VIRTUAL MACHINE MEMORY DATA MIGRATION METHOD

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application claims priority to a Chinese Patent Application No. 2024103624674, filed to the China National Intellectual Property Administration on March 27, 2024, and entitled "Virtual Machine Memory Data Migration Method, Device, Computer Program Product, and Storage Medium", the entire contents of which are incorporated herein by reference.

TECHNICAL FIELD

[0002] The present disclosure relates to the field of cloud computing technology, and in particular to a virtual machine memory data migration method.

BACKGROUND

[0003] Live migration of a virtual machine refers to a process of migrating a running virtual machine from a source host to a destination host. During the process of migrating, work tasks on the virtual machine are not interrupted, and users are unaware of the live migration.

[0004] Memory data migration is an important part of a process of live migrating. Currently, the memory data migration causes significant fluctuations in an usage of physical memory in the source host, which leads to serious memory contention problems on the source host and affect a memory performance of the source host.

SUMMARY

[0005] Various aspects of the present disclosure provide a virtual machine memory data migration method, to reduce occupation of physical memory in a source host caused by memory data migration.

[0006] Some embodiments of the present disclosure provide a virtual machine memory data migration method, applied to a virtual machine manager in a source host. The virtual machine memory data migration method includes that:

[0007] in response to executing an initial migration for a target virtual memory page, a page fault state of the target virtual memory page is detected;

[0008] in response to the target virtual memory page being in the page fault state, memory data corresponding to the target virtual memory page is acquired without triggering a page fault recovery operation;

[0009] the memory data is migrated to the destination host.

[0010] Some embodiments of the present disclosure further provide a virtual machine memory data migration method, applied to a kernel interface in a source host that is used for supporting memory data migration. The virtual machine memory data migration method includes that:

[0011] a page fault state query instruction sent by a virtual machine manager in the source host when executing an initial migration for a target virtual memory page is received;

[0012] page fault state description information is generated for the target virtual memory page, according to the page fault state query instruction, where the page fault state description information is used for indicating a page fault state of the target virtual memory page;

[0013] the page fault state description information is provided to the virtual machine manager, so that after the target virtual memory page is detected to be in the page fault state, memory data corresponding to the target virtual memory page is acquired without triggering a page fault recovery operation, and the memory data is migrated to the destination host.

[0014] Some embodiments of the present disclosure further provide a computing device, including a memory, a processor, and a communication component;

[0015] the memory is configured to store one or more computer instructions;

[0016] the processor is coupled with the memory and the communication component, and is configured to execute the virtual machine memory data migration method.

[0017] Some embodiments of the present disclosure further provide a computer-readable storage medium storing a computer program, where the computer program, when executed by one or more processors, causes the one or more processors to perform the virtual machine memory data migration method.

[0018] Some embodiments of the present disclosure further provide a computer program product, where the computer program, when executed by one or more processors, causes the one or more processors to perform the virtual machine memory data migration method.

[0019] Some embodiments of the present disclosure further provide a computer program product, including a non-transitory computer-readable storage medium, storing a computer program, where the computer program, when executed by a processor, implements the virtual machine memory data migration method.

[0020] Some embodiments of the present disclosure further provide a computer program, where the computer program, when executed by a processor, implements the virtual machine memory data migration method.

[0021] In embodiments of the present disclosure, a memory data migration scheme during a process of virtual machine migration is improved, detection of a page fault state of each virtual memory page when executing an initial migration for each virtual memory page in a virtual machine is proposed. For each virtual memory page in the page fault state, the virtual machine manager in the source host acquires memory data corresponding to each virtual memory page without triggering a page fault recovery operation. In this way, during the process of virtual machine migration, even if partial virtual memory pages are in the page fault state, no page fault recovery operation is triggered, thereby occupation of physical memory in the source host caused by page fault recovery operations is avoided and occupation of physical memory in the source host caused by memory data migration is reduced. Accordingly, the memory data migration no longer causes significant fluctuations in the usage of physical memory in the source host, thereby exacerbation of memory contention problems in the source host is avoided.

BRIEF DESCRIPTION OF THE DRAWINGS

[0022] Drawings described herein are used for providing a further understanding of the present disclosure and constitute a part of the present disclosure. Illustrative embodiments of the present disclosure and descriptions thereof are used for explaining the present disclosure and do not constitute improper limitations on the present disclosure. In the drawings:

[0023] FIG. 1 is a schematic flowchart of a virtual machine memory data migration method according to some exemplary embodiments of the present disclosure;

[0024] FIG. 2 is a schematic logic diagram of a virtual machine memory data migration method according to some exemplary embodiments of the present disclosure;

[0025] FIG. 3 is a schematic logic diagram of an optional implementation of a virtual machine memory data migration method according to some exemplary embodiments of the present disclosure;

[0026] FIG. 4 is a schematic structural diagram of page fault state description information according to some exemplary embodiments of the present disclosure;

[0027] FIG. 5 is a flowchart of a virtual machine memory data migration method according to another exemplary embodiment of the present disclosure; and

[0028] FIG. 6 is a schematic structural diagram of a computing device according to yet another exemplary embodiment of the present disclosure.

DETAILED DESCRIPTION

[0029] To make the objectives, technical solutions, and advantages of the present disclosure clearer, the technical solutions of the present disclosure is described clearly and completely below with reference to specific embodiments of the present disclosure and corresponding drawings. Obviously, described embodiments are only part, rather than all, of the embodiments of the present disclosure. Based on the embodiments of the present disclosure, all other embodiments obtained by a person of ordinary skill in the art without creative effort fall within the protection scope of the present disclosure.

[0030] Before providing a detailed description of the technical solutions provided by the embodiments of the present disclosure, several technical concepts involved in the present disclosure are explained as follows.

[0031] Memory contention is a phenomenon that occurs on a host when physical memory is insufficient. The physical memory in the host is typically over-provisioned to each virtual machine running thereon, mainly in consideration of the fact that each virtual machine typically do not simultaneously use memory in full, and therefore it is expected to guarantee memory usage requirement of each virtual machine through time-division multiplexing, thereby utilization rate of the physical memory is improved. However, in some cases, the memory usage of each virtual machine has increased significantly, and total amount of memory actually required by each virtual machine has already exceeded total amount of physical memory in the host, resulting in a memory contention problem. The memory contention problem causes partial

virtual machines to be unable to use memory normally, and is a relatively serious performance problem on the host.

[0032] Virtual machine migration refers to a technology of migrating a virtual machine from a source host to the destination host. The virtual machine migration is divided into live migration and cold migration. The live migration refers to migrating a running virtual machine from the source host to the destination host, and downtime required by the virtual machine during a process of live migration is typically short enough to be virtually imperceptible to users. The cold migration refers to migrating a shut-down virtual machine from the source host to the destination host, and the cold migration requires the virtual machine to be shut down for a relatively long period of time.

