ADDRESS TRANSLATION METHOD, FAST MAPPING TABLE CONSTRUCTION METHOD, DEVICE, STORAGE MEDIUM AND PROGRAM PRODUCT

The present disclosure claims priority to Chinese Patent Application No. 202410330180.3, filed with the China National Intellectual Property Administration on March 20, 2024 and entitled "ADDRESS TRANSLATION METHOD, RAPID MAPPING TABLE CONSTRUCTION METHOD, DEVICE, STORAGE MEDIUM, AND PROGRAM PRODUCT", the entire content of which is incorporated herein by reference.

TECHNICAL FIELD

The present disclosure relates to the field of virtualization technology, and in particular to an address translation method, a rapid mapping table construction method, device, storage medium, and program product.

BACKGROUND

Virtualization technology is a core technology of cloud computing. It creates one or more virtualization instances (also referred to as guests) on physical resources of a host to achieve the purpose of partitioning and centrally managing the physical resources. The host has a host operating system (Host Operating System, Host OS) and physical memory seen by the host (actual physical memory), and an address of the physical memory is an HPA (Host Physical Address, host physical address). A guest has a guest operating system (Guest OS) and physical memory seen by the guest, and an address of the physical memory is referred to as a GPA (Guest Physical Address, guest physical address).

In a virtualization environment, memory management is performed through EPT (Extended Page Tables, extended page tables), where by providing each guest with an independent multi-level page table structure, a GPA of the guest is translated into an HPA of the host, and isolation between different guests is achieved at the same time. The GPA of a guest is translated into the HPA of the host through the EPT, and access is directed to the correct physical memory location through a hardware-supported address translation

mechanism. However, this translation process often requires queries through multiple levels of page tables, and there is a problem of low efficiency.

SUMMARY

Various aspects of the present disclosure provide an address translation method, a rapid mapping table construction method, a device, a storage medium, and a program product, to address translation efficiency and achieve rapid address translation.

An embodiment of the present disclosure provides an address translation method, including: pre-constructing a first rapid mapping table corresponding to a memory page table, the memory page table including multi-level page tables corresponding to each of at least one guest physical address; the first rapid mapping table storing address information of a last-level page table entry corresponding to each of the at least one guest physical address; in response to a first address translation trigger event, determining a first guest physical address for which address translation is required, and querying, in the first rapid mapping table, address information of a last-level page table entry corresponding to the first guest physical address; and if the address information of the last-level page table entry corresponding to the first guest physical address is found by the query, generating a host physical address corresponding to the first guest physical address based on the address information of the last-level page table entry corresponding to the first guest physical address and an address offset of the first guest physical address.

An embodiment of the present disclosure further provides a rapid mapping table construction method, including: dividing an address space of guest physical addresses corresponding to a memory page table into at least one address space, and creating a base structure for each of the at least one address space, the base structure corresponding to a set page table level, the memory page table including multi-level page tables corresponding to each of at least one guest physical address; for any guest physical address in the memory page table, determining, based on address information of the any guest physical address and address spaces corresponding to base structures, a first structure corresponding to the any guest physical address from among the base structures; and recording, based on a relationship between a level of a last-level page table corresponding to the any guest physical address and the set page table level, through the first structure, address information of a last-level page table entry corresponding to the any guest physical address, to construct a first rapid mapping table corresponding to the memory page table, the first rapid mapping table storing address

information of a last-level page table entry corresponding to each of the at least one guest physical address.

An embodiment of the present disclosure further provides a physical machine, including: a memory and a processor; the memory is configured to store a computer program; the processor is coupled to the memory and configured to execute the computer program to implement the steps in the address translation method and the rapid mapping table construction method provided by the embodiments of the present disclosure.

An embodiment of the present disclosure further provides a computer-readable storage medium storing a computer program, where when the computer program is executed by a processor, the processor is caused to implement the steps in the address translation method and the rapid mapping table construction method provided by the embodiments of the present disclosure.

An embodiment of the present disclosure further provides a computer program product, including a computer program/instructions, where when the computer program/instructions are executed by a processor, the processor is caused to implement the steps in the address translation method and the rapid mapping table construction method provided by the embodiments of the present disclosure.

In the embodiments of the present disclosure, a first rapid mapping table adapted to a memory page table is constructed, and address information of a last-level page table entry corresponding to each of at least one guest physical address in the memory page table is stored in the first rapid mapping table, so that when address translation for a guest physical address is performed, the first rapid mapping table corresponding to the memory page table may be queried first, and if the address information of the last-level page table entry corresponding to the guest physical address is found by query in the first rapid mapping table, a host physical address corresponding to the guest physical address may be directly generated based on the address information of the last-level page table entry corresponding to the guest physical address and an address offset of the guest physical address, without the need to query level by level in multi-level page tables corresponding to the guest physical address included in the memory page table, which can improve address translation efficiency and achieve rapid address translation.

In some optional embodiments of the present disclosure, a second rapid mapping table adapted to the memory page table is further constructed, and address translation information of at least one host physical address to a corresponding guest physical address is stored in the second rapid mapping table, to support address translation from a host physical address to a

corresponding guest physical address, addressing the deficiency that only address translation from a guest physical address to a host physical address can be achieved based on the memory page table, and meeting address translation requirements from a host physical address to a guest physical address in various virtualization environments.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings described herein are used to provide a further understanding of the present disclosure and constitute a part of the present disclosure. The illustrative embodiments of the present disclosure and descriptions thereof are used to explain the present disclosure and do not constitute an improper limitation on the present disclosure. In the drawings:

FIG. 1a is a schematic flowchart of an address translation method provided by an exemplary embodiment of the present disclosure;

FIG. 1b is a schematic flowchart of a rapid mapping table construction method provided by an exemplary embodiment of the present disclosure;

FIG. 2a is a schematic structural diagram of a physical machine provided by an exemplary embodiment of the present disclosure;

FIG. 2b is a schematic structural diagram of another physical machine provided by an exemplary embodiment of the present disclosure;

FIG. 2c is a system structural diagram of the physical machine shown in FIG. 1b operating in root mode provided by an exemplary embodiment of the present disclosure;

FIG. 2d is a system structural diagram of the physical machine shown in FIG. 1b operating in non-root mode provided by an exemplary embodiment of the present disclosure;

FIG. 3a is a schematic flowchart of switching a host operating system from root mode to non-root mode provided by an exemplary embodiment of the present disclosure;

FIG. 3b is a schematic flowchart of creating a memory page table provided by an exemplary embodiment of the present disclosure;

FIG. 3c is a schematic flowchart of creating an information carrying object provided by an exemplary embodiment of the present disclosure;

FIG. 3d is a schematic flowchart of mode switching provided by an exemplary embodiment of the present disclosure;

FIG. 4 is a schematic structural diagram of an address translation apparatus provided by an exemplary embodiment of the present disclosure;

FIG. 5 is a schematic structural diagram of a rapid mapping table construction apparatus provided by an exemplary embodiment of the present disclosure;

FIG. 6 is a schematic structural diagram of a physical machine provided by an exemplary embodiment of the present disclosure.

DETAILED DESCRIPTION

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

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 users or fully authorized by all parties, and the collection, use, and processing of relevant data need to comply with relevant laws, regulations, and standards of relevant regions, and corresponding operation entries are provided for the users to choose to authorize or reject.

In a virtualization environment, address translation between GPA and HPA is performed through a memory page table. Optionally, the memory page table may be implemented as an EPT. In the following embodiments of the present disclosure, description is made taking the memory page table implemented as an EPT as an example, but not limited thereto. The EPT stores multi-level page tables corresponding to each of at least one GPA. However, there is a problem of low address translation efficiency. For this technical problem, in the embodiments of the present disclosure, a first rapid-mapping (Rapid-Mapping, RMAP) table is constructed for the EPT, and the first RMAP table stores address information of a last-level page table entry corresponding to each of at least one GPA, to establish a rapid mapping relationship between GPA and HPA, improve address translation efficiency, and achieve rapid address translation.

Further, the EPT only supports address translation from GPA to HPA, and cannot support reverse address translation from HPA to GPA. In embodiments of the present disclosure, a second RMAP table is created for the EPT, and the second RMAP table stores address translation information from at least one HPA to a corresponding GPA, to establish a rapid mapping relationship from HPA to GPA, improve address translation efficiency, and achieve rapid address translation.

Further, in the embodiments of the present disclosure, the construction process, management process, and state synchronization update process with the EPT of the first RMAP table and the second RMAP table are also described in detail, and reference may be made to the detailed description in the following embodiments.

It is stated here that the technical solutions provided by the embodiments of the present disclosure are applied in a virtualization environment. The virtualization environment refers to a runtime environment produced by virtualizing physical resources on a host using virtualization technology and creating one or more virtualization instances on the virtualized resources. In the embodiments of the present disclosure, relative to the host, the virtualization instance hosted on the virtualized resources are referred to as a guest. The guest has its own OS and GPA, and the host has its own OS and HPA.

In the embodiments of the present disclosure, the implementation manner of the virtualization environment is not limited. In an optional embodiment, implementation manner A of the virtualization environment is: adding a hypervisor in the host OS, where the hypervisor virtualizes hardware resources on the host and creates one or more virtual machines on the virtualized resources. In this virtualization environment, address translation between the GPA of the guest and the HPA of the host is performed through multi-level page tables corresponding to each of at least one GPA in the EPT.

In another optional embodiment, implementation manner B of the virtualization environment is: adding a virtualization layer between the hardware resources of the host and the host operating system, at least implementing a virtual resource providing function in the virtualization layer, where the virtualization layer switches the running host operating system from root mode (root) to non-root mode (non-root), provides at least one type of virtual resource in the non-root mode and at least runs on the virtual resource an application that the host operating system cannot run due to insufficient resources in the root mode. The virtual resource providing function can be flexibly implemented for the running host operating system without reinstalling the operating system, which solves the problem that applications cannot run due to insufficient resources caused by reasons such as not enabling or missing system

functions, supports the running of more applications, and is beneficial to expanding the capabilities of the host operating system.

The root mode is a working mode in which the host operating system has direct access privilege to hardware resources, and the non-root mode is a working mode in which the virtualization layer virtualizes hardware resources and schedules and accesses the virtualized resources on behalf of the host operating system. In the non-root mode, the access of the virtualization layer to hardware resources is restricted. Correspondingly, the host operating system in the non-root mode may also be referred to as a guest operating system. The virtualization layer maintains EPT, and the memory page table is used to store the mapping relationship between guest physical addresses in the non-root mode and host physical addresses in the root mode. The virtualization layer may convert GPA to HPA based on the multi-level page tables of the EPT.

In embodiments of the present disclosure, whether in the virtualization environment implemented in manner A or in the virtualization environment implemented in manner B, a first RMAP table may be created for the EPT, and rapid mapping from GPA to HPA may be implemented based on the first RMAP table. Further, a second RMAP table may also be created, and reverse mapping from HPA to GPA may be implemented based on the second RMAP table, solving the problem that the EPT cannot implement reverse mapping from HPA to GPA, and satisfying the requirement for reverse mapping from HPA to GPA.

In manner A, the process of creating the first RMAP table, performing address translation based on the first RMAP table, creating the second RMAP table, and performing address translation based on the second RMAP table may be completed by the host OS, and specifically, may be completed by the hypervisor in the host OS. In manner B, the process of creating the first RMAP table, performing address translation based on the first RMAP table, creating the second RMAP table, and performing address translation based on the second RMAP table may be completed by the virtualization layer in the non-root mode. Whether by the host OS in manner A or by the virtualization layer in manner B, the process of creating the first RMAP table, performing address translation based on the first RMAP table, creating the second RMAP table, and performing address translation based on the second RMAP table is the same or similar.

The following provides a detailed description of a solution provided by the embodiments of the present disclosure with reference to the accompanying drawings.

FIG. 1a is a schematic flowchart of an address translation method provided by an exemplary embodiment of the present disclosure. As shown in FIG. 1a, the method includes:

R11, pre-construct a first RMAP table corresponding to EPT, where the EPT includes multi-level page tables corresponding to each of at least one GPA; the first RMAP table stores address information of a last-level page table entry corresponding to each of the at least one GPA;

R12, in response to a first address translation trigger event, determine a first GPA for which address translation is required, and query, in the first RMAP table, address information of a last-level page table entry corresponding to the first GPA;

R13, if the address information of the last-level page table entry corresponding to the first GPA is found by query, generate an HPA corresponding to the first GPA based on the address information of the last-level page table entry corresponding to the first GPA and an address offset of the first GPA.

In this embodiment, on one hand, the EPT adopts a multi-level page table structure, and on the other hand, the EPT adopts mixed mapping granularity, supporting not only small page granularity but also large page granularity, which can meet the requirements of different physical page mappings and improve the flexibility of memory mapping. Specifically, on one hand, the EPT includes multi-level page tables corresponding to each of the at least one GPA, and the multi-level page tables may be 2-level memory page tables, 4-level memory page tables, or 5-level memory page tables, etc. Taking the EPT including 4-level memory page tables as an example, the EPT includes: page global directory (Page Global Directory, PGD), page upper directory (Page Upper Directory, PUD), page middle directory (Page Middle Directory, PMD), and page table entry (Page Table Entry, PTE). A page table entry pgd_t in a PGD records the address information of a lower-level page table (e.g., PUD); a page table entry pud_t in a PUD records the address information of a lower-level page table (e.g., PMD); a page table entry pmd_t in a PMD records the address information of a lower-level page table (e.g., PTE); and a page table entry pte_t in a PTE records the starting address of the physical page corresponding to the HPA. Of course, if a PGD, PUD, or PMD is a last-level page table, then page table entries respectively corresponding to the PGD, PUD, and PMD record the starting address of the physical page corresponding to the HPA. It should be noted that a GPA address may be divided into several parts, each serving as an index, to obtain multiple index values and an address offset, where the address offset refers to an offset relative to the corresponding physical page. Taking the EPT including 4-level memory page tables as an example, a GPA may be divided into 5 parts, including: a PGD-level index value, a PUD-level index value, a PMD-level index value, a PTE-level index value, and an address offset. Normally, the HPA

corresponding to the GPA can be obtained by querying the EPT based on each index value and the address offset in the GPA address.

On the other hand, the EPT is created with different page granularities for different address spaces (GPA). For example, for some address spaces, multi-level page tables of the EPT are constructed through small page granularity. If the small page granularity is 4k (corresponding to PTE page granularity), then the 4-level page tables corresponding to the address space are PGD, PUD, PMD, and PTE, and the last-level page table is PTE. For some other address spaces, the EPT is created using large page granularity. If the large page granularity is 2m (corresponding to PMD page granularity), then the 3-level page tables corresponding to the address space are PGD, PUD, and PMD, and the last-level page table is PMD. For still some other address spaces, the EPT is created using large page granularity. If the large page granularity is 1G (corresponding to PUD page granularity), then the 2-level page tables corresponding to this address space are PGD and PUD, and the last-level page table is PUD. That is to say, the EPT adopts mixed mapping granularity, and the last-level page table corresponding to any GPA in the EPT may be PGD, PUD, PMD, or PTE. For a detailed introduction to EPT page tables, reference may be made to the following embodiments, and details are not repeated here.

To facilitate recording the rapid mapping relationship between GPA and HPA, in this embodiment, a data structure of a first RMAP table is provided, and the data structure stores address information of the last-level page table entry corresponding to each of at least one GPA, thereby bypassing the multi-level page table translation in the EPT and improving address translation efficiency.

In this embodiment, since the last-level page tables of GPAs in the EPT page table may be of mixed granularity, the address information of the last-level page table entries stored in the first RMAP table may be address information of at least one type of page tables among PGD, PUD, PMD, and PTE.

In this embodiment, the timing of pre-constructing the first RMAP table corresponding to the EPT is not limited. For example, the first RMAP table may be created according to the correspondence between GPA and the last-level page table of the EPT after the EPT is created. For another example, when the last-level page table of the EPT is created, an add function of the first RMAP table may be called to establish the corresponding first RMAP table, so that the first RMAP table is established as the EPT is established.

In this embodiment, address translation trigger events are maintained. For ease of distinction and description, an address translation trigger event that triggers GPA to HPA

translation is referred to as a first address translation trigger event. The first address translation trigger event may be memory mapping management, a memory hot/cold management event, and a memory swap management event for GPA, etc.

Correspondingly, upon the first address translation trigger event, in response to the first address translation trigger event, a first GPA for which address translation is required may be determined, and address information of a last-level page table entry corresponding to the first GPA may be queried in the first RMAP table. If the address information of the last-level page table entry corresponding to the first GPA is found by query, an HPA corresponding to the first GPA is generated based on the address information of the last-level page table entry corresponding to the first GPA and the address offset of the first GPA. For example, by parsing the address information of the last-level page table entry corresponding to the first GPA, the starting address of the host physical address recorded in the last-level page table entry, the alignment value of the host physical address (HPA_align), may be obtained. The address offset of the first GPA refers to the offset relative to this starting address. Therefore, the HPA corresponding to the first GPA can be generated by adding the HPA_align and the address offset of the first GPA.

In an optional embodiment, a base structure may be created for the GPA address space corresponding to the EPT, and based on the base structure, the first RMAP table corresponding to the EPT may be pre-constructed. Compared with performing address translation level by level through the EPT, performing address translation based on the base structure does not need to go through multi-level page tables, and the address translation efficiency is higher.

Specifically, considering that the EPT has multiple different page table levels, a set page table level may be selected therefrom as a base page table level. The set page table level may be any page table level. For example, the set page table level may be the PUD page table level, the PMD page table level, the PTE page table level, or the PGD page table level, etc. The GPA address space corresponding to the EPT is divided into at least one address space, and a base structure corresponding to the base page table level (i.e., the set page table level) is created for the at least one address space, respectively. The GPA address space corresponding to the EPT may be divided according to a page granularity corresponding to the set page table level. For example, the set page table level is the PMD page table level, the granularity of the PMD page table is 2m, and the GPA address space corresponding to the EPT may be divided into at least one address space with a granularity of 2m, and a base structure is created for each 2m address space. For another example, the set page table level is the PUD page table level, the granularity of the PUD page table is 1G, and the GPA address space corresponding to the EPT may be

divided into at least one address space with a granularity of 1G, and a base structure is created for each 1G address space.

For any GPA in the EPT, a first structure corresponding to the any GPA is determined from base structures according to the address information of the any GPA and the address spaces corresponding to the base structures. The address space corresponding to each base structure may be understood as an address range, and the size of the address range corresponds to the page granularity corresponding to the set page table level. And any GPA is a piece of address information, and the address information can fall into the address space corresponding to a certain base structure. For ease of distinction and description, the base structure corresponding to the address space to which the address information of the any GPA belongs is referred to as a first structure.

According to the relationship between the level of the last-level page table corresponding to any GPA and the set page table level, the address information of the last-level page table entry corresponding to the any GPA is recorded through the first structure. The level of the last-level page table and the set page table level may be the same, or the level of the last-level page table is an upper level or a lower level of the set page table level. Additionally, the address information of the last-level page table entry corresponding to any GPA may be recorded by the first structure, or the address information of the last-level page table entry corresponding to any GPA may be recorded by an extended second structure of the first structure. For a detailed introduction to the second structure, reference may be made to the foregoing embodiments.

