

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第6345698号  
(P6345698)

(45) 発行日 平成30年6月20日 (2018. 6. 20)

(24) 登録日 平成30年6月1日 (2018. 6. 1)

(51) Int. Cl.	F I
<b>G 0 6 F 12/00 (2006. 01)</b>	G O 6 F 12/00 5 1 O B
<b>G 0 6 F 3/06 (2006. 01)</b>	G O 6 F 12/00 5 1 4 Z
	G O 6 F 3/06 3 O 1 W
	G O 6 F 3/06 3 O 1 S

請求項の数 15 (全 32 頁)

(21) 出願番号	特願2015-556181 (P2015-556181)	(73) 特許権者	515208784
(86) (22) 出願日	平成26年1月31日 (2014. 1. 31)		シンボリック アイオー コーポレーショ ン
(65) 公表番号	特表2016-509310 (P2016-509310A)		アメリカ合衆国, ニュージャージー州 O 7 7 3 3, ホルムデル, 1 0 1 クローフ ォーズ コーナー ロード
(43) 公表日	平成28年3月24日 (2016. 3. 24)		
(86) 国際出願番号	PCT/US2014/014225	(74) 代理人	100114775
(87) 国際公開番号	W02014/121109		弁理士 高岡 亮一
(87) 国際公開日	平成26年8月7日 (2014. 8. 7)	(74) 代理人	100121511
審査請求日	平成29年1月19日 (2017. 1. 19)		弁理士 小田 直
(31) 優先権主張番号	13/908, 239	(74) 代理人	100202751
(32) 優先日	平成25年6月3日 (2013. 6. 3)		弁理士 岩堀 明代
(33) 優先権主張国	米国 (US)	(74) 代理人	100191086
(31) 優先権主張番号	13/797, 093		弁理士 高橋 香元
(32) 優先日	平成25年3月12日 (2013. 3. 12)		
(33) 優先権主張国	米国 (US)		

最終頁に続く

(54) 【発明の名称】 格納されたデータにおける冗長削減

(57) 【特許請求の範囲】

【請求項 1】

データを非キャッシュ記録媒体上に格納するための方法であって、前記方法が、

i . データを非キャッシュ記録媒体に書き込むための命令を受信することであって、前記命令が、ユーザーが認識する論理ブロックアドレスおよびユーザー提供のバッファを含むことと、

i i . 前記ユーザー提供のバッファをユーザー提供のバッファユニットに分割することであって、書き込まれる各ユーザー提供のバッファユニットが、前記非キャッシュ記録媒体上の次の隣接するブロックに書き込まれることと、

i i i . 暗号化ハッシュ関数を各ユーザー提供のバッファユニットに適用し、それにより、各ユーザー提供のバッファユニットに対して生成されたハッシュ値を生成することと、

i v . コンピュータプログラムにハッシュ値テーブルをアクセスさせて、各生成されたハッシュ値が、前記ハッシュ値テーブル内のハッシュ値と重複しているかどうかを判断させる、アルゴリズムを含む前記コンピュータプログラムを起動することであって、前記ハッシュ値テーブルが、複数の格納されたハッシュ値の各々を異なる格納されたバッファユニット、および本当の論理ブロックアドレスと関連付け、

A . 前記生成されたハッシュ値が前記ハッシュ値テーブル内にない場合は、前記ユーザー提供のバッファユニットを非キャッシュ記録媒体内のブロックに書き込み、前記ハッシュ値テーブルを、前記ユーザー提供のバッファユニット、前記生成されたハッシュ値お

10

20

よび前記ユーザー提供のバッファユニットが格納されている本当の論理ブロックアドレスの相関関係を含むように更新して、メディアータ上で、前記ユーザー提供のバッファが書き込まれている位置に対応する本当の論理ブロックアドレスを、前記ユーザー提供のバッファに対する前記ユーザーが認識する論理ブロックアドレスと相互に関連付け、

B．前記生成されたハッシュ値が前記ハッシュ値テーブル内の格納されたハッシュ値と重複している場合は、矛盾があるかどうかを問い合わせ、矛盾は、重複しているハッシュ値が、格納されたバッファユニットおよび前記ユーザー提供のバッファユニットと関連付けられていて、かつ前記格納されたバッファユニットおよび前記格納されたバッファユニットの使用および前記ユーザー提供のバッファユニットが異なる値を有する状況として定義され、

10

a．矛盾がある場合は、前記ユーザー提供のバッファユニットを前記非キャッシュ記録媒体内のブロックに書き込み、前記格納されたバッファユニットと前記格納されたハッシュ値との間の前記ハッシュ値テーブル内の関連付けを非アクティブにするか、または削除して、前記ハッシュ値テーブルを、前記ユーザー提供のバッファユニット、前記生成されたハッシュ値および前記ユーザー提供のバッファユニットが格納されている本当の論理ブロックアドレスの相関関係を含むように更新して、前記ユーザー提供のバッファユニットが書き込まれている位置に対応する本当の論理ブロックアドレスおよび前記ユーザーが認識する論理ブロックアドレスを前記メディアータ上に書き込み、

b．矛盾がない場合は、前記ユーザー提供のバッファユニットと同じである、前記非キャッシュ記録媒体上に格納されたバッファユニットの本当の論理ブロックアドレスを前記メディアータ上に書き込み、それを、前記ユーザー提供のバッファユニットを前記非キャッシュ記録媒体上に書き込むことなく、前記ユーザー提供のバッファに対する前記ユーザーが認識する論理ブロックアドレスと相互に関連付けることと、を含む、方法。

20

#### 【請求項2】

前記方法が、複数のユーザー提供のバッファユニットの各々に対して、( i i i ) ~ ( i v ) を繰り返すことをさらに含み、全体として、前記複数のユーザー提供のバッファユニットがファイルに対応する、請求項1に記載の方法。

#### 【請求項3】

前記ユーザー提供のバッファがサイズにおいて 5 1 2 B ( バイト ) と 2 M B ( メガバイト ) との間にある、請求項1に記載の方法。

30

#### 【請求項4】

ユーザーが認識する論理ブロックアドレスに対して、前記メディアータが、前記本当の論理ブロックアドレスを複数回、相互に関連付ける、請求項1に記載の方法。

#### 【請求項5】

前記バッファユニットがサイズにおいて 5 1 2 B ( バイト ) または 4 K B ( キロバイト ) である、請求項1に記載の方法。

#### 【請求項6】

前記ハッシュ値がサイズにおいて 8 B ( バイト ) または 1 6 B ( バイト ) である、請求項5に記載の方法。

40

#### 【請求項7】

前記ユーザー提供のバッファユニットが、前処理によって形成され、前記前処理が：

( i ) 複数のデジタル2値信号を受信することであって、前記デジタル2値信号が複数のチャンクレットに編成されており、各チャンクレットがNビット長で、Nが1より大きい整数であり、前記チャンクレットが順序を有することと、

( i i ) X 個のマーカのセットから、1つのマーカを各チャンクレットに割り当てて、複数のマーカのセットを形成することであって、Xがチャンクレット内のビットの異なる組合せの数以下であり、同一のサブユニットが前記マーカに割り当てられていることと、

( i i i ) 前記マーカをバッファユニットとして使用することと、

50

を含む、請求項 1 に記載の方法。

【請求項 8】

前記ユーザー提供のバッファユニットが、前処理によって形成され、前記前処理が：

( i ) 複数のデジタル 2 値信号を受信することであって、前記デジタル 2 値信号が複数のチャンクレットに編成されており、各チャンクレットが N ビット長で、N が 1 より大きい整数であり、前記チャンクレットが順序を有することと、

( i i ) 各チャンクレットを均一なサイズのサブユニットに分割して、X 個のマーカのセットから、1 つのマーカを各サブユニットに割り当てて、複数のマーカのセットを形成することであって、X がサブユニット内のビットの異なる組合せの数以下であり、同一のサブユニットが前記マーカに割り当てられていることと、

( i i i ) 前記マーカをバッファユニットとして使用することと、  
を含む、請求項 1 に記載の方法。

10

【請求項 9】

少なくとも 1 つのマーカが、サイズにおいてサブユニットより小さい、請求項 8 に記載の方法。

【請求項 10】

ステップ ( i i ) が、ビットマーカテーブルおよび頻度変換器のうちの 1 つにアクセスすることを含む、請求項 7 ~ 8 のいずれか一項に記載の方法。

【請求項 11】

データを格納するためのシステムであって、前記システムが、

( a ) 永続的メモリであって、前記永続的メモリが、格納されたバッファユニットを格納されたハッシュ値および本当の論理ブロックアドレスと関連付けるように構成されているハッシュ値テーブルを格納する、永続的メモリと、

( b ) 持続性媒体内に格納されているコンピュータプログラムを格納しているか、またはそれに動作可能に結合されている、中央処理装置であって、前記コンピュータプログラムが、実行される際に、自動的に

i . ハッシュ値アルゴリズムを 1 つ以上のユーザー提供のバッファユニットの各々に適用して、生成されたハッシュ値を作成し、

i i . 前記生成されたハッシュ値が、格納されたバッファユニットと関連付けられている前記ハッシュ値テーブル内の格納されたハッシュ値の重複であるかどうかを判断し、  
そうである場合は、矛盾が存在するかどうかを判断し、前記矛盾が、2 つの異なるバッファユニットと関連付けられているハッシュ値として定義されて、矛盾が存在する場合は、前記ハッシュ値テーブルを更新して、前記ハッシュ値テーブル内のハッシュ値を、前記格納されたバッファユニットではなく、前記ユーザー提供のバッファユニットと関連付けさせる、

実行可能コードを含む、中央処理装置と、

( c ) 非キャッシュ記録媒体であって、前記非キャッシュ記録媒体がブロックレベル記憶に対して構成されており、書き込まれる各ユーザー提供のバッファユニットが、前記非キャッシュ記録媒体上の次の隣接するブロックに書き込まれる、非キャッシュ記録媒体と

、  
( d ) メディエータであって、前記メディエータが、本当の論理ブロックアドレスとユーザーが認識する論理ブロックアドレスとの相関関係を格納する、メディエータと、  
を備える、システム。

【請求項 12】

前記コンピュータプログラムが、

i . 前記生成されたハッシュ値が、前記ハッシュ値テーブル内のハッシュ値の重複ではないか、または

i i . 前記生成されたハッシュ値が、格納されたハッシュ値の重複であり、かつ矛盾があるか、のいずれか

の場合に限り、前記ユーザー提供のバッファユニットを前記非キャッシュ記録媒体に書き

20

30

40

50

込む、請求項 1 1 に記載のシステム。

【請求項 1 3】

複数のユーザーファイルに対して、ユーザーが認識する論理ブロックアドレスと本当の論理ブロックアドレスの相関関係があり、かつ、異なるファイルに対する複数の相関関係内に、前記本当の論理ブロックアドレスであるが、異なるユーザーが認識する論理ブロックアドレスが 1 つ以上ある、請求項 1 1 に記載のシステム。

【請求項 1 4】

前記システムが取得モジュールをさらに備え、前記取得モジュールが、前記メディアータにアクセスして、ハッシュ値テーブルにアクセスすることなく、複数のバッファユニットを、前記メディアータによって指示される順序で再結合することにより、データファイルを再構成するように構成され、かつ、前記メディアータによって指示される順序が、前記非キャッシュ記録媒体上での前記バッファユニットの順序とは異なる、請求項 1 1 ~ 1 3 のいずれか一項に記載のシステム。

10

【請求項 1 5】

コンピュータプログラムを格納しているコンピュータ可読記憶媒体であって、前記コンピュータプログラムが、コンピュータ上で実行される際に、前記コンピュータに、請求項 1 ~ 1 0 のいずれか一項に記載の方法を含む、ファイルシステム内で重複を排除し、データブロックを管理するための方法を実装させる、コンピュータ可読記憶媒体。

【発明の詳細な説明】

【技術分野】

20

【0 0 0 1】

本発明は、データの格納に関する。

【背景技術】

【0 0 0 2】

2 1 世紀は、人々および企業が生成および格納するデジタル化された情報の量において急激な増加を示してきた。この情報は、通常、ディスクなどの磁気面上に格納されている電子データから成る。これらのディスクは、サイズがサブマイクロメータで、データの個々のバイナリ断片を格納可能な、小さな領域を含む。

【0 0 0 3】

任意の所与の実体が格納する膨大な量のデータ内に、しばしば、情報の著しい重複がある。例えば、同じ会社のレターヘッドが何千もの文書に出現し得、そして、このデータに対応する各ファイルは、そのレターヘッドをコード化するビットを含むであろう。歴史的に、多数の実体は、この種の重複が自身のファイル内に存在すること、および同じ情報を重複して格納する非効率性は事業を行うためのコストであることを容認してきた。

