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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 March 2011 (17.03.2011)

PCT

(10) International Publication Number
WO 2011/031356 A1

(51) International Patent Classification:
G06F 9/46 (2006.01)

(21) International Application Number:
PCT/US2010/037496

(22) International Filing Date:
4 June 2010 (04.06.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/557,971 11 September 2009 (11.09.2009) US

(71) Applicant *(for all designated States except US)*: **EMPIRE TECHNOLOGY DEVELOPMENT LLC** [US/US]; 2711 Centerville Road, Suite 400, Wilmington, DE 19808 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **WOLFE, Andrew** [US/US]; 108 Leewood Court, Los Gatos, CA 95032 (US). **CONTE, Thomas, M.** [US/US]; 2022 Oak Grove Road Ne, Atlanta, GA 30345 (US).

(74) Agents: **HAYDEN, Matthew, P.** et al.; Baker & Hostetler LLP, 312 Walnut Street, Suite 3200, Cincinnati, OH 45202-4074 (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, 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, PE, PG, PH, PL, PT, RO, RS, RU, 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, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: THREAD SHIFT: ALLOCATING THREADS TO CORES

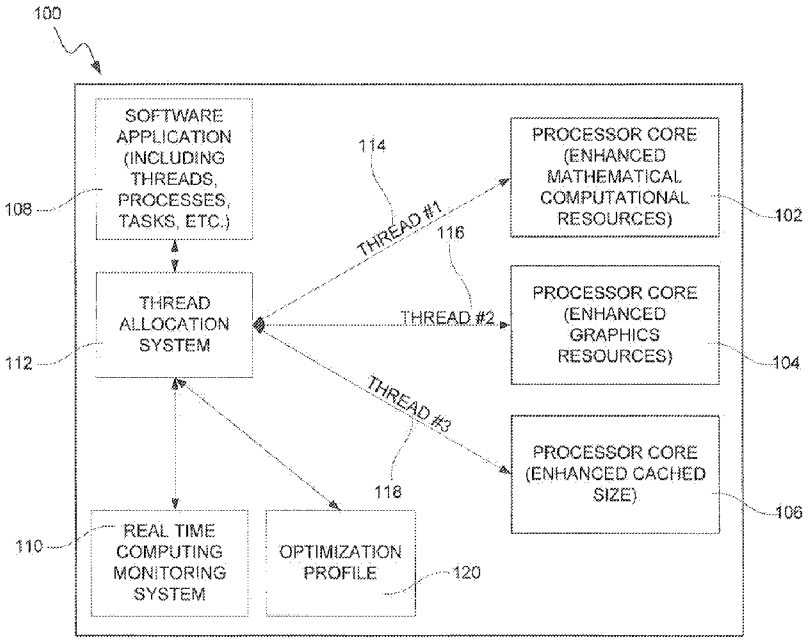

FIG. 1

(57) Abstract: Techniques are generally described for allocating a thread to heterogeneous processor cores. Example techniques may include monitoring real time computing data related to the heterogeneous processor cores processing the thread, allocating the thread to the heterogeneous processor cores based, at least in part, on the reai time computing data, and/or executing the thread by the respective allocated heterogeneous processor core.

WO 2011/031356 A1

Published:

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

PATENT COOPERATION TREATY APPLICATION
Docket No. JBS02-GN007W

Title: THREAD SHIFT: ALLOCATING THREADS TO CORES

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. Patent Application Serial No. 12/557,971 entitled, "THREAD SHIFT: ALLOCATING THREADS TO CORES," which was filed on September 11, 2009, the disclosure of which is hereby incorporated in its entirety by reference.

[0002] This application may be related to co-pending U.S. patent application Ser. No. 12/427,602, entitled "THREAD MAPPING IN MULTI-CORE PROCESSORS," filed April 21, 2009, by Wolfe et al., U.S. patent application Ser. No. 12/557,985, entitled "MAPPING OF COMPUTER THREADS ONTO HETEROGENEOUS RESOURCES," filed September 11, 2009, by Wolfe et al., and/or co-pending U.S. patent application Ser. No. 12/557,864, entitled "CACHE PREFILL ON THREAD MIGRATION," filed September 11, 2009, by Wolfe et al., the entire disclosures of which are incorporated herein by reference.

BACKGROUND

[0003] The present disclosure is related to multi-core computer systems and, more particularly, to efficiently tuning a thread's core allocation based, at least in part, on real time computing data related to the multi-core computer system.

SUMMARY OF THE DISCLOSURE

[0004] The present disclosure generally relates to multi-core computer processing. Specifically, the present disclosure relates to tuning core allocation of threads based on real time computing data.

[0005] A first aspect of the present disclosure generally describes methods of allocating a thread onto heterogeneous processor cores. Such methods may include monitoring real time computing data related to the heterogeneous processor core(s),

and allocating the thread to the heterogeneous processor core(s) based, at least in part, on the monitored real time computing data.

[0006] In some examples of the first aspect, the method may also include executing the thread by the respective allocated heterogeneous processor core. In some examples, monitoring real time computing data may include determining when the thread is falling behind a thread completion target deadline. In some examples, allocating the thread may be based, at least in part, on whether the thread has fallen behind the thread completion target deadline. In some examples, the real time computing data may include an indication that the thread is falling behind a thread completion target deadline.

[0007] In some examples of the first aspect, the real time computing data may include a cycles per instruction rate. In some examples, the cycles per instruction rate may be determined, at least in part, by a ratio of total number of clock cycles per instruction completed over a time interval.

[0008] Some examples of the first aspect may also include comparing the cycles per instruction rate to a target cycles per instruction rate. In such examples, allocating the thread may be based, at least in part, on the comparison of the cycles per instruction rate to the target cycles per instruction rate. In some examples, allocating the thread may include dynamically allocating the thread to the heterogeneous processor core(s).

[0009] In some examples of the first aspect, the method may also include transmitting the monitored real time computing data to a thread allocating software tool, and allocating the thread to a predetermined heterogeneous processor core with the thread allocating tool software based on the monitored real time computing data.

[0010] In some examples, the real time computing data may include checkpoint(s) instructing that the thread is to be allocated from a first heterogeneous processor core to a second heterogeneous processor core at a predetermined time. In such examples, allocating the thread may be based, at least in part, on the checkpoint(s).

[0011] In some examples of the first aspect, the method may also include independently allocating each of a plurality of threads to the heterogeneous processor core(s). In such examples, the thread may include the plurality of threads, each being associated with the real time computing data.

[0012] A second aspect of the present disclosure generally describes methods of allocating a thread from a first heterogeneous processor core to a second heterogeneous processor core. Such methods may include monitoring a total number of clock cycles associated with the first heterogeneous processor core, monitoring a total number of instructions completed by the first heterogeneous processor core, calculating a CPI value based, at least in part, on a ratio of total number of clock cycles per instruction completed over a predetermined time interval, and allocating the thread to second heterogeneous processor core based, at least in part, on the CPI value.

