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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 December 2018 (20.12.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/227549 A1

(51) International Patent Classification:
G06F 9/50 (2006.01)

(21) International Application Number:
PCT/CN2017/088619

(22) International Filing Date:
16 June 2017 (16.06.2017)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **ALIBABA GROUP HOLDING LIMITED** [—/CN]; Fourth Floor, One Capital Place, P.O. Box 847, George Town, Grand Cayman (KY).

(72) Inventors: **KINGSUM, Chow**; Hangzhou, Zhejiang (US). **LI, Chengdong**; Hangzhou, Zhejiang (CN). **ZHU, Wanyi**; Hangzhou, Zhejiang (CN).

(74) Agent: **BEIJING TSINGYUANHUI INTELLECTUAL PROPERTY LAW FIRM**; Beijing Suzhou Street No. 362, Haidian District 3, Beijing 100080 (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, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, 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, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, 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, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) Title: DETERMINING PROCESSOR UTILIZATION OF MULTIPROCESSING SYSTEM WITH VIRTUALIZATION

(57) Abstract: The techniques described herein provide a solution for determining CPU usage of a multiprocessing processor core with operating system level virtualization. Co-existing software processes operated with a multiprocessing processor core are identified, and one or more containers are identified as associated to the co-existing software processes. The CPU usage of the multiprocessing processor core is attributed to the one or more containers.

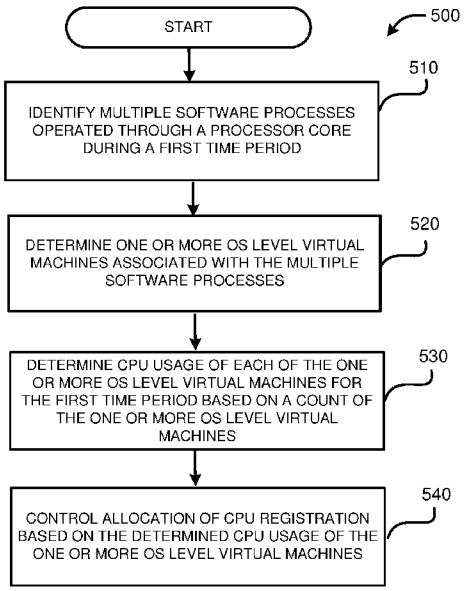


FIG. 5

WO 2018/227549 A1

DETERMINING PROCESSOR UTILIZATION OF MULTIPROCESSING SYSTEM WITH VIRTUALIZATION

BACKGROUND

[0001] Multiprocessing refers to the use of multiple processing components within a single computing system and/or the capacity of a system to support multiple processes and/or threads and allocate the processes and/or threads among multiple processing components. A multi-core processor is a single computing component with multiple (commonly even number of) independent physical processor cores. The multiple cores are either fabricated onto a single integrated circuit die (known as a chip multiprocessor or CMP), or onto multiple dies in a single chip package. Multithreading is a processor design/ability where a single processor or a single processor core of a multi-core processor to execute multiple threads simultaneously. In multithreading, the multiple threads share the resources of a single processor core, e.g., the computing units, the cache, and the translation lookaside buffer (TLB). From system design perspective, a processor core with the capacity of executing multiple program threads is called a multi-threading processor core and each such capacity is called a logical processor or a hardware thread as compared to a program thread executed in the "hardware thread."

[0002] CPU utilizations obtained from operating systems (OS) are common metrics that have been used for many purposes like product sizing, computer capacity planning, job scheduling, etc. However, with the advances in computer architecture designs including multi-processing systems, and especially hyper-threading, the CPU utilization as reported by the operating systems may be unreliable.

[0003] Virtualization is a computer architecture where multiple virtual machines (or virtual images) are multiplexed in the same hardware system. An operating system

level virtualization creates an abstraction layer between a host operating system and user applications. For example, operating system level virtualization may include multiple virtual servers (or containers) on a single physical server and the host operating system(s) to utilize the same hardware and software resources in, e.g., data center application(s).

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] To describe technical solutions in the embodiments of the present disclosure more clearly, the accompanying drawings are briefly described herein. The accompanying drawings described herein merely represent some embodiments of the present disclosure, and one of ordinary skill in the art may further derive other drawings from these accompanying drawings without making any creative effort. The use of the same reference numbers in different figures indicates similar or identical items.

[0005] FIG. 1 illustrates an example multiprocessing system.

[0006] FIG. 2 illustrates an example computing system including a multiprocessing system and operating system level virtualization.

[0007] FIG. 3 illustrates an example system for controlling CPU resource allocation.

[0008] FIG. 4 illustrates an example operating environment of the example system of FIG. 3.

[0009] FIG. 5 illustrates an example operation process for controlling CPU resource allocation.

[0010] FIG. 6 illustrates a detailed example operation process for controlling CPU resource allocation.

[0011] FIG. 7 illustrates another detailed example operation process for controlling CPU resource allocation.

DETAILED DESCRIPTION

[0012] The disclosure provides a solution to determine a processor (CPU) utilization of a multiple processing system with operating system (OS) level virtualization. In the following detailed description, references are made to the accompanying drawings that form a part hereof, and in which are shown, by way of illustration, specific configurations or examples, in which like numerals represent like elements throughout the several figures.

[0013] FIG. 1 illustrates an example multiprocessing system 100 including an example symmetric multi-core processing unit 105 ("processing unit 105") including multiple (here for example, four) processor cores 110, each having their own caches "L1 caches" 120. Processor cores 110 also share a cache "L2 cache" 130. Each of processor cores 110 includes multi-threading capacity through multiple (here for illustrative example, two) hardware threads 140. In the description herein, processing unit 105 and the components and operations thereof are used as an example to illustrate the current technical solutions. It should be appreciated that the scope of the disclosure shall not be limited to the specific architecture of multiprocessing system 100 and/or processing unit 105 or any other specific multi-processing system architecture. For example, processor core 110 could be a lower level processor or CPU. In this disclosure, the term "processor" and "CPU" are used interchangeably and do not have any variation in definition and interpretation. Further, multiprocessing system 100 may include multiple tiers of multiprocessing unit 105 such that each processor core 110 itself may be a lower level multiprocessing unit 105 and/or each multiprocessing unit 105 may be

a processor core 110 of a higher level multiprocessing unit 105. For example, multiprocessing system 100 may include a hierarchical clustering structure with tight coupling of multiprocessing components (e.g., processing units 105 and/or processor cores 110) within a cluster. In another example, multiprocessing system 100 and/or processing unit 105 may be a virtual machine (hardware level virtualization) capable of virtual symmetric multiprocessing. In another example, multiprocessing system 100 is a symmetric multiprocessing system including multiple tiers of processing units 105 and processor cores 110. All now existing and future developed multiprocessing systems and architectures could be used with the current disclosure to determine processor utilization and to allocate computing resources, and all are included in the disclosure.

[0014] In the description herein, a "processor core" refers to any computing component that have multithreading capacity and/or multithreading processing, and a "processing unit" references a multiprocessing computing component having multiple processor cores. The terms "processor usage/utilization" and "CPU usage/utilization" are used interchangeably and all refer to a metric that indicates an amount of time that a processor is used during a certain period of time, usually represented as a percentage.

[0015] FIG. 2 illustrates an example computing system 200 including a multiprocessing processing unit 105 and operating system level virtualization. As shown in FIG. 2, computing system 200 includes hardware layer 210 and a host operating system (OS) layer 220. Hardware layer 210 may include one or more multi-processing unit 105. Virtualization layer 230 lies inside host OS layer 220 (i.e., OS level virtualization) and include virtual images (virtual machines) 240. Each virtual image 240 may include a virtual server 242 (e.g., a container) capable of operating multiple processes (PID) 244 (here each virtual server 242 is shown as having 2 PIDs

244 for illustrative purposes only). In the description herein, a virtual image, a virtual machine, a container, and/or a virtual server may be used interchangeably, each of which do not limit the scope of the disclosure. A virtual server 242 may be used herein as an example of a virtual image 240 for simplicity.

[0016] The software processes 244 operated through each virtual server 242 may be executed in hardware threads 140 of a same processor core 110 or may be executed by hardware threads 140 of different processor cores 110. As illustrated in FIG. 2 as examples, two PIDs 244A of virtual server 242A are executed by hardware threads 140A of a same processor core 110A; and a PID 244B of virtual server 242B and a PID 244C of virtual server 242C are each executed through one hardware thread 140B of processor core 110B.

[0017] Further, software processes PID 244 of virtual servers 242 may migrate between different hardware threads 140 and/or processor cores 110 due to the complex event registration and allocation schemes of computing system 100/200. Such complexity may be further exaggerated in the case of multiple levels of virtualizations, e.g., a combination of CPU virtualization and OS level virtualization.

