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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 March 2006 (16.03.2006)

PCT

(10) International Publication Number
WO 2006/028520 A1

(51) International Patent Classification[7]: G06F 9/46

(21) International Application Number:
PCT/US2005/013122

(22) International Filing Date: 18 April 2005 (18.04.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/608,173 7 September 2004 (07.09.2004) US
60/608,177 7 September 2004 (07.09.2004) US

(71) Applicant *(for all designated States except US)*: **STARENT NETWORKS, CORP.** [US/US]; 30 International Place, Tewksbury, MA 01876 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **MORTSOLF, Timothy, G.** [US/US]; 686 Independence Drive, #5, Palatine, IL 60074 (US).

(74) Agents: **KANABE, George, L.** et al.; Wilmer Cutler Pickering Hale and Dorr LLP, 399 Park Avenue, New York, NY 10022 (US).

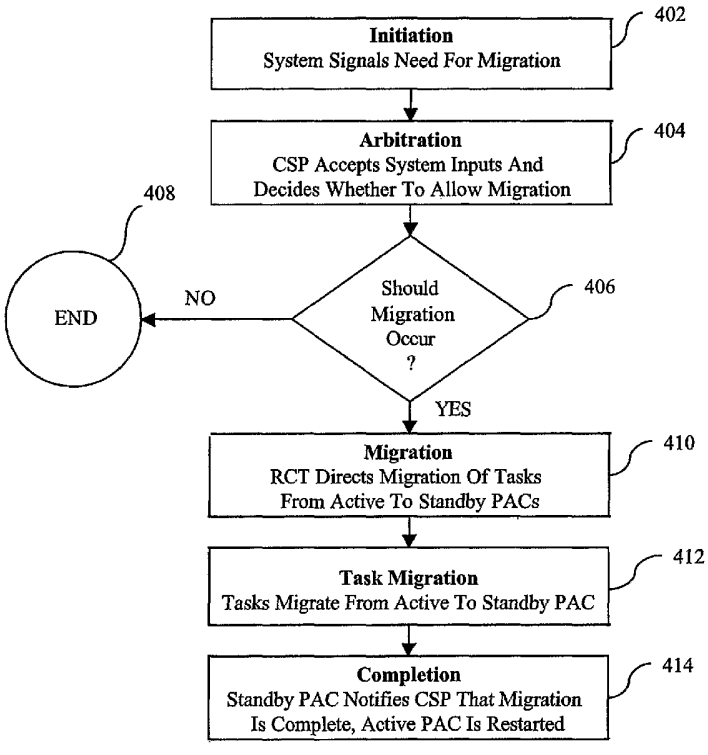
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK,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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: MIGRATION OF TASKS IN A COMPUTING SYSTEM



(57) Abstract: Methods and systems are provided for migrating tasks (406, 410, 412, 414) between computing devices in a computing system, such as a communications computing systems. For example, according to the invention, one computing device can take over providing services previously provided by another computing device without disrupting such services Moreover, according to various embodiments of the present invention, migration can be performed efficiently (404, 406) by avoiding treatment of all tasks alike. More particularly, tasks to be migrated may be separated into groups including critical tasks, restartable tasks, and migratable tasks, each of which is handled in a unique manner according to the invention.

WO 2006/028520 A1

MIGRATION OF TASKS IN A COMPUTING SYSTEM

Cross-Reference to Related Application

[0001] This application claims priority under 35 U.S.C. §119(e) to U.S. Provisional Patent Application No. 60/608,173, filed on September 7, 2004, and to U.S. Provisional Patent Application No. 60/608,177, filed on September 7, 2004, both of which are hereby incorporated by reference herein in their entirety.

Field of the Invention

[0002] The present invention relates to computing systems. More particularly, this invention relates to the migration of tasks between computing devices in computing systems, such as communications systems.

Background of the Invention

[0003] In many computing environments, including communications computing environments, fast, powerful and flexible computing systems are required for the efficient operation and/or execution of many different types of tasks (also called processes, operations and applications). This is especially true where multiple tasks are running on a computing system at the same time.

[0004] As persons versed in the art will appreciate, with increased demand for speed and flexibility in recent years, the configuration of computing systems has increased greatly. For example, users of a computing system in environments where there may be many competing demands, many different problem types, and/or continually changing computing needs may need to be able to quickly and easily change various characteristics of the computing system, including its capacity, speed, and/or its configuration. Additionally, users may want to expand the work capacity of a computing system without stopping execution of tasks running on the computing system. Moreover, users may want to change system configurations of a computing system, in order to better utilize the existing resources so that each task will have an optimum computing configuration.

[0005] Traditionally, the need for enhanced computing speed has been addressed by using a computing architecture where different tasks (or groups of tasks) have respective computing resources, such as central processing units (CPUs), dedicated to them. Initially,

computing systems of this type of architecture simply used a single central processing unit, where the power and speed of these computing systems were increased by increasing the clock rate of the single central processing unit.

[0006] More recently, computing systems have been developed which use several processors working in conjunction, rather than a single processor working alone. In this manner, multiple processors can be used in connection with a complex task demanding a relatively large amount of processing power, rather than the task having to rely solely on a single processor (which may have insufficient processing power). Generally speaking, such computing systems consist of several CPUs, each of which is controlled by a single operating system.

[0007] In a variant of the multiple processor computing system described above, called symmetric multiprocessing (SMP), the tasks running on the computing system are distributed equally across all processors. In many cases, these processors also share memory. In yet another variant, commonly referred to as asymmetric multiprocessing (AMP), a single processor acts as a "master" processor, while the remainder of the processors act as "slave" processors. In this configuration, all tasks, including those of the operating system, must pass through the master processor before being passed onto the slave processors.

[0008] Each of the multiprocessing architectures described above has the advantage that performance can be increased by adding additional processors. However, these architectures suffer from the disadvantage that the software running on such systems must be carefully written to take advantage of the multiple processors being used.

[0009] Computing performance can also be increased by dedicating computing resources (e.g., machines, CPUs, cards, etc.) to a task and optimally tuning the computing resources to that particular task. However, this approach has not been widely adopted because many (or most) situations involve uncoordinated application development, and because it is especially difficult (e.g., expensive) to dedicate resources among tasks in environments where the task mix is constantly changing. Further, with dedicated resources, it is essentially impossible (or at least very difficult) to quickly and easily migrate resources from one computing device to another, especially if different developers have been involved. Moreover, even when such a migration can be performed, it typically involves the

intervention of a system administrator, and may require at least some of the computer systems to be powered down and rebooted.

[0010] Alternatively, a computing system can be partitioned with hardware to make a subset of the resources on the computing system available to one or more particular tasks only. This approach avoids dedicating the resources permanently, given that the partitions can be changed, but potentially problematic issues still remain concerning, for example, performance improvements by means of load balancing of resources among partitions, and resource availability.

[0011] To address various availability and maintainability issues, a "shared everything" model has been proposed in which a large and robust centralized server that contains most of the computing resources is networked with (and services) many small, uncomplicated client network computers. Alternatively, clusters of computing systems may be used in which each system or node has its own memory and is controlled by its own operating system. The various computing systems interact by sharing data storage resources and passing messages among themselves using a communications network. A cluster system has the advantage that additional systems can easily be added to the cluster as needed. However, networks and clusters may suffer from a lack of shared memory, as well as from limited interconnect bandwidth which generally places limitations on performance.

[0012] As known by persons versed in the art, in many (or all) instances, it is highly desirable to be able to modify computer configurations on the fly (e.g., without rebooting any of the systems involved). Several prior art approaches have been used to attempt this accommodation. For example, a design called a virtual machine (VM), developed and marketed by International Business Machines Corporation, of Armonk, NY, uses a single physical machine, with one or more physical processors, in combination with software which simulates multiple virtual machines. Each of those virtual machines has access, at least in principle, to all the physical resources of the underlying real (physical) computing system. The assignment of resources to each virtual machine is controlled by a program called a "hypervisor." There is only one hypervisor in the system, and it is responsible for all the physical resources. Consequently, the hypervisor, not the other operating systems, deals with the allocation of physical hardware. The hypervisor intercepts requests for resources from the various operating systems that may be operating in the computing system, and deals with the requests in a globally-correct manner.