[0013] In some examples of the second aspect, the method may also include, prior to allocating the thread, comparing the CPI value to a target CPI value prior to allocating the thread. In such examples, allocating may be based, at least in part, on the CPI value equaling the target CPI value. Some examples may include comparing the CPI value to a target CPI value prior to allocating the thread. In such examples, allocating may be based, at least in part, on the CPI value exceeding the target CPI value. In some examples, the target CPI value may be defined prior to and/or during the execution of a software application associated with the thread.

[0014] In some examples of the second aspect, the real time computing data may include checkpoint(s) instructing that the thread is to be allocated from the first heterogeneous processor core to the second heterogeneous processor core at a predetermined time. In such examples, allocating may be based, at least in part, on the predetermined time associated with the checkpoint(s).

[0015] A third aspect of the present disclosure generally describes articles such as a storage medium having machine-readable instructions stored thereon. When executed by processing unit(s), such machine-readable instructions may enable a computing platform to monitor real time computing data related to a first heterogeneous

processor core processing a thread, and allocate the thread to a second heterogeneous processor core based, at least in part, on the monitored real time computing data.

[0016] In some examples of the third aspect, the machine-readable instructions may further enable a computing platform to determine if the thread is falling behind a thread completion target deadline. In such examples, the allocating the thread may be based, at least in part, on whether the thread has fallen behind the thread completion target deadline.

[0017] In some examples of the third aspect, the articles may also include a thread allocating software tool configured to receive the real time computing data and allocate the thread to a predetermined heterogeneous processor core based at least in part on the real time computing data. In some examples, the machine-readable instructions may further enable a computing platform to dynamically allocate the thread to a second heterogeneous processor core based, at least in part, on the monitored real time computing data.

[0018] A fourth aspect of the present disclosure generally describes multi-core processors. Such multi-core processors may include a first heterogeneous processor core and a second heterogeneous processor core. The multi-core processor may be configured to monitor real time computing data related to the first heterogeneous processor core processing a thread, allocate the thread to the second heterogeneous processor core based, at least in part, on the monitored real time computing data, and execute the thread by the second heterogeneous processor core after the thread is allocated to the second heterogeneous processor core.

[0019] In some examples, the first processor core may have a first capability and the second processor core may have a second capability that is different from the first capability such that the multi-core processor includes heterogeneous hardware. In some examples, thee first capability and the second capability each correspond to a graphics resource, a mathematical computational resource, an instruction set, an accelerator; an SSE, a cache size and/or a branch predictor.

[0020] 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

[0021] The foregoing and other features of the present disclosure will become more fully apparent from the following description and appended claims, taken in conjunction with the accompanying drawings. Understanding that these drawings depict only several embodiments in accordance with the disclosure and are, therefore, not to be considered limiting of its scope, the disclosure will be described with additional specificity and detail through use of the accompanying drawings.

[0022] In the drawings:

FIG. 1 is a block diagram illustrating an example multi-core system having multiple heterogeneous cores;

FIG. 2 is a flowchart depicting an example method of allocating a thread to at least one of a plurality of heterogeneous processor cores;

FIG. 3 is a flowchart depicting an example method of allocating a thread from a first heterogeneous processor core to a second heterogeneous processor core;

FIG. 4 is a schematic diagram illustrating an example article including a storage medium comprising machine-readable instructions; and

FIG. 5 is a block diagram illustrating an example computing device that may be arranged for thread allocating implementations, all configured in accordance with at least some embodiments of the present disclosure.

DETAILED DESCRIPTION

[0023] In the following detailed description, reference is made to the accompanying drawings, which form a part hereof. In the drawings, similar symbols typically identify

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

[0024] This disclosure is drawn, *inter alia*, to methods, systems, devices and/or apparatus related to multi-core computer systems and, more particularly, to efficient tuning of a thread's core allocation based, at least in part, on real time computing factors related to the multi-core computer system.

[0025] The present disclosure contemplates that some computer systems may include a plurality of processor cores. In a multi-core system with heterogeneous hardware, some cores may have certain hardware capabilities not available to other cores. In some example computer systems, at least one thread (which may be a sequence of instructions and which may execute in parallel with other threads) may be assigned to an appropriate core. Thread allocation may be utilized to associate threads with appropriate cores. In some example computer systems, a thread may be reassigned from one core to another core during execution of the thread. While the present disclosure generally refers to threads, it is within the scope of the disclosure to include the mapping of processes, tasks and the like.

[0026] In an example embodiment, data associated with threads may be used to determine when a thread should be allocated to another core and/or to which core the thread should be allocated. For example, a system may use real time computing data relating to a thread to determine whether the thread is falling behind a target deadline. If the thread is falling behind the target deadline, the thread may be migrated to a faster core, for example.

[0027] FIG. 1 is a block diagram illustrating an example multi-core system having

multiple heterogeneous core arranged in accordance with at least some embodiments of the present disclosure. An example multi-core system 100 may include a plurality of processor cores 102, 104, and/or 106. In an example embodiment, a multi-core system 100 may include one or more cores 102, 104, and/or 106 each core having different capabilities. In other words, a multi-core system 100 may include heterogeneous hardware. For example, one core 102 may include enhanced graphics resources, another core 104 may include enhanced mathematical computational resources and another core 106 may include a large capacity cache.

[0028] As depicted in Fig. 1, a software application 108 may include several threads (or processes and/or tasks). To efficiently allocate the threads to an appropriate core based on real time computing factors, the thread allocation methods described herein may be implemented by a thread allocation system 112 operably connected to the software application 108. In some embodiments, a real time computing monitoring system 110 may also be operably coupled to the thread allocation system 112 to assist in efficiently allocating threads 114, 116 and/or 118 to one or more cores 102, 104 and/or 106, which will be described in more detail below.

[0029] In the example embodiment of Fig. 1, the thread allocation system 112 may allocate threads to an appropriate core. Specifically, the thread allocation system 112 may map a first thread (thread #1) 114 to core 102, may map a second thread (thread #2) 116 to core 102, and may map a third thread (thread #3) 118 to core 106.

[0030] In some example embodiments, a thread which may initially benefit from enhanced graphics capabilities may be initially executed on a core having enhanced graphics resources. Based at least in part on the expectation that the thread may later benefit from enhanced mathematical computational capabilities, data pertaining to the thread may be transmitted to a core having enhanced mathematical computational to complete its execution.

[0031] In some example embodiments, cores may include different instruction sets; different accelerators (e.g., DSPs (digital signal processors) and/or different SSEs (streaming SIMD (single instruction, multiple data) extensions)); larger and/or smaller

caches (such as L1 and L2 caches); different branch predictors (the parts of a processor that determine whether a conditional branch in the instruction flow of a program is likely to be taken or not); and/or the like. Based at least in part on these and/or other differences between cores, different cores may provide different capabilities for certain tasks.

[0032] In some example embodiments, threads may be allocated (utilizing the thread allocation system 112, for example)to individual cores based at least in part on the hardware capabilities of the cores. For example, a thread associated with a large L1 cache (memory) demand may be allocated to a core including large L1 cache hardware. Similarly, a thread associated with a large SSE (instruction set) demand may be allocated to a core including native SSE hardware implementation. These examples are non-limiting, and it will be understood that threads may be allocated based at least in part on any hardware characteristic, instruction set, and/or other characteristic of a core and/or a thread. Further, the present disclosure contemplates that, if the threads are not allocates to a core based on hardware capabilities, the thread may instead be processed using software emulation, which may increase processing time for that thread.

