

MEMORY MIGRATION METHOD, APPARATUS, AND COMPUTING DEVICE

TECHNICAL FIELD

[1] Embodiments of this application relate to the field of computer application
5 technologies, and in particular, to a memory migration method, an apparatus, and a
computing device.

BACKGROUND

[2] Virtual machine (Virtual Machine) live migration refers to a process of migrating a
10 running virtual machine from one physical machine to another, during which the virtual
machine can still provide services externally. The virtual machine live migration mainly
includes live migrations of CPU, memory, and input/output (IO). The live migrations of
CPU and IO are simple, and only their states are transferred, while memory migration is the
most time-consuming part.

15 [3] Currently, during memory migration, a migration thread is first created, and the
migration thread copies dirty pages to a destination end in an iterative manner. With the
increase of the iteration round number, the number of dirty pages decreases. When the
number of dirty pages converges to a threshold, a virtual machine at a source end can be
suspended, and the remaining dirty pages, CPU states, and IO states are copied to a virtual
20 machine at a destination end. After the copy is complete, the virtual machine at the
destination end resumes running, and the live migration ends.

[4] During the migration, the migration thread executes a next round of iteration only
after dirty pages at a previous round are all sent to the destination end. In each iteration
process, the migration thread needs to perform a plurality of processing operations. If a
25 bottleneck occurs in any processing operation, the memory migration efficiency will be
affected.

[8] According to a third aspect, an embodiment of this application provides a memory migration method, including:

creating a processing thread, a sending thread, a traversal thread, and a synchronization thread for a memory region; where

5 the synchronization thread iteratively executes a synchronization operation, where the synchronization operation includes acquiring a dirty bitmap of the memory region and sending the dirty bitmap to a corresponding traversal thread;

the traversal thread iteratively executes a traversal operation, where the traversal operation includes traversing an assigned dirty bitmap and assigning a dirty page address to the processing thread;

10 the processing thread iteratively executes a processing operation, where the processing operation includes reading dirty page data based on the assigned dirty page address, and adding round number flag information to the dirty page data to generate a to-be-migrated dirty page; and

15 the sending thread iteratively executes a transmission operation, where the transmission operation includes sending the to-be-migrated dirty page to a destination end, and the destination end sequentially restores respective dirty page data to destination memories based on iteration round numbers in different to-be-migrated dirty pages.

20 [9] According to a fourth aspect, an embodiment of this application provides a memory migration method, including:

receiving at least one to-be-migrated dirty page transmitted by a source end;

parsing round number flag information in the at least one to-be-migrated dirty page to determine the iteration round number corresponding to the at least one to-be-migrated dirty page; and

25 sequentially restoring dirty page data in the at least one to-be-migrated dirty page to a destination memory based on the iteration round number in the at least one to-be-migrated dirty page.

[10] According to a fifth aspect, an embodiment of this application provides a memory migration apparatus, including:

30

[22] In the embodiments of this application, a processing thread, a sending thread, and a traversal thread for a memory region are created, and the different threads iteratively execute respective operations independently, and add round number flag information in dirty pages, so that even if dirty pages at different rounds have not sequentially arrived at a destination end, the destination end can sequentially restore dirty page data based on the round number flag information in the dirty pages, ensuring memory consistency. By creating a variety of threads, a transmit end does not need to consider order preservation. Different threads can execute different processing operations in parallel at a same time point. Different threads do not need to be synchronized with or wait for each other, and the overall migration process will not be blocked by a slower processing operation, thereby improving the efficiency and success rate of memory migration.

[23] These and other aspects of this application will be more concise and comprehensible in the description of the following embodiments.

BRIEF DESCRIPTION OF DRAWINGS

[24] To describe the technical solutions in the embodiments of this application or in the prior art more clearly, the following briefly describes the accompanying drawings for describing the embodiments or the prior art. The accompanying drawings in the following description show some embodiments of this application, and persons of ordinary skill in the art may still derive other drawings from these accompanying drawings without creative efforts.

[25] FIG. 1 is a flowchart of an embodiment of a memory migration method according to this application;

[26] FIG. 2 is a flowchart of another embodiment of a memory migration method according to this application;

[27] FIG. 3 is a flowchart of another embodiment of a memory migration method according to this application;

[39] The flows described in the specification, claims, and accompanying drawings of this application include a plurality of operations in a specific order. However, it should be clearly understood that the operations can be executed in an order different from the order in the specification, or may be executed in parallel, and that the operation numbers
5 such as 101 and 102 are only used to differentiate different operations, but do not represent any precedence. In addition, the flows may include more or fewer operations, and the operations can be executed in order or in parallel. It should be noted that the descriptions "first", "second" and the like in the specification are used to differentiate different messages, devices, modules and the like, but neither represent a precedence
10 order, nor define that "first" and "second" are different types.

