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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 August 2010 (12.08.2010)

PCT

(10) International Publication Number
WO 2010/089795 A1

(51) International Patent Classification:
G06F 12/12 (2006.01)

(21) International Application Number:
PCT/JP2009/000416

(22) International Filing Date:
3 February 2009 (03.02.2009)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **HITACHI, LTD.** [JP/JP]; 6-6 Marunouchi 1-chome, Chiyoda-ku, Tokyo, 1008280 (JP).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **JINDO, Kazue** [JP/JP]; c/o Hitachi, Ltd. Disk Array Systems Division, 322-2 Nakazato, Odawara-shi, Kanagawa, 2500872 (JP). **KUROKAWA, Isamu** [JP/JP]; c/o Hitachi, Ltd. Disk Array Systems Division, 322-2 Nakazato, Odawara-shi, Kanagawa, 2500872 (JP). **MORI, Akihiro** [JP/JP]; c/o Hitachi, Ltd. Disk Array Systems Division, 322-2 Nakazato, Odawara-shi, Kanagawa, 2500872 (JP). **MUTO, Junichi** [JP/JP]; c/o Hitachi, Ltd. Disk Array Systems Division, 322-2 Nakazato, Odawara-shi, Kanagawa, 2500872 (JP). **OGATA, Ran** [JP/JP]; c/o Hitachi, Ltd. Disk Array Systems Division, 322-2 Nakazato, Odawara-shi, Kanagawa, 2500872 (JP).

(74) Agent: **WILLFORT INTERNATIONAL**; Kanda Center Bldg., 5F, 3-2, Kajicho 2-chome, Chiyoda-ku, Tokyo, 1010044 (JP).

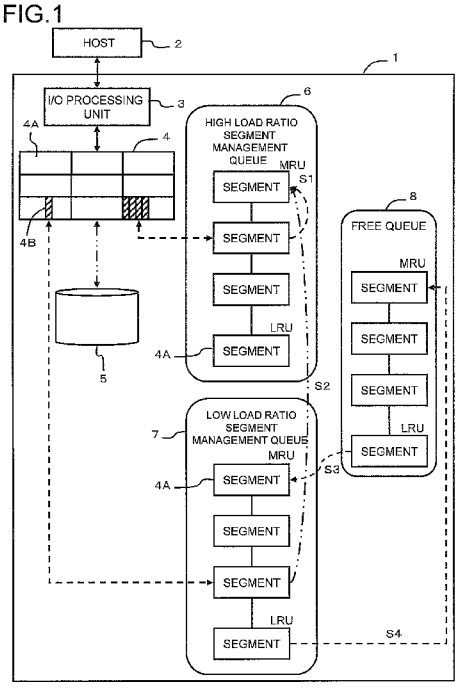
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, 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, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: STORAGE CONTROL DEVICE, AND CONTROL METHOD FOR CACHE MEMORY

FIG.1



(57) Abstract: The storage control device of the present invention uses a plurality of queues to manage cache segments which are in use, so as to retain cache segments which contain large amounts of data for long periods of time. One of the queues manages segments in which large amounts of valid data are stored. Another queue manages segments in which small amounts of valid data are stored. If the number of unused segments becomes insufficient, then a segment which is positioned at the LRU end of the other queue is released, and is shifted to a free queue. Due to the use of this other queue, it is possible to retain segments in which comparatively large amounts of data are stored for comparatively long periods of time.

WO 2010/089795 A1

Published:

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

Description

STORAGE CONTROL DEVICE, AND CONTROL METHOD FOR CACHE MEMORY

Technical Field

[0001] The present invention relates to a storage control device, and to a control method for a cache memory.

Background Art

[0002] A storage control device uses a plurality of storage devices such as hard disk drives, and supplies a storage region to a host computer on the basis of RAID (Redundant Array of Inexpensive Disks). A cache memory is used in transfer of data between the host computer and the storage devices. When a write command is outputted from the host computer, the data is written to a storage device after having been temporarily stored in the cache memory. And when a read command is outputted from the host computer, data which has been read out from the storage device is supplied to the host computer via the cache memory.

[0003] The LRU (Least Recently Used) algorithm is per se known as one algorithm for managing the cache memory. In the LRU algorithm, if the vacant capacity in the cache memory has become low, then this vacant capacity is increased by throwing away that data for which the elapsed time from when it was last used is the longest. In other words, among a plurality of segments which are present within the cache memory, the segments in which is stored data which is used the least is released as an unused segment, and subsequently new data is stored therein.

The effectiveness of the LRU algorithm decreases if the access by the host computer is random access, and moreover if the access size of the host computer is smaller than the segment size.

[0004] The hit rate decreases the more, the smaller is the access size of the host computer as compared to the segment size. For example, it will be supposed that one segment consists of eight 8 KB blocks, so that the segment size is 64 KB (= 8 KB X 8). Moreover, it will be supposed that the minimum size for random access by the host is one block.

[0005] Segments are used according to the occurrence of random accesses. In the extreme case, sometimes it happens that only 8 KB of data is stored in a segment of 64 KB. The proportion which the data occupies in this segment does not exceed 8/64 = 1/8. In this manner, the more segments occur in which the proportion of data is low, the more does the efficiency of utilization of the cache memory decrease, and as a result the hit rate decreases.

[0006] Due to this, a prior art technique described in Patent Document JP-A-2002-251322 has proposed the provision of a plurality of cache memories which can be set to different segment sizes. With this prior art technique, the size of an access from the host computer is detected, and that cache memory is selected which has a segment size appropriate for this access size which has been detected. Thus, with this prior art technique, it is aimed to enhance the hit rate by bringing the segment sizes and the size of accesses by the host computer closer to one another.

Patent Citation 1: JP-A-2002-251322

Disclosure of Invention

Technical Problem

[0007] With the above described prior art technique, it is necessary to provide a plurality of cache memories of different segment sizes, and moreover it is necessary to select a segment size for each I/O (Input/Output) by the host computer. Due to this, with this prior art technique, the structure becomes complicated.

Moreover, if the segment size is made small to match the size of an access by the host computer, then the number of segments increases. Accordingly, the size of the management table for managing the segments becomes great, so that the efficiency of utilization of the cache memories decreases. Since the management table is stored by using a portion of the cache memory, accordingly the greater the size of the management table becomes, the smaller does the region for storing user data become.

[0008] Thus, one object of the present invention is to provide a storage control device, and a control method for a cache memory, which make it possible to enhance the efficiency of utilization of the cache memory, without changing the segment size. Another object of the present invention is to provide a storage control device, and a control method for a cache memory, which make it possible to enhance the efficiency of utilization of the cache memory and to enhance the hit rate, if the host computer performs random accesses at a smaller size than the segment size. Other objects of the present invention will become clear from the following description.

Technical Solution

[0009] In order to solve the problems described above, according to a first aspect of the present invention, there is proposed a storage control device which performs data input and output between a host computer and a storage device via a cache memory, comprising: a cache control unit for controlling data input and output to and from a plurality of segments which the cache memory possesses; and a plurality of queues which are used by the cache control unit for managing the segments; wherein, in pre-determined circumstances, by controlling the queues according to the amounts of data stored in the segments, the cache control unit permutes segments in which the data

amounts are relatively small, so as to give priority to segments whose data amounts are relatively large.

[0010] And, according to a second aspect of the present invention, in the first aspect, the plurality of queues includes a first queue for managing segments in which the amount of data is relatively large, a second queue for managing segments in which the amount of data is relatively small, and a third queue for managing unused segments; the predetermined circumstances are that the size of accesses by the host computer is smaller than the size of the segments, and moreover that the host computer performs random accesses; and the cache control unit: if data requested by the host computer is not stored in any of the segments, stores the data requested by the host computer in an unused segment which is being managed with the third queue, and manages the segment with the second queue; if the data requested by the host computer is stored in one of the segments, manages the segment in which the data is stored with the first queue; and, if the number of the unused segments which are being managed with the third queue has become less than or equal to a predetermined value, manages with the third queue as an unused segment, that segment, among the segments being managed with the second queue, for which the elapsed time from its last use is the longest.

[0011] And, according to a third aspect of the present invention, in the first aspect, the predetermined circumstances are that the size of accesses by the host computer is smaller than the size of the segments, and moreover that the host computer performs random accesses.

[0012] And, according to a fourth aspect of the present invention, in the first aspect, the plurality of queues includes a first queue for managing segments in which the amount of data is relatively large, a second queue for managing segments in which the amount of data is relatively small, and a third queue for managing unused segments.

And, according to a fifth aspect of the present invention, in the fourth aspect, one single queue 105 is divided into two regions and, and the first region is used as the first queue, while the second region is used as the second queue.

And, according to a sixth aspect of the present invention, in the first aspect, the cache control unit: if data requested by the host computer is not stored in any of the segments, stores the data requested by the host computer in an unused segment which is being managed with the third queue, and manages the segment with the second queue as the segment which has been most recently used; and, if the data requested by the host computer is stored in one of the segments, manages the segment in which the data is stored with the first queue as the segment which has been used the longest ago.

[0013] And, according to a seventh aspect of the present invention, in the sixth aspect, if the number of the unused segments which are being managed with the third queue has become less than or equal to a predetermined value, the cache control unit manages

with the third queue, as an unused segment, that segment, among the segments being managed with the second queue, for which the elapsed time from its last use is the longest.

[0014] And, according to an eighth aspect of the present invention, in the sixth aspect, if the number of the unused segments which are being managed with the third queue has become less than or equal to a predetermined value, the cache control unit: compares together a first elapsed time of that segment, among the segments stored in the first queue, which has been used the longest ago, and a second elapsed time of that segment, among the segments stored in the second queue, which has been used the longest ago; and: if the first elapsed time is greater than or equal to the second elapsed time, manages with the third queue, as an unused segment, the segment, among the segments being managed with the first queue, which has been used the longest ago; and, if the first elapsed time is less than the second elapsed time, manages with the third queue, as an unused segment, the segment, among the segments being managed with the second queue, which has been used the longest ago.

[0015] And, according to a ninth aspect of the present invention, in the sixth aspect, if the number of the unused segments which are being managed with the third queue has become less than or equal to a predetermined value, the cache control unit calculates the ratio between the number of segments which are being managed with said first queue, and the number of segments which are being managed with the second queue; and manages with the third queue, as an unused segment, either that segment, among the segments being managed with the first queue, which has been used the longest ago, or that segment, among the segments being managed with the second queue, which has been used the longest ago, so that the ratio approaches a target ratio which is set in advance.

[0016] And, according to a tenth aspect of the present invention, in the sixth aspect, the storage device includes a high speed storage device whose access speed is relatively high and a low speed storage device whose access speed is relatively low; and, if the number of the unused segments being managed with the third queue is less than or equal to a predetermined value, the cache control unit manages with the third queue, as an unused segment, a segment, among the segments being managed with the second queue, which corresponds to the high speed storage device.

[0017] And, according to an eleventh aspect of the present invention, in the first aspect, if the amount of data which is stored in segments being managed with the first queue has dropped below a predetermined value which is set in advance, shifts the segment in which the amount of data has dropped below the predetermined value to the second queue; and, if the amount of data which is stored in segments being managed with the second queue is greater than the predetermined value which is set in advance, shifts the

segment in which the amount of data is greater than or equal to the predetermined value to the first queue.

[0018] And, according to a twelfth aspect of the present invention, in the first aspect: each of the segments consists of a plurality of blocks; each of the blocks consists of a plurality of sub-blocks; when predetermined data which has been requested from the host computer is not stored in any one of the segments, the cache control unit reads out the predetermined data from the storage device according to either a first mode, a second mode, or a third mode; the first mode is a mode in which only data in a sub-block corresponding to the predetermined data is read out from the storage device and stored in the cache memory; the second mode is a mode in which data from a sub-block corresponding to the predetermined data to the final sub-block in the segment which contains the sub-block is read out from the storage device and stored in the cache memory; and the third mode is a mode in which the data in all of the sub-blocks in the segment which contains a sub-block corresponding to the predetermined data is read out from the storage device and stored in the cache memory; and the cache control unit: if the predetermined data has been read out in the first mode, manages the segment in which the predetermined data is stored with the second queue; and, if the predetermined data has been read out in the second mode or the third mode, manages the segment in which the predetermined data is stored with the first queue.

[0019] And, according to a thirteenth aspect of the present invention, in the first aspect, in other than the predetermined circumstances, the cache control unit releases and reuses segments, among the segments, in order from that segment for which the elapsed time since its last use is the longest.

[0020] And, according to a fourteenth aspect of the present invention, a control method for a cache memory is for performing data input and output between a host computer and a storage device via a cache memory, and: the size of accesses by the host computer is smaller than the size of segments which make up the cache memory; a decision is made as to whether or not data to which access has been requested from the computer is stored in any of the segments; if the data to which access has been requested from the computer is stored in one of the segments, the segment in which the data is stored is shifted to a first queue for managing segments in which the proportion of data which has been staged is relatively large; if the data to which access has been requested from the computer is not stored in one of the segments, a decision is made as to whether or not any unused segment exists; if such an unused segment exists, the data to which access has been requested from the computer is read out from the storage device and stored therein, and the segment in which the data which has been read out from the storage device has been stored is put at the newest end of a second queue for managing segments in which the proportion of data which has been staged is relatively small;

and, if such an unused segment does not exist, then either that segment, among the segments which are being managed with the first queue, for which the elapsed time from last access is the longest, or that segment, among the segments which are being managed with the second queue, for which the elapsed time from last access is the longest, is used as the unused segment.

[0021] Any one, any number, or all of the means, functions or steps of the present invention may be implemented in the form of a computer program which is executed by a computer system. If, in this manner, any one, any number, or all of the structure of the present invention is implemented in the form of a computer program, then this computer program may be fixed upon any of various types of storage medium and thereby distributed, or may be transmitted via a communication network.

Furthermore, the various aspects of the present invention described above may be combined in various manners other than those explicitly described above, and such combinations are also to be considered as falling within the scope of the present invention.

Brief Description of the Drawings

[0022] [fig.1]Fig. 1 is a conceptual figure showing an embodiment of the present invention.

[fig.2]Fig. 2 is a block diagram of a storage control device.

[fig.3]Fig. 3 is a schematic figure showing the structures of a cache memory and a cache segment.

[fig.4]Fig. 4 is an explanatory figure showing a management method for the cache memory.

[fig.5]Fig. 5 is an explanatory figure showing a table for management of slots and a table for management of segments.

[fig.6]Fig. 6 is an explanatory figure showing the operation of the queues during a segment hit.

[fig.7]Fig. 7 is an explanatory figure showing the operation of the queues during a segment miss.

[fig.8]Fig. 8 is an explanatory figure showing the operation of the queues during replacement.

[fig.9]Fig. 9 is a flow chart of a cache segment control procedure.

[fig.10]Fig. 10 is a flow chart showing dequeue processing performed during the procedure of Fig. 9.

[fig.11]Fig. 11 is a flow chart showing write command processing.

[fig.12]Fig. 12 is a table showing simulation results which demonstrate the benefits of the present invention.

[fig.13]Fig. 13 is a figure showing a table for deciding priority order for replacement,

which is used in a storage control device according to a second embodiment.

[fig.14]Fig. 14 is a flow chart showing a cache segment control procedure.

[fig.15]Fig. 15 is a flow chart showing a cache segment control procedure, which is executed by a storage control device according to a third embodiment.

[fig.16]Fig. 16 is a schematic figure showing a staging mode which is executed by a storage control device according to a fourth embodiment.

[fig.17]Fig. 17 is a flow chart of a cache segment control procedure.

[fig.18]Fig. 18 is a flow chart showing a cache segment control procedure, which is executed by a storage control device according to a fifth embodiment.

[fig.19]Fig. 19 is a flow chart showing processing for selecting a cache control mode, which is executed by a storage control device according to a sixth embodiment.

[fig.20]Fig. 20 is a flow chart showing processing for setting a cache control mode, which is executed by a storage control device according to a seventh embodiment.

[fig.21]Fig. 21 is an explanatory figure showing a device management table and a RAID group management table, which are used by the storage control device according to the seventh embodiment.

[fig.22]Fig. 22 is an explanatory figure showing a tier management table.

[fig.23]Fig. 23 is an explanatory figure showing a setting screen for the cache control mode.

[fig.24]Fig. 24 is an explanatory figure showing the structure of a queue, which is used by a storage control device according to an eighth embodiment.

[fig.25]Fig. 25 is an explanatory figure showing the structure of a queue, which is used by a storage control device according to a ninth embodiment.

Explanation of the Reference Symbols

[0023] 1 storage control device

2 host computer

4 cache memory

5 logical volume

6 queue for managing low load ratio segments

7 queue for managing high load ratio segments

8 free queue

10 storage control device

100 module

121 microprocessor

130 memory package

132 cache memory

1321 cache segment

Best Modes for Carrying Out the Invention

[0024] Fig.1 is an explanatory figure showing a summary of an embodiment of the present invention. The structure shown in Fig. 1 is not to be considered as being limitative of the range of the present invention; other structures than that shown in Fig. 1 are also included within the scope of the present invention. The storage control device 1 according to this embodiment is connected to a host computer 2 (hereinafter abbreviated as the "host"). As such a host 2, for example, there may be cited a computer device such as a server computer, a mainframe computer, an engineering workstation, a personal computer, or the like.

[0025] The detailed structure of the storage control device 1 will be further described hereinafter; here, the description will concentrate upon the functions of this storage control device 1. The storage control device 1, for example may comprise an I/O processing unit 3, a cache memory 4, a storage device 5, and a plurality of queues 6, 7, and 8.

[0026] The I/O processing unit 3 performs processing to receive write commands and read commands which have been issued from the host 2, and to transmit the results of processing to the host 2.

[0027] The cache memory 4 is provided between the host 2 and the storage device 5, and temporarily stores data. This cache memory 4 comprises a plurality of segments 4A. Each of the segments 4A comprises a plurality of blocks 4B. For example, the size of a segment 4A may be 64 KB, while the size of a block 4B may be 8 KB. It should be understood that each of the blocks 4B comprises a plurality of sub-blocks.

The host 2 accesses the blocks 4B, which are of equal size (for example 8 KB), randomly. If data which is desired by the host is not stored in the cache memory 4, then this data is read out from the storage device 5, and is transferred to the cache memory 4. This reading out of data in the storage device 5 and storing it in the cache memory 4 is termed "staging". And writing of data within the cache memory 4 into the storage device 5 is termed "destaging".

[0028] Various types of device may be used as the storage device 5, such as a hard disk drive, a semiconductor memory drive, an optical disk drive or the like. The storage device 5 may also sometimes be termed the "disk drive 5".

[0029] If a hard disk drive is used as the storage device 5, then, for example, an FC (Fiber Channel) disk, a SCSI (Small Computer System Interface) disk, a SATA disk, an ATA (AT Attachment) disk, a SAS (Serial Attached SCSI) disk, or the like may be used.

If a semiconductor memory is used as the storage device, then various types of memory device may be used, such as, for example, a flash memory device (SSD: Solid State Drive), an FeRAM (Ferroelectric Random Access Memory), a MRAM (Magnetoresistive Random Access Memory), a phase change memory (Ovonic Unified

Memory), an RRAM (Resistance RAM), or the like.

[0030] A plurality of queues 6 through 8 are implemented for managing the cache memory 4. A "first queue" is a high load ratio segment management queue 6, and this is a queue for managing segments for which staging of data is performed at greater than or equal to a predetermined level. A "second queue" is a low load ratio segment management queue 7, and this is a queue for managing segments for which staging of data is performed at less than the predetermined level. And a "third queue" is a free queue 8, and is a queue for managing segments which are unused. In the figures, sometimes unused segments are shown as being free segments.

[0031] The segment which was accessed most long ago is positioned at the head end of each of these queues 6 through 8 (which is its lower end in the figure). By the segment which was accessed most long ago, is meant that segment for which the time which has elapsed since it was last used is the longest, and this segment will sometimes be herein referred to as the "LRU segment" (the Least Recently Used segment). By contrast, the segment which was accessed most recently is positioned at the tail end of each of the queues 6 through 8 (which is its upper end in the figure), and this segment will sometimes be herein referred to as the "MRU segment" (the Most Recently Used segment).

[0032] It should be understood that, in the following explanation, for convenience, the high load ratio segment management queue 6 is sometimes termed the "high load ratio" queue 6, and the low load ratio segment management queue 7 is sometimes termed the "low load ratio" queue 7.

[0033] The status of data which is stored in a segment can be one of three: "dirty", "clean", and "free". The dirty state is the state of data which has been received from the host 2 before it is written into the storage device 5. Data in the dirty state is only present within the cache memory 4. The clean state is the state of data which has been received from the host 2 after it has been written into the storage device 5. Data in the clean state is present both in the cache memory 4 and in the storage device 5. And the free state means the unused state.

[0034] If the number of segments which are being managed in the free queue 8 has become low, then, as described hereinafter, one or more segments which are being managed with the low load ratio queue 7 or the high load ratio queue 6 are shifted to the free queue 8. However, only segments in the free queue for which the data status is the clean state can be shifted to the free queue 8. Thus, segments which contain data which is in the dirty state are not shifted to the free queue 8.