[0033] In some example embodiments, determining whether to allocate a thread to a different core and/or when to perform such an allocation may include evaluating of at least a portion of an execution profile that may include data related to a prior execution of the thread. In some example embodiments, the execution profile may be generated using a freeze-dried ghost page execution profile generation method as disclosed in U.S. Patent Application Publication No. 2007/0050605, which is incorporated by reference. This method may use a shadow processor, or in some embodiments a shadow core, to simulate the execution of at least a portion of a thread in advance and to generate performance statistics and measurements related to this execution.

[0034] In some example embodiments, processors and/or caches may be configured to collect information as a program executes. For example, such information may include which cache lines the program references. In some example

embodiments, data about cache usage may be evaluated to determine which threads should be replaced (e.g., by counting the number of lines of thread process remaining). In an example embodiment, a performance counter may be configured to track line evictions of running threads and/or may use that information to decide which tasks may be flushed out to begin a higher priority task. A performance counter may also be configured to track the line evictions since a task has started.

[0035] An example multi-core system may include a performance counter adapted for use in accordance with the present disclosure. Various cores may be operatively coupled to a performance counter. A performance counter may be configured to store a count indicating the number of hardware-related activities within a computer system, for example. Thread allocation (from one core to another core, for example) may be at least partially determined using data collected by the performance counter.

[0036] Some example embodiments may consider the size of a cache footprint for a particular task. In some example embodiments, Bloom filters may be used to characterize how big the cache footprint is for a thread. An example Bloom filter may be a space-efficient probabilistic data structure that may be used to test whether an element is a member of a set. When using some example Bloom filters, false positives are possible, but false negatives are not. In some example Bloom filters, elements may be added to the set, but may not be removed (though this can be addressed with a counting filter). In some example Bloom filters, the more elements that are added to the set, the larger the probability of false positives. An empty Bloom filter may be a bit array of m bits, all set to 0. In addition, k different hash functions may be defined, each of which may map or hash some set element to one of the m array positions with a uniform random distribution. To add an element, the element may be fed to each of the k hash functions to get k array positions. The bits at these positions may be set to 1. To query for an element (e.g., to test whether it is in the set), the element may be fed to each of the k hash functions to get k array positions. In some example Bloom filters, if the bit at any of these positions is 0, then the element is not in the set; if the element was in the set, then all of the bits at the k array positions would have been set to 1 when it was

inserted. In some example Bloom filters, if all of the bits at the k array positions are 1, then either the element is in the set, or the bits were set to 1 during the insertion of other elements.

[0037] In some example embodiments, a Bloom filter may be used to track which portions of the cache are being used by the current thread. For example, the filter may be emptied when the thread is first scheduled onto the core. Each time a cache line is used by the thread, it may be added to the filter set. A sequence of queries may be used to estimate the thread footprint in order to evaluate the cost of cache data migration. In some example embodiments, a simple population count of the number of "1" bits in the filter may be used to estimate the cache footprint of the thread. In some example embodiments, counting Bloom filters may be used. In a counting Bloom filter, each filter element may be a counter which may be incremented when a cache line is used by the thread and may be decremented when the cache line is invalidated.

[0038] In some example embodiments, the cache data for a thread migration may be pre-fetched. The prefetching may be performed by a hardware prefetcher as is known in the art. One such prefetcher is disclosed in U.S. Patent No. 7,318,125, which is incorporated by reference. That is, when the system is preparing to transmit a thread to a new core, references from the current core may be sent to the new core to prepare for the migration. Thus, the new core may be "warmed up" in preparation for the migration. In some embodiments, substantially all of the data relating to the thread to be migrated may be pre-fetched by the new core. In some other example embodiments, a portion of the data relating to the thread to be migrated may be pre-fetched by the new core. For example, the cache misses, hits, and/or line evictions may be pre-fetched. In some example embodiments, rather than caching the data in the new core (and thereby filling up the new core with data that may ultimately not be required), the data may be pre-fetched to a side/stream buffer, for example.

[0039] As used herein, "cache hit" may refer to a successful attempt to reference data that has been cached, as well as the corresponding data. As used herein, "cache miss" may refer to an attempt to reference data that has not been found in the cache, as

well as the corresponding data. As used herein, "line eviction" may refer to removing a cached line from the cache, such as to make space for different data in the cache. Line eviction may also include a write-back operation whereby modified data in the cache is written to main memory or a higher cache level prior to being removed from the cache.

[0040] FIG. 2 is a flowchart depicting an example method 200 of allocating a thread to at least one of a plurality of heterogeneous processor cores that is arranged in accordance with at least some embodiments of the present disclosure. Example methods 200 may include one or more of processing operations 202 and/or 204. Processing may begin at operation 202, which may include monitoring real time computing data related to one or more heterogeneous processor cores, each adapted for processing the thread. Processing may flow from operation 202 to operation 204. Operation 204 may include allocating the thread to one or more of the heterogeneous processor cores based, at least in part, on the monitored real time computing data.

[0041] In some embodiments, the monitoring operation 202 may include determining if the thread is falling behind a thread completion target deadline, and the allocating operation 204 may be based, at least in part, on whether the thread has fallen behind the thread completion target deadline. In some other embodiments, the real time computing data may include a determination that the thread is falling behind a thread completion target deadline.

[0042] In some embodiments, the monitoring operation 202 (which may be performed by separate hardware on another processor core) may occur concurrently with thread processing or may occur intermittently (i.e., occurring at checkpoints or scheduling interrupts). The thread may either be executing or suspended (i.e., paused) at any time. In some examples, the thread may have a deadline for completion and may likely be executing on one core until such time as it is reallocated to another. In some embodiments, real-time data may be evaluated concurrently. In some examples, the thread may be suspended, the real-time data may be evaluated, and the thread may be reallocated to the same core or a different core.

[0043] In an example embodiment, the real time computing data may include a

cycles per instruction rate, which may be determined, at least in part, as a ratio of total number of clock cycles per instruction completed over a predetermined time interval. In another example embodiment, the cycles per instruction rate may be compared to a target cycles per instruction rate, and the allocating operation 204 may be based, at least in part, on the comparison of the cycles per instruction rate to the target cycles per instruction rate. In still another example embodiment, the allocating operation 2045 may occur dynamically.

[0044] In yet another embodiment, the real time computing data may include at least one checkpoint instructing that the thread is to be allocated from one heterogeneous processor core to another heterogeneous processor core at a predetermined time, and the allocating operation 204 may be based, at least in part, on the at least one checkpoint.

[0045] In another embodiment, the method 200 may further include transmitting the real time computing data to a thread allocating software tool. The thread allocating software tool may be configured to allocate the thread to a predetermined heterogeneous processor core. In another embodiment, the thread may be a plurality of threads, where each of a plurality of threads may be associated with the real time computing data, and where each of the threads may be independently allocated to the heterogeneous processor cores.

[0046] FIG. 3 is a flowchart depicting an example method 300 of allocating a thread from a first heterogeneous processor core to a second heterogeneous processor core that is arranged in accordance with at least some embodiments of the present disclosure. The example methods 300 may include one or more of processing operations 302, 304 and/or 306.

