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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 December 2014 (24.12.2014)

WIPO | PCT

(10) International Publication Number
WO 2014/204453 A1

(51) International Patent Classification:
G06F 9/455 (2006.01)

(21) International Application Number:
PCT/US2013/046584

(22) International Filing Date:
19 June 2013 (19.06.2013)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **EMPIRE TECHNOLOGY DEVELOPMENT, LLC** [US/US]; 2711 Centerville Road, Suite 400, Wilminton, DE 19808 (US).

(72) Inventor: **KRUGLICK, Ezekiel**; 13842 Deergrass Ct, Poway, CA 92084-2276 (US).

(74) Agent: **TURK, Carl, K.**; Turk Ip Law, LLC, 2885 Sanford Ave. S.W., #23998, Grandville, MI 49418 (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, BN, 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, 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, PA, 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, 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: PROCESSOR-OPTIMIZED LIBRARY LOADING FOR VIRTUAL MACHINES

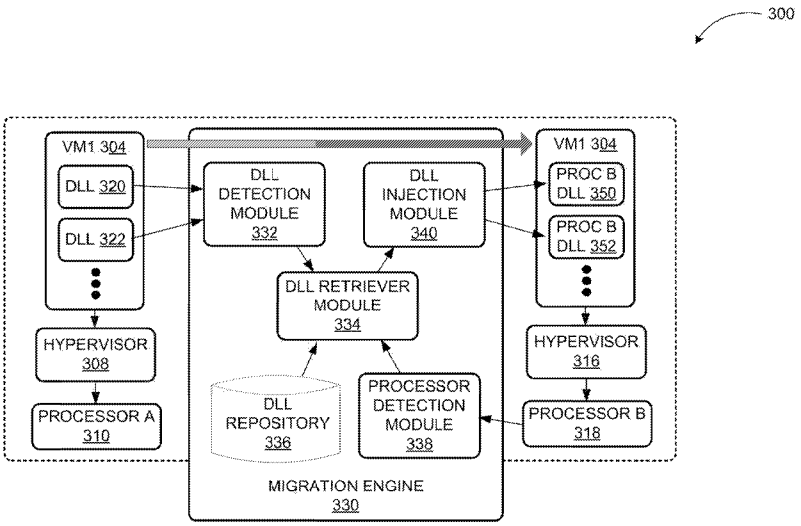

FIG. 3

(57) Abstract: Technologies are provided for loading processor-optimized library modules into virtual machines. In some examples, when a virtual machine is to be executed on a processor, the processor may be identified based on one or more processor characteristics. After the processor is identified, one or more library modules loaded into the virtual machine may be replaced with other library modules optimized for the identified processor.

WO 2014/204453 A1

PROCESSOR-OPTIMIZED LIBRARY LOADING FOR VIRTUAL MACHINES

BACKGROUND

[0001] Unless otherwise indicated herein, the materials described in this section are not prior art to the claims in this application and are not admitted to be prior art by inclusion in this section.

[0002] Cloud-based systems, used in many web-enabled applications, continue to grow in popularity. The datacenters that underpin these cloud-based systems may attempt to provide hardware- or platform-agnostic services, virtual machines, or instances for flexibility and ease of migration. However, the actual hardware in many datacenters may vary. For example, a datacenter may include processors from different manufacturers, with different processor architectures, different processor types, and/or different processor models.

[0003] There are challenges in trying to realize virtualized and load balanced cloud computation. One of the more prominent challenges is that for many important and demanding applications that make intensive use of computing resources, true performance optimization may not just be instruction set reliant but may actually depend on optimization to a particular processor make and model. However, datacenters may not allow users to select their favorite processor for practical reasons. Different processors may provide different amounts of real compute power for the same nominal instance type, so most users may simply select the "best" processor. Another issue is that, users used to picking their favorite processor type may become attached to those processors with processor-specific code making it difficult for datacenters to upgrade to stay competitive (and making it difficult for newer datacenters to attract customers to newer hardware).

SUMMARY

[0004] The present disclosure generally describes techniques for loading processor-optimized library modules into virtual machines.

[0005] According to some examples, a method is provided for loading processor-optimized library modules into virtual machines. The method may include detecting a first library module loaded into a virtual machine (VM) to be executed on a first processor, determining a processor

characteristic associated with the first processor, selecting a second library module based on the first library module and the determined processor characteristic, and loading the second library module into the VM to replace the first library module.

[0006] According to other examples, a virtual machine manager (VMM) is provided to load processor-optimized library modules into virtual machines. The VMM may include a library detection module, a processing module, and a library injection module. The library detection module may be configured to detect a first library module loaded into a virtual machine (VM) to be executed on a first processor. The processing module may be configured to determine a processor characteristic associated with the first processor and select a second library module based on the first library module and the determined processor characteristic. The library injection module may be configured to load the second library module into the VM to replace the first library module.

[0007] According to further examples, a cloud-based datacenter is provided to load processor-optimized library modules into virtual machines. The datacenter may include at least one virtual machine (VM) operable to be executed on one or more physical machines and a datacenter controller. The datacenter controller may be configured to detect a first library module loaded into the at least one VM to be executed on a first physical machine, determine a processor characteristic associated with the first physical machine, select a second library module based on the first library module and the determined processor characteristic, and load the second library module into the VM to replace the first library module.

[0008] According to yet further examples, a computer readable medium may store instructions for loading processor-optimized library modules into virtual machines. The instructions may include detecting a first library module loaded into a virtual machine (VM) to be executed on a first processor, determining a processor characteristic associated with the first processor, selecting a second library module based on the first library module and the determined processor characteristic, and loading the second library module into the VM to replace the first library module.

[0009] The foregoing summary is illustrative only and is not intended to be in any way limiting. In addition to the illustrative aspects, embodiments, and features described above, further aspects, embodiments, and features will become apparent by reference to the drawings and the following detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The foregoing and other features of this disclosure will become more fully apparent from the following description and appended claims, taken in conjunction with the accompanying drawings. Understanding that these drawings depict only several embodiments in accordance with the disclosure and are, therefore, not to be considered limiting of its scope, the disclosure will be described with additional specificity and detail through use of the accompanying drawings, in which:

FIG. 1 illustrates an example datacenter-based system where loading of processor-optimized library modules into virtual machines may be implemented;

FIG. 2 illustrates an example datacenter-based system having multiple processors;

FIG. 3 illustrates an example migration engine configured to load processor-optimized library modules into virtual machines;

FIG. 4 illustrates a general purpose computing device, which may be used to provide loading of processor-optimized library modules into virtual machines;

FIG. 5 is a flow diagram illustrating an example method for loading processor-optimized library modules into virtual machines that may be performed by a computing device such as the computing device in FIG. 4; and

FIG. 6 illustrates a block diagram of an example computer program product,

all arranged in accordance with at least some embodiments described herein.

DETAILED DESCRIPTION

[0011] In the following detailed description, reference is made to the accompanying drawings, which form a part hereof. In the drawings, similar symbols identify similar components, unless context dictates otherwise. The illustrative embodiments described in the detailed description, drawings, and claims are not meant to be limiting. Other embodiments may be utilized, and other changes may be made, without departing from the spirit or scope of the subject matter presented herein. It will be readily understood that the aspects of the present disclosure, as generally described herein, and illustrated in the Figures, can be arranged, substituted, combined, separated, and designed in a wide variety of different configurations, all of which are explicitly contemplated herein.

[0012] This disclosure is generally drawn, *inter alia*, to methods, apparatus, systems, devices, and/or computer program products related to loading processor-optimized library modules into virtual machines.

[0013] Briefly stated, technologies are generally described for loading processor-optimized library modules into virtual machines. In some examples, when a virtual machine is to be executed on a processor, the processor may be identified based on one or more processor characteristics. After the processor is identified, one or more library modules loaded in the virtual machine may be replaced with other library modules optimized for the identified processor.

[0014] A datacenter as used herein refers to an entity that hosts services and applications for customers through one or more physical server installations and one or more virtual machines executed in those server installations. Customers of the datacenter, also referred to as tenants, may be organizations that provide access to their services for multiple users. An example configuration may include an online retail service that provides retail sale services to consumers (users). The retail service may employ multiple applications (e.g., presentation of retail goods, purchase management, shipping management, inventory management, etc.), which may be hosted by one or more datacenters. Thus, a consumer may communicate with those applications of the retail service through a client application such as a browser over one or more networks and receive the provided service without realizing where the individual applications are actually executed. This scenario contrasts with configurations where each service provider would execute their applications and have their users access those applications on the retail service's own servers physically located on retail service premises. One result of the networked approach described herein is that customers like the retail service may move their hosted services / applications from one datacenter to another without the users noticing a difference.

[0015] FIG. 1 illustrates an example datacenter-based system where loading of processor-optimized library modules into virtual machines may be implemented, arranged in accordance with at least some embodiments described herein.

[0016] As shown in a diagram 100, a physical datacenter 102 may include one or more physical servers 110, 111, and 113, each of which may be configured to provide one or more virtual machines 104. For example, the physical servers 111 and 113 may be configured to provide four virtual machines and two virtual machines, respectively. In some embodiments, one

or more virtual machines may be combined into one or more virtual datacenters. For example, the four virtual machines provided by the server 111 may be combined into a virtual datacenter 112. The virtual machines 104 and/or the virtual datacenter 112 may be configured to provide cloud-related data/computing services such as various applications, data storage, data processing, or comparable ones to a group of customers 108, such as individual users or enterprise customers, via a cloud 106.

