(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5695040号
(P5695040)
(45)発行日　平成27年4月1日(2015.4.1)　(24)登録日　平成27年2月13日(2015.2.13)

(51)Int.Cl.　F I
*G06F 12/00 (2006.01)*　G06F 12/00 520J

請求項の数 36 (全 52 頁)

(21)出願番号　特願2012-517787(P2012-517787)
(86)(22)出願日　平成22年6月25日(2010.6.25)
(65)公表番号　特表2012-531675(P2012-531675A)
(43)公表日　平成24年12月10日(2012.12.10)
(86)国際出願番号　PCT/US2010/040058
(87)国際公開番号　WO2010/151813
(87)国際公開日　平成22年12月29日(2010.12.29)
審査請求日　平成25年6月25日(2013.6.25)
(31)優先権主張番号　61/269,633
(32)優先日　平成21年6月26日(2009.6.26)
(33)優先権主張国　米国(US)

(73)特許権者　511314120
シンプリヴィティ・コーポレーション
アメリカ合衆国マサチューセッツ州０１５８１－１７５６，ウエストボロー，テクノロジー・ドライブ　８
(74)代理人　100140109
弁理士　小野　新次郎
(74)代理人　100075270
弁理士　小林　泰
(74)代理人　100080137
弁理士　千葉　昭男
(74)代理人　100096013
弁理士　富田　博行
(74)代理人　100120112
弁理士　中西　基晴

最終頁に続く

(54)【発明の名称】ファイルシステム

(57)【特許請求の範囲】
【請求項１】
１つ又は複数のコンピュータストレージデバイスでのファイルの命名及び記憶のためのコンピュータファイルシステムであって、
前記ファイルシステムは、オブジェクトストアにアクセスする名前空間ファイルシステムを含み、前記名前空間ファイルシステムは、メモリと、該メモリと通信するハードウェアプロセッサとを含み、前記プロセッサは、オブジェクトフィンガープリントを用いて前記オブジェクトストアにアクセスするためにプログラム命令を実行し、前記オブジェクトストアはファイル、データ及びメタデータをオブジェクトとして保持し、各オブジェクトは、該オブジェクトのコンテンツから導出され、前記オブジェクトストアにアクセスするために用いられるグローバルに一意なオブジェクトフィンガープリントを有し、
各ファイルのオブジェクトは、該ファイルの前記データのオブジェクト又はメタデータのオブジェクトに対するオブジェクトフィンガープリントのマッピングを含み、前記ファイルのオブジェクトは、該ファイル内の前記オブジェクトの前記フィンガープリントから導出された前記ファイルのオブジェクト自体のオブジェクトフィンガープリントを有し、
前記オブジェクトストアは、更に、
ファイルシステムのｉノード番号とオブジェクトフィンガープリントとのマッピングを含むｉノードマップのオブジェクトであって、該オブジェクトフィンガープリントは、前記ｉノード番号を一定とすることを可能にしつつ、前記ファイルのコンテンツに変更が生じると変更される、ｉノードマップのオブジェクトと、

各々がｉノード番号とファイル名とのマッピングを含む複数のディレクトリのオブジェクトと、
を含み、前記ｉノードマップのオブジェクトとディレクトリのオブジェクトとの各々は、それぞれのオブジェクトのコンテンツから導出した自分自身のオブジェクトフィンガープリントを有することを特徴とするファイルシステム。
【請求項２】
請求項１に記載のファイルシステムであって、オブジェクト参照は、前記オブジェクトフィンガープリントによってマッピングされることを特徴とするファイルシステム。
【請求項３】
請求項１又は２に記載のファイルシステムであって、前記ファイルのオブジェクトのマッピングは、線形リスト、ツリー構造又は間接指定テーブルを含むことを特徴とするファイルシステム。
【請求項４】
請求項１から３の何れか一項に記載のファイルシステムであって、前記ファイルのオブジェクトはルートオブジェクトを含み、前記ルートオブジェクトは、前記ファイルシステム内の前記オブジェクトの全てから導出された該ルートオブジェクト自体のオブジェクトフィンガープリントを有し、それにより、前記ファイルシステム内のあらゆるオブジェクトは前記ルートオブジェクトを通じてアクセス可能であることを特徴とするファイルシステム。
【請求項５】
請求項１から４の何れか一項に記載のファイルシステムであって、前記名前空間ファイルシステムは、仮想ファイルシステム層とブロックストレージ抽象化層の間の、ストレージスタックにおける層として提供されることを特徴とするファイルシステム。
【請求項６】
請求項２から５の何れか一項に記載のファイルシステムにおいて、前記オブジェクトストアは、オブジェクトフィンガープリント、オブジェクト位置及びオブジェクト参照カウントのインデックスを含むことを特徴とするファイルシステム。
【請求項７】
請求項６に記載のファイルシステムであって、前記オブジェクトストアのインデックスは、不揮発性メモリに記憶されたことを特徴とするファイルシステム。
【請求項８】
請求項１から７の何れか一項に記載のファイルシステムであって、前記フィンガープリントは、前記オブジェクトのコンテンツの暗号ハッシュダイジェストであることを特徴とするファイルシステム。
【請求項９】
請求項１から８の何れか一項に記載のファイルシステムであって、前記オブジェクトのサイズは可変であることを特徴とするファイルシステム。
【請求項１０】
請求項１から９の何れか一項に記載のファイルシステムであって、前記ファイルシステムはＰＯＳＩＸ規格準拠ファイルシステムであることを特徴とするファイルシステム。
【請求項１１】
請求項１から１０の何れか一項に記載のファイルシステムであって、前記オブジェクトストアは、更に、
各オブジェクトに対する参照カウントを含む位置インデックスを含み、概参照カウントは前記オブジェクトが参照される回数を示すことを特徴とするファイルシステム。
【請求項１２】
請求項１１に記載のファイルシステムであって、
読出し、書込み、削除及び参照カウント更新を含むオブジェクトアクティビティのトランザクションログ
を含むファイルシステム。

【請求項１３】
請求項４又は請求項４に従属する場合の請求項５から１２の何れか一項に記載のファイルシステムであって、ファイルシステムの何れかのオブジェクトのコンテンツの変更は前記ルートオブジェクトの変更を生じ、前記ルートオブジェクトにおける変更を追跡することは、ファイルシステムアクティビティの履歴を提供することを特徴とするファイルシステム。
【請求項１４】
請求項１から１３の何れか一項に記載のファイルシステムであって、
前記オブジェクトの生成及び記憶の１つ又は複数を実施するためのハードウェアアクセラレータ
を含むファイルシステム。
【請求項１５】
請求項１から１４の何れか一項に記載のファイルシステムであって、
スタック
を含み、前記オブジェクトストアは前記スタックの下部を構成し、前記ファイルシステムは前記スタックの上部を構成することを特徴とするファイルシステム。
【請求項１６】
請求項７又は請求項７に従属する場合の請求項８から１５の何れか一項に記載のファイルシステムであって、前記不揮発性メモリはソリッドステートディスクを含むことを特徴とするファイルシステム。
【請求項１７】
請求項１６に記載のファイルシステムであって、前記メモリはフラッシュメモリを含むことを特徴とするファイルシステム。
【請求項１８】
請求項１から１７の何れか一項に記載のファイルシステムであって、前記名前空間ファイルシステム及び前記オブジェクトストアは、デジタル電子回路、コンピュータハードウェア、ファームウェア、非一時的マシン読み取り可能ストレージデバイス内のコンピュータプログラム又はそれらの組み合わせのうちの１つ又は複数において実装されたことを特徴とするファイルシステム。
【請求項１９】
名前空間ファイルシステムによりオブジェクトストアにアクセスするステップであって、該オブジェクトストアは、ファイル、データ又はメタデータをオブジェクトとして保持し、各オブジェクトは、グローバルに一意であり、該オブジェクトのコンテンツから導出され、前記オブジェクトストアにアクセスするために用いられるオブジェクトフィンガープリントを有し、各ファイルのオブジェクトは、オブジェクトフィンガープリントの、該ファイルの前記データのオブジェクト又はメタデータのオブジェクトに対するマッピングを含み、前記ファイルのオブジェクトは、該ファイル内の前記オブジェクトの前記フィンガープリントから導出された該ファイルのオブジェクト自体のオブジェクトフィンガープリントを有する、ステップと、
ファイルシステムのｉノード番号とオブジェクトフィンガープリントとのマッピングを含むｉノードマップのオブジェクトを前記オブジェクトストアに維持するステップであって、該オブジェクトフィンガープリントは、前記ｉノード番号を一定とすることを可能にしつつ、前記ファイルのコンテンツに変更が生じると変更される、ステップと、
前記オブジェクトストアにおいてディレクトリのオブジェクトを維持するステップであって、各ディレクトリのオブジェクトはｉノード番号とファイル名とのマッピングを含む、ステップと
を含み、前記ｉノードマップのオブジェクト及びディレクトリのオブジェクトの各々は、それぞれの前記オブジェクトのコンテンツから導出した自分自身のオブジェクトフィンガープリントを有することを特徴とする方法。
【請求項２０】

請求項１９に記載の方法であって、オブジェクトフィンガープリントと前記オブジェクトの物理的位置とをマッピングするための位置インデックスを維持するステップを含む方法。
【請求項２１】
請求項２０に記載の方法であって、前記位置インデックスは、前記オブジェクトに対する参照カウントを含むことを特徴とする方法。
【請求項２２】
請求項２１に記載の方法であって、前記フィンガープリント、位置インデックス、ｉノードマップのオブジェクト、ファイルのオブジェクト及びディレクトリのオブジェクトは、ファイルシステムを構成することを特徴とする方法。
【請求項２３】
請求項１９から２２の何れか一項に記載の方法であって、
前記ファイルシステムのスナップショットを前記ｉノードマップのオブジェクトの前記フィンガープリントから生成するステップ
を含む方法。
【請求項２４】
請求項１９から２３の何れか一項に記載の方法であって、
前記ｉノードマップのフィンガープリントを別のオブジェクトストアに発行するステップ
を含む方法。
【請求項２５】
請求項１９から２４の何れか一項に記載の方法であって、
前記ｉノードマップのフィンガープリントを前記ファイルシステムのスナップショットとして用いて障害回復を行うステップ
を含む方法。
【請求項２６】
請求項１９から２５の何れか一項に記載の方法であって、前記ｉノードマップのオブジェクトは以前のｉノードマップのフィンガープリントを含むことを特徴とする方法。
【請求項２７】
請求項２６に記載の方法であって、前記以前のｉノードマップのフィンガープリントは、前記ファイルシステムのスナップショットの履歴を構成することを特徴とする方法。
【請求項２８】
請求項１９から２７の何れか一項に記載の方法であって、
前記オブジェクトストアにおいてオブジェクト名と物理的オブジェクト位置との位置インデックスを維持するステップ
を含む方法。
【請求項２９】
請求項２１又は請求項２１に従属する場合の請求項２２から２８の何れか一項に記載の方法であって、
前記ファイルシステムに対する変更の際に、前記ｉノードマップのオブジェクトの下のあらゆるオブジェクトの前記参照カウントを調節するステップ
を含む方法。
【請求項３０】
請求項２９記載の方法であって、調節する前記ステップは、あらゆる入力／出力トランザクションに関して行われて、継続的データ保護が提供されることを特徴とする方法。
【請求項３１】
請求項２９に記載の方法であって、調節する前記ステップは、周期的に、オンデマンドで、又は、特定のイベントにおいて行われて、スナップショットが生成されることを特徴とする方法。
【請求項３２】

請求項１９から３１の何れか一項に記載の方法であって、前記ファイルのオブジェクトのマッピングは、前記ファイルの前記コンテンツへのオフセットによってインデックス付けされ、線形リスト、ツリー構造又は間接指定テーブルを含むことを特徴とする方法。
【請求項３３】
請求項２１から３２の何れか一項に記載の方法であって、読出し、書込み、削除及び参照カウント更新を含む全てのオブジェクトアクティビティのトランザクションログを生成するステップ
を含む方法。
【請求項３４】
請求項１９から３３の何れか一項に記載の方法であって、
前記ファイルのオブジェクトを追加、修正又は削除し、ファイルの新しいオブジェクトフィンガープリントを生成するステップ
を含む方法。
【請求項３５】
請求項３４に記載の方法であって、
ハードウェアアクセラレーションを利用してオブジェクトの生成、圧縮及び暗号化のうちの１つ又は複数を行うステップ
を含む方法。
【請求項３６】
プロセッサに、請求項１９から３５の何れか一項に記載の方法の前記ステップを実行させるプログラムコードを含む、非一時的マシン読み取り可能ストレージデバイスに記憶されたコンピュータプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
[001]本発明は、コンピュータファイルシステムデータ構造、並びに、ファイルの命名及び格納のための方法及び装置に関する。
【背景技術】
【０００２】
[002]十分な機能を有するストレージソリューションは、生のディスク、ファイルシステム、スナップショット、ファイルバージョニング、圧縮、暗号化、内蔵容量最適化（例えば、データ重複排除）、監査及びタンパーレジスタンス等他のセキュリティ機能、障害回復のためのオフサイト位置への効率的なレプリケーション等を含む場合がある。これらの機能の多くは、別々の機器において達成され、これらの機器は次いで、経験豊富な技術者によって接続されなければならない。
【０００３】
[003]数テラバイト（ＴＢ）のデータ向けの、今日の技術によるそのようなストレージソリューションの構築は、結果として、１０万ドルものコストを容易に上回る可能性のあるマルチボックスソリューションとなり、そのような十分な機能を有するストレージソリューションが、多数の企業及び顧客にとっては使用できないものとなることが多い。
【０００４】
[004]このマルチボックス、アドホックなソリューションは、ストレージの基本的な側面ではなく、むしろ、ファイルシステムのアーキテクチャ及び実装が、他の技術開発についてきていないということである。例えば、大抵のファイルシステムアーキテクチャは、より高速なコンピュータ処理装置（ＣＰＵ）、フラッシュメモリ、及び、ネットワーク帯域幅、ディスク密度及びディスクアクセスレートの間の異なるバランスを十分に活用するほどに、発展してきていない。
【発明の概要】
【発明が解決しようとする課題】
【０００５】

[005]データアクセシビリティを、アクセス帯域幅とアドレス指定可能ストレージの比として定義する場合、データのアクセシビリティは低下しつつある。ストレージ密度は、ディスクへのアクセスより速く増大しつつあるので、所与のデータセットサイズでは、データにアクセスするために必要とされる時間は増大しつつある（且つ、よって、アクセシビリティの低減を引き起こす）。ストレージアーキテクチャへの影響は、以下の通りであり、即ち、データを格納した後、絶対に必要でない限り、データを移動するべきではない。この単純な観察事実は、データが読み込まれ、再び書き出されることが絶えず行われている現在のストレージアーキテクチャでは、何度も妨げられる。この結果、著しい余分な支出（例えば、１０チャネル、ＣＰＵ、電力、時間、管理）となる。

【課題を解決するための手段】

【０００６】

[006]本発明の一実施形態によれば、ファイルシステム提供され、このファイルシステムは、

デジタルに署名されたファイルシステムであって、該システムにおいてデータ、メタデータ及びファイルはオブジェクトであり、各オブジェクトは、グローバルに一意でありコンテンツ導出のフィンガープリントを有し、オブジェクト参照はフィンガープリントによってマップされる、デジタルに署名されたファイルシステムを含み、

ファイルシステムは、ファイルシステム内の全てのオブジェクトのフィンガープリントのマッピングを含むルートオブジェクトを有し、

ファイルシステムに対する変更は、ルートオブジェクト内の変更を生じ、ルートオブジェクト内の変更を追跡することは、ファイルシステムアクティビティの履歴を提供する。

【０００７】

一実施形態では、

ファイルシステムは、ファイルオブジェクトのフィンガープリントへのｉノード番号のマッピングを含むｉノードマップオブジェクトを含み、ｉノードマップオブジェクトのフィンガープリントは、ファイルシステムのスナップショットを含む。

【０００８】

本発明の別の実施形態によれば、格納されたオブジェクトをインデックス付けする方法のための実行可能プログラム命令を含むコンピュータ可読媒体が提供され、上記方法は、

データ、メタデータ及びファイルをオブジェクトとして提供するステップと、

オブジェクトごとにフィンガープリントを提供するステップであって、該フィンガープリントはグローバルに一意でありオブジェクトのコンテンツから導出される、ステップとを含み、

ファイルシステムルートオブジェクトが提供され、該オブジェクトは、ファイルシステム内の全てのオブジェクトのフィンガープリントのマッピングを含み、ファイルシステムに対する変更がルートオブジェクト内の変更を生じ、ルートオブジェクト内の変更を追跡することはファイルシステムアクティビティの履歴を提供する。

【０００９】

一実施形態では、この方法は、

ファイルシステムｉノードマップオブジェクトを提供するステップであって、該オブジェクトはファイルオブジェクトのフィンガープリントへのｉノード番号のマッピングを含む、ステップを含み、

ｉノードマップオブジェクトのフィンガープリントは、ファイルシステムのスナップショットを含む。

【００１０】

一実施形態では、この方法は、

ｉノードマップフィンガープリントを、別個のオブジェクトストア上の別のコンピュータシステムに発行するステップを含む。

【００１１】

一実施形態では、この方法は、

ｉノードマップフィンガープリントを、障害回復のために、ファイルシステムのスナップショットとして使用するステップを含む。

【００１２】

一実施形態では、

ｉノードマップオブジェクトは、以前のｉノードマップのフィンガープリントを含む。

一実施形態では、

以前のｉノードマップフィンガープリントは、ファイルシステムのスナップショットの履歴を含む。

【００１３】

一実施形態では、

オブジェクトは、参照カウントを有し、

ファイルシステムに対する変更があると、ｉノードマップオブジェクトの下のあらゆるオブジェクトのオブジェクト参照カウントを調節するステップである。

【００１４】

一実施形態では、

調節するステップは、あらゆるＩＯトランザクションにおいて行われて、継続的データ保護が提供される。

【００１５】

一実施形態では、

調節するステップは、周期的に、オンデマンドで又は特定のイベントにおいて行われて、スナップショットが生成される。

【００１６】

一実施形態では、

オブジェクトは、参照カウントを有し、

参照カウントに対する調節は、新しいデータコンテンツのみが格納されるように、データ重複排除のために利用される。

【００１７】

本発明の別の実施形態によれば、１つ又は複数のコンピュータストレージデバイス上でのファイルの命名及び格納のためのコンピュータファイルシステムが提供され、このシステムは、

名前空間ファイルシステムを含み、ファイル、データ及びメタデータはオブジェクトであり、各オブジェクトは、オブジェクトのコンテンツから導出されたグローバルに一意なフィンガープリントを有し、各ファイルオブジェクトは、ファイルのデータオブジェクト及びメタデータオブジェクトのうちの一方又は双方のためのオブジェクトフィンガープリントのマッピングを含み、ファイルオブジェクトは、ファイル内のオブジェクトのフィンガープリントから導出された該ファイルオブジェクト自体のオブジェクトフィンガープリントを有し、このシステムは、ファイルオブジェクトのフィンガープリントへのｉノード番号のマッピングを含む。

【００１８】

一実施形態では、

オブジェクト参照は、オブジェクトフィンガープリントによって定まる。

一実施形態では、

ファイルオブジェクトのマッピングは、線形リスト、ツリー構造又は間接指定テーブルを含む。

【００１９】

一実施形態では、

ファイルオブジェクトはルートオブジェクトを含み、このルートオブジェクトは、ファイルシステム内のオブジェクトの全てから導出された該ルートオブジェクト自体のオブジェクトフィンガープリントを有し、ファイルシステム内のあらゆるオブジェクトはルートオブジェクトを通じてアクセス可能である。

【００２０】
一実施形態では、
名前空間ファイルシステムは、仮想ファイルシステム層とブロックストレージ抽象化層の間の、ストレージスタックにおける層として提供される。
【００２１】
一実施形態では、このシステムは、
オブジェクトフィンガープリント、オブジェクト位置及びオブジェクト参照カウントのインデックスを含むオブジェクトストアを更に含む。
【００２２】
一実施形態では、
オブジェクトストアインデックスは、不揮発性メモリに格納される。
一実施形態では、
フィンガープリントは、オブジェクトコンテンツの暗号ハッシュダイジェストである。
【００２３】
一実施形態では、
オブジェクトサイズは可変である。
一実施形態では、
ファイルシステムはＰＯＳＩＸ準拠ファイルシステムである。
【００２４】
本発明の別の実施形態によればある方法が提供され、この方法は、
ファイルシステム内のデータオブジェクトのためのオブジェクトフィンガープリントを生成するステップであって、データオブジェクトはデータ又はメタデータを含み、オブジェクトフィンガープリントは、データオブジェクトのコンテンツから導出されたグローバルに一意なフィンガープリントを含む、ステップと、
ファイルオブジェクトのためのオブジェクトフィンガープリントを生成するステップであって、各ファイルオブジェクトは、ファイル内の複数のデータオブジェクトのフィンガープリントを含み、ファイルオブジェクトのフィンガープリントは、ファイルオブジェクトのコンテンツから導出されたグローバルに一意なフィンガープリントを含む、ステップと、
ファイルシステム内のオブジェクトフィンガープリントの全てのマッピングを含むルートオブジェクトを生成するステップと
を含む。
【００２５】
一実施形態では、この方法は、
オブジェクトごとに参照カウントを維持し、オブジェクトへの参照が追加又は削除されたときに、オブジェクトの参照カウントを更新するステップを含む。
【００２６】
一実施形態では、この方法は、
読出し、書込み、削除及び参照カウント更新を含むオブジェクトアクティビティのトランザクションログを生成するステップを含む。
【００２７】
一実施形態では、この方法は、
ファイル内のデータオブジェクトを追加、修正又は削除し、新しいファイルオブジェクトのフィンガープリントを生成するステップを含む。
【００２８】
一実施形態では、
ファイルオブジェクト又はデータオブジェクトのコンテンツが変更されるとき、変更をルートオブジェクトまで伝えるステップである。
【００２９】
一実施形態では、この方法は、