[0047] Processing may begin at operation 302, which may include monitoring one or more of a total number of clock cycles associated with the first heterogeneous processor core processing the thread and/or a total number of instructions completed by the first heterogeneous processor core. Processing may continue from operation 302 to operation 304, which may include calculating a cycles per instruction (CPI) value based,

at least in part, on a ratio of a total number of clock cycles per instruction completed over a time interval. Processing may then continue from operation 304 to operation 306, which may include allocating the thread to second heterogeneous processor core based, at least in part, on the CPI value.

[0048] In an example embodiment, the method may further include comparing the CPI value to a target CPI value prior to the allocating operation 308, where the allocating operation 308 may be based, at least in part, on the CPI value approximately equaling the target CPI value. In another embodiment, the method may further include comparing the CPI value to a target CPI value prior to the allocating operation 308, where the allocating operation 308 may be based, at least in part, on the CPI value exceeding the target CPI value.

[0049] In another embodiment, the target CPI value may be defined prior to or during the execution of a software application associated with the thread.

[0050] In yet another embodiment, the real time computing data may include one or more checkpoints configured to instruct that the thread is to be allocated from one heterogeneous processor core to another heterogeneous processor core at a predetermined time. In this manner, the allocating operation 308 may be based, at least in part, on the predetermined time associated with the at least one checkpoint.

[0051] FIG. 4 is a schematic diagram of an example article including a storage medium 400 comprising machine-readable instructions arranged in accordance with at least some embodiments of the present disclosure. When executed by one or more processing units, the machine readable instructions may operatively enable a computing platform to monitor real time computing data related to a first heterogeneous processor core processing a thread (operation 402); allocate the thread to a second heterogeneous processor core based, at least in part, on the real time computing data (operation 404).

[0052] In some embodiments, the machine-readable instructions may be further configured to operatively enable a computing platform to determine if the thread is falling behind a thread completion target deadline. In this manner, the allocating

operation 404 may be based, at least in part, on whether the thread has been fallen behind the thread completion target deadline.

[0053] In another embodiment, the article may further include a thread allocating software tool configured to receive the real time computing data. The thread allocating software tool being may be configured to allocate the thread to a predetermined heterogeneous processor core.

[0054] In some embodiments, the machine-readable instructions may be further configured to operatively enable a computing platform to dynamically allocate the thread to another heterogeneous processor core based, at least in part, on the monitored real time computing data.

[0055] In some embodiment, a multi-core processor may include a first heterogeneous processor core and a second processor core. Such a multi-core processor may be configured to monitor real time computing data related to the first heterogeneous processor core processing a thread, allocate the thread to the second heterogeneous processor core based, at least in part, on the monitored real time computing data, and execute the thread by the second heterogeneous processor core after the thread is allocated to the second heterogeneous processor core.

[0056] FIG. 5 is a block diagram illustrating an example computing device 900 that is arranged for thread allocation in accordance with at least some embodiments of the present disclosure. In a very basic configuration 901, computing device 900 may typically include one or more processors 910 and system memory 920. A memory bus 930 can be used for communicating between the processor 910 and the system memory 920.

[0057] Depending on the desired configuration, processor 910 can be of any type including but not limited to a microprocessor (μP), a microcontroller (μC), a digital signal processor (DSP), or any combination thereof. Processor 910 can include one more levels of caching, such as a level one cache 911 and a level two cache 912, a processor core 913, and registers 914. The processor core 913 can include an arithmetic logic unit (ALU), a floating point unit (FPU), a digital signal processing core

(DSP Core), or any combination thereof. A memory controller 915 can also be used with the processor 910, or in some implementations the memory controller 915 can be an internal part of the processor 910.

[0058] Depending on the desired configuration, the system memory 920 can 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 920 typically includes an operating system 921, one or more applications 922, and program data 924. Application 922 may include a thread allocating algorithm 923 that may be arranged to monitor real time computing data and allocate a thread to at least one heterogeneous hardware component based, at least in part, on the monitored real time computing data. Program Data 924 may include thread allocation data 925 that may be useful for allocating the thread to an appropriate heterogeneous hardware component. In some embodiments, application 922 can be arranged to operate with program data 924 on an operating system 921 such that a thread may be efficiently allocated in accordance with the various methods described herein. This described basic configuration is illustrated in FIG. 5 by those components within dashed line 901.

[0059] Computing device 900 can have additional features or functionality, and additional interfaces to facilitate communications between the basic configuration 901 and any required devices and interfaces. For example, a bus/interface controller 940 can be used to facilitate communications between the basic configuration 901 and one or more data storage devices 950 via a storage interface bus 941. The data storage devices 950 can be removable storage devices 951, non-removable storage devices 952, 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 can 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.

[0060] System memory 920, removable storage 951 and non-removable storage 952 are all 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 can be used to store the desired information and which can be accessed by computing device 900. Any such computer storage media can be part of device 900.

[0061] Computing device 900 can also include an interface bus 942 for facilitating communication from various interface devices (e.g., output interfaces, peripheral interfaces, and communication interfaces) to the basic configuration 901 via the bus/interface controller 940. Example output devices 960 include a graphics processing unit 961 and an audio processing unit 962, which can be configured to communicate to various external devices such as a display or speakers via one or more A/V ports 963. Example peripheral interfaces 970 include a serial interface controller 971 or a parallel interface controller 972, which can 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 973. An example communication device 980 includes a network controller 981, which can be arranged to facilitate communications with one or more other computing devices 990 over a network communication via one or more communication ports 982. The communication connection is 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 includes any information delivery media. A "modulated data signal" can 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 can include wired media such as a wired network or direct-wired connection, and wireless media

such as acoustic, radio frequency (RF), infrared (IR) and other wireless media. The term computer readable media as used herein can include both storage media and communication media.

[0062] Computing device 900 can 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 900 can also be implemented as a personal computer including both laptop computer and non-laptop computer configurations.

[0063] The herein described subject matter sometimes illustrates different components contained within, or connected with, different other components. It is to be understood that such depicted architectures are merely examples, and that in fact many other architectures may be implemented which achieve the same functionality. In a conceptual sense, any arrangement of components to achieve the same functionality is effectively "associated" such that the desired functionality is achieved. Hence, any two components herein combined to achieve a particular functionality may be seen as "associated with" each other such that the desired functionality is achieved, irrespective of architectures or intermedial components. Likewise, any two components so associated may also be viewed as being "operably connected", or "operably coupled", to each other to achieve the desired functionality, and any two components capable of being so associated may also be viewed as being "operably couplable", to each other to achieve the desired functionality. Specific examples of operably couplable include but are not limited to physically mateable and/or physically interacting components and/or wirelessly interactable and/or wirelessly interacting components and/or logically interacting and/or logically interactable components.

[0064] With respect to the use of substantially any plural and/or singular terms herein, those having skill in the art may translate from the plural to the singular and/or from the singular to the plural as is appropriate to the context and/or application. The

various singular/plural permutations may be expressly set forth herein for sake of clarity.

