METHOD AND APPARATUS FOR WRITING DATA ONTO MAGNETIC TAPE

1. BACKGROUND OF THE INVENTION

This invention relates in general to magnetic tape storage. More particularly this invention relates to a method for writing data onto magnetic tape.

Conventional magnetic tape storage devices store data on magnetic tape in the form of blocks of data using conventional read-after-write heads. Conventional magnetic tape media block topologies require well ordered and sequential block placement on the magnetic tape storage medium.

Such conventional magnetic tape media block topologies suffer from two undesirable characteristics. First, the conventional read-after-write configuration coupled with the required data block sequentiality results in storage capacity loss whenever block re-writes are necessary due to write errors. Second, existing magnetic tape media block topologies do not readily permit continued writing if a read/write channel problem or localized media anomaly arises.

The present invention is directed to overcoming, or at least reducing the effects of, one or more of the problems set forth above. In particular, the present invention provides a method of writing data onto a magnetic tape storage media in the form of data envelopes which include one or more data entities. The data entities each include data blocks and error correction code (ECC) blocks. The storage of data blocks and ECC blocks of the data entities contained within the data envelopes can be in any order. The resulting block storage topology eliminates most of the storage capacity loss due to re-writes of data and ECC blocks. The resulting block storage topology also permits continued writing of data and ECC blocks, despite the presence of read/write channel problems and/or localized media anomalies, by migrating blocks onto other good channels or away from media aberrations. Thus the present invention provides a significant improvement in tape storage capacity and overall reliability is enhanced.

2. SUMMARY OF THE INVENTION

In accordance with one aspect of the present invention, a method of writing data onto a magnetic tape storage medium in the form of data blocks is provided in which data is received from a data source, the data received is formatted into data and ECC blocks, data entities are then created which include the data blocks and ECC blocks, data envelopes are then defined which include one or more of the data entities, and the data envelopes are written onto the magnetic

tape storage medium in sequential fashion. The data and ECC blocks within the data envelopes can be randomly placed and do not require sequential placement on the tape medium.

3. BRIEF DESCRIPTION OF DRAWINGS

Figure 1 is a block diagram of a host computer operably coupled to a tape drive by a system bus.

Figure 2 illustrates a sequence of data envelopes on a magnetic tape storage medium each including one or more data blocks.

Figure 3 illustrates block migration within a data envelope.

Figure 4 illustrates envelope closure.

4. DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

The following descriptions of illustrative embodiments of the invention are presented as lists of operations performed in illustrative methods, annotated with the explanatory notes below. The descriptions are derived from certain claims in the application as originally filed, but of course the claims are not intended and should not be deemed to be limited to the illustrative em-

bodiments so described. Bracketed numbers correspond to the notes below, while unbracketed reference numerals correspond to identifying numerals in the drawings.

It is to be understood that the particular implementations described are intended as illustrations of, and not as limiting the scope of, the claims. In the interest of clarity, not all the routine features of the implementations are described. It will of course be appreciated that in the development of any such actual implementation, numerous implementation-specific decisions must be made to achieve the developers' specific goals such as compliance with system- and business-related constraints and that these goals will vary from one implementation to another. Moreover, it will be appreciated that such a development effort might be complex and time-consuming, but would nevertheless be a routine undertaking of engineering for those of ordinary skill having the benefit of this disclosure.

Illustrative Method

Figures 1-4 show the context of an illustrative method [1] of writing data onto a magnetic tape storage medium 100 [2] using a host computer 105 [3] operably coupled to a tape drive 110 [4] by a system bus 115 [5]. The illustrative method generally comprises:

(a) receiving data from a data source [6];

(b) formatting the data into data blocks [7] and error correction code blocks, referred to as ECC blocks [8];

(c) creating at least one data entity, each data entity including data blocks and ECC blocks [9];

(d) defining at least one data envelope [10], each data envelope including at least one of the data entities [11]; and

(e) writing the data envelopes sequentially [12] onto the magnetic tape storage medium [13].