[0017] According to some examples, a virtual machine may be provided with a processor-model-specific computation library, which may be replaced based on a processor the virtual machine is being executed on. In particular, tagged optimized dynamic link libraries (DLLs) may be placed in a virtual machine and then the DLLs recognized and injected with replacements during migration so that as the VM migrates from one processor model to another processor model, the DLL is also changed between the matching processors.

[0018] FIG. 2 illustrates an example datacenter-based system having multiple processors, arranged in accordance with at least some embodiments described herein.

[0019] As shown in a diagram 200, a physical server 202 (similar to the physical servers 111 and 113 in FIG. 1) may include a number of physical processors. For example, the physical server 202 may include a processor A 210, a processor B 218, and potentially other processors (not shown). Each processor may have an associated hypervisor or virtual machine manager that manages the virtual machines or instances that are executed on the processor. For example, a first hypervisor 208 may be associated with the processor A 210. The first hypervisor 208 may in turn manage virtual machines VM1 204, VM2 206, and potentially other virtual machines (not shown), each of which are configured to execute on the processor A 210. Similarly, a second hypervisor 216 may be associated with the processor B 218 and may manage virtual machines VM3 212, VM4 214, and potentially other virtual machines (not shown) configured to execute on the processor B 218.

[0020] As mentioned above, the hardware in datacenters may vary, and a given datacenter may include different kinds or types of processors. For example, the processor A 210 may differ from the processor B 218 in manufacturer, processor architecture, processor type, processor model, or any other processor characteristic. While the datacenter with the physical server 202 may attempt to maintain consistent performance across different hardware, implementation differences across different processors may make this difficult. For example, a particular model

of processor may support a particular instruction set, while another model of processor may not, even if the two processor models have the same manufacturer, architecture, and/or type. Nevertheless, a datacenter may migrate instances or virtual machines between processors having relatively similar performance, even if the processors actually differ significantly. For example, if the processor A 210 and the processor B 218 have relatively similar performance, the virtual machine VM1 204 may be migrated between the processor A 210 and the processor B 218 automatically and transparently for, e.g., load distribution purposes.

[0021] At the same time, cloud developers may wish to optimize the performance of their applications or virtual machines in the datacenter environment, even to the point of taking advantage of hardware-specific implementation details. For example, a developer may include particular instructions or software library modules (also known as "dynamic linked libraries" or DLLs) optimized for a particular type or model of processor in a virtual machine in order to increase performance. If the virtual machine is executed on the corresponding type or model of processor, then the optimized library modules may allow increased performance. However, if the virtual machine is then migrated to another processor of a different type or model, then performance may be lost, and the virtual machine may even fail if the other processor does not support the particular instructions or library modules. As a result, the developer may have no choice other than to either accept the performance degradation or shut down and restart the virtual machine, hopefully on a more suitable processor. In the latter case, the developer may sometimes end up wasting already-paid-for time on the original processor.

[0022] To address this issue, techniques and systems for detecting and replacing DLLs in a virtual machine to be migrated with DLLs optimized for the destination hardware may be provided, as described below.

[0023] FIG. 3 illustrates an example migration engine configured to load processor-optimized library modules into virtual machines, arranged in accordance with at least some embodiments described herein.

[0024] According to a diagram 300, a virtual machine VM1 304 (similar to the VM1 204) may be configured to execute on a processor A 310 (similar to the processor A 210) and to be managed by a first hypervisor 308 (similar to the first hypervisor 208). The virtual machine VM1 304 may be configured to operate using one or more library modules or dynamic linked libraries (DLLs) to perform particular functions, such as data manipulations or processing. For example,

the virtual machine VM1 may operate using a DLL 320, a DLL 322, and other DLLs as appropriate. In some embodiments, the DLLs may be configured to provide video, image, and/or file compression functionality. The DLLs may also be configured to provide matrix processing functionality and/or real-time analytics processing.

[0025] At some point, the virtual machine VM1 304 may be migrated to a different processor. For example, the virtual machine VM1 304 may be migrated to another processor on the same physical server, on another physical server at the datacenter, or even to another datacenter. In the example shown in the diagram 300, the virtual machine VM1 304 may be migrated to a processor B 318 (similar to the processor B 218) and managed by a hypervisor 316 (similar to the hypervisor 216).

[0026] In some embodiments, a migration engine 330 may control the migration of the virtual machine VM1 304 from the processor A 310 to the processor B 318. The migration engine 330 may include components configured to detect DLLs in a virtual machine to be migrated, search for equivalent DLLs optimized for the destination processor, and replace the detected DLLs with the equivalent DLLs. For example, a DLL detection module 332 in the migration engine 330 may be configured to detect the DLLs 320 and 322 in the virtual machine VM1 304 and send information about the detected DLLs 320 and 322 to a DLL retriever module 334. In some embodiments, the DLLs 320 and 322 may each be tagged with identification data that allows the DLL detection module 332 to identify the particular DLL and/or its function. The identification data may include an identification string and/or a deduplication hash block (i.e., a data block that, when subject to a hashing procedure, results in a known hash signature), and may contain information about the function of the DLL and/or hardware it is optimized for. In some embodiments, DLL tag information may be present in the memory of the virtual machine VM1 304, and the DLL detection module 332 may search the virtual machine memory to identify the DLLs.

[0027] A processor detection module 338 may be configured to detect one or more characteristics associated with the destination processor B 318, such as processor type, model number, serial number, architecture, or any other suitable processor characteristic. The processor detection module 338 may then send the detected processor characteristic(s) to the DLL retriever module 334.

[0028] The DLL retriever module 334 may then search for DLLs equivalent to the detected DLLs 320 and 322 and optimized for the destination processor B 318. For example, the DLL retriever module 334 may search for equivalent DLLs in a DLL repository 336. The DLL repository 336 may store a number of different DLLs, some having equivalent function but written and optimized for different hardware configurations. In some embodiments, the different DLLs in the DLL repository 336 may be provided by processor manufacturers and/or service providers dedicated to providing hardware-optimized DLLs to datacenters or other cloud providers. Each DLL in the DLL repository 336 may be tagged with identification information (e.g., similar to the DLLs 320 and 322 as described above). The DLL retriever module 334 may then search the DLL repository 336 by looking for DLLs tagged with identification information that correspond to those of the DLLs 320 and 322.

[0029] After finding or identifying equivalent DLLs (if any) in the DLL repository 336, the DLL retriever module 334 may then provide the equivalent DLLs to a DLL injection module 340 for injection into the virtual machine to be migrated. For example, the DLL retriever module 334 may find a DLL 350 equivalent to the DLL 320 and optimized for the processor B 318 and a DLL 352 equivalent to the DLL 322 and optimized for the processor B 318. The DLL retriever module 334 may then provide the DLLs 350 and 352 to the DLL injection module 340, which may in turn inject the DLLs 350 and 352 into the virtual machine VM1 304 being migrated. The DLL injection module 340 may inject the DLLs 350 and 352 into the virtual machine VM1 304 to replace the DLLs 320 and 322 while the virtual machine VM1 304 is being migrated to the processor B 318. Before the DLLs 320 and 322 are replaced, the DLL injection module 340 may be configured to determine if any processes in the virtual machine VM1 304 are currently using either the DLL 320 or the DLL 322. In response to determining that process(es) are using the DLL 320 or DLL 322, the DLL injection module may wait for the process(es) to complete before replacing the DLLs 320/322 with the DLLs 350/352.

[0030] In some embodiments, the DLL injection module 340 may be configured to inject DLLs into a virtual machine when the virtual machine is being started up on a processor (i.e., not being migrated or transferred from one processor to another). The DLL injection module 340 may also be configured to inject DLLs into a virtual machine as the virtual machine is shutting down. For example, suppose the virtual machine VM1 304 is being transferred from a development environment to a production environment (i.e., actually serving customers). In the

development environment, the virtual machine VM1 304 may be using general-purpose DLLs that may not be specifically optimized for any particular hardware or processor. In this case, the DLL injection module 340 may replace the general-purpose DLLs with processor-optimized DLLs when the virtual machine VM1 304 shuts down in the development environment. As a result, the virtual machine VM1 304 can be started up in the production environment already having the appropriate processor-optimized DLLs.

[0031] FIG. 4 illustrates a general purpose computing device, which may be used to provide loading of processor-optimized library modules into virtual machines, arranged in accordance with at least some embodiments described herein.

[0032] For example, the computing device 400 may be used to provide loading of processor-optimized library modules into virtual machines as described herein. In an example basic configuration 402, the computing device 400 may include one or more processors 404 and a system memory 406. A memory bus 408 may be used for communicating between the processor 404 and the system memory 406. The basic configuration 402 is illustrated in FIG. 4 by those components within the inner dashed line.

[0033] Depending on the desired configuration, the processor 404 may be of any type, including but not limited to a microprocessor (μP), a microcontroller (μC), a digital signal processor (DSP), or any combination thereof. The processor 404 may include one more levels of caching, such as a level cache memory 412, a processor core 414, and registers 416. The example processor core 414 may include an arithmetic logic unit (ALU), a floating point unit (FPU), a digital signal processing core (DSP Core), or any combination thereof. An example memory controller 418 may also be used with the processor 404, or in some implementations the memory controller 418 may be an internal part of the processor 404.

[0034] Depending on the desired configuration, the system memory 406 may be of any type including but not limited to volatile memory (such as RAM), non-volatile memory (such as ROM, flash memory, etc.) or any combination thereof. The system memory 406 may include an operating system 420, a migration module 422, and program data 424. The migration module 422 may include a DLL replacement module 426 and a processor detection module 428 to provide loading of processor-optimized library modules as described herein. The program data 424 may include, among other data, DLL data 430 or the like, as described herein.