[0065] It will be understood by those within the art that, in general, terms used herein, and especially in the appended claims (e.g., bodies of the appended claims) are generally intended as "open" terms (e.g., the term "including" should be interpreted as "including but not limited to," the term "having" should be interpreted as "having at least," the term "includes" should be interpreted as "includes but is not limited to," etc.). It will be further understood by those within the art that if a specific number of an introduced claim recitation is intended, such an intent will be explicitly recited in the claim, and in the absence of such recitation no such intent is present. For example, as an aid to understanding, the following appended claims may contain usage of the introductory phrases "at least one" and "one or more" to introduce claim recitations. However, the use of such phrases should not be construed to imply that the introduction of a claim recitation by the indefinite articles "a" or "an" limits any particular claim containing such introduced claim recitation to inventions containing only one such recitation, even when the same claim includes the introductory phrases "one or more" or "at least one" and indefinite articles such as "a" or "an" (e.g., "a" and/or "an" should typically be interpreted to mean "at least one" or "one or more"); the same holds true for the use of definite articles used to introduce claim recitations. In addition, even if a specific number of an introduced claim recitation is explicitly recited, those skilled in the art will recognize that such recitation should typically be interpreted to mean at least the recited number (e.g., the bare recitation of "two recitations," without other modifiers, typically means at least two recitations, or two or more recitations). Furthermore, in those instances where a convention analogous to "at least one of A, B, and C, etc." is used, in general such a construction is intended in the sense one having skill in the art would understand the convention (e.g., "a system having at least one of A, B, and C" would include but not be limited to systems that have A alone, B alone, C alone, A and B together, A and C together, B and C together, and/or A, B, and C together, etc.). In those instances where a convention analogous to "at least one of A, B, or C, etc." is used, in general such a construction is intended in the sense one having skill in the art would understand the convention (e.g., "a system having at least one of A, B, or C"

would include but not be limited to systems that have A alone, B alone, C alone, A and B together, A and C together, B and C together, and/or A, B, and C together, etc.). It will be further understood by those within the art that virtually any disjunctive word and/or phrase presenting two or more alternative terms, whether in the description, claims, or drawings, should be understood to contemplate the possibilities of including one of the terms, either of the terms, or both terms. For example, the phrase "A or B" will be understood to include the possibilities of "A" or "B" or "A and B."

[0066] While various aspects and embodiments have been disclosed herein, other aspects and embodiments will be apparent to those skilled in the art. The various aspects and embodiments disclosed herein are for purposes of illustration and are not intended to be limiting, with the true scope and spirit being indicated by the following claims.

PATENT COOPERATION TREATY APPLICATION
Docket No. JBS02-GN007W

What is claimed is:

1. A method of allocating a thread to one or more heterogeneous processor cores, the method comprising:

monitoring real time computing data related to the one or more heterogeneous processor cores; and

allocating the thread to one or more of the heterogeneous processor cores based, at least in part, on the monitored real time computing data.

2. The method of claim 1, further comprising:

executing the thread by the respective allocated heterogeneous processor core.

3. The method of claim 1, wherein monitoring real time computing data comprises determining when the thread is falling behind a thread completion target deadline; and

wherein allocating the thread is based, at least in part, on whether the thread has fallen behind the thread completion target deadline.

4. The method of claim 1, wherein the real time computing data comprises an indication that the thread is falling behind a thread completion target deadline.

5. The method of claim 1, wherein the real time computing data comprises a cycles per instruction rate.

6. The method of claim 5, further comprising:

determining the cycles per instruction rate is determined, at least in part, by a ratio of total number of clock cycles per instruction completed over a time interval.

7. The method of claim 6, further comprising:

comparing the cycles per instruction rate to a target cycles per instruction rate; and

wherein allocating the thread is based, at least in part, on the comparison of the

cycles per instruction rate to the target cycles per instruction rate.

8. The method of claim 1, wherein allocating the thread comprises dynamically allocating the thread to one or more of the heterogeneous processor cores.

9. The method of claim 1, further comprising:

transmitting the monitored real time computing data to a thread allocating software tool, and allocating the thread to a predetermined one of the heterogeneous processor cores with the thread allocating tool software based on the monitored real time computing data.

10. The method of claim 1, the real time computing data comprising one or more checkpoints instructing that the thread is to be allocated from a first heterogeneous processor core to a second heterogeneous processor core at a predetermined time, wherein allocating the thread is based, at least in part, on the one or more checkpoints.

11. The method of claim 1, further comprising:

independently allocating each of a plurality of threads to one or more of the heterogeneous processor cores, wherein the thread includes the plurality of threads, each of the plurality of threads being associated with the real time computing data.

12. A method of allocating a thread from a first heterogeneous processor core to a second heterogeneous processor core, the method comprising:

monitoring a total number of clock cycles associated with the first heterogeneous processor core;

monitoring a total number of instructions completed by the first heterogeneous processor core;

calculating a CPI value based, at least in part, on the a ratio of total number of clock cycles per instruction completed over a predetermined time interval;

allocating the thread to second heterogeneous processor core based, at least in

part, on the CPI value.

13. The method of claim 12, the method further comprising:

prior to allocating the thread, comparing the CPI value to a target CPI value;

wherein the allocating operation is based, at least in part, on the CPI value equaling the target CPI value.

14. The method of claim 12, the method further comprising:

prior to allocating the thread, comparing the CPI value to a target CPI value;

wherein the allocating operation is based, at least in part, on the CPI value exceeding the target CPI value.

15. The method of claim 12, wherein the target CPI value is defined prior to the execution of a software application associated with the thread.

16. The method of claim 12, wherein the target CPI value is defined during the execution of a software application associated with the thread.

17. The method of claim 12,

wherein the real time computing data includes at least one checkpoint instructing that the thread is to be allocated from the first heterogeneous processor core to the second heterogeneous processor core at a predetermined time; and

wherein the allocating operation is based, at least in part, on the predetermined time associated with the at least one checkpoint.

18. An article comprising:

a storage medium comprising machine-readable instructions stored thereon, which, when executed by one or more processing units, operatively enable a computing platform to:

monitor real time computing data related to a first heterogeneous

processor core processing a thread; and

allocate the thread to a second heterogeneous processor core based, at least in part, on the monitored real time computing data.

19. The article of claim 18,

wherein the machine-readable instructions further operatively enable a computing platform to determine if the thread is falling behind a thread completion target deadline; and

wherein allocate the thread is based, at least in part, on whether the thread has fallen behind the thread completion target deadline.

20. The article of claim 18, further comprising:

a thread allocating software tool configured to receive the real time computing data, the thread allocating software tool being configured to allocate the thread to a predetermined one of the heterogeneous processor cores based at least in part on the real time computing data.

21. The article of claim 18, wherein the machine-readable instructions are further operative to enable a computing platform to dynamically allocate the thread to a second heterogeneous processor core based, at least in part, on the monitored real time computing data.

22. A multi-core processor, comprising:

a first heterogeneous processor core; and

a second heterogeneous processor core;

wherein the multi-core processor is configured to monitor real time computing data related to the first heterogeneous processor core processing a thread, allocate the thread to the second heterogeneous processor core based, at least in part, on the monitored real time computing data, and execute the thread by the second heterogeneous processor core after the thread is allocated to the second

heterogeneous processor core.