30

【0 0 0 4】

記憶装置に対する費用が増加し、記憶装置の可用性が低下してきているので、実体は、ファイル間で重複した情報の全てよりは少ない情報を格納するための手段を探り始めている。理論上は、重複した情報の格納を回避するか、または重複した情報が格納される回数を最小限にしようと努める実体は、自身のデータセット内で一意のビットまたはバイトパターンを識別し、その一意のビットまたはバイトパターンを最小限の回数だけ、格納しようとし得る。これらの方法を実施するために、新しいファイルが、格納のために準備されているとき、それらのファイル内の情報が、既に格納された情報の参照セットと比較され得、検討されているビットまたはバイトパターンが一意の場合に限り、格納されるであろう。一意でない場合、冗長データは、その重複である格納されたデータを指す、データよりも小さいサイズの参照と置き換えられるであろう。

40

【0 0 0 5】

重複した情報が格納される回数を削減することの目標は：( 1 ) 冗長を検査するための十分な速度を維持すること；( 2 ) 取得のためのデータの再構成において十分な速度を維持すること；( 3 ) 冗長の検査または元のファイルに対応する情報の格納のいずれかのプロセス中に、データが失われないことを確実にすること；( 4 ) 格納された情報に対する

50

権限のないアクセスを防ぐこと；および（５）データのスナップショットの取得、データの複製およびデータの復元の、全部ではないにしろ、１つ以上と関連して使用され得る効率的な技術および方法を提供すること、を含むが、それらに限定されない、いくつかの課題を提示する。本発明の様々な実施形態は、これらの課題の１つ以上を克服することに関する。

【発明の概要】

【課題を解決するための手段】

【０００６】

本発明は、冗長データが複数回、不必要に格納される程度を最小限にしながら、データを格納および取得する効率を向上させるための方法、システムおよびコンピュータプログラム製品を提供する。本発明の様々な実施形態を使用することにより、データを効率的に格納してアクセスすることができる。本発明のこれらの様々な実施形態を通して、データを変形し得、かつ／または変形もしくは変換されたデータが格納される物理装置を変更し得る。これは、実行される際に、本発明の方法またはプロセスの１つ以上を実施する、コンピュータプログラム製品を含むか、またはそれに動作可能に結合されているコンピュータを採用する自動化プロセスを通して達成され得る。これらの方法またはプロセスは、例えば、コンピュータアルゴリズムもしくはスクリプトで具現化されるか、またはそれを含み、任意選択で、１つ以上のモジュールを通して、システムによって実施され得る。

【０００７】

第１の実施形態によれば、本発明は、データを非キャッシュ記録媒体上に格納するための方法を対象とし、本方法は：（ｉ）データを非キャッシュ記録媒体に書き込むための命令を受信することであって、その命令が、ユーザーが認識する論理ブロックアドレス（「ＬＢＡ」）およびユーザー提供のバッファを含み、ユーザー提供のバッファが、例えば、５１２バイト～２メガバイト、または５１２バイト～６４Ｋから成る、データを非キャッシュ記録媒体に書き込むための命令を受信すること；（ｉｉ）ユーザー提供のバッファをユーザー提供のバッファユニットに分割して、暗号化ハッシュ関数をユーザー提供のバッファユニットの各々に適用し、それにより、生成されたハッシュ値を生成すること；（ｉｉｉ）コンピュータプログラム製品にハッシュ値テーブルをアクセスさせて、生成されたハッシュ値が、ハッシュ値テーブル内の格納されたハッシュ値と重複しているかを判断させるアルゴリズムを含むコンピュータプログラム製品を起動することであって、ハッシュ値テーブルが、複数の格納されたハッシュ値の各々を異なる格納されたバッファユニットおよび本当の論理ブロックアドレスと相互に関連付けて；（Ａ）生成されたハッシュ値がハッシュ値テーブル内にない場合は、ユーザー提供のバッファユニットを非キャッシュ記録媒体内のブロックに書き込み、ユーザー提供のバッファユニット、生成されたハッシュ値およびユーザー提供のバッファユニットが格納されている本当の論理ブロックアドレスの相関関係を含むようにハッシュ値テーブルを更新して、ユーザー提供のバッファユニットが書き込まれている位置に対応する本当の論理ブロックアドレスおよびユーザー提供のバッファに対するユーザーが認識する論理ブロックアドレス（または複数のアドレス）をメディアータ上に書き込み、（Ｂ）生成されたハッシュ値がハッシュ値テーブル内に格納されたハッシュ値と重複している場合は、矛盾があるかどうかを問い合わせ、矛盾は、同じハッシュ値が格納されたバッファユニットおよび現在のユーザー提供のバッファユニットと関連付けられていて、その２つのバッファユニットが異なる内容を有する状況として定義され、（ａ）矛盾がある場合は、ユーザー提供のバッファユニットを非キャッシュ記録媒体内のブロックに書き込み、格納されたバッファユニットと格納されたハッシュ値との間のハッシュ値テーブル内の関連付けを非アクティブにするか、または削除して、ユーザー提供のバッファユニット、生成されたハッシュ値およびユーザー提供のバッファユニットが格納されている本当の論理ブロックアドレスの相関関係を含むようにハッシュ値テーブルを更新して、ユーザー提供のバッファユニットが書き込まれている位置に対応する本当の論理ブロックアドレスおよびユーザーが認識する論理ブロックアドレスをメディアータ上に書き込み、（ｂ）矛盾がない場合は、ユーザー生成のバッファユニットと同じで

10

20

30

40

50

ある、非キャッシュ記録媒体上に格納されたバッファユニットの本当の論理ブロックアドレスをメディアータ上に書き込み、それを、ユーザー提供のバッファユニットを非キャッシュ記録媒体上に書き込むことなく、ユーザー提供のバッファに対してユーザーが認識する論理ブロックアドレスと相互に関連付ける、コンピュータプログラム製品を起動すること、を含む。

【0008】

ステップ(A)において、ユーザー提供のバッファユニットを書き込む際に、当業者は、本方法は、テーブル内のハッシュ値と関連付けられていないと判断されているユーザー提供のバッファユニットを書き込む必要があることを理解するであろう。

【0009】

しばしば、ユーザーは、ハッシュ値アルゴリズムが入力として受け入れるように構成されているユーザー提供のバッファユニットよりも大きい、ストリーム内またはユニット内のデータ(ユーザー提供のバッファ)を提供するであろう。これらの場合、ユーザー提供のバッファユニットは、ホストによって送信された生データをさらに小さいユニットにフラグメント化すること(ブレーキングとも呼ばれる)により形成され得、それは、ホストがそれらをフラグメント化するか、または本発明のシステムもしくは方法がそれを行うかどうかに関わらず、ユーザー提供のバッファユニットと見なされ得る。従って、これらのフラグメント化されたユーザー提供のバッファユニットは、暗号化ハッシュ関数に対する入力として機能し得る。限定されない例として、ユーザー提供されたデータは、16K~2MBであり得、各フラグメント化されたユーザー提供のバッファユニットは、512バイト~4Kまで、例えば、512バイトまたは4Kである。従って、いくつかの実施形態では、フラグメント化されたユーザー提供のバッファユニットは、フラグメンテーション前のユーザー提供のバッファサイズのわずか1/4またはわずか1/16またはわずか1/64である。フラグメンテーションのステップが、ハッシュ値アルゴリズムに入る前に使用される場合、ハッシュ値テーブルは、ユーザー提供のバッファユニットとハッシュ値の相関関係を含み得、記憶装置への書き込みは、より大きなデータバッファユニットではなく、ユーザー提供のバッファユニットにフラグメント化され、メディアータが、ユーザー提供のバッファユニットのユーザーが認識するアドレス(または複数のアドレス)を複数のフラグメント化されたユーザー提供のバッファユニットと相互に関連付けるであろう。

【0010】

本発明の本方法の様々なステップは、1つ以上のモジュール、例えば、バッファおよびユーザーが認識する論理ブロックアドレスの受信モジュール、フラグメンテーションモジュール、ハッシュ値検索モジュール、ハッシュ値の重複分析モジュール、矛盾モジュール、および書き込みモジュール内に格納され得る。同様に、ファイルの読取りおよび再構成モジュールがあり得る。これらのモジュールは、持続性媒体内に実行可能コードの形で格納され得る。

【0011】

第2の実施形態によれば、本発明は、データを格納するためのシステムを提供し、本システムは:(a)永続的メモリであって、格納されたバッファユニットを格納されたハッシュ値および本当の論理ブロックアドレスと関連付けるように構成されている、ハッシュ値テーブルを格納する、永続的メモリ;(b)持続性媒体内に格納されているコンピュータプログラム製品を含むか、またはそれに動作可能に結合されている、中央処理装置であって、コンピュータプログラム製品が、実行される際に、自動的に(i)ハッシュ値アルゴリズムを1つ以上のユーザー提供のバッファユニットの各々に適用して、生成されたハッシュ値を生成し、(ii)その生成されたハッシュ値が、格納されたバッファユニットと関連付けられているハッシュ値テーブル内の格納されたハッシュ値と重複しているかどうかを判断し、そうである場合は、矛盾が存在するかどうかを判断し(矛盾は、2つの異なるバッファユニットと関連付けられているハッシュ値として定義される)、矛盾が存在する場合は、ハッシュ値テーブルを更新して、テーブル内のハッシュ値を、格納されたバッファユニットではなく、ユーザー提供のバッファユニットと関連付けさせる、実行可能

コードを含む、中央処理装置；（ｃ）非キャッシュ記録媒体であって、ブロックレベル記憶に対して構成されている、非キャッシュ記録媒体；および（ｄ）メディエータであって、本当の論理ブロックアドレスとユーザーが認識する論理ブロックアドレスとの相関関係を格納する、メディエータ；を含む。

【００１２】

第３の実施形態によれば、本発明は、コンピュータ可読プログラムを含む、持続性コンピュータ利用可能媒体を含むコンピュータプログラム製品を提供し、コンピュータ可読プログラムは、コンピュータ上で実行される際に、コンピュータに、本発明の方法のいずれかを含む、ファイルシステム内での重複排除、およびデータブロックの管理のための方法を実装させる。

10

【発明の効果】

【００１３】

ほとんどの状況で、以前に書き込まれたデータの重複である大きなバッファユニットは、非キャッシュ記録媒体（NCM）に再度書き込む必要がないので、本発明の様々な実施形態を通して、データの格納および取得の効率が向上できる。代わりに、それが指すデータを収容する記憶装置よりもサイズが小さく、物理的に別個の構造中にある、メディエータ上のポインタが、コンピュータに、そのデータの以前に格納されたコピーを指示する。効率の向上は、一般に適用される方法で使用されるよりも少ない記憶空間を使用し、情報を格納および／または取得する活動において、より少ない時間および努力を投資することにより、実現され得る。さらにいくつかの実施形態では、本発明は、文書の格納および取得における速度の向上をもたらす。従って、本発明の技術および方法は、データを格納するために要求される物理的な記憶装置の総量を削減するのに役立つ。これは、重複したデータが書き込まれて格納される回数を最小限にすることにより、また、重複するデータがある状況では、以前に格納されたデータをポイントするメディエータを使用することにより、達成される。

20

【図面の簡単な説明】

【００１４】

【図１】本発明の実施形態に従ってデータを書き込むための方法の表現である。

【図２】本発明の方法に従って矛盾を解決するためのプロトコルの表現である。

【図３】本発明の一実施形態に従って、情報を読み取るための方法の表現である。

30

【発明を実施するための形態】

【００１５】

ここで、本発明の様々な実施形態に対する参照を詳細に行うが、その例が付随する図に示されている。以下の詳細な記述では、多数の具体的な詳細が、本発明の完全な理解を提供するために記載されている。しかし、別段の指示がないか、またはコンテキストから暗黙的でない限り、詳細は例であることを意図し、本発明の範囲をいかなる方法でも制限すると見なされるべきでない。

【００１６】

定義

【００１７】

別段の指示がないか、またはコンテキストから暗黙的でない限り、次の用語および句は、以下に規定する意味を有する。

40

【００１８】

用語「ビット」は、二進数字を指す。それは、２つの値のうちの１つを有することができる。各値は、０または１のいずれかによって表され得る。

【００１９】

用語「ブロック」は、所定の長さを有するデータのバイトまたはビットのシーケンスを指す。記録媒体上で、物理媒体は、ブロックサイズによって定義されるユニットに分割され得る。記録媒体上の各ブロックは、論理ブロックアドレスによって識別され得る。業界では、現在のところ、５１２バイトがブロックの標準的なサイズである。しかし、４０９

50

6 バイトを標準として使用する動向がある。さらに、当業者であれば理解するように、句「ブロックサイズ」および「セクターサイズ」はしばしば、当業者によって区別しないで使用される。

