US007484208B1

(12) **United States Patent**
Nelson

(10) **Patent No.: US 7,484,208 B1**
(45) **Date of Patent: Jan. 27, 2009**

(54) **VIRTUAL MACHINE MIGRATION**

(76) Inventor: **Michael Nelson**, 888 Forest La., Alamo, CA (US) 94507

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 425 days.

(21) Appl. No.: **10/319,217**

(22) Filed: **Dec. 12, 2002**

(51) **Int. Cl.**
G06F 9/455 (2006.01)
G06F 12/00 (2006.01)

(52) **U.S. Cl.** **718/1**; 711/6

(58) **Field of Classification Search** 718/1; 709/224; 711/153, 6
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,075,938	A *	6/2000	Bugnion et al.	703/27
6,698,017	B1 *	2/2004	Adamovits et al.	717/168
2004/0010787	A1 *	1/2004	Traut et al.	718/1

OTHER PUBLICATIONS

Theimer, Marvin M., Lantz, Keith A, and Cheriton, David R., "Preemptable Remote Execution Facilities for the V-System," Association for Computing Machinery, pp. 2-12, Dec. 1985.

* cited by examiner

Primary Examiner—Li B Zhen

(57) **ABSTRACT**

A source virtual machine (VM) hosted on a source server is migrated to a destination VM on a destination server without first powering down the source VM. After optional pre-copying of the source VM's memory to the destination VM, the source VM is suspended and its non-memory state is transferred to the destination VM; the destination VM is then resumed from the transferred state. The source VM memory is either paged in to the destination VM on demand, or is transferred asynchronously by pre-copying and write-protecting the source VM memory, and then later transferring only the modified pages after the destination VM is resumed. The source and destination servers preferably share common storage, in which the source VM's virtual disk is stored; this avoids the need to transfer the virtual disk contents. Network connectivity is preferably also made transparent to the user by arranging the servers on a common subnet, with virtual network connection addresses generated from a common name space of physical addresses.

