



- (51) **International Patent Classification:**
G06F 9/50 (2006.01) *G06F 9/40* (2006.01)
- (21) **International Application Number:**
PCT/US2013/036982
- (22) **International Filing Date:**
17 April 2013 (17.04.2013)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (71) **Applicant:** EMPIRE TECHNOLOGY DEVELOPMENT LLC [US/US]; 2711 Centerville Road, Suite 400, Wilmington, DE 19808 (US).
- (72) **Inventors:** SHEAFFER, Gad, S.; 80 Grinboim, 34987 Haifa (IL). UR, Shmuel; Shorashim, 20164 Galil (IL).
- (74) **Agent:** LEE, David, S.; Brundidge & Stanger, P.C., 2318 Mill Road, Suite 1020, Alexandria, VA 22314 (US).
- (81) **Designated States** (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY,

BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

Published:

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

- (54) **Title:** SCHEDULING COMPUTING TASKS FOR MULTI-PROCESSOR SYSTEMS

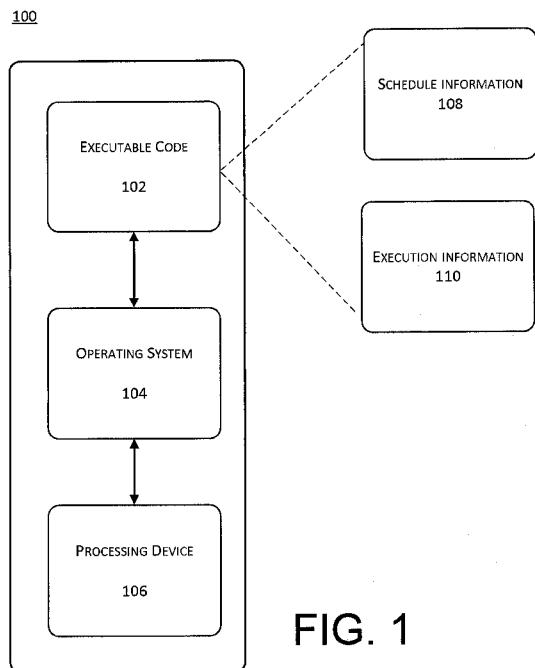


FIG. 1

(57) **Abstract:** In an example embodiment, one or more series of executable components may be configured to execute a respective processes, and one or more corresponding scheduling components may be configured to direct migration of each of the corresponding one or more series of executable components to a processing element thereof.

SCHEDULING COMPUTING TASKS FOR MULTI-PROCESSOR SYSTEMS**TECHNICAL FIELD**

[0001] The embodiments described herein pertain generally scheduling hardware utilization in a multi-processor system.

BACKGROUND

[0002] Unless otherwise indicated herein, the approaches described in this section are not prior art to the claims in this application and are not admitted to be prior art by inclusion in this section.

[0003] In a multi-processor system, the operating system, or any kernels external to an application, may schedule execution of one or more applications across a number of processors. Current solutions intended to improve hardware utilization include scheduling threads according to processor availability, typically to a predetermined type of available processor.

SUMMARY

[0004] Technologies are generally described for scheduling computing tasks for multi-processor systems, for example, heterogeneous systems. The various techniques may be implemented in various devices, methods and/or systems.

[0005] In some examples, various techniques may be implemented as a computer-readable medium. Some example computer-readable mediums storing a data structure may include executable code including one or more executable components that are configured to execute a respective process, and one or more scheduling components, at least some of which respectively corresponds to at least one of the executable components and configured to direct migration of the corresponding one of the executable components to a particular processing element.

[0006] In some examples, various techniques may be implemented as methods. Some methods may include registering a processor-enabled application program, compiling a scheduling component maintained by the application program, and migrating a thread to a processing element as directed by the compiled scheduling component.

[0007] In some additional examples, various techniques may be implemented as apparatuses. Some apparatuses may include a first component, including a first hardware element, to analyze executable instructions to generate at least a description of a particular processing element, a second component including a second hardware element, coupled to the first component, to integrate metadata maintained by at least one thread of the executable instructions, and a third component including a third hardware element, coupled to the second component, to migrate execution of the at least one thread to a processing element as directed by the integrated metadata.

[0008] The foregoing summary is illustrative only and is not intended to be in any way limiting. In addition to the illustrative aspects, embodiments, and features described above, further aspects, embodiments, and features will become apparent by reference to the drawings and the following detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] In the detailed description that follows, embodiments are described as illustrations only since various changes and modifications may be made in view of the following detailed description. The use of the same reference numbers in different figures indicates similar or identical items. In the drawings:

FIG. 1 shows an example computing environment in which one or more embodiments of scheduling for multi-processor systems may be implemented;

FIG. 2 shows an example configuration of executable code by which at least portions of scheduling for multi-processor systems may be implemented;

FIG. 3 shows an example configuration of a scheduling component by which at least portions of scheduling for multi-processor systems may be implemented;

FIG. 4 shows an example configuration of an operating system by which at least portions of scheduling for multi-processor systems may be implemented;

FIG. 5 shows an example configuration of a processing device by which at least portions of scheduling for multi-processor systems may be implemented;

FIG. 6 shows an example configuration of a processing flow of operations for scheduling for multi-processor systems;

FIG. 7 shows an example configuration of an apparatus by which scheduling for multi-processor systems may be implemented; and

FIG. 8 shows a block diagram illustrating an example computing device by which various example solutions described herein may be implemented;

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

DETAILED DESCRIPTION

[0010] In the following detailed description, reference is made to the accompanying drawings, which form a part of the description. In the drawings, similar symbols typically identify similar components, unless context dictates otherwise. Furthermore, unless otherwise noted, the description of each successive drawing may reference features from one or more of the previous drawings to provide clearer context and a more substantive explanation of the current example embodiment. Still, the example embodiments described in the detailed description, drawings, and claims are not meant to be limiting. Other embodiments may be utilized, and other changes may be made, without departing

from the spirit or scope of the subject matter presented herein. The aspects of the present disclosure, as generally described herein and illustrated in the drawings, may be arranged, substituted, combined, separated, and designed in a wide variety of different configurations, all of which are explicitly contemplated herein.

[0011] FIG. 1 shows an example computing environment 100 in which one or more embodiments of scheduling for multi-processor systems may be implemented, in accordance with at least some embodiments described herein. Computing environment 100 may be implemented, for example, in one or more computing devices communicatively coupled to each other via a network. As depicted, computing environment 100 may include executable code 102, an operating system 104, and a processing device 106, which are operatively/communicatively coupled to each other. In at least some examples, executable code 102 may include schedule information 108 and execution information 110. Schedule information 108 may communicate minimal or adequate requirements, an optimal or adequate execution environment, and/or other performance parameters to operating system 104. Operating system 104 may utilize such received information to schedule tasks related to execution of at least portions of executable code 102 on processing device 106 or resources corresponding thereto. However, this configuration is an example only, and other configurations are possible.

[0012] Executable code 102 may be designed, programmed, and/or configured to provide, to operating system 104, information regarding allocation of resources corresponding to processing device 106 to implement efficient execution performance and hardware utilization. Executable code 102 may be further designed, programmed, and/or configured as computer program products, applications, or other forms of software code including vector codes, or other forms of computer-readable instructions that are executable by one or more processors. Thus, in accordance with examples of scheduling for multi-processor systems, scheduling components of executable code 102 may provide operating system 104 with information to enhance execution performance and hardware utilization for processing device

106; and executable components of executable code 102 may include executable instructions for at least portions of processing device 106 to, *e.g.*, write data to storage systems, to render three-dimension models, to perform scientific calculations, or to perform other operations each of which may have different hardware utilization parameters, such as minimum hardware requirements.