【0020】

句「ブート可能性コード」、「ブート可能性情報」および「ブート可能性特徴」は、ブート可能状態に入るための手段を提供する情報を指し、ブートセクター上に格納され得る。ブートセクターは、ファームウェアによってRAM（ランダムアクセスメモリ）内にロードされるように構成されている機械コードを含み得、それは、その結果として、ブートプロセスがプログラムを記憶装置から、または記憶装置にロードするのを可能にする。例として、マスターブートレコードが、アクティブなパーティションを見つけて、ボリュームブートレコードを呼び出すコードを含み得、ボリュームブートレコードは、オペレーティングシステムまたは他のスタンドアロンプログラムをロードして呼び出すためのコードを含み得る。

10

【0021】

句「バッファユニット」は、ハッシュ値アルゴリズムへの入力としての使用に対して互換性があるサイズの、一連のビットを指す。バッファユニットは、チャンクレットと同じサイズであり得る。しかし、いくつかの実施形態では、それは、チャンクレットのサイズの分数であるか、またはチャンクレットのサイズの倍数であり得る。

【0022】

用語「バイト」は、8ビットのシーケンスを指す。

20

【0023】

用語「キャッシュ」は、データに対する将来の要求がより迅速に対応されるため、またはバッファリングを目的として、データが一時的に格納される位置を指す。L1キャッシュ（レベル1キャッシュ）は、例えば、プロセッサコアと統合されている、スタティックメモリを指す。L1キャッシュは、CPU（汎用処理装置）が同じデータを複数回アクセスする場合にデータアクセス速度を向上するために使用され得る。L2キャッシュ（レベル2キャッシュ）は、通常、L1キャッシュよりも大きく、データファイルが検索されたが、L1内で見つからなかった場合に、検索は、外部メモリを目指す前に、L2キャッシュで行われ得る。いくつかの実施形態では、L1キャッシュは、中央処理装置内ではない。代わりに、それは、DDR、DIMMまたはDRAM内に置かれ得る。追加または代替として、L2キャッシュは、PCI 2.0/3.0の一部であり得、それは、マザーボードに入る。従って、L1キャッシュおよびL2キャッシュの各々は、マザーボードの別々の部分にあり得る。いくつかの実施形態では、本発明の方法が実装される際に、ハッシュ値テーブルは、L2キャッシュ内に常駐する。

30

【0024】

用語「チャンクレット」は、セクタークラスタに対応し得るビットのセットを指す。チャンクレットのサイズは、記憶システムによって決定され得、チャンクレットサイズを有し得る。慣例的に、チャンクレットサイズは、CHS方式によって導出されたが、CHS方式は、シリンダー、ヘッドおよびセクターを、ハードディスク上にそれらが出現したところで定義したタプルによってブロックをアドレス指定した。ごく最近では、チャンクレットサイズは、論理ブロックアドレス（LBA）測定から導出されている。例として、チャンクレットサイズは、512B、1K、2K、4K、8K、16K、32K、64Kまたは1MBであり得る。当業者として、1K=1024Bが分かっている。チャンクレットは、生データとしてホストから受信され得る。

40

【0025】

用語「矛盾」は、異なる入力（例えば、バッファユニット）に対する、ハッシュ値アルゴリズムなどの、関数による、同じ出力（例えば、ハッシュ値）の生成の出現を指す。

【0026】

「ファイル」は、ビットまたはバイトで測定され得る長さをもつサイズのファイルを提供するために結合する関連したバイトまたはビットの集合である。ファイルは、チャンク

50



レットよりも小さいか、チャンクレットと同じサイズか、またはチャンクレットよりも大きい可能性がある。

【 0 0 2 7 】

句「ファイル名」は、コンピュータが特定のファイルを識別して、そのファイルを他のファイルから区別できるようにする表記法またはコードを指す。

【 0 0 2 8 】

句「ファイルシステム」は、ファイルのセットを格納、取得、および更新するために使用される抽象化を指す。従って、ファイルシステムは、ファイルのデータおよびメタデータへのアクセス、ならびにデータを含む記憶装置上の利用可能な空間を管理するために使用されるツールである。いくつかのファイルシステムは、例えば、サーバー上に常駐し得る。ファイルシステムの例は、Unix（登録商標）ファイルシステムおよびその関連したディレクトリテーブルおよびiノード、Windows（登録商標）FAT16およびFAT32ファイルシステム（FATは、ファイルアロケーションテーブルを指す）、マスターファイルテーブルに基づく、Windows（登録商標）NTFS、ならびに、HFSまたはHFSプラスを使用する、Apple Mac OS Xを含むが、それらに制限されない。

【 0 0 2 9 】

句「ハッシュ関数」、「暗号化ハッシュ関数」、「暗号化ハッシュ関数値アルゴリズム」、および「ハッシュ関数値アルゴリズム」は、（同じ長さまたは可変長の）大きなデータセットを、特定のハッシュ関数に対して固定長を有する、より小さいデータセットにマッピングする、アルゴリズムまたはサブルーチンを指す。「ハッシュ関数値」は、ハッシュ関数アルゴリズムの適用後に返される出力を指す。アルゴリズムが返す値は、ハッシュ値、ハッシュコード、ハッシュ合計、チェックサムまたはハッシュとも呼ばれ得る。例えば、MD5を使用する場合、出力は128ビットであり、他方、SHA-1を使用する場合、出力は160ビットである。従って、いくつかの実施形態では、ハッシュ値は32～512ビットの長さである。

【 0 0 3 0 】

用語「ホスト」、「ユーザー」および「イニシエータ」は、区別しないで使用され得、データを格納のために、本発明のデータ記憶および取得仲介システムに送信するエンティティまたはシステムを指す。ホストは、1つ以上のタイプの文書またはファイルに対応するデータを送信して、データを受信し得る。好ましくは、任意の入力/出力（I/O）ストリーム内で、データは、単一の文書タイプのファイルに対応する。

【 0 0 3 1 】

用語「含む（including）」および「包含する（comprising）」は、制限のない方式で使用され、従って、「～を含むが～に制限されない」ことを意味すると解釈されるべきである。

【 0 0 3 2 】

略語「LBA」は、「論理ブロックアドレス指定（logical block addressing）」または「論理ブロックアドレス」を指す。LBAは、リニアアドレス指定方式であり、ある記憶媒体、例えば、ハードディスク、内に格納されるデータブロックの位置を指定するために使用されるシステムである。LBA方式では、ブロックは、整数によって見つけられ、1つだけの番号がデータをアドレス指定するために使用される。通常、第1のブロックはブロック0である。ユーザーは、データが特定のLBA上に格納されていると、信じ得る。データが格納されるとユーザーが認識する位置は「ユーザーが認識する論理ブロックアドレス」である。これは、データが実際に格納されている場所と異なり得る。データがNCM上に実際に格納される位置は、「本当の論理ブロックアドレス」と呼ばれ得る。

【 0 0 3 3 】

略語「LUN」は、論理ユニット番号（logical unit number）を指し、論理ユニットを識別するために使用される。LUNは、通常、SANを介して共有

10

20

30

40

50

されるブロックストレージレイを管理するために使用される。

【 0 0 3 4 】

用語「マネージャ」は、コンピュータプログラム製品、例えば、持続性媒体内に格納され得、かつ1つ以上の他の動作（例えば、データの受信、送信、または処理）をとらせるコードを指す。それは、ハードウェア、ソフトウェアまたはそれらの組合せ上に格納され得る。いくつかの実施形態では、マネージャは、マネージャがその意図する機能を実施するのを可能にするように構成されている、コンピュータおよび/またはシステムの一部であり得る。

【 0 0 3 5 】

用語「メディエータ」は、ハードウェア、ソフトウェアまたはそれらの組合せ上に格納され得、かつ少なくとも1つの非キャッシュ媒体内の記憶空間の1つ以上のユニットをファイル名と相互に関連付ける、コンピュータプログラム製品を指す。従って、それは、ユーザーが認識するLBAを本当のLBAと相互に関連付け得る。メディエータは、それがポイントする非キャッシュ媒体よりも数桁小さい可能性がある。例えば、メディエータは、おおよそ、通常のシリンダーのサイズの約0.2%の大きさしかない可能性がある。いくつかの実施形態では、メディエータは、コンピューティングクラウド内に存在し、一方、他の実施形態では、メディエータは、持続性有形的記録媒体内に存在する。メディエータは、データが実際には、記録媒体の異なるトラック内に出現している間に、ホストが、記録媒体のあるトラック内にあると認識する位置内のデータの、編成、翻訳、変換および格納の制御を行うことが可能であり得るか、または、全てではないにしろ、これらの機能の1つ以上に対応するマネージャに動作可能に結合され得る。さらに、メディエータは、セクターマップ、テーブル、または物理装置もしくは構造内に配置され得るデータの他の編成を含み得、従って、メディエータの内容は、物理装置もしくは構造にあるジオメトリを持たせ得る。いくつかの実施形態では、メディエータは、L2キャッシュ上に常駐する。

【 0 0 3 6 】

用語「メタデータ」は、データのコンテナに関する管理情報を指す。メタデータの例は、読み取られているファイルの長さまたはバイトカウント；ファイルが修正された最後の時間に関する情報；ファイルタイプおよびアクセス許可を記述する情報；ならびにLUN、QoS、VMおよびWORMを含むが、それらに制限されない。他のタイプのメタデータは、オペレーティングシステム情報、自動初期化情報、グループ許可、および文書タイプ内のビットの頻度を含む。

【 0 0 3 7 】

略語「NCM」は、非キャッシュ記録媒体を指す。NCMの例は、ハードディスクおよびソリッドステートドライブを含むが、それらに制限されない。NCMは、例えば、100テラバイトのデータのバイトを保持するように構成され得る。NCMは、一意のバッファユニットを格納する。いくつかの実施形態では、NCMは、バッファユニットと格納されたハッシュ値の関連付けを含む、ダイジェストマップも格納する。これらの格納された関連付けは、サーバーのRAM内のハッシュ値テーブルにデータを投入するために使用され得る。RAMに、NCMの永続記憶装置からこの情報を取り込むことにより、高速ロードが達成され得る。追加または代替として、NCMは単位ブロック当たり1ビットのビットマップを格納し、それは、本発明の記憶装置節約を追跡するために使用され得る。実用性の問題として、ダイジェストマップのおよびビットマップの格納は、NCM上に格納されたデータのブロック当たり、5~10バイトの間、例えば、8.125バイト、のわずかなオーバーヘッドを必要とする。代替として、ダイジェストマップおよびビットマップは、本発明の方法またはシステム利用の結果として書き込まれるバッファユニットを格納するのとは異なる記憶媒体上に格納され得る。

【 0 0 3 8 】

句「動作可能に結合された」は、用語「結合された」と区別しないで使用されて、システム、装置、および/またはモジュールが、互いにまたは相互に通信するように構成され

10

20

30

40

50

て、通信時または通信した後に、それらの意図する目的を実施することが可能であることを意味する。この句および用語は、間接的、直接的、光学的、有線または無線接続を含む。従って、第1の装置が第2の装置と動作可能に結合されている場合、その接続は、直接的な電氣的接続、他の装置および接続を経由した間接的な電氣的接続、光学的接続、もしくは無線接続、またはそれらの組合せを通し得る。

【0039】

句「オペレーティングシステム」は、コンピュータハードウェア資源を管理するソフトウェアを指す。オペレーティングシステムの例は、Microsoft Windows（登録商標）、Linux（登録商標）、およびMac OS Xを含むが、それらに制限されない。

10

【0040】

用語「パーティション」は、記憶媒体、例えば、ディスクドライブをユニットに分割するフォーマットを指す。従って、パーティションは、ディスクパーティションとも呼ばれ得る。パーティションの例は、GUIDパーティションテーブルおよびAppleパーティションマップを含むが、それらに制限されない。

【0041】

句「記録媒体」は、ビットに対応する磁気信号をその中に格納できる、持続性有形的コンピュータ可読記憶媒体を指す。例として、記録媒体は、ハードドライブ、ハードディスク、フロッピーディスク、コンピュータテープ、ROM、EEPROM、不揮発性RAM、CD-ROMおよびパンチカードなどの、NCMを含むが、それらに制限されない。

20

【0042】

用語「セクター」は、ディスク、例えば、磁気ディスク上のトラックの下位区分を指す。各セクターは、一定量のデータを格納する。ディスクに対する一般的なセクターサイズは、512バイト（512B）、2048バイト（2048B）、および4096バイト（4K）である。チャンクレットが4Kのサイズであり、各セクターが512Bのサイズである場合、各チャンクレットは8セクターに相当する（ $4 * 1024 / 512 = 8$ ）。セクターは、トラックを有し、プラッター上に配置される。一般に、2つまたは4つのプラッターが1つのシリンダーを構成し、255のシリンダーがハードディスクおよび媒体装置を構成する。

