(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 2011/031355 A1

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

(21) International Application Number:
PCT/US2010/037489

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

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/557,864 11 September 2009 (11.09.2009) US

(71) Applicant *(for all designated States except US)*: **EMPIRE TECHNOLOGY DEVELOPMENT LLC** [US/US]; 2711 Centerville Road NE, 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 320, 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).

Published:

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

(54) **Title**: CACHE PREFILL ON THREAD MIGRATION

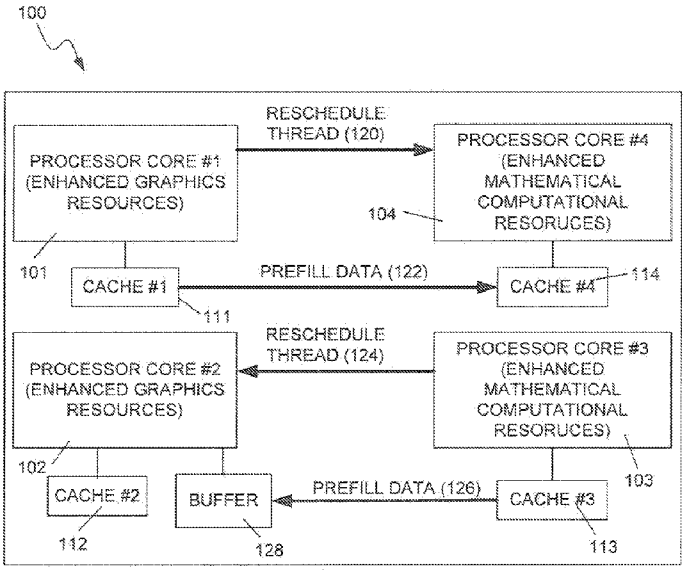

FIG. 1

(57) **Abstract**: Techniques for pre~fiiling a cache associated with a second core prior to migration of a thread from a first core to the second core are generally disclosed. The present disclosure contemplates that some computer systems may include a plurality of processor cores, and that some cores may have hardware capabilities different from other cores, in order to assign threads to appropriate cores, thread/core mapping may be utilized and, in some cases, a thread may be reassigned from one core to another core, In a probabilistic anticipation that a thread may be migrated from a first core to a second core, a cache associated with the second core may be pre-fiiled (e.g., may become filled with some data before the thread is rescheduled on the second core). Such a cache may be a iocai cache to the second core and/or an associated buffer cache, for example.

WO 2011/031355 A1

Title: CACHE PREFILL ON THREAD MIGRATION

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to U.S. Patent Application Serial No. 12/557,864 entitled, "CACHE PREFILL ON THREAD MIGRATION," 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. 12/557,971, entitled "THREAD SHIFT: ALLOCATING THREADS TO CORES," filed September 11, 2009, by Wolfe et al., and/or co-pending 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., the entire disclosures of which are incorporated herein by reference.

BACKGROUND

[0003] The present disclosure is generally related to multi-core computer systems and, more particularly, to transferring data in anticipation of thread migration between cores.

SUMMARY OF THE DISCLOSURE

[0004] The present disclosure generally relates to multi-core computer processing. Specifically, the present disclosure relates to migrating threads among processor cores of multi-core computer systems.

[0005] A first aspect of the present disclosure generally describes methods of migrating a thread from a first processor core to a second processor core. Such methods may include anticipating that a thread is to be migrated from a first processor core (associated with a first cache) to a second processor core (associated with a buffer and/or a second cache). Such methods may also include transferring data associated

with the thread from the first cache to the buffer and/or the second cache, and, after transferring data associated with the thread, migrating the thread from the first processor core to the second processor core.

[0006] In some examples of the first aspect, the methods may also include at least partially executing the thread on the first processor core prior to anticipating that the thread is to be migrated. Some examples may also include at least partially executing the thread on the second processor core after migrating the thread.

[0007] In some examples of the first aspect, data may include a cache miss, a cache hit and/or a cache line eviction associated with the thread.

[0008] In some examples, the second processor core may be associated with the second cache. In such examples, transferring the data may include transferring the data from the first cache to the second cache. In some examples of the first aspect, the second cache may include existing data associated with the thread. In such examples, transferring the data may include transferring new data associated with the thread.

[0009] In some examples of the first aspect, the second processor core may be associated with the buffer. In such examples, transferring the data may inclue transferring the data from the first cache to the buffer.

[0010] In some examples, anticipating that the thread is to be migrated to the second processor core may include determining that there is at least a threshold probability that the thread is to be migrated to the second processor core. In some examples, anticipating that the thread is to be migrated to a second processor core may be based, at least in part, on hardware capabilities of the second processor core.

[0011] A second 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 predict that a thread will be rescheduled from a first processor core to a second processor core, store data associated with the thread in a memory associated with the second core, and reschedule the thread from the first core to the second core after the data associated with the thread is stored in the memory

associated with the second core.

[0012] In some examples, the data associated with the thread may be new data associated with the thread, and the memory may include existing data associated with the thread. Some examples may enable the computing platform to predict that the thread will be rescheduled based, at least in part, upon a probability that the thread will be rescheduled.

[0013] In some examples of the second aspect, hardware capabilities associated with the first processor core may differ from hardware capabilities associated with the second processor core. In such examples, the instructions may enable the computing platform to predict that the thread will be rescheduled based, at least in part, upon the hardware capabilities associated with the first processor core, the hardware capabilities associated with the second processor core, and/or execution characteristic(s) associated with the thread.