Referring to Figure 3, the illustrative method of writing the data envelopes sequentially onto the magnetic tape storage medium may provide error correction within individual data envelopes that includes:

(1) detecting [14] an error in a data or ECC block previously written on a specified channel; and

(2) rewriting the data or ECC block in error onto the next circularly adjacent channel [15].

Referring to Figure 4, the illustrative method of writing the data envelopes sequentially onto the magnetic tape storage medium may also provide error correction during envelope closure that includes:

(1) initiating the next data envelope by writing data and/or ECC blocks [16];

(2) detecting an error in a data or ECC block previously written on a specified channel in the previous data envelope [17];

(3) killing the blocks written in the next data envelope [18]; and

(4) rewriting the data or ECC blocks in error into the previous data envelope [19].

Notes to Illustrative Method

[1] The method advantageously increases the storage capacity of a magnetic tape or other similar storage medium, permitting more data to be stored on a given amount of tape. Generally speaking, it does so by reducing the storage space loss that conventionally arises from errors in encoding the data onto the tape. In addition, the method allows data recording to be continued even in the event of a localized media anomaly (e.g., contamination) or data-channel malfunction (due, e.g., to tape-head damage, problems with the read-write circuit

[2] In the illustrative embodiment, the magnetic tape storage medium is a four-channel magnetic tape 100 with channels 0, 1, 2, and 3. Of course any magnetic or nonmagnetic storage medium (e.g., optical tape) may used with any number of read-write channels providing that the medium is sequential with read-after-write error detection.

[3] In the illustrative embodiment, the host computer 105 includes a main memory 120, a CPU 125, and a disk drive 130. The CPU 125 may be a general purpose programmable

computer or any number of commercially available microprocessors as will be recognized by a person of ordinary skill in the art. The CPU 125 retrieves data from the main memory 120 and/or the disk drive 130 for transmission to the tape drive 110 over the system bus 115. The operation and mechanics of programming such a CPU 125 to send and receive such data over the system bus 115 are considered well known to a person of ordinary skill in the art.

[4] In the illustrative embodiment, the tape drive 110 includes a tape controller 135 and a conventional read-after-write head 140. The tape controller 135 may include a buffer 145, a microprocessor 150, and program storage 155. The microprocessor 150 may be programmed to perform the method of the illustrative embodiment using industry standard programming languages such as, for example, assembly language, pascal, fortran, and C++. The programming may be encoded in an otherwise conventional program storage device such as, for example, a PROM , EEPROM, or flash memories. The read-after-write head 140 is controlled by the tape drive 110 using conventional servo-loop control. More generally, the storage medium may be any sequential storage medium with read-after-write capability. Therefore the present method may be adapted to encompass such storage media as, for example, optical tape storage with the "tape drive" replaced with analogous optical components to accommodate such alternative storage media as will be apparent to one of ordinary skill in the art. In addition, the control provided by the tape drive 110 may be provided by software resident in a microprocessor resident in the tape drive 100 or it may be augmented or replaced by a programmed logic array or other similar

hardware implementation as will be apparent to one of ordinary skill in the art.

[5] In the illustrative embodiment, the system bus 115 may comprise any number of industry standard serial or parallel communication busses.

[6] In the illustrative embodiment, the data source will be the main memory 120, CPU 125, or disk drive 130. Other data sources may also be used as will be apparent to one of ordinary skill in the art.