伝えるステップを、
Ｉ／Ｏトランザクションごとに、
周期的に、
オンデマンドで、
特定のイベントで、
のうち１つで行うステップを含む。
【００３０】
本発明の別の実施形態によればある方法が提供され、この方法は、
複数のデータオブジェクトを提供するステップであって、各データオブジェクトはデータ又はメタデータを含み、各データオブジェクトは、グローバルに一意であり該データオブジェクトのコンテンツから導出されたフィンガープリントを有する、ステップと、
複数の関連付けられたデータオブジェクトのための複数のデータオブジェクトのフィンガープリントを含むファイルオブジェクトを生成し、グローバルに一意でありファイルオブジェクトのコンテンツから導出されるファイルオブジェクトのフィンガープリントを生成するステップと、
ファイルオブジェクトのフィンガープリントへのｉノード番号のインデックスを維持するステップと
を含む。
【００３１】
一実施形態では、この方法は、
オブジェクトフィンガープリント及びオブジェクトの物理位置をマッピングするための位置インデックスを維持するステップを含む。
【００３２】
一実施形態では、
位置インデックスは、オブジェクトのための参照カウントを含む。
一実施形態では、
フィンガープリント及びインデックスは、ファイルシステムを含む。
【００３３】
一実施形態によれば、
プロセスによって実行されるとき、方法の請求項２７のステップを行うプログラムコード手段を含むコンピュータプログラム製品が提供される。
【００３４】
本発明の別の実施形態によれば、
格納されたオブジェクトをインデックス付けする方法のための実行可能プログラム命令を含むコンピュータ可読媒体が提供され、上記方法は、
グローバルに一意であり、データ及びメタデータのオブジェクトのコンテンツから導出されたフィンガープリントを生成するステップと、
データ及びメタデータのうちの一方又は双方のオブジェクトの複数のフィンガープリントを含むファイルオブジェクトを生成し、ファイルオブジェクトのフィンガープリントを生成するステップであって、ファイルオブジェクトのフィンガープリントは、グローバルに一意でありファイルオブジェクトのコンテンツについて導出された、ステップと、
データ、メタデータ及びファイルのオブジェクトのフィンガープリントの全てのマッピングを含むルートオブジェクトを生成するステップと
を含む。
【００３５】
本発明の別の実施形態によれば、
データ、メタデータ及びファイルへのアクセスを提供する物理プロセッサ及びストレージデバイスが提供され、
データ、メタデータ及びファイルはオブジェクトであり、各オブジェクトは、グローバルに一意でありコンテンツ導出のフィンガープリントを有し、オブジェクト参照はフィン

ガープリントによってインデックス付けされ、
　インデックス付けは、ファイルオブジェクトのフィンガープリントへのｉノード番号のマッピングを含む。
【００３６】
　本発明の別の実施形態によれば、
　データオブジェクト、及び、ファイルオブジェクトを含むデータオブジェクトの集まりを命名し格納するための処理及びストレージの装置であって、各データオブジェクトはデータ又はメタデータを含み、各オブジェクトは、コンテンツベースのグローバルに一意なフィンガープリントを該オブジェクトのオブジェクト名として有し、ファイルオブジェクトは、データオブジェクト名の集まりであり、該ファイルオブジェクト自体のコンテンツベースのグローバルに一意なフィンガープリントを該ファイルオブジェクトのファイルオブジェクト名として有する、処理及びストレージの装置と、
　２つの層を有するファイルシステムであって、これら２つの層は、
　　オブジェクト名及び物理オブジェクト位置のマッピングインデックスを含むオブジェクトストア層と、
　　ファイルオブジェクトごとのデータオブジェクト名のマッピングインデックスを含む名前空間層と
を含む、２つの層を有するファイルシステムと
が提供される。
【００３７】
　一実施形態では、
　名前空間層は、ファイルオブジェクト名へのｉノード番号のマッピングインデックスを含む。
【００３８】
　一実施形態では、
　オブジェクトストア層は、オブジェクトごとの参照カウントを含む。
　一実施形態では、
　オブジェクト名は、オブジェクトのコンテンツの暗号ハッシュダイジェストである。
【００３９】
　一実施形態では、
　システムは、オブジェクトの命名、圧縮及び暗号化のうち１つ又は複数のために機能するハードウェアアクセラレーション装置を含む。
【００４０】
　一実施形態では、
　オブジェクトストア層は、ファイルシステム内の全てのオブジェクトのグローバルインデックスを含み、グローバルなオブジェクトのインデックスのための主キーはオブジェクト名であり、オブジェクト名は、オブジェクトのコンテンツの暗号ハッシュダイジェストである。
【００４１】
　[007]本発明は、詳細な説明を以下の図面と共に参照することによって、より十分に理解されよう。
【図面の簡単な説明】
【００４２】
【図１】[008]オペレーティングシステムカーネル空間に統合された本発明の一実施形態を例示する概略ブロック図である。
【図２】[009]オブジェクトストアが様々な物理媒体上でホストされることを可能とする、オブジェクトストアの一実施形態の主なコンポーネントの概略ブロック図である。
【図３】[0010]重要な機能性を、オブジェクトストア設計に影響を与えることなくこの機能性が様々な方法で実装されることを可能としながら抽象化することが可能な、オブジェクトストアの一実施形態の概略ブロック図であり、実装は、純粋なソフトウェアソリュー

ションから、ハードウェアアクセラレーションを使用するものまで及ぶ場合がある。
【図４】[0011]統合ファイルシステムの基本構成要素として、コンストラクト（「ｈノード」）に共にグループ化されたオブジェクトのセットの一実施形態の概略ブロック図である。
【図５】[0012]ファイル、ディレクトリ及びｉマップ等、ファイルシステムによって必要とされるような他のデータ構造に特化されうる、ｈノードの一実施形態の概略ブロック図である。
【図６】[0013]ファイルシステムに対する変更がどのように追跡され、経時的に維持されるか、並びに、使用された技術がどのように自然に空間効率、不変性及びセキュリティの結果となるかを例示する、一実施形態の概略ブロック図である。
【図７】[0014]オブジェクトのためのグローバル一意名を提供しながら、圧縮、暗号化及び位置独立を透過的に扱うことができるオブジェクトの、一実施形態の概略ブロック図である。
【図８】[0015]ＦＵＳＥ、ユーザ空間ファイルシステムにより、ユーザ空間内で実装された本発明の代替実施形態の概略ブロック図であり、ＦＵＳＥは、ユーザ空間内でファイルシステムの構築を可能とする、ライブラリ及びカーネルモジュールのオープンソースセットである。
【図９】[0016]インデックス付けの発明の一実施形態により行われる、様々なインデックス付けオペレーションを例示する概略ブロック図である。
【図１０Ａ】[0017]この発明で使用されうるデータ構造の様々な実施形態を例示する図である。
【図１０Ｂ】この発明で使用されうるデータ構造の様々な実施形態を例示する図である。
【図１０Ｃ】この発明で使用されうるデータ構造の様々な実施形態を例示する図である。
【図１０Ｄ】この発明で使用されうるデータ構造の様々な実施形態を例示する図である。
【図１１】[0018]この発明の一実施形態によるルックアップオペレーションを例示する概略ブロック図である。
【図１２】[0019]この発明の一実施形態による挿入オペレーションを例示する概略ブロック図である。
【図１３】[0020]この発明の一実施形態による削除オペレーションの概略ブロック図である。
【図１４】[0021]この発明の一実施形態による更新オペレーションの概略ブロック図である。
【図１５Ａ】[0022]この発明の一実施形態によるフリー消去ブロックを生成するためのランダム読出し処理を例示する概略ブロック図である。
【図１５Ｂ】この発明の一実施形態によるフリー消去ブロックを生成するためのランダム読出し処理を例示する概略ブロック図である。
【図１６Ａ】[0023]スキャベンジング処理に従ってフリー消去ブロックを生成する別の方法を例示する概略ブロック図である。
【図１６Ｂ】スキャベンジング処理に従ってフリー消去ブロックを生成する別の方法を例示する概略ブロック図である。
【図１７】[0024]この発明の実装を例示するための６層のビュー又はスタックを例示する概略ブロック図である。
【図１８】[0025]この発明の一実施形態で使用されるようなレコードエントリの概略図である。
【図１９】[0026]図１９Ａは、この発明の一実施形態によるカッコウハッシング（ｃｕｃｋｏｏ　ｈａｓｈｉｎｇ）の実装を概略的に例示する図である。図１９Ｂは、この発明の一実施形態によるカッコウハッシングの実装を概略的に例示する図である。図１９Ｃは、この発明の一実施形態によるカッコウハッシングの実装を概略的に例示する図である。図１９Ｄは、この発明の一実施形態によるカッコウハッシングの実装を概略的に例示する図である。図１９Ｅは、この発明の一実施形態によるカッコウハッシングの実装を概略的に

例示する図である。
【図２０】[0027]この発明の一実施形態による、各バケットが複数のレコードを保持する、複数のバケットの概略的例示の図である。
【図２１】[0028]この発明の一実施形態によるバケットのコンテンツの概略図である。
【図２２】[0029]この発明の一実施形態による、複数のダイ、消去ブロック、ページ及びバケットを有する物理フラッシュチップの一例を例示する概略ブロック図である。
【図２３】[0030]図２３Ａは、この発明の一実施形態によるデバイス管理層のあるコンポーネントを例示する図である。図２３Ｂは、この発明の一実施形態によるデバイス管理層のあるコンポーネントを例示する図である。
【発明を実施するための形態】
【００４３】
Ａ．従来技術の（レガシー）ファイルシステムの従来のファイルシステムデータ構造及び制限。
[0031]従来のファイルシステムは、幾つかの基本データ構造を有する。ユーザが見ることができるディレクトリ及びファイルに加えて、内部構造は、スーパーブロック、ｉノード、アロケーションマップ、及びトランザクションログを含む。
【００４４】
[0032]アロケーションマップは、ディスク上のどのブロックが使用中であるか、或いは使用中でないかを示すデータ構造である。これらのデータ構造は、ビットマップと同じくらい簡素になり、或いは、Ｂツリーと同じくらい複雑になりうる。アロケーションマップは大きくなり、ほとんどメモリに合わない可能性がある。新しいブロックの単純なアロケーションは、低いディスク性能という結果となるが、前述のメモリ制限を考えると、最適な配置には高度なアロケーションアルゴリズムを要する。
【００４５】
[0033]ディレクトリは、ファイル及び他のディレクトリの名前のリストであり、多数のファイルシステムでは、単に異なるように解釈される別のファイルタイプとして扱われる。内部的には、ディレクトリは、ファイル名／ｉノード番号の対のリストである。ファイルシステムは、ファイル名へのアクセスを望むとき、ディレクトリ内のファイル名、及び対応するｉノード番号を発見しなければならない。
【００４６】
[0034]ファイルは、命名されたデータの集まりである。ファイル名は、それが参照するｉノードと共に、ディレクトリ構造に格納にされる。多数のファイルシステムは、異なるファイル名が同じデータ（ｉノード）をポイントすることができる、リンクの概念をサポートする。
【００４７】
[0035]トランザクションログは、原子性、一貫性、独立性及び持続性（ＡＣＩＤ）特性に従って、ファイルシステムを一貫性のあるものに保つために使用される。多数のファイルシステムは、メタデータ一貫性を保証するが、データについての異なるサービスレベルアグリーメント（ＳＬＡ）を有するようになる。
【００４８】
[0036]スーパーブロックは、ディスク又は永続媒体上の既知の位置に存在する小さいデータ構造である。スーパーブロックから、ｉノードテーブルのサイズ及び位置、アロケーションマップ、ルートディレクトリ等、ファイルシステムに関連する全ての他のデータ構造が発見されうる。ファイルシステムが搭載されるとき、最初にアクセスされるものがスーパーブロックである。安全性の理由のため、スーパーブロックはしばしば、ディスク上の様々なポイントでレプリケートされる。
【００４９】
[0037]恐らく、最も基本的なデータ構造は、ｉノード（「インデックスノード」）である。多数のファイルシステムに共通して、ｉノードは、ファイル等、コンテンツのための基本コンテナであるデータ構造である。ｉノード自体は、ファイル名を含まず、ファイル

名はディレクトリに格納される。ｉノードは、ディスク常駐データ構造（ｉノードテーブル）へのインデックスを示す整数によって識別される。テーブル内の各ｉノードエントリは、このファイルについて、ディスク上のどこでコンテンツが発見されうるかを記述する。この「マップ」は、線形リスト、間接指定テーブル（ｉｎｄｉｒｅｃｔｉｏｎ　ｔａｂｌｅ）、様々なツリータイプを含む、様々な形式を取ることができ、その各々は様々な速度／空間トレードオフを有する。重要なことは、マップが、論理ブロック番号（ＬＢＮ）等、物理又は論理アドレス指定を使用することである。ＬＢＮがどのディスクのために意図されるかを知る場合にのみ、ＬＢＮは意味をなす。

【００５０】

[0038]上記から、レガシーファイルシステムが、何か（コンテンツ）及びどこか（データの配置）についての厳重な制御を有することは、明らかになるべきである。この何か及びどこかが互いに混ざり合うこと、概して履歴の産物は、結果として、現代のストレージのニーズに届くことが困難であるアーキテクチャをもたらす。

【００５１】

Ｂ．本発明のファイルシステムの新規のデータ構造及び特徴。

[0039]本発明の様々な実施形態によれば、新しいデータ構造が、新しいタイプのファイルシステムを埋め込むために提供される。このファイルシステムは、他のファイルシステムと並んで存在し、働くことができ、レガシーファイルシステム及び既知のユーザレベルユーティリティと互換性がある。しかし、本発明の新しいデータ構造は、レガシーファイルシステムでは達成できない利益を提供する。これらの利益は、限定されないが、以下のうち１つ又は複数を含む。

【００５２】

・物理又は論理ブロックアドレス指定に頼らない、ファイルの命名及び格納のための抽象化のレベルを提供すること、

・データオブジェクトのコンテンツから導出されたグローバル一意フィンガープリントをオブジェクト名として利用することであって、各データオブジェクトはデータ又はメタデータを備えること、

・名前空間ファイルシステム内でオブジェクトフィンガープリントを利用することであって、全ての内部データ構造はオブジェクトであり、全てのオブジェクト間参照がオブジェクトフィンガープリントによって定義されることを可能とすること、

・「ｈノード」構造と呼ばれる新しいデータ構造を提供することであって、ファイルｈノードは、ファイル内の全てのデータオブジェクトフィンガープリントのマッピング構造であり、それ自体は、ファイルオブジェクトのコンテンツから導出されたグローバル一意フィンガープリントを有するオブジェクトであること、

・同様に、ルートｈノード（オブジェクト）は、ファイルシステム内の全てのオブジェクトフィンガープリントのマッピング構造であり、ファイルシステムに対するいかなる変更もルートオブジェクト内の変更を生じ、且つ、ルートオブジェクトに対する変更を追跡することがファイルシステムアクティビティの履歴を提供するようにすること、

・ファイルオブジェクトフィンガープリントへのｉノード番号のマッピングを備える、ｉノードマップオブジェクト（ｉマップ）を提供し、ファイルコンテンツが変化するときにオブジェクト名（フィンガープリント）が変化しながら、ｉノード番号が一定であることを可能にすることであって、ｉノードマップオブジェクトのフィンガープリントは、ファイルシステムのスナップショットを備えること。

【００５３】

[0040]開示された実施形態では、オブジェクト名、即ち、オブジェクトフィンガープリントは、オブジェクトのコンテンツの暗号ハッシュダイジェストである。これは、オブジェクト名がオブジェクトコンテンツのフィンガープリントとしてグローバル一意且つ識別可能となることを可能とする。フィンガープリントは、オブジェクトよりかなり、例えば、１００倍、１０００倍以上も小さく、よって、フィンガープリントを操作することはしばしば、下にあるコンテンツを操作することよりも高速且つ容易である。

【００５４】
[0041]データオブジェクトの組み合わせ又は集まりを、同じくオブジェクトフィンガープリントであるオブジェクト名を有するオブジェクトである、ｈノードとして提供することによって、ｈノードはグローバル一意であり、ｈノード内に含まれたデータオブジェクトのコンテンツから導出される。いかなる変更（例えば、追加、削除、メタデータ変更、読出し）も、ファイルシステムｈノードフィンガープリントが変更される結果となる。ｉマップに対する変更を追跡することによって、全てのファイルシステムアクティビティの完全な履歴が提供される。
【００５５】
[0042]本発明に固有のものは、ｉノードマップオブジェクト（別名ｉマップ）であり、ｉノード番号をオブジェクトフィンガープリントに変換するものである。これは、名前空間ファイルシステムがｉノード番号に対処することを可能とし、多数のユーザレベルアクティビティがｉノード番号を参照するので、中心のものである。ｉノード番号へのフィンガープリント（オブジェクト名）のｈノードマッピングは、従来の静的なｉノードテーブルを越えて、追加の間接指定（ｉｎｄｉｒｅｃｔｉｏｎ）（又は仮想化）の層を提供する。この間接指定テーブルを使用することによって、ｉノード番号は一定であることができるが、ｉノードに対応するファイルが変化するとき、関連付けられたオブジェクト名（フィンガープリント）は変化することができる。ｉマップ自体がオブジェクトであるので、ファイルシステムが修正されるとき、その名前もまた変化するようになる。ｉマップのフィンガープリントは本質的に、ファイルシステムの完全な「スナップショット」である。スナップショットフィンガープリントを有した後、ファイルシステム（書込み可能なスナップ）に取り組むことを継続し、且つ、今後の使用のため（例えば、障害回復のため）にそれを記憶していることが可能である。スナップショットフィンガープリントを、別個のオブジェクトストア上に位置する別のシステムに発行することもできる。他のオブジェクトストアは、スナップショットデータ（オブジェクト）の全てを十分にホストしない場合があるが、記載された機構はなお十分に一貫性があり、使用可能である。
【００５６】
[0043]本発明のこれら及び他の利益は、本発明の様々な実施形態を参照して、より詳細に後述される。
[0044]カーネル空間及び次いでユーザ空間の両方で実装される、新しいファイルシステムの具体例を記載する前に、本実施形態で利用される様々なコンポーネントのより全体的な説明が定義される。
【００５７】
オブジェクトストア
[0045]オブジェクトストアは、本実施形態では、不透明（ｏｐａｑｕｅ）データ（オブジェクト）のフラットな集まりである。各オブジェクトは固有であり、参照カウント（名前空間ファイルシステムによって参照される回数）を有する。オブジェクトの名前は、オブジェクトのコンテンツの暗号ハッシュであり、即ち、コンテンツを変更すると、名前は変化しなければならない。
【００５８】
[0046]任意の十分に強力な暗号ハッシュが、オブジェクト名（フィンガープリント）を生成するために受け入れ可能である。例として、セキュアハッシュアルゴリズム（ＳＨＡ）ハッシュ関数は、国家安全保障局（ＮＳＡ）によって設計され、ＮＩＳＴによって米国連邦情報処理規格として発行された、暗号ハッシュ関数のセットである。ＳＨＡ－Ｉは、既存のＳＨＡハッシュ関数の最もよく確立されたものであり、幾つかの広く使用されたセキュリティアプリケーション及びプロトコルで採用される。
【００５９】
[0047]実際には、オブジェクトサイズは典型的には２の累乗であり、５１２バイト（$2^9$）から最大１ＭＢ（$2^{20}$）以上に及ぶが、オブジェクトのサイズに対するアーキテクチャ上の制約はない。

【００６０】
[0048]典型的なオブジェクトサイズは、２ＫＢ（$2^{11}$バイト）である。８ＴＢ（$2^{43}$バイト）のファイルシステムでは、$2^{32}$ものオブジェクトであり、即ち、およそ２０億ものオブジェクトである。インデックス内の各オブジェクトのエントリは約３２（$2^{5}$）バイトであり、そのため、オブジェクトインデックスは、密に詰まっていると仮定すると、$2^{37}$、即ち１２８ＧＢ、即ち全ファイルシステム空間の約２％である。他のオブジェクトサイズは、適用性又は一般性を失わずに使用されうる。
【００６１】
[0049]オブジェクトは、オブジェクトのユーザに対して透過的に圧縮且つ暗号化される。オブジェクト名は、クリーンな非圧縮データ（及び任意選択のソルト）に基づく。オブジェクトに実際に格納されるものは、（クリーン）、（クリーン圧縮）、（クリーン、圧縮暗号化）又は（クリーン暗号化）データのうち１つである。
【００６２】
[0050]オブジェクトは典型的には、クリーンデータのみで読み／書きされ、圧縮／暗号化は、オブジェクトストアに内部で起こる。
[0051]強力な暗号ダイジェストを使用することは、オブジェクトがグローバル一意且つ一貫性のある名前を有することを可能とする。同じ名前を有する２つのオブジェクトは、実際上は、同じコンテンツを有するようになる。
【００６３】
名前空間
[0052]名前空間ファイルシステムは、本実施形態では、ファイル、ディレクトリ構造、リンク、スーパーブロック等を有する。
【００６４】
[0053]名前空間ファイルシステムは、データを直接的には含まず、代わりに全てのデータがオブジェクトに格納される。オブジェクトは比較的小さく、しばしばより大きいデータ構造が必要とされる。オブジェクトを集約する構造は、ｈノードと呼ばれる。
【００６５】
[0054]実際の方法としては、Ｕｎｉｘ(登録商標)又はＬｉｎｕｘ(登録商標)環境にプラグインするファイルシステムは、ｉノード番号をエクスポーズする必要がある。ｉノードは、ファイルを一意に識別する数である。
【００６６】
ｈノード
[0055]ｈノードは、本実施形態では、ファイル等、コンテンツを共に結び付けるデータ構造である。時として、コンテンツは大変大きくなる可能性があり（多ＧＢ）、ディスク又は永続媒体上で連続的に適合しない。そのコンテンツは分割され、複数の個別のユニットとして格納される。従来のファイルシステムの場合、これはディスク上のブロックとなる。本発明では、これらがオブジェクト名である。ｈノードは、マッピング構造内のオブジェクト名の全てのリストを保持する。線形リストは、そのようなマッピング構造の一例であるが、より複雑な間接指定テーブルもまた可能である。
【００６７】
[0056]ｈノードとｉノードの間には、２つの主な違いがある。第１に、ｈノードはオブジェクトのコンテンツを識別するオブジェクト名（フィンガープリント）を使用するのに対し、ｉノードは物理又は論理ブロックアドレス指定を使用することである。第２に、ｈノードは、明確に定義されたグローバル一意名（そのコンテンツのハッシュ）を有することである。後述の好ましい実施形態では、ｈノード名は、オブジェクトコンテンツのハッシュ及びソルトである。
【００６８】
ｉノードマップオブジェクト（ｉマップ）
[0057]本発明に固有のものは、ｉマップであり、ｉノード番号をオブジェクトフィンガープリント（名前）に変換するものである。このフィンガープリントは典型的にはｈノー