[0014] In some examples of the second aspect, memory may include a cache and/or a buffer. In some examples of the second aspect, the instructions may enable the computing platform to reschedule the thread from the first core to the second core subsequent to storage of substantially all of the data associated with the thread in the memory associated with the second core.

[0015] A third aspect of the present disclosure generally describes methods of prefilling a cache. Such examples may include identifying processor cores to which a thread is to be migrated, transferring data associated with the thread to a cache and/or a buffer associated with the processor cores to which the thread is to be migrated, and migrating the thread to the processor cores to which the thread is to be migrated.

[0016] In some examples of the third aspect, transferring the data may be substantially complete prior to migrating the thread. In some examples, identifying the processor core to which the thread may be migrated may be based, at least in part, on information collected using a performance counter associated with the processor core(s). In some examples, the information collected using the performance counter may include numbers of line evictions associated with individual threads running on the

processor cores.

[0017] In some examples of the third aspect, identifying the processor core to which the thread may be migrated may be based, at least in part, on real-time computing information associated with the thread. In such examples, when the real-time computing information indicates that the thread is falling behind a target deadline, the thread may be migrated to a faster processor core. In some examples, transferring the data associated with the thread may include transferring the data from a first cache associated with a current processor core to a second cache associated with the processor core to which the thread may be migrated.

[0018] A fourth aspect of the present disclosure generally describes multi-core systems. Such multi-core systems may include a first processor core, a first cache associated with the first processor core, a second processor core, and a second cache and/or a buffer associated with the second processor core. The multi-core system may be configured to transfer data from the first cache to the second cache and/or the buffer, and, subsequently, migrate a thread from the first processor core to the second processor core, the thread being associated with the data.

[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. In some examples, the data may include a cache miss, a cache hit, and/or a cache line eviction associated with the thread.

[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;

FIG. 2 is block diagram illustrating an example multi-core system including a performance counter;

FIG. 3 is a flowchart depicting an example method for migrating a thread from a first processor core to a second processor core;

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

FIG. 5 is a flowchart depicting an example method for prefilling a cache; and

FIG. 6 is a block diagram illustrating an example computing device that may be arranged for cache prefill 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 generally related to multi-core computers and, more particularly, to transferring data in anticipation of thread migration between cores.

[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. An example core may be associated with a cache, which may include a temporary storage area where frequently accessed data may be stored for rapid access. Such a cache may be a local cache and/or an associated buffer cache, for example. 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/core mapping 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 before execution of the thread is complete.

[0026] The present disclosure describes that when a thread is rescheduled from a first core to a second core, a cache associated with the second core may be pre-filled. In other words, the cache associated with the second core may be at least partially filled with thread-related data before the thread is rescheduled on the second core.

[0027] FIG. 1 is a block diagram illustrating an example multi-core system 100 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 101, 102, 103, and/or 104. Individual cores 101, 102, 103, and/or 104 may be associated with one or more caches 111, 112, 113, and/or 114, and/or buffers 128. In an example embodiment, a multi-core system 100 may include one or more cores 101, 102, 103, and/or 104, each core having different capabilities. In other words, a multi-core system 100 may include heterogeneous hardware. For example, cores 101 and 102 may include enhanced graphics resources and/or cores 103 and 104 may include enhanced mathematical computational resources.

[0028] In an example embodiment, a thread 120 which may initially benefit from enhanced graphics capabilities may be initially executed on core 101. Based at least in part on the expectation that thread 120 may benefit from enhanced mathematical computational capabilities, data 122 pertaining to thread 120 may be prefilled into cache 114, and thread 120 may be rescheduled to core 104 to complete its execution. Similarly, a thread 124 which may initially benefit from enhanced mathematical computational capabilities may be initially executed on core 103. Based at least in part on the expectation that thread 124 may benefit from enhanced graphics capabilities, data 126 pertaining to thread 124 may be prefilled into buffer 128, and thread 124 may be rescheduled to core 102. In this example embodiment, one or more of data 122 and 126 may be filled into cache 114 and/or buffer 128, respectively, prior to rescheduling threads 120 and 124 to cores 104 and 102, respectively.

[0029] 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.

[0030] In some example embodiments, some threads may be associated with one or more execution characteristics, which may be expressed and/or based on information collected by one or more performance counters, for example. In some example embodiments, thread mapping may be based at least in part on one or more of the execution characteristics.

[0031] In some example embodiments, threads may be mapped 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 mapped to a core including large L1 cache hardware. Similarly, a thread associated with a large SSE (instruction set) demand may be mapped to a core including native SSE hardware implementation.

These examples are non-limiting, and it will be understood that threads may be mapped based at least in part on any hardware characteristic, instruction set, and/or other characteristic of a core and/or a thread.

[0032] In some example embodiments, thread execution characteristics may vary over time based on a phase of the program running in the thread. For example, a thread may originally have a large L1 cache demand, but may have a minimal L1 cache demand at a later time. The thread may be mapped to different cores at different times during its execution, which may result in improved performance. For example, the thread may be mapped to a core including a relative large L1 cache when L1 demand is high, and/or the thread may be mapped to a core having a smaller L1 cache when L1 demand is lower.

[0033] In some example embodiments, determining whether or not to migrate a thread to a different core and/or when to perform such a migration 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, a thread scheduler within the operating system may establish probabilities for thread migration. For example, the scheduler may examine the pending thread queue to determine how many threads are waiting to be scheduled and how many of those threads would prefer to be scheduled on core 2. The scheduler may also estimate how long a current portion of the current thread executing on core 1 (thread A) will require in order to complete. An estimation may then be performed to determine the likelihood that one of the waiting threads will be scheduled on core 2 prior to thread A requesting rescheduling. If this probability estimate exceeds a predetermined threshold, then data related to thread A may be

migrated to the core 2 cache.