23. The multi-core processor of claim 22, wherein the first heterogeneous processor core has a first capability and the second heterogeneous processor core has a second capability that is different from the first capability such that the multi-core processor comprises heterogeneous hardware.

24. The multi-core processor of claim 23, wherein each of the first capability and the second capability corresponds to at least one of: a graphics resource, a mathematical computational resource, an instruction set, an accelerator; an SSE, a cache size and/or a branch predictor.

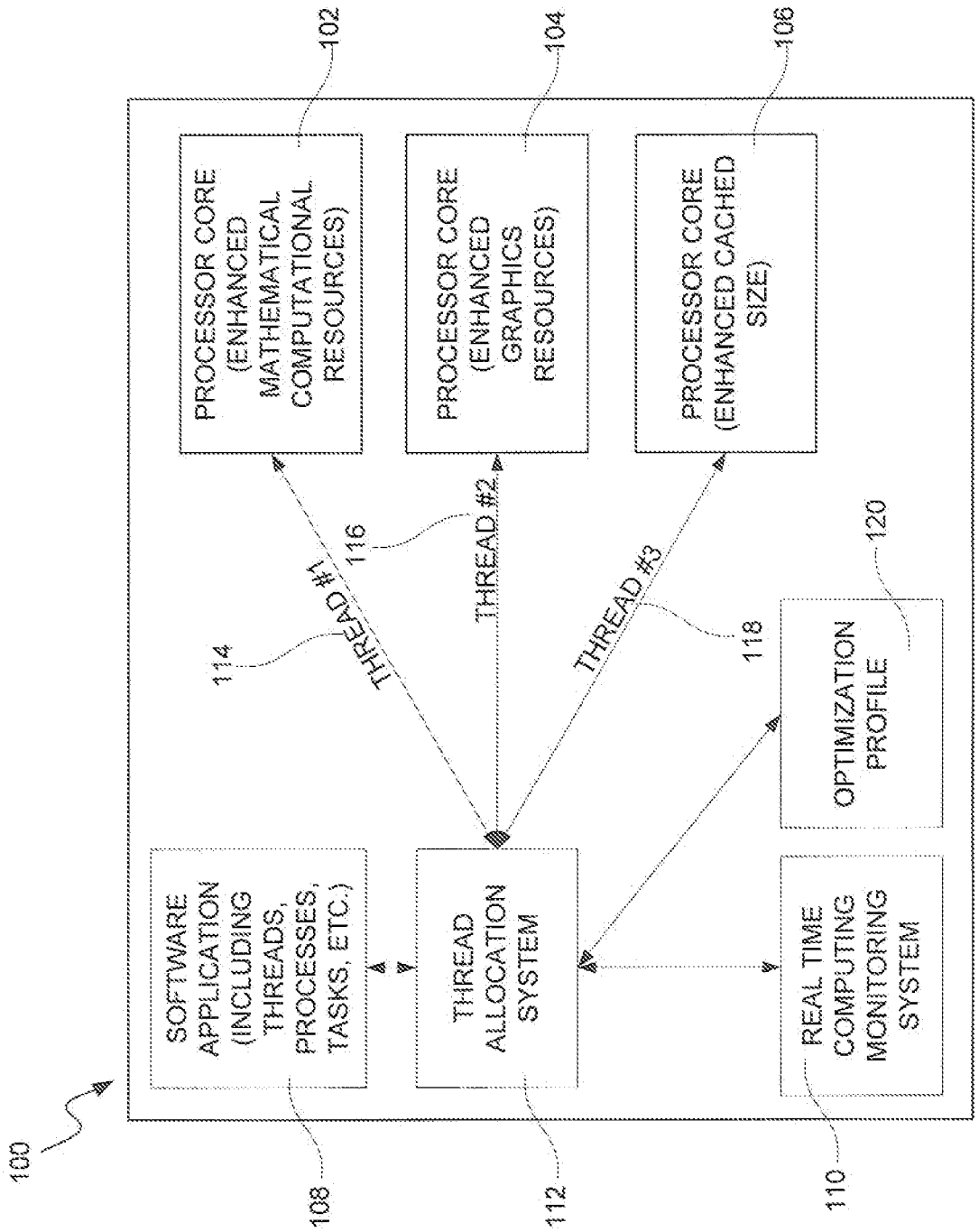
100
PROCESSOR CORE (ENHANCED MATHEMATICAL COMPUTATIONAL RESOURCES)
102
PROCESSOR CORE (ENHANCED GRAPHICS RESOURCES)
104
PROCESSOR CORE (ENHANCED CACHED SIZE)
106
SOFTWARE APPLICATION (INCLUDING THREADS, PROCESSES, TASKS, ETC.)
108
THREAD ALLOCATION SYSTEM
112
REAL TIME COMPUTING MONITORING SYSTEM
110
OPTIMIZATION PROFILE
120
THREAD #1
114
THREAD #2
116
THREAD #3
118

FIG. 1

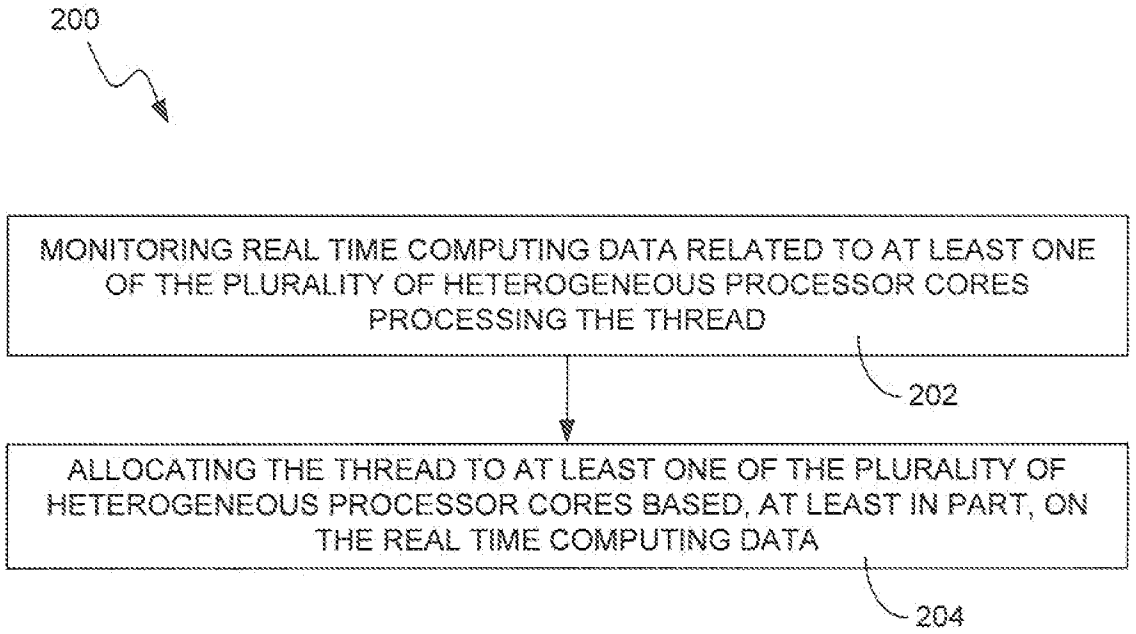
200
MONITORING REAL TIME COMPUTING DATA RELATED TO AT LEAST ONE OF THE PLURALITY OF HETEROGENEOUS PROCESSOR CORES PROCESSING THE THREAD
202
ALLOCATING THE THREAD TO AT LEAST ONE OF THE PLURALITY OF HETEROGENEOUS PROCESSOR CORES BASED, AT LEAST IN PART, ON THE REAL TIME COMPUTING DATA
204