ドであり、ｈノードは、コンテキストに応じて様々な方法で解釈される。これは、名前空間ファイルシステムの残りがｉノード番号に対処することを可能とし、多数のユーザレベルユーティリティがそのようなコンストラクトを見ることを必要とするので、必須のものである。ある意味で、これは、従来の静的なｉノードテーブルを越えて、追加の間接指定（又は仮想化）の層を提供する。

【００６９】

[0058]この間接指定テーブルを使用することによって、ｉノード番号は一定であることができるが、ｉノードに対応するファイルが変化するとき、関連付けられたオブジェクト名（フィンガープリント）は変化することができる。ｉマップ自体がオブジェクトであるので、ファイルシステムが修正されるとき、その名前もまた変化するようになる。

【００７０】

[0059]従来のファイルシステムでは、ルートディレクトリは既知のｉノード番号にあり、ｉマップの場合もまた、そうである。

[0060]ｉマップのフィンガープリントを有する場合、本質的に、ファイルシステムの完全な「スナップ」を有する。このフィンガープリントの下のあらゆる可視のオブジェクトの参照カウントをバンプすることは、スナップをロックし、他のファイルシステムアクティビティにかかわらず、スナップが削除されることを妨げる。

【００７１】

[0061]スナップフィンガープリントを有した後、ファイルシステム（書込み可能なスナップ）に取り組むことを継続し、且つ、今後の使用のために（恐らく、障害回復の目的で）それを記憶していることが可能である。スナップフィンガープリントを、別個のオブジェクトストア上に位置する別のシステムに発行することもできる。他のオブジェクトストアに気付く範囲内で、あるオブジェクトストアが特定のフィンガープリントの読出し要求を解決することができない場合、そのオブジェクトストアは、その要求をそれらのストアに転送してもよい。よって、スナップのフィンガープリントは、そのオブジェクトストアがスナップのデータ（オブジェクト）の全てを十分にホストしない場合のあるシステムに移動することがあるが、上記の機構を介して、なお十分に一貫性があり、使用可能である。

【００７２】

スーパーブロック

[0062]スーパーブロックは、本実施形態では、オブジェクトストアが永続媒体上で存続するとき、使用されるデータ構造である。スーパーブロックは、既知の（１つ又は複数の）位置内で存続する。スーパーブロックは、アロケーションマップ、ｉマップ、オブジェクトプール、インデックス及び他の構造が媒体上のどこで存続するかを記述する。オブジェクトストアは、常にグローバル一意識別子（ＧＵＩＤ）を有し、ＧＵＩＤは、オブジェクトストアのその固有インスタンスを表現する。

【００７３】

[0063]オブジェクトストアが大きいオブジェクトプールに参加する場合、スーパーブロックもまた、より大きいプールのＧＵＩＤ、及び、全てのメンバのＧＵＩＤ、及び、メンバの関係（取り除かれた、レプリケートされた、消去符号化された等）を含む。

【００７４】

ファイル

[0064]ファイルコンストラクトは、本実施形態では、ｈノードから導出される。ファイルコンストラクトは、読出し、書込み、開く、閉じる等、ファイルに関する通常（例えば、ＰＯＳＩＸ（登録商標））のセマンティクスの全てを有する。

【００７５】

ディレクトリ

[0065]ディレクトリは、本実施形態では、ｈノードの特化されたバージョンである。ディレクトリは、（ｉノード番号、オブジェクト名）の対のマップを含む。線形リスト、ベクトル、又は他のより複雑な構造は、実装の例である。マップは少なくとも、ｈノードに

対してそれとして持続するために、直列化可能且つ非直列化可能でなければならない。マッピング構造に応じて、ランダムアクセスもまた可能である。

【００７６】

追跡

[0066]ファイルシステムは、通常の書込み、削除及び読出しにより修正される（読出しがアクセス時間を変化させることを認める）ので、そのファイルシステムを構成するオブジェクト及びｈノードもまた変化する。これは、結果としてルートハッシュの履歴となり、ルートハッシュの履歴は、大変細かい粒度では、継続的データ保護（ＣＤＰ）と呼ばれ、より粗い粒度では、スナップと呼ばれる。違いは、どのくらいの頻度でルートハッシュが取り込まれるか、のみにある。

【００７７】

[0067]システム内のあらゆるオブジェクトは、少なくとも１つのルートハッシュを通じてアクセス可能でなければならない。

[0068]本実施形態では、ｈノードとしてＨが書き込まれ、新しいｈノードＨ’が作成され、より多くの変化が生じる場合、場合によってはＨ”である。これらの変化は蓄積してもよいが、ある時点で、最後の変化がルートまで遡って伝わる。この保留中の入力／出力（ＩＯ）は、ファイルシステムが変化を蓄積し、且つ、変化ごとにルートまで伝わらないことを可能とする。どのくらいの頻度でこれが発生するかは、ポリシーに基づく。変化リストＨ－＞Ｈ’－＞Ｈ”の途中のオブジェクトのための参照カウントは、それに応じて対処され、ダングリング参照又は到達不可能オブジェクトがないようにしなければならない。

【００７８】

Ｃ．本発明の実施例（実装）

[0069]ここで図１を参照すると、図示されるものは、オペレーティングシステムカーネル１０１内の様々なストレージコンポーネントである。Ｌｉｎｕｘ(登録商標)環境をもとに描かれるが、この図は十分に一般的であるので、Ｗｉｎｄｏｗｓ（登録商標）、Ｓｏｌａｒｉｓ（登録商標）及び他のＵｎｉｘ(登録商標)クラスのオペレーティングシステム等、他のオペレーティングシステムに適用される。

【００７９】

[0070]ＰＯＳＩＸ（登録商標）１０４のスタイルのファイルシステムの一実施例が示され、但し、ＰＯＳＩＸ（登録商標）は、一般性を失わずに、ＲｅｓｉｅｒＦｓ、Ｅｘｔｓ、ｂｔｒｆｓ及びｚｆｓ等、任意の数のファイルシステムのうち任意のものでありうる。仮想ファイルシステム（ＶＦＳ）層１０３は、ファイルシステムの多数の共通機能を抽象化するために使用され、一貫性のあるインタフェース１６０をユーザ空間１００及び他のコンポーネントに提供する。ＶＦＳ１０３はまた、いかなるファイルシステムも使用しなければならない（ＶＦＳ１０３層によって認識されることを期待する場合）、明確に定義された「より低いエッジ」のインタフェース１５０をも有する。実際には、典型的には、並行して働く多数のファイルシステムがある。

【００８０】

[0071]ファイルシステムは、ブロックドライバ１０５によって実装された、ブロックストレージ抽象化の上に正常に位置する。ブロックストレージは、論理装置番号ＬＵＮローカルストレージデバイス１０９上にあってもよく、或いは、ｉＳＣＳＩプロトコルを使用してリモートＬＵＮ上にあってもよい。ブロックドライバ１０５もまた、オペレーティングシステム内で明確に定義されたインタフェースを有する。

【００８１】

[0072]この実施形態では、新しいファイルシステムは、カーネル内の他のファイルシステムと並んで働く。新しいファイルシステムは、軽量オブジェクトファイルシステム１０８の上にスタックされる、名前空間ファイルシステム１０７からなる。２つのコンポーネント間のインタフェース１５２は、ＡＮＳＩ　Ｔ－１０オブジェクト標準等、様々な業界標準オブジェクトインタフェースの任意のものであってもよい。

【００８２】
[0073]オブジェクトファイルシステム（オブジェクトストア）１０８は次には、パーティションで区切られ、共通して使用される機能のライブラリである、ダイジェスト、インデックス付け、圧縮、暗号化（ＤＩＣＥ）ライブラリ３１０が抜き出されるようになる。ライブラリ３１０は、ソフトウェアにおいて完全に実現されてもよく、或いは、様々なハードウェアアクセラレーション１１３の技術をうまく利用してもよく、その１つが例示される。
【００８３】
[0074]オブジェクトファイルシステム１０８は、生のＬＵＮ、ディスク上のパーティション、又は大きいファイルの上に位置してもよい、オブジェクトコンテナを作成する。オブジェクトファイルシステム１０８はまた、ｉＳＣＳＩ又は他のリモートアクセスブロックプロトコル（ＦＣｏＥは、もう１つの例である）等、プロトコルを使用して、ネットワークスタック１０６を介して、コンテナを参照してもよい。ネットワークファイルシステム（ＮＦＳ）１０２は、（インタフェース１５４を介して）ネットワークスタック１０６の上に位置し、ＮＦＳは、ＶＦＳ１０３に接続される。ネットワークスタック１０６は、インタフェース１６０を介してＬＵＮ１０９に、インタフェース１５９を介してクラウド１１０に接続される。
【００８４】
[0075]図２を参照すると、オブジェクトストア１０８は更に分解される。オブジェクトストア１０８は、バイナリの不透明オブジェクトを含み、その例は、Ｐ２０１、Ｑ２０２及びＲ２０３である。オブジェクトは様々なサイズであってもよいが、好ましい実装では、２の累乗である。オブジェクトは、コンテナ内であるオフセットにて存在し、このオフセットは、バイトオフセット、又は、最小オブジェクトサイズを法とするオフセット（即ち、最小オブジェクトが５１２バイトである場合、バイトオフセットを得るために、オフセットに５１２が乗じられる）であってもよい。
【００８５】
[0076]各オブジェクトは、名前（フィンガープリント）を有し、名前（フィンガープリント）は、オブジェクトのコンテンツ全体の暗号ダイジェスト（ハッシュ）に、あるサイト特有のソルトを加えたものである。図２では、オブジェクト名は、Ｈ（Ｐ）、Ｈ（ｑ）及びＨ（ｒ）によって示される。
【００８６】
[0077]インデックス構造２０４は、オブジェクト名、オブジェクト位置、及びオブジェクト参照を追跡し続ける。オブジェクトの参照は、オブジェクトが書き込まれるたびに増分される。名前空間ファイルシステム１０７は、同じオブジェクトの多数のコピーであると思うものを生成してもよく、オブジェクトストア１０８は、１つのみを格納するが、名前空間がいくつを有すると実際に考えるかを追跡し続ける。
【００８７】
[0078]オブジェクトストア１０８は、幾つかのインタフェースクラスを有する。読出し、書込み、削除インタフェース１５２ａは、オブジェクトに対してまさにそれを行う。これに関連するオブジェクト削除は、実際にはオブジェクトの参照カウントの減分である。オブジェクトストア内部のオブジェクトのためのストレージは、参照カウントが０になるときにのみ、解放される。
【００８８】
[0079]インデックス付けオペレーション１５２ｂは、名前によるオブジェクトの列挙、参照カウント調節、及び、名前によるオブジェクトのルックアップを可能とする。
[0080]オブジェクトストア１０８は、トランザクションのセマンティクス（ＡＣＩＤ特性）を有し、トランザクション境界は、トランザクションオペレーション１５２ｃを通じて管理される。これは、保留中のトランザクションをリストすることに加えて、トランザクションの開始、コミット及び中断を含む。
【００８９】

[0081]プロビジョニングインタフェース１５２ｄは、オブジェクトストアが作成され、削除され、マージされ、分割され、集約されることを可能とする。

[0082]インデックス２０４はマップであり、その主キーはオブジェクト名である。他のところで論じたように、インデックスは大変大きくなりうる。システム内のあらゆるオブジェクトのためのインデックスエントリがある。各エントリは、以下を含む。

【００９０】

ａ）オブジェクトのコンテンツのフィンガープリント。フィンガープリントは、コンテンツ上の暗号ダイジェストによって生成され、少量の追加のコンテンツ（「ソルト」）が付加される。ソルトは、オブジェクトストア内の全てのオブジェクトに共通である。

【００９１】

ｂ）何回オブジェクトが参照されるかを示す参照カウント。参照カウントは、空間を節約するために飽和演算（ｓａｔｕｒａｔｉｎｇ　ａｒｉｔｈｍｅｔｉｃ）を使用してもよい。例えば、参照を追跡するために８ビットのみを使用してもよく、参照カウントは追加且つ減分されうるが、２５５に等しくなるか、或いは２５５を上回る場合、カウントは「飽和」し、更なる減分は許可されない。

【００９２】

ｃ）物理ロケータ。オブジェクトが物理ディスク上にある場合、これは論理ブロック番号ＬＢＮであってもよい。オブジェクトがホスト側プロバイダ（例えば、ＡｍａｚｏｎＳ３）によってホストされる場合、クラウドオブジェクトへの参照となる。

【００９３】

ｄ）様々な使用のためのフラグ。あるフラグは、オブジェクトが格納され、圧縮されるかどうかを示し、別のフラグは、暗号化されるかどうかを示す。他のフラグは使用可能であるが、具体的な使用に割り振られない。

【００９４】

[0083]アロケーションマップ２２０は、オブジェクトコンテナ２０６上で割り振られたブロックに対して使用された通常のビットマップである。

[0084]オブジェクトコンテナ２０６は、ランダムにアドレス指定可能な永続ストレージ抽象化である。例は、生のＬＵＮ、ファイル、ディスク上のパーティション、又は、広域ネットワークＷＡＮに渡るｉＳＣＳＩデバイスを含む。

【００９５】

[0085]オブジェクトコンテナ２０６は、幾つかのコンポーネント２０７～２１１（縮尺通りに図示されない）を有する。既知のオフセットにて存続するコンテナ記述子ブロック２０７を除いて、他のコンポーネントの順序は重要ではない。

【００９６】

[0086]インデックス２０８は、Ｂツリー等、コンテナ常駐部分、若しくはメモリ２０４内の部分、又は両方を有してもよい。アロケーションマップ２１０もまた、部分的にディスク上且つメモリ２２０内にあってもよい。２つの間のマイグレーションは、ページング技術により達成されうる。

【００９７】

[0087]オブジェクトストアが修正されるとき、トランザクションログ２１１が永続ストレージ上で保持される。ログは、読出し、書込み、削除、参照調節等を含む、全てのオブジェクトアクティビティを追跡する。ログは、時間順に保持され、周期的にメインインデックス２０８にロールインされる。オブジェクトアクティビティは、メインインデックスを検索する前に、最初にログ上で「ヒット」しなければならない。各ログエントリは、オペレーションタイプ１５２ａ、１５２ｂ、１５２ｃ、１５２ｄ、フィンガープリント、参照カウント、トランザクションＩＤ又はエポック番号、及びプール位置からなる。ログエントリは構造的に、トランザクションＩＤを追加したインデックスエントリに類似する。

【００９８】

[0088]グローバルオブジェクト命名は、一貫性のある命名及びアクセスをなお保ちながら、オブジェクトストアがオブジェクトをあちこちに移動させることを可能とする。オブ

ジェクトを移動させるための理由は、以下を含む。

【００９９】

ａ）性能の理由のため、物理ディスク上で関連オブジェクトを互いの近くに移動させること。

ｂ）故障境界（ｆａｕｌｔ　ｂｏｕｎｄａｒｉｅｓ）に渡ってオブジェクトをレプリケートすること。これは、２つの別々のローカルディスク、ローカルディスク及びリモートディスク、又は、それらの任意の複数のものに渡る可能性がある。レプリケーションはまた、読出し性能利益を与えることもできる。レプリケーションはまた、消去コードによる等、オブジェクトを分割することを含むこともできる。

【０１００】

ｃ）圧縮、解凍、暗号化、暗号解読等、オブジェクトにおけるバックグラウンドオペレーション。

ｄ）温度、即ち、それらの使用の頻度又は期待される頻度に基づいて、オブジェクトを移動させること。

【０１０１】

[0089]図３は、オブジェクトストア１０８の、ＤＩＣＥライブラリ３１０との関係を例示する。ライブラリ３１０は、ダイジェスト１５３ａ、インデックス付け１５３ｂ、圧縮１５３ｃ及び暗号化１５３ｄ等、オブジェクトストアの共通機能を抽象化する。

【０１０２】

[0090]一貫性のあるインタフェースを提供しながら、内部的には、ライブラリは、様々な技術を使用してサービスを達成してもよい。実装技術は、ソフトウェアのみ、部分的ハードウェア支援（例えば、Ｉｎｔｅｌ　ＱｕｉｃｋＡｓｓｉｓｔ（登録商標））、又は、大量のインデックスを格納することができるカスタムハードウェア実装、又は、上記の任意の組み合わせを含む。

【０１０３】

[0091]ハードウェアアクセラレータ１１３を使用する場合、そのアクセラレータは２つの幅広いサービスのクラスを有してもよい。即ち、１つは、計算集中的なオペレーション用１１１（圧縮、暗号化、フィンガープリンティング）、もう１つは、インデックス等、メモリ集中的なオペレーション用１１２である。ハードウェア実装は、一方若しくは他方又は両方を有してもよい。

【０１０４】

[0092]図４は、本実施形態におけるｈノード構造４０１の重要なコンポーネントを例示する。ｈノードは、レガシーｉノードが使用する物理／論理ブロックアドレス指定ではなく、オブジェクト識別子（フィンガープリント）を使用して、コンテンツを識別する。

【０１０５】

[0093]ｈノードは、ファイルのような、コンテンツのシーケンスであり、ランダムに読出し、書込み、付加、作成、削除及び切り捨て可能である。コンテンツは、任意のバイト境界上で、任意の範囲でアクセス可能である。どのようにコンテンツが解釈されるかは、コンテキストによって決まる。

【０１０６】

[0094]ｈノード４０１は、ｓｔａｔ構造４２０、例えば、ファイルメタデータに対して使用されるＰＯＳＩＸ（登録商標）構造を有してもよい。その構造の一部は、ファイル、即ち、この場合はｈノードのバイト長を含んでもよい。データシーケンスは、個別のオブジェクト、例えば、図４のＳ４１０、Ｔ４１１及びＵ４１２に分割される。各オブジェクトの名前は、マッピングテーブル４０２に格納され、マッピングテーブル４０２はＳ、Ｔ及びＵの各々のフィンガープリントを記録する。オブジェクトは、必ずしも同じ長さでなくてもよい。

【０１０７】

[0095]マッピングテーブル４０２は、一般性を失わずに、線形リスト、ツリー構造又は間接指定構造（ａｎ　ｉｎｄｉｒｅｃｔｉｏｎ　ｓｔｒｕｃｔｕｒｅ）を含む、様々な表

現を有してもよい。マッピングテーブル４０２は、標準のＵｎｉｘ(登録商標)のｉノード間接指定テーブルが動く方法に類似の方法で、どの（１つ又は複数の）オブジェクトが参照されるべきであるかを決定するために、コンテンツ（シーケンスＳ、Ｔ及びＵ）へのオフセットによって、インデックス付けされる。

【０１０８】

[0096]ｈノード自体はオブジェクトであり、よって、固有の名前を有する。ｓｔａｔ構造４２０、マッピングテーブル４０２、及び、参照されたオブジェクトの何れかのうち、何れか１つ又は複数が変化するとき、ｈノードの名前（フィンガープリント）もまた変化するようになる。

【０１０９】

[0097]ｈノードは、読出し、書込み及び付加の両方のために、ランダムにアクセスされうる。ｈノードは、書き込まれていないデータが既知の値（典型的には０）を返す、疎空間（ｓｐａｒｓｅ　ｓｐａｃｅ）をサポートする。

【０１１０】

[0098]ｈノードの名前はそのコンテンツの関数であるので、ｈノードに対するいかなる変更も、結果として新しいｈノードとなる。元のｈノードは、ファイルシステムポリシーに応じて、参照解除されても、（参照カウントを増すことによって）保持されてもよい。

【０１１１】

[0099]ｈノード４０１は、例えば、標準のＵｎｉｘ(登録商標)「ｓｔａｔ」構造４２０に加えて、追加の構造を有してもよい。

[0100]図５に示すように、ｈノード４０１は、ファイルに類似した、ランダムにアドレス指定可能なコンテンツのシーケンスである。どのようにそのコンテンツが解釈されるかは、コンテキストによって決まる。本発明の本実施形態では、ｈノードは更に、ファイル、ディレクトリ及びｉマップに特化される。オブジェクト指向プログラミングの専門用語では、クラスファイル、ディレクトリ及びｉマップは、ベースクラスｈノードから派生される。

【０１１２】

[0101]ファイル５０４は、ｈノードが、開き、閉じ、読出し、書込み等が可能な通常のＰＯＳＩＸ（登録商標）ファイルのように見えるようにする、薄いラッパであってもよい。

【０１１３】

[0102]ディレクトリ５０５は、ｈノード４０１の別の解釈である。ディレクトリ５０５は、ファイル名（文字列）へのｉノード番号（整数）のマッピング５０１である。マッピングは、限定されないが、線形リスト、Ｂツリー及びハッシュマップを含む、様々な形式を取ることができる。マップ５０１が完全にメモリ内にある場合、マップが直列化且つ非直列化されうることが要件である。

【０１１４】

[0103]ｉマップ（「ｉノードマップ」）５０２は、（ディレクトリ５０１からの）ｉノード番号をオブジェクトダイジェスト（フィンガープリント）に変換する。オブジェクトは、ｈノード（及び、従って拡張して、ファイル、ディレクトリ又は他のｉマップ）、スーパーブロック等の構造、又は他のデータを表現してもよい。

【０１１５】

[0104]ｉマップは、周知のオブジェクトのために、インデックス０、インデックス１等、位置を予約済みであってもよい。例は、（１つ又は複数の）以前のｉマップ、ファイルシステムスーパーブロック等を含む。

【０１１６】

[0105]図６は、コンテンツが追加されるとき、ファイルコンテンツ及びメタデータが最初の時間$T_0$　６１０から時間$T_1$　６１１までにどのように変化するかを例示する。コンテンツの削除は、類似の経路を辿る。