[0013] The VM architecture supports the concept of a logical partition (LPAR). Each LPAR contains some of the available physical CPUs (and other resources) which are logically assigned to the partition. The same resources can be assigned to more than one partition. LPARs are generally set up statically by an administrator, but are able to respond to changes in load (e.g., resource demand) dynamically, and without rebooting, in several ways. For example, assume that two logical partitions, each containing ten logical CPUs, are shared on a single physical computing system containing ten physical CPUs. In this case, if the logical partitions have complementary peak loads, each partition can take over the entire ten physical CPUs as the workload shifts (generally without requiring a re-boot or operator intervention).

[0014] In addition, the logical CPUs assigned to each partition can be turned on and off dynamically using normal operating system operator commands (also generally without requiring a re-boot). One significant limitation, however, is that the number of CPUs active at system initialization, which is ten in the example provided about, is the maximum number of CPUs that can be turned on in any partition.

[0015] Moreover, in cases where the aggregate workload demand of all logical partitions is more than can be delivered by the physical computing system, LPAR weights can be used to define the portion of the total CPU resources which is to be given to each partition. These weights can be changed by system administrators, on-the-fly, with no disruption. Nevertheless, the undesirable limitation still exists that the number of CPUs active at system initialization is the maximum number of CPUs that can be turned on in any partition.

[0016] Another known system is called a Parallel Sysplex, and is also marketed and developed by the International Business Machines Corporation. This architecture consists of a set of computers that are clustered via a hardware entity (called a "coupling facility") that is attached to each CPU. The coupling facilities on each node are connected (e.g., via a fiber-optic link), and each node operates as a traditional SMP machine, with a maximum of, e.g., 10 CPUs. Certain CPU instructions directly invoke the coupling facility. For example, a node registers a data structure with the coupling facility, and then the coupling facility takes care of keeping the data structures coherent within the local memory of each node.

[0017] An Enterprise 10000 Unix server, developed and marketed by Sun Microsystems, of Mountain View, CA, uses a partitioning arrangement called Dynamic System Domains to logically divide the resources of a single physical server into multiple partitions, or domains, each of which operates as a stand-alone server. Each of the partitions has CPUs, memory and input/output (I/O) hardware. Dynamic reconfiguration allows a system administrator to create, resize, or delete domains on the fly and without rebooting. Every domain remains logically isolated from any other domain in the system, isolating it completely from any software error or CPU, memory, or I/O error generated by any other domain. There is no sharing of resources between any of the domains.

[0018] Moreover, a Hive Project conducted at Stanford University concerns an architecture which is structured as a set of cells. When the system boots, each cell is assigned a range of nodes, each having memory and I/O devices, that the cell owns throughout execution. Each Hive cell manages the processors, memory and I/O devices on those nodes as if it were an independent operating system. The cells cooperate to present the illusion of a single system to user-level processes.

[0019] Hive cells are not responsible for deciding how to divide their resources between local and remote requests. Each cell is responsible only for maintaining its internal resources and for optimizing performance within the resources it has been allocated. Global resource allocation is carried out by a user-level process called "wax." The Hive system attempts to prevent data corruption by using certain fault containment boundaries between the cells. In order to implement the tight sharing expected from a multiprocessor system, despite the fault containment boundaries between cells, resource sharing is implemented through the cooperation of the various cell kernels. However, the policy is implemented outside the kernels, in the wax process. Both memory and processors can be shared.

[0020] A system called Cellular IRIX, developed and marketed by Silicon Graphics Inc., of Mountain View, CA, supports modular computing by extending traditional symmetric multiprocessing systems. The Cellular IRIX architecture distributes global kernel text and data into optimized SMP-sized chunks (cells), which represent a control domain consisting of one or more machine modules, where each module consists of processors, memory, and I/O devices. Tasks running on these cells rely extensively on a full set of local operating system services, including local copies of operating system text and kernel data structures, but only one instance of the operating system exists on the entire system. Inter-cell coordination

allows tasks to directly and transparently utilize processing, memory and I/O resources from other cells without incurring the overhead of data copies or extra context switches.

[0021] Another existing architecture is NUMA-Q, developed and marketed by Sequent Computer Systems, Inc., of Beaverton, OR. With NUMA-Q, groups of four processors (quads) per portion of memory are used as the basic building block for SMP nodes. Adding I/O to each quad further improves performance. The NUMA-Q architecture not only distributes physical memory, but also puts a predetermined number of processors and PCI slots next to each processor. The memory in each quad is not local memory in the traditional sense. Rather, it is a portion of the physical memory address space and has a specific address range. The address map is divided evenly over memory, with each quad containing a contiguous portion of address space. Only one copy of the operating system is running and, as in any SMP system, it resides in memory and runs processes without distinction and simultaneously on one or more processors.

[0022] As known by persons versed in the art, in various computing environments, such as the previously described computing environments, it is important to be able to change the configurations of a computing system without stopping execution of tasks running on the computing system. For example, it is often desirable to migrate tasks from a first computing device to another (e.g., in the case of a failure, or routine maintenance of the first computing device). Current computing systems that allow migration of tasks between computing devices within the computing system, however, have several drawbacks. For example, in these systems, it is often necessary to reboot before migration can be completed. Additional, for example, it is often necessary in these systems to keep the secondary computing device in lockstep with the primary computing device from which one or more tasks is to be migrated in the case of a failure or routine maintenance, for example. This requirement, however, creates inefficiency because of the need to have a separate "mirror" computing device for each primary computing device for which redundancy (or backup) is sought.

[0023] Accordingly, it is desirable to provide methods and systems for migrating tasks between computing devices in computing systems, such as communications systems, such that many or all of the benefits of the various computing systems described above are maintained, but not several of the drawbacks such as the requirement for mirroring or rebooting.

Summary of the Invention

[0024] In accordance with the principles of the present invention, methods and systems are provided for migrating tasks between computing devices in computing systems. According to various embodiments of the present invention, for example, one computing device can take over providing services previously provided by another computing device without disrupting such services. Moreover, according to various embodiments of the present invention, migration can be performed efficiently by avoiding treatment of all tasks alike.

[0025] In one embodiment, the invention provides a method for migrating tasks between computing devices in a computing system, where the method includes at least one of the sequential, sequence independent and non-sequential steps of receiving an indication that migration of one or more restartable and/or migratable tasks from a first computing device is to be initiated, starting one or more restartable tasks on a second computing device corresponding to one or more restartable tasks running on the first computing device, transmitting state information for one or more migratable tasks running on the first computing device to the second computing device, and starting the one or more migratable tasks on the second computing device using the transmitted state information.

[0026] In another embodiment, the invention provides a method for migrating tasks between computing devices in a computing system, where the method includes at least one of the sequential, sequence independent and non-sequential steps of initiating migration of a first task using a first technique for task migration between the first computing device and a second computing device, and initiating migration of a second task using a second technique for task migration between the first computing device and the second computing device, wherein the second computing device is not running in lockstep with the first computing device prior to the migration of the first and second tasks from the first computing device to the second computing device.

[0027] In yet another embodiment, the invention provides a system for migrating tasks between computing devices in a computing system, where the system includes means for initiating migration of a first task using a first technique for task migration between the first computing device and a second computing device, and means for initiating migration of a second task using a second technique for task migration between the first computing device and the second computing device, wherein the second computing device is not running in

lockstep with the first computing device prior to the migration of the first and second tasks from the first computing device to the second computing device.

