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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 March 2021 (25.03.2021)

WIPO | PCT

(10) International Publication Number
WO 2021/055281 A1

(51) International Patent Classification:
G06F 12/02 (2006.01) *G06F 12/1027* (2016.01)
G06F 12/1009 (2016.01) *G11C 8/18* (2006.01)

(21) International Application Number:
PCT/US2020/050714

(22) International Filing Date:
14 September 2020 (14.09.2020)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
16/573,490 17 September 2019 (17.09.2019) US

(71) Applicant: **MICRON TECHNOLOGY, INC.** [US/US]; 8000 South Federal Way, Boise, Idaho 83707 (US).

(72) Inventors: **CUREWITZ, Kenneth Marion**; 408 Ashland Court, Cameron Park, California 95682 (US). **GUNASEKARAN, Shivasankar**; 200 S Lexington Dr, Apt 935, Folsom, California 95630 (US). **AKEL, Ameen D.**; 3408 Grappa Way, Rancho Cordova, California 95670 (US). **WANG, Hongyu**; 1225 Meredith Way, Folsom, California 95630 (US). **ENO, Justin M.**; 3844 Yellowstone Lane, El Dorado Hills, California 95762 (US). **SWAMI, Shivam**; 200 South Lexington Drive, Apt 816, Folsom, California 95630 (US). **BRADSHAW, Samuel E.**; 6319 Surfside Way, Sacramento, California 95831 (US).

(74) Agent: **WARD, John P.** et al.; Greenberg Traurig, LLP, 77 West Wacker Drive, Suite 3100, Chicago, Illinois 60601 (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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, 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, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, 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,

(54) Title: ACCESSING STORED METADATA TO IDENTIFY MEMORY DEVICES IN WHICH DATA IS STORED

(57) Abstract: A computer system stores metadata that is used to identify physical memory devices that store randomly-accessible data for memory of the computer system. In one approach, access to memory in an address space is maintained by an operating system of the computer system. Stored metadata associates a first address range of the address space with a first memory device, and a second address range of the address space with a second memory device. The operating system manages processes running on the computer system by accessing the stored metadata. This management includes allocating memory based on the stored metadata so that data for a first process is stored in the first memory device, and data for a second process is stored in the second memory device.

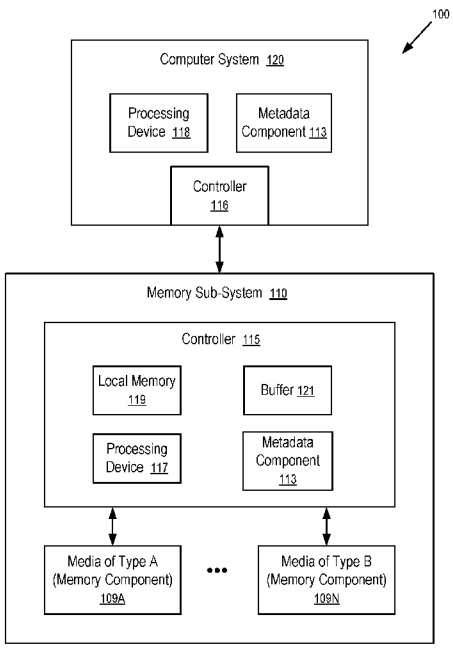

FIG. 1

WO 2021/055281 A1

[Continued on next page]

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))*

ACCESSING STORED METADATA TO IDENTIFY MEMORY DEVICES IN WHICH DATA IS STORED

RELATED APPLICATION

[0001] The present application claims priority to U.S. Pat. App. Ser. No. 16/573,490, filed Sep. 17, 2019 and entitled "ACCESSING STORED METADATA TO IDENTIFY MEMORY DEVICES IN WHICH DATA IS STORED," the entire disclosure of which is hereby incorporated herein by reference.

FIELD OF THE TECHNOLOGY

[0002] At least some embodiments disclosed herein relate to memory systems in general and more particularly, but not limited to accessing stored metadata to identify memory devices of a memory system in which data is stored.

BACKGROUND

[0003] Various types of memory devices can be used to store data in the main memory of a computer system. One type of volatile memory device is a dynamic random access memory (DRAM) device. Various types of non-volatile memory device can include a NAND flash memory device or a non-volatile random access memory (NVRAM) device.

[0004] In an operating system, memory management is responsible for managing the main memory of the computer system. The memory management tracks the status of memory locations in the main memory (e.g., a memory status of either allocated or free). Memory management further determines the allocation of memory among various processes running on the operating system. When memory is allocated to a process, the operating system determines the memory locations that will be assigned to the process.

[0005] In one approach, an operating system uses paged allocation to divide the main memory into fixed-sized units called page frames. A virtual address space of a software program is divided into pages having the same size. A hardware memory management unit maps pages to frames in physical memory. In a paged memory management approach, each process typically runs in its own address

space.

[0006] In some cases, a memory management unit (MMU) is called a paged memory management unit (PMMU). The MMU manages all memory references used by the operating system and performs the translation of virtual memory addresses to physical addresses. The MMU typically divides a virtual address space, which is the range of addresses used by the processor, into pages.

[0007] In some approaches, the MMU uses a page table containing page table entries to map virtual page numbers to physical page numbers in the main memory. In some cases, a cache of page table entries called a translation lookaside buffer (TLB) is used to avoid the need to access a page table stored in the main memory when a virtual address is mapped. When using virtual memory, a contiguous range of virtual addresses can be mapped to several non-contiguous blocks of physical memory.

[0008] More generally, a computer system can have one or more memory sub-systems. A memory sub-system can be a memory module, such as a dual in-line memory module (DIMM), a small outline DIMM (SO-DIMM), or a non-volatile dual in-line memory module (NVDIMM). A memory sub-system can be a storage system, such as a solid-state drive (SSD), or a hard disk drive (HDD). A memory sub-system can include one or more memory components that store data. The memory components can be, for example, non-volatile memory components and volatile memory components. Examples of memory components include memory integrated circuits. Some memory integrated circuits are volatile and require power to maintain stored data. Some memory integrated circuits are non-volatile and can retain stored data even when not powered. Examples of non-volatile memory include flash memory, Read-Only Memory (ROM), Programmable Read-Only Memory (PROM), Erasable Programmable Read-Only Memory (EPROM) and Electronically Erasable Programmable Read-Only Memory (EEPROM) memory, etc. Examples of volatile memory include Dynamic Random-Access Memory (DRAM) and Static Random-Access Memory (SRAM). In general, a computer system can utilize a memory sub-system to store data at the memory components and to retrieve data from the memory components.

[0009] For example, a computer system can include one or more memory sub-systems attached to the computer system. The computer system can have a central processing unit (CPU) in communication with the one or more memory sub-

systems to store and/or retrieve data and instructions. Instructions for a computer can include operating systems, device drivers, and application programs. An operating system manages resources in the computer and provides common services for application programs, such as memory allocation and time sharing of the resources. A device driver operates or controls a particular type of device in the computer; and the operating system uses the device driver to offer resources and/or services provided by the type of device. A central processing unit (CPU) of a computer system can run an operating system and device drivers to provide the services and/or resources to application programs. The central processing unit (CPU) can run an application program that uses the services and/or resources. For example, an application program implementing a type of application can instruct the central processing unit (CPU) to store data in the memory components of a memory sub-system and retrieve data from the memory components.

[0010] An operating system of a computer system can allow an application program to use virtual addresses of memory to store data in, or retrieve data from, memory components of one or more memory sub-systems of the computer system. The operating system maps the virtual addresses to physical addresses of one or more memory sub-systems connected to the central processing unit (CPU) of the computer system. The operating system translates the memory accesses specified at virtual addresses to the physical addresses of the memory sub-systems.

[0011] A virtual address space can be divided into pages. A page of virtual memory can be mapped to a page of physical memory in the memory sub-systems. The operating system can use a paging technique to access a page of memory in a storage device via a page of memory in a memory module. At different time instances, the same virtual page of memory in a memory module can be used as a proxy to access different physical pages of memory in the storage device or another storage device in the computer system.

[0012] A computer system can include a hypervisor (or virtual machine monitor) to create or provision virtual machines. A virtual machine is a computing device that is virtually implemented using the resources and services available in the computer system. The hypervisor presents the virtual machine to an operating system as if the components of virtual machine were dedicated physical components. A guest operating system runs in the virtual machine to manage resources and services available in the virtual machine, in a way similar to the host operating system running

in the computer system. The hypervisor allows multiple virtual machines to share the resources of the computer system and allows the virtual machines to operate on the computer substantially independently from each other.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The embodiments are illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate similar elements.

[0014] **FIG. 1** illustrates an example computer system having a memory sub-system, in accordance with some embodiments.

[0015] **FIG. 2** shows a mobile device that accesses different types of memory in a memory module using a memory bus, in accordance with some embodiments.

[0016] **FIG. 3** illustrates an example computer system that stores metadata used to access memory devices in a memory sub-system, in accordance with some embodiments.

[0017] **FIG. 4** shows a memory module configured for memory bus access by a host computer system to volatile and non-volatile memory of the memory module, in accordance with some embodiments.

[0018] **FIG. 5** shows a host operating system accessing a memory module using memory bus access, in accordance with at least some embodiments.

[0019] **FIG. 6** shows a method for managing memory for processes in an address space of a computer system based on stored metadata that associates virtual address ranges for the processes in the address space with physical addresses for memory devices in the computer system, in accordance with some embodiments.

[0020] **FIG. 7** is a block diagram of an example computer system in which embodiments of the present disclosure can operate.

DETAILED DESCRIPTION

[0021] At least some embodiments herein relate to accessing stored metadata to identify memory devices of a memory system in which data is stored. In various embodiments as discussed herein, the metadata can be stored and accessed by various types of computer systems. In one example, the computer system is a system-on-chip (SoC) device that stores metadata for managing memory usage by

one or more processes running on the SoC device. In one example, a mobile device uses a SoC device to manage allocation of main memory for one or more applications that are running on the mobile device.

[0022] Prior computer systems often use different types of memory devices for storing data. One type of memory device typically used is DRAM, which is generally considered to provide fast read and write access. DRAM is commonly used for storing data in the main memory of a computer system.

[0023] Other memory devices, such as flash memory, are typically considered to be slower than DRAM. For example, read or write access latency for a DRAM is typically significantly less than read or write access latency for flash memory. As a particular example, write access latency for some memory devices can be tens or even hundreds of times greater than for DRAM devices.

[0024] In prior computer systems that use different types of physical memory devices for storing data in main memory, a technical problem exists in which the processor does not have an awareness of how memory for various processes is actually mapped to the memory devices. For example, the processor may allocate a virtual address range to a process. However, the processor is unaware of how the virtual address range is mapped to the different memory devices.

[0025] In one example, if the virtual address range for a process is mapped to physical memory devices (e.g., flash memory) that are significantly slower than other memory devices (e.g., DRAM), then the process can be forced to run slowly due to an inability to rapidly access data from main memory that is needed for continuing execution of the process. For example, the process may need a response from main memory in order to continue data computations or other processing (e.g., a response that includes data for a read access request made by a processor to main memory to obtain data needed during execution by the process). If the data needed from main memory is actually stored in a slow physical memory device, then the processing is significantly delayed while waiting for the response.

[0026] Various embodiments of the present disclosure provide a technological solution to one or more of the above technical problems. In some embodiments, a computer system stores data regarding the latency of memory devices used by the computer system (e.g., memory devices used to provide main memory). In one example, the latencies of various memory regions that are visible to a processor of the computer system are known (e.g., as represented by information collected and/or

aggregated in stored metadata, as discussed below).

[0027] In some embodiments, the processor, an operating system, and/or an application (as programmed by a software designer) can initiate and/or perform actions by the computer system to avoid significant process delays due to slow memory access. For example, a high priority process that needs fast memory response can be configured to run in DRAM.

[0028] In another example, a priority for an application executing on a mobile device can be monitored. When the priority for the application increases (e.g., changes from low to high), then the processor and/or operating system can automatically transfer the application out of an address range of main memory that corresponds to a slow memory device, and move the application to a new address range that corresponds to a fast memory device.

[0029] In one example, memory device types include DRAM, NVRAM, and NAND flash. The priority of a process is determined by the processor (e.g., based on data usage patterns by the process). Based on stored metadata regarding address range mapping to these memory device types, the processor allocates the process to an address range having an appropriate memory latency. For example, the processor can determine whether a process has a low, intermediate, or high priority. Based on determining that the process has an intermediate priority, software and/or data associated with the process are stored in an address range corresponding to physical storage in the NVRAM memory device type, which has an intermediate latency.

[0030] In one example, the NVRAM device type is a 3D XPoint memory. In one example, the NVRAM device type can be resistive random-access memory, magnetoresistive RAM, phase-change RAM, and/or ferroelectric RAM. In one example, an NVRAM chip is used as main memory of a computer system (e.g., NVDIMM-P). In one example, an NVRAM device is implemented using non-volatile 3D XPoint memory in a DIMM package.