[0013] Operating system 104 may be designed, programmed, and/or configured to assign certain computing resources of processing device 106 to execution of at least portions of executable code 102, based on the aforementioned information received from executable code 102. In some examples, operating system 104 may be designed, programmed, and/or configured to schedule execution of at least portions of executable code 102 on processing device 106, in accordance with availability of corresponding resources, and a “first-in-first-out” queue, based on the aforementioned information received from executable code 102. Further to the examples, execution of a portion or portions of executable code 102 may be delayed while processing device 106 executes other portions of executable code 102 or is otherwise occupied by operating system 104. Thus, in accordance with the information received from executable code 102 prior to execution thereof, on-hold portions of executable code 102 may be added to a “first-in-first-out” (FIFO) queue, with a portion of executable code 102 that is first added to the FIFO queue being assigned to processing device 106 for execution when processing device 106 is available. In other examples, when processing device 106 is configured as a multi-core system including more than one cores of different types, the information received from executable code 102 prior to execution thereof may instruct operating system 104 to schedule execution of certain portions of executable code 102 to a particular type of core. For example, rendering three-dimension models may only be assigned to Graphics Processing Units (GPU).

[0014] Operating system 104 may refer to any proprietary or open-source operating system including, but not limited to, Microsoft® Windows®, UNIX®, Berkeley Software Distribution (BSD) and its descendants, Apple® OS X®, Linux® and GNU, and Google® Chrome® OS. Other operating system(s) may

be used for operating system 104. Operating system 104 may or may not be hosted on processing device 106 or one or more elements thereof.

[0015] Processing device 106 may refer to, as examples, any one of Graphics Processing Units (GPU), Central Processing Units (CPU), or other types of processing units and cores thereof. After being assigned to process certain portions of executable code 102 by operating system 104, based on the information received from executable code 102 prior to execution thereof, processing device 106 may carry out the instructions embedded in executable code 102 and perform arithmetical, logical, input/output, or other operations of computing environment 100. In accordance with some examples, processing device 106 may include one or more processing elements so that operating system 104 may be configured to migrate, or schedule, the execution of certain portions of executable code 102 to different ones of the one or more processing elements.

[0016] Thus, **FIG. 1** shows an example environment for implementation of the scheduling mechanism or method in accordance with various embodiments.

[0017] **FIG. 2** shows an example configuration 200 of executable code 102 by which at least portions of scheduling for multi-processor systems may be implemented, arranged in accordance with at least some embodiments described herein. Example configuration 200 includes or may be embodied as a non-transitory computer readable medium having a data structure stored thereon that includes executable code 102 and scheduling components 204. As depicted, example configuration 200 of executable code 102 includes schedule information 108 and execution information 110. Execution information 110 may include executable components 202A – 202N and schedule information 108 may include scheduling components 204A – 204N. Scheduling components 204A – 204N may provide information regarding, for example, minimal requirements and/or an optimal execution environment of one or more of executable components 202A – 202N. Each of scheduling components 204A - 204N may respectively correspond to at least one of the executable components 202A – 202N. However, the

configuration is an example only and other configurations are possible. While executable code 102 may be implemented as software programs, applications, or processes, other examples of executable components 202A – 202N may include one or more smaller sequences or blocks of executable code 102, *e.g.*, processes of applications or threads of processes. Such examples are not intended to be exhaustive, and therefore should not be interpreted to be so. Unless a distinction of one or more of executable components 202A – 202N is needed to enhance the description for scheduling for multi-processor systems, general references may be made herein to “executable components 202.” Similarly, scheduling components 204A – 204N may be referred as “scheduling components 204” if no distinction is needed.

[0018] Executable components 202, in accordance with some examples, may be configured to execute respective processes, which may be smaller sequences or blocks of executable code 102 including one or more threads of a respective process.

[0019] Scheduling components 204, each of which may respectively correspond to at least one of the executable components 202, may direct migration of each of corresponding executable components 202 to a preferred or adequate processing element by describing minimal or adequate processing requirements or parameters and/or a preferred or adequate execution environment for executing one or more corresponding executable components 202. Scheduling components 204 may describe an optimal or adequate processing element, which may identify the preferred or adequate execution environment for a corresponding one of executable components 202. The preferred or adequate processing element may be one of a series of processing elements including any type of processing units, such as GPUs, GPUs, and cores.

[0020] As set forth above, executable code 102 may be designed, programmed, and/or configured to provide, to operating system 104, scheduling information regarding allocation of computing resources corresponding to processing device 106 to implement preferred or adequate

execution performance and hardware utilization. To establish communication with operating system 104, in accordance with at least in some example embodiments, executable code 102 may use a system call to register scheduling components 204 with operating system 104, a run-time environment, a device driver, a hyper-visor and/or other element for the location and organization of scheduling components 204. In some example embodiments, the characteristics of executable components 202 may change and cause scheduling components 204 to change the description of the minimal or adequate requirements and/or optimal or adequate execution environment, thus executable code 102 may initiate a new system call to notify operating system 104, the run-time environment, the device driver, the hyper-visor, and/or other element.

[0021] Thus, FIG. 2 shows an example configuration 200 of executable code 102 by which at least portions of scheduling for multi-processor systems may be implemented in accordance with various embodiments.

[0022] FIG. 3 shows an example configuration 300 of one of scheduling components 204 by which at least portions of scheduling for multi-processor systems may be implemented, arranged in accordance with at least some embodiments described herein. As depicted, example configuration 300 of one of scheduling components 204 includes at least one power requirement 302A, at least one time constraint 302B, at least one memory capacity requirement 302C, and/or other types of information 302N (including alternative or additional values for 302A-302C etc. in some embodiments).

[0023] Scheduling components 204 store information regarding minimal or adequate requirements, including minimal amounts of identified types of computing resources and/or a preferred execution environment, regarding execution of corresponding ones of executable components 202. The minimal requirements and/or preferred execution environment may further include power requirements 302A, time constraints 302B, memory capacity requirements 302C, and/or other types of information 302N usable to provide at least adequate execution performance; however, this

configuration is an example only, and other configurations are possible. Scheduling components 204 may further provide information to describe current and/or upcoming program phases in terms that are generic so as to allow flexibility for operating system 104 to schedule certain computing resources of processing device 106. For example, a length of a next phase or phases of executable code 102 may be included in the information from a respective one of scheduling components 204 so that operating system 104 may schedule the computing resources before the next phase or phases begin.

[0024] In one example embodiment of scheduling components 204, a compiler may generate executable code 102 for a specific loop and provide information in scheduling components 204 as to the number of elements in a specific vector, whether specific vector architecture support is required or usable, a data footprint in a first level cache for which a corresponding portion of executable code 102 is optimized, and/or other information. Additionally, a dependency between different computing resources of processing device 106 may be provided in one or more of scheduling components 204. For example, when a large matrix is operated on in sub-matrices, at least one of which may be processed by one of the computing resources, the dependency of between the computing resources may be provided in scheduling components 204. In another example embodiment of scheduling components 204, a compiler may provide multiple code paths for specific portions of executable code 102 and may further indicate an appropriate code path for each part of processing device 106 for which one or more portions of executable code 102 may be scheduled.

[0025] Information stored in respective ones of scheduling components 204 may be provided in a plurality of mechanisms and combinations thereof. Further, in accordance with some example embodiments for providing the information, such information may be agnostic to any programming language by which executable code 102 and/or scheduling components 204 are designed, programmed, and/or configured. For example, a programmer may describe the information in a natural language, *e.g.*, English. Alternatively, a programmer may partly or fully provide the information by pragmas and/or

declarations, via keywords in a programming language. In yet another example, the information may be provided in a format that is capable of being analyzed by a compiler with or without profiling. Further still, the information may be collected from sampling of run-time code.

[0026] In accordance with some example embodiments for providing the information in scheduling components 204, a programmer may not be required to input or determine such minimal or adequate requirements and/or preferred or adequate execution environment. Instead, such information may be proactively assigned to executable code 102 from a real time computing environment. For example, during execution of certain portions of executable components 202, *e.g.*, rendering a three-dimension model, executable code 102 may collect statistical performance data and generate a minimal requirement for execution thereof, such for example a GPU with 1GB (Gigabytes) memory being preferred or suggested, or even optimal, for execution of such portions of executable components 202. Such minimal requirement or other information may then be stored in one or more of scheduling components 204 as memory capacity requirement 302C. Further, information contained in scheduling components 204 may be modifiable during execution of executable code 102, such as when the characteristics of executable code 102 change during different phases, as described previously.