4 Claims, 3 Drawing Sheets

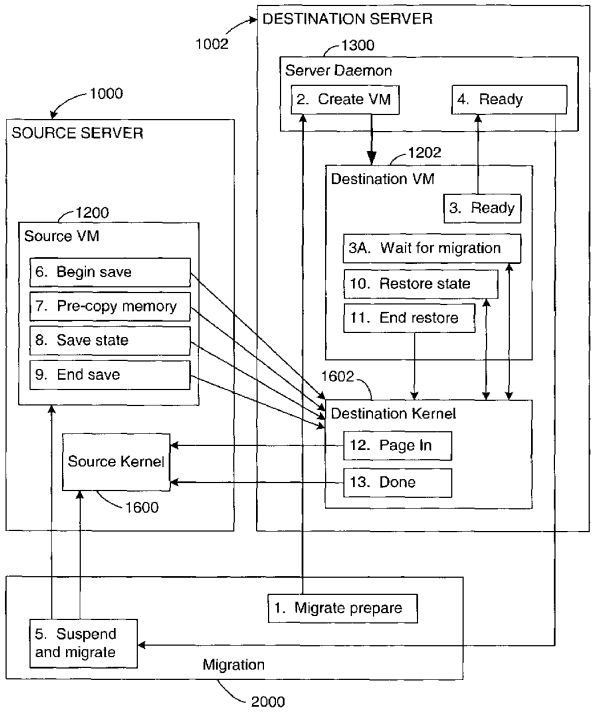
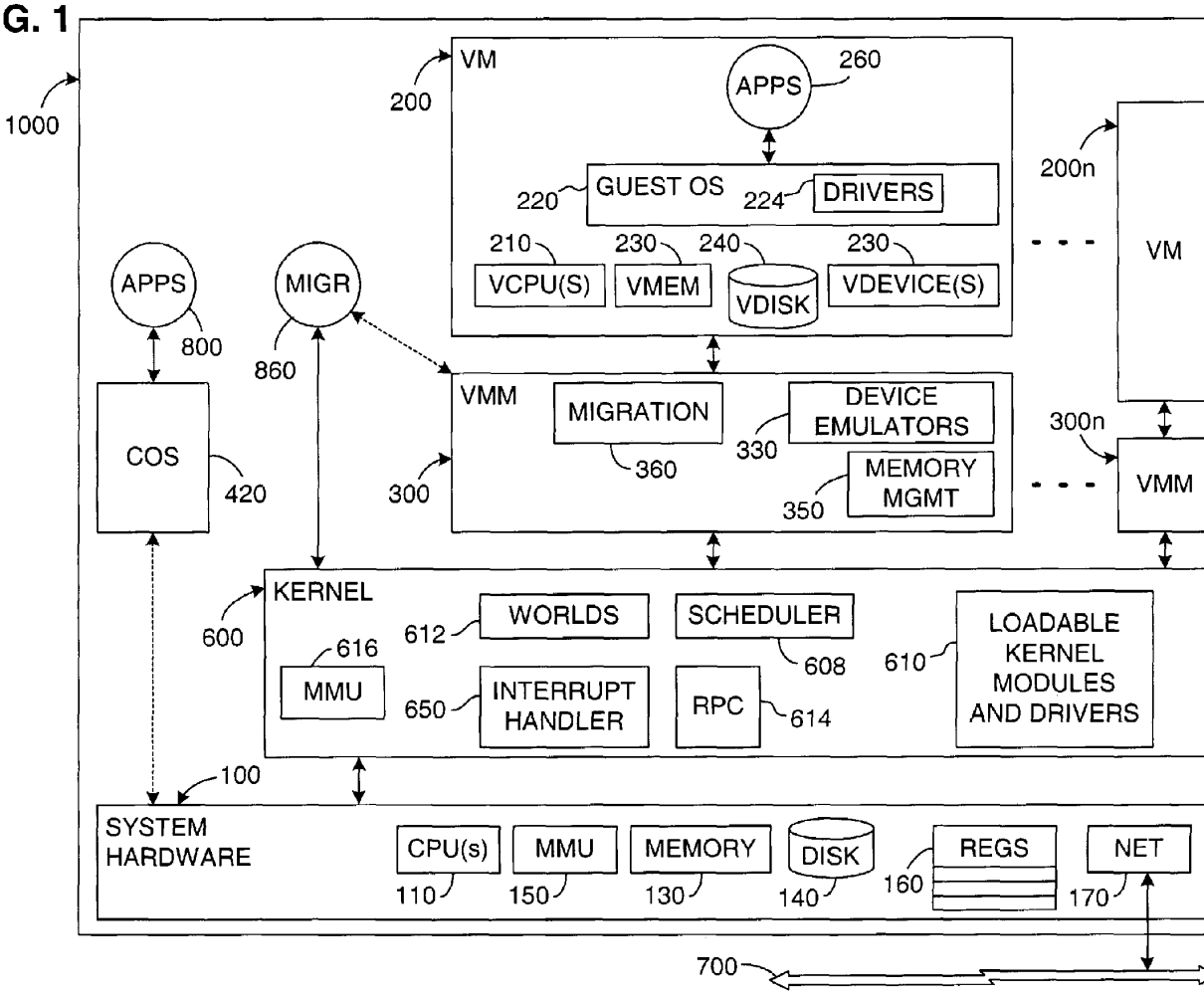