Optionally, the implementation manner of recording the address information of the last-level page table entry corresponding to the any GPA through the first structure according to the relationship between the level of the last-level page table corresponding to any GPA and the set page table level is not limited. The method provided by this optional embodiment is: if the level of the last-level page table is the same as the set page table level, the address information of the last-level page table entry is recorded using the first structure; if the level of the last-level page table is not the same as the set page table level, other page table levels different from the set page table level are represented through extended mapping. For example, for the first structure, a second structure is extended, the address information of the last-level page table entry is recorded in the second structure, and the address information of the second structure is recorded in the first structure.

The following provides exemplary descriptions.

Example C1: If the level of the last-level page table corresponding to any GPA is the same as the set page table level, it is considered that there is no need to extend the first structure,

and the address information of the last-level page table entry is directly recorded using the first structure. The address information of the last-level page table entry corresponding to the any GPA is added to an information field in the first structure, and flag information in the first structure is set to a first value. The first value indicates that the address information of the last-level page table entry is stored in the information field of the first structure. The base structure may be a rapid mapping header (rmap_head) structure, for example, struct rmap_head {unsigned long val}, where struct denotes a structure, unsigned long indicates that the data type is unsigned long, and val is the information field.

Example C2: If the level of the last-level page table corresponding to any GPA is not the same as the set page table level, it is considered that the first structure may be extended to generate a second structure adapted to the level of the last-level page table corresponding to the any GPA. The address information of the last-level page table entry corresponding to the any GPA may be added to an information field in the second structure, address information of the second structure may be added to an information field in the first structure, and flag information in the first structure may be set to a second value. The second value indicates that the address information of the second structure is stored in the information field of the first structure.

It should be noted that the flag information in the first structure is mainly used to identify the type of address information stored in the information field of the first structure. If the flag information is the first value, it indicates that the address information of the last-level page table entry is stored in the information field of the first structure. If the flag information is the second value, it indicates that the address information of the second structure is stored in the information field of the first structure. The values of the first value and the second value are different. For example, the first value and the second value may be represented by different bits in different information fields, and a bit with a value of 1 indicates that a corresponding value is valid. For another example, the lower three bits in the information field may be used as the flag information. the value range of the lower three bits is 0-7, and the first value and the second value may be any two different values among 0-7.

Further optionally, the implementation manner of creating the second structure adapted to the level of the last-level page table corresponding to any GPA is not limited, and an exemplary description is provided below: the address information of the last-level page table entry is written into the second structure, and the address information of the second structure is written into the base structure; in this case, all GPAs in the pud

corresponding base structures may share the same second structure, and there is no need to create a second structure for each base structure

Example D1: In the case that the level of the last-level page table corresponding to any GPA is the level above the set page table level, in an optional embodiment, multiple base structures may be created for the last-level page table, and for each base structure, a second structure adapted to the level of the last-level page table is generated, the address information of the last-level page table entry is recorded in each second structure, and the address information of each second structure is written into the corresponding base structure. In another optional embodiment, since the address information of the last-level page table entry is recorded in each second structure, multiple base structures may share the same second structure, and the address information of the shared second structure is recorded in each base structure.

For example, the set page table level is the PMD level, that is, the base structure corresponds to the PMD page table level, and the level of the last-level page table corresponding to any GPA is the level above the set page table level, that is, the PUD level. One PUD large page includes 512 PMD pages, each PMD page corresponds to one base structure, and one PUD large page corresponds to 512 base structures. Multiple base structures may share the same second structure, the second structure stores the address information of the PUD page, and the information field in each base structure stores the address information of the shared second structure.

Based on the above, in the case that the level of the last-level page table corresponding to any GPA is the level above the set page table level, it may be determined whether the second structure already exists; if the second structure does not exist, a new structure adapted to the level of the last-level page table corresponding to any GPA is created as the second structure, the address information of the second structure is added to the information field in the first structure corresponding to any GPA, and the flag information in the first structure is set to the second value.

Optionally, if the second structure already exists, the address information of the existing second structure is added to the information field in the first structure corresponding to any GPA, and the flag information in the first structure is set to the second value.

Here, in the case that the level of the last-level page table corresponding to any GPA is the level above the configured page table level, the second structure may be represented as rmap_cmpt {unsigned long val; atomic_t num; }. struct denotes a structure, unsigned long

indicates that the data type is unsigned long, val is the information field, and atomic_t num indicates the number of GPA-HPA fast mappings.

Example D2: In the case that the level of the last-level page table corresponding to any GPA is the level below the preset page table level, a new structure adapted to the level of the last-level page table corresponding to any GPA is created as the second structure.

For example, the preset page table level is the PMD page table level, and the level of the last-level page table corresponding to any GPA is the level below the preset page table level, that is, the PTE page table level. A page table at the preset page table level may be extended to multiple page tables at the next lower level. For example, a 2m page table at the PDM level is mapped to 512 page tables at the PTE level. Correspondingly, the first structure corresponding to any GPA may be extended to multiple (e.g., 512) second structures, and the multiple second structures are used to store the address information of the page tables at the next lower level (PTE page tables) mapped from the page table at the preset page table level (e.g., PMD page table). Further, a unified structure may be used to organize and manage the multiple second structures, the unified structure may organize and manage the multiple second structures in an array manner, and the address information stored in each second structure is treated as an element in the array. The unified structure may be a fast-mapping extension (rmap_extd) structure described below, and the address information stored in the second structure is an element of the array included in the rmap_extd structure. The unified structure is represented as rmap_extd{unsigned long val[VAL_EXTD_NR]; atomic_t num; }. Here, struct denotes a structure, unsigned long indicates that the data type is unsigned long, and atomic_t num indicates the number of GPA-HPA fast mappings. val[VAL_EXTD_NR] is an array, and the address information stored in the second structure is an element of the val[VAL_EXTD_NR] array.

Here, the offset of the granularity of the lower-level page table (e.g., 4k) within the page table at the preset page table level (e.g., 2m) may be determined, a new structure corresponding to the offset may be created, and the new structure may be used as the second structure corresponding to the any GPA. For example, if any GPA is located in the third 4k page within 2m, a new structure corresponding to offset 3 is created, and the new structure is used as the second structure corresponding to the any GPA. The address information of the last-level page table (e.g., PTE page table) corresponding to the any GPA may also be added to the information field of the second structure, and the flag information in the second structure may be set to the second value.

In an optional embodiment, in the case that the first RMAP table already exists, operations such as a query operation, an addition operation, a deletion operation, a modification operation, an extension operation, and a collapsing operation may be performed on any GPA. An exemplary description is provided below.

Example E1: A query operation performed on a first GPA is described as an example.

Here, the scenario of querying an HPA corresponding to the first GPA is not limited. For example, when reading or writing data for the first GPA, querying the HPA corresponding to the first GPA may be triggered, so that the HPA corresponding to the first GPA is quickly obtained based on the first RMAP table.

In the case of querying the HPA corresponding to the first GPA, the address information of the last-level page table entry corresponding to the first GPA is queried in the first RMAP table; if the address information of the last-level page table entry corresponding to the first GPA is found by query, the HPA corresponding to the first GPA is generated based on the address information of the last-level page table entry corresponding to the first GPA and the address offset of the first GPA. Optionally, if the address information of the last-level page table entry corresponding to the first GPA is not found, it is determined that no HPA exists for the first GPA, and a page fault may be triggered.

In an optional embodiment, based on the address information of the first GPA and address spaces corresponding to base structures included in the first RMAP table, the first structure corresponding to the first GPA is determined from the base structures, where the first structure is the base structure corresponding to the first GPA among the base structures included in the first RMAP table; the address information of the last-level page table entry corresponding to the first GPA is obtained through the first structure corresponding to the first GPA. The HPA corresponding to the first GPA is generated based on the address information of the last-level page table entry corresponding to the first GPA and the address offset of the first GPA.

Optionally, obtaining the address information of the last-level page table entry corresponding to the first GPA through the first structure corresponding to the first GPA includes: obtaining the flag information from the first structure; depending on the flag information, the manner of obtaining the address information of the last-level page table entry corresponding to the first GPA differs. If the flag information is the first value, it indicates that the information field of the first structure records the address information of the last-level page table entry, and the address information of the last-level page table entry corresponding to the first GPA may be directly obtained from the information field of the first structure; if the flag information is the second value, it indicates that the information field of the first structure

records the address information of a second structure, the second structure stores the address information of the last-level page table entry, the address information of the second structure may be obtained from the information field of the first structure, and the address information of the last-level page table entry corresponding to the first GPA may be obtained from the information field of the second structure based on the address information of the second structure.

Further optionally, the information field of the first structure stores the address information of the unified structure and the offset (offset, oft) of the second structure in the array of the unified structure, where the offset oft represents the offset of any GPA address at the last-level page granularity within the page table at the set page table level. For example, if the set page table level is the PMD level and the last-level page granularity is the PTE level, the offset oft refers to the offset at the 4k granularity within the 2m at the PMD level for any GPA address. If the offset is oft3, it indicates that the any GPA is located in the third 4k address space within the 2m. Correspondingly, obtaining the address information of the last-level page table entry corresponding to the first GPA from the information field of the second structure includes: based on the offset oft, selecting the oft-th element from the array of the unified structure (e.g., rmap_extd), where the value of the element is the address information of the last-level page table entry corresponding to the first GPA.

Example E2: An addition operation performed on a second GPA is described as an example.

Here, the scenario of adding the second GPA is not limited. For example, in a case that GPAs are insufficient, a GPA may be added, and in this case, an addition operation on the GPA address can be triggered. In the present embodiment, one or more GPAs may be added, and adding the second GPA is described below as an example.

In an optional embodiment, in a case of newly adding the second GPA in the EPT, the address information of the last-level page table entry corresponding to the second GPA is added in the first RMAP table.

Optionally, based on the address information of the second GPA and address spaces corresponding to base structures included in the first RMAP table, the first structure corresponding to the second GPA is determined from the base structures; if it is determined that the address information of the last-level page table entry corresponding to the second GPA is not recorded through the first structure, it indicates that the address information of the last-level page table entry corresponding to the second GPA is not recorded in the first RMAP table, and the address information of the last-level page table entry corresponding to the second GPA

is recorded through the first structure based on the relationship between the level of the last-level page table corresponding to the second GPA and the set page table level. For the detailed implementation, reference may be made to the foregoing description of "recording, through the first structure, the address information of the last-level page table entry corresponding to any GPA based on the relationship between the level of the last-level page table corresponding to any GPA and the set page table level," which is not repeated herein.

Example E3: A deletion operation performed on a third GPA is described as an example.

Here, the scenario of deleting a GPA is not limited. For example, in a case that virtual memory needs to be scaled down, some GPAs may be deleted, and the translation relationship between the GPAs and HPAs may be deleted. Deleting the third GPA is described below as an example.

In an optional embodiment, in a case of deleting the third GPA from the EPT, the address information of the last-level page table entry corresponding to the third GPA is deleted from the first RMAP table.

Optionally, based on the address information of the third GPA and address spaces corresponding to base structures included in the first RMAP table, the first structure corresponding to the third GPA is determined from the base structures; if the level of the last-level page table corresponding to the third GPA and the page table level indicated by the flag information of the first structure are adapted, the address information of the last-level page table entry corresponding to the third GPA is deleted through the first structure based on the relationship between the level of the last-level page table corresponding to the third GPA and the set page table level.

Here, the adaptation between the level of the last-level page table corresponding to the third GPA and the page table level indicated by the flag information of the first structure may be understood as: the level of the last-level page table is the same as the page table level to which the address information indicated by the flag information belongs. The flag information of the first structure is used not only to identify the type of the address information stored in the information field of the first structure, but also, in a case that the flag information indicates the address information of the last-level page table entry, to identify the page table level to which the address information recorded in the information field of the first structure belongs, that is, the page table level of the page table entry whose address information is recorded. For example, in the case that the flag information indicates the address information of the last-level page table entry, the flag information is a second value, and the second value may be X0 or X1, where X1 indicates that the address information is the address information of a page table

entry at the level below the set page table level (e.g., PTE level), and X0 indicates that the address information is the address information corresponding to a page table entry at the level above the set page table level (e.g., PUD level).

Example E4: A modification operation performed on a fourth GPA is described as an example.

Here, the scenario of modifying an HPA address corresponding to a GPA is not limited. For example, in a case of performing a memory swap operation, an original HPA page corresponding to the GPA is swapped to a disk, and the original HPA corresponding to the GPA can be modified to a new HPA. Modifying the fourth GPA is described below as an example.

In an optional embodiment, in a case of modifying the first address information of the last-level page table corresponding to the fourth GPA to the second address information in the EPT, the first address information of the last-level page table corresponding to the fourth GPA is modified to the second address information in the first RMAP table.

Optionally, based on the address information of the fourth GPA and address spaces corresponding to base structures included in the first RMAP table, the first structure corresponding to the fourth GPA is determined from the base structures; the first address information of the last-level page table entry corresponding to the fourth GPA is modified to the second address information through the first structure.

Further optionally, the flag information is obtained from the first structure; if the flag information is the first value, the first address information recorded in the information field of the first structure is modified to the second address information; if the flag information is the second value, the address information of the second structure is obtained from the information field of the first structure, and based on the address information of the second structure, the first address information recorded in the information field of the second structure is modified to the second address information.

Example E5: The last-level page table in the EPT is extended to obtain a lower-level page table. Extending the last-level page table is described below as an example.

In an optional embodiment, in a case of extending the last-level page table in the EPT to lower-level page tables, in the first RMAP table, the address information of the last-level page table entry corresponding to each GPA in the last-level page table is modified to the address information of a corresponding lower-level page table.

Optionally, in the EPT, the address information recorded in each page table entry of the last-level page table (i.e., the address information of the physical page corresponding to an

HPA) is cleared, and in the first RMAP table, the address information of the last-level page table entry corresponding to each GPA in the last-level page table is cleared; lower-level page tables corresponding to the last-level page table are created, the address information of the lower-level page tables is recorded in the page table entries of the last-level page table, and the address information of the physical pages corresponding to respective HPAs is recorded in the page table entries of the lower-level page tables; and in the first RMAP table, based on the relationship between the level of the lower-level page table and the set page table level, the address information of a lower-level page table entry corresponding to each GPA in the last-level page table is recorded through the first structure.

Example E6: Collapsing is performed on the last-level page table in the EPT to obtain an upper-level page table. The following description uses collapsing of the last-level page table as an example.

In an optional embodiment, in a case of collapsing last-level page tables with contiguous addresses in the EPT to an upper-level page table, in the first RMAP table, the address information of the last-level page table entries is cleared, and the address information of the upper-level page table entry corresponding to each GPA in the last-level page table is added.

Optionally, in the EPT, the address information recorded in each page table entry of the last-level page tables with contiguous addresses is cleared, and in the first RMAP table, the address information of the last-level page table entry corresponding to each GPA in the last-level page table is cleared; an upper-level page table corresponding to the last-level page tables is created in the EPT, and according to the page granularity corresponding to the upper-level page table entry, base addresses of the physical pages corresponding to respective HPAs are respectively recorded in the page table entries of the upper-level page table; and in the first RMAP table, based on the relationship between the level of the upper-level page table and the set page table level, the address information of the upper-level page table entry corresponding to each GPA in the last-level page table is recorded through the first structure.

In an optional embodiment, considering that the EPT page table cannot implement reverse translation from HPA to GPA, while the underlying layer mostly uses HPA as the management unit during memory management, so as to facilitate subsequent memory management, hot and cold management, and memory swapping. Therefore, a fast mapping relationship from HPA to GPA is needed.

In the present optional embodiment, a second RMAP table corresponding to the EPT is pre-constructed, and the second RMAP table stores address translation information from at least one HPA to the corresponding GPA; the address translation information of the

corresponding GPA refers to the base address of the GPA page to which the corresponding GPA belongs, and the size of the GPA page corresponds to the size of the HPA page. For example, the second RMAP table stores, through a third structure, the address information of HPAs partitioned at 2m granularity, and the GPA found directly by reverse lookup is also correspondingly partitioned at 2m granularity; as for other granularities such as 4k or 1G, a next-level extension structure (e.g., a fourth structure) is needed for storage. For example, if the size of the HPA page corresponding to the last-level page table entry maintained in the EPT page table is 4k, then the 4k-size HPA page corresponds to a 4k-size GPA page, and in the second RMAP table, the GPA page at 2m granularity is recorded through the third structure, and further, by extending the 2m-granularity GPA page, a plurality of 4k-granularity pages are obtained, and the plurality of 4k-granularity pages are recorded through the fourth structure; if the size of the HPA page corresponding to the last-level page table entry maintained in the EPT page table is 2m, then the 2m-size HPA page corresponds to a 2m-size GPA page, and in the second RMAP table, the 2m-granularity GPA page is recorded through the third structure; if the size of the HPA page corresponding to the last-level page table entry maintained in the EPT page table is 1G, then the 1G-size HPA page corresponds to a 1G-size GPA page, the 1G-size GPA page is split into a plurality of 2m-granularity GPA pages, and in the second RMAP table, each 2m-granularity GPA page is recorded through the third structure, and the address information of the GPA large page to which the 2m-granularity pages belong is recorded through the fourth structure, where the 512 2m-granularity pages under the 1G page share the address information of the same GPA large page. In response to a second address translation trigger event, a first HPA for which address translation is required is determined, and the address translation information from the first HPA to the corresponding GPA is queried in the second RMAP table; and based on the first HPA, the address translation information from the first HPA to the corresponding GPA, and the address offset of the first HPA, the GPA corresponding to the first HPA is obtained. For example, the GPA corresponding to the first HPA is obtained by adding the offset of the first HPA to the address translation information. The second address translation trigger event may be memory mapping management for GPA, memory hot and cold management events, memory swap management events, or the like.

Optionally, the implementation of pre-constructing the second RMAP table corresponding to the EPT includes: partitioning the HPA address space corresponding to the EPT into at least one address space, and creating a base structure for each of the at least one address space, where the base structure corresponds to a set page granularity, and the set page granularity may be 4k, 2m, 1G, or the like. For any HPA corresponding to a GPA in the EPT, based on the address

information of the HPA and address spaces corresponding to base structures, a third structure corresponding to the HPA is determined from the base structures, where the third structure is the base structure corresponding to the HPA in the second RMAP table, and the HPA may be regarded as being located in the address space corresponding to the third structure. For example, the third structure may be an rmap_head structure. Based on the relationship between the page granularity corresponding to any HPA and the set page granularity, the address translation information from any HPA to the corresponding GPA is recorded through the third structure. The page granularity corresponding to any HPA is known, and the page granularity corresponding to the HPA is the same as the set page granularity, or is not the same as the set page granularity.

Further optionally, if the page granularity corresponding to any HPA is the same as the set page granularity, the address translation information from the HPA to the corresponding GPA is recorded through the information field in the third structure, and the first flag information in the third structure is set to a third value, where the third value indicates that the information field of the third structure records the address translation information from the HPA to the corresponding GPA, and the third value may be 0, 1, or the like. If the page granularity corresponding to any HPA is not the same as the set page granularity, a fourth structure adapted to the page granularity corresponding to the HPA is created, where the granularity of the address space corresponding to the fourth structure is the same as the page granularity corresponding to the HPA. For example, the fourth structure corresponding to a 4k page granularity may be rmap_extd, and the fourth structure corresponding to a 1G page granularity may be rmap_cmpt. The address translation information from any HPA to the corresponding GPA is recorded through the information field in the fourth structure, and the address information of the fourth structure is added to the information field in the third structure, and the first flag information in the third structure is set to a fourth value. The fourth value indicates that the information field of the third structure records the address information of the fourth structure, and the fourth value may be 0 or 1. The third value and the fourth value may be represented using the low three bits of the information field of the third structure; for example, the third value and the fourth value may be any two different values from 0 to 7. Alternatively, the third value and the fourth value may be represented through two flag bits, where a flag bit of 1 indicates that the value is valid.