1. Example Systems and Environments

[0018] FIG. 3 illustrates a system diagram showing aspects of an illustrative example computing system 300 for controlling computing resource allocation. As shown in FIG. 3, computing system 300 may include a memory unit 310 containing computer executable instructions, which when executed by a processing unit, configures the processing unit to implement a computing resource allocation system 320. Computing resource allocation system 320 may include a PID identification unit 322, a VM determination unit 330, a CPU usage determination unit 332 and a control

unit 334. PID identification unit 322 may further include an event detection unit 324, a multi-threading identification unit 326 and a registration retrieval unit 328.

[0019] Computing system 300 may also include one or more processing units (PU) 340, one or more interfacing units 350, one or more networking units 360 and other components 370.

[0020] It should be appreciated that units of system 300 may reside on a single computing device, e.g., a PC, or in multiple computing devices in a distributed computing environment/system, and all are included in the disclosure. Details of a computing device or a distributed computing environment is not required to understand the disclosure and do not limit the scope of the disclosure.

[0021] FIG. 4 illustrates an example operation environment 400 of the computing resource allocation system 320. As shown in FIG. 4, computing resource allocation system 320 may communicate with multiple computing systems 200 through network 420. One or more of computing systems 200 may include a multi-threading processing core architecture(s) and OS level virtualization(s), e.g., containers. FIG. 4 illustrates that computing resource allocation system 320 functions remotely with computing systems 200. But the resource allocation system 320 may also physically reside within/together with a computing system 200.

[0022] In operation, PID identification unit 322 may be generally configured to identify multiple software processes 244 operated through a same processor core 110 during a period of time. Any approaches may be used in this identification and all are included in the disclosure. In an example, the multiple software processes 244 operated through the processor core 110 during a period of time includes detecting that the multiple software processes share a hardware resource, e.g., a cache, of the processor core 110. In another example, PID identification unit 322 may identify multiple

software processes 244 that are operated simultaneously at a same level of a multi-threading architecture under the processor core 110.

[0023] Within PID identification unit 322, event detection unit 324 may be configured to detect a registration event of a software process 244 being registered with a processor core 110 during a sampling interval or to detect a termination event of a software process 244 being terminated with a processor core 110 during a sampling interval. The determination of registration/termination status of a software process may be event based. That is, PID identification unit 322 and the units thereof may not continuously monitor the registration and/or termination status of software processes 244 with the processor core(s) 110 and/or the hardware threads 140 thereof. Instead, the registration/termination status of a software process 244 is checked upon an event being detected by event detection unit 324.

[0024] Multi-threading identification unit 326 may be configured to identify all co-existing software processes 244 operated through a processor core 110. In an example, the identifying all co-existing software processes 244 operated through a processor core 110 includes detecting that the multiple software processes 244 share a hardware resource, e.g., L1 cache 120, of the processor core 110.

[0025] Registration retrieval unit 328 may be configured to determine a most recent registration time of a software process 244 with a processor core 110. After a registration event of a software process 244 is detected by event detection unit 324, a registration time may be recorded and stored. As is appreciated, a software process 244 may register and/or terminate with multiple processor cores 110 sequentially and may migrate from one processor core 110 to another. Registration retrieval unit 328 may retrieve such stored registration time(s) to determine a most recent registration time of a software process 244 with a processor core 110.

[0026] VM determination unit 330 may be configured to determine one or more operating system (OS) level virtual images 240 (including, e.g., virtual servers 242) associated with the multiple software processes 244. For example, as illustrative in FIG. 2, processes PID 244B and 244C are co-existing software processes through processor core 110B, and virtual servers 242B and 242C are determined as associated to software processes 244B and 244C, respectively.

[0027] CPU usage determination unit 332 may be configured to determine CPU usage of each of the one or more OS level virtual machines 242 for a time period based on a count of the one or more OS level virtual machines 242. For example, CPU usage determination unit 332 may attribute the CPU usage evenly to each of the one or more OS level virtual machines 242 for the time period. Therefore, each of the one or more OS level virtual machines 242 may be attributed with a same amount of CPU usage for the time period. That is, the CPU usage is determined on the container level instead of the software process level. If two OS level virtual machines 242 each have at least one software process 244 operated through a same processor core 110 during a period of time, the two OS level virtual machines will be attributed with same amount of CPU usage, even if the two OS level virtual machines may have different number of software processes 244 simultaneously operated through the same processor core 110.

[0028] Control unit 334 may be configured to control an allocation of CPU registration of a processor core 110 based on the determined CPU usage of the one or more OS level virtual machines with the processor core 110.

2. Example Operation Flows

[0029] Referring now to FIG. 5, which illustrates an example flow chart of a process 500 of an example operation of system 300. In example operation 510, PID

identification unit 322 may identify multiple software processes 244 operated through a same processor core 110 during a first time period. Any approaches may be used in the identification and all are included in the disclosure. In an example, the identifying the multiple software processes 244 operated through the same processor core 110 during the first time period includes detecting that the multiple software processes 244 share a hardware resource of the processor core 110. In another example, PID identification unit 322 may identify multiple software processes 244 that are operated simultaneously at a same level of a multi-threading architecture under the processor core 110. For illustrative example, with respect to the example of FIG. 2, PID identification unit 322 may identify two software processes 244A operated through a same processor core 110A and software processes 244B and 244C operated through a same processor core 110B. A first time period may be set by any approaches and all are included. In an example, the first time period may be included within a sampling interval and may fully overlap with the sampling interval.

[0030] In example operation 520, VM determination unit 330 may determine one or more operating system (OS) level virtual images/virtual machines 240 (including, e.g., virtual servers 242) associated with the multiple software processes 244. Any approaches may be used to determine the virtual machine(s) 242 associated to the software processes 244 and all are included in the disclosure. As illustrative in FIG. 2, for illustrative example, software processes PID 244B and PID 244C are associated to virtual servers 242B and 242C and two PIDs 244A are associated to virtual server 242A.

[0031] It should be appreciated that any and all approach(es) for collecting data for the purposes of example operations 510, 520 may be used and all are included in the disclosure. For example, Windows-based and/or Unix-based real time performance monitoring tools and/or log-based performance monitoring tools may be used to collect

system operation/CPU registration data for processes 244 and/or virtual machines 242 with regarding to threads 140 and/or processor cores 110. The disclosure is not limited to any specific way of data collection and/or the data collected to be used in the operation(s).

[0032] In example operation 530, CPU usage determination unit 332 may determine CPU usage of each of the one or more OS level virtual machines 240 for the first time period based on a count of the one or more OS level virtual machines 240. For example, CPU usage determination unit 332 may attribute evenly the CPU usage to each of the one or more OS level virtual machines 240 for the first time period. For illustrative example, in FIG. 2, OS level virtual servers 242B and 242C each includes at least one software process 244 (PID 244B and PID 244C) operated through a same processor core 110B during a period of time. Based on the count of virtual machines 240, i.e., two, the two OS level virtual machines 240 will be attributed with same amount of CPU usage, e.g., 50% of the CPU usage of processor core 110B for the period of time. Following also the example of FIG. 2, two software processes PIDs 244A of virtual server 242A are operated with the same processor core 110A for the period of time. Based on the count of virtual machines involved, here, one virtual machine, 100% of CPU usage of processor core 110A may be attributed to virtual server 242A for the period of time.

[0033] In example operation 540, control unit 334 may control an allocation of CPU registration of a processor core 110 based on the determined CPU usage of the one or more OS level virtual machines 240/242. Control unit 334 may use the determined CPU usage of a virtual machine 242 in various ways to control the allocation of CPU registration to a virtual machine 242, and all are included in the disclosure. For example, in a case that a virtual machine 242 has a set amount of CPU

usage as part of a container agreement, the determined CPU usage (and/or the accumulation thereof) may be used to determine the remaining computing resources accessible by the virtual machine 242. For another example, the determined CPU usage (and/or the accumulation thereof) may be used to prioritize and/or de-prioritize the access to hardware computing resources entitled to a virtual machine 242.

[0034] FIG. 6 shows a detailed operating process of computing resource allocation system 320. In example operation 610, event detection unit 324 may detect a registration event of a software process 244 being registered with a processor core 110 during a sampling interval. For an illustrative example, at time point t_1 within a sampling interval t_0 to t_2, event detection unit 324 may detect software process PID 244B registered with processor core 110B. In an example, a sampling interval is set to be smaller than a threshold time interval, during which a software process 244 is capable of migrating from one hardware thread 140 to another hardware thread 140.

[0035] In example operation 620, event detection unit 324 may identify a first portion (e.g., t_1 to t_2) of the sampling interval (e.g., t_0 to t_2) after the registration event (e.g., at t_1) as the first time period.

