EUROPEAN PATENT SPECIFICATION

Date of publication and mention of the grant of the patent:
08.11.2017 Bulletin 2017/45

Application number: 05103697.8

Date of filing: 03.05.2005

Non-volatile memory cache performance improvement

Leistungsverbesserung eines nicht-flüchtigen Cachespeichers

Amélioration de performance d’une antémémoire à mémoire non-volatile

Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU MC NL PL PT RO SE SI SK TR

Priority: 03.05.2004 US 837986

Date of publication of application:
09.11.2005 Bulletin 2005/45

Proprietor: Microsoft Technology Licensing, LLC
Redmond, WA 98052 (US)

Inventors:
- Ergan, Cenk
  98052, Redmond (US)
- Fortin, Michael R.
  98052, Redmond (US)
- Obr, Nathan Steven
  98052, Redmond (US)
- Stemen, Patrick L.
  98052, Redmond (US)
- Nagar, Rajeev Y.
  98052, Redmond (US)
- Verma, Surenda
  98052, Redmond (US)
- Powell, Threron
  98052, Redmond (US)
- Westerinen, William J.
  98052, Redmond (US)
- Nicholson, Clark D.
  98052, Redmond (US)
- Teodosiu, Dan
  98052, Redmond (US)
- DeWhitt, Dean L.
  98052, Redmond (US)
- Hill, Emily Nicole
  98052, Redmond (US)
- Kodavalla, Hanumantha R.
  98052, Redmond (US)
- Parchem, John M.
  98052, Redmond (US)
- Zbikowski, Mark Joseph
  98052, Redmond (US)
- Zwilling, Michael J.
  98052, Redmond (US)

Representative: Grünecker Patent- und Rechtsanwälte
PartG mbB
Leopoldstraße 4
80802 München (DE)

References cited:
WO-A-01/75581

Note: Within nine months of the publication of the mention of the grant of the European patent in the European Patent Bulletin, any person may give notice to the European Patent Office of opposition to that patent, in accordance with the Implementing Regulations. Notice of opposition shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).
CROSS REFERENCE TO RELATED PATENT APPLICATIONS

[0001] This application is related to the following previously-filed applications:


FIELD OF THE INVENTION

[0002] This invention relates generally to computer systems, and more particularly, to improving the performance of computer systems by using non-volatile memory in conjunction with a long-term storage unit.

BACKGROUND OF THE INVENTION

[0003] Certain storage devices are used on computer systems in order to provide persistent memory for the functioning of the computer system. For example, a magnetic hard disk drive assembly is well known in the art. However, such persistent memory has some drawbacks. It is time consuming to prepare the magnetic hard drive for reading or writing data. In order to do so, in most cases, it is necessary to physically initiate spinning of the disk, and to position a read/write head over the proper location.

[0004] The preparation for reading or writing from the storage devices is time- and power-consuming. Additionally, noise may be generated by such a magnetic hard drive memory and its components. Thus, there is a need for an improved memory for computer systems which will remedy some of these drawbacks.

[0005] New compositions and configurations of persistent storage devices may provide some improvements in performance related to the storage. However, simply using traditional techniques for accessing persistent storage will not provide the full benefit of such new compositions or configurations. Generally, an operating system is used to control the operations of the computer system. The operating system also mediates between other applications in the computer system and the hard disk. Thus, there is a need for methods such as application programming interfaces (APIs) to provide enhanced performance. Similarly, there is a need for systems which create new configurations and new control techniques for storage for use by a computer systems.

[0006] WO 01/75581 A relates to a computer system and disk driver where a memory buffer is used during disk drive operations. The disk driver may "pre-fetch" additional and unrequested sectors from a disk drive into a buffer portion of the memory. Further, the number of hard disk drive accesses may be minimized by storing data for a number of write operations in the memory buffer which will later be written to the disk drive in a single write operation to fulfill all write requests at once. Further, the memory of the described computer system may be one of a number of types of random access memories. In addition, an access log may be used to assist the disk driver in reducing hard disk drive accesses and by employing the memory as a write or read buffer.

[0007] US 2004/044849 A1 describes a data storage system having a non IC based memory and an IC based non-volatile memory for storing user data. In one example, the IC based non-volatile memory is implemented with MRAM. Examples of non IC based memory include e.g. hard disks, tape, and compact disks. In some examples, the IC based memory is utilized to store user data from an information device in order to increase the speed and/or the effective storage capacity of the data storage system. In some examples, a portion of a standard size block of user data can be stored on spaces of the non IC based memory that are deficient for storing a standard size block with the remaining portion being stored in IC based memory. Portions of a file of user data may be non-volatily stored in the IC based memory in order to more quickly provide the file to an information device. For example, data of a file, that if stored in a location on the non IC based media would significantly increase the retrieval time of the file, can be stored in the IC based media.

SUMMARY OF THE INVENTION

[0008] It is the object of the invention to provide an improved memory technique for a computer system.

[0009] This object is solved by the present invention as claimed in the independent claims.

[0010] Preferred embodiments are defined by the dependent claims.
A persistent storage device in which one or more long-term storage media are included along with a non-volatile memory is used with an operating system providing or responsive to application program interfaces, commands, and interrupts which allow the operating system or an application program to effectively use the non-volatile memory in order to increase performance time and efficiency. While it is contemplated that the non-volatile memory be included physically with the long-term storage medium, it is also possible for the non-volatile memory to be separate form the long-term storage medium but to function along with it to provide persistent storage.

In one embodiment, application program interfaces (APIs) are exposed which allow for a request that the long-term storage medium be spun up or spun down. Other APIs allow for requests that the non-volatile memory be flushed or populated with data from specified addresses of the long-term storage medium. Still others allow data to be written directly to the long-term storage medium, allow data to be written as an atomic write, or allow data written to the non-volatile memory cache to be ignored rather than saved for writing to the long-term storage medium. Additionally, a request to find and give details about existing non-volatile memory available for such storage is provided for.

In one embodiment, the non-volatile memory can be used in one of two ways. In a first way, a cache is provided in non-volatile memory. This cache is used as a read cache for data read from the long-term storage medium and also as a write buffer for data being written to the long-term storage medium. In this way, reads and writes to the long-term storage medium, instead of occurring sporadically, may be clustered together, requiring fewer reads from, writes to, and preparations of the long-term storage medium.

A portion of the non-volatile memory can be owned by the operating system or another application directly. Certain logical sector addresses (LSAs) from the long-term storage medium are mapped to the non-volatile memory, and reads from or writes to these LSAs will be performed in the non-volatile memory. This mapping can occur alongside the caching in a non-volatile memory portion described above.

Other features of the invention are described below.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing summary, as well as the following detailed description of preferred embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there is shown in the drawings example constructions of the invention; however, the invention is not limited to the specific methods and instrumentalities disclosed. In the drawings:

- Figure 1 is a block diagram generally illustrating an exemplary computer system on which the present invention resides;
- Figure 2 is a block diagram of the exemplary computer system of figure 1 generally illustrating locations where the present invention may be implemented;
- Figure 3 is a block diagram of a persistent storage device according to one aspect of the invention;
- Figure 4 is a block diagram of a persistent storage device according to one aspect of the invention;
- Figure 5 is a block diagram illustrating the flow of data between the operating system, a non-volatile memory, and a long-term storage medium according to one embodiment of the invention;
- Figure 6 is a state diagram of the state of the cache portion of a non-volatile memory according to one embodiment of the invention;
- Figure 7 is a flow diagram of the operations of the boot state according to one embodiment of the invention;
- Figure 8 is a flow diagram of the operations in the running state according to one embodiment of the invention;
- Figure 9 is a flow diagram of the operations in the shut down state according to one embodiment of the invention; and
- Figure 10 is a flow diagram of the operations during a flush of the mapped portion of the non-volatile memory according to one embodiment of the invention.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

In the description that follows, the term "NV memory" will be used to refer to solid state non-volatile memory. NV memory means any type of memory that retains its data when power is removed and that can be erased and reprogrammed with new data. The NV memory may include battery-backed up memory. The NV memory may be located in a computer as a separate component of the computer such as a personal computer slot card or as a component in system memory. Alternatively, it could be a component of a persistent storage device such as a hard drive, in a docking station of a notebook device and the like. On mobile systems, the track buffer memory the disk drives have can be used (i.e., the invention can also be implemented using the track buffer memory).