【０１１７】

[0106]この図は、インタフェース１５２によって分離された、オブジェクトストア１０８コンポーネント、及び、名前空間１０７コンポーネントを共に示す。

[0107]時間$T_0$　６１０で、$Root_0$ディレクトリ$Root_0$　６４０は、２つのファイルＦＯＯ６４１及びＢＡＲ６４２を有する。ファイルＦＯＯ６４１は、オブジェクトＰ６５２及びＱ６５５に分割されたコンテンツからなる。Ｐ６５２及びＱ６５５のためのオブジェクト名は、ＦＯＯ６４１の、以前に例示された（図４）マッピングテーブルに格納される。同様に、ファイルＢＡＲ６４２は、コンテンツＱ６５５を有する。ルートディレクトリ６４０もまた、$Root_0$　６５３によって示されたオブジェクトである。同様に、ファイル（ｈノード）ＦＯＯ６４１及びＢＡＲ６４２は、それぞれオブジェクト６５１及び６５４において表現される。オブジェクト$Root_0$　６５３を有するルートディレクトリ$Root_0$　６４０がそうであるように、最初の$iImap_0$　５０２ａもまた、オブジェクト$iImap_0$　６５０において表現される。

【０１１８】

[0108]オブジェクトＱ６５５は、ファイルＦＯＯ６４１及びＢＡＲ６４２の両方に共通であるので、参照カウントが２であるのに対して、オブジェクトＰ６５２は、時間$T_0$　６１０で参照カウントが１のみである。

【０１１９】

[0109]ルートディレクトリ６４０は、２つのエントリを含み、それぞれＦＯＯ及びＢＡＲのものである。ＦＯＯのエントリは、ｉノードインデックスが４であり、ＢＡＲのｉノードインデックスは９である。

【０１２０】

[0110]$imap_0$　５０２ａはｈノードであり、オブジェクト６５０等として格納される。図面を複雑にすることを避けるため、ｉマップはｈノードであり、ｈノードは多数のオブジェクト上にマップしてもよいが、ここでは１つのオブジェクトとして図示される。

【０１２１】

[0111]慣例により、ルートディレクトリのダイジェストは常に、ｉマップインデックス２に格納される。ｉマップのダイジェストは、ファイルシステムへのフルアクセスを可能とする。ｉマップに関連付けられたオブジェクトを読み出すことによって、ルートディレクトリが得られ、そこから、任意の後続のディレクトリ及びファイルのうちの一方又は双方が得られる。更に、ｉマップのダイジェストは、ダウンストリームのファイルシステム全体のコンテンツを、正確に、且つ、曖昧でなく定義する。

【０１２２】

[0112]不変性：例えば、オブジェクトＱが変化する場合、名前が変化する（オブジェクトの名前は、そのコンテンツの関数である）。修正されたＱにポイントするいかなるマッピングテーブルも、このときはポイントせず、従って、修正されたＱは「可視」ではない。同様の引数は、マップのダイジェストによって参照可能であるいかなるオブジェクトにも適用される。

【０１２３】

[0113]時間$T_1$　６１１で、ファイルＢＡＲ６４２にはコンテンツＳ６５８が付加されているので、新しいファイルＢＡＲ６４４が作成される。新しいファイルＢＡＲは、ダイジェスト及びオブジェクト名が一貫性のあるものになるように、作成されなければならない。新しいコンテンツＳ６５８が追加されるので、それを参照するあらゆるものもまた更新され、新しいバージョンが作成される。これは、より新しいバージョンのＢＡＲ６４４、ルートディレクトリ６４３、及び最も重要なことには、新しいｉマップテーブル５０２ｂに当てはまる。時間$T_0$　６１０でのオブジェクト参照カウント６１４は、コンテンツが追加／除去されるときに調節されるので、時間$T_1$で、$T_1$オブジェクト参照カウント６１５は、$T_0$に固有である、$T_1$に固有であるコンテンツ、及び、共通であるコンテンツを表現する。

【０１２４】

[0114]時間$T_1$　６１１で、本質的に、多くの共通コンテンツを有する２つのファイル

システムがある。これらの２つのファイルシステムは、それらのそれぞれのｉマップ、ｉｍａｐ$_0$　５０２ａ及びｉｍａｐ$_1$　５０２ｂのダイジェストによって十分に特定される。例えば、時間Ｔ$_0$　６１０で、オブジェクトＱ６５５は、経路（６４０ａ、６４１ｂ）、（６４０ｂ、６４２ａ）、（６４３ａ、６４１ｂ）及び（６４３ｂ、６４４ａ）を通じて参照されうる。

【０１２５】

[0115]ファイルのコンテンツが修正（追加、削除、修正）されるとき、ファイルのマッピングテーブルもまた変更される。次には、ファイルマッピングを含むオブジェクト、ｈノードもまた変化する。様々な理由（性能、管理インタフェース）のため、あらゆる変化をツリー中ずっと、ルートディレクトリまで、且つｉマップへと伝わることは、適切でない場合がある。しかし、あらゆるＩＯトランザクション上で行われる場合、システムはＣＤＰを暗黙的に実装し、ＣＤＰではｉマップのあらゆるダイジェストが特定のＩＯトランザクションを表現する。周期的に（例えば、毎時等）、オンデマンドで、或いは、特定のイベント（ファイル閉じ）において行われる場合、ビヘイビアはファイルシステムスナップショットに類似する。

【０１２６】

[0116]同一のオブジェクトがある程度まで、オブジェクトが参照カウントを有するとき、重複排除はシステムに固有である。ファイルシステムが、修正の結果として変化するとき、大部分では、変化のみが、新しいコンテンツがストレージプールに追加される結果となる。

【０１２７】

[0117]図７では、オブジェクト７０１は、不透明のバイナリデータのシーケンスであり、オブジェクト名７０３を有する。オブジェクトサイズは任意であるが、好ましい実装では、割振り及び永続ストレージ管理をより容易にするために、２の累乗のサイズであってもよい。

【０１２８】

[0118]オブジェクトのユーザに対して、コンテンツは常に、クリーンオブジェクトコンテンツ７１０として読出し、書込み且つアクセスされる。オブジェクトストアは、任意選択の圧縮７１１及び暗号化７１２のうちの一方又は双方を含んでもよい形式で、オブジェクトを内部で格納する。よって、ユーザにとって２０４８バイトのオブジェクトのように見える可能性のあるものは、（４：１の圧縮比を仮定すると）５１２バイトのデータとして内部で格納され、このデータは更に暗号化される。オブジェクトストアは暗号化ドメインであり、全てのオブジェクトが暗号化に関して同様に扱われることを意味する。これは、オブジェクトの呼出元が使用する場合のあるいかなる暗号化とも別個である。

【０１２９】

[0119]図８では、ユーザ空間１００で実装される、ファイルシステムの代替実施形態が例示される。オープンソースＦＵＳＥフレームワークを使用して、名前空間ファイルシステム１０７は、ユーザモードＦＵＳＥライブラリ８０２に対してリンクされる。名前空間ファイルシステムは、オブジェクトストア１０８への同じプライベートインタフェース１５２を有する。オブジェクトストア１０８もまた、ＤＩＣＥライブラリ３１０への同じインタフェース１５３を有する。ＤＩＣＥライブラリは、ハードウェア支援１１３を任意選択で使用してもよい。垂直ファイルシステム１０３は、カーネル１０１内に存在し、それはハードウェア支援１１３及び（ＶＦＳ１０３及びＦＵＳＥライブラリ８０２の両方に接続された）ＦＵＳＥモジュール８０１と同様である。

【０１３０】

Ｄ．代替実施形態

[0120]機能の組み合わせを統合するファイルシステムを、従来のシステムのコストの何分の１かで構築する新規の方法が上述された。様々な修正は、代替実施形態の構築時に当業者には明らかになるであろう。

【０１３１】

[0121]新しいファイルシステムは、任意の他のファイルシステムとしてコンピュータ上で実行する、純粋なソフトウェア形式で実現されうる。更に、統合ファイルシステムの編成は、レガシーファイルシステムでは可能ではない、固有のハードウェアアクセラレーション技術に役立つ。ハードウェアアクセラレーションは、所与のコストに対してより高い性能、又は、所与の性能レベルに対してより低い所有コスト総額を可能とする。

【０１３２】

[0122]上記の実施形態では、このファイルシステムは、統合された機能セットを提供する。このファイルシステムは、２つの別個のファイルシステムである、オブジェクトファイルシステム及び名前空間ファイルシステムを含む、スタックとして実装される。スタックは完全にＰＯＳＩＸ（登録商標）準拠であり、第２の拡張ファイルシステム（ＥＸＴ２）、第３の拡張ファイルシステム（ＥＸＴ３）、ＲｅｉｓｅｒＦｓ等、ＰＯＳＩＸ（登録商標）準拠ファイルシステムが求められる場合は常に使用されうる。

【０１３３】

[0123]スタックのより下位の部分は、オブジェクトファイルシステムである。オブジェクトベースのファイルシステムは、オブジェクトの形式でデータをホストするために使用される。オブジェクトは、不透明のバイナリデータのシーケンスである。オブジェクトは、生データであっても、メタデータ（例えば、生データの作成のレコード、及び、生データに対する任意の変更）であってもよい。オブジェクトサイズは様々でありうるが、典型的には、数キロバイト（ＫＢ）の範囲に制限される。しかし、これは、本発明の訂正オペレーションには必要とされない。オブジェクトの名前（本明細書では、フィンガープリントとも呼ばれる）は、例えば、強力な暗号ハッシュを使用して、オブジェクトのコンテンツから導出される。これは、オブジェクト名がグローバル一意且つ識別可能であること、即ち、コンテンツのフィンガープリントを可能とする。オブジェクトファイルシステムは、主としてマシン指向である。

【０１３４】

[0124]等しい２つのフィンガープリントは、どこでそれらのフィンガープリントが計算されたかにかかわらず、実際上は同じコンテンツを表現するようになる。反対に、異なる２つのフィンガープリントは、異なるコンテンツを表現する。フィンガープリントは、オブジェクトよりかなり（例えば、１００倍、１０００倍以上も）小さいので、フィンガープリントを操作することはしばしば、下にあるコンテンツを操作することよりも高速且つ容易である。

【０１３５】

[0125]上記の実施形態に記載されたオブジェクトファイルシステムは、軽量且つフラットであり、ＡＮＳＩ　Ｔ－Ｉ０仕様に記載されたもの等、重量のオブジェクトファイルシステム、又は、市販のＥＭＣ　Ｃｅｎｔｅｒａ（登録商標）、若しくはＨｉｔａｃｈｉの製品（Ａｒｃｈｉｖａｓを介して取得）等、コンテンツアドレス指定可能ファイルシステムとは異なる。本明細書で使用されるようなオブジェクトは、Ｃ＋＋及びＪａｖａ(登録商標)等、プログラミング言語で使用されるようなオブジェクトと混同されるべきではない。

【０１３６】

[0126]オブジェクトファイルシステムは、オブジェクトの全てを追跡する「インデックス」を有する。そのようなインデックスの構築及び管理は、何百万又は数十億ものエントリさえもがインデックス内にある可能性がある場合、オブジェクトファイルシステムにとって大きな課題となりうる。

【０１３７】

[0127]記載された実施形態によれば、ストレージスタックの上に、ファイル、ディレクトリ等を有する名前空間ファイルシステムが提供される。既知のもの（例えば、ＰＯＳＩＸ（登録商標）ファイルシステム）との違いは、しかし、コンテンツにアクセスするために、論理ブロック番号アドレス指定（ＬＢＮ）を使用する代わりに、オブジェクトフィンガープリントが使用されることである。更に、名前空間ファイルシステムの全ての内部デ

ータ構造は、それら自体でオブジェクトである。よって、ストレージスタック全体（名前空間及びオブジェクト層）は、オブジェクト参照によって共に「結び付け」られ、ルートを表現するオブジェクトのフィンガープリントを有することは、ファイル構造全体を完全に、且つ曖昧でなく定義することを可能とする。

【０１３８】

[0128]あらゆる変更（追加、削除、メタデータ変更、読出し）も、ファイルシステムのシグネチャが変更される結果となる。ルートシグネチャを追跡することによって、従って、全てのファイルシステムアクティビティの完全な履歴を得ることができる。

【０１３９】

[0129]本発明の開示された実施形態によれば、２つの別々のコンポーネント（名前空間１０７及びオブジェクトストア１０８）への仕事の分割、及び、これらのコンポーネントがどのように対話するかは、ＰＯＳＩＸ（登録商標）セマンティクスをなお保ちながら、重複排除、スナップ、書込み可能スナップ、継続的データ保護（ＣＤＰ）、広域ネットワーク効率、バージョニング、ファイルシステム整合性チェック及び不変性が自然に起こるような方法で行われる。

【０１４０】

[0130]開示された実施形態によれば、ファイルシステムの編成は、ハードウェア支援の適用を可能とする。ハードウェア支援は、２つの形式を取ってもよい。１つの形式は、圧縮、暗号化及び暗号ダイジェスト等、計算アクセラレーション用である。第２の形式は、実際のオブジェクトストアを構築するために使用される、大きいインデックスのための構築及び維持用である。

【０１４１】

[0131]かなりのＣＰＵ資源が、暗号ハッシング、圧縮及び暗号化で費やされる。より高速なＣＰＵクロック及びより多くのＣＰＵコアは、これをある程度まで軽減するが、性能要件が増大するにつれて、これらの機能の一部又は全部を専用ハードウェア（アクセラレーション）へオフロードすることが望ましい。これを達成することができる、幾つかの市販のチップセット（例えば、Ｈｉｆｎ、Ｃａｖｉｕｍ）がある。

【０１４２】

[0132]オブジェクトストアインデックスは大きくなる可能性があり、実際的なメモリ制限をすぐに上回る場合がある。ランダムに読出し及び書込みされる（そのようなインデックスのための主キーは暗号ハッシュであり、ランダムな分布を有する）、グローバルオブジェクトインデックス（即ち、ストレージの全てのためのインデックス）は、ページング及びキャッシングアルゴリズムを効果のないものにする場合がある。そのようなインデックスを、ソリッドステートディスク（ＳＳＤ）等、より高速の不揮発性ストレージ上に置くことは、よって、性能利益を提供することになる。

【０１４３】

[0133]ＳＳＤは、読出し速度が書込み速度よりかなり速い（即ち、Ｓｅａｇａｔｅ　ｘｘｘは、３５，０００ｉｏｐｓ／読出し且つ３０００ｉｏｐｓ／書込みを達成することができる）ように構築される。インデックスアクセスが読出しと書込みの間で均等に分割される場合、ＳＳＤの利益の多くは実現されない。

【０１４４】

[0134]ＦＬＡＳＨ及びＦＰＧＡでできたカスタムビルドのインデックス付けソリューションは、インデックス付け帯域幅を更により増大させることができる。３１０

[0135]ハードウェア支援は、ＤＩＣＥライブラリによって、予め記載されたように、管理されうる。

【０１４５】

[0136]本発明の実施形態は、デジタル電子回路において、或いは、コンピュータハードウェア、ファームウェア、ソフトウェアにおいて、或いは、それらの組み合わせにおいて実装されうる。本発明の実施形態は、コンピュータプログラム製品として、即ち、情報キャリア内で、例えば、マシン可読ストレージデバイス内で、データ処理装置、例えば、プ

ログラマブルプロセッサ、コンピュータ又は複数のコンピュータによる実行のため、或いはそれらのオペレーションを制御するために、有形に実施されたコンピュータプログラムとして実装されうる。コンピュータプログラムは、コンパイラ型又はインタープリタ型言語を含む任意の形式のプログラミング言語で書かれてよく、スタンドアロンのプログラム又はモジュール、コンポーネント、サブルーチン若しくはコンピューティング環境で使用するために適した他のユニットを含む任意の形式で配置されうる。コンピュータプログラムは、１つのコンピュータ又は複数のコンピュータ上で実行されるように配置することができ、これら複数のコンピュータは、１つの場所にあるか又は複数の場所に渡って分散され通信ネットワークによって相互接続される。

【０１４６】

[0137]本発明の実施形態の方法ステップは、１つ又は複数のプログラマブルプロセッサがコンピュータプログラムを実行して、入力データに関して動作し、出力を生成することにより、本発明の機能を行うことによって、行われうる。方法ステップはまた、専用論理回路、例えば、ＦＰＧＡ（フィールドプログラマブルゲートアレイ）又はＡＳＩＣ（特定用途向け集積回路）によって行われてもよく、本発明の装置は、専用論理回路、例えば、ＦＰＧＡ又はＡＳＩＣとして実装されうる。

【０１４７】

[0138]コンピュータプログラムの実行に適したプロセッサは、例として、汎用及び専用マイクロプロセッサ、及び、任意の種類のデジタルコンピュータの任意の１つ又は複数のプロセッサを含む。一般に、プロセッサは、命令及びデータを、読出し専用メモリ若しくはランダムアクセスメモリ又は両方から受信するようになる。コンピュータの必須要素は、命令を実行するためのプロセッサ、並びに、命令及びデータを格納するための１つ又は複数のメモリデバイスである。一般に、コンピュータはまた、データを格納するための１つ又は複数の大容量ストレージデバイス、例えば、磁気、光磁気ディスク若しくは光ディスクをも含むか、或いは、それらからデータを受信し、若しくはそれらへデータを転送するように動作可能に結合されるか、或いはその両方となる。コンピュータプログラム命令及びデータを実施するために適した情報キャリアは、例として、例えば、ＥＰＲＯＭ、ＥＥＰＲＯＭ及びフラッシュメモリデバイス等、半導体メモリデバイスと、例えば、内蔵ハードディスク又はリムーバブルディスク等、磁気ディスクと、光磁気ディスクと、ＣＤＲＯＭ及びＤＶＤ－ＲＯＭディスクとを含む、全ての形式の不揮発性メモリを含む。プロセッサ及びメモリは、専用論理回路によって補われるか又は専用論理回路に組み込まれうる。

【０１４８】

Ｅ．ｈノード（コンテンツにソルトを加えたもの）

[0139]一実施形態では、ｈノード名は、そのコンテンツのハッシュにソルトを加えたものである。ソルトは、８から３２バイト程度の小さい値であり、シグネチャが計算される前に、内部で自動的にあらゆるオブジェクトの前に付加又は後に付加される。オブジェクトが書き出されるとき、ソルトは格納されない。

【０１４９】

[0140]例えば、ユーザがパスワードを入力し、そこから、暗号鍵生成のために使用される様々な標準技術の任意のものを使用して、ソルトが生成される。ユーザは、任意の他のパスワードのように、このパスワードを保護するようになる。人がその販売を得る場合であっても、元のパスワードを生成することは計算的に可能ではない。

【０１５０】

[0141]ソルトは、主として、不正が行われたデータに対する防御機構であり、この実施例では、以下の通りである。

－あらゆるオブジェクトは、そのフィンガープリントを使用して追跡される、

－フィンガープリントは、インデックス内に格納される、

－フィンガープリントの配布はフラットであり、即ち、全てのフィンガープリントは等しく発生する可能性がある、

－インデックス付けアルゴリズムは、鍵（フィンガープリント）の一律の配布を前提とする。

【０１５１】

[0141]ファイルシステムが、例えば、ＳＨＡ－１等、特定のフィンガープリントアルゴリズムを使用することを、悪意のあるエンティティが知る場合、このエンティティは、大変狭い範囲に入るフィンガープリントを有するコンテンツを容易に生成することができる。そうするために、このエンティティは、ランダムコンテンツを生成し続け、それをフィンガープリントし、特定の狭い範囲に入るコンテンツのみを保持する。それは、インデックス付けアルゴリズムが大変低い性能を有するようにさせることとなる。

【０１５２】

[0142]しかし、暗号ハッシュの性質は、オブジェクトのコンテンツのわずか１ビットを変更する場合、フィンガープリントのビットのおよそ５０％が変化するようになるものである。元のコンテンツの異なるビットを変更するとき、どの５０％であるかもまたランダム化される。

【０１５３】

[0143]ソルトを追加すること（即ち、比較的小さい変更）は、よって、フィンガープリントをランダム化し、インデックス付けアルゴリズムを「操る」ことを非常に困難にする。

【０１５４】

Ｇ．スケーラブルなインデックス付け（実施形態）

[0144]本発明の方法及び装置は、本願と同日（２０１０年６月２５日）に出願され、２００９年６月２６日出願の米国仮特許出願第６１/２６９，６３３号に対する優先権を主張する、本願と同じ発明者であるＰ．Ｂｏｗｄｅｎ及びＡ．Ｊ．Ｂｅａｖｅｒｓｏｎによる、同時係属の、所有者が同じである「Ｓｃａｌａｂｌｅ　Ｉｎｄｅｘｉｎｇ（スケーラブルなインデックス付け）」という名称の米国特許出願第１２/８２３，４５２号に記載された、以下のインデックス付けアルゴリズム及びメモリ技術と共に実装されうる。両出願に対する優先権が本明細書で主張され、各々の完全な開示はそれらの全体として参照により本明細書に組み込まれる。

【０１５５】

[0145]前述及び以下の記載は、本発明の範囲を例示するためのものであり、限定するように意図されないことを理解されたい。

１）概要

[0146]本発明の一実施形態によれば、ユニフォームアクセスインデックス付け処理によって、ノンユニフォームアクセスメモリに格納されたインデックスにアクセスする方法が提供され、この方法は、

インデックス付け処理によって生成された論理バケット識別子を、メモリの物理バケット位置にマップして、インデックス内の各レコードデータエントリにアクセスするために、変換テーブルを維持するステップと、

インデックスに書き込まれる複数のレコードデータエントリをキャッシュ内に集めるステップであって、該ステップは、メモリの少なくとも１つの物理バケット位置へのエントリの集まりの後続の順次書込みに先立つ、ステップと