Data arriving from the host computer 105 are broken down into substantially uniform fixed size data blocks by software resident in the microprocessor 150. In the illustrative embodiment, sixteen such sequentially received data blocks are grouped together and four ECC blocks are also generated. This generation is done prior to writing the any of the data blocks to tape. The resulting twenty blocks are called an entity. As illustrated in Figure 2, one entity containing sixteen data blocks (numbered 1-16) and four ECC blocks (numbered 17-20) are sequentially packaged into a first data envelope 205 while another entity containing sixteen data blocks (numbered 1-16) and four ECC blocks (numbered 17-20) are non-sequentially packaged into a second, and sequentially positioned, data envelope 210. Although the illustrative embodiments only show a single entity in the envelopes, an envelope may contain any number of entities, but is governed by a practical implementation-wide maximum. Entities always exist

within a single envelope and are not split or segmented between envelopes. The data envelopes 205 and 210 may contain any number of entities with the blocks of the entities contained within the data envelopes freely positioned within the data envelope (i.e. sequential ordering of the blocks is not required and the blocks of the plurality of entities within a specific data envelope may be freely intermixed).

A significant advantage of the present method is that the ECC blocks are created prior to writing the data blocks onto the tape 100. This has the effect of permitting the data blocks and ECC blocks to be written to the tape media 100 in an essentially arbitrary order. This represents a considerable improvement over prior techniques in which the ECC blocks were required to be written to the tape media after the data blocks were written. In addition, the prior techniques required the data blocks and the ECC blocks to be written sequentially and in the correct order. The prior techniques requirement of sequentiality and ordering led to data capacity loss when block errors occurred because blocks that had originally been written sequentially were required to be rewritten sequentially in the original order. The present method, by contrast, permits the writing of blocks in any order.

[7] One format for the data blocks is an adaptation of that used in existing products such as, for example, the DLT-2000, manufactured by the Quantum Corporation and further described in the ECMA Standard No. 209, published in December of 1994. Other types of data

block formatting, which is driven by the particular platform selected, may also be utilized as will be recognized by persons of ordinary skill in the art.

[8] One format for the ECC blocks is disclosed in United States Patent No. 5,107,506, issued on April 21, 1992, to Weng et al., the disclosure of which is incorporated herein by reference. Other types of ECC formatting may also be utilized as will be recognized by persons of ordinary skill in the art.

[9] In the illustrative embodiment, sixteen data blocks and four ECC blocks are used in each entity. More generally, any number and combination of data and ECC blocks though may be utilized though depending upon the specific error correction algorithm selected.

[10] The blocks within an envelope are tagged with control fields identifying the particular data envelope in which they are contained. In this manner, the data envelopes may be sequentially ordered while the blocks within may be ordered in any fashion desired (e.g., sequential, random, ordered, chaotic, or some other pseudo-ordered arrangement).

[11] The data envelopes may include from one up to any number of entities. In the illustrative embodiment, the data entities include data blocks and ECC blocks of substantially uniform size. More generally, it is possible for the data entities to not include ECC blocks and also

to utilize blocks of non-uniform size.

[12] The data envelopes are written onto the magnetic tape media in sequential order while the blocks within the data envelopes may be in any order. Blocks from any of the entities within a defined envelope may be written to tape 100 in any order and may be freely repeated within the envelope.

An error of any block detected by the read-after-write circuitry will be migrated onto the next circularly adjacent channel. All blocks within a defined envelope are completely written to media (error free) before the next envelope is started. The boundary between consecutive envelopes is called the envelope closure. During envelope closure, errors detected will result in any blocks currently being written to tape 100 from the next envelope to be killed ("k-bitted") so that inter-envelope sequentiality can be maintained.

The write logic is responsible for enforcing the block topology as well as the rules governing envelope closure. It is also responsible for making heuristic decisions regarding block migration and error management. Read operations are responsible for reordering the blocks within a specified envelope prior to transmitting them back to the host computer 105.

[13] The storage medium in the illustrative embodiment is a four-channel magnetic

tape 100 with read-write channels 0, 1, 2, and 3. The term "writing onto" a magnetic tape will of course be recognized to mean that certain particulate matter on the flexible tape substrate is selectively magnetized.

[14] Detecting an error in a previously written block is provided by the conventional operation of read-after-write. In the illustrative embodiment illustrated in Figure 3, the data envelope 305 includes block 5* with errors. The error in block 5* is detected during the subsequent conventional read-after-write operation.

