(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5302886号
(P5302886)

(45)発行日　平成25年10月2日(2013.10.2)　(24)登録日　平成25年6月28日(2013.6.28)

(51)Int.Cl.　　F I
G06F 12/00 (2006.01)　　G06F 12/00 520P
G06F 3/06 (2006.01)　　G06F 3/06 301G
　　G06F 12/00 520E

請求項の数 10　(全 26 頁)

(21)出願番号　特願2009-518192 (P2009-518192)
(86)(22)出願日　平成19年6月25日 (2007.6.25)
(65)公表番号　特表2009-543198 (P2009-543198A)
(43)公表日　平成21年12月3日 (2009.12.3)
(86)国際出願番号　PCT/US2007/014664
(87)国際公開番号　WO2008/005211
(87)国際公開日　平成20年1月10日 (2008.1.10)
審査請求日　平成21年6月16日 (2009.6.16)
(31)優先権主張番号　11/477,886
(32)優先日　平成18年6月29日 (2006.6.29)
(33)優先権主張国　米国(US)

前置審査

(73)特許権者　303039534
ネットアップ，インコーポレイテッド
アメリカ合衆国　カリフォルニア　９４０８９，　サニーヴェール，　イースト　ジャバ　ドライブ　４９５
(74)代理人　100087642
弁理士　古谷　聡
(74)代理人　100076680
弁理士　溝部　孝彦
(74)代理人　100121061
弁理士　西山　清春
(72)発明者　ツェン，リン
アメリカ合衆国カリフォルニア州９４０８９，サニーヴェール，イースト・ジャバ・ドライブ・４９５

最終頁に続く

(54)【発明の名称】ブロック指紋を読み出し、ブロック指紋を使用してデータ重複を解消するシステム、及び方法

(57)【特許請求の範囲】
【請求項１】
ストレージシステムにおいて重複データを識別するために、コンピュータで実施される方法であって、
複数のデータブロックにバイト単位でデータを記憶するステップと、
各チェックサム値が各データブロックについて計算される、複数のチェックサム値を計算するステップと、
前記データブロック内の所定のオフセットからデータをサンプリングするステップと、
各チェックサム値及び前記サンプリングされたデータを複数の指紋のうちの１つの指紋に記憶するステップであって、各指紋が、前記複数のデータブロックのうちの１つのデータブロックに関連し、前記データブロックについて計算されたチェックサム値、及び前記データブロックからのサンプリングされたデータを含む、各チェックサム値及び前記サンプリングされたデータを複数の指紋のうちの１つの指紋に記憶するステップと、
（１）前記複数の指紋のうちの同一の指紋を識別し、（２）前記同一の指紋に関連する前記識別された重複データブロックを比較し、前記識別された重複データブロックが実際に重複していることを確認することにより、前記複数のデータブロックのうちの重複データブロックを識別するステップであって、前記比較は、前記識別された重複データブロックのバイトごとの比較を実施することを含む、前記複数のデータブロックのうちの重複データブロックを識別するステップと

を含む、方法。
【請求項２】
前記複数のデータブロックのうちの１つに記憶されたデータを変更するステップと、
前記所定のオフセットからサンプリングされたデータが、前記複数のデータブロックのうちの１つの中の前記所定のオフセットに記憶された前記変更されたデータとは異なることに応答して、前記複数の指紋のうちの１つを削除するステップと
をさらに含む、請求項１に記載のコンピュータで実施される方法。
【請求項３】
前記複数のデータブロックのうちの１つに関連するメタデータを記憶するステップと、
前記メタデータを前記複数のデータブロックのうちの１つに関連する指紋レコードにコピーするステップと、
前記複数のデータブロックのうちの１つに関連する指紋を前記指紋レコードに記憶するステップと
をさらに含む、請求項１または請求項２に記載のコンピュータで実施される方法。
【請求項４】
前記メタデータは、生成番号コピーとして前記指紋レコードにコピーされたインデックス・ノード（ｉｎｏｄｅ）の生成番号を含み、前記方法は、
前記ｉｎｏｄｅを削除するステップと、
書き込みオペレーションに応答して、前記ｉｎｏｄｅを再割り当てするステップと、
前記生成番号を変更するステップと、
前記生成番号コピーが、変更された生成番号とは異なることに応答して、前記指紋レコードを削除するステップと
をさらに含む、請求項３に記載のコンピュータで実施される方法。
【請求項５】
前記メタデータは、ＣＰカウントコピーとして前記指紋レコードにコピーされたコンシステンシ・ポイント（ＣＰ）を含み、前記方法は、
書き込みオペレーションに応答して、前記ＣＰカウントを変更するステップと、
前記ＣＰカウントコピーが、変更されたＣＰカウントとは異なることに応答して、前記指紋レコードを削除するステップと
をさらに含む、請求項３または請求項４に記載のコンピュータで実施される方法。
【請求項６】
前記メタデータは、データコンテナ内のデータブロックの数に関連するサイズ値を含み、さらに、ｆｂｎコピーとして前記指紋レコードにコピーされたファイルブロック番号（ｆｂｎ）を含み、前記方法は、
前記データコンテナ内のデータブロックの数の変化に応答して、前記サイズ値を変更するステップと、
前記ｆｂｎコピーが、変更されたサイズ値に関連するデータブロックの数よりも大きなオフセットを示していることに応答して、前記指紋レコードを削除するステップと
をさらに含む、請求項３～５のいずれか一項に記載のコンピュータで実施される方法。
【請求項７】
ストレージオペレーティングシステムを実行するプロセッサを含むストレージシステムであって、前記ストレージオペレーティングシステムが、
ファイルシステム層と、
前記ファイルシステム層と協働し、書き込みオペレーションを処理するストレージモジュール層であって、バイト単位でデータブロックを記憶し、前記データブロックについてチェックサム値を計算するようにさらに構成されたストレージモジュール層と、
前記データブロックの１以上のオフセットにおけるデータをサンプリングし、前記データブロックについての前記チェックサム値及び前記データブロックからの前記サンプリングされたデータを含む指紋を生成し、前記指紋を指紋データベースに記憶するように構成された指紋抽出層であって、前記指紋データベースが、複数の指紋を含み、各指紋が、

複数のデータブロックのうちの１つのデータブロックに関連する、指紋抽出層と、
（１）前記指紋データベース中の同一の指紋を識別し、（２）前記同一の指紋に関連する前記識別された重複データブロックを比較し、前記識別された重複データブロックが実際に重複していることを確認することにより、重複データブロックを識別するように構成された重複解消モジュールであって、前記比較は、前記識別された重複データブロックのバイトごとの比較を実施することを含む、重複解消モジュールと
を含む、ストレージシステム。
【請求項８】
前記書き込みオペレーションに応答して、前記複数の指紋のうちの少なくとも１つの指紋を記憶するように構成された指紋キューをさらに含み、前記重複解消モジュールは、前記指紋キュー、及び前記指紋データベースの中から同一の指紋を識別することにより、重複データブロックを識別するようにさらに構成される、請求項７に記載のストレージシステム。
【請求項９】
前記ストレージモジュール層は、ＲＡＩＤ（Redundant Array of Inexpensive Disks）プロトコルを実施するように構成され、前記チェックサム値は、前記ＲＡＩＤプロトコルに従って計算される、請求項７または請求項８に記載のストレージシステム。
【請求項１０】
前記チェックサム値は、３２ビットＲＡＩＤ（Redundant Array of Inexpensive Disks）チェックサム値である、請求項１～６のいずれか一項に記載のコンピュータで実施される方法。
【発明の詳細な説明】
【技術分野】
【０００１】
[発明の分野]
本発明は、ストレージシステムに関し、特に、ストレージシステムにおいて重複データを効率的に減らす技術に関する。
【背景技術】
【０００２】
[発明の背景]
ストレージシステムは一般に１以上の記憶装置を備え、要望に応じてその中に情報を入れたり、そこから情報を取り出したりすることができる。ストレージシステムは、ストレージオペレーティングシステムを含み、ストレージオペレーティングシステムは、とりわけ、システムによって実施されるストレージサービスを支える種々のストレージオペレーションを実施することにより、システムを機能的に編成する。ストレージシステムは、種々のストレージアーキテクチャに従って実施することができ、限定はしないが、例えば、ネットワーク・アタッチド・ストレージ環境、ストレージ・エリア・ネットワーク、あるいは、クライアントコンピュータ、又はホストコンピュータに直接取り付けられたディスクアセンブリとして実施される場合がある。記憶装置は通常、ディスクアレイとして編成され、ストレージプロトコルに従って管理される幾つかのディスクドライブである。ここで、「ディスク」という用語は一般に、内蔵型の回転式磁気媒体記憶装置を意味する。この文脈におけるディスクという用語は、ハード・ディスク・ドライブ（ＨＤＤ）やダイレクト・アクセス・ストレージ・デバイス（ＤＡＳＤ）と同じような意味である。
【０００３】
ディスクアレイへの情報の記憶は、好ましくは、ディスク空間の全体的論理構成を定義する、物理ディスクの１以上のストレージ「ボリューム」として実施される。ボリューム内のディスクは通常、１以上のグループに編成され、各グループが、ＲＡＩＤ（Redundant Array of Independent or Inexpensive Disks）として運用される。大半のＲＡＩＤ実施形態は、ＲＡＩＤグループ内の所与の数の物理ディスクにわたってデータ「ストライプ」を冗長書き込みし、ストライピングされたデータに関する冗長情報（パリティ）を適切

に記憶することにより、データ記憶の信頼性／完全性を向上させる。各ＲＡＩＤグループの物理ディスクには、ストライピングされたデータを記憶するように構成されたディスク（すなわち、データディスク）と、データのパリティを記憶するように構成されたディスク（すなわち、パリティディスク）とがある。その後、ディスクが故障したときに、失われたデータの復旧を可能にするために、パリティは読みされる場合がある。「ＲＡＩＤ」という用語、及びその種々の実施形態は周知であり、１９９８年６月、データ管理に関する国際会議（ＳＩＧＭＯＤ）の議事録において、D.A. Patterson、G.A. Gibson、及びR.H. Katzにより、「A Case for Redundant Arrays of Inexpensive Disks (RAID)」と題して開示されている。

【０００４】

ストレージシステムのストレージオペレーティングシステムは、ファイルシステムのような上位モジュールを実施し、情報を入れるためのデータコンテナを論理編成する場合がある。例えば、情報は、ディレクトリ、ファイル、及びブロックの階層構造としてディスク上に記憶される場合がある。例えば、「ディスク上」の各ファイルは、ファイルの実際のデータのような情報を記憶するように構成された一組のデータ構造、すなわちディスクブロックとして実施される場合がある。これらのデータブロックは、ファイルシステムによって管理されるボリュームブロック番号（ｖｂｎ）空間の中に編成される。ファイルシステムは更に、ファイル内の各データブロックに、対応する「ファイルオフセット」、又はファイルブロック番号（ｆｂｎ）を割り当てる場合がある。ファイルシステムは通常、ファイルごとに連続したｆｂｎを割り当てるのに対し、ｖｂｎは、より大きなボリュームアドレス空間に割り当てられる。ファイルシステムは、ｖｂｎ空間内において幾つかのデータブロックを「論理ボリューム」として編成する。そして、もちろん必須ではないが、各論理ボリュームは、その論理ボリューム独自のファイルシステムに関連付けられる場合がある。ファイルシステムのサイズが、ｎ＋１ブロックである場合、ファイルシステムは通常、ゼロからｎまでの連続した範囲のｖｂｎから構成される。

