

(19) World Intellectual Property Organization
International Bureau



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

PCT

(10) International Publication Number
WO 2006/028521 A1

(51) International Patent Classification⁷: **G06F 9/46**

(21) International Application Number:
PCT/US2005/013126

(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): **STAR-
ENT NETWORKS, CORP.** [US/US]; 30 International
Place, Tewksbury, MA 01876 (US).

(72) Inventor; and

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

(74) Agents: **KANABE, George, L.** et al.; Wilmer Cutler Pick-
ering 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.

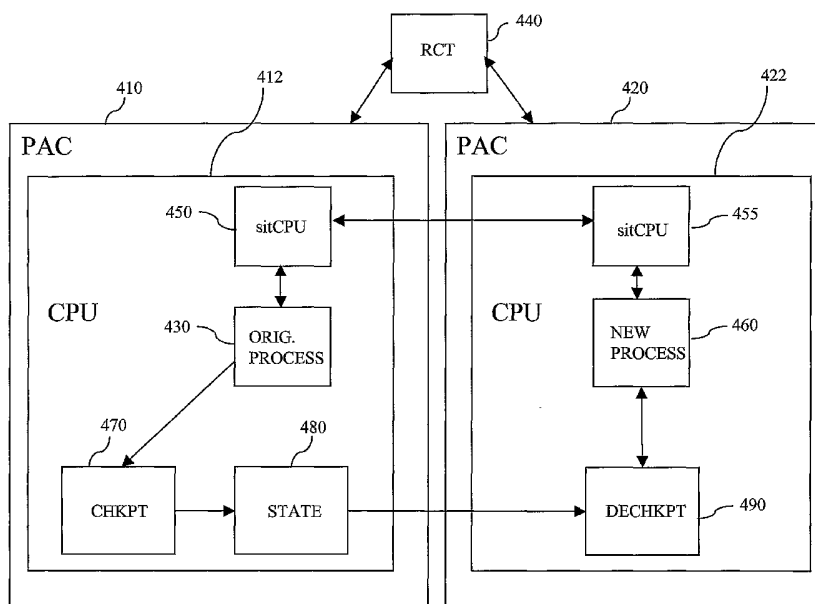
(84) Designated States (unless otherwise indicated, for every
kind of regional protection available): ARIPO (BW, GH,
GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,
ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),
European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,
FR, GB, GR, HU, IE, IS, IT, LT, 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 "Guid-
ance Notes on Codes and Abbreviations" appearing at the begin-
ning of each regular issue of the PCT Gazette.

(54) Title: PROCESS CHECKPOINTING AND MIGRATION IN COMPUTING SYSTEMS



(57) Abstract: Methods and systems are provided for process checkpointing and migration in computing systems, such as commu-
nications systems. According to various embodiments, a first standard operating system call is invoked to checkpoint a first process
(430) running on a computing device (410) to produce checkpointing results (480). At least a portion of the checkpointing results are
transmitted to a second computing device (420). A second standard operating system call is invoked to use the transmitted check-
pointing results to de-checkpoint (490) a second process (460) running on the second computing device.

WO 2006/028521 A1

PROCESS CHECKPOINTING AND MIGRATION IN COMPUTING SYSTEMS

Cross-Reference to Related Applications

[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 process checkpointing and migration 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 of many applications and/or the efficient execution of many processes (also called tasks or operations). For example, heavy demand for CPU processing is common in wireless (mobile) communications systems. Moreover, the need for fast, powerful, and flexible computing systems is especially present where multiple applications and/or processes 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 in complexity. 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 processes 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 process will have an optimum computing configuration.

[0005] Generally, in distributed network computing, adaptability of process assignment is desirable for high throughput and resource utilization, especially for a long-running application. A process is a piece of a program in execution. It represents a job assigned to a computer (also called a computing device or machine) during the execution of a program. A program may comprise one or more processes running on single or multiple computers.

[0006] The software and hardware on a computer create a distinct computing platform. Typically, in the development of a user application, an executable file is provided via a compiler for that particular computer. The executable file contains a sequence of machine instructions in form of platform-specific binary code. One or more processes are created on a computer before an executable file can be executed on a computer, so that the operating system of the computer can load those instructions into the computer's main memory and assign the instructions to the central processing unit (CPU).

[0007] In building an executable file, a programmer can write a program (or source code) in the form of a high-level computer language such as C, C++, or FORTRAN, and pass it to a compiler for that language. A program comprises a global data definition area and a description of functions. Each function description comprises of parameter variable declarations, local variable declarations, and programming language statements. The compiler translates the program source code into the platform-specific binary code, and stores them in an executable file. During compilation, the compiler can also optimize the machine instructions according to specific features of the computing platform. At runtime, the operating system loads the executable file into the computer's memory. The loaded executable file is then ready to be executed by the CPU and is recognized as one or more processes or tasks.