[15] Rewriting the block 5* to the next circularly adjacent channel means writing the block again to another adjacent channel. In the illustrative embodiment, the block 5* was initially written to channel 0 and is then subsequently rewritten to the circularly adjacent channel 1. Note that the final block number 20 of the data envelope 305 is written onto all four channels to finish out the data envelope 305. This is not required, but for a four-channel read-write head, as a matter of convenience, it is practical to write any remaining data to all four channel locations to finish out the data envelope.

[16] Figure 4 illustrates the error correction during envelope closure. A first data envelope 305 including blocks 1-20 are initially written onto the magnetic tape medium 100. The second data envelope 310 is then begun by writing blocks 1-4.

[17] During the writing of blocks 1-4 of the second data envelope 310, the conventional read-after-write operation detects an error in block 19*.

[18] The initial blocks numbered 1-4 of the second data envelope 310 are then killed by being "k-bitted." K-bitting a block entails using a control bit in a control field which marks all of the blocks as bad thereby indicating subsequent read operations to ignore them. The "killing" of the initial blocks of the second data envelope 310 is required because, while blocks within the data envelopes may be in any order, the blocks within the second data envelope 310 must be sequential to the blocks within the first data envelope 305. Consequently, the first data envelope 305 must be completed before the second data envelope 310 can begin.

[19] The first data envelope 305 is then completed by writing the bad block 19* again. Note that for the case of error correction during envelope closure, the rewriting can be to any channel (i.e., channel circularity is not required). Once the first data envelope 305 is completed by rewriting the block 19, the second data envelope 310 can then begin again by writing the blocks 1-4.

* * *

The illustrative method and its numerous variations offers several significant advantages over the prior techniques employed for block topology:

(1) Errors detected within an envelope result in the block in error being rewritten, but no other blocks are killed or rewritten. Thus the illustrative method is extremely capacity-efficient.

(2) Errors detected at the envelope boundary result in some capacity loss, but these errors encompass only a small percentage of all errors ordinarily encountered in practice.

(3) Block migration allows continued writing of the magnetic tape storage medium during temporary read-write channel problems or contaminations of localized media perturbations. Thus the overall likelihood of a fatal write error is dramatically reduced.

(4) The architecture is essentially independent of the number of channels and can be implemented to be easily scalable onto future products.

(5) The architecture encapsulates all knowledge and maintenance of envelopes and their associated control so that only a single software module has responsibility for the block topology. The higher layers of the software - the bus handlers, servers, and the host programs - need have no inherent knowledge of the block topology and thus are isolated from it. The entire scheme is thus highly transportable and scales onto future designs.

* * *

It will be appreciated by those of ordinary skill having the benefit of this disclosure that numerous variations from the illustrations in the notes will be possible without departing from the inventive concept described herein.



5. WHAT IS CLAIMED IS:

1. A method of writing data onto a magnetic tape storage medium including one or more channels, comprising:

(a) receiving said data from a data source;

(b) formatting said data into data blocks and error correction code blocks;

(c) creating data entities, each data entity including data blocks and error correction code blocks;

(d) defining data envelopes, each said data envelope including at least one of said data entities; and

(e) writing said data envelopes sequentially onto said magnetic tape storage medium.

2. The method of claim 1, wherein said data blocks and said error correction code blocks are arranged within said data envelopes in sequential order.

3. The method of claim 1, wherein said data blocks and said error correction code blocks are arranged within said data envelopes in nonsequential order.

4. The method of claim 1, wherein said data blocks and said error correction code blocks are of substantially uniform size.

5. The method of claim 1, wherein said data blocks and said error correction code blocks are nonuniform in size.

6. The method of claim 1, wherein writing said data envelopes sequentially onto said magnetic tape storage medium includes:

(1) detecting an error in a data block or in an error correction code block previously written onto a specified channel of said magnetic tape storage medium; and

(2) rewriting said data block or error correction code block in error onto a circularly adjacent channel of the magnetic tape storage medium.