【０００５】

一つの既知のタイプのファイルシステムは、ディスク上でデータを上書きしないｗｒｉｔｅ－ａｎｙｗｈｅｒｅファイルシステムである。ディスクからストレージシステムのメモリ内へデータブロックを取り出し（読み出し）、新たなデータで「汚す」（すなわち、更新、又は変更）するとき、書き込み性能を最適化するために、以後、そのデータブロックは、ディスク上の新たな場所に記憶（書き込み）される。ｗｒｉｔｅ－ａｎｙｗｈｅｒｅファイルシステムは、データがディスク上で実質的に連続的に配置されるような最適レイアウトを最初に仮定する場合がある。この最適なディスクレイアウトにより、ディスクに対する効率的なアクセスオペレーションが可能となり、特に、シーケンシャル読み出しオペレーションの場合に効率的なアクセスオペレーションが可能となる。ストレージシステム上で動作するように構成されたｗｒｉｔｅ－ａｎｙｗｈｅｒｅファイルシステムの一例は、カリフォルニア州サニーベイルにあるネットワーク アプライアンス，インコーポレイテッドから販売されているＷｒｉｔｅ Ａｎｙｗｈｅｒｅ Ｆｉｌｅ Ｌａｙｏｕｔ（ＷＡＦＬ）ファイルシステムである。

【０００６】

大きなファイルシステムでは、個々のデータブロックの重複的発生を見付けることが一般的である。データブロックの重複が発生するのは、例えば、２以上のファイルその他のデータコンテナが、共通のデータを共有する場合や、あるいは、所与のファイル内の複数の場所に、所与のデータセットが発生する場合である。データブロックの重複の結果、ストレージシステムによって提供される複数の異なる場所に同一のデータが記憶されることにより、記憶空間は、十分に利用されないことになる。

【０００７】

この問題を解決するために使用されている一つの技術として、「ファイル・フォールディング」と呼ばれるものがある。ファイル・フォールディングの基本原理は、アクティブファイルシステム上のファイルの新たなデータが、永久的イメージ中のそのファイルの古

いデータと同一のものである場合、新たなデータと古いデータとの間におけるディスクブロックの共有を許可するというものである。この技術は、カリフォルニア州、サニーベイルにあるネットワーク　アプライアンス，インコーポレイテッドから市販されている種々のストレージシステムにおいて既に実施されている。具体的には、それらのストレージシステムは、指定されたデータセットの空間保守的な、コンシステンシ・ポイント・イメージ（ＰＣＰＩ）を生成する機能を有する。ＰＣＰＩとは、名前によるアクセスが可能な、空間保守的な、何らかの過去の時点におけるデータ（例えばストレージシステム）の読み出し専用イメージであり、何らかの過去の時点におけるそのデータの一貫性のあるイメージを提供する。具体的には、ＰＣＰＩとは、記憶装置（例えばディスク）その他の永久的メモリに記憶されたアクティブファイルシステム、ファイル、又はデータベースのような記憶要素をある時点において表現した物であり、自分自身を他の時点において作成されたＰＣＰＩから区別するための名前その他の識別子を有する。ＰＣＰＩの一例は、ネットワーク　アプライアンス，インコーポレイテッドから市販されているストレージシステムにおいて実施されるようなＳｎａｐｓｈｏｔである。「ＰＣＰＩ」という用語と「Ｓｎａｐｓｈｏｔ」という用語は、ネットワークアプライアンスの商標権を失墜させることなく、交換可能に使用される。

【０００８】

ＰＣＰＩを生成する技術については、１９９８年１０月６日に発行された「Method For Maintaining Consistent States of a File System and for Creating User-Accessible Read-Only Copies of a File System」と題するDavid Hitz他による本件と同じ譲受人の米国特許第５，８１９，２９２号に詳しく記載されている。例えば、「スナップショット化」されたデータコンテナの中のブロックが変更された場合、ストレージシステムは、そのデータコンテナのもう一つの完全なコピー（変更後のもの）を作成する代わりに、単に、そのデータコンテナの変更されたブロックをアクティブファイルシステム上に生成する。未変更のブロックについては、ストレージシステムは単に、ＰＣＰＩ中の対応するブロックへのポインタをそのデータコンテナに与えるだけである。このようにして、ＰＣＰＩ中の未変更のブロックは、ＰＣＰＩとアクティブファイルシステムとの間で共有されることになる。この技術については、２００２年３月２２日に出願された「File Folding Technique」と題する本件と同じ譲受人の米国特許出願第２００３／０１８２３１７号に詳しく記載されている。

【０００９】

ファイルフォールディングは、記憶空間の利用効率を向上させるためには役に立たない。しかしながら、ＳｎａｐｓｈｏｔのようなＰＣＰＩに頼ることなく、アクティブファイルシステムにおけるデータ重複を減らすことが望ましい。また、重複ファイル、又は単一ファイル内の重複データによりアクティブファイルシステム上に発生することがある重複データブロックを識別し、減らすことが望まれている。より一般的には、ストレージシステムにおいて、のデータの位置とは無関係にデータ重複を減らすことが望まれている。

【００１０】

データ重複の低減を実現する他の技術は、１９９９年１１月２３日に発行された「Partitioning a Block of Data into Blocks and for Storing and Communicating Such Subblocks」と題するRoss Williamsによる米国特許第５，９９０，８１０（以下、‘８１０特許）に記載されている。’８１０特許に記載の方法は、まず、ローリングハッシュ関数を使用して、複数のサブブロックのデータを生成する。ローリングハッシュ関数は、固定サイズウィンドウのデータを使用し、２つのサブブロック間に位置する境界を生成する。一つのデータブロックを複数のサブブロックに分割した後、各サブブロックのハッシュ値が計算され、ハッシュ値のテーブルが形成される。次に、このハッシュテーブルを使用して、新たなサブブロックが、ハッシュテーブルに既に記憶されたハッシュ値を有するいずれかのサブブロックと同一であるか否かが判定される。この判定を実施するために、新たなサブブロックのハッシュ値が計算され、それが、ハッシュテーブルに含まれるハッシュ値と比較される。もし、新たなサブブロックのハッシュ値が、ハッシュテーブルに既に記憶

されていた場合、記憶されたハッシュ値を有するものとして識別されたサブブロックは、新たなサブブロックと同一であるとみなされる。その場合、新たなサブブロックは、既に記憶されているサブブロックへのポインタで置き換えられ、それによって、サブブロックに必要とされる記憶空間の量が低減される。‘８１０特許に記載されている技術の大きな欠点は、計算上、膨大な数の集中的なハッシュ計算の処理能力が要求される点にあり、そのような計算は、そのような方法を実施するストレージシステムの全体的処理能力に悪影響を与えることがある。もう一つの大きな欠点は、データセットのサイズが増大するのにつれて、ハッシュテーブルも大きくなるため、テラバイトやペタバイトのデータのような大きなデータセットには応用できないことがある点である。

【００１１】

重複データを無くすさらに別の技術は、２００５年４月１３日に出願された「Method and Apparatus for Identifying and Eliminating Duplicate Datablocks and Sharing Data Blocks in a Storage System」と題するLing Zheng他による本件と同じ譲受人の米国特許出願第１１／１０５，８９５に記載されており、この文献の内容は、参照により、本明細書に援用される。この特許出願に記載されているシステムでは、全てのデータ重複解消オペレーションが、例えば４キロバイト（ＫＢ）サイズといった固定サイズのブロックに対して実施される。新たなブロックを記憶する場合、ハッシュ値は、４ＫＢブロックの指紋として計算される。次に、この指紋は、以前に記憶された種々のブロックの指紋を含むハッシュテーブルと比較される。もし、新たなブロックの指紋が、以前に記憶されたブロックのものと同一であった場合、その新たなブロックは、以前に記憶されたブロックと同一である可能性が高い。その場合、新たなブロックは、以前に記憶されたブロックへのポインタで置き換えられ、それによって、ストレージリソースの消費が低減される。

【００１２】

しかしながら、このシステムの欠点は、新たな４ＫＢブロックを記憶するときにハッシュ計算が必要となる点にある。指紋やデータブロックのサイズによっては、そのような計算は、処理リソースをさらに必要とする場合がある。そして、それらのサイズは、さらに、ストレージシステムのサイズによって決まる場合がある。また、可能性は非常に低いが、同一の複数の指紋が、重複データを意味しないこともありうる。例えば、重複解消プロセスにおいて、同一の指紋を有するデータブロックは、それらが実際に同一であることを確認するために、さらに比較される場合がある。従って、その後のデータ重複解消プロセスにおけるデータブロックの比較を容易にするためには、同一の指紋が重複データを意味するであろうという適度な確率を維持しながらも、より少ない処理リソースを使用して指紋を生成できることが望ましい。さらに、ＮｅｔＡｐｐ　Ｄａｔａ　ＯＮＴＡＰオペレーティングシステムのような現代のストレージシステムの中には、システムに記憶されるデータブロックの完全性を確保するために、既にチェックサムオペレーションを実施しているものもある。既存のストレージ機能を利用して処理リソースを節約する重複解消プロセスが、望まれている。

【発明の概要】

【課題を解決するための手段】

【００１３】

[発明の概要]

本発明は、ストレージシステムにおける重複データブロックを効率的に識別できるようにブロック指紋を最適化することにより、従来技術の欠点を克服する。識別された重複データブロックは、次に、データ重複解消モジュールによって比較され、ストレージシステムの処理能力を最適化するために、重複データは削除される。

【００１４】

本発明の一実施形態によれば、ストレージシステムは、ファイルシステムのような高レベルモジュールを実施し、ストレージモジュールによって管理される複数のディスクのような記憶装置に記憶される情報を論理編成する。上位モジュールとストレージモジュールの間の境界において、指紋抽出層は、ストレージシステムにより記憶される各データブロ

ックについて、例えば６４ビットのような所定サイズの指紋を生成する。例えば、指紋の最初の所定数のビットは、ストレージモジュールによって実施されるチェックサム計算の結果から得られ、指紋の次の所定数のビットは、データブロック中の１以上の所定のオフセットに記憶されているデータから得られる。あるいは、ストレージモジュールと重複解消モジュールは、同じ６４ビットチェックサム値を使用して、異なる目的に従事する場合がある。得られた指紋は、データブロックの中身を十分に反映し、望ましくない誤認識率を生じることなく、重複の識別を可能にする。また、指紋の生成のために、ハッシュ関数のような他の暗号化関数をさらに実施する必要はないため、プロセッサ使用率も低減される。

【００１５】

各指紋は、指紋レコードに記憶され、指紋レコードはさらに、指紋データベースに記憶され、データ重複解消モジュールによってアクセスされる。データ重複解消モジュールは、指紋を定期的に比較し、重複する指紋を識別し、さらに重複データブロックを識別する場合がある。

【００１６】

例示的実施形態によれば、各指紋レコードは、指紋が付けられたデータブロックに関する付加情報を有する。データ重複解消モジュールは、この付加情報をデータブロックに関連するメタデータと定期的に比較し、指紋データベースが、ストレージシステムの最新の状態を正確に反映するようにする。従って、「古くなった」指紋（例えば、消去されたデータブロックや変更されたデータブロックに関連する指紋）は、指紋データベースから削除される場合がある。あるいは、古くなった指紋を削除するプロセスは、指紋データベースが所定のサイズに達するといった、所定の条件に応じて実施され、又は、古くなった指紋を削除するプロセスは、指紋を比較する上記のプロセスの直前に実施される場合がある。