Further optionally, the implementation of creating the fourth structure adapted to the page granularity of any HPA includes: in a case that the page granularity of any HPA is an upper level of the set page granularity, determining whether the fourth structure already exists; if not,

creating a new structure adapted to the page granularity of the HPA, as the fourth structure; in a case that the page granularity of the HPA is a lower level of the set page granularity set page table level, creating a new structure adapted to the level of the last-level page table corresponding to the GPA, as the fourth structure. For the detailed description of this embodiment, reference may be made to the implementation of creating the second structure adapted to the level of the last-level page table corresponding to the GPA, which is not repeated here. For example, the page granularity in order of level from top to bottom may be: 512G, 1G, 2m, and 4k. If the page granularity of any HPA is 1G and the set page granularity is 2m, then the page granularity of the HPA is regarded as being the upper level of the set page granularity. If the page granularity of any HPA is 4k and the set page granularity is 2m, then the page granularity of the HPA is regarded as being the lower level of the set page granularity.

In an optional embodiment, the mapping between HPA and GPA may be a one-to-one correspondence, or is a one-to-many relationship; for example, one HPA corresponds to a plurality of GPAs. Correspondingly, the implementation of recording, through the information field in the third structure, the address translation information from any HPA to the corresponding GPA includes: in a case that the HPA corresponds to a plurality of GPAs, creating a fifth structure, where the fifth structure is an extension of the third structure, and is a structure that records the address translation information from the HPA to the plurality of GPAs in a case that the page granularity of the HPA is the same as the set page granularity. The address information of the fifth structure is added to the information field in the third structure, and the address translation information from the HPA to the plurality of GPAs is respectively added to the fifth structure, and a flag bit indicating a one-to-many relationship is added in the third structure. In a case that any HPA corresponds to one GPA, the address translation information from the HPA to the corresponding GPA is directly added to the information field in the third structure, without the need to set a flag bit indicating a one-to-many relationship, and by default the HPA and GPA have a one-to-one relationship.

Correspondingly, recording, through the information field in the fourth structure, the address translation information from any HPA to the corresponding GPA includes: in a case that the HPA corresponds to a plurality of GPAs, creating a sixth structure, where the sixth structure is an extension of the fourth structure, and is a structure that records the address translation information from the HPA to the plurality of GPAs in a case that the page granularity of the HPA is not the same as the set page granularity. The address information of the sixth structure is added to the information field in the fourth structure, and the address

translation information from the HPA to the plurality of GPAs is respectively added to the sixth structure, and a flag bit indicating a one-to-many relationship is added in the fourth structure; in the case that the HPA corresponds to one GPA, the address translation information from the HPA to the corresponding GPA is directly added to the information field in the fourth structure.

It should be noted that the fifth structure and the sixth structure may be implemented as an rmap_desc structure, i.e., rmap_desc {unsigned long val[VAL_DESC_NR]; struct rmap_desc *more;}. val[VAL_DESC_NR] represents being used to store the address translation information from any HPA to a plurality of corresponding GPAs, each element in the array represents the address translation information from the HPA to one corresponding GPA, and the *more pointer may point to the next rmap_desc, and the next rmap_desc structure also includes a val[VAL_DESC_NR] array, for storing address translation information of more GPAs.

The following provides illustrative examples of addition, query, deletion, modification, extension, collapsing, and extension operations based on the first RMAP table and the second RMAP table as shown below, but is not limited thereto.

An EPT page table is a four-level page table structure, including four types of page tables: PGD, PUD, PMD, and PTE, and the last-level page table of the EPT may be at least one of PGD, PUD, PMD, and PTE. Correspondingly, the last-level page table entry may be pgd_t, pud_t, pmd_t, or pte_t. The base structure is identified by rmap_head. The unified structure at the PTE level is rmap_extd, and the address information of the last-level page table entry stored in the second structure is an element in the val[VAL_EXTD_NR] array of the unified structure. The second structure at the PUD level is rmap_cmpt.

I. Querying the HPA corresponding to a first GPA through the first RMAP table.

1. First, based on the address information of the first GPA and the address spaces corresponding to the base structures included in the first RMAP table, the first structure (rmap_head) corresponding to the first GPA is determined from the base structures. The first RMAP table may be regarded as a large array, the elements in the large array are base structures, and the address space corresponding to each base structure is 2m; therefore, based on the address information of the first GPA, the base structure corresponding to the first GPA is selected from the large array at a granularity of 2m, and the selected base structure is used as the first structure.

2. Based on the flag information of the information field (val) in the first structure rmap_head, different query methods are selected.

1) If the flag information is the second value corresponding to rmap_cmpt, it indicates that the page granularity corresponding to the last-level page table entry of the second GPA is 1G.

a) Parse out the rmap_cmpt structure from the information field (val) of rmap_head, i.e., obtaining the address information of rmap_cmpt;

b) Parse the information field in rmap_cmpt (the second structure) into a new rmap_head; here, rmap_head is a universal structure, and the new rmap_head may be understood as the value contained in the information field of rmap_cmpt;

c) Determine whether the val value in the new rmap_head is the address information of a page table entry. For example, based on the flag information in the val value, it may be determined whether the val value is the address information of a page table entry. If yes, then based on the last-level address information of the first GPA recorded in the new rmap_head, the base address of the physical page recorded in the address information is read, and the address offset of the first GPA is added to the base address to generate the HPA corresponding to the first GPA, where the address offset is the offset within the 1G memory page. If no, it indicates that the val value in the new rmap_head has a next-level extension structure, the val value is the address information of the next-level extension structure, and the next-level extension structure may be a structure corresponding to a one-to-many relationship.

2) If the flag information is the second value corresponding to rmap_extd, it indicates that the page granularity corresponding to the last-level page table entry of the second GPA is 4k.

a) Parse out the rmap_extd structure from the information field of rmap_head, i.e., obtain the address information of rmap_extd;

b) Based on the offset oft of the second GPA at 4k granularity within 2m, select the oft-th val element from the val[VAL_EXTD_NR] array in rmap_extd, which is the address information of the second structure;

c) Parse the oft-th val element as a new rmap_head;

d) Determine whether the val value in the new rmap_head is the address information of a page table entry. If yes, then based on the last-level address information of the first GPA recorded in the new rmap_head, the base address of the physical page recorded in the address information is read, and the address offset of the first GPA is added to the base address to generate the HPA corresponding to the first GPA, where the address offset is the offset within the 4k memory page. If no, it indicates that the val value in the new rmap_head has a next-level extension structure, the val value is the address information of the next-level extension

structure, and the next-level extension structure may be a structure corresponding to a one-to-many relationship.

3) If the flag information is the first value corresponding to rmap_head or no flag information is set, then the page granularity corresponding to the last-level page table entry is 2m, and there is no need to parse out a lower-layer structure.

a) Determine whether the val value in the new rmap_head is the address information of a page table entry. If yes, then based on the last-level address information of the first GPA recorded in the new rmap_head, the base address of the physical page recorded in the address information is read, and the address offset of the first GPA is added to the base address to generate the HPA corresponding to the first GPA, where the address offset is the offset within the 2m memory page. If no, it indicates that the val value in the new rmap_head has a next-level extension structure, the val value is the address information of the next-level extension structure, and the next-level extension structure may be a structure corresponding to a one-to-many relationship.

The present embodiment further includes querying the GPA corresponding to the first HPA through the second RMAP table, and this process is similar to the implementation of querying the HPA corresponding to the first GPA through the first RMAP table, and is not repeated here.

In addition, in the case of storing a one-to-many relationship between HPA and GPA, when querying the GPA corresponding to the first HPA in the second RMAP table, a one-to-many mapping lookup operation may be further included. The following focuses on the parts of the one-to-many that differ from the one-to-one, and other details are not repeated.

For example, find the GPA corresponding to a second HPA. If the flag information in the third structure is the fourth value, the fourth structure is rmap_cmpt, the flag information in the fourth structure rmap_cmpt indicates that a one-to-many mapping relationship exists, and the fourth structure stores the address information of the fifth structure rmap_desc, it is required to iterate through each val recorded in rmap_desc, determine whether the val is the address information of a page table entry, and if yes, directly return the val value. If no, the val value is the address information of a next-level extension structure, and the val may be obtained from the next-level extension structure.

When querying rmap_desc, all elements in the val[VAL_DESC_NR] array of rmap_desc are obtained, and the corresponding GPA is calculated according to the address translation information of the GPA recorded in each element and the offset of the first HPA. Of course, a certain element in the array may be returned according to a set filtering condition, and the

address translation information of the GPA recorded in the element plus the offset of the first HPA yields the GPA corresponding to the first HPA. If the length of val[VAL_DESC_NR] is 5, it indicates that the array stores 5 elements, and the array is linked to a subsequent array via a *more pointer. To obtain the 13th element of the array, the address information of a second rmap_desc is obtained via the more pointer in a first rmap_desc, and based on the address information of the second rmap_desc, the address information of a third rmap_desc pointed to by the more pointer in the second rmap_desc is determined. Based on the address information of the third rmap_desc, the 3rd element in val[VAL_DESC_NR] of the third rmap_desc is obtained. The 3rd element is the address translation information of the GPA corresponding to the first HPA, and adding the address offset of the first HPA can yield the GPA corresponding to the first HPA. If the length of val[VAL_DESC_NR] is 5 and the 3rd element of the array is to be obtained, the 3rd element may be directly obtained from val[VAL_DESC_NR] of the first rmap_desc, i.e., the address translation information of the GPA corresponding to the first HPA, and adding the address offset of the first HPA can yield the GPA corresponding to the first HPA.

II. Adding an HPA corresponding to a second GPA to the first RMAP table.

1. First, a first structure (rmap_head) corresponding to the second GPA is determined from the base structures according to the address information of the second GPA and the address spaces corresponding to the base structures included in the first RMAP table. The first RMAP table may be regarded as a large array, elements in the large array are base structures, and the address space corresponding to each base structure is 2m. Therefore, based on the address information of the second GPA, a base structure corresponding to the second GPA is selected from the large array with a granularity of 2m, and the selected base structure is used as the first structure.

2. The page table level (level) of the last-level page table of the second GPA is determined; flag information of the first structure is obtained, and it is determined whether the flag information indicates that the address information stored in the information field of the first structure corresponds to a structure of the page table level (level). If yes, it is determined that the structure of the page table level (level) already exists. If no, it is determined that the structure corresponding to the page table level (level) does not exist, and the structure corresponding to the page table level (level) is created. For example, if the page table level level is the PUD level but the flag information of the first structure is not set to the value corresponding to rmap_cmpt, an rmap_cmpt structure is allocated, and the rmap_cmpt structure is written into the information field of the first structure.

3. It is determined whether the address information of the last-level page table entry corresponding to the second GPA exists in the structure corresponding to the page table level level. If it exists, the address information is directly returned;

4. If it does not exist, an HPA is added for the second GPA. At this time, different addition methods may be selected according to the flag information of the information field (val) in the first structure rmap_head.

1) If the flag information is the second value corresponding to rmap_cmpt, it indicates that the page granularity corresponding to the last-level page table entry of the second GPA is 1G.

It should be noted that, when a 1G granularity mapping is established for a GPA for the first time, an rmap_cmpt structure may be allocated, the address information of the HPA aligned to the 1G GPA is recorded in the rmap_cmpt structure, and a count is incremented. The address information of 512 rmap_heads contained in the 1G memory page is associated with the rmap_cmpt, i.e., the address information of the rmap_cmpt is recorded in the information field of each rmap_head. When subsequently establishing mappings for other GPAs in the 1G memory page, the mapping count in the rmap_cmpt structure is incremented.

2) If the flag information is the second value corresponding to rmap_extd, it indicates that the page granularity corresponding to the last-level page table entry of the second GPA is 4k.

a) Parse out the rmap_extd structure from the information field of rmap_head, i.e., obtain the address information of rmap_extd;

b) According to the offset oft of the second GPA at 4k granularity within 2m, select the oft-th val element in val[VAL_EXTD_NR] from rmap_extd, i.e., the address information of the second structure;

c) Parse the oft-th val element as a new rmap_head;

d) Add a map value (i.e., the address information of the pte_t page table entry) to the new rmap_head; i.e., add the address information of the pte_t page table entry to the information field of the second structure.

e) Increment the mapping count num in the rmap_extd structure.

3) If the flag information is the first value corresponding to rmap_head or flag information is not set, the page granularity corresponding to the last-level page table entry is 2m, and parsing out a lower-layer structure is not required.

a) Directly add a map value, i.e., the address information of the last-level page table entry (pmd_t), to the information field of rmap_head.

The present embodiment further includes adding a GPA corresponding to a second HPA to the second RMAP table, and the implementation is similar to that of adding the HPA corresponding to the second GPA to the first RMAP table, which is not repeated here.

In addition, in the case that a one-to-many mapping of HPA to GPA is stored, when the GPA corresponding to the second HPA is added to the second RMAP table, the addition operation of the one-to-many mapping may be further included. The following focuses on the differences of the "one-to-many" compared to the "one-to-one," and other details are not repeated.

An example is described in which the address translation information from the second HPA to the corresponding GPA is recorded in a third structure, where the implementation of recording the address translation information of the corresponding GPA in a fourth structure is the same or similar and is not repeated here.

If it is the first time that the address translation information of the corresponding GPA is added to the third structure, the address translation information of the GPA corresponding to the second HPA is directly recorded in the information field of the third structure.

In the case that the second HPA corresponds to multiple GPAs, it is determined whether flag information of the third structure includes mark information indicating "one-to-many." If no mark information exists, it indicates that the third structure is extended for the first time, and a fifth structure (rmap_desc) structure may be allocated. The rmap_desc structure includes: val[VAL_DESC_NR], and the array may be initialized to a special value, as long as the special value is not used as address information. For example, ffffffff, where f represents the maximum value in hexadecimal, i.e., 15, and ffffffff is the last address in a 4GB memory space. The address translation information of the GPA recorded in the third structure is recorded at the position of the first element in val[VAL_DESC_NR], and the address translation information of the GPA to be added for the second time is recorded at the position of the second element in val[VAL_DESC_NR]. The address information of the rmap_desc structure is recorded in the information field of the third structure, and a flag bit indicating a one-to-many relationship is added to the third structure.

If the mark information exists in the third structure, it indicates that at least one extension has been performed, and a free unrecorded position in rmap_desc is first searched for. If such a position exists, the position is selected to record a map value, i.e., the address translation information of the GPA corresponding to the second HPA. If no recording position is found in rmap_desc, a new rmap_desc is allocated and linked after the more pointer of the last rmap_desc, and then a free position is selected from the array of the newly allocated rmap_desc

to record a map value, i.e., the address translation information of the GPA corresponding to the second HPA.

III. Deleting an HPA corresponding to a third GPA from the first RMAP table.

1. First, a first structure (rmap_head) corresponding to the second GPA is determined from the base structures according to the address information of the second GPA and the address spaces corresponding to the base structures included in the first RMAP table. The first RMAP table is regarded as a large array, elements in the large array are base structures, and the address space corresponding to each base structure is 2m. Therefore, based on the address information of the second GPA, a base structure corresponding to the second GPA is selected from the large array with a granularity of 2m, and the selected base structure is used as the first structure.

2. The page table level (level) of the last-level page table of the second GPA is determined; flag information of the first structure is obtained, and it is determined whether the flag information indicates that the address information stored in the information field of the first structure corresponds to a structure of the page table level (level). If yes, it is determined that the structure of the page table level (level) already exists, and step 3 is performed. If no, it is determined that the structure corresponding to the page table level (level) does not exist, nor does the HPA corresponding to the third GPA exist, and an error is returned. For example, if the page table level level is the PUD level but the flag information of the first structure is not set to the value corresponding to rmap_cmpt, an error is returned.

3. Different deletion methods are selected according to the flag information of the information field (val) in the first structure rmap_head.

1) If the flag information is the second value corresponding to rmap_cmpt, it indicates that the page granularity corresponding to the last-level page table entry of the second GPA is 1G.

a) Parse out the rmap_cmpt structure from the information field (val) of rmap_head, i.e., obtain the address information of rmap_cmpt;

b) Parse the information field in the rmap_cmpt structure is parsed as a new rmap_head;

c) Delete a map value, i.e., the address information of pud_t, from the new rmap_head;

d) Decrement the mapping count num in the rmap_cmpt structure;

e) If the mapping count num becomes 0, release the rmap_cmpt structure.

It should be noted that, when the rmap_cmpt structure is released, the rmap_cmpt structure corresponds to a 1G memory page, and val values in rmap_head respectively corresponding to

512 2m memory pages under the 1G memory page store address information of the rmap_cmpt structure; if the rmap_cmpt structure is released, the val values in the respective rmap_head are cleared.

2) If the flag information is the second value corresponding to rmap_extd, it indicates that the page granularity corresponding to the last-level page table entry corresponding to the second GPA is 4k.

a) Parse out the rmap_extd structure from the information field of head_rmap, i.e., obtain the address information of rmap_extd;

b) According to an offset oft of the second GPA at 4k granularity within 2m, select the oft-th val element in rmap_extd from the val[VAL_EXTD_NR] array, that is, the address information of the second structure;

c) Parse the oft-th val element as a new rmap_head;

d) Delete the map value, i.e., the address information of pte_t, from the new rmap_head, and decrement the mapping count by 1; when the count is 0, release the rmap_extd structure, and simultaneously clear the mapping in the upper-level rmap_head, that is, clear the address information of rmap_extd recorded in val of the rmap_head.

3) If the flag information is the first value corresponding to rmap_head or no flag information is set, the page granularity corresponding to the last-level page table entry is 2m, and no lower-layer structure needs to be parsed out.

a) Delete the map value recorded in the information field of rmap_head, where the map value is the address information of pmd_t.

The present embodiment further includes deleting the GPA corresponding to the third HPA from the second RMAP table; this process is similar to the implementation of deleting the HPA corresponding to the third GPA from the first RMAP table, and details are not repeated herein.

In addition, in a scenario of storing a one-to-many mapping of HPA to GPA, when the GPA corresponding to the third HPA in the second RMAP table is deleted, an addition operation for the one-to-many mapping may be further included. The following focuses on the parts of the "one-to-many" that differ from the "one-to-one," and other details are not repeated.

An example of deleting, in the third structure, the address translation information from the third HPA to the corresponding GPA is used for description, where an implementation of

deleting, in the fourth structure, the corresponding address translation information of the GPA is the same or similar and details are not repeated herein.

For the case where the third HPA corresponds to a plurality of GPAs, the address information of the rmap_desc structure is obtained from the information field of the third structure, and each element recorded in the rmap_desc structure is iterated to determine whether it is the address translation information from the third HPA to the corresponding GPA that needs to be deleted;

If the address translation information from the third HPA to the corresponding GPA that needs to be deleted (hereinafter referred to as the map value) is found and the rmap_desc structure exists, the last position j in rmap_desc that has a map value is first found;

The value at the last position j is replaced to the position i to be deleted, and the value at position j is deleted; if j is not 0 at this time, the process returns; if j is 0, it indicates that the entire rmap_desc is empty, and the rmap_desc may be deleted; if a previous rmap_desc exists, the rmap_desc information linked under the more of the current rmap_desc is updated to the more linked list of the previous rmap_desc; if no previous rmap_desc exists, the value of val in rmap_head is associated with the rmap_desc pointed to by the more of the current rmap_desc.