FIG. 2

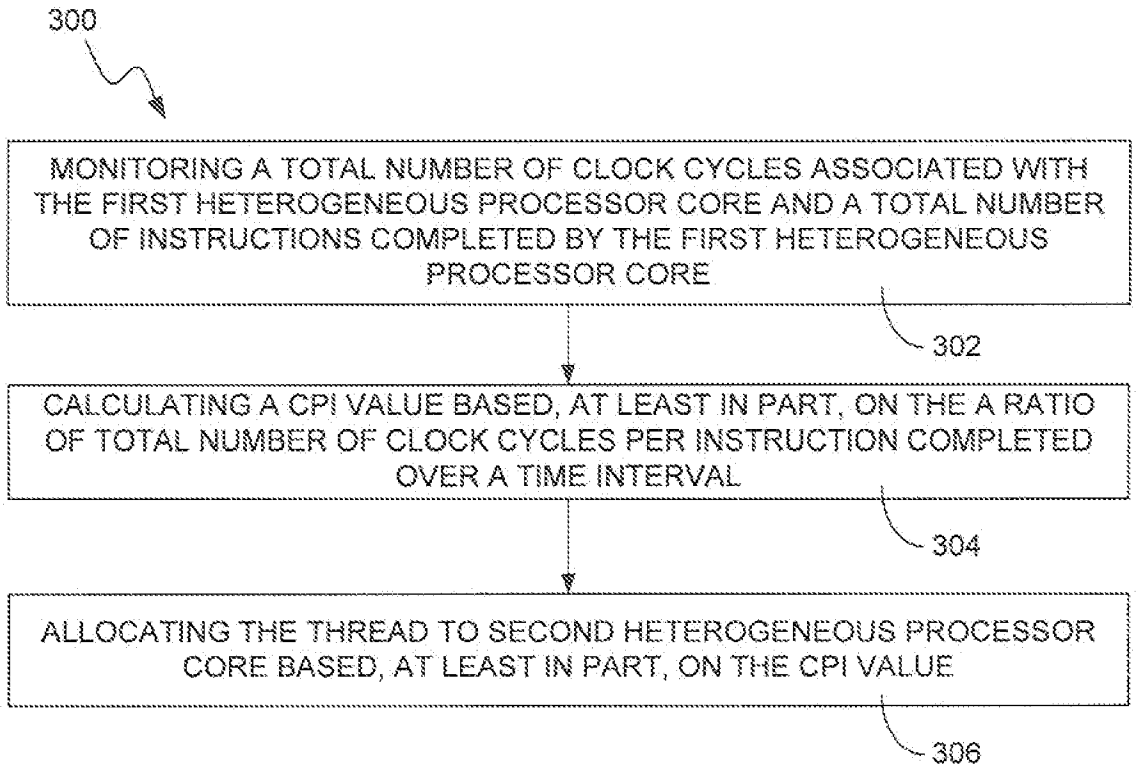
300
MONITORING A TOTAL NUMBER OF CLOCK CYCLES ASSOCIATED WITH THE FIRST HETEROGENEOUS PROCESSOR CORE AND A TOTAL NUMBER OF INSTRUCTIONS COMPLETED BY THE FIRST HETEROGENEOUS PROCESSOR CORE
302
CALCULATING A CPI VALUE BASED, AT LEAST IN PART, ON THE A RATIO OF TOTAL NUMBER OF CLOCK CYCLES PER INSTRUCTION COMPLETED OVER A TIME INTERVAL
304
ALLOCATING THE THREAD TO SECOND HETEROGENEOUS PROCESSOR CORE BASED, AT LEAST IN PART, ON THE CPI VALUE
306

FIG. 3

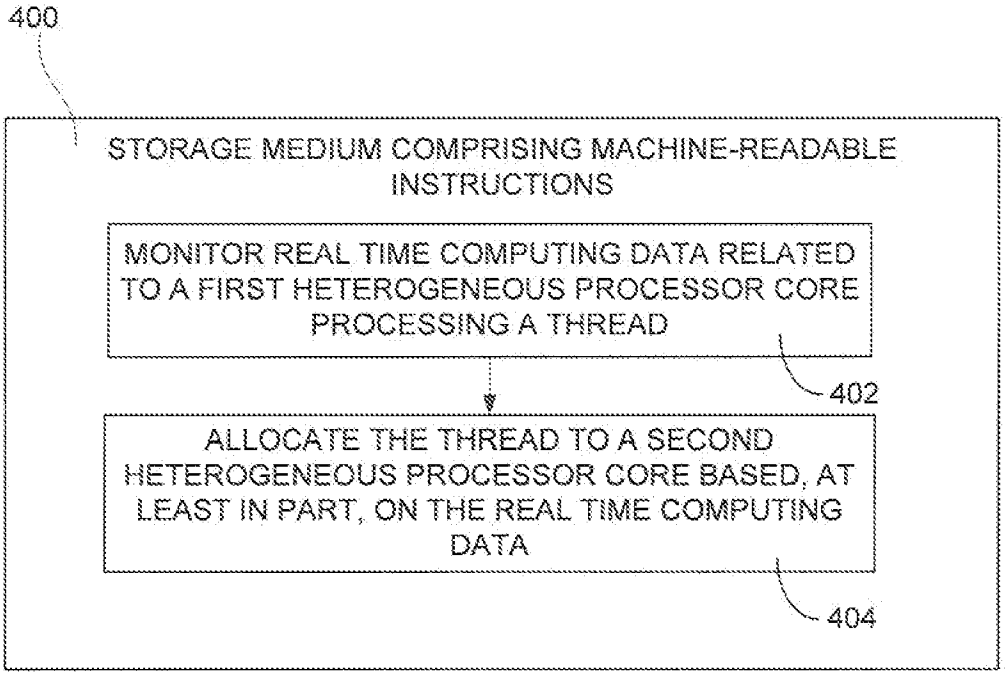
400
STORAGE MEDIUM COMPRISING MACHINE-READABLE INSTRUCTIONS
MONITOR REAL TIME COMPUTING DATA RELATED TO A FIRST HETEROGENEOUS PROCESSOR CORE PROCESSING A THREAD
402
ALLOCATE THE THREAD TO A SECOND HETEROGENEOUS PROCESSOR CORE BASED, AT LEAST IN PART, ON THE REAL TIME COMPUTING DATA
404

