US006073232A

United States Patent [19]

Kroeker et al.

[11] Patent Number: 6,073,232

[45] Date of Patent: *Jun. 6, 2000

[54] **METHOD FOR MINIMIZING A COMPUTER'S INITIAL PROGRAM LOAD TIME AFTER A SYSTEM RESET OR A POWER-ON USING NON-VOLATILE STORAGE**

[75] Inventors: **Richard Mark Kroeker**, Morgan Hill; **Richard Henry Mandel, III**, San Jose, both of Calif.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **08/806,135**

[22] Filed: **Feb. 25, 1997**

[51] **Int. Cl.**[7] **G06F 9/445**

[52] **U.S. Cl.** **713/1**; 713/2; 713/100

[58] **Field of Search** 395/651, 652, 395/653; 713/1, 2, 100

[56] **References Cited**

U.S. PATENT DOCUMENTS

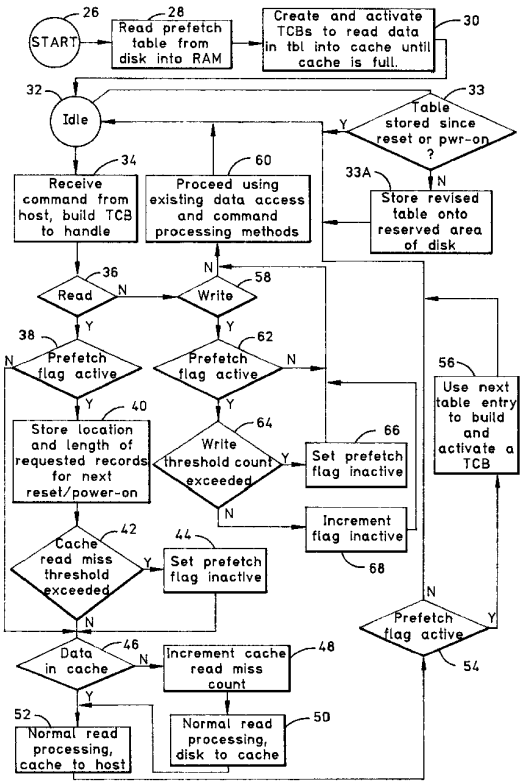
4,663,707	5/1987	Dawson	364/200
5,210,875	5/1993	Bealkowski et al.	395/700
5,230,052	7/1993	Dayan et al.	395/700
5,269,022	12/1993	Shinjo et al.	395/700
5,307,497	4/1994	Feigenbaum et al.	395/700
5,325,532	6/1994	Crosswy et al.	395/700
5,355,498	10/1994	Provinp et al.	395/700
5,404,527	4/1995	Irwin et al.	395/700
5,410,699	4/1995	Bealkowski et al.	395/700

Primary Examiner—Ayaz R. Sheikh
Assistant Examiner—Tim Vo
Attorney, Agent, or Firm—Gray Cary Ware Freidenrich

[57] **ABSTRACT**

A method for increasing boot speed of a host computer with associated hard disk drive generates a prefetch table that contains pointers to disk locations and lengths of the records of an application program requested by the host computer during an initial power-on/reset. During the next power-on/reset, before the host computer is ready for data but after the disk drive has completed its reset routine, using the prefetch table the disk drive accesses the previously requested data and copies it onto the cache of the disk drive, from where it is transferred to the host computer when the host computer requests it. The prefetch table is updated to reflect disk location changes for the various records, or to reflect new records that were requested by the host computer but not found in cache during the previous power-on/reset.