IV. In the first RMAP table, the first address information of the last-level page table corresponding to the fourth GPA is modified to the second address information. In the second RMAP table, the first address translation information of the GPA corresponding to the fourth HPA is modified to the second address translation information. An example where the last-level page table is at the PMD level is used for description.

1. Clear the original PMD-level HPA-GPA mapping, that is, clear the first address translation information of the GPA corresponding to the fourth HPA recorded in the second RMAP table;

2. Establish a new PMD-level page table mapping in the EPT, and record the physical page corresponding to the new HPA in the PMD-level page table entry;

3. Simultaneously establish a new PMD-level HPA-GPA mapping, that is, record the second address translation information of the GPA corresponding to the fourth HPA in the second RMAP table;

4. Flush the EPT page table so that the modifications take effect.

During subsequent host operation, operations such as page table extension, collapsing, and update may be performed; at this time, when the EPT is updated, the first RMAP table and the second RMAP table may be synchronously updated.

V. In a case of extending the last-level page table in the EPT to lower-level page tables, the address information of the last-level page table entry corresponding to each GPA in the last-level page table is modified in the first RMAP table to the address information of the corresponding lower-level page table.

The following uses extending a PMD-level EPT page table to the PTE level as an example for description.

1. Determine whether EPT page table extension is required, that is, whether the EPT page table is already at the PTE level; if yes, return directly; otherwise, continue;

2. Allocate a page EPTP for recording lower-level PTE-level page tables. EPTP stands for EPT page table pointer (extended-page-table pointer), and EPTP is a field that stores the physical address of the EPT page table.

3. Clear the previous PMD-level page table mapping, including clearing the address information of the physical page recorded in the original PMD page table;

4. Clear the original GPA_HPA mapping (also referred to as GPA_EPT mapping) in the first RMAP table, and the HPA_GPA mapping;

5. For each PTE-level page table at the lower level, establish the corresponding mapping relationship (that is, write the base address of the physical page in each PTE-level page table entry), and record the address information of the PTE-level page table entry in the allocated page EPTP;

6. During the establishment of each PTE-level page entry, a new GPA_HPA mapping and a new HPA_GPA mapping are synchronously established, to obtain the first RMAP table and the extended second RMAP table after extension, respectively.

7. Finally, associate the newly allocated page EPTP with the original PMD-level page table entry;

8. Flush the EPT page table so that the updates take effect.

VI. In a case of collapsing last-level page tables with contiguous addresses in the EPT into an upper-level page table, the address information of the last-level page table entry corresponding to each GPA in the last-level page table is modified in the first RMAP table to the address information of an upper-level page table entry.

The following uses collapsing PTE page tables into a PMD page table as an example for description.

1. Determine whether EPT page table collapsing is required, i.e., whether the EPT page table is already at the PMD level; if yes, return directly; otherwise, proceed;

2. Determine whether the lower-level PTE page tables can be collapsed, i.e., whether all mapping entries in the lower-level PTE page tables are empty, or whether a contiguous HPA range is recorded and the first HPA addresses are PMD-level aligned;

3. For each lower-level PTE-level page table, clear the corresponding mapping relationships, such as clearing the base addresses of physical pages recorded in the page table entries of the PTE page table;

4. During the process of clearing each PTE-level page entry, the old GPA_EPT mapping and HPA_GPA mapping are also synchronously cleared, i.e., the corresponding mapping entries in the first RMAP table and the second RMAP table are cleared;

5. Establish a new PMD-level page table mapping, record the HPA mapping information in the EPT page table entry, thereby disassociating the original lower-level PTE page tables;

6. At the same time, establish new PMD-level RMAP table information, including the GPA_EPT mapping and the HPA_GPA mapping;

7. Flush the EPT page table to make the updates take effect;

8. Release the pages that originally recorded the PTE-level page tables.

VII. It may further include: clearing page table entries at a set page table level; for example, when a page is swapped out, updating the HPA_GPA mapping;

The following description uses clearing PMD-level page table entries as an example.

1. Clear the original PMD-level HPA_GPA mapping;

2. Clear the old PMD-level page table mapping recorded in the PMD-level page table entry;

3. Flush the EPT page table to make the updates take effect.

In this embodiment, in addition to the address translation method described above, a fast mapping table construction method is also provided. As shown in FIG. 1b, the method includes:

R21, divide a GPA address space corresponding to an EPT into at least one address space, and create a base structure for each of the at least one address space respectively, where the base structure corresponds to a set page table level, and the EPT includes multi-level page tables corresponding to at least one GPA respectively;

R22, for any GPA in the EPT, determine a first structure corresponding to the any GPA from among base structures according to address information of the any GPA and address spaces corresponding to the base structures;

R23, record, according to a relationship between a level of a last-level page table corresponding to the any GPA and the set page table level, through the first structure, address information of the last-level page table entry corresponding to the any GPA, to construct a first

RMAP table corresponding to the EPT, where the first RMAP table stores address information of a last-level page table entry corresponding to each of the at least one GPA.

Regarding the detailed implementations and beneficial effects of the steps in the methods shown in FIG. 1a and FIG. 1b provided by the embodiments of the present disclosure, detailed descriptions have been provided in the foregoing embodiments and will not be elaborated herein.

In addition to the above methods, the embodiments of the present disclosure also include an introduction to the virtualization layer, as follows:

In an embodiment of the present disclosure, as shown in FIG. 2a, a hos includes hardware resources, a host operating system (host operating system, host OS) runs on the hardware resources, and various applications run on the host operating system; relative to the hardware resources, the host operating system and the applications belong to software resources of the host. The host operating system is the most basic system software in the host, and is responsible for controlling and managing the entire hardware resources and software resources of the host, and is capable of reasonably scheduling the work and resource allocation of the host, and can provide convenient interfaces and environments to users and other software or devices. The host operating system can provide various system-level functions, such as file management, memory management, and various Input/Output (IO) device management, etc. Based on these system functions, the host operating system can provide various resources required by applications during operation to support the running of the applications.

Among these system functions, at least some system functions need to be configured when installing the host operating system, such as whether to enable them, which mode to enable, etc. System functions that are not enabled cannot be used during the operation of the host operating system. Due to the unavailability of such system functions, applications that depend on those system functions will operate abnormally or fail to run because they cannot obtain the resources provided by those system functions. However, with the operation of the host operating system and changes in application requirements, it may be required to use these unenabled system functions, or to upgrade already-enabled system functions, i.e., the host operating system faces a capability expansion problem. For a traditional host, either the host operating system is reinstalled, or some system functions are forcibly enabled online. The approach of reinstalling the host operating system requires interrupting the running of all applications, resulting in low efficiency and high cost; the approach of forcibly enabling the corresponding system functions online will seriously affect the stability of the system.

In an embodiment of the present disclosure, in order to solve the capability expansion problem faced by the host operating system without reinstalling the host operating system and without affecting the running performance of the host operating system, a new physical machine architecture is provided, which provides two running modes for the host operating system in a running state, namely a root mode and a non-root mode. A virtualization layer is inserted between the hardware resources and the host operating system, at least a virtual resource providing function is implemented in the virtualization layer, and the virtualization layer switches the running host operating system from the root mode to the non-root mode. In the non-root mode, at least one virtual resource is provided and at least an application that cannot run due to insufficient resources in the root mode of the host operating system is run on the virtual resource. Without reinstalling the operating system, the virtual resource providing function can be flexibly implemented for the host operating system in a running state, solving the problem that applications cannot run due to insufficient resources caused by unenabled or missing system functions, supporting the running of more applications, and facilitating the expansion of capabilities of the host operating system.

For example, assuming that the host operating system supports function A, function B, function C, and function D, etc., and when installing the host operating system, function A, function B, and function C are enabled, function D is not enabled, and function E is missing (i.e., the host operating system does not support function E). Using the physical machine provided by the embodiment of the present disclosure, a virtualization layer is inserted between the hardware resources and the host operating system, and at least a virtual resource providing function is implemented in the virtualization layer. For example, function D that is not enabled and function E that is missing in the host operating system are handled by the virtualization layer, which switches the host operating system from the root mode to the non-root mode, and provides virtual resources corresponding to function D and function E in the non-root mode, and applications that originally could not run due to function D not being enabled and function E being missing run on the virtual resources. Without reinstalling the host operating system, the virtual resource providing function can be flexibly implemented for the host operating system in a running state, solving the problem that corresponding applications cannot run due to function D not being enabled and function E being missing, and facilitating the expansion of capabilities of the host operating system. Further, in order to compensate for the deficiencies of the host operating system, virtual resources corresponding to function D and function E may be preferentially provided in the non-root mode, solving the running problem of applications that depend on function D and function E; of course, for compatibility of the host operating

system and to avoid frequent mode switching, virtual resources corresponding to function A, function B, and function C, etc., may also be provided in the non-root mode, so as to run corresponding applications that depend on function A, function B, and function C on those resources.

It is noted herein that in the embodiments of the present disclosure, the focus is on the switching process of the host operating system between the two modes, and no limitation is imposed on which virtual resources corresponding to which system functions are provided in the non-root mode and how the virtual resources corresponding to those system functions are provided. The physical machine architecture provided by the embodiments of the present disclosure and the process of switching the host operating system from the root mode to the non-root mode will be described in detail below with reference to the accompanying drawings.

FIG. 2b is a schematic structural diagram of a physical machine provided by an exemplary embodiment of the present disclosure. As shown in FIG. 2b, the physical machine includes: hardware resources 10 and a host operating system 20 running on the hardware resources, and a virtualization layer 30 is implemented between the hardware resources 10 and the host operating system 20. Further, an application layer 40 is also included above the host operating system 20. The application layer 40 includes various applications.

Optionally, the hardware resources 10 include: at least one physical computing resource object 101 and a physical storage medium 102, where the physical storage medium 102 is responsible for providing a physical address space. In addition, the hardware resources 10 on the physical machine may further include: IO devices, communication components, a display, power supply components, audio components, and other components, which are not illustrated.

The physical computing resource object 101 may be a central processing unit (Central Processing Unit, CPU), a graphics processing unit (Graphics Processing Unit, GPU), a data processing unit (Data Processing Unit, DPU), a tensor processing unit (Tensor Processing Unit, TPU), a cloud infrastructure processing unit (Cloud Infrastructure Processing Unit, CIPU), an application specific integrated circuit (Application Specific Integrated Circuit, ASIC), or other physical resource objects having computing capabilities. The host operating system 20 may run on at least one physical computing resource object 101.

The physical storage medium 102 includes permanent and non-permanent, removable and non-removable media that may be implemented by any method or technology for information storage. The information may be computer-readable instructions, data structures, program modules, or other data. The physical storage medium 102 includes, but is not limited to: phase-change random access memory (Phase-change Random Access Memory, PRAM), static

random access memory (Static Random Access Memory, SRAM), dynamic random access memory (Dynamic Random Access Memory, DRAM), other types of random access memory (RAM), read-only memory (ROM), electrically erasable programmable read-only memory (EEPROM), flash memory or other memory technologies, compact disc read-only memory (CD-ROM), digital video disc (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other non-transmission medium that can be used to store information accessible by a computing device.

The physical memory is an actual memory chip or module existing in the physical machine, used for storing data and program code. The physical memory may include non-permanent memory in the physical storage medium 102, random access memory (Random Access Memory, RAM), and/or non-volatile memory, etc., such as read-only memory (ROM) or flash memory (flash RAM).

The physical address space is an address range corresponding to the physical memory. The physical address space is typically related to the hardware architecture and the host operating system, and describes the physical location and size of actual memory modules. In a virtual memory system, the physical address space also involves mechanisms such as memory page tables, where the memory page table mechanism is used to implement the mapping from guest physical addresses to host physical addresses.

In this embodiment, the host operating system 20 corresponds to a root mode and a non-root mode. The root (root) mode is a working mode in which the host operating system 20 has direct access privilege to the hardware resources 10, and the non-root mode (non-root) is a working mode in which the virtualization layer 30 virtualizes the hardware resources 10 and schedules and accesses the virtualized resources on behalf of the host operating system 20. In the non-root mode, access of the virtualization layer 30 to the hardware resources 10 is restricted. Correspondingly, the host operating system 20 may also be referred to as a guest operating system in the non-root mode.

In this embodiment, a lightweight, powerful, and efficient virtualization layer 30 is inserted between the hardware resources 10 and the host operating system 20. The virtualization layer 30 may be pre-developed and inserted between the hardware resources 10 and the host operating system 20. The manner of inserting the virtualization layer 30 is not limited. For example, after the host operating system 20 is installed on the physical machine, the virtualization layer 30 may be inserted while the host operating system 20 is in a normal running state.

The virtualization layer 30 has virtual resource provision capabilities and is at least capable of providing virtual resources corresponding to resources that the host operating system 20 is unable to provide due to system functions that are not enabled or are missing in root mode. That is to say, resources that the host operating system 20 is unable to provide in root mode due to system functions that are not enabled or are missing can be provided by the virtualization layer 30 as corresponding virtual resources, so as to facilitate running of corresponding applications. Specifically, the virtualization layer 30 can switch the running host operating system 20 from root mode to non-root mode, provide at least one type of virtual resource in the non-root mode, and run at least one type of application on the at least one type of virtual resource. Without the need to reinstall the operating system, the virtual resource provision function can be flexibly implemented for the running host operating system, solving the problem that the host operating system 20 is unable to provide corresponding resources to run corresponding applications due to system functions that are not enabled or are missing in root mode, and the operation of more applications can be supported, facilitating expansion of the capabilities of the host operating system. Furthermore, out of compatibility considerations for the host operating system and to reduce frequent mode switching of the host operating system, in the non-root mode, virtual resources are provided not only for system functions that are not enabled or are missing in the host operating system 20 in root mode, but also for system functions that are already enabled in the host operating system 20 in root mode, thereby achieving support for more functions of the host operating system 20.

By way of example, at least one target function implemented in the virtualization layer 30 includes, but is not limited to: a memory management function, a file management function, a virtualization function, a scheduling optimization function, and the like. The memory management function, the file management function, the scheduling optimization function, and the like herein may be newly extended functions that the host operating system originally does not support, or may be functions that the host operating system originally supports but that are not enabled during the installation of the host operating system. Regardless of whether they are newly extended or not enabled functions, the memory management function specifically includes, but is not limited to: memory page fault management, memory swap management, memory mapping management, and the like; the file management function specifically includes, but is not limited to: directory management, permission management, file backup, creation, query, deletion, modification, and the like; the virtualization function includes, but is not limited to: virtualization of various elastic resources (such as CPU, GPU, memory, network, etc.); and the scheduling optimization function includes, but is not limited to: scheduling of

physical computing resource objects and scheduling of various virtualized resources, and the like.

Furthermore, FIG. 2b is a schematic diagram of an overall architecture of a physical machine. Compared with FIG. 2b, FIG. 2c and FIG. 2d show internal architectures of the physical machine operating in root mode and non-root mode, respectively. Before a mode switch is performed, the host operating system operates in root mode, and in this root mode, as shown in FIG. 2c, the internal architecture of the running physical machine from bottom to top is: hardware resources 10, the host operating system 20, and an application layer 40. After the mode switch is performed, the system operates in non-root mode, and in this non-root mode, as shown in FIG. 2d, the architecture of the running physical machine from bottom to top is: the hardware resources 10, the virtualization layer 30, a guest operating system, and the application layer 40. The guest operating system is a name given to the host operating system 20 when it is running in the non-root mode.

In an embodiment of the present disclosure, two running modes, namely root mode and non-root mode, are provided for the host operating system in a running state, a virtualization layer is inserted between the hardware resources and the host operating system, a virtual resource provision function is implemented at least in the virtualization layer, and the virtualization layer switches the running host operating system from the root mode to the non-root mode, virtual resources are provided in the non-root mode, and at least applications that the host operating system is unable to run in the root mode due to insufficient resources are run on the virtual resources. The virtual resource provision function can be flexibly implemented for the running host operating system without reinstalling the host operating system, supporting the running of more applications and expanding the capabilities of the host operating system.

It is noted herein that, from the perspective of program code, implementation code involved in the embodiments of the present disclosure for "the function of providing virtual resources" and "switching from root mode to non-root mode" and the like belongs to the virtualization layer 30, and this code resides under the host operating system. However, from the perspective of the running state, before the mode switch is actually completed, at least part of the program code in the virtualization layer 30 is executed during the running of the host operating system. In other words, at least part of the program code in the virtualization layer 30 is executed in root mode, where the program code in the virtualization layer 30 that is executed in root mode at least includes: program code for switching from root mode to non-root mode, such as the upper half of a switching function mentioned in the following embodiments. Correspondingly, after the mode switch is actually completed, part of the

program code is executed in non-root mode, such as the lower half of the switching function mentioned in the following embodiments. Additionally, the program code in the virtualization layer 30 responsible for providing virtual resources are also run in root mode, but this part of the code is run after the switch is completed. The virtualization layer 30 also externally provides some interface code, part of which runs in non-root mode and part of which requires switching to root mode to run, depending on the function of the interface code.

In an optional embodiment, as shown in FIG. 3a, the virtualization layer 30 switching the running host operating system from root mode to non-root mode includes three parts:

S1: Create a memory page table for storing a mapping relationship between guest physical addresses in the non-root mode and host physical addresses in root mode;

S2: Create an information carrier object required for mode switching, where the information carrier object is used to synchronize context information between the root mode and the non-root mode;

S3: Switch the running host operating system from the root mode to non-root mode based on the information carrier object and the memory page table.

The following describes the detailed implementation of the above three parts.

Step S1: Create a memory page table

In the root mode, the physical address space has host physical addresses, where a host physical address is represented by HPA (Host Physical Address), and the host physical addresses are addresses of physical memory in physical memory resources; in the non-root mode, the physical address space corresponds to a virtual address space, where the virtual address space has guest physical addresses, a guest physical address is represented by GPA (Guest Physical Address), and the guest physical addresses are addresses of virtual memory in virtual memory resources.

In this embodiment, the virtualization layer 30 can create a memory page table, where the memory page table is used to store a mapping relationship between guest physical addresses in the non-root mode and host physical addresses in the root mode. The memory page table may be represented by an EPT (Extended Page Tables). It should be noted that, when the memory page table is initially created, since the host operating system has not yet run in the non-root mode and has not performed operations such as memory reclamation or memory swapping, the guest physical addresses in the non-root mode are the same as the host physical addresses in the root mode. Subsequently, as the host operating system is used in the non-root mode, for example, through memory reclamation or memory swapping, the guest physical addresses may no longer be equal to the host physical addresses, and the memory page table can dynamically

maintain the mapping relationship between the guest physical addresses and the host physical addresses.

In an optional embodiment, as shown in FIG. 3b, the process of creating the memory page table includes: S11, establish a page table structure corresponding to the memory page table; S12, allocate a root page of the memory page table from a physical address space; S13, generate an entry address of the memory page table according to a host physical address of the root page, and add the entry address to the page table structure; S14, create a multi-level memory page table according to an address range of the physical address space and a page granularity used for mapping guest physical addresses to host physical addresses.