Brief Description of the Drawings

[0028] Additional embodiments of the invention, its nature and various advantages, will be more apparent upon consideration of the following detailed description, taken in conjunction with the accompanying drawings, in which like reference characters refer to like parts throughout, and in which:

[0029] FIG. 1 is a simplified illustration of a chassis in a computing system that includes multiple PACs among which migration of tasks according to the principles of the present invention may be accomplished;

[0030] FIG. 2 is a simplified illustration of an active packet accelerator card (PAC) that includes four CPUs from which migration of tasks according to the principles of the present invention may be accomplished;

[0031] FIG. 3 is a simplified illustration of a standby PAC that includes four CPUs to which migration of tasks according to the principles of the present invention may be accomplished;

[0032] FIG. 4 is a flow chart illustrating the steps performed according to one embodiment of the present invention in migrating tasks from a first computing device to a second computing device in a computing system;

[0033] FIG. 5 is a more detailed flow chart of one of the steps depicted in FIG.4 according to one embodiment of the present invention;

[0034] FIG. 6 is a simplified illustration of an active PAC that hosts critical, restartable, and migratable tasks to be migrated to a standby PAC, and a special task sitCPU which monitors these critical, restartable, and migratable tasks according to the principles of the present invention;

[0035] FIG. 7 is a simplified illustration of an standby PAC that hosts a special task sitCPU for monitoring critical, restartable, and migratable tasks according to the principles of the present invention; and

[0036] FIG. 8 is a simplified illustration of a standby PAC that hosts critical, restartable, and migratable tasks which have been migrated from an active PAC, and a special task sitCPU which monitors these critical, restartable, and migratable tasks according to the principles of the present invention.

Detailed Description of the Invention

[0037] Methods and systems are provided for migrating tasks between computing devices in computing systems, such that many or all of the benefits of the various computing systems described above are maintained, but not several of the drawbacks such as the requirement for mirroring or rebooting. It will be understood that certain features which are well known in the art are not described in great detail in order to avoid complication of the subject matter of the present invention. Moreover, it will be understood that although particular reference is made herein to the effective and enhanced migration of tasks in a communications computing system, it will be understood that the invention is not limited in this manner.

[0038] Tasks with heavy demand for CPU processing are common in many fields, including communications systems, and particularly wireless (mobile) communications systems. Such systems have, for example, demanding, ongoing tasks that present significant challenges with respect to dynamic reconfiguration, e.g., reconfiguration under control of the operating system(s) running on the computing system and without system administrator intervention. Moreover, as also described above, attempts have been made at providing a flexible computer system having mechanisms for reconfiguring the computing system on the fly.

[0039] As known by persons versed in the art, computing systems generally use multiple computing devices, such as electronic circuitry cards, for handling various tasks. For example, in the case of a wireless communications computing system using a communications oriented computing platform called ST-16, by Starent Networks Corporation, of Tewksbury, MA, electronic circuitry cards referred to as packet accelerator cards (PACs) are used. Although references to the migration of tasks from a first PAC to a second PAC are made below, it will be understood that the invention is not limited in this manner, and that migration of tasks according to the invention may be accomplished in connection with any suitable type of computing devices.

[0040] FIG. 1 shows a chassis 100 in a computing system that includes multiple PACs 101-114 among which migration of tasks according to the invention may be accomplished. In the chassis 100 shown in FIG. 1, PAC 102 serves as a redundant or backup PAC for operational PAC 101, PAC 104 serves as a backup PAC for operational PAC 103, and so on up to PAC 114, which serves as a backup PAC for operational PAC 113. It will be understood that, although the chassis 100 shown in FIG. 1 shows a 1:1 redundancy of backup PACs to operational PACs, the invention is not limited in this manner. Rather, it will be understood that a 1:N redundancy is used according to various embodiments of the invention, as explained below. Chassis 100 shown in FIG. 1 also includes a management card, such as a Switch Processor Card (SPC) 115 as developed by Starent Networks Corporation of Tewksbury, MA, for controlling some or all of the chassis operations (e.g., starting chassis 100, managing PACs 101-114, handling recovery tasks, etc.). According to various embodiments, as shown in FIG. 1, chassis 100 also includes a redundant SPC (or RSPC) 116.

[0041] PACs can be classified as either active PACs or standby PACs. A standby PAC serves as a redundant or backup PAC that can take over and assume many or all tasks that are running when an active PAC fails. The process of transferring tasks from an active to a standby PAC is known as PAC migration, and can occur for any of a number of reasons as described below. In the embodiment of the invention illustrated by FIG. 2, an active PAC 200 includes four CPUs 202, 204, 206, and 208. Similarly, in the embodiment of the invention illustrated by FIG. 3, a standby PAC 300 includes four CPUs 302, 304, 306, and 308. The number of CPUs included in each of PACS 200 and 300, however, is not limited in this manner. Moreover, each CPU 202, 204, 206 and 208 of PAC 200, and each CPU 302, 304, 306, and 308 of PAC 300, executes a set of tasks specific to the host PAC. In addition, each of these CPUs executes a special task, referred to as sitCPU or a monitoring task, which keeps track of (e.g., monitors) all the other tasks running on the respective CPU. For example, the sitCPUs of CPUs 202, 204, 206 and 208 (of active PAC 200) may maintain a current list of all the other tasks running on the respective CPUs, their task ID numbers and task types (e.g., critical, restartable, or migratable), etc.

[0042] Generally speaking, two types of PAC migrations can occur: graceful and ungraceful. In the case of a graceful migration, tasks are transferred between a first PAC and a second PAC while the first PAC is still fully functional. A graceful migration may take place, for example, maintenance purposes. In the past, graceful migrations have required that

the second PAC (to which tasks are transferred) mirror the first PAC's state from initialization and up to the point of the migration. As appreciated by persons versed in the art, however, this requirement is often burdensome and subject to errors, due, e.g., to timing inaccuracies. In some situations, such as in an ungraceful migration, it may be necessary to completely migrate tasks to the second PAC when the first PAC fails. It will be understood that, generally speaking, when mirroring is required to be used for either a graceful or ungraceful migration, the first and second PACs are not said to be running in "active" and "standby" mode.

[0043] According to the invention, migration of tasks from an active PAC to a standby PAC is provided that only requires that the standby PAC be fully booted and be ready to initialize any tasks it must assume. Accordingly, because it is not necessary for the standby PAC to be running as a mirror to the active PAC, as mentioned above, the present invention permits a 1:N redundancy. In other words, it is not required that a separate standby PAC be running in lockstep (as a mirror) for each active PAC for which redundancy is desired. Rather, because a standby PAC is able to resume some or all of the functions of an active PAC without previously mirroring the standby pack, is it possible for a single standby PAC to provide redundancy (e.g., in the event that migration is required for failure reasons or simply desired for maintenance reasons) for more than one active PAC.

[0044] FIG. 4 is a flow diagram outlining the steps involved during a PAC migration according to one embodiment of the invention. At step 402, a migration is initiated by any one (or more than one) of several system inputs. For example, the administrator of the computing system in which PACs 200 and 300 operate may want to service a particular active PAC 200 while ensuring that any tasks currently running on the active PAC 200 are not lost. By using system Command Line Interface (CLI) commands, for example, the administrator can indicate the particular active PAC 200 from which tasks are to be migrated to a standby PAC 300. The administrator can also initiate a migration, for example, by manually removing an active PAC 200 card. In order to remove the card, the administrator generally manipulates a physical device called a trigger lock, which sends a signal indicating that a migration is necessary for the active PAC 200. A migration can also be initiated if the diagnostic system senses that a particular active PAC 200 card is experiencing one or more failures, for example, and needs to be shut down. It will be understood by persons versed in

the art that a migration may be initiated at step 402 by other means as well, and that the invention is not limited to the particular examples provided above.

[0045] At step 404, a dedicated process or task is used which monitors some or all of the chassis components (e.g., PACs 101-114 of chassis 100), and monitors system and manual inputs to determine whether a migration is necessary. This task, which is also referred to herein as a Card/Slot/Port task (CSP), generally runs on either an SPC (e.g., SPC 115) or, when the SPC is not available or is not functioning properly, on an RSPC (e.g., RSPC 116) of a chassis (e.g., chassis 100). For example, if a diagnostic system determines that a specific active PAC 200 card is overheating, it sends a request for migration to the CSP. The CSP then determines whether the active PAC 200 card is in the correct state for task migration, or whether the active PAC should continue to run for a certain amount of time before migrating its tasks. In this manner, the CSP acts as an arbitrator for all the inputs and determines whether any particular PAC 200 should be allowed to migrate its tasks to a standby PAC 300.