Future disks for mobile systems may draw from battery power to keep its memory alive without actually drawing power to keep the disk spinning. The NV memory may store file data, file metadata, as well as known disk sectors such as the boot sector and partition tables. During power down or entry into a hibernate mode, the computer system determines...
the static and dynamic configuration data to be stored in NV memory based on a history of prior uses. Data is also stored in the NV memory during system operation to reduce the number of times the disk drive is spun up. For mobile PCs and the like, reducing the number of times the disk drive is spun up improves user experience because power is reduced, thereby allowing smaller and lighter mobile PCs and the like that last longer on a given battery capacity. It also reduces the need to run noisy, high speed fans that are loud and annoying to the user.

[0019] Turning to the drawings, wherein like reference numerals refer to like elements, the invention is illustrated as being implemented in a suitable computing environment. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multi-processor systems, microprocessor based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0020] Figure 1 illustrates an example of a suitable computing system environment 100 on which the invention may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0021] The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multi-processor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0022] The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

[0023] With reference to Figure 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Associate (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0024] Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustics, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer readable media.

[0025] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS),
containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Figure 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0026] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Figure 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0027] The drives and their associated computer storage media discussed above and illustrated in Figure 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In Figure 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers hereto illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 110 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 150 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195.

[0028] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the personal computer 110, although only a memory storage device 181 has been illustrated in Figure 1. The logical connections depicted in Figure 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0029] When used in a LAN networking environment, the personal computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the personal computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Figure 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0030] In the description that follows, the invention will be described with reference to acts and symbolic representations of operations that are performed by one or more computers, unless indicated otherwise. As such, it will be understood that such acts and operations, which are at times referred to as being computer-executed, include the manipulation by the processing unit of the computer of electrical signals representing data in a structured form. This manipulation transforms the data or maintains it at locations in the memory system of the computer, which reconfigures or otherwise alters the operation of the computer in a manner well understood by those skilled in the art. The data structures where data is maintained are physical locations of the memory that have particular properties defined by the format of the data. However, while the invention is being described in the foregoing context, it is not meant to be limiting as those of skill in the art will appreciate that various of the acts and operation described hereinafter may also be implemented in hardware.

Inclusion of NV Memory in the Computer 110

[0031] Turning now to figure 2, the NV memory 200 may be located in the computer 110 as a separate component
202 of the computer 110 such as a card to be put in a pc slot or as a component 204 in system memory. Alternatively, it could be a component 206 of a persistent storage device 300 such as a hard drive 141, in a docking station of a notebook device and the like. Another alternative location for NV memory 200 is in the form of a removable solid state non-volatile memory device which can be plugged in to the computer system, such as one according to the CompactFlash specification as maintained by the CompactFlash Association, or similar. The NV memory 200 as a separate component 202 provides the capability to improve performance on legacy systems that do not have NV memory 200 without having to install a hard drive having NV memory.

Turning now to figure 3, the persistent storage device 300 has a disk assembly 302 that includes a non-volatile memory device which can be plugged in to the computer system, such as one according to the CompactFlash notebook device and the like. Another alternative location for NV memory 200 is in the form of a removable solid state non-volatile memory device which can be plugged in to the computer system, such as one according to the CompactFlash specification as maintained by the CompactFlash Association, or similar. The NV memory 200 as a separate component 202 provides the capability to improve performance on legacy systems that do not have NV memory 200 without having to install a hard drive having NV memory.

Operating System and Host Application Use of NV Memory

With reference again to Figure 2, in order to take advantage of an NV memory 200 to improve performance, according to one embodiment of the invention, the operating system (OS) 134 is provided with various interfaces in order to provide for improved use of the NV memory 200.

First a detection interface is provided. OS 134 can detect whether there is an NV memory 200, and the size of the NV memory. This functionality is exposed as an interface which the application programs 135, and other program modules 136 can use in order to determine if NV memory 200 exists in the computer 110. Additionally, OS components, such as file systems or other object stores, can use the detection interface to determine the existence and size of NV memory 200 on the computer 110. Other information regarding the NV memory 200 may also be provided such as usage information or type information. This information may be useful because certain NV memories have limited lifespans, and this may affect the usage of the NV memory 200 in certain situations.

In order to read from or write to a disk 304 in disk assembly 302, the disk must be spun up, and the read/write head 306 must be moved to the appropriate location by spindle/armature assembly. This preparatory action takes time and causes a delay before data can be read from or written to disk assembly 302. The preparatory action in writing to the disk 304 will be referred to as spinning up the disk, even though it encompasses moving the read/write head 306 as well. Indeed, while the disk is pictured as a rotating media, it is contemplated that in a persistent storage device 300, a NV memory 200 may be included along with another form of the long-term storage medium (other than disk assembly 302) which requires preparatory action before data can be written to or from the memory. Any preparatory action for the use of the long-term storage medium on the persistent storage device 300 will be known as "spin up" or "spinning up" the memory, even though it is contemplated that there may be additional preparatory actions, or even that an actual spinning of a rotating medium may not be involved.

Because an application or the operating system 134 itself may recognize in advance of a write to the persistent storage device 300, that a significant probability exists that such a write requiring a spin up of the long-term storage medium on that device 300 will occur, delays caused by the need to wait for the spin up of long-term storage medium can be obviated through the use of an interface which allows an application or the operating system 134 to request a spin up of the long-term storage medium.

Similarly, a spin down interface is also provided. In order to save power consumption, if no writes are being made to the disk assembly 302 of persistent storage device 300, then the spinning of the disk 304 may be halted. Again, while the persistent storage device 300 is shown as including a rotating medium, where an actual spin down will reduce power consumption, other forms of long-term storage media on the persistent storage device 300 aside from NV memory 200 are contemplated, and such long-term storage media may have actions analogous to the spin down of the disk assembly 302, in which some benefit is gained by ceasing the action which allows immediate or relatively swift reads and writes to that memory. Such cessation of an action may not actually include a physical spinning down of a rotating medium, however, such cessations are contemplated by the "spin down" or "spinning down" referred to herein.
The spin down and spin up actions may be performed at less than their maximum speeds. Performing such operations at slower speeds can lessen the acoustic noise associated with these actions and lessen any associated spikes in power consumption. Because the spin up and spin down (or other preparatory actions for the long term storage medium) can be performed in advance of an absolute need for such actions, these advantages may be obtained without a performance impact. When a disk is spun up slowly, rather than quickly, the disk will be available when it is fully spun up, but the effects on the user and the rest of the system are minimized.

Some or all of the NV memory 200 may be used as a write cache for the disk assembly 302 of a persistent storage device 300. In such a case, an interface is provided for the situation in which an application or the operating system 134 may determine that for performance reasons, the NV memory 200 should be flushed to the other memory in the persistent storage device 300. This interface flushes the NV memory 200 to the long-term storage medium.

Different portions of NV memory 200 may be marked with different priorities. For example, an area of NV memory 200 may be "pinned" or marked with a high priority should be maintained in NV memory 200 rather than being flushed to long-term storage medium preferably to other areas of NV memory 200 which are not pinned or marked in this way. One embodiment of the invention allows one of a number of priorities to be assigned to areas in NV memory 200. For example, there may be three priorities, a highest priority (or "pinned" memory or "priority 1") memory, a second priority ("priority 2") and a lowest priority (or "priority 3"). Where space is needed in the NV memory 200, the lowest priority memory is flushed to long-term storage medium before higher priority memory areas will be flushed. Thus, according to one embodiment of the invention, several interfaces options for flushing the NV memory 200 cache may be presented - e.g., one which flushes the entire cache, one which flushes lower-priority areas of the cache, or one which flushes all of the cache with a priority equal to or less than a priority specified in call to the interface.

