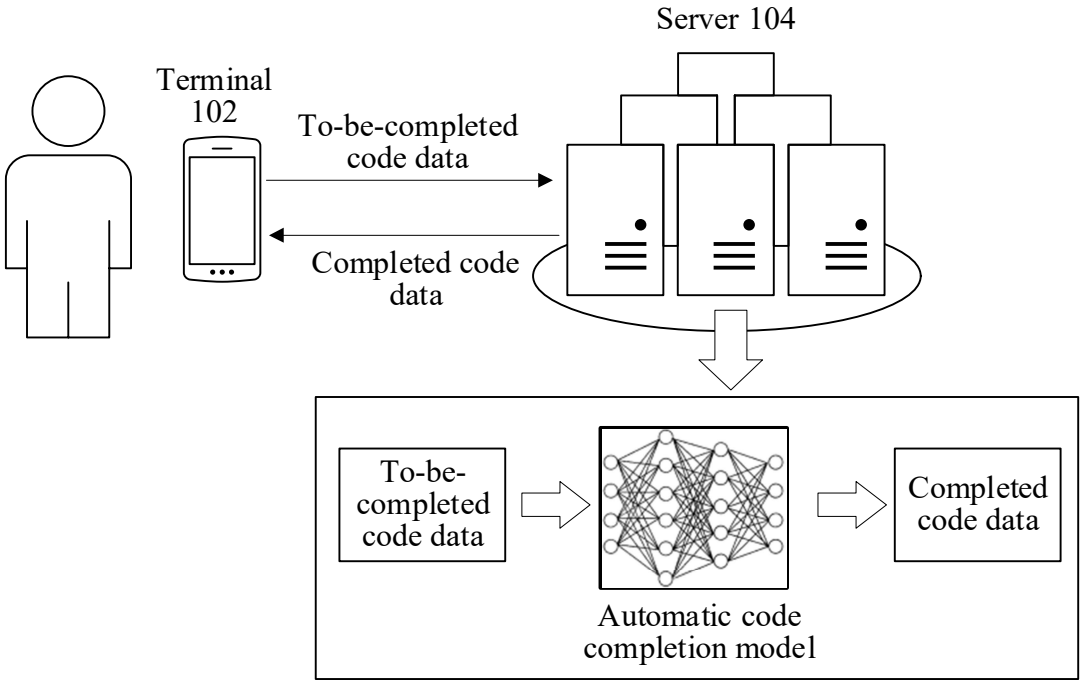

Server 104
Terminal
102
To-be-completed
code data
Completed code
data
To-be-
completed
code data
Completed
code data
Automatic code
completion model