を含む。

【０１５６】

一実施形態では、この方法は、

レコードデータエントリの集まりを、キャッシュからメモリのバケット位置へ、順次書込みとして書き込むステップと、

集まりのレコードデータエントリのためのバケット位置により、変換テーブルを更新するステップと

を含む。

【０１５７】

一実施形態では、この方法は、
１つ又は複数の順次レコードデータエントリを、メモリからキャッシュへ読み出すステップと、
１つ又は複数のエントリが読み出されたメモリ内の物理位置を、フリーとして指定するステップと
を含む。
【０１５８】
一実施形態では、この方法は、
ブロック内の任意の有効エントリをキャッシュに読み出すこと、及び、そのようなエントリが読み出されたメモリ内の物理位置を、フリーとして指定することによって、メモリ内の複数の順次物理バケット位置をフリーブロックとするステップを含む。
【０１５９】
一実施形態では、
インデックス付け処理が、一律に配布され一意のインデックスキーに基づいて、インデックスへのランダムアクセス要求を生成する。
【０１６０】
一実施形態では、
キーは、暗号ハッシュダイジェストを含む。
一実施形態では、
インデックス付け処理は、置き換えハッシングの処理を含む。
【０１６１】
一実施形態では、
置き換えハッシングは、カッコウハッシング処理を含む。
一実施形態では、
メモリは、フラッシュ、相変化及びソリッドステートディスクのメモリデバイスのうちの１つ又は複数を含む。
【０１６２】
一実施形態では、
メモリは、ランダム書込みアクセス時間、ランダム読出し－修正－書込みアクセス時間、順次書込み、アラインメント制約、消去時間、消去ブロック境界及び摩耗のうちの１つ又は複数によって制限される。
【０１６３】
一実施形態では、
物理バケットのサイズは、メモリの最小書込みサイズを含む。
一実施形態では、
物理バケットのサイズは、ページ又は部分ページを含む。
【０１６４】
一実施形態では、
メモリは、複数のページを含む消去ブロックを有する。
一実施形態では、この方法は、
メモリ内のどのバケット位置が有効であるかを追跡するためのバケット有効テーブルを維持するステップを含む。
【０１６５】
一実施形態では、
メモリ内のバケットは、１つ又は複数のレコードデータエントリのセット、及び、バケット変換テーブルへのセルフインデックスを含む。
【０１６６】
一実施形態では、
バケット内のレコードデータエントリは順序付けされない。
一実施形態では、この方法は、

メモリに順次書き込まれたレコードデータエントリを、キャッシュ内で読出し専用として指定するステップを含む。

【０１６７】

一実施形態では、

バケット変換テーブルは、永続メモリに格納される。

一実施形態では、この方法は、

消去ブロック内のフリーバケットの数を追跡し、フリーバケットのしきい値が満たされたとき、フリー消去ブロックを生成するための処理を実施するステップを含む。

【０１６８】

一実施形態では、

インデックス付け処理は、レコードが挿入、削除、ルックアップ及び修正のうちの１以上をされる要求に基づいて、インデックス付けオペレーションを行う。

【０１６９】

一実施形態では、

インデックス付け処理は、インデックスのレコードを格納する物理バケットを読み出し該物理バケットに書き込むための、論理バケットオペレーションを提示する。

【０１７０】

一実施形態では、

物理バケットオペレーションは、ランダム読出し及び順次書込みを含む。

一実施形態では、

物理バケットオペレーションは、トリムコマンドを更に含む。

【０１７１】

一実施形態では、

メモリは、ノンユニフォーム読出し及び書込みアクセス、並びに、サイズ、アラインメント及びタイミングについての不変性によって特徴付けられた物理デバイス層を含む。

【０１７２】

一実施形態では、

レコードデータエントリは、キー、参照カウント及び物理ブロックアドレスのためのフィールドを含む。

【０１７３】

一実施形態では、

キーは、データの暗号ハッシュダイジェストを含み、

物理ブロックアドレスフィールドは、ストレージデバイス上に格納されたデータの物理ブロックアドレスへのポインタを含む。

【０１７４】

一実施形態では、

論理バケット位置は、複数のハッシュ関数によって生成される。

一実施形態では、

メモリは、複数の消去ブロックを含むフラッシュメモリデバイスを含み、各消去ブロックは複数のページを含み、各ページは複数のバケットを含む。

【０１７５】

本発明の別の実施形態によれば、

プロセッサによって実行されるとき、前述の方法のステップを行うプログラムコード手段を含むコンピュータプログラム製品が提供される。

【０１７６】

本発明の別の実施形態によればコンピュータ可読媒体が提供され、このコンピュータ可読媒体は、ユニフォームアクセスインデックス付け処理によってノンユニフォームアクセスメモリに格納されたインデックスにアクセスする方法のための実行可能プログラム命令を含み、上記方法は、

インデックス付け処理によって生成された論理バケット識別子を、メモリの物理バケッ

ト位置にマップして、インデックス内の各レコードデータエントリにアクセスするために、変換テーブルを維持するステップと、
インデックスに書き込まれる複数のレコードデータエントリをキャッシュ内に集めるステップであって、該ステップは、メモリの少なくとも１つの物理バケット位置へのエントリの集まりの後続の順次書込みに先立つ、ステップと
を含む。

【０１７７】
本発明の別の実施形態によれば、物理プロセッサと、コンピュータ可読媒体を含むメモリデバイスとを含むシステムが提供され、
該コンピュータ可読媒体は、ユニフォームアクセスインデックス付け処理によってノンユニフォームアクセスメモリに格納されたインデックスにアクセスする方法のための実行可能プログラム命令を含み、上記方法は、
インデックス付け処理によって生成された論理バケット識別子を、メモリの物理バケット位置にマップして、インデックス内の各レコードデータエントリにアクセスするために、変換テーブルを維持するステップと、
インデックスに書き込まれる複数のレコードデータエントリをキャッシュ内に集めるステップであって、該ステップは、メモリの少なくとも１つの物理バケット位置へのエントリの集まりの後続の順次書込みに先立つ、ステップと
を含む。

【０１７８】
一実施形態では、
インデックスを格納するメモリは、ノンユニフォーム読出し及び書込みアクセス、並びに、サイズ、アラインメント及びタイミングについての不変性によって特徴付けられた物理デバイス層を含む。

【０１７９】
一実施形態では、
インデックスを格納するメモリは、フラッシュ、相変化及びソリッドステートディスクのメモリデバイスのうちの１つ又は複数を含む。

【０１８０】
一実施形態では、
インデックスを格納するメモリは、複数の消去ブロックを含むフラッシュメモリデバイスを含み、各消去ブロックは複数のページを含み、各ページは複数のバケットを含む。

【０１８１】
本発明の別の実施形態によれば、ユニフォームアクセスインデックス付け処理によって、ノンユニフォームアクセスメモリに格納されたインデックスにアクセスする方法が提供され、この方法は、
インデックス内のレコードデータエントリごとに論理バケット識別子をメモリの物理バケット位置にマップする変換テーブルに、インデックス付け処理によって生成された論理バケット識別子を提供するステップと、
論理バケット識別子にマップされた物理バケット位置にアクセスするステップと、
インデックスに書き込まれるレコードデータエントリをキャッシュ内に集めるステップと、
続いて、レコードデータエントリの集まりを、キャッシュから、メモリの少なくとも１つの新しい物理バケット位置におけるインデックスに順次書き込むステップと、
変換テーブルを更新して、少なくとも１つの新しい物理バケット位置を論理バケット識別子に関連付けるステップと
を含む。

【０１８２】
本発明の別の実施形態によればあるコンピュータシステムが提供され、このコンピュータシステムは、

ノンユニフォームアクセスメモリであって、メモリの物理バケット位置においてレコードデータエントリを含むインデックスが格納される、ノンユニフォームアクセスメモリと、
ユニフォームアクセスインデックス付け処理によって生成された論理バケット識別子を、レコードデータエントリの各々のためのメモリの物理バケット位置にマップするための、変換テーブルと、
インデックスに書き込まれる集められたレコードデータエントリのためのキャッシュと、
メモリの物理バケット位置にアクセスする手段であって、該物理バケット位置は、インデックス付け処理によって変換テーブルに供給された論理バケット識別子にマップされた、手段と、
レコードデータエントリの集まりを、キャッシュから、メモリの少なくとも１つの物理バケット位置のインデックスに順次書き込む手段と、
変換テーブルを更新して、少なくとも１つの物理バケット位置を論理バケット識別子に関連付ける手段と
を含む。

【０１８３】

２）図面

[0147]インデックス付けの発明は、以下の図面と共に、詳細な説明を参照することによって、より十分に理解される。

【０１８４】

図９は、本発明の一実施形態により行われる、様々なインデックス付けオペレーションを例示する概略ブロック図である。
図１０Ａないし１０Ｄは、本発明で使用されうるデータ構造の様々な実施形態を例示する図である。

【０１８５】

図１１は、本発明の一実施形態によるルックアップオペレーションを例示する概略ブロック図である。
図１２は、本発明の一実施形態による挿入オペレーションを例示する概略ブロック図である。

【０１８６】

図１３は、本発明の一実施形態による削除オペレーションの概略ブロック図である。
図１４は、本発明の一実施形態による更新オペレーションの概略ブロック図である。
図１５Ａ及び１５Ｂは、本発明の一実施形態によるフリー消去ブロックを生成するためのランダム読出し処理を例示する概略ブロック図である。

【０１８７】

図１６Ａ及び１６Ｂは、スキャベンジング処理に従ってフリー消去ブロックを生成する別の方法を例示する概略ブロック図である。
図１７は、本発明の実装を例示するための６層のビュー又はスタックを例示する概略ブロック図である。

【０１８８】

図１８は、本発明の一実施形態で使用されるようなレコードエントリの概略図である。
図１９Ａ～１９Ｅは、本発明の一実施形態によるカッコウハッシングの実装を概略的に例示する図である。

【０１８９】

図２０は、本発明の一実施形態による、各バケットが複数のレコードを保持する、複数のバケットの概略的例示の図である。
図２１は、本発明の一実施形態によるバケットのコンテンツの概略図である。

【０１９０】

図２２は、本発明の一実施形態による、複数のダイ、消去ブロック、ページ及びバケッ

トを有する物理フラッシュチップの一例を例示する概略ブロック図である。
図２３Ａ～図２３Ｂは、本発明の一実施形態によるデバイス管理層のあるコンポーネントを例示する図である。
【０１９１】
３．概観
[0148]本発明の１つ又は複数の実施形態によれば、特化されたメモリ技術及びアルゴリズムが使用されて、多数のレコード及びトランザクション要件を同時に有するインデックスが構築される。一実施形態は、置き換え（ｄｉｓｐｌａｃｅｍｅｎｔ）ハッシングインデックス付けアルゴリズム、例えばカッコウハッシングを利用する。本発明は、フラッシュ、相変化及びソリッドステートディスク（ＳＳＤ）メモリデバイス等、ノンユニフォームアクセスメモリ技術の使用を可能とする。
【０１９２】
[0149]本発明の様々な実施形態では、メモリデバイスが、メモリデバイスにとって効率的であるＩＯ（入力／出力）パターンに遭遇しながら、インデックス付けアルゴリズムがアルゴリズムにとって自然（効率的）である方法で行うことを保証するために、新しいデータ構造及び方法が提供される。
【０１９３】
[0150]インデックス付けアルゴリズムによって見なされた論理バケットを、メモリデバイス上の物理バケットにマップする、あるデータ構造である間接指定テーブルが作成される。このマッピングは、ノンユニフォームアクセスメモリデバイスへの書込み性能が強化されるようにするものである。
【０１９４】
[0152]キャッシュの追い出し及びライトバックポリシーの一部として、バケットを集め、メモリデバイスへ順次書き出すために、別のデータ構造である連想キャッシュが使用される。
【０１９５】
[0153]インデックス付けアルゴリズムによって必要とされる（レコードの）バケットでキャッシュをポピュレートするために、方法が使用される。追加のバケットは、デマンド読出し中に又はスキャベンジング処理によって、メモリデバイスからキャッシュへ読み出されてもよい。
【０１９６】
[0154]間接指定テーブルと併せた、キャッシュの使用は、メモリデバイスへの大きい順次書込みを可能にする。
[0155]フラッシュ技術は、インデックス付け問題のために必要とされた容量及びＩＯレートを達成する基本機能を有するが、フラッシュアクセス特性はノンユニフォームである。このノンユニフォーム性はかなり十分なので、通常のインデックス付けアルゴリズムは、フラッシュメモリデバイスと共に、たとえ機能するとしても、うまく機能しない。
【０１９７】
[0156]本発明で使用されるノンユニフォームアクセスフラッシュメモリは、数百から数千ビットもの大きいブロックサイズで、即ち、バイトレベルのランダムアクセスではなく、読出し、書込み及び消去が行われなければならない、電気的消去可能プログラマブル読出し専用メモリ（ＥＥＰＲＯＭ）である。物理的には、フラッシュは、フローティングゲートトランジスタからなるメモリセルのアレイに情報を格納する、不揮発性メモリ形式である。２つのタイプのフラッシュメモリデバイス、ＮＡＮＤフラッシュ及びＮＯＲフラッシュがある。ＮＡＮＤフラッシュは、ＮＯＲフラッシュより高密度及び大容量をより低コストで提供し、より高速な消去、順次書込み及び順次読出し速度を有する。本願において、且つ本発明において使用されるとき、「フラッシュ」メモリは、ＮＡＮＤフラッシュメモリを包含するものであり、ＮＯＲメモリを包含しないものである。ＮＡＮＤは、各セルが１ビットの情報のみを格納するシングルレベルセル（ＳＬＣ）デバイス、及び、１つのセルにつき２ビット以上を格納することができる、より新しいマルチレベルセル（ＭＬＣ

）デバイスを共に含む。ＮＡＮＤフラッシュは高速なアクセス時間を提供するが、ＰＣ内でメインメモリとして使用された揮発性ＤＲＡＭメモリほど高速ではない。フラッシュメモリデバイスは、フラッシュファイルシステムを含んでも含まなくてもよい。フラッシュファイルシステムは、典型的には、ウェアレベリング及び誤り訂正を行うために内蔵コントローラを有していない、内蔵フラッシュメモリと共に使用される。

【０１９８】

[0157]典型的なＮＡＮＤフラッシュチップは、数ＧＢのコンテンツを格納してもよい。コンピュータに取り付けられたメモリとは異なり、フラッシュチップ上のメモリは、あるサイズで、且つ、ある境界上でアクセスされなければならない。更に、メモリのセクションに書き込まれた後、それらのメモリ位置が再び書込み可能となる前に、消去オペレーションが行われなければならない。また、位置は徐々になくなるので、全ての位置が同様の数の書込みを得ることを保証することは、使用を更に複雑にする。読出し時間、書込み時間及び消去時間は、大幅に（数マイクロ秒から数ミリ秒まで）変わる可能性がある。よって、タイミング、ウェアレベリング及びアラインメント制約は、フラッシュの実際的な使用を困難にするのがせいぜいである。

【０１９９】

[0158]フラッシュメモリデバイスは、１つ又は複数のダイ（シリコンウエハ）を含んでもよい。各ダイは、大部分では、独立してアクセスされうる。

[0159]ダイは、数千もの消去ブロックからなる。消去ブロックは典型的には、１２８～５１２ＫＢのサイズである。データがクリアされる必要があるとき、消去ブロック境界上でクリアされなければならない。

【０２００】

[0160]ＮＡＮＤフラッシュのもう１つの制限は、データが順次書き込まれることのみが可能であることである。更に、書込みのためのセットアップ時間は長く、読出しのセットアップ時間のおよそ１０倍である。

【０２０１】

[0161]データは、ページ粒度で読み出される。ページは、特定のフラッシュチップに応じて、１ＫＢから４ＫＢまでに及ぶ場合がある。各ページに関連付けられるものは、誤り訂正符号（ＥＣＣ）チェックサムのために使用されうる、数バイトである。

【０２０２】

[0162]データは、ページ粒度で書き込まれる。書き込まれた後、ページは、その消去ブロック（ページを含む）が消去されるまで、再度書き込まれない場合がある。消去ブロックは、数十から１００を超えるページを含む場合がある。

【０２０３】

[0163]上記の読出し及び書込みページ粒度に対する１つの例外は、サブページ書込み、又は、部分ページプログラミングである。技術に応じて、ページは、消去が必要とされる前に最大４回、部分的に書き込まれる場合がある。

【０２０４】

[0164]ＮＡＮＤフラッシュブロック内のページは順次、且つ、ブロック消去オペレーションの間に一度のみ書き込まれうるので、後続の書込みは、典型的には異なるフラッシュブロック内に位置する異なるページへの書込みを必要とする。ブロック消去の問題点は、フラッシュファイルシステムの機能である、書込み可能フラッシュブロックのプールを作成することによって、対処される。

【０２０５】

[0165]消去ブロックを消去することは、数ミリ秒を取りうるので、時間的に最も費用のかかるオペレーションである。（トラフィック的に）多用されるデバイスでは、消去ブロックが生成されうる速度（即ち、どのくらい速くフリー消去ブロックが使用可能にされうるか）は、フラッシュ設計における制限要因であることが多い。

【０２０６】

[0166]多数のＳＳＤ（ソリッドステートディスク）は、フラッシュ技術を使用する。Ｓ

ＳＤにおけるファームウェアは、フラッシュ変換層（ＦＴＬ）と呼ばれる層において、前述のアクセスの問題点に対処する。そうすることにおいて、しかし、ファームウェアは、どのようにＳＳＤが使用されるか（例えば、大部分は読出し、大部分は書込み、読出し及び書込みのサイズ及びアラインメント）について仮定を行い、これらの仮定の結果として、ＳＳＤの性能特性は、インデックス付けアルゴリズムにとって準最適であることが多い。

【０２０７】

[0167]人が文献において、また実際に発見する多数のインデックス付けアルゴリズムは、ユニフォームメモリアクセスモデルに基づいており、即ち、全てのメモリが、読出しでも書込みでも時間的に等しくアクセス可能であり、アクセスサイズ又はアラインメントにおけるいかなる一次制約もない。

【０２０８】

[0168]インデックス付けソリューションを考察する場合、挿入、削除、ルックアップ及び修正等のオペレーションは典型的には、より多くの多様な時間の量を必要とし、ブロック、典型的には小さいブロック（４ＫＢくらい）の読出し及び書込みは、より短い時間を必要とする。ブロックはランダムであるように見え、即ち、任意のブロックが読み出されてよく、任意の他のブロックが書き込まれてよい。幾つかのアルゴリズムでは、ランダム読出し－修正－書込みＩＯプロファイルがあり、即ち、ランダムブロックが読み出され、次いで、わずかに修正されたデータで同じ位置に書き戻される。

【０２０９】

[0169]インデックス付けアルゴリズムの効率的な動作を必要とするこのランダムＩＯは、フラッシュが提供するように意図されるものではない。フラッシュは、ランダム読出しをうまく扱うことができるが、読出し－修正－書込みがそうであるように、ランダム書込みは困難である。この理由は、すでに書き込まれたものを上書きすることはできず、まずそれを消去しなければならないからである。この状況を更に複雑にすることには、消去は時間を要し、大きい境界（典型的には６４ＫＢ）上で起こらなければならない。

【０２１０】

[0170]消去ブロックが消去されるとき、そのブロック内のいかなる有効データも他のところへ移動される必要がある。アルゴリズムがフラッシュデバイス中でランダムな４ＫＢブロックを書き込む場合、単純な実装は、その間ずっとブロックが消去され続ける結果となる。消去時間は遅いので、性能がかなり損なわれることになる。

【０２１１】

[0171]本発明によれば、フラッシュへの書込みを順次にすることを可能にするため、インデックス付けアルゴリズムが期待する論理ランダムアクセスをなお保ちながら、変換又は間接指定テーブルが作成される。このテーブルは、インデックス付けアルゴリズムによって必要とされるような（レコードの）論理バケットを、フラッシュデバイスの物理バケット（例えば、ページ）にマップする。

【０２１２】

[0172]インデックス付けアルゴリズムがバケット（例えば、フラッシュからのデータのページ）を読み込むとき、バケットコンテンツを修正（挿入、更新又は削除オペレーション）するために、バケットはキャッシュに移動される。フラッシュデバイス上の対応するバケットにはこのとき、有効ではない（フリー）としてマークが付けられうる。ＳＳＤの場合、これはＴＲＩＭコマンドの形式を取りうる。

【０２１３】

[0173]本発明の更なる実施形態によれば、フリー消去ブロックを生成するために、方法が提供される。いかなる所与の時間でも、消去ブロックは、有効及び無効データの組み合わせを有する場合がある。消去ブロックを解放するため、全ての有効データは、そのブロックから離れるように移動されなければならない。これを達成するために使用されうる、２つの機構がある。１つは、インデックス付けアルゴリズムによって生成されたランダム読出しを使用して、（インデックス付けアルゴリズムによって必要とされるよりも）より

多くを読出し、消去ブロックを解放するようにすることである。インデックス付けアルゴリズムはランダム読出しを生成する傾向があるので、経時的に、全ての消去ブロックは最終的に読み出され、空のページのために集積される。例えば、読出しを含む消去ブロックが、幾つかのフリーページ及び幾つかの有効ページを有する場合、アルゴリズムは、消去ブロック全体を読み込み、且つ、全ての有効ページをキャッシュに入れることを選択してもよい。これは、後続の消去及び次いで書込みのために、その消去ブロックを解放する効果を有する。

【０２１４】

[0174]別法として、例えば、前述のランダム読出し処理が十分に高速でない場合、消去ブロックを読み出し、別の消去ブロックと合体するために有効ページをキャッシュに入れるために、分離したスキャベンジング処理（例えば、スレッド）が使用されうる。

【０２１５】

[0175]キャッシュが一杯になるとき、エントリが書き出されなければならない。キャッシュエントリのセットが集められ、部分ページの隣接するセット（部分ページ書込みがフラッシュデバイスによって可能にされる場合）、複数のページ及び１つ又は複数の消去ブロックのうちの１以上へと、順次書き込まれるようになる。キャッシュエントリがフラッシュデバイスに書き込まれるとき、間接指定テーブルは更新されるので、インデックス付けアルゴリズムはなお、エントリが固定された論理アドレスにあるものと見なす。

【０２１６】

４．インデックス付けオペレーション

[0176]本発明の様々な実施形態がここで、添付の図９～１４を利用して説明され、本発明に従って行われた様々なインデックス付けオペレーションが例示される。図１５～１６は、ストレージ媒体（例えば、フラッシュメモリ）の効率的利用のためにフリー消去ブロックを生成する、２つの方法を例示する。これらの実施形態は例示的なものであり、限定するものではない。

