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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 March 2011 (31.03.2011)

PCT

(10) International Publication Number
WO 2011/036015 A1

(51) International Patent Classification:
G06F 3/06 (2006.01)

(21) International Application Number:
PCT/EP2010/062385

(22) International Filing Date:
25 August 2010 (25.08.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09171182.0 24 September 2009 (24.09.2009) EP

(71) Applicant *(for all designated States except US)*: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York 10504 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **FUENTE, Carlos, Francisco** [GB/GB]; IBM United Kingdom Limted, M/P 148, Husley Park, Hursley, Winchester Hampshire SO21 2JN (GB). **SCALES, William, James** [GB/GB]; IBM United Kingdom Limited, M/P 148, Hursley Park, Hursley, Winchester Hampshire SO21 2JN (GB). **WHYTE, Barry, Douglas** [GB/GB]; IBM United Kingdom Limited, M/P 148, Hursley Park, Hursley, Winchester Hampshire SO21 2JN (GB).

(74) Agent: **STRETTON, Peter, John**; IBM United Kingdom Limited, Intellectual Property Law, Hursley Park, Winchester Hampshire SO21 2JN (GB).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK,

[Continued on next page]

(54) Title: DATA STORAGE USING BITMAPS

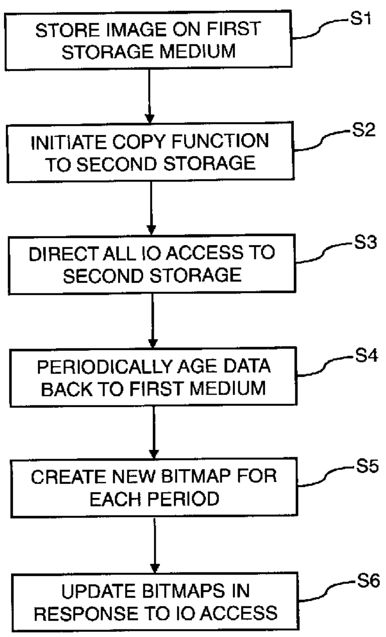

Fig. 6

(57) Abstract: A data storage system comprises a controller, a first lower performance storage medium and a second higher performance storage medium. The controller is connected to the storage mediums and is arranged to control IO access to the storage mediums. The controller is further arranged to store an image on the first storage medium, initiate a copy function from the first storage medium to the second storage medium, direct all IO access for the image to the second storage medium, periodically age data from the second storage medium to the first storage medium, create a new empty bitmap for each period, and in response to an IO access for data in the image, update the latest bitmap to indicate that the data has been accessed and update the previous bitmaps to indicate that the data has not been accessed.

WO 2011/036015 A1

EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

DATA STORAGE USING BITMAPS

DESCRIPTION

This invention relates to a data storage system and to a method of operating the data storage system. In one embodiment, a FlashCopy process is used to separate frequently accessed data from less frequently accessed data using a single FlashCopy target with multiple bitmaps to reduce capacity usage.

Within a large data storage system there are likely to be multiple different types of storage media in use. The various types can be classified by a number of metrics, such as capacity, availability, performance and cost. Examples of different properties that distinguish the different classes of storage might include attachment technology (for example, SATA or FC/SCSI) drives, redundancy scheme (for example, RAID-5, RAID-1 or RAID-10), and space-saving algorithm (for example, compression, de-duplication or non-compressed). A new class of storage technology that is emerging is storage-class memory, of which Flash Memory is a preferred example. The different applications and servers that are being hosted by the storage media will have varying requirements with respect to these metrics. Each application will tend to have its own requirements, and a given application's requirements will also tend to vary over time as the demands on that application vary.

It is a therefore a very complex task to optimise the allocation of storage amongst applications, to ensure the best use of the available resources, in order to meet the requirements of a business as a whole. The work required to perform a change to try and optimise the resource allocation can also be expensive. Re-allocating storage to a server can require server downtime, plus significant administrator effort in order to perform the change. The use of a product that implements storage virtualization, such as SAN Volume Controller (SVC), can eliminate the server downtime, and significantly reduce the administrative effort. Such products provide a data migration feature that allows data to be relocated from one class of storage to another, transparently to the using system. One limit of these changes, however, is that the most convenient unit of management is at the vdisk, volume or logical unit (LUN) level. A whole vdisk must be given the same class of storage. There are many examples though, where the vdisks in a data storage system may not comprise data with homogeneous requirements. In fact, it is likely that a completely homogenous vdisk is the exception rather than the norm. One particular case where there will be different regions of data with different requirements is for performance.

Conventional caching techniques can help with this problem to a limited degree. A write cache can allow a host to perceive a low write service time, even where backed by slow access storage. A write cache can also absorb a short burst of write activity quickly, and feed it to slower backend storage over a longer period of time, emulating higher performance storage. Both these capabilities are known, and implemented in, for example, SVC. But write caching as currently used is limited in its capabilities. Firstly, write IO must ultimately be sent to the backing storage from the write cache. It is possible to eliminate a small proportion of the traffic, but the majority must still be processed there. If the backing storage is unable to sustain the host application write traffic, then the write cache becomes full and the advantage is lost. Additionally, the size of a write cache is small compared to the total amount of system storage, possibly less than 1%. Together, these mean that a conventional write cache is not sufficient to allow storage within a low-performance class to be used for those portions of a disk that have higher performance requirements.