[0035] The computing device 400 may have additional features or functionality, and additional interfaces to facilitate communications between the basic configuration 402 and any desired devices and interfaces. For example, a bus/interface controller 430 may be used to facilitate communications between the basic configuration 402 and one or more data storage devices 432 via a storage interface bus 434. The data storage devices 432 may be one or more removable storage devices 436, one or more non-removable storage devices 438, or a combination thereof. Examples of the removable storage and the non-removable storage devices include magnetic disk devices such as flexible disk drives and hard-disk drives (HDD), optical disk drives such as compact disk (CD) drives or digital versatile disk (DVD) drives, solid state drives (SSD), and tape drives to name a few. Example computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data.

[0036] The system memory 406, the removable storage devices 436 and the non-removable storage devices 438 are examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD), solid state drives, or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which may be used to store the desired information and which may be accessed by the computing device 400. Any such computer storage media may be part of the computing device 400.

[0037] The computing device 400 may also include an interface bus 440 for facilitating communication from various interface devices (e.g., one or more output devices 442, one or more peripheral interfaces 444, and one or more communication devices 466) to the basic configuration 402 via the bus/interface controller 430. Some of the example output devices 442 include a graphics processing unit 448 and an audio processing unit 450, which may be configured to communicate to various external devices such as a display or speakers via one or more A/V ports 452. One or more example peripheral interfaces 444 may include a serial interface controller 454 or a parallel interface controller 456, which may be configured to communicate with external devices such as input devices (e.g., keyboard, mouse, pen, voice input device, touch input device, etc.) or other peripheral devices (e.g., printer, scanner, etc.) via

one or more I/O ports 458. An example communication device 466 includes a network controller 460, which may be arranged to facilitate communications with one or more other computing devices 462 over a network communication link via one or more communication ports 464. The one or more other computing devices 462 may include servers at a datacenter, customer equipment, and comparable devices.

[0038] The network communication link may be one example of a communication media. Communication media may be embodied by computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport mechanism, and may include any information delivery media. A "modulated data signal" may be a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media may include wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, radio frequency (RF), microwave, infrared (IR) and other wireless media. The term computer readable media as used herein may include both storage media and communication media.

[0039] The computing device 400 may be implemented as a part of a general purpose or specialized server, mainframe, or similar computer that includes any of the above functions. The computing device 400 may also be implemented as a personal computer including both laptop computer and non-laptop computer configurations.

[0040] Example embodiments may also include methods for instance processing customization at migration for continuously optimized datacenter video processing. These methods can be implemented in any number of ways, including the structures described herein. One such way may be by machine operations, of devices of the type described in the present disclosure. Another optional way may be for one or more of the individual operations of the methods to be performed in conjunction with one or more human operators performing some of the operations while other operations may be performed by machines. These human operators need not be collocated with each other, but each can be with a machine that performs a portion of the program. In other examples, the human interaction can be automated such as by pre-selected criteria that may be machine automated.

[0041] FIG. 5 is a flow diagram illustrating an example method for loading processor-optimized library modules into virtual machines that may be performed by a computing device

such as the computing device in FIG. 4, arranged in accordance with at least some embodiments described herein.

[0042] Example methods may include one or more operations, functions or actions as illustrated by one or more of blocks 522, 524, 526, and/or 528, and may in some embodiments be performed by a computing device such as the computing device 400 in FIG. 4. The operations described in the blocks 522-528 may also be stored as computer-executable instructions in a computer-readable medium such as a computer-readable medium 520 of a computing device 510.

[0043] An example process for loading processor-optimized library modules into virtual machines may begin with block 522, "DETECT DLLS IN A VIRTUAL MACHINE TO BE MIGRATED", where a DLL detection module (e.g., the DLL detection module 332) may detect DLLs in a virtual machine to be migrated to a different processor, as described above. For example, the DLL detection module 332 may detect the DLLs 320 and 322 in the VM1 304 to be migrated to the processor B 318.

[0044] Block 522 may be followed by block 524, "DETERMINE THE DESTINATION PROCESSOR TYPE TO WHICH THE VIRTUAL MACHINE IS TO BE MIGRATED", where a processor detection module (e.g., the processor detection module 338) may identify the type of destination processor to which the virtual machine is to be migrated by detecting one or more processor characteristics, as described above. For example, the processor detection module 338 may detect one or more characteristics associated with the processor B 318, such as processor type, model number, serial number, architecture, or any other suitable processor characteristic.

[0045] Block 524 may be followed by block 526, "RETRIEVE DLLS CORRESPONDING TO THE DETECTED DLLS AND SUITABLE FOR THE DETERMINED DESTINATION PROCESSOR TYPE", where a DLL retriever module (e.g., the DLL retriever module 334) searches for and retrieves DLLs suitable for the detected processor type and similar to the detected DLLs in the virtual machine to be migrated, as described above. For example, the DLL retriever module 334 may search for and retrieve DLLs optimized for the processor B 318 and equivalent in function to the DLLs 320 and 322.

[0046] Finally, block 526 may be followed by block 528, "INJECT THE RETRIEVED DLLS INTO THE MIGRATED VIRTUAL MACHINE", where a DLL injection module (e.g., the DLL injection module 340) may replace the original DLLs in the virtual machine to be

migrated with the DLLs retrieved by the DLL retriever module, as described above. For example, the DLL injection module 340 may replace the DLLs 320 and 322 in the virtual machine VM1 304 with the equivalent DLLs 350 and 352, respectively.

[0047] FIG. 6 illustrates a block diagram of an example computer program product, arranged in accordance with at least some embodiments described herein.

[0048] In some examples, as shown in FIG. 6, the computer program product 600 may include a signal bearing medium 602 that may also include one or more machine readable instructions 604 that, when executed by, for example, a processor, may provide the functionality described herein. Thus, for example, referring to the processor 404 in FIG. 4, the migration module 422 may undertake one or more of the tasks shown in FIG. 6 in response to the instructions 604 conveyed to the processor 404 by the medium 602 to perform actions associated with loading processor-optimized library modules into virtual machines as described herein. Some of those instructions may include, for example, detecting dynamic linked libraries (DLLs) in a virtual machine to be migrated, determining the destination processor type to which the virtual machine is to be migrated, retrieving DLLs corresponding to the detected DLLs and suitable for the determined destination processor type, and/or injecting the retrieved DLLs into the migrated virtual machine, according to some embodiments described herein.

[0049] In some implementations, the signal bearing medium 602 depicted in FIG. 6 may encompass a computer-readable medium 606, such as, but not limited to, a hard disk drive, a solid state drive, a Compact Disc (CD), a Digital Versatile Disk (DVD), a digital tape, memory, etc. In some implementations, the signal bearing medium 602 may encompass a recordable medium 608, such as, but not limited to, memory, read/write (R/W) CDs, R/W DVDs, etc. In some implementations, the signal bearing medium 602 may encompass a communications medium 610, such as, but not limited to, a digital and/or an analog communication medium (e.g., a fiber optic cable, a waveguide, a wired communications link, a wireless communication link, etc.). Thus, for example, the program product 600 may be conveyed to one or more modules of the processor 604 by an RF signal bearing medium, where the signal bearing medium 602 is conveyed by the wireless communications medium 610 (e.g., a wireless communications medium conforming with the IEEE 802.11 standard).

[0050] According to some examples, a method for loading processor-optimized library modules into virtual machines may include detecting a first library module loaded into a virtual

machine (VM) to be executed on a first processor, determining a processor characteristic associated with the first processor, selecting a second library module based on the first library module and the determined processor characteristic, and loading the second library module into the VM to replace the first library module.

[0051] According to some embodiments, the second library module may be optimized for the first processor. The method may further include selecting the second library module such that the first library module and the second library module provide substantially similar functionality. The functionality may include video compression, image compression, file compression, matrix processing, and/or real-time analytics processing. The method may further include loading the second library module into the VM in response to migrating the VM from another processor to the first processor, where the first library module is associated with the other processor, starting up the VM on the first processor, and/or shutting down the VM.

[0052] According to other embodiments, the method may further include determining whether any processes of the VM are using the first library module, and in response to determining that one or more processes of the VM are using the first library module, waiting for the process(es) to complete before loading the second library module into the VM to replace the first library module. The first library module may be a general-purpose library module. The processor characteristic may include processor type, processor model number, processor serial number, and/or processor architecture. The second library module may include a dynamic linked library (DLL) file.

[0053] According to further embodiments, the method may further include selecting the second library module based on identification data, where the second library module is tagged with the identification data. The identification data may include an identification string and/or a deduplication hash block. The method may further include selecting the second library module from a library module repository having multiple library modules. The library modules may be provided by a manufacturer of the processor and/or a library service provider.

[0054] According to other examples, a virtual machine manager (VMM) to load processor-optimized library modules into virtual machines at datacenters may include a library detection module, a processing module, and a library injection module. The library detection module may be configured to detect a first library module loaded into a virtual machine (VM) to be executed on a first processor. The processing module may be configured to determine a processor

characteristic associated with the first processor and select a second library module based on the first library module and the determined processor characteristic. The library injection module may be configured to load the second library module into the VM to replace the first library module.

[0055] According to some embodiments, the second library module may be optimized for the first processor. The first library module and the second library module may provide substantially similar functionality. The functionality may include video compression, image compression, file compression, matrix processing, and/or real-time analytics processing. The library injection module may be further configured to load the second library module into the VM in response to a migration of the VM from another processor to the first processor, where the first library module is associated with the other processor, a start-up process of the VM on the first processor, and/or a shut-down process of the VM.