[0046] If it is determined by the CSP at step 406 that a migration should not occur, then at step 408, the process illustrated by the flow chart of FIG. 4 ends (and tasks are not migrated from active PAC 200 to standby PAC 300). According to other embodiments of the present invention, a time delay, for example, may be introduced into the migration process, rather than the process coming to an end. On the other hand, if the CSP determines that a migration should occur, at step 410, a Recovery Control Task (RCT) running on the SPC (or RSPC) begins to direct migration. According to various embodiments, the RCT may be any suitable task (such as develpoped by Starent Networks Corporation of Tewksbury, MA) that initiates some (or all) of the desired recovery actions when a task has failed and needs to be restarted. According to various embodiments, the RCT may handle card level recoveries (where all the tasks on a card require recovery), CPU level recoveries (where all the tasks on a CPU require recovery), and/or single task failure recoveries. In step 410, according to various embodiments, the RCT asks each CPU of active PAC 200 to begin migrating its respective tasks (as identified by the respective sitCPU of each CPU) to one or more CPUs of standby PAC 300. According to other embodiments, rather than asking, RCT instructs, or orders one or more of the CPUs of active PAC 200 to migrate its tasks .

[0047] At step 412, the migration process continues, as each CPU of active PAC 200 migrates its respective tasks to a corresponding CPU of standby PAC 300. For example,

according to one embodiment, tasks from CPU 202 migrate to CPU 302, from CPU 204 to CPU 304, and so on. At step 414, once all the tasks from each CPU of PAC 200 have migrated, standby PAC 300 notifies the CSP that migration is complete. The CSP then optionally restarts active PAC 200 as a standby PAC, and standby PAC 300 serves as an active PAC for the tasks which have migrated to it.

[0048] FIG. 5 is a more detailed flow diagram outlining the migration process associated with step 412 of the flow chart shown in FIG. 4. Prior to describing the various steps of the flow chart shown in FIG. 5, it should be noted that, as shown in FIG. 6, each active PAC 200 hosts several tasks which fall into at least one of the categories of critical tasks (CTs), restartable tasks (RTs), and migratable tasks (MTs). CTs are those tasks that are necessary for the basic operation of all PAC cards, and therefore, generally run on both active and standby PACs. Examples of CTs include a resource manager task that provides the CPU load, and a task that monitors the PAC card's temperature. It should also be noted that, in order to simplify the description of the present invention, the steps of the flow chart of FIG. 5 are described below only with respect to the migration of tasks from CPU 202 of active PAC 200 to CPU 302 of standby PAC 300. However, it will be understood that the various steps of the flow chart of FIG. 5 are generally carried out for each CPU of PAC 200 from which tasks are to be migrated.

[0049] Even before the migration of tasks is to take place from PAC 200 to PAC 300, the CTs 240 running on CPU 202 are generally also running on CPU 302 of PAC 300 (because CTs, in general, are necessary for the basic operation of all PAC cards). When this is the case, at step 502, standby PAC 300 reports to the CSP that it is in a ready, or operational state. If CTs 240 are not already running on CPU 302 of PAC 300 (e.g., because of a failure, or because PAC 300 is not yet operational), then a report indicating this is sent to the CSP at step 502, and according to various embodiments of the invention, a different standby PAC 300 is used for the migration process. According to various other embodiments of the present invention, a report will not be sent when CTs 240 are running on CPU 302 of PAC 300, but rather, will only be sent when they are not (e.g., to indicate that another PAC 300 must be used). According to yet other embodiments, a report will never be sent, regardless of whether CTs 240 are running on CPU 302 of PAC 300.

[0050] Next, at step 504 of FIG. 5, the RTs 250 running on CPU 202 of PAC 200 are migrated to CPU 302 of standby PAC 300 (as requested or ordered by the RCT). In

particular, once sitCPU 270 of CPU 202 shown in FIG. 6 determines which tasks are restartable, it terminates each RT 250 running on CPU 202, and sends a message to sitCPU 370 of CPU 302 (which is shown in FIG. 7) to restart each RT 250. At this point, the RTs 250 originally running on CPU 202 can be restarted on CPU 302, and can reacquire their state at PAC 300 by communicating with other components of the system. Thus, it is not necessary to send state information to standby PAC 300 from active PAC 200 for this migration to occur. Once the RTs 250 originally running on CPU 202 of PAC 200 have migrated to CPU 302 of standby PAC 300, as described above, according to various embodiments of the invention, these tasks no longer exist on CPU 202. According to other embodiments, however, these tasks may continue to exist on active CPU 202 (when they are not terminated by sitCPU 270).

[0051] Next, according to the principles of the present invention, the MTs 260 running on CPU 202 of PAC 200 are migrated to CPU 302 of standby PAC 300. The process for migrating MTs 260 differs from the process used to transfer RTs 250, as is now explained in detail.

[0052] At step 506, sitCPU 270 executes pre-migration for each MT 260 running on CPU 202. This includes invoking one or more checkpointing operating system (OS) kernel calls (e.g., using a standard software LINUX tool). For example, during this checkpointing operation, the OS saves the full state of each MT 260, which includes most (or all) of the memory controlled by the MTs 260, and the internal CPU registers used by the MTs 260.

[0053] At step 508, the checkpointing OS kernel call records state information associated with each MT 260, and sitCPU 270 of active PAC 200 transmits the state information to sitCPU 370 of standby PAC 300. Next, at step 510, sitCPU 370 of standby PAC 300 invokes a de-checkpointing OS kernel call for each MT 260 using the received state information. For example, during this de-checkpointing operation, the OS restores the full state of each MT 260 (including the respective memory and the internal CPU registers), and resumes executing the MTs 260 from the state at which the checkpointing operation took place.

[0054] Finally, at step 512, sitCPU 370 of standby PAC 300 executes post-migration, including, e.g., task-specific post-migration. For example, once the checkpointing and de-checkpointing operations performed by the OS are complete, it may be necessary to

reestablish any resources required on the new system PAC 300. This may include, among other things, reopening files that were previously being used (because the files are no longer "seen" once the MTs 260 have been moved), or reestablishing network connections with other internal or external components.

[0055] Once the RTs and MTs running on each CPU of PAC 200 have been migrated to respective CPUs of standby PAC 300 in the manner described above, the migration of tasks from PAC 200 to PAC 300 is complete, and notification is provided by standby PAC 300 to the CSP (as explained above in connection with step 414 of the flow chart of FIG. 4). FIG. 8 is a simplified illustration showing the tasks running on CPU 302 after PAC migration has completed, where CTs 340 correspond to CTs 240 of PAC 200, and RTs 350 and MTs 360 respectively correspond to migrated versions of RTs 250 and MTs 260 of PAC 200. Although not shown, the other CPUs of PAC 300 will also have similar tasks running thereon (corresponding to the tasks originally running on other CPUs of PAC 200) after migration from PAC 200 to PAC 300 is complete.

[0056] It will be understood that, while migration of different tasks from active PAC 200 to standby PAC 300 is described above in accordance with a particular sequence, other sequences are also contemplated according to the invention. For example, according to various embodiments of the present invention, MTs may be migrated from active PAC 200 to standby PAC 300 before RTs are started on standby PAC 300. The invention is not limited in this manner.

[0057] Moreover, while migration of tasks has been described above without the use of a standby PAC that mirrors an active PAC, it will be understood that the invention is not limited in this manner. For example, according to various alternate embodiments of the present invention, the technique known as lockstep or software mirroring may be used in conjunction with the above described process to facilitate migration. For example, a wrapper process may used so that each time an event occurs on certain active PACs, a message is transmitted to respective standby PACs so that the PACs execute in lockstep. The events may include, for example, messages and data. Time, data, and messages are coordinated, and memory is managed to help avoid instances in which the PACs react differently to the same event. For example, as a check, each of the PACs may be directed to report its time so that synchronization can analyzed. It will be understood that, using a lockstep or software mirroring technique, for example, there may be respective standby PACs 300 for certain (but

not all) active PACs 200, with a 1:1 ratio, while other standby PACs 300 which are not in lockstep with any active PAC 200 may be able to take over the tasks at any given time for one of a plurality of active PACs 200. The invention is not limited in this manner.

