22 DEC 2005



*166166*

*G00002*

# Specification

Title of the Invention:

Data writing method, data reading method, disk drive apparatus and disk drive apparatus controller

Detailed Description of the Invention:

[Field of the invention]

The present invention relates to data reading and writing by a disk drive apparatus, and more particularly, to a method for quickly reading and writing data.

[Background art]

When a host computer issues a plurality of write requests to a hard disk drive apparatus as an external storage apparatus of a computer, it takes a considerable time to execute and complete an actual data write to the hard disk drive apparatus. Therefore, the hard disk drive apparatus notifies a host computer that the hard disk drive apparatus has received a command about a write request and at the same time retains a write command waiting to be executed.

In the case where there is a plurality of write commands waiting to be executed, commands to be executed are sequentially selected according to RPO (Rotational Position Optimization). When a command waiting to be executed is executed, RPO is a technique for estimating a seek time after a seek for a target track on a magnetic disk is started until the relevant track is reached and a disk rotation waiting time after the track is reached until access to a target sector is started, then selecting a command waiting to be executed that corresponds to the smallest sum of this seek time and rotation waiting time as the next command to be executed.

[Problems to be solved by the invention]

RPO allows a command execution time to be shortened, which improves the performance of the hard disk drive apparatus. However, of course, it is desirable to further shorten a command execution time by RPO and improve the performance of the hard disk drive apparatus.

On the other hand, there are cases where while there is a write command of which status is waiting execution, the hard disk drive apparatus receives a request from a host computer for reading data stored in the magnetic disk. In this case, a conventional hard disk drive apparatus executes a read command and performs a data read only after all write commands waiting to be executed are executed, that is, after writes to the magnetic disk are completed. Therefore, in the case where there are write commands of which status is waiting execution, it takes a considerable time to start reading. From this standpoint, too, it is desirable to improve the performance of the hard disk drive apparatus.

Thus, it is an object of the present invention to improve the writing or reading efficiency when there are write commands, which are waiting to be executed and improve the performance of the hard disk drive apparatus.

[Summary of the invention]

The present invention is a method of writing write data that is requested to write by a host device on a recording medium and solves the above-described problem through a data writing method characterized by including a positional relation determining step of determining positional relation of a logical block address of a leading write command and a logical block address of a following write command by comparing the logical block address of the leading write command of which status is waiting execution with the logical block address of the following write command that is newly issued; a distance

determining step of determining whether distance between the logical block address of the leading write command and the logical block address of the following write command is equal to or less than predetermined distance or not in the case where the positional relation determining step determines that the logical block address of the leading write command and the logical block address of the following write command is separate; and a data writing step of writing data corresponding to the leading write command and data corresponding to the following write command sequentially on the recording medium in the case where the distance determining step determines that distance between the logical block address of the leading write command and the logical block address of the following write command is equal to or less than predetermined distance.

In the data writing method of the present invention, in the case where there is a plurality of the leading write commands of which distance is determined by the distance determining step to be equal to or less than predetermined distance, it is desirable that the write data corresponding to the following write command with the least distance be written in the data writing step immediately after writing the write data corresponding to the leading write command.

The present invention is a method of reading data when a host device issues a data read request while a data write request is waiting to be executed and solves the above-described problem through a data reading method characterized by including a positional relation determining step of determining whether there is any overlap between a logical block address of a write command and a logical block address of a read command by comparing the logical block address of the write command of which status is waiting execution and the logical block address of the read command that is newly issued; and a data reading step of executing the read command before all the write commands are executed in the case where the positional relation determining step determines that there is no overlap between the logical block address of the write command and the logical

block address of the read command.

In the data reading method of the present invention, the read command can be executed following the execution of the write command, which corresponds to the least distance between the head of the logical block address of the read command and the tail of the logical block address of the write command.

Furthermore, the present invention is a disk drive apparatus characterized by including a randomly accessible disk-shaped recording medium; command retaining means for retaining a write command sent from an external device until the command is executed; command relation determining means for determining whether there is any overlap between the logical block address of the write command retained in the command retaining means and the logical block address of the command newly sent from the external device; command distance determining means for determining whether the distance between the logical block address of the write command retained in the command retaining means and the logical block address of the newly sent write command is greater than predetermine distance or not in the case where the command relation determining means determines that there is no overlap; and write instructing means for instructing that write data corresponding to the two commands be written on the recording medium sequentially in the case where the command distance determining means determines that the distance between the write command retained in the command retaining means and the newly sent write command is not greater than the predetermined distance.

In the disk drive apparatus of the present invention, the command distance determining means specifies a write command retained in the command retaining means corresponding to the least distance for the newly sent write command in the case where there is a plurality of write commands retained in the command retaining means of which distance is equal to or less than the predetermined distance.