【0043】

句「セクターマップ」は、ホストから要求を受信して、ファイルが格納されている記憶装置内の位置を相互に関連付けるツールを指す。セクターマップは、例えば、iSCSI（インターネット小規模コンピュータシステムインタフェース）プロトコルによって定義されるパラメータ下で動作し得る。本発明のいくつかの実施形態では、セクターマップは、メタデータのビットフィールド内に配置され得る。

30

【0044】

用語「トラック」は、全てのセクターをトラバースするディスク内の円形の単位を指す。「トラックセクター」は、任意の1つのセクター内のトラックである。「トラッククラスタ」は、2つ以上のセクターに及ぶ。

【0045】

好ましい実施形態

40

【0046】

本発明は、データを非キャッシュ記録媒体上に格納するための方法、これらの方法を実施するためのコンピュータプログラム製品、およびこれらの方法を実施するように構成されているシステムを提供する。本発明の様々な実施形態を通して、重複したデータが格納される回数を減らすことにより、データを効率的に格納および取得することができる。

【0047】

一実施形態によれば、本発明は、データを非キャッシュ記録媒体上に格納するための方法を提供する。いくつかの実施形態では、受信されるデータは、ユーザー定義の論理ブロックアドレス（LBA）およびユーザー提供のバッファを含むが、基本的にそれらから成

50

るか、またはそれらから成る。ユーザー提供のバッファのサイズは、例えば、512バイト～64Kまたはそれ以上、例えば、最大で2メガバイトであり得る。受信されるデータは、 $(LBA_x, バッファ_x)$ の形であり得、ここで $x = 1 \sim n$ および $n$  = ユーザーが送信するバッファ数である。バッファは生データであり、従って、それらは、任意の文書タイプ、例えば、J P E G、P D F、W O R D文書、M P E GおよびT X T文書、に対応し得る。

#### 【0048】

ユーザーが送信する総ストリームは、Nバイトの形式であり得る。ストリームは、有線または無線のネットワークを経由し、I/Oストリームを伝送するための既知の方法および技術を通して、受信され得る。ユーザーは、データをフォーマット： $(LBA_x, バッファ_x)$ にフォーマットし得るか、またはユーザーはデータを、そのデータをこのフォーマットにフォーマットするサーバーに送信し得る。従って、Nは、バッファユニットのサイズよりも大きい可能性がある。好ましくは、ユーザーは、データをファイル名および/またはファイル識別子とともに伝送する。

#### 【0049】

Nがバッファユニットのサイズよりも大きい場合、データのストリームの受信後に、サーバーは、Nバイトをバッファユニットにフラグメント化し得る。従って、バッファは、任意のサイズであり得るデータのストリームであるが、バッファユニットは、固定サイズであり、N C M上の記憶ユニットのサイズに対応する。例えば、ユーザーは、1024バイトの単一バッファを、L B A 1 0で始まるためのファイルに対する指定とともに送信し得る。本発明の実施態様のバッファユニットサイズが512バイトである場合、データは、L B A 2 0およびL B A 2 1に格納され得る。とりわけ、いくつかの実施形態では、ユーザーは、開始L B Aのみを伝送し得、そのシステムは、ファイルが、そのアドレスから開始して、そのファイルに対して十分な空間があるように、連続したブロックまで伸びる、N C M上に格納されることを認識するであろう。

#### 【0050】

このフラグメンテーションは、総ストリームまたはチャンクレットについて実行され得る。例えば、本方法が、4Kのサイズのバッファユニットを受信するように構成されているが、ユーザーが16Kのサイズのチャンクレットを伝送する場合、サーバーがそのチャンクレットを受信後または受信時に、それは、各チャンクレットを、各々が4Kのサイズの4つのバッファユニットに分割するためのプロトコルを開始するであろう。従って、いくつかの実施形態によれば、データを $(LBA_x, バッファ_x)$ の形式で受信するか、または受信するデータを、この形式に変換する。ユーザーがデータを、必要なバッファユニットサイズになるように構成されているか、またはそのサイズに変換されるパケットで提供するかに関わらず、ハッシュ値アルゴリズムに対する入力として機能するバッファユニットの各々は、「ユーザー提供のバッファユニット」と呼ばれる。

#### 【0051】

データが正しいフォームで受信された（または正しいフォームに変換された）後、暗号化ハッシュ関数値アルゴリズムが、各バッファユニットに適用されて、そのバッファユニットに対する生成されたハッシュ値を形成する。生成されるハッシュ値は、生成されたハッシュ値と呼ばれ得る。暗号化ハッシュ値アルゴリズムは、例えば、持続性記憶媒体内に格納されている、コンピュータプログラム製品またはコンピュータプログラム製品内のプロトコルの形式であり得る。これらのタイプのアルゴリズムの例は、MD5ハッシュ（メッセージダイジェストアルゴリズムとも呼ばれる）、MD4ハッシュおよびS H A - 1を含むが、それらに制限されない。ハッシュ関数値アルゴリズムから出力される値は、ハッシュ値、チェックサムまたは和（sum）と呼ばれ得る。いくつかの実施形態では、ハッシュ値は、64、128、もしくは256ビットまたは8バイトのサイズ、あるいは中間の任意の値である。I/Oストリーム内のデータの非常に繰り返しが多い特質のために、矛盾するハッシュ値、すなわち、同じであるが、異なるバッファユニットに対応するハッシュ値、を生成する確率は、比較的低い。本方法は、先入れ先出し（「F I F O」）プロ

10

20

30

40

50

トコルに従ってハッシュ値を取得し得、I/Oストリームが受信されている間、ハッシュ値が生成されている間、または全てのI/Oストリームが受信されて、フラグメント化され、必要ならば、ハッシュ関数値アルゴリズムを適用された後のいずれかに、関連ファイルのアクセスを開始する。

【0052】

生成されたハッシュ値がユーザー提供のバッファユニットに対して取得された後、その生成されたハッシュ値を生成した、異なるコンピュータプログラム製品または同じコンピュータプログラム製品内の異なるモジュールが、アクセスされる。このコンピュータプログラム製品は、ハッシュ値テーブルをアクセスする。ハッシュ値テーブルは、例えば、永続的メモリ内に格納されて、アクセスおよび使用のためにL2キャッシュに置かれ得る。ハッシュ値テーブル内で、複数の格納されたハッシュ値が各々、格納されたバッファユニットのセット内の異なる格納されたバッファユニットおよびNCM上のバッファユニットの本当のLBAと関連付けられる。句「格納されたハッシュ値」は、ハッシュ値アルゴリズムにより特定のユーザー提供のバッファユニットに対して生成されるハッシュ値と対比するために使用され、それは、生成されたハッシュ値と呼ばれ得る。

10

【0053】

ハッシュ値テーブルは、当初、特定のハッシュ値アルゴリズムによって関連付けられたとおりの既知のハッシュ値およびバッファユニットのセットで生成され得る。これらの既知の値は、アルゴリズムの経験的な先使用、例えば、1つのホストのファイルまたは同様の産業におけるホストのファイルに対して以前に生成されたもの、に基づき判断されている可能性がある。代替として、それは、当初は空であり得、最初のユーザー提供のバッファユニットが関連付けの空白(null)のセットと比較される。使用時には、ハッシュ値テーブルは、例えば、サーバー上の、RAM内に常駐し得る。従って、ハッシュ値テーブルまたはそれに投入するためのデータは、バッファユニットを格納するNCM上、またはシステムのブートもしくは再ブート時にRAMを再設定するためにアクセスできるフォーマット内のどこかなど、永続記憶装置内に常駐し得る。更新される場合、更新は、RAM内のテーブルに対して、および永続的メモリに対しても、行われる。

20

【0054】

いくつかの実施形態では、いかなる時でも、テーブル内で、所与の格納されたハッシュ値が、わずか1つのバッファユニットとアクティブに関連付けられる。バッファユニットがハッシュ値テーブル内にあり、格納されたハッシュ値と関連付けられている場合、そのバッファユニットは「格納されたバッファユニット」である。ハッシュ値テーブル内の特定のハッシュ値と関連付けられているバッファユニットは、時間とともに変わり得、以下で説明するように矛盾が起こった場合、最新のバッファユニット、ハッシュ値アルゴリズムによって判断されたハッシュ値関連付けが、本明細書で説明する方法の適用後に、テーブル内のアクティブな関連付けになるであろう。句「アクティブな関連付け」および「アクティブに関連付けられた」は、テーブルが、生成されたハッシュ値との比較のために抽出されるデータを、提供するか、構成するか、または示す条件を指す。アクティブな関連付けは、ハッシュ値アルゴリズムの適用時に、ハッシュ値の生成をもたらす、バッファユニットを最後に受信したNCM上の本当の論理ブロックアドレスを含み得る。一旦、ハッシュ値が格納されたハッシュ値になると、それは、テーブル内にとどまるが、時間とともに、それが関連付けられているバッファユニットは変わり得る。従って、格納されたバッファユニットは、格納されたバッファユニットのままでない可能性がある。

30

40

【0055】

当業者は、ある実施形態では、ハッシュ値テーブルを使用するのではなく、マルチマップを使用し得ることを認識するであろう。マルチマップを使用する場合、ハッシュ値は、2つ以上のバッファユニットと関連付けられ得、従って、矛盾がある場合に、以前に格納されたハッシュ値が除去される必要がない。

【0056】

ハッシュ値アルゴリズムは、複数の異なるバッファユニットに対して同じハッシュ値を

50

生成できるので、手続きは、テーブル内でのこれらの出現を扱う方法を判断する必要がある。格納されたハッシュ値と同じである、ハッシュ値が、ユーザー提供のバッファユニットに対して生成されている場合、次の3つの手続きのうちの1つが使用され得る：(1) 既存の関連付けが新しい関連付けで上書きされ得る；(2) 既存の関連付けが削除され得、新しいエントリがテーブル内に、例えば、新しい位置で作成され得る；または(3) 既存の関連付けが非アクティブなファイルに移動され得るか、または別の方法でそれが非アクティブであることを示すように修正され得るが、情報は、例えば、アクティブなファイル内に保持され、適切に設計されたコンピュータプログラムを通じて、後にアクセスできる。条件(1)または(3)は、関連付けを非アクティブにする例である。当業者であれば認識するように、NCMからのデータ取得は、ハッシュ値テーブルへのアクセスを必要としないので、任意のアーカイブされた情報は、本発明の様々な実施形態の実装のために必要でない。しかし、この情報は、同じハッシュ値が異なるユーザー提供のバッファユニットまたはフラグメント化されたユーザー提供のバッファユニットに対して生成されている程度を再検討するために使用され得る。

#### 【0057】

データの非常に繰り返しが多い特質のために、矛盾するハッシュ値が、ハッシュ値アルゴリズムの適用の結果として生成されている確率は低い。例えば、SHA-1の160ビットのハッシュは、異なるパターンに対して同じハッシュ値をランダムに生成する、 $10^{24}$ 分の1の確率を有する。本発明は、重複したバッファユニットが格納される程度を最小限にするために、ハッシュ値アルゴリズムのこの特徴を利用する。これを達成するために、本方法は：(1) 新しく生成されたハッシュ値が、それに対して格納されたバッファユニットがある、格納されたハッシュ値と同じであるか；そうであれば、(2) そのハッシュ値に対して格納されたバッファユニットがユーザー提供のバッファユニットと異なるような矛盾があるか、を問い合わせる、2つのモジュールまたは2つの別個のコンピュータプログラムアルゴリズムの割当てを提供する。

#### 【0058】

前述の問い合わせの結果は、どの新しい情報がNCM上に格納されるか、およびどのような状況下で、ならびに何がメタデータに書き込まれるか、を判断する。最も簡単な場合には、本発明のプロトコルは、ハッシュ値の重複がないことを判断する。ハッシュ値の重複がないことは、ユーザー提供のバッファユニットがハッシュ値テーブル内にないことを示す。その結果として、バッファユニットが、NCMの新しいブロック内に書き込まれて、ユーザーが後に同じバッファユニットを、異なるデータストリームもしくはデータパケットの一部として、または後に同じデータストリーム内で、投入する場合に、それがNCM上に既に格納されているデータと同じであることを方法が検出できるように、更新がハッシュ値テーブルに対して行われる。ハッシュ値テーブルは、バッファユニットが書き込まれる本当のLBAを含むようにも更新される。生成されたハッシュ値およびユーザー提供のバッファユニットをハッシュ値テーブルに書き込むと、それらは、それぞれ、格納されたハッシュ値および格納されたバッファユニットとなる。

#### 【0059】

方法が、ハッシュ値の重複があると判断する場合、第2の問い合わせが行われる。前述のように、この第2の問い合わせでは、本方法は、アルゴリズムをユーザー提供のバッファユニットに対して適用すると、たとえ2つのバッファユニットが異なっているとしても、既に格納されたバッファユニットが関連付けられている同じハッシュ値の生成をもたらすかどうかを検討する。

