(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 July 2013 (04.07.20 13)

W I P O | P C T

(10) International Publication Number
WO 2013/100962 Al

(51) International Patent Classification:
G06F 9/22 (2006.01) *G06F 21/22* (2006.01)
G06F 9/44 (2006.01)

(21) International Application Number:
PCT/US201 1/067582

(22) International Filing Date:
28 December 201 1 (28. 12.201 1)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US):* **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, MS: RNB-4-150, Santa Clara, California 95052 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only):* **CASTELINO, Manohar R.** [IN/US]; 1063 Morse Avenue, 11-103, Sunnyvale, California 94089 (US). **SHANBHOGUE, Vedvyas** [IN/US]; 13686 NW Henninger Ln., Portland, Oregon 97229 (US). **RODRIGUEZ, Sergio** [MX/US]; 15344 Dermody Ave., San Lorenzo, California 94580 (US).

(74) Agent: **YEE, Edward W.;** Garrett IP, LLC, c/o CPA Global, P.O. Box 52050, Minneapolis, Minnesota 55402 (US).

(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, QA, RO, RS, RU, RW, 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, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, 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, RS, 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: SYSTEMS, METHODS AND COMPUTER PROGRAM PRODUCTS FOR BOOTSTRAPPING A TYPE 1 VIRTUAL MACHINE MONITOR AFTER OPERATING SYSTEM LAUNCH

(57) Abstract: Systems, methods, and computer program products that provide for the use of a type 2 VMM to de-link or isolate underlying processor hardware from an operating system. This may allow the launching of a task that requires direct access to processor hardware, where such access requires the absence of an operating system. Such a task may take the form of a type 1 VMM, such as an information security or integrity VMM, e.g., an anti-malware VMM.

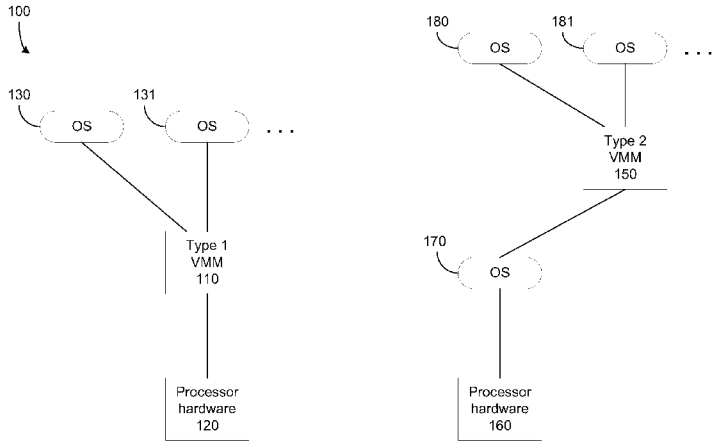

FIG. 1
(prior art)

WO 2013/100962 A1

SYSTEMS, METHODS AND COMPUTER PROGRAM PRODUCTS FOR BOOTSTRAPPING A TYPE 1 VIRTUAL MACHINE MONITOR AFTER OPERATING SYSTEM LAUNCH

BACKGROUND

A virtual machine monitor (VMM), also known as a hypervisor, is a mechanism for virtualizing hardware computing resources, allowing multiple operating systems (known as guests) to run concurrently on a host processor. A VMM may present to the guest operating systems a virtual operating platform and may manage the execution of the guest operating systems.

There are two forms of VMMs. A type 1 VMM may run directly on processor hardware, controlling the hardware to manage guest operating systems. In this case, a guest operating system may therefore run on a level above the VMM. A type 2 VMM may run within a conventional operating system environment. Such a VMM may therefore operate at a second software level, above the hardware and above the operating system. A guest operating system may therefore run at a third level in this setting.

Currently, a type 1 VMM may be launched in one of three ways. First, such a VMM may be launched prior to booting the operating system, using a custom boot strap process. Second, a type 1 VMM may be launched early in the boot phase by the operating system itself. This may involve direct access to operating system's core data structures, which in turn may require a variety of security permissions. This approach may require an operating system that is specially designed to support the launch of a type 1 VMM. Third, a type 1 VMM may be launched by first modifying one or more core operating system data structures without knowledge of the operating system. This represents an unconventional approach not envisioned by operating system designers. Moreover, this approach may result in significant instability. A computer system in such a situation may be susceptible to failure, particularly in the event of operating system changes or updates.

As a result, there is currently no solution that safely allows the launch of a type 1 VMM after an operating system has been enabled.

BRIEF DESCRIPTION OF THE DRAWINGS/FIGURES

FIG. 1 is a block diagram illustrating type 1 and 2 VMMs.

FIG. 2 is a flowchart illustrating the processing described herein, according to an embodiment.

FIG. 3 is a flowchart illustrating the operation of a type 2 VMM, according to an embodiment.

FIG. 4 is a flowchart illustrating the transition from a type 2 VMM to a type 1 VMM, according to alternative embodiment.

FIG. 5 is a flowchart illustrating the transition from a type 2 VMM to a type 1 VMM, according to an alternative embodiment.