[0033] Memory data migration: during a process of the virtual machine migration, the memory data of the virtual machine needs to be migrated from the source host to the destination host. Typically, since the cold migration requires the virtual machine to be shut down and the memory is transitory the cold migration typically does not involve the memory data migration. The live migration, on the other hand, does not require the virtual machine to be shut down, and therefore the memory data of the virtual machine needs to be fully and correctly migrated to the destination host, to ensure that the virtual machine is able to run correctly on the destination host after the memory data migration is completed.

[0034] Memory virtualization is a virtualization technology that allows the physical memory to be expanded into a logically larger memory space, so that programs are able to access more memory resources. This technology divides virtual address space into fixed-size pages (referred to as virtual memory pages), and maps these virtual memory pages to the physical memory, so that each process or resource (e.g., the virtual machine) is able to have its own virtual memory space.

[0035] A page fault is a memory exception derived from memory virtualization technology. The page fault typically refers to a virtual memory page not being mapped to a physical memory page, and therefore the memory data corresponding to the virtual memory page is inaccessible. However, this memory exception is recoverable, and a operation of recovering from the page fault is a page fault recovery operation.

[0036] The page fault recovery operation is to reallocate the physical memory page for the page-faulted virtual memory page, and restore the memory data into the reallocated physical memory page.

[0037] During a research process, the inventors found that, currently, during the process of the virtual machine migration, when the virtual machine manager in the source host performs memory data migration, the virtual machine manager traverses the virtual memory pages occupied by the virtual machine and indiscriminately sends memory access requests to these virtual memory pages. For the memory access requests sent to page-faulted virtual memory pages, the page fault recovery operation is caused in an operating system of the source host, and the page fault recovery operation needs to occupy physical memory in the source host. Therefore, as described in the background, the memory data migration causes relatively large fluctuations in the usage of physical memory in the source host. The inventors also found during the research process that a reason for the virtual machine migration is typically insufficient physical memory in the source host, and it is expected to improve an insufficient physical memory problem for the source host by migrating out partial virtual machines. However, under such circumstances, the memory data migration in turn needs to occupy a relatively large amount of physical memory in the source host, which aggravates the insufficient physical memory problem for the source host and further exacerbates the memory contention problem in the source host.

[0038] To this end, the present embodiment provides a virtual machine memory data migration method, by improving a process of memory data migration, which is able to effectively reduce physical memory occupation caused in the memory data migration.

[0039] The technical solutions provided by the embodiments of the present disclosure are described in detail below with reference to the drawings.

[0040] FIG. 1 is a schematic flowchart of a virtual machine memory data migration method provided by some exemplary embodiments of the present disclosure, and FIG. 2 is a schematic logic diagram of a virtual machine memory data migration method provided by some exemplary embodiments of the present disclosure. The virtual machine memory data migration method is performed by a virtual machine manager in a source host, the virtual machine manager is implemented as software, hardware, or a combination of software and hardware, and the virtual

machine manager is integrated in the source host. Referring to FIG. 1, the virtual machine memory data migration method includes the following steps.

[0041] In step 100, in response to executing an initial migration for a target virtual memory page, a page fault state of the target virtual memory page is detected.

[0042] In step 101, in response to the target virtual memory page being in the page fault state, memory data corresponding to the target virtual memory page is acquired without triggering a page fault recovery operation.

[0043] In step 102, the memory data is migrated to the destination host.

[0044] In the present embodiment, a virtual machine manager (VMM) adopts a hypervisor or a Quick Emulator (QEMU)-Kernel-based Virtual Machine (KVM), or the like. Furthermore, the virtual machine manager typically includes a component located in a kernel mode and a component located in a user mode. In the present embodiment, the component located in the user mode included in the virtual machine manager is described as a user-mode component, for example, the QEMU component; and the component located in the kernel mode included in the virtual machine manager is described as a kernel-mode component, for example, the KVM component. The virtual machine memory data migration method provided in the present embodiment is primarily executed by the user-mode component in the virtual machine manager.

[0045] The virtual machine memory data migration method provided in the present embodiment is applied to a scenario of virtual machine live migration mentioned above. Of course, in a case where a memory data migration requirement also exists in a scenario of virtual machine cold migration, the virtual machine memory data migration method provided in the present embodiment is equally applied to the scenario of virtual machine cold migration. In other words, the present embodiment does not limit the application scenario.

[0046] During the research process, the inventors found that the memory data migration process roughly includes two migration phases: a full initial migration phase and a dirty page migration phase.

[0047] The full initial migration phase is understood as: after migration work begins, full memory data of the virtual machine is traversed, and the full memory data is migrated from the source host to the destination host. The dirty page migration phase is understood as: after a

certain virtual memory page completes the initial migration, since the virtual machine is still running normally in the source host, relevant operations occurring in the virtual machine cause memory data corresponding to partial virtual memory pages to change, these virtual memory pages whose memory data has changed are dirty pages, and the dirty pages need to be migrated to the destination host. Moreover, the dirty page migration phase requires multiple rounds of migration, until it is detected that time required by the latest round of dirty page migration is less than a specified time threshold, and after the latest round of dirty page migration is completed, the dirty page migration phase is ended.

[0048] On this basis, improvements to the full initial migration phase are proposed in the present embodiment to reduce the physical memory occupation caused in the full initial migration phase.

[0049] It is worth noting that, during the full initial migration phase, the initial migration needs to be performed for virtual memory pages occupied by the virtual machine. For ease of description, in the present embodiment, a target virtual memory page among the virtual memory pages is taken as an example to describe a process of the initial migration in detail. It should be understood that the virtual machine memory data migration method provided in the present embodiment is applied to any virtual memory page occupied by a virtual machine to be migrated.

[0050] Referring to FIG. 1, the step 100 proposes that, in a case where the initial migration is performed for the target virtual memory page, a page fault state of the target virtual memory page is detected. It is understood that, in the present embodiment, when the initial migration is performed for the target virtual memory page, a memory access request for the target virtual memory page is not directly sent; instead, the page fault state of the target virtual memory page is detected first.

[0051] In the present embodiment, in step 100, multiple implementation manners are used for detecting the page fault state of the target virtual memory page. FIG. 3 is a logic schematic diagram illustrating an optional implementation manner of a virtual machine memory data migration method according to some exemplary embodiments of the present disclosure. Referring to FIG. 3, an exemplary implementation manner in step 100 are the following steps.

[0052] The virtual machine manager sends, to a kernel interface for supporting memory data migration in the source host, a page fault state query instruction for the target virtual memory

page, where the page fault state query instruction is used for triggering the kernel interface to return page fault state description information corresponding to the target virtual memory page.

[0053] In response to the page fault state description information indicating that the target virtual memory page is in the page fault state, it is determined that the target virtual memory page is detected to be in the page fault state.