[0036] In example operation 630, multi-threading identification unit 326 may identify all co-existing software process 244 operated through a same processor core 110 as the newly registered software process 244. In an example, all co-existing software processes 244 operated through a processor core 110 includes detecting that the software processes 244 share a hardware resource, e.g., a cache, of the processor core 110. Here, following the example, multi-threading identification unit 326 may identify that software process PID 244C is a co-existing software process operated through the same processor core 110B as the newly registered software process PID 244B.

[0037] In example operation 640, it may be determined whether the newly registered software process 244 and the identified co-existing software process 244 are associated to the same virtual machine 242. As shown in FIG. 5, example operation 520 may determine one or more OS level virtual machines 242 associated with the multiple software processes 244. Here, following the example, it may be determined that software process PID 244B is associated to virtual server 242B and software process PID 244C is associated to virtual server 242C. The two software processes 244 are not associated to a same virtual machine 242.

[0038] In the case that the two software processes are not associated to a same virtual machine 242, the process goes to example operation 650, where registration retrieval unit 328 may determine a most recent registration time of the co-existing software process 244 with the processor core 110. In an example, the determining the most recent registration time of the co-existing software process 244 with the processor core 110 is restricted within the sampling interval.

[0039] Following the illustrative example, for co-existing software process 244C, registration retrieval unit 328 may retrieve a most recent registration time t_r with the processor core 110B. if t_r is earlier than t_0, the starting point of the relevant sampling interval, t_0, may be used as the most recent registration time of co-existing software process 244C.

[0040] In example operation 660, CPU usage determination unit 332 may determine a CPU usage of a virtual machine based on a count of virtual machines involved in the co-existing software processes 244 (the newly registered software process 244 is a co-existing software process itself). For the first time period, t_1 to t_2, the CPU usage of the processor core 110B is attributed at least to the OS level virtual machine 242C of the co-existing software process PID 244C and the OS level virtual

machine 242B of the newly registered software process PID 244B. For example, for the first time period, t_1 to t_2, a same amount of 50% of the CPU usage of processor core 110B may be attributed to each of virtual server 242B and virtual server 242C.

[0041] For a second portion of the sampling interval before the registration event (t_1) of the newly registered software process PID 244B and after the most recent registration time (t_0) of the co-existing software process PID 244C, i.e., t_0 to t_1, the CPU usage of the processor core 110B is attributed at least to OS level virtual machine 242C of the co-existing software process PID 244C.

[0042] FIG. 7 shows another detailed operation process of computing resource allocation system 320. In example operation 710, event detection unit 324 may detect a termination event of a software process 244 being terminated with a processor core 110 during a sampling interval. For an illustrative example, at time point t_1 within a sampling interval t_0 to t_2, event detection unit 324 may detect software process PID 244A being terminated with processor core 110A, with reference also to FIG. 2.

[0043] In example operation 720, event detection unit 324 may identify a first portion (e.g., t_0 to t_1) of the sampling interval (e.g., t_0 to t_2) before the termination event (e.g., at t_1) as the first time period.

[0044] In example operation 730, multi-threading identification unit 326 may identify all co-existing software process 244 operated through a same processor core 110 as the terminated software process 244. In an example, the identifying all co-existing software processes 244 operated through a processor core 110 includes detecting that the multiple software processes 244 share a hardware resource of the processor core 110. Here, following the example, multi-threading identification unit 326 may identify that another software process PID 244A is a co-existing software

process operated through the same processor core 110A together with the newly terminated software process PID 244A.

[0045] In example operation 740, it may be determined whether the newly terminated software process 244 and the identified co-existing software process 244 are associated to the same virtual machine 242. As shown in FIG. 5, example operation 520 may determine one or more OS level virtual machines 242 associated with the multiple software processes 244. Here, following the example, it may be determined that the newly terminated software process PID 244A and the co-existing software process PID 244A are associated to the same virtual server 242A.

[0046] In the case that the newly terminated software process 244 is not associated with a same OS level virtual machine 242 as any of the co-existing software process(s) 244, the operation process may proceed to example operation 750. In example operation 750, the OS level virtual machine 242 of the terminated software process 244 is removed from the count of the one or more OS level virtual machines 242 in attributing CPU usage of the processor core 110. For illustrative example, with reference to FIG. 2, software processes PID 244B and PID 244C are co-existing software processes operated through processor core 110B. In a case that PID 244B is terminated with processor core 110B, virtual server 242B, to which PID 244B is associated, may be removed from the count of OS level virtual servers 242 and 100% of the CPU usage of processor core 110B may be attributed to virtual server 242C of PID 244C.

[0047] In the case that the newly terminated software process 244 is associated to a same virtual machine 242 with at least one co-existing software process 244, in example operation 760, a same portion of CPU usage may be attributed to the same OS level virtual machine 242 for the first time period. For illustrative example, even if one

of software process PID 244A is terminated with processor core 110A, another software PID 244A of virtual server 242A is still operated with processor core 110A, and therefore the same 100% of CPU usage is attributed to virtual server 242A.

[0048] The processes described above in association with FIGS. 5-7 can be implemented in hardware, software, or a combination thereof. In the context of software, the operations represent computer-executable instructions stored on one or more computer-readable storage media that, when executed by one or more processors, perform the recited operations. Generally, computer-executable instructions include routines, programs, objects, components, data structures, and the like that perform particular functions or implement particular abstract data types. In other embodiments, hardware components perform one or more of the operations. Such hardware components may include or be incorporated into processors, application-specific integrated circuits (ASICs), programmable circuits such as field programmable gate arrays (FPGAs), or in other ways. The order in which the operations are described is not intended to be construed as a limitation, and any number of the described operations can be combined in any order and/or in parallel to implement the processes.

[0049] The memory may include computer readable media such as a volatile memory, a Random Access Memory (RAM), and/or non-volatile memory, e.g., Read-Only Memory (ROM) or flash RAM, and so on. The memory is an example of a computer readable medium.

[0050] Computer readable media include non-volatile, volatile, mobile and non-mobile media, and can implement information storage through any method or technology. The information may be computer readable instructions, data structures, program modules or other data. Examples of storage media of a computer include, but not limited to, Phase-change RAMs (PRAMs), Static RAMs (SRAMs), Dynamic

RAMs (DRAMs), other types of RAMs, ROMs, Electrically Erasable Programmable Read-Only Memories (EEPROMs), flash memories or other memory technologies, Compact Disk Read-Only Memories (CD-ROMs), Digital Versatile Discs (DVDs) or other optical memories, cassettes, cassette and disk memories or other magnetic memory devices or any other non-transmission media, and can be used for storing information accessible to the computation device. According to the definitions herein, the computer readable media exclude transitory media, such as modulated data signals and carriers.

[0051] It should be further noted that, the terms "include", "comprise", or any variants thereof are intended to cover a non-exclusive inclusion, such that a process, a method, a product, or a device that includes a series of elements not only includes such elements but also includes other elements not specified expressly, or may further include inherent elements of the process, method, product, or device. In the absence of more restrictions, an element limited by "include a/an..." does not exclude other same elements existing in the process, method, product, or device that includes the element.

[0052] Described above are merely the examples of the present application, which are not used to limit the present application. For those skilled in the art, the present application may have various alterations and changes. Any modification, equivalent replacement, improvement, and the like made within the spirit and principle of the present application shall be included in the scope of the claims of the present application.

[0053] The disclosure may be appreciated with the following clauses:

[0054] Clause 1: a computer-implemented system, comprising: one or more processors; and computer readable storage medium communicatively coupled with the one or more processors, the computer readable medium having computer executable

instructions stored therein, which when executed, configure the one or more processors to implement a computing resource allocation system operable to: identify multiple software processes operated through a processor core during a first time period; determine one or more operating system (OS) level virtual machines associated with the multiple software processes; and determine CPU usage of each of the one or more OS level virtual machines for the first time period based on a count of the one or more OS level virtual machines.

[0055] Clause 2: the computer-implemented system of clause 1, wherein the computing resource allocation system is further operable to control an CPU registration of the processor core based on the determined CPU usage of the one or more OS level virtual machines.

[0056] Clause 3: the computer-implemented system of clause 1, wherein the identifying the multiple software processes includes: detecting a registration event of a software process being newly registered with the processor core during a sampling interval; identifying a first portion of the sampling interval after the registration event as the first time period; identifying a co-existing software process operated through the processor core; determining whether the co-existing software process and the newly registered software process are associated with a same OS level virtual machine; and in a case that the co-existing software process and the newly registered software process are not associated with the same OS level virtual machine, determining a most recent registration time of the co-existing software process with the processor core.

[0057] Clause 4: the computer-implemented system of clause 3, wherein the determining the most recent registration time of the co-existing software process with the processor core is restricted within the sampling interval.

[0058] Clause 5: the computer-implemented system of clause 3, wherein: for a second portion of the sampling interval before the registration event and after the most recent registration time of the co-existing software process, the CPU usage of the processor core is attributed at least to an OS level virtual machine of the co-existing software process; and for the first time period, the CPU usage of the processor core is attributed at least to the OS level virtual machine of the co-existing software process and an OS level virtual machine of the newly registered software process.