As another performance improvement, because data written to the NV memory 200 may later be deleted, an interface is presented which allows the OS 134 or an application to signal to the NV memory 200 that certain data written to the NV memory 200 is no longer required, and can be discarded from the NV memory 200 write cache. This interface allows the OS 134 or an application to tell the NV memory 200 that the data is no longer required, and can be discarded from the NV memory 200 write cache. When the memory is flushed, the discarded areas will not be written to the long-term storage medium, and when a read is performed on the discarded area, an error or random data may be returned.

This interface can be used, for example, in order to manage transactions. Often transaction logs contain intermediate data which is not required to be stored long term. For example, in one case, when a transaction occurs, metadata about the transaction is first committed to the log. If this log is stored in NV memory 200, the write will not require a spinning up of the disk. When the transaction is completed and the changes committed, the stored meta-data about the transaction is changed or deleted to reflect the completed transaction. If, however, the transaction is interrupted before it can be completed and committed, the data in the NV memory 200 will persist and thus can be used to determine what transaction was occurring and solve problems regarding such interrupted transactions.

In another example, this interface can be used to avoid flushing to disk any data or metadata related to temporary files. Initially, these files and their metadata get created in the NV memory 200. When the files are deleted, the OS 134 or an application may signal NV memory 200 that the file data and metadata written to the NV memory 200 is no longer required, and can be discarded from the NV memory 200 write cache.

A NVM circumvention interface is also provided. In certain situations, use of the NV memory may not be efficient for read or write operations. With reference again to Figure 3, for example, if an application is writing to persistent storage device 300 data that is large relative to the size of the NV memory 200, writing some of that data to the NV memory 200 may not be generally useful, as the NV memory 200 will fill and the data will then need to be flushed from the NV memory 200 to the long-term storage medium. Thus, the NVM circumvention interface allows the operating system or an application using the interface to specify that data to be read from or written to the persistent storage device 300 should be directly read from or written to the physical memory addresses on the long-term storage medium rather than to the NV memory 200.

In one embodiment, an interrupt is sent from the persistent storage device 300 to the host computer 110 when the disk controller in the persistent storage device 300 is preparing to or actually spinning up the disk assembly 302. This interrupt may be transmitted to the operating system 134 or its components or to an application. Upon receipt of such an interrupt, a recipient which is properly programmed to act opportunistically can ensure that data is moved from the NV memory 200 to the long-term storage medium or from the long-term storage medium to the NV memory 200, as the long-term storage medium is soon to be spun up and ready for such movements. This reduces the number of long-term storage medium spin-ups and therefore reduces disk drive power consumption. Additionally on receipt of the interrupt indicating that the memory is spinning or spun up, the OS can use the NVM circumvention interface to flush any non-persistent write buffers or caches directly to the long-term storage medium.

Additionally, OS components and applications may use an atomic write interface which allows a write to the persistent storage device 300 to be treated as atomic. All data to be written atomically is stored in the NV memory 200. When this is completed, the persistent storage device ensures that all data to be written atomically is written to long-term storage medium. If a failure occurs before all of the data to be written atomically is stored in the NV memory 200, an error is returned, and the data which has been received is discarded. If, after the storage on the NV memory 200 is
completed, the atomic write fails due to a system or power failure, the failure is detected when the system restarts, and
the atomic set of data stored in the NV memory is rewritten to the long-term storage medium. This allows improved
performance in applications and settings where atomic writes of large data are preferable, e.g. in database applications,
since it does not require these applications to deal with the problem known in the art as "torn writes".

[0046] In one embodiment, the operating system 134, its components, or applications use the NV memory 200 to
store specific files or objects that are used regularly. Thus more expensive writes/reads from long-term storage medium
are avoided. For example, database transaction logs, file system metadata, and other metadata are stored in NV memory
200. Where a priority scheme is being used for assigning different priority to areas in NV memory 200 storing data, in
one embodiment, such frequently-accessed files are accorded a high priority, which further helps to avoid expensive
writes/reads to/from the long-term storage medium.

Partitioning of the NV Memory

[0047] In one embodiment, the NV memory 200 includes a partition which allows a portion of the NV memory 200 to
be used exclusively by the operating system. One implementation of this is to allow the operating system 134 to partition
the NV memory 200, and to have an OS partition of the NV memory 200 which is controlled directly by the disk controller
firmware with input from the operating system.

[0048] Thus, through the disk controller, the OS 134 requests the population of the NV memory 200 cache with specific
sectors that it are likely to be needed next. For example, these sectors are the sectors which are needed or are thought
likely to be needed at the time of the next boot, or when the system resumes from hibernation. As a further example,
these sectors are sectors that the OS 134 predicts it will need in the near future.

[0049] Thus, in such an embodiment, each time the long-term storage medium is spun up the OS can repopulate the
OS partition of the read cache in the NV memory 200 by sending a list of logical sector addresses (LSAs) to the disk
drive controller. This list can include priority information to be used according to a priority scheme as described above
to indicate how readily the disk drive controller will overwrite the read cache data with write buffering data.

[0050] Within the OS 134, a mechanism is provided to map a specific set of LSAs to the NV memory 200. This allows
a section of the NV memory 200 to be directly under the OS 134's control. Reads and writes to these LSAs will then
always access the NV memory 200. The data in these LSAs will not be flushed to the storage unit unless the NV memory
is reconfigured by the operating system. In one embodiment, in this section of NV memory 200, the OS 134 stores "black
box recorder information," i.e. a recent history of operations that is used to diagnose system failures. The OS 134 stores
the boot recovery console and files required for safe boot so that the system can still boot and diagnose mechanical
disk failures. It may also store a possibly compressed package of files required for the next boot to be used in place of
or in addition to the regular representation of system files.

[0051] The interfaces and commands as described above are set forth in the following Table 1:

<table>
<thead>
<tr>
<th>Command</th>
<th>Data for Command</th>
<th>Drive Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query NVM</td>
<td></td>
<td>Determine the existence of an NV memory for use; additionally can be used to determine the size or other information regarding the NV memory</td>
</tr>
<tr>
<td>Spin Up Disk</td>
<td></td>
<td>Prepare the disk for read/write (R/W) access. Opportunistically flush data from write buffer to disk.</td>
</tr>
<tr>
<td>Spin Down Disk</td>
<td></td>
<td>Park the R/W heads and turn off the disk spindle to save power and reduce wear.</td>
</tr>
<tr>
<td>Flush NVM Cache</td>
<td></td>
<td>Move all modified sectors from the NV memory cache to the disk.</td>
</tr>
<tr>
<td>Flush NVM Cache - unpin</td>
<td></td>
<td>Same as Flush NVM cache but pinned blocks are also removed.</td>
</tr>
<tr>
<td>Flush NVM Cache - All</td>
<td></td>
<td>Flush all modified data in mapped portion and in R/W cache to disk.</td>
</tr>
<tr>
<td>Populate NVM Cache</td>
<td>List of LSAs and priorities (e.g., 1 = pin, 2 = remove after priority 3 sectors, 3 = remove as needed)</td>
<td>Spin up disk. Flush cache to disk. Copy data from the disk to the NV memory cache as defined by list. Store list in NV memory cache along with their LSAs and priority.</td>
</tr>
</tbody>
</table>
### Copy-on-Write Functionality

[D0052] Databases and database products, among other applications and products, may use copy-on-write functionality in order to reduce the consumption of resources such as time and memory. Copy-on-write allows previous versions of data to be made available. This is useful because, in addition to a current version of the information stored in, for example, a database, previous versions of the information may be useful to a user.