Furthermore, the present invention is a disk drive apparatus characterized by including a randomly accessible disk-shaped recording medium; command retaining means for retaining a write command sent from an external device until the command is executed; command comparing means for determining relation between a logical block address of a read command for a read request sent from the external device and a logical block address of a write command retained in the command retaining means; and read instructing means for instructing a read based on the comparison result of the command comparing means.

In the above-described disk drive apparatus, the command comparing means can determine whether there exists the read command that overlaps the write command at the logical block addresses or not.

Furthermore, in the above-described disk drive apparatus, the read instructing means can instruct the command retaining means to execute the read command after executing the overlapping read command from among the write commands retained in the command retaining means in the case where the command comparing means determines that there exists the read command that overlaps the write command at the logical block addresses.

Furthermore, the read instructing means can instruct that the read command be executed while the write command waiting to be executed is retained in the command retaining means in the case where the command comparing means determines that the write command and the read command do not overlap at the logical block addresses.

Furthermore, the present invention provides a disk drive apparatus controller that controls data writing to a randomly accessible disk-shaped recording medium characterized by including command retaining means for retaining a leading write command sent in advance from an external device until the command is executed; command comparing means for determining whether there exists the leading write command of which

distance from the following write command newly sent from the external device is equal to or less than predetermined distance at logical block addresses; and write instructing means for instructing that the following write command be executed ahead of the leading write command in the case where the command comparing means determines that there exists the leading write command of which distance is equal to or less than the predetermined distance.

Furthermore, the present invention provides a disk drive apparatus controller that controls data reading from a randomly accessible disk-shaped recording medium characterized by including command retaining means for retaining a plurality of leading write commands sent in advance from an external device until the commands are executed; command comparing means for determining whether there exists the leading write command that overlaps a read command newly sent from the external device at logical block addresses or not; and read instructing means for instructing that the read command be executed ahead of execution of the write command in the case where the command comparing means determines that there exists no leading write command that overlaps the read command at the logical block addresses.

[Preferred embodiments]

The present invention will be explained based on embodiments below.

Figure 1 illustrates an outlined configuration of an HDD (Hard Disk Drive) apparatus 10 as an external storage apparatus of a computer that implements the present invention.

As shown in the figure, the HDD apparatus 10 is provided with a magnetic disk 21 that stores data as a recording medium and a spindle motor 22 that rotates and/or drives the magnetic disk 21. A magnetic head 23 records and/or reproduces (writes/reads) data to the magnetic disk 21. A head arm 24 is

provided with the magnetic head 23 at the end and moves over the recording surface of the magnetic disk 21. Moreover, an actuator 25 holds the head arm 24 and rotates and/or drives the head arm 24. In this way, the magnetic head 23 is configured so as to move in the quasi-radial direction of the magnetic disk 21 and randomly access the recording surface of the magnetic disk 21.

A driving mechanism configured by the magnetic disk 21, spindle motor 22, magnetic head 23 and actuator 25 is controlled by a control circuit 30. This control circuit 30 is provided with an HDC (Hard Disk Controller) 31, ROM 32, RAM 33 and host I/F (interface) 34 and these are mutually connected via a bus 35.

The HDC 31 controls the entire HDD apparatus 10 as an internal controller of the HDD apparatus 10 according to a control program and control data stored in the ROM 32. This HDC 31 performs operation processing for servo control and data write/read control. In this way, the HDC 31 drives the spindle motor 22 and actuator 25 and performs writing/reading using the recording head and reproduction head of the magnetic head 23. Furthermore, the HDC 31 performs control for retaining part of the data recorded in the magnetic disk 21 as the recording medium in the RAM 33 and reading in advance part of the data stored in the magnetic disk 21 and storing the data in the RAM 33.

The ROM 32 stores the control program to be executed by the HDC 31 and control data used by this control program.

The RAM 33 temporarily stores write data to be recorded in the magnetic disk 21 and has a function as a cache memory to temporarily store the read data read from the magnetic disk 21.

On the other hand, the host I/F 34 is an interface circuit that sends/receives data and commands to/from the host 40.

Figure 2 shows a main functional block of the HDC 31.

The HDC 31 is provided with an MPU (micro processing unit) 41, an MM (memory manager) 42, a RAM 43 and an SG (segment handler) 44.

The RAM 43 stores a table in a configuration shown in Figure 16 and write segments in a configuration shown in Figure 17 in a table form. The table in Figure 16 stores NO OF HIT (number of OVERLAY HITs), HEAD BEST SKIP LENGTH, HEAD BEST SKIP SEGMENT ID, TAIL BEST SKIP LENGTH, and TAIL BEST SKIP SEGMENT ID, which will be described later. Moreover, the write segment in Figure 17 is configured by 4 words, of which LBA uses the first and second words, the block length (LENGTH) uses the third word and HIT STATUS uses the fourth word.