#### 【0060】

当業者であれば認識するように、この2段階のアプローチは、重複したデータがNCM上に格納される回数を減らすための方法に効率をもたらす。第1のステップでは、ハッシュ値が比較される。これらの値は、バッファユニットよりも小さく(例えば、バッファユニットよりも少なくとも2分の1、少なくとも10分の1、少なくとも100分の1、または少なくとも1000分の1小さい)、従って、バッファユニット自体よりも比較しや

10

20

30

40

50

すい。それらの値がハッシュ値の重複を示す場合に限り、システムは実際のバッファユニットを比較する。従って、バッファユニットを互いに対してチェックする前に、ハッシュ値テーブル内にまだないハッシュ値と関連付けられているバッファユニットを除去することにより、システムは効率的である。

#### 【0061】

問合わせの第2の段階では、2つのバッファユニットを相互に比較する。矛盾がない、すなわち、（格納されたものおよび現在のユーザー提供の）バッファユニットの本当の識別がある場合；NCMへの書込みの追加のステップが必要とされない。代わりに、バッファユニットが以前に格納されたブロックのメディエータに付けられた印があることを必要とするだけである。従って、これらのバッファユニットは再度書き込まれる必要がないので、効率が向上する。それが属するファイルを追跡するために、メディエータ内で、LBAは、ユーザーが認識するLBA、および任意選択で、ユーザーが生成したファイル名またはファイルシステムと関連付けられる。

#### 【0062】

問合わせの第2の段階の他の結果は、（格納されたバッファユニットと関連付けられた際の）格納されたハッシュ値が、生成されたハッシュ値と同じであるが、格納されたバッファユニットおよびユーザー提供のバッファユニットが異なるので矛盾がある場合に生じる。これらの場合、本発明の方法は、ユーザー提供のバッファユニットがNCMへの書込みを必要としていると考える。本方法は、ハッシュ値テーブルも変更させて、それが、共通のハッシュ値をユーザー提供のバッファユニットと関連付けるようにする。共通のハッシュ値の以前の関連付けが、非アクティブにされるか、または削除される。その結果として、重複したバッファユニットを検索する後続の問合わせでは、より最近格納されたハッシュ値、バッファユニット関連付けが検討される。これらの方法では、矛盾が生じた場合には、ハッシュ値の拡張は絶対に必要ない。

#### 【0063】

様々な実施形態では、各LBA<sub>x</sub>に対して、本方法は、メディエータにLBA<sub>y</sub>をLBA<sub>x</sub>とともに格納させ、LBA<sub>y</sub>は、ユーザー提供のバッファユニットと同一である、バッファユニットの実際の位置を指す。LBA<sub>x</sub>は、バッファユニットが格納されているとユーザーが信じる位置であり、LBA<sub>y</sub>は、バッファユニットが実際に格納されている位置であるので、LBA<sub>y</sub>は、ハッシュ値テーブル内にも格納される。しかし、LBA<sub>x</sub>は、ハッシュ値テーブルから省かれ得、いくつかの実施形態では、メディエータ上にのみ存在する。任意選択で、メディエータは、ユーザー作成のファイル名および/またはファイル識別子も格納して、そのファイル名を1つ以上のユーザーが認識する論理ブロックアドレスと関連付ける。

#### 【0064】

前述のように、ほとんどのバッファユニットはNCM上に一度だけ格納される。従って、複数のユーザーファイルに対して、ユーザーが認識する論理ブロックアドレスと本当の論理ブロックアドレスとの相関関係があり、異なるユーザーファイルに対応する複数の（メディエータ上の）相関関係内には、同じ本当の論理ブロックアドレスであるが、異なるユーザーが認識する論理ブロックアドレスが1つ以上ある。実際のユーザーファイル内で最も繰り返しが多いバッファユニットは、最多数の異なる相関関係内に出現するであろう。加えて、ユーザーはバッファを提供し、それがNCM上の連続したサイトに格納されていると信じ得るので、単一のLBA<sub>x</sub>とそのバッファとの関連付けを記録して、データをLBA<sub>x</sub>から始まる連続したLBAに格納されていると見なし得る。しかし、格納は実際にはバッファユニットレベル上であり、しばしば、所与のバッファに対して連続した位置に格納されないので、メディエータは、単一のユーザーが認識するLBA（または、暗黙的もしくは明示的に）複数の連続したLBAを、連続していない複数の本当のLBAとともに格納し得る。例えば、ユーザーは、サイズが4096Bで、ユーザーが認識するLBAが10のバッファを供給し得る。バッファユニットが512Bのサイズである場合、ユーザーは暗黙的に、そのデータはLBA 10から始まる8つの連続した記憶サイトにあ

10

20

30

40

50

ると信じ得る。しかし、実際には、それらは、L B A 4、L B A 3、L B A 2、L B A 2、L B A 3、L B A 3、L B A 9、L B A 4であり得、メディアータは、それらの位置を指すであろう。とりわけ、全部ではないにしろ、ユーザーが送信するバッファ内のデータに対応する、ほとんどの重複したバッファユニットに対して、メディアータはN C M上の同じL B Aを指すであろう。従って、メディアータ上に、いくつかの実施形態では、全てのデータの本当の位置（例えば、全ての本当のL B A）の、ユーザーから受信した際のユーザーが認識する位置または複数の位置との相関関係があり得る。

【 0 0 6 5 】

従って、これらの方法では、拡張を使用して、ハッシュ値アルゴリズムの同じハッシュ値を生成する可能性を考慮に入れるのではなく、（ i ）比較可能なハッシュ値が、任意のバッファユニットと関連付けられているとして、まだハッシュ値テーブル内にない；または（ i i ）任意の所与のハッシュ値に対して、矛盾がある：場合に限り、前述した関連付けの最新の出現を使用して、特定の提供されたバッファユニットをN C Mに書き込む。これらの後者の場合、以前に書き込まれている、同じデータのN C Mへの何回かの書き込みがあり得る。しかし、実用性の問題として、まれにしか起こらないであろう。

【 0 0 6 6 】

いくつかの実施形態では、バッファユニットのN C Mへの書き込みは連続的である、すなわち、書き込まれる各ユーザー提供のバッファユニットは、N C M上の次の隣接するブロックに書き込まれ得る。従って、N C M上でのデータの散在が最小限であるか、または全くなく、それは、読取り / 書き込み性能を向上して、記憶空間の節約を可能にする。加えて、実際のN C Mへの書き込みが少ないので、読取り / 書き込み性能が向上する。これは、その結果として、オペレーティングシステムのキャッシュがより良く機能するのを可能にする。さらに、N C M上のデータは、メディアータおよびハッシュ値テーブルがない限り、ファイルを再構成するために使用できないので、データのセキュリティを向上できる。

【 0 0 6 7 】

本発明のさらなる利点は、データがどのように読み取られるかを考慮すると、理解され得る。ユーザーは、ファイルを読み取る要求を送信し得る。要求は、ファイル識別子および1つ以上のユーザーが認識する論理ブロックアドレスに関する情報を含む。関連するメディアータにアクセスすることにより、実際の論理ブロックアドレス（複数可）を判断して、関連するデータを取得できる。とりわけ、書き込みプロトコルとは対照的に、取得および読取りステップ中に、ハッシュ値アルゴリズムまたはテーブルは必要ない。代わりに、読取り装置は、メディアータが指すN C M上のサイトからデータを取得し、また、メディアータは、1つ以上のファイルに対して、1つ以上のバッファユニットを複数回、指し得る。

【 0 0 6 8 】

本発明は、データを効率的に格納するためのシステムも提供する。これらのシステムは：（ a ）永続的メモリ；（ b ）中央処理装置（ C P U ）；（ c ）非キャッシュ記録媒体；および（ d ）メディアータ；を含み得る。構成要素の各々は、それらの指定された機能を実行するために、1つ以上の他の構成要素と動作可能に結合され得る。

【 0 0 6 9 】

永続的メモリは、ハッシュ値テーブルを含み、以下で説明する非キャッシュ記録媒体の一部であり得るか、またはそれとは異なり得る。ハッシュ値テーブルは、複数の格納されたハッシュ値の各々を異なる格納されたバッファユニットと関連付ける。ハッシュ値は、ハッシュ値アルゴリズムのバッファユニットに対する適用によって判断される。各格納されたバッファユニットは、本当の論理ブロックアドレスとも関連付けられる。当業者は、ハッシュ値テーブルが、3つのタイプのデータ：格納されたハッシュ値、格納されたバッファユニットおよび本当の論理ブロックアドレスの間の関連付けを1つのテーブル内に含み得ることを認識するであろう。あるいは、ハッシュ値テーブルは、1つのテーブル内に、最初の2つのタイプのデータだけの関連付けを含み得、同じか、または異なるメモリ装置（例えば、メディアータ）内の、別のテーブル内に、本当のL B Aと格納されたバッ



ァユニットを相互に関連付けるテーブルが格納され得る。永続的メモリは、CPU内に格納され得るか、またはCPUに動作可能に結合され得る。

【0070】

中央処理装置は、ハードウェアまたはハードウェアとソフトウェアの組合せから成る。CPUは、永続的メモリにアクセスして、ハッシュ値テーブルを検索するように構成される。CPUは、NCMおよびメディアータへの書込みを含むがそれらに制限されず、本発明の方法の1つ以上を実行するようにも構成される。さらに、CPUは、例えば、サーバーを通じて、無線または有線ネットワークを通して1つ以上のリモートユーザーと通信するように構成される。

【0071】

NCMは、ブロックレベル記憶に対して構成される。NCMは、CPUとは別個であり得るか、またはCPUの一部であり得る。

【0072】

前述したシステムのいくつかの実施形態では、メディアータ、ハッシュ値テーブル、CPUおよび非キャッシュ記録媒体の各々は、互いにリモートに格納される。それらは、同じ筐体内の別個の構造内または異なる筐体内にあり得る。

【0073】

前述のように、（サーバーを通して制御され得る）本発明のシステムは、メディアータを調べた後にハッシュ値テーブルにアクセスすることなく、ファイルを再作成してデータをユーザーに伝送することができる。従って、一実施形態では、本発明のシステムは、取得モジュールを含み、取得モジュールは、メディアータにアクセスして、ハッシュ値テーブルにアクセスすることなく、複数のバッファユニットを、メディアータによって指示される順序で再結合することにより、データファイルを再構成するように構成され、メディアータによって指示される順序は、非キャッシュ記録媒体上でのバッファユニットの順序とは異なる。

【0074】

本発明の様々な方法は、マネージャによって自動的に制御され得る。マネージャは、1つ以上のモジュールを含み、ローカルコンピュータ上、ネットワーク上、またはクラウド内または例えば、CPU内に常駐し得る。マネージャは、情報自体の受信を調整するか、または情報自体を受信して、この情報をメディアータに転送するか、または、情報の受信を直接メディアータによって制御するように構成され得る。従って、本方法は、イニシエータからの情報が、本発明の重複排除方法のため、マネージャを通り、マネージャの指示で、メディアータに、またはシステムの他の構成要素に流れるが、マネージャを通して流れないように、設計できる。

【0075】

いくつかの実施形態では、マネージャは、1つまたは複数のメディアータを制御し、それと通信し、かつ、その活動を調整し得る。各メディアータに対して、マネージャは、パラメータのセットを受信する（または、その受信を調整する）。これらのパラメータは、ファイルシステム情報、ブート可能性情報、およびパーティション分割情報の1つ、2つ、もしくは3つ全部を含むか、基本的にそれらから成るか、またはそれらから成り得る。

【0076】

メディアータは、例えば、（a）トラックの第1のセット；（b）トラックの第2のセット；（c）トラックの第3のセット；および（d）トラックの第4のセット：を含み得る。マネージャは、ファイルシステム情報、ブート可能性情報、およびパーティション分割情報を、メディアータ上のトラックの第1のセット内に格納させ、それは、予約1またはR<sub>1</sub>と呼ばれ得る。この情報は、ファイルシステム情報の識別を含み得、それは、予約ブロックがどのように使用されるかを指示する。例えば、NTFSを使用する場合、セクター1～2がMBR（マスターブートレコード）用であり得、セクター3が\$MFT用であり得る。任意選択で、これらのトラックが、トラックの第2のセットにコピーされ得、それは、予約2またはR<sub>2</sub>と呼ばれ得る。

10

20

30

40

50

## 【 0 0 7 7 】

これらの実施形態では、マネージャは、前のパラグラフで説明したパラメータに加えて、メタデータも受信し得る。メタデータは、メディアータ上のトラックの第3のセット内に格納され得る。マネージャがパラメータおよびメタデータを受信する時、またはその後に、それは、非キャッシュ媒体上に格納するための1つ以上のファイルも受信し得る。各ファイルは、ファイル名および1つ以上のユーザーが認識するLBAとともに受信される。ファイル名は、ファイルを伝送するホストによって生成され、ホストのファイルシステムによって定義され得る。例えば、SANもしくはNASもしくはそれらの組合せであるか、またはその一部であり得る、マネージャは、ファイルをファイル名とともに受信すると、本当のLBAおよびユーザーが認識するLBAをトラックの第4のセットのビットフィールド内に格納することを含め、格納のための本明細書で説明するステップを自動的に実行できる。