[0031] In another example, if the processor and/or operating system are not configured to automatically transfer the application to a different address range in response to a priority change, the software code of the application itself can be configured to read one or more values from the stored metadata. Based on the read values, the application itself can manage data storage so that data is preferentially stored in address ranges that correspond to faster memory devices.

In one example, the application can determine relative latencies of available memory devices in a computer system based on reading or otherwise being provided access to the stored metadata. In one example, the stored metadata specifies what data is on which memory device of various different memory devices having different latencies. By specifying the memory device in this manner, the application can determine a latency of access for particular data depending on the memory device being used to store the data.

[0032] In one example, an application on a mobile device reads stored metadata when requesting an allocation of main memory by an operating system (e.g., executing on a system-on-chip device). In one example, the application makes a request for an address range in main memory that corresponds to a particular type of memory device and/or a particular latency associated with memory read or write access.

[0033] In one example, the application reads or otherwise accesses the stored metadata to determine which memory is fast, and which memory is slow. In a first context of the mobile device, the application makes a request for allocation of fast memory. In a second context of the mobile device, the application makes a request for allocation of slow memory. In one example, in response to the detection of a predetermined context, the application initiates or makes a request for a change in allocation of memory. In one example, the application determines a change in context based on an updated query made to the stored metadata (e.g., by the processor), and/or data (e.g., operating characteristics of the mobile device) provided to the application by the processor of the computer system.

[0034] In one embodiment, a computer system includes a first memory device (e.g., DRAM) and a second memory device (e.g., NVRAM or NAND flash), and one or more processing devices (e.g., a CPU or system on a chip (SoC)). The computer system further includes memory containing instructions configured to instruct the one or more processing devices to: access memory in an address space maintained by an operating system, the accessing including accessing the first memory device and the second memory device using addresses in the address space; store metadata that associates a first address range of the address space with the first memory device, and a second address range of the address space with the second memory device; and manage, by the operating system based on the stored metadata, processes including a first process and a second process, where data for the first

process is stored in the first memory device, and data for the second process is stored in the second memory device.

[0035] In one embodiment, a computer system uses memory device types including DRAM, NVRAM, and NAND flash. In one example, the DRAM is faster than the NVRAM, which is faster than the NAND flash. The computer system is configured so that all three different types of memory can be accessed directly by a processor of the computer system using a virtual memory address. In one example, the processor communicates with a memory management unit to implement a virtual to physical address mapping system.

[0036] In one embodiment, an application is not pre-programmed or otherwise configured to manage or handle optimization of memory allocation based on different types of memory devices. For example, this may occur for legacy software programs. In this type of situation, an operating system can be configured to manage optimization of memory allocation for the application.

[0037] In one example, the operating system detects or otherwise determines one or more characteristics of an application. Based on the characteristics, the operating system uses the stored metadata to assign one or more address ranges in main memory to the application. In one example, the characteristics are determined based on information provided by the application itself (e.g., when the application is launched on a mobile device). In another example, the characteristics are provided by a computing device other than the computer system on which the application is being executed. In one example, a central repository is used to store and update a database or table of characteristics for applications. In one example, the central server provides an indication to the operating system regarding a type of physical memory to use.

[0038] In one embodiment, the operating system determines a context associated with a computer system and/or execution of an application. Based on this context, the operating system uses the stored metadata to assign one or more address ranges in main memory to the application.

[0039] In one embodiment, stored metadata is used for identifying devices in which data is stored. A memory sub-system has multiple physical memory devices (e.g., DRAM, NVRAM, and NAND flash) that can be addressed by a processor (e.g., SoC) in a memory address space. The metadata is used to specify which memory address regions are mapped to which physical memory devices. The metadata can

be loaded into the DRAM and/or the processor (e.g., loaded into a cache of the processor) to determine what data is on which device, and/or used to estimate the latency of access for the respective data.

[0040] In one embodiment, an application is executing on a mobile device having a processor that uses a main memory. The application requests that an operating system of the mobile device allocate a portion of main memory for use by the application. The allocated memory is in a logical/virtual memory space (e.g., memory addresses as seen by the programmer and by the execution units of the processor are virtual). In one embodiment, the virtual memory addresses are mapped to real/physical memory by page tables. A portion of the mapping data in the page tables is cached in a buffer of the processor. In one example, the buffer is a translation lookaside buffer (TLB).

[0041] In one embodiment, a computer system includes DRAM, NVRAM, and NAND flash memory devices. A processor of the computer system randomly accesses main memory by address. Addresses within the main memory correspond to physical locations of data storage on these three types of memory devices. In one example, each of the devices is accessed by the processor using a synchronous memory bus. In one example, the DRAM is synchronous dynamic random access memory (SDRAM) having an interface synchronous with a system bus carrying data between a CPU and a memory controller hub.

[0042] **FIG. 1** illustrates an example computing environment 100 having a memory sub-system 110, in accordance with some embodiments. The memory sub-system 110 can include media, such as memory components 109A to 109N. The memory components 109A to 109N can be volatile memory components, non-volatile memory components, or a combination of such. In some embodiments, the memory sub-system 110 is a memory module. Examples of a memory module include a DIMM and an NVDIMM. In some embodiments, the memory sub-system 110 is a hybrid memory/storage sub-system. In general, the computing environment 100 can include a computer system 120 that uses the memory sub-system 110. For example, the computer system 120 can write data to the memory sub-system 110 and read data from the memory sub-system 110.

[0043] The computer system 120 can be a computing device such as a mobile device, IoT device, desktop computer, laptop computer, network server, or such computing device that includes a memory and a processing device. The computer

system 120 can include or be coupled to the memory sub-system 110 so that the computer system 120 can read data from or write data to the memory sub-system 110. The computer system 120 can be coupled to the memory sub-system 110 via a physical host interface. As used herein, "coupled to" generally refers to a connection between components, which can be an indirect communicative connection or direct communicative connection (e.g., without intervening components), whether wired or wireless, including connections such as electrical, optical, magnetic, etc. Examples of a physical host interface include, but are not limited to, a serial advanced technology attachment (SATA) interface, a peripheral component interconnect express (PCIe) interface, universal serial bus (USB) interface, Fiber Channel, Serial Attached SCSI (SAS), a double data rate (DDR) memory bus, etc. The physical host interface can be used to transmit data between the computer system 120 and the memory sub-system 110. The computer system 120 can further utilize an NVM Express (NVMe) interface to access the memory components 109A to 109N when the memory sub-system 110 is coupled with the computer system 120 by the PCIe interface. The physical host interface can provide an interface for passing control, address, data, and other signals between the memory sub-system 110 and the computer system 120.

[0044] **FIG. 1** illustrates memory sub-system 110 as an example. In general, the computer system 120 can access multiple memory sub-systems via a shared communication connection, multiple separate communication connections, and/or a combination of communication connections. In one example, each memory sub-system 110 can be a different type of memory device that is randomly accessed by processing device 118 over a memory bus.

[0045] The computer system 120 includes the processing device 118 and a controller 116. The processing device 118 can be, for example, a microprocessor, a central processing unit (CPU), a processing core of a processor, an execution unit, etc. In some instances, the controller 116 can be referred to as a memory controller, a memory management unit, and/or an initiator. In one example, the controller 116 controls the communications over a bus coupled between the computer system 120 and one or more memory sub-systems 110.

[0046] In general, the controller 116 can send commands or requests to the memory sub-system 110 for desired access to memory components 109A to 109N. The controller 116 can further include interface circuitry to communicate with the

memory sub-system 110. The interface circuitry can convert responses received from memory sub-system 110 into information for the computer system 120.

[0047] The controller 116 of the computer system 120 can communicate with controller 115 of the memory sub-system 110 to perform operations such as reading data, writing data, or erasing data at the memory components 109A to 109N and other such operations. In some instances, the controller 116 is integrated within the same package of the processing device 118. In other instances, the controller 116 is separate from the package of the processing device 118. The controller 116 and/or the processing device 118 can include hardware such as one or more integrated circuits and/or discrete components, a buffer memory, a cache memory, or a combination thereof. The controller 116 and/or the processing device 118 can be a microcontroller, special purpose logic circuitry (e.g., a field programmable gate array (FPGA), an application specific integrated circuit (ASIC), etc.), or another suitable processor.

[0048] The memory components 109A to 109N can include any combination of various different types of non-volatile memory components and/or volatile memory components. An example of a non-volatile memory component includes a negative-AND (NAND) type flash memory. In one example, each of the memory components 109A to 109N can include one or more arrays of memory cells such as single level cells (SLCs) or multi-level cells (MLCs) (e.g., triple level cells (TLCs) or quad-level cells (QLCs)). In some embodiments, a particular memory component can include both an SLC portion and a MLC portion of memory cells. Each of the memory cells can store one or more bits of data (e.g., data blocks) used by the computer system 120.

[0049] Although non-volatile memory components such as NAND type flash memory are one example, the memory components 109A to 109N can be based on any other type of memory such as a volatile memory. In some embodiments, the memory components 109A to 109N can be, but are not limited to, random access memory (RAM), read-only memory (ROM), dynamic random access memory (DRAM), synchronous dynamic random access memory (SDRAM), phase change memory (PCM), magneto random access memory (MRAM), Spin Transfer Torque (STT)-MRAM, ferroelectric transistor random-access memory (FeTRAM), ferroelectric RAM (FeRAM), conductive bridging RAM (CBRAM), resistive random access memory (RRAM), oxide based RRAM (OxRAM), negative-or (NOR) flash

memory, electrically erasable programmable read-only memory (EEPROM), nanowire-based non-volatile memory, memory that incorporates memristor technology, and a 3D XPoint array of non-volatile memory cells. A cross-point array of non-volatile memory can perform bit storage based on a change of bulk resistance, in conjunction with a stackable cross-gridded data access array. Additionally, in contrast to many flash-based memories, cross-point non-volatile memory can perform a write in-place operation, where a non-volatile memory cell can be programmed without the non-volatile memory cell being previously erased. Furthermore, the memory cells of the memory components 109A to 109N can be grouped as memory pages or data blocks that can refer to a unit of the memory component used to store data.

[0050] The controller 115 of the memory sub-system 110 can communicate with the memory components 109A to 109N to perform operations such as reading data, writing data, or erasing data at the memory components 109A to 109N and other such operations (e.g., in response to commands scheduled on a command bus by controller 116). The controller 115 can include hardware such as one or more integrated circuits and/or discrete components, a buffer memory, or a combination thereof. The controller 115 can be a microcontroller, special purpose logic circuitry (e.g., a field programmable gate array (FPGA), an application specific integrated circuit (ASIC), etc.), or another suitable processor. The controller 115 can include a processing device 117 (processor) configured to execute instructions stored in local memory 119. In the illustrated example, the local memory 119 of the controller 115 includes an embedded memory configured to store instructions for performing various processes, operations, logic flows, and routines that control operation of the memory sub-system 110, including handling communications between the memory sub-system 110 and the computer system 120. In some embodiments, the local memory 119 can include memory registers storing memory pointers, fetched data, etc. The local memory 119 can also include read-only memory (ROM) for storing micro-code. While the example memory sub-system 110 in **FIG. 1** has been illustrated as including the controller 115, in another embodiment of the present disclosure, a memory sub-system 110 may not include a controller 115, and can instead rely upon external control (e.g., provided by an external host, or by a processor or controller separate from the memory sub-system).

[0051] In general, the controller 115 can receive commands or operations from

the computer system 120 and can convert the commands or operations into instructions or appropriate commands to achieve the desired access to the memory components 109A to 109N. The controller 115 can be responsible for other operations such as wear leveling operations, garbage collection operations, error detection and error-correcting code (ECC) operations, encryption operations, caching operations, and address translations between a logical block address and a physical block address that are associated with the memory components 109A to 109N. The controller 115 can further include host interface circuitry to communicate with the computer system 120 via the physical host interface. The host interface circuitry can convert the commands received from the computer system into command instructions to access the memory components 109A to 109N as well as convert responses associated with the memory components 109A to 109N into information for the computer system 120.

[0052] The memory sub-system 110 can also include additional circuitry or components that are not illustrated. In some embodiments, the memory sub-system 110 can include a cache or buffer 121 (e.g., DRAM or SRAM) and address circuitry (e.g., a row decoder and a column decoder) that can receive an address from the controller 115 and decode the address to access the memory components 109A to 109N.

[0053] The computing environment 100 includes a metadata component 113 in the computer system 120 that stores metadata used to identify memory devices in which data is stored (e.g., as discussed in various embodiments above). A portion of metadata component 113 can reside on computer system 120 and/or memory sub-system 110. In one example, a portion of the metadata is stored in local memory 119 and/or buffer 121. In one example, a portion of the metadata is alternatively and/or additionally stored in a cache of controller 116 (e.g., stored in a translation lookaside buffer).

[0054] In one example, the memory sub-system 110 can provide access for computer system 120 to data in different types of memory devices via a DDR or other type of synchronous memory bus. In one embodiment, the access is provided to data in NVRAM on a DIMM and to data in a DRAM. In one example, data is made accessible in a random access memory address space of the computer system 120 for access during host read/write requests made over the DDR memory bus.