FIG. 1



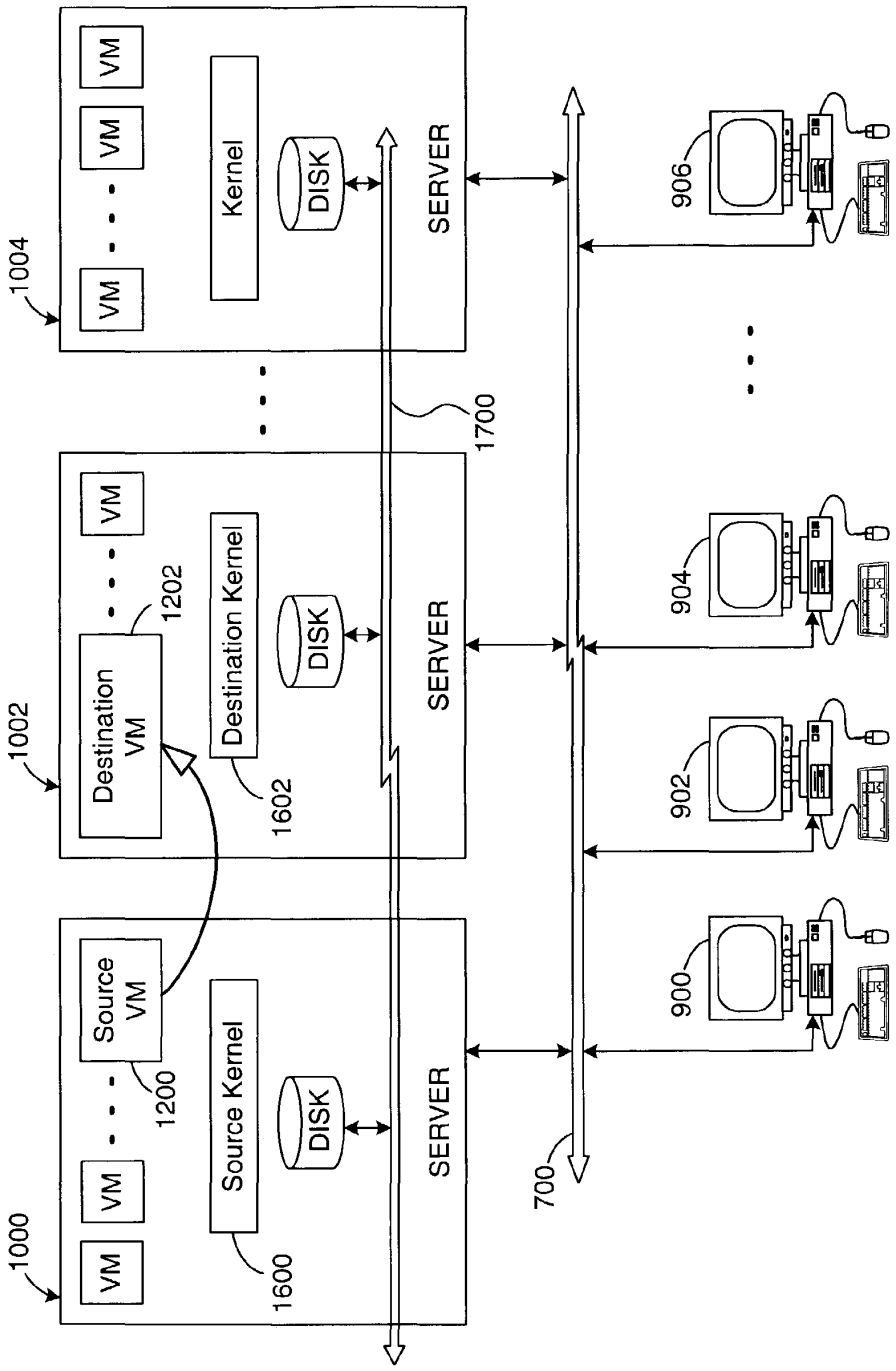
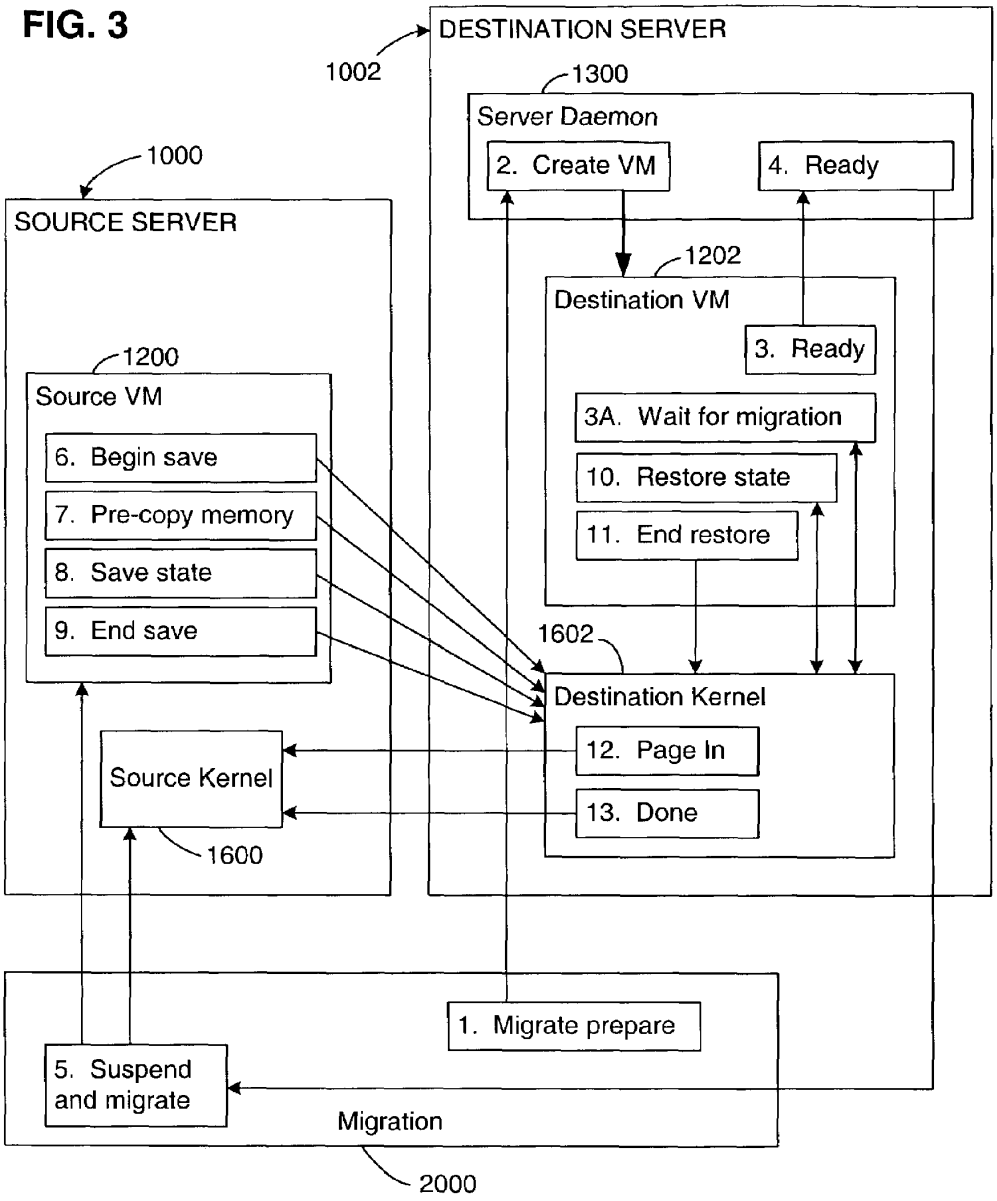