[D0053] Without copy-on-write, one way to provide information on previous versions of data is to make a complete copy of the database at a point in time which may be of interest to a user. When this occurs, all files associated with the database are stored. However, this technique is space intensive, because for a database which is stored in a certain amount of storage space, each copy will also require that amount of storage space. This approach may also be time consuming, as the copy itself involves movement of lots of data.

[D0054] Thus, instead, changes from the original version may be stored. Where NV memory 200 is available, it can be used for performing copy-on-write activities. In a first implementation of copy-on-write, the original data which is to be changed is stored in the system memory 130. Before it is updated, then, a copy must be written to stable storage, in order to have the unchanged version of the data available. Where NV memory 200 is available, such storage may be to the NV memory 200. This allows the storage of the unchanged copy to be performed quickly. As described above, this copy may be moved to the disk storage at a later point. Database systems using block-oriented caching can use this to reduce time due to memory writes.

[D0055] In a second copy-on-write implementation, the original version may be stored in a long-term storage medium. In such a case, if the data is located in the NV memory 200, an additional copy may be made in NV memory 200 before the write occurs. If the data is not stored in the NV memory, then the original stored copy may be treated as the archive copy - the copy which stores the unchanged version. The NV memory 200 stores the changed version. When this version is flushed from NV memory 200 or otherwise moved to the disk, the version will be written to a new location in memory 200, since the original stored copy on disk is now the archive copy of the data before changes.

### Defragmentation Using NV Memory

[D0056] Large data stores, e.g. file systems and databases, through constant storage and deletion of data may exhibit a condition known as fragmentation. This occurs when large ranges of free memory on a storage device become more and more sparse, and may lead to performance degradation. In order to correct this, defragmentation is used. Defragmentation processes reunitate files in successive address spaces. In order to perform defragmentation, data files are moved within physical memory locations on the disk to create contiguously stored files. Thus, temporary storage locations are used; as the space occupied by a first file is used by a second file, the first file must be stored in some location so it is not lost.
In one embodiment, NV memory is used as a temporary storage location during defragmentation. Data is written to the NV memory. The location of other data is changed in order to increase the contiguousness of the storage of the other data. Then the data from the NV memory is written to the storage device being defragmented. When the data which has been temporarily stored is restored to the storage device (such as the hard disk of a computer system) the temporary copy is removed from NV memory. Thus defragmentation can occur without requiring the use of the long-term storage medium and the associated costs of such use for storing temporary data.

Control over the Persistent Storage Device

As shown in Figure 4 and described above, a persistent storage device 300 can include both a long-term storage medium 430 (e.g. the disk assembly 302 from Figure 3). In such a situation, data from the CPU 120 travels between the host computer 110 and the persistent storage device 300. This flow is mediated by host computer interface 420 in host computer 110 and host computer interface 420 in the persistent storage device 300. Device control firmware 440 may directly or indirectly control storage in the NV memory 200 and the long-term storage medium 430.

In order to begin use of the persistent storage device 300, the operating system 134 queries the storage unit configuration, as described above, to determine whether a NV memory 200 is available and, if so, size and other parameters of the NV memory 200, such as type, utilization history, etc. Then a determination is made regarding how the NV memory 200 will be utilized.

Utilization of the NV memory 200 may be controlled in two ways. First, some portions of NV memory 200 may be under the control of the storage unit. Second, some portions of the NV memory 200 may be under the direct control of the OS 134. As discussed above, when a portion of the NV memory 200 is under direct control of the OS, that portion is used to map a specific set of LSAs. Reads and writes to these LSAs will then always access the NV memory 200 at the mapped portions. The data in these LSAs will not be flushed to the storage unit unless the NV memory is reconfigured by the operating system.

The portion of the NV memory 200 which is not under direct operating system control is used as a cache. This cache portion serves dual duty as a read cache and as a write buffer. When the write buffer is flushed to the disk the OS opportunistically re-populates the NV memory 200 with data from the disk to serve as a read cache. This cache data is overwritten with write buffer data. Eventually the read cache is completely overwritten by the write buffer data, at which time the buffer is full, the storage unit controller spins up the disk, flushes the buffer data to the disk, repopulates the read cache, and spins down the disk.

The use of an NV memory with both a cache portion and a mapped portion is shown in Figure 5. NV memory 200 is divided into a cache portion 500 and a mapped portion 510. As shown by arrow 520, data read from or written to LSAs which are not mapped to the mapped portion 510 is written to the cache portion 500; as shown by arrow 530, if the data being read from an LSA is not present in the cache portion, then it is read from the long-term storage medium 430. As discussed, the cache portion 500 functions as both a read cache and a write cache, arrow 540 shows that data in the cache portion 540 is flushed to the long-term storage medium 430, and data is copied from the long-term storage medium 430 to the cache portion 500 when there is room to use the cache portion 500 as a write buffer.

The mapped portion 510 is mapped to specific LSAs. Thus, as shown by arrow 550, when these LSAs are read from or written to, these reads and writes are directly from the mapped portion 510. Data in the mapped portion 510 does not get flushed to the long-term storage medium 430, nor does long-term storage medium 430 provide data to the mapped portion. The only exception to this is upon remapping. As shown by arrow 560, when a new mapping is created, data from the former mapping is copied to the long-term storage medium 430 and data from the disk for the new LSAs mapped is copied to the NV memory 200 (specifically, to the mapped portion 510.)

Control of Cache Portion 500

The control of the cache portion is defined by the state diagram shown in Figure 6. As shown in Figure 6, a boot state 600 is entered either during initialization (603) or during reboot after shutdown (625). After the boot state, as shown by transition 605, when the operating system moves to a running state 610, and sends a command to flush the cache and repopulate, along with a list of logical sector addresses with which to initially populate the cache. As shown by transition 613, each time this command to flush the cache and repopulate is received from the operating system 134, the operations of the running state 610 are performed again. In order to move to shut down state 620, transition 615, occurs. When a shutdown occurs, according to one embodiment, a command to flush and repopulate the cache, with an LSA list, is sent.

The disk drive controller firmware 440 reduces power consumption and improves reliability and performance by using the NV memory 200 in certain cases rather than using the slower disk 430. The disk drive controller 440 receives commands from the OS 136 to populate the cache portion 500 of the NV memory 200 with data from a specified list of logical sector addresses (LSAs). The list of LSAs also includes a prioritization of the sectors where highest priority sectors
are least likely to be removed from the cache and lowest priority sectors are most likely to be removed. Once the cache 500 is populated, the disk 430 is spun down and the cache 500 is used as a write buffer. Sectors cached for read are overwritten as the buffer fills up according to their priority. The disk is only spun-up when there is a read cache miss, when the write buffer is full, or when an explicit spin-up command is received from the OS 134. In this way the disk 430 will remain off for long periods of time relative to the time it is turned on, thus reducing power and improving reliability. The highest priority LSAs are permanently pinned in the NV memory 200 cache until told otherwise by the OS 134.

[0066] The operations in the boot state 600 are shown in Figure 7. As shown in Figure 7, step 700, the boot state blocks from the NV memory 200 are provided, as requested by the BIOS and the operating system 134. In parallel, the long-term storage medium 430 (e.g. disk) is spun up.

[0067] After this state is complete, the running state 610 is entered, and the OS sends a command to flush and repopulate the cache with specified LSAs. The operations of the running state 610 are shown in Figure 8. As shown in Figure 8, in the running state, if the disk is not already spun up, the disk is spun up, step 800. In step 810, any modified sectors in the NV memory 200 are flushed to the disk/long-term storage medium 430. The LSAs specified by the OS are then read from the disk to the NV memory 200 for the read cache in step 820. The disk is then spun down, step 830. In step 840, the NV memory 200 is used as a read cache and a write buffer. Cached sectors are provided as requested; sectors written to are saved in the NV memory buffer, overwriting read cache sectors according to priority. When the write buffer is over the full threshold, step 850, the NV memory is flushed and repopulated, steps 800 through 830.