[0055] In one example, computer system 120 sends a page-in request (for access to a page) to controller 115. In response to receiving the page-in request, controller 115 moves a page from a slow media such a non-volatile memory device to a volatile memory device (e.g., DRAM on memory sub-system 110).

[0056] In one example, computer system 120 sends a page-out request to controller 115. In response to receiving the page-out request, controller 115 moves data out of volatile memory (e.g., DRAM on memory sub-system 110) to non-volatile memory via buffer 121.

[0057] In some embodiments, the controller 116 and/or the processing device 118 in the computer system 120 includes at least a portion of the metadata component 113. For example, the controller 116 and/or the processing device 118 can include logic circuitry implementing the metadata component 113. For example, the processing device 118 (processor) of the computer system 120 can be configured to execute instructions stored in memory for performing operations that identify in which devices data is stored for the metadata component 113 as described herein. In some embodiments, the metadata component 113 is part of an operating system of the computer system 120, a device driver, or an application (e.g., an application executing on computer system 120).

[0058] In some embodiments, the controller 115 and/or the processing device 117 in the memory sub-system 110 includes at least a portion of the metadata component 113. For example, the controller 115 and/or the processing device 117 can include logic circuitry implementing the metadata component 113.

[0059] In one example, a central processing unit (CPU) can access memory in a memory system connected to the CPU. For example, the central processing unit (CPU) can be configured to access the memory based on a query to stored metadata of metadata component 113.

[0060] **FIG. 2** shows a mobile device 200 that accesses different types of memory in a memory module 205 using a memory bus 203, in accordance with some embodiments. **FIG. 2** shows a computer system having different types of memory. The computer system of **FIG. 2** includes a mobile device 200, and a memory module 205 connected to the mobile device 200 via memory bus 203. The memory module 205 is an example of the memory sub-system 110 illustrated in **FIG. 1**.

[0061] The mobile device 200 includes processing device 118, which can be a central processing unit or a microprocessor with one or more processing cores. The

mobile device 200 can have a cache memory 211. At least a portion of the cache memory 211 can be optionally integrated within the same integrated circuit package of the processing device 118.

[0062] The memory module 205 illustrated in **FIG. 2** has multiple types of memory (e.g., 221 and 223). For example, memory of type A 221 (e.g., DRAM) is faster than memory of type B 223 (e.g., NVRAM). For example, the memory bus 203 can be a double data rate bus. In general, several memory modules (e.g., 205) can be coupled to the memory bus 203.

[0063] The processing device 118 is configured via instructions (e.g., an operating system and/or one or more device drivers) to access a portion of memory in the computer system using metadata component 113. For example, memory of type B 223 (e.g., NVRAM) of the memory module 205 can be accessed or memory of type A 221 (e.g., DRAM) of the memory module 205 can be accessed. In one embodiment, memory of type B 223 of the memory module 205 is accessible only through addressing the memory of type A 221 of the memory module 205.

[0064] A controller 227 can be provided in the memory module 205 to manage data access to the memory of type A 221 and the memory of type B 223. In one embodiment, controller 227 multiplexes access to DRAM or NVRAM by mobile device 200 and memory module 205 when transferring data to or from buffer 121. In one example, memory bus 203 provides a host DDR channel as the DDR interface between mobile device 200 and memory module 205. In one example, once a page is retrieved from NVRAM memory into buffer 121, the page can be loaded for access by the mobile device via a conventional DDR4 slot (e.g., a host DDR channel).

[0065] In general, the memory sub-systems (e.g., 205) can include media, such as memory (e.g., 221, ..., 223). The memory (e.g., 221, ..., 223) can include volatile memory, non-volatile memory (NVM), and/or a combination of such. The processing device 118 can write data to each of the memory sub-systems (e.g., memory module 205) and read data from the memory sub-systems (e.g., memory module 205) directly or indirectly.

[0066] In one embodiment, memory module 205 provides memory bus access to non-volatile memory or volatile memory by using buffer 121. In one example, memory module 205 is a DIMM coupled to a mobile device 200 via a DDR bus. The storage media is, for example, cross-point memory.

[0067] In one embodiment, the mobile device communicates with the memory module via a communication channel for read/write operations (e.g., using a DDR4 bus). The mobile device can have one or more Central Processing Units (CPUs) to which computer peripheral devices, such as the memory module, may be attached via an interconnect, such as a computer bus (e.g., Serial AT Attachment (SATA), Peripheral Component Interconnect (PCI), PCI eXtended (PCI-X), PCI Express (PCIe)), a communication portion, and/or a computer network.

[0068] In one embodiment, the memory module can be used to store data for a processor in the non-volatile or volatile storage media. The memory module has a host interface that implements communications with the mobile device using the communication channel. In one embodiment, the memory module 205 has a controller 227 running, for example, firmware to perform operations responsive to communications from the processing device 118. In one example, the memory module includes volatile Dynamic Random-Access Memory (DRAM) and NVRAM. The DRAM and NVRAM store data accessible by the processing device 118 in a memory address space.

[0069] As illustrated, the computer system of **FIG. 2** is used to implement a mobile device. The processing device 118 can read data from or write data to the memory sub-systems (e.g., 205).

[0070] A physical host interface can be used to transmit data between the processing device 118 and the memory sub-system (e.g., 205). The physical host interface can provide an interface for passing control, address, data, and other signals between the memory sub-system (e.g., 205) and the processing device 118.

[0071] In general, a memory sub-system (e.g., memory module 205) includes a printed circuit board that connects a set of memory devices, such as memory integrated circuits, that provides the memory (e.g., 221, ..., 223). The memory (e.g., 221, ..., 223) on the memory sub-system (e.g., 205) can include any combination of the different types of non-volatile memory devices and/or volatile memory devices.

[0072] In some implementations, the memory (e.g., 221, ..., 223) can include, but are not limited to, random access memory (RAM), read-only memory (ROM), dynamic random access memory (DRAM), static random access memory (SRAM), synchronous dynamic random access memory (SDRAM), phase change memory (PCM), magneto random access memory (MRAM), negative-or (NOR) flash memory,

electrically erasable programmable read-only memory (EEPROM), and/or a cross-point array of non-volatile memory cells.

[0073] A memory sub-system (e.g., memory module 205) can have a controller (e.g., 227) that communicates with the memory (e.g., 221, …, 223) to perform operations such as reading data, writing data, or erasing data in the memory (e.g., 221, …, 223) and other such operations, in response to requests, commands or instructions from the processing device 118. The controller (e.g., 227) can include hardware such as one or more integrated circuits and/or discrete components, a buffer memory, or a combination thereof. The controller (e.g., 227) can be a microcontroller, special purpose logic circuitry (e.g., a field programmable gate array (FPGA), an application specific integrated circuit (ASIC), etc.), or another suitable processor. The controller (e.g., 227) can include one or more processors (processing devices) configured to execute instructions stored in local memory.

[0074] The local memory of the controller (e.g., 227) can include an embedded memory configured to store instructions for performing various processes, operations, logic flows, and routines that control operation of the memory sub-system (e.g., 205), including handling communications between the memory sub-system (e.g., 205) and the processing device 118, and other functions described in greater detail below. Local memory of the controller (e.g., 227) can include read-only memory (ROM) for storing micro-code and/or memory registers storing, e.g., memory pointers, fetched data, etc.

[0075] While the example memory sub-system 205 in **FIG. 2** has been illustrated as including controller 227, in another embodiment of the present disclosure, a memory sub-system (e.g., 205) may not include a controller (e.g., 227), and can instead rely upon external control (e.g., provided by a processor or controller separate from the memory sub-system (e.g., 205)).

[0076] In general, the controller (e.g., 227) can receive commands, requests or instructions from the processing device 118 in accordance with a standard communication protocol for the communication channel (e.g., 203) and can convert the commands, requests or instructions in compliance with the standard protocol into detailed instructions or appropriate commands within the memory sub-system (e.g., 205) to achieve the desired access to the memory (e.g., 221, …, 223). For example, the controller (e.g., 227) can be responsible for operations such as address translations between a logical address and a physical address that are associated

with the memory (e.g., 221, ..., 223). The controller (e.g., 227) can further include host interface circuitry to communicate with the processing device 118 via the physical host interface. The host interface circuitry can convert the commands received from the processing device 118 into command instructions to access the memory devices (e.g., 221, ..., 223) as well as convert responses associated with the memory devices (e.g., 221, ..., 223) into information for the processing device 118.

[0077] The memory sub-system (e.g., 205) can also include additional circuitry or components that are not illustrated. In some implementations, the memory sub-system (e.g., 205) can include a cache or buffer (e.g., DRAM) and address circuitry (e.g., a row decoder and a column decoder) that can receive an address from the controller (e.g., 227) and decode the address to access the memory (e.g., 221, ..., 223).

[0078] In one example, the memory bus 203 has one or more connectors to provide the memory sub-system (e.g., 205) with power and/or communicate with the memory sub-system (e.g., 205) via a predetermined protocol; and the memory sub-system (e.g., 205) has one or more connectors to receive the power, data and commands from the processing device 118. In one example, the processing device 118 can execute one or more operating systems to provide services, including memory access in which a portion of memory (e.g., a page stored in NVRAM) in the computer system is accessed using synchronous memory access.

[0079] **FIG. 3** illustrates an example computer system 300 that stores metadata 320 used to access memory devices in a memory sub-system 302, in accordance with some embodiments. The memory devices accessed in memory sub-system 302 include DRAM 304, NVRAM 306, and NAND flash 308. In one embodiment, computer system 300 alternatively and/or additionally stores metadata 322 in DRAM 304 that is used to access the memory devices.

[0080] In one embodiment, a processing device 310 of computer system 300 accesses memory in an address space. In one example, the memory is main memory used by processing device 310 when executing one or more applications. The processing device 310 accesses different memory devices using addresses in the address space.

[0081] In one embodiment, metadata 320, 322 associates a first address range of the address space with a memory device (e.g., DRAM 304) and a second address

range of the address space with a different memory device (e.g., NVRAM 306 or NAND flash 308). In one example, the latency of DRAM 304 is less than the latency of NVRAM 306 and NAND flash 308.

[0082] The applications executing on processing device 310 include application 312, which is configured to include a memory type 314. When the application 312 is initially launched, application 312 provides memory type 314 to processing device 310 along with a request for an allocation of memory in the main memory of computer system 300.

[0083] In response to the request for the allocation of memory, processing device 310 makes a query to metadata 320 and/or sends a query to metadata 322. Based on a result from one or both of these queries, processing device 310 allocates an address range in the address space to application 312.

[0084] In one embodiment, application 312 makes a request to processing device 310 for an indication of latency associated with the memory devices. Processing device 310 accesses metadata 320, 322 to obtain a result, and based on this result provides the indication of latency to application 312. In response to receiving the indication of latency, application 312 makes a request for an allocation of memory corresponding to a specific one of the memory devices, a memory device corresponding to memory type 314, or a request for an allocation of memory that has performance characteristics meeting at least one or more predetermined thresholds and/or requirements.

[0085] In one embodiment, metadata 322 stores data that associates an address range in a virtual address space with physical addresses in the memory devices of memory sub-system 302. In one example, metadata 322 stores address range 324 for NVRAM, and address range 326 for NAND flash. In one example, address range 324 maps a virtual or logical address of processing device 310 to a physical address of NVRAM 306. In one example, address range 326 maps a virtual or logical address of processing device 310 to a physical address of NAND flash 308. In one embodiment, metadata 320 or 322 stores one or more address ranges mapping addresses of processing device 310 for data stored in DRAM 304.

[0086] In one embodiment, metadata 322 is stored as part of page table 328, which provides a mapping of virtual addresses to physical addresses for a memory management unit 316 of computer system 300. Processing device 310 provides a virtual address to memory management unit 316, which accesses a translation

lookaside buffer 318 to obtain a physical address in one of the memory devices of memory sub-system 302.

[0087] In one embodiment, translation lookaside buffer 318 is a cache that stores a portion of the data from page table 328. In one example, buffer 318 stores a portion of metadata 322. In one embodiment, a portion of metadata 320 stored on computer system 300 is copied to translation lookaside buffer 318 for access by memory management unit 316 when accessing a memory device in memory sub-system 302.

[0088] In one embodiment, processing device 310 provides memory characteristics of the different memory devices to application 312. Application 312 makes a request for an allocation of memory based on the provided memory characteristics.

[0089] In one embodiment, processing device 310 receives a requested latency from application 312. An address range is allocated to the application 312 based on the requested latency.

[0090] In one embodiment, processing device 310 determines a priority associated with application 312. The address range allocated to application 312 is based on the determined priority. In one example, a faster memory device type is selected for use with the determined priority. Processing device 310 uses metadata 320, 322 to select an address range that physically stores data in a memory device of the selected faster memory device type.

[0091] In one embodiment, processing device 310 determines a change in priority of application 312. In one example, based on an increase in priority of application 312, processing device 310 changes a memory allocation that is used for application 312 in the address space. In one example, in response to the increase in priority, processing device 310 accesses metadata 320, 322 to determine an address range that corresponds to a faster physical memory device.

