

(51) International Patent Classification:
G06F 12/08 (2006.01)(21) International Application Number:
PCT/CN2010/071456(22) International Filing Date:
31 March 2010 (31.03.2010)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant (for all designated States except US): **INTEL CORPORATION** [US/US]; 2200 Mission College Blvd., Santa Clara, California 95052 (US).

(72) Inventor; and

(75) Inventor/Applicant (for US only): **TIAN, Kun** [CN/CN]; No.880 Zi Xing Road, Shanghai Zizhu Science Park, Shanghai 200241 (CN).(74) Agent: **NTD PATENT & TRADEMARK AGENCY LIMITED BEIJING OFFICE**; 10th Floor, Block A, Investment Plaza, 27 Jinrongdajie, Beijing 100033 (CN).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM,

AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— with international search report (Art. 21(3))

(54) Title: HOT-SWAPPING ACTIVE MEMORY FOR VIRTUAL MACHINES WITH DIRECTED I/O

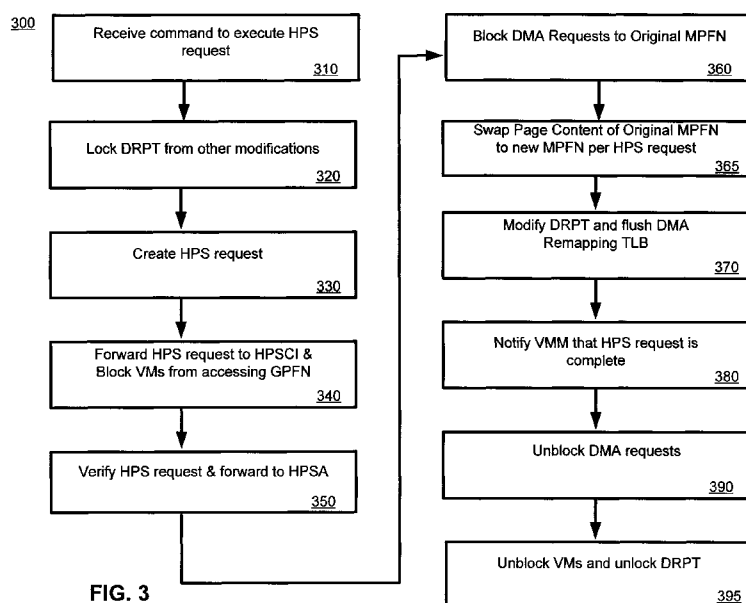


FIG. 3

(57) Abstract: Embodiments of the invention describe a DMA Remapping unit (DRU) to receive, from a virtual machine monitor (VMM), a hot-page swap (HPS) request, the HPS request to include a virtual address, in use by at least one virtual machine (VM), mapped to a first memory page location, and a second memory page location. The DRU further blocks DMA requests to addresses of memory being remapped until the HPS request is fulfilled, copies the content of the first memory page location to the second memory page location, and remaps the virtual address from the first memory page location to the second memory page location.

HOT-SWAPPING ACTIVE MEMORY FOR VIRTUAL MACHINES WITH DIRECTED I/O

FIELD

[0001] Embodiments of the invention generally pertain to system memory management, and more particularly to management of system memory assigned to virtual machines.

BACKGROUND

[0002] Systems including virtual machines (VMs) allow the sharing of an underlying physical machine and its resources. The software layer providing virtualization to the VMs is referred to as a virtual machine monitor (VMM) or hypervisor. A VMM acts as a host to the VMs. The VMM presents system software executing on the VMs (e.g., operating system software and application software) with an abstraction of the physical machine. The VMM is further able to retain selective control of processor resources, physical memory, interrupt management and data I/O.

[0003] I/O virtualization describes the capability for a single physical I/O unit to be shared by more than one VM. Prior art solutions accomplish I/O virtualization via a software module. Thus all accesses to VM memory pages are initiated from the CPU, and thereby incurring significant processing overhead. Furthermore, this type of virtualization does not allow devices assigned to VMs to execute memory operations directly via Direct Memory Access (DMA).

[0004] Other prior art solutions accomplish I/O virtualization via hardware logic, also known as directed I/O. In these solutions, a device's DMA requests are restricted to memory domains and memory regions assigned (by the VMM) to the VM that owns the device. One of these hardware prior art solutions may be described as DMA remapping hardware — i.e., a hardware unit positioned between the DMA capable peripheral devices and the system's physical memory. Unmodified drivers may run seamlessly in a VM with full privilege on I/O resources assigned to that VM; however, because DMA protection is enforced at the chipset level, access to physical memory outside of the VM-owned physical pages is prevented for I/O devices assigned to that VM.

[0005] Some advanced computer server memory features require the ability to “hot-remove” or “hot-swap” memory (e.g., removing DIMMs while they are in use). Hot swapping memory involves moving the content of (at least) one in-use page to another page, without breaking ongoing executions.

[0006] Hot-swapping memory is possible in the prior art if I/O virtualization is done via a software module. In this scenario, a VMM freezes the respective VM to prevent race access when page content is being swapped. After new mapping is established, the VM is thawed and subsequent memory accesses are redirected to newly mapped page.