[0035] The operation of this embodiment will now be explained in a simple manner. If data which is desired by the host 2 is present in any one of the segments 4A, then this is termed a "segment hit". If the data which is desired by the host 2 is present within a

segment 4A which is being managed with the high load ratio queue 6, then this segment is shifted so as to become the MRU segment of the high load ratio queue 6 (a step S1). And, if the data which is desired by the host 2 is present within a segment 4A which is being managed with the low load ratio queue 7, then this segment is shifted so as to become the MRU segment of the high load ratio queue 6 (a step S2).

[0036] If the data which is desired by the host 2 is not present within any of the segments 4A within the cache memory 4, then, among the unused segments which are being managed with the free queue 8, that unused segment which is the LRU segment is selected.

[0037] The storage control device 1 reads out the data which is desired by the host 2 from the storage device 5, and stores it in the segment which has been selected. And then this segment in which the data desired by the host 2 has been stored is linked into the low load ratio queue 7 as its MRU segment (a step S3).

[0038] In this manner, according to access from the host 2, data which is requested by the host 2 is read out from the storage device 5 and is stored in a segment 4A. If the number of unused segments becomes low, then the segment which is positioned as the LRU segment of the low load ratio queue 7 (a segment which is in the clean state) is shifted to the free queue 8 (a step S4). It should be understood that, as the method of selecting this segment which is returned to the free queue 8, any of a plurality of variant embodiments may be employed. These variant embodiments will be described hereinafter with reference to the figures.

[0039] According to this embodiment, the segments 4A are managed according to the amount of data which is being staged in the segments 4A (to put it in another manner, according to the proportion of the segment size which is occupied by the data size). In this embodiment, if the vacant capacity in the cache memory 4 has become low, then a segment 4A in the low load ratio queue 7 is released as an unused segment, and is reused.

[0040] Accordingly, in this embodiment, it is possible to keep segments for which the amount of data storage is high for longer periods of time than segments for which the data storage amount is low. Due to this, it is possible to increase the efficiency of utilization of the cache memory 4, and to enhance the hit rate. In the following, embodiments of the present invention will be explained in detail.

Embodiment 1

[0041] Fig. 2 shows the entire information processing system included in the storage control device 10. First the way this corresponds to Fig. 1 will be explained. The storage control device 10 corresponds to the storage control device 1; the host 30 corresponds to the host 2; the cache memory 132 corresponds to the cache memory 4; the segment 1321 (refer to Fig. 4) corresponds to the segment 4A; the block 1322 (refer to Fig. 4)

corresponds to the block 4B; the storage device 151 corresponds to the storage device 5; the queue 101 (refer to Fig. 6) corresponds to the queue 6; the queue 102 (refer to Fig. 6) corresponds to the queue 7; and the queue 103 (refer to Fig. 6) corresponds to the queue 8.

[0042] The storage control device 10 comprises two modules 100(1) and 100(2), and one service processor 170 (hereinafter termed the "SVP 170"). When it is not necessary particularly to distinguish between the modules 100(1) and 100(2), reference will be made to a "module 100".

[0043] Each of these modules 100 may comprise, for example, a front end package 110 (in the figure, termed a "FEPK 110"), a microprocessor package 120 (in the figure, termed a "MPPK 120"), a memory package 130 (in the figure, termed a "memory PK" 130), a back end package 140 (in the figure, termed a "BEPK 140"), and a switch 160. And each of the modules 110 can utilize a plurality of corresponding storage devices 151.

[0044] The front end package 110 is a control board which is in charge of communication with the host 30. This front end package 110 comprises a plurality of communication ports 111. The communication ports 111 are connected to the host 30 via a communication network 41. For this communication network, for example, a FC_SAN (Fiber Channel_Storage Area Network) or an IP_SAN (Internet Protocol_SAN) may be used.

[0045] In the case of a FC_SAN, the communication between the host 30 and the front end package 110 is performed according to a fiber channel protocol. If the host 30 is a mainframe, then, for example, data communication may be performed according to a communication protocol such as FICON (Fiber Connection: registered trademark), ESCON (Enterprise System Connection: registered trademark), ACONARC (Advanced Connection Architecture: registered trademark), FIBARC (Fiber Connection Architecture: registered trademark) or the like. In the case of an IP_SAN, the communication between the host 30 and the front end package 110 is performed according to the TCP/IP (Transmission Control Protocol / Internet Protocol) or the like.

[0046] The back end package 140 is a control board which is in charge of communication with the storage devices 151. This back end package 140 comprises a plurality of communication ports 141, and each of the communication ports 141 is connected to the storage devices 151.

[0047] The memory package 130 has a shared memory region 131 and a cache memory region 132. In the shared memory region 131 there are stored, for example, commands which have been received from the host 30 and various types of information for control and so on. And in the cache memory region 132 there are stored, for example, user data and tables and so on for managing the cache memory region 132. In the following explanation, the cache memory region 132 will be termed the cache memory 132.

[0048] The microprocessor package 120 is a control board for controlling the operation of this module 100. This microprocessor package 120 comprises a microprocessor 121 and a local memory 122.

[0049] The microprocessor package 120 executes commands which have been issued by the host 30, and transmits the results of their execution to the host 30. For example, if the front end package 110 has received a read command, then the microprocessor package 120 acquires the data which has been requested from the cache memory 132 or from a storage device 151, and transmits it to the host 30. And, if the front end package 110 has received a write command, then the microprocessor package 120 writes the write data into the cache memory 132, and notifies the host 30 that processing has been completed. This data which has been written into the cache memory 132 is subsequently written into a storage device 151.

[0050] Furthermore, the microprocessor package 120 performs queuing management which will be described hereinafter.

[0051] As described above, the storage devices 151 are storage devices such as hard disk drives or flash memory devices or the like. A plurality of these storage devices 151 may be collected together into one group 150. This group 150 may be, for example, a so called RAID group or parity group. And one or a plurality of logical volumes are defined in the grouped storage region.

[0052] The switches 160 are circuits for connecting the modules 100 together. Thereby, one of the modules 100(1) is able to access the memory package 130 and the storage devices 151 of the other module 100(2) via the switches 160. And, in a similar manner, that other module 100(1) is able to access the memory package 130 and the storage devices 151 of the first module 100(1) via the switches 160.

[0053] For example, the case may be investigated in which, when the microprocessor package 120 within one of the modules 100(1) is processing a command from the host 30, the data which is being requested by the host 30 is present in the cache memory 132 or a storage device 151 of the other module 100(2). In this case, the microprocessor package 120 within the first module 100(1) acquires this data from the cache memory 132 or the storage device 151 of the other module 100(2). And the microprocessor package 120 within the first module 100(1) transmits this data which has been acquired to the host 30 via the front end package 110 of this first module 100(1).

[0054] The SVP 170 is a control circuit for gathering various types of information within the storage control device 10 and supplying it to a management terminal 20, and for storing set values and so on which have been inputted from the management terminal 20 in a shared memory 131. This SVP 170 is connected to the management terminal 20 via a communication network such as, for example, a LAN (Local Area Network).

[0055] Fig. 3 is an explanatory figure, schematically showing the relationship between the

structure of the cache memory 132 and host access size and so on.

[0056] The host 30 can access all or a portion of a logical volume 152. The range which the host 30 can access will be termed the "host access range". In normal circumstances, for reasons such as keeping down the cost of manufacture and so on, the size of the cache memory 132 is set to be smaller than the host access range.

[0057] The cache memory 132 is made up of a plurality of segments 1321. The size of each segment 1321 may be, for example, 64 KB. Each of the segments 1321 is made up of a plurality of blocks 1322. The size of each block 1322 may be, for example, 8 KB. In this embodiment, the size of accesses by the host 30 (the host access size) and the block size are equal.

[0058] Each of the blocks 1322 is made up of a plurality of sub-blocks 1323. The size of each of the sub-blocks 1323 may be, for example, 512 bytes. A staging bitmap T1 is a table for managing, among the sub-blocks 1323 which are included in each segment 1321, in which of the sub-blocks 1323 the data is valid.

[0059] To express this simply, the staging bitmap T1 is a table which shows, for the various sub-blocks which make up a segment 1321, in which sub-blocks 1323 data is stored.

As explained in Fig. 3, the size of the cache memory 132 is small as compared with the host access range, which is the range that the host 30 can access. Furthermore, the host 30 performs random accesses in a size (= the block size = 8 KB) which is smaller than the size of the segments 1321. Accordingly, it becomes very important to determine which segments are preferentially left in the cache memory 132, and which segments to release.

[0060] Thus, in this embodiment, it is arranged to pay particular attention to the amount of data which is stored in a segment (= the proportion of the segment size occupied by the data size = the data load ratio), and to leave segments which include more valid data preferentially in the cache.

[0061] Fig. 4 is an explanatory figure showing a method for managing the data stored in the cache memory. In order to be able to search the data which is stored in the cache memory 132 efficiently, the storage control device 10 manages it according to the following layered structure.

[0062] First, when an I/O request is issued from the host 30, the microprocessor package 120 obtains a VDEVSLOT number (VDEVSLOT#) on the basis of the LBA (Logical Block Address) which is included in this I/O request.

[0063] And, on the basis of this VDEVSLOT number, the microprocessor package 120 refers to a VDSLOT-PAGE table T11, and acquires a pointer to the next layer. A pointer to a PAGE-DIR table T12, which is the next table, is included in the VDSLOT-PAGE table T11.

[0064] A pointer to a PAGE-GRPP table T13 is included in the PAGE-DIR table T12.

Moreover, a pointer to a GRPT1 table T14 is included in the PAGE-GRPP table T13. And a pointer to a GRPT2 table T15 is included in the GRPT1 table T14. Finally, a pointer to a SLCB table T16 (a slot control table) is included in the GRPT2 table T15.

[0065] The SLCB table T16 is reached by successively referring to these various tables T11 through T15 on the basis of the LBA. At least one or more SGCB tables T17 (segment control block tables) are established in correspondence to this SLCB table T16. These SGCB tables T17 store control information related to segments 1321, which are the minimum units in cache management. One to four segments 1321 can be put into correspondence with a single slot. For example, 64 KB of data may be stored in a single segment 1321.

[0066] Although the minimum unit for cache management is a segment, state transition between the dirty data state (data in its state of not yet having been written to a storage device 151) and the clean data state (data in its state of having already been written to a storage device 151) is performed in units of slots. Taking (i.e. reserving) and also releasing (i.e. releasing or replacing) cache regions may be performed in units of slots or units of segments.

[0067] Fig. 5 is an explanatory figure showing examples of the structure of the SLCB table T16 and the SGCB table T17. As shown in Fig. 5, the SLCB table T16 may include, for example, a backward pointer, a forward pointer, a VDEVSLOT number, a queue status, a slot status, ad a SGCB pointer.

[0068] The queue status includes a queue classification and a queue number, held in this SLCB table T16 in mutual correspondence. The slot status includes the state of the slot which corresponds to this SLCB table T16. And the SGCB pointer includes a pointer for specifying the SGCB table T17 which corresponds to this SLCB table T16.

In the SGCB table T17, for example, there may be included a backward pointer, a forward pointer, a staging bitmap T1, and a SLCB pointer.

[0069] The cache algorithm will now be explained on the basis of Figs. 6 through 10. Fig. 6 is an explanatory figure showing the case when a segment hit has occurred. Here, a segment hit means that data which is the subject of access by the host 30, and data which is stored in a segment 1321, agree with one another.

[0070] In this embodiment, the cache is controlled by using queues 101 through 103 of three types. The first queue is a high load ratio segment management queue 101 for managing segments in which a lot of data is included. The second queue is a low load ratio segment management queue 102 for managing segments in which not a great deal of data is included. And the third queue is a free queue 103 for managing segments which are unused. In the following, the abbreviated terms " high load ratio queue 101" and " low load ratio queue 102" will be employed. It should be understood that, for the convenience of depiction, the terms "high load queue" and "low load queue" are

sometimes used in the drawings.

[0071] It should be understood that, for example, the high load ratio queue 101 may be termed the queue for managing segments which are kept preferentially, while the low load ratio queue 102 may be termed the queue for managing normal segments.

[0072] When a segment in which data which is the subject of access is stored is being managed with the high load ratio queue 101, this segment is shifted to be the MRU segment of the high load ratio queue 101. By contrast, when a segment in which data which is the subject of access is stored is being managed with the low load ratio queue 102, likewise, this segment is shifted to be the MRU segment of the high load ratio queue 101. In other words, in this embodiment, a segment which has been hit is subsequently managed with the high load ratio queue 101.

[0073] Fig. 7 is an explanatory figure showing the case when a segment miss has occurred. A segment miss means the state in which data which is the subject of access by the host 30 is stored neither in the cache memory 132, nor in any segment 1321.

[0074] If a segment miss has occurred, then, among the unused segments which are being managed with the free queue 103, that unused segment which is positioned as the LRU segment is selected, and the data which is the subject of access is stored in this segment which has been selected. And this segment in which the data which is the subject of access has been stored is shifted to become the MRU segment of the low load ratio queue 102.

[0075] Fig. 8 shows a situation in which a segment in which data is stored is removed from the low load ratio queue and is shifted to the free queue. If the vacant capacity of the cache memory 132 has become low, then replacement of some data is performed. To put this in another manner, if the number of unused segments which are being managed with the free queue has become low, then some of the segments in which data is stored are made into unused segments, and are shifted to the free queue 103.

[0076] In this embodiment, the segment which is positioned as the LRU of the low load ratio queue 102 is shifted to be the MRU of the free queue 103, and is re-used as an unused segment.

[0077] Fig. 9 is a flow chart showing processing for controlling the cache segments. It should be understood that the various flow charts shown below provide summaries of various processes as far as required for the understanding and implementation of the present invention, but may differ from the computer programs which are actually used in various ways. A person skilled in the art will be able to make various changes or alterations to the steps shown in these figures, or to add new steps or the like.

[0078] The microprocessor 121 of the microprocessor package 120 first makes a decision as to whether or not a segment miss has occurred (a step S10). If a segment hit has occurred (NO in the step S10), then the microprocessor 121 shifts this segment which

has been hit to the MRU end of the high load ratio queue 101 (a step S11).

[0079] On the other hand, if a segment miss has occurred (YES in the step S10), then the microprocessor 121 makes a decision as to whether or not any unused segment is present in the free queue 103 (a step S12). If some unused segment remains (YES in the step S11), then the microprocessor 121 takes this unused segment (a step S13). And the microprocessor 121 stores the data which is the subject of access in this segment which has been taken, and then connects this segment into the low load ratio queue 102 at its MRU end (a step S14).

[0080] If no unused segment is present in the free queue 103 (NO in the step S12), then the microprocessor 121 performs dequeue processing (a step S15).

Fig. 10 is a flow chart showing this dequeue processing which is performed during the step S15 of Fig. 9. In this processing, fundamentally, a segment is taken from the LRU end of the low load ratio queue 102, and is shifted to the free queue 103 (a step S29).

[0081] However, if within the high load ratio queue 101 a segment is present which has been managed for a long time (YES in the step S23), or if the ratio of the length of the high load ratio queue 101 and the length of the low load ratio queue 102 is outside some fixed range (NO in the step S28), then this segment is removed from the LRU of the high load ratio queue 101 and is shifted to the free queue 103 (a step S24).

[0082] For the segment which is positioned at the LRU end of the high load ratio queue 101, the microprocessor 121 acquires its LRU time period TH (a step S20). This LRU time period is the time period which has elapsed from when this segment was positioned at the LRU end of this queue.

[0083] And, for the segment which is positioned at the LRU end of the low load ratio queue 102, the microprocessor 121 also acquires its LRU time period TL (a step S21). Then the microprocessor 121 calculates the time difference dT (delta-T) between the LRU time period TH for the high load ratio queue 101 and the LRU time period TL for the low load ratio queue 102 (a step S22).

[0084] The microprocessor 121 makes a decision as to whether or not this time difference dT is greater than or equal to a predetermined time period TS which is set in advance (a step S23). If the time difference dT is greater than or equal to the predetermined time period TS (YES in the step S23), then the microprocessor 121 removes the segment which is positioned at the LRU end of the high load ratio queue 101, and shifts it to the MRU end of the free queue 103 (a step S24). This is because the segment which is positioned at the LRU end of the high load ratio queue 101 has not been used for a longer time period than the segment which is positioned at the LRU end of the low load ratio queue 102, and it may be anticipated that it will not be used much in the future.

[0085] If the time difference dT is less than the predetermined time period TS (NO in the step S23), then the microprocessor 121 acquires the number NH of segments which are being managed with the high load ratio queue 101 (a step S25). In a similar manner, the microprocessor 121 acquires the number NL of segments which are being managed with the low load ratio queue 102 (a step S26). And the microprocessor calculates the ratio RN (=NL/NH) between the number of segments NL in the low load ratio queue 102 and the number of segments NH in the high load ratio queue 101 (a step S27).

[0086] Then the microprocessor 121 makes a decision as to whether or not this ratio RN is greater than or equal to a predetermined ratio NTh which is set in advance (a step S28). If the ratio RN is greater than or equal to the predetermined ratio NTh (YES in the step S28), then the microprocessor 121 removes the segment which is positioned in the LRU position in the low load ratio queue 102, and shifts it to the MRU position in the free queue 103 (a step S29).

[0087] However, if the ratio RN is less than the predetermined ratio NTh (NO in the step S28), then the microprocessor 121 removes the segment which is positioned in the LRU position in the high load ratio queue 101, and shifts it to the MRU position in the free queue 103 (a step S24). Since the number of segments NH in the high load ratio queue 101 is much greater than the number of segments NL in the low load ratio queue 102, accordingly a segment which is being managed with the high load ratio queue 101 is removed, in order to obtain a balance between these numbers of segments. The above completes the explanation of the method of cache memory management in this embodiment. Next, the method of processing a normal command will be explained.

[0088] Fig. 11 shows the processing when a write command has been issued from the host 30. When the host 30 issues a write command (a step S40), the front end package 110 of the first module 100(1) receives this write command (a step S41).

[0089] The microprocessor 121 of the first module 100(1) checks the write destination of this write command (a step S42). And the microprocessor 121 makes a decision as to whether or not the write destination of the write command is a logical volume 152 which is under the management of this first module 100(1) (a step S43).

[0090] If the first module 100(1) has the logical volume 152 which is the write destination (YES in the step S43), then the microprocessor 121 writes the write data to the logical volume 152 which is the write destination, via the back end package 140 (a step S44). And then the microprocessor 121 notifies the host 30 that the write has been completed (a step S45).

[0091] But, if the write destination specified in the write command is a logical volume 152 which is being managed by the second module 100(2) (NO in the step S43), then the microprocessor 121 of the first module 100(1) stores the write data in the cache memory 132 within the first module 100(1) (a step S46). And the microprocessor 121

of the first module 100(1) delegates the processing of the write command to the microprocessor 121 of the second module 100(2) (a step S47).

[0092] When the microprocessor 121 of the second module 100(2) receives the delegation for processing the write command (a step S48), it reads out the write data from the cache memory 132 of the first module 100(1) (a step S49).

[0093] The microprocessor 121 of the second module 100(2) writes the write data to the logical volume 152 of the second module 100(2) (a step S50), and then notifies the microprocessor 121 of the first module 100(1) that the write has been completed (a step S51).

[0094] When the microprocessor 121 of the first module 100(1) receives this notification from the microprocessor 121 of the second module 100(2), it notifies the host 30 that the processing of the write command has been completed (a step S52). And, by receiving this notification, the host 30 is able to confirm that the processing of the write command has been completed (a step S53).

[0095] The case of a read command is a little different, but is processed in almost a similar manner, according to the flow chart shown in Fig. 11. For reference, the way in which a read command is processed will now be explained. If the logical volume upon which the data which is the subject of being read is held by the first module 100(1), then the first module 100(1) reads out the data which has been requested from that logical volume 152 and transfers it to the cache memory 132. And the first module 100(1) transmits this read subject data to the host 30.

[0096] But, if the logical volume 152 in which the read data is stored is under the management of the second module 100(2), then the first module 100(1) delegates the processing of the read command to the second module 100(2). The second module 100(2) reads out the read subject data from the logical volume 152, and stores it in the cache memory 132 of the second module 100(2). And the second module 100(2) notifies the address which is the destination of storage of the read subject data to the first module 100(1). Then the first module 100(1) reads out the read subject data from the cache memory 132 within the second module 100(2), and transmits this data to the host 30.

[0097] In this embodiment, as explained above, the cache segments are managed so as to retain segments in which a lot of data is stored as long as possible, while paying attention to discrepancies in the data load ratio (the staging ratio). Accordingly, as shown in Fig. 12, by increasing the efficiency of utilization of the cache memory 132, it is possible to enhance the hit rate.

[0098] Fig. 12 is a table showing the simulation results which demonstrate the benefits of this embodiment of the present invention. In "cache size / host access range" in the left column of the table there is displayed, in percent, the proportion of the range which the

host accesses, which the cache size occupies. For example, "100%" means that the cache size and the host access range are equal, while "25%" means that the cache size is 1/4 of the host access range.

[0099] On the right side of the table, for each size of the cache segments, the respective hit rates are shown. In this simulation, calculations have been performed for four different segment sizes: 64 KB, 32 KB, 16 KB, and 8 KB. It should be understood that the size of the accesses of the host 30 is 8 KB, and moreover it is hypothesized that the host 30 is performing accesses randomly.