[0056] According to other embodiments, the processing module may be further configured to determine whether any processes of the VM are using the first library module, and in response to determination that one or more processes of the VM are using the first library module, causing the library injection module to wait for the process(es) to complete before loading the second library module into the VM to replace the first library module. The first library module may be a general-purpose library module. The processor characteristic may include processor type, processor model number, processor serial number, and/or processor architecture. The second library module may include a dynamic linked library (DLL) file.

[0057] According to further embodiments, the second library module may be tagged with identification data, and the processing module may be further configured to select the second library module based on the identification data. The identification data may include an identification string and/or a deduplication hash block. The VMM may further include a library module repository having multiple library modules, and the processing module may be further configured to select the second library module from the library module repository. The library modules may be provided by a manufacturer of the processor and/or a library service provider.

[0058] According to further examples, a cloud-based datacenter configured to load processor-optimized library modules into virtual machines at the datacenter may include at least one virtual machine (VM) operable to be executed on one or more physical machines and a datacenter controller. The datacenter controller may be configured to detect a first library module

loaded into the at least one VM to be executed on a first physical machine, determine a processor characteristic associated with the first physical machine, select a second library module based on the first library module and the determined processor characteristic, and load the second library module into the VM to replace the first library module.

[0059] According to some embodiments, the second library module may be optimized for the first physical machine. The first library module and the second library module may provide substantially similar functionality. The functionality may include video compression, image compression, file compression, matrix processing, and/or real-time analytics processing. The controller may be further configured to load the second library module into the VM in response to a migration of the VM from another physical machine to the first physical machine, where the first library module is associated with the other physical machine, a start-up process of the VM on the first physical machine, and/or a shut-down process of the VM.

[0060] According to other embodiments, the controller may be further configured to determine whether any processes of the VM are using the first library module, and in response to determination that one or more processes of the VM are using the first library module, wait for the process(es) to complete before loading the second library module into the VM to replace the first library module. The first library module may be a general-purpose library module. The processor characteristic may include processor type, processor model number, processor serial number, and/or processor architecture. The second library module may include a dynamic linked library (DLL) file.

[0061] According to further embodiments, the second library module may be tagged with identification data, and the controller may be further configured to select the second library module based on the identification data. The identification data may include an identification string and/or a deduplication hash block. The datacenter may further include a library module repository having multiple library modules, and the controller may be further configured to select the second library module from the library module repository. The library modules may be provided by a manufacturer of the processor and/or a library service provider.

[0062] According to yet further examples, a computer readable storage medium may store instructions, which when executed on one or more computing devices execute a method for loading processor-optimized library modules into virtual machines. The instructions may include detecting a first library module loaded into a virtual machine (VM) to be executed on a first

processor, determining a processor characteristic associated with the first processor, selecting a second library module based on the first library module and the determined processor characteristic, and loading the second library module into the VM to replace the first library module.

[0063] According to some embodiments, the second library module may be optimized for the first processor. The instructions may further include selecting the second library module such that the first library module and the second library module provide substantially similar functionality. The functionality may include video compression, image compression, file compression, matrix processing, and/or real-time analytics processing. The instructions may further include loading the second library module into the VM in response to migrating the VM from another processor to the first processor, where the first library module is associated with the other processor, starting up the VM on the first processor, and/or shutting down the VM.

[0064] According to other embodiments, the instructions may further include determining whether any processes of the VM are using the first library module, and in response to determining that one or more processes of the VM are using the first library module, waiting for the process(es) to complete before loading the second library module into the VM to replace the first library module. The first library module may be a general-purpose library module. The processor characteristic may include processor type, processor model number, processor serial number, and/or processor architecture. The second library module may include a dynamic linked library (DLL) file.

[0065] According to further embodiments, the instructions may further include selecting the second library module based on identification data, where the second library module is tagged with the identification data. The identification data may include an identification string and/or a deduplication hash block. The instructions may further include selecting the second library module from a library module repository having multiple library modules. The library modules may be provided by a manufacturer of the processor and/or a library service provider.

[0066] There is little distinction left between hardware and software implementations of aspects of systems; the use of hardware or software is generally (but not always, in that in certain contexts the choice between hardware and software may become significant) a design choice representing cost vs. efficiency tradeoffs. There are various vehicles by which processes and/or systems and/or other technologies described herein may be effected (e.g., hardware, software,

and/or firmware), and that the preferred vehicle will vary with the context in which the processes and/or systems and/or other technologies are deployed. For example, if an implementer determines that speed and accuracy are paramount, the implementer may opt for a mainly hardware and/or firmware vehicle; if flexibility is paramount, the implementer may opt for a mainly software implementation; or, yet again alternatively, the implementer may opt for some combination of hardware, software, and/or firmware.

[0067] The foregoing detailed description has set forth various embodiments of the devices and/or processes via the use of block diagrams, flowcharts, and/or examples. Insofar as such block diagrams, flowcharts, and/or examples contain one or more functions and/or operations, it will be understood by those within the art that each function and/or operation within such block diagrams, flowcharts, or examples may be implemented, individually and/or collectively, by a wide range of hardware, software, firmware, or virtually any combination thereof. In one embodiment, several portions of the subject matter described herein may be implemented via Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs), digital signal processors (DSPs), or other integrated formats. However, those skilled in the art will recognize that some aspects of the embodiments disclosed herein, in whole or in part, may be equivalently implemented in integrated circuits, as one or more computer programs executing on one or more computers (e.g., as one or more programs executing on one or more computer systems), as one or more programs executing on one or more processors (e.g., as one or more programs executing on one or more microprocessors), as firmware, or as virtually any combination thereof, and that designing the circuitry and/or writing the code for the software and or firmware would be well within the skill of one of skill in the art in light of this disclosure.

[0068] The present disclosure is not to be limited in terms of the particular embodiments described in this application, which are intended as illustrations of various aspects. Many modifications and variations can be made without departing from its spirit and scope, as will be apparent to those skilled in the art. Functionally equivalent methods and apparatuses within the scope of the disclosure, in addition to those enumerated herein, will be apparent to those skilled in the art from the foregoing descriptions. Such modifications and variations are intended to fall within the scope of the appended claims. The present disclosure is to be limited only by the terms of the appended claims, along with the full scope of equivalents to which such claims are

entitled. It is also to be understood that the terminology used herein is for the purpose of describing particular embodiments only, and is not intended to be limiting.

[0069] In addition, those skilled in the art will appreciate that the mechanisms of the subject matter described herein are capable of being distributed as a program product in a variety of forms, and that an illustrative embodiment of the subject matter described herein applies regardless of the particular type of signal bearing medium used to actually carry out the distribution. Examples of a signal bearing medium include, but are not limited to, the following: a recordable type medium such as a floppy disk, a hard disk drive, a Compact Disc (CD), a Digital Versatile Disk (DVD), a digital tape, a computer memory, a solid state drive, etc.; and a transmission type medium such as a digital and/or an analog communication medium (e.g., a fiber optic cable, a waveguide, a wired communications link, a wireless communication link, etc.).

[0070] Those skilled in the art will recognize that it is common within the art to describe devices and/or processes in the fashion set forth herein, and thereafter use engineering practices to integrate such described devices and/or processes into data processing systems. That is, at least a portion of the devices and/or processes described herein may be integrated into a data processing system via a reasonable amount of experimentation. Those having skill in the art will recognize that a data processing system may include one or more of a system unit housing, a video display device, a memory such as volatile and non-volatile memory, processors such as microprocessors and digital signal processors, computational entities such as operating systems, drivers, graphical user interfaces, and applications programs, one or more interaction devices, such as a touch pad or screen, and/or control systems including feedback loops and control motors (e.g., feedback for sensing position and/or velocity of gantry systems; control motors for moving and/or adjusting components and/or quantities).

[0071] A data processing system may be implemented utilizing any suitable commercially available components, such as those found in data computing/communication and/or network computing/communication systems. The herein described subject matter sometimes illustrates different components contained within, or connected with, different other components. It is to be understood that such depicted architectures are merely exemplary, and that in fact many other architectures may be implemented which achieve the same functionality. In a conceptual sense, any arrangement of components to achieve the same functionality is effectively "associated"

such that the desired functionality is achieved. Hence, any two components herein combined to achieve a particular functionality may be seen as "associated with" each other such that the desired functionality is achieved, irrespective of architectures or intermediate components. Likewise, any two components so associated may also be viewed as being "operably connected", or "operably coupled", to each other to achieve the desired functionality, and any two components capable of being so associated may also be viewed as being "operably couplable", to each other to achieve the desired functionality. Specific examples of operably couplable include but are not limited to physically connectable and/or physically interacting components and/or wirelessly interactable and/or wirelessly interacting components and/or logically interacting and/or logically interactable components.

[0072] With respect to the use of substantially any plural and/or singular terms herein, those having skill in the art can translate from the plural to the singular and/or from the singular to the plural as is appropriate to the context and/or application. The various singular/plural permutations may be expressly set forth herein for sake of clarity.

[0073] It will be understood by those within the art that, in general, terms used herein, and especially in the appended claims (*e.g.*, bodies of the appended claims) are generally intended as "open" terms (*e.g.*, the term "including" should be interpreted as "including but not limited to," the term "having" should be interpreted as "having at least," the term "includes" should be interpreted as "includes but is not limited to," etc.). It will be further understood by those within the art that if a specific number of an introduced claim recitation is intended, such an intent will be explicitly recited in the claim, and in the absence of such recitation no such intent is present. For example, as an aid to understanding, the following appended claims may contain usage of the introductory phrases "at least one" and "one or more" to introduce claim recitations. However, the use of such phrases should not be construed to imply that the introduction of a claim recitation by the indefinite articles "a" or "an" limits any particular claim containing such introduced claim recitation to embodiments containing only one such recitation, even when the same claim includes the introductory phrases "one or more" or "at least one" and indefinite articles such as "a" or "an" (*e.g.*, "a" and/or "an" should be interpreted to mean "at least one" or "one or more"); the same holds true for the use of definite articles used to introduce claim recitations. In addition, even if a specific number of an introduced claim recitation is explicitly recited, those skilled in the art will recognize that such recitation should be interpreted to mean at

least the recited number (*e.g.*, the bare recitation of "two recitations," without other modifiers, means at least two recitations, or two or more recitations).