[0035] In some example embodiments, processors and/or caches may be adapted 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. Performance counter data may be incorporated into the estimates of rescheduling probabilities discussed above.

[0036] FIG. 2 is block diagram illustrating an example multi-core system 200 including a performance counter 218, arranged in accordance with at least some embodiments of the present disclosure. Cores 202, 204, and/or 206 (which may be associated with caches 212, 214, and/or 216) may be operatively coupled to a performance counter 218. Performance counter 218 may be configured to store the counts for hardware-related activities within the computer system, for example. Thread 220 migration (from core 202 to core 204, for example) may be at least partially determined using data collected by performance counter 218. In some example embodiments, data 222 may be prefilled into cache 214 from cache 212 prior to migration of thread 220.

[0037] 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.

[0038] 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.

[0039] In some example embodiments, data associated with threads may be evaluated to determine when a thread should be migrated to another core and/or to which core the thread should be migrated. For example, a system may use real-time computing (RTC) 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 (e.g., a core operating at a higher clock speed), for example.

[0040] 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 migrate 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.

[0041] 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.

[0042] Thread migration may be expected and/or anticipated based at least partially on, for example, variation of thread execution characteristics over time, data associated with a performance counter, and/or data associated with threads (e.g., RTC computing data).

[0043] FIG. 3 is a flowchart depicting an example method 300 for migrating a thread from a first processor core to a second processor core, arranged in accordance with at least some embodiments of the present disclosure. Example methods 300 may include one or more of processing operations 302, 304, 306, 308 and/or 310.

[0044] Processing may begin at operation 304, which may include anticipating that the thread is to be migrated from a first processor core associated with a first cache to a second processor core, the second processor core being associated with one or more

of a buffer and/or a second cache. Operation 304 may be followed by operation 306, which may include transferring data associated with the thread from the first cache to one or more of the buffer and/or the second cache. Operation 306 may be followed by operation 308, which may include migrating the thread from the first processor core to the second processor core.

[0045] Some example methods may include operation 302 prior to operation 304. Operation 302 may include at least partially executing the thread on the first processor core. Some example methods may include operation 310 after operation 308. Operation 310 may include at least partially executing the thread on the second processor core.

[0046] FIG. 4 is a schematic diagram illustrating 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 predict that a thread will be rescheduled from a first processor core to a second processor core (operation 402); store data associated with the thread in a memory associated with the second core (operation 404); and reschedule the thread from the first core to the second core (operation 406).

[0047] FIG. 5 is a flowchart depicting an example method 500 for prefilling a cache in accordance with at least some embodiments of the present disclosure. Example methods 500 may include one or more of processing operations 502, 504, and/or 506.

[0048] Processing for method 500 may begin at operation 502, which may include identifying one or more processor cores to which a thread may be migrated. Operation 502 may be followed by operation 504, which may include transferring data associated with the thread to one or more of a cache and/or a buffer associated with the processor core to which the thread may be migrated. Operation 504 may be followed by operation 506, which may include migrating the thread to the processor core to which the thread may be migrated.

[0049] FIG. 6 is a block diagram illustrating an example computing device 900 that

is arranged for cache prefill in accordance with at least some embodiments of the present disclosure. In a very basic configuration 901, computing device 900 typically may 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.

[0050] 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.

[0051] 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 cache prefill algorithm 923 that may be arranged to anticipate rescheduling and prefill a cache. Program Data 924 may include cache prefill data 925 that may be useful for prefilling a cache, as will be further described below. In some embodiments, application 922 can be arranged to operate with program data 924 on an operating system 921 such that a cache may be prefilled in accordance with the techniques described herein. This described basic configuration is illustrated in FIG. 6 by those components within dashed line 901.

[0052] 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.

[0053] 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.

[0054] 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.

[0055] 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.

[0056] 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.

[0057] 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.

[0058] 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."

[0059] 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.

What is claimed is:

1. A method of migrating a thread from a first processor core to a second processor core, the method comprising:

anticipating that a thread is to be migrated from a first processor core associated with a first cache to a second processor core, the second processor core being associated with one or more of a buffer and/or a second cache;

transferring data associated with the thread from the first cache to one or more of the buffer and/or the second cache; and

after transferring data associated with the thread, migrating the thread from the first processor core to the second processor core.

2. The method of claim 1, further comprising, prior to anticipating that the thread is to be migrated, at least partially executing the thread on the first processor core.

3. The method of claim 1, further comprising, after migrating the thread, at least partially executing the thread on the second processor core.

4. The method of claim 1, wherein the data includes one or more of a cache miss, a cache hit, and/or a cache line eviction associated with the thread.

5. The method of claim 1, wherein the second processor core is associated with the second cache; and wherein transferring the data includes transferring the data from the first cache to the second cache.

6. The method of claim 5, wherein the second cache includes existing data associated with the thread; and wherein transferring the data includes transferring new data associated with the thread.

7. The method of claim 6, wherein the new data includes one or more of a cache miss, a cache hit, and/or a cache line eviction associated with the thread.

8. The method of claim 1, wherein the second processor core is associated with the buffer; and wherein transferring the data includes transferring the data from the first cache to the buffer.

9. The method of claim 1, wherein anticipating that the thread is to be migrated to the second processor core comprises determining that there is at least a threshold probability that the thread is to be migrated to the second processor core.

10. The method of claim 1, wherein anticipating that the thread is to be migrated to a second processor core is based at least in part on one or more of hardware capabilities of the second processor core.