[0007] However, there is no ability to hot swap pages of VM with I/O virtualization that allows directed I/O. This is due, in part, to the fact that a VMM can freeze VM execution, but has no control over devices assigned to said VM. These assigned devices can issue DMA requests to VM pages at any time, even when hot swapping of that page is in progress. This race condition is dangerous in that it may cause inconsistency between old and new pages.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The following description includes discussion of figures having illustrations given by way of example of implementations of embodiments of the invention. The drawings should be understood by way of example, and not by way of limitation. As used herein, references to one or more "embodiments" are to be understood as describing a particular feature, structure, or characteristic included in at least one implementation of the invention. Thus, phrases such as "in one embodiment" or "in an alternate embodiment" appearing herein describe various embodiments and implementations of the invention, and do not necessarily all refer to the same embodiment. However, they are also not necessarily mutually exclusive.

[0009] FIG. 1 is a block diagram of a system utilizing an embodiment of the invention.

[0010] FIG. 2 is a block diagram of an embodiment of a DMA Remapping Unit.

[0011] FIG. 3 is a flow diagram of processing a hot page swap request according to an embodiment of the invention.

[0012] FIG. 4 illustrates an example process that minimizes the number of DMA requests that need to be blocked during a hot page swap operation.

[0013] Descriptions of certain details and implementations follow, including a description of the figures, which may depict some or all of the embodiments described below, as well as discussing other potential embodiments or implementations of the inventive concepts presented herein. An overview of embodiments of the invention is provided below, followed by a more detailed description with reference to the drawings.

DETAILED DESCRIPTION

[0014] Embodiments of the present invention relate to enhanced DMA functionality for VMs with directed I/O. Embodiments of the present invention allow for hot-swapping of active memory units (e.g., pages) in a race-free and efficient manner.

[0015] Example embodiments of the present invention describe an enhanced DMA remapping unit (DRU) at the chipset level. The enhanced DRU unit works in conjunction with a VMM. The DRU includes an interface to accept page swap-requests from the VMM. Upon receiving the request, the DRU executes the page swap while (at least partially) blocking DMA requests directed towards the page to be swapped.

[0016] Embodiments of the invention may be included in systems and servers that utilize features that require memory hot-swap capability. Said features include, for example, Reliability, Availability and Serviceability (RAS), memory low-power mode such as offline state (where power to DIMMs are cut off to save power), and Non-Uniform Memory Access (NUMA) node affinity wherein cross-node VM pages are reorganized to a single node when available memory resources change. These features require the ability to hot swap content of one in-use page to another, without breaking ongoing executions.

[0017] FIG. 1 is a block diagram of a system utilizing an embodiment of the invention. System 100 may be included in a system server, desktop computer device, a mobile computer device, or any other any device utilizing a processor and system memory.

[0018] As shown in the example embodiment of FIG. 1, system 100 includes system memory 110, processor 120, system devices 130-139 (e.g., external disk drives, I/O devices) and memory controller 150. In this embodiment, devices 130-139 are DMA capable devices.

[0019] System 100 may further include VMM 170 to manage VM 160, including allocating portions of memory (i.e., pages) to VM 160. VM 160 may contain a guest operating system running various applications (i.e., guest software). Both VM 160 and VMM 170 may be executed via processor 120.

[0020] VMM 170 may further manage devices 130-139 and assign at least one of said devices to VM 160. In this example embodiment, devices 130, 131 and 139 are assigned to VM 160. It is to be understood that in alternative embodiments, a system may include a plurality of VMs, and system devices may be assigned to each VM in any combination.

[0021] The memory address range that guest software is provided, i.e. the Guest Physical Address (GPA) range, may not be the same as the Machine Physical Address (MPA) range. DMA capable devices 130, 131 and 139 use the MPA to transfer the data to and from physical memory locations. However, in this embodiment, the guest software of VM 160 (i.e., the guest OS) include device drivers to control devices 130, 131 and 139, and thus provide GPA ranges instead of MPA ranges.

[0022] DRU 155 is used to do the required address conversion. Since the GPA is provided by VMM 170, GPA ranges must be converted to MPA ranges. In one embodiment, VMM 170 programs DRU with the GPA to MPA conversion information so the DMA remapping hardware is able to perform the necessary translation. This conversion information is referred to herein as the DMA remapping page table (DRPT). Using the DRPT, data from a device can now be transferred directly to the appropriate memory address rather than going through an intermediate software emulation layer.

[0023] Memory 110 may comprise several units of memory, e.g., a plurality of DIMMs. The ability to “hot-remove” or “hot-swap” memory involves removing DIMMs while they are in use. Hot swapping memory involves, for example, moving the content of (at least) one in-use page included in a first DIMM (that is to be power down/moved) to another page included in a second DIMM (that will not be powered down/moved), without breaking ongoing executions. The operation to swap a page in this manner is referred to herein as a Hot Page Swap (HPS).

[0024] During the swapping of a page, it is important that no access is made to the page. VMM 170 can freeze execution of VM 160, thus preventing VM 160 from accessing the page to be swapped. VMM 170 can alternatively just freeze VM accesses to the page to be swapped. However, VMM 170 has no control over devices 130, 131 and 139 with regards to DMA requests each device may issue.