[0059] Clause 6: the computer-implemented system of clause 1, wherein the determining the CPU usage of each of the one or more OS level virtual machines for the first time period based on the count of the one or more OS level virtual machines includes attributing evenly the CPU usage to each of the one or more OS level virtual machines for the first time period.

[0060] Clause 7: the computer-implemented system of clause 1, wherein the multiple software processes are operated simultaneously at a same level of a multi-threading architecture under the processor core.

[0061] Clause 8: the computer-implemented system of clause 1, wherein the identifying the multiple software processes operated through the processor core during the first time period includes detecting that the multiple software processes share a hardware resource of the processor core.

[0062] Clause 9: the computer-implemented system of clause 3, wherein the sampling interval is set to be smaller than a threshold time interval, during which a software process is capable of migrating from one hardware thread to another hardware thread.

[0063] Clause 10: the computer-implemented system of clause 1, wherein the computing system includes a symmetric multiprocessing system including multiple tiers of processing units and processor cores.

[0064] Clause 11: the computer-implemented system of clause 1, wherein identifying the multiple software processes includes: detecting a termination event of a software process being terminated with the processor core during a sampling interval; identifying a first portion of the sampling interval before the termination event as the first time period; identifying a co-existing software process operated through the processor core; and determining whether the co-existing software process and the terminated software process are associated with a same OS level virtual machine.

[0065] Clause 12: the computer-implemented system of clause 11, wherein in a case that the co-existing software process and the terminated software process are associated with the same OS level virtual machine, a same portion of CPU usage is attributed to the same OS level virtual machine for a second portion of the sampling interval after the termination event as for the first time period.

[0066] Clause 13: the computer-implemented system of clause 11, wherein in a case that none of the co-existing software process is associated with an OS level virtual machine as the terminated software process, the OS level virtual machine of the terminated software process is removed from the count of the one or more OS level virtual machines in attributing CPU usage of the processor core.

[0067] Clause 14: a method, comprising: identifying multiple software processes operated through a processor core of a computing system during a first time period; determining one or more operating system (OS) level virtualization images associated with the multiple software processes; and attributing usage of a computing

resource associated to the processor core evenly among the one or more OS level virtualization images.

[0068] Clause 15: the method of clause 14, wherein the multiple software processes includes: detecting a registration event of a software process being newly registered with the processor core during a sampling interval; identifying a first portion of the sampling interval after the registration event within the first time period; identifying a co-existing software process operated through the processor core; determining whether the co-existing software process and the newly registered software process are associated with a same OS level virtualization image; and in a case that the co-existing software process and the newly registered software process are not associated with the same OS level virtualization image, determining a most recent registration time of the co-existing software process with the processor core.

[0069] Clause 16: the method of clause 14, wherein: for a second portion of the sampling interval before the registration event and after the most recent registration time of the co-existing software process, the CPU usage of the processor core is attributed at least to an OS level virtualization image of the co-existing software process; and for the first time period, the CPU usage of the processor core is attributed at least to the OS level virtualization image of the co-existing software process and an OS level virtualization image of the newly registered software process.

[0070] Clause 17: the method of clause 14, wherein the identifying the multiple software processes operated through the processor core during the first time period includes detecting that the multiple software processes share a hardware resource of the processor core.

[0071] Clause 18: the method of clause 14, wherein the identifying the multiple software processes includes: detecting a termination event of a software process being

terminated with the processor core during a sampling interval; identifying a first portion of the sampling interval before the termination event as the first time period; identifying a co-existing software process operated through the processor core; and determining whether the co-existing software process and the terminated software process are associated with a same OS level virtualization image.

[0072] Clause 19: the method of clause 18, wherein in a case that the co-existing software process and the terminated software process are associated with the same OS level virtualization image, a same portion of CPU usage is attributed to the same OS level virtualization image machine for a second portion of the sampling interval after the termination event as for the first time period.

[0073] Clause 20: a server for a data center application, the server comprising: a processor; and a memory communicatively coupled with the processor, the memory having computer executable instructions stored therein, which when executed, configure the processor to implement operations including: identifying multiple software processes operated through a processor core executing the data center application during a first time period; determining one or more containers associated with the multiple software processes; and attributing a same amount of system resource usage to each of the one or more containers for the first time period.

[0074] Clause 21: a method, comprising: identifying multiple software processes operated through a processor core of a computing system during a time period; determining one or more containers associated with the multiple software processes; determining CPU usage of each of the one or more containers for the time period based on a count of the containers; and controlling allocation of CPU resource based on the determined CPU usage of the containers.

[0075] Clause 22: the method of clause 21, wherein the controlling the allocation of the CPU resource based on the determined CPU usage of the containers comprising: controlling allocation of CPU registration based on the determined CPU usage of the containers.

[0076] Clause 23: the method of clause 22, wherein the CPU registration includes registering a software process with the processor core.

What is Claimed is:

1. A computer-implemented system, comprising:

one or more processors; and

computer readable storage medium communicatively coupled with the one or more processors, the computer readable medium having computer executable instructions stored therein, which when executed, configure the one or more processors to implement a computing resource allocation system operable to:

identify multiple software processes operated through a processor core during a first time period;

determine one or more operating system (OS) level virtual machines associated with the multiple software processes; and

determine CPU usage of each of the one or more OS level virtual machines for the first time period based on a count of the one or more OS level virtual machines.

2. The computer-implemented system of claim 1, wherein the computing resource allocation system is further operable to control an CPU registration of the processor core based on the determined CPU usage of the one or more OS level virtual machines.

3. The computer-implemented system of claim 1, wherein the identifying the multiple software processes includes:

detecting a registration event of a software process being newly registered with the processor core during a sampling interval;

identifying a first portion of the sampling interval after the registration event as the first time period;

identifying a co-existing software process operated through the processor core;

determining whether the co-existing software process and the newly registered software process are associated with a same OS level virtual machine; and

in the case that the co-existing software process and the newly registered software process are not associated with the same OS level virtual machine, determining a most recent registration time of the co-existing software process with the processor core.

4. The computer-implemented system of claim 3, wherein the determining the most recent registration time of the co-existing software process with the processor core is restricted within the sampling interval.

5. The computer-implemented system of claim 3, wherein:

for a second portion of the sampling interval before the registration event and after the most recent registration time of the co-existing software process, the CPU usage of the processor core is attributed at least to an OS level virtual machine of the co-existing software process; and

for the first time period, the CPU usage of the processor core is attributed at least to the OS level virtual machine of the co-existing software process and an OS level virtual machine of the newly registered software process.

6. The computer-implemented system of claim 1, wherein the determining the CPU usage of each of the one or more OS level virtual machines for the first time period based on the count of the one or more OS level virtual machines includes

attributing evenly the CPU usage to each of the one or more OS level virtual machines for the first time period.

7. The computer-implemented system of claim 1, wherein the multiple software processes are operated simultaneously at a same level of a multi-threading architecture under the processor core.

8. The computer-implemented system of claim 1, wherein the identifying the multiple software processes operated through the processor core during the first time period includes detecting that the multiple software processes share a hardware resource of the processor core.

9. The computer-implemented system of claim 3, wherein the sampling interval is set to be smaller than a threshold time interval, during which a software process is capable of migrating from one hardware thread to another hardware thread.

10. The computer-implemented system of claim 1, wherein the computing system includes a symmetric multiprocessing system including multiple tiers of processing units and processor cores.

11. The computer-implemented system of claim 1, wherein the identifying the multiple software processes includes:

detecting a termination event of a software process being terminated with the processor core during a sampling interval;

identifying a first portion of the sampling interval before the termination event as the first time period;

identifying a co-existing software process operated through the processor core; and

determining whether the co-existing software process and the terminated software process are associated with a same OS level virtual machine.

12. The computer-implemented system of claim 11, wherein in a case that the co-existing software process and the terminated software process are associated with the same OS level virtual machine, a same portion of CPU usage is attributed to the same OS level virtual machine for a second portion of the sampling interval after the termination event as for the first time period.

13. The computer-implemented system of claim 11, wherein in a case that none of the co-existing software process is associated with an OS level virtual machine as the terminated software process, the OS level virtual machine of the terminated software process is removed from the count of the one or more OS level virtual machines in attributing CPU usage of the processor core.

14. A method, comprising:

identifying multiple software processes operated through a processor core of a computing system during a first time period;

determining one or more operating system (OS) level virtualization images associated with the multiple software processes; and

attributing usage of a computing resource associated to the processor core evenly among the one or more OS level virtualization images.