[0068] When a command is received to shut down, the shut down state 620 is entered. The operations of the shut-down state 620 are shown in Figure 9. The disk is spun up, step 900. In step 910, any modified sectors in the NV memory 200 are flushed to the disk/long-term storage medium 430. The LSAs specified by the OS are then read from the disk to the NV memory 200 for the read cache in step 920. This allows the NV memory to have data available which will be requested at startup by BIOS and the operating system 134.

[0069] Other forms of shut-down or analogous pauses in functioning (for example, stand by and hibernate) also function according to the methods shown in Figures 6 through 9. Information needed to resume functioning is stored in the NV memory 200, which allows quicker resumption.

[0070] Again, as described above, the OS 134 and applications (via the OS 134) may give other hints to the device controller 440 to help it determine when to spin up or spin down the disk, and conversely, if the controller FW must spin the disk up for any reason, it sends an interrupt to the OS 134 indicating the disk is spinning up. The operating system and applications can then take advantage of this situation to write non-critical data that has been cached in volatile working memory to the storage unit.

[0071] In one embodiment, the controller 440 firmware also utilizes the benefits realized from NV memory 200 to lower acoustic noise in hard drive storage units. By setting the write buffer threshold low enough, the controller 440 is able to spin up more slowly and take longer to flush the write buffer to the disk 430, reducing acoustic noise from the spindle and the heads to seek servos. This technique also reduces the peak current draw from the power supply. The controller 440 firmware also takes advantage of relatively large amounts of data in the buffer to place data on the disk in contiguous blocks and avoid fragmentation when flushing the data to disk.

[0072] The exact state of the cache 500 is known only by the controller 440. The operating system can get a snapshot of it by querying the controller 440. At any point in time the operating system 134 may choose to flush the cache 500 and start with a fresh set of sectors it believes will be required in the near future by sending a command (transition 613 in Figure 6).

[0073] In one embodiment, in the absence of a specific command from the OS 134 to populate the cache portion 500 with files by presenting a list of LSAs (e.g. those required to boot or resume from hibernate on transition 615, or generally those given upon a transition from boot state 600 or in running state 610) the controller could do its own prefetching from disk to NV memory 200 by monitoring disk IOs immediately after boot or S4 resume. These IOs represent the most likely data to be required on the next boot or S4 resume. Once this data is collected by the controller 440 it is stored in a table in NV memory 200. On next shutdown or hibernate, the controller 440 uses this data to populate the NV memory 200 from disk before shutting down or hibernating.

Control of Mapped Portion 510

[0074] The operating system directly controls the mapped portion 510 of NV memory 200. For this portion, the operating system defines specific logical sector addresses that must map to the mapped portion 510. This gives the operating system direct control of the data stored in the NV memory 200. Such control, in one embodiment, may also be given to other applications.

[0075] One way in which this can be accomplished is where operating system 134 creates a portion of NV memory 200 that maps or shadows a specific but not necessarily contiguous set of LSAs. To create this portion the OS sends a command to the disk controller to flush the entire NV memory 200 which moves all modified data in the write buffer and in a previous mapping to the long-term storage medium 430. It then sends a command to map a list of LSAs to the
mapped portion 510 of the NV memory 200. The controller 440 copies sectors from the disk to the NV memory 200 and maintains them there until told otherwise by the operating system 136. If the logical sector addresses sent to the persistent storage device 300 for mapping extend beyond the current address space of the disk 430 then these are appended to the end of the disk’s address space. When a remapping occurs the OS 134 will need to copy the data in these extended addresses to locations in the disk’s address space to avoid losing data. The operating system is in direct control of these mapped sectors in NV memory 200.

[0076] The OS 134 may periodically flush the mapped portion 510 to disk 430 and map a new set of LSAs to the mapped portion 510. This occurs as shown in Figure 10. In step 1000, the disk 430 is prepared for reading/writing. In step 1010, the data from the mapped portion 510 of the NV memory 200 is flushed to the disk 430. In step 1020, the cache portion 500 is also flushed to the disk. In step 1030, the new LSAs to be mapped are copied from the disk 430 to the mapped portion 510 of the NV memory 200. Although not shown, the cache portion 500 in one embodiment, is filled with data from the disk 430 in order to function as a read cache. The disk 430 is then spun down, step 1040.

[0077] As an alternative to this mapping, the NV memory may also be configured as a separate partition or logical volume of the persistent storage device 300. Continuous logical sectors are then under direct control of the operating system 136 as described above.

Self Descriptive Memory Portion for the Persistent Storage Device 300

[0078] In addition to a cache portion 500 and a mapped portion 510 as shown in Figure 5, in one embodiment, a portion of the NV memory is set aside to store certain information about the persistent storage device 300 which will allow diagnostics or other information to be maintained.

[0079] In one embodiment, such descriptive information includes label information, the number of bad sectors in the long-term storage medium 430, statistical data regarding the number of sectors written to or from in a certain amount of time, the number of spin ups or spin downs of the disk in a certain amount of time, and the number of hours the long-term storage medium has run.

[0080] In one embodiment, this self-descriptive memory portion is read by specific interfaces which allow OS components or an application to access the memory. In another embodiment, the persistent storage device 300 includes a display. For example, an LCD display may be included in the persistent storage device 300. When a certain query is made to the persistent storage device 300, either via the computer 110 or directly (e.g. by pressing a button on the persistent storage device 300), data from the self-descriptive memory portion is displayed.

[0081] In one embodiment, the OS or an application use the information stored in the self-descriptive memory to determine when the storage device 300 is nearing its life expectancy and the data stored on said storage device 300 needs to be transferred to a new storage device.

[0082] Information may also be stored which allows the NV memory 200 to be used as a "black box" recorder, which stores data concerning the operation of the computer. In case of a failure, such as an abnormal shut-down, information regarding the operation of the computer is stored and can be retrieved in order to analyze the failure. Such information may include process information regarding processes running in said computer system; thread information regarding threads running in said computer system; data accesses; and device accesses. Similarly to a black box recorder in an airplane, a limited amount of space may be allotted to this operational information, and when more information is collected than can fit in that space, previously-stored operational information is overwritten with newer information.

Wear Leveling

[0083] Certain NV memory technologies suffer from wear-out after too many erase cycles. In order to delay the onset of such wear-out, the controller 440, in one embodiment, balances the erasure of all areas of the NV memory 200 in order to ensure that approximately equal use is made of each area. The controller 440 changes the physical locations used within the NV memory to avoid wearing out NV memory blocks associated with disk sectors that are written to frequently. In one embodiment, this requires a certain amount of overhead to be used to guarantee that there is always space available in the mapped portion to do the wear leveling. For example, if 1 MB of sectors are set aside for mapped portion 510, the disk controller may actually set aside 1.5MB to allow the wear leveling to work properly.

Conclusion

[0084] It is noted that the foregoing examples have been provided merely for the purpose of explanation and are in no way to be construed as limiting of the present invention. While the invention has been described with reference to various embodiments, it is understood that the words which have been used herein are words of description and illustration, rather than words of limitations. Further, although the invention has been described herein with reference to particular means, materials and embodiments, the invention is not intended to be limited to the particulars disclosed.
herein; rather, the invention extends to all functionally equivalent structures, methods and uses, such as are within the scope of the appended claims. Those skilled in the art, having the benefit of the teachings of this specification, may effect numerous modifications thereto and changes may be made without departing from the scope of the invention in its aspects.