[0092] In one embodiment, processing device 310 determines a priority of application 312 based on observing characteristics associated with data access by application 312 in the address space. The observed characteristics can be used for allocating memory usage for application 312. In one embodiment, processing device 310 determines one or more latencies associated with physical memory devices. Metadata 320, 322 stores data regarding the determined one or more latencies, which can be used by processing device 310 when initially allocating

and/or changing an allocation of main memory.

[0093] **FIG. 4** shows a memory module 401 configured for memory bus access by a host computer system (not shown) to volatile memory 402 and non-volatile memory 404, in accordance with some embodiments. Memory module 401 is an example of memory sub-system 302 or memory module 205. In one example, memory module 401 is a hybrid DIMM. Volatile memory 402 is for example DRAM.

[0094] Memory module 401 uses multiplexer 408 to provide access to volatile memory 402 and non-volatile memory 404 by memory controller 416. Memory controller 416 is coupled to host interface 406 for handling read/write access by a host system. In one embodiment, multiplexer 408 is controlled based on signals received from memory controller 416 in response to receiving read or write commands from the host system via host interface 406.

[0095] In one example, a host system accesses a memory space (e.g., DRAM memory address space) on the memory module 401 (e.g., a DIMM). The DIMM exposes itself to the host as a channel of DRAM. In one embodiment, a hypervisor of the host system controls data movement on the DIMM. For example, a request is made for moving memory blocks in and out of the DRAM address space and exposing the DRAM pages to software running on the host. The software is, for example, executing in a virtual machine (VM).

[0096] In one example, a page in/out control path is provided for a driver to request a page that is currently in DRAM or in NVRAM. In one example, the NVRAM has a much larger capacity than the DRAM.

[0097] In one example, memory module 401 is implemented as a DIMM. The non-volatile memory 404 is provided by 3D XPoint memory packages. In one example, pages of data obtained from the 3D XPoint memory are copied in and out of a buffer (page in/page out).

[0098] In one example, the host system has read/write access to any DRAM or NVRAM address using normal DDR4 timing. For example, the host can generate arbitrary traffic per DDR4 rules during those times.

[0099] In one example, the full DDR address space of the non-volatile memory 404 is exposed to the host system. According to various embodiments, a controller (e.g., controller 116) of computer system 120 can operate in the same way (e.g., same read/write and refresh timing cycles) as it would for access to a conventional DRAM.

[0100] **FIG. 5** shows a host operating system 241 accessing a memory module 502 using a memory bus, in accordance with at least some embodiments. Memory module 502 includes a buffer 410. Buffer 410 is an example of buffer 121. In one example, buffer 410 stores metadata 322 and/or at least a portion of page table 328. Commands and data are received from a host operating system 241 via host interface 406. In one example, host operating system 241 executes on computer system 120 or 300.

[0101] In one embodiment, a device driver 247 (e.g., a back-end driver) is configured for memory access via a hypervisor 245. In one example, the system of **FIG. 5** is implemented in a computer system of **FIGs. 1-3**.

[0102] In one example, the host operating system 241 runs on the processing device 118 of the computer system of **FIG. 1** or **2**, or processing device 310 of **FIG. 3**. The host operating system 241 includes one or more device drivers (e.g., 247) that provide memory services using the memory (e.g., 221, ..., 223) of memory sub-systems, such as the memory module 205 or memory sub-system 302.

[0103] In one embodiment, back-end driver 247 maintains a mapping table 246. For example, the driver 247 maintains mapping table 246 to include a mapping for pages of data stored in DRAM 304, NVRAM 306, and NAND flash 308.

[0104] In one embodiment, the host operating system 241 includes a hypervisor 245 that provisions a virtual machine 249. The virtual machine 249 has virtual hardware implemented via the resources and services provided by the host operating system 241 using the hardware of a computing system of **FIGs. 1-3**. For example, the hypervisor 245 can provision virtual memory as part of the virtual machine 249 using a portion of the memory (e.g., 221, ..., 223) of memory sub-systems, such as the memory module 205.

[0105] The virtual machine 249 allows a guest operating system 243 to provide resources and/or services to applications (e.g., 251, ..., 253) running in the guest operating system 243, in a way as the operating system 243 running on a physical computing machine that has the same or similar set of hardware as provisioning in the virtual machine. The hypervisor 245 manages the mapping between the virtual hardware provisioned in the virtual machine and the services of hardware in the computing system managed by the host operating system 241.

[0106] A device driver 248 (e.g., a front-end driver) communicates with back-end driver 247. Driver 247 and driver 248 can communicate for memory ballooning

when additional DDR capacity (e.g., capacity in DRAM or NVRAM) is available.

[0107] **FIG. 5** illustrates an instance in which a virtual machine 249 is provisioned by the hypervisor 245. In general, the hypervisor 245 can provision several virtual machines (e.g., 249) that can run the same guest operating system 243, or different guest operating systems. Different sets of users and/or application programs can be assigned to use different virtual machines.

[0108] In some instances, the host operating system 241 is specialized to provide services for the provisioning of virtual machines and does not run other application programs. Alternatively, the host operating system 241 can provide additional services to support other application programs, such as applications (e.g., 251, ..., 253).

[0109] In one embodiment, the device driver 247 can be configured to request page-in of a page from slower memory (e.g., NVRAM) to faster memory (e.g., DRAM) for use by the virtual machine 249. This request can be made in response to a request from an application (e.g., application 312 of **FIG. 3**). After requesting the page, the page is made available in the faster memory by loading and/or transferring the page of data from the slower memory to the faster memory. In one example, processing device 310 moves the page from slower memory to faster memory based on address range information stored as metadata 320, 322. In one example, the slower memory can be the non-volatile memory 404 in the memory module 401 and the faster memory be the volatile memory 402 in the same memory module 401.

[0110] In one embodiment, the transfer of data (e.g., performed in response to a page-in request by the host operating system 241) is performed within a same memory sub-system, such as within the same memory module 401, to avoid or reduce congestion in communication channels connected to the processing device 118, such as the memory bus 203. For example, data can be copied from the slower memory 223 (e.g., NVRAM or NAND flash) in the memory module 205 to the faster memory 221 (e.g., DRAM) in the memory module 205, under the control of controller 227 in the memory module 205 in response to one or more commands, requests, and/or instructions from the device driver 247.

[0111] In one embodiment, the hypervisor 245 not only requests the device driver 247 to access a memory (e.g., 221, ..., 223) in a memory sub-system (e.g., memory module 205), but also provides the device driver 247 with information that can be

used in managing pages in the memory (e.g., 221, ..., 223, ..., or 225) to be used. In one example, the provided information includes stored metadata 320 or 322.

[0112] In one example, driver 247 is a memory mode driver used to access a memory address space in memory module 502 (e.g., a DIMM). Driver 247 has control over which pages are in volatile memory of the DIMM at any one time. In one approach, for example, the memory address space is exposed to the guest operating system 243. In this hypervisor environment, the guest operating system 243 sees the full storage capacity of the non-volatile memory (e.g., NVRAM and DRAM) in the DIMM.

[0113] In one example, only a number of pages that are in the DDR DRAM are actively paged-in via the host operating system 241. If there is a guest access to a page that is not present, a page fault path in a memory management unit (MMU) of the host system triggers the driver 247 to cause loading (page in) of a page. In one example, the page gets loaded in through control registers. Once the page is actually present in the DDR DRAM, then the driver 247 can set up MMU mapping (via mapping table 246) so that a guest application can directly read and write that data.

[0114] In one example, a front-end driver of a guest and a back-end driver of a host communicate regarding access to the memory address space. In one example, when deciding that pages are stale (e.g., not being used frequently based on a predetermined threshold), a request is made that a portion of data that is currently mapped in the DDR memory address space be pushed back out to the NVRAM memory (e.g., via an SRAM buffer) to make space available in the DRAM memory for other pages to be paged in. The back-end driver 247 communicates the page out request to move data from the DDR DRAM to the NVRAM memory.

[0115] In one embodiment, back-end driver 247 operates as a memory mode driver. Until driver 247 loads, there is no access to the NVRAM memory capacity of memory module 502. During this operation as a memory mode driver, the guest operating system 243 sees the memory as normal, and the driver 247 reserves DRAM pages on the memory module for page-in and page-out operations.

[0116] The driver 247 exposes the NVRAM memory to the guest operating system 243 and maintains the page mapping (e.g., in mapping table 246). For example, the driver 247 maintains the mapping between pages that are currently in the DRAM and pages that are on the NVRAM memory.

[0117] In one example, the driver 247 sets up memory management unit mapping tables at the host system to map any pages that are currently stored in DRAM. A page fault path from the guest can be used if there is an access outside of a mapped page to trigger a page-in request. A page-out request can be performed to maintain some memory space in the DRAM.

[0118] In one embodiment, operation is not restricted to memory mode. Driver 247 can also be operated as a block mode driver for which NVRAM memory is exposed as block mode storage.

[0119] In one embodiment, the memory module 502 maintains its own mapping table including a list of pages that are in an SRAM buffer (not shown). The memory module 502 can return a page-in completion signal to the host system once a page has been moved to the SRAM buffer. These permit reducing the latency for the host system to access those particular page(s). The driver 247 ensures that until its mapping is set up, the host will not access that page(s) until the page-in request completes.

[0120] In one embodiment, driver 247 implements a page out operation. In one example, this operation is triggered as a thread. This operation trades free pages back out of the DRAM memory and changes the mapping of valid pages.

[0121] **FIG. 6** shows a method for managing memory for processes in an address space of a computer system based on stored metadata that associates virtual address ranges for the processes in the address space with physical addresses for memory devices in the computer system, in accordance with some embodiments. For example, the method of **FIG. 6** can be implemented in the system of **FIGs. 1-3**.

[0122] The method of **FIG. 6** can be performed by processing logic that can include hardware (e.g., processing device, circuitry, dedicated logic, programmable logic, microcode, hardware of a device, integrated circuit, etc.), software (e.g., instructions run or executed on a processing device), or a combination thereof. In some embodiments, the method of **FIG. 6** is performed at least in part by one or more processing devices (e.g., processing device 310 of **FIG. 3**).

[0123] Although shown in a particular sequence or order, unless otherwise specified, the order of the processes can be modified. Thus, the illustrated embodiments should be understood only as examples, and the illustrated processes can be performed in a different order, and some processes can be performed in parallel. Additionally, one or more processes can be omitted in various

embodiments. Thus, not all processes are required in every embodiment. Other process flows are possible.

[0124] At block 601, an operating system maintains memory in an address space. The memory is accessed including accessing a first memory device and a second memory device using addresses in the address space. In one example, the operating system executes on processing device 310 of **FIG. 3**. In one example, the first memory device is DRAM 304, and the second memory device is NVRAM 306. In one example, the first memory device is NVRAM 306, and the second memory device is NAND flash 308.

[0125] At block 603, metadata is stored that associates a first address range of the address space with the first memory device. The metadata also associates a second address range of the address space with the second memory device. In one example, the stored metadata is metadata 320 and/or 322 of **FIG. 3**. In one example, the first address range is address range 324, and the second address range is address range 326.

[0126] At block 605, processes running in a computer system are managed based on the stored metadata. The processes include a first process and a second process. Data for the first process is stored in the first memory device, and data for the second process is stored in the second memory device. In one example, data for the first process is stored in address range 324, and data for the second process is stored in address range 326. In one example, data for the first process is stored in an address range of metadata 320, 322 that corresponds to physical memory storage in DRAM 304. In one example, the computer system is computer system 120 or 300.

[0127] In one embodiment, a method comprises: accessing, by a processing device (e.g., processing device 310 of **FIG. 3**) of a computer system, memory in an address space, wherein memory devices of the computer system are accessed by the processing device using addresses in the address space; storing metadata (e.g., metadata 320 and/or 322) that associates a first address range of the address space with a first memory device (e.g., DRAM 304), and a second address range of the address space with a second memory device (e.g., NVRAM 306), wherein a first latency of the first memory device is different from a second latency of the second memory device; and allocating, based on the stored metadata, the first address range to an application (e.g., application 312) executing on the computer system.

[0128] In one embodiment, allocating the first address range to the application is performed in response to a request by the application.

[0129] In one embodiment, the method further comprises: in response to a first request by the application, providing an indication that the first latency is greater than the second latency; receiving a second request made by the application based on the indication; and in response to receiving the second request, allocating the second address range to the application.

[0130] In one embodiment, the first latency is less than the second latency, and the metadata is stored in the first memory device.

[0131] In one embodiment, the computer system uses a memory bus to access the first memory device and the second memory device, and wherein the metadata is stored in the second memory device.

[0132] In one embodiment, the metadata is stored in the first memory device, and the method further comprises loading at least a portion of the metadata into a buffer (e.g., translation lookaside buffer 318), wherein the processing device queries the buffer to determine a physical address corresponding to a virtual address in the first address range.