Some classes of storage, such as those that implement compression, or data de-duplication, present an extreme example of a low performance store. Though these might provide significant cost savings in terms of capacity required to satisfy a given server or application's storage requirements, the performance penalty for performing write IO against these stores means they cannot be used for general purpose IO. Their algorithms increase the cost of performing IO, and also place a limit on the peak throughput that they can sustain.

It is therefore an object of the invention to improve upon the known art.

According to a first aspect of the present invention, there is provided a data storage system comprising a controller, a first lower performance storage medium and a second higher performance storage medium, the controller connected to the storage mediums and arranged to control IO access to the storage mediums, wherein the controller is arranged to store an image on the first storage medium, initiate a copy function from the first storage medium to the second storage medium, direct all IO access for the image to the second storage medium, periodically age data from the second storage medium to the first storage medium, create a new empty bitmap for each period, and in response to an IO access for data in the image, update the latest bitmap to indicate that the data has been accessed and update the previous bitmaps to indicate that the data has not been accessed.

According to a second aspect of the present invention, there is provided a method of operating a data storage system, the system comprising a controller, a first lower performance storage medium and a second higher performance storage medium, the controller connected to the storage mediums and arranged to control IO access to the storage mediums, wherein the method comprises the steps of storing an image on the first storage medium, initiating a copy function from the first storage medium to the second storage medium, directing all IO access for the image to the second storage medium, periodically aging data from the second storage medium to the first storage medium, creating a new empty bitmap for each period, and in response to an IO access for data in the image, updating the latest bitmap to indicate that the data has been accessed and updating the previous bitmaps to indicate that the data has not been accessed.

According to a third aspect of the present invention, there is provided a computer program product on a computer readable medium for operating a data storage system, the system comprising a controller, a first lower performance storage medium and a second higher performance storage medium, the controller connected to the storage mediums and arranged to control IO access to the storage mediums, wherein the product comprises instructions for storing an image on the first storage medium, initiating a copy function from the first storage medium to the second storage medium, directing all IO access for the image to the second storage medium, periodically aging data from the second storage medium to the first storage medium, creating a new empty bitmap for each period, and in response to an IO access for data in the image, updating the latest bitmap to indicate that the data has been accessed and updating the previous bitmaps to indicate that the data has not been accessed.

Owing to the invention, it is possible to provide a data storage system in which the system will automatically identify and segregate high-performance regions from low-performance regions with a low processing overhead, using a single higher performance storage medium. Therefore, the portions of the original storage medium containing data which is important for system performance receive better service than the rest, and therefore the system as a whole is advantaged. Frequently accessed, high-performance-requiring data on a vdisk is separated from less-performance-requiring data and is assigned to the very highest-performing class of storage, leaving the rest of the vdisk data on a lower-performing class of storage. This capability maximises the performance of the applications and servers on the storage medium, for the lowest cost.

Preferably, the controller is arranged, when periodically aging data from the second storage medium to the first storage medium, to perform the aging procedure after a preset time period. The periodic aging process, which moves data stored by the higher performance second storage medium to the lower performance first storage medium, is best performed after a regular defined period, which can be set by an administrator or derived from usage statistics. This period is then used each time to trigger the aging process. In this way data on the higher performance second storage medium that has not been used recently can be merged back into the slower storage medium thereby freeing up more space on the faster storage medium.

Advantageously, the controller is arranged, when periodically aging data from the second storage medium to the first storage medium, to copy data from the second storage medium to the first storage medium that has not been accessed for a predetermined length of time. This length of time can be expressed as a set number of periods, where the period is the time gap between two aging procedures. For example, the length of time could be set at five periods, which means that any data that has not been accessed on the faster storage medium since the previous five aging procedures will be merged back to the slower storing medium in the current aging procedure.

Ideally, the controller is arranged, when periodically aging data from the second storage medium to the first storage medium, to perform the aging according to the oldest existing bitmap. The aging process can be carried out using the bitmaps, as this will document in the oldest existing bitmap which data has not been accessed since that bitmap was created. This can then be used to decide which data should be merged back onto the slower storage medium. In this case, it is advantageous that the controller is further arranged, following periodically aging data from the second storage medium to the first storage medium, to delete the oldest existing bitmap.

Embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings, in which:-

Figure 1 is a schematic diagram of a storage system,
Figure 2 is a schematic diagram of a second embodiment of the storage system,
Figure 3 is a further schematic diagram of the storage system showing an aging process,
Figure 4 is a schematic diagram of an alternative storage system,
Figure 5 is a further schematic diagram of the storage system showing bitmaps, and
Figure 6 is a flowchart of a method of operating the storage system.

A data storage system is shown in Figure 1, which comprises a controller 10, a first, lower performance storage medium 12 and a second, higher performance storage medium 14. The controller 10 is connected to the storage mediums 12 and 14 and is arranged to control the IO access to the storage mediums 12 and 14. Although the two storage media 12 and 14 are shown as the same size, their physical size and/or data storage capacity may be different. The lower performance storage medium 12 could be a hard disk and the higher performance storage medium 14 could be a solid state memory device, for example. IO access to the data storage, whether read or write access, is managed by the controller 10.