Claims

1. A method of operating a computer system having a persistent storage device (300) including a non-volatile memory (200) and a long-term storage medium (302, 430), the long-term storage medium being alternately put into a mode where it is readable and writeable, and into a mode where it is not readable or writeable, the method comprising:

   utilizing (840) a first portion (500) of said non-volatile memory as a read cache and a write buffer for intermediate storage of data from read and write operations (520, 540) from and to said long-term storage medium in a mode where the long-term storage medium is not readable or writeable, and for flushing the write buffer data to the long-term storage medium and repopulating the read cache in a mode where the long-term storage medium is readable and writeable;

   utilizing a second portion (510) of said non-volatile memory as a portion to which at least one logical sector address, LSA, of said long-term storage medium has been mapped by means of a command that receives a list of said at least one LSA, said command being from an operating system or application, wherein utilizing the second portion comprises:

   when said at least one LSA is written to, performing (550) a write operation to the second portion (510) of said non-volatile memory; and

   when said at least one LSA is read from, performing (550) a read operation from said second portion (510) of said non-volatile memory,

   wherein data in the second portion does not get flushed to the long-term storage medium, nor does the long-term storage medium provide data to the second portion, unless new LSAs are to be mapped to the second portion.

2. The method of claim 1, where said method further comprises providing an interface for requesting that a direct correspondence of at least one non-volatile memory storage element of said non-volatile memory with said at least one LSA be altered.

3. The method of claim 2, further comprising, in response to said request that said direct correspondence be altered:

   reading second data from each non-volatile memory storage element to said corresponding long-term storage element of said non-volatile memory;

   reading third data from at least one long-term storage element to non-volatile memory according to a second direct correspondence of at least one non-volatile memory storage element of said non-volatile memory with at least one long-term storage element of said non-volatile memory;

   writing fourth data from write operations directed at said long-term storage elements to said non-volatile storage element in accordance with said second direct correspondence; and

   reading fifth data in response to read operations directed at said long-term storage element from said non-volatile storage element in accordance with said second direct correspondence.

4. The method of claim 1, further comprising:

   providing an interface for requesting that data written to said write buffer be transferred to said long-term storage medium.

5. The method of claim 4, further comprising:

   providing an interface for requesting that a preparatory action be performed on said long-term storage medium to prepare said long-term storage medium for writing.

6. The method of claim 5, where said long-term storage medium comprises a rotating storage medium, and said preparatory action comprises spinning up (800, 1000) said rotating storage medium.
7. The method of claim 6, where said preparatory action occurs more slowly than the fastest possible speed for said preparatory action.

8. The method of claim 5, further comprising:
   detecting that said preparatory action has been performed; and
   transferring (1020) first data written to said write buffer to said long-term storage medium.

9. The method of claim 8, further comprising:
   determining that a set of second data is available for transfer to said long-term storage; and
   upon detection that said preparatory action has been performed, transferring said second data to said long-term storage.

10. The method of claim 1, further comprising:
    providing an interface for requesting that an action be performed on said long-term storage medium to deactivate it temporarily for writing.

11. The method of claim 10, where said long-term storage medium comprises a rotating storage medium, and said action comprises spinning down (830, 1040) said rotating storage medium.

12. The method of claim 11, where said deactivation action occurs more slowly than the fastest possible speed for said preparatory action.

13. The method of claim 4, further comprising:
    providing an interface for requesting that certain specified data in one or more write operations be written directly to said long-term storage medium rather than to said write buffer.

14. The method of claim 4, where, for each element of data stored in said write buffer, a priority indication is assigned to said element, where said interface for requesting that data be transferred to said long-term storage medium comprises an indication of a level of priority, and where, for each element of data stored in said write buffer, whether said element is transferred to said long-term storage medium is based on said indication and said assigned level of priority for said element.

15. The method of claim 4, further comprising:
    providing an interface for indicating that certain specified data stored in said non-volatile memory is invalid.

16. The method of claim 15, where said invalid data is not transferred to said long-term storage medium.

17. The method of claim 16, where a location in said non-volatile memory used for said invalid data is made available for use to store other data.

18. The method of claim 4, further comprising:
    providing an interface for receiving information regarding said non-volatile memory.

19. The method of claim 18, where said information comprises whether said non-volatile memory is present.

20. The method of claim 19, where said information comprises the size of said non-volatile memory.

21. The method of claim 4, where said information comprises at least one selected from among the following: usage information regarding said non-volatile memory; and type of said non-volatile memory.

22. The method of claim 4, further comprising:
providing a control ensuring that different physical locations within said non-volatile memory are erased with approximately equal frequency.

23. The method of claim 1, further comprising:

providing an interface for requesting that certain data be read from said long-term storage media into said first portion of non-volatile memory.

24. The method of claim 23, further comprising:

providing an interface for requesting that a preparatory action be performed on said long-term storage medium to prepare said long-term storage medium for reading.

25. The method of claim 24, where said long-term storage medium comprises a rotating storage medium, and said preparatory action comprises spinning up (800, 1000) said rotating storage medium.

26. The method of claim 23, further comprising:

providing an interface for requesting that an action be performed on said long-term storage medium to deactivate it temporarily for reading.

27. The method of claim 26, where said long-term storage medium comprises a rotating storage medium, and said action comprises spinning down (830, 1040) said rotating storage medium.

28. The method of claim 23, where, for each element of data stored in said read buffer, a priority indication is assigned to said element, where portions of said read buffer are periodically reassigned for other purposes, and where whether a first portion of said read buffer is reassigned before a second portion of said read buffer is based on the relative values of a first priority indication assigned to elements of data in said first portion and a second priority indication assigned to elements of data stored in said second portion.

29. The method of claim 28, where at least one of said other uses comprise use as a write buffer for intermediate storage of data from write operations to said long-term storage medium; and where said method further comprises:

providing an interface for requesting that data written to said write buffer be transferred to said long-term storage medium.

30. A computer-readable medium having computer-executable instructions to perform the method of one of claims 1 to 29.

31. A computer system comprising logic to perform the method of one of claims 1 to 29.

Patentansprüche

1. Verfahren zum Betreiben eines Computersystems mit einer persistenten Speichereinrichtung (300), die einen nicht-flüchtigen Speicher (200) und ein dauerhaftes Speichermedium (302, 430) enthält, wobei das dauerhafte Speichermedium alternativ in einen Modus, in dem es lesbar und beschreibbar ist, und in einen Modus, in dem es nicht lesbar oder beschreibbar ist, versetzt wird, wobei das Verfahren umfasst:

Nutzen (840) eines ersten Teils (500) des nicht-flüchtigen Speichers als eines Lesecaches und eines Schreibpuffers für das Zwischenspeichern von Daten aus Lese- und Schreiboperationen (520, 540) von und zu dem dauerhaften Speichermedium in einem Modus, in dem das dauerhafte Speichermedium nicht lesbar oder beschreibbar ist, und zum Verschieben der Schreibpufferdaten zu dem dauerhaften Speichermedium und zum erneuten Füllen des Lesecaches in einem Modus, in dem das dauerhafte Speichermedium lesbar und beschreibbar ist,

Nutzen eines zweiten Teils (510) des nicht-flüchtigen Speichers als eines Teils, auf den wenigstens eine Logiksektoradresse (LSA) des dauerhaften Speichermediums mittels eines Befehls, der eine Liste der wenigstens einen LSA empfängt, gemappt wurde, wobei der Befehl von einem Betriebssystem oder einer Anwendung stammt, wobei das Nutzen des zweiten Teils umfasst:
wenn zu der wenigstens eine LSA geschrieben wird, Durchführen (550) einer Schreiboperation zu dem zweiten Teil (510) des nicht-flüchtigen Speichers, und wenn von der wenigstens einen LSA gelesen wird, Durchführen (550) einer Leseoperation von dem zweiten Teil (510) des nicht-flüchtigen Speichers, wobei Daten in dem zweiten Teil nicht zu dem dauerhaften Speichermedium verschoben werden und auch das dauerhafte Speichermedium keine Daten zu dem zweiten Teil vorsieht, wenn keine neuen LSAs auf den zweiten Teil zu mappen sind.