[0054] It should be understood that the physical memory in the source host is managed in a kernel state, and therefore the memory data migration requires support of partial interfaces in the kernel state of the source host. In the present embodiment, such interfaces are described as kernel interfaces in the source host that are used for supporting the memory data migration. The virtual machine manager, in cooperation with the kernel interfaces, completes the memory data migration. The kernel interfaces in the source host that are used for supporting the memory data migration are kernel interfaces provided by the operating system of the source host, for example, a memory manager (MM), or the like. Of course, the kernel interfaces for the memory data migration in the source host are also interfaces provided by a kernel-state component in the virtual machine manager, for example, a memory access interface provided by a KVM component, or the like. There is no limitation herein on the provider of the kernel interfaces in the source host that are used for supporting the memory data migration. Correspondingly, the virtual machine manager has calling permissions for relevant interfaces provided by the kernel state of the source host; for example, a QEMU component has calling permissions for various interfaces provided by the KVM component, and also has calling permissions for partial kernel interfaces provided by the operating system of the source host.

[0055] In the exemplary implementation manner, a communication protocol between the virtual machine manager and the kernel interface is improved, so that both parties reach a consistent understanding of the page fault state query instruction. In practical applications, the virtual machine manager sends the page fault state query instruction in an instruction format agreed upon with the kernel interface, and the kernel interface determines, based on the instruction format, whether a received instruction is the page fault state query instruction. The instruction format, for example, is carrying a special identifier or field in the received instruction, and there is no limitation herein. During the research process, the inventors found that the virtual machine manager is used for managing each virtual machine on the source host, and therefore the virtual

machine manager is aware of which virtual memory pages are occupied by the virtual machine to be migrated in a virtual memory space provided by the source host. An address of a virtual memory page is a host virtual address (HVA). Based on this, in the exemplary implementation manner, an identifier of the target virtual memory page, for example, the HVA, is carried in the page fault query instruction, so that the page fault query instruction is directed to the target virtual memory page.

[0056] Furthermore, in the exemplary implementation manner, processing logic for page fault state query is further introduced in the kernel interface, so that after the page fault state query instruction is received, the kernel interface executes the processing logic for page fault state query. The processing logic for page fault state query in the kernel interface includes the following steps.

[0057] In response to the page fault state query instruction sent by the virtual machine manager when performing the initial migration for the target virtual memory page, page fault state description information is generated for the target virtual memory page; and the page fault state description information is provided to the virtual machine manager as a response result to the page fault state query instruction.

[0058] During the research process, the inventors also found that the kernel interface is responsible for address translation between a host virtual address (HVA) and a host physical address (HPA) during the process of the memory data migration, and the address translation is accomplished by relying on a page table. For example, the memory manager (MM) maintains a memory mapping (MMap), which is a type of page table. In the page table maintained by the kernel interface, a page table entry maintained for a virtual memory page includes a presence flag bit (represented by a present flag bit). When the presence flag bit=1, it indicates that a valid translation from the HVA to the HPA is performed, or in other words, it indicates that the HPA to which the HVA is mapped exists, and therefore it is determined that the virtual memory page corresponding to the presence flag bit is not in the page fault state; when the presence flag bit=0, it indicates that the valid translation from the HVA to the HPA is not performed, or in other words, it indicates that the HPA to which the HVA is mapped does not exist, and therefore it is determined that the virtual memory page corresponding to the presence flag bit is in the page fault state.

[0059] On this basis, the kernel interface queries the page table; and the page fault state of the target virtual memory page is determined based on a value of the presence flag bit associated with the target virtual memory page. Correspondingly, the page fault state description information generated by the kernel interface for the target virtual memory page indicates the page fault state of the target virtual memory page.

[0060] In the exemplary implementation manner, for the virtual machine manager, the page fault state description information returned by the kernel interface is parsed; and in response to the page fault state description information indicating that the target virtual memory page is in the page fault state, it is determined that the target virtual memory page is detected to be in the page fault state.

[0061] It is understood that, in the exemplary implementation manner, the virtual machine manager triggers the kernel interface to query the page fault state of the target virtual memory page, by calling the kernel interface for supporting the memory data migration in the source host, so that the virtual machine manager acquires the page fault state of the target virtual memory page from the kernel interface, and accuracy is ensured.

[0062] In the present embodiment, in addition to the exemplary implementation manners, in step 100, the virtual machine manager also adopts other exemplary implementation manners to detect the page fault state of the target virtual memory page. For example, the virtual machine manager reads, from the kernel interface, a page table entry corresponding to the target virtual memory page, and based on this, the virtual machine manager detects the page fault state of the target virtual memory page based on the value in the presence flag bit in the page table entry. As another example, there are usually other interfaces in the kernel state of the source host that are capable of providing a mapping relationship between the virtual memory page and the physical memory page, and the virtual machine manager also communicates with such interfaces in the kernel state of the source host that are not for supporting the memory data migration, to obtain required information from such interfaces, thereby the page fault state of the target virtual memory page is detected. No further elaboration or additional examples are provided herein.

[0063] Continuing to refer to FIG. 1, the step 101 proposes that, in response to the target virtual memory page being in a page fault state, the memory data corresponding to the target virtual memory page is acquired without triggering a page fault recovery operation.

[0064] During the research process, the inventors found that the page fault recovery operation is caused by sending the memory access request for the virtual memory page that is in the page fault state. To this end, in step 101, after determining that the target virtual memory page is in the page fault state, the virtual machine manager no longer sends the memory access request for the target virtual memory page, so that the page fault recovery operation is not triggered. Accordingly, in the present embodiment, a memory access manner is no longer used for acquiring memory data that has been swapped out, so that triggering the page fault recovery operation is avoided. Other than the memory access manner, any other data acquiring manner capable of acquiring the memory data that has been swapped out is applied to the present embodiment, for acquiring the memory data that has been swapped out. It should be understood that the present embodiment does not limit the data acquiring manner that is adopted in step 101, and exemplary data acquiring manners are provided in subsequent embodiments.

[0065] Furthermore, as mentioned above, memory management in the source host is performed in the kernel state, and therefore an actual storage location of the memory data corresponding to the virtual memory page in the page fault state is managed in the kernel state of the source host. On this basis, in step 101, based on a communication capability between the virtual machine manager and the kernel state of the source host that the virtual machine manager already possesses, the virtual machine manager seamlessly determines the actual storage location of the memory data corresponding to the target virtual memory page, and further implements access logic capable of reaching the actual storage location, thereby the memory data corresponding to the target virtual memory page is acquired.

[0066] It is seen that, in the present embodiment, in step 101, in response to the target virtual memory page having the memory data corresponding to the target virtual memory page, the virtual machine manager acquires data from the actual storage location of the memory data, instead of acquiring data through the memory access request. This not only ensures that the virtual machine manager is able to acquire the memory data corresponding to the virtual memory page in the page fault state, but also ensures that the page fault recovery operation is not triggered. Therefore, when the virtual machine manager acquires the memory data corresponding to the virtual memory page in the page fault state, no incremental occupation of physical memory pages in the source host is caused.