FIG. 6 is a block diagram illustrating the computing environment of a software or firmware embodiment of the system described herein.

In the drawings, the leftmost digit(s) of a reference number identifies the drawing in which the reference number first appears.

DETAILED DESCRIPTION

An embodiment is now described with reference to the figures, where like reference numbers indicate identical or functionally similar elements. While specific configurations and arrangements are discussed, it should be understood that this is done for illustrative purposes only. A person skilled in the relevant art will recognize that other configurations and arrangements can be used without departing from the spirit and scope of the description. It will be apparent to a person skilled in the relevant art that this can also be employed in a variety of other systems and applications other than what is described herein.

The systems, methods, and computer program products described herein provide for the use of a type 2 VMM to de-link or isolate underlying processor hardware from an operating system. This may allow the launching of a task that requires direct access to processor hardware, where such access requires the absence of an operating system. Such a task may take the form of a type 1 VMM, such as an information security or integrity VMM, e.g., an anti-malware VMM. In alternative embodiments, a type 1 VMM of another type may be used.

Generally, a type 1 VMM may operate directly on processor hardware, without an intervening operating system. This is illustrated in FIG. 1. Here, a type 1 VMM 110 is shown operating on processor hardware 120. The VMM 110 may allow virtualization of hardware resources for each of several guest operating systems 130, 131, etc. In contrast, a type 2 VMM 150 may virtualize resources of hardware 160, but operates within the context of an operating system 170. A typical type 2 VMM may allow virtual access to hardware 160 by each of several guest operating systems 180, 181, etc.

The overall processing of the system described herein is illustrated in FIG. 2, according to an embodiment. At 210, a basic I/O system (BIOS) may be initiated. At 220 a master boot record may be loaded in advance of a subsequent boot-up. At 230, the operating system may be loaded. At 240, the operating system may be started. A 250, a type 2 VMM may be run, where this VMM de-links the operating system from the underlying processor, i.e., central processing unit (CPU) hardware, effectively freezing the operating system and isolating the processor from the operating system. At 260, a task that requires direct access to the processor may be launched, where this access requires that the operating system not be active. An example of such a task may be a type 1 VMM, such as an anti-malware VMM. Alternatively, such a task may be a scan for system failures, or a deep probing process of computing hardware.

The running of the above type 2 VMM (250 in FIG. 2) is illustrated in greater detail in FIG. 3, according to an embodiment. At 310, memory may be allocated for this type 2 VMM. At 320, the image for the type 1 VMM may be loaded, where this latter VMM may be launched at a subsequent point, as will be described below. As discussed above, an image for a different task (other than a type 1 VMM) requiring direct access to the processor may be loaded instead. At 330, both static and dynamic guest operating system states may be saved. At 340, a virtual machine control structure (VMCS) may be set, along with a VMCS control field. At 350, the guest and host OS states in the VMCS may be populated. At 360, the type 2 VMM may be launched, shown here as a virtualization extension launch, or VMX. At 370, the host and guest states may be saved and the operating system effectively halted. Note that in a multi-processor environment, the OS states may be saved and halted for all processors.

At 375, a transition from the type 2 VMM to the type 1 VMM may be performed, in a manner to be described in greater detail below. At 380, the type 1 VMM (or other task requiring direct access to the processor hardware) may be launched. At this point, the processor may look as if a boot had just taken place, and the processor may be put into any state desired by the type 1 VMM.

A process for transitioning from the type 2 VMM to the type 1 VMM (375 in FIG. 3) is illustrated in FIG. 4, according to an embodiment. At 410, the processor may be reconfigured to a state needed by the type 1 VMM. At 420, the halted guest operating system state may be passed to the type 1 VMM, which may then be launched as shown at 380 of FIG. 3.

In an alternative embodiment, shown in FIG. 5, a scanning utility may be employed to perform system checks prior to the launch of the type 1 VMM. In an embodiment, these system checks may be security-related, and may detect compromises to system security. Alternatively or in addition,

these checks may detect failure conditions. At 510, the processor may be reconfigured to a state required by such a scanning utility. At 520, this scanning utility may be executed. At 530, the processor may be reconfigured to a state needed by the type 1 VMM. At 540, the halted guest operating system state may be passed to the type 1 VMM, which may then be launched as shown at 380 of FIG. 3.

One or more features disclosed herein may be implemented in hardware, software, firmware, and combinations thereof, including discrete and integrated circuit logic, application specific integrated circuit (ASIC) logic, and microcontrollers, and may be implemented as part of a domain-specific integrated circuit package, or a combination of integrated circuit packages. The term software, as used herein, refers to a computer program product including at least one computer readable medium having computer program logic stored therein to cause a computer system to perform one or more features and/or combinations of features disclosed herein. The computer readable medium may be transitory or non-transitory. An example of a transitory computer readable medium may be a digital signal transmitted over a radio frequency or over an electrical conductor, through a local or wide area network, or through a network such as the Internet. An example of a non-transitory computer readable medium may be a compact disk, a flash memory, ROM, RAM, or other data storage device.