【図面の簡単な説明】

【００１７】

【図１】本発明の一実施形態によるストレージシステム環境を示す略ブロック図である。

【図２】本発明の一実施形態によるストレージオペレーティングシステムを示す略ブロック図である。

【図３】本発明の一実施形態によるｉｎｏｄｅを示す略ブロック図である。

【図４】本発明の一実施形態によるデータコンテナのバッファツリーを示す略ブロック図である。

【図５】本発明の一実施形態によるデータブロックへのポインタを有する間接ブロックを示す略ブロックである。

【図６】本発明の一実施形態による変更されたデータブロックへのポインタを有する間接ブロックを示す略ブロック図である。

【図７】本発明の一実施形態による重複するデータブロックへのポインタを有する間接ブロックを示す略ブロック図である。

【図８】本発明の一実施形態による指紋を示す略ブロック図である。

【図９】本発明の一実施形態による指紋レコードを示す略ブロック図である。

【図１０】本発明の一実施形態による、指紋レコードを生成する手順のステップの詳細を示すフロー図である。

【図１１】本発明の一実施形態による、重複データブロックを識別する手順のステップの詳細を示すフロー図である。

【図１２Ａ】本発明の一実施形態による重複データブロックの識別を示す指紋キュー、及び指紋データベースを示す略ブロック図である。

【図１２Ｂ】本発明の一実施形態による重複データブロックの識別を示す指紋キュー、及び指紋データベースを示す略ブロック図である。

【図１２Ｃ】本発明の一実施形態による重複データブロックの識別を示す指紋キュー、及び指紋データベースを示す略ブロック図である。

【図１２Ｄ】本発明の一実施形態による重複データブロックの識別を示す指紋キュー、及び指紋データベースを示す略ブロック図である。
【図１３】本発明の一実施形態による、古くなった指紋レコードを削除する手順のステップの詳細を示すフロー図である。
【発明を実施するための形態】
【００１８】
[例示的実施形態の詳細な説明]
Ａ．ストレージシステム環境
図１は、本発明とともに有利に使用されるストレージシステム１２０を含む例示的ストレージシステム環境１００を示す略ブロック図である。ストレージシステムは、ディスクアレイ１６０のディスク１３０のような記憶装置上での情報の編成に関連するストレージサービスを提供するコンピュータである。ストレージシステム１２０は、システムバス１２５によって相互接続されたプロセッサ１２２、メモリ１２４、ネットワークアダプタ１２６、不揮発性ランダムアクセスメモリ（ＮＶＲＡＭ）１２９、及びストレージアダプタ１２８を含む。ストレージシステム１２０はさらに、ストレージオペレーティングシステム２００を含み、ストレージオペレーティングシステム２００は、好ましくは、ファイルシステムのような上位モジュールを実施し、情報をディレクトリ、ファイル、及び仮想ディスク（以下、「ブロック」）と呼ばれる特殊なタイプのファイルの階層構造として論理編成する。
【００１９】
例示的実施形態として、メモリ１２４は、ソフトウェアプログラムコードを記憶するために、プロセッサ１２２、及びアダプタによってアドレス指定可能な多数の記憶場所を有する。メモリの一部は、本発明に関連する指紋キューのようなデータ構造を記憶するために、「バッファキャッシュ」１７０としてさらに編成される場合がある。あるいは、ＮＶＲＡＭ１２９は、本発明に関連する指紋キュー、指紋データベース、その他のデータ構造を記憶するために使用される場合がある。さらに、システムの永久的コンシステンシ・ポイント・イメージ（ＰＣＰＩ）をサポートするシステムでは、ＮＶＲＡＭ１２９を使用して、コンシステンシ・ポイント間においてファイルシステムに対して施された変更を記憶する場合がある。そのような変更は、コンシステンシ・ポイントの進行中に不揮発性ログ（ＮＶＬＯＧ）１３１に記憶され、すなわち、ディスクにフラッシュ（書き込み）される場合がある。さらに、プロセッサ１２２、及びアダプタは、ソフトウェアコードを実行し、データ構造を操作するように構成された処理要素、及び／又は論理回路を含む場合がある。ストレージオペレーティングシステム２００は、その一部が通常、メモリに常駐し、処理要素によって実行され、とりわけ、ストレージシステムによって実施されるストレージオペレーションを実行することにより、システム１２０を機能的に編成する。当業者には明らかなように、本明細書に記載する本発明の技術に関連するプログラム命令の記憶、及び実行には、他の処理手段や、種々のコンピュータ読取可能媒体を含む他の記憶手段を使用してもよい。
【００２０】
ネットワークアダプタ１２６は、コンピュータネットワーク１４０を介してストレージシステム１２０をクライアント１１０に接続するために必要な機械的、電気的、及び信号的回路を含み、コンピュータネットワーク１４０は、ポイント・ツーポイント接続、又はローカルエリアネットワークのような共有媒体を含む場合がある。例えば、コンピュータネットワーク１４０は、イーサネット（Ｒ）ネットワーク、又はファイバチャネル（ＦＣ）ネットワークとして実施される場合がある。クライアント１１０は、トランスミッション・コントロール・プロトコル／インターネット・プロトコル（ＴＣＰ／ＩＰ）のような所定のプロトコルに従って個々のデータフレーム、又はデータパケット１５０をやりとりすることにより、ネットワーク１４０を介してストレージシステムと通信することができる。
【００２１】

クライアント１１０は、アプリケーション１１２を実行するように構成された汎用コンピュータであってよい。また、クライアント１１０は、クライアント／サーバモデルの情報配送に従ってストレージシステム１２０との間で情報をやりとりする場合がある。すなわち、ネットワーク１４０を介してパケット１５０をやりとりすることにより、クライアント１１０はストレージシステム１２０にサービスを要求する場合があり、システムはクライアント１１０から要求されたサービスの結果を返す場合がある。クライアント１１０は、ファイルやディレクトリの形をした情報をアクセスするときは、ＴＣＰ／ＩＰを介して、コモン・インターネット・ファイル・システム（ＣＩＦＳ）プロトコル、又はネットワーク・ファイル・システム（ＮＦＳ）プロトコルのようなファイルベースのアクセスプロトコルを含むパケットを発行する場合がある。一方、クライアントは、ブロックの形をした情報をアクセスするときは、ＳＣＳＩ　ｏｖｅｒ　ＴＣＰ（ｉＳＣＳＩ）プロトコルや、ＳＣＳＩ　ｏｖｅｒ　ＦＣ（ＦＣＰ）プロトコルのようなブロックベースのアクセスプロトコルを含むパケットを発行する場合がある。

【００２２】

ストレージアダプタ１２８は、システム１２０上で実行されているストレージオペレーティングシステム２００と協働し、ユーザ（すなわち、クライアント）から要求された情報をアクセスする。情報は、書き込み可能なストレージデバイス媒体の任意のタイプのアタッチド・アレイに記憶され、媒体には、例えば、ビデオテープ、光学的、ＤＶＤ、磁気テープ、バブルメモリ、電気的ランダムアクセスメモリ、又はＭＥＭＳデバイスなどがあり、あるいはデータやパリティ情報を記憶するように構成された任意の他の適当な媒体も含まれる。ただし、本明細書で例示するように、情報は、アレイ１６０のハードディスクドライブ（ＨＤＤ）、及び／又はダイレクト・アクセス・ストレージ・デバイス（ＤＡＳＤ）のようなディスク１３０に記憶されることが好ましい。ストレージアダプタ１２８は、従来の高性能ＦＣシリアルリンクトポロジのようなＩ／Ｏ相互接続構成によってディスクに接続される入出力（Ｉ／Ｏ）インタフェース回路を含む。

【００２３】

アレイ１６０への情報の記憶は、好ましくは、１以上のストレージ「ボリューム」として実施され、ボリュームは、一群の物理記憶ディスク１３０を含み、それらが
協働して、ボリューム（複数の場合もあり）上のボリュームブロック番号（ｖｂｎ）空間の全体的論理構成を規定する。当然ながら必須ではないが、各論理ボリュームは一般に、各自、自分のファイルシステムに関連している。論理ボリューム／ファイルシステム内のディスクは通常、１以上のグループに編成され、各グループは、ＲＡＩＤ（Redundant Array of Independent or Inexpensive Disks）として運用され、ＲＡＩＤプロトコルに従って管理される場合がある。ＲＡＩＤ４レベル実施形態のような大半のＲＡＩＤ実施形態は、ＲＡＩＤグループ内の所与の数の物理ディスクにわたってデータ「ストライプ」を冗長書き込みし、更に、そのストライピングされたデータに関連するパリティ情報を適切に記憶することにより、データ記憶の信頼性／完全性を向上させる。例示的ＲＡＩＤプロトコルは更に、ディスクに書き込まれる各データブロックについて、データの完全性を確認するために３２ビットチェックサム値を実施する。ＲＡＩＤ実施形態の一例は、ＲＡＩＤ４レベル実施形態であるが、当然ながら、本明細書に記載する本発明の原理によれば、他のタイプ、又は他のレベルのＲＡＩＤ実施形態を使用することも可能であるものと考えられる。同様に、論理ボリューム／ファイルシステムのディスクの編成には、他の実施形態、及び／又はプロトコルが使用される場合もある。

【００２４】

Ｂ．ストレージオペレーティングシステム

ディスク１３０へのアクセスを容易にするために、ストレージシステム２００は、ｗｒｉｔｅ－ａｎｙｗｈｅｒｅファイルシステムを実施する。このファイルシステムは、仮想化モジュールと協働し、ディスク１３０により提供される記憶空間を仮想化する。例えば、ファイルシステムは、情報を名前付きディレクトリ、及びファイルの階層構造としてディスク上に論理編成する。「ディスク」上の各ファイルは、データのような情報を記憶す

るように構成された一組のディスクブロックとして実施される場合がある一方、ディレクトリは、特殊な形のファイルとして実施され、その中に他のファイルやディレクトリの名前、あるいはそれらへのリンクが記憶される場合がある。仮想化モジュールによれば、ファイルシステムは、情報をディスク上のブロックの階層構造としてさらに論理編成し、それらを名前付きの論理ユニット番号（ＬＵＮ）としてエキスポートすることが可能になる。

【００２５】

例示的実施形態として、ストレージオペレーティングシステムは、カリフォルニア州サニーベイルにあるネットワーク　アプライアンス，インコーポレイテッドから販売されているＮｅｔＡｐｐ　Ｄａｔａ　ＯＮＴＡＰオペレーティングシステムであることが好ましい。ただし、当然ながら、任意の適当なストレージオペレーティングシステムを本明細書に記載する本発明の原理に従って拡張して使用することも可能であるものと考えられる。したがって、「Ｄａｔａ　ＯＮＴＡＰ」という用語を使用した場合でも、この用語は、本発明の教示に適合する任意のストレージオペレーティングシステムを指すものとして広い意味で捉えなければならない。

【００２６】

図２は、本発明とともに有利に使用されるストレージオペレーティングシステム２００を示す略ブロック図である。ストレージオペレーティングシステムは、統合ネットワークプロトコルスタックを形成するように編成された一連のソフトウェア層を含む。すなわち、より一般的に言えば、ストレージオペレーティングシステムは、ストレージシステムに記憶された情報をブロックアクセスプロトコル、及びファイルアクセスプロトコルを使用してアクセスするときのデータパスをクライアントに提供するマルチプロトコルエンジンを含む。プロトコルスタックは、ＩＰ層２１２、並びにその支援搬送手段であるＴＣＰ層２１４、及びユーザ・データグラム・プロトコル（ＵＤＰ）層２１６のような種々のネットワークプロトコル層との間のインタフェースとして機能するネットワークドライバ（例えば、ギガビット・イーサネット（Ｒ）・ドライバ）のメディアアクセス層２１０を含む。ファイルシステムプロトコル層は、マルチプロトコルファイルアクセスを可能にし、その目的のために、ダイレクトアクセスファイルシステム（ＤＡＦＳ）プロトコル２１８、ＮＦＳプロトコル２２０、ＣＩＦＳプロトコル２２２、及びハイパーテキスト・トランスファ・プロトコル（ＨＴＴＰ）プロトコル２２４をサポートする。仮想インタフェース（ＶＩ）層２２６は、ＶＩアーキテクチャを実施し、ＤＡＦＳプロトコル２１８によって必要とされるようなＲＤＭＡのようなダイレクト・アクセス・トランスポート（ＤＡＴ）機能を提供する。

【００２７】

ｉＳＣＳＩドライバ層２２８は、ＴＣＰ／ＩＰネットワークプロトコル層を介したブロックプロトコルアクセスを提供する一方、ＦＣドライバ層２３０は、ストレージシステムとの間で、ブロックアクセス要求、及び応答の送受信を行う。ＦＣドライバ、及びｉＳＣＳＩドライバは、ブロックに対するＦＣ固有の、又はｉＳＣＳＩ固有のアクセス制御を提供し、したがって、ストレージシステム上のブロックをアクセスするときに、ｉＳＣＳＩとＦＣＰのいずれか一方、あるいは両方へのＬＵＮのエキスポートを管理する。さらに、ストレージオペレーティングシステムは、ＲＡＩＤシステム２４０として実施され、Ｉ／Ｏオペレーションに従ってボリューム／ディスクに対する情報の記憶、又は読み出しを管理するストレージ（例えばＲＡＩＤ）プロトコルを実施するストレージモジュールと、ＳＣＳＩプロトコルのようなディスクアクセスプロトコルを実施するディスクドライバシステム２５０とを含む。