3. Verfahren nach Anspruch 2, das weiterhin in Reaktion auf die Anfrage des Ändern der direkten Entsprechung umfasst:

Lesen von zweiten Daten von jedem nicht-flüchtigen Speicherelement zu dem entsprechenden dauerhaften Speicherelement des dauerhaften Speichermediums,
Lesen von dritten Daten von wenigstens einem dauerhaften Speicherelement zu dem nicht-flüchtigen Speicher gemäß einer zweiten direkten Entsprechung wenigstens eines nicht-flüchtigen Speicherelements des dauerhaften Speicherelements des dauerhaften Speichermediums,
Schreiben von vierten Daten von an die dauerhaften Speicherelemente gerichteten Schreiboperationen zu dem nicht-flüchtigen Speicherelement gemäß der zweiten direkten Entsprechung, und
Lesen von fünften Daten in Reaktion auf an das dauerhafte Speicherelement gerichtete Leseoperationen von dem nicht-flüchtigen Speicherelement gemäß der zweiten direkten Entsprechung.

4. Verfahren nach Anspruch 1, das weiterhin umfasst:


5. Verfahren nach Anspruch 4, das weiterhin umfasst:

Vorsehen einer Schnittstelle für das Anfragen des Durchführens einer vorbereitenden Aktion auf dem dauerhaften Speichermedium, um das dauerhafte Speichermedium für das Schreiben vorzubereiten.

6. Verfahren nach Anspruch 5, wobei das dauerhafte Speichermedium ein sich drehendes Speichermedium ist und wobei die vorbereitende Aktion das Hochfahren (800, 1000) des sich drehenden Speichermediums umfasst.

7. Verfahren nach Anspruch 6, wobei die vorbereitende Aktion langsamer erfolgt als die schnellstmögliche Geschwindigkeit für die vorbereitende Aktion.

8. Verfahren nach Anspruch 5, das weiterhin umfasst:

Erfassen, dass die vorbereitende Aktion durchgeführt wurde, und Übertragen (1020) von zu dem Schreibpuffer geschriebenen ersten Daten zu dem dauerhaften Speichermedium.

9. Verfahren nach Anspruch 8, das weiterhin umfasst:

Bestimmen, dass ein Satz von zweiten Daten für die Übertragung zu dem dauerhaften Speicher verfügbar ist, und nach dem Erfassen, dass die vorbereitende Aktion durchgeführt wurde, Übertragen der zweiten Daten zu dem dauerhaften Speicher.

10. Verfahren nach Anspruch 1, das weiterhin umfasst:

Vorsehen einer Schnittstelle für das Anfragen des Durchführens einer Aktion auf dem dauerhaften Speichermedium, um dieses vorübergehend für ein Schreiben zu deaktivieren.

11. Verfahren nach Anspruch 10, wobei das dauerhafte Speichermedium ein sich drehendes Speichermedium ist und
wobei die Aktion das Herunterfahren (830, 1040) des sich drehenden Speichermediums umfasst.

12. Verfahren nach Anspruch 11, wobei die deaktivierende Aktion langsamer erfolgt als die schnellstmögliche Geschwindigkeit für die vorbereitende Aktion.

13. Verfahren nach Anspruch 4, das weiterhin umfasst:


15. Verfahren nach Anspruch 4, das weiterhin umfasst:


17. Verfahren nach Anspruch 16, wobei die Position in dem nicht-flüchtigen Speicher, die für die ungültigen Daten verwendet wird, für die Verwendung zum Speichern von anderen Daten verfügbar gemacht wird.

18. Verfahren nach Anspruch 4, das weiterhin umfasst:

Vorsehen einer Schnittstelle für das Empfangen von Informationen in Bezug auf den nicht-flüchtigen Speicher.

19. Verfahren nach Anspruch 18, wobei die Informationen umfassen, ob der nicht-flüchtige Speicher anwesend ist.

20. Verfahren nach Anspruch 19, wobei die Informationen die Größe des nicht-flüchtigen Speichers umfassen.


22. Verfahren nach Anspruch 4, das weiterhin umfasst:

Vorsehen einer Steuerung, die sicherstellt, dass verschiedene physikalische Positionen in dem nicht-flüchtigen Speicher mit einer annähernd gleichen Frequenz gelöscht werden.

23. Verfahren nach Anspruch 1, das weiterhin umfasst:


24. Verfahren nach Anspruch 23, das weiterhin umfasst:

Vorsehen einer Schnittstelle für das Anfragen, dass eine vorbereitende Aktion auf dem dauerhaften Speichermedium durchgeführt wird, um das dauerhafte Speichermedium für ein Lesen vorzubereiten.

25. Verfahren nach Anspruch 24, wobei das dauerhafte Speichermedium ein sich drehendes Speichermedium ist und wobei die vorbereitende Aktion das Hochfahren (800, 1000) des sich drehenden Speichermediums umfasst.

26. Verfahren nach Anspruch 23, das weiterhin umfasst:
Vorsehen einer Schnittstelle für das Anfragen des Durchführens einer Aktion auf dem dauerhaften Speichermedium, um dieses vorübergehend für ein Lesen zu deaktivieren.

27. Verfahren nach Anspruch 26, wobei das dauerhafte Speichermedium ein sich drehendes Speichermedium ist und die Aktion das Herunterfahren (830, 1040) des sich drehenden Speichermediums umfasst.


29. Verfahren nach Anspruch 28, wobei wenigstens eine der anderen Verwendungen eine Verwendung als Schreibpuffer für das Zwischenspeichern von Daten von Schreiboperationen zu dem dauerhaften Speichermedium umfasst und wobei das Verfahren weiterhin umfasst:


30. Computerlesbares Medium mit computerausführbaren Befehlen zum Durchführen des Verfahrens der Schritte 1 bis 29.

31. Computersystem mit einer Logik zum Durchführen des Verfahrens der Schritte 1 bis 29.

Revendications

1. Procédé pour faire fonctionner un système informatique comportant un dispositif de stockage persistant (300) incluant une mémoire non volatile (200) et un support de stockage à long terme (302, 430), le support de stockage à long terme étant alternativement placé dans un mode dans lequel il est accessible en lecture et écriture et dans un mode dans lequel il n'est pas accessible en lecture ou écriture, le procédé comprenant les étapes consistant à :

   utiliser (840) une première partie (500) de ladite mémoire non volatile en tant que cache de lecture et tampon d'écriture pour un stockage intermédiaire de données provenant d'opérations de lecture et d'écriture (520, 540) dans ledit support de stockage à long terme, dans un mode dans lequel le support de stockage à long terme n'est pas accessible en lecture ou écriture, et pour purger les données du tampon d'écriture vers le support de stockage à long terme et remplir le cache de lecture, dans un mode dans lequel le support de stockage à long terme est accessible en lecture et en écriture ;

   utiliser une seconde partie (510) de ladite mémoire non volatile en tant que partie mise en correspondance avec au moins une adresse de secteur logique (LSA) dudit support de stockage à long terme au moyen d'une commande qui reçoit une liste des au moins une LSA, ladite commande provenant d’un système d’exploitation ou d’une application, où l'utilisation de la seconde partie comporte :

   quand ladite au moins une LSA reçoit une écriture, mettre en oeuvre (550) une opération d'écriture dans la seconde partie (510) de ladite mémoire non volatile ; et

   quand ladite au moins une LSA est lue, mettre en oeuvre (550) une opération de lecture de la seconde partie (510) de ladite mémoire non volatile,

   où des données dans la seconde partie ne sont pas purgées vers le support de stockage à long terme et le support de stockage à long terme ne fournit pas de données à la seconde partie, à moins que de nouvelles LSA ne doivent être mises en correspondance avec la seconde partie.

2. Procédé selon la revendication 1, où ledit procédé comprend en outre la mise à disposition d'une interface pour demander la modification d'une correspondance directe entre au moins un élément de stockage de mémoire non volatile de ladite mémoire non volatile avec ladite au moins une LSA.