In an embodiment, some or all of the processing described herein may be implemented as software or firmware. Such a software or firmware embodiment is illustrated in the context of a computing system 600 in FIG. 6. System 600 may include a central processing unit (CPU) 620 and a body of memory 610 that may include one or more non-transitory computer readable media that may store computer program logic 640. Memory 610 may be implemented as a read-only memory (ROM) or random access memory (RAM) device, for example. CPU 620 and memory 610 may be in communication using any of several technologies known to one of ordinary skill in the art, such as a bus or a point-to-point interconnect. Computer program logic 640 contained in memory 610 may be read and executed by CPU 620. In an embodiment, One or more I/O ports and/or I/O devices, shown collectively as I/O 630, may also be connected to CPU 620 and memory 610.

In the embodiment of FIG. 6, computer program logic 640 may include a module 650 responsible for saving a host state. Computer program logic 640 may also include a module 660 responsible for saving a guest state. Modules 650 and 660 may allow for the halting of an operating system while retaining current states of the operating system and processor 620. In addition,

computer program logic 640 may include a module 670 responsible for launch of a type 1 VMM (or other task requiring direct access to processor 620 without an operating system).

Methods and systems are disclosed herein with the aid of functional building blocks illustrating the functions, features, and relationships thereof. At least some of the boundaries of these functional building blocks have been arbitrarily defined herein for the convenience of the description. Alternate boundaries may be defined so long as the specified functions and relationships thereof are appropriately performed.

While various embodiments are disclosed herein, it should be understood that they have been presented by way of example only, and not limitation. It will be apparent to persons skilled in the relevant art that various changes in form and detail may be made therein without departing from the spirit and scope of the methods and systems disclosed herein. Thus, the breadth and scope of the claims should not be limited by any of the exemplary embodiments disclosed herein.

WHAT IS CLAIMED IS:

1. A method, comprising:

given an operating system (OS) running on a processor,

running a type 2 virtual machine monitor (VMM) that isolates the OS from the processor; and

launching a task that requires access to the processor in isolation from the OS.

2. The method of claim 1, wherein the task comprises a type 1 VMM.

3. The method of claim 1, wherein said running of the type 2 VMM comprises:

saving a current state of the OS; and

halting operation of the OS,

wherein the saving and halting are performed for all processors if the method is performed in a multi-processor environment.

4. The method of claim 3, wherein the running of the type 2 VMM further comprises:

reconfiguring the processor to a state required by the type 1 VMM; and

passing the saved OS state to the type 1 VMM,

performed after the halting of the operation of the OS.

5. The method of claim 4, the running of the type 2 VMM further comprises:

reconfiguring the processor to a state required to start execution of a scanning utility; and

executing the scanning utility, to detect one or more of a system failure or a system compromise,

performed before said reconfiguration of the processor to the state required by the type 1 VMM.

6. The method of claim 2, wherein the type 1 VMM is an anti-malware VMM.

7. The method of claim 1, wherein the task comprises scanning for system failures.

8. The method of claim 1, wherein the task comprises a probing of system hardware.

9. The method of claim 1, wherein said running of the type 2 VMM further comprises:

allocating memory for the type 2 VMM;

setting a current virtual machine control structure (VMCS);

setting a control field for the VMCS; and

populating guest and host states in the VMCS,

performed before said running of the type 2 VMM.

10. A computer program product including at least one non-transitory computer readable medium having computer program logic stored therein, the computer program logic including:

logic to cause a processor to run a type 2 virtual machine monitor (VMM) that isolates the OS from the processor, given an operating system (OS) already running on the processor; and

logic to cause the processor to launch a task that requires access to the processor in isolation from the OS.

11. The computer program product of claim 10, wherein the task comprises a type 1 VMM.

12. The computer program product of claim 10, wherein the logic to cause the processor to run the type 2 VMM comprises:

logic to cause the processor to save a current state of the OS; and

logic to cause the processor to halt operation of the OS,

wherein, if the processor is part of a multi-processor environment, the saving and halting are performed for all processors in the multi-processor environment.

13. The computer program product of claim 12, wherein the logic to cause the processor to run the type 2 VMM further comprises:

logic to cause the processor to reconfigure to a state required by the type 1 VMM; and

logic to cause the processor to pass the saved OS state to the type 1 VMM,

wherein the reconfiguration and the passing are performed after the halting of the OS.

14. The computer program product of claim 13, wherein the logic to cause the processor to run the type 2 VMM further comprises:

logic to cause the processor to reconfigure to a state required to start execution of a scanning utility; and

logic to cause the processor to execute the scanning utility, to detect one or more of a system failure or a system compromise,

wherein the reconfiguration to the state required to start execution of the scanning utility and the execution of the scanning utility are performed before the reconfiguration to the state required by the type 1 VMM.

15. The computer program product of claim 11, wherein the type 1 VMM is an anti-malware VMM.

16. The computer program product of claim 10, wherein the task comprises scanning for system failures.

17. The computer program product of claim 10, wherein the task comprises a probing of system hardware.

18. The computer program product of claim 10, wherein said logic to cause the processor to run the type 2 VMM comprises:

logic to cause the processor to allocate memory for the type 2 VMM;

logic to cause the processor to set a current virtual machine control structure (VMCS);

logic to cause the processor to set a control field for the VMCS; and

logic to cause the processor to populate guest and host states in the VMCS,

all performed before said running of the type 2 VMM.