[0067] Continuing to refer to FIG. 1, in step 102, the memory data is migrated to the destination host.

[0068] In the present embodiment, from a perspective of the entire full initial migration phase, the virtual machine manager traverses and detects the page fault state of each virtual memory page occupied by the virtual machine to be migrated. For virtual memory pages that are not in the page fault state, the virtual machine manager sends the memory access requests for such virtual memory pages according to a traditional migration manner, to acquire corresponding memory data from the physical memory pages to which such virtual memory pages are mapped; it is understood that the memory access requests sent for such virtual memory pages do not cause incremental occupation of the physical memory pages in the source host. For the virtual memory pages that are in the page fault state, the virtual machine manager no longer sends the memory access requests according to the traditional manner, but instead acquires data from the actual storage location of the memory data; this change in the data acquiring manner not only ensures that the virtual machine manager is able to acquire the memory data corresponding to the virtual memory pages in the page fault state, but also ensures that the page fault recovery operation is not triggered, and therefore no incremental occupation of the physical memory pages in the source host is caused. This ensures that, throughout the entire full initial migration phase, the virtual machine manager does not trigger the page fault recovery operation, and therefore no incremental occupation of the physical memory pages in the source host is caused.

[0069] Corresponding to the dirty page migration phase included in the memory data migration process. In the present embodiment, after completing the initial migration for the target virtual memory page, in response to detecting that the target virtual memory page has become a dirty page, the virtual machine manager acquires dirty page data corresponding to the target virtual memory page from a physical memory page to which the target virtual memory page is mapped; and migrates the dirty page data acquired for the target virtual memory page to the destination host. Herein, the technical concept involved, i.e., the dirty page, refers to a memory page that has been modified.

[0070] It should be understood that, in a process of managing the virtual machine, the virtual machine manager detects the memory access request sent by the virtual machine for the virtual memory pages, and as mentioned above, in response to the target virtual memory page being in

the page fault state, the memory access request causes the page fault recovery operation to allocate the physical memory page for the target virtual memory page, and the memory data corresponding to the target virtual memory page is stored in the physical memory page. It is worth emphasizing that the page fault recovery operation occurring in this case is caused by the memory access request sent by the virtual machine, and such a memory access request is required during normal operation of the virtual machine, and therefore the occupation of the physical memory page in the source host resulting therefrom is reasonable and is not intervened in the present embodiment. In this way, in response to the virtual machine manager detecting that the target virtual memory page has become the dirty page, the target virtual memory page necessarily is no longer in the page fault state, and correspondingly, the virtual machine manager normally sends the memory access request for the target virtual memory page to acquire the dirty page data corresponding to the target virtual memory page and migrate the dirty page data to the destination host.

[0071] It is seen that in the present embodiment, during the dirty page migration phase, when the virtual machine manager performs dirty page data migration, no additional occupation of the physical memory pages in the source host is caused.

[0072] In summary, in the present embodiment, the memory data migration scheme during the virtual machine migration is improved, and it is proposed to detect the page fault state of the virtual memory pages when performing the initial migration for each virtual memory page in the virtual machine. For the virtual memory pages that are in the page fault state, the virtual machine manager in the source host acquires the memory data for such virtual memory pages without triggering the page fault recovery operation. In this way, during the process of the virtual machine migration, even if partial virtual memory pages are in the page fault state, no page fault recovery operation is triggered, thereby the occupation of the physical memory in the source host caused by the page fault recovery operation is avoided, and amount of occupation of the physical memory in the source host caused by the memory data migration is reduced. Accordingly, the memory data migration no longer causes significant fluctuation in the usage of the physical memory in the source host, thereby exacerbation of the memory contention problem in the source host is avoided.

[0073] In the above or below embodiments, step 101 employs various data acquisition schemes to acquire the memory data corresponding to the target virtual memory page without triggering the page fault recovery operation.

[0074] During the research process, the inventors found that there are many causes leading to the virtual memory page being in the page fault state, and under different causes, the actual storage location of the memory data corresponding to the virtual memory page differs. Therefore, an exemplary data acquisition scheme is proposed: for page fault states caused by the different causes, an adapted access logic is adopted to reach the actual storage location of the memory data, so as to acquire the memory data. Herein, the causes leading to the page fault state include, but are not limited to, swap and deferred memory allocation.

[0075] To this end, the data acquisition scheme proposes: in response to the memory data corresponding to the target virtual memory page having been swapped out to the swap space corresponding to the source host, the memory data corresponding to the target virtual memory page is acquired from the swap space without triggering the page fault recovery operation; and in response to the target virtual memory page not having been allocated a physical memory page in the source host, an empty file is determined as the memory data corresponding to the target virtual memory page.

[0076] In this data acquisition scheme, several possible actual storage locations of the memory data are provided for different causes of the page fault state: one is being swapped out to the swap space, and the other is having no actual storage location allocated (i.e., no memory data exists under the virtual memory page). Herein, several technical concepts are involved and are explained here. Swap is a swap technique in which, when the physical memory in a host is insufficient, a portion of memory data is saved to the swap space to release the physical memory in the host. The swap space is a technique used for virtual memory in a computer system, which allows an operating system to use disk space as temporary memory. When the physical memory in the host is insufficient, the swap space saves memory pages that are temporarily inactive, to release the physical memory for use by other programs. The swap space is typically located on a hard disk. Deferred memory allocation is a memory allocation mechanism. For the virtual machine on the host, before memory allocation, only a promise of memory space usage is acquired rather than actually allocating the physical memory to the virtual machine; when the

virtual machine needs to use memory, one or a group of physical memory pages are allocated to the virtual machine in the kernel mode of the host. In other words, partial virtual memory pages occupied by the virtual machine not yet have been accessed by the virtual machine, and such virtual memory pages are not allocated the physical memory pages; only when the virtual machine sends an access to such a virtual memory page, the kernel mode of the host is triggered to allocate a physical memory page for the virtual memory page. It is understood that, in response to the virtual memory page not having been allocated the physical memory, it indicates that no memory data has been generated by the virtual machine under the virtual memory page.

[0077] In this data acquisition scheme, access logic for acquiring the memory data is also provided for different actual storage locations respectively. In the case where the memory data corresponding to the target virtual memory page is located in the swap space, the virtual machine manager acquires the memory data corresponding to the target virtual memory page from the swap space. In the case where the target virtual memory page has not been allocated a physical memory page in the source host, it indicates that no memory data has been generated under the target virtual memory page, and the virtual machine manager directly determines the empty file as the memory data corresponding to the target virtual memory page.