33 Claims, 2 Drawing Sheets



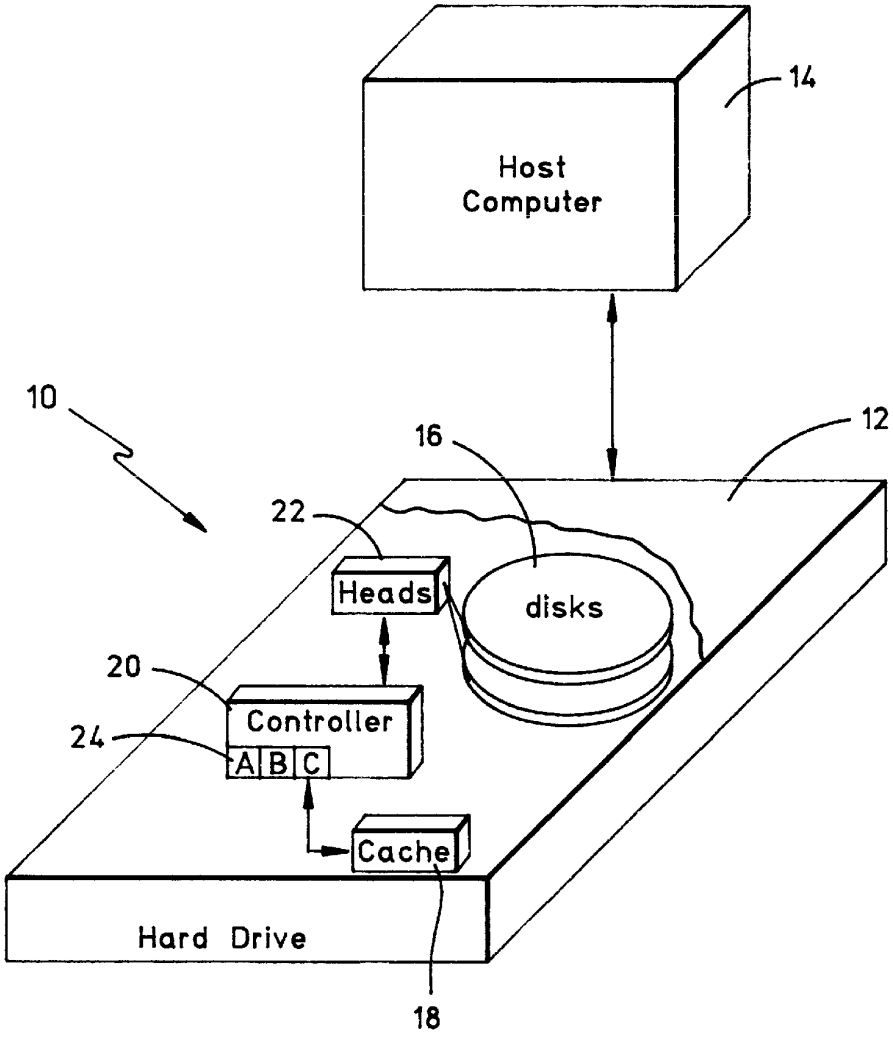


FIG. 1

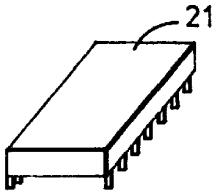


FIG. 2

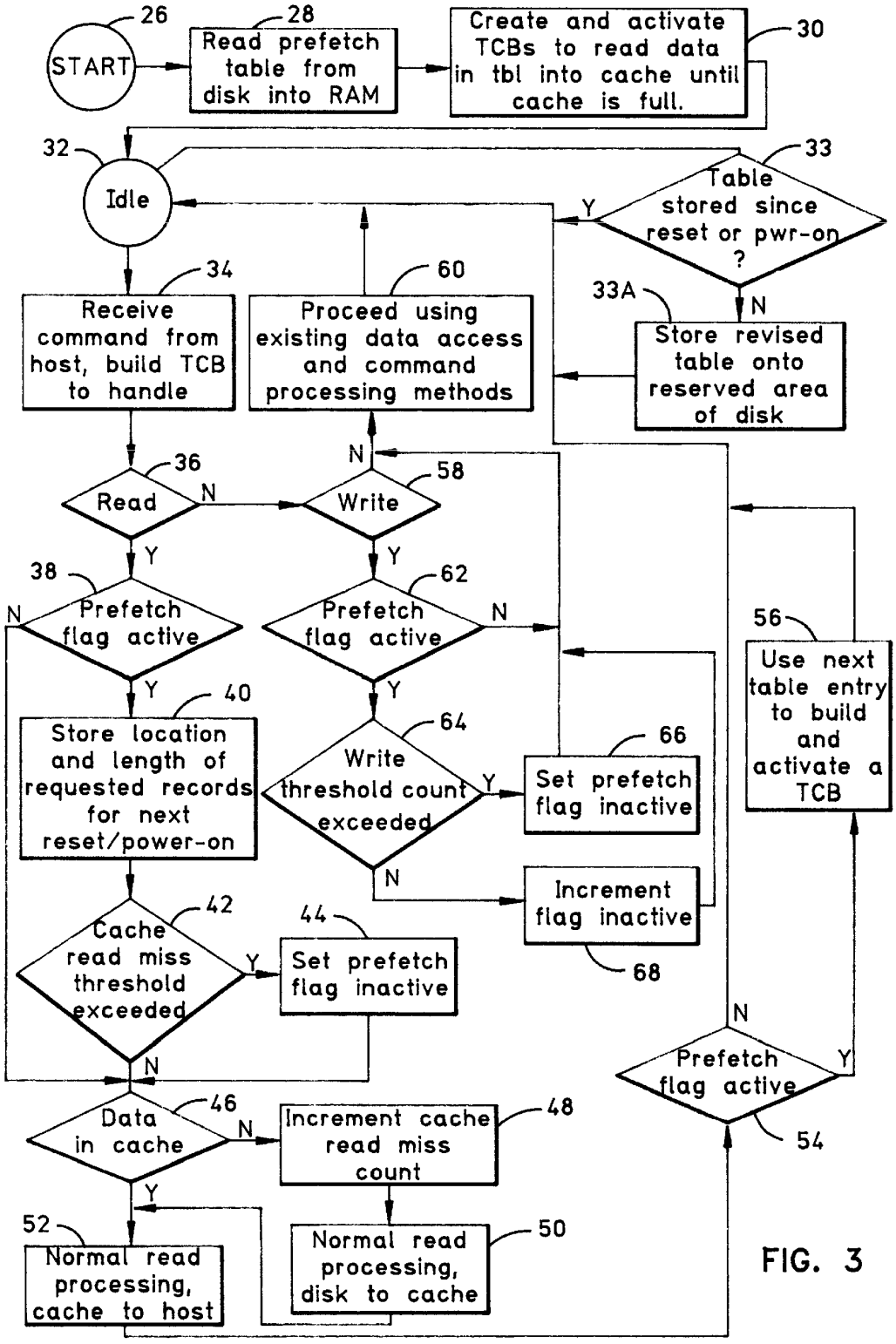


FIG. 3

METHOD FOR MINIMIZING A COMPUTER'S INITIAL PROGRAM LOAD TIME AFTER A SYSTEM RESET OR A POWER-ON USING NON-VOLATILE STORAGE

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to peripheral storage apparatus for computers, and more particularly to shortening the load time of computer programs from a hard disk drive to a host computer.

2. Description of the Related Art

When a computer undergoes a hardware reset (i.e., a power-on or reset), the computer executes procedures embodied in its power-on/reset firmware which prepare for loading an operating system into the computer to condition it for operation. Typically, execution of such procedures begins what is referred to as a "boot". While the computer is executing these power on/reset procedures, peripheral devices that are associated with the computer, such as, for example, a hard disk drive, also execute their own power-on/reset procedures embodied in firmware. When the computer finishes the above-described firmware-implemented portion of the "booting" process, it typically requests data from the disk drive as part of initializing user-selected software, e.g., a program marketed under one of the tradenames DOS, Windows, UNIX, OS/2, AIX, etc.

It happens that the transfer of the selected program to the computer is relatively slow, particularly when the program is a modern large operating system. Accordingly, methods have been disclosed for increasing the speed with which computers "boot". One example of such a method is disclosed in U.S. Pat. No. 5,307,497, which teaches that a portion of an operating system can be stored in read-only memory (ROM) for fast access of the portion during power-on or reset. Unfortunately, the portion stored in ROM is unchangeable. In other words, the method disclosed in the '497 patent does not adapt to changing user preferences regarding operating systems, or indeed to updated versions of a particular operating system.

Another example of a previous attempt to shorten the load time from a disk drive to its host computer is set forth in U.S. Pat. No. 5,269,022. As disclosed in the '022 patent, a snapshot of computer memory is stored in a backup memory that is separate from the disk drive associated with the computer, for use during the next succeeding boot. Unfortunately, the backup memory must be large, because it must store the entire computer memory. Also, the method disclosed in the '022 patent requires operating system intervention, which, because of security features common to many modern operating systems, renders the '022 invention unfeasible.