[0100] If the cache size and the host access range agree with one another (i.e. if the cache size / the host access range = 100%), then all of the data that can be accessed by the host 30 is stored in the cache memory 132. Accordingly in this case the hit rate is 100%, irrespective of the value of the segment size.

[0101] If the cache size is 75% of the host access range, then there are discrepancies in the hit rate, according to the relationship between the segment size and the host access size. For example, if the cache segment size is 64 KB, then the hit rate of the normal LRU algorithm becomes 27%. This is because, if the host access size (=8 KB) is too small with respect to the size of the cache segments (=64 KB), then it is not possible to utilize the cache segments effectively. However, according to the present invention, it is possible to enhance the hit rate to 68%.

[0102] In this manner, according to this embodiment, queuing is performed according to the number of data stored in the cache segments, and cache segments in which more data is stored are retained for longer times than cache segments in which the amount of data stored is low. Thus, according to this embodiment, as shown in Fig. 12, when the host access size is small as compared to the cache segment size, and when moreover the host performs random access, then it is possible to increase the efficiency of utilization of the cache memory and to enhance the hit rate

Embodiment 2

[0103] A second embodiment will now be explained on the basis of Figs. 13 and 14. The following embodiment, including this second embodiment, are equivalent to variant embodiments of the first embodiment. Accordingly, in the following descriptions of these embodiments, the explanation will focus upon the aspects in which they differ from the first embodiment. In this second embodiment, the replacement is controlled according to the type of the storage device and the amount of data which is stored in the cache segments.

[0104] In this embodiment a mixture of high speed storage devices such as SSDs and low speed storage devices such as SATAs is used. For example, the storage control device 10 may include high speed logical volumes which are defined on the basis of SSDs and low speed logical volumes which are defined on the basis of SATAs. Or, as another

example, a structure may be conceived in which one region in a logical volume is defined on the basis of an SSD, and another region is defined on the basis of a SATA.

[0105] It should be understood that, in the flow chart which is described hereinafter, for the convenience of illustration, the abbreviation "low speed devices" is used for the low speed storage devices, and the abbreviation "high speed devices" is used for the high speed storage devices.

[0106] Fig. 13 shows a table T20 for deciding a priority order for replacement, according to the type of storage device and the type of queue. If the number of unused segments which are being managed with the free queue 103 has become low, then segments which are in used are returned to being free segments, according to the priority order shown in this table T20.

[0107] Among the cache segments which correspond to high speed storage devices, those segments which are being managed with the low load ratio queue 102 are the highest ones in the priority order. And, among the cache segments which correspond to low speed storage devices, those segments which are being managed with the low load ratio queue 102 are the second (or third) ones in the priority order. Among the cache segments which correspond to high speed storage devices, those segments which are being managed with the high load ratio queue 101 are the second (or third) ones in the priority order. And, among the cache segments which correspond to low speed storage devices, those segments which are being managed with the high load ratio queue 101 are the lowest ones in the priority order.

[0108] Since the high speed storage devices are the ones whose access speeds are high, accordingly the penalty is small if a segment miss occurs. By contrast, since the low speed storage devices are the ones whose access speeds are low, accordingly the penalty is high if a segment miss occurs. Thus, in this embodiment, it is arranged to keep segments which contain more data and whose data is stored in low speed storage devices (i.e., fourth order segments) in the cache memory for as long as possible.

[0109] Fig. 14 is a flow chart showing the cache segment control procedure. If a segment miss has occurred (YES in the step S10), and there are no unused segments in the free queue 103 or if the number of unused segments is insufficient (NO in the step S12), then the microprocessor 121 makes a decision as to whether or not the segment which is positioned at the LRU end of the low load ratio queue 102 corresponds to a high speed device (a step S60).

[0110] If the segment which is positioned at the LRU end of the low load ratio queue 102 corresponds to a high speed device (YES in the step S60), then the microprocessor 121 dequeues this segment which is positioned at the LRU end of the low load ratio queue 102, and shifts it to the free queue 103 (a step S61). In other words, the microprocessor 121 changes the status of the segment of the first order shown in Fig. 13 to the free

state (the step S61).

[0111] On the other hand, if the segment which is positioned at the LRU end of the low load ratio queue 102 does not correspond to a high speed device (NO in the step S60), then the microprocessor 121 makes a decision as to whether or not the segment which is positioned at the LRU end of the high load ratio queue 101 corresponds to a high speed ratio storage device (a step S62).

[0112] If the segment which is positioned at the LRU end of the high load ratio queue 101 corresponds to a high speed device (YES in the step S62), then the flow of control is transferred to the step S61, and dequeues the segment which is positioned at the LRU end of the low load ratio queue 102. In other words, in this case, this segment which is positioned at the LRU end of the low load ratio queue 102, and which corresponds to a low speed storage device, is dequeued.

[0113] Thus, if the segment which is positioned at the LRU end of the high load ratio queue 101 corresponds to a high speed device (YES in the step S62), then the dequeue processing described in Figs. 9 and 10 is performed (the steps S15 and S20 through S29).

[0114] With this second embodiment having the structure described above, a similar advantageous effect is obtained as in the case of the first embodiment. In addition thereto, with this second embodiment, since the replacement is performed according to the access speeds of the storage devices, accordingly it is possible to leave segments which are in low speed storage devices and which contain a lot of data in the cache memory for comparatively long periods of time.

Embodiment 3

[0115] A third embodiment will now be explained on the basis of Fig. 15. In this third embodiment, the queuing is performed according to the amounts of data stored in the segments (i.e., the data load ratios).

[0116] Fig. 15 is a flow chart showing the cache segment control procedure. If a segment miss has occurred (YES in the step S10), and moreover there are enough unused segments present in the free queue 103 (YES in the step S12), then the microprocessor 121 selects an unused segment and stores the write data in it.

[0117] The microprocessor 121 selects a segment to be connected into a queue according to the amount of data stored in the segment (a step S70). And the microprocessor 121 makes a decision as to whether or not the amount of data which the amount of data which is stored in this segment is greater than or equal to 30% of the segment size (the step S70). In this specification, the proportion of the amount of data which is stored in a segment to the size of that segment is termed the "data load ratio". The data load ratio may be obtained by referring to the staging bitmap T1.

[0118] If the data load ratio is greater than or equal to 30% (YES in the step S70), then the

microprocessor 121 positions this segment in which the data is stored at the MRU end of the high data load ratio queue 101 (a step S14). By contrast, if the data load ratio is less than 30% (NO in the step S70), then the microprocessor 121 positions this segment in which the data is stored at the MRU end of the low data load ratio queue 102 (a step S71).

[0119] If a segment hit has occurred (NO in the step S10), then the microprocessor 121 makes a decision as to whether or not the segment which has been hit is being managed with the high data load ratio queue 101 (a step S72). If this segment is being managed with the high data load ratio queue 101 (YES in the step S72), then the microprocessor 121 shifts this segment to the MRU end of the high data load ratio queue 101 (a step S73).

[0120] But, if the segment which has been hit is being managed with the low data load ratio queue 102 (NO in the step S72), then the microprocessor 121 makes a decision as to whether or not the data load ratio of this segment is greater than or equal to 30% (a step S74).

[0121] If the data load ratio is greater than or equal to 30% (YES in the step S74), then the microprocessor 121 positions this segment at the MRU end of the high data load ratio queue 101 (a step S75). But, if the data load ratio is less than 30% (NO in the step S74), then the microprocessor 121 positions this segment at the MRU end of the low data load ratio queue 102 (a step S76).

[0122] With this third embodiment having the structure described above, a similar advantageous effect is obtained as in the case of the first embodiment. In addition thereto, with this third embodiment, since the data load ratios of the segments are detected by referring to the staging bitmap T1, accordingly it is possible to use the high data load ratio queue 101 and the low data load ratio queue 102 separately on the basis of the staging mode. It should be understood that the 30% above is only one example of the threshold value; the present invention should not be considered as being limited by this value of 30%.

Embodiment 4

[0123] A fourth embodiment will now be explained on the basis of Figs. 16 and 17. In this fourth embodiment, usage is divided between the high data load ratio queue 101 and the low data load ratio queue 102, according to the type of staging mode (transfer mode) by which the data is transferred from the storage device 151 to the cache memory 132.

[0124] Fig. 16 is an explanatory figure, schematically showing this staging mode. In this embodiment, three different staging modes may be used: record load, half load, and full load.

[0125] Record load, which is a "first mode", is a mode in which only a record (i.e. a sub-

block) which has been requested is staged. Half load, which is a "second mode", is a mode in which data within the segment to which the requested record belongs, from that record up to the final record, is staged. And full load, which is a "third mode", is a mode in which all of the data in the segment to which the requested record belongs is staged.

[0126] An access pattern history management unit 123 functions for managing the history of the access pattern of the host 30. This staging mode determination unit 124 functions for determining the staging mode, and determines any one among the three staging modes described above, on the basis of the history of the access pattern. And data is transferred from the storage devices 151 to the cache memory 132 according to the staging mode which has thus been confirmed.

[0127] And an I/O (Input/Output) processing unit 125 performs processing of commands from the host 30, using the data which is stored in the cache memory 132.

[0128] Fig. 17 is a flow chart showing the cache segment control procedure. If a segment miss has occurred (YES in the step S10), then, from among the unused segments which are being managed with the free queue 103, the microprocessor 121 reserves only the necessary unused segments (YES in the step S12, and the step S13).

[0129] And the microprocessor 121 makes a decision as to whether or not the current staging mode is full load or half load (a step S80). If the current staging mode is either full load or half load (YES in the step S80), then the microprocessor 121 positions the segment in which the data is stored at the MRU end of the high data load ratio queue 101 (a step S14). Since a segment from which data has been read out at full load or half load is one in which a lot of data is stored, accordingly it is managed with the high data load ratio queue (the step S14).

[0130] By contrast, if the current staging mode is record load (NO in the step S80), then the microprocessor 121 positions the segment in which the data is stored at the MRU end of the low data load ratio queue 102 (a step S81). Since, in the case of record load, only the data which has been requested is staged, accordingly it is considered that the proportion of the segment which the data occupies (in other words, the data load ratio) is comparatively small. Accordingly, this segment is managed with the low data load ratio queue 102 (a step S81).

[0131] If there has been a segment hit (NO in the step S10), then the microprocessor 121 makes a decision as to whether or not the segment which has been hit is being managed with the high data load ratio queue 101 (a step S82). And, if this segment which has been hit is being managed with the high data load ratio queue 101 (YES in the step S82), then the microprocessor 121 positions this segment at the MRU end of the high data load ratio queue 101 (a step S83).

[0132] But, if the segment which has been hit is being managed with the low data load ratio

queue 102 (NO in the step S82), then the microprocessor 121 makes a decision as to whether or not the staging mode in relation to this segment which has been hit is either full load or half load (a step S84).

[0133] If the staging mode related to this segment is either full load or half load (YES in the step S84), then the microprocessor 121 positions this segment at the MRU end of the high data load ratio queue 101 (a step S85). But, if the staging mode related to this segment is record load (NO in the step S84), then the microprocessor 121 positions this segment at the MRU end of the low data load ratio queue 102 (a step S86).

[0134] With this fourth embodiment having the structure described above, a similar advantageous effect is obtained as in the case of the first embodiment. In addition thereto, with this fourth embodiment, since the high data load ratio queue 101 and the low data load ratio queue 102 are used separately on the basis of the staging mode, accordingly it is possible to simplify the structure, as compared to the method of referring to a staging bitmap T1.

Embodiment 5

[0135] A fifth embodiment will now be explained on the basis of Fig. 18. In this fifth embodiment, if the segment which has been hit is being managed with the high data load ratio queue 101 (YES in a step S90), then, if the data load ratio of this segment is less than a first threshold value (for example, 60%) (NO in a step S91), then this segment is positioned, not to the MRU end of the high data load ratio queue 101, but rather to an intermediate position in the high data load ratio queue 101 (a step S93).

[0136] Fig. 18 is a flow chart of this cache segment control procedure. If a segment hit has occurred (NO in the step S10), then the microprocessor 121 makes a decision as to whether or not the segment which has been hit is being managed with the high data load ratio queue 101 (a step S90).

[0137] If the segment which has been hit is being managed with the high data load ratio queue 101 (YES in the step S90), then the microprocessor 121 makes a decision as to whether or not the data load ratio of this segment is greater than or equal to 60% (a step S91). It should be understood that this first threshold value is not limited to being 60%; some other value would also be acceptable.

[0138] If the data load ratio is less than 60% (NO in the step S91), then the microprocessor 121 positions this segment at an intermediate position in the high load ratio queue 101 (for example, at the very center of this queue) (a step S93). Due to this, a segment for which the effective data amount which is stored is small (i.e. a segment for which the data load ratio is small) does not become positioned at the MRU end of the high load ratio queue 101, even if it is a segment which is being managed with the high load ratio queue 101.

[0139] On the other hand, if the segment which has been hit is being managed with the low

load ratio queue 102 (NO in the step S90), then the microprocessor 121 makes a decision as to whether or not the data load ratio of this segment is greater than or equal to 30% (a step S94). And, if the data load ratio is greater than or equal to 30% (YES in the step S94), then the microprocessor 121 shifts this segment from the low load ratio queue 102 to an intermediate position in the high load ratio queue 101 (a step S93).

[0140] On the other hand, if the data load ratio is less than 30% (NO in the step S94), then the microprocessor 121 positions this segment at the MRU end of the low load ratio queue 102 (a step S95).

[0141] With this fifth embodiment having the structure described above, a similar advantageous effect is obtained as in the case of the first embodiment. In addition thereto, with this fifth embodiment, in the case of a segment hit, the position in the queue is changed according to the data load ratio of that segment. Accordingly it is possible to perform queuing in a more detailed manner.

Embodiment 6

[0142] A sixth embodiment will now be explained on the basis of Fig. 19. In this sixth embodiment, different cache algorithms are used when access by the host 30 is sequential access, and when it is random access.

[0143] Fig. 19 is a flow chart showing processing for selecting the cache control mode (i.e. the cache control algorithm). The microprocessor 121 analyzes the command which it has received from the host 121, and determines whether or not sequential access is set in that command (a step S100). In other words, the microprocessor 121 decides whether or not the access by the host 30 is sequential access, by analyzing the command received from the host 30.

[0144] If it has been decided that the access is sequential access (YES in the step S100), then the microprocessor 121 selects normal LRU control (a step S101). In other words, in this normal LRU control, if the number of unused segments has become low, then the segment which is positioned at the LRU end is returned to the heap of unused segments.

[0145] But if sequential access is not set within the command received from the host 30 (NO in the step S100), then the microprocessor 121 makes a decision as to whether or not the slot status shown in Fig. 5 is set to "sequential" (a step S102). For example if it has been decided by an algorithm which learns the access pattern that sequential access is taking place, then "sequential" is set in the slot status.

[0146] If the slot status is set to sequential (YES in the step S102), then the flow of control is transferred to the step S101 and normal LRU control is selected (the step S101). But, if the slot status is not set to sequential, then the microprocessor 121 makes a decision as to whether or not the hit rate by normal LRU control is greater than or equal to a predetermined value which is set in advance (a step S103).

[0147] If the hit rate during LRU control is greater than or equal to the predetermined value (YES in the step S103), then the microprocessor 121 decides that sequential access is taking place, and the flow of control is transferred to the step S101. But, if the hit rate during LRU control is less than the predetermined value (NO in the step S103), then the microprocessor 121 selects control as described with reference to the first embodiment of the present invention. In other words, in the case of random access, control is performed so as preferentially to retain segments whose data load is high.

It should be understood that the action at this stage is not limited to being processing according to the first embodiment; in this step S104, it would also be acceptable to arrange to select control according to some other embodiment of the present invention.

[0148] With this sixth embodiment having the structure described above, a similar advantageous effect is obtained as in the case of the first embodiment. In addition thereto, with this sixth embodiment, normal LRU control is selected during sequential access, and cache control according to the present invention is selected during random access. Accordingly it is possible to select a control method which is well adapted to the access pattern of the host 30.

Embodiment 7

[0149] A seventh embodiment will now be explained on the basis of Figs. 20 through 23. In this seventh embodiment, it is possible for the user to specify in advance, from the management terminal 20, whether normal LRU control is to be performed, or the special cache control according to the present invention is to be performed.

[0150] Fig. 20 is a flow chart showing processing which the user executes using the management terminal 20, for setting a control mode for each logical volume. The user accesses the SVP 160 via the management terminal 20, and causes the management screen to be displayed (a step S110).

[0151] Via the management screen, the user edits various cache segment control management tables T30 through T32 as appropriate, and stores them (a step S111). These cache management tables T30 through T32 will be described hereinafter.

The microprocessor 121 refers to these management tables T30 through T32 (a step S112), and makes a decision as to whether or not normal LRU control is set as the cache control mode (a step S113).

[0152] If normal LRU control is set (YES in the step S113), then the microprocessor 121 performs normal LRU control (a step S114). But, if the control method according to the present invention is selected (NO in the step S113), then, as described in connection with the various embodiments, control is performed so as preferentially to retain segments whose data load ratios are high (a step S115).

[0153] Examples of the structures of the management tables T30 through T32 will now be explained with reference to Figs. 21 and 22. Fig. 21 is an explanatory figure showing

the device management table T30 and the RAID group management table T31.

[0154] The device management table T30 is a table for management of the structure of the logical volumes 152, which are logical storage devices. This device management table T30, for example, may perform its management by maintaining, in mutual correspondence, device numbers C301, RAID levels C302, and RAID group IDs C303.

[0155] The device numbers C301 are information for identifying the logical volumes 152 which belong to this storage device 10. The RAID levels C302 are information specifying the RAID structures which are applied to these logical volumes 152. And the RAID group IDs C303 are information for identifying the RAID groups 150 to which the various logical volumes 152 belong.

[0156] The RAID group management table T31 is a table for management of the RAID groups 150. This RAID group management table T31, for example, may perform its management by maintaining, in mutual correspondence, RAID group IDs C311 and drive types C312. The RAID group IDs C311 are the same as the RAID group IDs C303 which are managed by the device management table T30. And the drive types C312 are information which specifies the types of the storage devices 151 which belong to the RAID groups 150.

[0157] Fig. 22 is an explanatory figure showing the tier management table T32. This tier management table T32 is a table for managing each storage layer which the storage control device 10 possesses. The tier management table T32, for example, may perform its management by maintaining, in mutual correspondence, tier types C321 and cache control modes C322.

[0158] The tier types C321 define the tier types as combinations of drive types C3211 and RAID levels C3212. The drive types may be, for example, SATA, FC, SSD, or the like. The RAID levels may be, for example, RAID 1, RAID 5, RAID 6, or the like. Moreover, in the case of RAID 1, there is a difference according to whether the drive structure is 2D+2D or 4D+4D. It should be understood that, in the second column of this table T32, "D" means "data drive", and "P" means "parity drive".

[0159] The cache control mode to be applied to each tier is set in the cache control mode C322. The cache control mode either may be a mode in which, as explained above in connection with the various embodiments of the present invention, segments whose data load ratio is high are retained within the cache memory for as long as possible, or may be a mode in which normal LRU is performed.

[0160] It should be understood that the control method of the present invention (i.e. the control mode) is not to be considered as being limited by the structure of the first embodiment, and it could be: a method of determining the order of replacement according to the type of the storage device 151 (as in the second embodiment); a method of dividing the segments between two queues 101 and 102 according to the data load ratio

(as in the third embodiment); a method of dividing the segments between two queues 101 and 102 according to the staging mode (as in the fourth embodiment); a method of, upon a segment hit, shifting the segment to an intermediate position in the queue 101 (as in the fifth embodiment); a method of changing over between performing normal LRU control during sequential access, and applying a control method according to the present invention during random access (as in the sixth embodiment); a method of using three queues as will be described hereinafter (as in the eighth embodiment); or a method of setting a plurality of regions within a single queue as will be described hereinafter (in the ninth embodiment). According to requirements, the user may also set different ones of the control methods described above, individually for each tier.

[0161] Furthermore, while in Fig. 22 a case was shown in which the cache control mode was set for each tier unit individually, this should not be considered as being limitative of the present invention; it would also be acceptable to arrange to set the cache control mode individually for each of the logical volumes 152.

[0162] Moreover, as has been described above in connection with the sixth embodiment of the present invention, it would also be acceptable to arrange to decide automatically, according to the pattern of access by the host 30, whether to employ the normal LRU control mode or to employ the control mode according to the present invention, and to register this decision result in advance in the tier management table T32. The user would also be able manually to change the control mode which has been registered in advance.

[0163] Fig. 23 is an explanatory figure showing a setting screen G10 for the cache control mode, which is displayed upon the screen of the management terminal 20. Via this setting screen G10, the user is able manually to set the cache control mode for each tier type.

[0164] The setting screen G10 may include, for example, a list display unit G11 which shows the cache control mode for each tier type, a designation unit G12 for selecting either normal LRU control ("Normal Cache Mode") or the control method according to the present invention ("Rapid Cache Mode"), and an "apply" button B11.

[0165] With this seventh embodiment having the structure described above, a similar advantageous effect is obtained as in the case of the first embodiment. In addition thereto, with this seventh embodiment the convenience of use is enhanced, since it is possible for the user to set the cache control mode in advance.