[0078] Furthermore, in the present embodiment, a preferred implementation scheme for acquiring the memory data corresponding to the target virtual memory page from the swap space is also provided. The preferred implementation scheme proposes: the virtual machine manager sends a data acquisition request for the target virtual memory page to a kernel interface for supporting memory data migration in the source host, where the data acquisition request is used for triggering the kernel interface to read the memory data corresponding to the target virtual memory page from the swap space; and acquires the memory data corresponding to the target virtual memory page provided by the kernel interface.

[0079] In other words, in the preferred implementation scheme, the virtual machine manager does not need to directly access the swap space, but uses the kernel interface in the source host to read required memory data from the swap space, so that the virtual machine manager acquires the memory data corresponding to the target virtual memory page from the kernel interface.

[0080] Referring to FIG. 3, to further facilitate the virtual machine manager in acquiring the required memory data from the kernel interface, the preferred implementation scheme further

proposes: to support data transfer between the virtual machine manager and the kernel interface, a cache space is preset; on this basis, the virtual machine manager allocates a cache address for the target virtual memory page in the preset cache space; and carries an identifier of the target virtual memory page and the cache address in the data acquisition request, to trigger the kernel interface to read the memory data corresponding to the target virtual memory page from the swap space and store the memory data in the cache address in the preset cache space. For the kernel interface, according to the identifier of the target virtual memory page carried in the data acquisition request, the memory data corresponding to the target virtual memory page is hit in the swap space, thereby the memory data corresponding to the target virtual memory page is accurately read. Based on this, the virtual machine manager reads the memory data corresponding to the target virtual memory page from the cache address. This effectively improves efficiency of the virtual machine manager acquiring the memory data corresponding to the target virtual memory page from the swap space, reduces logic complexity in the virtual machine manager, and makes full use of the kernel interface in the source host.

[0081] It should be understood that the several possible actual storage locations of the memory data described above are merely exemplary, and the present embodiment is not limited thereto. Moreover, the access logic provided for the several possible actual storage locations of the memory data is also merely exemplary, and the present embodiment is not limited thereto either, as long as the virtual machine manager is able to adopt appropriate access logic to reach the actual storage location of the memory data corresponding to the virtual memory in the page fault state.

[0082] Furthermore, to enable the virtual machine manager to more conveniently and more efficiently determine the cause of the target virtual memory page being in the page fault state, a preferred scheme is also proposed in the present embodiment. The preferred scheme inherits basic concept mentioned above of sending a page fault state query command to the kernel interface in the source host for supporting memory data migration in the source host. Based on this, the preferred scheme proposes that the page fault state description information returned by the kernel interface indicates a page fault type. Herein, the page fault type corresponds to the cause of the page fault state. For example, in response to the cause of the page fault state being swap, the page fault type corresponding to the cause is a memory swap type; in response to the

cause of the page fault state being deferred memory allocation, the page fault type corresponding to the cause is a deferred allocation type.

[0083] On this basis, in the preferred scheme: the virtual machine manager determines the page fault type corresponding to the target virtual memory page by parsing the page fault state description information returned by the kernel interface for the target virtual memory page; in response to the page fault state description information indicating that the page fault type corresponding to the target virtual memory page is the memory swap type, determines that the memory data corresponding to the target virtual memory page has been swapped out to the swap space; and in response to the page fault state description information indicating that the page fault type corresponding to the target virtual memory page is the deferred allocation type, determines that the target virtual memory page has not been allocated the physical memory page.

[0084] FIG. 4 is a schematic structural diagram of page fault state description information according to some exemplary embodiments of the present disclosure. Referring to FIG. 4, the page fault state description information includes a page fault state flag identifier field and a page fault type field.

[0085] Based on this, in response to a value of the page fault state identifier field being a first value, it is indicated that the target virtual memory page is in the page fault state; in response to the value of the page fault state identifier field being a second value, it is indicated that the target virtual memory page is not in the page fault state. In practical applications, the first value is 1 and the second value is 0. And, in response to a value of the page fault type field being a third value, it is indicated that the page fault type of the target virtual memory page is the memory swap type; in response to the value of the page fault type field being a fourth value, it is indicated that the page fault type of the target virtual memory page is the deferred allocation type.

[0086] FIG. 4 also shows other fields in the page fault state description information, such as an information type field, used for indicating that current information is the page fault state description information for identification by the virtual machine manager. No further examples of other fields that is included in the page fault state description information are provided herein.

[0087] It should be understood that, in addition to the preferred scheme, other implementation schemes are also adopted in the present embodiment to support the virtual machine manager in determining the cause of the target virtual memory page being in the page fault state. For

example, the virtual machine manager applies to the kernel mode of the source host for page table monitoring permission, to maintain monitoring of modification events in the page table, so that the virtual machine manager determines, based on the modification events, which virtual memory pages have experienced which type of event that causes a page fault, and thus the cause of the page fault state is determined for the virtual memory page in the page fault state. No further examples of implementation schemes are provided herein.

[0088] In summary, in the present embodiment, the virtual machine manager fully interacts with the kernel interface in the source host for supporting memory data migration in the source host, so as to use the kernel interface to provide the virtual machine manager with a reference basis for detecting the page fault state, and to use the kernel interface to read the memory data for the virtual memory page in the page fault state. This enables the virtual machine manager to more efficiently detect the page fault state of the virtual memory page and to more efficiently acquire the memory data corresponding to the virtual memory page in the page fault state without triggering the page fault recovery operation.

[0089] FIG. 5 is a flowchart of a method for virtual machine memory data migration provided by another exemplary embodiment of the present disclosure. The virtual machine memory data migration method is applied to a kernel interface in a source host that is used for supporting memory data migration. Referring to FIG. 5, the virtual machine memory data migration method includes that:

[0090] In step 500, a page fault state query instruction sent by a virtual machine manager in the source host when performing an initial migration on a target virtual memory page is received.

[0091] In step 501, according to the page fault state query instruction, page fault state description information is generated for the target virtual memory page, where the page fault state description information is used for indicating a page fault state of the target virtual memory page.

[0092] In step 502, the page fault state description information is provided to the virtual machine manager, so that after the target virtual memory page is detected to be in the page fault state, memory data corresponding to the target virtual memory page is acquired without triggering a page fault recovery operation, and the memory data is migrated to the destination host.

[0093] It should be understood that, in the present embodiment, the kernel interface still maintains its original work related to the memory data migration, for example, responding to the memory access request for the virtual memory page, and the original work related to the memory data migration undertaken by the kernel interface is not elaborated in detail herein.

[0094] In the present embodiment, processing logic shown in FIG. 5 is added to the kernel interface, so that the kernel interface is able to cooperate with the virtual machine manager to support the virtual machine manager in avoiding triggering the page fault recovery operation during a process of the memory data migration.

[0095] The following merely further clarifies partial technical details in the kernel interface.