[0074] Furthermore, in those instances where a convention analogous to "at least one of A, B, and C, etc." is used, in general such a construction is intended in the sense one having skill in the art would understand the convention (*e.g.*, " a system having at least one of A, B, and C" would include but not be limited to systems that have A alone, B alone, C alone, A and B together, A and C together, B and C together, and/or A, B, and C together, etc.). It will be further understood by those within the art that virtually any disjunctive word and/or phrase presenting two or more alternative terms, whether in the description, claims, or drawings, should be understood to contemplate the possibilities of including one of the terms, either of the terms, or both terms. For example, the phrase "A or B" will be understood to include the possibilities of "A" or "B" or "A and B."

[0075] As will be understood by one skilled in the art, for any and all purposes, such as in terms of providing a written description, all ranges disclosed herein also encompass any and all possible subranges and combinations of subranges thereof. Any listed range can be easily recognized as sufficiently describing and enabling the same range being broken down into at least equal halves, thirds, quarters, fifths, tenths, etc. As a non-limiting example, each range discussed herein can be readily broken down into a lower third, middle third and upper third, etc. As will also be understood by one skilled in the art all language such as "up to," "at least," "greater than," "less than," and the like include the number recited and refer to ranges which can be subsequently broken down into subranges as discussed above. Finally, as will be understood by one skilled in the art, a range includes each individual member. Thus, for example, a group having 1-3 cells refers to groups having 1, 2, or 3 cells. Similarly, a group having 1-5 cells refers to groups having 1, 2, 3, 4, or 5 cells, and so forth.

[0076] While various aspects and embodiments have been disclosed herein, other aspects and embodiments will be apparent to those skilled in the art. The various aspects and embodiments disclosed herein are for purposes of illustration and are not intended to be limiting, with the true scope and spirit being indicated by the following claims.

CLAIMS

WHAT IS CLAIMED IS:

1. A method for loading processor-optimized library modules into virtual machines, the method comprising:

detecting a first library module loaded into a virtual machine (VM) to be executed on a first processor;

determining a processor characteristic associated with the first processor;

selecting a second library module based on the first library module and the determined processor characteristic; and

loading the second library module into the VM to replace the first library module.

2. The method of claim 1, wherein the second library module is optimized for the first processor.

3. The method of claim 1, further comprising selecting the second library module such that the first library module and the second library module provide substantially similar functionality.

4. The method of claim 3, wherein the functionality includes one or more of video compression, image compression, file compression, matrix processing, and real-time analytics processing.

5. The method of claim 1, further comprising loading the second library module into the VM in response to one or more of:

migrating the VM from another processor to the first processor, wherein the first library module is associated with the other processor;

starting up the VM on the first processor; and

shutting down the VM.

6. The method of claim 1, further comprising:

determining whether any processes of the VM are using the first library module; and

in response to determining that one or more processes of the VM are using the first library module, waiting for the one or more processes to complete before loading the second library module into the VM to replace the first library module.

7. The method of claim 1, wherein the first library module is a general-purpose library module.

8. The method of claim 1, wherein the processor characteristic includes one or more of processor type, processor model number, processor serial number, and processor architecture.

9. The method of claim 1, wherein the second library module includes a dynamic linked library (DLL) file.

10. The method of claim 1, further comprising selecting the second library module based on identification data, wherein the second library module is tagged with the identification data.

11. The method of claim 10 wherein the identification data includes one or more of an identification string and a deduplication hash block.

12. The method of claim 1, further comprising selecting the second library module from a library module repository having a plurality of library modules.

13. The method of claim 12, wherein the plurality of library modules is provided by one or more of a manufacturer of the processor and a library service provider.

14. A virtual machine manager (VMM) to load processor-optimized library modules into virtual machines at datacenters, the VMM comprising:

a library detection module configured to detect a first library module loaded into a virtual machine (VM) to be executed on a first processor;

a processing module configured to:

determine a processor characteristic associated with the first processor; and

select a second library module based on the first library module and the determined processor characteristic; and

a library injection module configured to load the second library module into the VM to replace the first library module.

15. The VMM of claim 14, wherein the second library module is optimized for the first processor.

16. The VMM of claim 14, wherein the first library module and the second library module provide substantially similar functionality.

17. The VMM of claim 16, wherein the functionality includes one or more of video compression, image compression, file compression, matrix processing, and real-time analytics processing.

18. The VMM of claim 14, wherein the library injection module is further configured to load the second library module into the VM in response to one or more of:

a migration of the VM from another processor to the first processor, wherein the first library module is associated with the other processor;

a start-up process of the VM on the first processor; and

a shut-down process of the VM.

19. The VMM of claim 14, wherein the processing module is further configured to:

determine whether any processes of the VM are using the first library module; and

in response to determination that one or more processes of the VM are using the first library module, causing the library injection module to wait for the one or more processes to complete before loading the second library module into the VM to replace the first library module.

20. The VMM of claim 14, wherein the first library module is a general-purpose library module.

21. The VMM of claim 14, wherein the processor characteristic includes one or more of processor type, processor model number, processor serial number, and processor architecture.

22. The VMM of claim 14, wherein the second library module includes a dynamic linked library (DLL) file.

23. The VMM of claim 14, wherein the second library module is tagged with identification data and the processing module is further configured to select the second library module based on the identification data.

24. The VMM of claim 23, wherein the identification data includes one or more of an identification string and a deduplication hash block.

25. The VMM of claim 14, further comprising a library module repository having a plurality of library modules, and wherein the processing module is further configured to select the second library module from the library module repository.

26. The VMM of claim 25, wherein the plurality of library modules is provided by one or more of a manufacturer of the first processor and a library service provider.

27. A cloud-based datacenter configured to load processor-optimized library modules into virtual machines at the datacenter, the datacenter comprising:

at least one virtual machine (VM) operable to be executed on one or more physical machines; and

a datacenter controller configured to:

detect a first library module loaded into the at least one VM to be executed on a first physical machine;

determine a processor characteristic associated with the first physical machine;

select a second library module based on the first library module and the determined processor characteristic; and

load the second library module into the VM to replace the first library module.

28. The datacenter of claim 27, wherein the second library module is optimized for the first physical machine.

29. The datacenter of claim 27, wherein the first library module and the second library module provide substantially similar functionality.

30. The datacenter of claim 29, wherein the functionality includes one or more of video compression, image compression, file compression, matrix processing, and real-time analytics processing.

31. The datacenter of claim 27, wherein the controller is further configured to load the second library module into the VM in response to one or more of:

a migration of the VM from another physical machine to the first physical machine, wherein the first library module is associated with the other physical machine;

a start-up process of the VM on the first physical machine; and

a shut-down process of the VM.

32. The datacenter of claim 27, wherein the controller is further configured to:

determine whether any processes of the VM are using the first library module; and

in response to determining that one or more processes of the VM are using the first library module, wait for the one or more processes to complete before loading the second library module into the VM to replace the first library module.

33. The datacenter of claim 27, wherein the first library module is a general-purpose library module.

34. The datacenter of claim 27, wherein the processor characteristic includes one or more of processor type, processor model number, processor serial number, and processor architecture.

35. The datacenter of claim 27, wherein the second library module includes a dynamic linked library (DLL) file.

36. The datacenter of claim 27, wherein the second library module is tagged with identification data and the controller is further configured to select the second library module based on the identification data.

37. The datacenter of claim 36, wherein the identification data includes one or more of an identification string and a deduplication hash block.

38. The datacenter of claim 27, further comprising a library module repository having a plurality of library modules, and wherein the controller is further configured to select the second library module from the library module repository.

39. The datacenter of claim 38, wherein the plurality of library modules is provided by one or more of a manufacturer of a processor in the physical machine and a library service provider.

40. A computer readable storage medium with instructions stored thereon, which when executed on one or more computing devices execute a method for loading processor-optimized library modules into virtual machines, wherein the method includes action of claims 1 through 13.