[0058] Other embodiments are within the scope of the following claims. For example, larger or smaller numbers of categories of tasks may be used. Tasks may be categorized ahead of time or on the fly. As explained above, different types of tasks may be migrated in different orders from that described above. Partial migration may be also performed. Moreover, migration may be performed from between active PACs 200. Additionally, for example, some or all of the critical tasks, restartable tasks, and migratable tasks being migrated from active PAC 200 to standby PAC 300 may remain on active PAC 200 even after migration is complete. Additionally, according to various embodiments, tasks from a single CPU of PAC 200 are migrated to multiple CPUs of PAC 300 (with, or without redundancy). According to various other embodiments, tasks from multiple CPUs of PAC 200 are migrated to a single CPU of PAC 300. These and other variations are within the scope of the present invention.

[0059] Therefore, although the invention has been described and illustrated in the foregoing illustrative embodiments other embodiments, it will be understood that extensions and modifications of the ideas presented above are comprehended and should be within the reach of one versed in the art upon reviewing the present disclosure. Accordingly, the scope of the present invention in its various aspects should not be limited by the examples presented above. The individual aspects of the present invention, and the entirety of the invention should be regarded so as to allow for such design modifications and future developments within the scope of the present. The present invention is limited only by the claims which follow.

What is claimed is:

1. A method for migrating tasks between computing devices in a computing system, the method comprising at least one of the sequential, sequence independent and non-sequential steps of:

receiving an indication that migration of one or more restartable and/or migratable tasks from a first computing device is to be initiated;

starting one or more restartable tasks on a second computing device corresponding to one or more restartable tasks running on the first computing device;

transmitting state information for one or more migratable tasks running on the first computing device to the second computing device; and

starting the one or more migratable tasks on the second computing device using the transmitted state information.

2. The method of claim 1, wherein one or more critical tasks running on the first computing device are already running on the second computing device before the receiving an indication.

3. The method of claim 1, wherein the migration of tasks from the first computing device to the second computing device is initiated for maintenance purposes.

4. The method of claim 1, wherein the migration of tasks from the first computing device to the second computing device is initiated following one or more failures by the first computing device.

5. The method of claim 1, wherein the receiving an indication comprises using one or more CLI commands of the computing system.

6. The method of claim 5, wherein the one or more CLI commands are issued by an administrator of the computing system.

7. The method of claim 1, wherein receiving an indication comprises manually removing the first computing device from the computing system.

8. The method of claim 7, wherein the manually removing the first computing device from the computing system comprises manipulating a trigger lock.

9. The method of claim 1, wherein receiving an indication comprises receiving a signal from a diagnostic system of the computing system.

10. The method of claim 1, further comprising determining that a migration of the one or more tasks from the first computing device to the second computing device should take place in response to receiving the indication.

11. The method of claim 1, wherein the first computing device comprises one or more central processing units and the second computing device comprises one or more central processing units.

12. The method of claim 1, wherein starting one or more restartable tasks on the second computing device comprises acquiring the respective states of the one or more restartable tasks by communication with components of the computing system other than the first computing device.

13. The method of claim 1, further comprising terminating the one or more restartable tasks running on the first computing device.

14. The method of claim 1, further comprising the steps of:

invoking a checkpointing call at the first computing device; and

recording state information for the one or more migratable tasks running on the first computing device.

15. The method of claim 14, wherein the checkpointing call is a checkpointing operating system kernel call.

16. The method of claim 1, wherein starting the one or more migratable tasks on the second computing device comprises invoking a de-checkpointing call using the received state information.

17. The method of claim 16, wherein the de-checkpointing call is a de-checkpointing operating system kernel call.

18. The method of claim 1, further comprising indicating that the migration of one or more restartable and/or migratable tasks from a first computing device to a second computing device is complete.

19. The method of claim 1, wherein at least one of the one or more critical, restartable, and/or migratable tasks remains on the first computing device following the migration of tasks to the second computing device.

20. The method of claim 1, wherein the first computing device and the second computing device are both electronic circuitry cards.

21. The method of claim 20, wherein the first computing device and the second computing device are both packet accelerator cards.

22. The method of claim 21, wherein the first computing device is an active packet accelerator card and the second computing device is a standby packet accelerator card.

23. The method of claim 21, wherein the first computing device and the second computing device are both active packet accelerator cards.

24. The method of claim 1, wherein the computing system is a communications computing system.

25. The method of claim 24, wherein the computing system is a wireless communications computing system.

26. The method of claim 1, wherein the second computing device is not running in lockstep with the first computing device prior to the migration of one or more restartable and/or migratable tasks from the first computing device to the second computing device.

27. A method for migrating tasks between computing devices in a computing system, the method comprising at least one of the sequential, sequence independent and non-sequential steps of:

initiating migration of a first task using a first technique for task migration between the first computing device and a second computing device; and

initiating migration of a second task using a second technique for task migration between the first computing device and the second computing device;

wherein the second computing device is not running in lockstep with the first computing device prior to the migration of the first and second tasks from the first computing device to the second computing device.

28. The method of claim 27, wherein migration of the first and second tasks from the first computing device to the second computing device is accomplished without rebooting the computing system.

29. The method of claim 27, further comprising receiving an indication that migration of one or more restartable and/or migratable tasks from a first computing device is to be initiated.

30. The method of claim 27, further comprising the steps of:

determining that migration is complete; and

treating the second computing device as an active computing device.

31. A system for migrating tasks between computing devices in a computing system, the system comprising:

means for initiating migration of a first task using a first technique for task migration between the first computing device and a second computing device; and

means for initiating migration of a second task using a second technique for task migration between the first computing device and the second computing device;

wherein the second computing device is not running in lockstep with the first computing device prior to the migration of the first and second tasks from the first computing device to the second computing device.