19. A system, comprising:

a processor; and

a memory in communication with said processor, wherein the memory stores a plurality of processing instructions configured to direct said processor to

run a type 2 virtual machine monitor (VMM) that isolates the OS from the processor, given an operating system (OS) already running on the processor; and

launch a task that requires access to the processor in isolation from the OS.

20. The system of claim 19, wherein the task comprises a type 1 VMM.

21. The system of claim 20, wherein said processing instructions configured to direct said processor to run the type 2 VMM comprise instructions configured to direct said processor to:

save a current state of the OS; and

halt operation of the OS,

wherein the saving and halting is performed for all processors if the system comprises a multi-processor environment.

22. The system of claim 21, wherein the plurality of processing instructions configured to direct said processor to run the type 2 VMM further comprises instructions configured to direct said processor to:

reconfigure to a state required by the type 1 VMM; and

pass the saved OS state to the type 1 VMM,

wherein the reconfiguration and the passing are performed after the halting of the operation of the OS.

23. The system of claim 21, wherein the plurality of processing instructions configured to direct said processor to run the type 2 VMM further comprises instructions configured to direct said processor to:

reconfigure to a state required to start execution of a scanning utility; and

execute the scanning utility, to detect one or more of a system failure or a system compromise,

wherein the reconfiguration to the state required to start execution of the scanning utility and the execution of the scanning utility are performed before said reconfiguration to the state required by the type 1 VMM.

24. The system of claim 20, wherein the type 1 VMM is an anti-malware VMM.

25. The system of claim 19, wherein the task comprises scanning for system failures.

26. The system of claim 19, wherein the task comprises a probing of system hardware.

27. The system of claim 19, wherein said instructions configured to direct said processor to run the type 2 VMM further comprises instructions configured to direct said processor to:

allocate memory for the type 2 VMM;

set a current virtual machine control structure (VMCS);

set a control field for the VMCS; and

populate guest and host states in the VMCS,

all performed before said running of the type 2 VMM.