As recognized by the present invention, however, it is possible to provide, without operating system intervention, a method for adaptively preparing a disk drive to effect rapid application program loading to a host computer. Specifically, we have found that during hardware resets the disk drive associated with a host computer typically completes its booting process before the host computer is ready for program transfer, and as recognized by the present invention, the disk drive can be configured during this period for rapidly communicating a program to the host computer.

Accordingly, it is an object of the present invention to provide a method for rapidly communicating a computer program from a disk drive to a host computer.

Another object of the present invention is to provide a method for rapidly communicating a computer program from a disk drive to a host computer which adapts to changes in user program preference, and to changes in program storage location on the disks of the disk drive. Yet another object of the present invention is to provide a method for rapidly communicating a computer program from a disk drive to a host computer which does not require excessively large storage space, and which can be undertaken entirely by the disk drive itself, transparent to the host computer. Still another object of the present invention is to provide a method for rapidly communicating a computer program from a disk drive to a host computer which is easy to use and cost-effective.

SUMMARY OF THE INVENTION

The invention is embodied in an article of manufacture—a machine component—that is used by a digital processing apparatus and that tangibly embodies a program of instructions that are executable by the digital processing apparatus to rapidly communicate a computer program from a hard disk drive to the drive's host computer.

This invention is realized in a critical machine component that causes a digital processing apparatus to adaptively store a computer program on the cache of the hard disk drive and communicate the program to the host computer. Hereinafter, the machine component is referred to as a "computer program product".

In accordance with the present invention, steps executed by the digital processing apparatus include, after an initial power-up or reset of the hard disk drive and a host computer associated with the drive, receiving an initial read command from the host computer for transferring to the host computer a plurality of data records of a program stored on the disk. A prefetch table is then generated, with the table representing a disk location and length of each data record requested by the initial read command. Then, after a subsequent power-on or reset of the hard disk drive, and during a second power-on or reset of the host computer, the prefetch table is accessed to read into the data cache the data records. In response to a subsequent read command from the host computer, it is determined whether records requested by the subsequent read command are stored in the data cache. If they are, the records are communicated from the cache to the host computer; otherwise, the records are communicated from the disk to the host computer.

Preferably, the accessing and determining steps are repeated for each power-on or reset of the host computer. The steps executed by the digital processing apparatus further include either incrementing a read counter toward a predetermined value, fixed in the algorithm or programmed by the user or adaptively determined based on system environmental conditions, or decrementing a counter towards zero using the same process described for the incrementing condition when it is determined during the determining step that records requested by the subsequent read command are not stored in the data cache. As used generally herein, then, "incrementing" a counter refers both to incrementing and decrementing a counter. Additionally, the steps further include updating the data prefetch table, communicating the records from the disk to the host computer when the read counter exceeds a predetermined threshold, and setting a prefetch flag to inactive when the read counter exceeds the predetermined threshold.

In the presently preferred embodiment, the invention also includes setting the prefetch flag to inactive when a prede-

termined number of write commands from the host computer to the hard disk drive have been received. Moreover, the invention can include, after the communicating step and when the prefetch flag is inactive, determining whether the hard disk drive is idle and if so, storing the prefetch table on the disk. The computer program product is disclosed in combination with the hard disk drive, and in combination with the host computer.

In another aspect, a computer program product is disclosed for use with a host computer that is coupled to a hard disk drive. The hard disk drive includes at least one storage disk having a program stored thereon and a data cache, and the computer program product includes a data storage device which includes a computer usable medium having computer readable program means. As disclosed in detail below, these code means are for enhanced loading of the program from the hard disk drive to the host computer during power-on or reset of the host computer. In accordance with the present invention, code means receive a command from the host computer during a power-up or reset of the host computer. When the command is a read command, code means generate a prefetch table representative of at least the disk location of the records requested by the read command for transfer of the records from the disk to the cache for a subsequent power-on or reset of the host computer. Moreover, code means are provided for determining whether the records have been stored in the cache in response to a previous power-on or reset of the host computer, and the records are communicated to the host computer in response.