11. 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:

predict that a thread will be rescheduled from a first processor core to a second processor core;

store data associated with the thread in a memory associated with the second core; and

reschedule the thread from the first core to the second core after the data associated with the thread is stored in the memory associated with the second core.

12. The article of claim 11, wherein the data associated with the thread is new data associated with the thread; and wherein the memory includes existing data associated with the thread.

13. The article of claim 11, wherein the instructions enable the computing platform to predict that the thread will be rescheduled based at least in part upon a probability that the thread will be rescheduled.

14. The article of claim 11, wherein one or more hardware capabilities associated with the first processor core differ from one or more hardware capabilities associated with the second processor core; and wherein the instructions enable the computing platform to predict that the thread will be rescheduled based at least in part upon the one or more hardware capabilities associated with the first processor core, the one or more hardware capabilities associated with the second processor core, and one or more execution characteristics associated with the thread.

15. The article of claim 11, wherein the memory includes one or more of a cache and/or a buffer.

16. The article of claim 11, wherein the instructions enable the computing platform to reschedule the thread from the first core to the second core subsequent to storage of substantially all of the data associated with the thread in the memory associated with the second core.

17. A method of prefilling a cache comprising:

identifying one or more processor cores to which a thread is to be migrated;

transferring data associated with the thread to one or more of a cache and/or a buffer associated with the processor cores to which the thread is to be migrated; and

migrating the thread to the processor cores to which the thread is to be migrated.

18. The method of claim 17, wherein transferring the data is substantially complete prior to migrating the thread.

19. The method of claim 17, wherein identifying the processor core to which the thread may be migrated is based at least in part on information collected using a performance counter associated with at least one of the processor cores.

20. The method of claim 19, wherein the information collected using the performance

counter includes numbers of line evictions associated with individual threads running on the processor cores.

21. The method of claim 17, wherein identifying the processor core to which the thread may be migrated is based at least in part on real-time computing information associated with the thread; and wherein, when the real-time computing information indicates that the thread is falling behind a target deadline, the thread is migrated to a faster one of the processor cores.

22. The method of claim 17, wherein transferring the data associated with the thread includes transferring the data from a first cache associated with a current processor core to a second cache associated with the processor core to which the thread may be migrated.

23. A multi-core system comprising:

a first processor core;

a first cache associated with the first processor core;

a second processor core; and

one or more of a second cache and/or a buffer associated with the second processor core;

wherein the multi-core system is configured to transfer data from the first cache to one or more of the second cache and/or the buffer and, subsequently, migrate a thread from the first processor core to the second processor core, the thread being associated with the data.

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

25. The multi-core system of claim 24, 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.

26. The multi-core system of claim 23, wherein the data comprises one or more of a cache miss, a cache hit, and/or a cache line eviction associated with the thread.

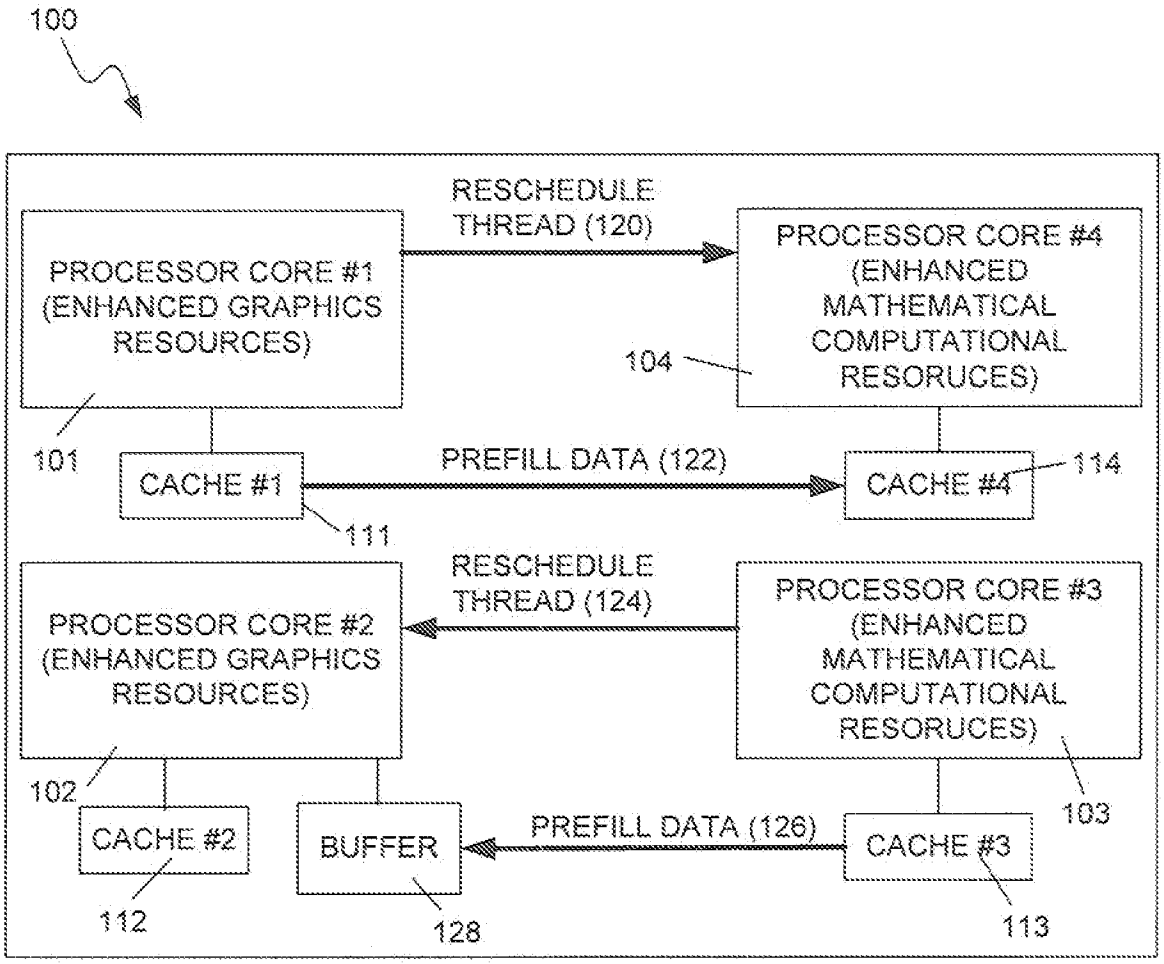
100
RESCHEDULE THREAD (120)
PROCESSOR CORE #1 (ENHANCED GRAPHICS RESOURCES)
PROCESSOR CORE #4 (ENHANCED MATHEMATICAL COMPUTATIONAL RESORUCES)
104
101
CACHE #1
PREFILL DATA (122)
CACHE #4
114
111
RESCHEDULE THREAD (124)
PROCESSOR CORE #2 (ENHANCED GRAPHICS RESOURCES)
PROCESSOR CORE #3 (ENHANCED MATHEMATICAL COMPUTATIONAL RESORUCES)
102
103
CACHE #2
BUFFER
PREFILL DATA (126)
CACHE #3
112
128
113