[0025] It is possible to modify the device drivers in VM 160 for each of devices 130, 131 and 139 to proactively freeze assigned devices. Before starting a page swap operation, VMM 170 may send a freeze request via VM 160 to the modified device drivers. The modified device drivers may then signal that it is safe to start the hot-swap operation. Upon completion of the hot-swap, VMM 170 may send an unfreeze request to the modified device drivers. However, this solution is not ideal for several reasons.

[0026] Commands to freeze/unfreeze (i.e., stop/resume) devices are time expensive operations. Uncertain latency is required to ensure all ongoing DMA requests for each device are drained, and further time is required to wait for each device to enter a quiescent state. This can further impact I/O throughput and affect system performance. This additional latency may further prevent hardware optimization. More importantly, modification to existing drivers is better to be avoided for well-known maintenance and distribution concerns.

[0027] Therefore, embodiments of the invention include enhanced DRU unit 155 to receive DMA requests from devices 130-139 and forward said requests to memory controller

150. DRU unit 155 also handles hot page swap requests issued by VMM 170. By removing hot swap execution from VMM 170, and thus removing VMM from having to eliminate race conditions, device DMA requests are controlled in a more efficient manner.

[0028] FIG. 2 is a block diagram of DRU 155 according to an embodiment of the invention. DRU 155 includes Hot Page Swap Command Interface (HPSCI) 200, through which the system VMM can notify the DRU of an HPS request.

[0029] An HPS request generated by the VMM may include information indicating the target Guest Page Frame Number (GPFN) to be updated, the original Machine Page Frame Number (MPFN) mapped by the target GPFN, the new MPFN to be mapped by the target GPFN, and a status field to indicated error and completion information for the HPS request. The GPFN may alternatively be referred to as a “virtual address.” It is to be understood that the original MPFN and new MPFN are included in system memory 110.

[0030] HPSCI 200 performs various validation checks on an HPS request. For example, HPSCI 200 may ensure the mapping of the target GPFN is consistent with the original MPFN, or if the requested swap cannot be completed because the memory is not assigned to the issuing VM.

[0031] After an HPS request is validated, Hot Page Swap Agent (HPSA) 210 may execute the page swap. HPSA 210 reads out the contents from the original MPFN, and then writes to the new MPFN. In one embodiment, these memory access requests are interleaved with on-going device DMA requests. Upon finish of the memory copy, HPSA 210 then changes the DRPT to reflect the latest mapping of the target GPFN to the new MPFN. HPSA 210 may further invalidate the internal DMA Remapping TLB to avoid stale mapping entries after the HPS is executed.

[0032] DRU 155 further includes device interface 220 to accept and fulfill DMA requests for system devices. Thus, DRU 155 has control over device DMA requests (as opposed to the VMM), and can block DMA requests to pages being swapped if necessary.

[0033] The HPSCI described above may be implanted in a synchronous or asynchronous manner. FIG. 3 is a flow diagram of an HPSCI processing an HPS request according to an embodiment of the invention. Flow diagrams as illustrated herein provide examples of sequences of various process actions. Although shown in a particular sequence or order, unless otherwise specified, the order of the actions can be modified. Thus, the illustrated implementations should be understood only as examples, and the illustrated processes can be performed in a different order, and some actions may be performed in parallel. Additionally,

one or more actions can be omitted in various embodiments of the invention; thus, not all actions are required in every implementation. Other process flows are possible.

[0034] Process 300 begins with a VMM receiving a command to execute an HPS request, 310. This HPS request may be a processor instruction issued to the VMM. The VMM will lock the DRPT from being modified by other commands in response to receiving said command, 320.

[0035] The VMM compiles the necessary hot-swap information, such as target GPFN, original MPFN and new MPFN, into an HPS request, 330. VMM sends the HPS request to an HPSCI of a DRU while also blocking VMs from accessing the corresponding GPFN, 340. In this embodiment, the VMM is further blocked from receiving other HPS requests.

[0036] The HPSCI will then verify and forward the HPS request to the HPSA, 350. Access to the original MPFN should be blocked during the HPS operation. As mentioned above, VMs are blocked from accessing the original MPFN (via the corresponding GPFN). Devices must also be blocked from accessing the original MPFN via DMA requests. This blocking is done via the DRU, 360.

[0037] The HPS request is then fulfilled by swapping the content of the original MPFN to the new MPFN, 365. The HPSA further modifies the DRPT and flushes the DMA Remapping TLB, 370. The HPSA notifies the HPSCI that the HPS request has been fulfilled, and the HPSCI furthers this notification to the VMM, 380. The DRU unblocks DMA requests from system devices, 390. The VMM unblocks the VMs and unlocks the DRPT, 395.

[0038] In an alternative embodiment, the HPSCI is implemented in an asynchronous manner. In this embodiment, the VMM is not blocked from receiving other HPS requests during the hot-swap operation. In this embodiment, VMM can continue to queue other HPS requests, and the HPSCI may handle these queued requests in parallel. In one embodiment, the VMM sets a threshold for the number of parallel requests that may be serviced.

