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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 August 2016 (18.08.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/130487 A1

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

(21) International Application Number:
PCT/US2016/017024

(22) International Filing Date:
8 February 2016 (08.02.2016)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
201510069338.7 10 February 2015 (10.02.2015) CN
15/017,070 5 February 2016 (05.02.2016) US

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

(72) Inventor: **ZHAO, Yongke**; Alibaba Group Legal Department, 5/F, Building 3, No. 969 West Wen YI Road, Yu Hang District, Hangzhou, 311121 (CN).

(74) Agent: **SOCHOR, Michael, D.**; Murabito Hao & Barnes, LLP, 2 N. Market St., 3rd Floor, San Jose, CA 95113 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Published:

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

(54) Title: RESOURCE MANAGEMENT

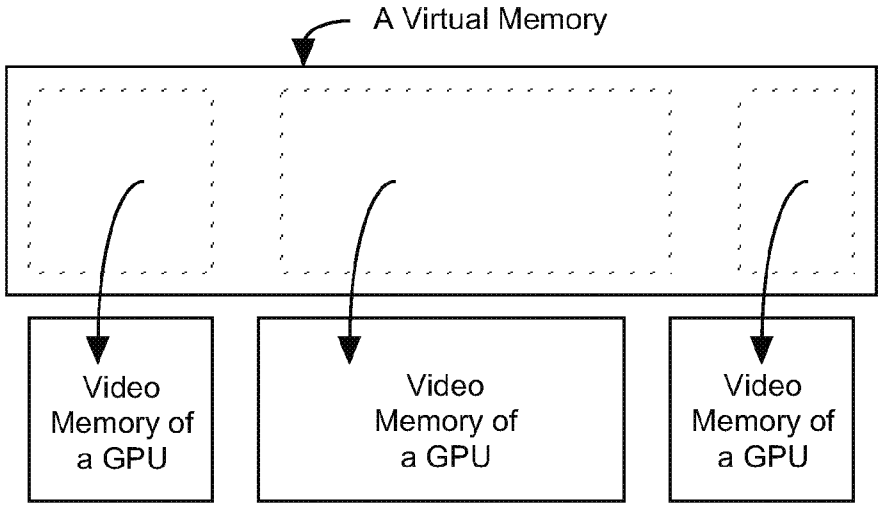

FIG. 5

(57) Abstract: Embodiments of the present invention provide resource managing methods and systems. The method comprises: receiving a request to allocate resources sent from host code of an application program located on a first device; in accordance with the allocation request and a maintained mapping logic mapping available hardware resources of at least one graphics processing unit (GPU) of the first device to a unified virtual GPU resource, allocating required resources for a device code of the application program from the available hardware resources of at least one GPU of the first device; and forwarding information of the allocated resource back to the host code. The present invention can efficiently utilize GPU resources and reduce implementation costs.

WO 2016/130487 A1

RESOURCE MANAGEMENT

[001] This application claims priority to Chinese Patent Application No. 201510069338.7, filed on February10, 2015, which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[002] Embodiments according to the present disclosure relate generally to the field of software technology and, more particularly, to a method and apparatus for resource management.

BACKGROUND

[003] With a constant increase in data volume and computing scale, more and more application programs are choosing to use graphics processing units (GPU) for computational resources. During the process of using a GPU, an operating system is responsible for obtaining low-level GPU hardware details and providing those low-level GPU hardware details to an application program. The application program determines the required GPU resources and requests the required GPU resources from the GPU according to the low-level GPU hardware details.

[004] Some video card manufacturers, such as NVIDIA and AMD, continually release GPUs with new structures for the purpose of increasing video card efficiency. However, the capacities of the newly released video cards are usually so small that a single video card is incapable of meeting the increasing needs of the computational requirements. When there is a need to further improve computational capabilities, coordinative computing by multiple GPUs is usually selected as the compute mode. One of the most common modes is to use one computer with multiple GPUs to coordinate computing large amounts of data.

[005] Since the GPU-based application programs on the market are designed to run on hardware with a single GPU card, programmers need to redesign application programs in order for them run on the hardware structures incorporated within multiple GPUs. These new application programs need to be capable of using hardware resources based on the details of the low level hardware, such as, the video memory capacities, the number of computing units, and the number of GPUs to accommodate the hardware structures of multiple GPUs. Moreover, the application programs need to be redesigned with respect to the different structures of a computer having

multiple GPUs. Even though GPU resources can be used more efficiently when there is a specific application program for respective hardware structures, there is a significant cost to have a specific design maintained for each of the hardware structures.

SUMMARY OF INVENTION

[006] The present disclosure overcomes the deficiencies described above by providing a technique to avoid redesigning the application programs and to reduce the cost associated with application development. Embodiments of the present disclosure include a method and apparatus for resource management that uses GPU resources more efficiently and reduces costs.

[007] An embodiment comprises a method including receiving a request to allocate resources sent from host code of an application program located on a first device. The method also includes allocating resources for device code of the application program from available hardware resources of at least one GPU of the first device in accordance with 1) the allocation request and 2) a maintained mapping logic mapping the available hardware resources of at least one GPU of the first device to a unified virtual GPU resource. The method further includes forwarding the information of the allocated resource back to the host code to enable the host code to control the device code to use the allocated resource to run on at least one GPU.

[008] Another embodiment comprises a device for managing resources that includes a receiving module configured to receive a request to allocate resources sent from host code of an application program on a first device. The device also includes an allocating module configured to allocate required resources for device code of the application program from available hardware resources of at least one GPU of the first device in accordance with the allocation request and further configured to maintain mapping logic mapping the available hardware resources of at least one GPU of the first device to a unified virtual GPU resource. The device further includes a sending module configured to forward information of the allocated resource back to the host in order to enable the host code to control the device code, and allowing the device code to run on at least one GPU based on the allocated resource.

[009] Embodiments of the present disclosure maintain mapping logic between a GPU and an application program that maps the available hardware resources of at least one GPU of the first device to a unified virtual GPU resource. The available hardware resources provided by the at least one GPU of the first device can be presented to the application program in the form of a unified

virtual GPU resource by use of mapping logic. Specifically, when a request to allocate resources sent from host code is received, according to 1) the allocation request and 2) the maintained mapping logic mapping the available hardware resources of at least one GPU of the first device to a unified virtual GPU resource, resources are selected from the available hardware resources of at least one GPU to be allocated for device code of the application program and then provided to the host code. When receiving the resources, the host code can control the device code and have the device code run on the GPU based on the required resources. The method of the present disclosure implements a separation between the application program and the details of the low level GPU hardware, in addition to maintaining effective use of GPU resources. As the application programs do not need to take the details of the low level hardware into account, there is advantageously no need to redesign the application programs. Compared with the conventional techniques, the techniques of the embodiments of the present disclosure reduce costs, e.g., associated with redesigning an application.