100

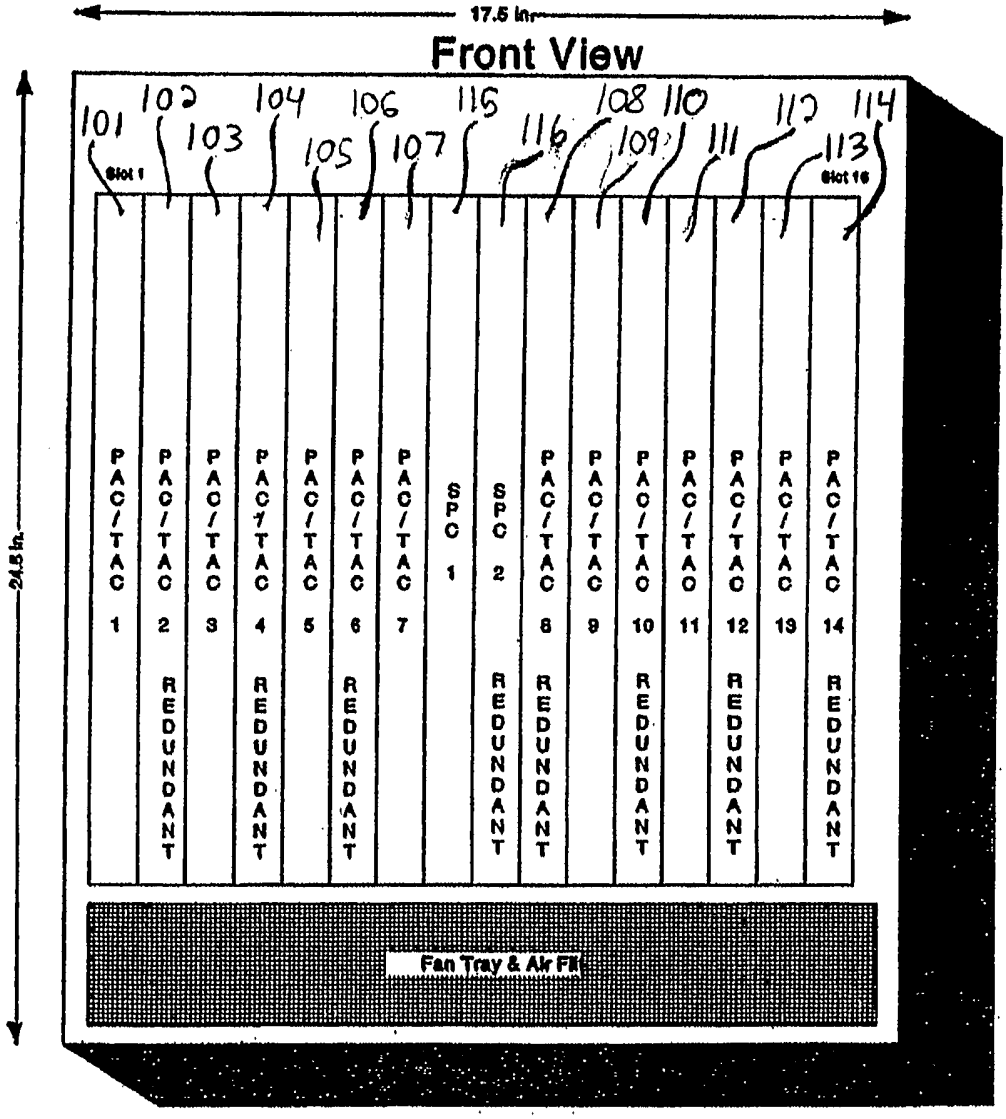

17.5 in.
Front View
24.5 in.
101
102
103
104
105
106
107
115
116
108
109
110
111
112
113
114
Slot 1
Slot 16
PAC/TAC 1
PAC/TAC 2 REDUNDANT
PAC/TAC 3
PAC/TAC 4 REDUNDANT
PAC/TAC 5
PAC/TAC 6 REDUNDANT
PAC/TAC 7
SPC 1
SPC 2 REDUNDANT
PAC/TAC 8 REDUNDANT
PAC/TAC 9
PAC/TAC 10 REDUNDANT
PAC/TAC 11
PAC/TAC 12 REDUNDANT
PAC/TAC 13
PAC/TAC 14 REDUNDANT
Fan Tray & Air Fil