[0008] It can be useful to be able to restart a program after it has halted for one or more reasons. To facilitate recovery of a program, especially a long running program, intermediate results or states of the program can be taken at particular intervals, which is referred to as checkpointing the program. Checkpointing enables the program to be restarted from the last checkpoint, rather than from the beginning.

[0009] Process migration, where the execution of a process is suspended on one computer and then resumed on another computer (or, for example, on different CPUs

associated with the same computer), is a mechanism to adapt process and resource assignment. The process being migrated can be transferred, for example, via direct network-to-network communication (network migration), or file migration. Applications of process migration include load distribution, which involves migrating processes from overloaded computers to underloaded computers to make use of otherwise unused computing cycles; fault resilience, which involves migrating processes from computers that may be experiencing partial or total failure; resource sharing, which involves migrating processes to computers with special hardware or other unique resources such as databases or peripherals required for computations; and data access locality, which involves migrating processes towards the source of the data. In addition to mobile computing, both clustered network computing and ubiquitous computing, for example, often demand efficient process migration.

[0010] The mobile agent approach to process migration is an alternative to "true" process migration. Mobile agents are implemented on top of safe or interpreted languages, such as Java, which are more secure and promising for certain applications. In these languages, the interpreter acts as a virtual machine to create an artificial homogeneous environment. However, these languages are less powerful, slow, and require rewrites of existing software.

[0011] Checkpointing for process migration has been developed primarily for fault tolerance, and involves transferring and restarting one or more checkpointed processes on properly functioning machines. Checkpointing requires access to file systems and roll backs to a consistent global state in parallel processes. Typically, checkpointing a process requires the additional step of saving the data contents of a process to a file periodically. Later, when recovery is needed, data from the checkpointed file is read and restored in a new process to resume execution of the application.

[0012] With process migration, all data necessary for future execution of the process is first collected and then restored in the data segment of the new process on another machine. Accordingly, there is a need for efficient methods to recognize, collect, and restore data contents of a process. While efforts have been made to collect and restore the data contents of a process in a computer in a time-sensitive manner, prior migration methods and systems suffer from various problems (e.g., inefficiencies) as will be appreciated by persons versed in the art.

[0013] Accordingly, it is desirable to provide methods and systems for use in process checkpointing and migration in computing systems, such as communications systems, that alleviate at least some of the problems associated with existing methods and systems associated with process checkpointing and migration.

Summary of the Invention

[0014] In accordance with the principles of the present invention, methods and systems are provided for use in process checkpointing and migration in computing systems, such as communications systems.

[0015] According to various embodiments of the invention, implementations of the invention may provide one or more of the following advantages. For example, duties of a process can be efficiently transferred from one computing device (e.g., an accelerator card) to another computing device (e.g., another accelerator card). Additionally, for example, a supervisor task can monitor the status of migration of a process.

[0016] According to one embodiment, the invention provides a method for use in process checkpointing and migration in a computing system, where the method includes invoking a first standard operating system call to checkpoint a first process running on a first computing device to produce checkpointing results, transmitting at least a portion of the checkpointing results to a second computing device, and invoking a second standard operating system call to use the transmitted checkpointing results to de-checkpoint a second process running on the second computing device.

[0017] According to a second embodiment, the invention provides a system for checkpointing and migrating processes between a first computing device and a second computing device in a computing system, where the system includes means for invoking a first standard operating system call to checkpoint a first process running on the first computing device to produce checkpointing results, means for transmitting at least a portion of the checkpointing results to the second computing device, and means for invoking a second standard operating system call to use the transmitted checkpointing results to de-checkpoint a second process running on the second computing device.

[0018] According to a third embodiment, the invention provides a system for checkpointing and migrating processes between a first computing device and a second

computing device in a computing system, where the system includes a first computing device on which a first process is running, and a second computing device on which a second process is running, wherein at least a portion of the checkpointing results produced by invoking a first standard operating system call to checkpoint the first process are transmitted to the second computing device, and wherein a second standard operating system call uses the checkpointing results received by the second computing device to de-checkpoint the second process.

Brief Description of the Drawings

[0019] 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:

[0020] FIG. 1 is a simplified illustration of a chassis in a computing system that includes multiple PACs among which migration of processes according to the principles of the present invention may be accomplished;

[0021] FIG. 2 is a simplified illustration of an active packet accelerator card (PAC) that includes four CPUs from which migration of processes according to the principles of the present invention may be accomplished;

[0022] FIG. 3 is a simplified illustration of a standby PAC that includes four CPUs to which migration of processes according to the principles of the present invention may be accomplished;

[0023] FIG. 4 is a simplified illustration showing two PACs that may be involved in the checkpointing and migration of one or more processes in accordance with the principles of the present invention; and

[0024] FIG. 5 is a flow chart illustrating the steps performed according to one embodiment of the present invention in the checkpointing and migration of processes from a first computing device to a second computing device in a computing system.