[0039] In the example embodiment discussed above, devices are prevented from issuing a DMA requests during the host-swap process. These DMA requests may be held in some internal buffer, either at the DRU or on each device, and may be processed only when new mapping is re-established. This embodiment runs the risk of buffer overflow if too many DMA requests are issued during a hot-swap operation, and overall performance may be negatively affected.

[0040] FIG. 4 illustrates an example process that minimizes the number of DMA requests that are blocked during an HPS operation. As discussed above, to hot swap an in-use

page, the whole content of the old page has to be copied to the new page. This copy process must ensure the new page includes latest content.

[0041] In some systems, there are still DMA accesses (i.e. write-ops) issued from system devices even though the VM that owns these devices is blocked. Furthermore, though guest software usually manages memory in page units (i.e., 4Kb), memory controllers often accept burst accesses to memory DIMMs in cache line units (i.e., 64 bytes).

[0042] Thus, copying an old page to a new page incurs dozens of “cache line” copies — e.g., if one page is to be hot-swapped, then there are 64 “cache line” sized writes that occur for that operation. While it is easier to block device DMA requests until a whole page copy is finished, it adds latency and risk of device internal buffer overflow. Process 400 illustrates handling HPS operations and DMA requests in an interleaved manner, therefore allowing device to access an in-swap page in most cases.

[0043] In this embodiment, an internal counter is used to track which “cache line” is currently being copied during an HPS operation. Thus, when the DRU receives a device DMA request, 410, the DRU determines the location of the memory requested in relation to the internal counter, 420.

[0044] If the DMA request is to a “cache line” smaller than the internal counter, 430, this indicates that this particular portion of memory has already been swapped, so the DMA target address will be changed and instead targeted to the new “cache line” location (i.e., the new MPFN), 435. The updated DMA request is then allowed, 445.

[0045] If the DMA request is to a “cache line” larger than the internal counter, 440, this indicates that this particular portion of memory has not been swapped, and it is still safe to allow the DMA request to original “cache line” location (i.e., the original MPFN), 445.

[0046] If the DMA request is to a “cache line” equal to the internal counter, this indicates that this particular portion of memory is currently being swapped, and therefore this DMA request must be blocked to prevent a race condition or memory inconsistency, 450.

[0047] In one embodiment, the DRU unit waits for the current “cache line” to be swapped, 460, and then updates the target address of the (now unblocked) DMA request to the new MPFN (i.e., operation 435). The updated DMA request is then allowed, 445. It will be understood that the example process of FIG. 4 reduces the number of DMA requests that need to be blocked during an HPS operation.

[0048] Various components referred to above as processes, servers, or tools described herein may be a means for performing the functions described. Each component described

herein includes software or hardware, or a combination of these. The components can be implemented as software modules, hardware modules, special-purpose hardware (e.g., application specific hardware, ASICs, DSPs, etc.), embedded controllers, hardwired circuitry, etc. Software content (e.g., data, instructions, configuration) may be provided via an article of manufacture including a computer storage readable medium, which provides content that represents instructions that can be executed. The content may result in a computer performing various functions/operations described herein. A computer readable storage medium includes any mechanism that provides (i.e., stores and/or transmits) information in a form accessible by a computer (e.g., computing device, electronic system, etc.), such as recordable/non-recordable media (e.g., read only memory (ROM), random access memory (RAM), magnetic disk storage media, optical storage media, flash memory devices, etc.). The content may be directly executable ("object" or "executable" form), source code, or difference code ("delta" or "patch" code). A computer readable storage medium may also include a storage or database from which content can be downloaded. A computer readable medium may also include a device or product having content stored thereon at a time of sale or delivery. Thus, delivering a device with stored content, or offering content for download over a communication medium may be understood as providing an article of manufacture with such content described herein.

CLAIMS

1. A system comprising:
 - a memory;
 - a processor;
 - a virtual machine (VM) to be executed via the processor;
 - a device;
 - a virtual machine monitor (VMM) to be executed via the processor, the VMM to
 - assign the device to the VM;
 - control the VM's access to the memory, the processor and the device,
 - issue a hot-page swap (HPS) request, the HPS request to include a virtual address in use by at least one of the VM and the device, a first memory page location mapped to the virtual address, and a second memory page location, and
 - block access to the virtual address from the VM until the HPS request is complete; and
 - a DMA Remapping Unit (DRU) operatively coupled to the processor and the memory, the DRU to
 - receive and verify the HPS request,
 - copy content of the first memory page location to the second memory page location;
 - remap the virtual address from the first memory page location to the second memory page location, and
 - block DMA requests from the device to addresses of memory being remapped.
2. The system of claim 1, the DRU to further block all DMA requests until the HPS request is fulfilled.
3. The system of claim 1, the DRU to further block all DMA requests to the first memory page location until the HPS request is fulfilled.
4. The system of claim 1, the DRU to further:
 - fulfill DMA requests based on the second memory page location if the address of DMA requested memory has already been remapped; and
 - fulfill DMA requests based on the first memory page location if the address of DMA requested memory has yet to be remapped.
5. The system of claim 1, the VMM to further block additional HPS requests until the HPS request is complete.