[0133] In one embodiment, the computer system is a system-on-chip device, and the buffer is a translation lookaside buffer.

[0134] In one embodiment, the method further comprises: providing, to the application, memory characteristics of the first memory device and the second memory device; wherein allocating the first address range to the application is in response to a request made by the application based on the provided memory characteristics.

[0135] In one embodiment, the method further comprises receiving a requested latency from the application, wherein allocating the first address range to the application is further based on the requested latency.

[0136] In one embodiment, the method further comprises determining a priority associated with the application, wherein allocating the first address range to the application is further based on the priority.

[0137] In one embodiment, the first latency is less than the second latency; prior to allocating the first address range to the application, the application is allocated to the second address range; and allocating the first address range to the application is performed in response to determining an increase in a priority associated with the

application.

[0138] In one embodiment, determining the increase in the priority associated with the application is based on one or more observations regarding data access by the application in the address space.

[0139] In one embodiment, the method further comprises determining, by the processing device, latencies associated with the memory devices, wherein storing the metadata further comprises storing the determined latencies.

[0140] In one embodiment, a system comprises: a first memory device; a second memory device; at least one processing device; and memory containing instructions configured to instruct the at least one processing device to: access memory in an address space maintained by an operating system, the accessing including accessing the first memory device and the second memory device using addresses in the address space; store metadata that associates a first address range of the address space with the first memory device, and a second address range of the address space with the second memory device; and manage, by the operating system based on the stored metadata, processes including a first process and a second process, wherein data for the first process is stored in the first memory device, and data for the second process is stored in the second memory device.

[0141] In one embodiment, the first process has a first priority, the second process has a second priority, and the first memory device is selected to store the data for the first process in response to determining that the first priority is higher than the second priority.

[0142] In one embodiment, the first process corresponds to a first application; the instructions are further configured to instruct the at least one processing device to receive a request from the first application that indicates a type of memory to use for storing data; and the first memory device is selected to store the data for the first process based on the indicated type of memory.

[0143] In one embodiment, the system further comprises a buffer to store the metadata, wherein the operating system receives a virtual address in the first address range from the first process, and accesses the buffer to determine a physical address of the first memory device corresponding to the virtual address.

[0144] In one embodiment, a read latency of the first memory device is less than a read latency of the second memory device, and the instructions are further configured to instruct the at least one processing device to store the metadata in the

first memory device.

[0145] In one embodiment, the system further comprises a memory management unit (e.g., memory management unit 316) configured to, when accessing the stored data for the first process, map a virtual address in the first address range to a physical address in the first memory device.

[0146] In one embodiment, a non-transitory machine-readable storage medium stores instructions which, when executed on at least one processing device, cause the at least one processing device to at least: access memory in an address space, wherein memory devices of a computer system are accessed by the at least one processing device using addresses in the address space; store metadata that associates a first address range of the address space with a first memory device, and a second address range of the address space with a second memory device; provide, to an application executing on the computer system, first data indicating that a first latency of the first memory device is less than a second latency of the second memory device; in response to providing the first data to the application, receive a request from the application that second data associated with the application be stored in the first memory device; in response to a request by the application to store the second data, query the stored metadata to provide a result; and store, based on the result, the second data in the first memory device.

[0147] **FIG. 7** is a block diagram of an example computer system in which embodiments of the present disclosure can operate. **FIG. 7** illustrates an example machine of a computer system 600 within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, can be executed. In some embodiments, the computer system 600 can correspond to a host system (e.g., the computer system 120 of **FIG. 1**) that includes, is coupled to, or utilizes a memory sub-system (e.g., the memory sub-system 110 of **FIG. 1**) or can be used to perform the operations of a metadata component 113 (e.g., to execute instructions to perform operations corresponding to the metadata component 113 described with reference to **FIGS. 1-6**). In alternative embodiments, the machine can be connected (e.g., networked) to other machines in a LAN, an intranet, an extranet, and/or the Internet. The machine can operate in the capacity of a server or a client machine in client-server network environment, as a peer machine in a peer-to-peer (or distributed) network environment, or as a server or a client machine in a cloud computing infrastructure or environment.

[0148] The machine can be a personal computer (PC), a tablet PC, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a web appliance, a server, a network router, a switch or bridge, an Internet of Things (IOT) device, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine. Further, while a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

[0149] The example computer system 600 includes a processing device 602, a main memory 604 (e.g., read-only memory (ROM), flash memory, dynamic random access memory (DRAM) such as synchronous DRAM (SDRAM) or Rambus DRAM (RDRAM), static random access memory (SRAM), etc.), and a data storage system 618, which communicate with each other via a bus 630 (which can include multiple buses).

[0150] Processing device 602 represents one or more general-purpose processing devices such as a microprocessor, a central processing unit, or the like. More particularly, the processing device can be a complex instruction set computing (CISC) microprocessor, reduced instruction set computing (RISC) microprocessor, very long instruction word (VLIW) microprocessor, or a processor implementing other instruction sets, or processors implementing a combination of instruction sets. Processing device 602 can also be one or more special-purpose processing devices such as an application specific integrated circuit (ASIC), a field programmable gate array (FPGA), a digital signal processor (DSP), network processor, or the like. The processing device 602 is configured to execute instructions 626 for performing the operations and steps discussed herein. The computer system 600 can further include a network interface device 608 to communicate over the network 620.

[0151] The data storage system 618 can include a machine-readable storage medium 624 (also known as a computer-readable medium) on which is stored one or more sets of instructions 626 or software embodying any one or more of the methodologies or functions described herein. The instructions 626 can also reside, completely or at least partially, within the main memory 604 and/or within the processing device 602 during execution thereof by the computer system 600, the main memory 604 and the processing device 602 also constituting machine-readable storage media. The machine-readable storage medium 624, data storage system

618, and/or main memory 604 can correspond to the memory sub-system 110 of **FIG. 1**.

[0152] In one embodiment, the instructions 626 include instructions to implement functionality corresponding to a metadata component 113 (e.g., the metadata component 113 described with reference to **FIGS. 1-6**). While the machine-readable storage medium 624 is shown in an example embodiment to be a single medium, the term "machine-readable storage medium" should be taken to include a single medium or multiple media that store the one or more sets of instructions.

[0153] **Closing**

[0154] The disclosure includes various devices which perform the methods and implement the systems described above, including data processing systems which perform these methods, and computer readable media containing instructions which when executed on data processing systems cause the systems to perform these methods.

[0155] The description and drawings are illustrative and are not to be construed as limiting. Numerous specific details are described to provide a thorough understanding. However, in certain instances, well-known or conventional details are not described in order to avoid obscuring the description. References to one or an embodiment in the present disclosure are not necessarily references to the same embodiment; and, such references mean at least one.

[0156] Reference in this specification to "one embodiment" or "an embodiment" means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the disclosure. The appearances of the phrase "in one embodiment" in various places in the specification are not necessarily all referring to the same embodiment, nor are separate or alternative embodiments mutually exclusive of other embodiments. Moreover, various features are described which may be exhibited by some embodiments and not by others. Similarly, various requirements are described which may be requirements for some embodiments but not other embodiments.

[0157] In this description, various functions and operations may be described as being performed by or caused by software code to simplify description. However, those skilled in the art will recognize what is meant by such expressions is that the functions result from execution of the code by one or more processors, such as a microprocessor, Application-Specific Integrated Circuit (ASIC), graphics processor,

and/or a Field-Programmable Gate Array (FPGA). Alternatively, or in combination, the functions and operations can be implemented using special purpose circuitry (e.g., logic circuitry), with or without software instructions. Embodiments can be implemented using hardwired circuitry without software instructions, or in combination with software instructions. Thus, the techniques are not limited to any specific combination of hardware circuitry and software, nor to any particular source for the instructions executed by a computing device.

[0158] While some embodiments can be implemented in fully functioning computers and computer systems, various embodiments are capable of being distributed as a computing product in a variety of forms and are capable of being applied regardless of the particular type of machine or computer-readable media used to actually effect the distribution.

[0159] At least some aspects disclosed can be embodied, at least in part, in software. That is, the techniques may be carried out in a computing device or other system in response to its processor, such as a microprocessor, executing sequences of instructions contained in a memory, such as ROM, volatile RAM, non-volatile memory, cache or a remote storage device.

[0160] Routines executed to implement the embodiments may be implemented as part of an operating system, middleware, service delivery platform, SDK (Software Development Kit) component, web services, or other specific application, component, program, object, module or sequence of instructions referred to as "computer programs." Invocation interfaces to these routines can be exposed to a software development community as an API (Application Programming Interface). The computer programs typically comprise one or more instructions set at various times in various memory and storage devices in a computer, and that, when read and executed by one or more processors in a computer, cause the computer to perform operations necessary to execute elements involving the various aspects.

[0161] A machine readable medium can be used to store software and data which when executed by a computing device causes the device to perform various methods. The executable software and data may be stored in various places including, for example, ROM, volatile RAM, non-volatile memory and/or cache. Portions of this software and/or data may be stored in any one of these storage devices. Further, the data and instructions can be obtained from centralized servers or peer to peer networks. Different portions of the data and instructions can be

obtained from different centralized servers and/or peer to peer networks at different times and in different communication sessions or in a same communication session. The data and instructions can be obtained in entirety prior to the execution of the applications. Alternatively, portions of the data and instructions can be obtained dynamically, just in time, when needed for execution. Thus, it is not required that the data and instructions be on a machine readable medium in entirety at a particular instance of time.

[0162] Examples of computer-readable media include but are not limited to recordable and non-recordable type media such as volatile and non-volatile memory devices, read only memory (ROM), random access memory (RAM), flash memory devices, solid-state drive storage media, removable disks, magnetic disk storage media, optical storage media (e.g., Compact Disk Read-Only Memory (CD ROMs), Digital Versatile Disks (DVDs), etc.), among others. The computer-readable media may store the instructions.

[0163] In general, a tangible or non-transitory machine readable medium includes any mechanism that provides (e.g., stores) information in a form accessible by a machine (e.g., a computer, mobile device, network device, personal digital assistant, manufacturing tool, any device with a set of one or more processors, etc.).

[0164] In various embodiments, hardwired circuitry may be used in combination with software and firmware instructions to implement the techniques. Thus, the techniques are neither limited to any specific combination of hardware circuitry and software nor to any particular source for the instructions executed by a computing device.

[0165] Although some of the drawings illustrate a number of operations in a particular order, operations which are not order dependent may be reordered and other operations may be combined or broken out. While some reordering or other groupings are specifically mentioned, others will be apparent to those of ordinary skill in the art and so do not present an exhaustive list of alternatives. Moreover, it should be recognized that the stages could be implemented in hardware, firmware, software or any combination thereof.

[0166] In the foregoing specification, the disclosure has been described with reference to specific exemplary embodiments thereof. It will be evident that various modifications may be made thereto without departing from the broader spirit and scope as set forth in the following claims. The specification and drawings are,

accordingly, to be regarded in an illustrative sense rather than a restrictive sense.

[0167] Various embodiments set forth herein can be implemented using a wide variety of different types of computing devices. As used herein, examples of a “computing device” include, but are not limited to, a server, a centralized computing platform, a system of multiple computing processors and/or components, a mobile device, a user terminal, a vehicle, a personal communications device, a wearable digital device, an electronic kiosk, a general purpose computer, an electronic document reader, a tablet, a laptop computer, a smartphone, a digital camera, a residential domestic appliance, a television, or a digital music player. Additional examples of computing devices include devices that are part of what is called "the internet of things" (IOT). Such "things" may have occasional interactions with their owners or administrators, who may monitor the things or modify settings on these things. In some cases, such owners or administrators play the role of users with respect to the "thing" devices. In some examples, the primary mobile device (e.g., an Apple iPhone) of a user may be an administrator server with respect to a paired “thing” device that is worn by the user (e.g., an Apple watch).

[0168] In some embodiments, the computing device can be a computer or host system, which is implemented, for example, as a desktop computer, laptop computer, network server, mobile device, or other computing device that includes a memory and a processing device. The host system can include or be coupled to a memory sub-system so that the host system can read data from or write data to the memory sub-system. The host system can be coupled to the memory sub-system via a physical host interface. In general, the host system can access multiple memory sub-systems via a same communication connection, multiple separate communication connections, and/or a combination of communication connections.

CLAIMS

What is claimed is:

1. A method comprising:
   accessing, by a processing device of a computer system, memory in an address space, wherein memory devices of the computer system are accessed by the processing device using addresses in the address space;
   storing metadata that associates a first address range of the address space with a first memory device, and a second address range of the address space with a second memory device, wherein a first latency of the first memory device is different than a second latency of the second memory device; and
   allocating, based on the stored metadata, the first address range to an application executing on the computer system.

2. The method of claim 1, wherein allocating the first address range to the application is performed in response to a request by the application.

3. The method of claim 1, further comprising:
   in response to a first request by the application, providing an indication that the first latency is greater than the second latency;
   receiving a second request made by the application based on the indication; and
   in response to receiving the second request, allocating the second address range to the application.