The SG 44 accesses the RAM 43 via the MM 42 and searches the table in the RAM 43. Furthermore, the SG 44 is provided with a register that temporarily stores SKIP LENGTH, which will be described later.

This embodiment determines positional relation (hereinafter referred to as "LBA") between a logical block address of a leading write command already sent from the host 40 and a logical block address of a following write command newly sent from the host 40. Information on the LBA, that is, the head and tail LBAs for the write command and block length are sent from the host 40 together with the write command and write data. This write data is temporarily stored in the RAM 33 and the information on the LBA is stored in the write segment shown in Figure 17.

In this embodiment, suppose a leading write command is referred to as "buffer data" and a following write command is simply referred to as "write command." Furthermore, in this embodiment, positional relation on LBA may be simply referred to as "relation" or "positional relation."

In this embodiment, as a first stage, it is determined, based on the LBA, which of 8 types of NO HIT-A, TAIL_SEQ, OV_TAIL,

OV_ALL, OV_COVER, OV_HEAD, HEAD_SEQ and NO HIT-B shown in Figure 3 the relation between the buffer data and write command falls into. This determination is performed based on the flow shown in Figure 4 to Figure 6.

Then, as a second stage, the relation between the buffer data and write command is determined in detail based on the flow shown in Figure 7 to Figure 14 also using the LBA. Then, it is the HDD apparatus according to this embodiment that executes RPO based on this determination result.

First, the 8 types determined in the first stage will be explained based on Figure 3.

In Figure 3, the LBA at the tail to which buffer data is written is described as "LLBA." On the other hand, the block length of the buffer data is described as "LENGTH." Furthermore, the LBA at the head to which a write command is written is described as "SLBA" and the LBA at the tail is described as "ELBA." The block length of the LBA of the write command is described as "SLEN."

#1 to #8 in Figure 3 indicate write commands. First, it should be noted that #1 to #8 are described here to classify write commands for convenience, and #1 to #8 do not to indicate the order in which the commands are sent from the host 40.

In this embodiment, the positional relation of write commands relative to the buffer data is classified into 8 types as shown in Figure 3. The content of each type is as shown below:

<NO HIT-A>

Write command #1 exists in a location distant from the LLBA of the buffer data. As shown above, in the case where write command #1 has no overlap with the buffer data, this case is determined as NO HIT-A.

<TAIL_SEQ>

As in the case of write command #2, in the case where the tail of the buffer data is sequentially connected by the head of write command #2, this case is determined as TAIL_SEQ.

<OV_TAIL>

As in the case of write command #3, where the buffer data overlaps with write command #3 on the tail side of the buffer data, this case is determined as OV_TAIL.

<OV_ALL>

Write command #4 is completely included in the buffer data. This case is determined as OV_ALL.

<OV_COVER>

Write command #5 includes the whole buffer data. This case is determined as OV_COVER.

<OV_HEAD>

As in the case of write command #6, where the buffer data overlaps with write command #6 on the head side of the buffer data, this case is determined as OV_HEAD.

<HEAD_SEQ>

As in the case of write command #7, where the head of the buffer data is sequentially connected to the tail of command #7, this case is determined as HEAD_SEQ.

<NO HIT_B>

As in the case of write command #8, where write command #8 is distant from the buffer data on the head side of the buffer

data, this case is determined as NO HIT_B.

The above-described determination is made by the SG (segment handler) 44 of the HDC 31 comparing the LBA information of the write command with the LBA information of the segment table stored in the RAM 33. The specific flow will be explained based on Figure 4 to Figure 6.

In Figure 4 to Figure 6, it is determined whether LLBA<SLBA-1 or not in step S11 and the process moves on to step S12 in the case where LLBA<SLBA-1 and moves on to step S13 otherwise. The determination in step S11 is based on a positional comparison between the LBA at the tail of the buffer data and LBA at the head of the write command. In the case where LLBA<SLBA-1, since there is no overlap on LBA between the buffer data and write command in step S12, the relation of the write command with the buffer data, that is, HIT STATUS is determined to be HIT-A. "A" indicates that the write command exists behind the buffer data.

In step S13, it is determined whether LLBA=SLBA-1 or not, that is, whether the LBA at the tail of the buffer data is sequentially connected to the LBA at the head of the write command or not. In the case where the LBA at the tail of the buffer data is sequentially connected to the LBA at the head of the write command, the hit status is determined as TAIL_SEQ in step S14. In the case where LLBA=SLBA-1 is not the case, the process moves on to step S15 (hereinafter Figure 5).