BRIEF DESCRIPTION OF THE DRAWINGS

[010] Embodiments of the present disclosure will be better understood from a reading of the following detailed description, taken in conjunction with the accompanying figures, in which like reference characters designate like elements and in which:

[011] FIG. 1 is an exemplary flow diagram of a GPU-based computing application program in accordance with an embodiment.

[012] FIG. 2 is a block diagram of a first multi-level structure, from the low level hardware to an upper level application program in accordance with an embodiment.

[013] FIG. 3 is a block diagram of a second multi-level structure, from the low level hardware to an upper level application program in accordance with an embodiment.

[014] FIG. 4 is a flowchart of an exemplary method for managing resources in accordance with an embodiment.

[015] FIG. 5 is a diagram of an exemplary mapping between a virtual memory and a video memory resource in accordance with an embodiment.

[016] FIG. 6 is a block diagram of an apparatus for managing resources in accordance with an embodiment.

[017] FIG. 7 is a block diagram of an apparatus for managing resources in accordance with an embodiment.

DETAILED DESCRIPTION

[018] Reference will now be made in detail to the embodiments of the present invention. While the invention will be described in conjunction with these embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications, and equivalents which may be included within the spirit and scope of the invention as defined by the appended claims. It will be apparent from the foregoing disclosure to those skilled in the art that variations and modifications of such embodiments and methods may be made without departing from the spirit and scope of the invention.

[019] A GPU-based computing application program includes host code and device code. The host code is executed on a CPU (central processing unit) and the device code, which is also referred to as a kernel function, is executed on a GPU. FIG. 1 illustrates an exemplary flow diagram of a GPU-based computing application program that includes:

[020] (1) Copying, under the control of a CPU, data for processing from a computer memory to a GPU video memory;

[021] (2) The CPU sending a device code to the GPU;

[022] (3) Multiple processors in the GPU executing the device code using the data in the video memory. This can include the GPU reading and writing video memory data repeatedly during the computing process and storing computing results of the device code in the video memory; and

[023] (4) Copying the computing results in the video memory to the computer memory under the control of the CPU.

[024] The current GPU-based application programs are designed for single GPU-based systems. The process described above is suitable for the scenarios in which a single GPU-based hardware system is utilized. Programmers need to redesign application programs to accommodate the hardware structure of a computer with multiple GPU cards and to use hardware resources based on the details of the low level hardware of the computer with multiple GPU cards, such as video memory capacities, the number of computing units and the number of GPU cards. However, generally the cost to redesign the application program is high.

[025] Addressing the issues explained above, the present disclosure includes a method for resource management. By using this method, a GPU-based computing application program can

be executed on devices containing GPU cards and CPUs without taking the details of the low level hardware into account or redesigning application programs. The method, in accordance with the present invention, uses GPU resources effectively and reduces the cost.

[026] Before presenting a method of the present disclosure, an operating environment will be described herein. For ease of description, a device containing both GPU cards and one or more CPUs will be referred to as the first device. The first device can be any device including a combination of a computer with multiple GPU cards, which means that there is at least one CPU card associated with the first device accompanying the GPUs.

[027] In the field of computer technology, there are multiple levels or layers from the low level hardware to an upper level application program. FIG. 2 illustrates one of the commonly used multi-level structures, including at least one GPU, drivers, an operating system, a runtime library, and application programs.

[028] Drivers are modules designed specifically to communicate with hardware devices that are usually provided by hardware manufactures (e.g., NVIDIA and AMD GPU drivers are provided by NVIDIA and AMD respectively). Drivers provide software interfaces for hardware devices to support the operating system. An operating system possesses the capability to manage resources and allocate resources for multiple users and processes to prevent competition between the users or the processes. The upper level above the operating system is a runtime library provided for the application programs for the users to communicate with the low level hardware through appointed ports to prevent unauthorized hardware access by users.

[029] In order to support the operation method in the present disclosure, a virtual middleware is introduced to the multi-level structures presented in FIG. 2. The virtual middleware is used to map available hardware resources of at least one GPU to a unified virtual GPU resource, which can also be interpreted as mapping the hardware resources to resources on a virtual GPU for further processing. The virtual middleware is also used to maintain the mapping logic which provides mapping of the available hardware resources of at least one GPU to a unified virtual GPU resource to avoid the process of redesigning application programs. This enables application programs to be used on a device that combines a computer and multiple GPU cards in order to reduce costs associated with application programs.

[030] It should be noted that the virtual middleware can be placed between any two levels to implement embodiments of the present invention. FIG. 3 illustrates that the virtual middleware is placed between the drivers and the operating system and has a high efficiency when

incorporated in such a structure. In an embodiment with such a structure, the virtual middleware manages the low level hardware directly by using the drivers to map at least one GPU to a virtual GPU. The middleware can also be placed between the operating system and the runtime library or between the runtime library and the application programs.

[031] It should be noted that the virtual middleware can be embodied as the resource management device in subsequent embodiments.

[032] FIG. 4 is a flowchart of an exemplary method for resource management. In step 401, a request to allocate resources is sent from a host code of an application on a first device. In step 402, resources are allocated for a device code of the application program from the available hardware resources of at least one GPU on the first device in accordance with the allocation request and mapping logic is maintained that maps available hardware resources of at least one GPU on the first device to a unified virtual GPU resource. In step 403, the allocated resources are forwarded back to the host code.

[033] According to an embodiment, a resource management device implements the method for managing resources. The resource management device is located on the first device between the GPUs and application programs and maintains the mapping logic (mapping the available hardware resources of at least one GPU of the first device to a unified virtual GPU resource). The resource management device is in charge of managing and dispatching GPU resources and implementing communication between the application programs and the low level GPU hardware. The resource management device can be embodied as the virtual middleware in FIG. 3.

[034] According to an embodiment, a kernel layer or a driver layer of the operating system can execute the method above, meaning that the virtual middleware can also be embedded between the operating system and the driver layer in order to implement resource management. Specifically, the host code of the application program runs on the CPU of the first device. The host code sends requests to allocate resources to the resource management device or, alternatively, the CPU that runs the host code sends requests to allocate resources to the resource management device. The resource management device receives the requests to allocate resources from the host code and allocates resources for a device code of the application program from available hardware resources of at least one GPU of the first device according to the allocation requests and maintains mapping logic (mapping the available hardware resources of at least one GPU of the first device to the unified virtual GPU resource). The allocated resources are then provided to the host code. In this way, the

host code can control the device code and have the device code run on the GPU based on the allocated resources.