FIG. 1

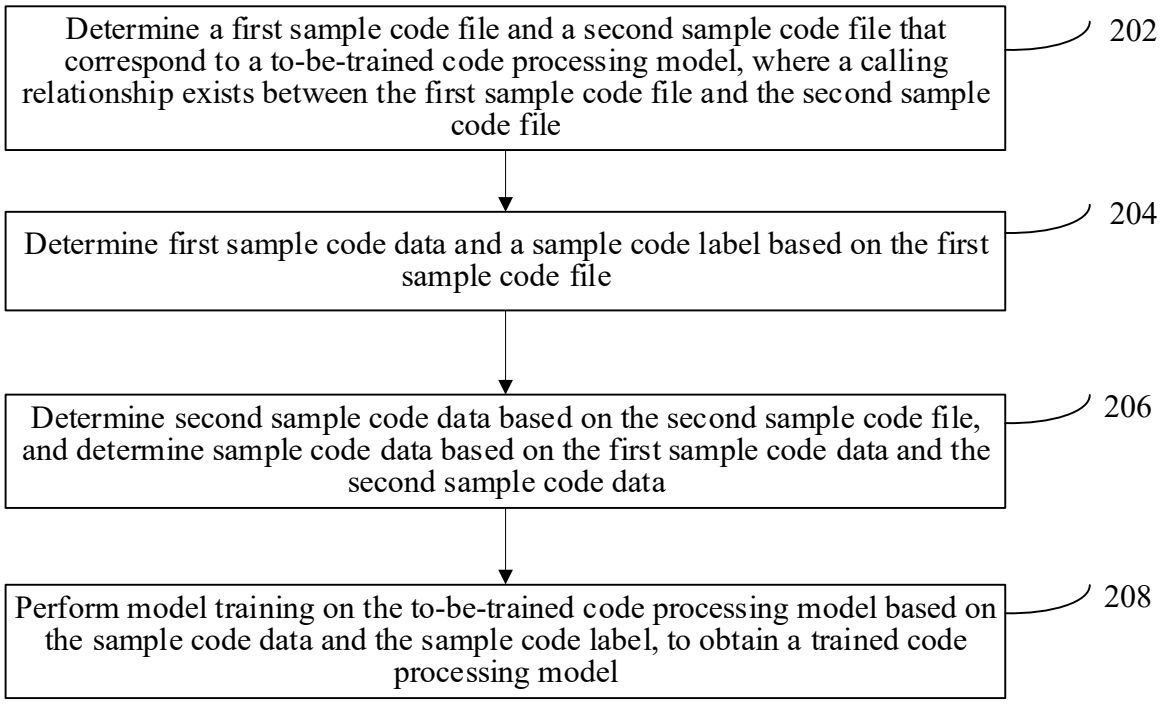

Determine a first sample code file and a second sample code file that correspond to a to-be-trained code processing model, where a calling relationship exists between the first sample code file and the second sample code file
202
Determine first sample code data and a sample code label based on the first sample code file
204
Determine second sample code data based on the second sample code file, and determine sample code data based on the first sample code data and the second sample code data
206
Perform model training on the to-be-trained code processing model based on the sample code data and the sample code label, to obtain a trained code processing model
208