【００２８】

ディスクソフトウェア層を統合ネットワークプロトコルスタック層に橋渡しするのは、仮想化システムである。仮想化システムはファイルシステム２８０によって実施され、ファイルシステム２８０は、例えばｖディスクモジュール２９０、及びＳＣＳＩターゲットモジュール２７０として実施される仮想化モジュールとの間で情報をやりとりする。ｖデ

ィスクモジュール２９０は、ファイルシステム２８０の上に層として形成され、ストレージシステムに対するユーザ（システム管理者）発行コマンドに応答して、ユーザインタフェース（ＵＩ）２７５のような管理インタフェースによるアクセスを有効化にする。ＳＣＳＩターゲットモジュール２７０は、ＦＣ、ｉＳＣＳＩドライバ２２８，２３０と、ファイルシステム２８０との間に配置され、ＬＵＮがブロックとして表現される場合に、ブロック（ＬＵＮ）空間とファイルシステム空間の間の変換層を仮想化システムに提供する。ＵＩ２７５は、種々の層、及びシステムへの管理者、又はユーザによるアクセスが可能となるような形で、ストレージオペレーティングシステム上に配置される。

【００２９】

本発明の一実施形態によれば、ストレージオペレーティングシステム２００は、重複解消モジュール２８４、及び指紋抽出層２８６をさらに含み、指紋抽出層２８６は、ファイルシステム２８０とＲＡＩＤシステム２４０の間に配置される。重複解消モジュール２８４は、所定の条件（例えば、ストレージシステムが所定の容量に達する等）に応答して定期的に、及び／又はＵＩ２７５を通じて入力された管理者、又はユーザからのコマンドに応答して、データ重複解消プロセスを実施する場合がある。重複解消モジュール２８４、及び指紋抽出層２８６は、指紋キュー２８７、及び指紋データベース２８８との間で情報をやりとりする。指紋キュー２８７、及び指紋データベース２８８はそれぞれ、指紋レコードを記憶し、指紋レコードを使用して、重複データブロックを識別する。例えば、代替実施形態として、指紋キュー２８７、及び指紋データベース２８８は、ストレージシステムメモリ１２４、及び／又はＮＶＲＡＭ１２９に常駐する場合もあるが、それらは個別に、ディスク、及び／又は任意の他のコンピュータ読取可能媒体に記憶される場合もある。また、代替実施形態によれば、ディスクアレイ１６０への書き込みアロケーションオペレーションの実行中にデータ重複解消手順を実施するために、データ重複解消モジュール２８４は、書き込みアロケータプロセス２８２の中に統合される場合もある。「Method and Apparatus for Identifying and Eliminating Duplicate Data Blocks and Sharing Data Blocks in a Storage System」と題する上で援用した米国特許出願に記載されているように、そのようなデータ重複解消手順は、同一のデータを有する複数のデータブロックを指し示す代わりに単一データブロックを指し示すように、間接ブロック中のポインタを変更する場合がある。

【００３０】

ファイルシステムは例えば、ディスクのような記憶装置に記憶された情報をアクセスする際に使用される論理ボリューム管理機能を備えたメッセージベースのシステムである。すなわち、ファイルシステム２８０は、ファイルシステムセマンティックを提供するだけでなく、通常ならばボリュームマネージャに関係する機能も提供する。そうした機能には、例えば、（１）ディスクのアグレゲーション、（２）ディスクの記憶帯域幅のアグレゲーション、及び（３）ミラーリング、及び／又はパリティ（ＲＡＩＤ）のような信頼性保証がある。ファイルシステム２８０は、例えば４キロバイト（ＫＢ）ブロックを使用し、インデックス・ノード（「ｉｎｏｄｅ」）を使用してファイルやファイル属性（例えば、作成時刻、アクセス許可属性、サイズ、及びブロック位置）を識別するブロックベースのオンディスク・フォーマット表現を有するｗｒｉｔｅ－ａｎｙｗｈｅｒｅファイルシステム（例えば、カリフォルニア州サニーベイルにあるネットワーク・アプライアンス，インコーポレイテッドから販売されているＷＡＦＬファイルシステムなど）を実施する。ファイルシステムは、ファイルを使用して、自分が有するファイルシステムのレイアウトを表すメタデータを記憶する。そのようなメタデータファイルとしては、とりわけ、ｉｎｏｄｅファイルがある。ディスクからｉｎｏｄｅを読み出すためには、ファイルハンドル、すなわちｉｎｏｄｅ番号を含む識別子が使用される。

【００３１】

簡単に言えば、ｗｒｉｔｅ－ａｎｙｗｈｒｅファイルシステムのｉｎｏｄｅは全て、ｉｎｏｄｅファイルとして編成される。ファイルシステム（ＦＳ）ｉｎｆｏブロックは、ファイルシステム内の情報のレイアウトを指定し、ファイルシステムの他の全てのｉｎｏｄ

ｅを含むファイルのｉｎｏｄｅを含む。各論理ボリューム（ファイルシステム）は、ＦＳ　ｉｎｆｏブロックを有し、ＦＳ　ｉｎｆｏブロックは、例えばＲＡＩＤグループ内の固定位置に記憶されることが好ましい。ｉｎｏｄｅファイルのｉｎｏｄｅはｉｎｏｄｅファイルのブロックを直接参照（指し示す）場合もあれば、ｉｎｏｄｅファイルの間接ブロックを参照し、さらに、その間接ブロックが、ｉｎｏｄｅファイルのブロックを指し示す場合もある。ｉｎｏｄｅファイルの直接ブロックの中にはそれぞれ、種々のｉｎｏｄｅが埋め込まれ、それらのｉｎｏｄｅがそれぞれ、間接ブロックを参照し、さらに、その間接ブロックが、ファイルのデータブロックを参照する。

【００３２】

動作として、クライアント１１０からの要求は、パケット１５０としてコンピュータネットワーク１４０を介してストレージシステム１２０へ転送され、そこで、ネットワークアダプタ１２６により受信される。（層２１０、又は層２３０）のネットワークドライバは、そのパケットを処理し、必要に応じて、それをネットワークプロトコル層、及びファイルアクセス層へ渡し、更なる処理を施した後、ｗｒｉｔｅ－ａｎｙｈｗｅｒｅファイルシステム２８０へと転送する。ここで、ファイルシステムは、要求されたファイルが、「コア」内、すなわちバッファキャッシュ１７０内に無ければ、それをディスク１３０からロードするオペレーションを生成する。情報がキャッシュ内に無ければ、ファイルシステム２８０は、ｉｎｏｄｅ番号を使用してｉｎｏｄｅファイルを検索し、適当なエントリにアクセスし、論理ｖｂｎを読み出す。次にファイルシステムは、その論理ｖｂｎを含むメッセージ構造をＲＡＩＤシステム２４０へ渡す。そして、論理ｖｂｎは、ディスク識別子、及びディスクブロック番号（ｄｉｓｋ，ｄｂｎ）にマッピングされ、ディスクドライバシステム２５０の適当なドライバへと送られる。ディスクドライバは、指定されたディスク１３０からｄｂｎをアクセスし、ストレージシステムによる処理に備えて、要求されたデータブロック（複数の場合も有り）をバッファキャッシュ１７０に読み出す。要求が完了すると、ストレージシステム（及び、オペレーティングシステム）は、ネットワーク１４０を介してクライアント１１０に返答を返す。

【００３３】

クライアント１１０からストレージシステム１２０に対して発行される書き込みオペレーションは、そのオペレーションが、後のディスク１３０へのコミットに備えて、ＮＶＲＡＭ１２９のＮＶＬＯＧ１３１に一時的に記憶（「キューイング」）されることがある点を除けば、ストレージシステム１００を貫通する同じ「パス」を通る。同様に、書き込みオペレーションは、その書き込みオペレーションによって変更された論理ｖｂｎに対応する指紋レコードを作成するために、指紋抽出層２８６、及び／又は重複解消モジュール２８４が、書き込みオペレーションの際にｉｎｏｄｅファイル中の適当なエントリに関連するメタデータのようなデータを抽出する場合がある点を除けば、ストレージオペレーティングシステム２００を貫通する同じソフトウェア「パス」を通る。

【００３４】

なお、ストレージシステムにおいて受信されたクライアント要求に対し、データストレージアクセスや書き込みオペレーションを実施するために必要となる上記のストレージオペレーティングシステム層を貫通するソフトウェア「パス」は、代替としてハードウェアで実施してもよい。すなわち、本発明の代替実施形態では、ストレージアクセス要求データパスは、フィールド・プログラマブル・ゲート・アレイ（ＦＰＧＡ）、又は特定用途向け集積回路（ＡＳＩＣ）の中に埋め込まれた論理回路として実施される場合がある。この種のハードウェア実施形態によれば、クライアント１１０により発行された要求に応答してストレージシステム１２０によって提供されるストレージサービスの性能を向上させることができる。また、本発明の更に別の代替実施形態として、アダプタ１２６、１２８の処理要素は、パケット処理オペレーション、及びストレージアクセスオペレーションの負荷の一部、又は全部をプロセッサ１２２から取り除くように構成される場合があり、それによって、システムにより提供されるストレージサービスの処理能力を向上させる場合がある。当然ながら、本明細書に記載する種々のプロセス、アーキテクチャ、及び手順は、

単一のシステムとして実施しても、分散システムとして実施してもよく、また、ハードウェアで実施しても、ファームウェアで実施しても、ソフトウェアで実施してもよい。

【００３５】

本明細書では、「ストレージオペレーティングシステム」という用語は通常、ストレージシステムにおいてストレージ機能を実施するためのコンピュータ実行可能コードを意味し、例えば、データアクセスを管理し、ファイルサーバの場合には、ファイルシステムセマンティックを実施することもあるコンピュータ実行可能コードを意味する。その意味で、Ｄａｔａ　ＯＮＴＡＰソフトウェアは、マイクロカーネルとして実施されるストレージオペレーティングシステムであって、かつＷＡＦＬファイルシステムセマンティックを実施し、データアクセスを管理するためのファイルシステム２８０を含む、そのようなストレージオペレーティングの一例である。例えば、本発明によれば、ストレージオペレーティングシステム２００は、指紋を比較し、データ重複解消を実施するための重複解消モジュール２８４を含む。しかしながら、代替実施形態では、指紋比較機能やデータ重複解消機能は、ストレージオペレーティングシステム２００の他のモジュールで実施される場合もある。また、ストレージオペレーティングシステムは、ＵＮＩＸやＷｉｎｄｏｗｓ　ＸＰのような汎用オペレーティングシステム上で動作するアプリケーションプログラムとして実施してもよいし、本明細書に記載するようなストレージアプリケーションに合わせて構成された構成変更機能を備えた汎用オペレーティングシステムとして実施してもよい。

【００３６】

さらに、当業者には分かるように、本明細書に記載する本発明の技術は、いかなるタイプの特殊目的のコンピュータ（例えばファイルサーバ、ファイラ、又はマルチプロトコルストレージアプライアンス）にも、汎用のコンピュータにも適用することができ、例えば、ストレージシステム１２０として実施され、又はストレージシステム１２０を含むスタンドアロンのコンピュータ、あるいはその一部にも適用することができる。本発明とともに有利に使用されるストレージシステムアプライアンスの一例は、２００２年８月８日に出願された「Protocol Storage Appliance that Provides Integrated Support for File and Block Access Protocols」と題するBrian Pawlowski他による米国特許出願公報第２００４／００３０６６８Ａ１号に記載されている。また、本発明の教示は、種々のストレージシステムアーキテクチャに適合させることができ、限定はしないが、例えば、ネットワーク・アタッチド・ストレージ環境、ストレージ・エリア・ネットワーク、あるいは、クライアントコンピュータやホストコンピュータに直接取り付けされたディスクアセンブリにも適用することが可能である。したがって、「ストレージシステム」という用語は、ストレージ機能を実施するように構成され、他の装置又はシステムに関連する何らかのサブシステムだけでなく、そのような構成も含むものとして広い意味で捉えなければならない。