The controller 10 is arranged to store an image on the first storage medium 12. This image will be data representing multiple applications and servers. The controller 10 is further arranged to initiate a copy function from the first storage medium 12 to the second storage medium 14. The copy function will copy data to the second storage medium 14 when there is a read or write access to any data within the image stored by the first storage medium 12. The controller is arranged to direct all IO access for the image to the second storage medium

14, and additionally will periodically age data from the second storage medium 14 to the first storage medium 12. This process of aging the data will be described in more detail below.

Figure 1b shows the situation after there has been IO access to the sector labelled 2 of the first storage medium 12. This IO access could be to overwrite some of the data in this sector. The controller resolves this by using the incoming IO and the original sector 2 to create the new sector 2a on the faster storage medium 14. Since all IO access is directed to the faster storage medium 14, any future read or write accesses to the data in sector 2 will be handled directly by the faster storage medium 14, without the need to make any access to the original image stored by the slower storage medium 12.

A preferred embodiment of the storage system is shown in Figure 2. This embodiment uses the FlashCopy and Space-Efficient (thin provisioning) features of IBM SAN Volume Controller, which are further extended to achieve the desired data separation between the most frequently used data and less frequently used data. The letters FC in the Figure refer to the Flashcopy function with the arrow indicating the source storage medium 12 and the target storage medium 14 of the copy function. At the most fundamental level, a Space-Efficient FlashCopy image (vdisk) is held on the high-performance, high-cost storage media 14, with the remainder of the storage held as a (typically) fully-allocated image (vdisk) on the low-performance lower-cost storage medium 12.

In this version of the storage system, all application IO is directed at the left-most FlashCopy image stored on the storage medium 14. This image contains the most recently accessed data, by virtue of the FlashCopy function between the two storage volumes. The image stored by the higher performance storage medium grows as the recent working set for the application expands. The right image of the first storage medium 12 contains the full image of the application disk, however this medium 12 does not contain the most recent updates, which are held on the left-most disk 14. Before any merging of data back onto the disk 12 takes place that storage disk 12 maintains a point-it-time copy as of the time that the FlashCopy was taken.

Write accesses to the image are handled using a conventional FlashCopy algorithm, using a copy-on-write technique. Therefore, if the data to be over-written is already located on the left-most disk 14, then it is over-written in place. If data has not been recently accessed, it will be found on the right-most disk 12. Wherever the data is found, it is read from its current location, merged with the new write data, and written to the left-most disk 14. In this mode of operation therefore, the left-most disk 14 is accumulating changes that amount to the recent working set of the application. Many future IO accesses will be serviced directly by the left-most disk 14. Some IO accesses will not have been accumulated there and will instead be serviced by the right-most-disk 12.

If the arrangement described above were static, then the left-most disk 14 would, over time, grow to be closer, in terms of the size of the data being stored, to the full-size of the right-most disk 12. This will happen, as even infrequently accessed data, once accessed, would begin to consume space on the storage medium 14. Eventually all of the data on the lower performance disk 12 would be accessed, and the data stored by the higher performance disk 14 would be the entirety of the original data image stored on the slower storage medium 12, with IO amendments. Therefore an ageing process is needed to remove infrequently accessed data from the left-most disk 16.

Figure 3 illustrates the aging process. The aging process can be triggered after a specific time has elapsed, or based on the data being stored by the higher performance medium 14 reaching a certain size, or other metrics or combination of metrics. Different triggering methods can be used in different system arrangements at different times. Figure 3a shows the higher performance storage medium after there has been IO access to the data stored in sectors 2, 3 and 5. These sectors have been copied across to the higher performance storage medium 14 and combined with the new data contained in the IO accesses (assuming that they were data writes). Any further IO accesses to these sectors, whether read or write, will be handled by the left-most medium 14.

After the periodic aging process has taken place, then the data storage system will be as shown in Figure 3b. The data stored by the sector 2a, on the higher performance storage medium 14, has been copied back to the lower performance storage medium 12. This process updates the image stored by the lower performance medium 12, and the original space for the data stored on the medium 14 can be de-allocated. Although, in this example, the data merged back to the slower medium 12 was the first bit of data to be copied across to the faster disk 14, this is not necessarily the action undertaken in the merging process. The data being merged back is the data that has not been accessed most recently. If there had been a recent read access to the sector 2a on the second medium 14, then this sector would not have been merged back.

The storage system uses a FlashCopy target with multiple bitmaps to reduce capacity usage. The system as described above, with reference to Figures 1 to 3, can provide the separation of frequently and less frequently accessed data for the purposes of creating tiers between different classes of storage and providing fine granularity of progression of data between tiers and over longer periods of time than a single target instance. However, if such a system is extended with multiple target space-efficient disks, for example using two faster storage media 14 and 16, as shown in Figure 4, then that storage system requires more capacity over time, especially as the number of targets is increased. A larger numbers of tiers of targets and/or a larger numbers of targets are required to maintain finer granularity of the "frequency" of access over longer periods of time of hours, and days etc.

In a system using multiple tiers or multiple targets, during the time period of hours and days etc., the amount of newly written data, or frequently re-read data, has to be stored on each target and some percentage of this has to be moved between targets when data is re-written, or frequently re-read. This places overheads on the infrastructure which has to maintain lots of targets and increases the necessary bandwidth for the additional grains copied between targets. The storage system according to the present invention provides an adaptation of the system of Figure 4 that eliminates the complexity of the multiple targets with no additional bandwidth requirements, while still providing the advantages of that system.