【０２１７】

[0177]図９は、本発明の一実施形態による、バケット変換テーブル１７’及びキャッシュ２３’を利用する幾つかのインデックス付けオペレーションの概観である。図９の上部に、３つのインデックスオペレーション１２’～１４’が、ルックアップ関数１５’及び変換関数１６’への代替入力として示される。第１のインデックスオペレーション１２’は、キーのための（レコードエントリ）から衛星データを返すための「キーをルックアップする」である。第２のインデックスオペレーション１３’は、キーのためのレコードエントリを更新（修正）するための「キーのための衛星データを更新する」である。第３のインデックスオペレーション１４’は、新しいレコードエントリを挿入するための「新しいキーを挿入する」である。もう１つのインデックスオペレーションである削除は、図９では図示されないが、図１３に関して後述される。

【０２１８】

[0178]全ての３つのインデックスオペレーションは、最初にルックアップ関数１５’を行い、キーのある関数ｆ（キー）が使用されて、インデックスが生成され、ここでは、ハッシュテーブルルックアップをサポート（例えば、加速）する論理バケット識別子である。バケット識別子（インデックス）は、変換関数１６’への入力であり、論理バケット識別子のある関数ｆ（インデックス）は、フラッシュメモリ内の物理バケット位置を生成する。この変換関数はバケット変換テーブル１７’によって実装され、バケット変換テーブル１７’は、対象フラッシュメモリ位置（フラッシュ内の物理バケット位置）への（インデックス付けアルゴリズムによって提供されるような）論理バケット識別子のマップである。フラッシュメモリ２６’に格納されたディクショナリ（インデックス）は、ルックアップキー（例えば、オブジェクト名）を衛星データ（例えば、ディスク上に格納されたオブジェクトへの位置ポインタ）にマップするレコードを備えてもよい。

【０２１９】

[0179]次に、３つのインデックス付けオペレーションのうちどれが行われているか（ル

ックアップ、更新又は挿入）に応じて、図９の下半分に示されたステップのうち１つ又は複数が行われる。

【０２２０】

[0180]ルックアップオペレーション１８’では、変換関数によって識別されたバケットエントリは、キャッシュルックアサイドによる、フラッシュメモリ内の対象バケット２２’からの読出し３０’である（例えば、対象バケットがキャッシュに格納される場合、フラッシュメモリ２６’からではなく、キャッシュ２３’から読み出されてもよい）。

【０２２１】

[0181]更新オペレーション１９’では、変換関数によって識別されたバケットエントリ（元のバケットエントリ）が、フラッシュメモリ（又はキャッシュ）の消去ブロック２１ａ’内の対象バケット２２’から読み出され３０’、バケットが更新され、キャッシュへ移動され３２’、且つ、後続の書込み２４’では、複数のキャッシュバケットエントリが、フラッシュメモリ内の部分ページの隣接するセット、複数のページ、及び、消去ブロック（例えば、新しい消去ブロック２１ｂ’）のうちの１以上に順次読み出される。この処理は、フラッシュ内の全ての移動されたバケットのステータスを、有効ではないデータ（例えば、フリー、又は、トリムオペレーションのために使用可能）に更新する３３’。

【０２２２】

[0182]挿入オペレーション２０’では、対象バケットがフラッシュから再度読み出され、修正されたバケットエントリが、フラッシュメモリ内の新しい位置への後続の順次書込み２４’のために再度、キャッシュに移動される３４’。

【０２２３】

[0183]図９は、隣接するフラッシュメモリバケットへの、キャッシュバケットエントリの集まりの順次書込み２４’を行うことに先立って、複数のバケットエントリを集めるためのキャッシュ２３’を概略的に示す。一実施形態では、スキャベンジングオペレーション２５’が、フリー消去ブロックを作成するために使用され、この処理は、スキャベンジング処理中に（消去ブロックからの）任意の有効バケットをキャッシュに格納すること、及び、フラッシュ消去ブロックをフリーとして再割振りすることを含む。

【０２２４】

[0184]図１０に例示された新しいデータ構造の考察に続いて、図９で参照されたインデックス付けオペレーションが、図１１～１４のフロー図に関してより具体的に説明される。

【０２２５】

５．データ構造

[0186]図１０は、本発明において有用なデータ構造の様々な実施形態を例示する。そのようなデータ構造は例示的なものであり、限定するものではない。

【０２２６】

[0187]図１０Ａは、（インデックス付けアルゴリズムによって生成された）論理バケットインデックスを、物理フラッシュバケットアドレスに変換するための、バケット変換テーブル（ＢＴＴ）３００’の一実施形態を例示する。ＢＴＴテーブルエントリは、３つのフィールド、即ち、有効３０１’、フラッシュ物理バケットアドレス３０２’、及び、拡張バケット状態３０３’を有して図示される。バケットアドレス粒度は、フラッシュデバイスの最小書込みサイズであり、即ち、（例えば、ＳＬＣ　ＮＡＮＤでの）部分ページ書込み、又は、（例えば、ＭＬＣ　ＮＡＮＤでの）ページ書込みである。ＢＴＴは、論理から物理バケットエントリへの１：１マッピングである。このテーブルは、より高いランダム性能（インデックス付けアルゴリズムによるランダム読出し及びランダム書込み）のために、フラッシュバケット割当ての再編成を可能とする。アルゴリズムのアクセラレーションを可能とするために、追加の状態情報がＢＴＴへ、第３のフィールド内に追加されてもよい。

【０２２７】

[0188]図１０Ｂは、バケット有効テーブル（ＢＶＴ）３０５’の一実施形態を示す。こ

のテーブルは、トリミングのためのブロックへのバケットのスキャベンジングを管理するために、フラッシュ内のどの物理バケットが有効であるかを追跡する。一例として、有効とラベル付けされたフィールド３０６’は、小型のビット配列（１ビット／バケット）であってもよい。ＢＶＴのサイズは、フラッシュバケットエントリの総数であり、そのサブセットのみがＢＴＴによって使用中である。

【０２２８】

[0189]図１０Ｃは、バケット内に含まれた複数のレコード３１０’、３１１’、３１２’．．．を、逆ＢＴＴポインタ３１３’（バケット変換テーブル１７’へのセルフインデックス）と共に有する、フラッシュバケット３０９’の一実施形態を例示する。よって、各バケットは、１つ又は複数のレコードのセット、及び、フラッシュバケット（例えば、ページ）が挿入、移動又は削除されるとき、ＢＴＴを更新するための逆ポインタを含む。バケットの各要素（レコード又はポインタ）には、データ構造の個々の信頼性を向上させ、且つ、ストレージデバイスの有用寿命を著しく延ばすために、追加のＥＣＣビット等、冗長コンテンツが追加されてもよい。例えば、任意選択のシーケンス番号フィールドが、停電発生中にデータ一貫性チェックを行うためにフラッシュバケット３０９’に追加されてもよく、他の最適化フラグも同様に提供されてもよい。

【０２２９】

[0190]レコードサイズはバケットサイズに比べて小さいので、これは、個々のレコードベースで追加の誤り回復情報を実装する（任意選択の）機会を提供する。この任意選択の機能は、訂正される可能性があるので下にあるストレージ技術の実効動作寿命を延ばす可能性のある、ビット誤り及び障害の数を増すことによって、ソリューションの全体の信頼性を向上させるようになる。

【０２３０】

[0191]図１０Ｄは、複数の消去ブロック３１６’（１からＭ）を含む、ＳＬＣ　ＮＡＮＤフラッシュデバイス３１５’の一例を示す。各消去ブロックは、複数のページ３１７’（１からＮ）を含む。この例では、各ページは４ＫＢであり、各ページは複数のバケット３１８’（１からＢ）を含み、各バケットは１ＫＢである。この例では、デバイスは部分ページ書込みをサポートする。

【０２３１】

[0192]バケットは、フラッシュデバイスの最小書込みサイズを表現する。典型的には、バケットは１ページとなる。部分ページ書込みが可能にされる場合、１ページにつき４つのバケットをサポートする４部分ページＳＬＣ　ＮＡＮＤデバイス等、１フラッシュページにつき１つ又は複数のバケットが提供されてもよい。

【０２３２】

[0193]１つの消去ブロックにつき、複数のフラッシュページが提供される。１つのフラッシュデバイスにつき複数の消去ブロックがあり、各ブロックは個々に消去される。

[0194]典型的なフラッシュサブシステムは、複数のフラッシュデバイスからなる。ＮＡＮＤフラッシュデバイスは、消去オペレーション間に所与のブロック内で１ページ（又は部分ページ）につき１回、順次書き込まれ、複数のブロックが、同時に書込み及び読出しを行うために使用可能である。

【０２３３】

６．処理フローチャート

[0195]図１１は、キーの存在を検証し、関連付けられた衛星データを返すためのルックアップオペレーション処理の一実施形態を例示する。ステップ１　４１’で、ルックアップキーがルックアップ関数に入力される。ステップ２　４２’で、ルックアップ関数ｆ（キー）は、ハッシュテーブルルックアップをサポート（例えば、加速）する論理バケット識別子を生成する。論理バケット識別子は変換関数に入力され、変換関数は、ステップ３　４３’で、バケット変換テーブル（ＢＴＴ）１７’を介してフラッシュメモリ（物理バケット）位置にマップされる。ステップ４　４４’で、バケットがキャッシュに格納されるまで、フラッシュメモリ内の対象バケットがフラッシュメモリから読み出され４５ａ’

、バケットがキャッシュに格納された場合、キャッシュ２３’から読出し可能である４５ｂ’。ステップ６　４６’で、キーのための衛星（レコード）データがインデックス付けアルゴリズムに返される。

【０２３４】

[0196]図１２は、挿入オペレーション処理の一実施形態を示す。最初のステップ７１’は、キーをルックアップ関数に入力する。ステップ２　７２’で、ルックアップ関数ｆ（キー）は、インデックス、ここでは論理バケット識別子を生成する。ステップ３　７３’で、バケット識別子が変換関数に入力され、変換関数は、バケット変換テーブル（ＢＴＴ）１７’を利用して、バケット識別子を、挿入が起こるべきフラッシュメモリ物理バケット位置へマップする。ステップ４　７４’で、挿入処理は、対象バケット位置を変換関数から受信する。ステップ５　７５’で、挿入処理は、対象バケット２２’をフラッシュメモリの消去ブロック２１ａ’から７５ａ’、或いはキャッシュから７５ｂ’、読み出す。ステップ６　７６’で、挿入処理は、レコードエントリを対象バケットに挿入し、修正済みバケットをキャッシュに書き込む。ステップ７　７７’で、挿入処理によって、複数のバケットエントリ（修正済み対象バケットを含む）がキャッシュ７３’から読み出される。ステップ８　７８’で、挿入処理は、キャッシュから読み出された修正済み対象バケット及び他のバケットを、フラッシュ２６’内の新しい位置（消去ブロック２１ｂ’内のページ）へ書き込む。ステップ９　７９’で、挿入処理は、キャッシュからフラッシュへ移動された全てのバケットのための新しい位置により、バケット変換テーブル１７’を更新し７９ａ’、また、移動された全てのバケットについて、ＢＶＴ内のバケット有効エントリをも更新する７９ｂ’。ステップ１０　８０’で、挿入処理は、移動されたキャッシュエントリに、読出し専用（使用可能）とマークを付ける。ステップ１１　８１’で、挿入処理は、元のフラッシュバケット（このとき、新しい消去ブロックに移動済み）に、フリーとしてマークを付ける。

【０２３５】

[0197]図１３は、削除オペレーション処理の一実施形態を例示する。最初のステップ９１’では、キーがルックアップ関数に提供される。ステップ２　９２’で、ルックアップ関数ｆ（キー）は、インデックス、ここでは論理バケット識別子を生成する。ステップ３　９３’で、バケット識別子が変換関数に提供され、変換関数は、バケット変換テーブル１７’を利用して、バケット識別子を物理フラッシュメモリバケット位置へマップする。ステップ４　９４’で、削除処理は、フラッシュメモリ位置を受信する。ステップ５　９５’で、対象バケットがフラッシュから読み出される。ステップ６　９６’で、処理は、バケット内の元のレコードエントリを削除し、修正済みバケットを（削除済みエントリと共に）キャッシュ２３’に書き込む。ステップ７　９７’で、バケットのグループ（集まり）がキャッシュから読み出される。ステップ８　９８’で、キャッシュ２３’から読み出された更新済み対象バケット及び他のバケットが、フラッシュ内のフリーページの隣接するセットに順次書き込まれる。ステップ９　９９’で、削除処理は、全ての移動されたバケットのためのフラッシュ内の新しい位置により、バケット変換テーブルを更新し９９ａ’、ＢＶＴ内のそれらのバケット有効ステータスを更新する９９ｂ’。ステップ１０　１００’で、削除処理は、キャッシュエントリに、読出し専用としてマークを付ける。ステップ１１　１０１’で、削除処理は、このとき、フラッシュ内の新しい位置に移動された元のフラッシュバケットに、フリーとしてマークを付ける。

【０２３６】

[0198]図１４は、フラッシュメモリに格納されたインデックス内のレコードを修正するための更新オペレーション処理の一実施形態を例示する。最初のステップ５１’で、キーが入力としてルックアップ関数に提供される。ステップ２　５２’で、ルックアップ関数ｆ（キー）は、インデックス、ここでは論理バケット識別子を生成する。バケット識別子は変換関数に入力される。ステップ３　５３’で、変換関数は、バケット変換テーブル１７’を利用して、バケット識別子を、更新が起こるべきフラッシュメモリ内の物理バケットへマップする。ステップ５　５５’で、対象バケットはフラッシュから５５ａ’、或い

はキャッシュから５５ｂ’、読み出される。ステップ６　５６’で、エントリを更新した後、更新済みバケットがキャッシュ２３’に書き込まれる。ステップ７　５７’で、バケットのグループがキャッシュ２３’から読み出され、ステップ８　５８’で、キャッシュからフラッシュメモリ２６’内の新しい位置へ順次書き込まれる。ステップ９　５９’で、更新処理は、全ての移動されたバケットのための新しい位置により、バケット変換テーブル１７’を更新し５９ａ’、ＢＶＴ内のそれらの有効ステータスを更新する５９ｂ’。ステップ１０　６０’で、更新処理は、移動されたエントリに、キャッシュ２３’内で読出し専用（及び、よって上書きされるために使用可能）としてマークを付ける。最後に、ステップ１１　６１’で、更新処理は、このとき新しい位置に移動された元のフラッシュバケットに、フリー（使用可能）としてマークを付ける。

【０２３７】

[0199]図１５Ａは、フリー消去ブロックを生成するための処理の一実施形態を例示し、（ルックアップ、挿入又は修正等、アップストリームインデックス付けオペレーションによって生成された）デマンド読出しは、（対象バケットと）同じ消去ブロック内の追加のバケットを読み出す。図１５Ａで、この処理は、更新要求と共に例示される。ステップ１　１１１’で、キーがルックアップ関数に提供される。ステップ２　１１２’で、ルックアップ関数ｆ（キー）は、インデックス、ここでは論理バケット識別子を生成する。ステップ３　１１３’で、バケット識別子が、フラッシュ内の物理対象バケット位置にマップされる。ステップ４　１１４’で、更新及びスキャベンジ処理は、対象フラッシュメモリ位置を受信する。ステップ５　１１５’で、処理は、対象バケットと同じ消去ブロック内の全ての有効バケットを識別する。ステップ６　１１６’で、更新処理は、対象バケット及び全ての識別された有効バケットを、対象バケットを含むフラッシュブロックから読み出す。ステップ７　１１７’で、処理は、対象バケット内のレコードエントリを更新し、フラッシュブロックからの全ての有効バケットをキャッシュ２３’に書き込む。ステップ８　１１８’で、更新処理は、キャッシュからブロックのグループを読み出す。ステップ９　１１９’で、更新処理は、キャッシュ２３’から読み出された更新済み対象バケット及び他のバケットを、フラッシュ２６’に書き込む。ステップ１０　１２０’で、更新処理は、移動された（キャッシュからフラッシュ内の新しい消去ブロック２１ｂ’に書き込まれた）全てのバケットのための新しい位置により、バケット変換テーブル１７’を更新し１２０ａ’、ＢＶＴ内のバケットエントリを更新する１２０ｂ’。ステップ１１　１２１’で、更新処理は、このとき古いキャッシュエントリに、読出し専用としてマークを付ける。ステップ１２　１２２’で、更新処理は、元のフラッシュブロック（対象ブロック内の全てのバケット）に、フリーとしてマークを付ける。

【０２３８】

[0200]図１５Ｂは、フリー消去ブロックを生成するためにのみ記載された、ランダム読出し処理の特定の実施形態を例示する。

[0201]この実施形態では、置き換えハッシングインデックス付けアルゴリズム１２５’は、論理バケット１２６’を生成する。インデックス付けアルゴリズムによって見なされる論理バケットサイズは、インデックス付けアルゴリズム及びフラッシュメモリが互換性を有するように、フラッシュ消去ブロックサイズに関連付けられる。これらのバケットは、インデックス読出し及び更新の結果として、ランダムに読み出されるようになる。

【０２３９】

[0202]バケット変換（間接指定）テーブル１２７’は、論理バケットインデックスを物理フラッシュデバイスバケット位置に変換する。この間接指定テーブルは、インデックス付けアルゴリズムが、読出し、書込み及び更新のために、ランダムに動作し、且つ、フラッシュデバイスレベルで行われる大きい順次書込みをなお有することを可能とする。好ましくは、間接指定テーブルは永続メモリに格納されるが、揮発性メモリに格納される場合、必要に応じて再構築されうる。

【０２４０】

[0203]間接指定テーブルの出力、即ち、物理デバイスバケット位置は、入力として、完

全連想バケットキャッシュ１２８’に提供される。この実施形態では、空の消去ブロックｆｉｆｏ１２９’のコンテンツがハイウォーターマークＱ未満である場合、消去ブロック全体（対象４ＫＢバケットを含む）が読み出される。

【０２４１】

[0204]消去ブロックは論理バケットをホストし、典型的な構成は、１つの消去ブロックが１６個の４ＫＢ論理バケットを保持するものである。物理デバイスは、負荷、例えば、９０％のために構成され、バケットの９０％が使用中であることを意味する。キャッシング及び犠牲化（ｖｉｃｔｉｍｉｚａｔｉｏｎ）（追い出し）は、残りのバケットの１０％の大部分がフリー消去ブロックに集中させられるように、フラッシュメモリ内で論理バケットをパックする（集中させる）ために、使用される。

【０２４２】

[0205]キャッシュ犠牲化（追い出し処理）は、キャッシュ内に集められた１６個のバケットを取り、それら１６個のバケットをキャッシュからフリー消去ブロック１３０’に書き出す。消去ブロックはランダム読出しオペレーションによってランダムに触れられるので、読出しオペレーションは、フリー消去ブロックを生成するために使用されうる。論理バケット識別子を生成するための暗号ハッシュ関数の使用は、読出しオペレーションのランダム性を増大させ、よって、フリー消去ブロックのランダム読出し生成を改善するようになる。

【０２４３】

[0206]図１６Ａ及び１６Ｂは、フリー消去ブロックを生成するための代替スキャベンジング処理を例示する。このスキャベンジング処理は、何れかのインデックス付けオペレーションの一部ではない。むしろ、より下位レベルのデバイス管理層の一部として実装される。この処理では、フラッシュ消去ブロック内の物理バケットのグループ（一部又は全部）がフラッシュから直接読み出され、バケット有効テーブル２７’が、消去ブロック内のどのバケットが有効であるかを決定するために使用される。

【０２４４】

[0207]図１６Ａに例示されるように、ステップ１　２２０’で、スキャベンジング処理２５’は、完全な消去ブロック２１ａ’を読み出す。ステップ２　２２２’で、スキャベンジング処理は、バケット有効テーブル２７’を使用して、有効であるそれらの読出しの全てのバケットを識別する。ステップ３　２２４’で、各有効バケットについて、論理バケット識別子がバケットから抽出される。ステップ４　２２６’で、有効バケットがキャッシュ２３’に格納され、各々はその論理バケット識別子によってインデックス付けされる。

【０２４５】

[0208]図１６Ｂは、ステップ１で、スキャベンジング処理２５’がバケット［９４，９７］を、９４及び９７を含めて読み出す一例を示す。ステップ２で、処理は、９５及び９６でのバケットが有効であると決定する。有効バケットは、「１」によって指定されたバケット有効テーブルにおいて示され、非有効バケットは「０」による。ステップ３で、バケット９５及び９６のための論理バケット識別子、即ち、タグ２３及び４９がそれぞれバケットから抽出される。ステップ４で、この２つのタグ並びにそれらの各バケット９５及び９６は、それらの各タグ２３、４９をインデックスとして使用して、キャッシュに挿入される。

【０２４６】

７．スタックレベルビュー及び実装

[0209]本発明のもう１つのより具体的な実施例が、ここで図１７～２４を参照して説明される。

【０２４７】

[0210]図１７は、物理フラッシュメモリデバイス２１１’によって望まれるものとは大変異なるビューである、インデックス付けアルゴリズム２０３によって望まれたＩＯ使用プロファイルビューを、フラッシュアダプテーション層２０７’が適合させる、本発明の

一実装を例示するための６層のビュー又はスタック２００’を示す。最高レベル２０１’で、レコードのディクショナリ（インデックス）が提供され、それに対して、あるインデックス付けオペレーション２０４’（レコードをルックアップ、削除、挿入且つ修正する）が必要とされる。インデックス付けアルゴリズム層２０３’は、１つ又は複数のインデックス付けアルゴリズムによりディクショナリを実装し、例えば、カッコウ置き換えハッシングアルゴリズムが一例である。インデックス付けアルゴリズムは、インデックスへのキーがどのようにインデックス永続層２０５’によって格納されるかのビューを有する。インデックス付けビューは、論理ビューであり、論理アドレス位置を特定する。このビューは更に、サイズ、アラインメント及びタイミングについて、インデックスへのユニフォームアクセスが存在するようになること、及び、インデックスが可変（安定）ストレージ上に格納されることを仮定する。