【００３７】

Ｃ．オンディスク・ファイルシステム構造

例示的実施形態として、Ｗｒｉｔｅ－ａｎｙｗｈｅｒｅファイルシステムでは、ファイル（又は、他のデータコンテナ）は、ディスク１３０への記憶に適したｉｎｏｄｅデータ構造として表現される。図３は、ｉｎｏｄｅ３００を示す略ブロック図である。ｉｎｏｄｅ３００は、メタデータ部３１０、及びデータ部３５０を有することが好ましい。各ｉｎｏｄｅ３００のメタデータ部３１０に記憶される情報はファイルを表し、例えば、ファイルのタイプ３１２（例えば、通常、ディレクトリ、仮想ディスクなど）、ファイルのサイズ３１４、ファイルのタイムスタンプ（アクセス、及び／又は変更）３１６、ファイルの所有者、例えばユーザＩＤ（ＵＩＤ３１８）、及びグループＩＤ（ＧＩＤ３２０）、コンシステンシ・ポイント（ＣＰ）フィールド３２２、並びに生成番号フィールド３２４を含む。

【００３８】

ＣＰカウントフィールド３２２は、特定バージョンのｉｎｏｄｅが生成された時点におけるＣＰカウントを識別する。例えば、各ＣＰは、単調増加ＣＰカウンタによって生成さ

れた一意のＣＰ番号によって識別される。代替実施形態では、ＣＰカウンタは、各ＣＰに一意の識別子を与えるような他の技術を使用して生成される場合がある。生成番号フィールド３２４は、特定ｉｎｏｄｅの生成を識別する。例えば、ｉｎｏｄｅが削除され、再使用されるたびに、そのｉｎｏｄｅに関連する生成番号３２４は、インクリメントされる。ｗｒｉｔｅ　ａｎｙｗｈｅｒｅファイルシステムでは、ｉｎｏｄｅが変更されたときは常に、新たなコピーがディスクに書き込まれる。そのようなとき、すなわち、ｉｎｏｄｅ書き込みアロケーションの際に、それらのフィールド３２２、３２４は更新される。従って、生成番号は、ｉｎｏｄｅの生成／ア割り当てを反映し、ＣＰカウントは、ｉｎｏｄｅの変更を反映する。フィールド３２２、３２４は、特定のデータコンテナの指紋が生成された後、ｉｎｏｄｅによって表されるそのデータコンテナに変更があったか否かを迅速に判定するために使用される場合がある。

【００３９】

ただし、各ｉｎｏｄｅのデータ部３５０の中身は、タイプフィールド３１２の中に規定されるファイル（ｉｎｏｄｅ）のタイプによって、異なる解釈をされる場合がある。例えば、ディレクトリｉｎｏｄｅのデータ部３５０は、ファイルシステムによってコントロールされるメタデータを有する一方、通常ｉｎｏｄｅのデータ部は、ファイルシステムデータを有する場合がある。後者の場合、データ部３５０は、ファイルに関連するデータの表現物を含む場合がある。

【００４０】

例示的実施形態によれば、通常オンディスクｉｎｏｄｅのデータ部３５０は、ファイルシステムデータ、又はポインタを含む場合があり、後者は、ファイルシステムデータを記憶するために使用されるディスク上の４ＫＢデータブロックを参照する。ディスク上のデータをアクセスするときのファイルシステムとＲＡＩＤシステム２４０の間の効率を上げるために、各ポインタは、論理ｖｂｎであることが好ましい。ｉｎｏｄｅのサイズには制限があるため（例えば１２８バイト）、６４バイト以下のサイズのファイルシステムデータは、その全体が、そのｉｎｏｄｅのデータ部の中に表現される。一方、ファイルシステムデータのサイズが６４バイトよりも大きく、且つ６４ＫＢ以下である場合、ｉｎｏｄｅ（例えば、第１レベルｉｎｏｄｅ）のデータ部は、最大で１６個までのポインタを有し、各ポインタが、ディスク上の４ＫＢブロックのデータを参照する。

【００４１】

また、データのサイズが６４ＫＢよりも大きく、且つ６４メガバイト（ＭＢ）以下である場合、ｉｎｏｄｅ（例えば、第２レベルｉｎｏｄｅ）のデータ部３５０にある各ポインタは、１０２４個のポインタを有する間接ブロック（例えば、第１レベルブロック）を参照し、各ポインタが、ディスク上の４ＫＢデータブロックを参照する。６４ＭＢよりも大きなサイズのファイルシステムデータの場合、ｉｎｏｄｅ（例えば、第３レベルｉｎｏｄｅ）のデータ部３５０にある各ポインタは、１０２４個のポインタを含む二重間接ブロック（例えば、第２レベルブロック）を参照し、さらに、各ポインタが、間接（例えば、第１レベル）ブロックを参照する。そして、その間接ブロックが、１０２４個のポインタを有し、各ポインタが、ディスク上の４ＫＢのブロックを参照する。ファイルをアクセスする場合、ファイルの各ブロックは、ディスク１３０からバッファキャッシュ１７０へロードされる場合がある。

【００４２】

オンディスクｉｎｏｄｅ（又は、ブロック）が、ディスク１３０からバッファキャッシュ１７０へロードされるとき、それに対応するコア内構造が、オンディスク構造に埋め込まれる。例えば、ｉｎｏｄｅ３００の周りを囲む点線は、オンディスクｉｎｏｄｅ構造のコア内表現を示している。コア内構造は、オンディスク構造、及びメモリ上の（ただし、ディスク上に無い）データの管理に必要となる他の情報を記憶する一つのメモリブロックである。他の情報には、例えば、「ダーティ」ビット３６０がある。例えば、書き込みオペレーションによる命令に従って、ｉｎｏｄｅ（又はブロック）内のデータが更新／変更された後、変更されたデータは、ダーティビット３６０を使用して、「汚れた」ものとし

てマーキングされ、その後、そのｉｎｏｄｅ（ブロック）はディスクに「フラッシュ」（記憶）できるようになる。ｉｎｏｄｅ、及びｉｎｏｄｅファイルを含む、ＷＡＦＬファイルシステムのコア内構造、及びオンディスクフォーマット構造については、１９９８年１０月６日に発行された「Method for Maintaining Consistent States of a File System and for Creating User-Accessible Read-Only Copies of a File System」と題するDavid Hitz他による米国特許第５，８１９，２９２号に開示、及び記載されており、この文献は、参照により本明細書に援用される。

【００４３】

図４は、本発明とともに有利に使用されるデータコンテナのバッファツリーを示す略ブロック図である。バッファツリーは、バッファキャッシュ１７０にロードされ、ｗｒｉｔｅ－ａｎｙｗｈｅｒｅファイルシステム２８０によって管理されるデータコンテナ（例えばファイル４００）の種々のブロックの内部表現である。埋め込みｉｎｏｄｅのようなルート（最上位）ｉｎｏｄｅ３００は、間接（例えば、第１レベル）ブロック４０４を参照する。間接ブロック（及び、ｉｎｏｄｅ）は、ファイル４００の実際のデータの記憶に使用されるデータブロック４０６を最終的に最終的に参照するポインタ４０５を含む。すなわち、ファイル４００のデータは、種々のデータブロックに記憶され、それらのブロックの位置が、ファイルの間接ブロックに記憶される。各レベル１間接ブロック４０４は、１０２４個もの数のデータブロックへのポインタを有する場合がある。ファイルシステムの「ｗｒｉｔｅ　ａｎｙｗｈｅｒｅ」な性質によれば、これらのブロックは、ディスク１３０上のどこに配置されることもありうる。

【００４４】

Ｄ．データ重複解消

図５は、本発明の一実施形態による、データブロック４０６への複数のポインタを含むレベル１間接ブロック４０４を示す略ブロック図である。例えば、各データブロック４０６は、４ＫＢのデータを含む。特に、ｗｒｉｔｅ　ａｎｙｗｈｅｒｅファイルレイアウトによれば、レベル０データブロックは、ファイルシステム内のどこに配置されることもあり得る（すなわち、それらが、ディスク１３０上の物理的に連続したブロックに対応している必要はない）。データを削除、及び／又は重複解消するように構成されたストレージオペレーティングシステム２００では、各データブロック４０６内のデータに、指紋を関連付ける場合がある。例えば、例示的なデータコンテナは、例えばｖｂｎ１、ｖｂｎ２、ｖｂｎ３、及びｖｂｎ４に記憶された一連のデータブロックを含む。各一意のデータブロックには、一意の指紋、例えば、Ａ、Ｂ、Ｃ、及びＤが関連付けられる。同様に、間接ブロック４０４の中にある一連のポインタ４０５、例えば、Ｐ１、Ｐ２、Ｐ３、及びＰ４は、データブロックｖｂｎ１、ｖｂｎ２、ｖｂｎ３、及びｖｂｎ４をそれぞれ参照する。

【００４５】

図６は、書き込みオペレーションによってデータコンテナに変更が加えられた後の、図５のレベル１間接ブロック４０４を示す略ブロック図である。例えば、ｖｂｎ３、及びｖｂｎ４に以前に記憶された２つのデータブロックは、現在は、ｖｂｎ１に記憶された第１のデータブロックのコピーを有するように変更され、その結果、それらに関連する指紋は同一になっている。変更が加えられたデータブロックは、今度は、指紋Ａ、Ｂ、Ａ、及びＡにそれぞれ関連している。ｗｒｉｔｅ　ａｎｙｗｈｅｒｅファイルレイアウトによれば、変更の加えられたデータには、２つの新たなｖｂｎ（ｖｂｎ５とｖｂｎ６）が割り当てられる。その結果、重複解消モジュール２８４は、それらの同一の指紋を使用して、ｖｂｎ５、及びｖｂｎ６に記憶されているデータブロック４０６が、ｖｂｎ１にあるデータブロック４０６に記憶されているデータの複製を有しているか否かを判定する場合がある。同一データの３つのコピーを有することは、ファイルシステムリソースの浪費である。従って、重複解消モジュール２８４は、ポインタＰ３、及びＰ４をｖｂｎ１を参照するように変更し、それによって、データコンテナの完全性を維持しつつも、ｖｂｎ５、及びｖｂｎ６の割り当てが解除され、記憶空間は節約される。

【００４６】

図７は、例示的なデータ重複解消手順の実施後の、図６のレベル１間接ブロック４０４を示す略ブロック図である。例示的手順の後、各一意のデータブロックは、一意の指紋に関連し、データコンテナの複数の同一のファイルブロックが、一意のデータブロックに関連し、さらに、その一意のデータブロックが、一意の指紋に関連する。重複解消手順を実施する技術の詳細については、「Method and Apparatus for Identifying and Eliminating Duplicate Data Blocks and Sharing Data Blocks in a Storage System」と題する、上で援用した米国特許出願に記載されている。

【００４７】

Ｅ．指紋レコードの生成

本発明の一実施形態によれば、ストレージシステム１２０は、ファイルシステム２８０のような上位モジュールを実施し、ＲＡＩＤシステム２４０のようなストレージモジュールによって管理されるディスク１３０のような複数の記憶装置に記憶される情報を論理編成する。上位モジュールとストレージモジュールの間の境界において、指紋抽出層２８６は、ストレージシステム１２０によって記憶される各データブロック４０６について、例えば６４ビットのような所定サイズの指紋を生成する。例えば、指紋の最初の所定数のビット、例えば、最初の３２ビットは、ストレージモジュールによって実施されるチェックサム計算の結果から得られ、指紋の次の所定数のビット、例えば、次の３２ビットは、データブロック４０６中の１以上の所定のオフセットに記憶されたデータから得られる。得られた指紋は、データブロック４０６の中身を十分に反映し、望ましくない誤認識率を生じることなく、重複の識別を可能にする。また、指紋の生成のために、ハッシュ関数のような他の暗号化関数をさらに実施する必要はないため、プロセッサ使用率も低減される。