[035] It should be noted that the allocation requests can include information such as types and quantities of required resources. The resource management device allocates resources, such as the types and amounts requested for the device code, according to the information of the allocation requests.

[036] The main function of the mapping logic mapping the available hardware resources of at least one GPU of the first device to the unified virtual GPU resource is to map the available hardware resources of at least one GPU of the first device to a virtual GPU resource. In this way, from an exterior perspective, what is observed is one virtual GPU resource instead of the available hardware resources of a GPU. Correspondingly, according to the mapping logic, the process of allocating of resources for the device code of the application program from the available hardware resources of at least one GPU of the first device allocates resources for the device code according to the unified virtual GPU resource.

[037] An embodiment includes a method that maintains the mapping logic between the GPU and the application program that maps the available hardware resources of at least one GPU of the first device to the unified virtual GPU resource, so that the available hardware resources of at least one GPU of the first device are presented to the application program in the form of a unified virtual GPU resource. The method uses the mapping logic to allocate resources for the device code running on the GPU of the application program, which implements a separation between the application program and the details of the low level GPU hardware. The application program no longer needs to take the details of the low level GPU hardware into account or determine and allocate GPU resources. Hence, the process and logic embodying the execution of the device code can be designed to be used by a single GPU card instead of being designed to additionally meet the requirements embodying interactions between multiple GPU cards in different scenarios. This avoids the need to redesign the application programs and, more specifically, avoids the need to redesign the logic for the execution of the device code in the application programs. By using this method, the GPU resources will be used more effectively and at a lower cost. Moreover, the method in accordance with the present invention extends the scale of using small capacity GPUs, releases the manufactures from developing GPUs with larger capacities, and also reduces application development costs. The method allows freedom from the restrictions of the low level hardware details and has portability that can be transferred to different structures.

[038] In an embodiment, in order to manage and dispatch the hardware resources provided by the GPU, before allocating resources for the application program from the available hardware resources of the GPU, the resource management device obtains, in advance, details of the hardware resources, also referred to as resource details. The resource details are provided by each of the GPUs of a set of GPUs and the resource management device generates the mapping logic mapping available hardware resources of at least one GPU to a unified virtual GPU resource mapped in accordance with the resource details of each GPU. The resource details are used to describe the available hardware resources of the GPU. For example, the resource details can include types, quantities, working conditions of the hardware resources, etc. In this embodiment, the available hardware resources of each GPU can include, but are not limited to, video memory resources and computational resources. The details of the video memory resource can include a type identifier used to identify a video memory resource, a capacity of video memory, availability of video memory, etc. The details of a computational resource can include a type identifier used to identify a computational resource, a quantity of the resources, working conditions of the resources, etc.

[039] The resource management device sends a request to obtain resource details to each of the GPUs of the set of GPUs and receive resource details forwarded back from each GPU in response to the request. Optionally, before sending the requests to obtain resource details to the GPUs, the resource management device monitors the initialization of the first device to detect the GPUs. Another way to obtain resource details is to let the resource management device receive registration requests including resource details sent from each of the GPUs of the set of GPUs and extract resource details from the registration requests. For example, each GPU can proactively register with the resource management device after the initialization of the first device so that the resource management device receives resource details and manages the GPUs according to the resource details.

[040] In another embodiment, the available hardware resources of at least one GPU include video memory resources and computational resources. Based on this, the resource management device can access, in accordance with the allocation requests, the maintained mapping logic mapping the available hardware resources of at least one GPU of the first device to the unified virtual GPU resource, and the resource management device can allocate requested computational resources from the computational resources and allocated requested video memory resources from the video memory resources for the device code, according to the mapping logic. For ease of

description, the computational resource allocated to the device code will be referred to as the first computational resource and the video memory resource allocated to the device code as the first video memory resource.

[041] Since the video memory resources are mainly used to provide application programs with access to data, the resource management device can provide for a virtual memory having an identical amount of resources as the total amount of the available video memory resources of at least one GPU from the system memory of the first device according to the mapping logic (mapping the available hardware resources of at least one GPU of the first device to the unified virtual GPU resource).The resource management device can map the virtual memory with the available video memory resource of at least one GPU to generate a mapping relation between the two. Based on the foregoing, the process of the resource management device allocating the first video memory resources for the device code from the available hardware resources of at least one GPU according to the mapping logic can include the following: allocating a first storage resource for the device code from the virtual memory and identifying a first video memory resource corresponding to the first storage resource according to the mapping logic to identify the first video memory resource allocated to the device of the application program. Optionally, the mapping relation between the virtual memory and the available hardware resources of at least one GPU can be embodied in a specific way by mapping the addresses of the virtual memory and the video memory resource.

[042] FIG. 5 illustrates a mapping relation between the virtual memory and the available hardware resources of at least one GPU.

[043] In an embodiment, the resource management device can flexibly allocate resources for application programs according to the usage of computational resources. For example, when the computational resources are constrained, the resource management device can prioritize the allocation of computational resources for application programs that have high priority and high real-time requirements to let those application programs receive computational resources from the GPU first. The resource management device can also postpone the allocation of computational resources for application programs that have intense input/output (I/O) performance and low efficiency. Specifically, the resource management device can determine the sequencing to allocate computational resources for the device code of the application programs according to the priorities of the requests to allocate resources. The resource management device can also allocate the first computational resource for the device code of the application programs from the available

computational resources of at least one GPU according to the mapping logic mapping the available hardware resources of at least one GPU to the unified virtual GPU resource.

[044] The priorities of the requests to allocate resources can be determined according to the resource requirements and the scenarios of the applications such as the application's requirements for intensity of I/O performance and real-time performance.

[045] In the foregoing embodiment, the resource management device can apply for virtual memory from the system memory, which acts as a virtual GPU having enough storage resources from the perspective of the application programs. The application programs no longer need to take into consideration the details of the low level hardware, including the capacities of the video memories or the number of the GPU cards. By using the mapping logic, the resource management device automatically manages requests between the application programs and the bottom GPU hardware resources. Thus there is no need to redesign application programs, which simplifies the developing, dispatching, and maintaining work and thereby reduces costs.

[046] Furthermore, after the resource management device allocates resources for the device code of the application programs, the allocated resources can be sent to the host code of the application program to enable the host code to control the device code and have the device code run on the GPU. Specifically, the resource management device can send an identifier of the first computational resource and addresses of the first storage resource to the host code. The host code can then send the device code to the first computational resource to have the first computational resource run the device code and, in the meantime, save data produced from running the device code to the first storage resource. The resource management device can save the data produced from running the device code to the identified first video memory resource that corresponds to the first storage resource. It should be noted that, for the application program, the first storage resource in the foregoing is actually virtual memory and the address of the first storage resource is the address of the virtual memory.