【０２４８】

[0211]インデックス永続層２０５’は、インデックスのレコードを格納する物理バケットを読み出し、それらの物理バケットに書き込むための論理バケットオペレーション２０６’を提示するようになる。これらの論理バケットオペレーション２０６’は、フラッシュアダプテーション層２０７’に提示され、フラッシュアダプテーション層２０７’は、前述のように、（インデックス付け処理の）論理バケットをフラッシュストレージデバイス上の物理バケット位置に変換する。フラッシュアダプテーション層は、よって、上記のインデックス付けアルゴリズムによって望まれたビュー及びＩＯ使用プロファイルを、下記の物理ストレージデバイス（フラッシュメモリ２１１’）によって望まれた大変異なるビューに適合させる。ここで、物理バケットオペレーション２０８’は、ランダム読出し及び集約（ブロック順次）書込みを含み、これらのランダム読出し及び集約（ブロック順次）書込みは、バケットアクセスのノンユニフォームモデルを構成する。この実施例の物理バケットオペレーションは、トリムコマンドを更に含んでもよい。

【０２４９】

[0212]物理バケットオペレーションは、物理フラッシュデバイス上の資源を追跡且つ調節する、デバイス管理層２０９’によって実装される。これらの物理デバイスオペレーション２１０’はここでは、ランダム読出し、大きい順次書込み及びトリムコマンドを含む。

【０２５０】

[0213]物理デバイス層２１１’は、そのノンユニフォーム読出し及び書込み、並びに、サイズ、アラインメント及びタイミングについての不変性によって特徴付けられる。そのような物理デバイスの例は、生のフラッシュ、相変化、ＳＳＤ、及びデバイス上に存在するフラッシュファイルシステムを有するフラッシュのうちの１以上を含む。

【０２５１】

[0214]本発明は、以下のもの等、デバイス管理層より下の追加の任意選択の強化を可能とする。

・バケットトリミング（細かいページトリミング）及びページ内のバケット追跡のモデルは、ＳＳＤ又は同等のストレージデバイスのフラッシュファイルシステムに直接組み込まれる場合、よりよい消去ブロック管理を可能とする。

【０２５２】

・フラッシュページ上へのバケットのマッピングは、抽象化である。変化ごとにフラッシュに書き込まれたデータの量を最小化することによって、それらのデバイスの寿命を延ばすために、バケットは、ＳＬＣ　ＮＡＮＤのための部分ページへマップすることができる。バケットはまた、これがシステム全体の性能にとって有益であった場合、複数のフラッシュページ上にマップすることもできる。

【０２５３】

[0215]図１８は、インデックスレコードの一例を示す。レコード１４０’は、フィンガープリント（キー）を格納するための最初の２０バイトのフィールド１４１’を含め、合計で３２バイトである。フィンガープリントは、好ましくはデータコンテンツの暗号ハッ

シュダイジェスト、例えば、ＳＨＡ－１ハッシュアルゴリズムである。例示を容易にするため、フィンガープリントを、「ＡＢ９２３４５Ｅ２０３．．．」等、１６進数で入力するのではなく、個々のフィンガープリントは、図１９～２２において、Ｐ、Ｑ、Ｒ、Ｓ、Ｔ等、単一の大文字によって指定される。やはり例示のために簡潔にするため、これらの大文字はまた、レコード全体の代理としての役割も果たすようになる。レコードのフィールドはまた、２バイトの参照カウントフィールド１４２’、５バイトの物理ブロックアドレスフィールド１４３’、１バイトのフラグフィールド１４４’、及び、４バイトの雑フィールド１４５’をも含む。ＰＢＡフィールド１４３’は、指定されたフィンガープリント１４１’のために、ディスク上で格納されたデータの物理ブロックアドレスへのポインタを含む。参照カウントは、ディスク上で格納されたデータへの参照の数を追跡する。

【０２５４】

[0216]本発明の一実施形態によれば、インデックスレコードからのフィンガープリント１４１’は、前述（図９）のルックアップ関数ｆ（キー）への入力キーとして使用され、この実施例では、関数ｆ（キー）は、４つのハッシュ関数$H_0$、$H_1$、$H_2$及び$H_3$のセットを備える。一般に、２つ以上のハッシュ関数の任意のセットを使用することができる。ハッシュ関数$H_x$は、フィンガープリントを、０及びＮ－１を含めて範囲[０，Ｎ－１]へマップし、但し、Ｎはハッシュテーブルのサイズである。この実施例では、フィンガープリント自体がハッシュであることを考えれば、ＢｉｔＦｉｅｌｄを抽出して、以下の４つのハッシュ値のファミリを生成することができる。

【０２５５】

$H_0$（ｘ）＝ｘ＜０：３１＞ｍｏｄ　Ｎ
$H_1$（ｘ）＝ｘ＜０３２：６３＞ｍｏｄ　Ｎ
$H_2$（ｘ）＝＜０６４：９５＞ｍｏｄ　Ｎ
$H_3$（ｘ）＝ｘ＜０９６：１２７＞ｍｏｄ　Ｎ

[0217]抽出されたＢｉｔＦｉｅｌｄ幅は、$log_2$（Ｎ）以上である。ばらばらのビットの任意の組み合わせを使用することができ、$log_2$（Ｎ）制約を受ける。図１８に例示されるように、最初のフィールド１４１’内のフィンガープリントのみがハッシュされて、キーが形成される。レコード１４０’の残りのコンテンツ（フィールド１４２’～１４５’）は、値又はペイロードを備える。

【０２５６】

[0218]図１９は、カッコウハッシングとして知られる、置き換えハッシングインデックス付けアルゴリズムの一例を例示する。例示を容易にするため、２つの関数のみが使用される。図１９Ａは、フィンガープリントＰがそれぞれ関数$H_0$（ｘ）及び$H_1$（ｘ）からハッシュ値２及び５を生成する一方、フィンガープリントＱがこれらの同じ関数からハッシュ値１及び３を生成する、２×３グリッドを示す。カッコウハッシングアルゴリズムは、０～６のラベルが付けられた７個のスロット（図１９Ｂ）のうち１つにＰ及びＱを入れるための、２つの代替ハッシュ値の中から選択するようになる。Ｐは、２つの位置のうち１つ、２又は５に入る可能性があり、Ｑは、２つの位置のうち１つ、１又は３に入る可能性がある。図１９Ｃに示すように、このアルゴリズムは、Ｑを最下位の空スロット１に、Ｐをスロット２に入れる。この実施例では、レコードコンテナは、１つのレコードを保持するスロットとして見なされるが、本発明はそのように限定されず、インデックス付けアルゴリズムはまた、複数のレコードを保持するバケットをもコンテナと見なすことを理解されたい。ここでは、説明を簡素化するため、単一のレコードスロットが使用される。

【０２５７】

[0219]このとき、もう１つのフィンガープリントＲが提供され、Ｒは、同じハッシュ関数から１及び２のハッシュ値を生成する（図１９Ｄのテーブルを参照）。ハッシングアルゴリズムは、Ｒを左の位置、即ちスロット１に入れ、現在のエントリＱを変位させるようになる（図１９Ｅ）。Ｑはこのとき、$H_1$（Ｑ）によって指定された他の任意選択の位置、即ち位置３に移動されるようになる。このアルゴリズムは、各レコードが空のスロットに入るまで、レコードを変位させ続けるようになる。

【０２５８】
[0219]この実施例では、「Ｒを挿入する」オペレーションを達成するため、インデックス付けアルゴリズムは、以下の読出し及び書込み要求を生成する。
１を読み出す（Ｑを得る）
２を読み出す（Ｐを得る）
１を書き込む（Ｒを書き込む）
３を読み出す（妥当性チェック）
３を書き込む（Ｑ）
[0220]最初の２つの読出しは、Ｒがインデックス内にすでに存在しているのではないことを検証するために使用される。妥当性チェック（３を読み出す）は、スロット番号３が空であるかどうかを決定し、そうである場合、Ｑがスロット３に書込み可能となり、どのエントリもスロット３に再書込みされなかったとき、このアルゴリズムは終了される。スロット３が空でなかった場合、スロット３内の現在のエントリが別のスロットに移動されることが必要となる。我々がビットマップを有する場合、スロット３のコンテンツは既知であり、そうでない場合、我々は、そのステータスを決定するために、スロット３内のエントリを読み出す必要がある。各エントリは、そのエントリが有効であるかどうかを示す有効ビットを含む。有効とは、それが使用中である（且つ、その位置を現在占有しているものが変位されなければならない）ことを意味する。有効ではないとは、その位置が空であることを意味し、処理中のレコードがそこに書込み可能である。有効ビットのコンテンツもまた、あるメモリを犠牲にして、別のビットマップに格納されうる。
【０２５９】
[0221]カッコウハッシングアルゴリズムは、空のエントリに着地するまで、複数のエントリに渡って書き込み、以前のコンテンツを変位させ続けるという点において、再帰的である。実際には、この処理はめったに１回の変位を越えることはない。
【０２６０】
[0222]インデックス付けアルゴリズムは、バケット及び個々のレコードオペレーションを共に有する。インデックス付けアルゴリズムは、１つのレコードを１つのコンテナ（スロット）に入れることとして（図１９で）上述されるが、インデックス付けアルゴリズムによって、レコードがバケット、即ち、複数のレコードを含むバケットに集約されてもよいことは理解される。よって、上記の実施例は非限定的であり、概してレコードオペレーションを例示するものである。
【０２６１】
[0223]前述のように、個々のレコードの読出し及び書込みは、フラッシュメモリにとって効率的ではないので、個々のレコードはバケットに集約される。図２０は、各々が２つ以上のレコードを含む、４つのそのようなバケットを例示し、即ち、レコード位置０及び１を有するバケット$Ｂ_0$、レコード位置２及び３を有する$Ｂ_1$、レコード位置４及び５を有する$Ｂ_2$、並びに、レコード位置６及びｘを有する$Ｂ_3$である。バケットサイズはフラッシュデバイスによって決定された最小書込みサイズ、即ち、完全ページ書込み又は部分ページ書込みの関数である（且つ、好ましくはそれに等しい）。典型的なバケットサイズは、４ＫＢであってもよい。レコードの具体的な順序付けは、バケット内で必要とされない――ルックアップオペレーション中にバケット全体が有効レコードについて検索されるので、レコードはバケット内のいかなる点でも挿入可能である。変位時、カッコウハッシングアルゴリズムによれば、バケット内のエントリはランダムに変位されうる。インデックス付けアルゴリズムは、よって、論理バケットを、ランダムな位置のように見えるものに、１つずつ書き込み、最終的にフラッシュアダプテーション層によって、フラッシュデバイスへのより大きい、物理的に隣接する（順次）書込みに集約される。
【０２６２】
[0224]図２１は、バケットエントリ１６０’の一例を例示する。４ＫＢのバケットサイズは、下にあるデバイスの最小書込みサイズ、ここでは４ＫＢのページに基づく。４ＫＢのバケットは、バケットエントリ内のレコードの数を指定する、４バイトの最初のフィー

ルド１６１’を含む。４バイトのタグフィールド１６２’は、論理バケット識別子を指定する。この識別子（タグ）は論理アドレスであり、物理的なものではない。変換テーブルは、アルゴリズムバケットアドレス（ＡＢＡ）をフラッシュバケットアドレスＦＢＡにマップする。キャッシュは、（ＣＰＵの用語で）仮想キャッシュとして動作し、各キャッシュライン（エントリ）は、タグ、この場合はＡＢＡによって識別される。アルゴリズムがレコードを要求するとき、キャッシュを通過中にそれが知ることは、要求されたＡＢＡがキャッシュされることのみであり、ＡＢＡがマップされるところ（ＦＢＡ）は、キャッシュの下端（例えば、図１０Ｃにおける、ＢＴＴへの逆ポインタ３１３’を参照）である。バケットは、複数のレコード$R_0$、$R_1$、$R_2$．．．を保持するためのフィールド１６３’を含み、各レコードは、３２バイトのサイズである。この例では、４ＫＢのバケットは、１バケットにつき（４０９６－４－４）／３２個のレコード、即ち、約１２７個のレコードを保持するようになる。

【０２６３】

[0225]図２２は、一実施形態における、バケット、ページ及び消去ブロックの相対サイズを例示する、フラッシュメモリデバイス１６４’の概略図である。物理フラッシュデバイスは、２ＧＢのサイズであるチップ（パッケージ）１６５’である。チップ上に、２つのダイ（シリコンウエハ）１６６ａ’、１６７ｂ’がある。各ダイ上に、２＾１４個の消去ブロックがあってもよく、各消去ブロック１６７’は典型的には６４ＫＢである。ページ１６８’は、書込み可能である最小サイズ、ここでは４ＫＢであり、スタック（図１７を参照）内のより上位で使用されるような、同じく４ＫＢである、バケット１６９’のサイズを決定する。

【０２６４】

[0226]図２３は、物理フラッシュデバイス上の資源を追跡且つ調節するためのデバイス管理層（図１７の２０９’）の一実施形態による、選ばれたコンポーネントを例示する。図２３Ａは、（上部に）複数のページ（バケット）１７０’と、その後に続く、どのページが有効であるか（１は有効、０は有効でない）を示す、ページアロケーションマップ１７１’を示す。この下は、将来トリムされることになるが、まだそうされていないページの、保留中トリムマップ１７２’である。ページアロケーション及び保留中トリムマップは、前述のような本発明の様々な実施形態では、バケットが有効データを保持するかどうかを決定する（図９に例示されたバケット有効テーブル２７’を参照）するために使用されうる。

【０２６５】

[0227]図２３Ｂは、消去ブロックインデックスによってインデックス付けされた、消去ブロック記述子テーブル１７５’の一例を例示する。各消去ブロック記述子エントリ１７６’は、消去済みの数１７７’、部分書込みの数１７８’、部分読出しの数１７９’、完全読出しの数１８０’、完全書込みの数１８１、及び、誤りの数１８２’を含む、複数のフィールドを含む。この情報は、本発明の様々な実施形態において前述されたように、フリー消去ブロックの生成において使用されうる。