Embodiment 8

[0166] A eighth embodiment will now be explained on the basis of Fig. 24. In this eighth embodiment, a high load ratio queue 101, a low load ratio queue 102, a free queue 103, and a medium load ratio queue 104 are used. In other words, the three type of queue 101, 102, and 104 are used for managing the segments which are in use.

[0167] The medium load ratio queue 104 is a queue for managing those segments which have data load ratios which are intermediate between the data load ratios which correspond to the high load ratio queue 101 and the data load ratios which correspond to the low load ratio queue 102. In this manner, with this eighth embodiment having the structure described above, a similar advantageous effect is obtained as in the case of the first embodiment.

Embodiment 9

[0168] A ninth embodiment will now be explained on the basis of Fig. 25. In this ninth embodiment, one queue 105 is used by being divided into two regions 105A and 105B. One of these regions 105A is a region for managing segments whose data load ratio is high. The other of these regions 105B is a region for managing segments whose data load ratio is low. It should be understood that the unused segments are managed with the free queue 103. In this manner, with this ninth embodiment having the structure described above, a similar advantageous effect is obtained as in the case of the first embodiment.

[0169] It should be understood that the present invention is not limited to the embodiments described above. A person skilled in the art will be able to make various additions and alterations within the scope of the present invention. For example, it would be possible to combine the above embodiments in various appropriate ways.

Claims

[1] A storage control device (1, 10) which performs data input and output between a host computer (2, 30) and a storage device (5, 151) via a cache memory (4,132), comprising:
a cache control unit (120) for controlling data input and output to and from a plurality of segments (4A, 1321) which said cache memory possesses; and
a plurality of queues (101 - 103) which are used by said cache control unit for managing said segments;
wherein, in predetermined circumstances, by controlling said queues according to the amounts of data stored in said segments, said cache control unit permutes segments in which said data amounts are relatively small, so as to give priority to segments whose data amounts are relatively large.

[2] A storage control device according to Claim 1, wherein:
said plurality of queues includes a first queue (6, 101) for managing segments in which the amount of data is relatively large, a second queue (7, 102) for managing segments in which the amount of data is relatively small, and a third queue (8, 103) for managing unused segments;
said predetermined circumstances are that the size of accesses by said host computer is smaller than the size of said segments, and moreover that said host computer performs random accesses; and
said cache control unit:
if data requested by said host computer is not stored in any of said segments, stores said data requested by said host computer in an unused segment which is being managed with said third queue, and manages said segment with said second queue;
if said data requested by said host computer is stored in one of said segments, manages the segment in which said data is stored with said first queue; and
if the number of said unused segments which are being managed with said third queue has become less than or equal to a predetermined value, manages with said third queue as an unused segment, that segment, among said segments being managed with said second queue, for which the elapsed time from its last use is the longest.

[3] A storage control device according to Claim 1, wherein said predetermined circumstances are that the size of accesses by said host computer is smaller than the size of said segments, and moreover that said host computer performs random accesses.

[4] A storage control device according to Claim 1, wherein said plurality of queues

includes a first queue for managing segments in which the amount of data is relatively large, a second queue for managing segments in which the amount of data is relatively small, and a third queue for managing unused segments.

[5] A storage control device according to Claim 4, wherein one single queue 105 is divided into two regions (105A) and (105B), and said first region is used as said first queue, while said second region is used as said second queue.

[6] A storage control device according to Claim 1, wherein said cache control unit:
if data requested by said host computer is not stored in any of said segments, stores said data requested by said host computer in an unused segment which is being managed with said third queue, and manages said segment with said second queue as the segment which has been most recently used; and
if said data requested by said host computer is stored in one of said segments, manages the segment in which said data is stored with said first queue as the segment which has been used the longest ago.

[7] A storage control device according to Claim 6, wherein, if the number of said unused segments which are being managed with said third queue has become less than or equal to a predetermined value, said cache control unit manages with said third queue, as an unused segment, that segment, among said segments being managed with said second queue, for which the elapsed time from its last use is the longest.

[8] A storage control device according to Claim 6, wherein, if the number of said unused segments which are being managed with said third queue has become less than or equal to a predetermined value, said cache control unit:
compares together a first elapsed time of that segment, among the segments stored in said first queue, which has been used the longest ago, and a second elapsed time of that segment, among the segments stored in said second queue, which has been used the longest ago; and:
if said first elapsed time is greater than or equal to said second elapsed time, manages with said third queue, as an unused segment, said segment, among said segments being managed with said first queue, which has been used the longest ago; and
if said first elapsed time is less than said second elapsed time, manages with said third queue, as an unused segment, said segment, among said segments being managed with said second queue, which has been used the longest ago.

[9] A storage control device according to Claim 6, wherein, if the number of said unused segments which are being managed with said third queue has become less than or equal to a predetermined value, said cache control unit calculates the ratio between the number of segments which are being managed with said first

queue, and the number of segments which are being managed with said second queue; and manages with said third queue, as an unused segment, either that segment, among said segments being managed with said first queue, which has been used the longest ago, or that segment, among said segments being managed with said second queue, which has been used the longest ago, so that said ratio approaches a target ratio which is set in advance.

[10] A storage control device according to Claim 6, wherein said storage device includes a high speed storage device whose access speed is relatively high and a low speed storage device whose access speed is relatively low; and, if the number of said unused segments being managed with said third queue is less than or equal to a predetermined value, said cache control unit manages with said third queue, as an unused segment, a segment, among said segments being managed with said second queue, which corresponds to said high speed storage device.

[11] A storage control device according to Claim 1, wherein said cache control unit:
if the amount of data which is stored in segments being managed with said first queue has dropped below a predetermined value which is set in advance, shifts said segment in which the amount of data has dropped below said predetermined value to said second queue; and
if the amount of data which is stored in segments being managed with said second queue is greater than said predetermined value which is set in advance, shifts said segment in which the amount of data is greater than or equal to said predetermined value to said first queue.

[12] A storage control device according to Claim 1, wherein:
each of said segments consists of a plurality of blocks;
each of said blocks consists of a plurality of sub-blocks;
when predetermined data which has been requested from said host computer is not stored in any one of said segments, said cache control unit reads out said predetermined data from said storage device according to either a first mode, a second mode, or a third mode;
said first mode is a mode in which only data in a sub-block corresponding to said predetermined data is read out from said storage device and stored in said cache memory;
said second mode is a mode in which data from a sub-block corresponding to said predetermined data to the final sub-block in the segment which contains said sub-block is read out from said storage device and stored in said cache memory; and
said third mode is a mode in which the data in all of the sub-blocks in the

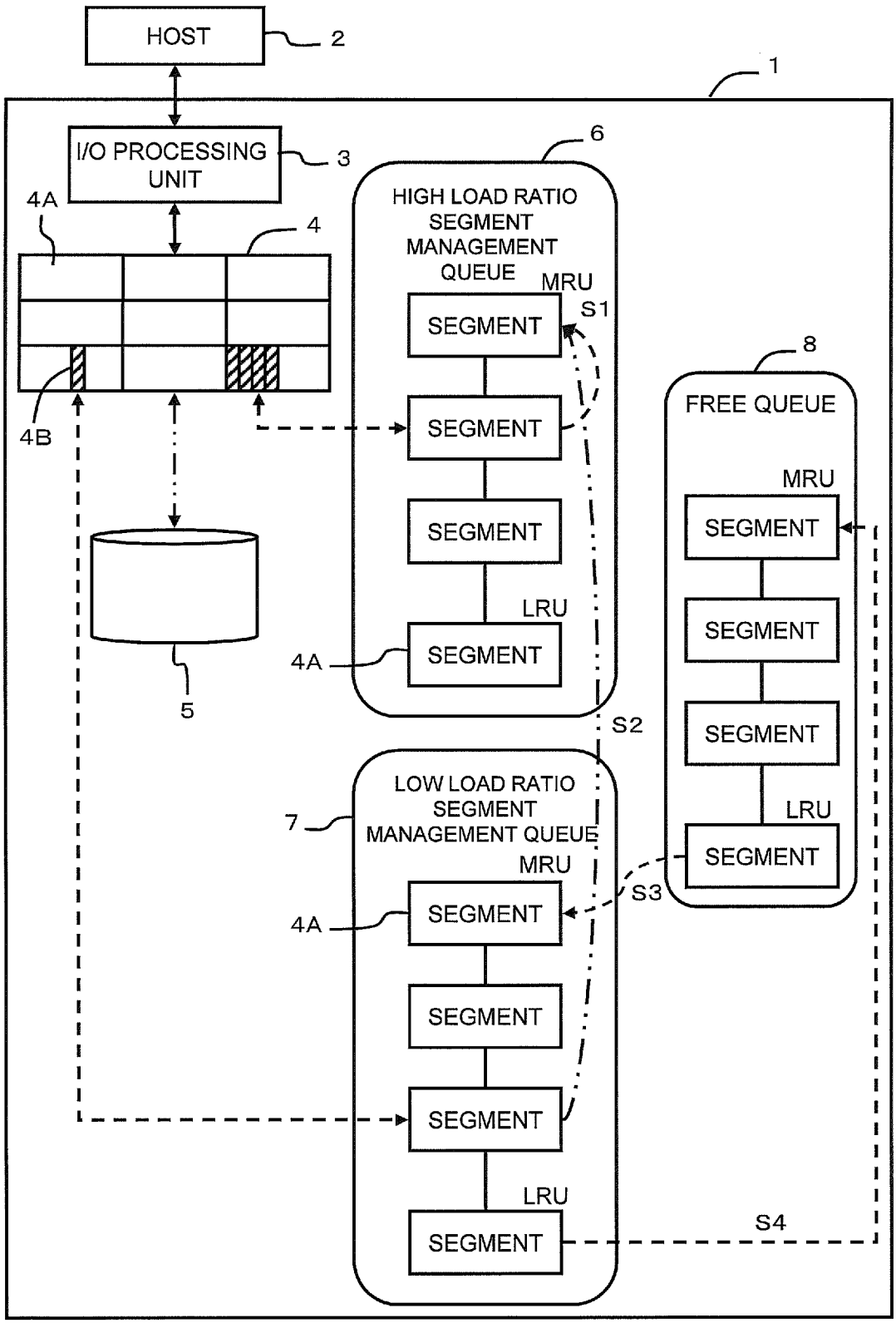
segment which contains a sub-block corresponding to said predetermined data is read out from said storage device and stored in said cache memory; and
said cache control unit:
if said predetermined data has been read out in said first mode, manages the segment in which said predetermined data is stored with said second queue; and
if said predetermined data has been read out in said second mode or said third mode, manages the segment in which said predetermined data is stored with said first queue.

[13] A storage control device according to Claim 1, wherein, in other than said predetermined circumstances, said cache control unit releases and reuses segments, among said segments, in order from that segment for which the elapsed time since its last use is the longest.

[14] A control method for a cache memory, for performing data input and output between a host computer (2, 30) and a storage device (5, 151) via a cache memory (4,132), wherein:
the size of accesses by said host computer is smaller than the size of segments (4A, 1321) which make up said cache memory;
a decision is made as to whether or not data to which access has been requested from said computer is stored in any of said segments;
if said data to which access has been requested from said computer is stored in one of said segments, said segment in which said data is stored is shifted to a first queue (6, 101) for managing segments in which the proportion of data which has been staged is relatively large;
if said data to which access has been requested from said computer is not stored in one of said segments, a decision is made as to whether or not any unused segment exists;
if such an unused segment exists, said data to which access has been requested from said computer is read out from said storage device and stored therein, and said segment in which said data which has been read out from said storage device has been stored is put at the newest end of a second queue (7, 102) for managing segments in which the proportion of data which has been staged is relatively small; and
if such an unused segment does not exist, then either that segment, among the segments which are being managed with said first queue, for which the elapsed time from last access is the longest, or that segment, among the segments which are being managed with said second queue, for which the elapsed time from last access is the longest, is used as said unused segment.

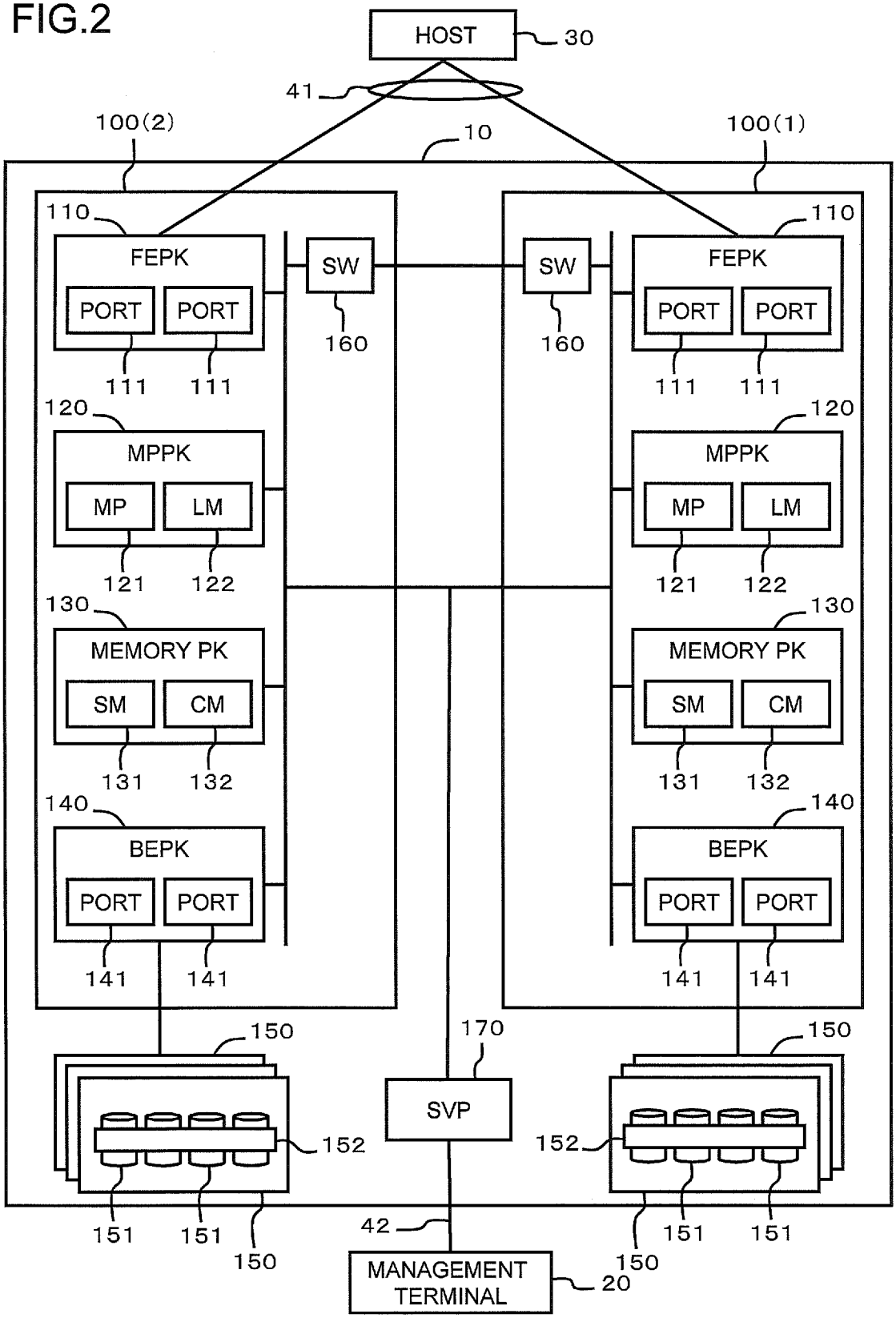
[Fig. 1]

FIG.1


HOST
2
1
I/O PROCESSING UNIT
3
6
HIGH LOAD RATIO SEGMENT MANAGEMENT QUEUE
MRU
SEGMENT
S1
SEGMENT
SEGMENT
LRU
SEGMENT
4A
4A
4
4B
5
8
FREE QUEUE
MRU
SEGMENT
SEGMENT
SEGMENT
LRU
SEGMENT
S2
S3
7
LOW LOAD RATIO SEGMENT MANAGEMENT QUEUE
MRU
4A
SEGMENT
SEGMENT
SEGMENT
LRU
SEGMENT
S4

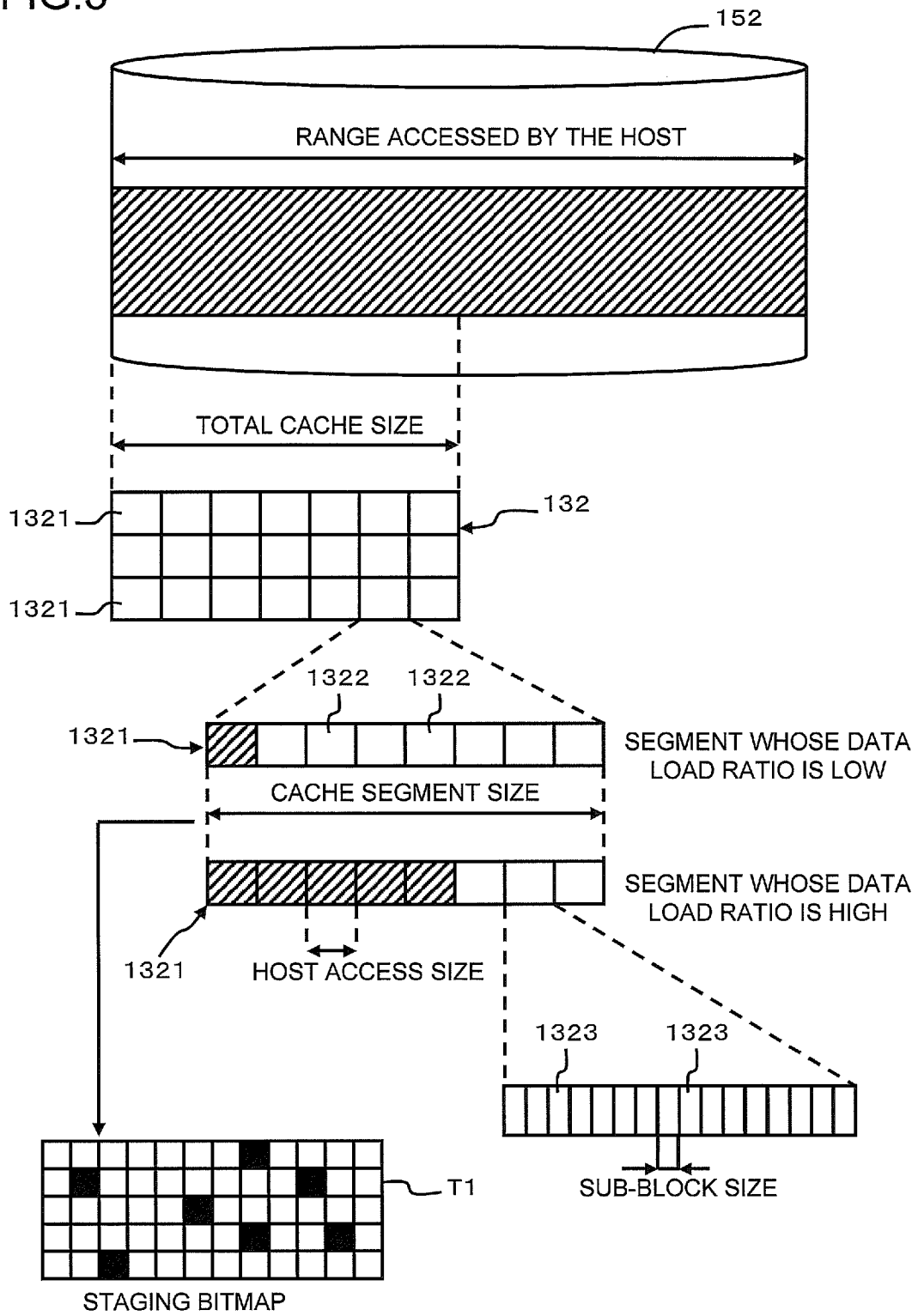
[Fig. 2]

FIG.2


HOST
30
41
100(2)
10
100(1)
110
FEPK
PORT
PORT
111
111
SW
160
SW
160
110
FEPK
PORT
PORT
111
111
120
MPPK
MP
LM
121
122
120
MPPK
MP
LM
121
122
130
MEMORY PK
SM
CM
131
132
130
MEMORY PK
SM
CM
131
132
140
BEPK
PORT
PORT
141
141
140
BEPK
PORT
PORT
141
141
150
152
151
151
150
170
SVP
42
MANAGEMENT TERMINAL
20
152
150
151
151
150