[0027] Computing resources, *e.g.*, a core, to which one or more portions of executable code 102 are scheduled, may fail. Thus, in one or more example embodiments, information in one or more of scheduling components 204 may include a pointer to a source code of specific portions of executable code 102 so that a compiler may generate a version of executable code 102, which may not have been deemed necessary when the code was statically compiled, but is currently executable by other available resources of processing device 106. In other example embodiments, executable code 102 may proactively change the information stored in scheduling components 204 and initiate a new system call to notify the operating system 104.

[0028] In some embodiments, scheduling components 204 may be maintained semantically close to a Central Processing Unit Identification (CPUID) tree in a homogeneous system including processors or, even in a heterogeneous system, in which processing device 106 includes multiple different processors, however, having essentially the same instruction set architecture. In other embodiments, in which processing device 106 includes processing elements having different structures, such as a vector-oriented main CPU and a thread-oriented GPU, scheduling components 204 may be abstract in specification.

[0029] Thus, **FIG. 3** describes an example configuration 300 of one of scheduling components 204 by which at least portions of scheduling for multi-processor systems may be implemented in accordance with various embodiments.

[0030] **FIG. 4** shows an example configuration 400 of operating system 104 by which at least portions of scheduling for multi-processor systems may be implemented, arranged in accordance with at least some embodiments described herein. As depicted, example configuration 400 includes operating system 104 and schedulers 402A – 402N.

[0031] Schedulers 402A – 402N may refer to components or modules corresponding to operating system 104 that are designed, programmed, and/or configured to schedule computing resources of processing device 106 for one or more portions of executable code 102. In some embodiments, schedulers 402A – 402N may form part of operating system 104 itself, while schedulers 402A – 402N may be external to operating system 104 in other embodiments. Unless a distinction of schedulers 402A – 402N is needed to enhance the description for the scheduling mechanism, general references may be made herein to “schedulers 402.” In some examples, operating system 104 may include a global scheduler that governs schedulers 402, which may be configured to perform the same function as referenced herein.

[0032] In accordance with some example embodiments, information from one or more scheduling components 204 may be received from operating system 104, and operating system 104 may convey the received information to schedulers 402, one or more of which may correspond to at least one of executable components 202. Schedulers 402 may then schedule computing resources of processing device 106 to the corresponding executable components 202.

[0033] In some example embodiments, when executable code 102 proactively changes information in scheduling components 204 during phase shifting, operating system 104 may consult scheduling components 204, and schedulers 402 may accordingly opt to migrate execution of executable components 202 between computing resources of processing device 106 that have different architectures. For example, if an example phase of executable code 102 may use a larger number of executable components, *e.g.*, threads, which are latency tolerant, time constraints 302B in scheduling components 204 may be updated to indicate that GPUs may be a preferred or adequate processing environment for that phase of executable code 102. Schedulers 402 may then migrate execution of executable code 102 from a CPU to a GPU accordingly.

[0034] In some alternative or additional examples, executable code 102 may have a data footprint for which, due to its considerable size, latency may be deemed to be significant. Accordingly, time constraints 302B and memory capacity requirements 302C may be updated to indicate that a preferred or adequate processing environment may include a core having caches of a certain capacity. Schedulers 402 may then migrate at least portions of execution of executable code 102 to a core of CPUs having caches of the suitable capacity, if available.

[0035] In yet another example, if processing of at least portions of executable code 102 value speed over precision, *e.g.*, executable code 102 includes IEEE 754 compliant floating point calculation, schedulers 403 may schedule all available computing resources of processing device 106 without scheduling any verification process to avoid undue consumption of computing resources.

[0036] Thus, **FIG. 4** shows an example configuration 400 of operating system 104 by which at least portions of scheduling for multi-processor systems may be implemented in accordance with various embodiments.

[0037] **FIG. 5** shows an example configuration 500 of processing device 106 by which at least portions of scheduling for multi-processor systems may be implemented, arranged in accordance with at least some embodiments described herein. As depicted, example configuration 500 includes processing device 106, processing elements 502A – 502N, and memory units 504A – 504N. Processing elements 502A – 502N may be any of CPUs, GPUs, cores, cloud computing resources, other types of resources, and/or any combination thereof. Memory units 504A – 504N may be any of Hard Disk Drive (HDD), Solid State Drive (SSD), Random Access Memory (RAM), cache, online storage space, other storage devices and/or any combination thereof. Unless a distinction of processing elements 502A – 502N is needed to enhance the description for the scheduling mechanism, general references may be made herein to “processing elements 502.” Similarly, memory units 504A – 504N may be referred as “memory units 504” if no distinction is needed.

[0038] Memory units 504 may be physically or communicatively coupled to processing elements 502, while processing elements 502 may be scheduled to different executable components 202 to store data temporarily or permanently for execution. Such configuration is an example only and other configurations are possible.

[0039] In a heterogeneous multi-core system, some of processing elements 502 may differ with regard to architecture and/or capabilities. For example, a respective one of processing elements 502 that is smaller but more energy efficient than others may be scheduled to execute those latency tolerant portions of executable components 202. In at least one other example embodiment, operating system 104 may include information that describes processing elements 502 and capabilities thereof. Such information may include CPU frequency, GPU frequency, cache size, the number of cores, and etc.

Executable code 102 may evaluate the information and calculate a preferred or adequate one or more of processing elements 502 for current and future phases. Scheduling components 204 may be updated accordingly.

[0040] In some examples, executable code 102 may have a data footprint for which, due to its considerable size, latency may be deemed to be substantial, and so time constraints 302B and memory capacity requirements 302C may be updated to indicate that the optimal or adequate processing units may be a core with caches of a particular capacity that have a different architecture. Schedulers 402 may schedule such a core having the suitable cache capacity among processing elements 502.

[0041] Thus, FIG. 5 shows an example configuration 500 of a processing device 106 by which at least portions of scheduling for multi-processor systems may be implemented in accordance with various embodiments.

[0042] FIG. 6 shows an example configuration of a processing flow 600 of operations for scheduling for multi-processor systems, arranged in accordance with at least some embodiments described herein. As depicted, processing flow 600 includes sub-processes executable by various components (including one or more processors or other hardware element) that are part of computing environment 100. However, processing flow 600 is not limited to such components, and modification may be made by re-ordering two or more of the sub-processes described here, eliminating at least one of the sub-processes, adding further sub-processes, substituting components, having various components assuming sub-processing roles accorded to other components in the following description, and/or combinations thereof. Processing flow 600 may include various operations, functions, or actions as illustrated by one or more of blocks 602, 604, and/or 606. Processing may begin at block 602.

[0043] Block 602 (Register Process-Enabled Program) may refer to operating system 104 registering executable code 102, via system calls or other registration technique initiated by executable code 102. Executable code 102 may include computer program products, processor-enabled programs,

applications, processes, or other computer-readable instructions that are executable by one or more processors, and may further include executable components 202 and scheduling components 204. Executable components 202 may be sub-sequences of executable code 102, *e.g.*, processes of applications or threads of processes. To establish communication with operating system 104, at least in some embodiments, executable code 102 may use a system call or other technique to register the location and organization of scheduling components 204 with operating system 104, a run-time environment, a device driver, a hyper-visor, and/or other element. In some embodiments, in which characteristics of executable components 202 may change and further cause scheduling components 204 to change a description of the minimal or adequate requirements and/or a preferred or adequate execution environment, executable code 102 may initiate a new system call or other communication to notify operating system 104, the run-time environment, the device driver, the hyper-visor, and/or other element. Processing may continue from block 602 to block 604.

[0044] Block 604 (Analyzing Scheduling Component) may refer to a compiler analyzing one of scheduling components 204 maintained by executable code 102, *e.g.*, processor-enabled program. Scheduling components 204 may thus comprise information regarding minimal or adequate requirements such as minimal amounts of identified computing resources and/or preferred or adequate execution environment for executing the corresponding executable components 202. The minimal or adequate requirements and/or preferred or adequate execution environment may further include power requirements 302A, time constraints 302B, memory capacity requirements 302C, and/or other types of information 302N. Processing may continue from block 604 to block 606.