[0096] In some optional embodiments, the page fault state description information is generated for the target virtual memory page includes that: a page table is queried; and based on a value on a presence flag bit associated with the target virtual memory page, the page fault state of the target virtual memory page is determined, so as to generate the page fault state description information for the target virtual memory page.

[0097] In some optional embodiments, the page fault state description information is further used for indicating a page fault type. The page fault state description information indicates that the page fault type corresponding to the target virtual memory page is the memory swap type, so that the virtual machine manager determines that the memory data corresponding to the target virtual memory page has been swapped out to the swap space. The page fault state description information indicates that the page fault type corresponding to the target virtual memory page is the deferred allocation type, so that the virtual machine manager determines that the target virtual memory page has not been allocated the physical memory page.

[0098] In some optional embodiments, the page fault state description information includes the page fault state identifier field and the page fault type field. In response to the value of the page fault state identifier field being a first value, it is indicated that the target virtual memory page is in the page fault state. In response to the value of the page fault state identifier field being a second value, it is indicated that the target virtual memory page is not in the page fault state. In response to the value of the page fault type field being a third value, it is indicated that the page fault type of the target virtual memory page is the memory swap type. In response to the value of

the page fault type field being a fourth value, it is indicated that the page fault type of the target virtual memory page is the deferred allocation type.

[0099] In some optional embodiments, in response to the data acquisition request sent by the virtual machine manager for the target virtual memory page, the memory data corresponding to the target virtual memory page is read from the swap space and provided to the virtual machine manager.

[0100] In some optional embodiments, in response to the data acquisition request sent by the virtual machine manager carrying an identifier of the target virtual memory page and a cache address, the memory data corresponding to the target virtual memory page read from the swap space is stored at the cache address, so that the virtual machine manager reads the memory data corresponding to the target virtual memory page from the cache address.

[0101] It is worth noting that, for more specific technical details regarding the kernel interface, reference is made to the descriptions above, which are not elaborated in detail herein, but this should not cause a loss to the scope of protection of the present disclosure.

[0102] In addition, some processes described in the above embodiments and the drawings include multiple operations appearing in a specific order, but it should be clearly understood that these operations is performed in an order different from the order in which they appear herein or is performed in parallel. Serial numbers of the operations, such as 801, 802, etc., are merely used for distinguishing respective different operations, and the serial numbers themselves do not represent any execution order. In addition, these processes include more or fewer operations, and these operations is performed sequentially or in parallel. It should be noted that the descriptions of “first”, “second”, and the like herein are used for distinguishing different values and the like, do not represent a sequential order, and do not limit the “first” and the “second” to being different types.

[0103] FIG. 6 is a schematic structural diagram of a computing device provided by yet another exemplary embodiment of the present disclosure. As shown in FIG. 6, the computing device is a source host where a virtual machine to be migrated is located. The computing device includes that: a memory 60, a processor 61, and a communication component 62. The processor 61 is coupled with the memory 60 and the communication component 62, and is configured to execute a computer program in the memory 60.

[0104] In some design solutions, the processor 61 executes the computer program in the memory 60 to serve as the virtual machine manager in the source host. In this scenario, the processor 61 is configured to:

[0105] detect the page fault state of the target virtual memory page, in response to executing the initial migration for the target virtual memory page;

[0106] acquire the memory data corresponding to the target virtual memory page without triggering the page fault recovery operation, in response to the target virtual memory page being in the page fault state; and

[0107] migrate the memory data to the destination host.

[0108] In some optional embodiments, when acquiring the memory data corresponding to the target virtual memory page without triggering the page fault recovery operation, the processor 61 is configured to:

[0109] acquire the memory data corresponding to the target virtual memory page from the swap space without triggering the page fault recovery operation, in response to the memory data corresponding to the target virtual memory page having been swapped out to the swap space corresponding to the source host; and

[0110] determine the empty file as the memory data corresponding to the target virtual memory page, in response to the target virtual memory page not having been allocated the physical memory page in the source host.

[0111] In some optional embodiments, the processor 61, when detecting the page fault state of the target virtual memory page, is configured to:

[0112] send the page fault state query instruction for the target virtual memory page to the kernel interface in the source host for supporting memory data migration in the source host, where the page fault state query instruction is used for triggering the kernel interface to return the page fault state description information corresponding to the target virtual memory page; and

[0113] determine that the target virtual memory page is detected to be in the page fault state, in response to the page fault state description information indicating that the target virtual memory page is in a page fault state.

[0114] In some optional embodiments, the page fault state description information is further used for indicating a page fault type corresponding to the target virtual memory page, and the processor 61 is further configured to:

[0115] determine that the memory data corresponding to the target virtual memory page has been swapped out to the swap space, in response to the page fault type being a memory swap type; and

[0116] determine that the target virtual memory page has not been allocated the physical memory page has not been allocated the physical memory page, in response to the page fault type being the deferred allocation type.

[0117] In some optional embodiments, the page fault state description information includes the page fault state identifier field and the page fault type field;

[0118] indicate that the target virtual memory page is in the page fault state, in response to the value of the page fault state identifier field being the first value; indicate that the target virtual memory page is not in the page fault state, in response to the value of the page fault state identifier field being the second value;

[0119] indicate that the page fault type of the target virtual memory page is the memory swap type, in response to the value of the page fault type field being the third value; and indicate that the page fault type of the target virtual memory page is the deferred allocation type, in response to the value of the page fault type field being the fourth value.

[0120] In some optional embodiments, the processor 61, when acquiring the memory data corresponding to the target virtual memory page from the swap space, is configured to:

[0121] send the data acquisition request for the target virtual memory page to the kernel interface, where the data acquisition request is used for triggering the kernel interface to read the memory data corresponding to the target virtual memory page from the swap space; and

[0122] acquire the memory data corresponding to the target virtual memory page provided by the kernel interface.

[0123] In some optional embodiments, the processor 61, when sending the data acquisition request for the target virtual memory page to the kernel interface, is configured to:

[0124] allocate the cache address for the target virtual memory page in the preset cache space; and

[0125] carry the identifier of the target virtual memory page and the cache address in the data acquisition request, to trigger the kernel interface to read the memory data corresponding to the target virtual memory page from the swap space and to store the memory data at the cache address in the preset cache space;

[0126] acquire the memory data corresponding to the target virtual memory page provided by the kernel interface, including that:

[0127] read the memory data corresponding to the target virtual memory page from the cache address.

[0128] In some optional embodiments, the processor 61 is further configured to:

[0129] acquire the memory data corresponding to the target virtual memory page from the physical memory page to which the target virtual memory page is mapped, in response to the target virtual memory page not being in a page fault state.

[0130] In some optional embodiments, the processor 61 is further configured to:

[0131] acquire the dirty page data corresponding to the target virtual memory page from the physical memory page to which the target virtual memory page is mapped, after completing the initial migration for the target virtual memory page, in response to detecting that the target virtual memory page has become the dirty page; and

[0132] migrate the dirty page data acquired for the target virtual memory page to the destination host.