4. The method of claim 1, wherein the first latency is less than the second latency, and the metadata is stored in the first memory device.

5. The method of claim 1, wherein the computer system uses a memory bus to access the first memory device and the second memory device, and wherein the metadata is stored in the first memory device or the second memory device.

6. The method of claim 1, wherein the metadata is stored in the first memory device, the method further comprising loading at least a portion of the metadata into a buffer, wherein the processing device queries the buffer to determine a physical address corresponding to a virtual address in the first address range.

7. The method of claim 6, wherein the computer system is a system-on-chip device, and the buffer is a translation lookaside buffer.

8. The method of claim 1, further comprising:
providing, to the application, memory characteristics of the first memory device and the second memory device;
wherein allocating the first address range to the application is in response to a request made by the application based on the provided memory characteristics.

9. The method of claim 1, further comprising receiving a requested latency from the application, wherein allocating the first address range to the application is further based on the requested latency.

10. The method of claim 1, further comprising determining a priority associated with the application, wherein allocating the first address range to the application is further based on the priority.

11. The method of claim 1, wherein:
the first latency is less than the second latency;
prior to allocating the first address range to the application, the application is allocated to the second address range; and
allocating the first address range to the application is performed in response to determining an increase in a priority associated with the application.

12. The method of claim 11, wherein determining the increase in the priority associated with the application is based on one or more observations regarding data access by the application in the address space.

13. The method of claim 1, further comprising determining, by the processing

device, latencies associated with the memory devices, wherein storing the metadata further comprises storing the determined latencies.

14. A system comprising:
a first memory device;
a second memory device;
at least one processing device; and
memory containing instructions configured to instruct the at least one processing device to:
access memory in an address space maintained by an operating system, the accessing including accessing the first memory device and the second memory device using addresses in the address space;
store metadata that associates a first address range of the address space with the first memory device, and a second address range of the address space with the second memory device; and
manage, by the operating system based on the stored metadata, processes including a first process and a second process, wherein data for the first process is stored in the first memory device, and data for the second process is stored in the second memory device.

15. The system of claim 14, wherein the first process has a first priority, the second process has a second priority, and the first memory device is selected to store the data for the first process in response to determining that the first priority is higher than the second priority.

16. The system of claim 14, wherein:
the first process corresponds to a first application;
the instructions are further configured to instruct the at least one processing device to receive a request from the first application that indicates a type of memory to use for storing data; and
the first memory device is selected to store the data for the first process based on the indicated type of memory.

17. The system of claim 14, further comprising a buffer to store the metadata, wherein the operating system receives a virtual address in the first address range from the first process, and accesses the buffer to determine a physical address of the first memory device corresponding to the virtual address.

18. The system of claim 14, wherein a read latency of the first memory device is less than a read latency of the second memory device, and wherein the instructions are further configured to instruct the at least one processing device to store the metadata in the first memory device.

19. The system of claim 14, further comprising a memory management unit configured to, when accessing the stored data for the first process, map a virtual address in the first address range to a physical address in the first memory device.

20. A non-transitory machine-readable storage medium storing instructions which, when executed on at least one processing device, cause the at least one processing device to at least:

access memory in an address space, wherein memory devices of a computer system are accessed by the at least one processing device using addresses in the address space;

store metadata that associates a first address range of the address space with a first memory device, and a second address range of the address space with a second memory device;

provide, to an application executing on the computer system, first data indicating that a first latency of the first memory device is less than a second latency of the second memory device;

in response to providing the first data to the application, receive a request from the application that second data associated with the application be stored in the first memory device;

in response to a request by the application to store the second data, query the stored metadata to provide a result; and

store, based on the result, the second data in the first memory device.

PAGE INTENTIONALLY LEFT BLANK

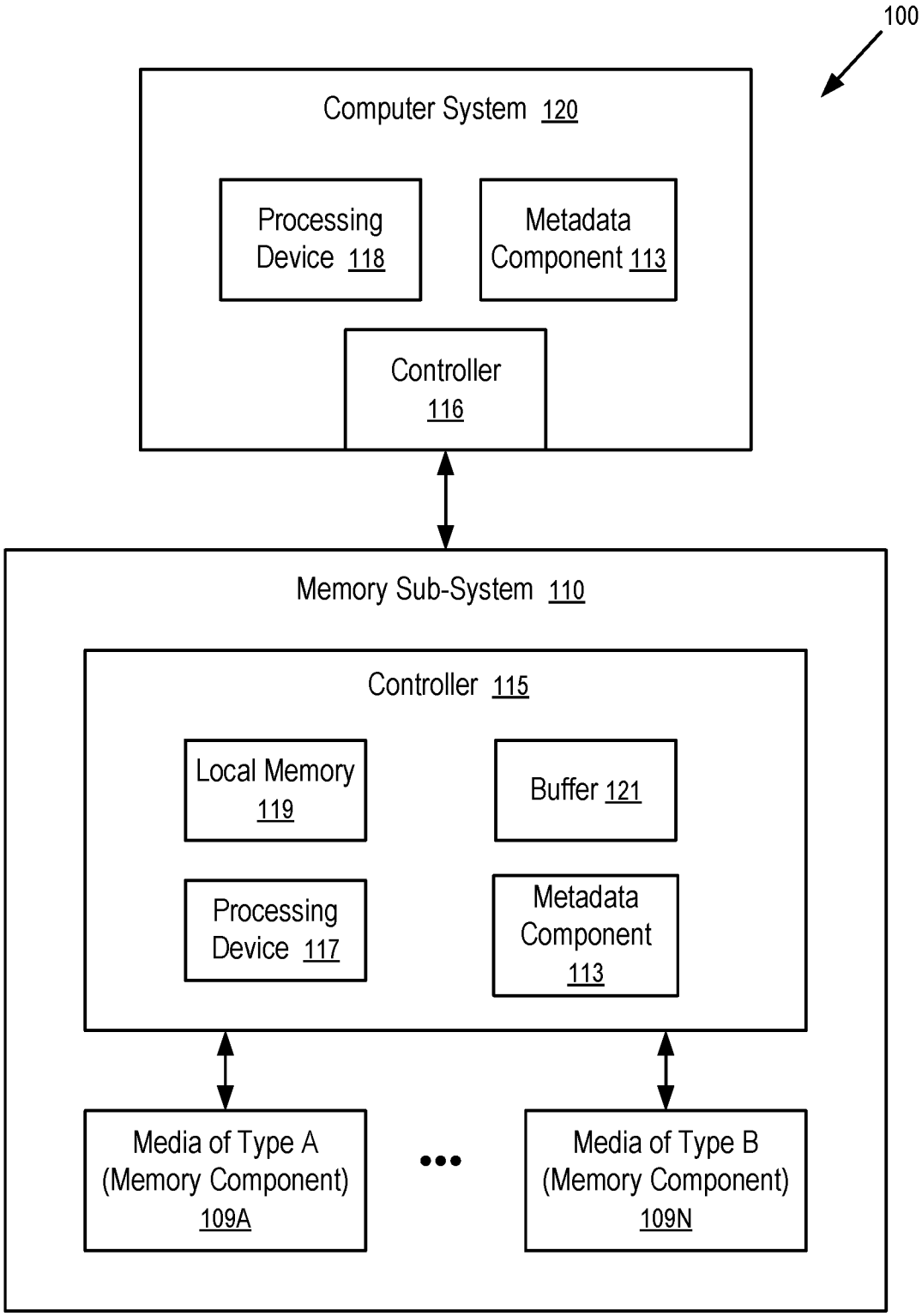
100
Computer System 120
Processing Device 118
Metadata Component 113
Controller 116
Memory Sub-System 110
Controller 115
Local Memory 119
Buffer 121
Processing Device 117
Metadata Component 113
Media of Type A (Memory Component) 109A
Media of Type B (Memory Component) 109N

FIG. 1

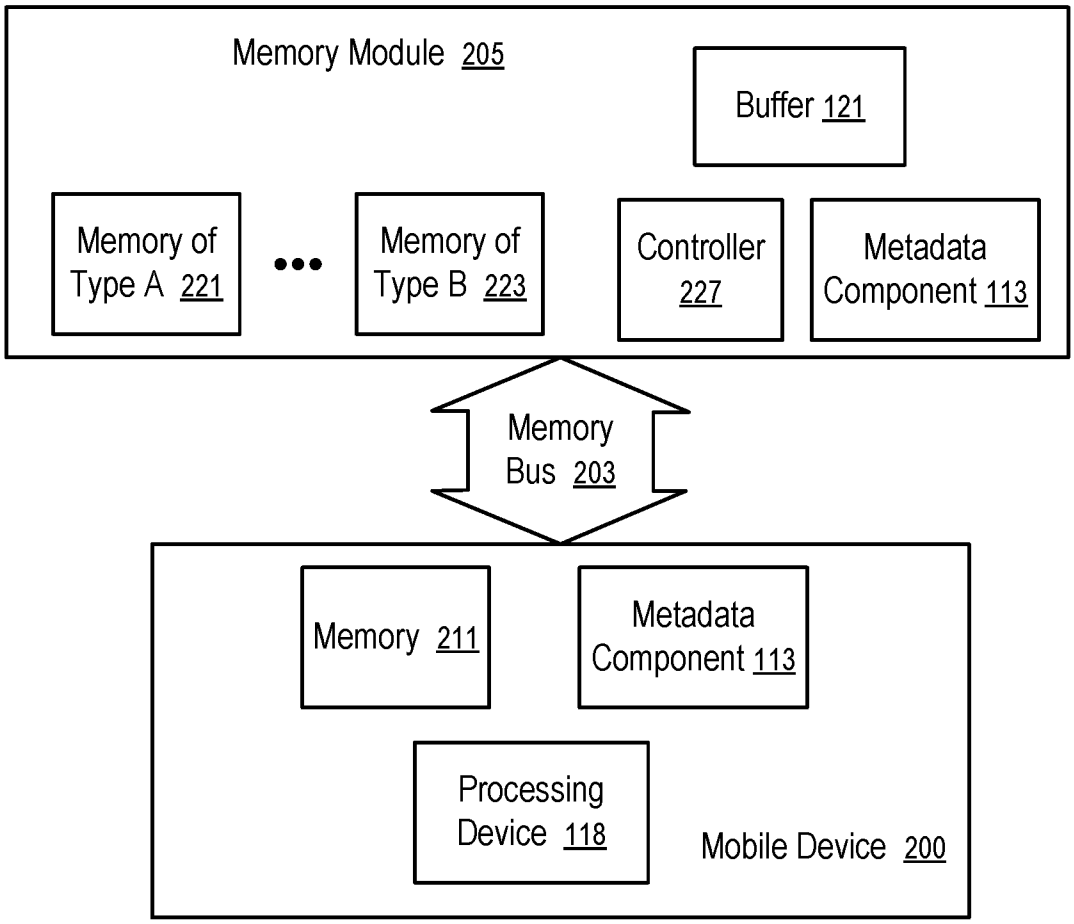
Memory Module 205
Buffer 121
Memory of Type A 221
Memory of Type B 223
Controller 227
Metadata Component 113
Memory Bus 203
Memory 211
Metadata Component 113
Processing Device 118
Mobile Device 200