In step S15, it is determined whether LENGTH<LLBA-SLBA+1 or not. In the case where LENGTH<LLBA-SLBA+1, the process moves on to step S19, and moves on to step S16 otherwise.

In step S16, it is determined whether LLBA<ELBA or not. In the case where LLBA<ELBA, the hit status is determined as OV_TAIL in step S17, and the hit status is determined as OV_ALL otherwise in step S18.

On the other hand, in step S19, it is determined whether LLBA≤ELBA or not and in the case where LLBA≤ELBA, the hit status is determined as OV_COVER in step S20, and the process moves on to step S21 otherwise.

In step S21, it is determined whether LENGTH<LLBA-ELBA or not. In the case where LENGTH<LLBA-ELBA, the hit status is determined as NO HIT-B in step S22. Otherwise, the process moves on to step S23 (hereinafter Figure 6).

In step S23, it is determined whether LENGTH=LLBA-ELBA or not and in the case where LENGTH=LLBA-ELBA, the hit status is determined as HEAD_SEQ. Otherwise, the hit status is determined as OV_HEAD.

This completes the determination in the first stage.

Then, the second-stage processing will be explained based on Figure 7 to Figure 14.

<NO HIT-A>

When the case is determined as NO HIT-A, the determination shown in Figure 7 is made. That is, in step S101, it is determined whether SKIP LENGTH is greater than SKIP CRITERIA or not. Here, SKIP LENGTH is a value defined as LLBA-SLBA+1 and indicates the distance between the tail of the buffer data and the head of the write command.

In the case where SKIP LENGTH is greater than SKIP CRITERIA, it is determined in step S102 that the write command is NO HIT with respect to the buffer data. That is, in the case where the distance between the head of the write command and the tail of the buffer data is greater than SKIP CRITERIA, which is predetermined distance, this case is determined as NO HIT. If determined as NO HIT, it is assumed that there is no relation between the buffer data and write command and processing such as a sequential write is not performed when a data write is

performed.

In the case where SKIP LENGTH is equal to or less than SKIP CRITERIA, it is determined in step S103 that the write command is TAIL HIT relative to the buffer data. That is, it is assumed that though the buffer data is separate from the write command and the distance is equal to or less than predetermined distance, and therefore these two are regarded as continuous data and a write will be performed sequentially in execution of a data write. Therefore, this is effective for performance improvement by RPO. This is the same as the case determined as NO HIT-B.

In the case where the case is determined as TAIL HIT, it is determined in step S104 whether TAIL BEST SKIP LENGTH so far is less than 0 or not. In the case where TAIL BEST SKIP LENGTH is less than 0, the processing ends. Otherwise, it is determined in step S105 whether SKIP LENGTH this time is smaller than TAIL BEST SKIP LENGTH or not. In the case where SKIP LENGTH is smaller than TAIL BEST SKIP LENGTH, SKIP LENGTH this time is designated as TAIL BEST SKIP LENGTH in step S106, and TAIL BEST SKIP LENGTH and TAIL BEST SKIP SEGMENT ID are updated. Otherwise, the processing ends.

Here, TAIL BEST SKIP LENGTH will be explained with reference to Figure 23. Figure 23 shows a write command resulting in TAIL HIT for two buffer data items. However, it is observed that SKIP LENGTH of the buffer data with segment ID of 2 is smaller than that of the buffer data with segment ID of 1. In this case, TAIL BEST SKIP LENGTH of the write command becomes SKIP LENGTH of the buffer data with segment ID of 2 and TAIL SKIP SEGMENT ID becomes 2. TAIL BEST SKIP LENGTH and TAIL SKIP SEGMENT ID are stored in the table shown in Figure 16.

<TAIL_SEQ>

For the case determined as TAIL_SEQ, the processing shown in Figure 8 is performed.

In the case of TAIL_SEQ, the hit status is determined as a TAIL HIT in step S201. Then, it is determined in step S202 whether TAIL BEST SKIP LENGTH so far is less than 0 or not. In the case where TAIL BEST SKIP LENGTH is less than 0, the processing ends. Otherwise, it is determined in step S203 whether SKIP LENGTH is smaller than TAIL BEST SKIP LENGTH or not. In the case where SKIP LENGTH is smaller than TAIL BEST SKIP LENGTH, TAIL BEST SKIP LENGTH and TAIL BEST SKIP SEGMENT ID are updated in step S204.

<OV_TAIL>

For the case determined as OV_TAIL, the processing shown in Figure 9 is performed.