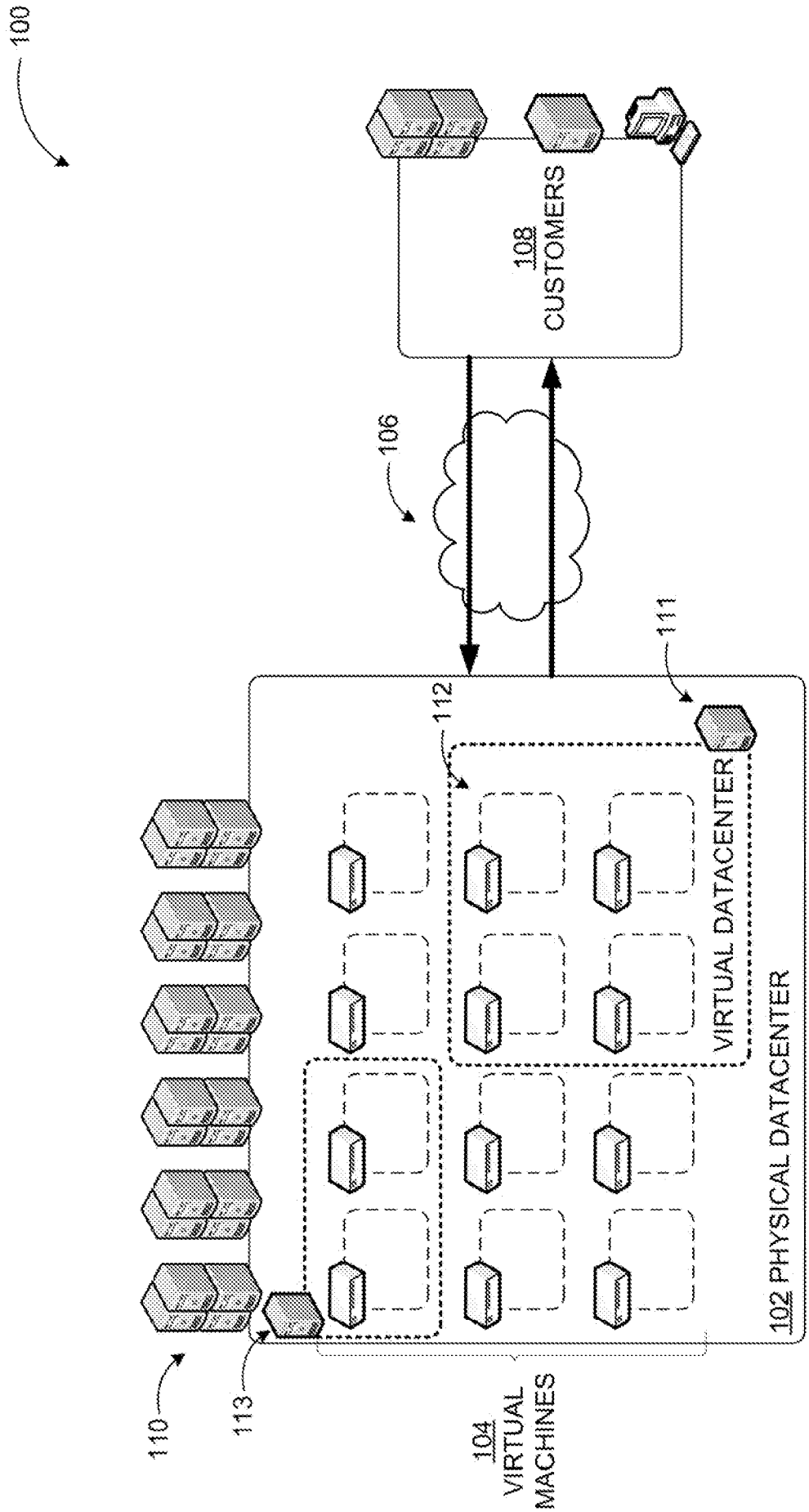
100
108 CUSTOMERS
106
111
112
VIRTUAL DATACENTER
102 PHYSICAL DATACENTER
104 VIRTUAL MACHINES
110
113

FIG. 1

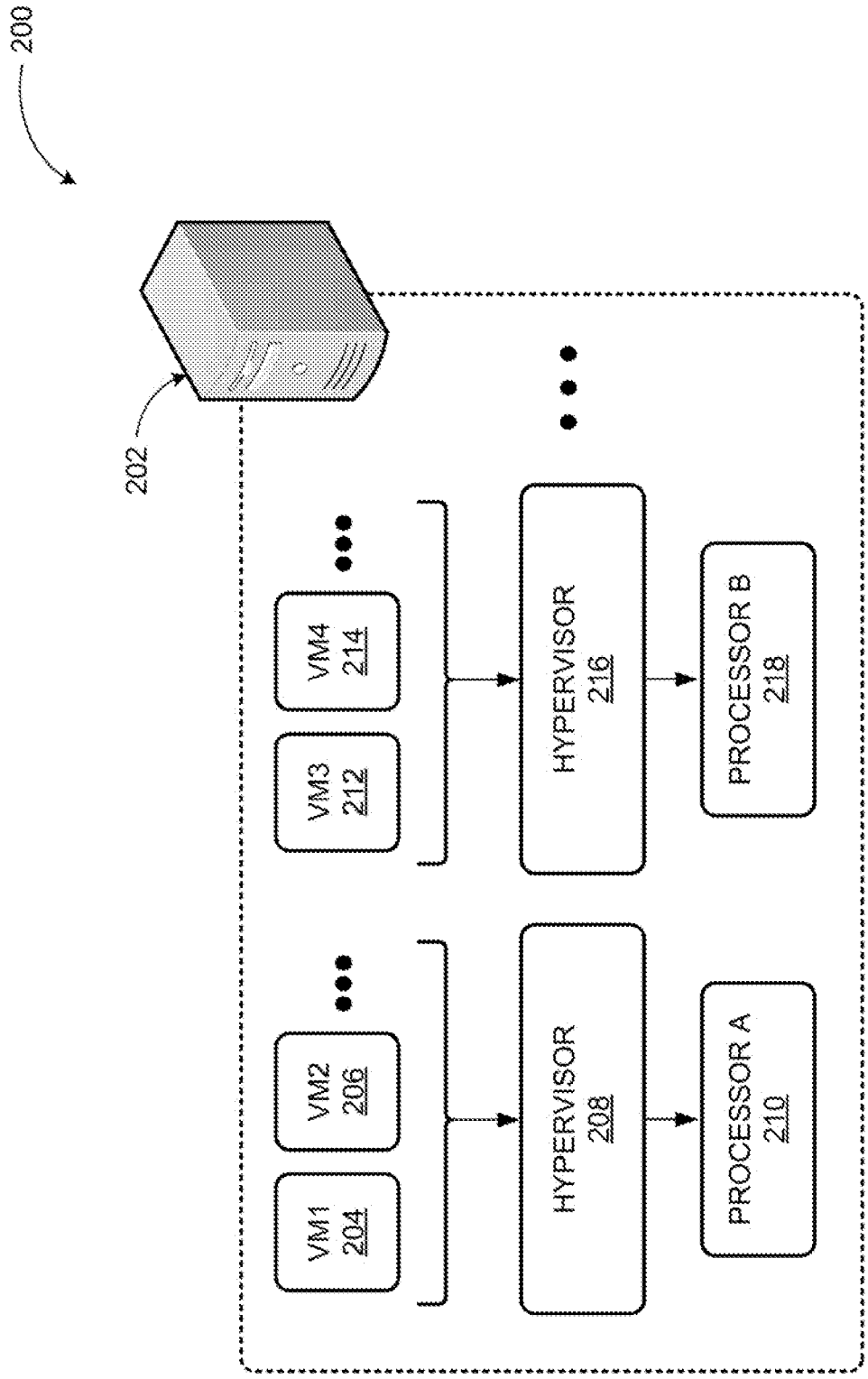
200
202
VM1
204
VM2
206
VM3
212
VM4
214
HYPERVISOR
208
HYPERVISOR
216
PROCESSOR A
210
PROCESSOR B
218

FIG. 2

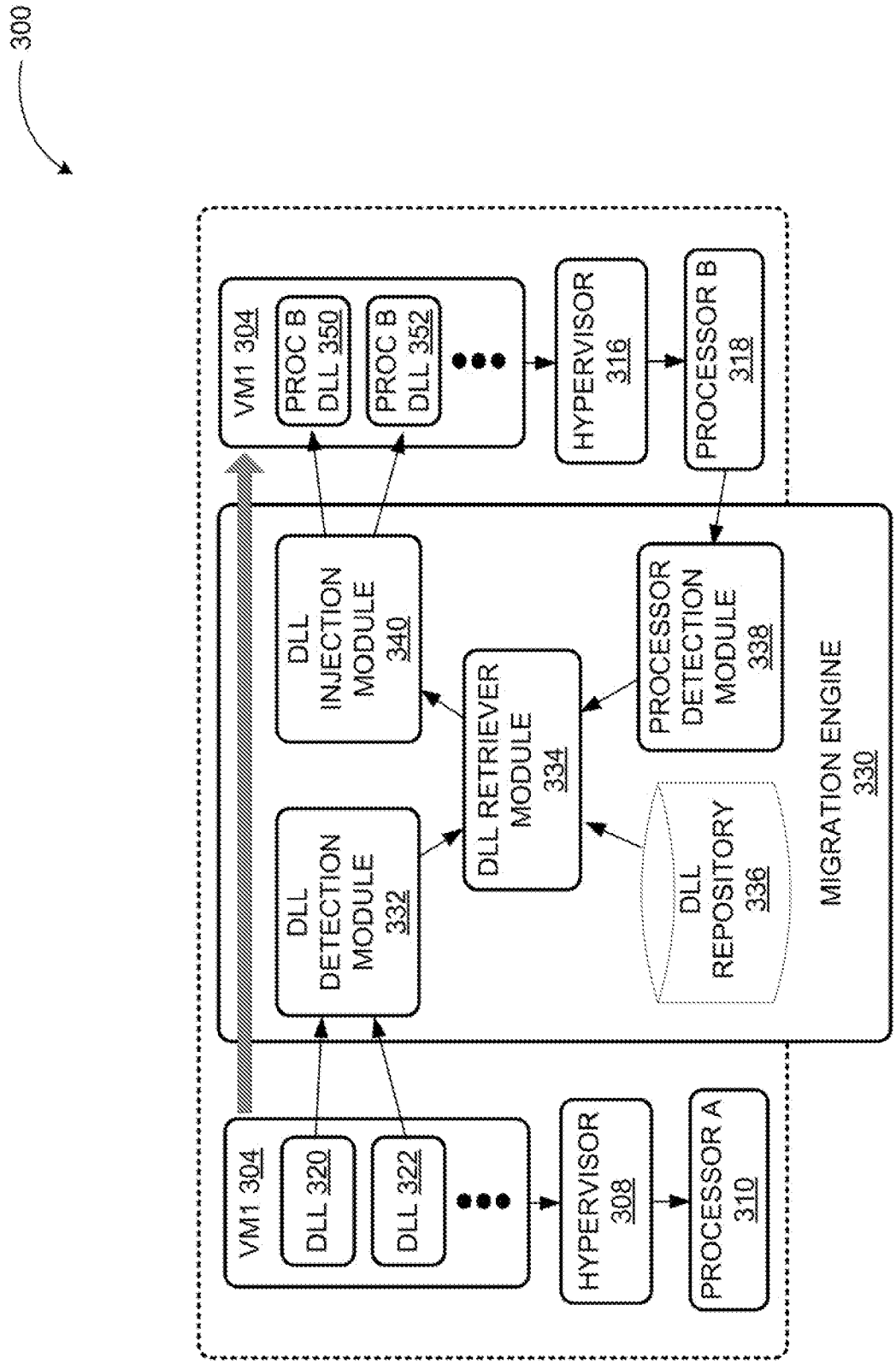
300
VM1 304
PROC B DLL 350
PROC B DLL 352
HYPERVISOR 316
PROCESSOR B 318
DLL INJECTION MODULE 340
DLL RETRIEVER MODULE 334
PROCESSOR DETECTION MODULE 338
DLL DETECTION MODULE 332
DLL REPOSITORY 336
MIGRATION ENGINE 330
VM1 304
DLL 320
DLL 322
HYPERVISOR 308
PROCESSOR A 310