FIG. 2

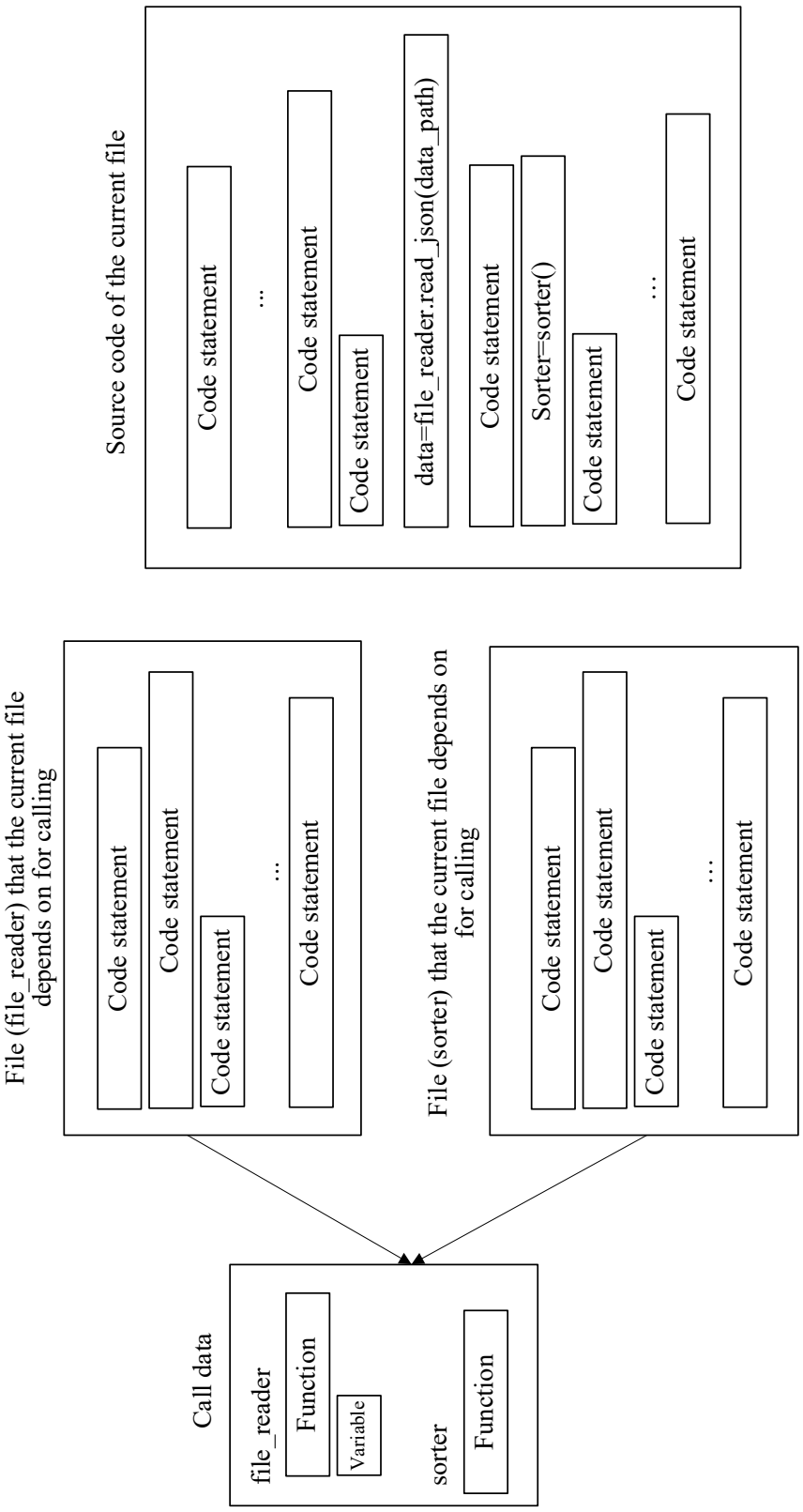

Source code of the current file
Code statement
...
Code statement
Code statement
data=file_reader.read_json(data_path)
Code statement
Sorter=sorter()
Code statement
...
Code statement
File (file_reader) that the current file depends on for calling
Code statement
Code statement
Code statement
...
Code statement
File (sorter) that the current file depends on for calling
Code statement
Code statement
Code statement
...
Code statement
Call data
file_reader
Function
Variable
sorter
Function

FIG. 3

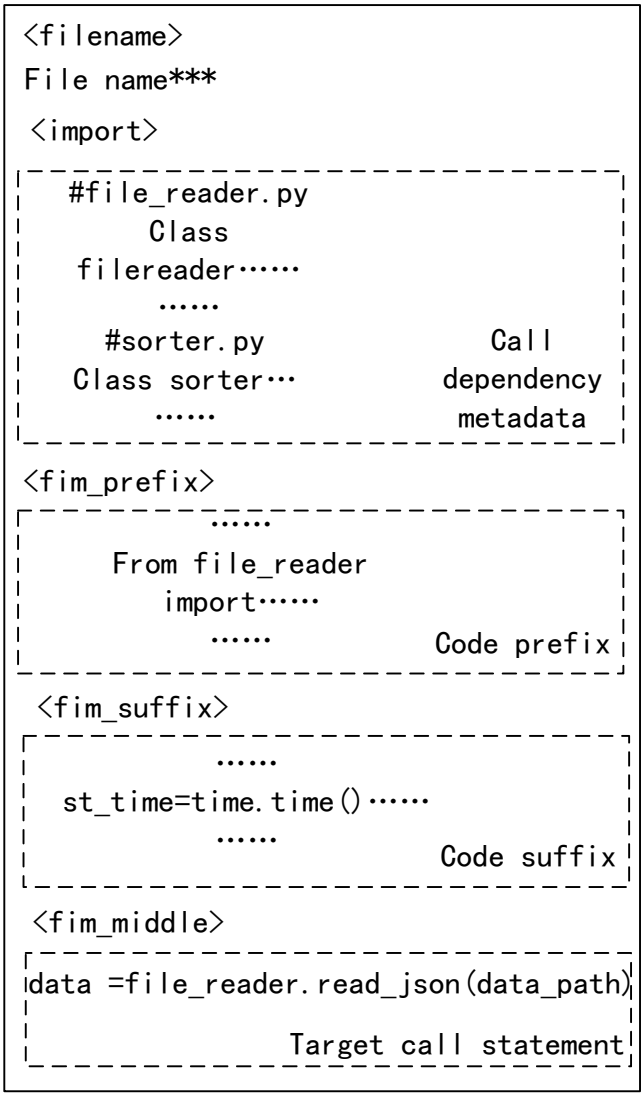

<filename>
File name***
<import>
#file_reader.py
Class
filereader……
……
#sorter.py
Class sorter…
……
Call
dependency
metadata
<fim_prefix>
……
From file_reader
import……
……
Code prefix
<fim_suffix>
……
st_time=time.time()……
……
Code suffix
<fim_middle>
data =file_reader.read_json(data_path)
Target call statement