[047] After obtaining the allocated first storage resource, the host code of the application program can access the first storage resource, such as requesting to write data to the first storage resource and read data from the first storage resource. Based on this fact, the resource management device can receive read requests sent from the host code to read data from the first storage device and forward the read data back to the host code. Alternatively, the resource management device can receive write requests sent from the host code to write data to the first storage device and synchronize the first video memory resource with the written data in accordance

with the identification of the first video memory resource, thereby maintaining its correspondence with the first storage resource. It should be noted that the read requests in the foregoing usually carry the address information of the first storage resource to indicate the location to read data from. The write requests also typically carry the address information of the first storage resource to indicate the location to write data to as well as information regarding what is to be written, such as the data to be written or a storage path for the data to be written.

[048] During the process of the application program using storage resources, the resource management device may synchronize data between the virtual memory and the available video memory resources. The synchronization of a single segment of data is difficult to achieve and expensive; therefore, a full synchronization is adopted to synchronize all of the data in the video memory resource of the whole GPU card with the data in the virtual memory corresponding to that GPU. This full synchronization causes a significant delay, goes against the computing potential of the GPU, and significantly reduces the efficiency and performance of the resource management device.

[049] Addressing the issues mentioned above, the resource management device uses paging to manage resources. When generating the mapping relation between the virtual memory and the available video memory resources of at least one GPU, the resource management device pages the virtual memory to obtain a memory page, pages the video memory resource to obtain a video memory page, and maps the memory page with the video memory page to obtain a mapping relation between the two.

[050] Based on the description in the foregoing, the process of the resource management device synchronously writing the data of the first storage resource to the first memory resource can be specifically interpreted as determining the memory page where the write data is to be located, which will be referred to as the first memory page for ease of description, and synchronizing the data on that first memory page to the first video memory page corresponding with the first memory page.

[051] The resource management device thereby only needs to synchronize the data in one page by using the page management instead of synchronizing all of the data, which is beneficial as it reduces the amount of data interactions between the virtual memory and the video memory resource, thereby further decreasing the delay of memory access and enabling the full computing potential of the GPU. Thus, this method reduces the processing burden of the resource management device and also improves its efficiency.

[052] It should be noted that, for ease of description, the embodiments in the foregoing are presented in a manner of combinations of a series of actions and procedures; however, those skilled in the art should acknowledge that the present disclosure should not be limited by the orders of the actions or procedures. According to the present disclosure, some of the procedures can be embodied in different sequences or simultaneously. Moreover, those skilled in the art should know that the embodiments of the present disclosure are preferred embodiments; however, related actions and procedures are not required to embody the present disclosure. For the parts not explained in detail in certain embodiments, related descriptions can be cross-referenced.

[053] FIG. 6 illustrates a block diagram of a resource management device according to an embodiment, including receiving module 61, allocating module 62, and sending module 63.Receiving module 61 is configured to receive the request to allocate resources sent from the host code of the application program on the first device. Allocating module 62 is configured to allocate resources, according to the request received by module 61, for the device code of the application program from the available hardware resources of at least one GPU of the first device, in accordance with the maintained mapping logic (mapping the available hardware resources of at least one GPU of the first device to the unified virtual GPU resource).Sending module 63 is configured to forward the information from the resources allocated by allocating module 62 back to the host code. The host code can then control the device code to run on at least one GPU based on the allocated resources.

[054] FIG. 7 illustrates a block diagram of a resource management device of an embodiment, including information obtaining module 64 and maintaining module 69. Information obtaining module 64 is configured to send requests to obtain resource details to each of the GPUs of a set of GPUs and receive resource details forwarded back from each of the GPUs based on the requests. Information obtaining module 64 can also be configured to receive registration requests sent from each of the GPUs of the set of GPUs. The registration requests contain resource details, which are used to describe the available hardware resources that each GPU can provide. Maintaining module 69 is configured to generate mapping logic mapping the available hardware resources of at least one GPU to the unified virtual GPU resource according to the resource details of each of the GPUs obtained by information obtaining module 64.

[055] In an embodiment, the available hardware resources of at least one GPU include video memory resources and computational resources. Allocating module 62 is specifically configured to read the mapping logic mapping the available hardware resources of at least one GPU to the unified virtual GPU resource according to the allocation requests. Based on that mapping

logic, allocating module 62 allocates the first computational resource for the device code from the computational resources and the first video memory resource for the device code from the video memory resources.

[056] In an embodiment, as illustrated in FIG. 7, the resource management device also includes applying module 65 and mapping module 66. Applying module 65 is configured to manage a virtual memory that possesses a total amount of resources identical to the available video memory resources of at least one GPU from the system memory of the first device according to the mapping logic mapping the available hardware resources of at least one GPU to the unified virtual GPU resource. Mapping module 66 is configured to generate the mapping logic between the virtual memory is managed by the applying module 65 and the available video memory resources of at least one GPU.

[057] In accordance with the foregoing, allocating module 62 is specifically configured to allocate the first computational resource for the device code from the computational resources and the first storage resource for the device code from the virtual memory; it is also used to mark the first video memory resource corresponding to the first storage resource according to the mapping relation generated by mapping module 66 to identify the first video memory resource allocated to the device code.

[058] In an embodiment, as illustrated in FIG. 7, the resource management device also includes read-write module 67. Read-write module 67 is configured to receive a read request sent from the host code, read data from the first storage resource according to the request, and forward the data back to the host code. Read-write module 67 is also configured to receive a write request sent from the host code, write data to the first storage resource according to the request, and synchronize the written data to the first video memory on the first video memory resource.

[059] In an embodiment, mapping module 66 is specifically configured to conduct a paging process to the virtual memory to obtain a memory page, conduct a paging process to the video memory resource to obtain a video memory page, and generate the mapping relation between the memory page and the video memory page.

[060] Read-write module 67, based on the foregoing, can be specifically configured to receive a write request sent from the host code to write data to the first storage resource, determine the first memory page where the data is written to, and synchronize the data on the first memory page to the video page corresponding to the first memory.

[061] In an embodiment, allocating module 62 is configured to determine the sequencing to allocate computational resources for the device code according to the priorities of allocation requests. It is also used to allocate the first computational resource for the device code from the computational resources and the first video memory resource for the device code from the video memory resources according to the mapping logic mapping the available hardware resource of at least one GPU to the unified virtual GPU resource.