7. The method of claim 1, wherein writing said data envelopes sequentially onto said magnetic tape storage medium comprises:



(1) initiating a next data envelope on said magnetic tape storage medium by writing data and/or error correction code blocks onto said magnetic tape storage medium;

(2) detecting an error in a data block or in an error correction code block previously written on a specified channel in a previous data envelope;

(3) killing said data and/or error correction code blocks written in said next data envelope; and

(4) rewriting said data or error correction code blocks in error into said previous data envelope.

8. The method of claim 7, wherein killing said blocks comprises k-bitting said blocks.

10. A method of writing data onto a storage medium including one or more channels, comprising:

(a) receiving said data from a data source;

(b) formatting said data into blocks;

(c) creating at least one data entity, each data entity including one or more of said blocks;



(d) defining data envelopes, each said data envelope including at least one of said data entities; and

(e) writing said data envelopes onto said storage medium.

11. The method of claim 10, wherein said blocks include data blocks and error correction code blocks.

12. The method of claim 10, wherein said blocks are substantially uniform in size.

13. The method of claim 10, wherein said blocks are nonuniform in size.

14. The method of claim 10, wherein writing said data envelopes onto said storage medium comprises writing said data envelopes sequentially onto said storage medium.



15. The method of claim 10, wherein said storage medium comprises a magnetic tape storage medium.

16. The method of claim 10, wherein writing said data envelopes onto said storage medium includes:

(1) detecting an error in a block previously written onto a specified channel of the storage medium; and

(2) rewriting said block in error to a circularly adjacent channel of said storage medium.

17. The method of claim 10, wherein writing said envelopes onto said storage medium comprises:

(1) initiating a next data envelope on said storage medium by writing blocks onto said storage medium;

(2) detecting an error in a block previously written on a specified channel in a previous data envelope;

(3) killing said blocks initially written in said next data envelope; and

(4) rewriting said block in error into said previous data envelope.



18. The method of claim 17, wherein killing said blocks comprises k-bitting said blocks.

20. An apparatus for writing data onto a storage medium including one or more channels in the form of blocks, comprising:

(a) a read-after-write recording device adapted to read and write sequentially onto said storage medium; and

(b) a controller operably coupled to said read-after-write recording device, said controller adapted to:

receive data from a data source;

format said data into blocks;

create at least one data entity, each data entity including at least one or more of said blocks;

define data envelopes, each said data envelope including at least one of said data entities; and

direct said read-after-write recording device to write said data envelopes sequentially onto said storage medium.



21. The apparatus of claim 20, wherein said blocks include data blocks and error correction code blocks.

22. The apparatus of claim 20, wherein said blocks are substantially uniform in size.

23. The apparatus of claim 20, wherein said blocks are nonuniform in size.

24. The apparatus of claim 20, wherein said controller is adapted to direct said read-after-write recording device to write said data envelopes onto said storage medium sequentially.

25 The apparatus of claim 20, wherein said read-after-write recording device comprises a magnetic tape read-after-write recording head.

26. The apparatus of claim 20, wherein said controller is adapted to:

* C L M S *

(1) detect an error in a block previously written onto a specified channel of the storage medium; and

(2) direct said read-after-write recording device to rewrite said block in error to a circularly adjacent channel of said storage medium.

27. The apparatus of claim 20, wherein said controller is adapted to:

(1) initiate a next data envelope on said storage medium by directing said read-after-write recording device to write blocks onto said storage medium;

(2) detect an error in a block previously written on a specified channel in a previous data envelope;

(3) kill said blocks initially written in said next data envelope; and

(4) direct said read-after-write recording device to rewrite said block in error into said previous data envelope.

28. The apparatus of claim 27, wherein said controller is adapted to kill said blocks by k-bitting said blocks.

29. The apparatus of claim 20, wherein said controller comprises a programmed microprocessor.

30. The apparatus of claim 20, wherein said controller comprises a logic array.