15. The method of claim 14, wherein the identifying the multiple software processes includes:

detecting a registration event of a software process being newly registered with the processor core during a sampling interval;

identifying a first portion of the sampling interval after the registration event as the first time period;

identifying a co-existing software process operated through the processor core;

determining whether the co-existing software process and the newly registered software process are associated with a same OS level virtualization image; and

in the case that the co-existing software process and the newly registered software process are not associated with the same OS level virtualization image, determining a most recent registration time of the co-existing software process with the processor core.

16. The method claim 14, wherein:

for a second portion of the sampling interval before the registration event and after the most recent registration time of the co-existing software process, the CPU usage of the processor core is attributed at least to an OS level virtualization image of the co-existing software process; and

for the first time period, the CPU usage of the processor core is attributed at least to the OS level virtualization image of the co-existing software process and an OS level virtualization image of the newly registered software process.

17. The method of claim 14, wherein the identifying the multiple software processes operated through the processor core during the first time period includes detecting that the multiple software processes share a hardware resource of the processor core.

18. The method of claim 14, wherein the identifying the multiple software processes includes:

detecting a termination event of a software process being terminated with the processor core during a sampling interval;

identifying a first portion of the sampling interval before the termination event as the first time period;

identifying a co-existing software process operated through the processor core; and

determining whether the co-existing software process and the terminated software process are associated with a same OS level virtualization image.

19. The method of claim 18, wherein in a case that the co-existing software process and the terminated software process are associated with the same OS level virtualization image, a same portion of CPU usage is attributed to the same OS level virtualization image machine for a second portion of the sampling interval after the termination event as for the first time period.

20. A server for a data center application, the server comprising:

a processor; and

a memory communicatively coupled with the processor, the memory having computer executable instructions stored therein, which when executed, configure the processor to implement operations including:

identifying multiple software processes operated through a processor core executing the data center application during a first time period;

determining one or more containers associated with the multiple software processes; and

attributing a same amount of system resource usage to each of the one or more containers for the first time period.

21. A method, comprising:

identifying multiple software processes operated through a processor core of a computing system during a time period;

determining one or more containers associated with the multiple software processes;

determining CPU usage of each of the one or more containers for the time period based on a count of the containers; and

controlling allocation of CPU resource based on the determined CPU usage of the containers.

22. The method of claim 21, wherein the controlling the allocation of the CPU resource based on the determined CPU usage of the containers comprising:

controlling allocation of CPU registration based on the determined CPU usage of the containers.

23. The method of claim 22, wherein the CPU registration includes registering a software process with the processor core.

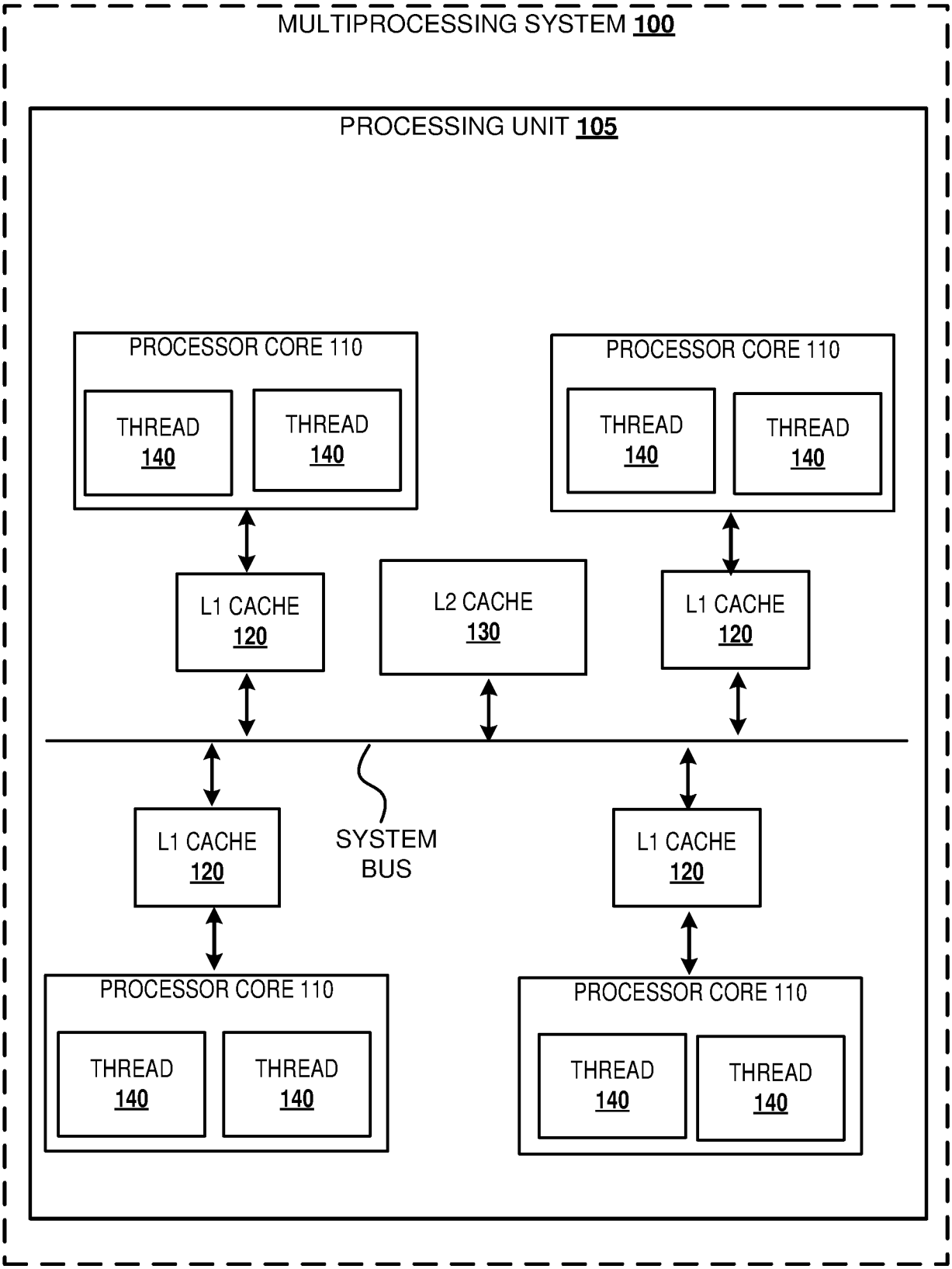

MULTIPROCESSING SYSTEM 100
PROCESSING UNIT 105
PROCESSOR CORE 110
THREAD 140
THREAD 140
PROCESSOR CORE 110
THREAD 140
THREAD 140
L1 CACHE 120
L2 CACHE 130
L1 CACHE 120
L1 CACHE 120
SYSTEM BUS
L1 CACHE 120
PROCESSOR CORE 110
THREAD 140
THREAD 140
PROCESSOR CORE 110
THREAD 140
THREAD 140