FIG. 4

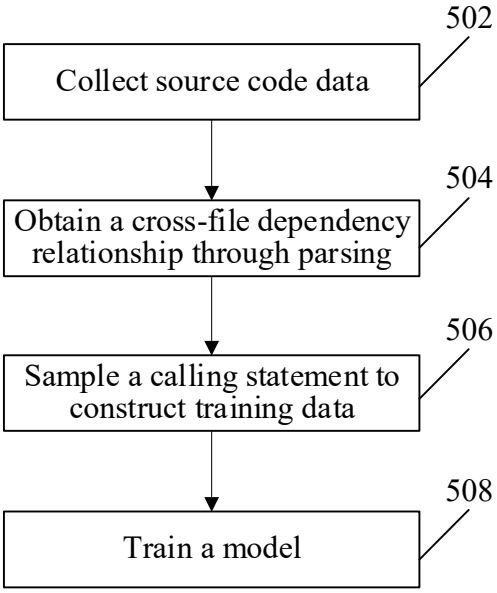

502
Collect source code data
504
Obtain a cross-file dependency relationship through parsing
506
Sample a calling statement to construct training data
508
Train a model

FIG. 5

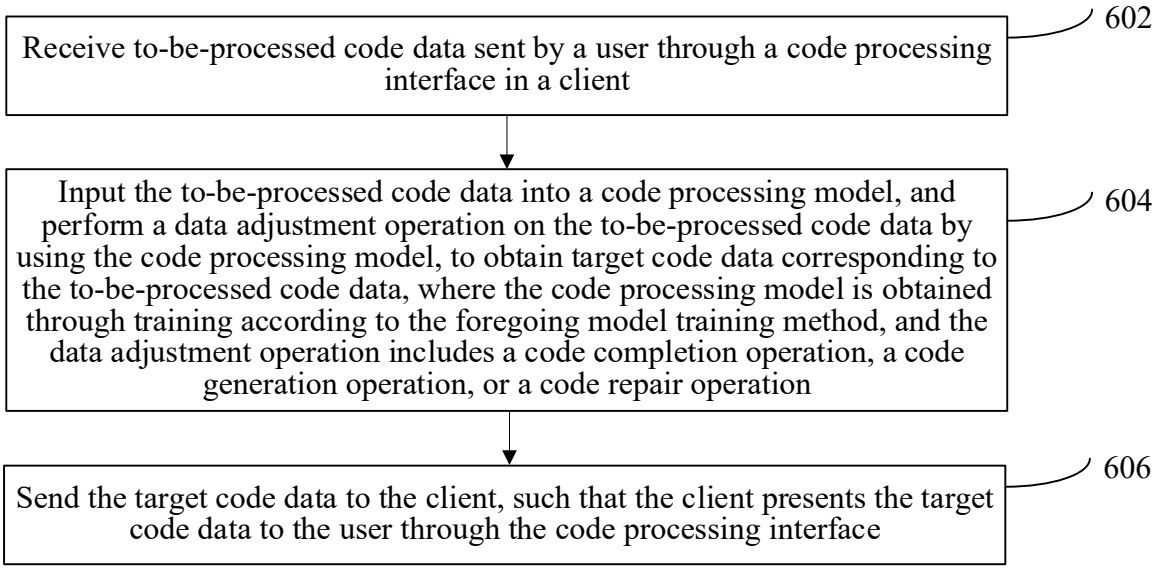

Receive to-be-processed code data sent by a user through a code processing interface in a client
602
Input the to-be-processed code data into a code processing model, and perform a data adjustment operation on the to-be-processed code data by using the code processing model, to obtain target code data corresponding to the to-be-processed code data, where the code processing model is obtained through training according to the foregoing model training method, and the data adjustment operation includes a code completion operation, a code generation operation, or a code repair operation
604
Send the target code data to the client, such that the client presents the target code data to the user through the code processing interface
606