【００４８】

さらに、指紋抽出層２８６は、特定のポリシーに基いて、データブロックのサブセットに対して指紋を生成するように選択される場合もある。例えば、ユーザファイルに属するデータブロックに対して（ファイルシステムによって使用される内部的なメタデータファイルとは対照的に）指紋が生成される場合がある。あるいは、レベル０ブロック４０６に対して（間接ブロック４０４とは対称的に）、指紋が生成される場合がある。

【００４９】

各指紋は指紋レコードに記憶され、指紋レコードは更に、指紋データベース２８８に記憶され、データ重複解消モジュール２８４によってアクセスされる。データ重複解消モジュール２８４は、指紋を定期的に比較し、重複する指紋を識別し、重複データブロック４０６を知らせる場合がある。次に、重複データブロック４０６が比較され、重複データは削除される。

【００５０】

本発明の一実施形態によれば、指紋抽出層２８６は、ストレージシステム１２０によって記憶される各データブロック４０６に対して６４ビットの指紋を生成する。図８は、例示的な指紋８００の種々の要素を示す略ブロック図である。指紋８００は例えば、ＲＡＩＤチェックサムフィールド８０５、及びサンプルデータフィールド８１０を含み、代替実施形態では、さらに別のフィールド８１５を有する場合がある。なお、代替実施形態では、更に別のフィールド、及び／又は異なるフィールドが使用される場合もある。あるいは、ＲＡＩＤチェックサムが、重複解消モジュールのニーズに十分に合致するチェックサム関数を含む場合、指紋８００は、ＲＡＩＤチェックサムだけしか有しない場合もある。

【００５１】

通常の書き込みオペレーションの一部として、ディスクに書き込まれた各データブロックに対するチェックサム計算を実施することにより、周知のＲＡＩＤプロトコルによってデータの完全性が確認される。例えば、データブロック４０６に対する各書き込みオペレーションについて、指紋抽出層２８６は、ＲＡＩＤプロトコルに従って計算された３２ビットのチェックサム値を識別し、それを指紋８００のＲＡＩＤチェックサムフィールド８０５にコピーする。従って、指紋を生成するプロセスは、各データブロック４０６に関連する事前計算されたチェックサム値の利点が得られるように最適化される。あるいは、Ｒ

ＡＩＤチェックサムフィールド８０５は、ＲＡＩＤプロトコルに従って計算された３２ビットチェックサムの一部だけを単独で、又は他の値と組み合わせて有する場合がある。同様に、ストレージプロトコルが、３２ビットＲＡＩＤチェックサム値以外のチェックサム値を生成する場合、指紋８００は、ＲＡＩＤチェックサムフィールド８０５において、そのような他のチェックサム値を実施する場合がある。あるいは、ＲＡＩＤチェックサムフィールド８０５は、データブロック４０６に記憶されたデータに関連する、ハッシュ関数のような他の暗号化関数の結果を含む場合がある。

【００５２】

例えば、サンプルデータフィールド８１０は、指紋８００に関連するデータブロック４０６内の１以上の所定のオフセットに記憶されたデータのコピーを有する。例えば、データブロック４０６が４ＫＢのデータを有している場合、サンプルデータフィールド８１０は、３２ビット（４バイト）の長さを有し、データブロック４０６に記憶されたデータの１０２４ビット目ごと（すなわち、１２８バイト目ごとにその最初のビット）のコピーを有する場合がある。あるいは、サンプルデータフィールド８１０は、データブロック４０６内の単一オフセットに記憶された３２個の連続したビットのコピーを有する場合がある。得られる指紋８００は、データブロック４０６の中身を十分に反映し、望ましくない誤認識率を生じることなく、重複の識別を可能にする。したがって、ここでも、指紋を生成するプロセスは、ストレージオペレーティングシステム２００によって更に別のハッシュ計算、及び／又は暗号化関数を実施する必要なく、ストレージオペレーティングシステム２００に既に提供されたデータの利点が得られるように最適化される。代替実施形態によれば、ＲＡＩＤチェックサムフィールド８０５、及びサンプルデータフィールド８１０は、例えば、ストレージオペレーティングシステム２００によって実施される更に別のハッシュ計算、及び／又は暗号化関数の結果といった、データブロック４０６に関連する他のデータを含むさらに別のフィールド８１５と組み合わされる場合がある。

【００５３】

各指紋は、データ重複解消モジュール２８４によって生成された指紋レコードに記憶される。図９は、本発明とともに有利に使用される指紋レコード９００を示す略ブロック図である。指紋レコード９００は、指紋８００を記憶するフィールド、ｉｎｏｄｅ番号フィールド９０５、ファイルブロック番号（ｆｂｎ）フィールド９１０、生成番号フィールド９１５、及びコンシステンシ・ポイント（ＣＰ）カウントフィールド９２０を含み、代替実施形態では、更に別のフィールド９２５を含む場合もある。ｉｎｏｄｅ番号フィールド９０５は、指紋が生成されたデータブロック４０６のｉｎｏｄｅ３００へのポインタを記憶する。ｆｂｎフィールド９１０は、データコンテナ内のブロックのオフセットを記憶する。例えば、データコンテナが複数のデータブロック４０６を含む場合、ｆｂｎフィールド９１０の値は、どのデータブロック４０６が指紋８００に対応するものかを識別する。指紋レコード９００の他のフィールド９１５、９２０、及び９２５は、書き込みオペレーションの際に、そのブロックのｉｎｏｄｅ３００のメタデータ部から収集される場合がある。例えば、ｉｎｏｄｅ３００のＣＰカウントフィールド３２２、及び生成番号フィールド３２４に記憶される値は、指紋レコード９００のＣＰカウントフィールド９２０、及び生成番号フィールド９１５からそれぞれコピーされる場合がある。従って、指紋レコード９００は、指紋８００に関連するデータを記憶し、更なる指紋処理機能をストレージオペレーティングシステムに与える。

【００５４】

図１０は、本発明の一実施形態による、指紋レコードを生成する手順１０００のステップの詳細を示すフロー図である。手順１０００は、ステップ１００５から開始され、ステップ１０１０へ進み、そこで、ストレージオペレーティングシステム２００のファイルシステム２８０は、書き込みコマンドをＲＡＩＤシステム２４０へ発行する。例えば、この書き込みコマンドは、ストレージシステム１２０に接続されたクライアント１１０によって生成される、新たなデータコンテナの新たなデータブロック４０６を書き込むためのコマンドである場合がある。あるいは、書き込みコマンドは、ｗｒｉｔｅ　ａｎｙｗｈｅｒ

ｅファイルレイアウトに従って新たなデータブロック４０６を割り当てることにより、既存のデータコンテナのデータを変更するコマンドである場合もある。

【００５５】

手順１０００はステップ１０２０へ進み、そこでＲＡＩＤシステムは、３２ビットチェックサム値を計算し、例えば、書き込みオペレーションをディスクドライバシステム２５０を通じて送信することにより、書き込みオペレーションを完了する。ステップ１０２５において、ＲＡＩＤシステム２４０は、次に、確認メッセージをファイルシステム２８０へ返す。なお、書き込みオペレーションを実施し、チェックサム値のような値を返すために、他のストレージプロトコル、すなわち、他のＲＡＩＤプロトコルを実施してもよい。ステップ１０３０において、指紋抽出層２８６は、ＲＡＩＤシステム２４０からの確認メッセージを捕捉し、及び／又は監視する。ステップ１０３５において、指紋抽出層２８６は、確認メッセージから３２ビットチェックサム値を抽出し、その値を指紋８００のＲＡＩＤチェックサムフィールド８０５にコピーする。また、指紋抽出層２８６は、データブロック４０６の種々のオフセットから３２ビットのデータをサンプリングし、そのデータを指紋８００のサンプルデータフィールド８１０にコピーする。従って、６４ビットの指紋８００が生成される。さらに、指紋抽出層２８６は、データブロック４０６に関連するメタデータを抽出する。指紋レコード９００が生成され（例えば指紋レコード９００の種々のフィールドを記憶するための１以上のメモリエントリを割り当てることにより）、抽出されたメタデータは、指紋レコード９００の適当なフィールドにコピーされる。代替実施形態によれば、指紋抽出層２８６は、書き込みコマンドに応答して、ファイルシステム２８０が提供するデータに基いてそれらのフィールドのデータを生成する場合がある。

【００５６】

手順１０００はステップ１０４０へと進み、そこで、指紋８００は指紋レコード９００に追加される。ステップ１０４５において、指紋レコード９００は、指紋キュー２８７に追加され、そこで、新たに生成された指紋は、図１１を参照して後で説明するような、重複解除モジュール２８４によって実施されるバッチ処理に備えて蓄積される。ステップ１０５０において、手順１０００は終了する。

【００５７】

なお、代替実施形態によれば、指紋レコード生成手順１０００は、定期的に実施される場合もあれば、書き込みオペレーション中に実施されるのではなく、例えばＵＩ２７５から受信される管理者コマンドのような、所定の条件に応答して実施される場合もある。そのような実施形態では、ファイルシステム２８０は、ディスクアレイ１６０、又はその一部をスキャンし、指紋レコードを生成する場合がある。

【００５８】

例えば、データ重複解消モジュール２８４は、重複する指紋を識別し、重複するデータブロックを示すために、定期的に指紋を比較する場合がある。なお、例示的実施形態によれば、重複する一対の指紋は、「誤認識」であることもある。「誤認識」とは、それらの指紋が、実際の重複データブロックには該当しないことを意味する。従って、重複解消を行う前に、識別されたデータブロックをバイトごとに比較し、それらが実際に重複であるか否かを確認する。例えば、各指紋８００が、３２ビットのＲＡＩＤチェックサム、及び３２ビットのサンプルデータを有している場合、指紋レコード生成手順１０００におけるストレージシステム処理リソースは、データ重複解消の際の誤認識の確率と引き換えに、節約される。ただし、誤認識の確率は非常に低く（４０億個のデータブロックにつき約１００個の誤認識）、事前計算されたデータを使用して指紋８００を生成することにより節約されるプロセッサリソースの量は、データ重複解消の際にバイトごとの比較を実施することにより消費されるプロセッサリソースの量を上回る。さらに、重複解消プロセスは、システムの活動が活発でない時、例えば深夜、又は週末に実施される場合がある。したがって、本明細書では、「同一」の、又は「重複」するデータブロックは、低い確率で実際には重複していないデータブロック（誤認識）も含む。

【００５９】

一方、代替実施形態によれば、各指紋８００において、ＲＡＩＤチェックサムフィールドとサンプルデータフィールド８０５、８１０に異なる値がコピーされる場合があり、及び／又は、更に別のフィールド８１５が実施される場合がある。例えば、指紋８００は、誤認識の確率が確実にゼロになるようにするために暗号ハッシュ関数の結果を含む場合がある。従って、代替実施形態によれば、データ重複解消の際にバイト単位のデータブロック比較を実施する必要は無い場合もある。

【００６０】

Ｆ．重複データブロックの識別

図１１は、本発明の一実施形態による重複データブロック４０６を識別する手順１１００のステップの詳細を示すフロー図である。手順１１００は、ステップ１１０５から開始され、手順１１１０へ進み、そこで、指紋キュー２８７に記憶されている指紋を指紋データベース２８８に記憶されている指紋と比較する。もし、同一の指紋があれば、手順１１００はステップ１１１５へ進み、そこで、重複解消モジュール２８４は、例えば、同一の指紋に関連するデータブロック間でバイトごとの比較を実施し、その後、間接ブロック中のポインタを変更することにより、データ重複解消を実施する。次に、ステップ１１２０において、指紋キュー２８７に記憶されている指紋を指紋キュー２８７に記憶されている他の指紋と比較する。例えば、この比較は、２つの指紋間におけるビットごとの比較であり、重複解消モジュール２８４によって実施される。代替実施形態によれば、この比較は、例えば、バイトごとの比較であってもよく、及び／又は同一データを識別する他の方法であってもよい。同一の指紋があれば、手順１１００はステップ１１２５へ進み、そこで、重複解消モジュール２８４は、図５、図６、及び図７を参照して上で例示したものと同様にして、データ重複解消を実施する。次に、手順１１００はステップ１１３０へ進み、そこで、指紋キュー２８７に記憶されている指紋レコードを指紋データベース２８８にコピーする。次に、ステップ１１３５において、指紋キュー２８７をフラッシュし、すなわち、指紋キュー２８７内の全ての指紋レコードを削除する。手順１１００はステップ１１４０で終了する。