First, a page table structure is created, and the function of this page table structure is to store various information related to the memory page table, such as storing an entry address of the memory page table, an address lock array (Address Lock Array) for operating on the memory page table, and the address of a special advanced programmable interrupt controller (Advanced Programmable Interrupt Controller, APIC) page, etc. The entry address of the memory page table is used to access the memory page table; for example, the entry address may be a start address of the memory page table. Each entry in the address lock array corresponds to a page table entry and contains a lock flag bit used to indicate whether the entry is locked; when not locked, a process can access the memory space corresponding to the page table entry. The address of the special APIC page is mainly used to access the special APIC page. After creating the page table structure, a physical page may be allocated from the physical address space provided by the physical storage medium to serve as the root page of the memory page table; the entry address of the memory page table is generated according to the host physical address of the root page and added to the page table structure. The form of the entry address of the memory page table varies depending on the type of the physical computing resource object and the manufacturer. For example, the host physical address of the root page may be directly used as the entry address of the memory page table. As another example, the host physical address of the root page plus a set offset or flag bits may be used as the entry address of the memory page table, which is not limited herein.

The physical address space has an address range; for example, the address range of the physical address space may be 0-8G or 0-16G, etc. Additionally, in this embodiment, in the non-root mode, a single page granularity supported by the host operating system may be broken, allowing the page granularity for address mapping to be set as needed; for example, the page granularity for address mapping can include, but is not limited to: 4K (Kilobyte), 2M (Megabyte), or 1G (Gigabyte), etc. In view of this, in an optional embodiment, the

virtualization layer 30 can create a multi-level memory page table according to the address range of the physical address space and the page granularity used for mapping guest physical addresses to host physical addresses. The multi-level memory page table may be a 2-level memory page table, a 4-level memory page table, or a 5-level memory page table, etc., specifically depending on the address range of the physical address space and the page granularity. For example, for the same address range, the smaller the page granularity, the more levels of the memory page table are required; as another example, for the same page granularity, the larger the address range, the more levels of the memory page table are required.

Optionally, in this embodiment, the physical address space is divided into a first address space and a second address space. The second address space is larger than the first address space; that is to say, the address size of the second address space is greater than the address size of the first address space. For example, when the size of the physical address space is 8G, the address range of the physical address space may be divided into 0G-4G and 4G-8G, where the first address space may be the address space corresponding to 0G-4G, and the second address space may be the address space above 4G; as another example, if the size of the physical address space is 16G, the physical address space may be divided into 0G-4G and 4G-16G, where the first address space may be the address space corresponding to 0G-4G, and the second address space may be the address space above 4G. Based on this, during the process of creating the multi-level memory page table, corresponding multi-level memory page tables may be created separately for the first address space and the second address space. The first address space is mainly used as a memory address space, and the second address space may be used at least as an IO address space. The memory address space is used to describe the storage locations of programs and data in memory, and the IO address space is used to describe the address range of I/O devices of the physical machine. For example, the physical address space is typically divided into several different address ranges. The address space with an address range of 0-4GB is referred to as the first address space, and the address space with an address range above 4GB is referred to as the second address space. The first address space is typically referred to as "basic memory" or "system memory," which includes memory used by the host operating system, applications, and basic hardware drivers; that is, the first address space is mainly used as a memory address space. The second address space is typically referred to as "large-capacity memory" or "extended memory." This memory capacity is typically used for devices such as high-performance computers, servers, and workstations, to support more complex computing tasks and multitasking. For example, the second address space may be used at least as an IO address space.

For the first address space, a first mapping relationship between guest physical addresses and host physical addresses is formed with a first page granularity; that is, the guest physical addresses and the host physical addresses are mapped according to a page size of the first page granularity, and a multi-level memory page table corresponding to the first address space is created according to the first mapping relationship; for the IO address space in the second address space, a second mapping relationship between guest physical addresses and host physical addresses are formed with a second page granularity; that is, the guest physical addresses and the host physical addresses are mapped according to a page size of the second page granularity, and a multi-level memory page table corresponding to the IO address space is created according to the second mapping relationship; the second page granularity is greater than the first page granularity, for example, the first page granularity is 2M and the second page granularity is 1G. Adopting the first page granularity (small page granularity) for memory mapping can reduce internal fragmentation in memory pages, reduce memory waste, and reduce the loss of page table entries in the cache, thereby improving address translation efficiency. Adopting the second page granularity (large page granularity) for memory mapping reduces page table entry data, reduces the number of levels in the memory page table, lowers memory management overhead, and improves the performance of the physical computing resource object.

Further optionally, the second address space may also be used as a memory address space. Based on this, the virtualization layer 30 may further form a third mapping relationship between guest physical addresses and host physical addresses for the memory address space in the second address space with a third page granularity; that is, the guest physical addresses and the host physical addresses are mapped according to a page size of the third page granularity, and a multi-level memory page table corresponding to the memory address space is created according to the third mapping relationship; where the third page granularity is less than the second page granularity, but the size of the third page granularity is not limited.

In an optional embodiment, the third page granularity may be equal to the first page granularity; for example, the first page granularity and the third page granularity are both 2M; that is to say, for the memory address space, a smaller page granularity may be used for address space mapping, and for the IO address space, a larger page granularity may be used for address space mapping, which can make reasonable use of the physical address spaces and improve memory management performance.

Further optionally, in some application scenarios, the memory address space in the second address space includes an APIC page, and the APIC page adopts a smaller page granularity,

for example, a fourth page granularity, where the third page granularity is greater than the fourth page granularity. For example, the third page granularity is 2M and the fourth page granularity is 4K.

In the case that the memory address space in the second address space contains an APIC page of the fourth page granularity, it means that the APIC page of the fourth page granularity belongs to a physical page of the third page granularity. The physical page of the third page granularity to which the APIC page of the fourth page granularity belongs is briefly referred to as a first physical page. Then, the first physical page of the third page granularity to which the APIC page belongs and a target memory page table corresponding to the first physical page can be obtained, where the target memory page table corresponding to the first physical page is a last-level page table in the multi-level memory page table, and the page table entries in this page table point to the address space in the first physical page. In order to adapt to the page granularity of the APIC page, the first physical page may be split into multiple sub-pages of the fourth page granularity, and a lower-level memory page table may be further extended below the target memory page table; that is, a lower-level memory page table is added below the target memory page table, and the lower-level memory page table is used to store the host physical addresses of the multiple split sub-pages. For example, the host physical address corresponding to each sub-page is stored in each page table entry of the lower-level memory page table, and this host physical address is the start address of the sub-page.

In this embodiment, when creating the memory page table, a hybrid mapping granularity approach combining the first page granularity, the second page granularity, the third page granularity, and the fourth page granularity is adopted, which supports not only small page granularity but also large page granularity, can meet the requirements of different physical page mappings, and improves the flexibility of memory mapping.

Furthermore, in the non-root mode provided by the embodiments of the present disclosure, when the memory page table is created, the physical address space of the host is no longer restricted, and it is allowed to create a memory page table for the entire physical address space, that is, full memory mapping is supported. Therefore, not only can a memory page table be created for user-mode pages, but also memory page table creation is supported for kernel-mode pages. By supporting memory mapping of the entire physical address space, a foundational framework is provided for subsequent various operations based on memory mapping. For example, when memory swapping is performed based on memory mapping, not only swapping of user-mode pages is supported, but also swapping of kernel-mode pages is supported, that is, full memory swapping is supported.

In this embodiment, taking the first page granularity as 2M, the second page granularity as 1G, the third page granularity as 2M, the fourth page granularity as 4K, and the multi-level memory page table implemented as a four-level memory page table as an example, the specific process of creating the memory page table is exemplarily described.

The following gives an example of a four-level memory page table. The four-level memory page table includes: a page global directory (Page Global Directory, PGD), a page upper directory (Page Upper Directory, PUD), a page middle directory (Page Middle Directory, PMD), and a page table entry (Page Table Entry, PTE). The PGD is the highest-level directory, and the PGD includes multiple global page directory entries pgd_t, each pgd_t can map a host physical address (HPA) of 512G in size, and pgd_t points to a lower-level page directory (PUD); the PUD includes multiple upper page directory entries pud_t, each pud_t can map a host physical address (HPA) of 1G in size, and pud_t points to a lower-level page directory (PMD); the PMD includes multiple middle page directory entries pmd_t, each pmd_t can map a host physical address (HPA) of 2M in size, and pmd_t points to a lower-level page directory (PTE); the PTE includes multiple direct page directory entries pte_t, each pte_t can map a host physical address (HPA) of 4K in size, and each pte_t points to a corresponding physical page in the host physical address (HPA).

1) For a first address space with an address range of 0G-4G, a first mapping relationship between guest physical addresses and host physical addresses is formed with a page granularity of 2M, and a multi-level memory page table corresponding to the first address space is created according to the first mapping relationship.

a1. According to the guest physical address (GPA) in the first address space, determine the page table entry pgd_t in the PGD-level page table according to the index value of the PGD corresponding to the guest physical address (GPA), where the pgd_t points to a PUD-level page table.

b1. According to a flag bit in pgd_t, determine whether the PUD-level page table exists; if it does not exist, allocate a page to serve as the PUD-level page table.

c1. If the PUD-level page table exists, then according to the first page granularity (e.g., 2M), determine whether the PUD-level page table needs to be extended; since each pud_t can map a host physical address (HPA) of 1G in size, and 1G is greater than the first page granularity (e.g., 2M), it is determined that a lower-level PMD-level page table needs to be extended, in preparation for subsequent mapping.

d1. According to the index value of PUD corresponding to the guest physical address (GPA), determine the page table entry pud_t in the PUD-level page table, where the pud_t points to a PMD-level page table.

e1. According to a flag bit in the pud_t, determine whether the lower-level PMD-level page table exists; if it does not exist, allocate a page to serve as the PMD-level page table.

f1. According to the first page granularity (e.g., 2M), determine whether the PMD-level page table needs to be extended. Since each pmd_t can map a host physical address (HPA) of 2M in size, it is determined that extension to the lower-level PTE-level page table is not needed.

g1. According to the index value of the PMD corresponding to the guest physical address (GPA), determine the page table entry pmd_t in the PMD-level page table.

h1. In a case that the mapped page granularity is 2M, directly establish a page table entry pmd_t in the PMD-level page table, where pmd_t points to a physical page of the host physical address (HPA), and the size of the physical page is 2M. Therefore, it is required to assemble the page table entry pmd_t of the PMD-level page table. The pmd_t may include a physical page frame number (Page Frame Number, PFN) corresponding to the respective HPA, and the page table entry is set to be accessible, and a large page flag is set to 1. For example, a large page may be a physical page having a size of 2M or 1G, and a physical page having a size of 4K does not belong to a large page.

i1. Fill the assembled page table entry pmd_t of the PMD-level page table into the PMD-level page table, complete one mapping at the first page granularity (e.g., 2M), and record the mapping completion progress and the next guest physical address that needs to be mapped, until, for the first address space, the mapping at the first page granularity is established.

j1. Determine whether the mapping at the first page granularity (e.g., 2M) is completed; if not completed, return to step b; otherwise, end.

2) For an input/output (IO) address space in the second address space, form a second mapping relationship between guest physical addresses and host physical addresses at a second page granularity, and create, according to the second mapping relationship, a multi-level memory page table corresponding to the IO address space. That is, according to the address space of the host operating system recorded in the IO memory resource (iomem_resource) (i.e., the physical memory space and the physical IO space), a multi-level memory page table at a second page granularity (e.g., 1G granularity) is established for the IO address space (e.g., the memory address space above 4G) in the second address space. The process of establishing the multi-level memory page table is similar to the above-described implementation of establishing the multi-level memory page table at the first page granularity, with the difference that:

establishing the multi-level memory page table at the second page granularity (e.g., 1G granularity) requires establishing pud_t in the PUD-level page table, where pud_t points to a physical page of a host physical address (HPA), and the size of the physical page is 1G.

3) For the memory address space in the second address space, form a third mapping relationship between guest physical addresses and host physical addresses at a third page granularity (e.g., 2M), and create, according to the third mapping relationship, a multi-level memory page table corresponding to the memory address space; where the third page granularity is smaller than the second page granularity. For the process of establishing the multi-level memory page table, reference can be made to the above-described implementation of establishing the multi-level memory page table at the first page granularity, and details are not repeated herein.

4) In a case that the memory address space includes an APIC page of a fourth page granularity, obtain a first physical page of the third page granularity to which the APIC page belongs and a target memory page table corresponding to the first physical page; split the first physical page into a plurality of sub-pages of the fourth page granularity, and add a lower-level memory page table under the target memory page table, where the lower-level memory page table is used to store host physical addresses of the plurality of split sub-pages.

5) Allocate a special APIC page, where the physical address of the APIC page is a guest physical address.

6) Establish a mapping between the host physical address of the APIC and the special APIC page at the fourth page granularity (e.g., 4k granularity). The default host physical base address of the APIC (APIC_DEFAULT_PHYS_BASE) is 0xfee00000, and this physical address corresponds to a memory address space in the second memory space. This process involves splitting of the already established page at the third page granularity (e.g., 2M granularity). The mapping process is as follows:

a2. Sequentially perform steps a1 through e1 in step 1). According to the fourth page granularity (e.g., 4K), determine whether the PMD-level page table is required to be extended. Since each pmd_t can map a host physical address (HPA) of 2M in size, and 2M is not equal to 4K, it is determined that extension to the lower-level PTE-level page table is required.

b2. According to the index value of the PTE corresponding to the guest physical address (GPA), determine the page table entry pte_t in the PTE-level page table, assemble the content of pte_t, including the physical page frame number corresponding to the HPA, and set the page table entry to be accessible, where the large page flag is not set to 1.

c2. Fill the assembled content of the page table entry of the PTE-level page table into the corresponding pte_t, to complete the current mapping.

S2. Create information carrying objects required for mode switching

In this embodiment, in order to perform mode switching for the host operating system, on one hand, a physical descriptor structure is created for any physical computing resource object 101, as shown in step S21 in FIG. 3c. In some subsequent descriptions, the physical descriptor structure may simply referred to as a pcpu. The physical descriptor structure is used to save, during mode switching, context information of any physical computing resource object running in root mode. The context information of any physical computing resource object in root mode may include, but is not limited to: values of various registers used by any physical computing resource object, state information of any physical computing resource object, state information of the host, and related configuration information. The related configuration information includes, but is not limited to: the frequency of a High Voltage (hv) timer, etc.

In this embodiment, the implementation of the physical descriptor structure is not limited. Any implementation structure capable of saving the context information of the physical computing resource object running in root mode is applicable to the embodiments of the present disclosure. In an optional embodiment, the physical descriptor structure may include, but is not limited to, the following parts or fields: a first register structure (regs), a field carrying state information of the physical computing resource object, a field carrying state information of the host, and a field carrying configuration information. The first register structure may include, but is not limited to: a general-purpose register field, a debug register field, a stack-related register field, a floating point unit (fpu) register field, and a model specific register (MSR) field, etc.

In order to perform mode switching for the host operating system, on the other hand, a virtualization descriptor structure is created for any physical computing resource object, as shown in step S22 in FIG. 3c. In some subsequent descriptions, the virtualization descriptor structure may be simply referred to as a vcpu. The virtualization descriptor structure is used to synchronize the context information in the physical descriptor structure during mode switching. The virtualization descriptor structure represents a virtual computing resource object obtained by virtualizing the physical computing resource object.

The implementation of the virtualization descriptor structure is not limited. Any implementation structure capable of synchronizing the context information in the physical computing resource object is applicable to the embodiments of the present disclosure. In an optional embodiment, the vcpu may include, but is not limited to, the following fields: a second

register structure (regs) corresponding to the first register structure; in addition, the vcpu further includes: a field carrying a virtual computing resource object identification instruction (e.g., cpu_id), a field carrying an interrupt vector number (apic_id), a field carrying the running state of the vcpu, a field carrying a switch marker for state changes during vCPU switching, a field carrying related configuration information of APIC registers, a field carrying configuration information of specific model registers (e.g., MSR registers), a field carrying configuration information of processor opcodes (e.g., CPUID), a field carrying a descriptor of a program interruption (PI), and a field carrying configuration information related to an exit event, etc.

Further, in order to perform mode switching for the host operating system, a virtualization control structure is created for any physical computing resource object, as shown in step S23 in FIG. 3c. For example, the virtualization control structure may be implemented as a vmcs (Virtual Machine Control System). The virtualization control structure is used to save runtime state information and runtime control information of any physical computing resource object in the non-root mode. For example, the runtime state information of the physical computing resource object in the non-root mode may be values of some registers and the state of the physical computing resource object, such as active, halted (HLT), or shutdown (shutdown) states. The runtime control information mainly refers to control information for the host operating system in the non-root mode. For example, the runtime control information saved in the virtualization control structure may include, but is not limited to, the following fields: a field carrying an exit event of the non-root mode, a field carrying a specific processor model register bitmap (msr_bitmap), a field carrying APIC virtualization, a field carrying interrupt virtualization, a field carrying a preemption timer, a field carrying clock configuration, or a field carrying IO control information, etc. The information in the corresponding fields may be configured as needed during initialization of the virtualization control structure according to actual circumstances.

In an optional embodiment, the virtualization layer 30 may further create an IO bitmap (io_bitmap), as shown in step S24 in FIG. 3c. The IO bitmap is used to record access permissions of any physical computing resource object to each IO port in the non-root mode, and the access permissions are related to the IO control information in the runtime control information. The access permissions may all default to 0, that is, access to all IO ports is allowed. Subsequently, if it is necessary to intercept some IOs, the io_bitmap may be configured as needed.

Further, the virtualization layer 30 may further create a new stack. For convenience of description and distinction, the new stack is referred to as a first stack, as shown in step S25 in

FIG. 3c. The new stack (i.e., the first stack) is relative to an old stack. For convenience of description and distinction, the old stack is referred to as a second stack. The old stack (i.e., the second stack) is a stack currently used by any physical computing resource object. After switching from the root mode to the non-root mode, any physical computing resource object will continue to use the old stack to run in the non-root mode. It is noted herein that after switching from the root mode to the non-root mode, the physical computing resource object may be implemented as a virtual physical computing resource object; for example, a physical cpu may be implemented as a virtual cpu. The first stack is a stack prepared for the virtualization layer 30, and is used for the virtualization layer 30 to perform memory access in the root mode based on the new stack (i.e., the first stack).

In an optional embodiment, the virtualization layer 30 may further initialize the virtualization descriptor structure and the virtualization control structure respectively, as shown in step S26 in FIG. 3c. The initialization process is a process of enabling required functions, allocating memory pages for the required functions, and configuring initial values. The initialization process of the virtualization descriptor structure may include adding initialization information corresponding to the functions to be enabled to corresponding fields of the virtualization descriptor structure. The initialization process of the virtualization control structure may include adding initialization information corresponding to the functions to be enabled to corresponding fields of the virtualization control structure.

Optionally, the virtualization layer 30 may configure, in the virtualization control structure, runtime control information for operation in the non-root mode for any physical computing resource object. For example, the runtime control information may include, but is not limited to, at least one of: MSR control information, PI control information, APIC control information, hardware register operation instruction information, an entry address of a memory page table, and related configuration information for exiting the non-root mode.