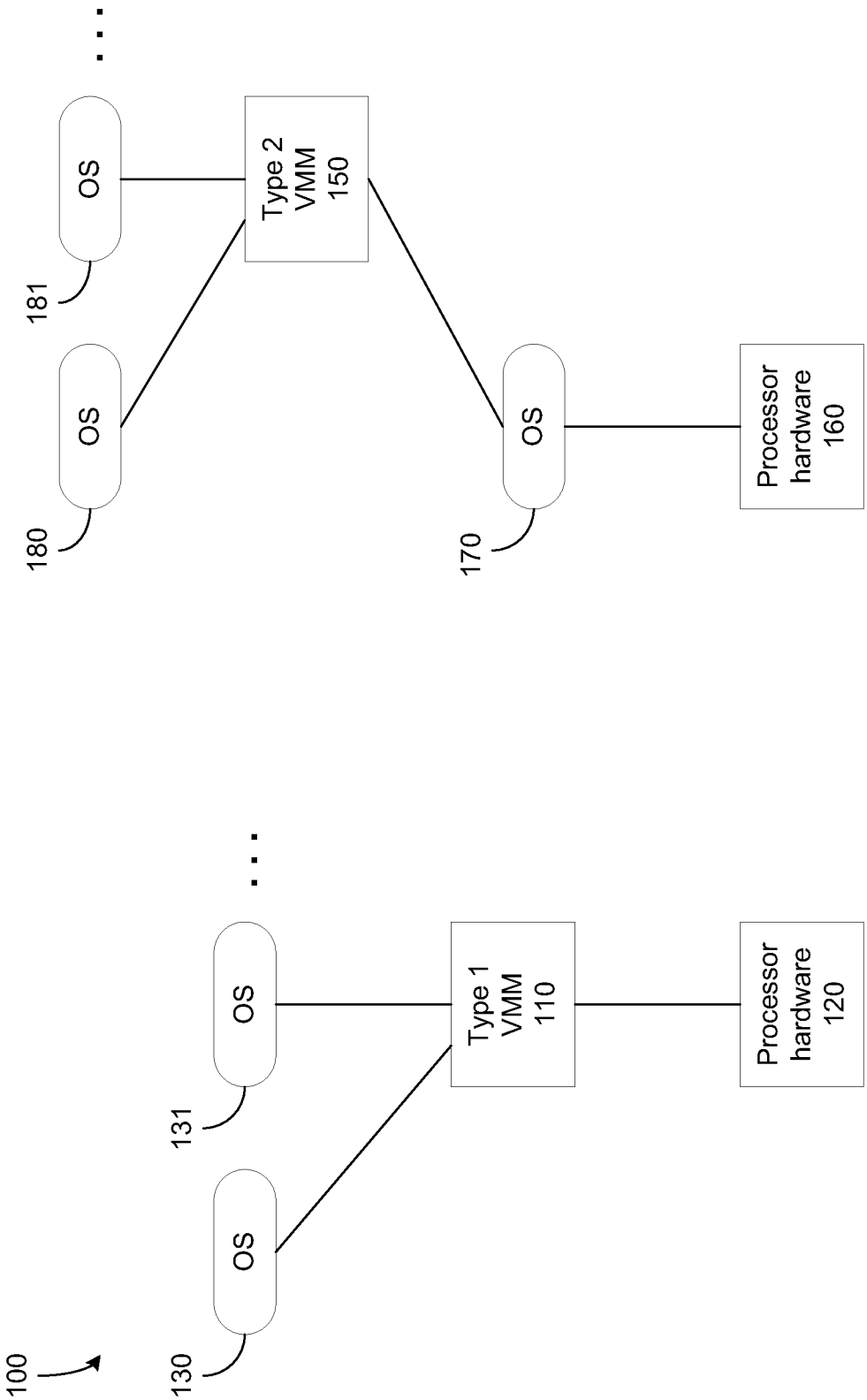

FIG. 1
(prior art)

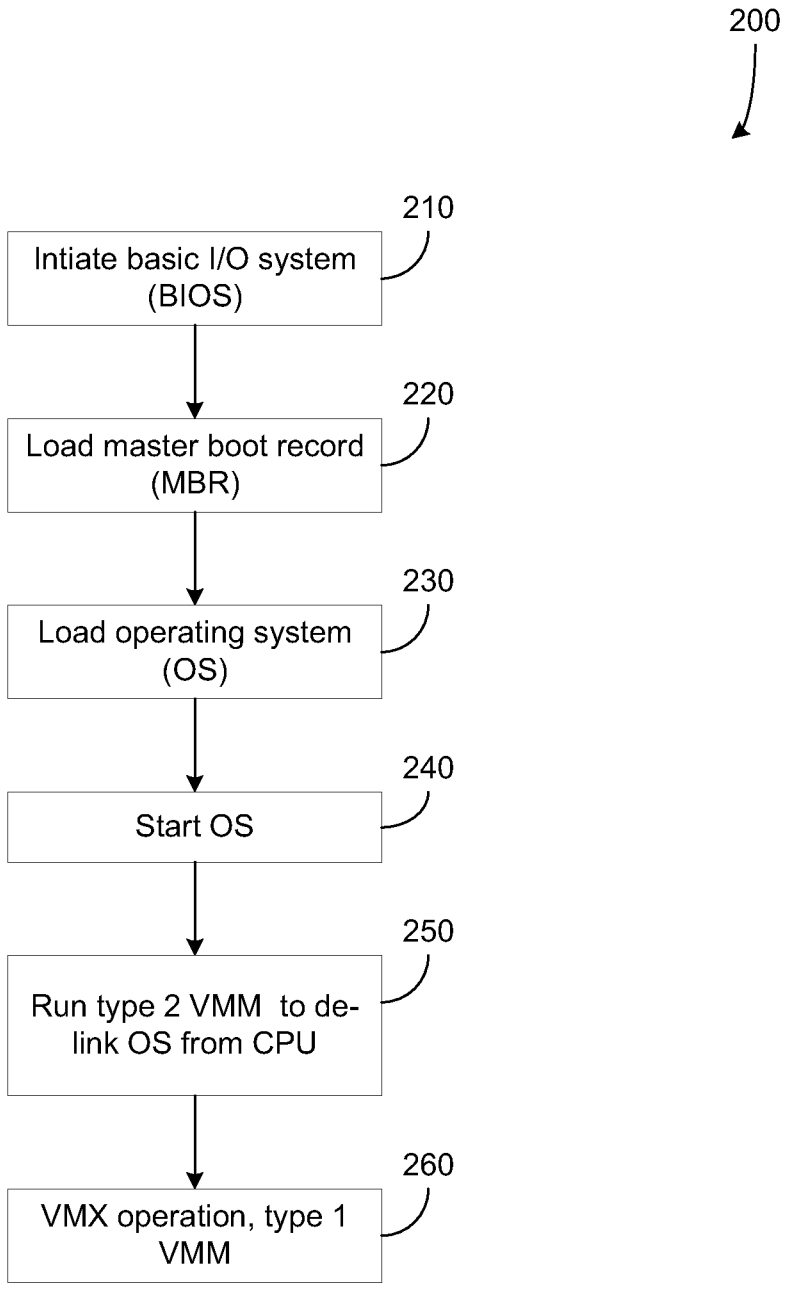
200
210
Intiate basic I/O system (BIOS)
220
Load master boot record (MBR)
230
Load operating system (OS)
240
Start OS
250
Run type 2 VMM to de-link OS from CPU
260
VMX operation, type 1 VMM