10

## 【 0 0 7 8 】

いくつかの実施形態では、生データを受信すると、本発明の方法は、受信の確認を自動的にホストに返させ得る。あるQoS（サービス品質）プロトコルでは、データファイルがI/Oを通して受信されて、直ちにL1キャッシュに送信される。受信されると、確認がL1キャッシュからI/Oを通して返送される。L1キャッシュから、データファイルがL2キャッシュに送信され得、L2キャッシュは確認をL1キャッシュに返送する。L2キャッシュは、データファイルを、本発明の実施形態の1つ以上を実行するシステムまたはシステムの一部にも送信し、本発明の重複排除プロトコルを経た後、メディアータに書き込み、いくつかの場合には、長期格納のために、非キャッシュ媒体（NCM）に書き込み得る。NCMは、同様に、確認をL2キャッシュに返送し得る。

20

## 【 0 0 7 9 】

いくつかの実施形態では、メディアータは、L1キャッシュ内のヒープ（動的に割り当てられるメモリ）内に常駐し得るか、またはそれに動作可能に結合され得る。代替として、メディアータは、カード内に常駐し得るか、またはL2キャッシュの一部であり得るか、もしくはL2キャッシュに動作可能に結合され得るか、またはソリッドステートドライブもしくは格納用の任意のブロック装置上にあり得る。

## 【 0 0 8 0 】

当業者であれば分かるように、メディアータを、L2に対してL1内に置くという判断は、格納されたデータの使用の頻度などの要因によって影響を受けるであろう。従って、L1キャッシュは、システムまたはエンドユーザーによって頻繁に使用されるデータを格納するために使用され、他方、L2キャッシュは、幾分頻繁にアクセスされるデータに対して使用され得る。

30

## 【 0 0 8 1 】

別のQoSプロトコルでは、I/Oを通して、データファイルがL1キャッシュによって受信される。データファイルは、L1キャッシュからL2キャッシュおよびNCMの両方に転送される。L2キャッシュおよびNCMの各々が、確認をL1キャッシュに送信する。L2キャッシュおよびNCMの一方または両方から確認を受信する前、または後のいずれかに、L1キャッシュはI/Oを通して確認を送信する。

40

## 【 0 0 8 2 】

前述のように、メディアータは、トラックの第1の予備セット（ $R_1$ ）およびトラックの第2の予備セット（ $R_2$ ）を含み得る。いくつかの実施形態では、トラックの第2の予備セット（ $R_2$ ）は、トラックの第1の予備セット（ $R_1$ ）のコピーである。追加として、いくつかの実施形態では、トラックの第2の予備セット（ $R_2$ ）を使用して、トラックの第1の予備セット（ $R_1$ ）内のエラーをチェックし得る。

## 【 0 0 8 3 】

$R_1$ は、ホストの開始のための中心点として機能するように構成され得る。従って、本発明の重複排除方法のいずれかの前に、ホストは、 $R_1$ に送信するパラメータを選択し得る。メディアータは、この情報をホストから直接に、またはマネージャを通して間接的に

50

、受信し得る。好ましくは、 $R_2$  は、決してホストに公開されない。従って、メディアータ自身またはマネージャのみが情報を  $R_2$  に格納できる。 $R_1$  および  $R_2$  の各々は、例えば、16のセクターを含み得、ホスト修飾子などの実際のデータで満たされ得る。慣例により、番号付けは、0から始まり得る。従って、 $R_1$  は、例えば、セクター（またはトラック）0～15を含み得、 $R_2$  は、セクター（またはトラック）16～31を含み得る。しかし、メディアータは、 $R_1$  および  $R_2$  の各々が、16トラックの初期サイズを超えて拡張するのを可能にするように構成され得る。

【0084】

いくつかの実施形態では、 $R_1$  は、一意の予備セクター情報およびパーティション情報を含む。パーティション情報内には、ファイルシステム情報を格納し得る。

10

【0085】

限定されない例として、また、当業者であれば分かるように、ボリュームをNTFSファイルシステムでフォーマットする際に、\$MFT（マスターファイルテーブル）、\$Bitmap、\$LogFileおよびその他などの、メタデータファイルを作成する。このメタデータは、NTFSボリューム上のファイルおよびフォルダの全てに関する情報を含む。NTFSボリューム上の第1の情報は、パーティションブートセクター（\$Bootメタデータファイル）であり得、セクター0に配置され得る。このファイルは、基本NTFSボリューム情報およびメインメタデータファイル\$MFTの位置を記述し得る。

【0086】

フォーマットプログラムは、\$Bootメタデータファイルに対して最初の16セクターを割り当てる。最初のセクターは、ブートストラップコードを有するブートセクターであり、次の15のセクターは、ブートセクターのIPL（初期プログラムローダー）である。

20

【0087】

$R_1$  および  $R_2$  のトラックに加えて、メディアータは追加のメタデータを格納し得る。このメタデータは、例えば、シンプロビジョニングストラテジの実行を可能にする情報に対応し、それは、装置が、実際に利用可能であるよりも多くの物理的資源を有しているように見えるのを可能にして、例えば、トラック32～39であり得る、 $R_2$  の後の8トラック内に含まれ得る。メタデータは、LUN、QoS、VMおよびWORMなどの特徴も提供し得る。

30

【0088】

最後に、メディアータは、ビットフィールドも含み得る。ビットフィールドは、データが記憶媒体内の物理的にどこに格納されているかを示す情報を含み、メタデータがトラック32～39に置かれている場合、ビットフィールドのセクター番号はトラック40から始まる。ホストのファイル名とデータの位置との間の相関関係が格納されているのはメディアータのビットフィールド内である。従って、それは、セクターマップを含むか、基本的にそれから成るか、またはそれから成り得る。

【0089】

事実上、好ましくは、メディアータは、バッファユニットデータが格納されているディスクまたは記録媒体上に置かれていない。加えて、好ましくは、メディアータは、対応するディスクまたは記録媒体の総メモリの約0.1～0.2%のみを必要とする。

40

【0090】

さらに詳細な説明のために、図が参照され得る。図1は、本発明の方法の表現であり、そのための命令が、持続性記録媒体内の永続記憶装置内に格納され得る。例示を目的として、図1に示す方法のステップ130から終わりまでが、単一のユーザー提供バッファユニットに対して示されているが、本方法は、複数のユーザー提供バッファユニットにフラグメント化されている所与のバッファに対して、複数回、繰り返され得る。ユーザーが認識するLBAと本当のLBAとの関連付けがメディアータに書き込まれるとき、それらの関連付けは、一緒にグループ化されて、特定のファイルに対応するとして示される。当業者であれば認識するように、ユーザー提供のバッファがユーザー提供のバッファユニット

50

と同じサイズである場合、それはフラグメント化する必要がなく、本発明の様々な実施形態は、この条件をチェックするように構成できる。本発明の様々な実施形態は、全てのバッファが等しいサイズのバッファユニットに分割され、そうでない場合は、それら全てを同じサイズにするために、バッファまたは最後のバッファユニットであろうもののビット列の端部に 0 を追加することが可能であることを確実にするようにも構成され得る。

【 0 0 9 1 】

図に示すように、情報を記憶媒体に書き込むための命令が、受信され得る。これらの命令は、ユーザーが認識する論理ブロックアドレス ( L B A ) または ( L B A <sub>x</sub> ) およびユーザー提供のバッファユニットの形であり得る ( 1 1 0 ) 。

【 0 0 9 2 】

前述のように、ユーザーが認識する L B A は、ユーザーが、彼または彼女のデータが格納されると信じる位置である。ユーザー提供のバッファユニット内のデータは、通常、それが格納される装置上のブロックのサイズになる。ユーザーが、装置上のブロックサイズよりも大きいデータを投入する場合、ユーザーが投入するデータは、各々がブロックのサイズの、複数のバッファユニットを含むように、前処理され得る。バッファユニットがブロックサイズよりも小さい場合には、ブロックのサイズになるまで、コンピュータプログラム製品が、バッファユニットの一方の端部に全て 0 を追加し得る。

【 0 0 9 3 】

受信された命令は、従って、( L B A , バッファ ) によって表され得る形式であるか、またはその形式に変換される。ユーザー提供のバッファユニットに対して、コンピュータプログラム製品は、受信された際にデータをフラグメント化し、必要であれば、ハッシュ値を計算する ( 1 2 0 ) 。複数の命令が受信される場合、各バッファユニットに対してハッシュ値が計算される。

【 0 0 9 4 】

アルゴリズムがハッシュ値を生成した後、アルゴリズムは、そのハッシュ値が、ハッシュ値テーブル内に既に存在する格納されたハッシュ値と重複しているかを問い合わせる ( 1 3 0 ) 。テーブルがその生成されたハッシュ値を含んでいない場合、アルゴリズムは、バッファユニットが N C M に対して新しいものであると結論付けて、そのバッファユニットを、以前に使用されていない N C M 内のブロックに書き込む ( 1 4 0 ) 。アルゴリズムはまた、N C M に新しく書き込まれた、このユーザー生成のバッファユニットと、その生成されたハッシュ値との関連付けを含むように、メモリ内に格納されているハッシュ値テーブルを更新する ( 1 5 0 ) 。この生成されたハッシュ値は格納されたハッシュ値になり、ユーザー提供のバッファユニットは格納されたバッファユニットになる。いくつかの実施形態では、ハッシュ値テーブルは、バッファユニットが N C M 上に格納されている本当の論理ブロックアドレスも識別するが、他方、他の実施形態では、ハッシュ値テーブルはこの情報を除外して、例えば、メタデータ上または別のデータファイル内など、他の場所に格納される。

【 0 0 9 5 】

ハッシュ値テーブルが更新された後、ユーザーが認識する L B A およびユーザー提供のバッファユニットが格納されている本当の L B A がメタデータ上に格納されて、互いに関連づけられる ( 1 6 0 ) 。

【 0 0 9 6 】

図 1 内のステップ 1 3 0 に戻り、問い合わせが、新しく計算されたハッシュ値が、ハッシュ値テーブル内の格納されたハッシュ値と重複しているという結論となる場合、方法は、矛盾があるかどうかを尋ねるプロトコルを開始する ( 1 7 0 ) 。

【 0 0 9 7 】

矛盾がない場合、プロトコルは、既に N C M 上にあり、ユーザー提供のバッファユニットと同じ、バッファユニットデータの位置と、ユーザーが認識する位置との相関関係を、メタデータに格納させる ( 1 6 0 ) 。従って、この重複したデータに対して、新しい情報は N C M に書き込まれない。

10

20

30

40

50

## 【 0 0 9 8 】

矛盾があるかどうかに関する問合わせ ( 1 7 0 ) に戻り、応答が、同じハッシュ値が異なるバッファユニットに割り当てられていることを意味する、y e s の場合、プロトコルは、ユーザー提供のバッファユニットの N C M への書込み ( 1 4 0 ) およびメモリ内のハッシュ値テーブルを、新しく書き込まれたバッファユニットとアクティブに関連付けられているハッシュ値に更新することを要求する。加えて、メディエータが、ユーザーが認識する位置 ( 複数可 ) と本当の位置 ( 複数可 ) との新しい相関関係を含むように更新される。

## 【 0 0 9 9 】

図 2 は、このステップをさらに説明する。図 1 と同様に、矛盾判断プロトコル ( 2 7 0 ) に入ると、y e s 出力がある場合、バッファユニットの一意のデータが N C M 上の新しいブロックに書き込まれる ( 2 4 0 ) 。新しいブロックの書込みに続いて、アルゴリズムはハッシュ値テーブルを更新する。

10

## 【 0 1 0 0 】

ハッシュ値テーブルを更新する場合、本方法は、以前に格納されたハッシュ値のバッファユニットとの関連付けを解除して、その格納されたハッシュ値をより最近受信されたバッファユニットと関連付けて ( 2 5 1 ) 、ハッシュ値と以前に格納されたバッファユニットとの関連付けを除去するか、または非アクティブにする ( 2 5 6 ) 。ハッシュ値テーブルが更新された後、プロトコルは、最後に受信されたバッファユニットの本当の論理ブロックアドレスをメディエータに書き込んで、それをユーザーが認識する論理ブロックアドレスと関連付ける ( 2 6 0 ) 。

20

## 【 0 1 0 1 】

図 3 は、読取り命令を表す。システムは、ブロックを読み取る、すなわち、( L B A , バッファ ) の情報を取得するための命令を受信し得る ( 3 1 0 ) 。要求は、1 つ以上のユーザーが認識する L B A とのユーザーのシステムの関連付けである、ファイル名による、ファイルに対する要求の形で到着し得る。