FIG. 6

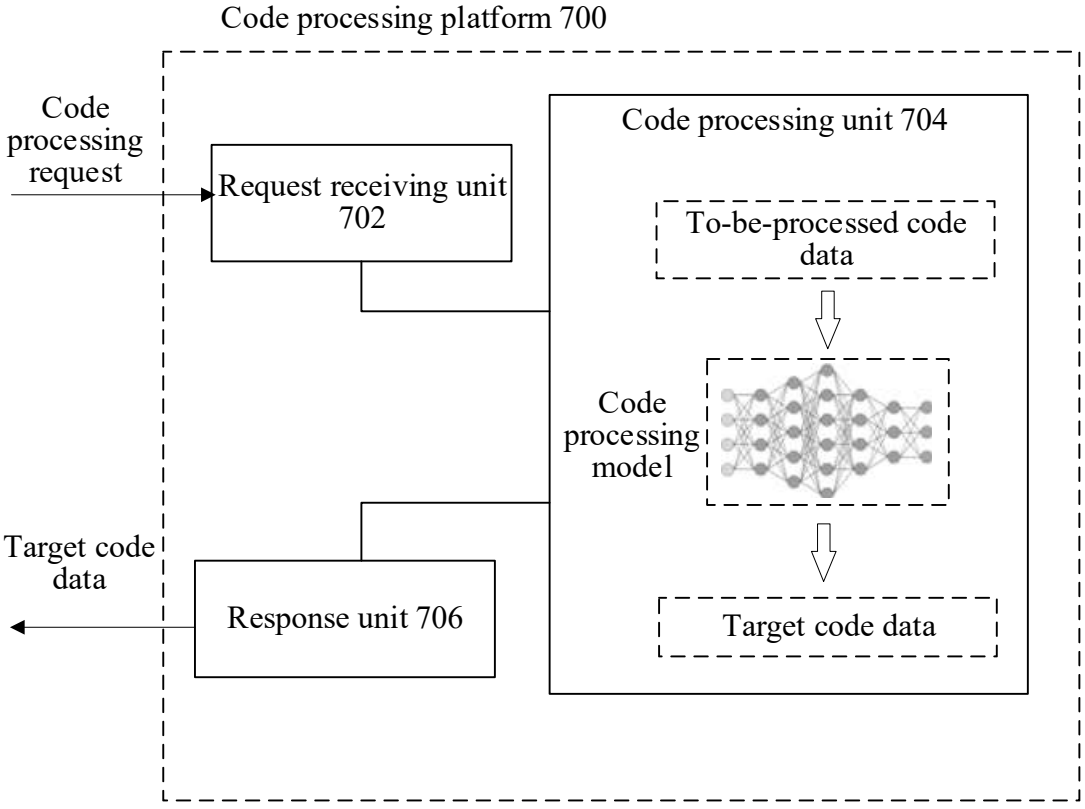

Code processing platform 700
Code processing request
Request receiving unit 702
Code processing unit 704
To-be-processed code data
Code processing model
Target code data
Response unit 706
Target code data