First, in step S301, the hit status is determined as a TAIL HIT. Then, in step S302 it is determined whether SKIP LENGTH is smaller than TAIL BEST SKIP LENGTH or not. In the case where SKIP LENGTH is smaller than TAIL BEST SKIP LENGTH, TAIL BEST SKIP LENGTH and TAIL BEST SKIP SEGMENT ID are updated in step S303. Otherwise, the processing ends.

<OV_ALL>

For the case determined as OV_AIL, the processing shown in Figure 10 and Figure 11 is performed.

First, in step S401, it is determined whether LLBA matches ELBA. In the case where LLBA matches ELBA, the process moves on to step S402 and in the case where LLBA does not match ELBA, the process moves on to step S408 in Figure 11.

In step S402, it is determined whether SLBA=LLBA-LENGTH+1 or not. In the case where SLBA=LLBA-LENGTH+1, the process moves on to step S403 or moves on to step 405 otherwise.

In step S404, in the case where it is determined that SLBA matches the LBA at the head of the buffer data, the hit status

is determined as COVER OVERLAY, then in step S404, the count of OVERLAY HIT is incremented by 1.

In step S405, the hit status is determined as TAIL HIT. Then, in step S406, it is determined whether SKIP LENGTH is smaller than TAIL BEST SKIP LENGTH or not. In the case where SKIP LENGTH is smaller than TAIL BEST SKIP LENGTH, TAIL BEST SKIP LENGTH and TAIL BEST SKIP SEGMENT ID are updated in step S407. The processing ends otherwise.

In step S408, it is determined whether SLBA=LLBA-LENGTH+1 or not. In the case where SLBA=LLBA-LENGTH+1, the process moves on to step S409, or moves on to step S412 otherwise.

In step S409, the hit status is determined as HEAD HIT. Then, in step S410, it is determined whether SKIP LENGTH (here SLEN) is smaller than HEAD BEST SKIP LENGTH or not. In the case where SKIP LENGTH is smaller than HEAD BEST SKIP LENGTH, the process moves on to step S411 or the process ends otherwise.

In step S411, HEAD BEST SKIP LENGTH and HEAD SKIP SEGMENT ID are updated and then the processing ends.

In S412, the hit status is determined as ALL OVERLAY and then in step S413, the count of OVERLAY HIT is incremented by 1 and the processing ends. Here, the count of OVERLAY HIT is incremented in the cell of NO OF HIT in the table shown in Figure 16.

<OV_COVER>

For the case determined as OV_COVER, the hit status is determined as ALL OVERLAY in step S501 as shown in Figure 12 and then in step S502, the count of OVERLAY HIT is incremented by 1 and the processing ends.

<OV_HEAD>

For the case determined as OV_HEAD, the processing shown in Figure 13 is performed.

In the case of OV_HEAD, the hit status is determined as HEAD HIT in step S601. Then, in step S602, it is determined in step S602 whether SKIP LENGTH (determined here as (LLBA-LENGTH+1)-ELBA) is smaller than HEAD BEST SKIP LENGTH or not. In the case where SKIP LENGTH is smaller than HEAD BEST SKIP LENGTH, HEAD BEST SKIP LENGTH and HEAD BEST SKIP SEGMENT ID are updated in step S603. The processing ends otherwise.

<HEAD_SEQ>

For the case determined as HEAD_SEQ, the processing shown in Figure 14 is performed.

In the case of HEAD_SEQ, the hit status is determined as HEAD HIT in step S701. Then, in step S702, it is determined whether BEST HIT SKIP LENGTH so far is negative or not. In the case where BEST HIT SKIP LENGTH so far is negative, the processing ends. In the case where BEST HIT SKIP LENGTH so far is not negative, the process moves on to step S703.

In step 703, it is determined whether SKIP LENGTH is smaller than HEAD BEST SKIP LENGTH or not. This SKIP LENGTH is zero. In the case where SKIP LENGTH is smaller than HEAD BEST SKIP LENGTH, in step S704, HEAD BEST SKIP LENGTH and HEAD BEST SKIP SEGMENT ID are updated and the processing ends.

<NO HIT-B>

For the case determined as NO HIT-B, the processing shown in Figure 15 is performed. That is, in step S801, it is determined whether (LLBA-LENGTH+1)-ELBA is greater than SKIP CRITERIA or not.

In the case where SKIP LENGTH is greater than SKIP CRITERIA, the hit status is determined as NO HIT in step S802.

In the case where SKIP LENGTH is equal to or smaller than SKIP CRITERIA, the hit status is determined as HEAD HIT in step S803 and the process moves on to step S804.