Further optionally, when configuring the runtime control information, the virtualization layer 30 is specifically configured to perform at least one of the following configuration operations:

a3. Establish a specific model register bitmap (e.g., msr_bitmap), and allocate a second physical page for saving the bitmap;

b3. Set a PI interrupt function, and set a PI notification vector;

c3. Set an APIC interrupt control function, allocate a third physical page from a physical address space for saving an APIC table, and save a base address of the page where the APIC table is located in an APIC field; additionally, it is also required to set APIC_ID to the apic_id

of the current physical computing resource object (e.g., CPU), and an APIC local vector register (APIC Local Vector Register, APIC_LVR) inherits the setting of the current physical computing resource object (e.g., CPU); if the APIC timer is intercepted, the configuration and handler function of the timer may also be configured;

d3. Set a processor opcode (e.g., CPUID), and establish a cache of the virtual computing resource object;

e3. Set PAUSE-Loop Exiting (PAUSE-Loop Exiting, PLE), where the PLE configuration is used to reduce waste of virtual computing resources caused by loop waiting.

f3. Save the entry address of the multi-level memory page generated in step S1 in the virtualization control structure, and further set, as required, the RIP instruction position to be executed after exiting the non-root mode;

g3. Default configuration of general-purpose registers in the root mode and the non-root mode.

The process of initializing each virtualization descriptor structure and virtualization control structure is as follows:

1) Sequentially load each virtualization descriptor structure (vcpu), and perform initialization settings for each vcpu structure;

2) Load the virtualization control structure (vmcs) structure corresponding to the vcpu as the current vmcs;

3) Configure runtime control information for the current vmcs, that is, execute the above steps a3 through g3;

4) Clear the current vmcs structure;

5) Determine whether vcpus that need to be initialized have been initialized; if initialization is not complete, repeat steps 1) through 4); if initialization is complete, end the initialization.

In summary, performing mode switching for the host operating system involves switching from physical compute resource objects to virtual compute resource objects, and additionally involves synchronization of context information, access control of IO operations, and switching of memory access. Therefore, a physical descriptor structure corresponding to a physical compute resource object and a virtualization descriptor structure corresponding to a virtual compute resource object are created to facilitate synchronization of context information during the mode switching process, and an IO bitmap is created to facilitate access control of IO operations during the mode switching process, and a first stack is created to facilitate switching of memory access by the virtualization layer 30 during the mode switching process.

Here, after the memory page table is created and various information carrier objects are created, the foundation for mode switching is established. On this basis, the running host operating system is switchable from the root mode to the non-root mode based on the information carrier objects and the memory page table.

S3: Switch the running host operating system from root mode to non-root mode based on the information carrier objects and the memory page table.

In the present embodiment, mode switching may be performed for any physical compute resource object, and the process of performing mode switching for each physical compute resource object is the same. For ease of description and distinction, the following description takes performing mode switching for a target physical compute resource object as an example. The target physical compute resource object may be any physical compute resource object.

As shown in FIG. 3d, the process of performing mode switching for the target physical compute resource object includes: step S31, in a case of performing mode switching for the target physical compute resource object, save context information of the target physical compute resource object in the root mode to a physical descriptor structure corresponding to the target physical compute resource object; step S32, synchronize values of respective registers and configuration information of segment registers in the context information in the physical descriptor structure respectively to a virtualization descriptor structure and a virtualization control structure corresponding to the target physical compute resource object; step S33, control the target physical compute resource object to run according to the virtualization descriptor structure, the virtualization control structure, and the memory page table, to switch the running host operating system from the root mode to the non-root mode.

For example, the values of the registers in the context information in the physical descriptor structure are synchronized to the virtualization descriptor structure corresponding to the target physical compute resource object, and the configuration information of the segment registers in the context information in the physical descriptor structure is synchronized to the virtualization control structure corresponding to the target physical compute resource object. Herein, the configuration information of the segment registers may include, but is not limited to: the size of a segment, a starting address of the segment, and management attributes of the segment. For example, the management attributes of the segment include: write-prohibited, execution-prohibited, or system-dedicated, etc.

In the present embodiment, step S1 and step S2 may be executed when the virtualization layer 30 is started or initialized, and step S3 may be executed on demand according to dynamic

requirements (external instructions). Here, whether to perform mode switching for the target physical compute resource object may be determined by an external instruction.

In an optional embodiment, the host operating system 20 of the present embodiment externally provides a symmetrical multi-processing (Symmetrical Multi-Processing, SMP) call interface, which allows an external entity to send an SMP call request to any physical compute resource object on the physical machine. The SMP call request is used to indicate that mode switching needs to be performed for the any physical compute resource object. For ease of distinction and description, the following description takes an external entity sending an SMP call request to the target physical compute resource object as an example. For example, an upper-layer application running on the host operating system 20 may send an SMP call request to the target physical compute resource object, or a developer may send an SMP call request to the target physical compute resource object through the host operating system 20.

Upon receiving the SMP call request, the target physical compute resource object may initiate a call request to a switching function to trigger mode switching. Correspondingly, the virtualization layer 30 provides the switching function (e.g., a switch_vcpu function) configured for mode switching of the host operating system, for the target physical compute resource object to execute the switching function so as to switch the host operating system from the root mode to the non-root mode, which specifically refers to executing the operation of saving the context information of the target physical compute resource object in root mode to the physical descriptor structure corresponding to the target physical compute resource object and subsequent operations (e.g., step S32 and step S33), so as to perform mode switching for the physical compute resource object. After entering the non-root mode, the virtualization layer 30 may also control the target physical compute resource object to exit from the non-root mode to the root mode when an exit event configured in the virtualization control structure occurs.

In an optional embodiment, during the mode switching process, before saving the context information, it may be determined whether a state of a switching flag corresponding to the target physical compute resource object is a to-be-switched state; if the determination result is yes, the switching flag is updated to a switching-in-progress state. For example, the target physical compute resource object may acquire the state of the switching flag; if the state of the switching flag is an unswitched state (e.g., VMX_OFF), update the switching flag to the switching-in-progress state (e.g., VMX_SWITCH); if the state of the switching flag is a switching-completed state (e.g., VMX_ON), directly return from the SMP call without performing mode switching. Optionally, on the basis of returning from the SMP call, some other cleanup operations may be performed. The virtualization layer 30 may also configure a

first instruction for first entry into the non-root mode, the first instruction being an entry address of the switching function.

Here, the switching function is divided into an upper half and a lower half. For example, the switching function may be entered through a function entry such as switch_vcpu. Here, as the host operating system switches between different modes, code paths after different entries into the switching function differ. For the first time switching from the root mode to the non-root mode, an SMP call may be sent to the target physical compute resource object, the call being used to trigger mode switching. First, the state of the switching flag is detected; upon determining the to-be-switched state, mode switching is triggered, and the upper half of the switching function is entered. In the upper half, the switching flag is first set to the switching-in-progress state (e.g., VMX_SWITCH), and the first instruction to be executed after entering the non-root mode is set, that is, the entry address of the switching function. Thereafter, in the upper half of the switching function, the operation of switching the host operating system from the root mode to the non-root mode is performed. After the target physical compute resource object enters the non-root mode, the above-set first instruction is executed. The first instruction is to execute the switching function. At this point, the flag has already been set to the switching-in-progress state, so the lower half of the switching function is entered. The switching is completed in the lower half of the switching function, and the SMP call returns.

Optionally, for ease of distinction and description, the stack used by the target physical compute resource object currently (i.e., before switching to the non-root mode is referred to as a second stack. When the virtualization layer 30 saves the context information of the target physical compute resource object in the root mode to the physical descriptor structure corresponding to the target physical compute resource object, it is specifically configured to: save the values of registers in the context information and top-of-stack and bottom-of-stack addresses of the second stack to the physical descriptor structure. For example, the values of the registers in the context information may be saved to the physical descriptor structure, which specifically includes saving control registers, segment registers, and general-purpose registers to a register structure in the physical descriptor structure, and saving the top-of-stack and bottom-of-stack addresses of the second stack to corresponding stack fields of a host (host) register in the physical descriptor structure (pcpu), which will be used in subsequent mode switching.

The virtualization layer 30 may also switch the second stack currently used by the target physical compute resource object to the newly created first stack, for the virtualization layer 30 to continue running in the root mode based on the first stack.

Further optionally, when the virtualization layer 30 controls the target physical compute resource object to run according to the virtualization descriptor structure, the virtualization control structure, and the memory page table, it is specifically configured to: load values of dedicated registers in the virtualization control structure and the virtualization descriptor structure. The dedicated registers are a group of registers having specific functions, and the dedicated registers have special purposes and can be accessed and used by specific instructions or hardware modules. For example, PI registers, debug registers (DR) registers, MSR registers, or segment registers, etc. Further, segment registers of the pcpu may be saved, and values of the segment registers in the vcpu are loaded. Runtime control information in the virtualization control structure is injected into the target physical compute resource object, and values of general-purpose registers in the virtualization descriptor structure are loaded. The general-purpose registers are a group of registers usable by a developer for storing general information such as data and addresses. The general-purpose registers can be accessed and used by any instruction in a program. For example, pointer registers or index registers, etc. Here, the values of corresponding registers are loaded, which means that the values of the registers stored in the vcpu are loaded to specific hardware registers, so that the target physical compute resource object is able to run in the non-root mode. A mode switching instruction is executed to control the target physical compute resource object to enter the non-root mode and start running from the first instruction, performing memory management and access based on the memory page table during the running process.

Further optionally, after switching the running host operating system from the root mode to the non-root mode, the virtualization layer 30 may also restore the second stack to the top-of-stack and bottom-of-stack addresses saved in the physical descriptor structure, and update the switching flag to the switching-completed state (e.g., VMX_ON). Here, after switching from the root mode to the non-root mode, the second stack (i.e., the old stack) is restored to the top-of-stack and bottom-of-stack addresses saved in the physical descriptor structure, which facilitates the target physical compute resource object to continue running in the non-root mode based on the second stack.

In the embodiments of the present disclosure, by leveraging hardware virtualization functionality, a lightweight virtualization layer is inserted between the host operating system and hardware resources. A virtual resource providing function is implemented in the virtualization layer. The virtualization layer switches the running host operating system from root mode to non-root mode, thereby providing at least one virtual resource in the non-root mode and running at least one application on the at least one virtual resource. The at least one

application includes at least an application that cannot run in the root mode due to insufficient resources. Without reinstalling the host operating system, the virtual resource providing function can be flexibly implemented for the running host operating system, solving the problem that applications cannot run due to insufficient resources caused by not enabling or missing relevant system functions in the root mode. This enables the host operating system to support running more applications, which is beneficial for extending the capabilities of the host operating system. Herein, after the host operating system is switched from the root mode to the non-root mode, the at least one application running on the virtual resources provided in the non-root mode includes, but is not limited to: performing target functions such as memory management, file management, virtualization functionality, and scheduling optimization in the non-root mode.

In the following embodiments, memory page fault management in the non-root mode is described in detail, taking memory page fault management in memory management as an example. It is noted here that an application scenario of memory page faults is memory swap (swap), that is, memory page faults may occur during the memory swap process, but memory page faults are not limited to the memory swap process. To facilitate demonstrating the advantages of the memory page fault management implemented in the non-root mode according to the embodiments of the present disclosure, the following description takes using memory page faults for memory swap as an example, and first provides a relevant introduction to memory swap.

The memory swap function in the host operating system (e.g., Linux system) is a mechanism that uses free disk space as memory extension. When memory pages in main memory (e.g., RAM) are insufficient, the swap function can move a portion of infrequently used memory pages to the disk to free up space for other pages to use. This process is called page swap out (swap out). When a process uses the swapped-out content again, a page swap-in (swap in) is triggered.

The swap function of the host operating system (e.g., Linux system) includes two aspects: a swap partition and a swap file. The swap partition is used as swap disk space, also referred to as swap space. It is typically allocated when the host operating system is installed, and may also be manually created through commands such as fdisk or mkswap. After free disk space is divided into a swap partition, it may be mounted as a swap partition through the swapon command. The advantages of the swap partition are high speed and high stability, but the partition size needs to be planned in advance and cannot be dynamically adjusted. Herein, the fdisk command is used to create, delete, modify, and display disk partitions. The mkswap

command is used to create a swap partition. The swapon command is used to enable a swap partition on the physical machine.

A swap file is a file created on a normal file system and may serve as swap space. The swap file is created through commands such as dd and fallocate, then formatted as a swap file system using the mkswap command, and finally mounted as swap space through the swapon command. Herein, both the dd command and the fallocate command are used to create files. The swap file has the advantage of high flexibility, supports dynamic size adjustment, and facilitates convenient management, but its performance is slightly inferior to that of the swap partition.

In the host operating system (e.g., Linux system), the size of the swap space is typically set to two or three times the physical memory. If the physical memory is sufficient, the utilization rate of the swap space is very low and does not have a significant impact on the performance of the physical machine. If the physical memory is insufficient, the utilization rate of the swap space is high, which seriously affects the performance of the physical machine. Therefore, the performance of the swap space is also critical to the stability of the physical machine performance.

In addition, the swap function of the kernel of the host operating system (e.g., Linux) targets user processes and does not directly affect swapping in and out of kernel pages. If memory is insufficient, the kernel attempts to reclaim some unnecessary pages and swaps them to the disk through the swap function to release memory space. These unnecessary pages include pages of user processes, cache pages, anonymous memory pages, etc., but do not include kernel pages, because kernel pages are typically non-swappable. Kernel code and data are typically locked in memory and are not swapped to the disk, and therefore are not affected by the swap function. Moreover, the swap function currently only supports 4K small pages, whereas in current cloud computing scenarios, large pages (e.g., 2M or 1G) are increasingly used, and the swap function for large pages is not yet supported. If the running operating system is not configured with the swap function, enabling it online can only support the file-based approach, which has low performance and seriously affects system stability.

Although some missing system functions of the host operating system may be implemented by loading functional modules, this approach is invasive and is constrained by various existing structures and function capabilities of the operating system, and may not solve all problems, such as the swapping of kernel-mode pages mentioned above. In the embodiments of the present disclosure, based on the virtualization layer, switching from the root mode to the non-root mode is performed, and virtual memory resources are provided in the non-root mode,

so that the memory swap function is implementable without being limited by whether the host operating system has enabled the memory swap function. In addition, the solution provided by the embodiments of the present disclosure is more transparent and imperceptible to upper-layer applications, and is capable of achieving more comprehensive functionality.

Further, in the embodiments of the present disclosure, when the memory page table is created, a hybrid mapping granularity approach combining a first page granularity, a second page granularity, a third page granularity, and a fourth page granularity is adopted, which supports not only small page granularity but also large page granularity, can meet the requirements of different physical page mappings and improve the flexibility of memory mapping. Using small page granularity for memory mapping can reduce internal fragmentation in memory pages and reduce memory waste; using large page granularity for memory mapping reduces page table entry data, reduces the number of levels of the memory page table, reduces memory management overhead, and improves the performance of the physical compute resource object.

Further, in the embodiments of the present disclosure, when the memory page table is created, the physical address space of the host is no longer restricted, and it is allowed to create the memory page table for the full physical address space, that is, full memory mapping is supported. Therefore, not only can the memory page table be created for user-mode pages but also memory page table creation is supported for kernel-mode pages. By supporting memory mapping of the full physical address space, a foundational framework is provided for various subsequent operations based on memory mapping. For example, when memory swap is performed based on memory mapping, not only swapping of user-mode pages is supported, but swapping of kernel-mode pages is also supported, that is, full memory swap is supported. The following is a brief description of the process of implementing memory page fault management after switching to non-root mode:

1) The memory page table (e.g., ept page table) prepared for the mode switching of the host operating system has a one-to-one correspondence between guest physical addresses (GPA) and host physical addresses (HPA). Normal operation in the non-root mode does not trigger a page fault. To extend the memory management function of the host operating system, some memory pages (the page granularity of which may be 4K or 2M) may be reclaimed by methods such as reclaiming free pages, compressing pages in use, swapping memory pages to disk, or extending new virtual memory space, and corresponding page table entries of the memory page table are cleared, so that subsequent access in the non-root mode may trigger a

page fault. Currently, a batch of free pages' memory page table entries may be directly cleared to trigger page faults.

2) Since a batch of memory pages has been cleared and reclaimed in step 1), a simple memory management system, for example, using status bits for memory management, may be used for subsequent page fault handling.

3) Before performing page fault exception handling, a handling function may be set in an exit event of the non-root mode, for example, setting a page fault handling function (ept_handler) associated with the exit reason EPT violation (EXIT_REASON_EPT_VIOLATION).

4) In the page fault handling function (ept_handler), an unused host physical page may be selected from the underlying memory management system, and is marked as used.

5) A mapping relationship is established between the physical address (HPA) of the unused host physical page and the guest physical address (GPA) where the page fault occurs, and a corresponding page table entry is established in the corresponding memory page table (e.g., EPT page table), so that the host operating system implements memory management in the non-root mode, where the memory management includes but is not limited to memory swapping and the like.

It should be noted that the execution entity of each step of the method provided in the above embodiments may be the same device, or the method is also executed by different devices as execution entities. For example, the execution entity of step R11 to step R12 may be device A; for another example, the execution entity of steps R11 and R12 may be device A, and the execution entity of step R13 may be device B; and so on.

In addition, in some processes described in the above embodiments and the accompanying drawings, multiple operations appearing in a particular order are included, but it should be clearly understood that these operations may not be executed in the order in which they appear herein or may be executed in parallel, and serial numbers of the operations, such as R11 to step R12, are merely used to distinguish different operations, and the serial numbers themselves do not represent any execution order. In addition, these processes may include more or fewer operations, and these operations are executed sequentially or in parallel. It should be noted that the descriptions of "first", "second", etc. herein are used to distinguish different messages, devices, modules, and the like, and do not represent a sequential order, nor limit that the "first" and the "second" are of different types.

FIG. 4 is a schematic structural diagram of an address translation apparatus provided by an exemplary embodiment of the present disclosure. As shown in FIG. 4, the apparatus

includes: a construction module 41, a determination module 42, a query module 43, and a generation module 44.

The construction module 41 is configured to pre-construct a first RMAP table corresponding to an EPT, where the EPT includes multi-level page tables corresponding to each of at least one GPA; the first RMAP table stores address information of a last-level page table entry corresponding to each of the at least one GPA.

The determination module 42 is configured to, in response to a first address translation trigger event, determine a first GPA for which address translation is required; the query module 43 is configured to query, in the first RMAP table, address information of a last-level page table entry corresponding to the first GPA.

The generation module 44 is configured to, if the address information of the last-level page table entry corresponding to the first GPA is found by query, generate an HPA corresponding to the first GPA according to the address information of the last-level page table entry corresponding to the first GPA and an address offset of the first GPA.

In an optional embodiment, the construction module is further configured to: pre-construct a second RMAP table corresponding to the EPT, the second RMAP table storing address translation information from at least one HPA to a corresponding GPA; the determination module is further configured to: in response to a second address translation trigger event, determine a first HPA for which address translation is required, and query, in the second RMAP table, address translation information from the first HPA to a corresponding GPA; the generation module is further configured to: generate a GPA corresponding to the first HPA according to the first HPA and the address translation information from the first HPA to the corresponding GPA and an address offset of the first HPA.

In an optional embodiment, the construction module is specifically configured to: divide a GPA address space corresponding to the EPT into at least one address space, and create a base structure for each of the at least one address space, the base structure corresponding to a set page table level; for any GPA in the EPT, determine a first structure corresponding to the any GPA from base structures according to address information of the any GPA and address space corresponding to the base structures; and record address information of a last-level page table entry corresponding to the any GPA through the first structure according to a relationship between a level of a last-level page table corresponding to the any GPA and the set page table level.