[0045] Block 606 (Migrate Thread to Processing Element) may refer to schedulers 402 migrating one of executable components 202, *e.g.*, a thread, from one of processing elements 502 to another thereof as directed by the compiled one of scheduling components 204. In at least one example embodiment, the one of processing elements 502 may be a core. When executable code 102

updates information in scheduling components 204 during phase shifting, operating system 104 may then consult scheduling components 204, and schedulers 402 may accordingly opt to migrate the execution of executable components 202 between various ones of processing elements 502 of processing device 106 to thereby improve hardware utilization. Further, processing flow 600 may include a sub-process for executing the one of executable components 202, *e.g.*, a thread, utilizing the one of processing elements 502, *e.g.*, a core, directed by the compiled one of scheduling components 204.

[0046] Thus, FIG. 6 shows an example configuration of a processing flow 600 of operations for scheduling for multi-processor systems in accordance with various embodiments.

[0047] FIG. 7 shows an example configuration of an apparatus 700, as an overall device that may include executable code 102, operating system 104, and processing device 106, by which scheduling for multi-processor systems may be implemented, arranged in accordance with at least some embodiments described herein. Apparatus 700 may be a processor, and therefore as depicted, apparatus 700 may include an analyzing element 702, an integrating element 704, and a migrating element 706, which may be operatively or communicatively coupled to each other, and at least some or portion of these elements may be implemented in hardware in various embodiments.

[0048] Analyzing element 702 may be a component that is configured to analyze executable instructions to identify scheduling components 204 that include at least a description of a particular processing element. In accordance with some examples, scheduling components 204 may store information regarding minimal or adequate requirements, including minimal amounts of identified types of computing resources and/or a preferred execution environment, regarding execution of corresponding ones of executable components 202. The executable instructions may be in the form of executable code 102 including applications, programs, functions, and/or other computer-readable

instructions executable by one or more processors. The executable instructions may further comprise executable components 202, which may include processing threads.

[0049] Integrating element 704 may be a component that is configured to integrate metadata maintained by at least one of executable components 202 of executable code 102. In some embodiments, integrating element 704 may be one of an interpreter, converter, compiler, operating system, virtual machine, runtime environment, and/or other element. The integrated metadata may be embedded in each of scheduling components 204 and may include processing resource preferences for the at least one of executable components 202. The processing resource preferences may include a type of core, power requirements, time constraints, memory capacity requirements, and/or other parameters.

[0050] Migrating element 706 may be a component that is configured to migrate execution of the at least one of executable components 202 to one of processing elements 502 as directed by the integrated metadata. Processing elements 502 may be any of Central Processing Units, Graphics Processing Units, cores, cloud computing resources, other resource, and/or any combination thereof. In some example embodiments, the processing resource preferences may indicate that a certain type of core may improve the hardware utilization if the at least one of executable components 202 (*e.g.*, threads) is executed by the type of core. Migrating element 706 may then migrate execution of the at least one of executable components to this type of core.

[0051] Thus, FIG. 7 shows an example configuration of an apparatus for implementing scheduling for multi-processor systems in accordance with various embodiments.

[0052] FIG. 8 is a block diagram illustrating an example computing device 800 that is arranged for scheduling computing tasks for multi-processor systems in accordance with the present disclosure. Example computing device 800 may include apparatus 700. In an example configuration 802, computing device 800 typically includes one or more processors 804, *e.g.*, processor 804A and processor 804B, and

a system memory 806. Processor 804A and processor 804B may be collectively referred as “processor 804” herein. A memory bus 808 may be used for communicating between processor 804 and system memory 806.

[0053] Depending on a particular configuration, processor 804 may be of any type including but not limited to a microprocessor (μ P), a microcontroller (μ C), a digital signal processor (DSP), or any combination thereof. Processor 804 may include one more levels of caching, such as a level one cache 810 and a level two cache 812, a processor core 814, and registers 816. An example processor core 814 may include an arithmetic logic unit (ALU), a floating point unit (FPU), a digital signal processing core (DSP Core), or any combination thereof. An example memory controller 818 may also be used with processor 804, or in some implementations memory controller 818 may be an internal part of processor 804.

[0054] Depending on a particular configuration, system memory 806 may be of any type including but not limited to volatile memory (such as RAM), non-volatile memory (such as ROM, flash memory, etc.) or any combination thereof. System memory 806 may include an operating system 820 (which may embody operating system 104 previously described above), one or more applications 822, and program data 824. Operating system 820 may be configured to perform similar functions as operating system 104. Application 822 may include a scheduling algorithm 826 that is arranged to perform the functions as described herein including those described in FIG 6. Program data 824 may include executable code 102 and multi-processor scheduling data 828 that may be useful for scheduling computing tasks for multi-processor system as is described herein. Scheduling data 828 may include schedule information 108 described in FIG. 1. In some embodiments, application 822 may be arranged to operate with program data 824 on operating system 820 such that implementations of scheduling computing tasks for multi-processor systems may be provided as described herein. This described configuration 802 is illustrated in FIG. 8 by those components within the inner dashed line.

[0055] Computing device 800 may have additional features or functionality, and additional interfaces to facilitate communications between configuration 802 and any particular devices and interfaces. For example, a bus/interface controller 830 may be used to facilitate communications between configuration 802 and one or more data storage devices 832 via a storage interface bus 834. Data storage devices 832 may be removable storage devices 836, non-removable storage devices 838, or a combination thereof. Examples of removable storage and non-removable storage devices include magnetic disk devices such as flexible disk drives and hard-disk drives (HDD), optical disk drives such as compact disk (CD) drives or digital versatile disk (DVD) drives, solid state drives (SSD), and tape drives to name a few. Example computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data.

[0056] System memory 806, removable storage devices 836 and non-removable storage devices 838 are examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which may be used to store information and which may be accessed by computing device 800. Any such computer storage media may be part of computing device 800.

[0057] Computing device 800 may also include an interface bus 840 for facilitating communication from various interface devices (*e.g.*, output devices 842, peripheral interfaces 844, and communication devices 846) to configuration 802 via bus/interface controller 830. Example output devices 842 include a graphics processing unit 848 and an audio processing unit 850, which may be configured to communicate to various external devices such as a display or speakers via one or more A/V ports 852. Example peripheral interfaces 844 include a serial interface controller 854 or a parallel

interface controller 856, which may be configured to communicate with external devices such as input devices (*e.g.*, keyboard, mouse, pen, voice input device, touch input device, *etc.*) or other peripheral devices (*e.g.*, printer, scanner, *etc.*) via one or more I/O ports 858. An example communication device 846 includes a network controller 860, which may be arranged to facilitate communications with one or more other computing devices 862 over a network communication link via one or more communication ports 864.

[0058] The network communication link may be one example of a communication media. Communication media may typically be embodied by computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport mechanism, and may include any information delivery media. A “modulated data signal” may be a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media may include wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, radio frequency (RF), microwave, infrared (IR) and other wireless media. The term computer readable media as used herein may include both storage media and communication media.

[0059] Computing device 800 may be implemented as a portion of a small-form factor portable (or mobile) electronic device such as a cell phone, a personal data assistant (PDA), a personal media player device, a wireless web-watch device, a personal headset device, an application specific device, or a hybrid device that include any of the above functions. Computing device 800 may also be implemented as a personal computer including both laptop computer and non-laptop computer configurations.

[0060] The above description of illustrated embodiments, including what is described in the Abstract, is not intended to be exhaustive or to be limited to the precise forms disclosed. While specific embodiments and examples are described herein for illustrative purposes, various equivalent

modifications are possible and can be made in light of the above detailed description. The terms used in the following claims should not be construed to be limited to the specific embodiments disclosed in the specification and the claims. Rather, the scope is to be determined entirely by the following claims, which are to be construed in accordance with established doctrines of claim interpretation.

WE CLAIM

1. A non-transitory computer-readable medium having a data structure stored thereon that comprises:

executable code including one or more executable components that are configured to execute a respective process; and

one or more scheduling components, at least some of which respectively correspond to at least one of the executable components and are configured to direct migration of the corresponding one of the executable components to a particular processing element.

2. The computer-readable medium of Claim 1, wherein the data structure includes an application or program.

3. The computer-readable medium of Claim 1, wherein the one or more scheduling components include at least a description of the particular processing element.

4. The computer-readable medium of Claim 1, wherein the one or more scheduling components include at least a description of the particular processing element based on power demand for the respective one of the executable components.

5. The computer-readable medium of Claim 1, wherein the one or more scheduling components include at least a description of the particular processing element based on time constraints for the respective one of the executable components.