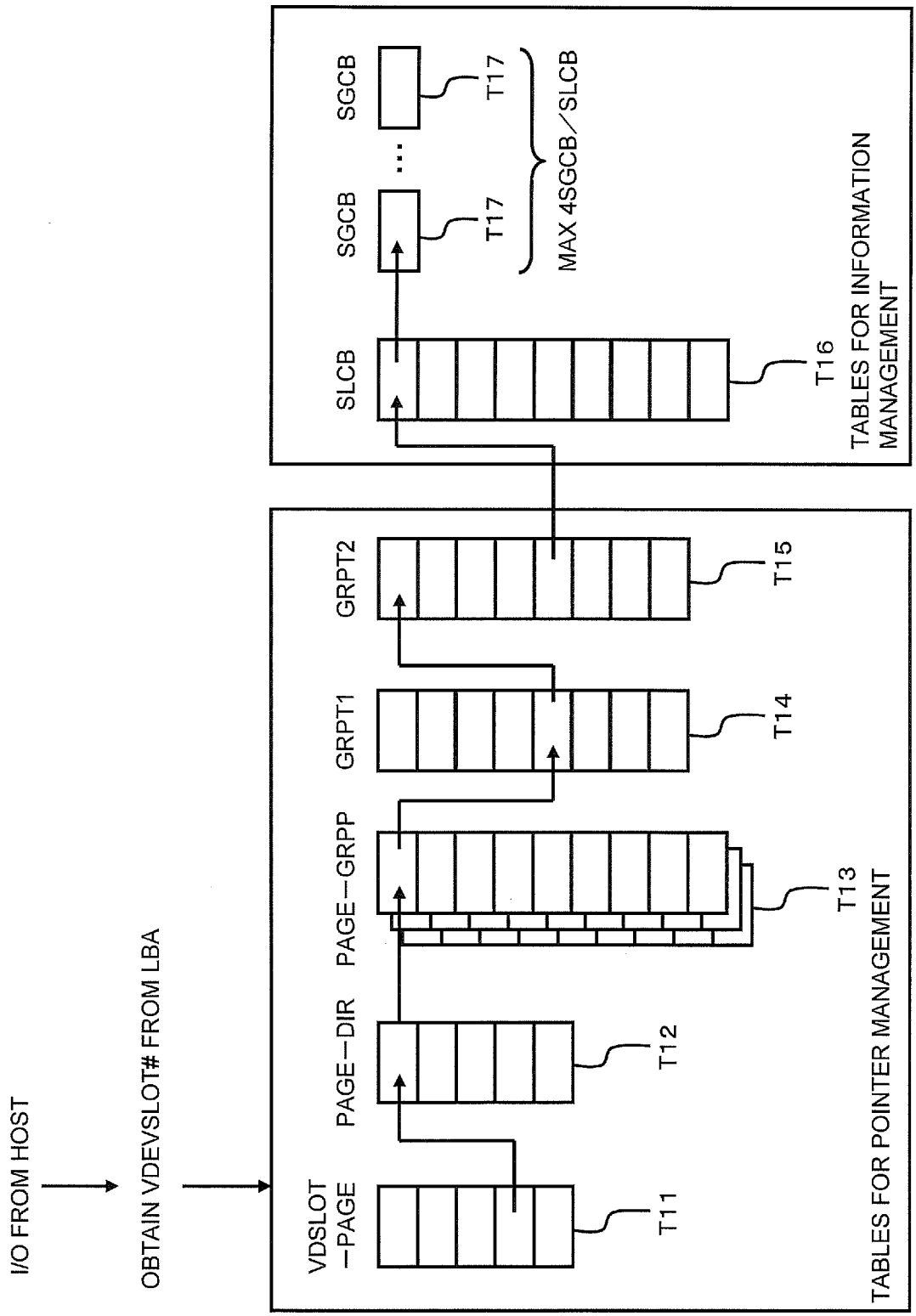
[Fig. 3]

FIG.3


152
RANGE ACCESSED BY THE HOST
TOTAL CACHE SIZE
1321
1321
132
1322
1322
1321
SEGMENT WHOSE DATA LOAD RATIO IS LOW
CACHE SEGMENT SIZE
SEGMENT WHOSE DATA LOAD RATIO IS HIGH
1321
HOST ACCESS SIZE
1323
1323
T1
SUB-BLOCK SIZE
STAGING BITMAP

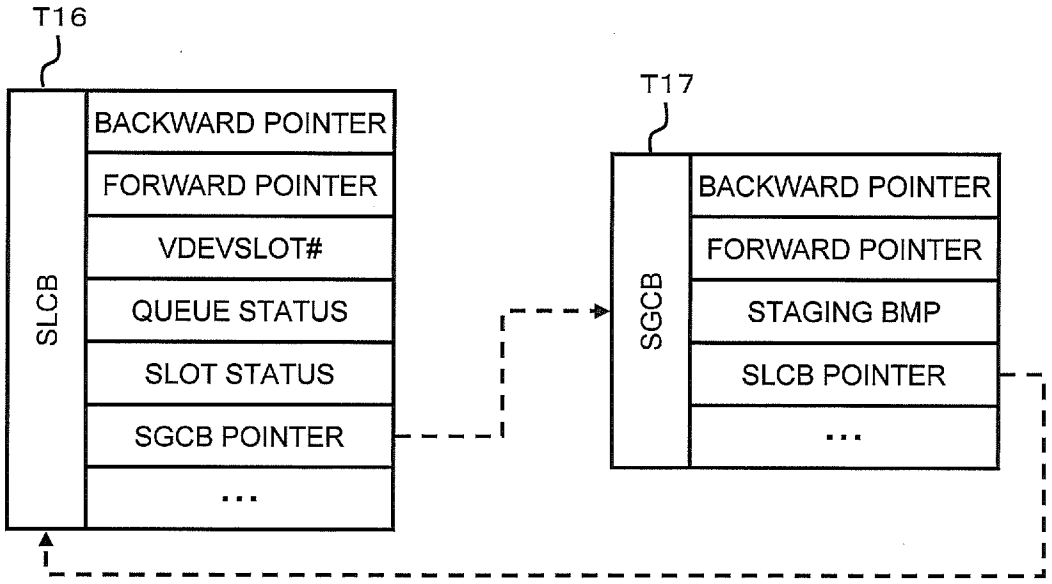
[Fig. 4]

FIG.4


I/O FROM HOST
OBTAIN VDEVSLOT# FROM LBA
VDSLOT
−PAGE
PAGE−DIR
PAGE−GRPP
GRPT1
GRPT2
SLCB
SGCB
SGCB
SGCB
T11
T12
T13
T14
T15
T16
T17
T17
MAX 4SGCB／SLCB
TABLES FOR POINTER MANAGEMENT
TABLES FOR INFORMATION MANAGEMENT

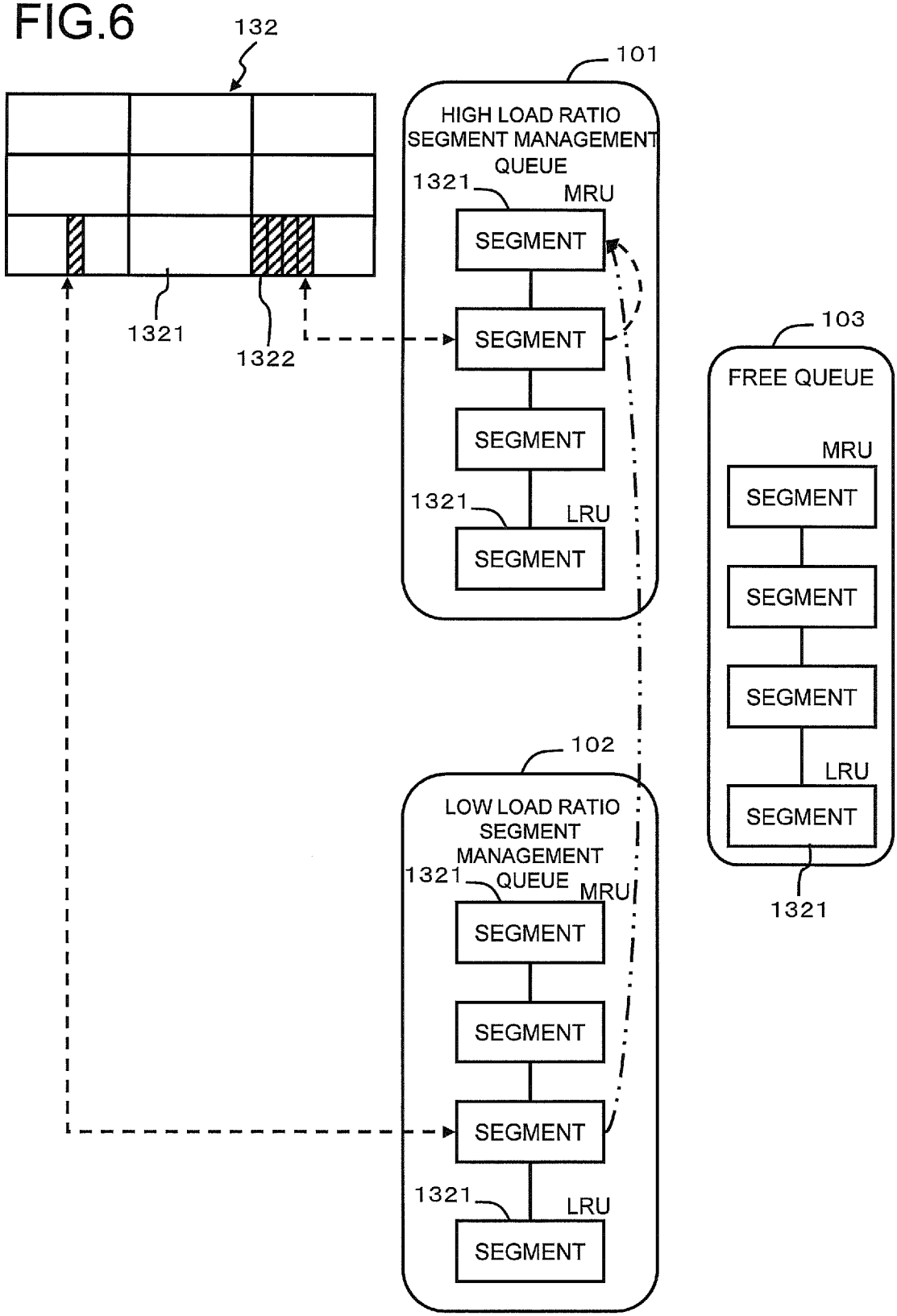
[Fig. 5]

FIG.5


T16
SLCB
BACKWARD POINTER
FORWARD POINTER
VDEVSLOT#
QUEUE STATUS
SLOT STATUS
SGCB POINTER
...
T17
SGCB
BACKWARD POINTER
FORWARD POINTER
STAGING BMP
SLCB POINTER
...

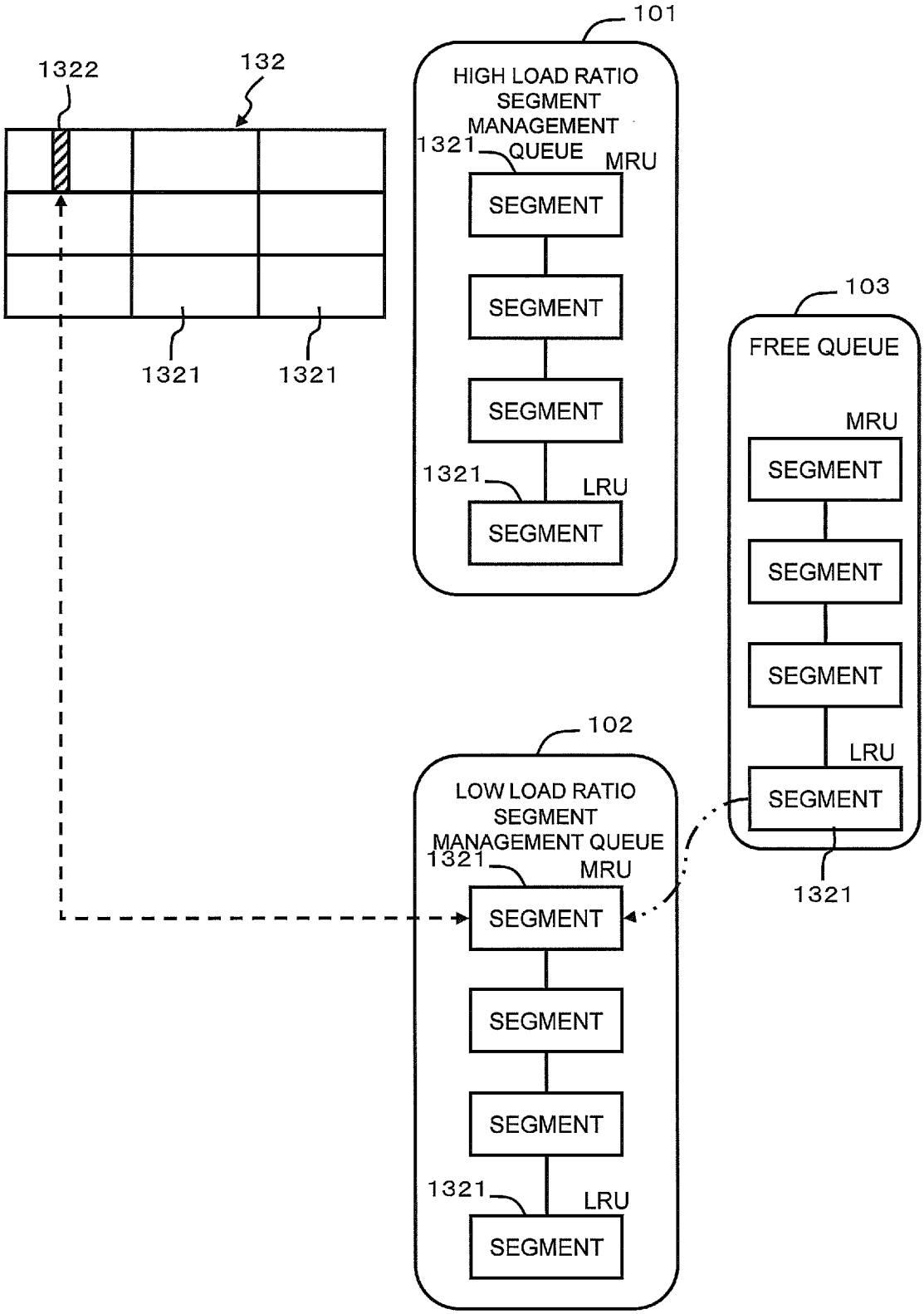
[Fig. 6]

FIG.6


132
101
HIGH LOAD RATIO SEGMENT MANAGEMENT QUEUE
1321
MRU
SEGMENT
SEGMENT
SEGMENT
1321
LRU
SEGMENT
1321
1322
103
FREE QUEUE
MRU
SEGMENT
SEGMENT
SEGMENT
LRU
SEGMENT
1321
102
LOW LOAD RATIO SEGMENT MANAGEMENT QUEUE
1321
MRU
SEGMENT
SEGMENT
SEGMENT
1321
LRU
SEGMENT

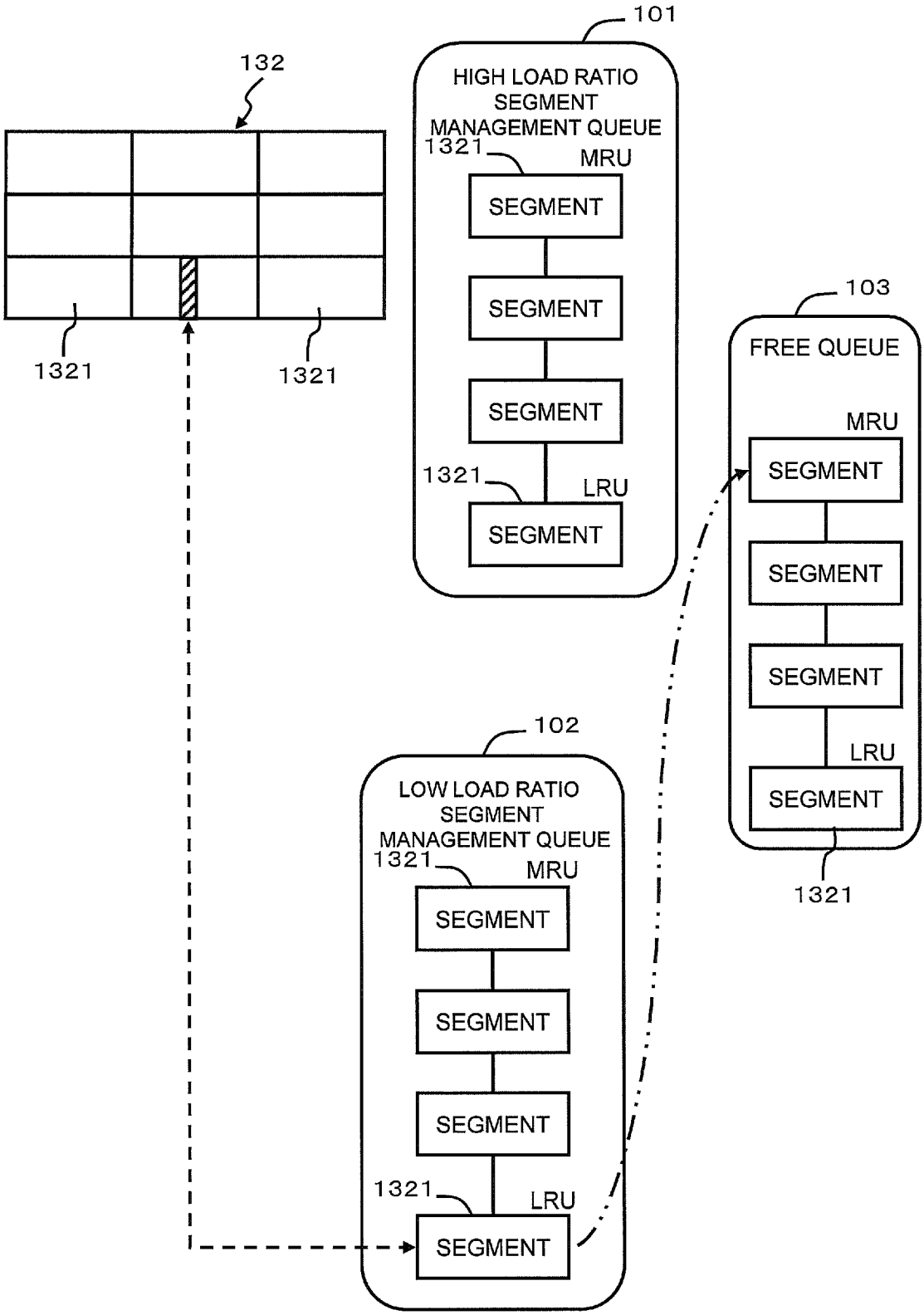
[Fig. 7]

FIG.7


101
HIGH LOAD RATIO SEGMENT MANAGEMENT QUEUE
1321
MRU
SEGMENT
SEGMENT
SEGMENT
1321
LRU
SEGMENT
1322
132
1321
1321
103
FREE QUEUE
MRU
SEGMENT
SEGMENT
SEGMENT
LRU
SEGMENT
1321
102
LOW LOAD RATIO SEGMENT MANAGEMENT QUEUE
1321
MRU
SEGMENT
SEGMENT
SEGMENT
1321
LRU
SEGMENT

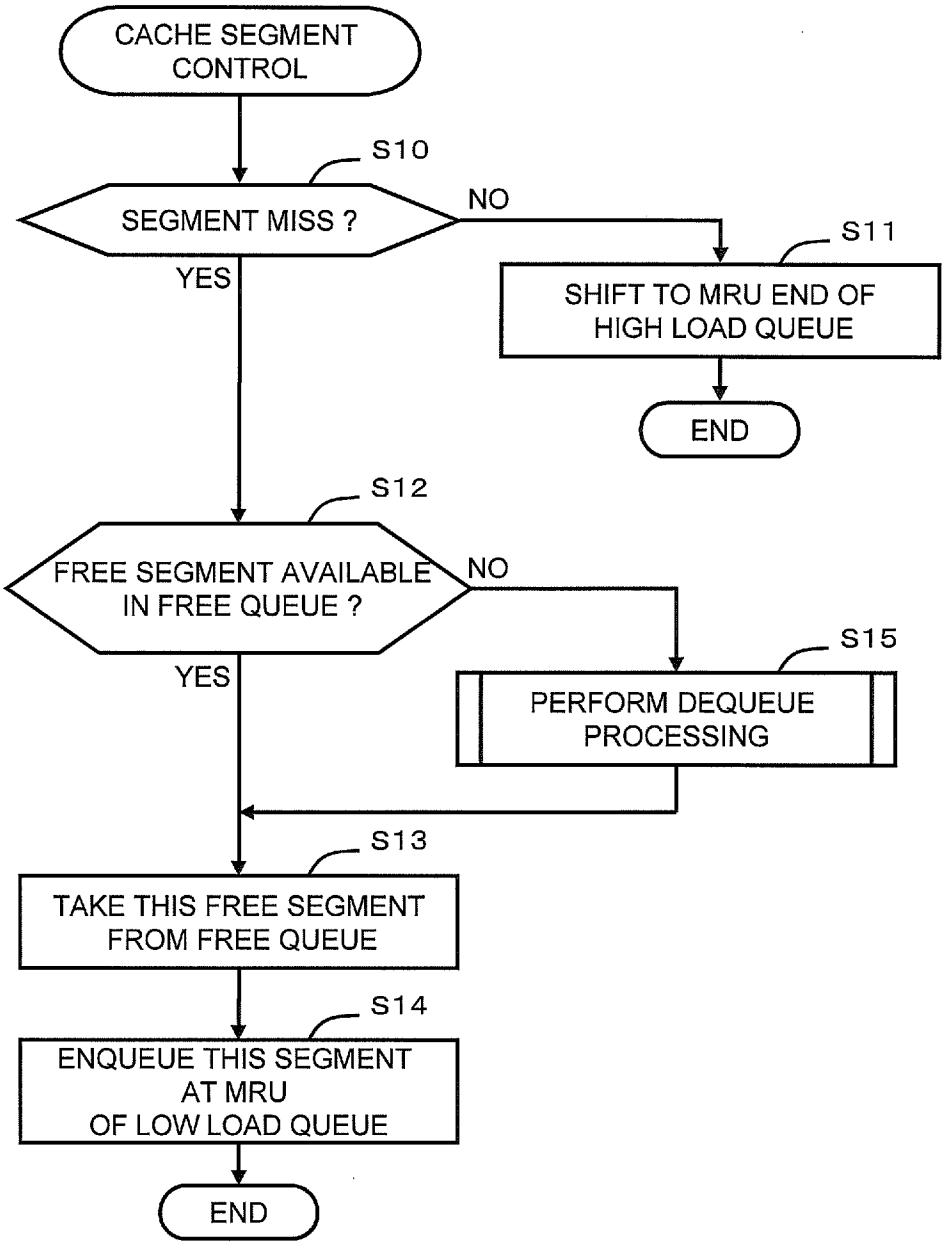
[Fig. 8]

