CLAIMS

Claim 1: A model compiling method, comprising:

obtaining a target computation sub-graph in a computation graph corresponding to a model to be compiled, wherein the target computation sub-graph comprises a plurality of memory-access-intensive operators and data corresponding to the plurality of memory-access-intensive operators, and the target computation sub-graph is one computation sub-graph contained in the computation graph;

determining, based on a plurality of threads set in an executor of the model and a computation task parallelism degree of each thread, a plurality of groups of computation tasks to be serially executed by a target thread, wherein each group of computation tasks comprises a plurality of computation tasks, each computation task is a task for sequentially adopting the plurality of memory-access-intensive operators to perform computation on the data, the computation task parallelism degree indicates a quantity of computation tasks executed by each thread in parallel, and the target thread is any one of the plurality of threads;

performing, for any group of computation tasks based on computational dependencies of the plurality of memory-access-intensive operators, sequential rearrangement on a plurality of computation tasks contained in the any group of computation tasks, to obtain the plurality of rearranged computation tasks, wherein the rearrangement enables different computation tasks corresponding to the same memory-access-intensive operator to be arranged in parallel;

generating, based on the plurality of rearranged computation tasks, an execution instruction sequence both corresponding to the target thread and the any group of computation tasks; and

determining, based on the execution instruction sequence both corresponding to the target thread and the any group of computation tasks, an executable program file for executing the model by the executor, wherein the executable program file comprises execution instruction

sequences respectively corresponding to a plurality of computation sub-graphs contained in the computation graph.

Claim 2: The model compiling method according to claim 1, wherein the model compiling method further comprises:

inserting a first data copy instruction before the execution instruction sequence both corresponding to the target thread and the any group of computation tasks, wherein the first data copy instruction is used for copying data corresponding to a next group of computation tasks of the any group of computation tasks from an off-chip memory of the executor to a first buffer in an on-chip memory of the executor, to enable the data to be read from the first buffer when the next group of computation tasks is executed.

Claim 3: The model compiling method according to claim 2, wherein the model compiling method further comprises:

inserting a second data copy instruction before the first data copy instruction in response to the any group of computation tasks being an initial computation task in the plurality of groups of computation tasks, wherein the second data copy instruction is used for copying data corresponding to the any group of computation tasks from the off-chip memory of the executor to a second buffer in the on-chip memory of the executor.

Claim 4: The model compiling method according to claim 3, wherein the data, corresponding to the next group of computation tasks, copied to the first buffer is data corresponding to an initial memory-access-intensive operator in the next group of computation tasks; and the data, corresponding to the any group of computation tasks, copied to the second buffer is data corresponding to an initial memory-access-intensive operator in the any group of computation tasks.

Claim 5: The model compiling method according to any one of claims 1 to 4, wherein determining, based on the execution instruction sequence both corresponding to the target thread and the any group of computation tasks, the executable program file for executing the model by the executor comprises:

determining, based on execution instruction sequences respectively corresponding to the plurality of groups of computation tasks respectively executed by the plurality of threads in the target computation sub-graph, an execution instruction sequence corresponding to the target computation sub-graph; and

fusing the execution instruction sequences respectively corresponding to the plurality of computation sub-graphs in the computation graph, to determine the executable program file of the model.

Claim 6: The model compiling method according to claim 5, wherein the model compiling method further comprises:

obtaining a plurality of execution instruction sequences corresponding to the target computation sub-graph at a plurality of computation task parallelism degrees; and

determining, based on running performance of the executor when separately executing the plurality of execution instruction sequences, a target execution instruction sequence corresponding to the target computation sub-graph; and

fusing the execution instruction sequences respectively corresponding to the plurality of computation sub-graphs in the computation graph, to determine the executable program file of the model comprises:

fusing target execution instruction sequences respectively corresponding to the plurality of computation sub-graphs, to determine the executable program file of the model.

Claim 7: The model compiling method according to claim 5, wherein fusing the execution instruction sequences respectively corresponding to the plurality of

computation sub-graphs in the computation graph, to determine the executable program file of the model comprises:

respectively determining distances between the plurality of computation sub-graphs and an initial computation sub-graph;

grouping the plurality of computation sub-graphs based on the distances, wherein distances corresponding to computation sub-graphs in the same group of computation sub-graphs are the same; and

inserting a global synchronization instruction after an execution instruction sequence corresponding to each group of computation sub-graphs, to determine the executable program file of the model.