In step S804, it is determined whether SKIP LENGTH is smaller than HEAD BEST SKIP LENGTH or not. Here, this SKIP LENGTH is defined as (LLBA-LENGTH+1)-ELBA. In the case where HEAD BEST SKIP LENGTH is equal to or smaller than SKIP LENGTH, the processing ends. In the case where SKIP LENGTH is smaller than HEAD BEST SKIP LENGTH, HEAD BEST SKIP LENGTH and HEAD BEST SKIP SEGMENT ID are updated in step S805 and the processing ends.

As is explained above, this embodiment adopts the technique of roughly classifying the relation between buffer data and a write command in the beginning as shown in Figure 3 and then more specifically determining the relation between the buffer data and command as shown in Figure 7 to Figure 15. Figure 24 shows the relations between NO HIT-A, TAQIL_SEQ, OV_TAIL, OV_ALL, OV_COVER, OV_HEAD, HEAD_SEQ, NO HIT_B and each hit status determined in Figure 7 to Figure 15 including the positional relation with the buffer data.

The above-described determinations are made every time the host 40 issues a write request by the segment handler (SG) accessing the RAM 43 in which the segment table is stored via the memory manager (MM). The write segment that makes up the segment table is configured by 4 words as shown in Figure 16. Of the 4 words, LBA uses the 1st and 2nd words, BLOCK LENGTH (LENGTH) uses the 3rd word and HIT STATUS uses the 4th word.

A conventional HDD has only one set of this write segment and does not use the 4th word. On the other hand, this embodiment provides 64 sets of the write segment shown in Figure 16 and moreover uses the 4th word, which has been unused conventionally, for the hit status. Therefore, the write

command is stored in the RAM 43.

The hit status stored in this write segment is reflected in RPO (Rotational Position Optimization). When a command waiting to be executed is executed, RPO estimates a seek time after a seek to a target track on the magnetic disk 21 is started until the relevant track is reached and a disk rotation waiting time after the relevant track is reached until access to the target sector is started. RPO is the technique of selecting a command waiting to be executed corresponding to the least sum of this seek time and rotation waiting time as the next command to be executed.

As described above, this embodiment can link write commands that can be regarded as a series of data from among a plurality of write commands by referencing the hit status stored in the write segment. That is, in the case where commands are not mutually NO-HIT, the commands can be deemed as the targets to be written sequentially as a cluster of a series of commands.

As described above this embodiment provides 64 sets of write segment, and therefore there can be a plurality of the above-described command clusters. In the case where such a plurality of command clusters exists, writing is executed starting with a command cluster corresponding to the least sum of a seek time and rotation waiting time according to the conventional RPO. Of course, there can also be write commands that do not form a command cluster, and therefore the write command execution sequence is determined between such write commands or command groups according to the conventional RPO.

The processing concept above will be explained using Figure 18.

Figure 18(a) shows a plurality of commands (1) to (6) requested to write by the host 40. Numbers (1) to (6) assigned to the commands denote the sequence of write requests sent from the host and the horizontal axis denotes LBA.

In Figure 18(a), based on the aforementioned comparison of LBA, it is determined that command (1) and command (3), and command (2) and command (5) form a series of data groups, respectively. However, command (4) and command (6) have nothing to do with other commands and can be determined as NO HIT. Therefore, commands (1) to (6) can be replaced with 6 data groups A to D shown in Figure 18(b). Then, the sequence of writing is arranged not in the sequence of the commands sent from the host 40 but in a sequence of A, B, C and D.

Then, the processing when writing of data groups determined as OVERLAY HIT is performed will be explained based on Figure 19.

As shown in Figure 19, suppose the LBA of a leading write request command (buffer data) is 1, 2 and 3 and the LBA of a following write request command (write command) is 0 and 1. The hit status of the write command corresponding to the buffer data is OVERLAY HIT and the block LBA1 overlaps. In the case where a buffer data write is executed after simply executing a write command according to RPO, the data of the write command is overwritten with the data of the buffer data for the LBA1 block. Since the data of the buffer data is older than the data of the write command, the above-described overwrite causes the old data to overwrite the new data.

Thus, this embodiment erases (DISCARDs) the overlapping block of the buffer data as shown on the right side of Figure 19. Then, this embodiment executes a write of the buffer data and then executes writing of a write command. This allows new data to be written also in the overlapping section.

Then, the processing in the case where the host 40 issues a read request command while a plurality of write commands is waiting to be executed will be explained.

When a read request is issued while write commands are waiting to be executed, the conventional HDD apparatus executes the read request command after all write commands waiting to be

executed are executed. However, executing the read request command after all write commands are executed would cause a data read to be delayed. What should be given priority at this moment is the read command. Thus, when the host 40 issues a read request command while a plurality of write commands is waiting to be executed, this embodiment allows a read command to be executed as an interrupt before all write commands are executed.