[062] In another embodiment, sending module 63 is configured to send the identifier information of the first computational resource and address information of the first storage resource to the host code to enable the host code to send the device code to the first computational resource to have the first computational resource run the device code and save the data produced in running the device code to the first storage resource and to the mark first video memory resource as allocated.

[063] The resource management device of the embodiment can be located between the GPU and the application program maintaining the mapping logic (mapping the available hardware resource of at least one GPU to the unified virtual GPU resource) to enable the available hardware resources of at least one GPU to be presented to the application program as the unified virtual GPU resource. The specific process is to allocate resources for the device code running on the GPU of the application program. In this way, there is a separation between the application program and the details of the low level GPU hardware so that the application program does not need to take the low level hardware details into account or determine how to allocate GPU resources. The logic of the running process of the device code can then be designed to be used by a single GPU. There is no need to consider having the interactive functions between multiple GPUs. The application programs advantageously do not need to be redesigned to meet the requirements to adapt to the combination of one computer with multiple GPUs, which thereby allows GPU resources to be effectively used and reduce costs.

[064] Those skilled in the art should appreciate that the specific working process(es) of the systems, devices, and methods in the foregoing can be referred to in the embodiments of the present disclosure, which will not be fully discussed herein.

[065] According to the embodiments of the present disclosure, it should be noted that the systems, devices, and methods can be embodied in other ways. The devices described above are just schematic. For example, the arrangement of the modules is just an arrangement of the function logic; there are other arrangements that will fulfill the requirements. For example, one or multiple modules or units can be combined or loaded to another system while omitting some features or

procedures. Moreover, the coupling and combination discussed can be embodied by indirect couplings or communication connections between some ports, devices, or modules; the indirect couplings or communication connections can be electrical, mechanical, or in other forms.

[066] The modules described as separate parts in the descriptions can be physically separated or connected; the modules presented as parts of the device can be or may not be physical modules, meaning that they can be located at the same location or scattered to multiple network units. The processes and devices in the embodiments of the present disclosure can be selected in part or in whole to fulfill the target function of the present disclosure.

[067] Moreover, the function modules in the embodiments of the present disclosure can be integrated into one processing unit or can be scattered to separate physical units; they can also be implemented by combining two or more function modules. The integrated unit can be embodied in the form of hardware or a combination of hardware and software.

[068] The integrated unit embodied in the form of software can be stored on a computer-readable storage medium containing several commands to enable a computer (including PCs, servers, or network devices) or a processor to execute part of the steps in the embodiments of the present disclosure. The computer-readable mediums include the following: USB drives, portable hard drives, read-only memories, random access memories, magnetic disks or CDs, and all mediums that can store program codes.

[069] It should be noted that, although certain preferred embodiments and methods have been disclosed herein, it will be apparent from the foregoing disclosure to those skilled in the art that variations and modifications of such embodiments and methods may be made without departing from the spirit and scope of the invention. It is intended that the invention shall be limited only to the extent required by the appended claims and the rules and principles of applicable law.

.

WHAT IS CLAIMED IS:

1. A method of resource management, the method comprising:

receiving an allocation request to allocate resources sent from a host code of an application program located on a first device;

allocating, in accordance with the allocation request and a maintained mapping logic mapping available hardware resources of at least one GPU of the first device to a unified virtual GPU resource, resources for a device code of the application program from the available hardware resources of the at least one GPU of the first device; and

responsive to the allocating, forwarding information of allocated resources back to the host code of the application program.

2. The method of Claim 1, further comprising, in accordance with the allocation request and the maintained mapping logic, and prior to the allocation of the resource for the device code of the application program from the available hardware resources of the at least one GPU of the first device, performing:

sending a request to obtain resource details to each one of the GPUs of a set of GPUs;

receiving resource details of each one of the GPUs forwarded back from each one of the GPUs in response to the request to obtain resource details, wherein the resource details describe the available hardware resources of each one of the GPUs;

generating the mapping logic mapping available hardware resources of the at least one GPU to the unified virtual GPU resource in accordance with the resource details of each one of the GPUs; or

receiving a registration request sent from each one of the GPUs of the set of GPUs, wherein the registration request comprises resource details; and

generating a mapping logic mapping available hardware resources of the at least one GPU to the unified virtual GPU resource in accordance with the resource details of

each one of the GPUs, wherein the resource details describes the available hardware resources of each one of the GPUs.

3. The method of Claim 1, wherein the available hardware resources of the at least one GPU comprise video memory resources and computational resources;

wherein further resources are allocated, in accordance with the allocation request and the maintained mapping logic, for the device code of the application program from the available hardware resources of the at least one GPU of the first device, and further comprising:

reading, in accordance with the allocation request, the mapping logic; and

allocating, in accordance with the mapping logic, a first computational resource for the device code from the computational resources and a first video memory resource for the device code from the video memory resources.

4. The method of Claim 3,further comprising, in accordance with the allocation request and the maintained mapping logic, and prior to the allocation of a resource for the device code of the application program from the available hardware resources of the at least one GPU of the first device, performing:

applying, in accordance with mapping logic, from a system memory, for a virtual memory providing a total amount of available memory resources analogous to the amount that of the at least one GPU; and

generating a mapping relation between the virtual memory and the video memory resource provided by the at least one GPU;

wherein the allocating, in accordance with the mapping logic, a first video memory resource for the device code from the video memory resources, further comprises:

allocating a first storage resource for the device code from the virtual memory; and

marking, according to the mapping relation, the first video memory resource corresponding to the first storage resource to identify the first video memory resource allocated to the device code.

5. The method of Claim 4, further comprising:

receiving a read request sent from the host code to read data from the first storage resource and forwarding the data read back to the host code; and

receiving a write request sent from the host code to write data to the first storage resource and synchronizing the marked first video memory resource with the data written.

6. The method of Claim 4, wherein the mapping relation between the virtual memory and the available video memory resource of the at least one GPU is generated through a process comprising:

conducting paging to the virtual memory to obtain a memory page;

conducting paging to the video memory resource to obtain a video memory page;

generating a mapping relation between the memory page and the video memory page; and

wherein the synchronizing the first video memory resource to the data written further comprises:

determining the first memory page where the data is written and synchronizing the data on the first memory page to the video memory page corresponding to that first memory page.

7. The method of Claim 3, wherein the allocating the first computational resource, in accordance with the mapping logic, for the device code from the computational resources further comprises:

determining a sequencing of allocating computational resources for the device code according to priorities of a plurality of allocation requests; and

allocating, in accordance with the sequencing and the mapping logic, the first computational resource for the device code from the computational resource.