FIG. 1

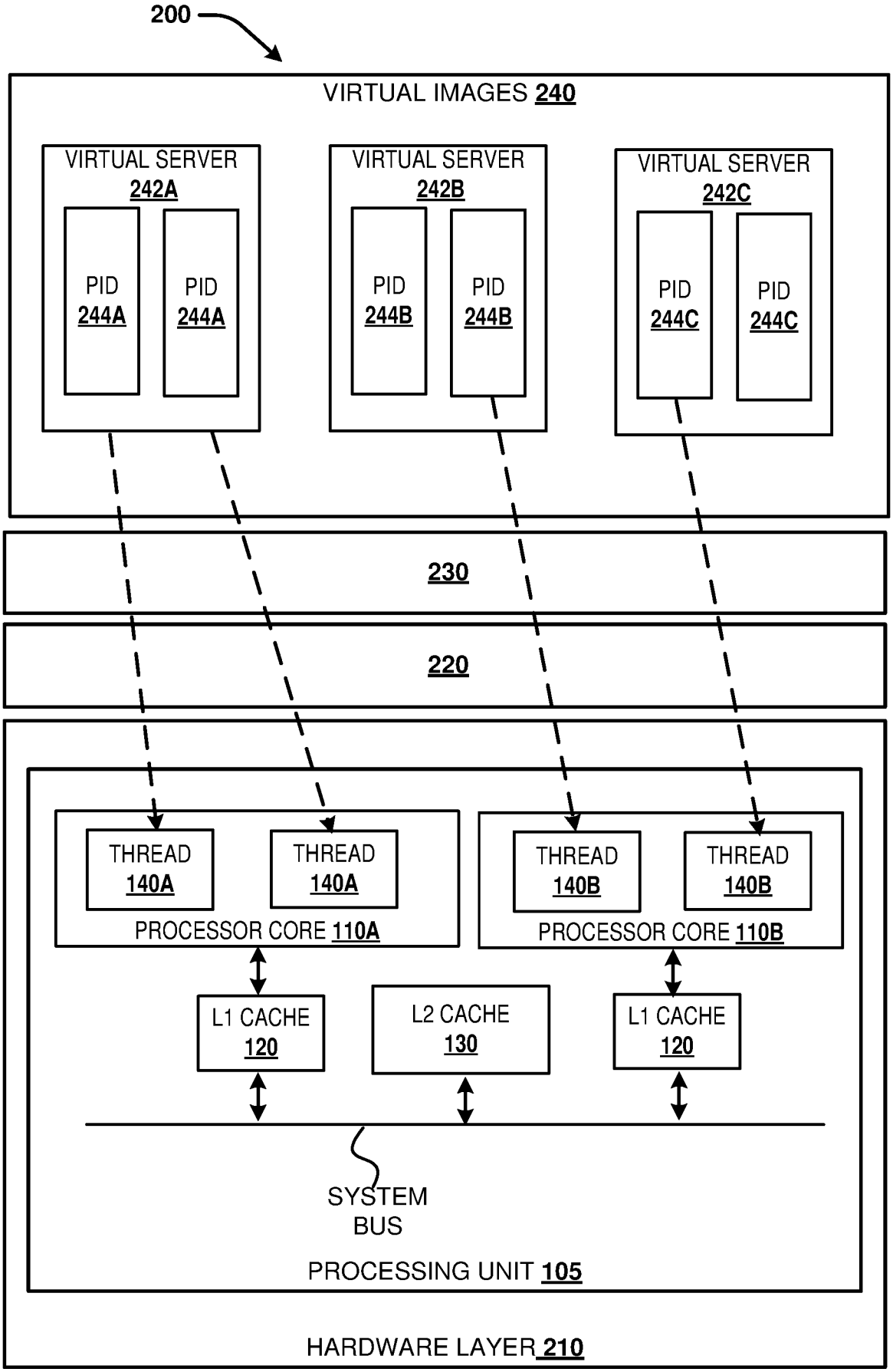

200
VIRTUAL IMAGES 240
VIRTUAL SERVER 242A
PID 244A
PID 244A
VIRTUAL SERVER 242B
PID 244B
PID 244B
VIRTUAL SERVER 242C
PID 244C
PID 244C
230
220
THREAD 140A
THREAD 140A
PROCESSOR CORE 110A
THREAD 140B
THREAD 140B
PROCESSOR CORE 110B
L1 CACHE 120
L2 CACHE 130
L1 CACHE 120
SYSTEM BUS
PROCESSING UNIT 105
HARDWARE LAYER 210

FIG. 2

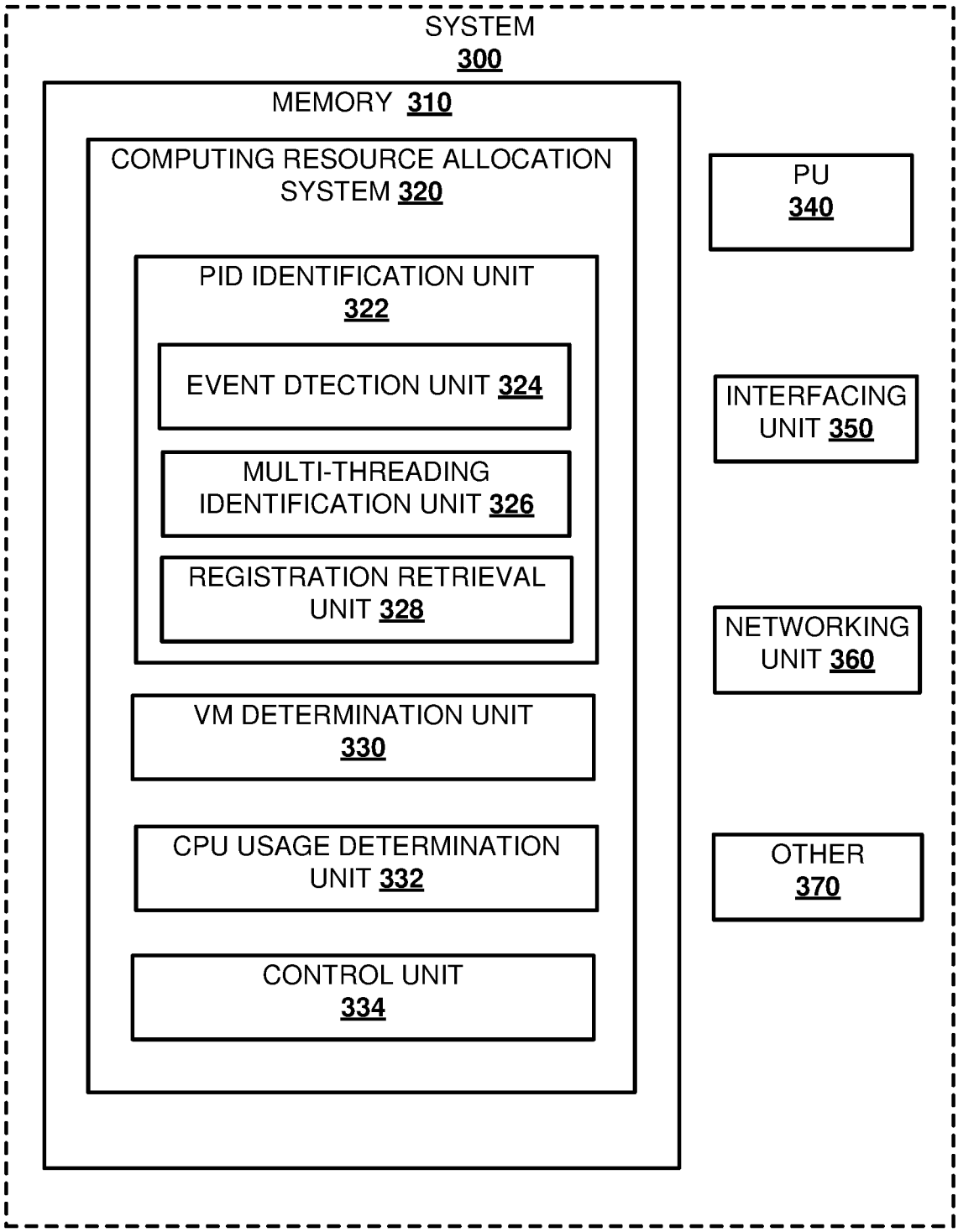

SYSTEM
300
MEMORY 310
COMPUTING RESOURCE ALLOCATION SYSTEM 320
PID IDENTIFICATION UNIT 322
EVENT DTECTION UNIT 324
MULTI-THREADING IDENTIFICATION UNIT 326
REGISTRATION RETRIEVAL UNIT 328
VM DETERMINATION UNIT 330
CPU USAGE DETERMINATION UNIT 332
CONTROL UNIT 334
PU 340
INTERFACING UNIT 350
NETWORKING UNIT 360
OTHER 370