【００６１】

なお、例示的実施形態によれば、上記手順１１００は、アクティブファイルシステムに対して実施される場合がある。従って、ストレージオペレーティングシステム２００には、２つ以上の指紋キューが存在する場合がある。手順１１００のにおいて重複解消モジュール２８４がキューの一つをアクセスしている間も、書き込みオペレーションは継続され、新たに生成された指紋レコードは、第２の指紋キューに記憶されるか、又は、一時的記憶場所に記憶される場合がある。新たに生成された指紋レコードを指紋データベース２８８に記憶するのではなく、指紋キュー２８７に記憶することにより、新たに書き込まれたデータブロック、又は最近変更が加えられたデータブロックに対応する指紋を一つのグループにまとめて「バッチ」処理することができる。あるいは、新たに生成された指紋レコードは、直ぐに比較し、処理するために、指紋データベース２８８に直接記憶される場合もある。また、指紋データベース２８８の一端に指紋キューを有効に生成するために、指紋データベース２８８は、指紋生成の順番に従ってさらにソートされる場合があり、ソートされた指紋データベース２８８は、独立した指紋キュー２８７を必要とせずに、手順１１００によってバッチ処理される場合がある。さらに別の実施形態によれば、新たに生成された指紋レコードは、指紋データベース２８８に直接記憶され、指紋値８００によってソートされる場合があり、重複解消モジュール２８４は、指紋データベース２８８の全体、又は種々の部分を定期的にスキャンすることにより、重複を調べる場合がある。

【００６２】

例示的実施形態によれば、指紋８００間の比較を容易にするために、指紋キュー２８７、及び指紋データベース２８８の指紋レコード９００は、指紋レコードが有している指紋８００の値に基いてソートされる場合がある。例えば、図１２Ａは、ソートされた指紋キュー２８７、及びソートされた指紋データベース２８８を示している。例えば、指紋８００は文字Ａ～Ｇで表され、指紋キュー２８７内の種々の場所はＱ１～Ｑ６で表され、指紋

データベース２８８内の種々の場所はＤ１～Ｄ４で表されている。

【００６３】

上記手順１１００のステップ１１１０、及び１１１５では、指紋キュー２８７と指紋データベース２８８の両方にある同一の指紋が識別され、それに対応するデータブロックの重複が解消される。例えば、図１２Ｂは、識別された重複する指紋（「Ｃ」、及び「Ｄ」）を示している。記憶場所Ｑ３、及びＤ１に記憶されている指紋レコードに関連するデータブロックを比較し、それらが、実際の重複データブロックであるか否かが、判定される。同様に、記憶場所Ｑ４、及びＤ２に記憶されている指紋レコードに関連するデータブロックが、比較される場合がある。このような比較を実施し、実際の重複データブロックの重複を解消する技術は、「Method and Apparatus for Identifying and Eliminating Duplicate Data Blocks and Sharing Data Blocks in a Storage System」と題する上記の米国特許出願に開示されている。

【００６４】

上記手順１１００のステップ１１２０、及び１１２５では、指紋キュー２８７内の同一の指紋が識別され、それらに対応するデータブロックの重複が解消される。例えば、図１２Ｃは、指紋キュー２８７内で識別された重複する指紋（「Ｇ」）を示している。記憶場所Ｑ５、及びＱ６に記憶されている指紋レコードに関連するデータブロックを比較し、それらが実際の重複データブロックであるか否かが、判定される場合がある。

【００６５】

実際の重複データブロック、及びそれに対応する指紋レコードは削除され、上記手順１１００のステップ１１３０において、指紋キュー２８７は指紋データベース２８８に追加される。結果得られるマージ後の指紋データベース２８８は、指紋値８００に従ってソートされる。例示的実施形態によれば、誤認識の確率がある場合、誤認識された指紋を含む指紋レコードは、保持される場合がある。例えば、図１２Ｄは、手順１１００の完了後の、マージされ、記憶された指紋データベース２８８を示している。例えば、指紋「Ｃ」、及び「Ｇ」は実際の重複データブロックに相当するが、指紋「Ｄ」は誤認識であった。従って、指紋「Ｃ」と「Ｇ」の重複は解消され、それらに対応するデータブロックの重複は解消された。一方、指紋「Ｄ」の重複は保持され、指紋データベース２８８にマージされた。もし、指紋「Ｄ」を有する新たなデータブロックが生成された場合、そのデータブロックは、指紋「Ｄ」を有する既存のデータブロックのいずれか、又は両方と比較することが望ましい。

【００６６】

Ｇ．古くなった指紋の削除

例示的実施形態によれば、各指紋レコード９００は、指紋が生成されたデータブロック４０６に関する更に別の情報を有する場合がある。データ重複解消モジュール２８４は、その別の情報を、データブロック４０６に関連するメタデータと定期的に比較し、指紋データベース２８８が、ストレージシステムの最新の状態を確実に反映するようにする。そのため、「古くなった」指紋（例えば、削除されたデータブロック、及び／又は変更されたデータブロックに関連する指紋）は、指紋データベース２８８から削除される場合がある。あるいは、古くなった指紋を削除するプロセスは、例えば、指紋データベース２８８が所定のサイズに達するといった所定の条件に応答して実施され、あるいは、古くなった指紋を削除するプロセスは、指紋を比較する上記プロセスの直前に実施される場合がある。あるいは、指紋は、指紋が古くなったとき、すなわち、対応するデータが削除され、又は変更されたときに削除される場合がある。

【００６７】

一実施形態によれば、ＲＡＩＤシステム２４０に対して書き込みオペレーションが実施されるたびに、新たな指紋レコード９００が生成される。従って、同じデータブロックに対して複数の書き込みオペレーションが実施された場合、複数の指紋レコードが生成されることもある。ただし、そのデータブロックを正確に反映するのは、最新の指紋レコードだけである。同様に、データブロックは、対応する指紋レコードが陳腐化したときに、削

除され、及び／又は再割り当てされる場合がある。従って、処理能力を向上させ、及び／又は指紋データベース２８８によって消費される空間を減らすためには、「古くなった」指紋レコード（例えば、削除されたデータブロック、及び／又は変更されたデータブロックに関連する指紋レコード）を、指紋データベース２８８から削除する必要がある。

【００６８】

図１３は、本発明の一実施形態による、古くなった指紋レコードを削除する手順１３００のステップを詳細に示すフロー図である。例えば、手順１３００は、重複解消モジュール２８４によって実行されるが、手順１３００は、ストレージオペレーティングシステム２００との間で情報をやりとりする機能を有するものであれば、他のいかなるモジュールによって実行されてもよい。手順１３００は、ステップ１３０５から開始され、ステップ１３１０へと進み、そこで、重複解消モジュール２８４は、指紋レコード９００に記憶されているｉｎｏｄｅ番号９０５を読み出す（ロードする）。次に、重複解消モジュール２８４は、そのｉｎｏｄｅ番号９０５を使用して、データブロック４０６のｉｎｏｄｅ３００をロードする。複数の指紋レコードのｉｎｏｄｅをロードしつつ、シーケンシャルディスクアクセスを可能にするため、手順１３００の前に、又は手順１３００において、指紋データベース２８８は、ｉｎｏｄｅ番号９０５、及びｆｂｎ９１０に従ってソートされる場合がある。ステップ１３１５において、重複解消モジュール２８４は、ｉｎｏｄｅ３００からメタデータを取り出し（読み出し）、指紋レコード９００に記憶されているメタデータと照合する。

【００６９】

例えば、ステップ１３２０では、生成番号３２４、９１５を比較し、指紋レコード９００の生成後に、ｉｎｏｄｅ３００が削除され、再利用された（すなわち、異なっている）か否かが判定される。そうであれば、指紋レコード９００は古くなっているので、手順１３００はステップ１３４０へ進み、そこで、指紋レコード９００は削除される場合がある。そうでなければ、手順１３００はステップ１３２５へ進み、そこで、ＣＰカウント３２２、９２０を比較し、指紋レコード９００の生成後に、ｉｎｏｄｅ３００が変更されたか否か（その結果、複数のＣＰカウントが発生する）が判定される。そうであれば、指紋レコード９００は古くなっており、このｉｎｏｄｅ３００が複数回変更されていれば、ｉｎｏｄｅ３００に対応する他の指紋レコードも、同様に古くなっている場合がある。従って、ステップ１３３０では、このｉｎｏｄｅ３００に対して最新の指紋レコード９００だけを保持し、他の指紋レコード（低いＣＰカウントを有する）は全て削除する場合がある。複数の指紋レコードのＣＰカウント間の比較を容易にするために、指紋データベース２８８は、まず、ｉｎｏｄｅ番号９０５によってソートされ、次に、ＣＰカウント９２０によってソートされる場合がある（複数の指紋レｋ－ドが同一のｉｎｏｄｅ番号を共有している場合）。

【００７０】

さらに、データコンテナは、切り詰められる場合がある（例えば、データコンテナ中の所与のオフセット以降のデータを全て削除することにより）。データコンテナの切り詰めは、関連ｉｎｏｄｅ３００に記憶されるメタデータに影響を与えないことがある。従って、ステップ１３３５において、手順１３００は、指紋８００のサンプルデータ８１０を使用して、データブロック４０６との間の整合性（すなわち、キューオフセットにおいてデータが相違するか否か）を確認する。サンプルデータ８１０（データブロック４０６内の１以上の所与のオフセットから最初にコピーされたもの）が、データブロック４０６のその所与のオフセットに現在記憶されているデータと相違する場合、手順１３００はステップ１３４０へと進み、そこで、指紋レコード９００を削除する場合がある。先行するデータブロックにあるデータコンテナも、切り詰められる場合がある（例えば、データコンテナに関連するｉｎｏｄｅ３００のサイズフィールドに記憶される値を減らすことにより）。従って、ステップ１３５０において、手順１３００は、指紋レコード９００のｆｂｎ９１０を関連ｉｎｏｄｅ３００のサイズ３１４と比較する。サイズ３１４がｆｂｎ９１０よりも小さかった場合、手順１３００はステップ１３４０へと進み、そこで指紋レコード９

００は削除される場合がある。そして、手順１３００はステップ１３４５において終了する。

【００７１】

例えば、古くなった指紋の削除手順１３００は、定期的に実施され、及び／又は所定の条件（例えば、ＵＩ２４５から受信される管理者コマンド等）に応じて実施される。また、手順１３００は、指紋データベース２８８が所定の閾値サイズを超えたときに、それに応じて実施される場合もある。例えば、複数のデータブロックが繰り返し書き込まれ、及び／又は変更される際に、指紋データベース２８８における指紋レコードの数は、ストレージシステムにおけるデータブロックの最大数を上回ることがある。なぜなら、指紋レコードの多くが、古くなっているからである。手順１３００は、例えば、指紋レコードの数がデータブロックの最大数の１２０％を超えたときに実施される場合がある。

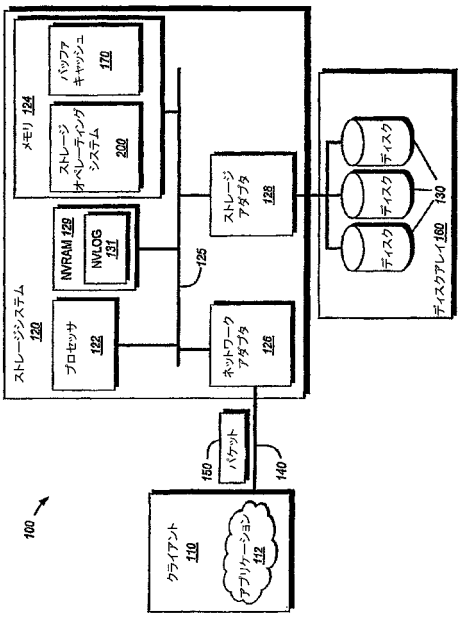
【００７２】

代替実施形態によれば、古くなった指紋の削除手順１３００は、重複指紋識別・データ重複解消手順１１００の一部として実施される場合もある。例えば、手順１３００は、手順１１００のステップ１１１０の直前に、重複解消モジュール２８４によって実施される場合がある。その結果、古くなった指紋、及びそれに関連するデータブロックを比較する際に消費されるストレージシステム処理リソースの量を減らせる場合がある。