FIG.8


101
HIGH LOAD RATIO SEGMENT MANAGEMENT QUEUE
1321
MRU
SEGMENT
SEGMENT
SEGMENT
1321
LRU
SEGMENT
132
1321
1321
103
FREE QUEUE
MRU
SEGMENT
SEGMENT
SEGMENT
LRU
SEGMENT
1321
102
LOW LOAD RATIO SEGMENT MANAGEMENT QUEUE
1321
MRU
SEGMENT
SEGMENT
SEGMENT
1321
LRU
SEGMENT

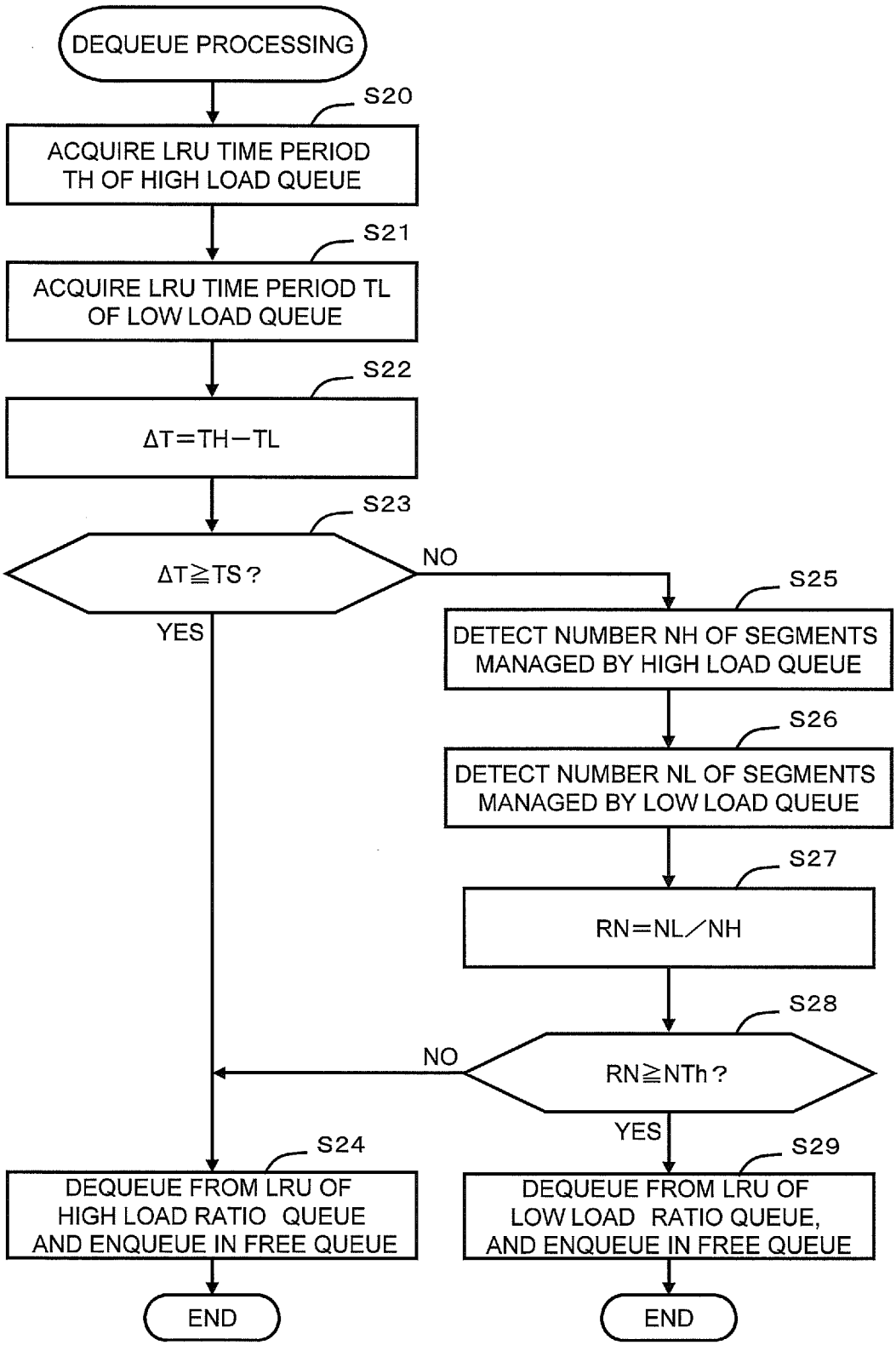
[Fig. 9]

FIG.9


CACHE SEGMENT CONTROL
S10
SEGMENT MISS ?
NO
S11
SHIFT TO MRU END OF HIGH LOAD QUEUE
END
YES
S12
FREE SEGMENT AVAILABLE IN FREE QUEUE ?
NO
S15
PERFORM DEQUEUE PROCESSING
YES
S13
TAKE THIS FREE SEGMENT FROM FREE QUEUE
S14
ENQUEUE THIS SEGMENT AT MRU OF LOW LOAD QUEUE
END

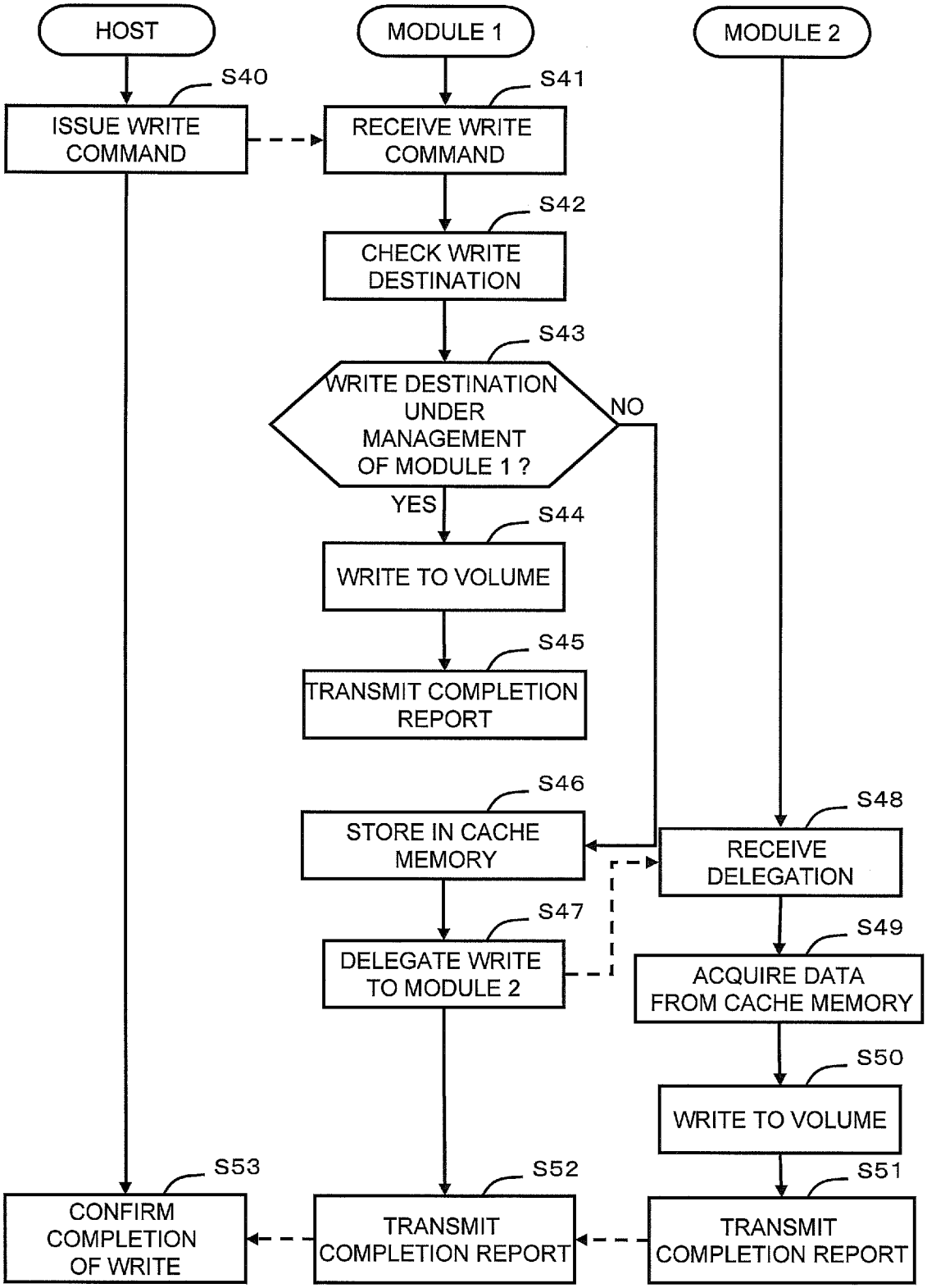
[Fig. 10]

FIG.10


DEQUEUE PROCESSING
S20
ACQUIRE LRU TIME PERIOD TH OF HIGH LOAD QUEUE
S21
ACQUIRE LRU TIME PERIOD TL OF LOW LOAD QUEUE
S22
ΔT=TH−TL
S23
ΔT≧TS？
NO
YES
S25
DETECT NUMBER NH OF SEGMENTS MANAGED BY HIGH LOAD QUEUE
S26
DETECT NUMBER NL OF SEGMENTS MANAGED BY LOW LOAD QUEUE
S27
RN=NL／NH
S28
RN≧NTh？
NO
YES
S24
DEQUEUE FROM LRU OF HIGH LOAD RATIO QUEUE AND ENQUEUE IN FREE QUEUE
S29
DEQUEUE FROM LRU OF LOW LOAD RATIO QUEUE, AND ENQUEUE IN FREE QUEUE
END
END

[Fig. 11]

FIG.11


HOST
MODULE 1
MODULE 2
S40
ISSUE WRITE COMMAND
S41
RECEIVE WRITE COMMAND
S42
CHECK WRITE DESTINATION
S43
WRITE DESTINATION UNDER MANAGEMENT OF MODULE 1 ?
NO
YES
S44
WRITE TO VOLUME
S45
TRANSMIT COMPLETION REPORT
S46
STORE IN CACHE MEMORY
S47
DELEGATE WRITE TO MODULE 2
S48
RECEIVE DELEGATION
S49
ACQUIRE DATA FROM CACHE MEMORY
S50
WRITE TO VOLUME
S51
TRANSMIT COMPLETION REPORT
S52
TRANSMIT COMPLETION REPORT
S53
CONFIRM COMPLETION OF WRITE

[Fig. 12]

FIG.12

SIMULATION RESULTS				
CACHE SIZE / HOST ACCESS RANGE (%)	HIT RATE FOR EACH CACHE SEGMENT SIZE (THE PRESENT INVENTION / THE PRIOR ART)			
	64KB	32KB	16KB	8KB
100	100	100	100	100
75	68/27	73/42	74/60	74/75
50	43/11	48/20	49/33	49/50
25	19/4	23/7	24/14	25/25

[Fig. 13]

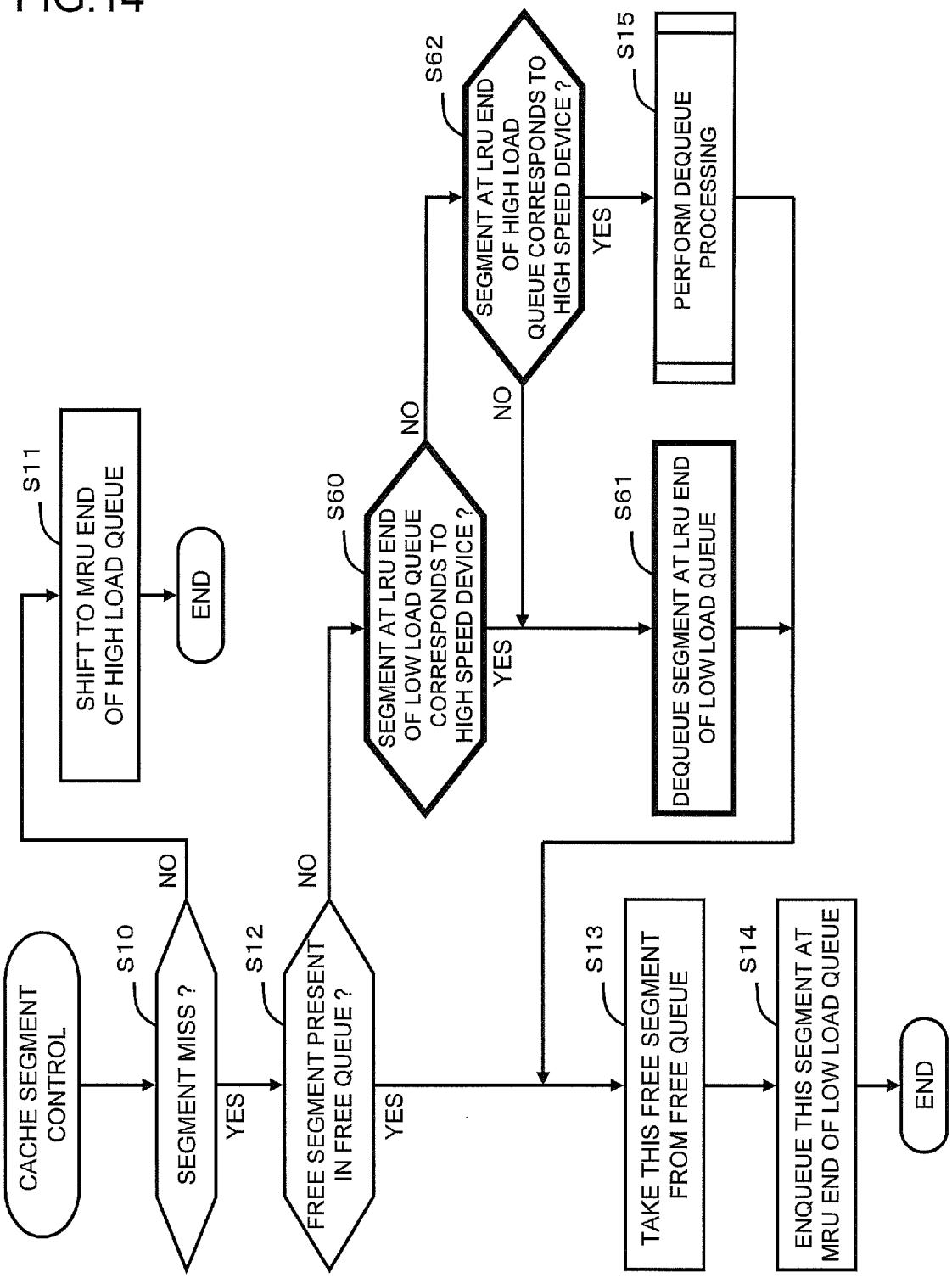
FIG.13

T20

PRIORITY ORDER FOR REPLACEMENT		
QUEUE TYPE / DEVICE TYPE	HIGH LOAD QUEUE	LOW LOAD QUEUE
LOW SPEED STORAGE DEVICE	4	2 or 3
HIGH SPEED STORAGE DEVICE	3 or 2	1

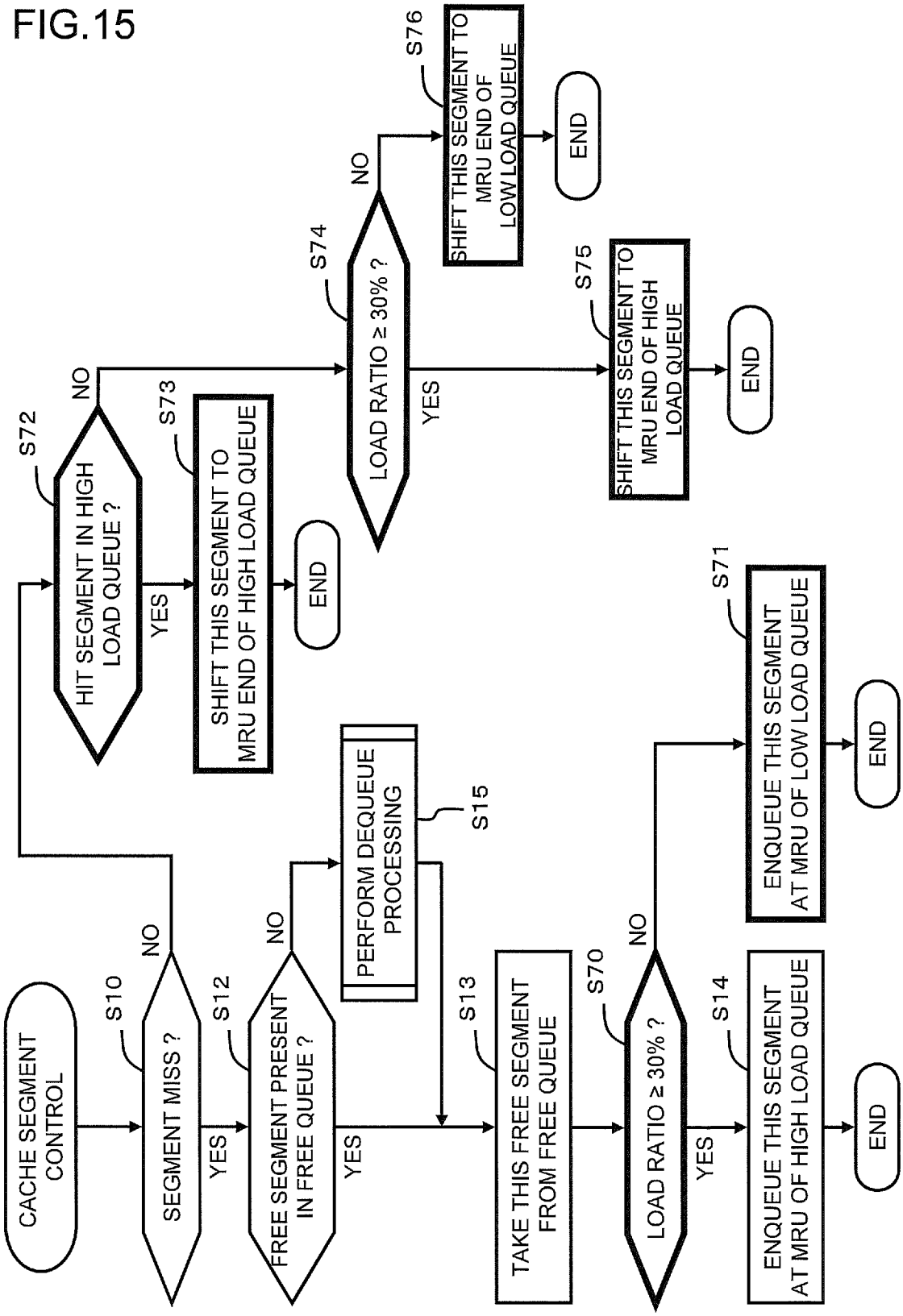
[Fig. 14]

FIG.14


CACHE SEGMENT CONTROL
S10
SEGMENT MISS ?
NO
YES
S11
SHIFT TO MRU END OF HIGH LOAD QUEUE
END
S12
FREE SEGMENT PRESENT IN FREE QUEUE ?
NO
YES
S60
SEGMENT AT LRU END OF LOW LOAD QUEUE CORRESPONDS TO HIGH SPEED DEVICE ?
NO
YES
S61
DEQUEUE SEGMENT AT LRU END OF LOW LOAD QUEUE
S62
SEGMENT AT LRU END OF HIGH LOAD QUEUE CORRESPONDS TO HIGH SPEED DEVICE ?
NO
YES
S15
PERFORM DEQUEUE PROCESSING
S13
TAKE THIS FREE SEGMENT FROM FREE QUEUE
S14
ENQUEUE THIS SEGMENT AT MRU END OF LOW LOAD QUEUE
END