In an environment such as that of Figure 4, the system allows multiple target space-efficient disks 14 and 16 to be used as storage "buckets" for frequently accessed data. In such a scheme, either X targets need to be created and managed at the initialisation of the process, or as time period Y has elapsed a new target must be created. As frequently accessed data is "re-hit" at the newest target, (re-hit meaning the data block already resides on one of the other earlier targets) then the block must be copied from the earlier target, merged with the new data I/O and written to the latest target. This places additional bandwidth requirements on the system.

The storage system according to the present invention provides a scheme by which a single target resource can be used to achieve the benefits of splitting the data between different classes of data storage media, but however it removes the need for any data to be re-copied between targets. The data stays in place on the one target storage medium 14 and so the management of the system is greatly simplified and the I/O re-direction does not need to continually adjust to point to the latest target. Overall capacity utilisation may also be reduced as the system does not end up with two (or more) allocated grains on multiple targets where some grains are simply marked as 'stale' but are still allocated.

Figure 5 shows the storage system, which uses bitmaps 18. Instead of using a cascade of, for example, four storage media P, Q, R and S (where S is the lower performance storage medium 12 and P, Q and R are different targets), only two disks R + S are used. It can be seen that although the storage system only has one space-efficient target 14, the system maintains multiple bitmaps that are associated with this single target 14. Each bitmap refers to the data that resides on the space-efficient target 14 at the end of one of an elapsed time periods. For example, bitmap1 covers data that was updated during period 2X to 3X (equivalent to target R), bitmap2 covers data that was updated during period X to 2X (equivalent to target Q), and bitmap3 covers data that was updated during period X (equivalent to target P).

When frequently accessed data is re-hit, as per the explanation above, instead of now having to copy the whole grain from target P to target R, the controller can simply update bitmap3 to show that the data has been accessed during the latest time period and then unset the bit in the bitmap1. Only the new I/O for this grain is now written to the single target disk 14. Thus, if an I/O was only 4K, and the grain size if 64K, there has been saved 60K of bandwidth in the system. Thus the system has now (by changing two bits in meta-data) “moved” the data to be on the latest target, without any physical data moving. Similarly the system does not end up with two copies of the same grain (on different targets and corresponding to different points in time) there is only the latest version of the grain on the disk 14, thus there has been a cut in the used capacity (on the target(s)) by 50% for data that is frequently accessed. As the system moves on to the next time period (say 4X) the controller will simply have to initialize a new bitmap 18 to all zeros. The system does not need to create a new target nor does it have to move the I/O re-direction layer point to a new target. The whole system simply resides on the single target 14.

The method of operating the storage system is summarised in Figure 6. The method comprises the first step S1 of storing an image on the first storage medium 12, followed by the steps S2 of initiating a copy function from the first storage medium 12 to the second storage medium 14 and S3 of directing all IO access for the image to the second storage medium 14. The next step is the step S4 of periodically aging data from the second storage medium 14 to the first storage medium 12, step S5 of creating a new empty bitmap 18 for each period, and finally, step S6 of, in response to an IO access for data in the image, updating the latest bitmap 18 to indicate that the data has been accessed and updating the previous bitmaps 18 to indicate that the data has not been accessed.

CLAIMS

1. A data storage system comprising a controller, a first lower performance storage medium and a second higher performance storage medium, the controller connected to the storage mediums and arranged to control IO access to the storage mediums, wherein the controller is arranged to:

- store an image on the first storage medium,
- initiate a copy function from the first storage medium to the second storage medium,
- direct all IO access for the image to the second storage medium,
- periodically age data from the second storage medium to the first storage medium,
- create a new empty bitmap for each period, and
- in response to an IO access for data in the image, update the latest bitmap to indicate that the data has been accessed and update the previous bitmaps to indicate that the data has not been accessed.

2. A system according to claim 1, wherein the controller is arranged, when periodically aging data from the second storage medium to the first storage medium, to perform the aging after a preset time period.

3. A system according to claim 1 or 2, wherein the controller is arranged, when periodically aging data from the second storage medium to the first storage medium, to copy data from the second storage medium to the first storage medium that has not been accessed for a predetermined length of time.

4. A system according to claim 1, 2 or 3, wherein the controller is arranged, when periodically aging data from the second storage medium to the first storage medium, to perform the aging according to the oldest existing bitmap.

5. A system according to any preceding claim, wherein the controller is further arranged, following periodically aging data from the second storage medium to the first storage medium, to delete the oldest existing bitmap.

6. A method of operating a data storage system, the system comprising a controller, a first lower performance storage medium and a second higher performance storage medium, the controller connected to the storage mediums and arranged to control IO access to the storage mediums, wherein the method comprises the steps of:

- storing an image on the first storage medium,
- initiating a copy function from the first storage medium to the second storage medium,
- directing all IO access for the image to the second storage medium,
- periodically aging data from the second storage medium to the first storage medium,
- creating a new empty bitmap for each period, and
- in response to an IO access for data in the image, updating the latest bitmap to indicate that the data has been accessed and updating the previous bitmaps to indicate that the data has not been accessed.

7. A method according to claim 6, wherein the step of periodically aging data from the second storage medium to the first storage medium comprises performing the aging after a preset time period.

8. A method according to claim 6 or 7, wherein the step of periodically aging data from the second storage medium to the first storage medium comprises copying data from the second storage medium to the first storage medium that has not been accessed for a predetermined length of time.