What should be noted when a read request is issued while a plurality of write commands is waiting to be executed and the write commands are interrupted by the read request command is a case where there is an overlap between the read request command and write request commands waiting to be executed. This will be explained using Figure 20.

Figure 20 shows relation between one write command and one read command for ease of understanding.

Figure 20(a) shows a case where there is no overlap on LBA between a leading write command ("W" in the figure) and a following read command ("R" in the figure). In this case, there is no problem in the case where read command R is executed first before the write command W is executed. However, as shown in Figure 20(b), in the case where there is an overlap on LBA between write command W and read command R, executing read command R first before executing write command W will involve the following problem:

The read request from the host 40 should be issued to the latest data. Figure 20(b) shows the case where data by write command W is the latest data. As described above, there is an overlap on LBAs between write command W and read command R. In this overlapping section, the data to be read by write command R is the data by write command W. Therefore, in the case where read command R is executed before write command W is executed and the data is actually written, the data read by execution of this read command R includes part that is not the latest part.

If this is the case, read command R cannot be considered to have been executed correctly.

In the case where a read request is issued when there is a plurality of write commands waiting to be executed, this embodiment checks the positional relation between the LBA of the write commands waiting to be executed and the LBA of the read command requested from the host 40. Then, in the case where there is a write command waiting to be executed which have the positional relation shown in Figure 20(b), the write command is executed first, then the read command is allowed to interrupt and executed ahead of other write commands waiting to be executed.

Figure 21 shows an example thereof. In Figure 21(a), suppose W1 to W4 are write commands waiting to be executed and R is a read command requested from the host 40 while write commands W1 to W4 are waiting to be executed.

As shown in Figure 21(a), there is an overlap on LBAs between read command R and write commands W2, W3. Therefore, in order to execute read command R, write commands W1, W2 and W3 are executed and then read command R is executed as shown in Figure 21(b). Then, write command W4 is executed after read command R is executed. By doing so, data read by execution of read command R includes the latest data. (1) to (5) in Figure 21(b) denote the execution sequence.

Figure 21 shows the case where there is an overlap on LBAs between read command R and write commands W2, W3. In the case where there is no overlap on LBAs between read command R and write commands as shown in Figure 22(a), it is also possible to allow read command R to interrupt write command W1 and execute read command R before write command W1 as shown in Figure 22(b).

[Advantages of the invention]

As explained above, when there are write commands waiting to be executed, the present invention can improve the efficiency of writing or reading and improve the performance of a hard disk drive apparatus.

Brief Description of the Drawings:

Figure 1 is a schematic configuration of an HDD (Hard Disk Drive) apparatus according to this embodiment;

Figure 2 is a main configuration of an HDC (Hard Disk Controller) of the HDD according to this embodiment;

Figure 3 is an illustration to explain relation between buffer data and write commands;

Figure 4 is a flow chart to determine relation between buffer data and write commands;

Figure 5 is a flow chart to determine relation between buffer data and write commands;

Figure 6 is a flow chart to determine relation between buffer data and write commands;

Figure 7 is a flow chart showing subsequent processing when relation between buffer data and write commands is NO HIT-A;

Figure 8 is a flow chart showing subsequent processing when relation between buffer data and write commands is TAIL_SEQ;

Figure 9 is a flow chart showing subsequent processing when relation between buffer data and write commands is OV_TAIL;

Figure 10 is a flow chart showing subsequent processing when

relation between buffer data and write commands is OV_ALL;

Figure 11 is a flow chart showing subsequent processing when relation between buffer data and write commands is OV_ALL;

Figure 12 is a flow chart showing subsequent processing when relation between buffer data and write commands is OV_COVER;

Figure 13 is a flow chart showing subsequent processing when relation between buffer data and write commands is OV_HEAD;

Figure 14 is a flow chart showing subsequent processing when relation between buffer data and write commands is HEAD_SEQ;

Figure 15 is a flow chart showing subsequent processing when relation between buffer data and write commands is NO HIT-B;

Figure 16 is a table stored in RAM 43;

Figure 17 is an illustration showing a configuration of a write segment;

Figure 18 is an illustration to explain a data write according to this embodiment;

Figure 19 is an illustration to explain a data write in the case of OVERLAY HIT according to this embodiment;

Figure 20 is an illustration showing relation between a write command and a read command;

Figure 21 is an illustration to explain a state in which a read command is issued when a plurality of write commands is waiting to be executed;

Figure 22 is an illustration to explain a state in which a read command is issued when a plurality of write commands is waiting to be executed;

Figure 23 is an illustration to explain BEST TAIL SKIP LENGTH; and

Figure 24 is an illustration showing positional relation between buffer data and hit statuses determined in Figure 7 to Figure 15.