6. The computer-readable medium of Claim 1, wherein the particular processing element includes a core of a multicore processor.

7. The computer-readable medium of Claim 1, wherein the particular processing element is a core within a different architecture than a current architecture for a currently executed process.

8. The computer-readable medium of Claim 1, wherein the one or more scheduling components is agnostic to a programming language of the at least one of description of the particular processing element.

9. The computer-readable medium of Claim 1, wherein the one or more scheduling components includes keywords written in a programming language.

10. The computer-readable medium of Claim 1, wherein the one or more scheduling components are configured to be analyzed by a compiler.

11. The computer-readable medium of Claim 1, wherein the one or more scheduling components are configured as sampling based runtime code.

12. The computer-readable medium of Claim 1, wherein the one or more scheduling components are configured to direct migration of the corresponding one of the executable components to the particular processing element thereof by listing one or more of minimal type of resources to execute the respective process or minimal amounts of the resources to execute the respective process.

13. The computer-readable medium of Claim 12, wherein the resources include memory capacity.

14. The computer-readable medium of Claim 1, wherein the executable code comprises a system call that is configured to register the one or more scheduling components with an operating system.

15. The computer-readable medium of Claim 14, wherein the executable code further comprises a system call that is configured to register the one or more scheduling components with a runtime environment.

16. The computer-readable medium of Claim 14, wherein the executable code further comprises a system call that is configured to register the one or more scheduling components with a device driver.

17. A method, comprising:

registering a processor-enabled application program;

analyzing a scheduling component maintained by the application program; and

migrating a thread to a processing element as directed by the compiled scheduling component.

18. The method of Claim 17, further comprising executing the thread utilizing the processing element directed by the compiled scheduling component.

19. The method of Claim 17, wherein the scheduling component is not written in a programming language.

20. The method of Claim 17, wherein the scheduling component includes keywords written in a programming language.

21. The method of Claim 17, wherein the processing element includes a core or a multicore processor.

22. The method of Claim 17, wherein said migrating includes migrating the thread to the processing element based on one or both of power demand or time constraints for the thread.

23. An apparatus, comprising:

a first component, including a first hardware element, to analyze executable instructions to identify metadata that includes at least a description of a particular processing element;

a second component including a second hardware element, coupled to the first component, to integrate the metadata maintained by at least one thread of the executable instructions; and

a third component including a third hardware element, coupled to the second component, to migrate execution of the at least one thread to a processing element as directed by the integrated metadata.

24. The apparatus of Claim 23, wherein the first, second, and third components are part of a processor.

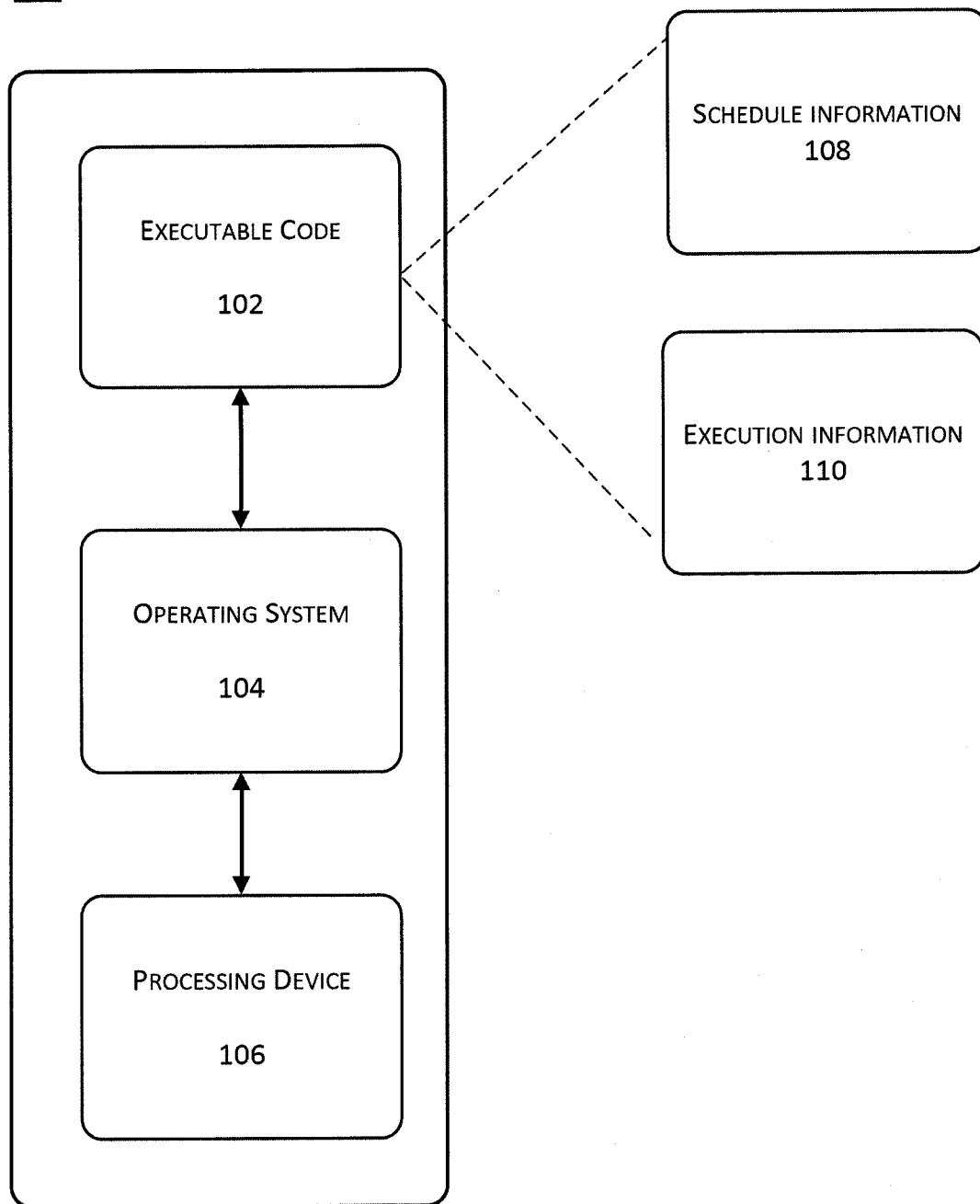
25. The apparatus of Claim 23, wherein the executable instructions include one of an application, program, or function.

26. The apparatus of Claim 23, wherein the second component to integrate includes one of an interpreter, converter, compiler, virtual machine, or runtime environment.

27. The apparatus of Claim 23, wherein the integrated metadata includes processing resource parameters for the at least one thread.

28. The apparatus of Claim 23, wherein the processing resource parameters include a type of core of a multicore processor.