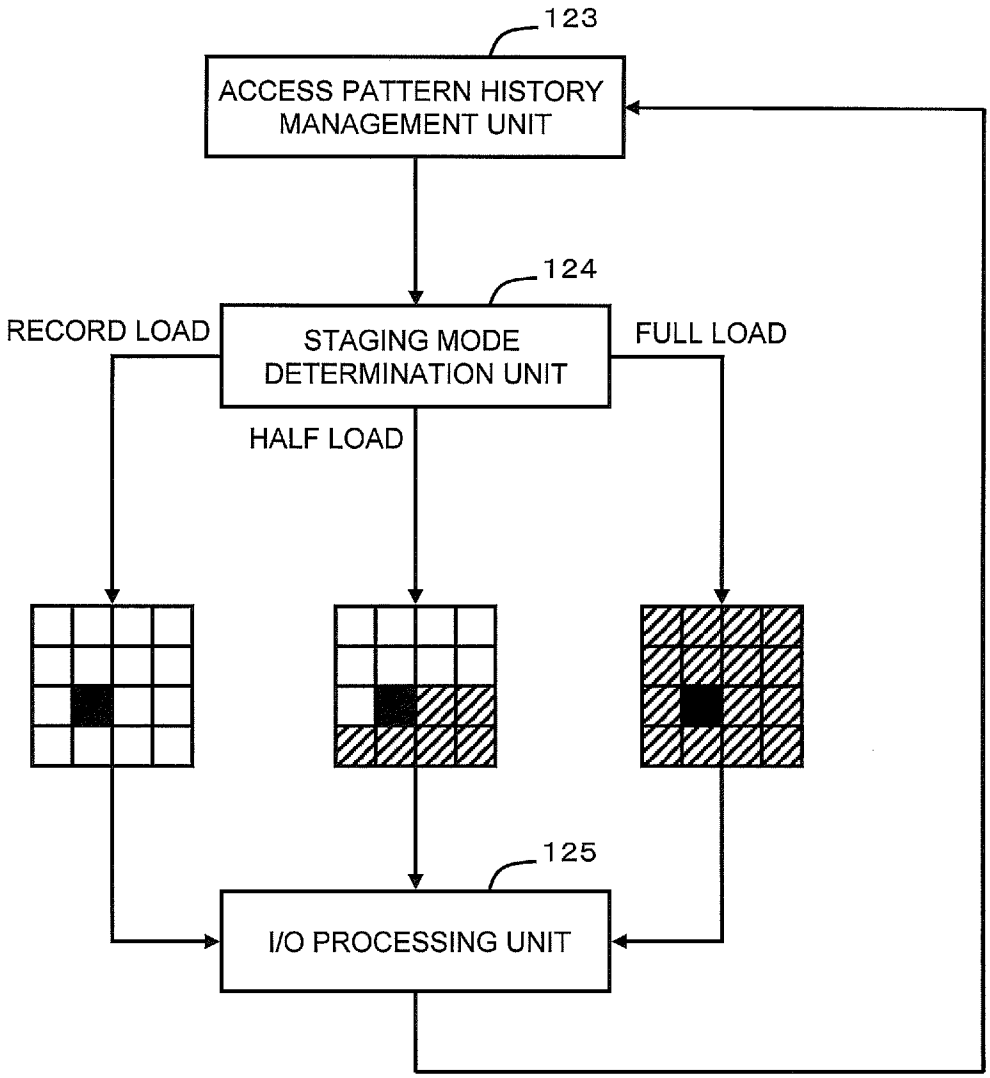
[Fig. 15]

FIG.15


CACHE SEGMENT CONTROL
S10
SEGMENT MISS ?
NO
YES
S12
FREE SEGMENT PRESENT IN FREE QUEUE ?
NO
YES
PERFORM DEQUEUE PROCESSING
S15
S13
TAKE THIS FREE SEGMENT FROM FREE QUEUE
S70
LOAD RATIO ≥ 30% ?
NO
YES
S14
ENQUEUE THIS SEGMENT AT MRU OF HIGH LOAD QUEUE
END
S71
ENQUEUE THIS SEGMENT AT MRU OF LOW LOAD QUEUE
END
S72
HIT SEGMENT IN HIGH LOAD QUEUE ?
NO
YES
S73
SHIFT THIS SEGMENT TO MRU END OF HIGH LOAD QUEUE
END
S74
LOAD RATIO ≥ 30% ?
NO
YES
S75
SHIFT THIS SEGMENT TO MRU END OF HIGH LOAD QUEUE
END
S76
SHIFT THIS SEGMENT TO MRU END OF LOW LOAD QUEUE
END

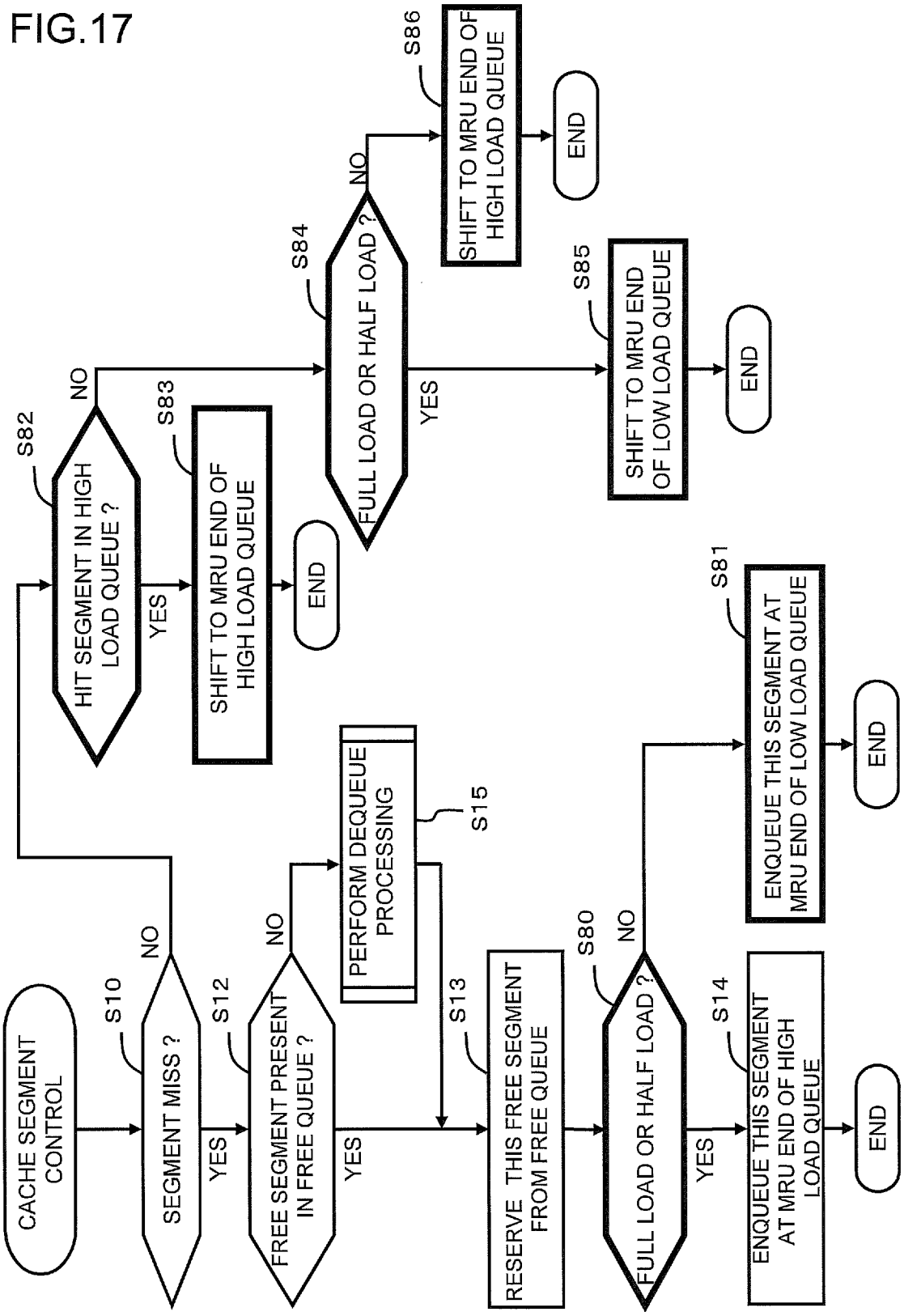
[Fig. 16]

FIG.16


123
ACCESS PATTERN HISTORY MANAGEMENT UNIT
124
RECORD LOAD
STAGING MODE DETERMINATION UNIT
FULL LOAD
HALF LOAD
125
I/O PROCESSING UNIT

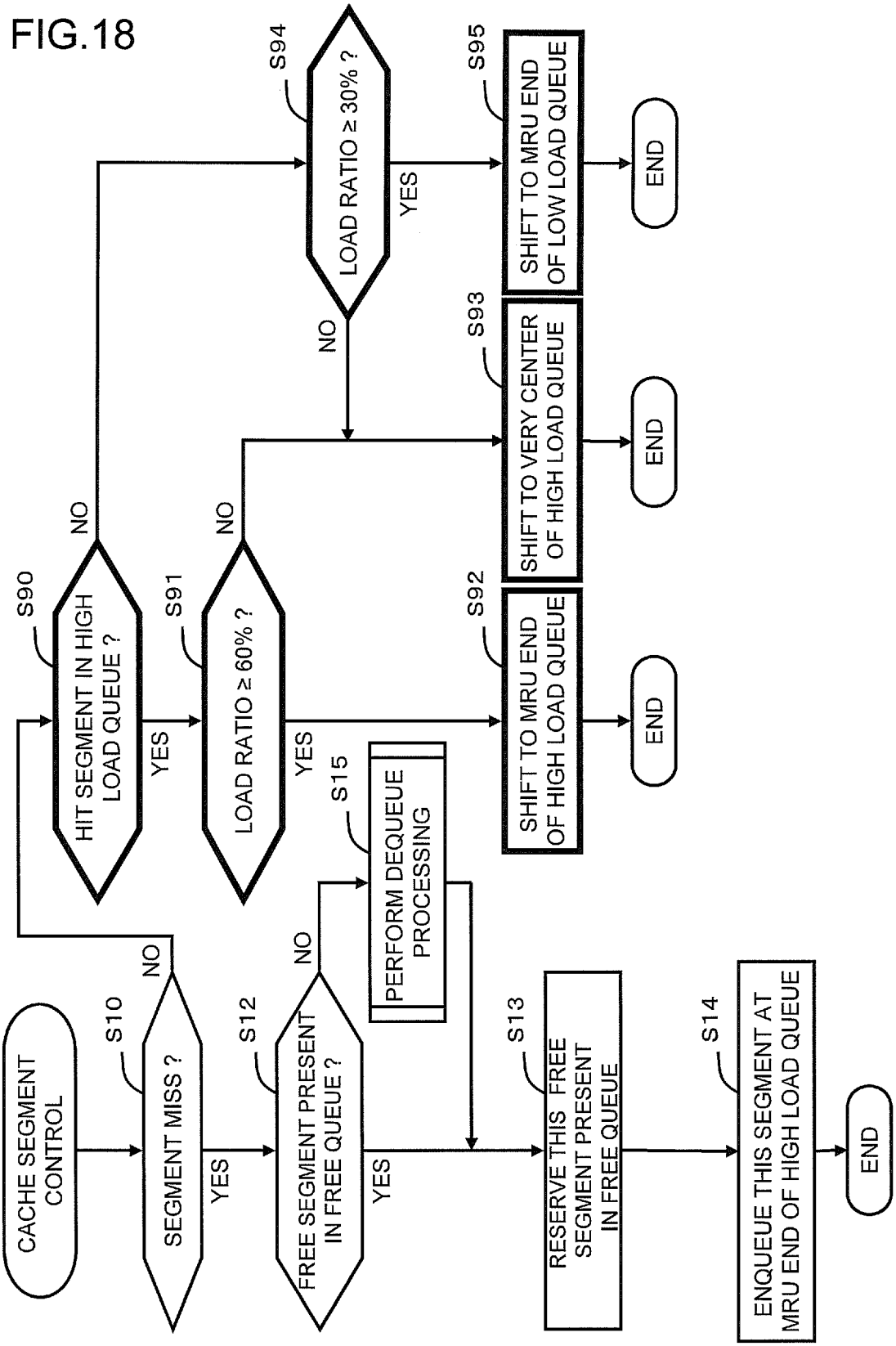
[Fig. 17]

FIG.17


CACHE SEGMENT CONTROL
S10
SEGMENT MISS ?
NO
YES
S12
FREE SEGMENT PRESENT IN FREE QUEUE ?
NO
YES
PERFORM DEQUEUE PROCESSING
S15
S13
RESERVE THIS FREE SEGMENT FROM FREE QUEUE
S80
FULL LOAD OR HALF LOAD ?
NO
YES
S81
ENQUEUE THIS SEGMENT AT MRU END OF LOW LOAD QUEUE
END
S14
ENQUEUE THIS SEGMENT AT MRU END OF HIGH LOAD QUEUE
END
S82
HIT SEGMENT IN HIGH LOAD QUEUE ?
NO
YES
S83
SHIFT TO MRU END OF HIGH LOAD QUEUE
END
S84
FULL LOAD OR HALF LOAD ?
NO
YES
S86
SHIFT TO MRU END OF HIGH LOAD QUEUE
END
S85
SHIFT TO MRU END OF LOW LOAD QUEUE
END

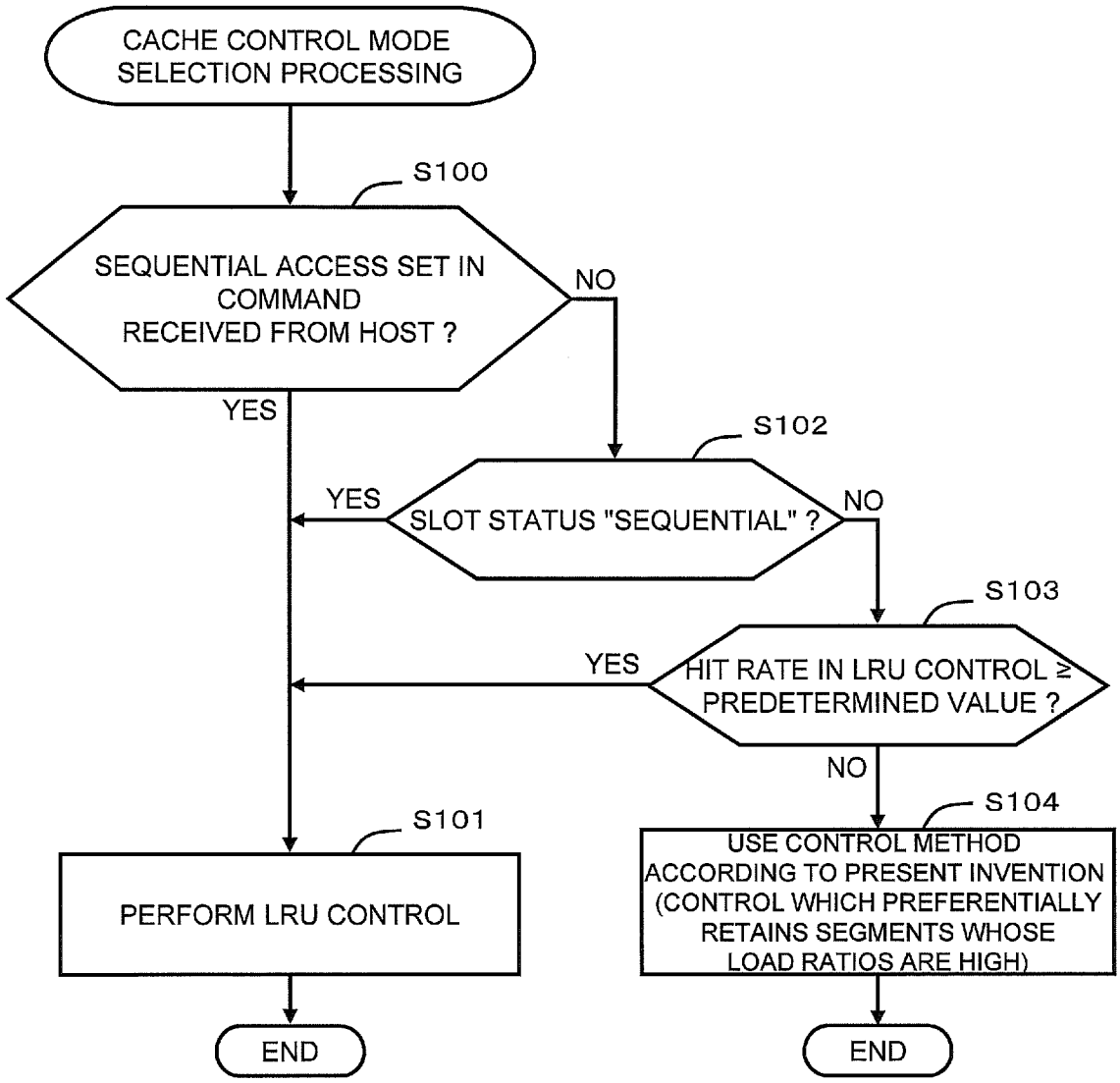
[Fig. 18]

FIG.18


CACHE SEGMENT CONTROL
S10
SEGMENT MISS ?
NO
YES
S12
FREE SEGMENT PRESENT IN FREE QUEUE ?
NO
YES
S15
PERFORM DEQUEUE PROCESSING
S13
RESERVE THIS FREE SEGMENT PRESENT IN FREE QUEUE
S14
ENQUEUE THIS SEGMENT AT MRU END OF HIGH LOAD QUEUE
END
S90
HIT SEGMENT IN HIGH LOAD QUEUE ?
NO
YES
S91
LOAD RATIO ≥ 60% ?
NO
YES
S92
SHIFT TO MRU END OF HIGH LOAD QUEUE
END
S93
SHIFT TO VERY CENTER OF HIGH LOAD QUEUE
END
S94
LOAD RATIO ≥ 30% ?
NO
YES
S95
SHIFT TO MRU END OF LOW LOAD QUEUE
END

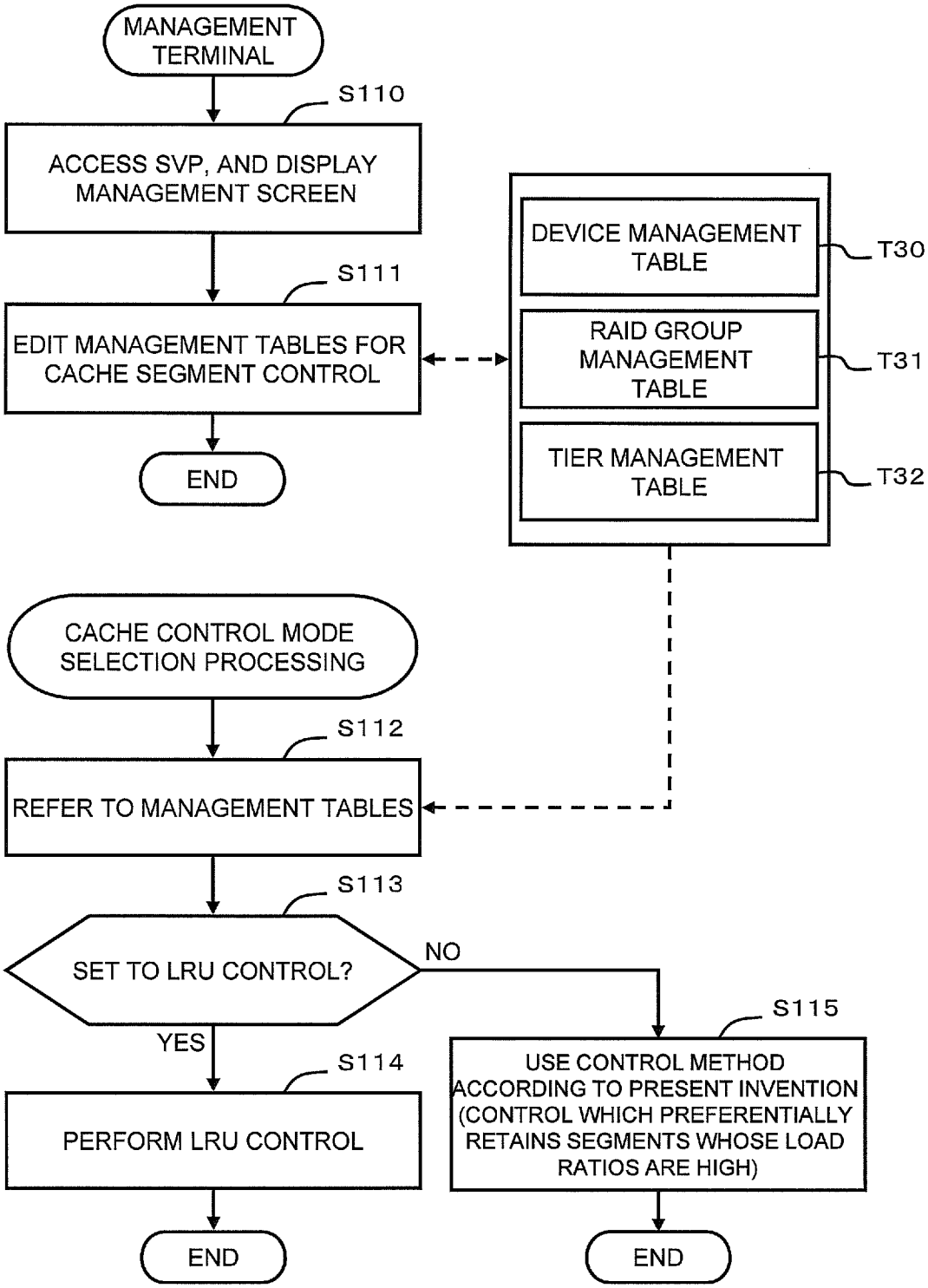
[Fig. 19]