[Description of symbols]

10 ... HDD (Hard Disk Drive) apparatus
21 ... Magnetic disk
22 ... Spindle motor
23 ... Magnetic head
24 ... Head arm
25 ... Actuator
30 ... Control circuit
31 ... HDC (Hard Disk Controller)
32 ... ROM
33 ... RAM
34 ... Host I/F (interface)
35 ... Bus
40 ... Host
41 ... MPU (micro processing unit)
42 ... MM (memory manager)
43 ... RAM
44 ... SG (segment handler)

CLAIMS:

1. A data writing method of writing write data received from a host device onto a recording medium, comprising:

comparing a first logical block address of a leading write command awaiting execution with a second logical block address of a following write command that is newly issued to determine a distance therebetween;

determining whether said distance is equal to or less than a predetermined distance in the case where said first and second logical block addresses are determined to be separate;

writing data corresponding to said leading write command and data corresponding to said following write command sequentially on said recording medium in the case where the distance is determined to be equal to or less than said predetermined distance; and

if there are a plurality of following write commands having each having a respective distance from said leading write command equal to or less than said predetermined distance, writing data corresponding a particular one of said plurality of following write commands having a least distance from said first logical block address immediately after writing data corresponding to said leading write command.

2. A data reading method when a host device issues a data read request while one or more data write commands are waiting to be executed, said method comprising:

determining whether there is any overlap between a first logical block address of a selected write command among said one or more write commands awaiting execution and a second logical block address of a subsequently issued read command; and

retaining said selected write command in a command cache and executing said read command before executing said selected write commands in response to determining that there is no overlap between said first and second logical block addresses.

3. The data reading method according to claim 2, wherein said read command is executed following the execution of a selected write command among said one or more write commands having a least distance between a head of said logical block address of said selected write command and a tail of said logical block address of said read command.

4. A disk drive apparatus comprising:

a command cache for retaining a write command sent from an external device until said write command is executed;

means for determining whether there is any overlap between a first logical block address of said write command retained in said command cache and a second logical block address of a subsequent write command newly sent from said external device;

means, responsive to determining that there is no overlap, for determining whether a distance between the first and the second logical block addresses is greater than a predetermined distance; and

means, responsive to determining that the distance between the first and second logical block addresses is not greater than said predetermined distance, for instructing that write data corresponding to said two commands be written in said recording medium sequentially, said means for instructing including:

means, if there are a plurality of following write commands having each having a respective distance from said leading write command equal to or less than said predetermined distance, for writing data corresponding a particular one of said plurality of following write

commands having a least distance from said first logical block address immediately after writing data corresponding to said leading write command.

5. The disk drive apparatus of claim 4, and further comprising:

a randomly accessible disk-shaped recording medium.

6. A disk drive apparatus comprising:

a command cache for retaining a write command sent from an external device until said command is executed;

means for determining a relation between a first logical block address of a read command for a read request sent from said external device and a second logical block address of said write command retained in said command cache; and

means for instructing a read access to said disk-shaped recording medium in response to determining the relation between said first and second logical block addresses said means for instructing including:

means, responsive to determining that said first and second logical block addresses do not overlap, for instructing that said read command be executed while said write command waiting to be executed is retained in said command cache.

7. The disk drive apparatus of claim 6, and further comprising:

a randomly accessible disk-shaped recording medium.

8. The disk drive apparatus according to claim 7, said means for determining a relation comprising means for determining whether or not said first and second logical block addresses overlaps.

9. The disk drive apparatus according to claim 7, and further comprising means, responsive to determining that said first and second logical block addresses overlap, for instructing said command cache to execute said read command after executing said write commands.

10. A disk drive apparatus that controls data writing to a randomly accessible disk-shaped recording medium, comprising:

a command cache for retaining a leading write command sent in advance from an external device until said leading write command is executed;

means for determining whether a distance between a first logical block address of said leading write command buffered in said command cache and a second logical block address of a following write command subsequently received from said external device is equal to or less than a predetermined distance; and

means, responsive to determining that there said distance is equal to or less than said predetermined distance, for instructing that said following write command is to be executed before said leading write command.

11. The disk drive apparatus of claim 10, and further comprising:

a randomly accessible disk-shaped recording medium.

12. A data writing method of writing write data received from a host device onto a disk recording medium, said method comprising:

buffering a leading write command sent in advance from an external device until said leading write command is executed;

determining whether a distance between a first logical block address of said leading write command and a second logical block address of a following write command subsequently received from said external device is equal to or less than a predetermined distance; and

in response to determining that said distance is equal to or less than said predetermined distance, causing said following write command to be executed before said leading write command.