FIG. 2

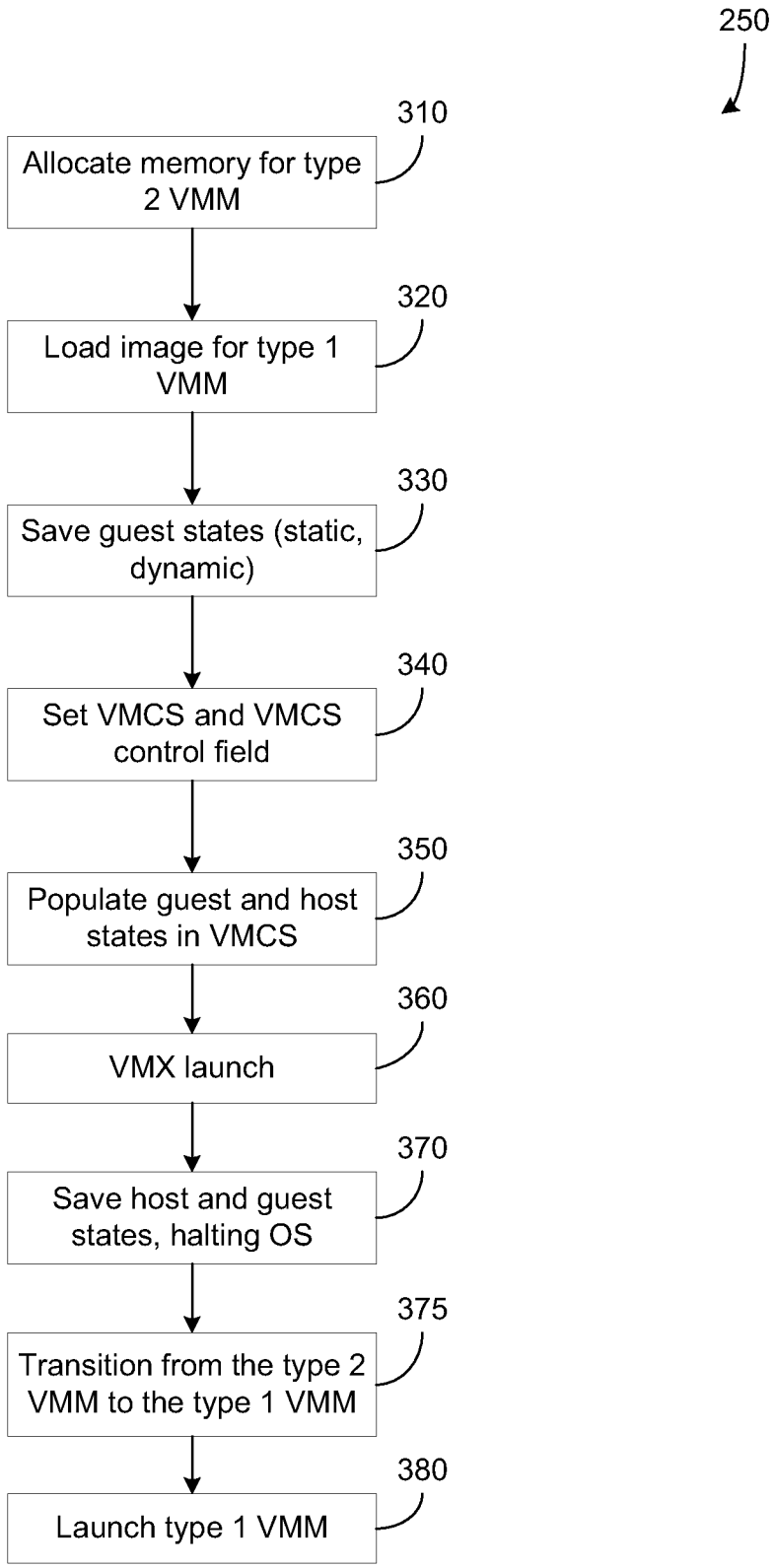
250
310
Allocate memory for type 2 VMM
320
Load image for type 1 VMM
330
Save guest states (static, dynamic)
340
Set VMCS and VMCS control field
350
Populate guest and host states in VMCS
360
VMX launch
370
Save host and guest states, halting OS
375
Transition from the type 2 VMM to the type 1 VMM
380
Launch type 1 VMM