Optionally, the construction module is specifically configured to: if a level of a last-level page table corresponding to any GPA is the same as the set page table level, add address

information of a last-level page table entry corresponding to the any GPA to an information field in the first structure, and set flag information in the first structure to a first value; if the level of the last-level page table corresponding to the any GPA is different from the set page table level, create a second structure adapted to the level of the last-level page table corresponding to the any GPA, add the address information of the last-level page table entry corresponding to the any GPA to an information field in the second structure, and add address information of the second structure to the information field in the first structure, and set the flag information in the first structure to a second value.

Further optionally, the construction module is specifically configured to: in a case that the level of the last-level page table corresponding to any GPA is an upper level of the set page table level, determine whether a second structure already exists; if the second structure does not exist, create a new structure adapted to the level of the last-level page table corresponding to the any GPA, as the second structure; in a case that the level of the last-level page table corresponding to the any GPA is a lower level of the set page table level, create a new structure adapted to the level of the last-level page table corresponding to the any GPA, as the second structure.

In an optional embodiment, the construction module is specifically configured to: divide an HPA address space corresponding to the EPT into at least one address space, and create a base structure for the at least one address space respectively, the base structure corresponding to a set page granularity; for any HPA corresponding to a GPA in the EPT, determine a third structure corresponding to the any HPA from base structures according to address information of the any HPA and address spaces corresponding to the base structures; and record address translation information from the any HPA to a corresponding GPA through the third structure according to a relationship between a page granularity corresponding to the any HPA and the set page granularity.

Optionally, the construction module is specifically configured to: if a page granularity corresponding to any HPA is the same as the set page granularity, record address translation information from the any HPA to a corresponding GPA through an information field in the third structure, and set first flag information in the third structure to a third value; if a page granularity corresponding to the any HPA is different from the set page granularity, create a fourth structure adapted to the page granularity of the any HPA, record the address translation information from the any HPA to the corresponding GPA through an information field in the fourth structure, and add address information of the fourth structure to the information field in the third structure, and set the first flag information in the third structure to a fourth value.

Further optionally, the construction module is specifically configured to: in a case that any HPA corresponds to a plurality of GPAs, create a fifth structure, add address information of the fifth structure to the information field in the third structure, and add address translation information from the any HPA to the plurality of GPAs respectively to the fifth structure, and add a flag bit indicating a one-to-many relationship in the third structure; in a case that any HPA corresponds to one GPA, directly add address translation information from the any HPA to the corresponding GPA to the information field in the third structure;

The construction module is specifically configured to: in a case that any HPA corresponds to a plurality of GPAs, create a sixth structure, add address information of the sixth structure to the information field in the fourth structure, and add address translation information from the any HPA to the plurality of GPAs respectively to the sixth structure, and add a flag bit indicating a one-to-many relationship in the fourth structure; in a case that any HPA corresponds to one GPA, directly add address translation information from the any HPA to the corresponding GPA to the information field in the fourth structure.

In an optional embodiment, the query module is specifically configured to: determine a first structure corresponding to the first GPA from the base structures according to address information of the first GPA and the address spaces corresponding to the base structures included in the first RMAP table, the base structure corresponding to a set page table level; and obtain address information of a last-level page table entry corresponding to the first GPA through the first structure corresponding to the first GPA.

Optionally, the query module is specifically configured to: obtain flag information from the first structure; if the flag information is a first value, obtain address information of a last-level page table entry corresponding to the first GPA from an information field in the first structure; if the flag information is a second value, obtain address information of a second structure from an information field in the first structure, and obtain address information of the last-level page table entry corresponding to the first GPA from an information field of the second structure according to the address information of the second structure.

In an optional embodiment, the apparatus further includes the following modules:

an addition module, configured to, in a case of adding a second GPA in the EPT, add, in the first RMAP table, address information of a last-level page table entry corresponding to the second GPA;

a deletion module, configured to, in a case of deleting a third GPA in the EPT, delete, in the first RMAP table, address information of a last-level page table entry corresponding to the third GPA;

a modification module, configured to, in a case of modifying first address information of a last-level page table corresponding to a fourth GPA to second address information in the EPT, modify, in the first RMAP table, the first address information of the last-level page table corresponding to the fourth GPA to the second address information;

an extension module, configured to, in a case of extending a last-level page table to lower-level page tables in the EPT, modify, in the first RMAP table, address information of the last-level page table entry corresponding to each GPA in the last-level page table to address information of a corresponding lower-level page table;

a collapsing module, configured to, in a case of collapsing last-level page tables with contiguous addresses to an upper -level page table in the EPT, modify, in the first RMAP table, address information of the last-level page table entry corresponding to each GPA in the last-level page tables to address information of an upper-level page table entry.

Optionally, the addition module is specifically configured to: determine a first structure corresponding to the second GPA from among the base structures according to address information of the second GPA and the address spaces corresponding to the base structures included in the first RMAP table, the first structure corresponding to a set page table level; if it is determined that address information of the last-level page table entry corresponding to the second GPA is not recorded through the first structure, record the address information of the last-level page table entry corresponding to the second GPA through the first structure according to a relationship between a level of a last-level page table corresponding to the second GPA and the set page table level.

Optionally, the deletion module is specifically configured to: determine a first structure corresponding to the third GPA from the base structures according to address information of the third GPA and the address spaces corresponding to the base structures included in the first RMAP table, the first structure corresponding to a set page table level; if a level of a last-level page table corresponding to the third GPA matches a page table level indicated by flag information of the first structure, delete the address information of the last-level page table entry corresponding to the third GPA through the first structure according to a relationship between the level of the last-level page table corresponding to the third GPA and the set page table level.

Optionally, the extension module is specifically configured to: clear, in the EPT, address information recorded in each page table entry of the last-level page table, and clear, in the first RMAP table, address information of the last-level page table entry corresponding to each GPA in the last-level page table; create lower-level page tables corresponding to the last-level page

table, record address information of the lower-level page tables in page table entries of the last-level page table, and record base addresses of physical pages corresponding to respective HPAs in page table entries of the lower-level page tables; in the first RMAP table, record address information of a lower-level page table entry corresponding to each GPA in the last-level page table through the first structure according to a relationship between a level of the lower-level page tables and the set page table level.

Optionally, the collapsing module is specifically configured to: clear, in the EPT, address information recorded in each page table entry of the last-level page tables with contiguous addresses , and clear, in the first RMAP table, the address information of the last-level page table entry corresponding to each GPA in the last-level page tables; create an upper-level page table corresponding to the last-level page tables, and record address information of physical pages corresponding to respective HPAs respectively in page table entries of the upper-level page table; in the first RMAP table, record address information of an upper-level page table entry corresponding to each GPA in the last-level page table through the first structure according to a relationship between a level of the upper-level page table and the set page table level.

In an optional embodiment, the execution entity of the apparatus is a hypervisor in a host operating system (OS); or, the execution entity of the apparatus is a virtualization layer between the host OS and host hardware resources, the virtualization layer being configured to switch the running host OS from root mode to non-root mode.

An embodiment of the present disclosure further provides a fast mapping table construction apparatus. As shown in FIG. 5, the apparatus includes:

a division module 51, configured to divide a GPA address space corresponding to the EPT into at least one address space; a creation module 52, configured to create a base structure for each of the at least one address space respectively, the base structure corresponding to a set page table level, the EPT including multi-level page tables corresponding to each of at least one GPA;

a determination module 53, configured to, for any GPA in the EPT, determine, from among base structures, a first structure corresponding to the any GPA according to address information of the any GPA and address spaces corresponding to the base structures;

a recording module 54, configured to, according to a relationship between a level of a last-level page table corresponding to the any GPA and a set page table level, record, through the first structure, address information of a last-level page table entry corresponding to any GPA, to construct a first RMAP table corresponding to the EPT, where the first RMAP table stores

address information of a last-level page table entry corresponding to each of the at least one GPA.

Detailed implementation manners and beneficial effects of steps in the apparatus shown in FIG. 4 and FIG. 5 provided by the embodiments of the present disclosure have been described in detail in the foregoing embodiments, and will not be elaborated here.

FIG. 6 is a schematic structural diagram of a physical machine provided by an exemplary embodiment of the present disclosure, where the physical machine is also referred to as an address translation device. As shown in FIG. 6, the device includes: a memory 64 and a processor 65.

The memory 64 is configured to store a computer program, and may be configured to store other various data to support operations on the address translation device. Examples of such data include instructions for any application programs or methods operating on the address translation device.

The processor 65, is coupled to the memory 64, and is configured to execute the computer program in the memory 64 to: pre-construct a first RMAP table corresponding to the EPT, where the EPT includes multi-level page tables corresponding to each of at least one GPA; the first RMAP table stores address information of a last-level page table entry corresponding to each of the at least one GPA; in response to a first address translation trigger event, determine a first GPA for which address translation is required, and query, in the first RMAP table, address information of a last-level page table entry corresponding to the first GPA; if the address information of the last-level page table entry corresponding to the first GPA is found by query, generate, according to the address information of the last-level page table entry corresponding to the first GPA and an address offset of the first GPA, an HPA corresponding to the first GPA.

In an optional embodiment, the processor 65 is further configured to: pre-construct a second RMAP table corresponding to the EPT, where the second RMAP table stores address translation information from at least one HPA to a corresponding GPA; in response to a second address translation trigger event, determine a first HPA for which address translation is required, and query, in the second RMAP table, address translation information from the first HPA to a corresponding GPA; generate, according to the first HPA, the address translation information from the first HPA to the corresponding GPA, and an address offset of the first HPA, a GPA corresponding to the first HPA.

In an optional embodiment, when pre-constructing the first RMAP table corresponding to the EPT, the processor 65 is specifically configured to: divide a GPA address space

corresponding to the EPT into at least one address space, and create a base structure for each of the at least one address space, where the base structure corresponds to a set page table level; for any GPA in the EPT, determine, according to address information of the any GPA and address spaces corresponding to base structures, a first structure corresponding to the any GPA from among the base structures; according to a relationship between a level of a last-level page table corresponding to the any GPA and the set page table level, record, through the first structure, address information of a last-level page table entry corresponding to the any GPA.

Optionally, when recording, according to the relationship between the level of the last-level page table corresponding to the any GPA and the set page table level, through the first structure, the address information of the last-level page table entry corresponding to the any GPA, the processor 65 is specifically configured to: if the level of the last-level page table corresponding to the any GPA is the same as the set page table level, add the address information of the last-level page table entry corresponding to the any GPA to an information field in the first structure, and set flag information in the first structure to a first value; if the level of the last-level page table corresponding to the any GPA is not the same as the set page table level, create a second structure adapted to the level of the last-level page table corresponding to the any GPA, add the address information of the last-level page table entry corresponding to the any GPA to an information field in the second structure, add address information of the second structure to the information field in the first structure, and set the flag information in the first structure to a second value.

Further optionally, when creating the second structure adapted to the level of the last-level page table corresponding to the any GPA, the processor 65 is specifically configured to: in a case that the level of the last-level page table corresponding to the any GPA is an upper level of the set page table level, determine whether the second structure already exists; if not, create a new structure adapted to the level of the last-level page table corresponding to the any GPA as the second structure; in a case that the level of the last-level page table corresponding to the any GPA is a lower level of the set page table level, create a new structure adapted to the level of the last-level page table corresponding to the any GPA as the second structure.

In an optional embodiment, when pre-constructing the second RMAP table corresponding to the EPT, the processor 65 is specifically configured to: divide an HPA address space corresponding to the EPT into at least one address space, and create a base structure for each of the at least one address space, where the base structure corresponds to a set page granularity; for any HPA corresponding to a GPA in the EPT, determine, according to address information of the any HPA and address spaces corresponding to base structures, a third structure

corresponding to the any HPA from among the base structures; according to a relationship between a page granularity corresponding to the any HPA and the set page granularity, record, through the third structure, address translation information from the any HPA to a corresponding GPA.

Optionally, when recording, according to a relationship between a page table level corresponding to the any HPA and the set page table level, through the third structure, the address translation information from the any HPA to the corresponding GPA, the processor 65 is specifically configured to: if the page granularity corresponding to the any HPA is the same as the set page granularity, record, through an information field in the third structure, the address translation information from the any HPA to the corresponding GPA, and set first flag information in the third structure to a third value; if the page granularity corresponding to the any HPA is not the same as the set page granularity, create a fourth structure adapted to the page granularity of the any HPA, record, through an information field in the fourth structure, the address translation information from the any HPA to the corresponding GPA, add address information of the fourth structure to the information field in the third structure, and set the first flag information in the third structure to a fourth value.

Further optionally, when recording, through the information field in the third structure, the address translation information from the any HPA to the corresponding GPA, the processor 65 is specifically configured to: in a case that the any HPA corresponds to a plurality of GPAs, create a fifth structure, add address information of the fifth structure to the information field in the third structure, add the address translation information from the any HPA to the plurality of GPAs respectively to the fifth structure, and add, in the third structure, a flag bit indicating a one-to-many relationship; in a case that the any HPA corresponds to one GPA, directly add the address translation information from the any HPA to the corresponding GPA to the information field in the third structure;

Accordingly, when recording, through the information field in the fourth structure, the address translation information from the any HPA to the corresponding GPA, the processor 65 is specifically configured to: in a case that the any HPA corresponds to a plurality of GPAs, create a sixth structure, add address information of the sixth structure to the information field in the fourth structure, add the address translation information from the any HPA to the plurality of GPAs respectively to the sixth structure, and add, in the fourth structure, a flag bit indicating a one-to-many relationship; in a case that the any HPA corresponds to one GPA, directly add the address translation information from the any HPA to the corresponding GPA to the information field in the fourth structure.

In an optional embodiment, when querying, in the first RMAP table, the address information of the last-level page table entry corresponding to the first GPA, the processor 65 is specifically configured to: determine, according to the address information of the first GPA and the address spaces corresponding to the base structures included in the first RMAP table, a first structure corresponding to the first GPA from among the base structures, where the base structures correspond to the set page table level; obtain, through the first structure corresponding to the first GPA, the address information of the last-level page table entry corresponding to the first GPA.

Optionally, when obtaining, through the first structure corresponding to the first GPA, the address information of the last-level page table entry corresponding to the first GPA, the processor 65 is specifically configured to: obtain flag information from the first structure; if the flag information is a first value, obtain, from an information field in the first structure, the address information of the last-level page table entry corresponding to the first GPA; if the flag information is a second value, obtain, from the information field in the first structure, address information of a second structure, and obtain, according to the address information of the second structure, from an information field of the second structure, the address information of the last-level page table entry corresponding to the first GPA.

In an optional embodiment, the processor 65 is further configured to perform at least one of the following operations:

in a case of newly adding a second GPA in the EPT, adding, in the first RMAP table, address information of a last-level page table entry corresponding to the second GPA;

in a case of deleting a third GPA from the EPT, deleting, in the first RMAP table, address information of a last-level page table entry corresponding to the third GPA;

in a case of modifying first address information of a last-level page table corresponding to a fourth GPA to second address information in the EPT, modifying, in the first RMAP table, the first address information of the last-level page table corresponding to the fourth GPA to the second address information;

in a case of extending the last-level page table to lower-level page tables in the EPT, modifying, in the first RMAP table, address information of the last-level page table entry corresponding to each GPA in the last-level page table to address information of a corresponding lower-level page table;

in a case of collapsing last-level page tables with consecutive addresses to an upper-level page table in the EPT, modifying, in the first RMAP table, address information of the last-level

page table entry corresponding to each GPA in the last-level page tables to address information of an upper-level page table entry.

In an optional embodiment, when adding, in the first RMAP table, the address information of the last-level page table entry corresponding to the second GPA, the processor 65 is specifically configured to: determine, according to the address information of the second GPA and the address spaces corresponding to the base structures included in the first RMAP table, from among the base structures, a first structure corresponding to the second GPA, where the first structure corresponds to the set page table level; if it is determined that the address information of the last-level page table entry corresponding to the second GPA is recorded through the first structure, record, according to a relationship between a level of the last-level page table corresponding to the second GPA and the set page table level, through the first structure, the address information of the last-level page table entry corresponding to the second GPA.

In an optional embodiment, when deleting, in the first RMAP table, the address information of the last-level page table entry corresponding to the third GPA, the processor 65 is specifically configured to: determine, according to the address information of the third GPA and the address spaces corresponding to the base structures included in the first RMAP table, from among the base structures, a first structure corresponding to the third GPA, where the first structure corresponds to the set page table level; if a level of the last-level page table corresponding to the third GPA matches a page table level indicated by flag information of the first structure, delete, according to a relationship between the level of the last-level page table corresponding to the third GPA and the set page table level, through the first structure, the address information of the last-level page table entry corresponding to the third GPA.

In an optional embodiment, when, in the case of extending the last-level page table to the lower-level page tables in the EPT, modifying, in the first RMAP table, the address information of the last-level page table entry corresponding to each GPA in the last-level page table to the address information of the corresponding lower-level page table, the processor 65 is specifically configured to: clear, in the EPT, the address information recorded in each page table entry of the last-level page table, and clear, in the first RMAP table, the address information of the last-level page table entries corresponding to each GPA in the last-level page table; create lower-level page tables corresponding to the last-level page table, record address information of the lower-level page tables in the page table entries of the last-level page table, and record, in page table entries of the lower-level page tables, base addresses of physical pages corresponding to respective HPAs; in the first RMAP table, record, according to a relationship

between a level of the lower-level page tables and the set page table level, through the first structure, address information of a lower-level page table entry corresponding to each GPA in the last-level page table.

In an optional embodiment, when, in the case of collapsing last-level page tables with consecutive addresses to an upper-level page table in the EPT, modifying, in the first RMAP table, the address information of the last-level page table entry corresponding to each GPA in the last-level page table to the address information of the upper-level page table entry, the processor 65 is specifically configured to: clear, in the EPT, address information recorded in each page table entry in the last-level page table with contiguous addresses, and clear, in the first RMAP table, the address information of the last-level page table entry corresponding to each GPA in the last-level page tables; create an upper-level page table corresponding to the last-level page tables, and record address information of physical pages corresponding to respective HPAs respectively in page table entries of the upper-level page table; in the first RMAP table, record, according to a relationship between a level of the upper-level page table and the set page table level, through the first structure, address information of an upper-level page table entry corresponding to each GPA in the last-level page table.

In an optional embodiment, the processor 65 performs the above operations through a hypervisor in a host operating system (OS); or, the processor 65 performs the above operations through a virtualization layer between the host OS and host hardware resources, where the virtualization layer is configured to switch a running host OS from root mode to non-root mode.

Detailed implementation manners and beneficial effects of steps in the device shown in FIG. 6 provided by the embodiments of the present disclosure have been described in detail in the foregoing embodiments, and will not be elaborated here.

Further, as shown in FIG. 6, the address translation device further includes other components, such as a communication component 66, a display 67, a power supply component 68, an audio component 69. FIG. 6 only schematically shows some components, and does not imply that the address translation device only includes the components shown in FIG. 6. In addition, the components within the dashed-line box in FIG. 6 are optional components rather than required components, depending on the product form of the address translation device. The address translation device of the present embodiment may be implemented as a terminal device such as a desktop computer, a laptop computer, a smartphone, or an IOT device, or may be implemented as a server-side device such as a conventional server, a cloud server, or a server array. If the address translation device of the present embodiment is implemented as a terminal device such as a desktop computer, a laptop computer, or a smartphone, it may include the

components within the dashed-line box in FIG. 6; if the address translation device of the present embodiment is implemented as a server-side device such as a conventional server, a cloud server, or a server array, it may not include the components within the dashed-line box in FIG. 6.