Detailed Description of the Invention

[0025] Methods and systems are provided for use in process checkpointing and migration in computing systems, such as communications systems. It will be understood that certain features that 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 checkpointing and migration of processes in a communications computing system, the invention is not limited in this manner.

[0026] According to various embodiments of the present invention described in greater detail below, a process or task is migrated from a first computing device to a second computing device by creating a matching process that runs on the second computing device, invoking a standard operating system call on the first computing device to save data reflecting the state of the process, transmitting at least some of the saved data to the second computing device, and invoking another standard operating system call on the second computing device to use the transmitted data to cause the matching process to assume the state of the process and execute accordingly. For example, in a particularly time-sensitive implementation in a wireless communications system, memory and register contents of a process on one communications computing device may be restored as the memory and register contents of a matching process on another communications computing device.

[0027] As known by persons versed in the art, computing systems generally use multiple computing devices, such as electronic circuitry cards, for handling various processes. 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 checkpointing and migration of processes 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 checkpointing and migration of processes according to the invention may be accomplished in connection with any suitable type of computing devices.

[0028] FIG. 1 shows a chassis 100 in a computing system that includes multiple PACs 101-114 among which migration of processes 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 (SPEC) 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 SPEC (or RPC) 116.

[0029] PACs such as those shown in FIG. 1 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 processes or tasks that are running when an active PAC fails. The act of transferring processes 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 invention is not, however, limited by the number of CPUs present in active PAC 200 or standby PAC 300.

[0030] 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 one or more processes or 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 processes running on the respective CPU. For example, the sitCPU of CPUs 202, 204, 206 and 208 (of active PAC 200) may maintain a current list of all the other processes running on the respective CPUs, their process ID numbers and process types (e.g., critical, restartable, or migratable), etc.

[0031] Generally speaking, two types of PAC migrations can occur: graceful and ungraceful. In the case of a graceful migration, processes 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, for maintenance purposes. In the past, graceful migrations have required that the second PAC (to which processes 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 processes 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.

[0032] FIG. 4 is a simplified illustration showing PACs 410 and 420 that may be involved in the checkpointing and migration of one or more processes in accordance with the principles of the present invention, e.g., from a CPU 412 of PAC 410 to a CPU 422 of PAC 420, as will now be explained in greater detail with reference to the flow chart shown in FIG. 5.

[0033] FIG. 5 is a flow chart outlining several steps involved in the checkpointing and migration of processes according to various embodiments of the invention. At step 502, migration of a process 430 of PAC 410 is initiated (e.g., because PAC 410 is to become disabled), and a Recovery Control Task (RCT) 440 (such as developed by Starent Networks Corporation of Tewksbury, MA) or a similar task is used to determine which card to migrate from (in this example, PAC 410) and which card to migrate to (in this example, PAC 420). In general, RCT 440 helps to coordinate process checkpointing and migration of an individual process or task, and may be running on a SPEC (or RPC) associated with the chassis (e.g., chassis 100) that includes PACs 410 and 420.

[0034] It is noted that the migration of process 430 may be initiated from any one (or more than one) of several system inputs. For example, the administrator of the computing system in which PACs 410 and 420 operate may want to service PAC 410 while ensuring that any processes currently running on it are not lost. By using, for example, system Command Line Interface (CLI) commands, the administrator can indicate that one or more processes are to be migrated from PAC 410 to PAC 420. The administrator can also initiate a migration, for example, by manually removing PAC 410 from the chassis in which it resides (e.g., chassis 100 shown in FIG. 1). In order to remove PAC 410, the administrator generally manipulates a physical device called a trigger lock, which sends a signal indicating that a migration is necessary for PAC 410. The migration of process 430 may also be initiated, for example, if a diagnostic system senses that PAC 410 is experiencing one or more failures and

needs to be shut down. It will be understood by persons versed in the art that migration of a process may be initiated by other means as well, and that the invention is not limited to the particular examples provided above.

[0035] At step 504, RCT 440 passes information regarding the process 430 to be migrated, and the PAC 420 to which migration is to take place, to sitCPU 450 of CPU 412 (the CPU on which process 430 is running). Next, at step 506, sitCPU 450 notifies and sends migration information to sitCPU 455 of CPU 422 (the CPU to which process 430 is being migrated) regarding the migration to take place, and sitCPU 455 creates a new process 460 corresponding to original process 430. Although not required, as explained above and shown in FIG. 4, new process 460 may run another PAC (in this case, PAC 420).