FIG. 3

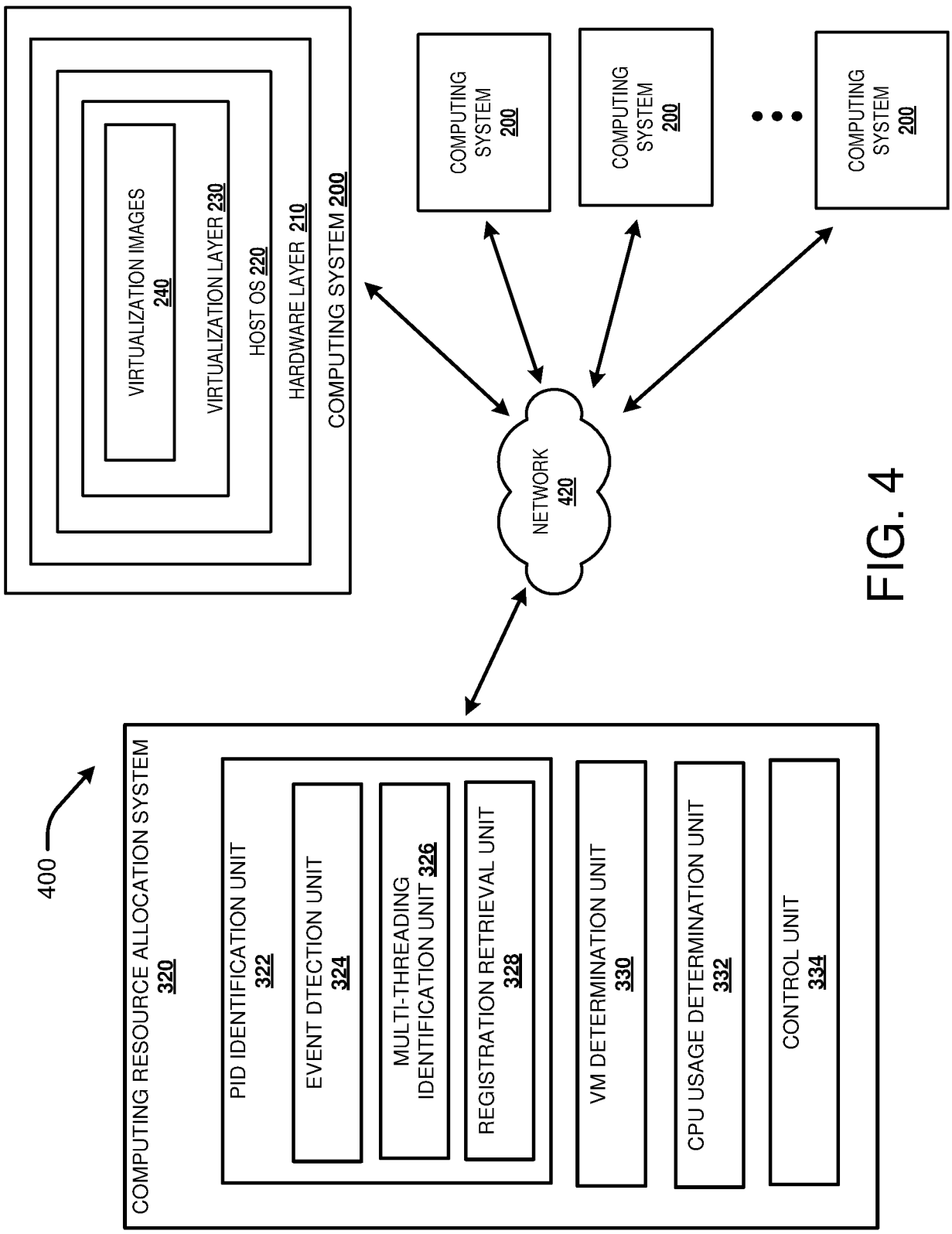

400
COMPUTING RESOURCE ALLOCATION SYSTEM 320
PID IDENTIFICATION UNIT 322
EVENT DTECTION UNIT 324
MULTI-THREADING IDENTIFICATION UNIT 326
REGISTRATION RETRIEVAL UNIT 328
VM DETERMINATION UNIT 330
CPU USAGE DETERMINATION UNIT 332
CONTROL UNIT 334
NETWORK 420
COMPUTING SYSTEM 200
COMPUTING SYSTEM 200
COMPUTING SYSTEM 200
COMPUTING SYSTEM 200
VIRTUALIZATION IMAGES 240
VIRTUALIZATION LAYER 230
HOST OS 220
HARDWARE LAYER 210

FIG. 4

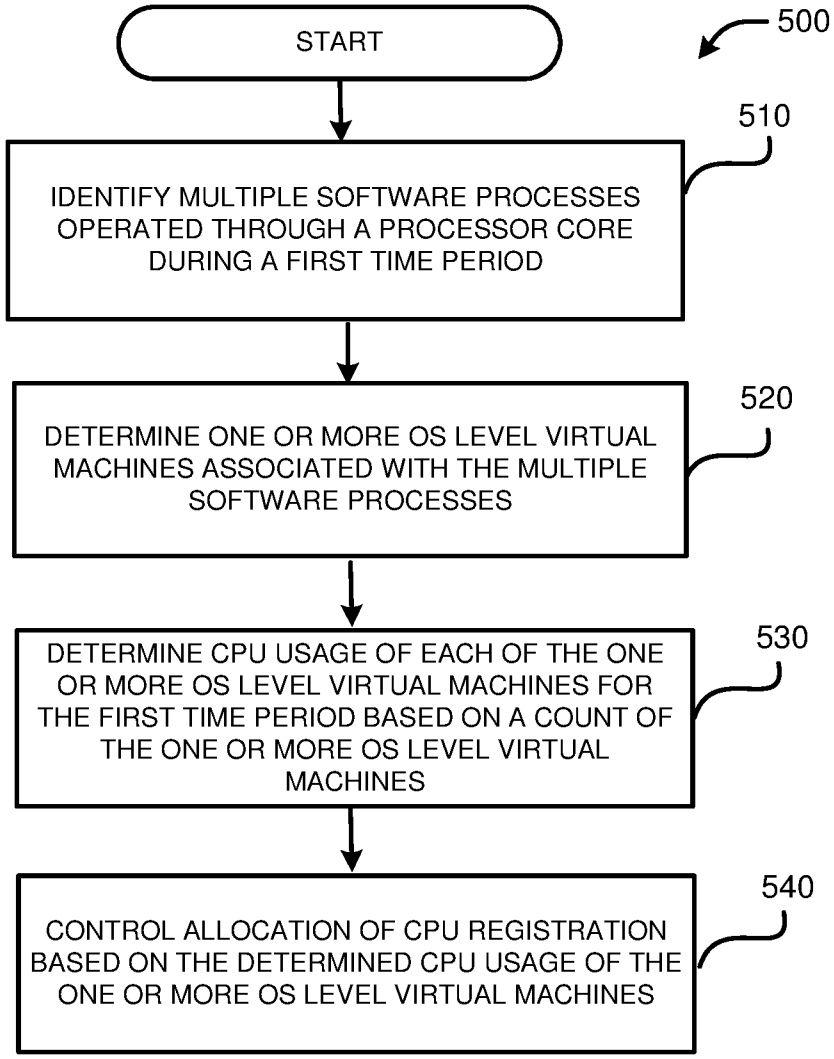

START
500
510
IDENTIFY MULTIPLE SOFTWARE PROCESSES OPERATED THROUGH A PROCESSOR CORE DURING A FIRST TIME PERIOD
520
DETERMINE ONE OR MORE OS LEVEL VIRTUAL MACHINES ASSOCIATED WITH THE MULTIPLE SOFTWARE PROCESSES
530
DETERMINE CPU USAGE OF EACH OF THE ONE OR MORE OS LEVEL VIRTUAL MACHINES FOR THE FIRST TIME PERIOD BASED ON A COUNT OF THE ONE OR MORE OS LEVEL VIRTUAL MACHINES
540
CONTROL ALLOCATION OF CPU REGISTRATION BASED ON THE DETERMINED CPU USAGE OF THE ONE OR MORE OS LEVEL VIRTUAL MACHINES