[0133] In other design solutions, the processor 61 is configured to execute a computer program in the memory 60 to serve as a kernel interface for supporting memory data migration in the source host. In this case, the processor 61 is configured to:

[0134] receive the page fault state query instruction sent by the virtual machine manager in the source host when executing the initial migration for the target virtual memory page;

[0135] generate the page fault state description information for the target virtual memory page, according to the page fault state query instruction, where the page fault state description information is used for indicating the page fault state of the target virtual memory page; and

[0136] provide the page fault state description information to the virtual machine manager, so that after the target virtual memory page is detected to be in the page fault state, memory data corresponding to the target virtual memory page is acquired without triggering a page fault recovery operation, and the memory data is migrated to the destination host.

[0137] Furthermore, as shown in FIG. 6, the computing device further includes: a power supply component 63 and other components. FIG. 6 only schematically illustrates some components, which does not mean that the computing device only includes the components shown in FIG. 6.

[0138] It is worth noting that for the technical details in the various embodiments of the computing device, reference is made to the relevant descriptions in the foregoing method embodiments regarding the virtual machine manager and the kernel interface that is used for supporting memory data migration. To save space, details are not repeated herein, but this should not cause a loss of the scope of protection of the present disclosure.

[0139] Accordingly, embodiments of the present disclosure further provide a computer-readable storage medium storing a computer program, where the computer program, when executed, is capable of implementing the steps in the foregoing method embodiments.

[0140] Accordingly, embodiments of the present disclosure further provide a computer program product, including a computer program, where when the computer program is executed by one or more processors, the one or more processors are caused to perform:

[0141] in response to executing the initial migration for the target virtual memory page, the page fault state of the target virtual memory page is detected;

[0142] in response to the target virtual memory page being in the page fault state, the memory data corresponding to the target virtual memory page is acquired without triggering the page fault recovery operation;

[0143] the memory data acquired for the target virtual memory page is migrated to the destination host.

[0144] The computer program product provided in this embodiment is capable of detecting the page fault state of the virtual memory page when executing the initial migration for each virtual memory page in the virtual machine. For the virtual memory page in the page fault state, the virtual machine manager in the source host acquires the memory data corresponding to such a virtual memory page without triggering the page fault recovery operation. In this way, during the process of the virtual machine migration, even if partial virtual memory pages are in the page fault state, no page fault recovery operation is triggered, thereby the occupation of physical memory in the source host caused by the page fault recovery operation is avoided and the amount of physical memory in the source host occupied due to memory data migration is reduced. Accordingly, the memory data migration no longer causes significant fluctuation in the usage of physical memory in the source host, thereby exacerbation of a memory squeeze problem in the source host is avoided.

[0145] The memory in FIG. 6 described above is configured to store the computer program and is configured to store various other data to support operations on the computing platform. Examples of such data include instructions for any application or method operating on the computing platform, contact data, phonebook data, messages, pictures, videos, and the like. The memory is implemented by any type of transitory or non-transitory storage device or a combination thereof, such as a Static Random Access Memory (SRAM), an Electrically Erasable Programmable Read-Only Memory (EEPROM), an Electrically Programmable Read-Only Memory (EPROM), a Programmable Read-Only Memory (PROM), a Read-Only Memory (ROM), a magnetic memory, a flash memory, a magnetic disk, or an optical disk.

[0146] The communication component in FIG. 6 described above is configured to facilitate wired or wireless communication between the device where the communication component is located and other devices. The device where the communication component is located is connected with a wireless network based on a communication standard, such as WiFi, 2G, 3G, 4G/Long-Term Evolution (LTE for short), 5G, or other mobile communication networks, or a combination thereof. In some exemplary embodiments, the communication component receives a broadcast signal or broadcast-related information from an external broadcast management system via a broadcast channel. In some exemplary embodiments, the communication component further includes a Near Field Communication (NFC) module to facilitate short-range communication. For example, the NFC module is implemented based on Radio Frequency

Identification (RFID) technology, Infrared Data Association (IrDA) technology, Ultra Wideband (UWB) technology, Bluetooth (BT) technology, and other technologies.

[0147] The power supply component in FIG. 6 described above provides power to various components of the device where the power supply component is located. The power supply component includes a power management system, one or more power supplies, and other components associated with generating, managing, and distributing power for the device where the power supply component is located.

[0148] Those skilled persons in the art should understand that the embodiments of the present disclosure are provided as methods, systems, or computer program products. Therefore, the present disclosure adopts a form of a complete hardware embodiment, a complete software embodiment, or some embodiments combining software and hardware aspects. Moreover, the present disclosure adopts a form of a computer program product implemented on one or more computer-usable storage media (including but not limited to a disk memory, a Compact Disc Read-Only Memory (CD-ROM), an optical memory, and the like) containing computer-usable program code.

[0149] The present disclosure is described with reference to flowcharts and/or block diagrams of methods, apparatuses (systems), and computer program products according to embodiments of the present disclosure. It should be understood that at least one each flow or each block in the flowcharts and/or block diagrams, as well as combinations of flows and/or blocks in the flowcharts and/or block diagrams, is implemented by computer program instructions. These computer program instructions are provided to a processor of a general-purpose computer, a special-purpose computer, an embedded processor, or other programmable data processing device to produce a machine, such that the instructions executed by the processor of the computer or other programmable data processing device create means for implementing the functions specified in a flow or flows of the flowchart and/or a block or blocks of the block diagram.

[0150] These computer program instructions are also stored in a computer-readable memory that direct a computer or other programmable data processing device to operate in a particular manner, such that instructions stored in the computer-readable memory produce an article of

manufacture including instruction means, and the instruction means implement the functions specified in a flow or flows of the flowchart and/or a block or blocks of the block diagram.

[0151] These computer program instructions are also loaded onto a computer or other programmable data processing device to cause a series of operational steps to be performed on the computer or other programmable device, to produce a computer-implemented process, such that the instructions executed on the computer or other programmable device provide steps for implementing the functions specified in a flow or flows of the flowchart and/or a block or blocks of the block diagram.

[0152] It should also be noted that the terms “including”, or any other variations thereof are intended to cover a non-exclusive inclusion, such that a process, a method, an article, or a device that includes a list of elements includes not only those elements but also other elements not expressly listed, or also includes elements inherent to such a process, a method, article, or a device. In the absence of further limitations, an element defined by the phrase “including a...” does not preclude the presence of additional identical elements in the process, the method, the article, or the device that includes the element.

[0153] It should be noted that the user information (including but not limited to user device information, user personal information, etc.) and data (including but not limited to data used for analysis, stored data, displayed data, etc.) involved in the present disclosure are all information and data authorized by the user or fully authorized by all parties, and collection, use, and processing of relevant data are required to comply with relevant laws, regulations, and standards of relevant countries and regions, and operation interfaces corresponding to the relevant data are provided for the user to select to authorize or refuse.