An embodiment of the present disclosure further provides a fast mapping construction device, where an implementation structure of the fast mapping construction device is the same as or similar to the implementation structure of the fast mapping construction device shown in FIG. 6, and reference can be made to the structure of the fast mapping construction device shown in FIG. 6. The difference between the fast mapping construction device provided by the present embodiment and the fast mapping construction device in the embodiment shown in FIG. 6 mainly lies in: functions implemented by the processors executing the computer programs stored in the memory are different. For the address translation device provided by the present embodiment, the processor executes the computer program stored in the memory, and may be configured to: divide a GPA address space corresponding to a memory page table EPT into at least one address space, create a base structure for each of the at least one address space, where the base structure corresponds to a set page table level, and the EPT includes multi-level page tables corresponding to each of at least one GPA; for any GPA in the EPT, determine, according to address information of the any GPA and address spaces corresponding to base structures, a first structure corresponding to the any GPA from among the base structures; according to a relationship between a level of a last-level page table corresponding to the any GPA and the set page table level, record, through the first structure, address information of a last-level page table entry corresponding to the any GPA, to construct a first RMAP table corresponding to the EPT, where the first RMAP table stores address information of a last-level page table entry corresponding to each of the at least one GPA.

Detailed implementation manners and beneficial effects of the fast mapping construction device provided by the embodiments of the present disclosure have been described in detail in the foregoing embodiments, and will not be elaborated here.

Accordingly, an embodiment of the present disclosure further provides a computer-readable storage medium storing a computer program, where when the computer program is executed, the steps in the method embodiments shown in FIG. 4 and FIG. 5 can be implemented.

Embodiments of the present disclosure further provide a computer program product, including a computer program/instructions, where when the computer program/instructions are

executed by a processor, the processor is caused to implement the steps in the methods shown in FIG. 4 and FIG. 5.

The above-described memory can be implemented by any type of volatile or non-volatile storage device or a combination thereof, such as a static random-access memory (Static Random-Access Memory, SRAM), an electrically erasable programmable read only memory (Electrically Erasable Programmable Read Only Memory, EEPROM), an Erasable Programmable Read Only Memory (EPROM), a programmable read-only memory (Programmable Read-Only Memory, PROM), a read-only memory (Read-Only Memory, ROM), a magnetic memory, a flash memory, a magnetic disk, or an optical disk.

The above-described communication component is configured to facilitate wired or wireless communication between the device in which the communication component is located and other devices. The device in which the communication component is located may access a wireless network based on a communication standard, such as WiFi, a 2G, 3G, 4G/LTE, or 5G mobile communication network, or a combination thereof. In an exemplary embodiment, the communication component receives a broadcast signal or broadcast-related information from an external broadcast management system via a broadcast channel. In an exemplary embodiment, the communication component further includes a near field communication (Near Field Communication, NFC) module to facilitate short-range communication. For example, the NFC module may be implemented based on radio frequency identification (Radio Frequency Identification, RFID) technology, infrared data association (Infrared Data Association, IrDA) technology, ultra wide band (Ultra Wide Band, UWB) technology, Bluetooth (BlueTooth, BT) technology, and other technologies.

The above-described display includes a screen, and the screen may include a liquid crystal display (Liquid Crystal Display, LCD) and a touch panel (TouchPanel, TP). If the screen includes a touch panel, the screen may be implemented as a touch screen to receive input signals from a user. The touch panel includes one or more touch sensors to sense touches, swipes, and gestures on the touch panel. The touch sensors may not only sense boundaries of a touch or swipe action, but also detect duration and pressure associated with the touch or swipe operation.

The above-described power supply component provides power to various components of the device in which the power supply component is located. The power supply component may include a power management system, one or more power supplies, and other components associated with generating, managing, and distributing power for the device in which the power supply component is located.

The above-described audio component may be configured to output and/or input audio signals. For example, the audio component includes a microphone (Microphone, MIC), and when the device in which the audio component is located is in an operation mode, such as a call mode, a recording mode, and a voice recognition mode, the microphone is configured to receive external audio signals. The received audio signals may be further stored in a memory or transmitted via the communication component. In some embodiments, the audio component further includes a speaker for outputting audio signals.

Those skilled in the art should understand that the embodiments of the present disclosure may be provided as a method, a system, or a computer program product. Accordingly, the present disclosure may take the form of an entirely hardware embodiment, an entirely software embodiment, or an embodiment combining software and hardware aspects. Furthermore, the present disclosure may take the form of a computer program product implemented on one or more computer-readable storage media having computer-usable program code embodied therein (including but not limited to a disk storage, a compact disc read-only memory (Compact Disc Read-Only Memory, CD-ROM), an optical storage, and the like).

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

These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means that implement the functions specified in one or more flows of the flowcharts and/or one or more blocks of the block diagrams.

These computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer-implemented

process, such that the instructions executed on the computer or other programmable apparatus provide steps for implementing the functions specified in one or more flows of the flowcharts and/or one or more blocks of the block diagrams.

In a typical configuration, a computing device includes one or more processors (Central Processing Unit, CPU), input/output interfaces, network interfaces, and memory.

The memory may include non-persistent storage in computer-readable media, a random access memory (Random Access Memory, RAM), and/or non-volatile memory, such as a read-only memory (ROM) or a flash memory (flash RAM). The memory is an example of a computer-readable medium.

Computer-readable media include persistent and non-persistent, removable and non-removable media, and information storage may be implemented by any method or technology. The information may be computer-readable instructions, data structures, modules of a program, or other data. Examples of computer storage media include, but are not limited to, a phase-change random access memory (Phase-change Random Access Memory, PRAM), a static random-access memory (SRAM), a dynamic random access memory (Dynamic Random Access Memory, DRAM), other types of random access memory (RAM), a read-only memory (ROM), an electrically erasable programmable read only memory (EEPROM), a flash memory or other memory technology, a compact disc read-only memory (CD-ROM), a digital video disc (Digital Video Disc, DVD) or other optical storage, a magnetic cassette, a magnetic tape, a magnetic disk storage or other magnetic storage device, or any other non-transmission medium that is used to store information accessible by a computing device. As defined herein, computer-readable media do not include transitory computer-readable media (transitory media), such as modulated data signals and carrier waves.

It should also be noted that the terms "including", "comprising", or any other variation thereof are intended to cover a non-exclusive inclusion, such that a process, method, article, or apparatus that includes a list of elements includes not only those elements but also other elements not expressly listed, or elements inherent to such process, method, article, or apparatus. Without further limitation, an element defined by the phrase "including a ..." does not preclude the presence of additional identical elements in the process, method, article, or apparatus that includes the element.

The foregoing is merely embodiments of the present disclosure and is not intended to limit the present disclosure. For those skilled in the art, the present disclosure may have various modifications and changes. Any modification, equivalent replacement, improvement, and the

like made within the spirit and principle of the present disclosure should be included within the scope of the claims of the present disclosure.

CLAIMS

1. An address translation method, comprising:

pre-constructing a first rapid mapping table corresponding to a memory page table, the memory page table comprising multi-level page tables corresponding to each of at least one guest physical address; the first rapid mapping table storing address information of a last-level page table entry corresponding to each of the at least one guest physical address;

in response to a first address translation trigger event, determining a first guest physical address for which address translation is required, and querying, in the first rapid mapping table, address information of a last-level page table entry corresponding to the first guest physical address; and

if the address information of the last-level page table entry corresponding to the first guest physical address is found by the query, generating a host physical address corresponding to the first guest physical address based on the address information of the last-level page table entry corresponding to the first guest physical address and an address offset of the first guest physical address.

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

pre-constructing a second rapid mapping table corresponding to the memory page table, the second rapid mapping table storing address translation information from at least one host physical address to a corresponding guest physical address;

in response to a second address translation trigger event, determining a first host physical address for which address translation is required, and querying, in the second rapid mapping table, address translation information from the first host physical address to a corresponding guest physical address; and

generating a guest physical address corresponding to the first host physical address based on the first host physical address, the address translation information from the first host physical address to the corresponding guest physical address, and an address offset of the first host physical address.

3. The method according to claim 1 or 2, wherein pre-constructing the first rapid mapping table corresponding to the memory page table comprises:

dividing an address space of guest physical addresses corresponding to the memory page table into at least one address space, and creating a base structure for each of the at least one address space, the base structure corresponding to a set page table level;

for any guest physical address in the memory page table, determining, based on address information of the any guest physical address and address spaces corresponding to base structures, a first structure corresponding to the any guest physical address from among the base structures; and

based on a relationship between a level of a last-level page table corresponding to the any guest physical address and the set page table level, recording, through the first structure, address information of a last-level page table entry corresponding to the any guest physical address.

4. The method according to claim 3, wherein based on the relationship between the level of the last-level page table corresponding to the any guest physical address and the set page table level, recording, through the first structure, the address information of the last-level page table entry corresponding to the any guest physical address comprises:

if the level of the last-level page table corresponding to the any guest physical address is the same as the set page table level, adding the address information of the last-level page table entry corresponding to the any guest physical address to an information field in the first structure, and setting flag information in the first structure to a first value; and

if the level of the last-level page table corresponding to the any guest physical address is not the same as the set page table level, creating a second structure adapted to the level of the last-level page table corresponding to the any guest physical address, adding the address information of the last-level page table entry corresponding to the any guest physical address to an information field in the second structure, adding address information of the second structure to the information field in the first structure, and setting the flag information in the first structure to a second value.

5. The method according to claim 4, wherein creating the second structure adapted to the level of the last-level page table corresponding to the any guest physical address comprises:

in a case that the level of the last-level page table corresponding to the any guest physical address is an upper level of the set page table level, determining whether the second structure already exists; if not, creating a new structure adapted to the level of the last-level page table corresponding to the any guest physical address, as the second structure;

in a case that the level of the last-level page table corresponding to the any guest physical address is a lower level of the set page table level, creating a new structure adapted to the level of the last-level page table corresponding to the any guest physical address, as the second structure.

6. The method according to claim 2, wherein pre-constructing the second rapid mapping table corresponding to the memory page table comprises:

dividing an address space of host physical addresses corresponding to the memory page table into at least one address space, and creating a base structure for each of the at least one address space, the base structure corresponding to a set page granularity;

for any host physical address corresponding to a guest physical address in the memory page table, determining, based on address information of the any host physical address and address spaces corresponding to base structures, a third structure corresponding to the any host physical address from among the base structures; and

based on a relationship between a page granularity corresponding to the any host physical address and the set page granularity, recording, through the third structure, address translation information from the any host physical address to a corresponding guest physical address.

7. The method according to claim 6, wherein based on the relationship between the page table level corresponding to the any host physical address and the set page table level, recording, through the third structure, the address translation information from the any host physical address to the corresponding guest physical address comprises:

if the page granularity corresponding to the any host physical address is the same as the set page granularity, recording the address translation information from the any host physical address to the corresponding guest physical address through an information field in the third structure, and setting first flag information in the third structure to a third value; and

if the page granularity corresponding to the any host physical address is not the same as the set page granularity, creating a fourth structure adapted to the page granularity corresponding to the any host physical address, recording the address translation information from the any host physical address to the corresponding guest physical address through an information field in the fourth structure, adding address information of the fourth structure to the information field in the third structure, and setting the first flag information in the third structure to a fourth value.

8. The method according to claim 7, wherein recording the address translation information of the any host physical address to the corresponding guest physical address through the information field in the third structure comprises:

in a case that the any host physical address corresponds to a plurality of guest physical addresses, creating a fifth structure, adding address information of the fifth structure to the information field in the third structure, adding address translation information from the any

host physical address to the plurality of guest physical addresses respectively to the fifth structure, and adding, in the third structure, a flag bit indicating a one-to-many relationship;

in a case that the any host physical address corresponds to one guest physical address, directly adding the address translation information from the any host physical address to the corresponding guest physical address to the information field in the third structure;

correspondingly, recording the address translation information from the any host physical address to the corresponding guest physical address through the information field in the fourth structure comprises:

in a case that the any host physical address corresponds to a plurality of guest physical addresses, creating a sixth structure, adding address information of the sixth structure to the information field in the fourth structure, adding the address translation information from the any host physical address to the plurality of guest physical addresses respectively to the sixth structure, and adding, in the fourth structure, a flag bit indicating a one-to-many relationship;

in a case that the any host physical address corresponds to one guest physical address, directly adding the address translation information from the any host physical address to the corresponding guest physical address to the information field in the fourth structure.

9. The method according to any one of claims 1-8, wherein querying, in the first rapid mapping table, the address information of the last-level page table entry corresponding to the first guest physical address comprises:

determining, based on address information of the first guest physical address and address spaces corresponding to base structures included in the first rapid mapping table, a first structure corresponding to the first guest physical address from among the base structures, the base structures corresponding to a set page table level; and obtaining, through the first structure corresponding to the first guest physical address, the address information of the last-level page table entry corresponding to the first guest physical address.

10. The method according to claim 9, wherein obtaining, through the first structure corresponding to the first guest physical address, the address information of the last-level page table entry corresponding to the first guest physical address comprises:

obtaining flag information from the first structure;

if the flag information is a first value, obtaining the address information of the last-level page table entry corresponding to the first guest physical address from an information field in the first structure; and

if the flag information is a second value, obtaining address information of a second structure from the information field in the first structure, and obtaining, based on the address

information of the second structure, the address information of the last-level page table entry corresponding to the first guest physical address from an information field in the second structure.

11. The method according to any one of claims 1-2 and 4-10, further comprising at least one of the following operations:

in a case of newly adding a second guest physical address in the memory page table, adding, in the first rapid mapping table, address information of a last-level page table entry corresponding to the second guest physical address;

in a case of deleting a third guest physical address from the memory page table, deleting, in the first rapid mapping table, address information of a last-level page table entry corresponding to the third guest physical address;

in a case of modifying first address information of a last-level page table corresponding to a fourth guest physical address in the memory page table to second address information, modifying, in the first rapid mapping table, the first address information of the last-level page table corresponding to the fourth guest physical address to the second address information;

in a case of extending a last-level page table to lower-level page tables in the memory page table, modifying, in the first rapid mapping table, address information of the last-level page table entry corresponding to each guest physical address in the last-level page table to address information of the lower-level page table;

in a case of collapsing last-level page tables with contiguous addresses to an upper-level page table in the memory page table, clearing, in the first rapid mapping table, the address information of the last-level page table entry, and adding address information of an upper-level page table entry corresponding to each guest physical address in the last-level page table.

12. The method according to claim 11, wherein adding, in the first rapid mapping table, the address information of the last-level page table entry corresponding to the second guest physical address comprises:

determining, based on address information of the second guest physical address and address spaces corresponding to base structures included in the first rapid mapping table, a first structure corresponding to the second guest physical address from among the base structures, the first structure corresponding to a set page table level; and

if it is determined that the address information of the last-level page table entry corresponding to the second guest physical address is not recorded through the first structure, recording, based on a relationship between a level of a last-level page table corresponding to the second guest physical address and the set page table level, the address information of the

last-level page table entry corresponding to the second guest physical address through the first structure.

13. The method according to claim 11, wherein deleting, in the first rapid mapping table, the address information of the last-level page table entry corresponding to the third guest physical address comprises:

determining, based on address information of the third guest physical address and address spaces corresponding to base structures included in the first rapid mapping table, a first structure corresponding to the third guest physical address from among the base structures, the first structure corresponding to a set page table level; and

if a level of a last-level page table corresponding to the third guest physical address matches a page table level indicated by flag information of the first structure, deleting, based on a relationship between the level of the last-level page table corresponding to the third guest physical address and the set page table level, the address information of the last-level page table entry corresponding to the third guest physical address through the first structure.

14. The method according to claim 11, wherein in the case of extending the last-level page table to the lower-level page tables in the memory page table, modifying, in the first rapid mapping table, the address information of the last-level page table entry corresponding to each guest physical address in the last-level page table to the address information of the lower-level page table comprises:

clearing, in the memory page table, address information recorded in each page table entry in the last-level page table, and clearing, in the first rapid mapping table, the address information of the last-level page table entry corresponding to each guest physical address in the last-level page table;

creating lower-level page tables corresponding to the last-level page table, recording address information of the lower-level page tables in page table entries of the last-level page table, and recording base addresses of physical pages corresponding to respective host physical addresses in page table entries of the lower-level page tables; and

in the first rapid mapping table, based on a relationship between a level of the lower-level page tables and the set page table level, recording, through the first structure, address information of a lower-level page table entry corresponding to each guest physical address in the last-level page table.

15. The method according to claim 11, wherein in the case of collapsing the last-level page tables with contiguous addresses to the upper-level page table in the memory page table, clearing, in the first rapid mapping table, the address information of the last-level page table

entry, and adding the address information of the upper-level page table entry corresponding to each guest physical address in the last-level page table comprises:

clearing, in the memory page table, address information recorded in each page table entry in the last-level page tables with contiguous addresses, and clearing, in the first rapid mapping table, the address information of the last-level page table entry corresponding to each guest physical address in the last-level page tables;

creating, in the memory page table, an upper-level page table corresponding to the last-level page tables, and recording, in page table entries of the upper-level page table according to a page granularity corresponding to the upper-level page table entry, base addresses of physical pages corresponding to respective host physical addresses respectively; and

in the first rapid mapping table, clearing the address information of the last-level page table entry, and based on a relationship between a level of the upper-level page table and the set page table level, recording, through the first structure, address information of the upper-level page table entry corresponding to each guest physical address in the last-level page table.

16. The method according to any one of claims 1-2, 4-10, and 12-15, wherein an execution body of the method is a hypervisor in a host operating system OS; or an execution body of the method is a virtualization layer between the host OS and host hardware resources, the virtualization layer being configured to switch a running host OS from root mode to non-root mode.

17. A rapid mapping table construction method, comprising:

dividing an address space of guest physical addresses corresponding to a memory page table into at least one address space, and creating a base structure for each of the at least one address space, the base structure corresponding to a set page table level, the memory page table comprising multi-level page tables corresponding to each of at least one guest physical address;

for any guest physical address in the memory page table, determining, based on address information of the any guest physical address and address spaces corresponding to base structure, a first structure corresponding to the any guest physical address from among the base structures; and

recording, based on a relationship between a level of a last-level page table corresponding to the any guest physical address and the set page table level, through the first structure, address information of a last-level page table entry corresponding to the any guest physical address, to construct a first rapid mapping table corresponding to the memory page table, the first rapid mapping table storing address information of a last-level page table entry corresponding to each of the at least one guest physical address.

18. A physical machine, comprising: a memory and a processor; the memory configured to store a computer program; the processor, coupled to the memory, configured to execute the computer program to implement the steps in the method according to any one of claims 1-16 and claim 17.

19. A computer-readable storage medium storing a computer program, wherein when the computer program is executed by a processor, the processor is caused to implement the steps in the method according to any one of claims 1-16 and claim 17.

20. A computer program product, comprising a computer program/instructions, wherein when the computer program/instructions are executed by a processor, the processor is caused to implement the steps in the method according to any one of claims 1-16 and claim 17.