9. A method according to claim 6, 7 or 8, wherein the step of periodically aging data from the second storage medium to the first storage medium comprises perform the aging according to the oldest existing bitmap.

10. A method according to any one of claims 6 to 9, and further comprising, following periodically aging data from the second storage medium to the first storage medium, deleting the oldest existing bitmap.

11. A computer program product on a computer readable medium for operating a data storage system, the system comprising a controller, a first lower performance storage medium and a second higher performance storage medium, the controller connected to the storage mediums and arranged to control IO access to the storage mediums, wherein the product comprises instructions for:

o storing an image on the first storage medium,

o initiating a copy function from the first storage medium to the second storage medium,

o directing all IO access for the image to the second storage medium,

o periodically aging data from the second storage medium to the first storage medium,

o creating a new empty bitmap for each period, and

o in response to an IO access for data in the image, updating the latest bitmap to indicate that the data has been accessed and updating the previous bitmaps to indicate that the data has not been accessed.

12. A computer program product according to claim 11, wherein the instructions for periodically aging data from the second storage medium to the first storage medium comprise instructions for perform the aging after a preset time period.

13. A computer program product according to claim 11 or 12, wherein the instructions for periodically aging data from the second storage medium to the first storage medium comprise instructions for copying data from the second storage medium to the first storage medium that has not been accessed for a predetermined length of time.

14. A computer program product according to claim 11, 12 or 13, wherein the instructions for periodically aging data from the second storage medium to the first storage medium comprise instructions for performing the aging according to the oldest existing bitmap.

15. A computer program product according to any one of claims 11 to 14, and further comprising instructions for, following periodically aging data from the second storage medium to the first storage medium, deleting the oldest existing bitmap.