[40] Embodiments of this application are applied to a virtual machine live migration scenario. The virtual machine live migration is time-demanding and requires that the total migration time to be as short as possible. Memory migration in the virtual machine live migration process is the most time-consuming. Therefore, how to optimize the
15 memory migration process and improve memory migration efficiency is a main technical problem to be resolved by the present invention.

[41] For ease of understanding, technical terms that may appear in the embodiments of this application will be explained first in the following.

[42] Virtual machine (Virtual Machine): refers to a software-simulated complete
20 computer system having complete hardware system functions and running in a completely isolated environment.

[43] Virtual machine live migration: refers to a process of migrating a running virtual machine from one physical machine to another, during which the virtual machine can still provide services externally. The virtual machine live migration mainly includes live
25 migrations of CPU, memory, and input/output (IO).

[44] Memory iterative copy: is a memory migration manner in the virtual machine live migration process, in which all memory pages are copied by a migration thread to a destination end in the first round, then dirty pages from a previous round are copied to the destination end in each round until the number of the remaining dirty pages
30 converges to a threshold, the source virtual machine is suspended, and all the remaining dirty pages are copied to a destination virtual machine.

[45] Dirty page: refers to a memory page in which data generated in a previous round has been modified during a memory iterative copy process.

[46] Dirty bitmap (dirty bitmap): is used to record which memory pages are dirty pages. Each bit in the dirty bitmap corresponds to a memory page. If a memory page is dirty, it is marked as 1, otherwise it is marked as 0 in an exemplary implementation.

[47] Dirty page flag: is identification information at the header of a dirty page, used to indicate a dirty page address, data length, and the like.

[48] At present, memory migration is executed by a migration thread, and after a round of iteration is completed, a next round of iteration proceeds. During each round of iteration, a plurality of processing operations such as traversing the dirty bitmap, acquiring dirty page data, and transmitting the dirty page data are performed. If a problem occurs in any one processing operation, a “barrel effect” will be caused, affecting the overall migration efficiency. In practical applications, in order to reduce the amount of transmitted data and improve the data transmission efficiency, so as to improve the memory migration efficiency, a compression thread may be created to compress the dirty page data for transmission. However, the migration thread can start transmission only after the compression thread completes the compression. A problem of the compression thread also affects the overall migration efficiency. In addition, in an actual service running process, there often exists a very large memory, and a large number of dirty pages are generated during the migration. If the memory migration efficiency is too low, memory iteration fails to converge, making it impossible to achieve virtual machine live migration ultimately. In the case that the compression thread has completed the current round of data compression and the migration thread starts transmission, the migration thread may not execute a next round of iteration before the transmission is completed. As a result, the compression thread stays in a waiting state and cannot be fully utilized, leading to waste of resources.

[49] The inventors have found in research that iterations in the memory migration process needs to be executed in order, so the migration thread needs to complete one round of iteration operations before starting a next round of iteration operations. Otherwise, it may occur at the destination end that dirty pages in an earlier round are delayed in reaching the destination end, and the delayed dirty pages overwrite dirty

pages in a new round, causing memory inconsistency between a source virtual machine and a destination virtual machine. Therefore, according to the prior-art memory migration method, before proceeding to a next round of iteration, the source end needs to ensure that all dirty pages in the current round are sent out.

5 [50] To improve memory migration efficiency, and ensure memory consistency between the source end and the destination end, the inventors have carried out a series of research and put forward the technical solution of this application. In the embodiments of this application, a processing thread, a sending thread, and a traversal thread for a memory region are created, and the different threads iteratively execute
10 respective operations independently, and add round number flag information for identifying an iteration round number to dirty pages, so that even if dirty pages at different rounds do not sequentially arrive at a destination end, the destination end can restore dirty page data based on the round number flag information in the dirty pages, ensuring memory consistency. By setting the round number flag information for
15 identifying the iteration round number, and creating a variety of threads, the transmit end does not need to consider order preservation. Different threads can execute different processing operations in parallel at a same time point. Different threads do not need to be synchronized with or wait for each other, and the overall migration process will not be blocked by a slower processing operation, eliminating the "barrel effect", and thereby
20 improving the efficiency and success rate of memory migration. Also, different threads iteratively execute respective operations at full load, improving resource utilization.