29. The apparatus of Claim 23, wherein the third component to migrate execution includes a driver.

100**FIG. 1**

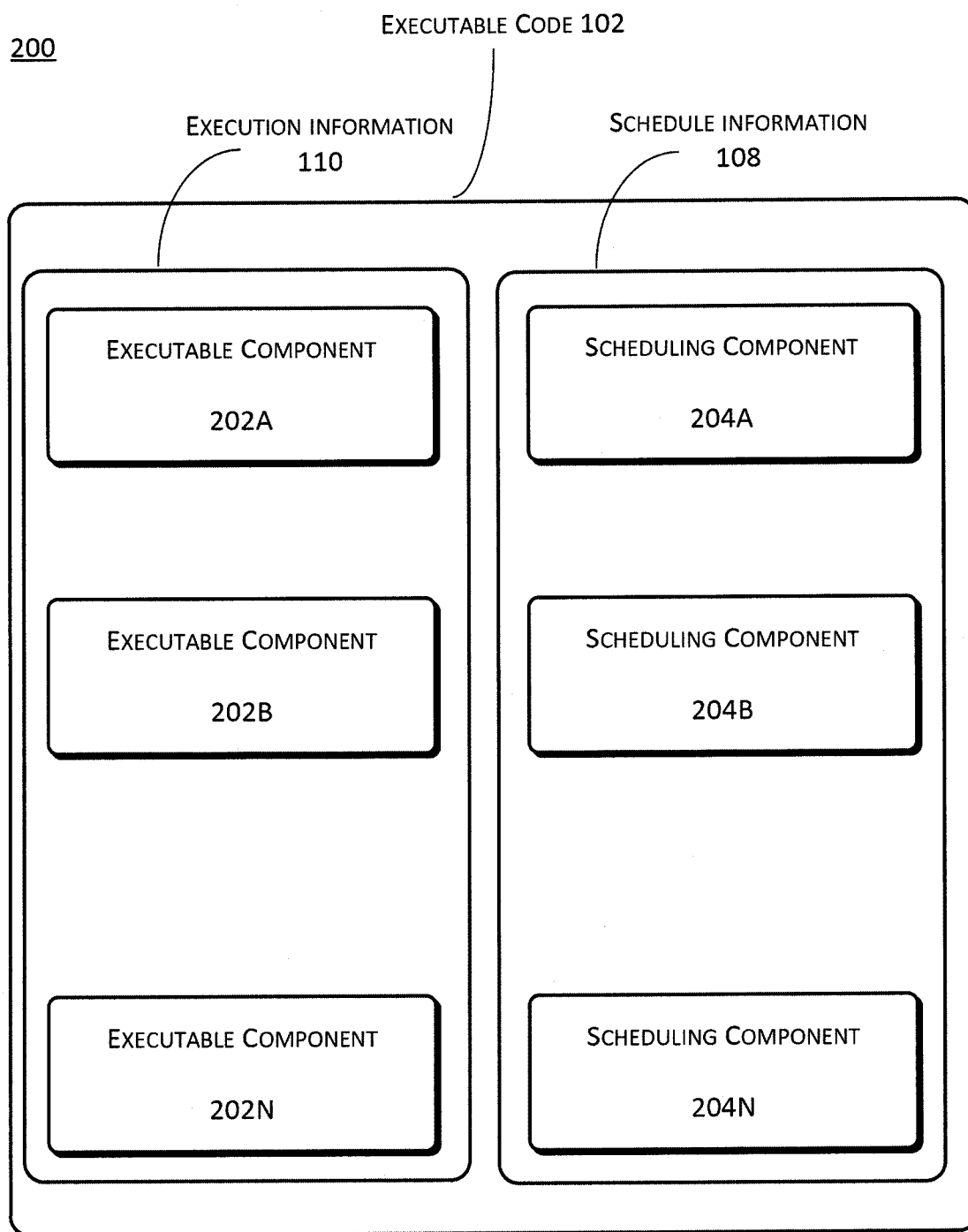


FIG. 2

300

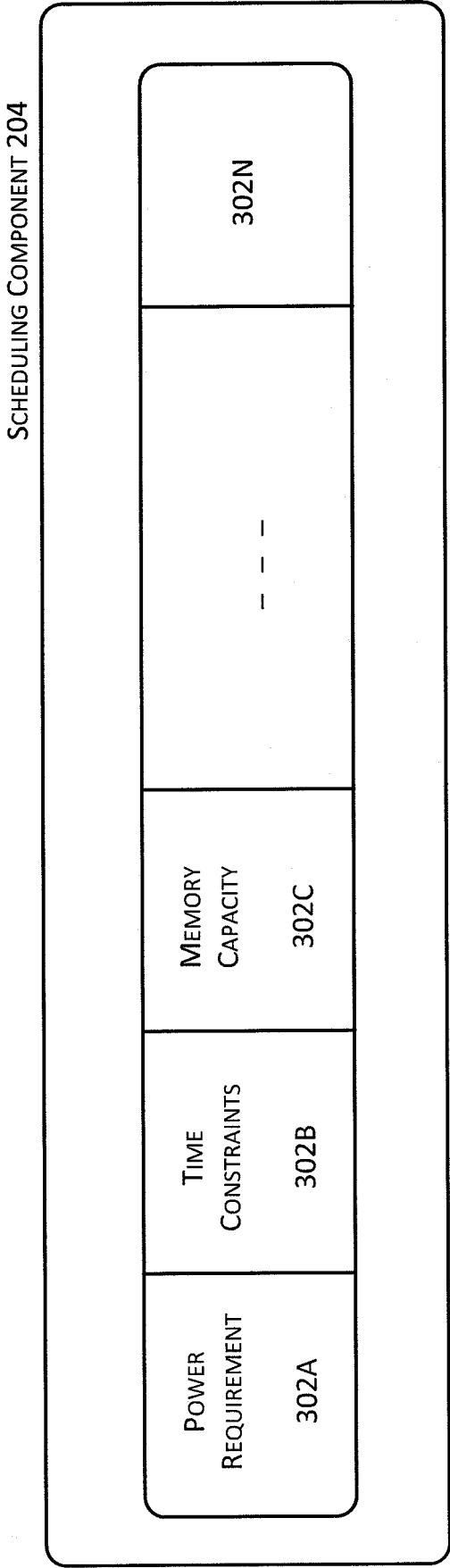