FIG. 2

FIG. 3



VIRTUAL MACHINE MIGRATION

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to a computer architecture, in particular, to an architecture that coordinates the operation of multiple virtual machines.

2. Description of the Related Art

The advantages of virtual machine technology have become widely recognized. Among these advantages is the ability to run multiple virtual machines on a single host platform. This makes better use the capacity of the hardware, while still ensuring that each user enjoys the features of a "complete," isolated computer.

General Virtualized Computer System

As is well known in the field of computer science, a virtual machine (VM) is a software abstraction—a "virtualization"—of an actual physical computer system. FIG. **1** illustrates, in part, the general configuration of a virtual machine **200**, which is installed as a "guest" on a "host" hardware platform **100**.

As FIG. **1** shows, the hardware platform **100** includes one or more processors (CPU's) **110**, system memory **130**, and a storage device, which will typically be a disk **140**. The system memory will typically be some form of high-speed RAM, whereas the disk (one or more) will typically be a non-volatile, mass storage device. The hardware **100** will also include other conventional mechanisms such as a memory management unit MMU **150**, various registers **160**, and any conventional network connection device **170** (such as a network adapter or network interface card—"NIC") for transfer of data between the various components of the system and a network **700**, which may be any known public or proprietary local or wide-area network such as the Internet, an internal enterprise network, etc.

Each VM **200** will typically include at least one virtual CPU **210**, a virtual disk **240**, a virtual system memory **230**, a guest operating system (which may simply be a copy of a conventional operating system) **220**, and various virtual devices **230**, in which case the guest operating system ("guest OS") will include corresponding drivers **224**. All of the components of the VM may be implemented in software using known techniques to emulate the corresponding components of an actual computer.