8. The method of Claim 3, wherein the forwarding of the information of the allocated resource back to the host to enable the host code to control the device code to have the device code run on the at least one GPU and further comprises:

sending labeling information of the first computational resource and address information of the first memory resource to the host code to enable the host code to send the device code to the first computational resource to have the first computational resource run the device code and save the data produced by running the device code to the first storage resource; and

saving the data produced in running the device code to the labeled first video memory resource.

9. An apparatus for resource management, the apparatus comprising:

a receiving module configured for receiving an allocation request to allocate a resource sent from a host code of an application program on a first device;

an allocating module coupled with the receiving module and configured for allocating, in accordance with the allocation request and a maintained mapping logic mapping hardware resources of at least one GPU on the first device to a unified virtual GPU, resources for running device code of the application program from the hardware resources of the at least one GPU on the first device; and

a sending module coupled with the allocating module configured for forwarding information of the allocated resource back to the host code of the application program.

10. The apparatus of Claim 9, further comprising:

an information obtaining module configured for:

sending a request to obtain resource details of each one of the GPUs of a set of GPUs;

receiving resource details forwarded back from each one of the GPUs in response to the request; or

receiving a registration request sent from each one of the GPUs of the set of GPUs, wherein the registration request comprises resource details;

wherein the resource details describe available hardware resources of each one of the GPUs; and

a maintaining module configured for generating a mapping logic mapping available hardware resources of the at least one GPU to a unified virtual GPU resource.

11. The apparatus of Claim 9, wherein the hardware resources of the at least one GPU comprises a video memory resource and a computational resource;

wherein further the allocating module is further configured for:

reading the mapping logic according to the allocation request; and

allocating, in accordance with the mapping logic, a first computational resource for the device code from the computational resource and allocating a first video memory resource for the device code from the video memory resource.

12. The apparatus of Claim 11, further comprising:

an applying module configured for managing, in accordance with the mapping logic, a virtual memory providing a total amount of available video memory resources analogous to the amount that the at least one GPU provides from a system memory of the first device;

a mapping module configured for generating a mapping relation between the virtual memory and the video memory resource provided by the at least one GPU;

wherein the allocating module is further configured for:

allocating a first computational resource for the device code from the computational resource, allocating a first storage resource for the device code from the virtual memory source, and labeling, in accordance with the mapping logic, the first video memory resource corresponding to the first storage resource to identify the first video resource allocated for the device code.

13. The apparatus of Claim 12, further comprising:

a read-write module configured for:

receiving a reading request sent from the host code to read data from the first storage resource and forwarding the data read back to the host code; and/or

receiving a writing request sent from the host code to write data to the first storage resource and synchronizing the first video memory resource with the data written according to the label of the first video memory resource.

14. The apparatus of Claim 12, wherein the mapping module is further configured for:

conducting paging to the virtual memory to obtain a memory page;

conducting paging to the video memory resource to obtain a video memory page;

generating a mapping relation between the memory page and the video memory page; and

wherein the read-write module is further configured for:

receiving a request to write sent from the host code to write data to the first storage resource, determining a first memory page where the data is written, and synchronizing the video memory page corresponding to the first memory page with the data in the first memory page.

15. The apparatus of Claim 11, wherein the allocating module is further configured for:

determining a sequencing of allocating the computational resources for the device code according to respective priorities of a plurality of allocation requests, and

allocating, according to the sequencing and the mapping logic, a first computational resource for the device code from the computational resource and a first video memory resource for the device code from the video memory resource.

16. The apparatus of Claim 11, wherein the sending module is further configured for:

sending labeling information of the first computational resource and address information of the first memory resource to the host code to let the host code send the device code to the first computational resource to have the first computational resource run the device code and save the data produced in running to the first memory resource and the labeled first video memory resource.

17. A computer system comprising a central processor and a plurality of GPUs coupled to memory wherein the memory comprises instructions that when executed implement a method of resource management, the method comprising:

receiving an allocation request to allocate resources sent from a host code of an application program located on a first device;

allocating, in accordance with the allocation request and a maintained mapping logic mapping available hardware resources of at least one GPU of the first device to a unified virtual GPU resource, resources for a device code of the application program from the available hardware resources of the at least one GPU of the first device; and

responsive to the allocating, forwarding information of allocated resources back to the host code of the application program.

18. The computer system of Claim 17, wherein the method further comprises, in accordance with the allocation request and the maintained mapping logic, and prior to the allocation of the resource for the device code of the application program from the available hardware resources of the at least one GPU of the first device, performing:

sending a request to obtain resource details to each one of the GPUs of a set of GPUs;

receiving resource details of each one of the GPUs forwarded back from each one of the GPUs in response to the request to obtain resource details, wherein the resource details describe the available hardware resources of each one of the GPUs;

generating the mapping logic mapping available hardware resources of the at least one GPU to the unified virtual GPU resource in accordance with the resource details of each one of the GPUs; or

receiving a registration request sent from each one of the GPUs of the set of GPUs, wherein the registration request comprises resource details; and

generating a mapping logic mapping available hardware resources of the at least one GPU to the unified virtual GPU resource in accordance with the resource details of

each one of the GPUs, wherein the resource details describes the available hardware resources of each one of the GPUs.

19. The computer system of Claim 17, wherein the available hardware resources of the at least one GPU comprise video memory resources and computational resources;

wherein further resources are allocated, in accordance with the allocation request and the maintained mapping logic, for the device code of the application program from the available hardware resources of the at least one GPU of the first device, and wherein the method further comprises:

reading, in accordance with the allocation request, the mapping logic; and

allocating, in accordance with the mapping logic, a first computational resource for the device code from the computational resources and a first video memory resource for the device code from the video memory resources.