FIG. 1

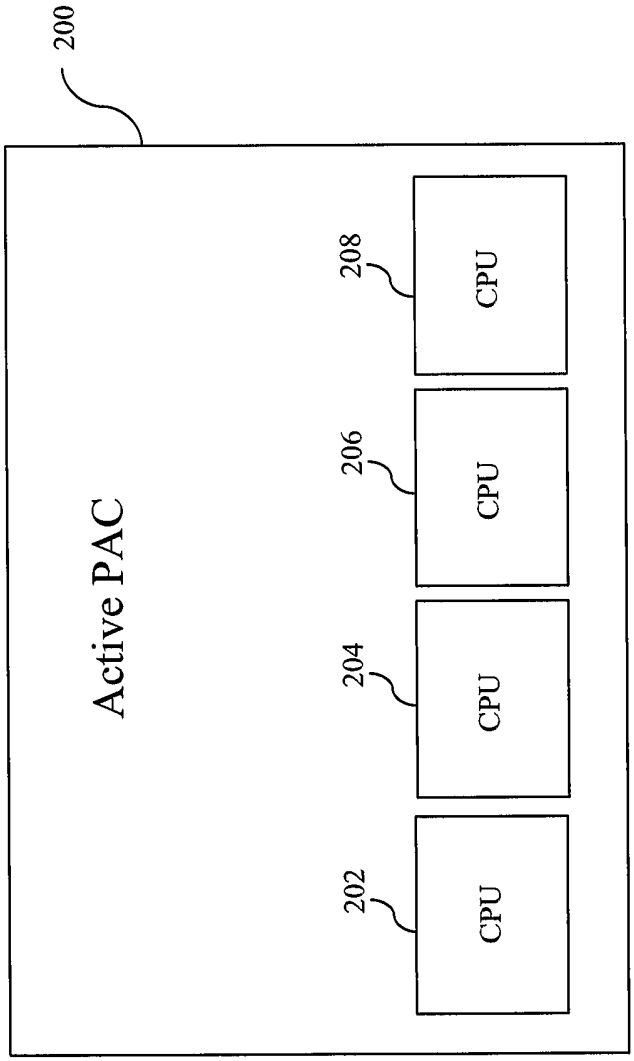


FIG. 2

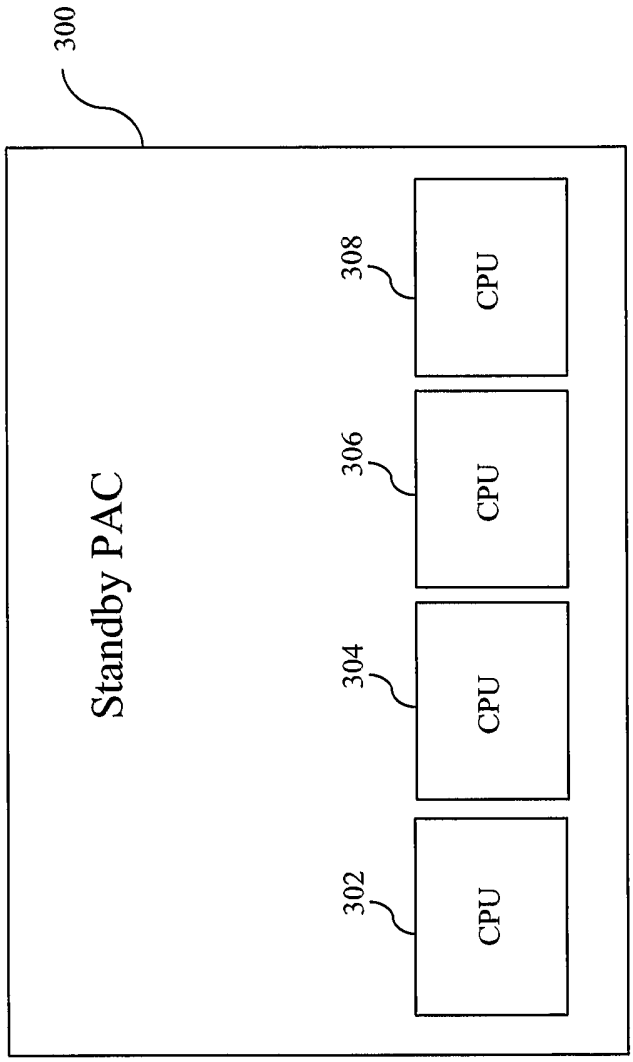


FIG. 3

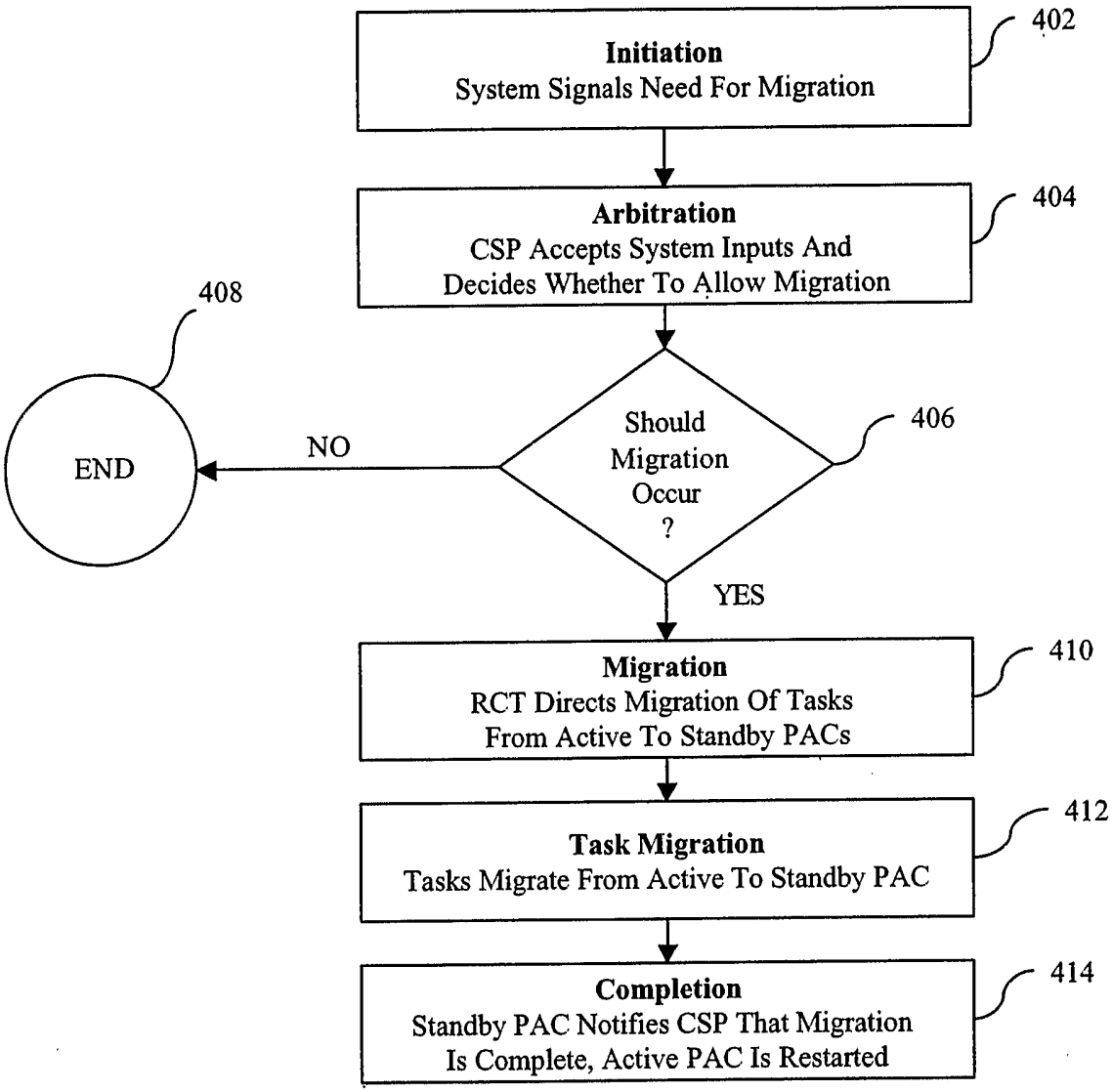

Initiation
System Signals Need For Migration
402
Arbitration
CSP Accepts System Inputs And
Decides Whether To Allow Migration
404
408
END
NO
Should
Migration
Occur
?
406
YES
Migration
RCT Directs Migration Of Tasks
From Active To Standby PACs
410
Task Migration
Tasks Migrate From Active To Standby PAC
412
Completion
Standby PAC Notifies CSP That Migration
Is Complete, Active PAC Is Restarted
414

FIG. 4

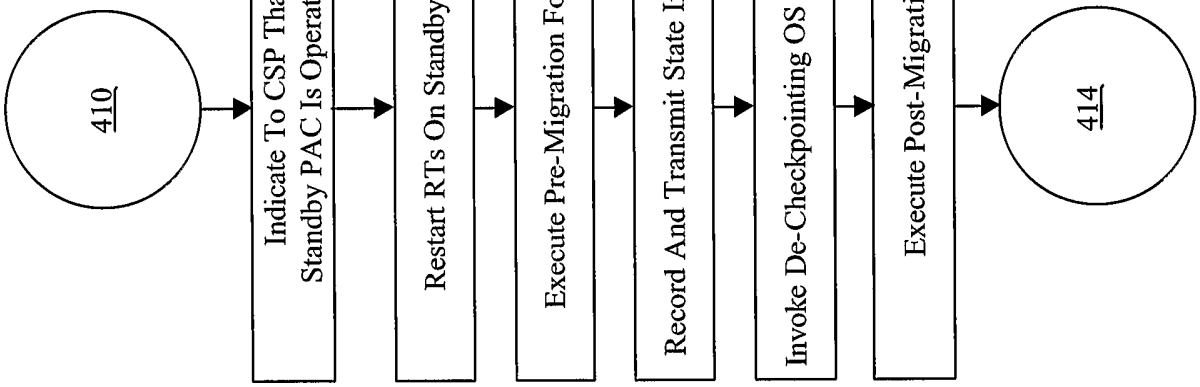

410
502
Indicate To CSP That Standby PAC Is Operational
504
Restart RTs On Standby PAC
506
Execute Pre-Migration For MTs
508
Record And Transmit State Information
510
Invoke De-Checkpointing OS Kernel Call
514
Execute Post-Migration
414