FIG. 3

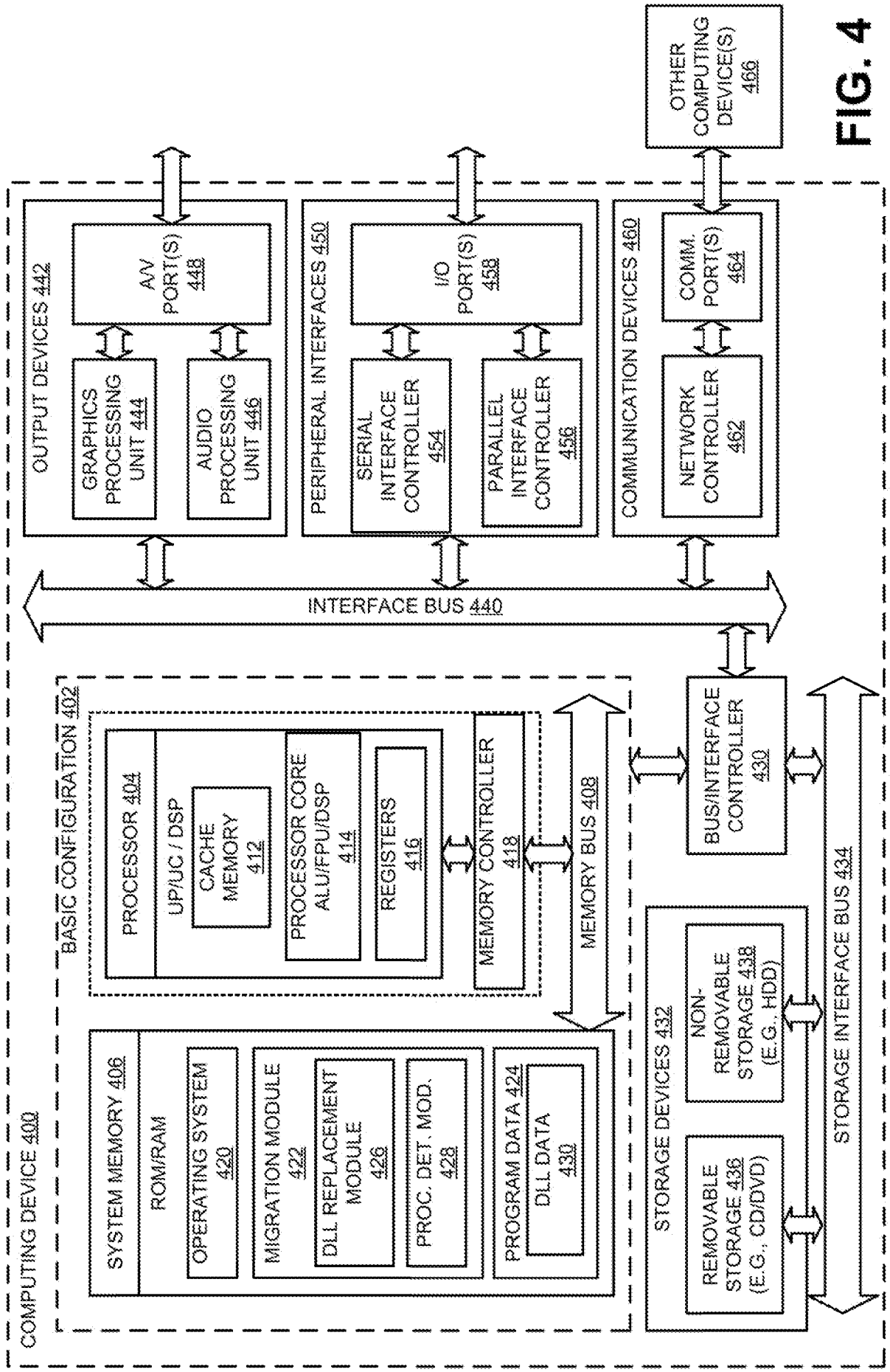
COMPUTING DEVICE 400
BASIC CONFIGURATION 402
PROCESSOR 404
UP/UC / DSP
CACHE MEMORY 412
PROCESSOR CORE ALU/FPU/DSP 414
REGISTERS 416
MEMORY CONTROLLER 418
MEMORY BUS 408
SYSTEM MEMORY 406
ROM/RAM
OPERATING SYSTEM 420
MIGRATION MODULE 422
DLL REPLACEMENT MODULE 426
PROC. DET. MOD. 428
PROGRAM DATA 424
DLL DATA 430
BUS/INTERFACE CONTROLLER 430
INTERFACE BUS 440
OUTPUT DEVICES 442
GRAPHICS PROCESSING UNIT 444
AUDIO PROCESSING UNIT 446
A/V PORT(S) 448
PERIPHERAL INTERFACES 450
SERIAL INTERFACE CONTROLLER 454
PARALLEL INTERFACE CONTROLLER 456
I/O PORT(S) 458
COMMUNICATION DEVICES 460
NETWORK CONTROLLER 462
COMM. PORT(S) 464
OTHER COMPUTING DEVICE(S) 466
STORAGE DEVICES 432
REMOVABLE STORAGE 436 (E.G., CD/DVD)
NON-REMOVABLE STORAGE 438 (E.G. HDD)
STORAGE INTERFACE BUS 434

FIG. 4

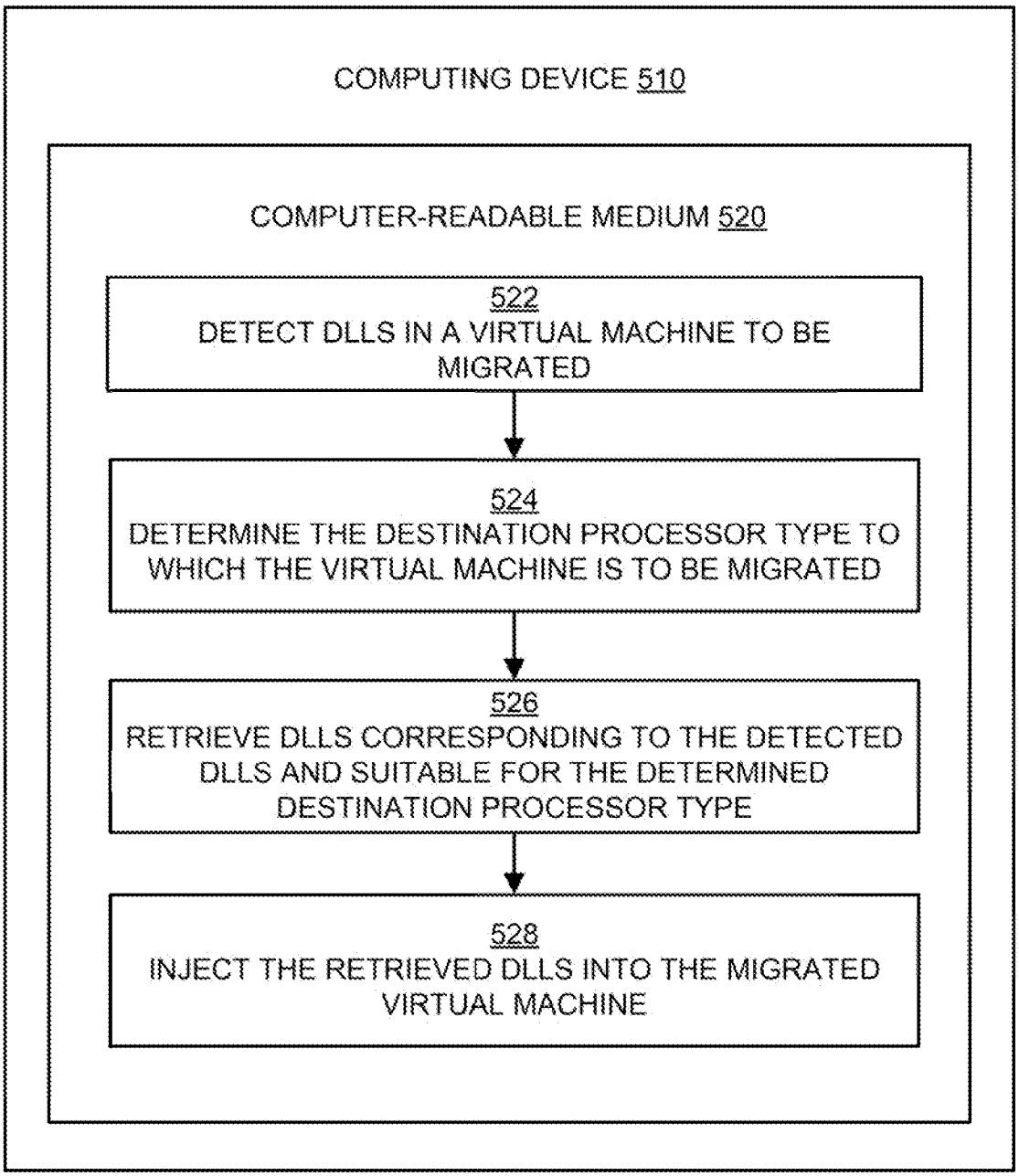
COMPUTING DEVICE 510
COMPUTER-READABLE MEDIUM 520
522
DETECT DLLS IN A VIRTUAL MACHINE TO BE MIGRATED
524
DETERMINE THE DESTINATION PROCESSOR TYPE TO WHICH THE VIRTUAL MACHINE IS TO BE MIGRATED
526
RETRIEVE DLLS CORRESPONDING TO THE DETECTED DLLS AND SUITABLE FOR THE DETERMINED DESTINATION PROCESSOR TYPE
528
INJECT THE RETRIEVED DLLS INTO THE MIGRATED VIRTUAL MACHINE