FIG. 1

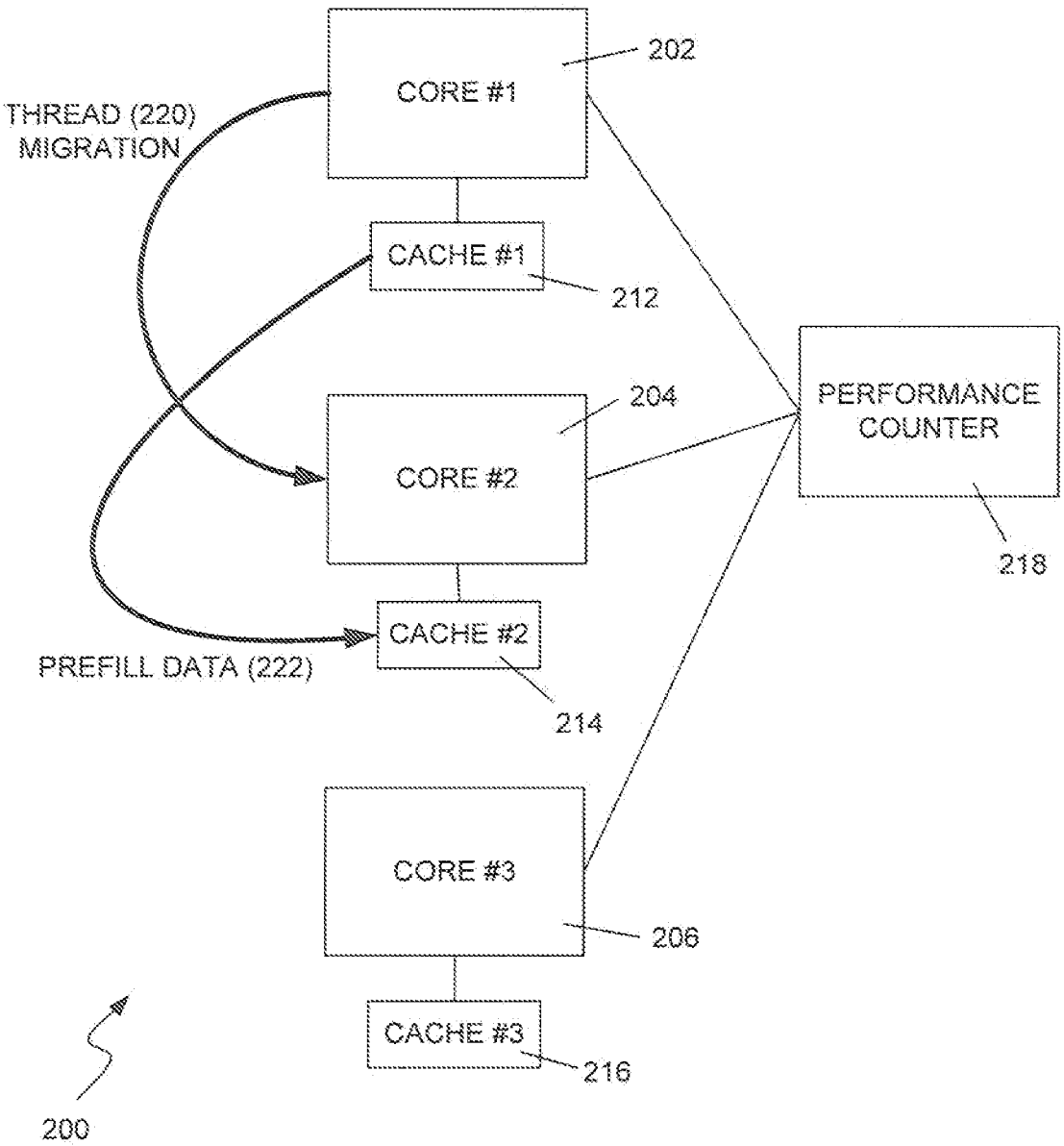
CORE #1
202
THREAD (220)
MIGRATION
CACHE #1
212
PERFORMANCE
COUNTER
CORE #2
204
218
CACHE #2
PREFILL DATA (222)
214
CORE #3
206
CACHE #3
216
200