FIG. 3

400

OPERATING SYSTEM 104

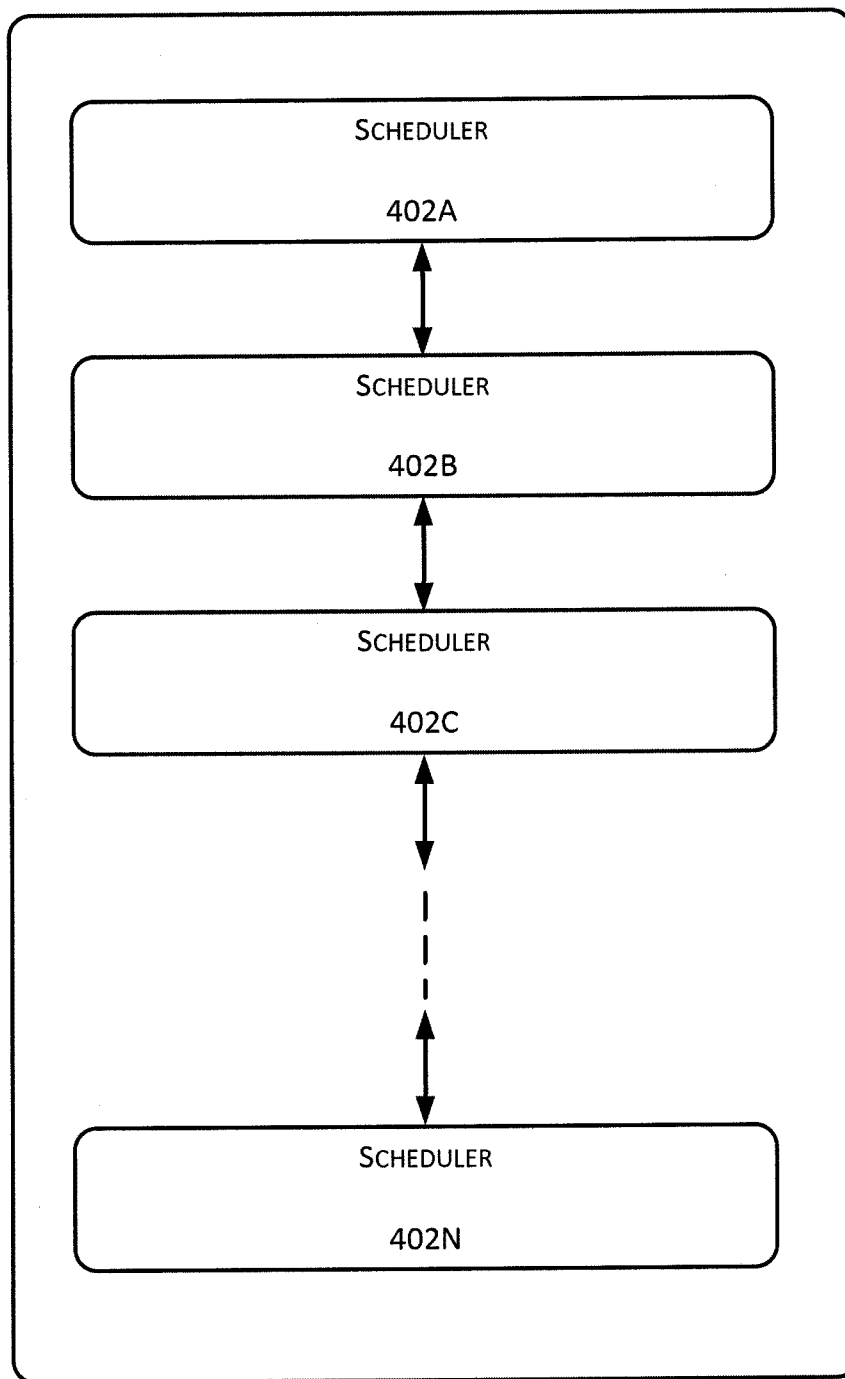


FIG. 4

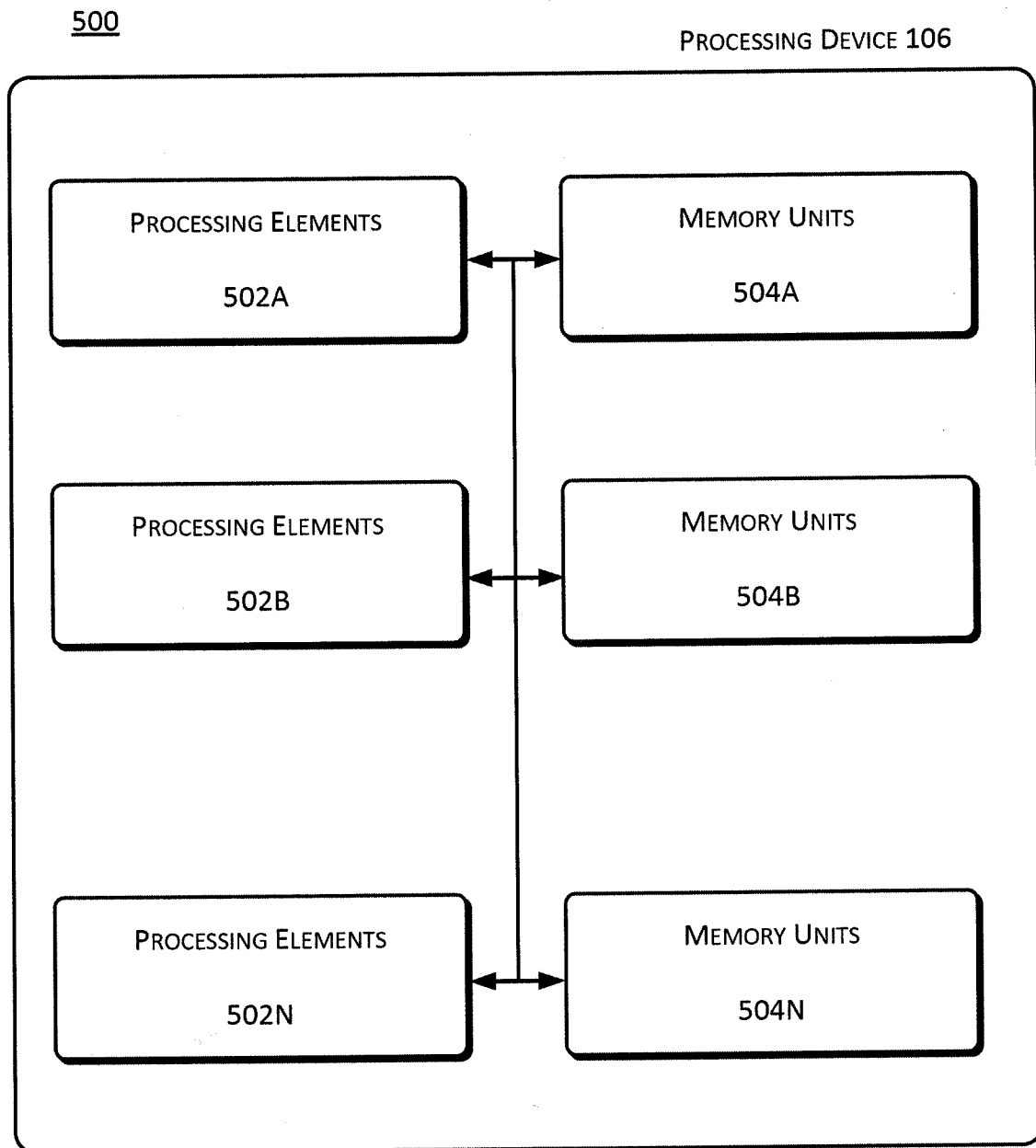
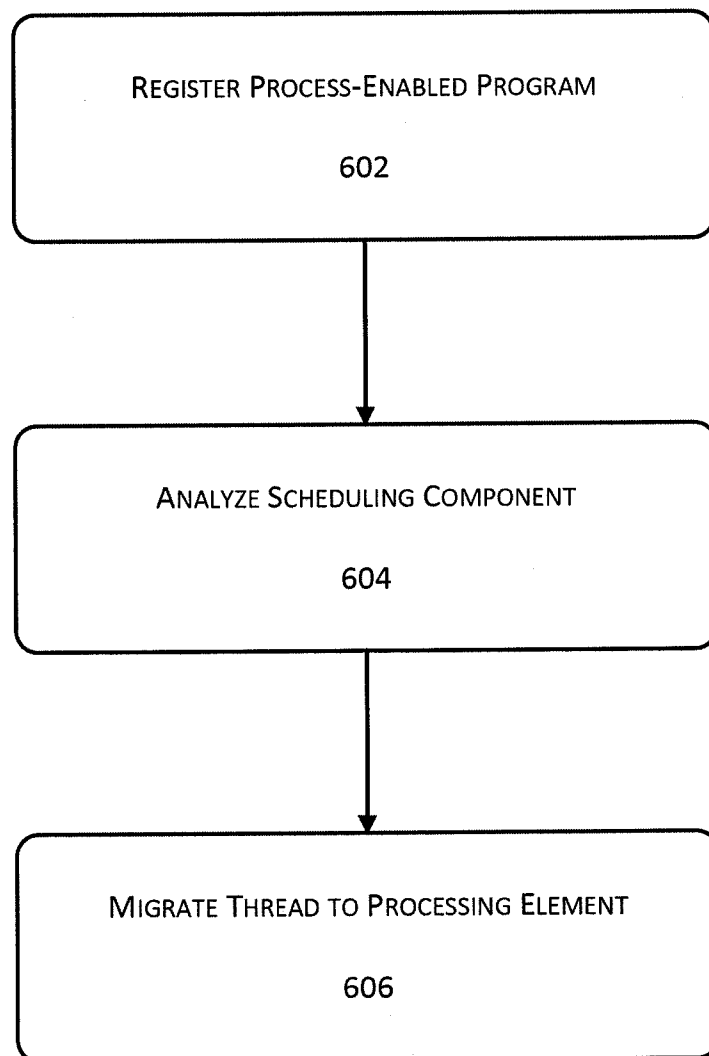
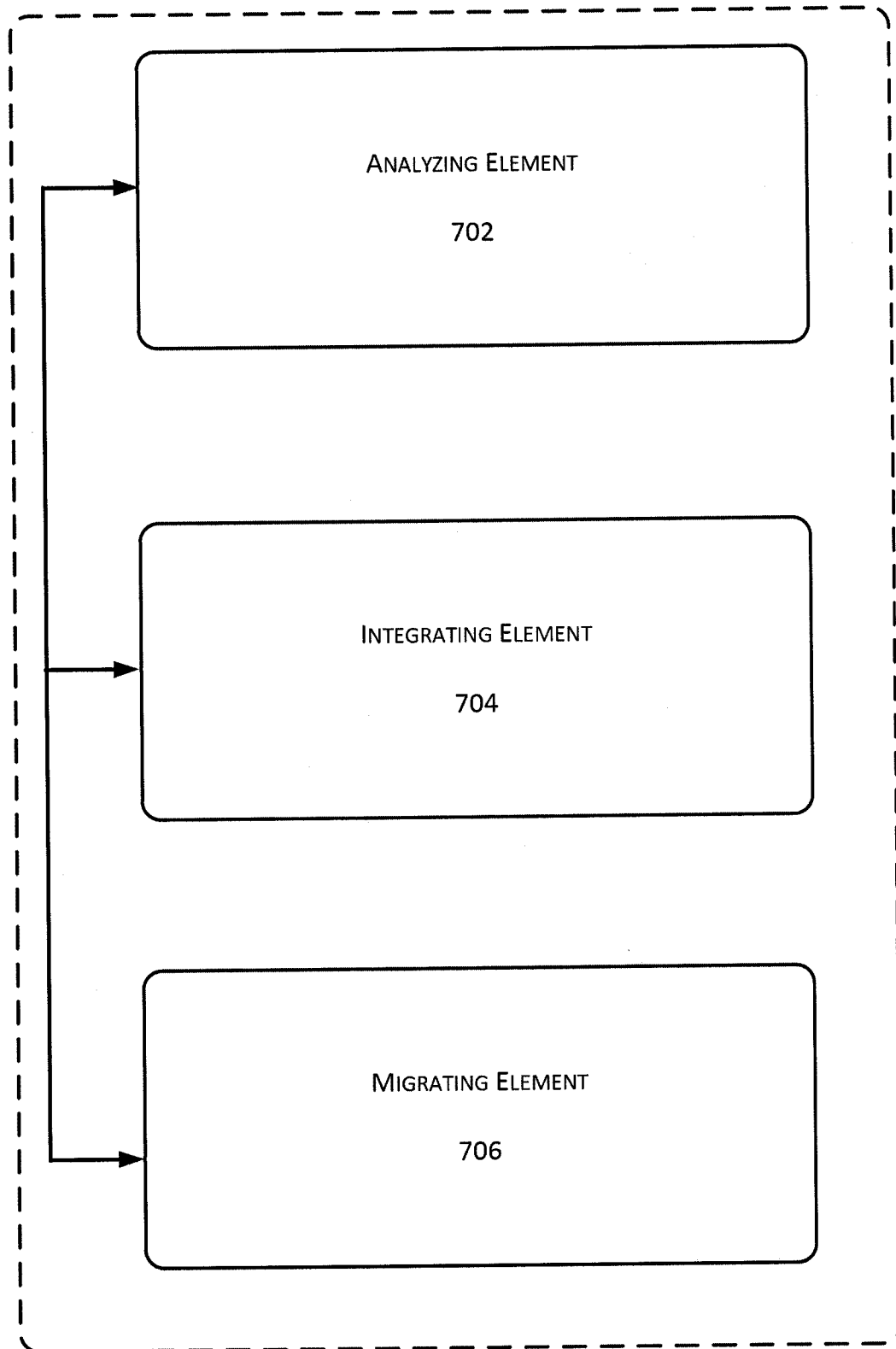