3. Procédé selon la revendication 2, comprenant en outre les étapes consistant à, en réponse à ladite demande de modification de ladite correspondance directe :
lire des secondes données dans chaque élément de stockage de mémoire non volatile audit élément de stockage à long terme correspondant dudit support de stockage à long terme ;
lire des troisièmes données dans au moins un élément de stockage à long terme à la mémoire non volatile conformément à une seconde correspondance directe d’au moins un élément de stockage de mémoire non volatile de ladite mémoire non volatile avec au moins un élément de stockage à long terme dudit support de stockage à long terme ;
écrire des quatrièmes données d’opérations d’écriture destinées auxdits éléments de stockage à long terme dans ledit élément de stockage non volatile conformément à ladite seconde correspondance directe ; et
lire des cinquièmes données en réponse à des opérations de lecture destinées audit élément de stockage à long terme dudit élément de stockage non volatile conformément à la seconde correspondance direct.

4. Procédé selon la revendication 1, comprenant en outre :

la mise à disposition d’une interface pour demander que des données écrites dans ledit tampon d’écriture soient transférées audit support de stockage à long terme.

5. Procédé selon la revendication 4, comprenant en outre :

la mise à disposition d’une interface pour demander qu’une action préparatoire soit mise en oeuvre sur ledit support de stockage à long terme afin de préparer le support de stockage à long terme à l’écriture.

6. Procédé selon la revendication 5, où ledit support de stockage à long terme comprend un support de stockage rotatif, et ladite action préparatoire comprend la mise en rotation (800, 1000) dudit support de stockage rotatif.

7. Procédé selon la revendication 6, où ladite action préparatoire s’effectue à une vitesse plus lente que la vitesse maximale possible pour ladite action préparatoire.

8. Procédé selon la revendication 5, comprenant en outre les étapes consistant à :

détecter que ladite action préparatoire a été réalisée ; et
transférer (1020) des premières données écrites dans ledit tampon d’écriture audit support de stockage à long terme.

9. Procédé selon la revendication 8, comprenant en outre les étapes consistant à :

déterminer qu’un ensemble de secondes données est disponible pour être transféré audit stockage à long terme ; et
quand il est détecté que l’action préparatoire a été réalisée, transférer lesdites secondes données audit stockage à long terme.

10. Procédé selon la revendication 1, comprenant en outre :

la mise à disposition d’une interface pour demander qu’une action soit mise en oeuvre sur ledit support de stockage à long terme pour le désactiver temporairement en vue de l’écriture.

11. Procédé selon la revendication 10, où ledit support de stockage à long terme comprend un support de stockage rotatif, et ladite action comprend le ralentissement de la rotation (830, 1040) dudit support de stockage rotatif.

12. Procédé selon la revendication 11, où ladite action de désactivation s’effectue à une vitesse plus lente que la vitesse maximale possible pour ladite action préparatoire.

13. Procédé selon la revendication 4, comprenant en outre :

la mise à disposition d’une interface pour demander que certaines données spécifiées dans une ou plusieurs opérations d’écriture soient écrites directement dans ledit support de stockage à long terme plutôt que dans le tampon d’écriture.

14. Procédé selon la revendication 4, où pour chaque élément de données stocké dans ledit tampon d’écriture, une
indication de priorité est assignée à l'élément, où ladite interface pour demander que les données soient trans-
ferées audit support de stockage à long terme comprend une indication d'un niveau de priorité et où pour chaque
élément de données stocké dans ledit tampon d'écriture, la question de savoir si ledit élément doit être transféré
audit support de stockage à long terme dépend de ladite indication et dudit niveau de priorité attribué à chaque
élément.

15. Procédé selon la revendication 4, comprenant en outre :
la mise à disposition d'une interface pour indiquer que certaines données spécifiées stockées dans ladite
mémoire non volatile sont non valides.

16. Procédé selon la revendication 15, où lesdites données invalides ne sont pas transférées audit support de stockage
à long terme.

17. Procédé selon la revendication 16, où un emplacement dans ladite mémoire non volatile utilisé pour lesdites données
non valides est mis à disposition pour être utilisé pour stocker d'autres données.

18. Procédé selon la revendication 4, comprenant en outre :
la mise à disposition d'une interface pour recevoir des informations concernant ladite mémoire non volatile.

19. Procédé selon la revendication 18, où lesdites informations comprennent la présence ou non de ladite mémoire
non volatile.

20. Procédé selon la revendication 19, où lesdites informations comprennent la taille de ladite mémoire non volatile.

21. Procédé selon la revendication 4, où lesdites informations comprennent au moins une information sélectionnée
parmi les suivantes : une information d'utilisation de ladite mémoire non volatile ; et un type de ladite mémoire non
volatile.

22. Procédé selon la revendication 4, comprenant en outre :
la mise à disposition d’une commande garantissant que divers emplacements physiques dans ladite mémoire
non volatile soient effacés avec une fréquence approximativement égale.

23. Procédé selon la revendication 1, comprenant en outre :
la mise à disposition d'une interface pour demander que certaines données soient lues dans ledit support de
stockage à long terme à destination de ladite première partie de la mémoire non volatile.

24. Procédé selon la revendication 23, comprenant en outre :
la mise à disposition d'une interface pour demander qu'une action préparatoire soit mise en oeuvre sur ledit
support de stockage à long terme pour préparer ledit support de stockage à long terme à la lecture.

25. Procédé selon la revendication 24, où ledit support de stockage à long terme comprend un support de stockage
rotatif, et ladite action préparatoire comprend la mise en rotation (800, 1000) dudit support de stockage rotatif.

26. Procédé selon la revendication 23, comprenant en outre :
la mise à disposition d'une interface pour demander qu'une action soit mise en oeuvre sur ledit support de
stockage à long terme pour le désactiver temporairement en vue de la lecture.

27. Procédé selon la revendication 26, où ledit support de stockage à long terme comprend un support de stockage
rotatif, et ladite action comprend le ralentissement de la rotation (830, 1040) dudit support de stockage rotatif.

28. Procédé selon la revendication 23, où pour chaque élément de données stocké dans ledit tampon de lecture, une
indication de priorité est assignée audit élément, où des parties dudit tampon de lecture sont périodiquement réas-
signées à d'autres usages, et où la question de savoir si une première partie dudit tampon de lecture doit être réassigné avant une seconde partie dudit tampon de lecture dépend des valeurs relatives d'une première indication de priorité assignée à des éléments de données dans la première partie et d'une seconde indication de priorité assignée à des éléments de données stockés dans la seconde partie.

29. Procédé selon la revendication 28, où au moins un desdits autres usages comprend une utilisation du tampon d'écriture pour le stockage intermédiaire de données d'opérations d'écriture dans ledit support de stockage à long terme ; et où ledit procédé comprend en outre :
   la mise à disposition d'une interface pour demander que des données écrites dans ledit tampon d'écriture soient transférées audit support de stockage à long terme.

30. Support de stockage à long terme comprenant des instructions exécutables par ordinateur pour mettre en oeuvre le procédé selon l'une des revendications 1 à 29.

31. Système informatique comprenant une logique pour mettre en oeuvre le procédé selon l'une des revendications 1 à 29.
FIG. 5
FIG. 6

Start

Provide boot state blocks from NVM as requested by BIOS and OS and spin up disk.

FIG. 7
Start

Spin up disk

800

Flush modified sectors from NVM buffer to disk

810

Copy LSAs from disk to NVM for read cache

820

Spin down disk

Yes

830

Utilize NVM for read cache and write buffer

840

No

Write buffer over full threshold?

850

FIG. 8
FIG. 9

Start

Spin up disk.

Flush modified sectors from NVM buffer to disk

Copy LSAs from disk to NVM
Start

1000

Spin up disk

1010

Flush data from NVM to disk for previously mapped LSAs

1020

Flush from NVM to disk for write buffer data

1030

Copy LSAs from disk to NVM for new mapped portions of the LSA space.

1040

Spin down disk

FIG. 10
REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader’s convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US 20040064647 A1 [0001]
- US 20040003223 A1 [0001]
- US 20040153694 A1 [0001]
- WO 0175581 A [0006]
- US 2004044849 A1 [0007]