[0036] Once new process 460 has been started and is ready to receive, e.g., a TCP connection, at step 508, sitCPU 450 of CPU 412 sends a message to process 430 instructing (or requesting) process 430 to being a migration. In response to the received request or instruction from sitCPU 450, at step 510, original process 430 executes pre-migration including, e.g., invoking a checkpointing operating system (OS) kernel call 470 (e.g., using a standard software LINUX tool). At step 512, the checkpointing OS kernel call records (saves) state information 480. The state information being recorded at step 512 may include, for example, the state of CPU registers being used by process 430. In addition, for example, the state information may include some (or all) of the memory or stack controlled by original process 430, shared memory, and/or dynamic load libraries. It will be understood by persons versed in the art that the foregoing list is not necessary exhaustive, but rather is illustrative of the items that may be included in the state information being recorded at step 512.

[0037] At step 514, original process 430 transmits the saved state information 480 to new process 460 (e.g., using RCT 440 or another task). The state information 480 may be transmitted, e.g., over a Transmission Control Protocol (TCP) connection or over a StarChannel link. The invention is not limited in this manner.

[0038] Upon receiving the state information transmitted by process 430, at step 516, new process 460 invokes a de-checkpointing OS kernel call 490 using the received state information 480. For example, during this de-checkpointing operation, the OS restores the full state of original process 430 to new process 460 (including, e.g., the respective memory and the internal CPU registers).

[0039] At step 518, new process 450 sends a message to sitCPU 450 indicating that migration is completed and executes post-migration including, e.g., process-specific post-migration. For example, the process-specific post-migration being executed by new process 460 may include closing, e.g., the TCP connection over which the state information 470 was transmitted, and fixing file descriptors and signal information, e.g., to refer to new process 460 rather than original process 430. Additionally, for example, if original process 430 had opened a socket, the interface to the socket may be modified so that new process 460 is in communication with the socket. In the case of migrating a Telnet process or task, for example, the TCP socket being used may be released in pre-migration and a new TCP socket may be started in post-migration for use with the new (migrated) Telnet process. Once new process 450 has executed post-migration, it begins to execute normally on PAC 420.

[0040] At step 520, original process 430 informs RCT 440 that migration is complete, and optionally terminates. Alternatively, original process 430 may be kept running, e.g., for load balancing or load sharing purposes.

[0041] Other embodiments are also within the scope of the present invention. For example, according to various embodiments of the invention, original process 430 has registered with a migration library to publish the process' mechanism for receiving a migration notification. Additionally, for example, the steps described above in connection with the flow chart of FIG. 5 may be executed in other sequences.

[0042] Additionally, according to various embodiments, processes associated with a single CPU of PAC 410 are migrated to multiple CPUs associated with PAC 420 (with, or without redundancy). According to various other embodiments, processes from multiple CPUs of PAC 410 are migrated to a single CPU of PAC 420. These and other variations are within the scope of the present invention. In addition, one or more processes can be migrated from one or more CPUs of PAC 410 to one or more other CPUs of PAC 410 (i.e., processes may be migrated between or among different CPUs in the same PAC in accordance with various embodiments of the invention). Additionally, migration of a process in accordance with the principles of the present invention does not necessarily involve migration of the entire process. Rather, according to various embodiments, it is contemplated that only a portion of a process is migrated. Moreover, while migration may occur from a active PAC to

a standby PAC, the invention is not limited in this manner. For example, migration may be performed from one active PAC to another active PAC.

[0043] 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 use in process checkpointing and migration in a computing system, the method comprising:
 - invoking a first standard operating system call to checkpoint a first process running on a first computing device to produce checkpointing results;
 - transmitting at least a portion of the checkpointing results to a second computing device; and
 - invoking a second standard operating system call to use the transmitted checkpointing results to de-checkpoint a second process running on the second computing device.
2. The method of claim 1, wherein the computing system is a communications system, and wherein the first and second computing devices are packet accelerator cards.
3. The method of claim 1, further comprising receiving an indication that the first process is to be migrated before the invoking a first standard operating system call.
4. The method of claim 3, wherein the indication is received in response to an action by an administrator of the computing system.
5. The method of claim 3, wherein the indication is received by a diagnostic system of the computing system.
6. The method of claim 1, further comprising invoking a recovery control task (RCT), wherein the RCT identifies the second computing device on which the second process will run.
7. The method of claim 6, further comprising indicating to the RCT, by the first process, that migration is complete.
8. The method of claim 1, further comprising creating the second process running on the second computing device in response to an indication that the first process is to be migrated.
9. The method of claim 1, wherein the checkpointing results comprises state information associated with the first process recorded by the first standard operating system call.
10. The method of claim 9, wherein the state information comprises the state of any central processing units being used by the first process.
11. The method of claim 9, wherein the state information comprises the memory or stack controlled by the first process.