## 【 0 1 0 2 】

プロトコルは、メディエータ内の相関関係テーブルのアクセスおよび、バッファまたはその一部に対応する各 L B A 位置での読取りを引き起こす ( 3 2 0 ) 。相関関係テーブルは、ユーザーが認識する L B A を本当の L B A と相互に関連付けるので、L B A ( または複数の L B A ) に対する情報が取得された後、プロトコルは、指定された実際の L B A または複数の L B A で N C M を読み取り、ブロック情報を取得して、バッファパラメータを満たす ( 3 3 0 ) 。この情報を用いて、システムは、生データを正しい順序で有し、それを、ホストのオペレーティングシステムを通して要求されたファイルの再構成のために、ホストに送信することが可能である。

30

## 【 0 1 0 3 】

とりわけ、読取りステップ中に、ハッシュ値アルゴリズムまたはテーブルにアクセスする必要がない。代わりに、N C M 上に格納されたとおりの L B A が、ユーザーが必要とするバッファユニットを取得するために読み取られ得る。いずれか 1 つ以上のバッファユニットが、読取り時に、L 1 または L 2 キャッシュのいずれか内にまだある場合、それらは、N C M からではなく、適切なキャッシュから読み取られ得る。

40

## 【 0 1 0 4 】

さらなる例として、以下の表が検討され得る。

【表 1】

ユーザーが認識する L B A	ユーザー提供のバッファ ユニット	ハッシュ値	本当のL B A位置
1	A	x	100
2	B	y	110
3	C	z	111
4	D	a	112
5	E	b	113
6	F	c	114
7	G	x	115
8	H	d	116
9	I	e	117
10	J	v	110
11	K	f	118
12	L	x	115
13	M	g	119
14	N	h	120
15	O	b	113
16	P	z	111
17	Q	v	121
18	R	x	122

以下のように仮定する：A ≠ G

$$G = L$$

$$B \neq Q$$

$$B = J$$

$$E = O$$

$$C = P$$

$$A = R$$

## 【0105】

ユーザーが格納のためにデータを投入すると、ユーザーは、バッファユニットのサイズが各々、例えば、512バイトである、18のブロックに対して、L B Aが1～18であると認識する。格納のためにデータを伝送する場合、ユーザーは、各バッファユニットを、異なるブロックに格納されていると仮定して、一意として処理する。

## 【0106】

ハッシュ値アルゴリズムを適用すると、2～3の重複したハッシュ値が生成される。見て分かるとおり、ハッシュ値xがバッファユニットA、G、LおよびRに適用され；ハッシュ値yがバッファユニットB、J、およびQに適用され；ハッシュ値zがバッファユニットCおよびPに適用され；ハッシュ値bがバッファユニットEおよびOに適用される。しかし、データの本当の重複は、AとR、GとL、BとJ、EとO、およびCとPとの間に見つけられる。それ故、矛盾は、AとG；GとR；BとQ；およびQとJとの間に生じる。

## 【 0 1 0 7 】

4 列目に示すように、データを書き込む際に、本当の重複に対して、新しいブロックは書き込まれない。例えば、ユーザーが認識する L B A 2 および 1 0 が格納されている実際の位置を参照されたい。両方がブロック 1 1 0 に格納されている。3 列目および 4 列目を検討すると、最初の 7 つのユーザー提供のバッファユニットが分析された後に、最初の重複したハッシュ値が生成された ( x 値 ) ことが分かる。しかし、A G なので、G は N C M 上のブロックに書き込まれる必要がある。これが生じた後、さらなる矛盾分析のために、メモリ内に格納されているハッシュ値テーブル内で、x が、A ではなく、G と関連付けられる。

## 【 0 1 0 8 】

10

1 0 番目のバッファユニット J が分析された後に、重複したハッシュ値 y が生成されていることが分かる。しかし、バッファユニットが同じであるので、矛盾はない。従って、ハッシュ値テーブルに対する更新は必要なく、新しいブロックが N C M に書き込まれる必要がない。代わりに、メディエータが、本当の L B A 1 1 0 を指して、それをユーザーが認識する L B A と相互に関連付けるように更新される。1 2 番目のユーザーが認識する L B A が分析された後、類似した本当の重複が明らかになり、1 5 番目および 1 6 番目が分析される。

## 【 0 1 0 9 】

バッファユニット Q が分析された後、ハッシュ値 y が生成される。しかし、Q B であるので、矛盾がある。その結果として、新しいブロックが L B A 1 2 1 に書き込まれ、ハッシュ値テーブルが、y を Q と関連付けるように更新されて、もはや y を ( J と同じである ) B と関連付けない。

20

## 【 0 1 1 0 】

ユーザー提供のバッファユニット R が分析された後、ハッシュ値 x が生成される。R = A である。しかし、A は、ハッシュ値テーブル内で x との最新の関連付けではない。その結果として、プロトコルは、R を、それが R を見たのが初めてであるかのように扱い、( 1 ) 新しい相関関係をハッシュ値テーブル内に保存するか、または古いものを上書きすること ( しかし、それは、メモリ内に格納された関連付けを回復しない )、および ( 2 ) 新しいブロックのデータを N C M 内に格納する ( ここではブロック 1 2 2 ) こと : の両方を行う必要がある。このように、本例では、N C M は、いくつかの重複を含むであろう。

30

## 【 0 1 1 1 】

表 1 は、例示目的であり、いくつかの実施形態では、第 1 および第 4 の列のみがメディエータの一部であり、それらは、ハッシュ値テーブルの一部ではない。メディエータのサイズ要件は小さい。例えば、いくつかの実施形態では、メディエータは、5 1 2 バイトまたは 4 K のサイズの N C M 上に格納されている各バッファユニットに対して、8 または 1 6 バイトを必要とする。

## 【 0 1 1 2 】

本発明の様々な実施形態を通して、N C M の物理記憶空間が大いに削減できる。例えば、それらは、少なくとも 5 0 %、少なくとも 1 0 0 %、少なくとも 2 0 0 %、少なくとも 5 0 0 %、または少なくとも 1 0 0 0 % だけ削減できる。従って、いくつかの実施形態では、必要な記憶容量は、標準状態下で必要であるよりも 5 0 ~ 1 5 0 分の 1 である。例えば、5 1 2 バイト ~ 4 K のバッファユニットに相当するものの、メディエータ上での記憶のために、5 ~ 1 0 バイトしか必要ないであろう。このように、同じバッファユニット ( またはフラグメント化されたバッファユニット ) を複数回、格納する必要を削減することにより、( 本発明の N C M の一例である ) 8 G B の U S B スティックが、1 . 5 3 T B のデータに相当するものを格納するために使用できる。

40

## 【 0 1 1 3 】

本発明の方法、システムおよびコンピュータプログラム製品は、格納されているバッファユニットが、たとえフラグメント化されていても、ユーザーから受信されたものと同じであると仮定して、説明されてきた。これらの実施形態では、それらは、ユーザーが認識

50

するものとは異なる L B A に、ユーザーが認識できない順序で格納されるので、情報を取得するためにメディアータが必要であり、ユーザーはそれなしでは、データを取得できない。それ故、それは、ある程度のセキュリティを提供する。

【 0 1 1 4 】

同じハッシュ値アルゴリズムを利用する異なるユーザーが、同じハッシュ値を生成するであろう。従って、それらのハッシュ値テーブルは、N C M 上のバッファユニットの位置を除いて、同様であろう。加えて、メディアータ内の相関関係情報が異なる。追加レベルのセキュリティとして、本発明の方法に入る前に、ユーザーは、データをコード化または変換することを望み得る。これらの動作は、前処理と呼ばれ得る。

【 0 1 1 5 】

いくつかの実施形態では、前述のプロトコルまたはシステムに入る前に、ユーザーのデータがまず、ビットマーカータブルもしくは頻度変換器またはサイズが小さく、かつ／もしくは符号化されているデータを生成するための他のハッシュ値アルゴリズムの使用を通して、変換される。これらの技術を実施するための方法、システムおよびコンピュータプログラム製品は、2013年2月2日に出願された「Bit Markers and Frequency Converters」という名称の米国13/756,921号；2013年3月12日に出願された「Data Storage and Retrieval Mediation Systems and Methods for Using Same」という名称の米国13/797,003号；および2013年6月3日に出願された「Methods and Systems for Storing and Retrieving Data」という名称の米国13/908,239号に開示されている。前述した出願の開示全体が、参照により、全体として組み込まれる。これらの方法の出力が、重複削減のためのバッファユニットを生成するために使用され得る。

【 0 1 1 6 】

これらの出願は、本発明に、前処理ステップとして、すなわち、本実施形態の重複排除方策の前に、組み込まれ得る、方法を記述する。これらの場合、バッファユニットは、データが、ビットマーカータブルまたは頻度変換器の使用を通して変換される前処理方法の出力に対応し得る。

【 0 1 1 7 】

従って、一実施形態では、この前処理ステップは：( i ) 複数のデジタル2値信号を受信することであって、デジタル2値信号が複数のチャンクレットに編成されており、各チャンクレットがNビット長で、Nが1より大きい整数であり、チャンクレットが順序を有する、複数のデジタル2値信号を受信すること；( i i ) 各チャンクレットを均一なサイズのサブユニットに分割して、X個のマーカのセットから、1つのマーカを各サブユニットに割り当てて、複数のマーカのセットを形成することであって、Xがサブユニット内のビットの異なる組合せの数以下であり、同一のサブユニットが同じマーカに割り当てられて、少なくとも1つのマーカがサブユニットのサイズよりも小さい、複数のマーカのセットを形成すること；および( i i i ) マーカをバッファユニットとして使用すること；を含む。この前処理ステップは、ハッシュ値テーブルと同じか、または異なる永続的メモリ内に格納され得るビットマーカータブルを利用する。データを読み取る際に、これらの前処理ステップが逆の順番で実施され得、データがN C M から取得された後に、N C M 上のバッファユニットが、メディアータによって指示された方法で再結合される。マーカをバッファユニットとして使用する場合、必要なサイズのバッファユニットを形成するために、マーカを結合または分割し得る。

【 0 1 1 8 】

ビットマーカータブルは、全てが同じサイズであるか、または異なるサイズのマーカを含み得る。異なるサイズの場合、サイズは、以下で説明するように、ビットまたはバイトの列の予測された頻度によって判断され得る。

【 0 1 1 9 】



さらなる例として、前処理ステップがビットマーカータブルを利用する場合、生データが、その生データを表す一連のマーカに翻訳される。生データは、ホストから受信されたデータに対応し、従って、例えば、J P E G、P D F、T I F FまたはW O R D文書などの、1つ以上のファイルを個々に、または全体として形成する1つ以上のチャンクレットであり得る。

#### 【0120】

チャンクレットは順番に受信される。例えば、ファイルが、システムによって連続的に受信される10のチャンクレットを含み得る。あるいは、所与のファイルに対する複数のチャンクレットが、それらが、ホストのオペレーティングシステムによるファイルの再作成および使用を可能にするような方法で、それらの相互の再関連付けを可能にする情報を含む場合、並行して、または一緒に伝送され得る。従って、いくつかの実施形態では、本発明の方法は、マーカを、チャンクレットが受信されるのと同じ順序で生成する。それに応じて、ホストがファイルの取得を要求する場合、対応する取得手段が、符号化されたデータを同じ順序で呼び戻して、それを適切な順序でチャンクレットに復号するであろう。

10

#### 【0121】

任意選択で、符号化の前に、システムは、チャンクレットをビットのグループ(サブユニットとも呼ばれる)に分割し得、その各々はAビット長である。システムがチャンクレットをサブユニットに分割する場合、サブユニットがビットマーカータブルと比較され得る。システムがチャンクレットをサブユニットに分割しない場合、各チャンクレットが

20

#### 【0122】

ビットマーカータブルは、ビットの一意のセットを一意のマーカと相互に関連付ける。いくつかの実施形態では、ビットマーカータブルは、サブユニットが使用される場合にはサイズA、またはサブユニットが使用されない場合にはサイズNの、各一意のビット列に対するマーカを含む。それ故、この方法では、コンピュータプログラムは、チャンクレットのセットを入力として受信し得る。それは、次いで、各チャンクレットを、同じサイズで、各々Aビット長のY個のサブユニットに分割し得、 $A/8$ は整数である。各一意のAに対して、テーブル内に1つのマーカがあり得る。