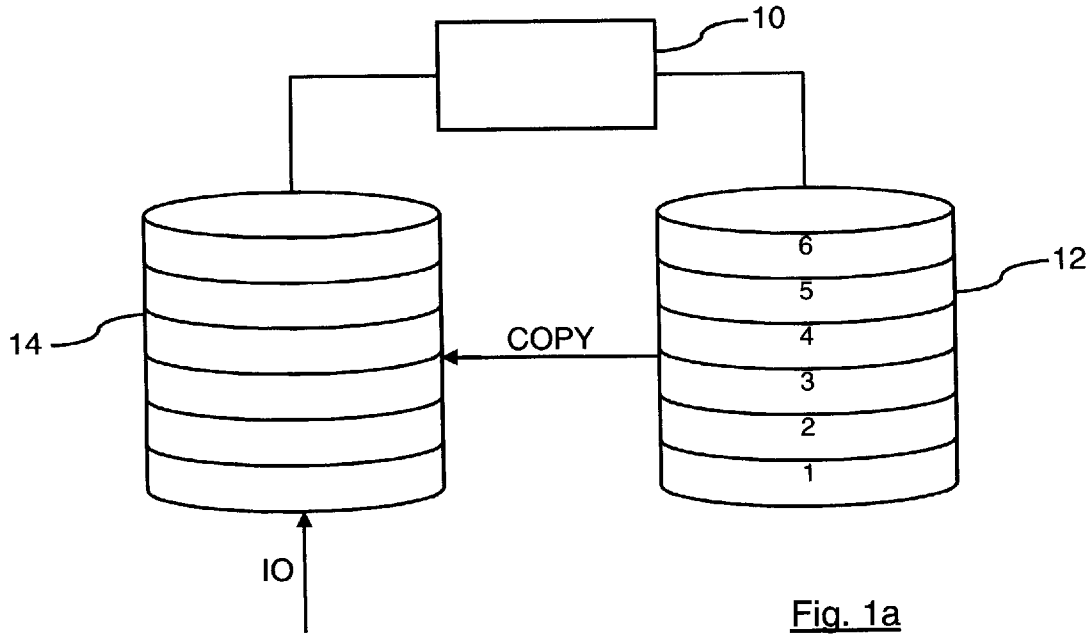

Fig. 1a

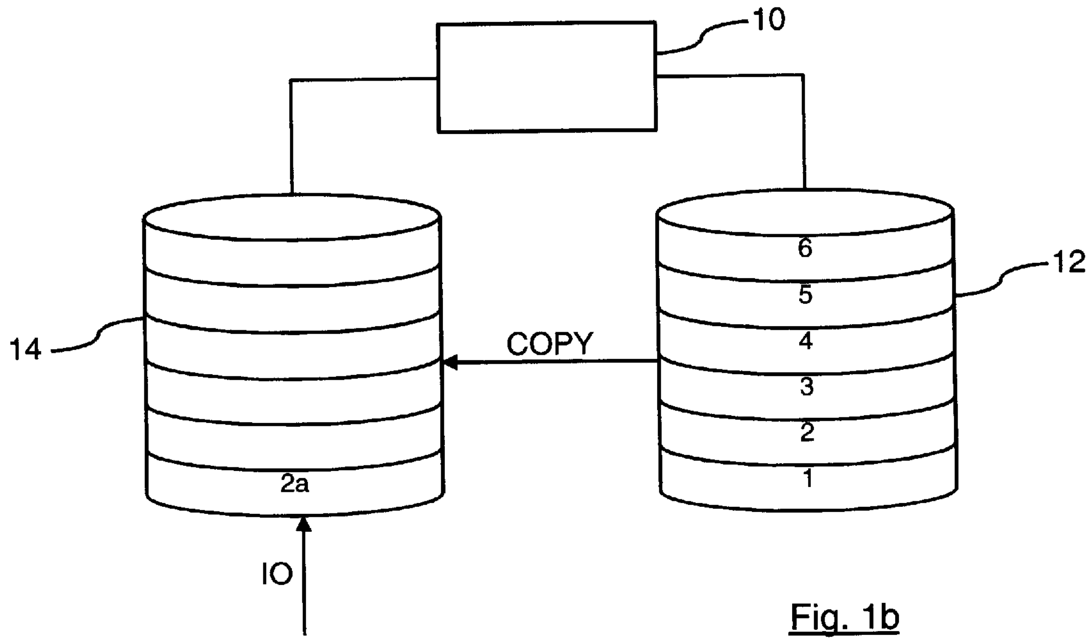

Fig. 1b

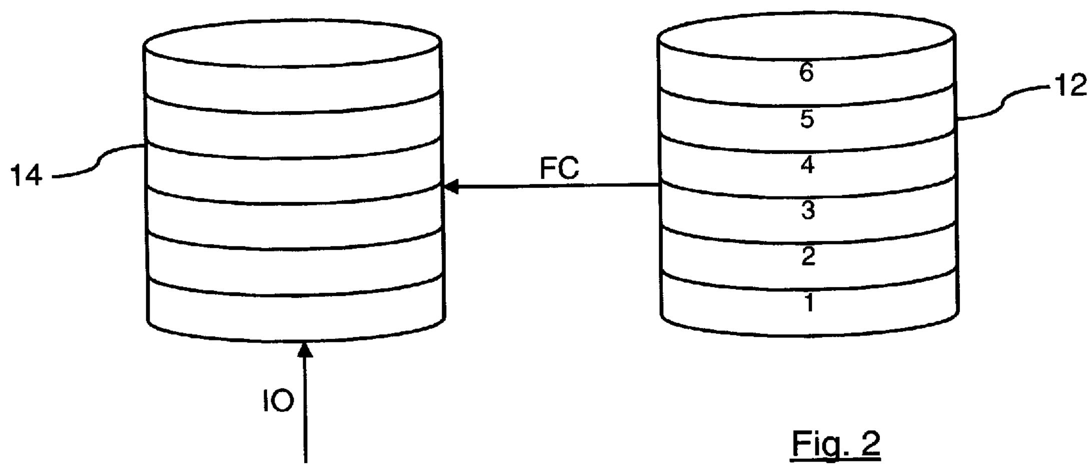

Fig. 2

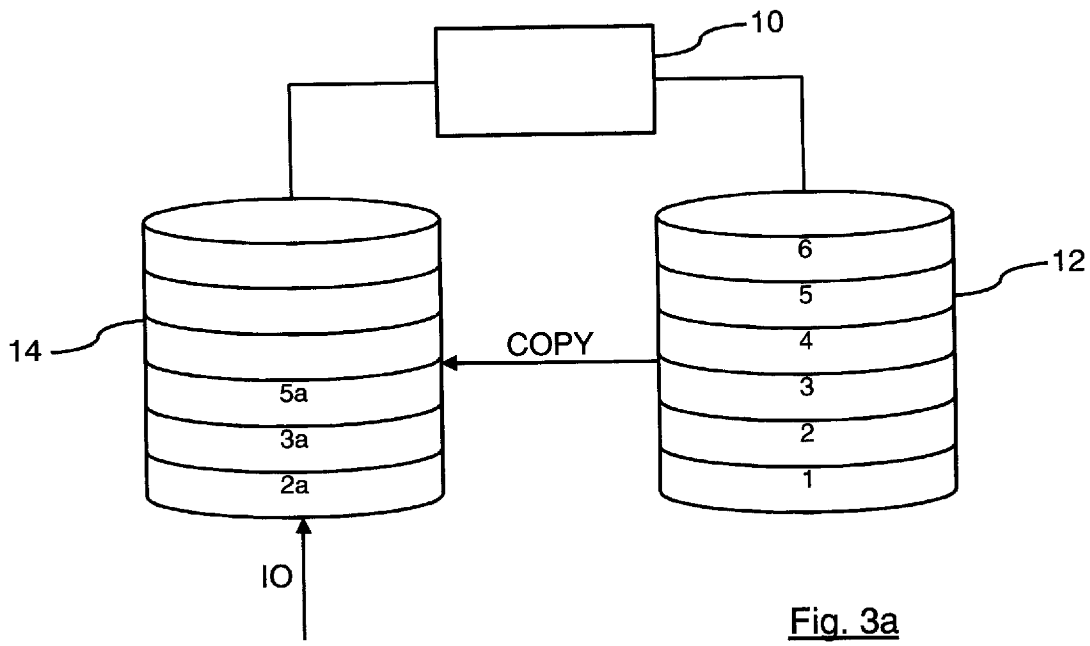

Fig. 3a

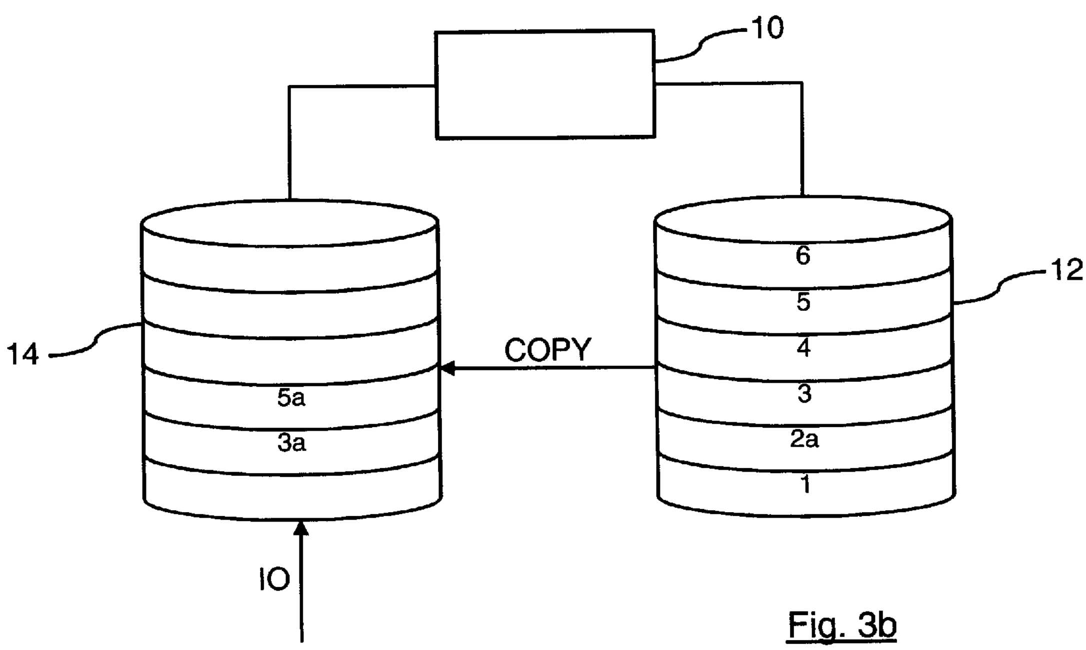

Fig. 3b

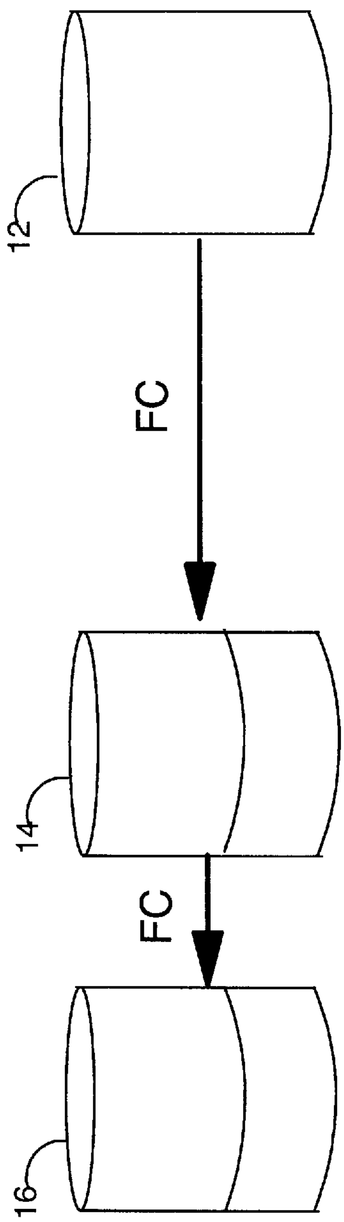

FIG. 4

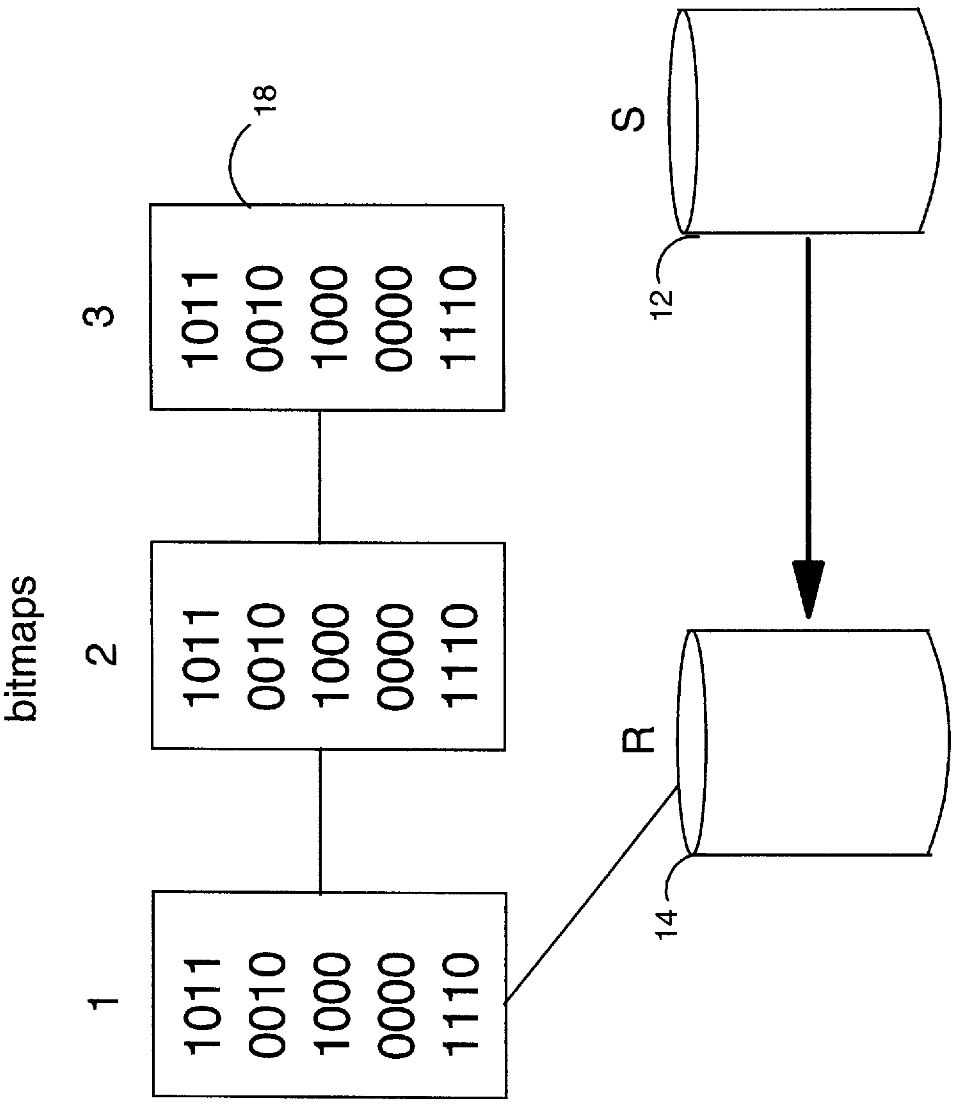

FIG. 5

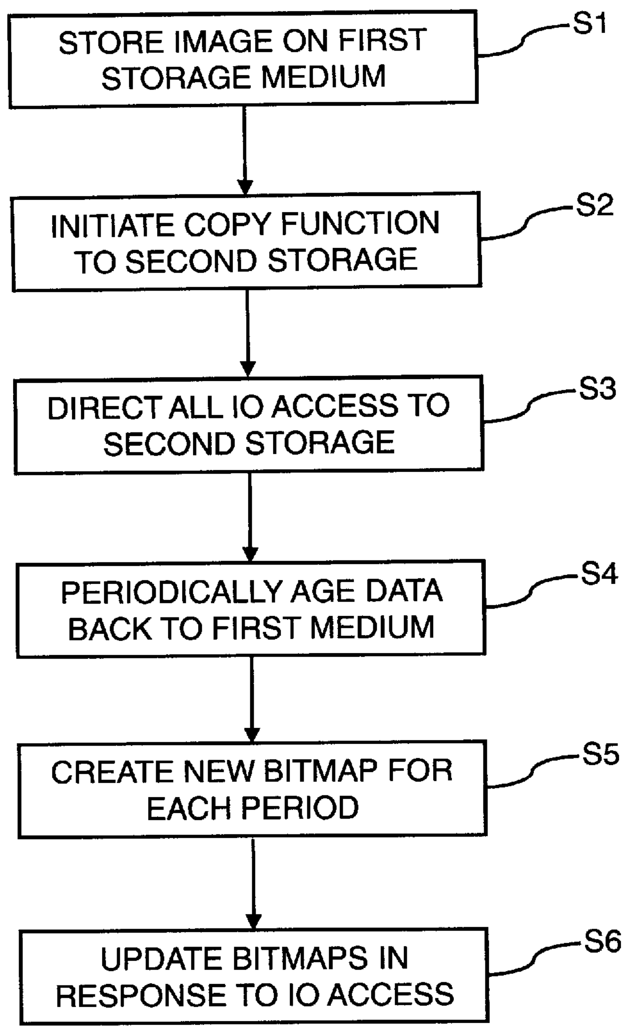
STORE IMAGE ON FIRST STORAGE MEDIUM
S1
INITIATE COPY FUNCTION TO SECOND STORAGE
S2
DIRECT ALL IO ACCESS TO SECOND STORAGE
S3
PERIODICALLY AGE DATA BACK TO FIRST MEDIUM
S4
CREATE NEW BITMAP FOR EACH PERIOD
S5
UPDATE BITMAPS IN RESPONSE TO IO ACCESS
S6

Fig. 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/062385

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F3/06
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

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

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

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2009/037662 A1 (LA FRESE LEE CHARLES [US] ET AL) 5 February 2009 (2009-02-05) figures 1-6 paragraph [0022] - paragraph [0028] paragraph [0033] - paragraph [0051] -----	1-15
Y	US 7 386 674 B1 (LANGO JASON ANSEL [US]) 10 June 2008 (2008-06-10) figures 1-4,24,32 column 7, line 40 - line 64 column 16, line 55 - column 17, line 13 column 18, line 13 - column 19, line 53 ----- -/--	1-15

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
24 September 2010	01/10/2010

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Alliot, Sylvain

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2010/062385

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	"N series Snapshot: A technical discussion" IBM 2007, XP007915000 Retrieved from the Internet: URL:http://www.redbooks.ibm.com/redpapers/pdfs/redp4132.pdf [retrieved on 2010-09-21] page 1 - page 12 -----	1-15

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2010/062385

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
US 2009037662	A1	05-02-2009	NONE	
US 7386674	B1	10-06-2008	NONE	