FIG. 5

FIG. 6

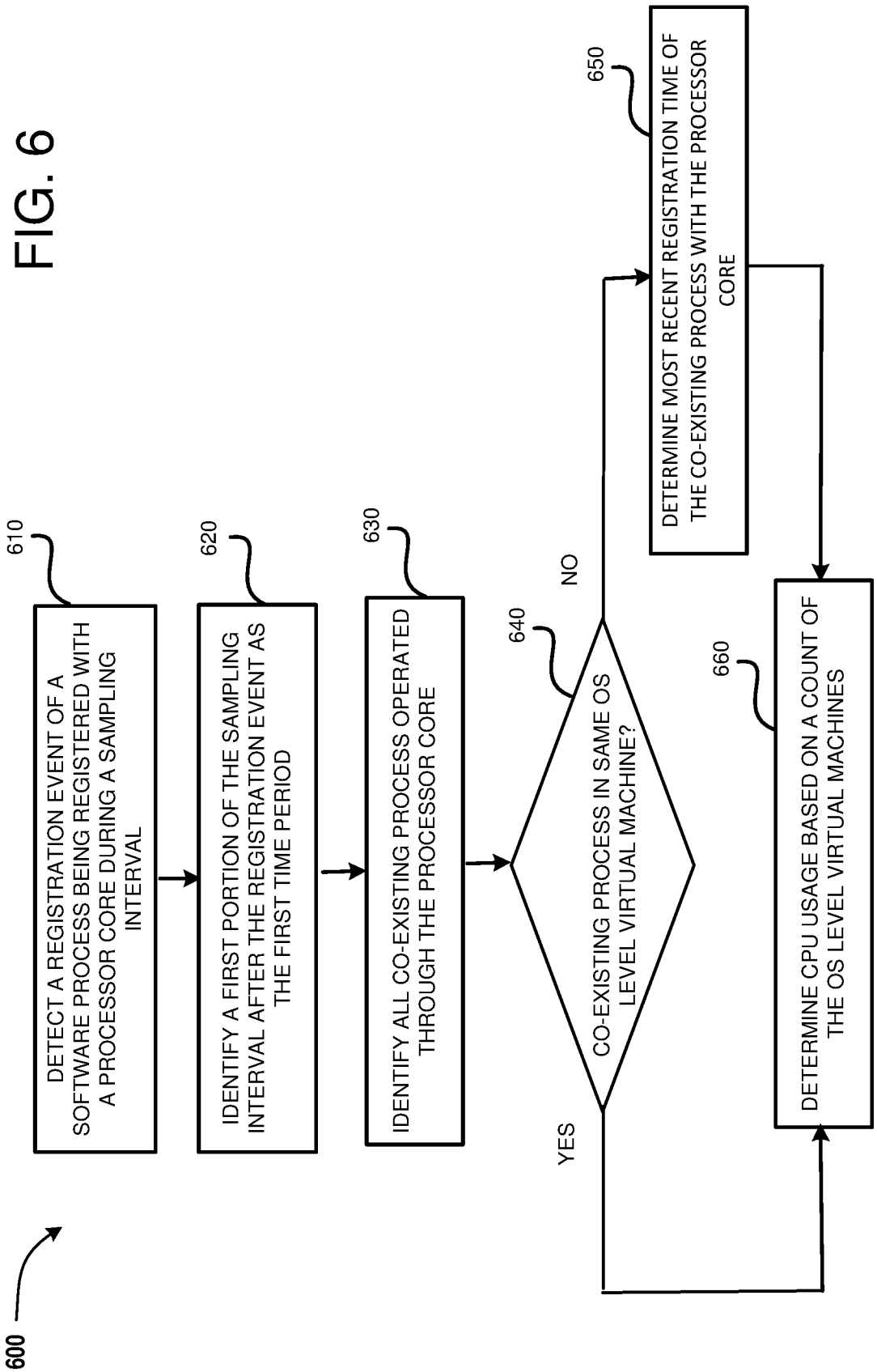
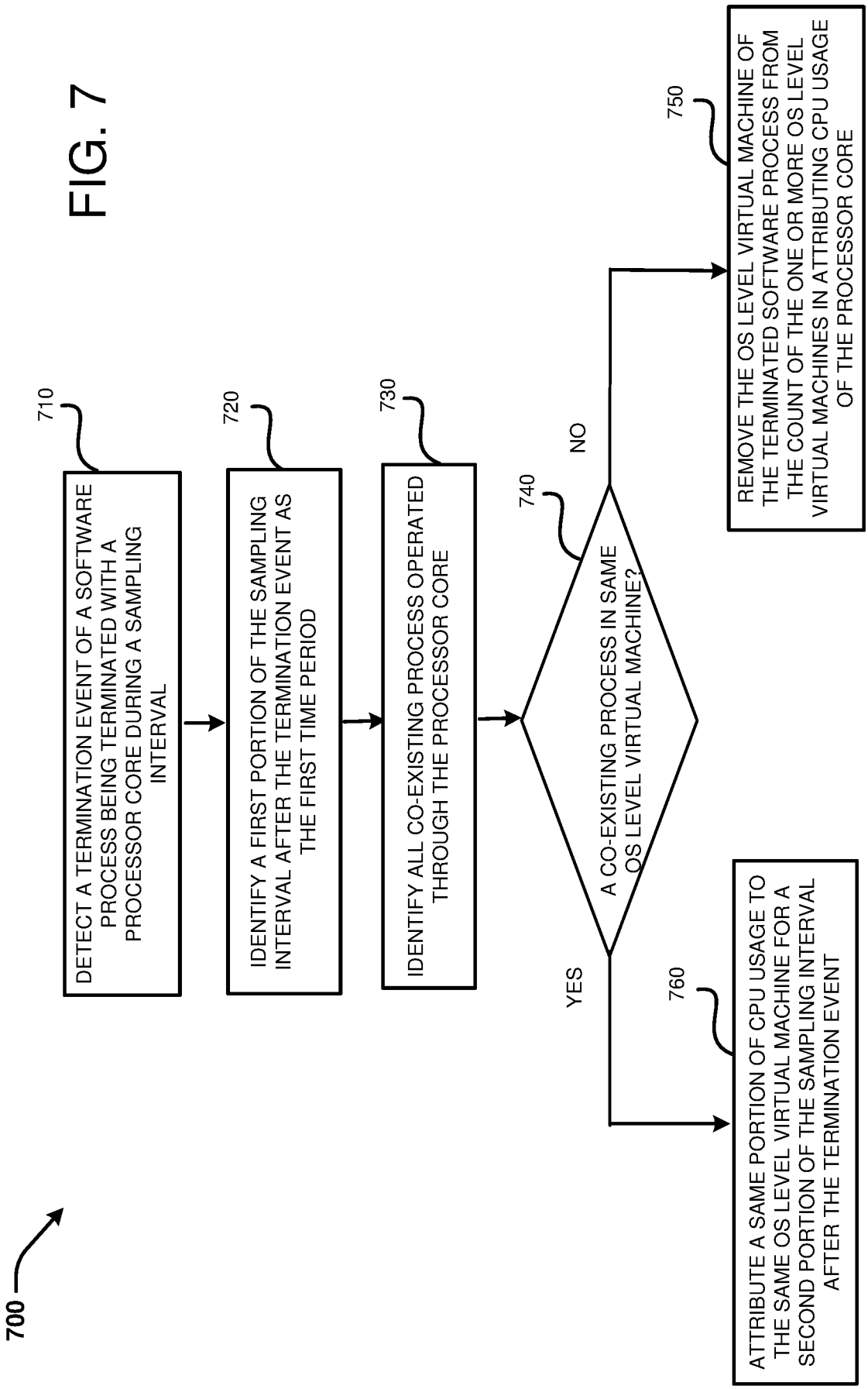

600
610
DETECT A REGISTRATION EVENT OF A SOFTWARE PROCESS BEING REGISTERED WITH A PROCESSOR CORE DURING A SAMPLING INTERVAL
620
IDENTIFY A FIRST PORTION OF THE SAMPLING INTERVAL AFTER THE REGISTRATION EVENT AS THE FIRST TIME PERIOD
630
IDENTIFY ALL CO-EXISTING PROCESS OPERATED THROUGH THE PROCESSOR CORE
640
CO-EXISTING PROCESS IN SAME OS LEVEL VIRTUAL MACHINE?
NO
YES
650
DETERMINE MOST RECENT REGISTRATION TIME OF THE CO-EXISTING PROCESS WITH THE PROCESSOR CORE
660
DETERMINE CPU USAGE BASED ON A COUNT OF THE OS LEVEL VIRTUAL MACHINES

FIG. 7


700
710
DETECT A TERMINATION EVENT OF A SOFTWARE PROCESS BEING TERMINATED WITH A PROCESSOR CORE DURING A SAMPLING INTERVAL
720
IDENTIFY A FIRST PORTION OF THE SAMPLING INTERVAL AFTER THE TERMINATION EVENT AS THE FIRST TIME PERIOD
730
IDENTIFY ALL CO-EXISTING PROCESS OPERATED THROUGH THE PROCESSOR CORE
740
A CO-EXISTING PROCESS IN SAME OS LEVEL VIRTUAL MACHINE?
NO
YES
750
REMOVE THE OS LEVEL VIRTUAL MACHINE OF THE TERMINATED SOFTWARE PROCESS FROM THE COUNT OF THE ONE OR MORE OS LEVEL VIRTUAL MACHINES IN ATTRIBUTING CPU USAGE OF THE PROCESSOR CORE
760
ATTRIBUTE A SAME PORTION OF CPU USAGE TO THE SAME OS LEVEL VIRTUAL MACHINE FOR A SECOND PORTION OF THE SAMPLING INTERVAL AFTER THE TERMINATION EVENT

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2017/088619

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/50(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)

G06F

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)

WPI, EPODOC, CNPAT, CNKI, IEEE: multi, core, process, OS, operating system, virtualization, virtual machine, register, registration, allocate, schedule, usage, utilization, CPU, interval, time, period

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CN 101169731 A (HUAWEI TECHNOLOGIES CO., LTD.) 30 April 2008 (2008-04-30) description, pages 3-10	1-23
A	US 2009064164 A1 (BOSE, PRADIP ET AL.) 05 March 2009 (2009-03-05) the whole document	1-23
A	US 2012096462 A1 (ELECTRONICS AND TELECOMMUNICATIONS RESEARCH INSTITUTE) 19 April 2012 (2012-04-19) the whole document	1-23
A	US 2010082938 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 01 April 2010 (2010-04-01) the whole document	1-23
A	US 2015120979 A1 (HITACHI LTD.) 30 April 2015 (2015-04-30) the whole document	1-23

☐ 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 another 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
11 March 2018	**19 March 2018**

Name and mailing address of the ISA/CN	Authorized officer
STATE INTELLECTUAL PROPERTY OFFICE OF THE P.R.CHINA **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China**	**ZHAO,Ting**
Facsimile No. **(86-10)62019451**	Telephone No. **(86-10)53961350**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2017/088619

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
CN 101169731 A	30 April 2008	None	
US 2009064164 A1	05 March 2009	WO 2009027153 A1	05 March 2009
US 2012096462 A1	19 April 2012	KR 20120040088 A	26 April 2012
US 2010082938 A1	01 April 2010	TW 201023046 A	16 June 2010
		WO 2010037745 A1	08 April 2010
US 2015120979 A1	30 April 2015	JP 2015088014 A	07 May 2015