FIG.19


CACHE CONTROL MODE SELECTION PROCESSING
S100
SEQUENTIAL ACCESS SET IN COMMAND RECEIVED FROM HOST ?
NO
YES
S102
YES
SLOT STATUS "SEQUENTIAL" ?
NO
S103
YES
HIT RATE IN LRU CONTROL ≥ PREDETERMINED VALUE ?
NO
S104
USE CONTROL METHOD ACCORDING TO PRESENT INVENTION (CONTROL WHICH PREFERENTIALLY RETAINS SEGMENTS WHOSE LOAD RATIOS ARE HIGH)
S101
PERFORM LRU CONTROL
END
END

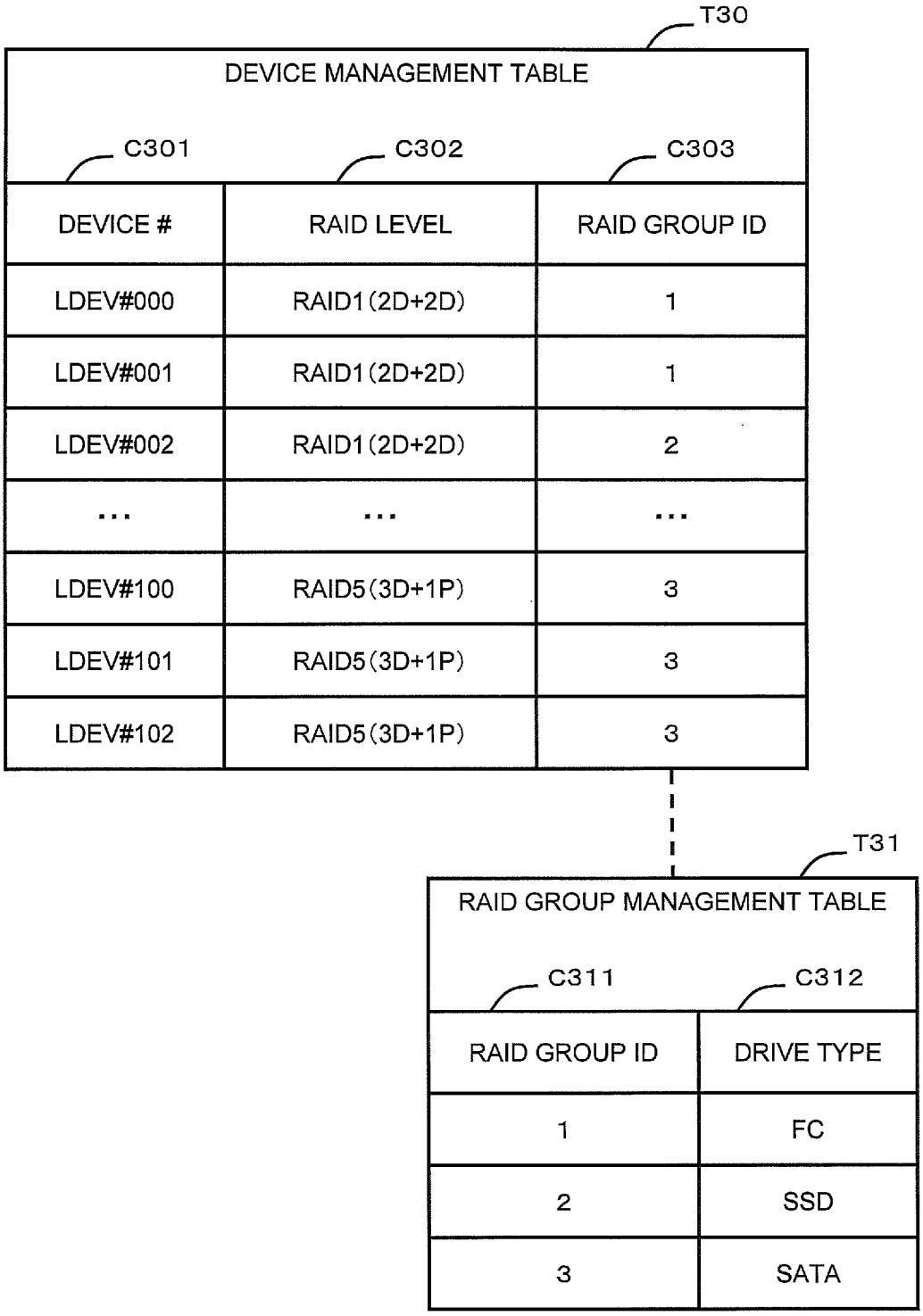
[Fig. 20]

FIG.20


MANAGEMENT TERMINAL
S110
ACCESS SVP, AND DISPLAY MANAGEMENT SCREEN
S111
EDIT MANAGEMENT TABLES FOR CACHE SEGMENT CONTROL
END
DEVICE MANAGEMENT TABLE
T30
RAID GROUP MANAGEMENT TABLE
T31
TIER MANAGEMENT TABLE
T32
CACHE CONTROL MODE SELECTION PROCESSING
S112
REFER TO MANAGEMENT TABLES
S113
SET TO LRU CONTROL?
NO
YES
S114
PERFORM LRU CONTROL
END
S115
USE CONTROL METHOD ACCORDING TO PRESENT INVENTION (CONTROL WHICH PREFERENTIALLY RETAINS SEGMENTS WHOSE LOAD RATIOS ARE HIGH)
END

[Fig. 21]

FIG.21


T30
DEVICE MANAGEMENT TABLE
C301
C302
C303
DEVICE #
RAID LEVEL
RAID GROUP ID
LDEV#000
RAID1(2D+2D)
1
LDEV#001
RAID1(2D+2D)
1
LDEV#002
RAID1(2D+2D)
2
...
...
...
LDEV#100
RAID5(3D+1P)
3
LDEV#101
RAID5(3D+1P)
3
LDEV#102
RAID5(3D+1P)
3
T31
RAID GROUP MANAGEMENT TABLE
C311
C312
RAID GROUP ID
DRIVE TYPE
1
FC
2
SSD
3
SATA

[Fig. 22]

FIG.22

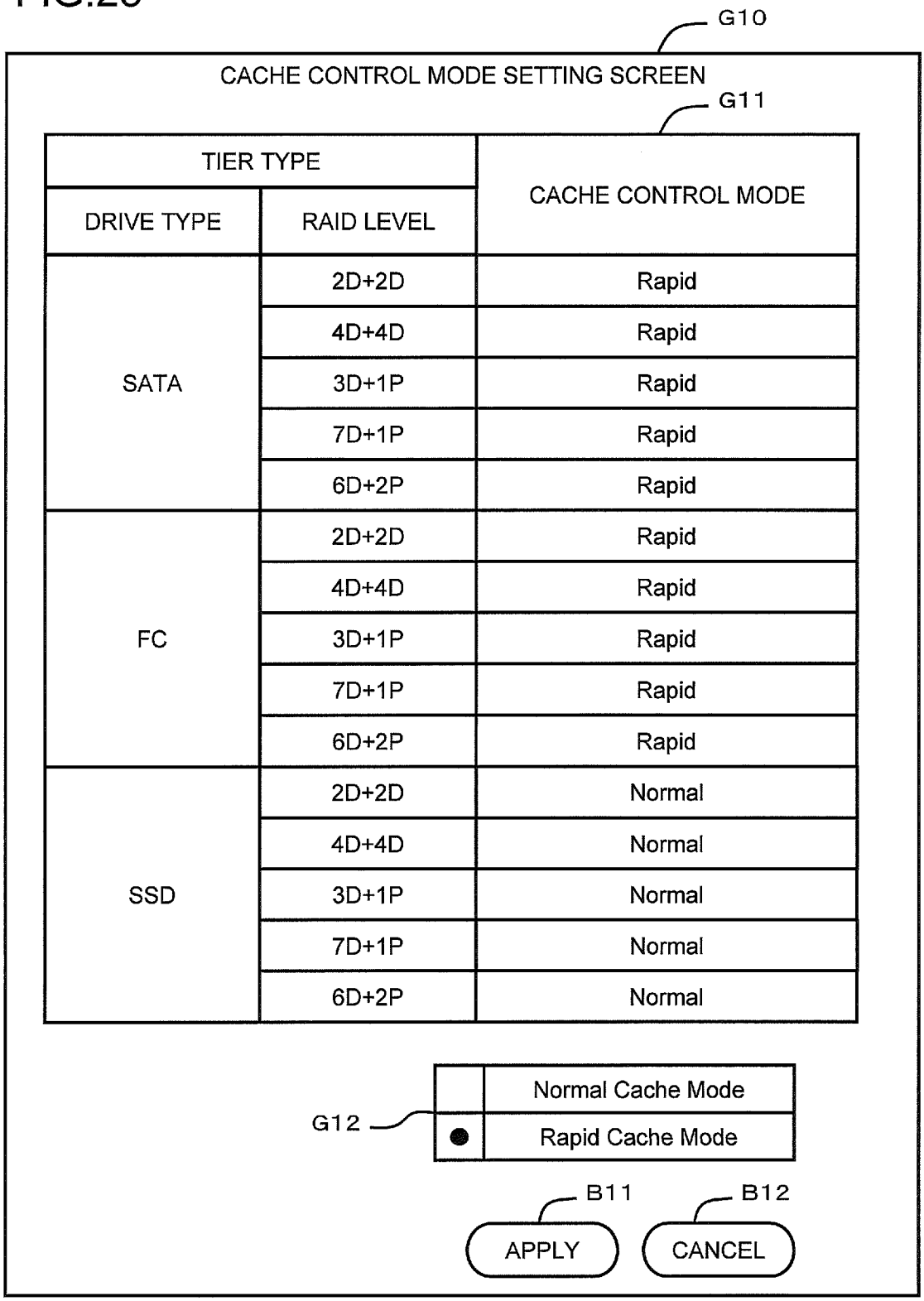
T32

TIER MANAGEMENT TABLE

C321 TIER TYPE		C322 CACHE CONTROL MODE
C3211 DRIVE TYPE	C3212 RAID LEVEL	
SATA	2D+2D	CONTROL ON THE BASIS OF LOAD RATIO
	4D+4D	
	3D+1P	
	7D+1P	
	6D+2P	
FC	2D+2D	CONTROL ON THE BASIS OF LOAD RATIO
	4D+4D	
	3D+1P	
	7D+1P	
	6D+2P	
SSD	2D+2D	LRU CONTROL
	4D+4D	
	3D+1P	
	7D+1P	
	6D+2P	

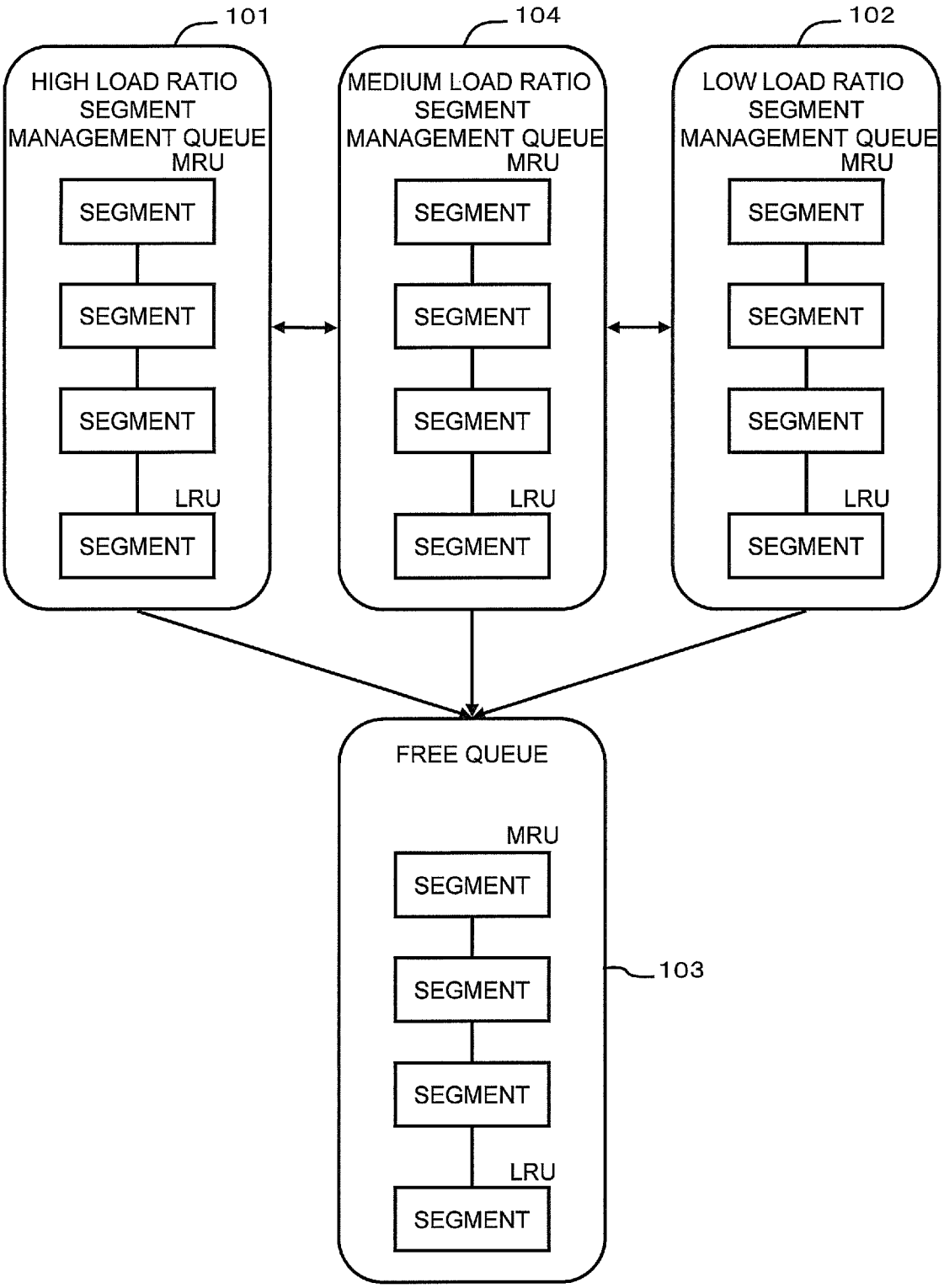
[Fig. 23]

FIG.23


G10
CACHE CONTROL MODE SETTING SCREEN
G11
TIER TYPE
DRIVE TYPE
RAID LEVEL
CACHE CONTROL MODE
SATA
2D+2D
Rapid
4D+4D
Rapid
3D+1P
Rapid
7D+1P
Rapid
6D+2P
Rapid
FC
2D+2D
Rapid
4D+4D
Rapid
3D+1P
Rapid
7D+1P
Rapid
6D+2P
Rapid
SSD
2D+2D
Normal
4D+4D
Normal
3D+1P
Normal
7D+1P
Normal
6D+2P
Normal
G12
Normal Cache Mode
Rapid Cache Mode
B11
B12
APPLY
CANCEL

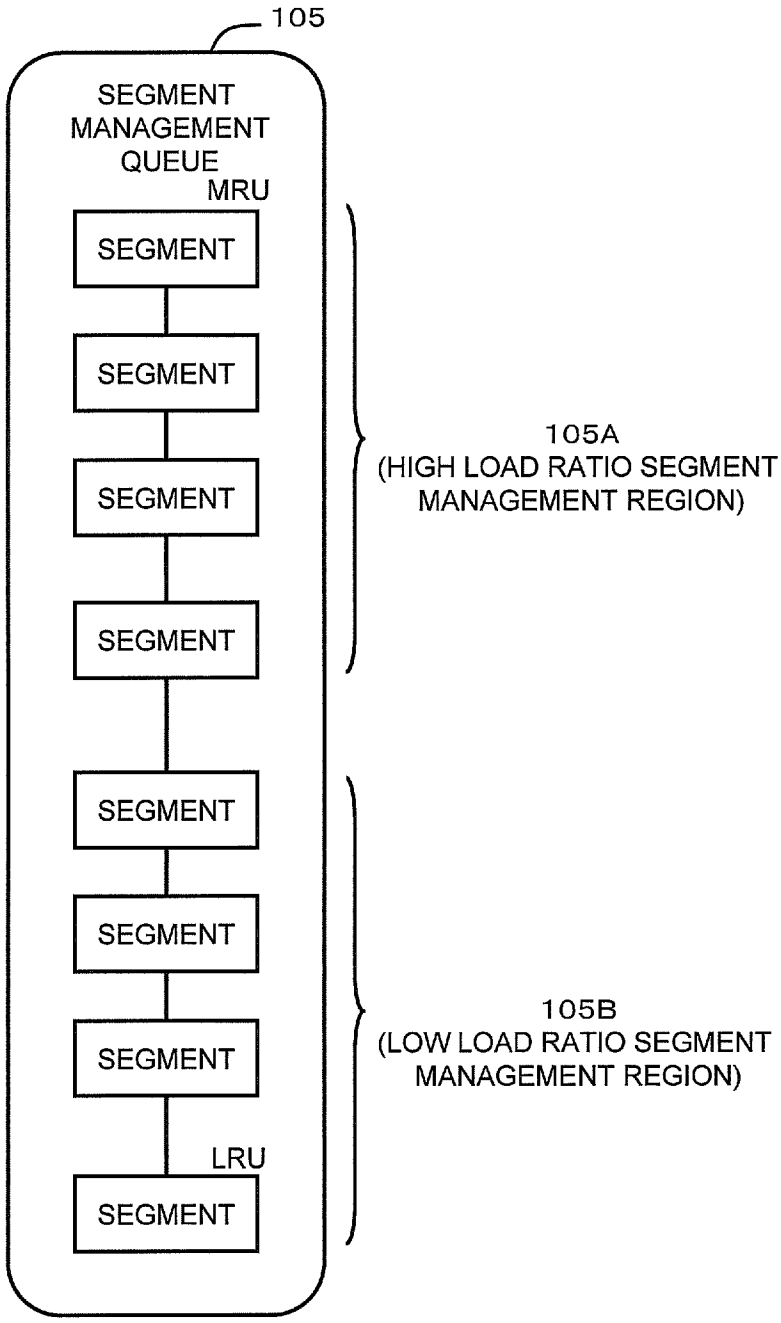
[Fig. 24]

FIG.24


101
HIGH LOAD RATIO SEGMENT MANAGEMENT QUEUE
MRU
SEGMENT
SEGMENT
SEGMENT
LRU
SEGMENT
104
MEDIUM LOAD RATIO SEGMENT MANAGEMENT QUEUE
MRU
SEGMENT
SEGMENT
SEGMENT
LRU
SEGMENT
102
LOW LOAD RATIO SEGMENT MANAGEMENT QUEUE
MRU
SEGMENT
SEGMENT
SEGMENT
LRU
SEGMENT
FREE QUEUE
MRU
SEGMENT
SEGMENT
SEGMENT
LRU
SEGMENT
103

[Fig. 25]

FIG.25


105
SEGMENT MANAGEMENT QUEUE
MRU
SEGMENT
SEGMENT
SEGMENT
SEGMENT
SEGMENT
SEGMENT
SEGMENT
LRU
SEGMENT
105A
(HIGH LOAD RATIO SEGMENT MANAGEMENT REGION)
105B
(LOW LOAD RATIO SEGMENT MANAGEMENT REGION)

INTERNATIONAL SEARCH REPORT

International application No
PCT/JP2009/000416

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F12/12

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

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6 012 126 A (AGGARWAL CHARU CHANDRA [US] ET AL) 4 January 2000 (2000-01-04) abstract; figure 1B column 3, line 35 - column 4, line 13	1-14
Y	US 6 701 393 B1 (KEMENY JOHN [US] ET AL) 2 March 2004 (2004-03-02) figure 3 column 1, line 50 - column 4, line 59	1-14
A	JP 2002 251322 A (HITACHI LTD; HITACHI ASAHI ELECTRON KK) 6 September 2002 (2002-09-06) the whole document	1-14

-/--

[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
16 October 2009	29/10/2009

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	Jardon, Stéphan

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/JP2009/000416

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2006/143394 A1 (PETEV PETIO G [BG] ET AL) 29 June 2006 (2006-06-29) paragraph [0096] - paragraph [0111]; figures 13A,13B,14	1-14
A	KAI CHENG ET AL: "LRU-SP: a size-adjusted and popularity-aware LRU replacement algorithm for web caching" COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, 2000. COMPSAC 2000. THE 24TH ANNUAL INTERNATIONAL TAIPEI, TAIWAN 25-27 OCT. 2000, LOS ALAMITOS, CA, USA,IEEE COMPUT. SOC, US, 25 October 2000 (2000-10-25), pages 48-53, XP010523744 ISBN: 978-0-7695-0792-7 page 48 - page 51 page 51, left-hand column	1-14

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/JP2009/000416

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6012126 A	04-01-2000	NONE	
US 6701393 B1	02-03-2004	NONE	
JP 2002251322 A	06-09-2002	NONE	
US 2006143394 A1	29-06-2006	NONE	