[51] The following describes the technical solutions in the embodiments of this application with reference to the accompanying drawings in the embodiments of this application. The described embodiments are only some rather than all of the
25 embodiments of this application. All other embodiments obtained by those skilled in the art based on the embodiments of this application without creative efforts shall fall within the protection scope of this application.

[52] FIG. 1 is a flowchart of an embodiment of a memory migration method according to this application. The method may include the following steps.

[73] A to-be-migrated dirty page includes dirty page data and a dirty page flag. The dirty page data is read from a memory page corresponding to the dirty page address, and the dirty page flag is added at the header to form a to-be-migrated dirty page. A destination end may restore the dirty page data based on the dirty page address in the dirty page flag, a data length, and other information. Therefore, optionally, round number flag information for identifying a corresponding iteration round number may be added to the dirty page flag, and the dirty page flag and the dirty page data constitute a to-be-migrated dirty page.

[74] In this embodiment of this application, round number flag information for identifying an iteration round number is added to the dirty page flag, so that the iteration round number can be determined based on the round number flag information.

[75] In one implementation, the round number flag information may directly be an Arabic numeral indicating the iteration round number, and the iteration round number starts counting from the number 1.

[76] In another implementation, the round number flag information may be time stamp information. The time stamp information may indicate synchronization time or traversal time of the dirty bitmap, or the like. Because during synchronization or traversal, a next round of iteration may proceed only after the current round is completed, the iteration round number may also be determined based on the sequence of the time stamps.

[77] Certainly, other methods may also be used to identify the iteration round number, and this is not specifically limited in this application.

[78] 104: The sending thread iteratively executes a transmission operation.

[79] The transmission operation includes sending at least one to-be-migrated dirty page to a destination end. Each sending thread may send at least one to-be-migrated dirty page to the destination end.

[80] Each processing thread may assign a generated to-be-migrated dirty page to the sending thread.

CLAIMS

1. A memory migration method, comprising:

creating a processing thread, a sending thread, and a traversal thread for a memory region; wherein

5 the traversal thread iteratively executes a traversal operation, wherein the traversal operation comprises traversing a dirty bitmap of the memory region and assigning a dirty page address to a corresponding processing thread;

the processing thread iteratively executes a processing operation, wherein the processing operation comprises reading dirty page data based on the assigned dirty page
10 address, and adding round number flag information to the dirty page data to generate a to-be-migrated dirty page; and

the sending thread iteratively executes a transmission operation, wherein the transmission operation comprises sending the to-be-migrated dirty page to a destination end, and the destination end sequentially restores respective dirty page data to
15 destination memories based on iteration round numbers in different to-be-migrated dirty pages.

2. The method according to claim 1, wherein the processing thread specifically iteratively executes the following processing operation:

reading dirty page data based on the assigned dirty page address;

20 compressing the dirty page data; and

adding round number flag information for identifying a corresponding iteration round number to the compressed dirty page data to generate a to-be-migrated dirty page.

3. The method according to claim 1, further comprising:

creating a synchronization thread for the memory region; wherein

the synchronization thread iteratively executes a synchronization operation, wherein the synchronization operation comprises acquiring a dirty bitmap of the memory region and sending the dirty bitmap to the traversal thread.

- 5 4. The method according to claim 1, wherein a plurality of processing threads are specifically created, and the plurality of processing threads run in parallel and iteratively execute the processing operation separately;

the traversal thread specifically iteratively executes the following traversal operation:

- 10 traversing a dirty bitmap of the memory region to determine a dirty page address and an iteration round number;

- assigning, based on the number of to-be-processed dirty pages, corresponding to-be-processed tasks to one or more processing threads with free resources, wherein the to-be-processed task comprises the iteration round number and at least one to-be-processed dirty page address.
- 15

5. The method according to claim 1, wherein the processing thread specifically iteratively executes the following processing operation:

reading dirty page data based on the assigned dirty page address;

- adding round number flag information for identifying an iteration round number to the dirty page data to generate a to-be-migrated dirty page; and
- 20

 caching the to-be-migrated dirty page to a dirty page cache page corresponding to the processing thread.

6. The method according to claim 5, wherein the sending thread specifically iteratively executes the following sending operation:

transmitting a to-be-migrated dirty page in the dirty page cache queue to a destination end.