12. The method of claim 9, wherein the transmitting at least a portion of the checkpointing results comprises transmitting at least a portion of the state information recorded by the first standard operating system call.
13. The method of claim 12, wherein the at least a portion of the state information is transmitted using at least one of a Transmission Control Protocol (TCP) connection and a StarChannel link.
14. The method of claim 1, wherein the transmitted checkpointing results comprises state information associated with the first process recorded by the first standard operating system call, and wherein the state information is used by the second standard operating system call to restore the full state of the first process to the second process.
15. The method of claim 1, further comprising executing post-migration by the second process.
16. The method of claim 15, wherein the executing post-migration by the second process comprises closing a TCP connection.
17. The method of claim 15, wherein the executing post-migration by the second process comprises modifying at least one of a file descriptor and signal information to refer to the second process.
18. The method of claim 15, wherein the executing post-migration by the second process comprises modifying the interface with a socket opened by the first process such that the second process is in communication with the socket.
19. A system for checkpointing and migrating processes between a first computing device and a second computing device in a computing system, the system comprising:
 - means for invoking a first standard operating system call to checkpoint a first process running on the first computing device to produce checkpointing results;
 - means for transmitting at least a portion of the checkpointing results to the second computing device; and
 - means for invoking a second standard operating system call to use the transmitted checkpointing results to de-checkpoint a second process running on the second computing device.
20. A system for checkpointing and migrating processes between a first computing device and a second computing device in a computing system, the system comprising:

a first computing device on which a first process is running; and

a second computing device on which a second process is running, wherein at least a portion of the checkpointing results produced by invoking a first standard operating system call to checkpoint the first process are transmitted to the second computing device, and wherein a second standard operating system call uses the checkpointing results received by the second computing device to de-checkpoint the second process.