FIG. 5

600**FIG. 6**

700**FIG. 7**

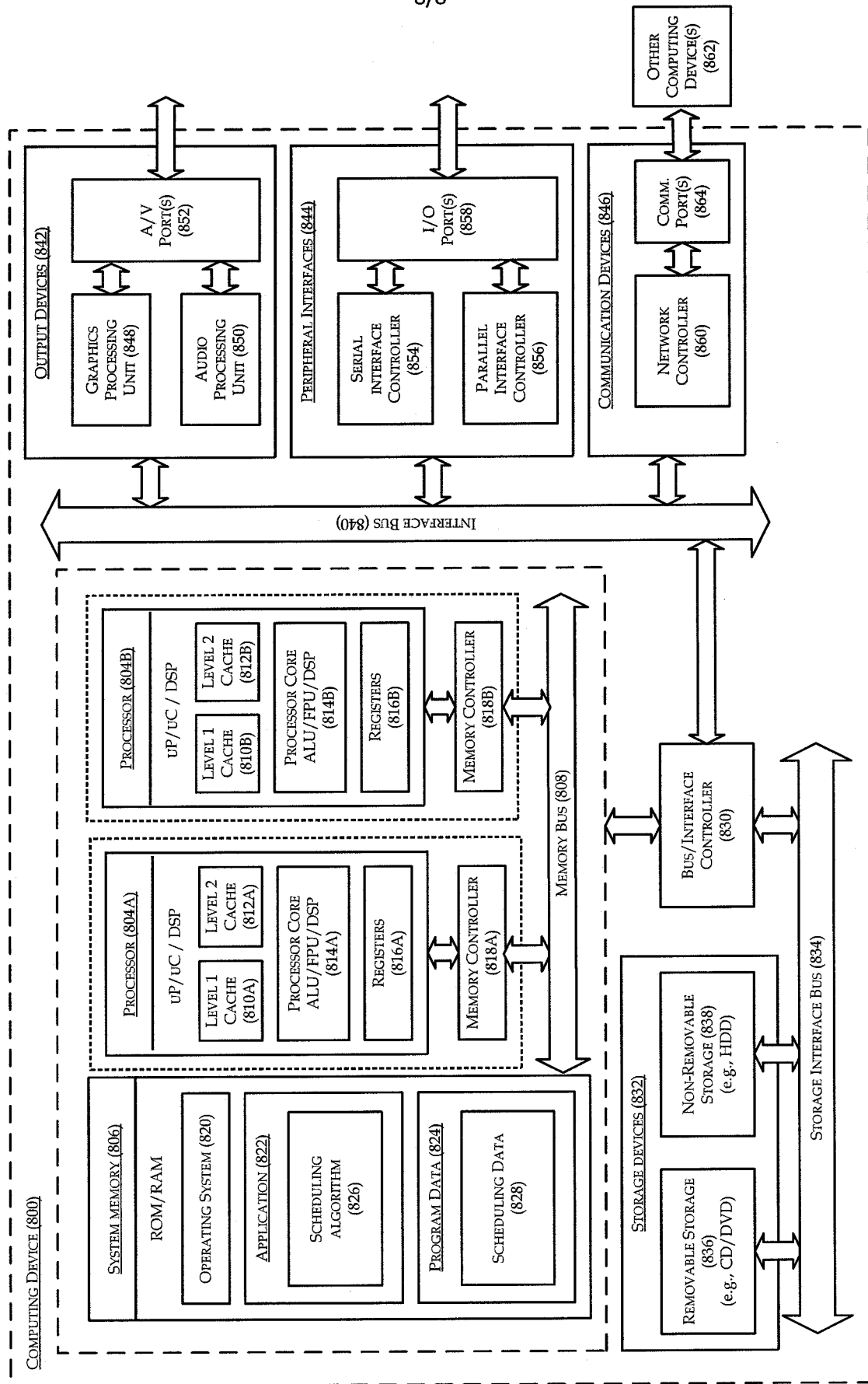


FIG. 8

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/50 (2006.01) G06F 9/40 (2006.01)

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)

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

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

WPI, EPODOC, Google Patents, Patentlens, Patentstorm

Keywords & Phrases: scheduling, executable code, compiled scheduling, resource utilization, scheduling in multi processor environment and like terms

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	Documents are listed in the continuation of Box C	



Further documents are listed in the continuation of Box C



See patent family annex

* "A"	Special categories of cited documents: document defining the general state of the art which is not considered to be of particular relevance	"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
"E"	earlier application or patent but published on or after the international filing date	"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
"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)	"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
"O"	document referring to an oral disclosure, use, exhibition or other means	"&"	document member of the same patent family
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 18 July 2013	Date of mailing of the international search report 18 July 2013
Name and mailing address of the ISA/AU AUSTRALIAN PATENT OFFICE PO BOX 200, WODEN ACT 2606, AUSTRALIA Email address: pct@ipaustalia.gov.au Facsimile No.: +61 2 6283 7999	Authorised officer Rajni Verma AUSTRALIAN PATENT OFFICE (ISO 9001 Quality Certified Service) Telephone No. 0399359635

INTERNATIONAL SEARCH REPORT		International application No.
C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		PCT/US2013/036982
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2007/128168 A1 (INTEL CORPORATION) 15 November 2007 Page 9, paragraph 1, page 6-paragraph 2-page 7, paragraph 1, page 16, paragraph 3, page 7, Figs 1, 3, 4, 5, 9 and associated text, claims 8, 14-16	1-29
Form PCT/ISA/210 (fifth sheet) (July 2009)		

INTERNATIONAL SEARCH REPORT Information on patent family members		International application No. PCT/US2013/036982
This Annex lists known patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.		
Patent Document/s Cited in Search Report		Patent Family Member/s
Publication Number	Publication Date	Publication Number Publication Date
WO 2007/128168 A1	15 Nov 2007	None
End of Annex		
Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001. Form PCT/ISA/210 (Family Annex)(July 2009)		