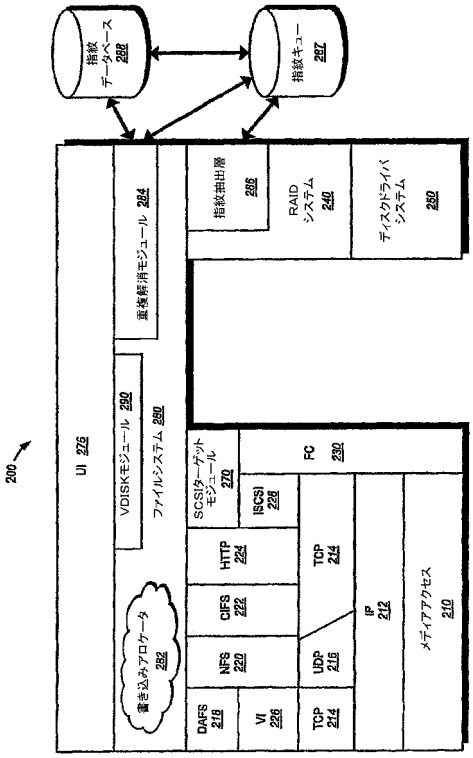
【００７３】

上記の説明は、本発明の特定の幾つかの実施形態に関するものである。しかしながら、当業者には明らかなように、それらの実施形態の利点の一部、又は全部を維持しながらも、記載した実施形態に対し、他にも変形、又は変更を施すことが可能である。当然ながら、本明細書に記載する手順、プロセス、及び方法は、他の順序で実施される場合もある。例えば、手順１１００の指紋キューと指紋データベースの比較ステップ１１２０は、指紋キューと指紋キューの比較ステップ１１１０の後に実施される場合や、それと同時に実施される場合もある。同様に、指紋生成、重複識別、及び古くなった指紋の削除等のプロセス／手順は、同時に実施される場合もあれば、個別に（すなわち、任意の順序で）実施される場合もあれば、プロセス／手順の所定の順番で実施される場合もある。また、本発明の教示は、ソフトウェア（コンピュータ上で実行されるプログラム命令を含むコンピュータ読取可能媒体を含む）でも、ハードウェアでも、ファームウェアでも実施することができ、あるいはそれらの組み合わせによって実施される場合もある。本明細書の説明は、ファイルシステムを例として書かれているが、本発明は、ファイルシステム以外のストレージ、例えば、ＬＵＮ、及び／又はブロックベースのストレージにも使用することが可能である。従って、本明細書の説明は、単なる例として捉えるべきものであり、本発明の範囲を何ら制限するものではない。従って、添付の特許請求の範囲の目的は、そうした変形や変更も、本発明の思想、及び範囲に含めることにある。

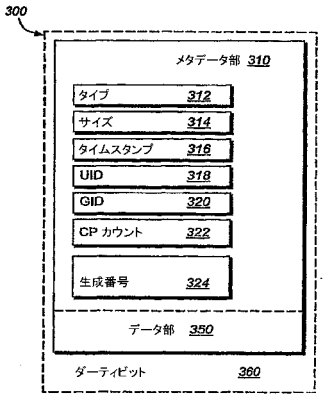
【図１】



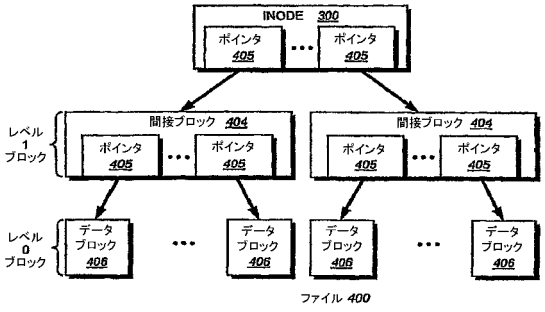
【図２】



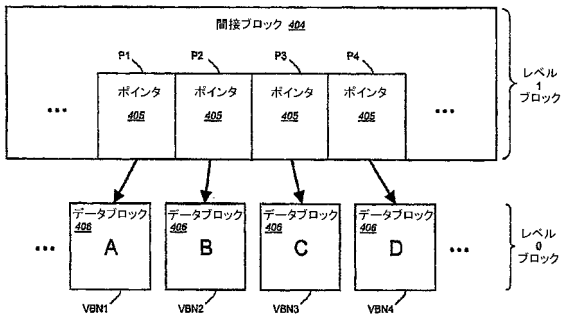
【図３】



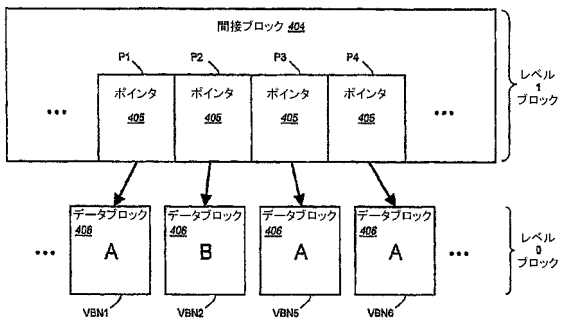
【図４】



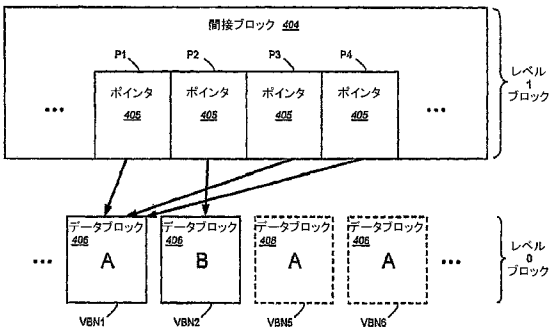
【図５】



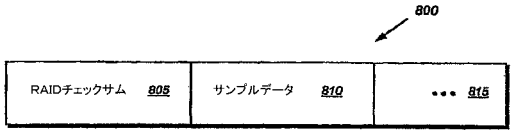
【図６】



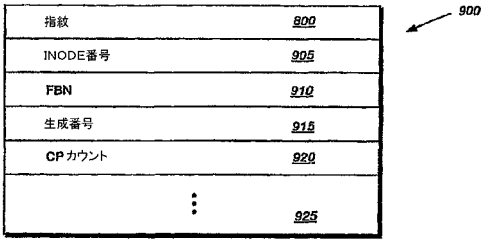
【図７】



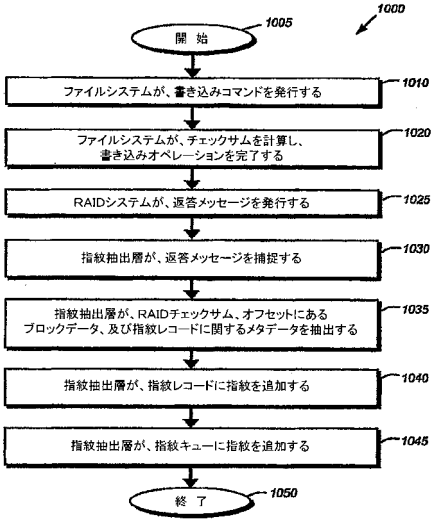
【図８】



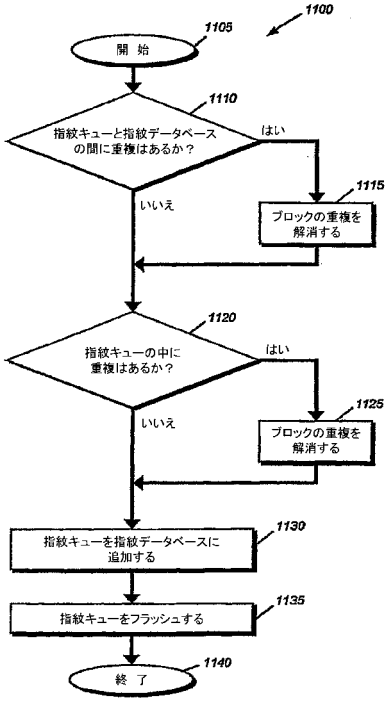
【図９】



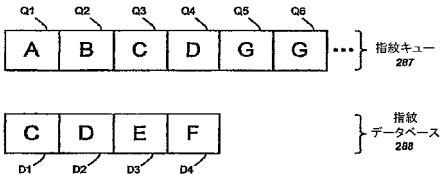
【図１０】



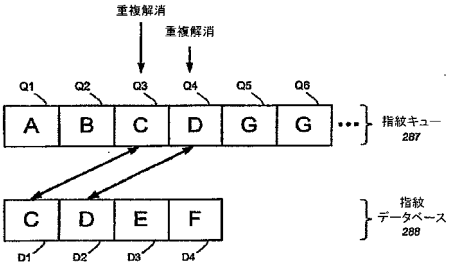
【図１１】



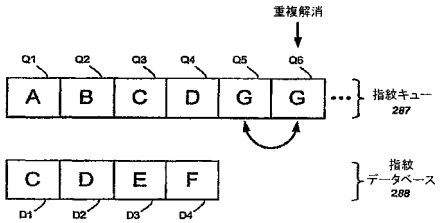
【図１２Ａ】



【図１２Ｂ】



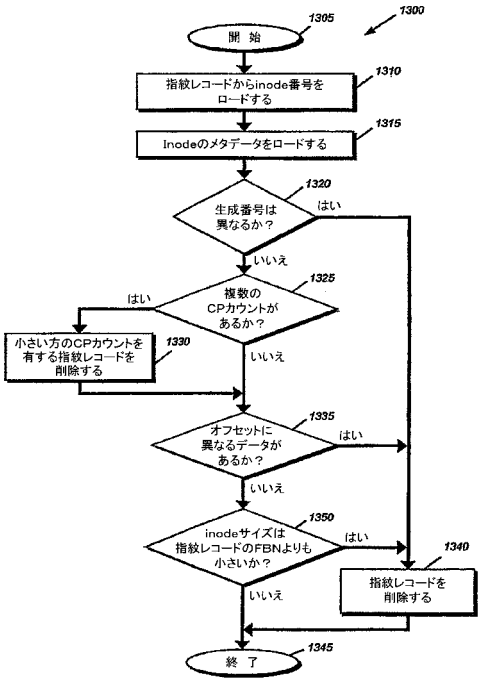
【図１２Ｃ】



【図１２Ｄ】


A
B
C
D
D
E
F
G
D1
D2
D3
D4
D5
D6
D7
D8
指紋
データベース
288

【図１３】


1300
1305
開 始
1310
指紋レコードからinode番号を
ロードする
1315
Inodeのメタデータをロードする
1320
生成番号は
異なるか？
はい
いいえ
1325
複数の
CPカウントが
あるか？
はい
いいえ
1330
小さい方のCPカウントを
有する指紋レコードを
削除する
1335
オフセットに
異なるデータが
あるか？
はい
いいえ
1350
inodeサイズは
指紋レコードのFBNよりも
小さいか？
はい
いいえ
1340
指紋レコードを
削除する
1345
終 了

フロントページの続き

(72)発明者　フォン，インフン
アメリカ合衆国カリフォルニア州９４０８９，サニーヴェール，イースト・ジャバ・ドライブ・４９５

審査官　桜井　茂行

(56)参考文献　特表２００３－５２４２４３（ＪＰ，Ａ）
国際公開第０１／０６１５６３（WO，Ａ１）
Bo Hong et al, "Duplicate Data Elimination in a SAN File System", Proceedings of the 21st IEEE / 12th NASA Goddard conference on mass stroage system and technologies, 米国, IEEE, ２００４年　4月, P. 103, [Online] http://www.ssrc.ucsc.edu/Papers/hong-mss04.pdf [2011/11/2取得]

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／００
Ｇ０６Ｆ　３／０６