FIG. 2

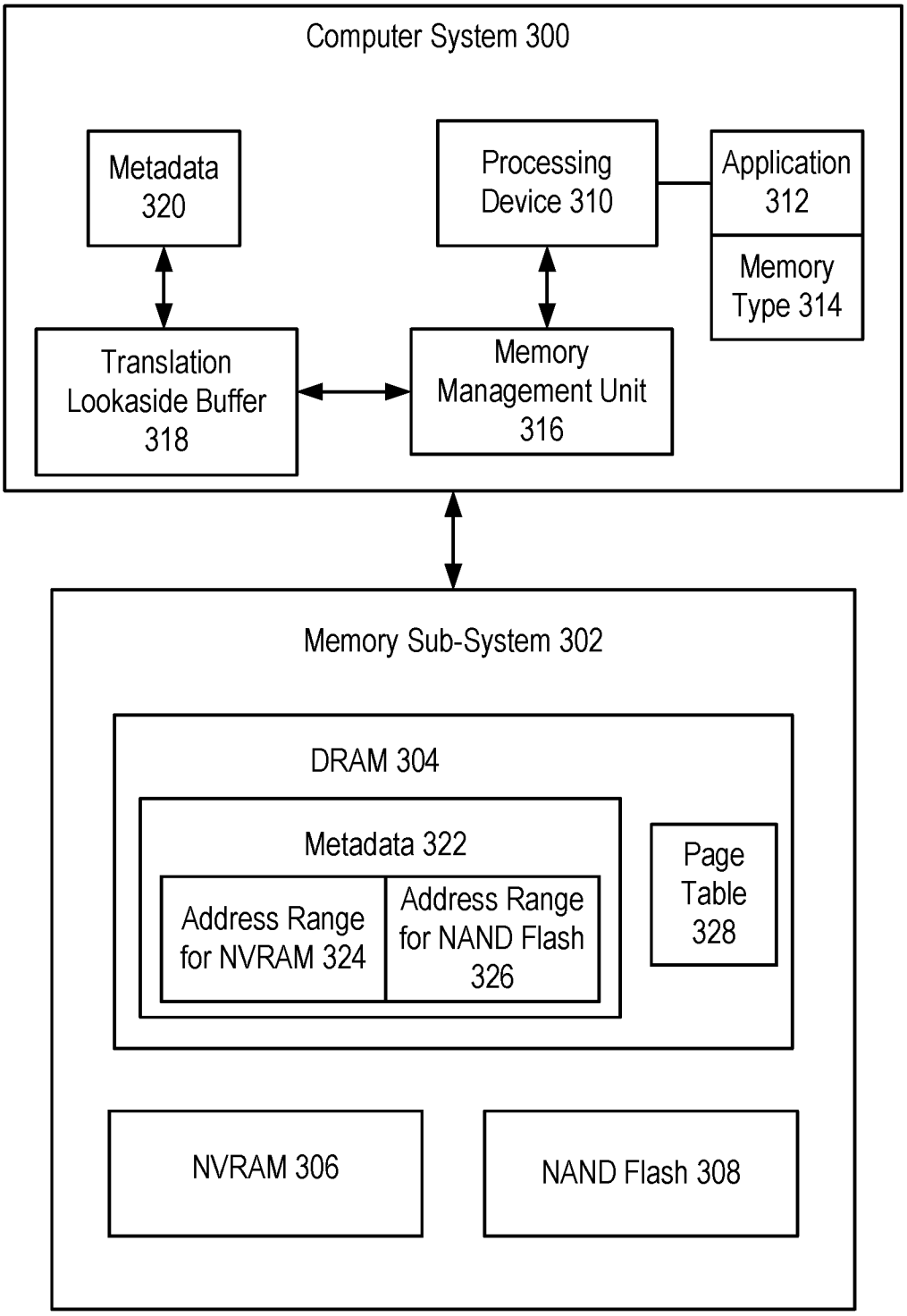
Computer System 300
Metadata 320
Processing Device 310
Application 312
Memory Type 314
Translation Lookaside Buffer 318
Memory Management Unit 316
Memory Sub-System 302
DRAM 304
Metadata 322
Address Range for NVRAM 324
Address Range for NAND Flash 326
Page Table 328
NVRAM 306
NAND Flash 308

FIG. 3

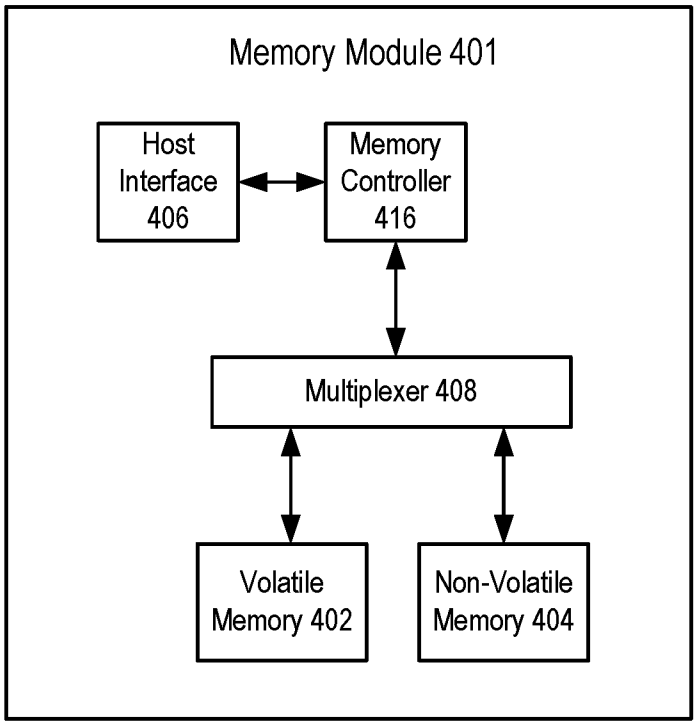
Memory Module 401
Host Interface 406
Memory Controller 416
Multiplexer 408
Volatile Memory 402
Non-Volatile Memory 404

FIG. 4

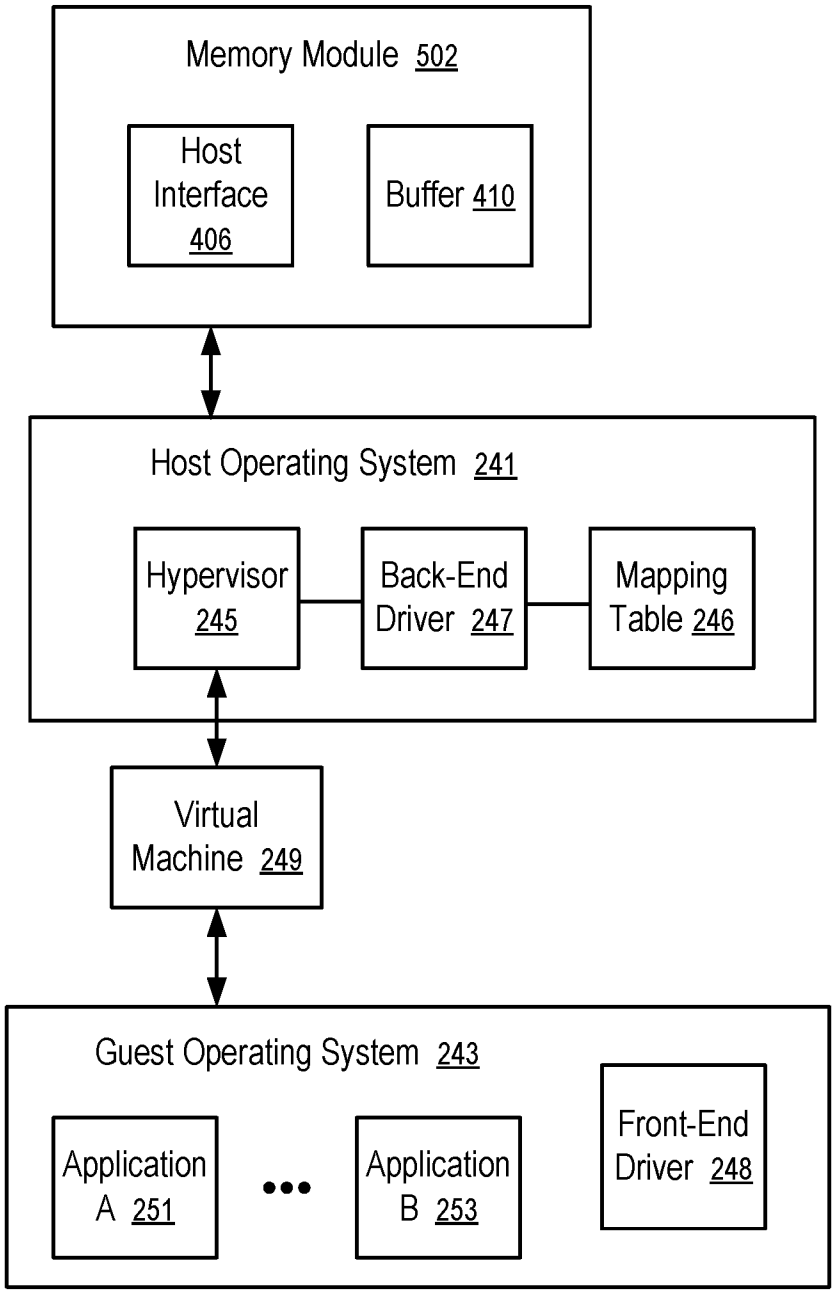
Memory Module 502
Host Interface 406
Buffer 410
Host Operating System 241
Hypervisor 245
Back-End Driver 247
Mapping Table 246
Virtual Machine 249
Guest Operating System 243
Application A 251
Application B 253
Front-End Driver 248

FIG. 5

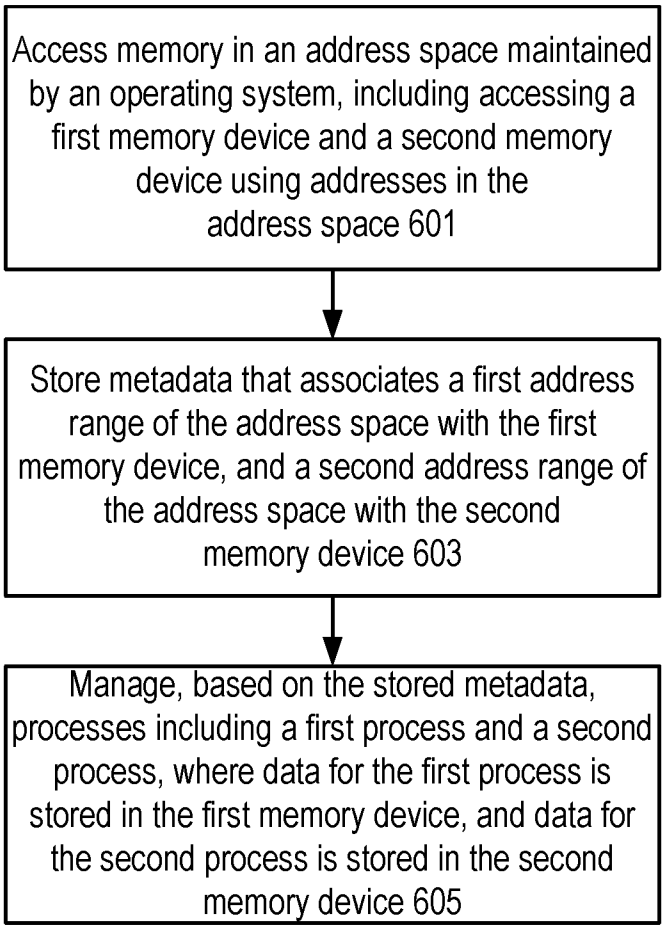
Access memory in an address space maintained by an operating system, including accessing a first memory device and a second memory device using addresses in the address space 601
Store metadata that associates a first address range of the address space with the first memory device, and a second address range of the address space with the second memory device 603
Manage, based on the stored metadata, processes including a first process and a second process, where data for the first process is stored in the first memory device, and data for the second process is stored in the second memory device 605

FIG. 6

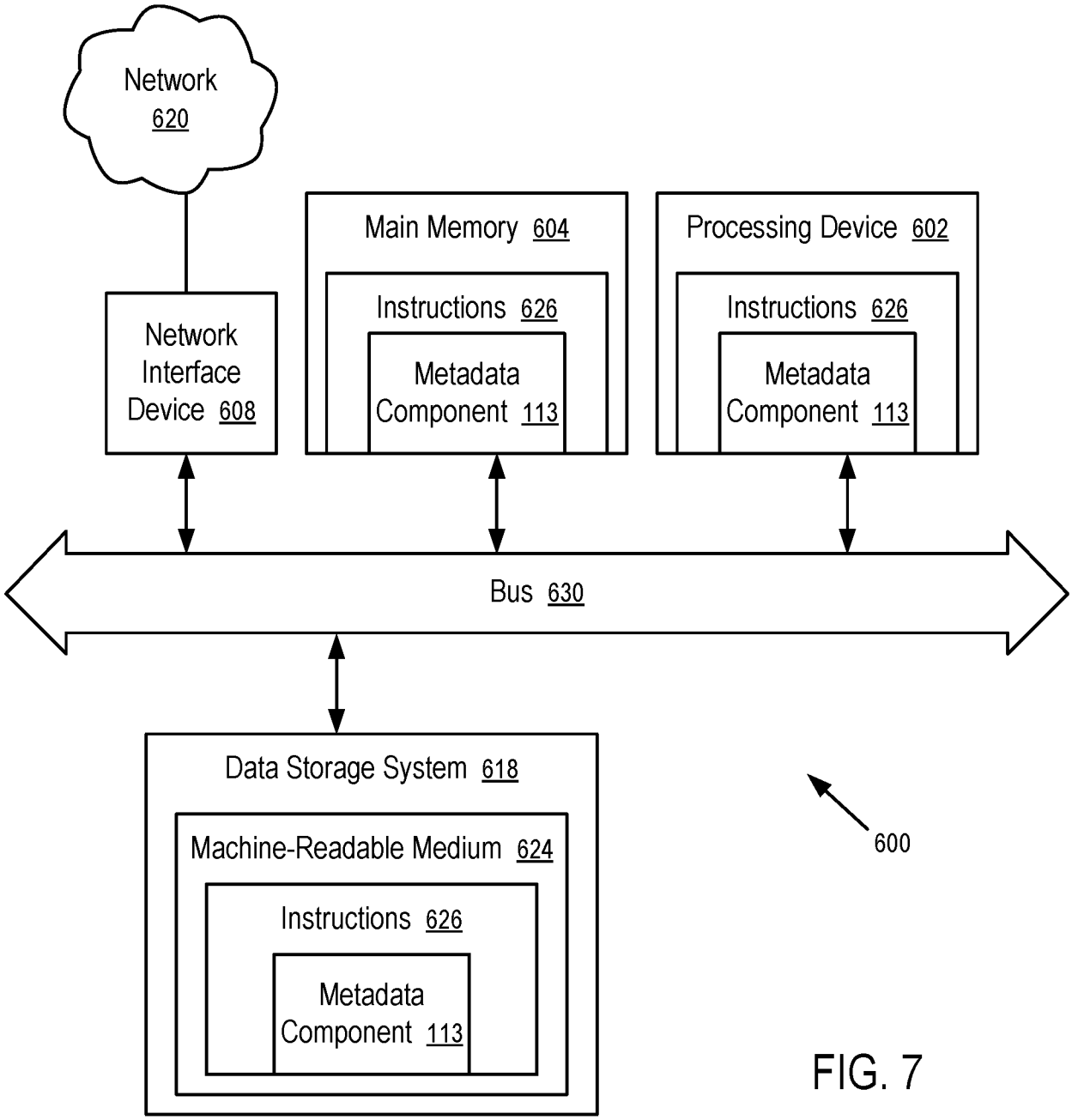
Network
620
Main Memory 604
Instructions 626
Metadata
Component 113
Processing Device 602
Instructions 626
Metadata
Component 113
Network
Interface
Device 608
Bus 630
Data Storage System 618
Machine-Readable Medium 624
Instructions 626
Metadata
Component 113
600