6. The system of claim 1, the VMM to further queue additional HPS requests, the DRU to process additional HPS requests in parallel with the HPS request.
7. The system of claim 1, the HPS request to enable at least one of power to be cut off to the first memory page location, and memory reorganization of the first memory page location.
8. A method comprising:
 - receiving and verifying, from a virtual machine monitor (VMM), a hot-page swap (HPS) request, the HPS request to include a virtual address in use by at least one virtual machine (VM), a first memory page location mapped to the virtual address, and a second memory page location;
 - blocking DMA requests to addresses of memory being remapped until the HPS request is complete;
 - copying content of the first memory page location to the second memory page location; and
 - remapping the virtual address from the first memory page location to the second memory page location.
9. The method of claim 8, further comprising blocking all DMA requests until the HPS request is fulfilled.
10. The method of claim 8, further comprising:
 - fulfilling DMA requests based on the second memory page location if the address of DMA requested memory has already been remapped; and
 - fulfilling DMA requests based on the first memory page location if the address of DMA requested memory has yet to be remapped.
11. The method of claim 8, further comprising blocking additional HPS requests until the HPS request is complete.
12. The method of claim 8, further comprising processing additional HPS requests in parallel with the HPS request.
13. The method of claim 8, further comprising cutting off power to the first memory page location.
14. The method of claim 8, further comprising reorganizing the memory of the first memory page location.
15. An apparatus comprising:

a first logic to receive and verify a hot-page swap (HPS), the HPS request to include a virtual address, a first memory page location mapped to the virtual address in use by at least one virtual machine (VM), and a second memory page location;

a device interface to receive DMA requests, the device interface to block DMA requests to addresses of memory being remapped until the HPS request is complete; and

a second logic to copy content of the first memory page location to the second memory page location, and to remap the virtual address from the first memory page location to the second memory page location.

16. The apparatus of claim 15, the device interface to further block all DMA requests until the HPS request is fulfilled.

17. The apparatus of claim 15, the device interface to further:

fulfill DMA requests based on the second memory page location if the address of DMA requested memory has already been remapped; and

fulfill DMA requests based on the first memory page location if the address of DMA requested memory has yet to be remapped.

18. The apparatus of claim 15, the first logic to further process additional HPS requests in parallel with the HPS request.

19. The apparatus of claim 15, the HPS request to enable at least one of power to be cut off to the first memory page location, and memory reorganization of the first memory page location.