If the VM is properly designed, then it will not be apparent to the user that any applications **260** running within the VM are running indirectly, that is, via the guest OS and virtual processor. Applications **260** running within the VM will act just as they would if run on a "real" computer, except for a decrease in running speed that will be noticeable only in exceptionally time-critical applications. Executable files will be accessed by the guest OS from the virtual disk or virtual memory, which will simply be portions of the actual physical disk or memory allocated to that VM. Once an application is installed within the VM, the guest OS retrieves files from the virtual disk just as if they had been pre-stored as the result of a conventional installation of the application. The design and operation of virtual machines is well known in the field of computer science.

Some interface is usually required between a VM and the underlying host platform (in particular, the CPU), which is responsible for actually executing VM-issued instructions and transferring data to and from the actual memory and storage devices. A common term for this interface is a "virtual machine monitor" (VMM), shown as component **300**. A VMM is usually a thin piece of software that runs directly on top of a host, or directly on the hardware, and virtualizes all the resources of the machine. Among other components, the VMM therefore usually includes device emulators **330**, which may constitute the virtual devices (**230**) that the VM **200** addresses. The interface exported to the VM is then the same as the hardware interface of the machine, so that the guest OS cannot determine the presence of the VMM. The VMM also usually tracks and either forwards (to some form of operating system) or itself schedules and handles all requests by its VM for machine resources, as well as various faults and interrupts.

Although the VM (and thus the user of applications running in the VM) cannot usually detect the presence of the VMM, the VMM and the VM may be viewed as together forming a single virtual computer. They are shown in FIG. **1** as separate components for the sake of clarity.

Virtual and Physical Memory

As in most modern computers, the address space of the memory **130** is partitioned into pages (for example, in the Intel x86 architecture) or regions (for example, Intel IA-64 architecture). Applications then address the memory **130** using virtual addresses (VAs), which include virtual page numbers (VPNs). The VAs are then mapped to physical addresses (PAs) that are used to address the physical memory **130**. (VAs and PAs have a common offset from a base address, so that only the VPN needs to be converted into a corresponding PPN.) The concepts of VPNs and PPNs, as well as the way in which the different page numbering schemes are implemented and used, are described in many standard texts, such as "Computer Organization and Design: The Hardware/Software Interface," by David A. Patterson and John L. Hennessy, Morgan Kaufmann Publishers, Inc., San Francisco, Calif., 1994, pp. 579-603 (chapter 7.4 "Virtual Memory"). Similar mappings are used in region-based architectures or, indeed, in any architecture where relocatability is possible.

An extra level of addressing indirection is typically implemented in virtualized systems in that a VPN issued by an application **260** in the VM **200** is remapped twice in order to determine which page of the hardware memory is intended. The first mapping is provided by a mapping module within the guest OS **202**, which translates the guest VPN (GVPN) into a corresponding guest PPN (GPPN) in the conventional manner. The guest OS therefore "believes" that it is directly addressing the actual hardware memory, but in fact it is not.

Of course, a valid address to the actual hardware memory must ultimately be generated. A memory management module **350** in the VMM **300** therefore performs the second mapping by taking the GPPN issued by the guest OS **220** and mapping it to a hardware (or "machine") page number PPN that can be used to address the hardware memory **130**. This GPPN-to-PPN mapping is typically done in the main system-level software layer (such as the kernel **600** described below), depending on the implementation: From the perspective of the guest OS, the GVPN and GPPN might be virtual and physical page numbers just as they would be if the guest OS were the only OS in the system. From the perspective of the system software, however, the GPPN is a page number that is then mapped into the physical memory space of the hardware memory as a PPN.

System Software Configurations in Virtualized Systems

In some systems, such as the Workstation product of VMware, Inc., of Palo Alto, Calif., the VMM is co-resident at system level with a host operating system. Both the VMM and the host OS can independently modify the state of the host processor, but the VMM calls into the host OS via a driver and a dedicated user-level application to have the host OS perform certain I/O operations of behalf of the VM. The virtual com-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.