7. The method according to claim 1, wherein a source memory is divided into a plurality of the memory regions, and different memory regions do not overlap each other; and

the creating a processing thread, a sending thread, and a traversal thread for a memory region comprises:

creating a traversal thread for the memory region, at least one sending thread, and at least one processing thread.

8. The method according to claim 1, wherein the creating a sending thread, a processing thread, and a traversal thread for a memory region comprises:

creating a sending thread, a processing thread, and a traversal thread for a memory region in one or more processing components.

9. The method according to claim 1, wherein a plurality of processing threads and a plurality of sending threads are specifically created; and

a plurality of processing threads run in parallel, and iteratively execute the processing operation separately, and a plurality of sending threads run in parallel, and iteratively execute the transmission operation separately.

10. A memory migration method, comprising:

creating a sending thread and a traversal thread for a memory region; wherein

the traversal thread iteratively executes a traversal operation, wherein the traversal operation comprises traversing a dirty bitmap of the memory region, reading

dirty page data based on a dirty page address, and adding round number flag information to the dirty page data to generate a to-be-migrated dirty page; and

the sending thread iteratively executes a transmission operation, wherein the transmission operation comprises sending the to-be-migrated dirty page to a destination end, and

the destination end sequentially restores respective dirty page data to destination memories based on iteration round numbers in different to-be-migrated dirty pages.

11. The method according to claim 10, further comprising:

creating a synchronization thread for the memory region; wherein

the synchronization thread iteratively executes a synchronization operation, wherein the synchronization operation comprises acquiring a dirty bitmap of the memory region and sending the dirty bitmap and the iteration round number to the traversal thread.

12. The method according to claim 10, wherein a plurality of sending threads are specifically created, and the plurality of sending threads run in parallel and iteratively execute the sending operation separately; and

the traversal thread specifically iteratively executes the following traversal operation:

traversing a dirty bitmap of the memory region to determine a dirty page address and an iteration round number;

reading dirty page data based on the dirty page address;

adding round number flag information for identifying the iteration round number to the dirty page data to generate a to-be-migrated dirty page; and

separately assigning, based on the number of to-be-processed dirty pages, at least one to-be-migrated dirty page to one or more sending threads with free resources.

13. The method according to claim 10, wherein a source memory is divided into a plurality of the memory regions, and different memory regions do not overlap each other; and

the creating a sending thread and a traversal thread for a memory region
5 comprises:

creating for the memory region a traversal thread and a sending thread.

14. The method according to claim 10, wherein the creating a sending thread and a traversal thread for a memory region comprises:

creating a sending thread and a traversal thread for a memory region in one or
10 more processing components.

15. The method according to claim 10, wherein a plurality of the sending threads are specifically created; and

a plurality of sending threads run in parallel, and iteratively execute the transmission operation separately.

15 16. A memory migration method, comprising:

creating a processing thread, a sending thread, a traversal thread, and a synchronization thread for a memory region; wherein

the synchronization thread iteratively executes a synchronization operation, wherein the synchronization operation comprises acquiring a dirty bitmap of the
20 memory region and sending the dirty bitmap to a corresponding traversal thread;

the traversal thread iteratively executes a traversal operation, wherein the traversal operation comprises traversing an assigned dirty bitmap and assigning a dirty page address to the processing thread;

the processing thread iteratively executes a processing operation, wherein the processing operation comprises reading dirty page data based on the assigned dirty page address, and adding round number flag information to the dirty page data to generate a to-be-migrated dirty page; and

5 the sending thread iteratively executes a transmission operation, wherein the transmission operation comprises sending the to-be-migrated dirty page to a destination end, and the destination end sequentially restores respective dirty page data to destination memories based on iteration round numbers in different to-be-migrated dirty pages.

10 17. The method according to claim 16, wherein a plurality of processing threads, a plurality of sending threads and a plurality of traversal threads are specifically created; and

 a plurality of traversal threads run in parallel, and iteratively execute the traversal operation separately; a plurality of processing threads run in parallel, and
15 iteratively execute the processing operation separately; and a plurality of sending threads run in parallel, and iteratively execute the sending operation separately.

18. A memory migration method, comprising:

 receiving at least one to-be-migrated dirty page transmitted by a source end;

 parsing round number flag information in the at least one to-be-migrated dirty
20 page, to determine the iteration round number corresponding to the at least one to-be-migrated dirty page; and

 sequentially restoring dirty page data in the at least one to-be-migrated dirty page to a destination memory based on the iteration round number in the at least one to-be-migrated dirty page.