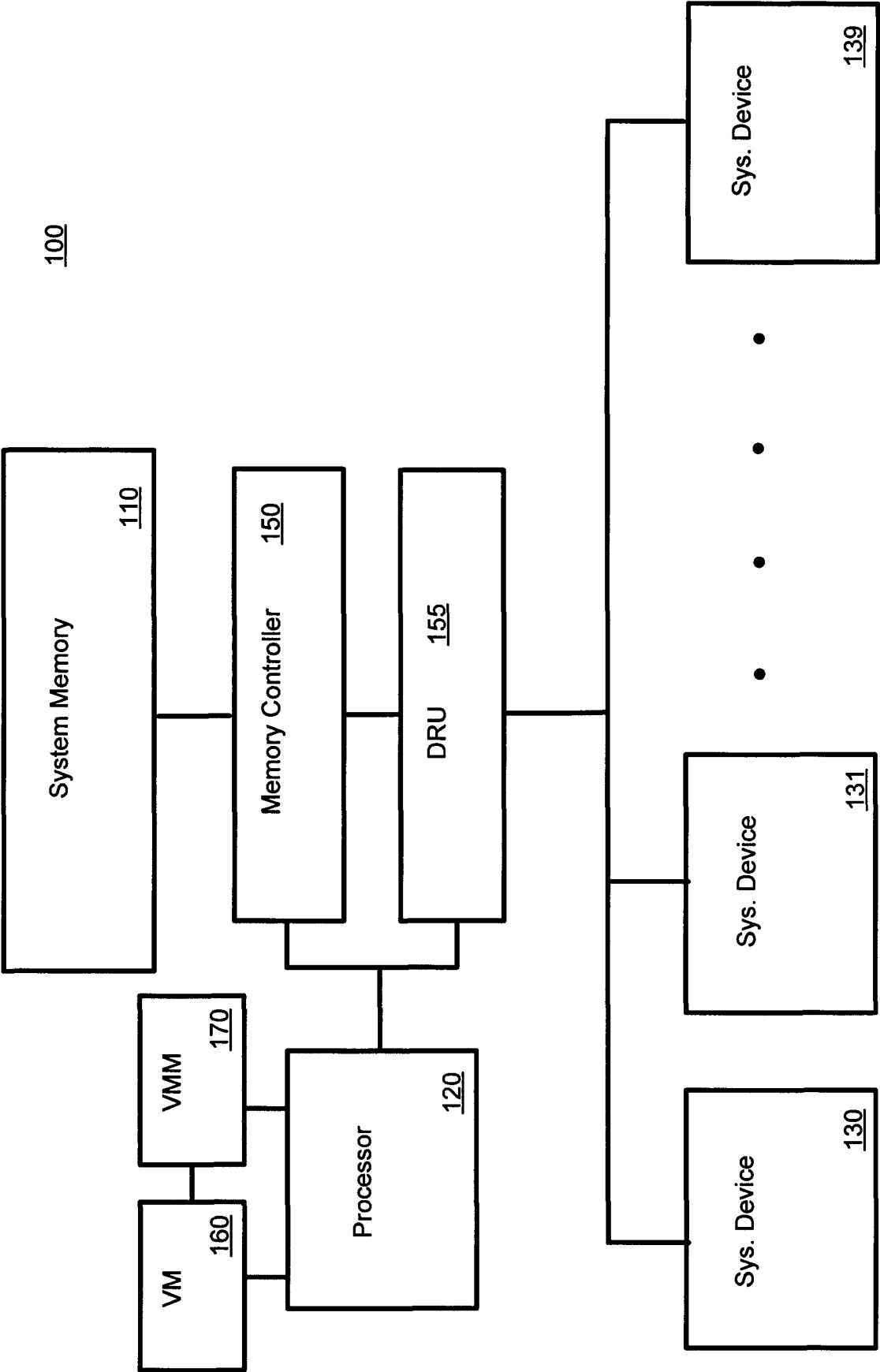


FIG. 1

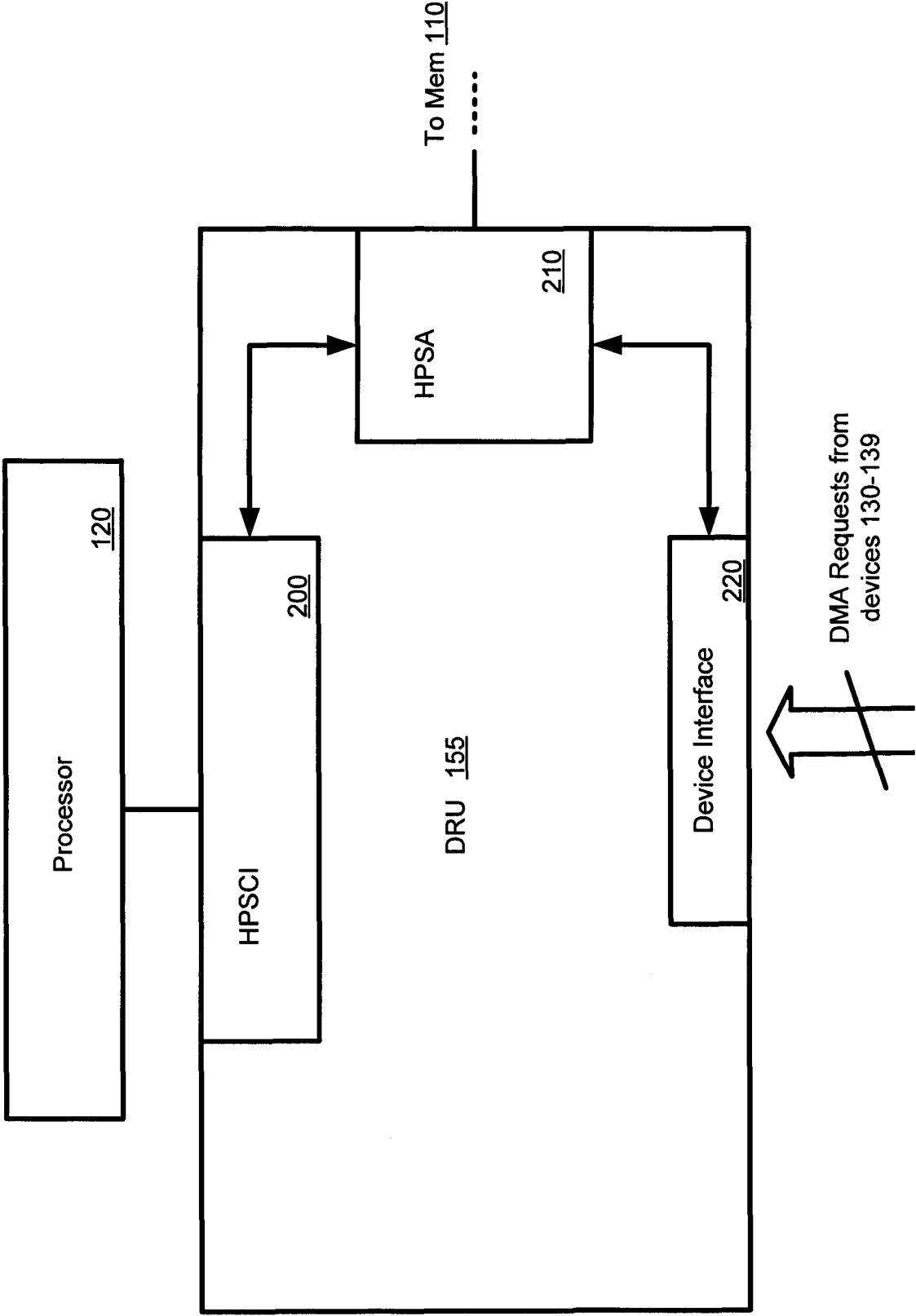