Claim 8: The model compiling method according to claim 2, wherein determining, based on the execution instruction sequence both corresponding to the target thread and the any group of computation tasks, the executable program file for executing the model by the executor comprises:

determining, based on execution instruction sequences respectively corresponding to the plurality of groups of computation tasks respectively executed by the plurality of threads in the target computation sub-graph, an execution instruction sequence corresponding to the target computation sub-graph; and

fusing the execution instruction sequences respectively corresponding to the plurality of computation sub-graphs in the computation graph, to determine the executable program file of the model.

Claim 9: The model compiling method according to claim 3, wherein determining, based on the execution instruction sequence both corresponding to the target thread and the any group of computation tasks, the executable program file for executing the model by the executor comprises:

determining, based on execution instruction sequences respectively corresponding to the plurality of groups of computation tasks respectively executed by the plurality of threads in the target computation sub-graph,

an execution instruction sequence corresponding to the target computation sub-graph; and

fusing the execution instruction sequences respectively corresponding to the plurality of computation sub-graphs in the computation graph, to determine the executable program file of the model.

Claim 10: The model compiling method according to claim 4, wherein determining, based on the execution instruction sequence both corresponding to the target thread and the any group of computation tasks, the executable program file for executing the model by the executor comprises:

determining, based on execution instruction sequences respectively corresponding to the plurality of groups of computation tasks respectively executed by the plurality of threads in the target computation sub-graph, an execution instruction sequence corresponding to the target computation sub-graph; and

fusing the execution instruction sequences respectively corresponding to the plurality of computation sub-graphs in the computation graph, to determine the executable program file of the model.

Claim 11: The model compiling method according to claim 1, wherein the model compiling method further comprises:

determining, based on the plurality of memory-access-intensive operators and data corresponding to the plurality of memory-access-intensive operators comprised in the target computation sub-graph, a quantity of computation tasks corresponding to the target computation sub-graph.

Claim 12: The model compiling method according to claim 11, wherein determining, based on the plurality of threads set in the executor of the model and the computation task parallelism degree of each thread, the plurality of groups of computation tasks to be serially executed by the target thread comprises:

determining, based on the plurality of threads set in the executor of the model and the quantity of computation tasks corresponding to the target

computation sub-graph, computation tasks corresponding to each thread; and

determining, based on the computation tasks corresponding to each thread and the computation task parallelism degree of each thread, the plurality of groups of computation tasks to be serially executed by the target thread.

Claim 13: A model compiling apparatus, comprising:

an obtaining module, configured to obtain a target computation sub-graph in a computation graph corresponding to a model to be compiled, wherein the target computation sub-graph comprises a plurality of memory-access-intensive operators and data corresponding to the plurality of memory-access-intensive operators, and the target computation sub-graph is one computation sub-graph contained in the computation graph; and

a compiling module, configured to: determine, based on a plurality of threads set in an executor of the model and a computation task parallelism degree of each thread, a plurality of groups of computation tasks to be serially executed by a target thread, wherein each group of computation tasks comprises a plurality of computation tasks, each computation task is a task for sequentially adopting the plurality of memory-access-intensive operators to perform computation on the corresponding data, the computation task parallelism degree indicates a quantity of computation tasks executed by each thread in parallel, and the target thread is any one of the plurality of threads; perform, for any group of computation tasks based on computational dependencies of the plurality of memory-access-intensive operators, sequential rearrangement on a plurality of computation tasks contained in the any group of computation tasks, to obtain the plurality of rearranged computation tasks, wherein the rearrangement enables different computation tasks corresponding to the same memory-access-intensive operator to be arranged in parallel; generate, based on the plurality of rearranged computation tasks, an execution instruction sequence both corresponding to the target thread and the any group of computation tasks; and determine, based on the execution instruction sequence both corresponding to the target thread and the any group of

computation tasks, an executable program file for executing the model by the executor, wherein the executable program file comprises execution instruction sequences respectively corresponding to a plurality of computation sub-graphs contained in the computation graph.

Claim 14: An electronic device, comprising: a memory, a processor, and a communication interface, wherein the memory stores executable code, and when the executable code is executed by the processor, the processor is enabled to perform the model compiling method as claimed in any one of claims 1 to 12.

Claim 15: A non-transitory machine-readable storage medium, wherein the non-transitory machine-readable storage medium stores executable code, and when the executable code is executed by a processor of an electronic device, the processor is enabled to perform the model compiling method as claimed in any one of claims 1 to 12.