19. The method according to claim 18, wherein the sequentially restoring dirty page data in the at least one to-be-migrated dirty page to a destination memory based on the iteration round number in the at least one to-be-migrated dirty page comprises:

5 caching the at least one to-be-migrated dirty page to a receiving cache queue corresponding to a respective iteration round number; and

 sequentially restoring dirty page data in to-be-migrated dirty pages in different receiving cache queues to destination memories based on the iteration round numbers.

20. The method according to claim 19, wherein the sequentially restoring dirty page data in to-be-migrated dirty pages in different receiving cache queues to destination
10 memories based on the iteration round numbers comprises:

 sequentially decompressing and restoring the dirty page data in the to-be-migrated dirty pages in the different receiving cache queues to the destination memories based on the iteration round numbers.

21. The method according to claim 18, further comprising:

15 creating a receiving thread and a restoration thread,

 wherein the receiving thread iteratively executes a receiving operation, comprising receiving at least one to-be-migrated dirty page transmitted by a source end, parsing round number flag information in the at least one to-be-migrated dirty page, to determine the iteration round number corresponding to the at least one to-be-migrated
20 dirty page, and putting the at least one to-be-migrated dirty page into a receiving cache queue corresponding to a respective iteration round number; and

 the restoration thread iteratively executes a restoration operation of sequentially restoring dirty page data in to-be-migrated dirty pages in different receiving cache queues to destination memories based on the iteration round numbers.

22. A memory migration apparatus, comprising:

a first creation module adapted to create a processing thread, a sending thread, and a traversal thread for a memory region;

5 a first execution module adapted to control the traversal thread to iteratively execute a traversal operation, wherein the traversal operation comprises traversing a dirty bitmap of the memory region and assigning a dirty page address to a corresponding processing thread;

10 a second execution module adapted to control the processing thread to iteratively execute a processing operation, wherein the processing operation comprises reading dirty page data based on the assigned dirty page address, and adding round number flag information to the dirty page data to generate a to-be-migrated dirty page; and

15 a third execution module adapted to control the sending thread to iteratively execute a transmission operation, wherein the transmission operation comprises sending the to-be-migrated dirty page to a destination end, and the destination end sequentially restores respective dirty page data to destination memories based on iteration round numbers in different to-be-migrated dirty pages.

23. A memory migration apparatus, comprising:

a second creation module adapted to create a sending thread, and a traversal thread for a memory region;

20 a fourth execution module adapted to control the traversal thread to iteratively execute a traversal operation, wherein the traversal operation comprises traversing a dirty bitmap of the memory region, reading dirty page data based on a dirty page address, and adding round number flag information to the dirty page data to generate a to-be-migrated dirty page; and

25 a fifth execution module adapted to control the sending thread to iteratively execute a transmission operation, wherein the transmission operation comprises sending

the to-be-migrated dirty page to a destination end, and the destination end sequentially restores respective dirty page data to destination memories based on iteration round numbers in different to-be-migrated dirty pages.

24. A memory migration apparatus, comprising:

5 a third creation module adapted to create a processing thread, a sending thread, a traversal thread, and a synchronization thread for a memory region;

a sixth execution module adapted to control the synchronization thread to iteratively execute a synchronization operation, wherein the synchronization operation comprises acquiring a dirty bitmap of the memory region and sending the dirty bitmap
10 to a corresponding traversal thread;

a seventh execution module adapted to control the traversal thread to iteratively execute a traversal operation, wherein the traversal operation comprises traversing an assigned dirty bitmap and assigning a dirty page address to a corresponding processing thread;

15 an eighth execution module adapted to control the processing thread to iteratively execute a processing operation, wherein the processing operation comprises reading dirty page data based on the assigned dirty page address, and adding round number flag information to the dirty page data to generate a to-be-migrated dirty page; and

20 a ninth execution module adapted to control the sending thread to iteratively execute a transmission operation, wherein the transmission operation comprises sending the to-be-migrated dirty page to a destination end, and the destination end sequentially restores respective dirty page data to destination memories based on iteration round numbers in different to-be-migrated dirty pages.