FIG. 2

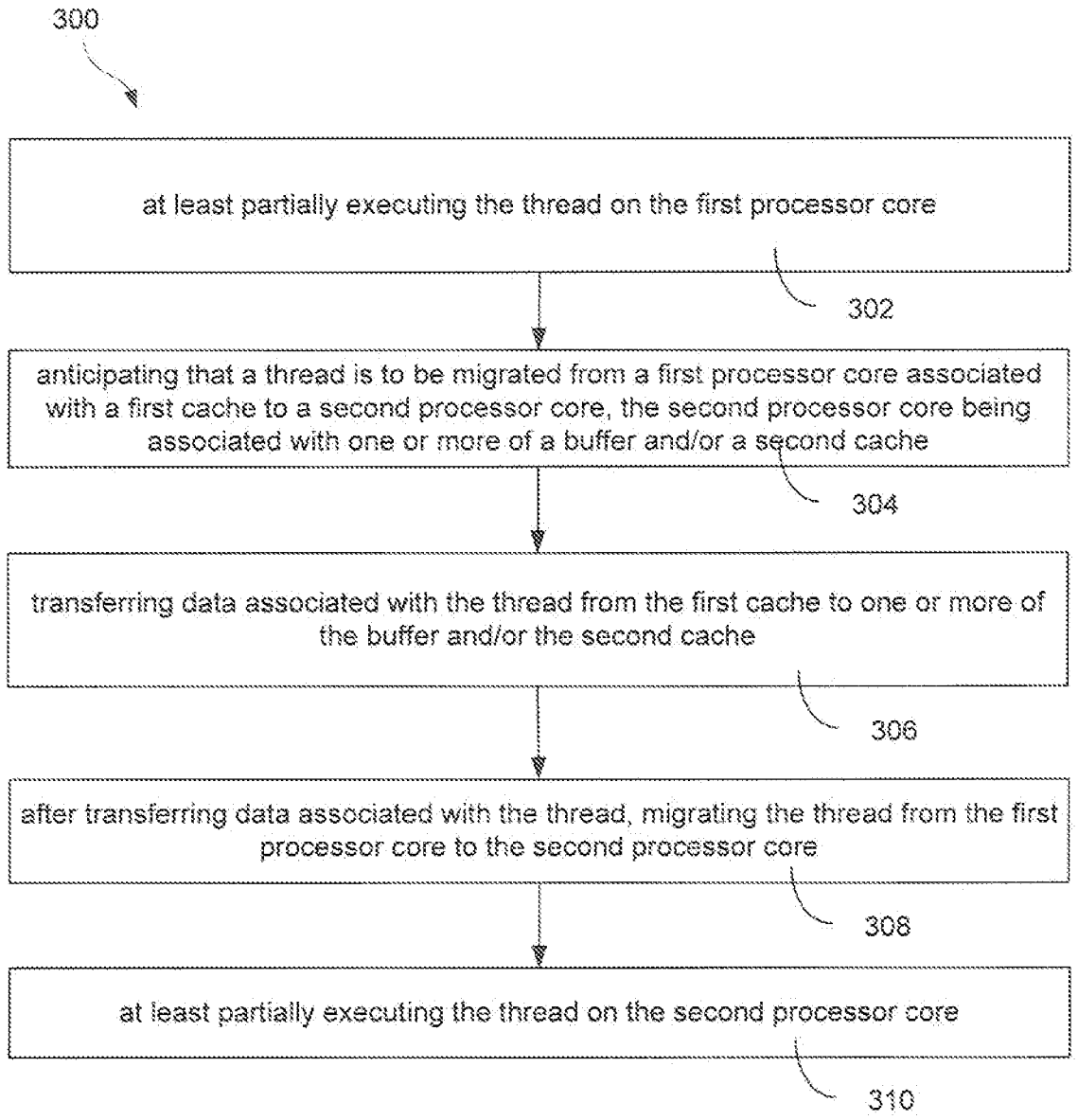
300
at least partially executing the thread on the first processor core
302
anticipating that a thread is to be migrated from a first processor core associated with a first cache to a second processor core, the second processor core being associated with one or more of a buffer and/or a second cache
304
transferring data associated with the thread from the first cache to one or more of the buffer and/or the second cache
306
after transferring data associated with the thread, migrating the thread from the first processor core to the second processor core
308
at least partially executing the thread on the second processor core
310