FIG. 4

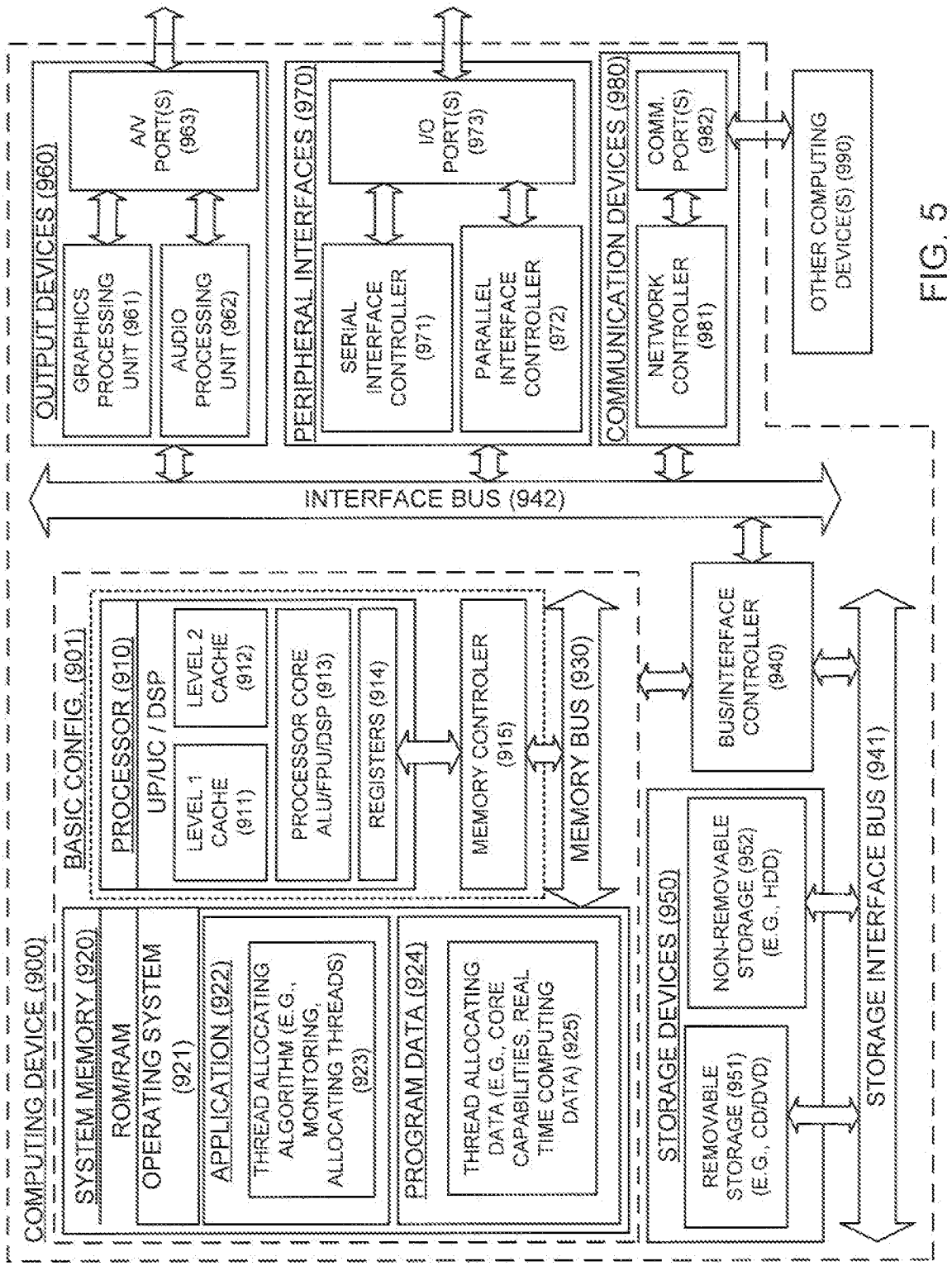
COMPUTING DEVICE (900)
BASIC CONFIG. (901)
PROCESSOR (910)
UP/UC / DSP
LEVEL 1 CACHE (911)
LEVEL 2 CACHE (912)
PROCESSOR CORE ALU/FPU/DSP (913)
REGISTERS (914)
MEMORY CONTROLER (915)
MEMORY BUS (930)
SYSTEM MEMORY (920)
ROM/RAM
OPERATING SYSTEM (921)
APPLICATION (922)
THREAD ALLOCATING ALGORITHM (E.G., MONITORING, ALLOCATING THREADS) (923)
PROGRAM DATA (924)
THREAD ALLOCATING DATA (E.G., CORE CAPABILITIES, REAL TIME COMPUTING DATA) (925)
STORAGE DEVICES (950)
REMOVABLE STORAGE (951) (E.G., CD/DVD)
NON-REMOVABLE STORAGE (952) (E.G., HDD)
STORAGE INTERFACE BUS (941)
BUS/INTERFACE CONTROLLER (940)
INTERFACE BUS (942)
OUTPUT DEVICES (960)
GRAPHICS PROCESSING UNIT (961)
AUDIO PROCESSING UNIT (962)
A/V PORT(S) (963)
PERIPHERAL INTERFACES (970)
SERIAL INTERFACE CONTROLLER (971)
PARALLEL INTERFACE CONTROLLER (972)
I/O PORT(S) (973)
COMMUNICATION DEVICES (980)
NETWORK CONTROLLER (981)
COMM. PORT(S) (982)
OTHER COMPUTING DEVICE(S) (990)

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 10/37496

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 9/46 (2010.01)
USPC - 718/100
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 9/46 (2010.01)
USPC: 718/100

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
USPC: 718/100, 102, 104, 105, 107 (keyword limited; terms below)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PubWest; Google Scholar; Google Patents; FreePatentsOnline. Search terms used: thread process fork, hardware, core processor processing-unit, allocate distribute, monitor track check surveillance observe, real-time realtime, heterogeneous diverse, clock-cycle, instructions, CPI cycles-per-instruction ...

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X -- Y	US 2009/0217277 A1 (JOHNSON et al.) 27 August 2009 (27.08.2009) entire document, especially Abstract; Fig. 3; para [0020]-[0023], [0026], [0030], [0031], [0033], [0046], [0052], [0061], [0063], [0075], [0085], [0086], [0088], [0089], [0093], [0097]	1, 2, 5-9, 11-16, 18,20-24 --- 3, 4, 10, 17, 19
Y	US 2005/0021931 A1 (ANDERSON et al.) 27 January 2005 (27.01.2005) entire document, especially Abstract; para [0002], [0079], [0080], [0096]	3, 4, 19
Y	US 5,659,721 A (SHEN et al.) 19 August 1997 (19.08.1997) entire document, especially Abstract; col 8, ln 21-63; col 13, ln 56-60; col 17, ln 11-28; col 78, ln 34-48	10, 17
A	US 2005/0086660 A1 (ACCAPADI et al.) 21 April 2005 (21.04.2005) entire document	1 - 24
A	US 5,968,115 A (TROUT) 19 October 1999 (19.10.1999) entire document	1 - 24
A	US 2005/0039185 A1 (HEIDARI-BATENI et al.) 17 February 2005 (17.02.2005) entire document	1 - 24

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

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

Date of the actual completion of the international search	Date of mailing of the international search report
15 September 2010 (15.09.2010)	21 SEP 2010
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-3201	Authorized officer: Lee W. Young PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (July 2009)