In still another aspect, a computer hard disk drive includes at least one data storage disk and a data storage cache. Furthermore, the hard disk drive includes means for recording onto the cache, immediately after a hardware reset of the hard disk drive, data on the disk that has been requested by a host computer during a first hardware reset of the host computer. Additionally, the disk drive includes means for communicating the data from the cache to the host computer during a second hardware reset of the host computer.

The details of the present invention, both as to its structure and operation, can best be understood in reference to the accompanying drawings, in which like reference numerals refer to like parts, and in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a partially schematic view of a disk drive with associated host computer, with portions broken away for clarity;

FIG. **2** is an illustration of memory such as read-only memory (ROM), random access memory (RAM), electrically-erasable programmable read only memory (EEPROM), or dynamic random access memory (DRAM) containing microcode, that embodies the invention as a program storage product; and

FIG. **3** is a flow chart showing the method steps of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring initially to FIG. **1**, a system is shown, generally designated **10**, for promoting rapid communication of a computer program from a hard disk drive **12** to a host computer **14** that is in data communication with the disk drive **12** in accordance with principles well-known in the art. In one intended embodiment, the host computer **14** may be a personal computer (PC) or laptop computer made by IBM Corp. of Armonk, N.Y. Or, the host computer **14** may be a Unix computer, or OS/2 server, or Windows NT server, or IBM RS/6000 250 workstation. Indeed, the host computer **14** can be an embedded controller that is part of a music synthesizer, or part of an industrial instrument. And, the hard disk drive **12** can be any hard disk drive suitable for computer applications, provided that the hard disk drive **12** includes at least one, and typically a plurality of, data storage disks **16** and an on-board, solid state, random access memory (RAM) data cache **18**.

As shown in FIG. **1**, the hard disk drive **12** also includes an onboard controller **20**. In accordance with principles well-known in the art, the onboard controller **20** is a digital processor which, among other things, controls read heads **22** in the disk drive **12** for effecting data transfer to and from the disks **16**.

Additionally, as intended by the present invention the onboard controller **20** includes an adaptive cache module **24**. Per the present invention, the adaptive cache module **24** is executed by the onboard controller **20** as a series of computer-executable instructions. These instructions are embodied as microcode in a memory, e.g., read-only memory (ROM) of the onboard controller **20**. Such a ROM is indicated by reference numeral **21** in FIG. **2**. The ROM **21** contains microinstructions that embody means and program steps that perform according to the invention. When in the ROM **21**, the microinstructions become part of the ROM **21**, and therefore, part of the hardware of the disk drive **12**.

Those skilled in the art will appreciate that the hard disk drive is merely illustrative of a particular tangible environment that is useful for understanding the concepts of our invention. Broadly, the hard disk drive **12** represents a peripheral storage apparatus. The hard disks **16** of the hard disk drive **12** represent data storage elements that are found in the general peripheral storage apparatus. The invention, therefore, applies to such a peripheral storage apparatus and a data storage element, and should not be limited to a hard disk drive.

FIG. **3** illustrates the structure of such microinstructions as embodied in a computer program. Those skilled in the art will appreciate that FIG. **3** illustrates the structures of computer program code elements that function according to this invention. Manifestly, the invention may be practiced in its essential embodiment by a machine component, embodied by the ROM **21**, that renders the computer program code elements in a form that instructs a digital processing apparatus (e.g., the onboard controller **20**) to perform a sequence of function steps corresponding to those shown in the Figures. The machine component is shown in FIGS. **1** and **2** as a combination of program code elements A–C in computer readable form that are embodied in a computer-usable data medium (the ROM **21**) of the onboard controller **20**. Such media can also be found in other semiconductor devices, on magnetic tape, on optical disks, on floppy diskettes, on a DASD array, on a conventional hard disk drive, in logic circuits, in other data storage devices, or even in a node of a network. In an illustrative embodiment of the invention, the computer-executable instructions would be in object code form, compiled or assembled from a C^{++} language program and stored, by conventional means, in the ROM **21**. Or, the code used can be an interpretative code such as Forth, Smalltalk, or Java and its derivatives.

Referring in detail to FIG. **3**, the method of the present invention can be seen. It is to be understood that in the presently preferred embodiment, the method begins immediately after the hard disk drive **12** has completed its power-on/reset (i.e., hardware reset) routine.

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.