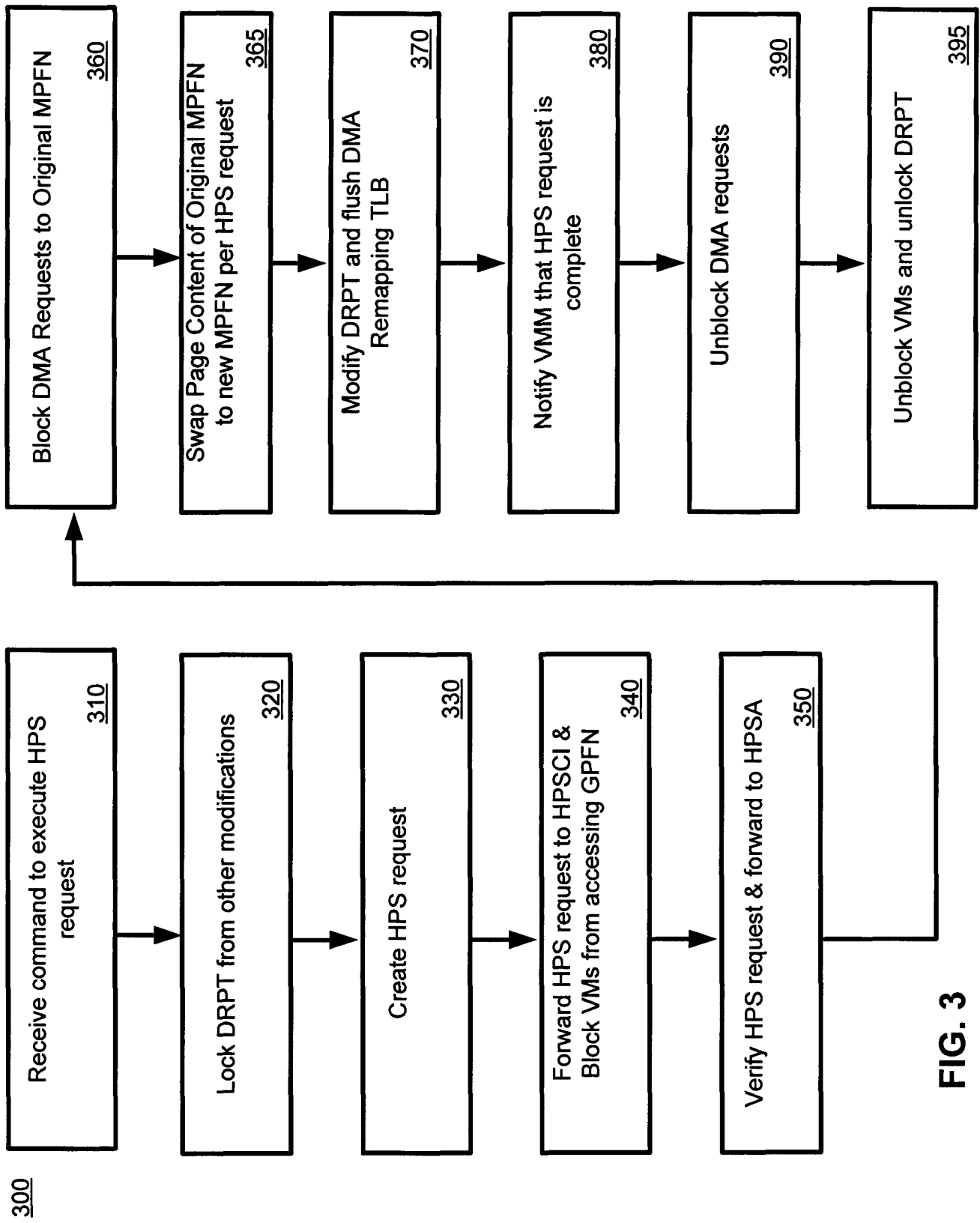
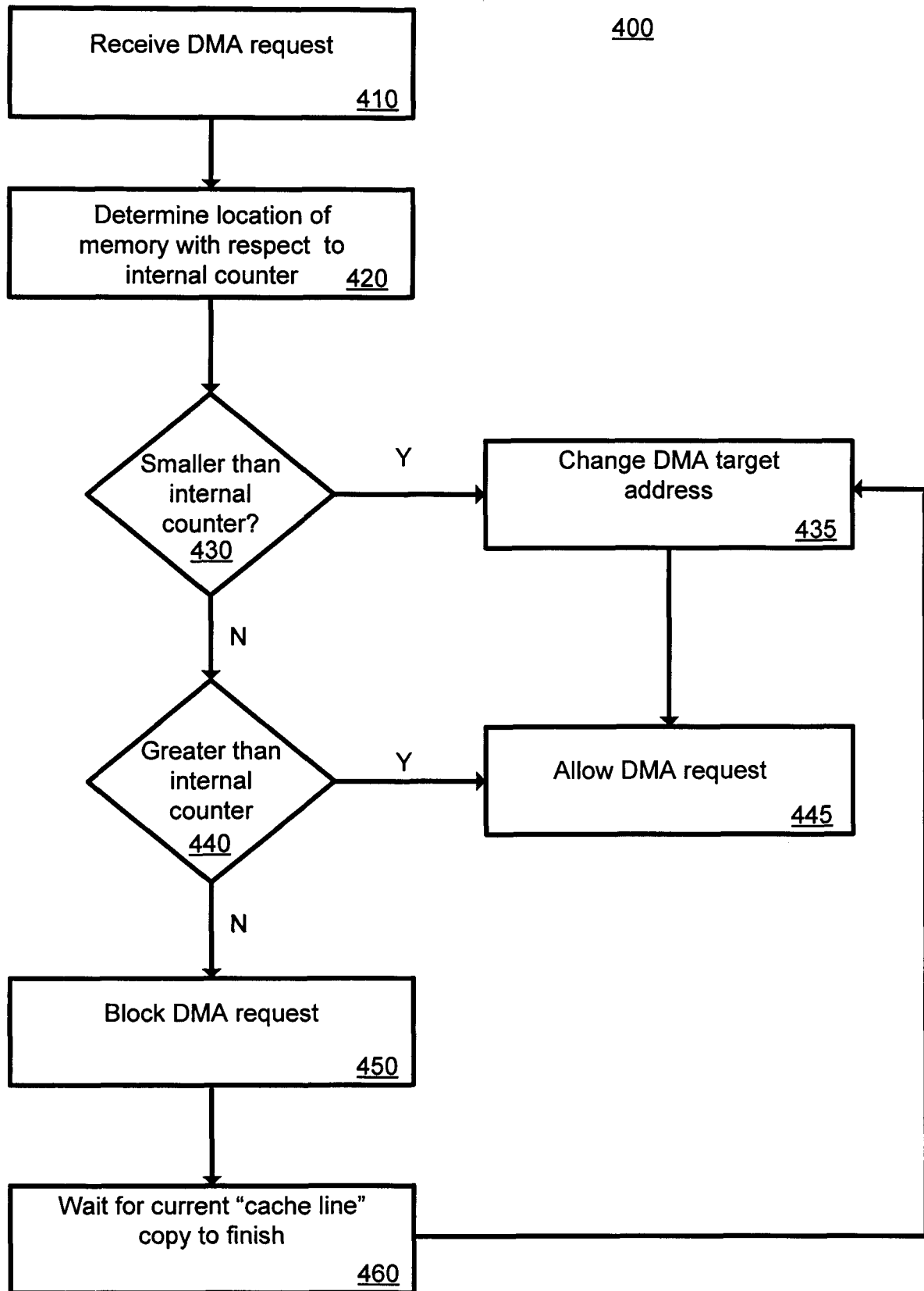