FIG. 3

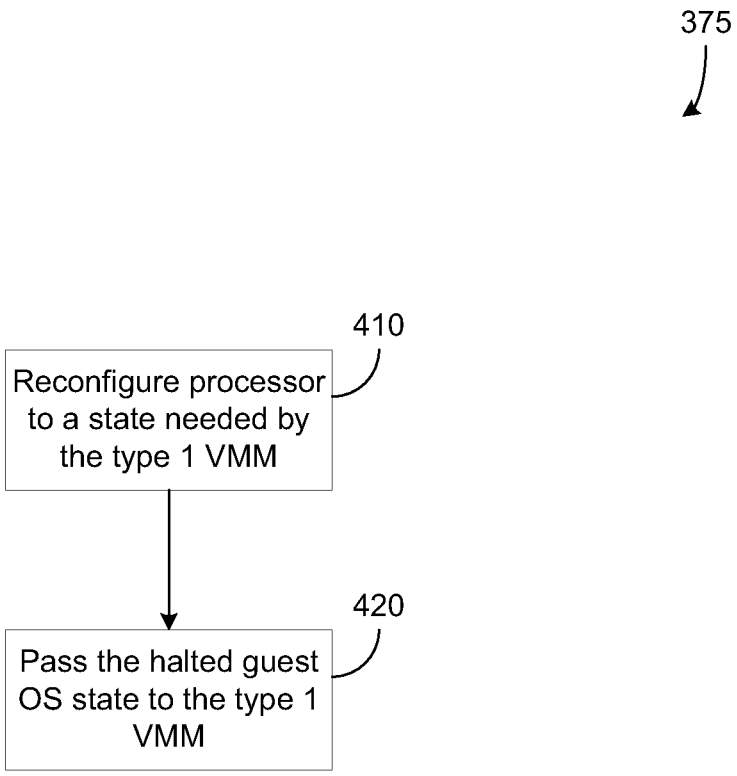
375
410
Reconfigure processor to a state needed by the type 1 VMM
420
Pass the halted guest OS state to the type 1 VMM

FIG. 4

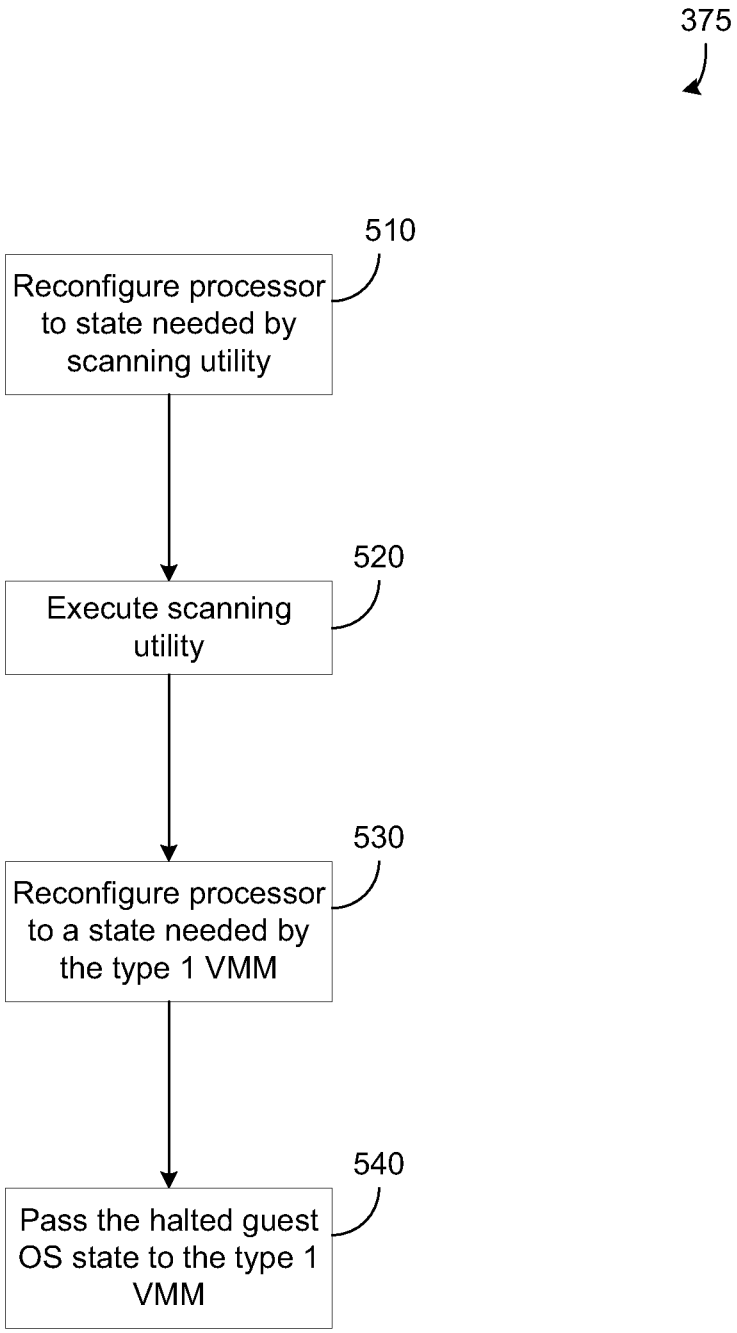
375
510
Reconfigure processor to state needed by scanning utility
520
Execute scanning utility
530
Reconfigure processor to a state needed by the type 1 VMM
540
Pass the halted guest OS state to the type 1 VMM