FIG. 5

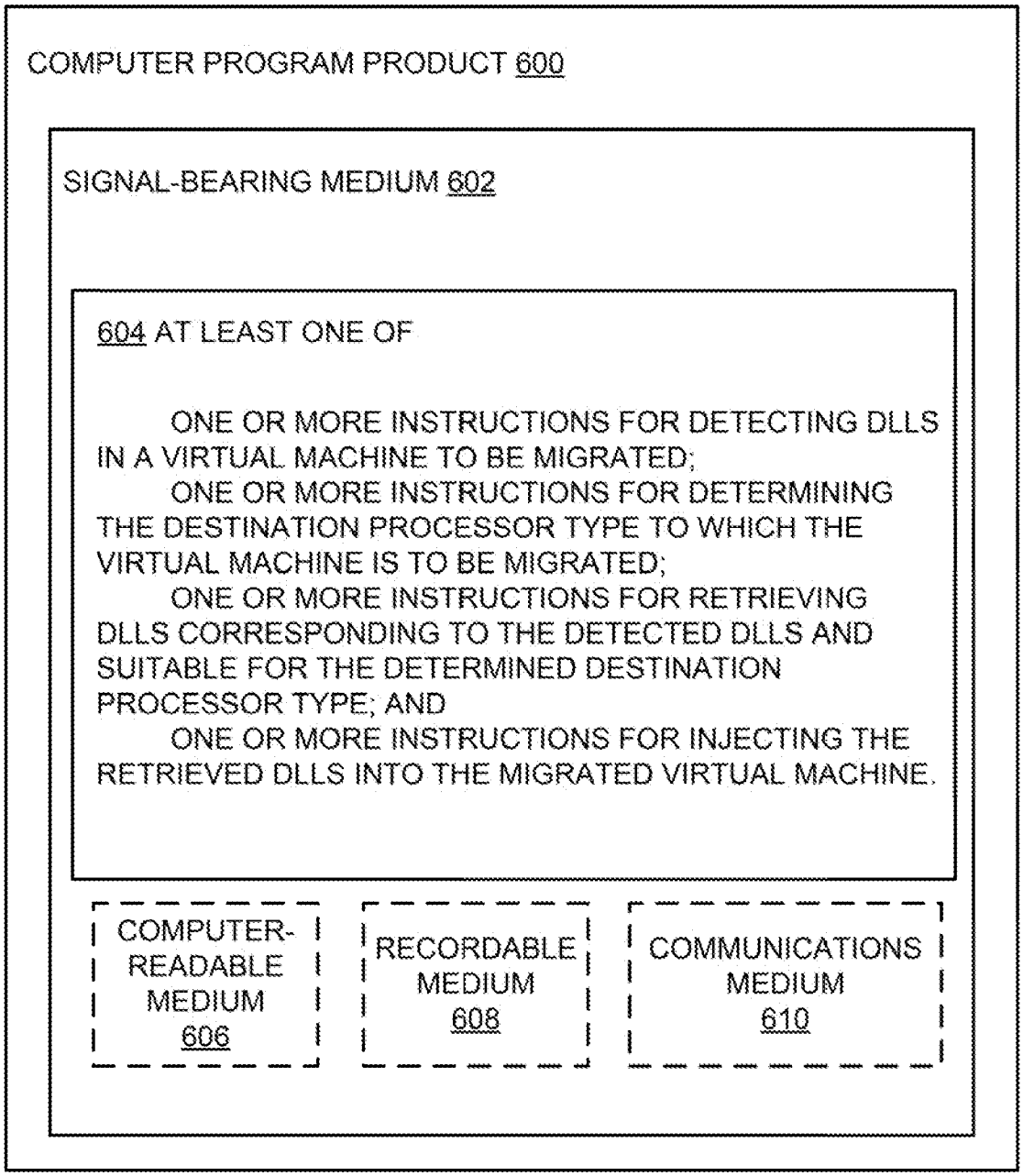
COMPUTER PROGRAM PRODUCT 600
SIGNAL-BEARING MEDIUM 602
604 AT LEAST ONE OF
ONE OR MORE INSTRUCTIONS FOR DETECTING DLLS IN A VIRTUAL MACHINE TO BE MIGRATED;
ONE OR MORE INSTRUCTIONS FOR DETERMINING THE DESTINATION PROCESSOR TYPE TO WHICH THE VIRTUAL MACHINE IS TO BE MIGRATED;
ONE OR MORE INSTRUCTIONS FOR RETRIEVING DLLS CORRESPONDING TO THE DETECTED DLLS AND SUITABLE FOR THE DETERMINED DESTINATION PROCESSOR TYPE; AND
ONE OR MORE INSTRUCTIONS FOR INJECTING THE RETRIEVED DLLS INTO THE MIGRATED VIRTUAL MACHINE.
COMPUTER-READABLE MEDIUM 606
RECORDABLE MEDIUM 608
COMMUNICATIONS MEDIUM 610

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 13/46584

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 9/455 (2013.01)
USPC - 718/1
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)
USPC: 718/1

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC: 707/774, 707/607, 707/713, 707/718, 707/769; 711/6, 711/104, 711/E12.001; 713/189; 717/151, 717/149, 717/144, 717/158, 717/150, 717/153, 717/148; 718/100, 718/102, 718/104 (keyword limited - see search terms below)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PatBase; Google; Google Scholar
Terms: virtual, machine, distributed, cloud, library, dll, resource, link, processor, node, client, specification, optimization, characteristics, performance, capability, migrate, allocate, transfer, assign, identifier, replace, substitute, tag, flag, hash, compression, matrix.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	US 2010/0138828 A1 (HANQUEZ et al.) 03 June 2010 (03.06.2010), entire document, especially abstract, Fig. 2A, para [0047], [0048], [0049], [0051], [0052], [0054], [0060], [0064], [0069].	1, 3, 5, 7-8, 10, 12-14, 16, 18, 20-21, 23, 25-27, 29, 31, 33-34, 36, 38-39 --- 2, 4, 6, 9, 11, 15, 17, 19, 22, 24, 28, 30, 32, 35, 37
Y	US 2012/0054367 A1 (RAMAKRISHNAN et al.) 01 March 2012 (01.03.2012), entire document, especially abstract, para [0004], [0019], [0028], [0095].	2, 15, 28
Y	US 5,634,070 A (ROBINSON) 27 May 1997 (27.05.1997), entire document, especially abstract, col. 1, ln 27-53, col. 3, ln 15-25, col. 4, ln 63 to col. 5, ln 13, col. 11, ln 26-62, col. 17, ln 1-17.	4, 17, 30
Y	US 2010/0313189 A1 (BERETTA et al.) 09 December 2010 (09.12.2010), entire document, especially abstract, para [0004], [0005], [0061], [0076], [0080].	6, 19, 32

☒ Further documents are listed in the continuation of Box C. ☐

* 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
24 November 2013 (24.11.2013)	23 DEC 2013
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

International application No.
PCT/US 13/46584

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2012/0011499 A1 (CONOVER et al.) 12 January 2012 (12.01.2012), entire document, especially abstract, para [0002], [0003], [0005].	9, 22, 35
Y	US 2010/0199104 A1 (VAN RIJNSWOU) 05 August 2010 (05.08.2010), entire document, especially abstract, para [0005], [0006], [0013], [0015].	11, 24, 37
A	US 2012/0089980 A1 (SHARP et al.) 12 April 2012 (12.04.2012), entire document.	1-39
A	US 2009/0070760 A1 (KHATRI et al.) 12 March 2009 (12.03.2009), entire document.	1-39
A	US 2003/0079213 A1 (CABILLIC et al.) 24 April 2003 (24.04.2003), entire document.	1-39
A	US 2011/0179176 A1 (RAVICHANDRAN et al.) 21 July 2011 (21.07.2011), entire document.	1-39

International application No.
PCT/US 13/46584

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☒ Claims Nos.: 40
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.