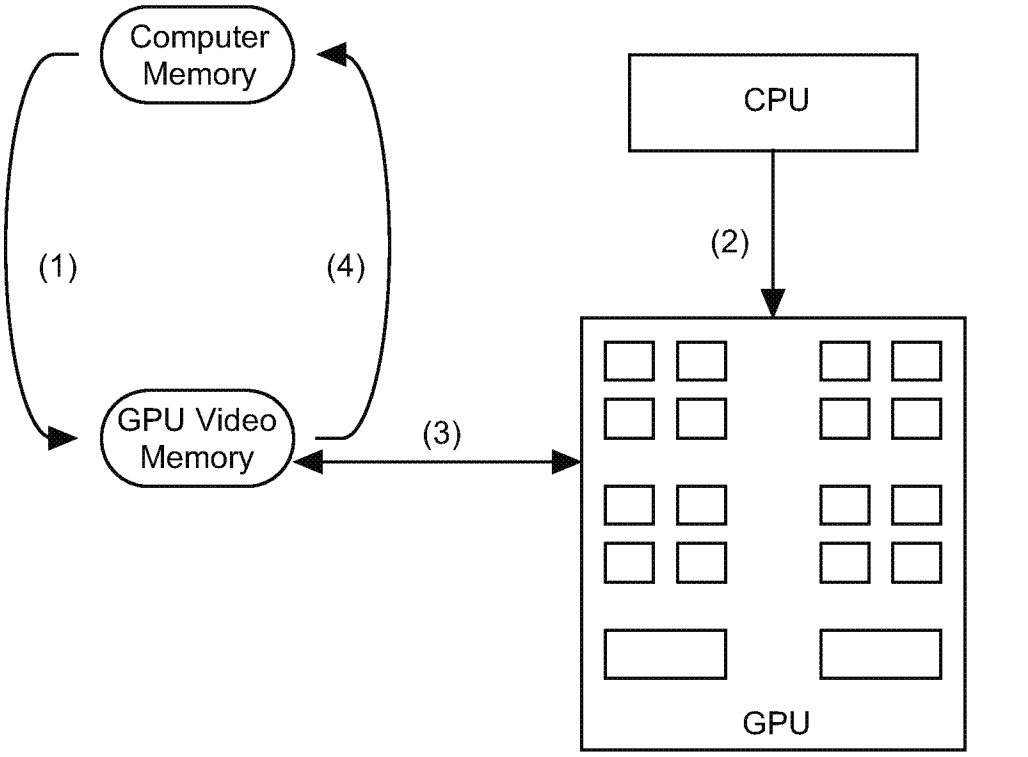
Computer Memory
CPU
(2)
(1)
(4)
GPU Video Memory
(3)
GPU
Multiple Graphics Processing Units

FIG. 1

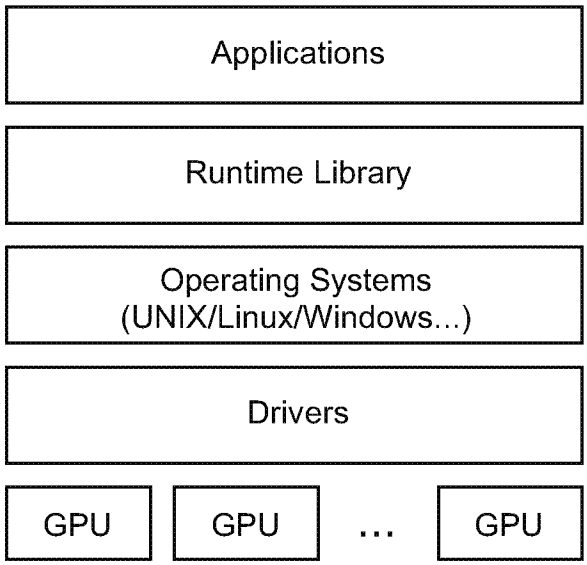
Applications
Runtime Library
Operating Systems (UNIX/Linux/Windows...)
Drivers
GPU
GPU
...
GPU

FIG. 2

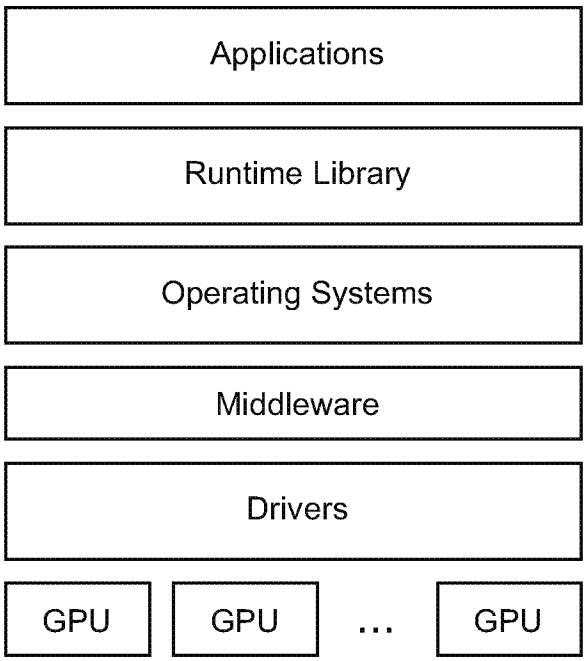
Applications
Runtime Library
Operating Systems
Middleware
Drivers
GPU
GPU
...
GPU

FIG. 3

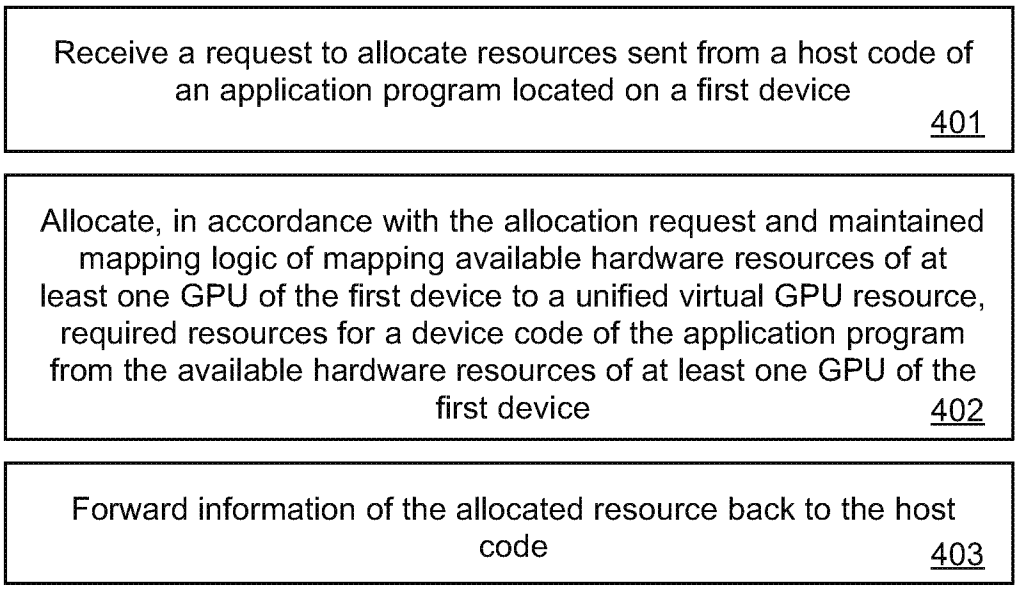
Receive a request to allocate resources sent from a host code of an application program located on a first device
401
Allocate, in accordance with the allocation request and maintained mapping logic of mapping available hardware resources of at least one GPU of the first device to a unified virtual GPU resource, required resources for a device code of the application program from the available hardware resources of at least one GPU of the first device
402
Forward information of the allocated resource back to the host code
403