FIG. 3

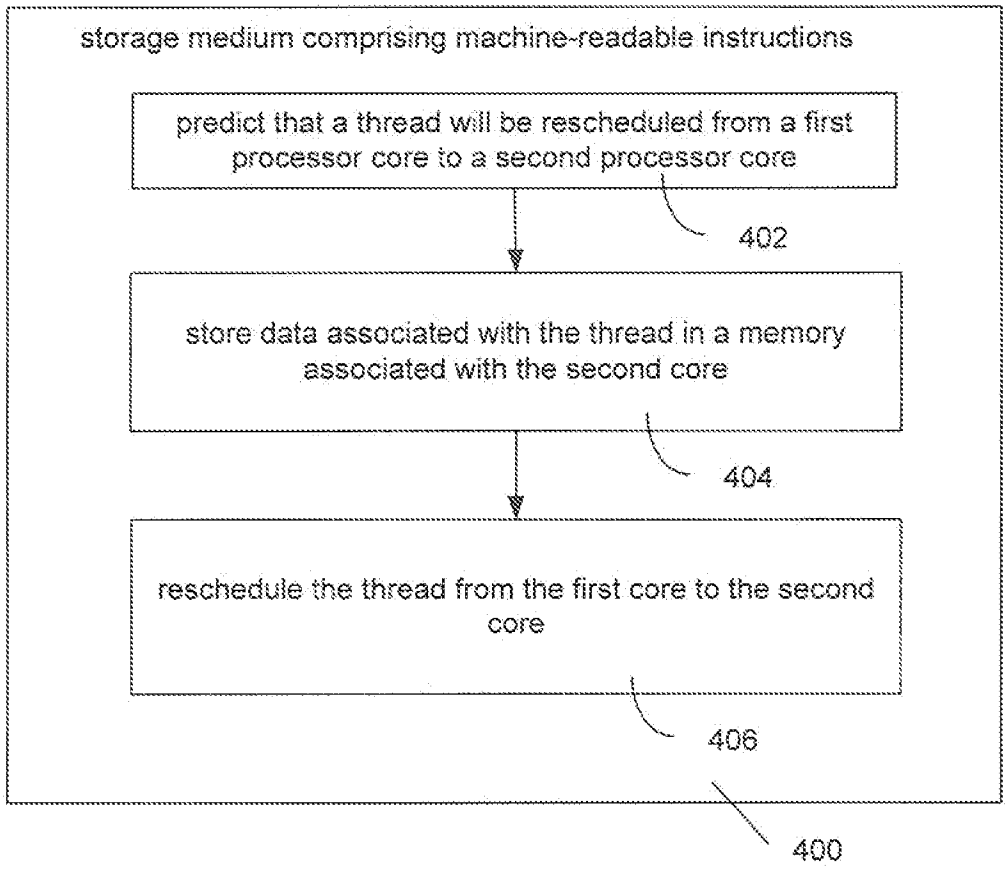
storage medium comprising machine-readable instructions
predict that a thread will be rescheduled from a first processor core to a second processor core
402
store data associated with the thread in a memory associated with the second core
404
reschedule the thread from the first core to the second core
406
400

FIG. 4

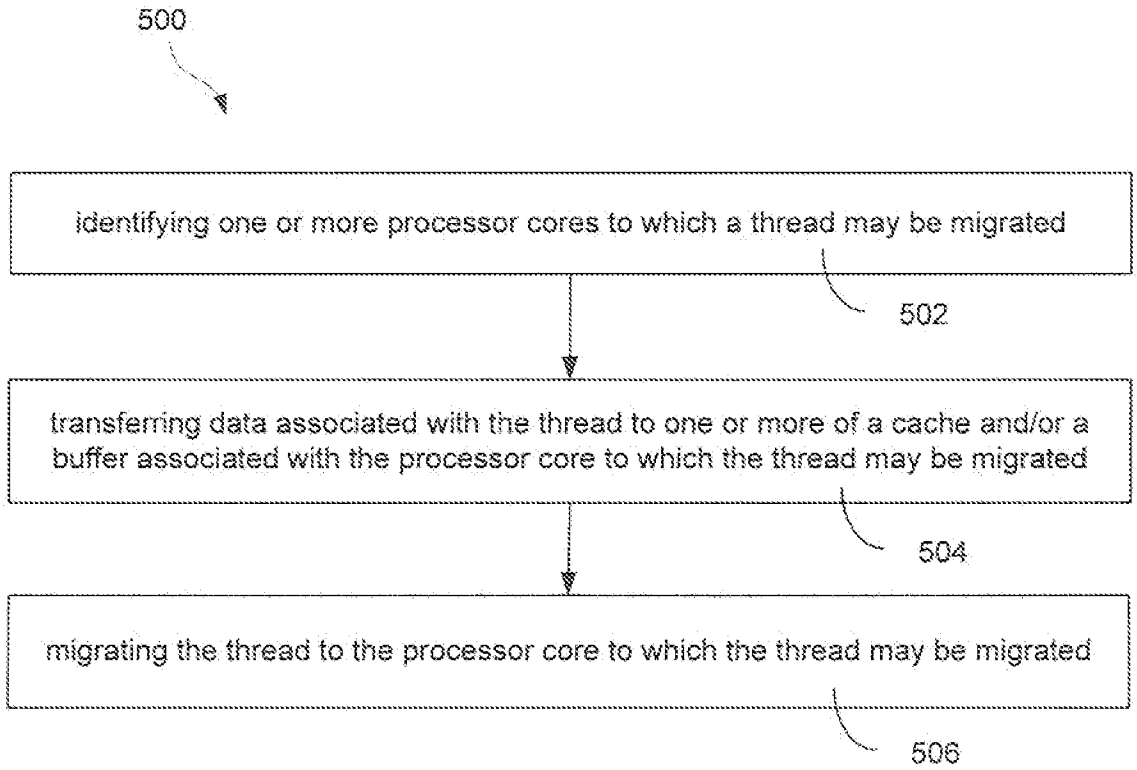
500
identifying one or more processor cores to which a thread may be migrated
502
transferring data associated with the thread to one or more of a cache and/or a buffer associated with the processor core to which the thread may be migrated
504
migrating the thread to the processor core to which the thread may be migrated
506