FIG. 7

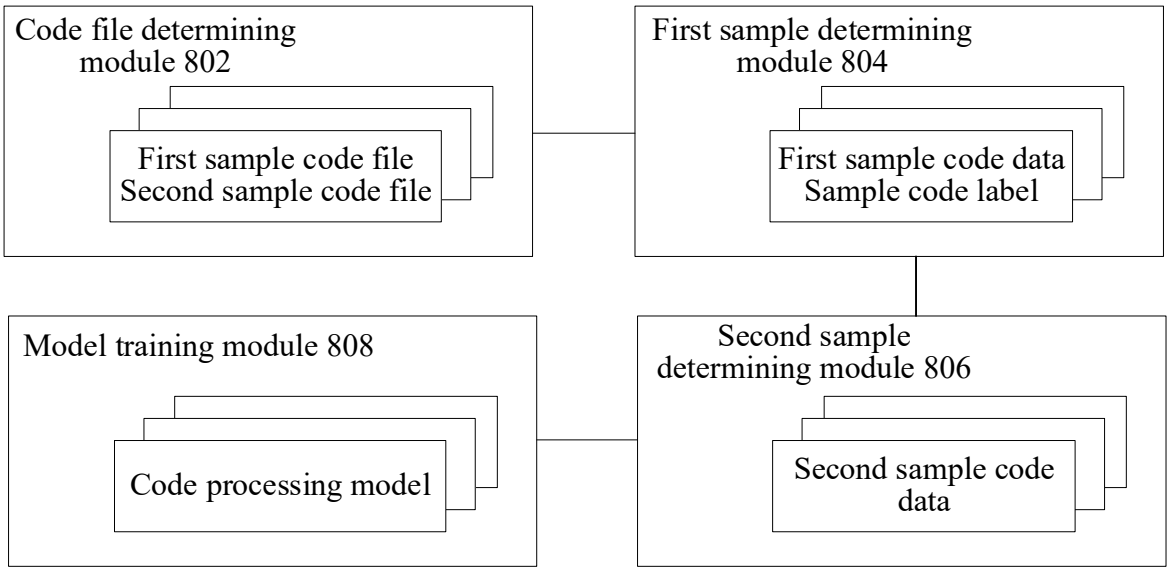

Code file determining module 802
First sample code file
Second sample code file
First sample determining module 804
First sample code data
Sample code label
Model training module 808
Code processing model
Second sample determining module 806
Second sample code data

FIG. 8

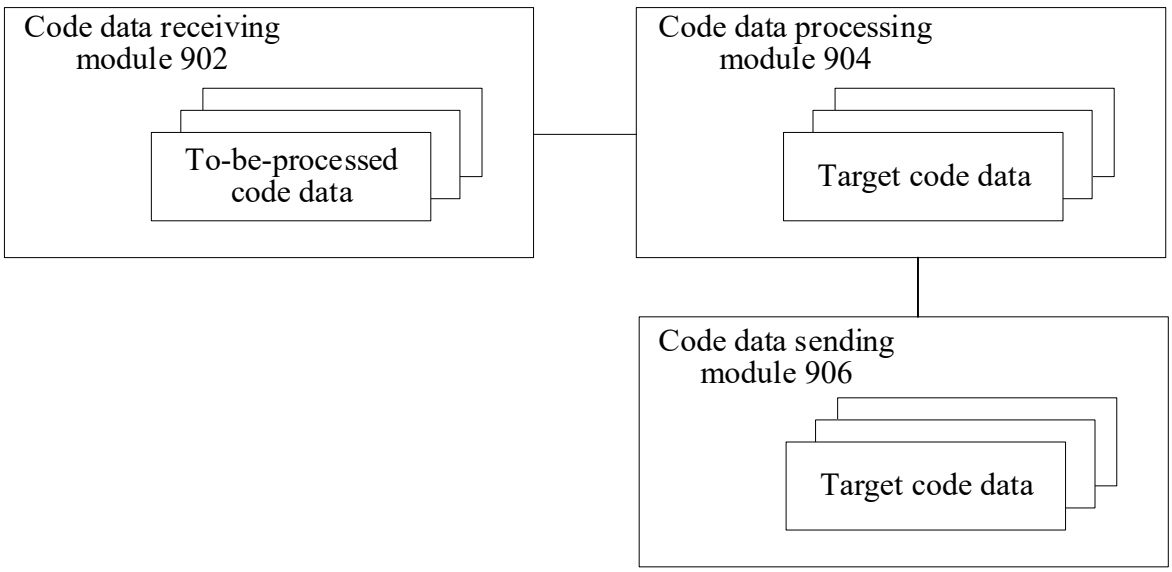

Code data receiving module 902
To-be-processed code data
Code data processing module 904
Target code data
Code data sending module 906
Target code data