FIG. 4

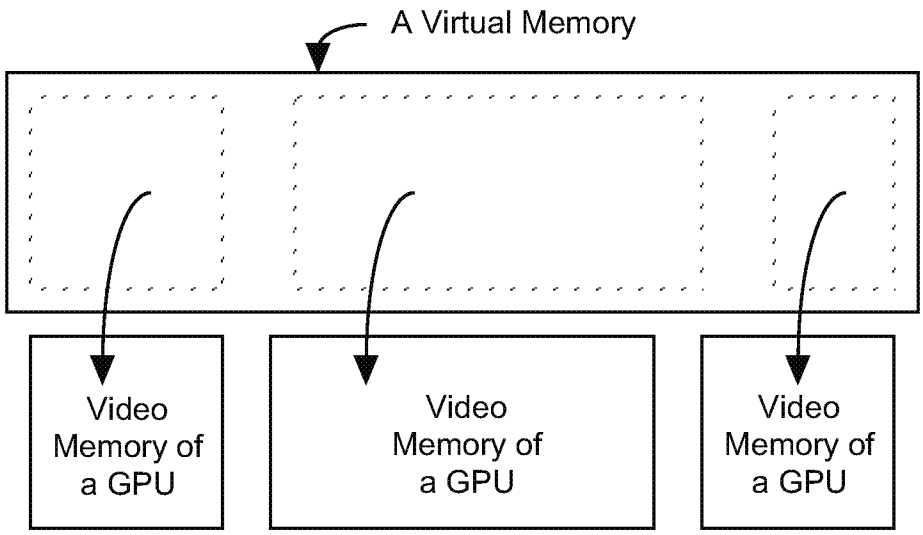

FIG. 5

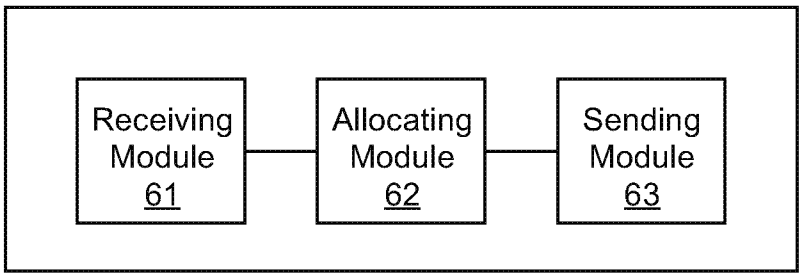

FIG. 6

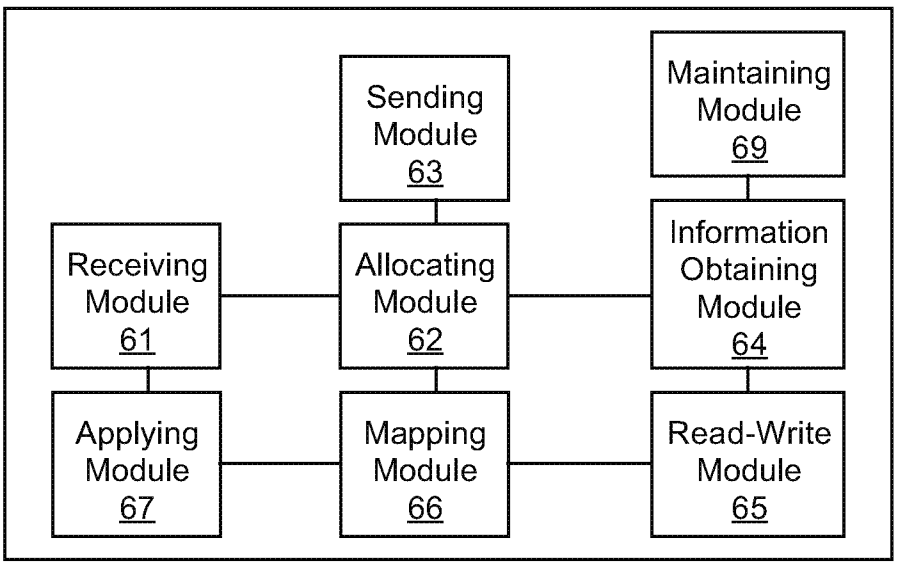

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/017024

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 9/455 (2016.01)
CPC - G06F 9/45558 (2016.01)
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC(8) - G06F 9/455, G06T 1/00, G06T 1/20 (2016.01)
CPC - G06F 9/45533, G06F 9/45537, G06F 9/4555, G06F 9/45558, G06F 9/5077, G06F 2009/45579, G06T 15/005 (2016.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC - 345/506, 345/522, 718/1 (keyword delimited)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Orbit, Google Patents, Google Scholar, Google
Search terms used: resource management, receive allocation request, allocate resources, host code of application program

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2006/0146057 A1 (BLYTHE) 06 July 2006 (06.07.2006), entire document	1-19
Y	KATO et al. "Gdev: First-class GPU resource management in the operating system." In: 2012 USENIX Annual Technical Conference (USENIX ATC 12). 15 June 2012 (15.06.2012) Retrieved from <https://www.usenix.org/system/files/conference/atc12/atc12-final319.pdf>, entire document	1-19
Y	WO 2012/083012 A1 (ADVANCED MICRO DEVICES, INC.) 21 June 2012 (21.06.2012), entire document	2, 10, 18
Y	US 2005/0168472 A1 (GOSALIA et al) 04 August 2005 (04.08.2005), entire document	4-6, 8, 12-14, 16
Y	US 2011/0202706 A1 (MOON et al) 18 August 2011 (18.08.2011), entire document	5, 13
A	US 2013/0021353 A1 (DREBIN et al) 24 January 2013 (24.01.2013), entire document	1-19
A	US 2013/0091500 A1 (EARL et al) 11 April 2013 (11.04.2013), entire document	1-19
A	US 2012/0011347 A1 (LITTLE et al) 12 January 2012 (12.01.2012), entire document	1-19
A	US 2015/0009222 A1 (NVIDIA CORPORATION) 08 January 2015 (08.01.2015), entire document	1-19

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

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

Date of the actual completion of the international search	Date of mailing of the international search report
30 March 2016	08 APR 2016
Name and mailing address of the ISA/ Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, VA 22313-1450 Facsimile No. 571-273-8300	Authorized officer Blaine R. Copenheaver PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774