[0154] The descriptions are merely embodiments of the present disclosure, and are not intended to limit the present disclosure. For those skilled persons in the art, the present disclosure is susceptible to various modifications and changes. Any modifications, equivalent substitutions, improvements, and the like made within the spirit and principles of the present disclosure should be included within the scope of protection of the present disclosure.

Industrial Applicability

[0155] In the method provided in the embodiments of the present disclosure, detection of a page fault state of each virtual memory page when executing an initial migration for each virtual memory page in a virtual machine is proposed. For each virtual memory page in the page fault state, the virtual machine manager in the source host acquires memory data corresponding to each virtual memory page without triggering a page fault recovery operation. In this way, during the process of virtual machine migration, even if partial virtual memory pages are in the page fault state, no page fault recovery operation is triggered, thereby occupation of physical memory in the source host caused by page fault recovery operations is avoided and occupation of physical memory in the source host caused by memory data migration is reduced. Accordingly, the memory data migration no longer causes significant fluctuations in the usage of physical memory in the source host, thereby exacerbation of memory contention problems in the source host is avoided.

What is claimed is:

1. A virtual machine memory data migration method, applied to a virtual machine manager in a source host, the virtual machine memory data migration method comprising:

in response to executing an initial migration for a target virtual memory page, detecting a page fault state of the target virtual memory page;

in response to the target virtual memory page being in the page fault state, acquiring memory data corresponding to the target virtual memory page without triggering a page fault recovery operation; and

migrating the memory data to a destination host.

2. The virtual machine memory data migration method as claimed in claim 1, wherein acquiring the memory data corresponding to the target virtual memory page without triggering the page fault recovery operation comprises:

in response to the memory data corresponding to the target virtual memory page having been swapped out to a swap space corresponding to the source host, acquiring the memory data corresponding to the target virtual memory page from the swap space without triggering the page fault recovery operation; and

in response to the target virtual memory page not having been allocated a physical memory page in the source host, determining an empty file as the memory data corresponding to the target virtual memory page.

3. The virtual machine memory data migration method as claimed in claim 1, wherein detecting the page fault state of the target virtual memory page comprises:

sending a page fault state query instruction for the target virtual memory page to a kernel interface for supporting memory data migration in the source host, wherein the page fault state query instruction is used for triggering the kernel interface to return page fault state description information corresponding to the target virtual memory page; and

in response to the page fault state description information indicating that the target virtual memory page is in the page fault state, determining that the target virtual memory page is detected to be in the page fault state.

4. The virtual machine memory data migration method as claimed in claim 3, wherein the page fault state description information is further used for indicating a page fault type corresponding to the target virtual memory page, the virtual machine memory data migration method further comprising:

in response to the page fault type being a memory swap type, determining that the memory data corresponding to the target virtual memory page has been swapped out to a swap space; and

in response to the page fault type being a deferred allocation type, determining that the target virtual memory page has not been allocated a physical memory page.

5. The virtual machine memory data migration method as claimed in claim 4, wherein the page fault state description information comprises a page fault state identifier field;

in response to a value of the page fault state identifier field being a first value, indicating that the target virtual memory page is in the page fault state.

6. The virtual machine memory data migration method as claimed in claim 1, wherein acquiring the memory data corresponding to the target virtual memory page without triggering the page fault recovery operation comprises:

in response to the memory data corresponding to the target virtual memory page having been swapped out to a swap space corresponding to the source host, sending a data acquisition request for the target virtual memory page to a kernel interface for supporting memory data migration in the source host without triggering the page fault recovery operation, wherein the data acquisition request is used for triggering the kernel interface to read the memory data corresponding to the target virtual memory page from the swap space; and

acquiring the memory data corresponding to the target virtual memory page provided by the kernel interface.

7. The virtual machine memory data migration method as claimed in claim 6, wherein sending the data acquisition request for the target virtual memory page to the kernel interface comprises:

allocating a cache address for the target virtual memory page in a preset cache space; and

carrying an identifier of the target virtual memory page and the cache address in the data acquisition request, to trigger the kernel interface to read the memory data corresponding to the target virtual memory page from the swap space and store the memory data in the cache address in the preset cache space.

8. The virtual machine memory data migration method as claimed in claim 1, wherein the virtual machine memory data migration method further comprises:

in response to the target virtual memory page not being in the page fault state, acquiring the memory data corresponding to the target virtual memory page from a physical memory page to which the target virtual memory page is mapped.

9. The virtual machine memory data migration method as claimed in claim 1, wherein the virtual machine memory data migration method further comprises:

after completing the initial migration for the target virtual memory page, in response to detecting that the target virtual memory page has become a dirty page, acquiring dirty page data corresponding to the target virtual memory page from a physical memory page to which the target virtual memory page is mapped; and

migrating the dirty page data for the target virtual memory page to the destination host.

10. A virtual machine memory data migration method, applied to a kernel interface for supporting memory data migration in a source host, the virtual machine memory data migration method comprising:

receiving a page fault state query instruction sent by a virtual machine manager in the source host when executing an initial migration for a target virtual memory page;

generating, according to the page fault state query instruction, page fault state description information for the target virtual memory page, wherein the page fault state description information is used for indicating a page fault state of the target virtual memory page; and

providing the page fault state description information to the virtual machine manager, so that after the target virtual memory page is detected to be in the page fault state, memory data corresponding to the target virtual memory page is acquired without triggering a page fault recovery operation, and the memory data is migrated to a destination host.

11. The virtual machine memory data migration method as claimed in claim 10, wherein the page fault state description information is further used for indicating a page fault type, the page fault type comprising a memory swap type or a deferred allocation type, the memory swap type is used for triggering the virtual machine manager to acquire memory data for the target virtual memory page from a swap space corresponding to the source host, and the deferred allocation type is used for triggering the virtual machine manager to determine an empty file as the memory data corresponding to the target virtual memory page.

12. The virtual machine memory data migration method as claimed in claim 4, wherein the page fault state description information comprises a page fault state identifier field;

in response to a value of the page fault state identifier field being a second value, indicating that the target virtual memory page is not in the page fault state.

13. The virtual machine memory data migration method as claimed in claim 4, wherein the page fault state description information comprises a page fault type field;

in response to a value of the page fault type field being a third value, indicating that the page fault type of the target virtual memory page is the memory swap type.

14. The virtual machine memory data migration method as claimed in claim 4, wherein the page fault state description information comprises a page fault type field;

in response to a value of the page fault type field being a fourth value, indicating that the page fault type of the target virtual memory page is the deferred allocation type.

15. The virtual machine memory data migration method as claimed in claim 7, wherein acquiring the memory data corresponding to the target virtual memory page provided by the kernel interface comprises:

reading the memory data corresponding to the target virtual memory page from the cache address.