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2020/050714

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/02(2006.01)i, G06F 12/1009(2016.01)i, G06F 12/1027(2016.01)i, G11C 8/18(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 12/02; G06F 12/00; G06F 13/00; G06F 3/06; G06F 12/1009; G06F 12/1027; G11C 8/18

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: memory, latency, address space, address range, metadata, application

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 7552282 B1 (MICHAEL BERMINGHAM et al.) 23 June 2009 claims 1-19 and figure 11	1-20
A	US 9448743 B2 (SANDISK ENTERPRISE IP LLC) 20 September 2016 claims 1-10 and figure 1	1-20
A	US 2014-0215129 A1 (RADIAN MEMORY SYSTEMS, LLC) 31 July 2014 claims 1-3 and figure 2	1-20
A	US 8452929 B2 (JON C. R. BENNETT) 28 May 2013 claims 1, 35, 36, 44 and figure 3A	1-20
A	KR 10-1377147 B1 (SANDISK CORPORATION) 24 March 2014 claims 1-4 and figure 8C	1-20

☐ 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
"D" document cited by the applicant in the international application
"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
16 December 2020 (16.12.2020)	**17 December 2020 (17.12.2020)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
YANG JEONG ROK
Telephone No. +82-42-481-5709

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2020/050714

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 7552282 B1	23/06/2009	None	
US 9448743 B2	20/09/2016	CN 101965559 A	02/02/2011
		CN 101965559 B	09/10/2013
		CN 103150275 A	12/06/2013
		CN 103150275 B	16/12/2015
		EP 2225642 A1	08/09/2010
		EP 2225642 B1	16/11/2016
		JP 2011-508349 A	10/03/2011
		JP 5272019 B2	28/08/2013
		US 2012-0072654 A1	22/03/2012
		US 2013-0262753 A1	03/10/2013
		US 2013-0339581 A1	19/12/2013
		US 2013-0339582 A1	19/12/2013
		US 2014-0101378 A1	10/04/2014
		US 2014-0108715 A1	17/04/2014
		US 2014-0237168 A1	21/08/2014
		US 2016-0034227 A1	04/02/2016
		US 7934052 B2	26/04/2011
		US 7978516 B2	12/07/2011
		US 8245101 B2	14/08/2012
		US 8386700 B2	26/02/2013
		US 8533384 B2	10/09/2013
		US 8621137 B2	31/12/2013
		US 8621138 B2	31/12/2013
		US 8738841 B2	27/05/2014
		US 8751755 B2	10/06/2014
		US 8762620 B2	24/06/2014
		US 8775717 B2	08/07/2014
		US 8959282 B2	17/02/2015
		US 8959283 B2	17/02/2015
		US 9152556 B2	06/10/2015
		US 9158677 B2	13/10/2015
		US 9239783 B2	19/01/2016
		US 9483210 B2	01/11/2016
		WO 2009-086357 A1	09/07/2009
		WO 2009-086359 A1	09/07/2009
		WO 2009-086365 A1	09/07/2009
		WO 2009-086371 A1	09/07/2009
		WO 2009-086376 A1	09/07/2009
		WO 2009-086404 A1	09/07/2009
		WO 2009-086412 A1	09/07/2009
		WO 2009-086419 A1	09/07/2009
		WO 2009-086421 A1	09/07/2009
		WO 2009-086424 A1	09/07/2009
US 2014-0215129 A1	31/07/2014	US 10445229 B1	15/10/2019
		US 10838853 B1	17/11/2020
		US 2014-0365719 A1	11/12/2014

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		US 9229854 B1	05/01/2016
		US 9400749 B1	26/07/2016
		US 9519578 B1	13/12/2016
		US 9652376 B2	16/05/2017
		US 9710377 B1	18/07/2017
		US 9727454 B2	08/08/2017
US 8452929 B2	28/05/2013	CN 101180596 B	19/05/2010
		CN 101681305 A	24/03/2010
		CN 101681305 B	30/10/2013
		CN 101727429 A	09/06/2010
		CN 101727429 B	14/11/2012
		CN 101836193 A	15/09/2010
		CN 101836193 B	03/10/2012
		CN 101872333 A	27/10/2010
		CN 101884033 A	10/11/2010
		CN 101884033 B	28/09/2016
		CN 101903866 A	01/12/2010
		CN 101903866 B	31/07/2013
		CN 102346694 A	08/02/2012
		CN 102346694 B	11/02/2015
		CN 102667738 A	12/09/2012
		CN 102667738 B	17/07/2018
		CN 103116565 A	22/05/2013
		EP 1872192 A2	02/01/2008
		EP 1872192 B1	12/09/2012
		EP 2132636 A2	16/12/2009
		EP 2132636 B1	29/10/2014
		EP 2193446 A2	09/06/2010
		EP 2193446 B1	21/10/2015
		EP 2201463 A2	30/06/2010
		EP 2232374 A2	29/09/2010
		EP 2232374 B1	07/05/2014
		EP 2378391 A1	19/10/2011
		EP 2378432 A1	19/10/2011
		EP 2378433 A1	19/10/2011
		EP 2383656 A1	02/11/2011
		EP 2383656 B1	19/06/2013
		EP 2383657 A1	02/11/2011
		EP 2383660 A1	02/11/2011
		EP 2383660 B1	26/06/2013
		EP 2383661 A1	02/11/2011
		EP 2467783 A2	27/06/2012
		EP 2467783 B1	27/05/2020
		EP 2793132 A2	22/10/2014
		EP 2793132 A3	03/12/2014
		EP 2793132 B1	12/10/2016
		EP 2860635 A1	15/04/2015
		EP 2860635 B1	02/11/2016
		EP 2996027 A1	16/03/2016

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2020/050714

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		EP 3696676 A1	19/08/2020
		JP 2011-502293 A	20/01/2011
		JP 2012-185851 A	27/09/2012
		JP 2013-168164 A	29/08/2013
		JP 2013-191217 A	26/09/2013
		JP 5258997 B2	07/08/2013
		JP 5654630 B2	14/01/2015
		KR 10-1132321 B1	05/04/2012
		KR 10-1271245 B1	07/06/2013
		KR 10-1307953 B1	12/09/2013
		KR 10-1331569 B1	21/11/2013
		KR 10-1375763 B1	19/03/2014
		KR 10-1411566 B1	25/06/2014
		KR 10-1427867 B1	11/08/2014
		KR 10-1448192 B1	07/10/2014
		KR 10-1502519 B1	13/03/2015
		KR 10-1520370 B1	19/05/2015
		KR 10-1573370 B1	01/12/2015
		KR 10-1710546 B1	27/02/2017
		KR 10-2012-0086695 A	03/08/2012
		KR 10-2012-0136412 A	18/12/2012
		KR 10-2013-0041314 A	24/04/2013
		KR 10-2013-0072270 A	01/07/2013
		KR 10-2013-0112953 A	14/10/2013
		KR 10-2014-0106752 A	03/09/2014
		KR 10-2015-0002900 A	07/01/2015
		KR 10-2016-0031048 A	21/03/2016
		US 10157016 B2	18/12/2018
		US 10176861 B2	08/01/2019
		US 10372366 B2	06/08/2019
		US 10417159 B2	17/09/2019
		US 2011-0126045 A1	26/05/2011
		US 2012-0221922 A1	30/08/2012
		US 2013-0042119 A1	14/02/2013
		US 2014-0310483 A1	16/10/2014
		US 2015-0178157 A1	25/06/2015
		US 2015-0242271 A1	27/08/2015
		US 2016-0085453 A1	24/03/2016
		US 2016-0239228 A1	18/08/2016
		US 2016-0246509 A1	25/08/2016
		US 2017-0199670 A1	13/07/2017
		US 2018-0341408 A1	29/11/2018
		US 2019-0303008 A1	03/10/2019
		US 7667914 B2	23/02/2010
		US 8112655 B2	07/02/2012
		US 8200887 B2	12/06/2012
		US 8726064 B2	13/05/2014
		US 9081713 B1	14/07/2015
		US 9189334 B2	17/11/2015
		US 9286198 B2	15/03/2016

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		US 9311182 B2	12/04/2016
		US 9384818 B2	05/07/2016
		US 9582449 B2	28/02/2017
		US 9632870 B2	25/04/2017
		US 9727263 B2	08/08/2017
		WO 2011-044515 A2	14/04/2011
		WO 2011-044515 A3	27/10/2011
KR 10-1377147 B1	24/03/2014	CN 101233479 A	30/07/2008
		CN 101233479 B	05/09/2012
		CN 101233480 A	30/07/2008
		CN 101233480 B	29/08/2012
		CN 101233498 A	30/07/2008
		CN 101233498 B	12/03/2014
		CN 101233499 A	30/07/2008
		CN 101258473 A	03/09/2008
		CN 101258473 B	30/05/2012
		CN 101263462 A	10/09/2008
		CN 101263462 B	16/05/2012
		CN 101278267 A	01/10/2008
		CN 101278267 B	22/08/2012
		CN 101288045 A	15/10/2008
		CN 101288045 B	29/08/2012
		EP 1910928 A2	16/04/2008
		EP 1913463 A2	23/04/2008
		EP 1913480 A1	23/04/2008
		EP 1913480 B1	29/12/2010
		EP 1917577 A2	07/05/2008
		EP 1920318 A2	14/05/2008
		EP 1920335 A1	14/05/2008
		EP 1920335 B1	11/05/2011
		EP 1920336 A2	14/05/2008
		EP 1920337 A2	14/05/2008
		JP 2009-518698 A	07/05/2009
		JP 4533956 B2	01/09/2010
		JP 4537481 B2	01/09/2010
		JP 4537482 B2	01/09/2010
		JP 4547028 B2	22/09/2010
		JP 4977703 B2	18/07/2012
		JP 5068754 B2	07/11/2012
		KR 10-1055324 B1	08/08/2011
		KR 10-1089150 B1	02/12/2011
		KR 10-1272642 B1	10/06/2013
		KR 10-1329068 B1	14/11/2013
		KR 10-1378031 B1	27/03/2014
		KR 10-2008-0033464 A	16/04/2008
		KR 10-2008-0038364 A	06/05/2008
		KR 10-2008-0038368 A	06/05/2008
		KR 10-2008-0042844 A	15/05/2008
		KR 10-2008-0042850 A	15/05/2008

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2020/050714

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
		KR 10-2008-0042851 A	15/05/2008
		KR 10-2008-0044254 A	20/05/2008
		TW 200728977 A	01/08/2007
		TW 200731065 A	16/08/2007
		TW 200731067 A	16/08/2007
		TW 200739342 A	16/10/2007
		TW 200741526 A	01/11/2007
		TW 200745929 A	16/12/2007
		TW 200745930 A	16/12/2007
		TW 200805134 A	16/01/2008
		US 2007-0033378 A1	08/02/2007
		US 2007-0186032 A1	09/08/2007
		US 7669003 B2	23/02/2010
		US 7949845 B2	24/05/2011
		US 7984084 B2	19/07/2011
		US 8055832 B2	08/11/2011
		US 8291151 B2	16/10/2012
		WO 2007-019155 A1	15/02/2007
		WO 2007-019174 A2	15/02/2007
		WO 2007-019174 A3	19/07/2007
		WO 2007-019175 A2	15/02/2007
		WO 2007-019175 A3	07/09/2007
		WO 2007-019197 A2	15/02/2007
		WO 2007-019197 A3	07/09/2007
		WO 2007-019198 A2	15/02/2007
		WO 2007-019198 A3	21/06/2007
		WO 2007-019217 A1	15/02/2007
		WO 2007-019220 A2	15/02/2007
		WO 2007-019220 A3	07/06/2007
		WO 2007-019258 A2	15/02/2007
		WO 2007-019258 A3	19/07/2007