FIG. 9

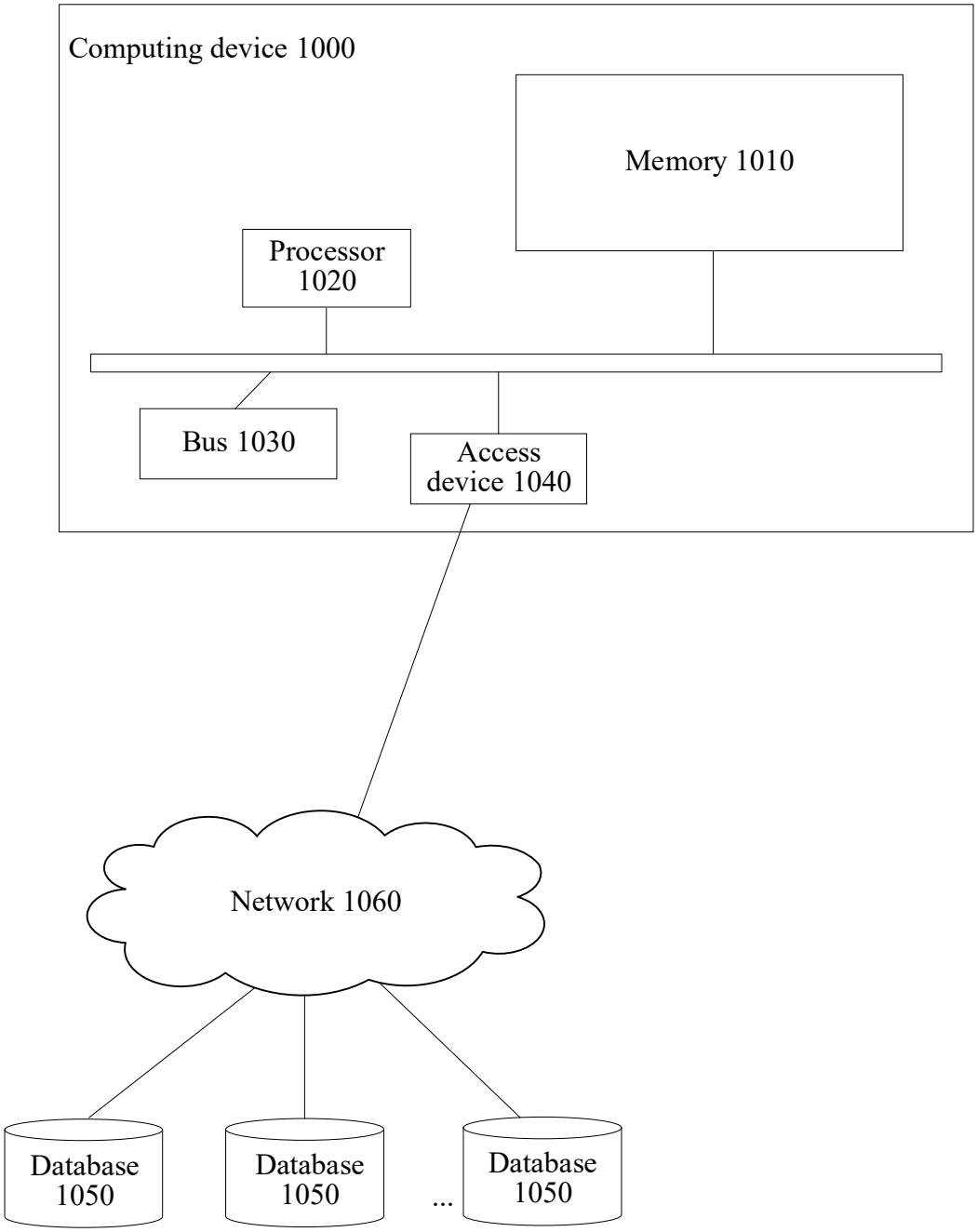

Computing device 1000
Memory 1010
Processor 1020
Bus 1030
Access device 1040
Network 1060
Database 1050
Database 1050
...
Database 1050

FIG. 10