FIG. 2



**FIG. 4**

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2010/071456

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/08 (2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: G06F 12/-, G06F 11/-, G06F 13/-

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Database: WPLEPODOC, VCN, VEN, CPRS, CNKI, ACM

Keywords: hot, page, swap, virtual, machine, dma, remap+, block+

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US2007156968A1(Madukkarumukumana et al.)05 Jul. 2007(05.07.2007) the abstract, the paragraphs [0002], [0020], [0023], [0034], [0038], the figures 2-5	1-19
Y	Carl A. Waldspurger, Memory resource management in VMware ESX server, ACM SIGOPS Operating Systems Review - OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation, Volume 36 Issue SI, Winter 2002 the abstract, the section 3.1	1-19
A	CN101031888A(INTEL CORP.)05 Sep. 2007(05.09.2007) the whole document	1-19
A	US6247079B1(Micron Electronics, INC)12 Jun. 2001(12.06.2001) the whole document	1-19

☐ Further documents are listed in the continuation of Box C.☒ See patent family annex.

* Special categories of cited documents:	“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
“A” document defining the general state of the art which is not considered to be of particular relevance	“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
“E” earlier application or patent but published on or after the international filing date	“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
“L” document which may throw doubts on priority claim (S) or which is cited to establish the publication date of another citation or other special reason (as specified)	“&” document member of the same patent family
“O” document referring to an oral disclosure, use, exhibition or other means	
“P” document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search 20 Dec. 2010(20.12.2010)	Date of mailing of the international search report 30 Dec. 2010 (30.12.2010)
Name and mailing address of the ISA/CN The State Intellectual Property Office, the P.R.China 6 Xitucheng Rd., Jimen Bridge, Haidian District, Beijing, China 100088 Facsimile No. 86-10-62019451	Authorized officer GUO, Yongqiang Telephone No. (86-10)62412084

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2010/071456

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
US2007156968A1	05.07.2007	WO2007078958A2	12.07.2007
		KR20080080594A	04.09.2008
		WO2007078958A3	03.01.2008
CN101031888A	05.09.2007	CN101351773A	21.01.2009
		US2006075285A1	06.04.2006
		DE112005002405T5	16.08.2007
		KR915714B1	04.09.2009
		US7340582B2	04.03.2008
		CN100498726C	10.06.2009
		TW315846B1	11.10.2009
		GB2432244A	16.05.2007
		KR20070047845A	07.05.2007
		WO2006039177A1	13.04.2006
US6247079B1	12.06.2001	JP2008515057T	08.05.2008
		GB2432244B	08.10.2008
		TW200627270A	01.08.2006
		None	