25 25. A memory migration apparatus, comprising:

a parsing module adapted to receive at least one to-be-migrated dirty page transmitted by a source end, and parse round number flag information in the at least one to-be-migrated dirty page, to determine the iteration round number corresponding to the at least one to-be-migrated dirty page; and

5 a restoring module adapted to sequentially restore dirty page data in the at least one to-be-migrated dirty page to a destination memory based on the iteration round number in the at least one to-be-migrated dirty page.

26. A computing device, comprising one or more processing components and one or more storage components, wherein the one or more processing components run source
10 virtual machines;

the one or more storage components store one or more computer instructions, and the one or more computer instructions are to be called and executed by the one or more processing components; and

the one or more processing components are adapted to:

15 create a processing thread, a sending thread, and a traversal thread for a memory region; wherein

the traversal thread iteratively executes a traversal operation, wherein the traversal operation comprises traversing a dirty bitmap of the memory region and assigning a dirty page address to a corresponding processing thread;

20 the processing thread iteratively executes a processing operation, wherein the processing operation comprises reading dirty page data based on the assigned dirty page address, and adding round number flag information to the dirty page data to generate a to-be-migrated dirty page; and

the sending thread iteratively executes a transmission operation, wherein the
25 transmission operation comprises sending the to-be-migrated dirty page to a destination end, and the destination end sequentially restores respective dirty page data to

destination memories based on iteration round numbers in different to-be-migrated dirty pages.

27. A computing device, comprising one or more processing components and one or more storage components,

5 wherein the one or more storage components store one or more computer instructions, and the one or more computer instructions are to be called and executed by the one or more processing components; and

 the one or more processing components are adapted to:

 create a sending thread and a traversal thread for a memory region; wherein

10 by the traversal thread iteratively executes a traversal operation, wherein the traversal operation comprises traversing a dirty bitmap of the memory region, reading dirty page data based on a dirty page address, and adding round number flag information to the dirty page data to generate a to-be-migrated dirty page; and

 the sending thread iteratively executes a transmission operation, wherein the
15 transmission operation comprises sending the to-be-migrated dirty page to a destination end, and the destination end sequentially restores respective dirty page data to destination memories based on iteration round numbers in different to-be-migrated dirty pages.

28. A computing device, comprising one or more processing components and one or
20 more storage components,

 wherein the one or more storage components store one or more computer instructions, and the one or more computer instructions are to be called and executed by the one or more processing components; and

 the one or more processing components are adapted to:

create a processing thread, a sending thread, a traversal thread, and a synchronization thread for a memory region; wherein

the synchronization thread iteratively executes a synchronization operation, wherein the synchronization operation comprises acquiring a dirty bitmap of the memory region and sending the dirty bitmap to the traversal thread;

the traversal thread iteratively executes a traversal operation, wherein the traversal operation comprises traversing an assigned dirty bitmap and assigning a dirty page address to the processing thread;

the processing thread iteratively executes a processing operation, wherein the processing operation comprises reading dirty page data based on the assigned dirty page address, and adding round number flag information to the dirty page data to generate a to-be-migrated dirty page; and

the sending thread iteratively executes a transmission operation, wherein the transmission operation comprises sending the to-be-migrated dirty page to a destination end, and the destination end sequentially restores respective dirty page data to destination memories based on iteration round numbers in different to-be-migrated dirty pages.

29. A computing device, comprising one or more processing components and one or more storage components,

wherein the one or more storage components store one or more computer instructions, and the one or more computer instructions are to be called and executed by the one or more processing components; and

the one or more processing components are adapted to:

receive at least one to-be-migrated dirty page transmitted by a source end;

parse round number flag information in the at least one to-be-migrated dirty page, to determine the iteration round number corresponding to the at least one to-be-migrated dirty page; and

5 sequentially restore dirty page data in the at least one to-be-migrated dirty page to a destination memory based on the iteration round number in the at least one to-be-migrated dirty page.

30. A computer-readable storage medium, storing a computer program, wherein when the computer program is executed by a computer, the memory migration method according to any one of claim 1 to claim 9 is implemented.

10 31. A computer-readable storage medium, storing a computer program, wherein when the computer program is executed by a computer, the memory migration method according to any one of claim 10 to claim 15 is implemented.

32. A computer-readable storage medium, storing a computer program, wherein when the computer program is executed by a computer, the memory migration method
15 according to any one of claim 16 and claim 17 is implemented.

33. A computer-readable storage medium, storing a computer program, wherein when the computer program is executed by a computer, the memory migration method according to any one of claim 18 to claim 21 is implemented.