FIG. 5

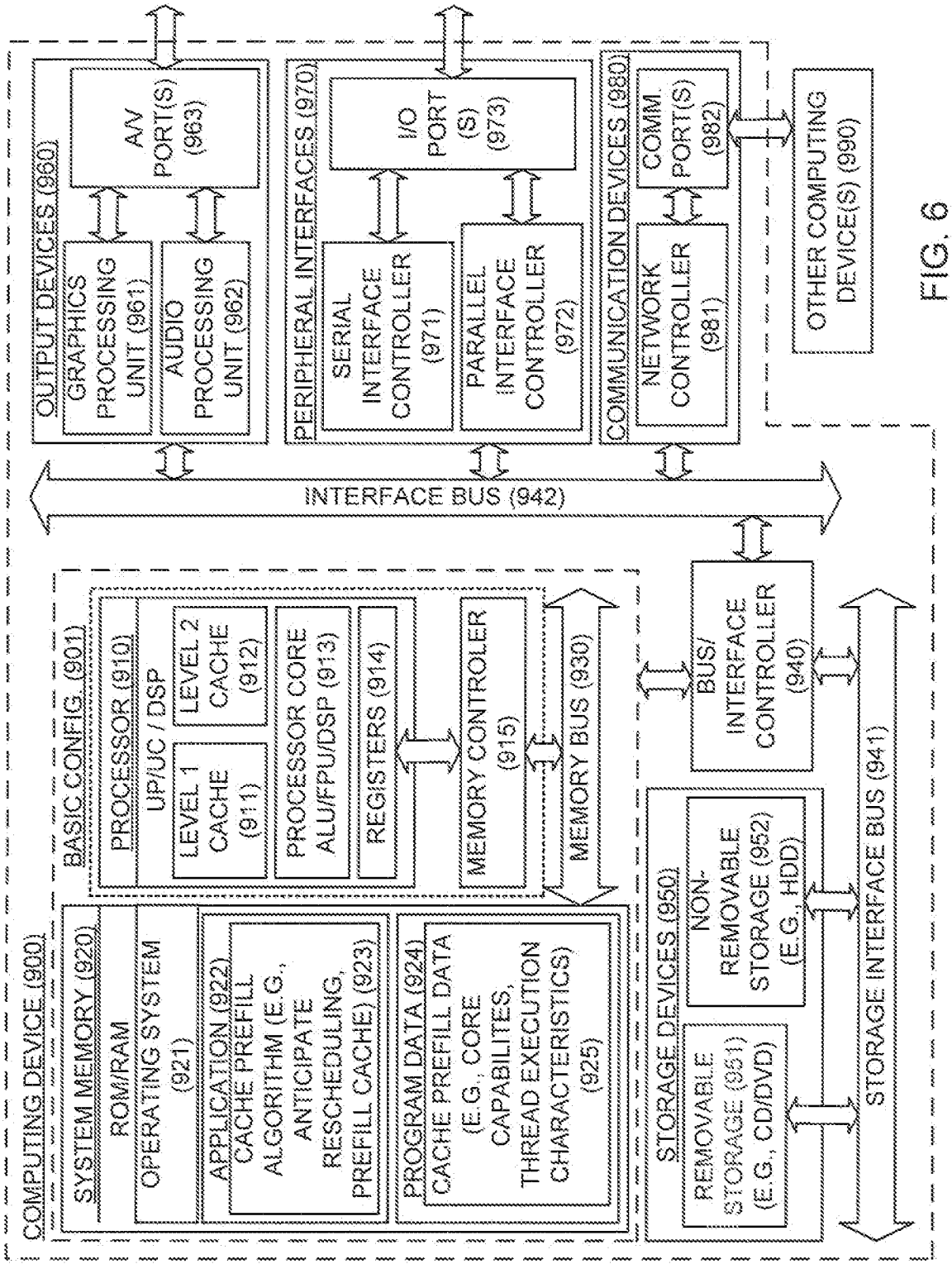
COMPUTING DEVICE (900)
BASIC CONFIG. (901)
PROCESSOR (910)
UP/UC / DSP
LEVEL 1 CACHE (911)
LEVEL 2 CACHE (912)
PROCESSOR CORE ALU/FPU/DSP (913)
REGISTERS (914)
MEMORY CONTROLER (915)
MEMORY BUS (930)
SYSTEM MEMORY (920)
ROM/RAM
OPERATING SYSTEM (921)
APPLICATION (922)
CACHE PREFILL ALGORITHM (E.G., ANTICIPATE RESCHEDULING, PREFILL CACHE) (923)
PROGRAM DATA (924)
CACHE PREFILL DATA (E.G., CORE CAPABILITES, THREAD EXECUTION CHARACTERISTICS) (925)
STORAGE DEVICES (950)
REMOVABLE STORAGE (951) (E.G., CD/DVD)
NON-REMOVABLE STORAGE (952) (E.G., HDD)
BUS/ INTERFACE CONTROLLER (940)
STORAGE INTERFACE BUS (941)
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. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 10/37489

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)
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, 103, 104, 106, 107; 711/1

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
PubWest(PGPB,USPT,EPAB,JPAB); Google Scholar. Terms: cache, miss, hit, eviction, thread, execute, migrate, processor, threshold, probability, move, multi-core, core, travel, buffer, anticipate, predict, detect, identify, cache, transfer, transmit, data, information

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2008/0244226 A1 (Li et al.) 02 October 2008 (02.10.2008) entire document (especially abstract, para [0002], [0010], [0015], [0023], [0030]-[0036])	1-26
Y	US 2006/0037017 A1 (Accapadi et al.) 16 February 2006 (16.02.2006) entire document (especially abstract, para [0009], [0038]-[0048])	1-26
Y	US 6,243,788 B1 (Franke et al.) 05 June 2001 (05.06.2001) entire document (especially col. 4, ln 30-58)	19, 20
A	US 2003/0163648 A1 (Smith) 28 August 2003 (28.08.2003) entire document (especially para [0066]-[0074])	1-26

☐ 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)	20 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)