【図１】

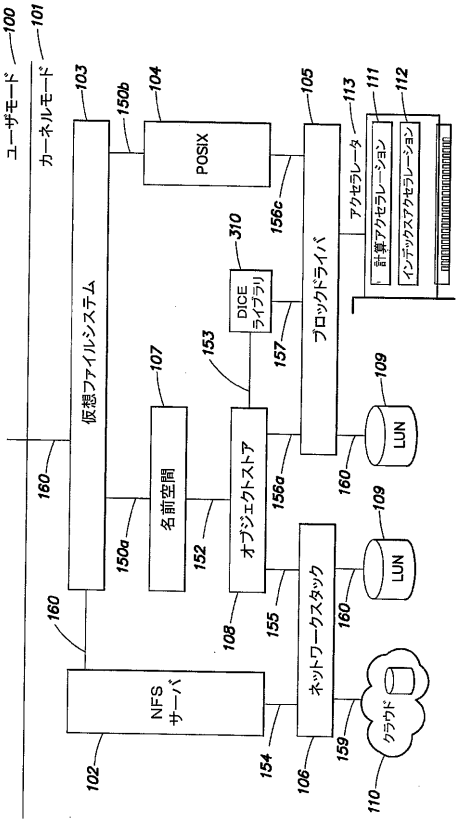


FIG. 1

【図２】

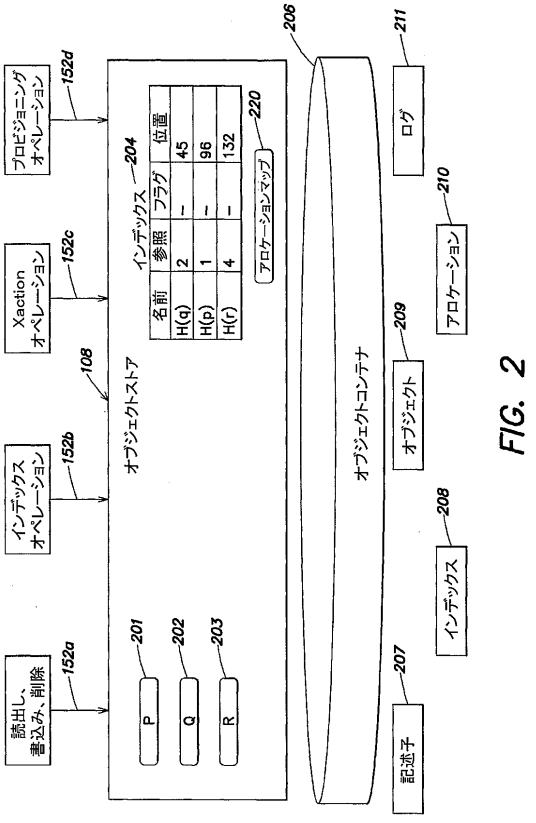


FIG. 2

【図３】

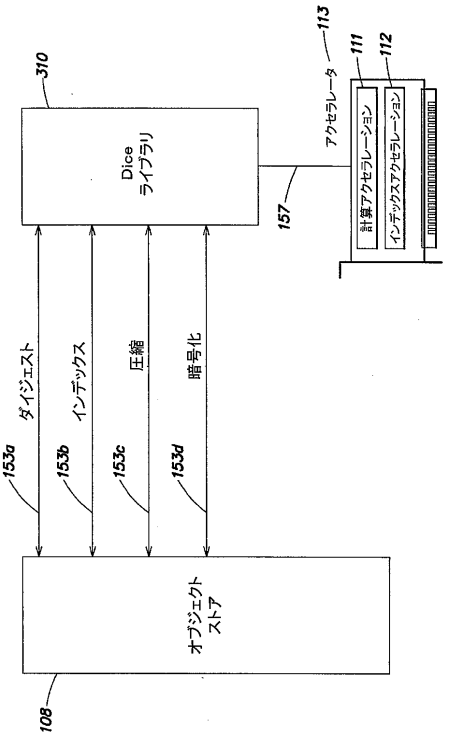


FIG. 3

【図４】

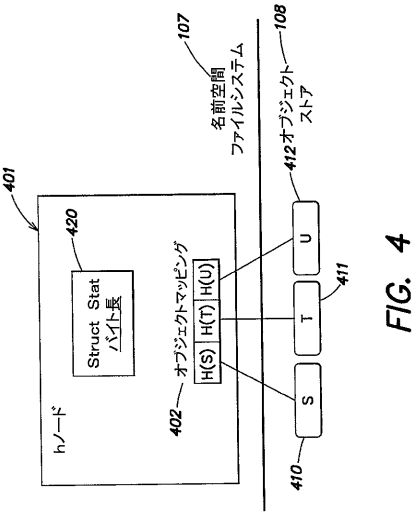


FIG. 4

【図５】

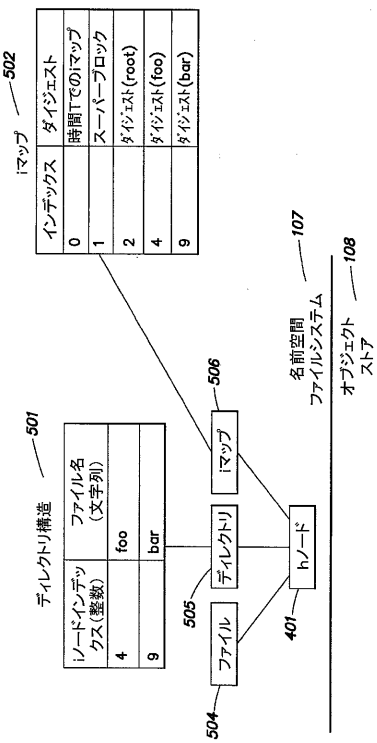


FIG. 5

【図６】

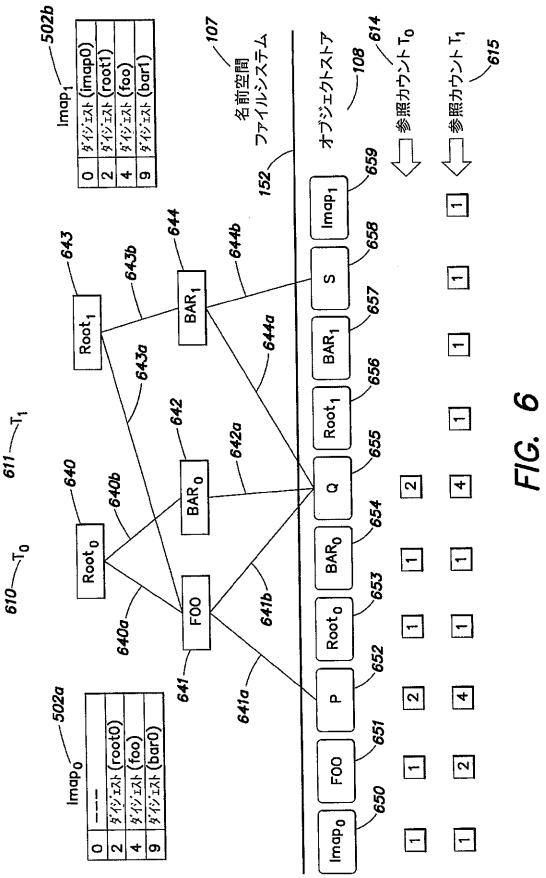


FIG. 6

【図７】

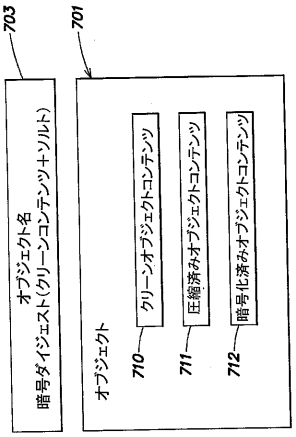


FIG. 7

【図８】

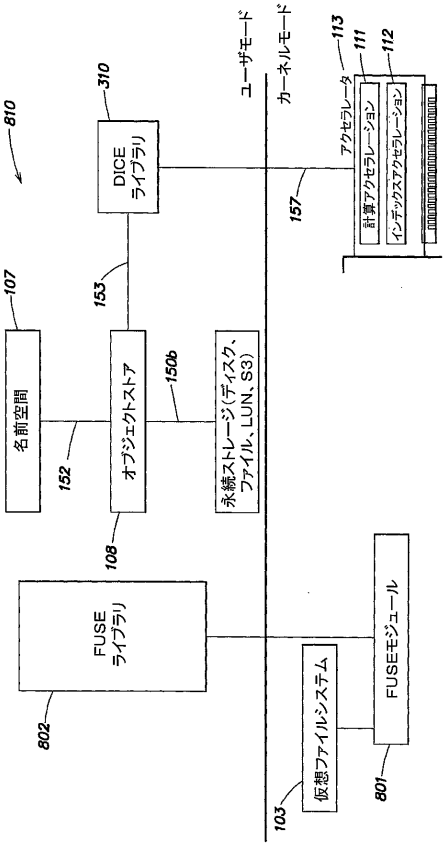


FIG. 8

【図９】

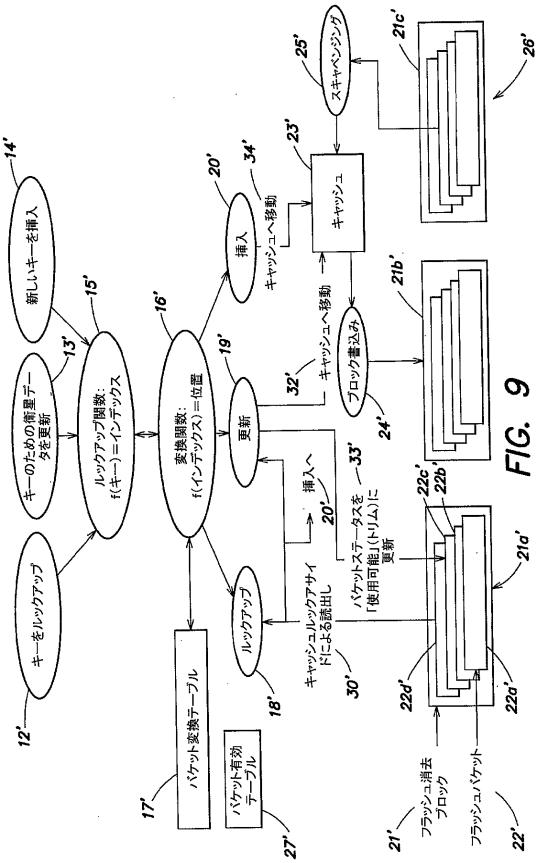


FIG. 9

【図１０Ａ】

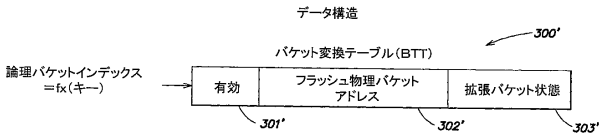


FIG. 10A

【図１０Ｂ】

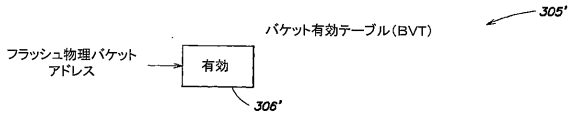


FIG. 10B

【図１０Ｃ】

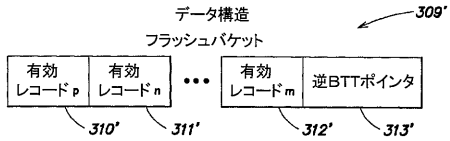


FIG. 10C

【図１０Ｄ】

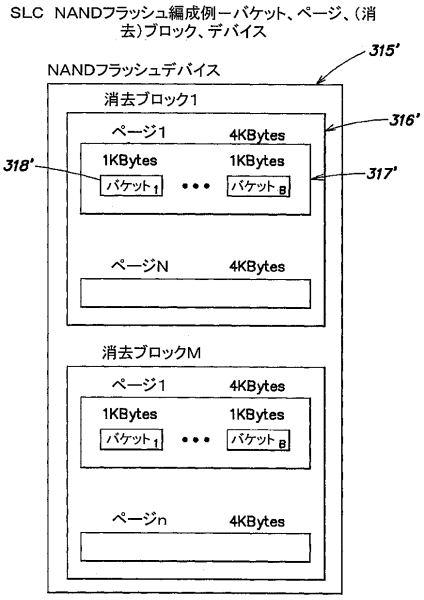


FIG. 10D

【図１１】

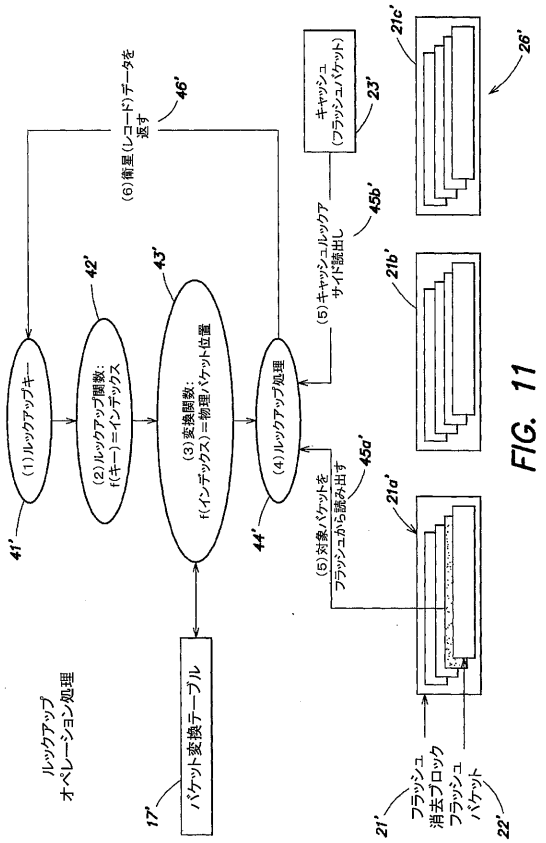


FIG. 11

【図１２】

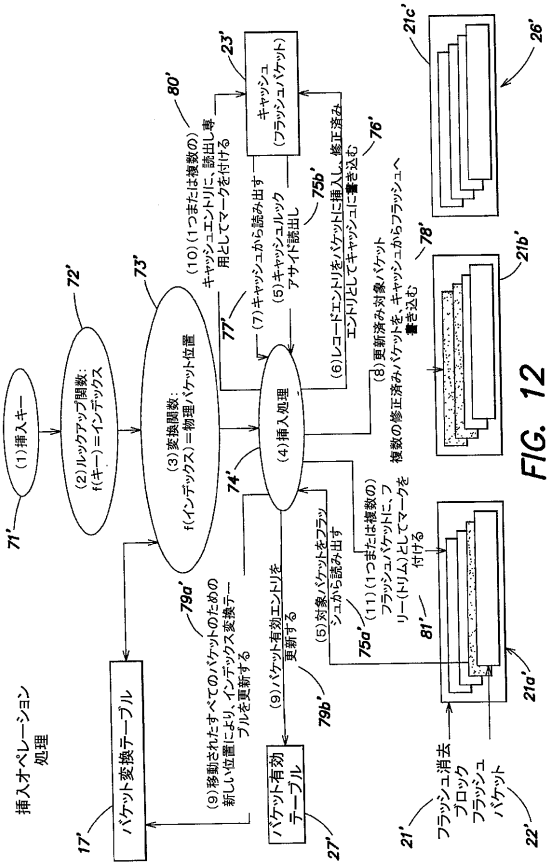


FIG. 12

【図１３】

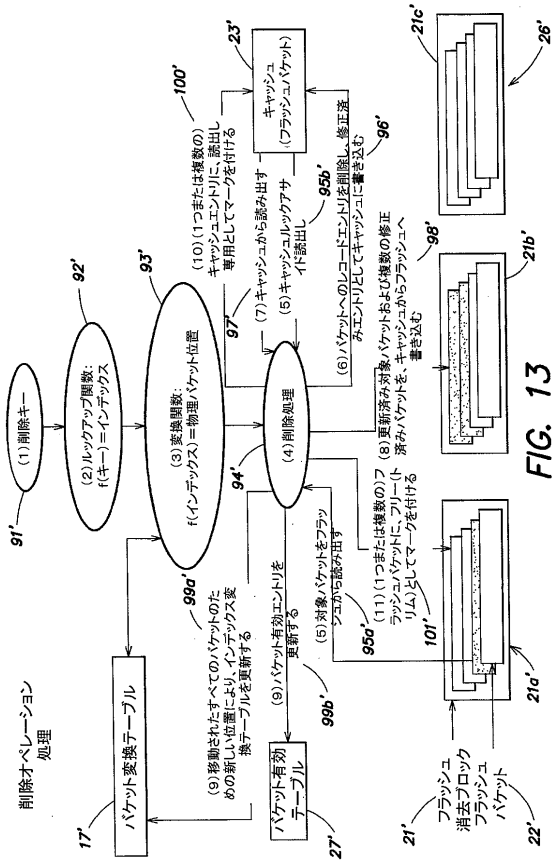


FIG. 13

【図１４】

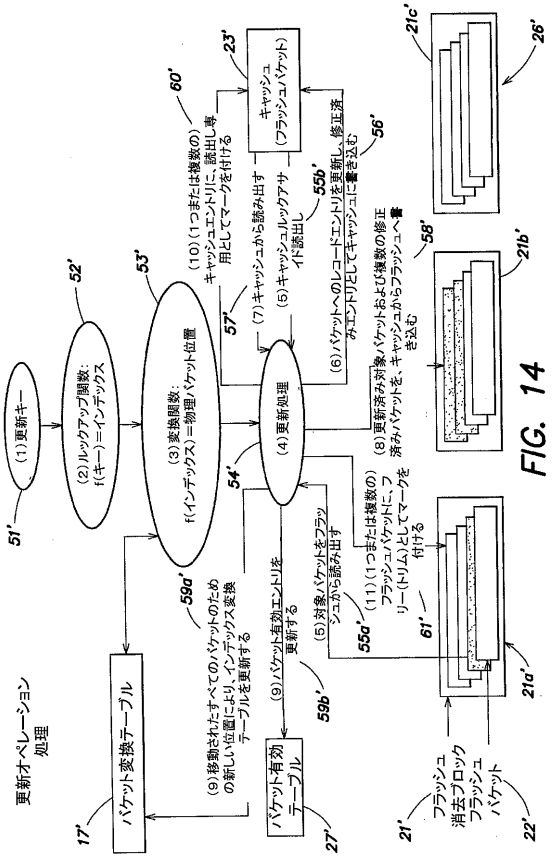


FIG. 14

【図１５Ａ】

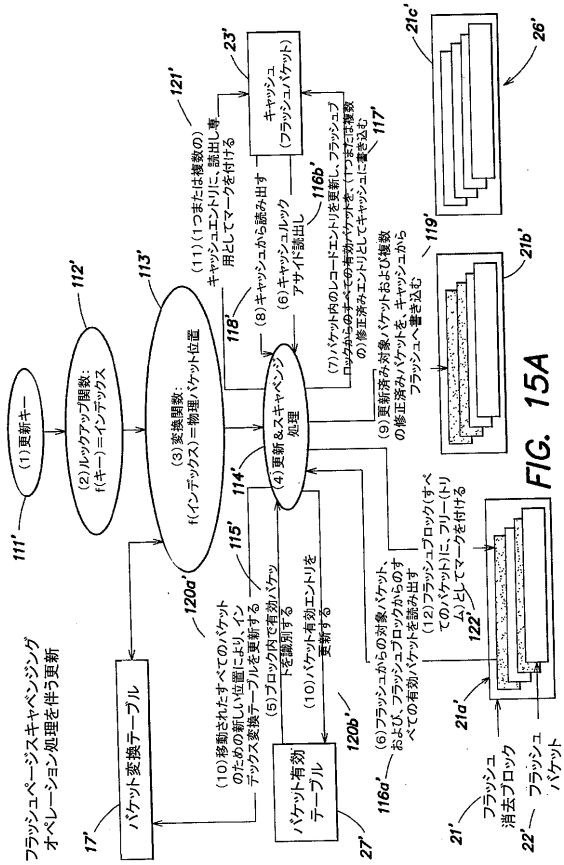


FIG. 15A

【図１５Ｂ】

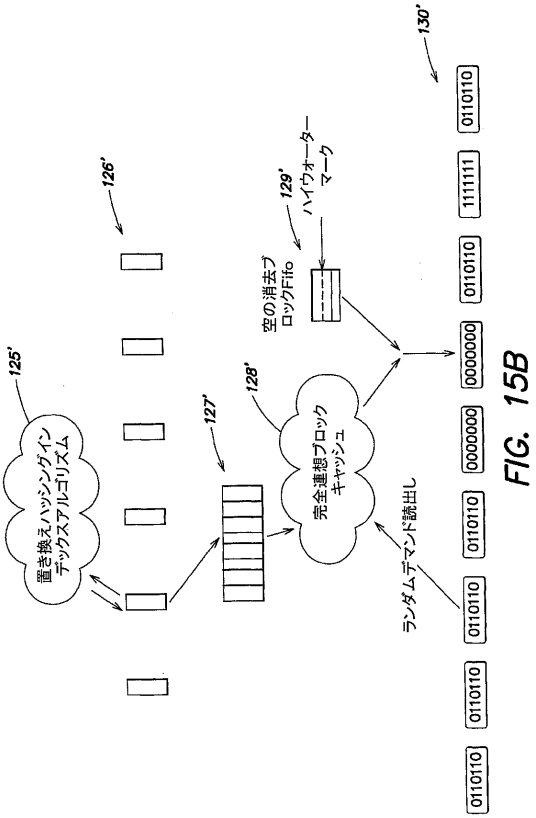


FIG. 15B

【図１６Ａ】

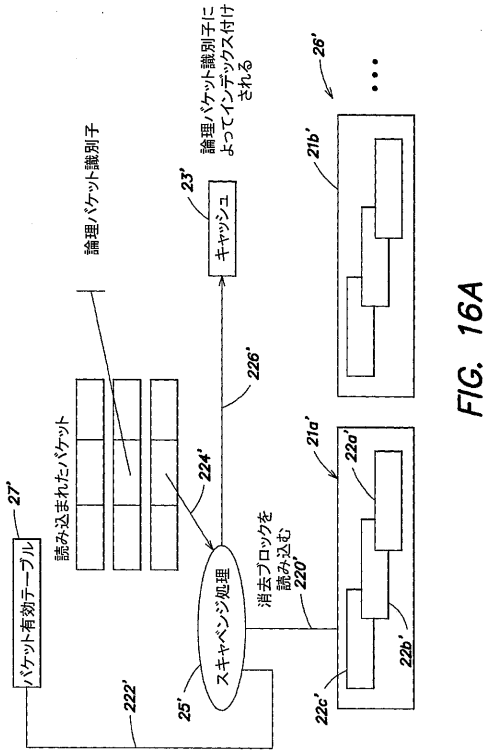


FIG. 16A

【図１６Ｂ】

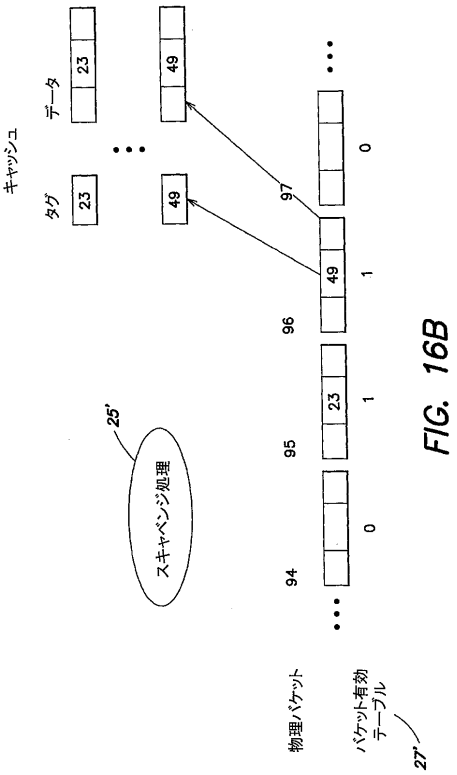


FIG. 16B

【図１７】

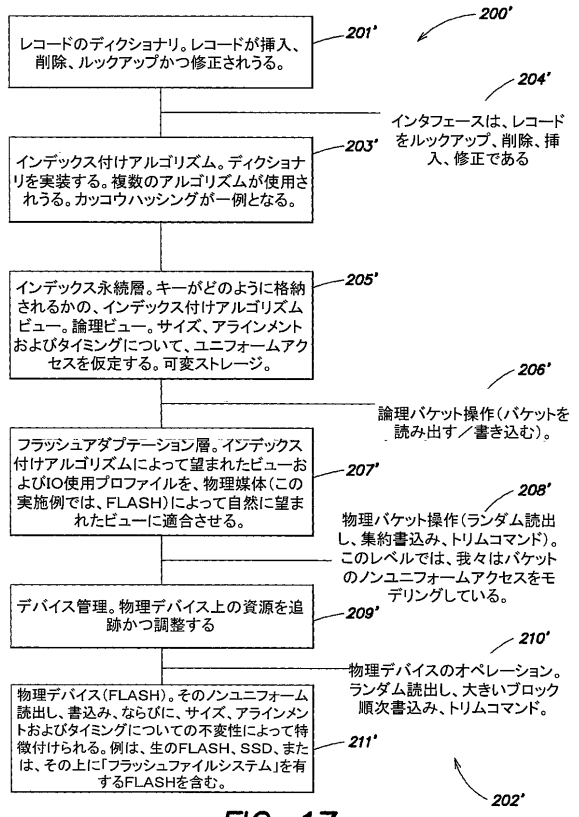


FIG. 17

【図１８】

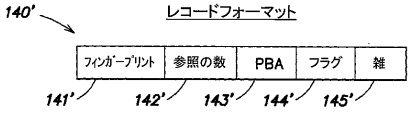


FIG. 18

【図１９】

置き換えハッシング（カッコウハッシング）

| | $H_0(X)$ | $H_1(X)$ |
|---|---|---|
| P | 2 | 5 |
| Q | 1 | 3 |

FIG. 19A

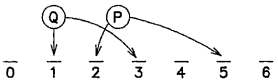


FIG. 19B

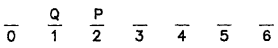


FIG. 19C

| | $H_0(R)$ | $H_1(R)$ |
|---|---|---|
| R | 1 | 2 |

FIG. 19D

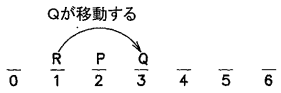


FIG. 19E

【図２０】

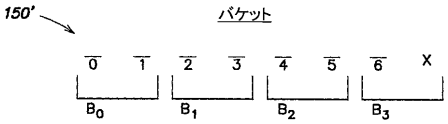


FIG. 20

【図２１】

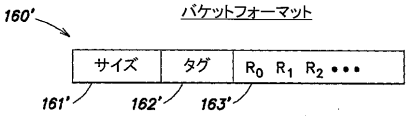


FIG. 21

【図２２】

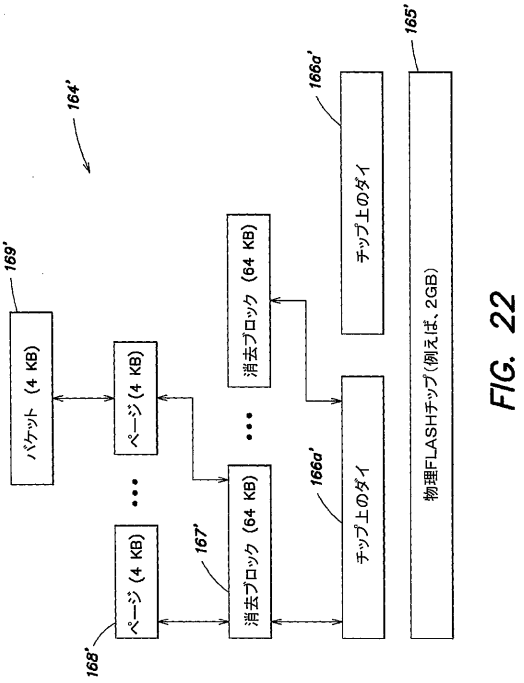


FIG. 22

【図２３】

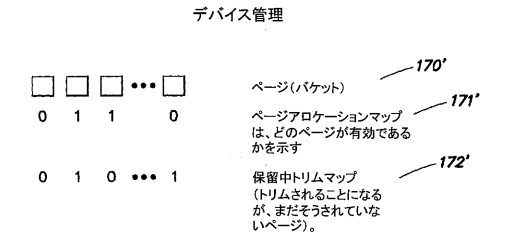


FIG. 23A

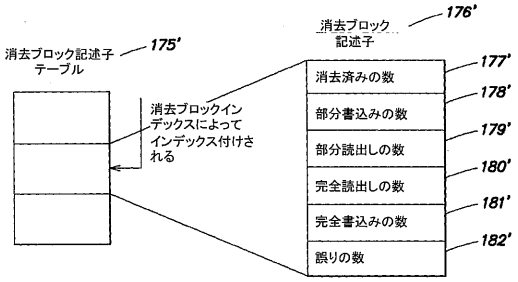


FIG. 23B

フロントページの続き

(72)発明者　ビーバーソン，アーサー・ジェイ
アメリカ合衆国マサチューセッツ州０１７１９，ボックスボロー，コブリー・ロード　３７
(72)発明者　ボウデン，ポール
アメリカ合衆国マサチューセッツ州０１５０３，バーリン，ライマン・ロード　１１３

審査官　池田　聡史

(56)参考文献　特開２００７－３０５１２２（ＪＰ，Ａ）
特開平０６－１９５２５３（ＪＰ，Ａ）
特開２００５－０１８７５８（ＪＰ，Ａ）
特開２００４－１１８８２４（ＪＰ，Ａ）
特開２００４－１５７９５８（ＪＰ，Ａ）
特開２００８－２５７７２５（ＪＰ，Ａ）
特開平１０－０８３２９９（ＪＰ，Ａ）
特開２００８－２５１０１０（ＪＰ，Ａ）
今泉貴史，ＵＮＩＸ流プログラミング：５１，ＵＮＩＸ　ＭＡＧＡＺＩＮＥ，日本，株式会社アスキー，１９９５年　1月　1日，第10巻　第1号，pp.166～173

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