FIG. 5

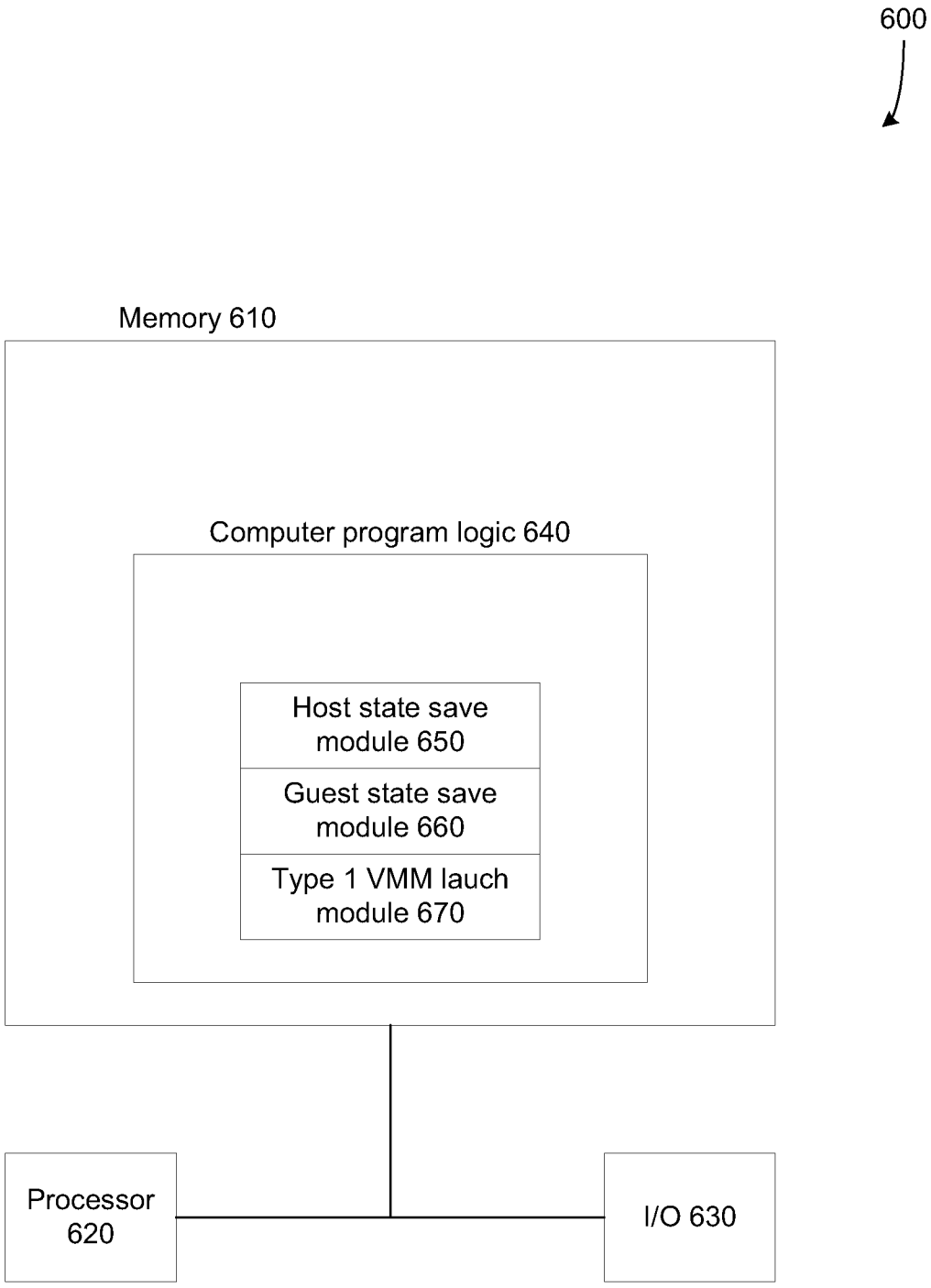
600
Memory 610
Computer program logic 640
Host state save module 650
Guest state save module 660
Type 1 VMM lauch module 670
Processor 620
I/O 630

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2011/067582

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/22(2006.01)i, G06F 9/44(2006.01)1, G06F 21/22(2006.01)1

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)

G06F 9/22; G06F 7/00; G06F 17/30; G06F 9/455; G06F 15/76; H04L 9/00; G06F 11/00

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords:operating system,VMM,task,isolation

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2011-0161955 Al (W0LLER THOMAS R. et al.) 30 June 2011 See paragraphs [0037] -[0043] and figures 3-8.	1-27
A	US 2009-0133097 Al (SMITH NED et al.) 21 May 2009 See paragraphs [0015] -[0050] and figures 1-2.	1-27
A	US 2011-0251992 Al (BETHLEHEM ALEXANDER et al.) 13 October 2011 See paragraphs [0125] -[0134] and figures 5-7.	1-27
A	US 7865762 B2 (SWANSON ROBERT C.) 04 January 2011 See column 5, line 5 - column 6, line 58.	1-27

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

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)

"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

"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

"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

"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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 SEPTEMBER 2012 (27.09.2012)	**27 SEPTEMBER 2012 (27.09.2012)**

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-70 1, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

LEE, Hee Bong

Telephone No. 82-42-481-8120

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2011/067582

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2011-0161955 A1	30.06.2011	WO 2011-090596 A2 WO 2011-090596 A3	28.07.2011 20.10.2011
US 2009-0133097 A1	21.05.2009	None	
US 2011-0251992 A1	13.10.2011	None	
US 7865762 B2	04.01.2011	US 2009-0144579 A1	04.06.2009