FIG. 5

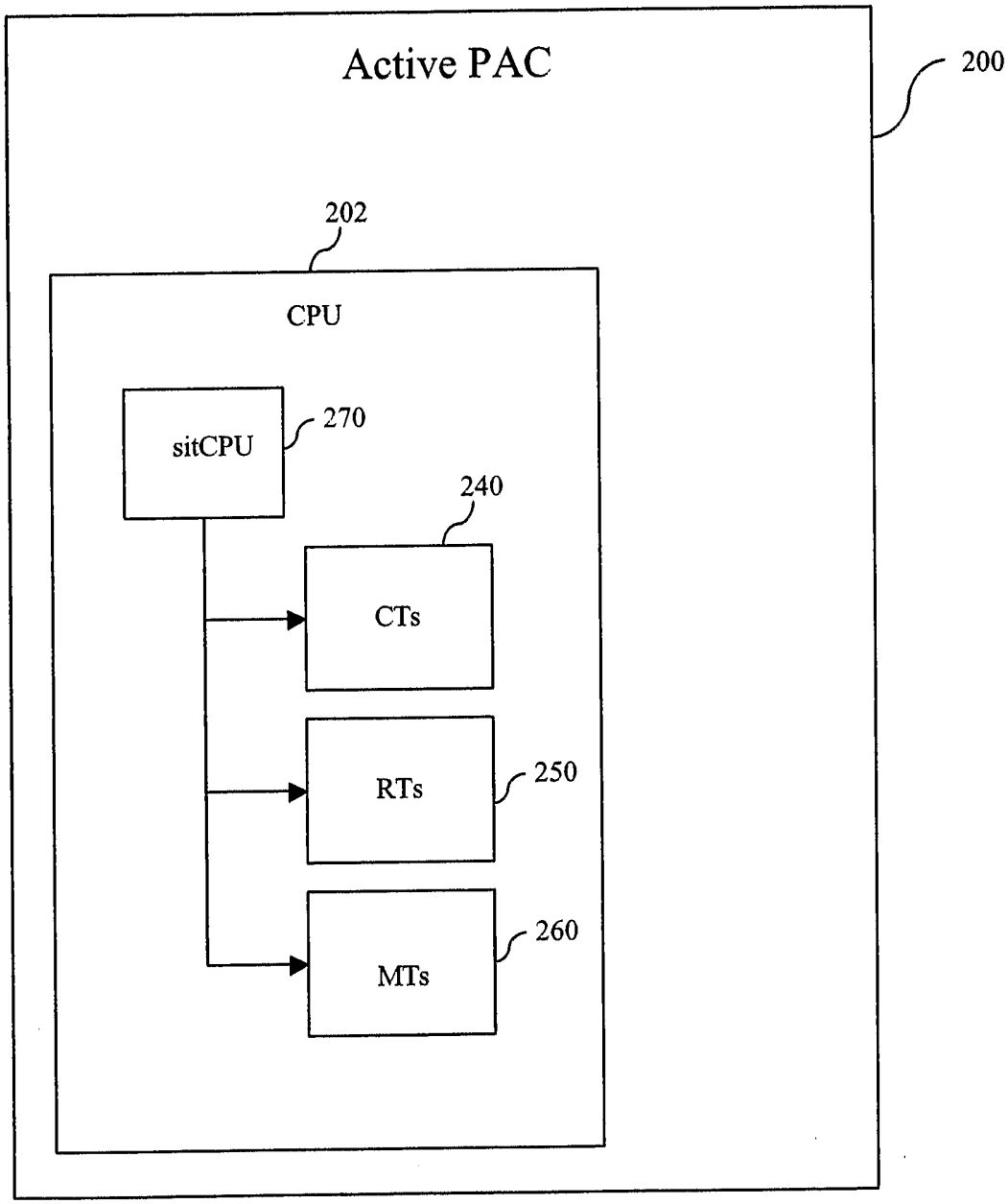


FIG. 6

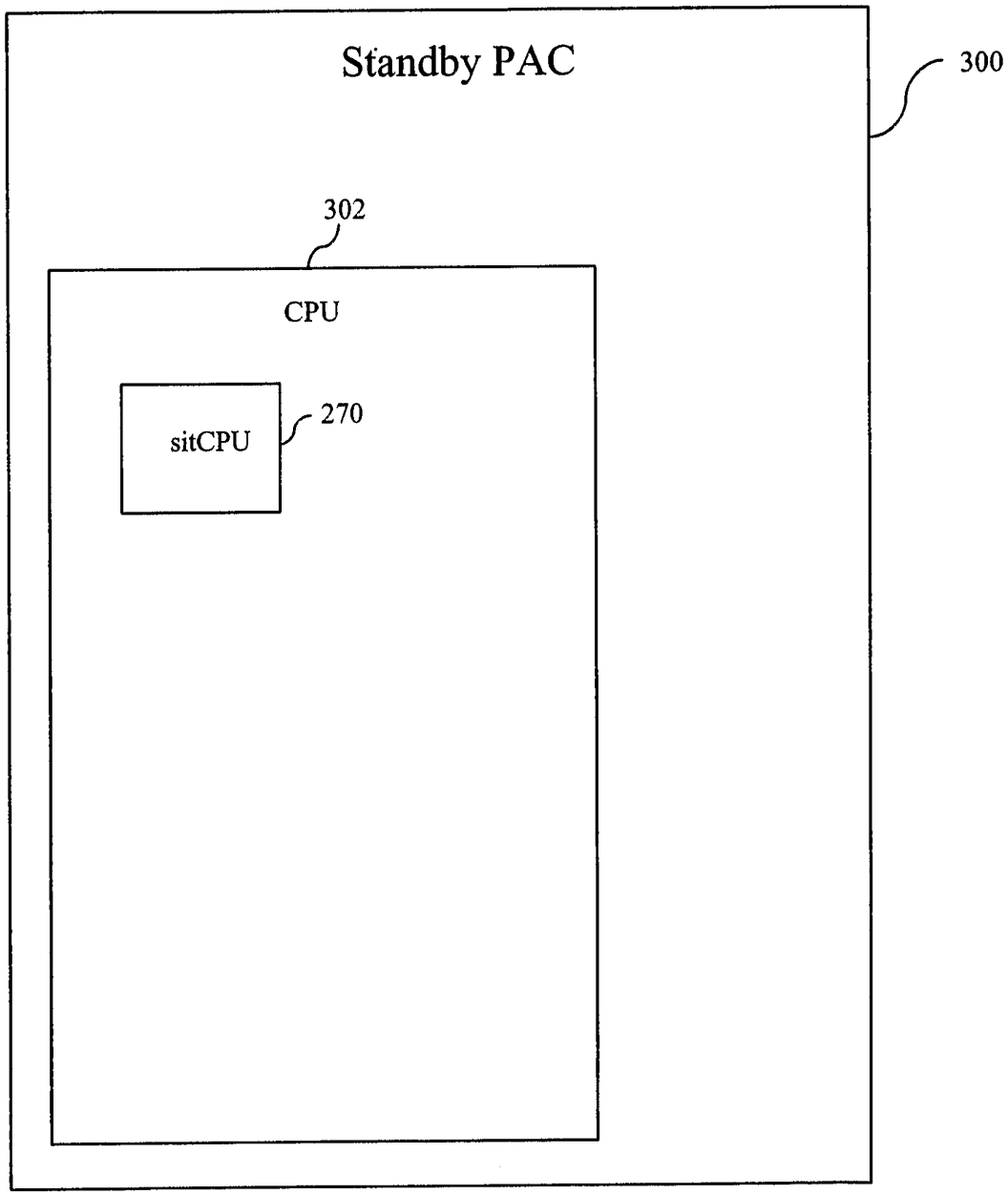


FIG. 7

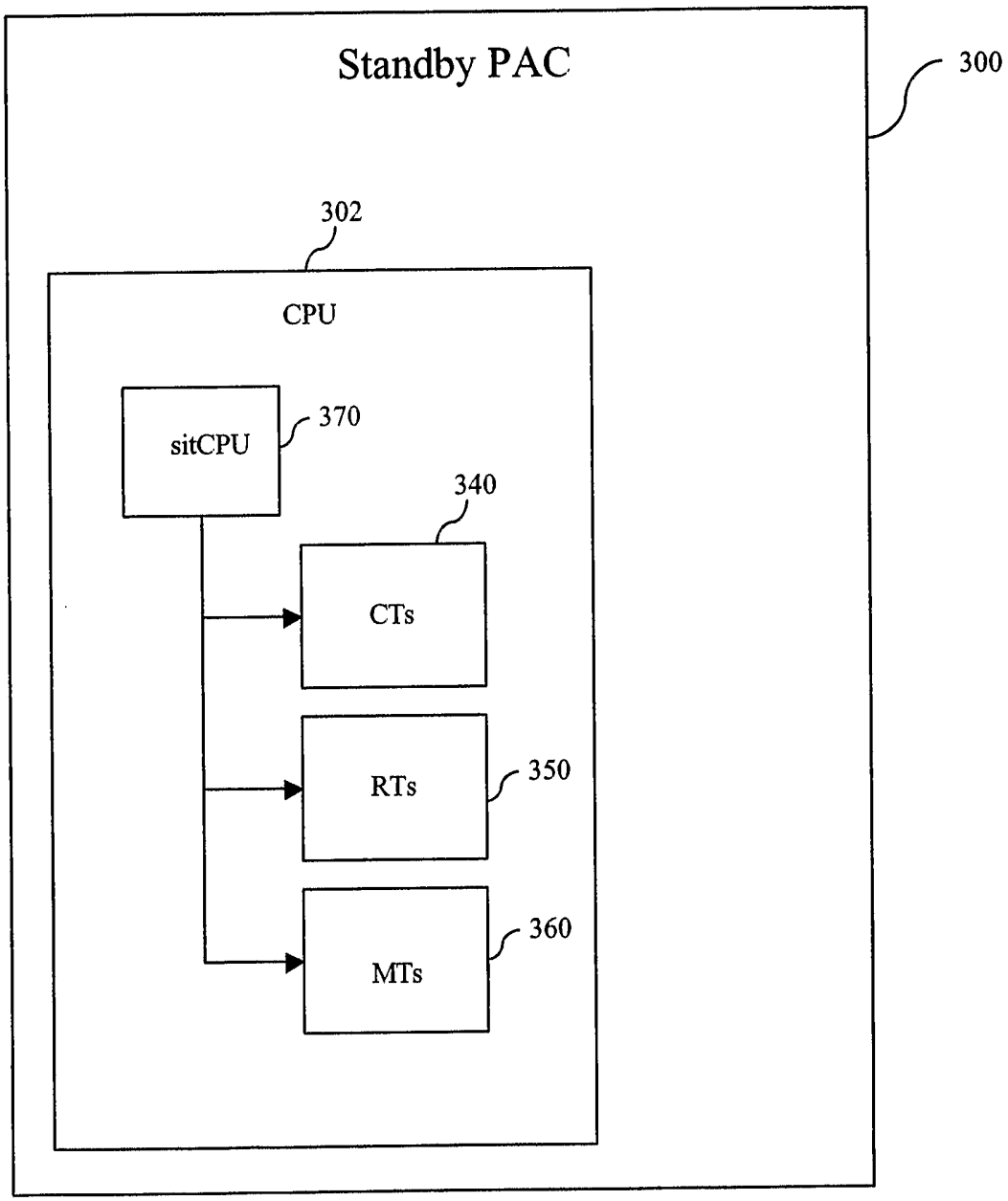


FIG. 8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US05/13122

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 9/46
US CL : 718/102
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)
U.S. : 718/102,106-108,100; 719/317; 709/202

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

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6,085,086 A (LA PORTA et al) 4 July 2000, col. 3, line 53 - col. 12,l ine 4.	1-31
A	US 6,769,121 B1 (KOYAMA et al) 27 July 2004, col. 6, line 13 - col. 16, line 34.	1-31
A	US 6,442,663 B1 (SUN et al) 27 August 2002, col. 13, line 15 - col. 29, line 44.	1-31
A	US 6,415,315 B1 (GLASS) 2 July 2002, col. 3, line 44 - col. 13, line 15.	1-31

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

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent 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
14 September 2005 (14.09.2005)

Date of mailing of the international search report
17 OCT 2005

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. (703) 305-3230

Authorized officer
for Michelle L. Esser
Meng-Ai An

Telephone No. 5712722100

Form PCT/ISA/210 (second sheet) (April 2005)