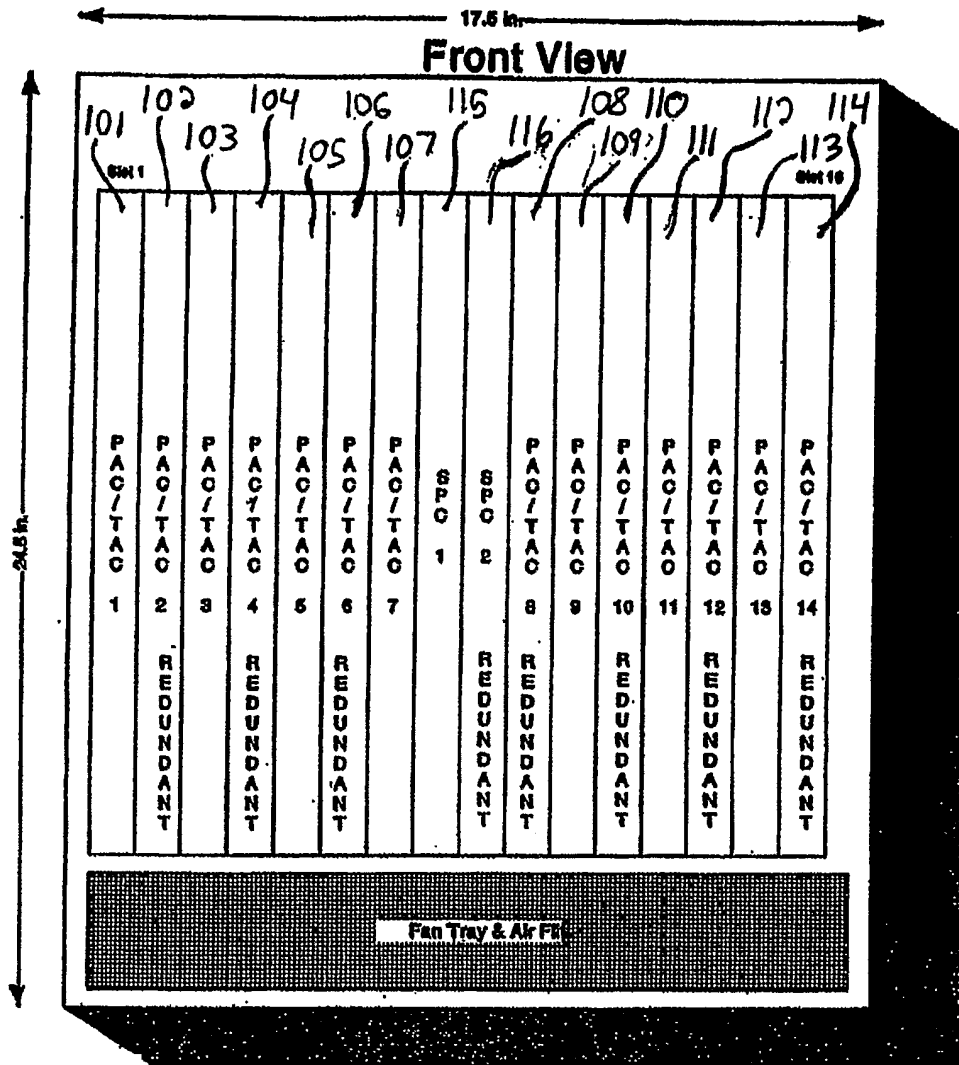
100

FIG. 1

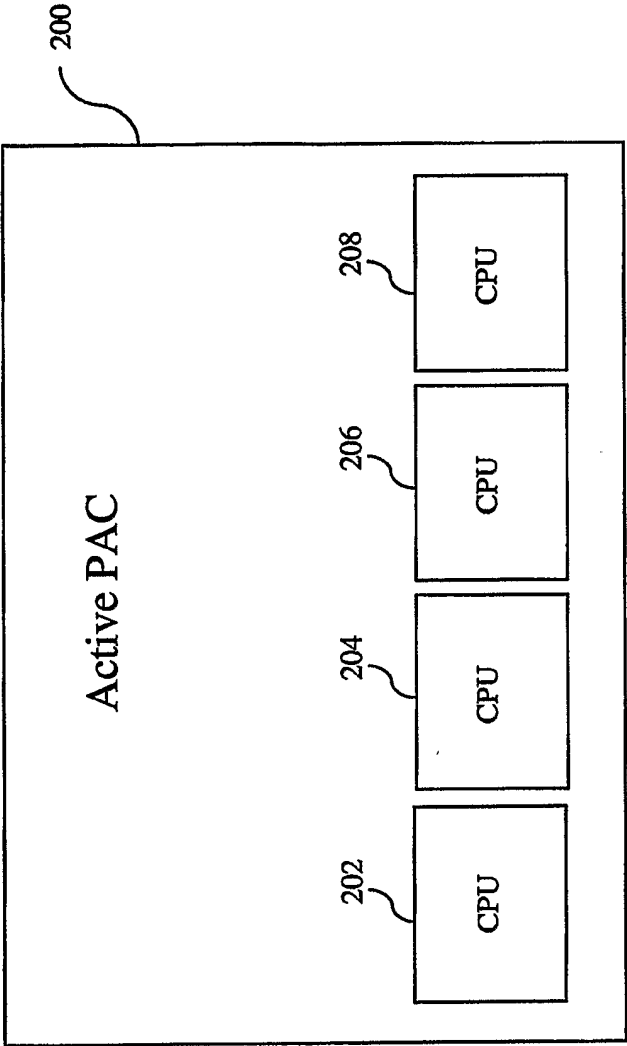


FIG. 2

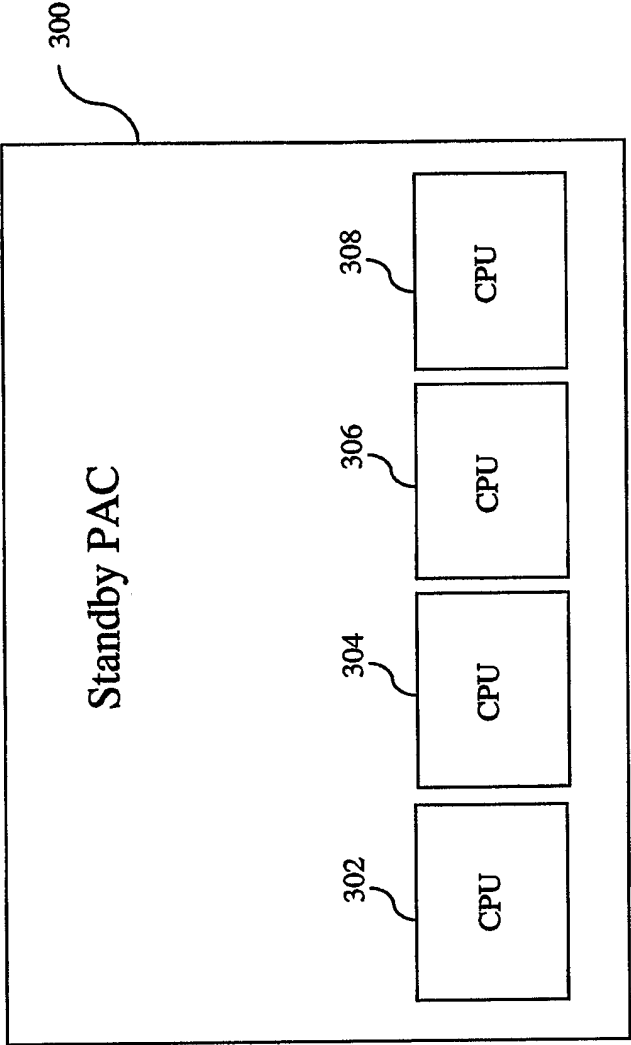


FIG. 3

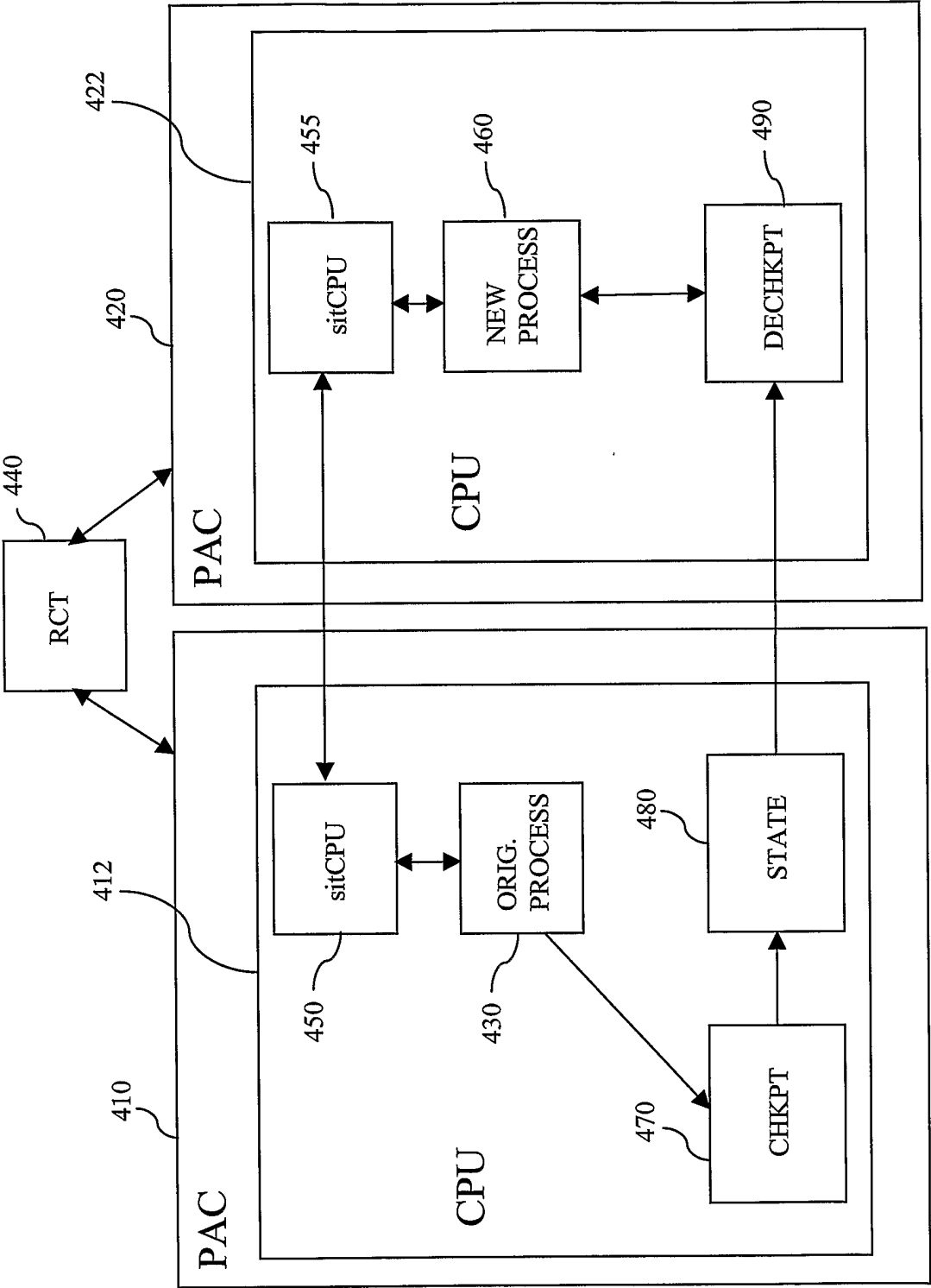


FIG. 4

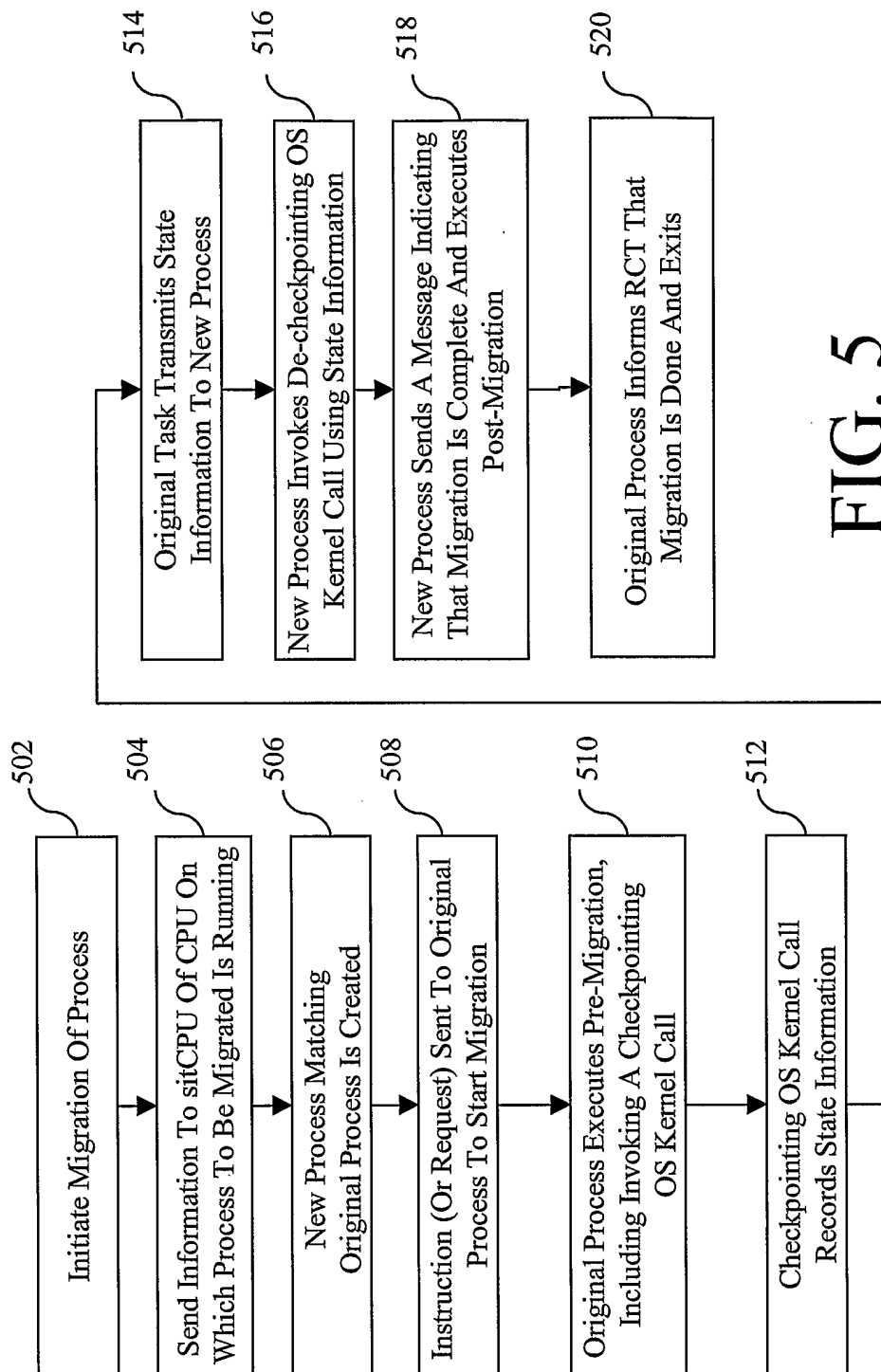


FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US05/13126

A. CLASSIFICATION OF SUBJECT MATTER IPC(7) : G06F 9/46 US CL : 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) U.S. : 718/100, 101, 102, 106-108; 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, line 4.	1-20
A	US 6,161,219 A (RAMKUMAR et al) 12 December 2000, col. 7, line 41 - col. 27, line 27.	1-20
A	US 6,161,193 A (GARG et al) 12 December 2000, col. 2, line 66 - col. 9, line 67.	1-20
A	US 5,893,912 A (FREUND et al) 13 April 1999, col.3, line 41 - col. 18, line 44.	1-20
☐ 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	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"E"	earlier application or patent published on or after the international filing date	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family
"P"	document published prior to the international filing date but later than the priority date claimed	
Date of the actual completion of the international search 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 <i>for Michelle L. Esen</i> Meng-Ai An Telephone No. 5712722100