#### 【0123】

このように、自動化プロトコルを通して、チャンクレットの受信後、コンピュータプログラム製品は、ビットマーカータブルをアクセスさせる。それに応じて、各チャンクレットまたはサブユニットは、入力として機能し得、各ビットマーカは、出力として機能し得、それによりマーカの出力セットを形成する。マーカの出力セットは、翻訳されたか、コード化されたか、または符号化されたデータと呼ばれ得る。各チャンクレットが細分されない実施形態では、各チャンクレットは1つのマーカを受信するであろう。チャンクレットが2つのサブユニットに分割される場合、それは、2つのマーカに翻訳されるか、または符号化されるであろう。従って、コンピュータプログラム製品は、各チャンクレットに対応する少なくとも1つのマーカを割り当てるために、マーカを入力と相互に関連付けるビットマーカータブルを使用する。コンピュータプログラム製品は、各個々のマーカに対応する異なる出力が生成されるように、各チャンクレットに対応するマーカのセットを含む異なる出力が生成されるように、または、完全なファイルに対応するマーカのセットを含む異なる出力が生成されるように、設計され得る。

30

40

#### 【0124】

前述のように、ビットマーカータブルは、X個のマーカを含む。いくつかの実施形態では、Xは、本方法がチャンクレットをサブユニットに分割しない場合には、長さNのチャンクレット内のビットの異なる組合せの数、または本方法がチャンクレットを分割する場合には、長さAのサブユニット内のビットの異なる組合せの数のいずれかに等しい。文書タイプが分かっているか、または所与の長さのサブユニットもしくはチャンクレットに対するビットの全ての組合せよりも少ないと予測される場合、X(マーカの数)は、

50

ビットの考えられる組合せの実数よりも小さい可能性がある。例えば、いくつかの実施形態では、全てのビットマーカ―が同じサイズであり、ビットマーカ―テーブル内のビットマーカ―の数が、サイズNまたはAのビット列内のビットの組合せの数に等しい。他の実施形態では、全てのビットマーカ―が同じサイズであり、ビットマーカ―テーブル内のビットマーカ―の数が、サイズNまたはAのビット列内のビットの組合せの数の、90%未満、80%未満、70%未満、もしくは60%未満である。

#### 【0125】

例として、いくつかの実施形態では、各チャンクレットは、複数の0および/または1から成るコード(すなわち、マーカ―)を割り当てられる。他の実施形態では、各チャンクレットは、各々が、複数の0および1から成るコード(すなわち、マーカ―)を割り当てられている複数のサブユニットに分割される。サブユニットは、長さAによって定義され得、 $N/A = Y$ で、Yは整数である。サブユニットがその数のビットを有していない場合、例えば、1つ以上のサブユニットが、システムが入力として受信するように構成されているビット数よりも少ないビット数しか持たない場合、システムはビット、例えば、ゼロを、全てのサブユニットが同じサイズになるまで、追加し得る。このステップは、例えば、チャンクレットがサブユニットに分割された後、全てのチャンクレットが同じサイズであるかどうかをまず確かめない場合に、実行され得る。あるいは、前述のように、それは、チャンクレットをサブユニットに分割する前に、チャンクレットレベルで実行され得る。

#### 【0126】

上の説明で示唆するように、アルゴリズムは、ビット列をコード化されたデータのセットに翻訳するように構成され得、アルゴリズムは、ビット列が、チャンクレットまたは、チャンクレットのサブユニットのいずれかに対応するように、設計され得る。好ましくは、コード化されたデータのセットは、ホストまたはクライアントから受信される際のファイルよりも小さい。しかし、コード化されたデータのセットが元のデータよりも小さいかどうかに関わらず、それは、変換してファイルのチャンクレットに戻すことが可能である。当業者であれば認識するように、格納のためにホストから受信されるデータは、生データであり、従って、任意の文書タイプに対応できる。マーカ―の出力は、前述のようなハッシュ値アルゴリズムへの入力のためのユーザー提供バッファユニットを形成するために、それらが結合されるのを可能にする順序であり得る。

#### 【0127】

符号化は、2つの独立した目的に役立ち得る。第1に、格納のためにデータを符号化することにより、セキュリティが向上する。コードを知っている(すなわち、ビットマーカ―テーブルにアクセスできる)人またはエンティティだけが、それを復号して文書を再構築することができる。第2に、コードが元の文書よりも少ないビットを使用して作成される場合、必要な記憶空間が少なくなつて、費用を節約できる。

#### 【0128】

テーブル内のビットの少なくとも複数の一意の組合せに対して、好ましくは、システムがチャンクレットをサブユニットに分割しない場合、マーカ―は、チャンクレット長Nよりも小さいか、またはシステムがチャンクレットをサブユニットに分割する場合には、サブユニット長Aよりも小さい。好ましくは、システムがチャンクレットをサブユニットに分割しない場合、どのマーカ―もチャンクレット長Nよりも大きくないか、またはシステムがチャンクレットをサブユニットに分割する場合、どのマーカ―もサブユニット長Aよりも大きくない。いくつかの実施形態では、全てのマーカ―は、Nよりも小さいか、またはAよりも小さい。加えて、いくつかの実施形態では、各マーカ―は、同じサイズであり得るか、または2つ以上のマーカ―が異なるサイズであり得る。

#### 【0129】

前述のように、ビットマーカ―テーブルは、マーカ―をビット列に、生データに対してランダムまたは非ランダムに割り当て得、ビットマーカ―は、均一または不均一なサイズであり得る。しかし、前述のようなビットマーカ―テーブルの代わりに、頻度変換器を使

10

20

30

40

50

用し得る。従って、ある文書タイプまたは文書のセット内により頻繁に出現すると見込まれる生データに対して、より小さいマーカを割り当て得る。この方策は、全ての情報のほぼ80%が、最も頻出するサブユニットのほぼ上位20%内に含まれるという事実を利用する。言い換えれば、データに対応するサブユニットは、非常に繰返しが多い。

#### 【0130】

いくつかの実施形態では、異なるサイズの複数の変換されたビット列の全てに対して、Aビット長の第1の変換されたビット列、およびBビット長の第2の変換されたビット列があり、ここで、 $A < B$ であって、第1の変換されたビット列のAビットの識別が、第2の変換されたビット列の最初のAビットの識別と同じでない。ビットマーカテーブルまたは頻度変換器のいずれかから、マーカを使用するとき、それらが異なるサイズである場合、それらは、どこで1つのマーカが終わって、次が始まるかをシステムが知ることができるフォーマットにされる必要がある。これは、例えば、最小限のマーカサイズを設定し、最小限のサイズの各ビット列がテーブルまたは変換器内で一意であるかどうかを問い合わせ、そうでない場合は、追加のビット(複数可)を読み取って、各追加のビットに対する問い合わせを繰返すことにより、ビット列を拡大し続ける、読取り分析を通して達成され得る。

#### 【0131】

情報が変換され得、出力コードは、ビットのグループを表すためにマーカが使用されるので、入力よりも小さくなるように構成できる。従って、好ましくは、テーブル内で、少なくとも1つの、複数の、少なくとも50%の、少なくとも60%の、少なくとも70%の、少なくとも80%の、少なくとも90%の、または少なくとも95%の、マーカが、サブユニットよりもサイズが小さい。しかし、変換されたデータを同じサイズにするか、またはホストから受信したか、もしくはハッシュ関数値アルゴリズムから生成したデータよりも長くするのを妨げる技術的障害はない。

#### 【0132】

別の実施形態によれば、前処理ステップは、(i) NバイトのI/Oストリームを(例えば、I/Oプロトコルを使用して)受信すること；(ii) Nバイトを、Xバイトのフラグメント化ユニットにフラグメント化すること；(iii) 暗号化ハッシュ関数(値アルゴリズム)をXバイトの各フラグメント化ユニットに適用して、Xバイトの各フラグメント化ユニットに対して生成されたハッシュ関数値を形成すること；(iv) 相関ファイルにアクセスすることであって、相関ファイルが、Yビットの格納されたハッシュ関数値を、複数の格納されたXバイトのシーケンスの各々と関連付けて、(a) Xバイトのフラグメント化ユニットに対して生成されたハッシュ関数値が相関ファイル内にある場合は、Yビットの格納されたハッシュ関数値をユーザー提供のバッファユニットとして使用し；(b) Xバイトのフラグメント化ユニットに対して生成されたハッシュ関数値が相関ファイル内にない場合は、Yビットの生成されたハッシュ関数値をXバイトのフラグメント化ユニットとともに相関ファイル内に格納し、生成されたハッシュ関数値をユーザー提供のバッファユニットとして使用する、相関ファイルにアクセスすること；を含む。

#### 【0133】

この前処理ハッシュ値アルゴリズムを使用する場合には、それらが矛盾している可能性に対処する必要がある。最新のハッシュ値関連付けが保持されて格納される、重複排除のための前述した方法に対する代替として、この任意選択の前処理ステップ中に、相関ファイル内の格納されたハッシュ関数値と同じであるが、ユーザー提供のチャンクレットが格納されたチャンクレットと異なる、ハッシュ関数値が生成される場合に、方法が、異なるZビットを格納されたハッシュ関数値および生成されたハッシュ関数値と関連付けさせる、矛盾解決モジュールを使用し得る。この技術は、2013年6月3日に出願された、米国特許出願第13/908,239号に記述されており、その開示全体が参照により組み込まれる。

#### 【0134】

このように、この前処理ステップは、第1のハッシュ値テーブルを利用し得、矛盾が存

10

20

30

40

50

在しない全てのハッシュ関数値に対して、Zビットが関連付けられ、Zビットは、例えば、8～16個のゼロの均一の長さである。限定されない例として、本方法は、チェックサムが、以前に格納されたチェックサムと矛盾しない場合、8バイトのチェックサムの端部に8個のゼロを結合させ得る。矛盾を識別した際には（例えば、異なるフラグメント化ユニットが同じチェックサムと関連付けられている）、最新のチェックサムが異なるZ値を割り当てられ得る。従って、関連ファイル内に格納された際のZ値が00000000の場合、第1の矛盾しているチェックサムに対するZ値は、00000001であり得、別の矛盾しているチェックサムがあるとすれば、00000010である。さらに矛盾しているチェックサムがある場合、矛盾しているチェックサムが識別されると、各矛盾しているチェックサムは次のZ値を割り当てられ得る。従って、関連ファイルがアクセスされた後、新しく生成されたハッシュ値が既に関連ファイル内にある場合に限り、矛盾モジュールがチェックとしてアクセスされ得る。矛盾モジュールは次いで、矛盾があるか、またはチェックサムおよび受信したファイルからのフラグメント化ユニットの両方が、既に関連ファイル内で互いに関連付けられているかを判断するであろう。これらの拡張ファイルは、格納されたハッシュ値の格納されたバッファユニットとの関連付けの置換え、または上書きに対する代替手段である。これらの拡張をもつチェックサムは、入力をユーザー提供のバッファユニットとして形成するために必要なサイズに結合され得る。

#### 【0135】

前処理ステップがハッシュ値アルゴリズムを使用する任意の場合に、適用される第1のハッシュ値アルゴリズムが、第1のハッシュ値アルゴリズムまたは第1のハッシュ値テーブルを利用する前処理ハッシュ値アルゴリズムと呼ばれ得、第2のハッシュ値アルゴリズムが、重複排除ハッシュ値アルゴリズムまたは第2のハッシュ値テーブルを利用する第2のハッシュ値アルゴリズムと呼ばれ得る。前処理ハッシュ値アルゴリズムおよび第2のハッシュ値アルゴリズムの両方が使用される場合には、前述のように、好ましくは、重複排除ハッシュ値アルゴリズムの使用時に矛盾に対処するために、対応するハッシュ値テーブルが、最新の関連付けを好んで選択することにより、矛盾している関連付けの間で解決し、他方、前処理ハッシュ値アルゴリズムの使用時に矛盾に対処するために、対応するハッシュ値テーブルが、前述のような拡張方法を使用する。

#### 【0136】

前処理技術を使用する場合、出力、例えば、ビットマーカは、バッファユニットと同じサイズであり得る。いくつかの実施形態では、ビットマーカは、バッファユニットのサイズよりも大きい可能性がある。これらの場合、システムは、それらをフラグメント化して、バッファユニットを形成し得るか、または、コード化するデータよりも大きい、いずれのビットマーカも拒絶して、代わりに、元の生データを使用してバッファユニットを形成し、それにより、ビットマーカテーブルへのアクセスを迂回する、デフォルトモジュールを含み得る。他の実施形態では、それらは、小さい可能性があり、結合してバッファユニットを形成するか、またはバッファユニットにフラグメント化されるバッファを形成する必要がある。これらのステップは、コンピュータプログラム製品内のモジュールに従い、サーバー上で、クラウド内で、またはCPUによって実施され得る。

#### 【0137】

データの前処理が書き込みプロセスの一部である場合、データの後処理が読取りプロセスの一部でなければならない。本発明の重複排除ステップの読取りとは異なり、後処理ステップは、前処理ステップと対称的であるが、逆の順番で実施される。

#### 【0138】

さらに、本発明の様々な実施形態が、データの損失を防ぐための他の方法と組み合わせて使用され得る。ある実施形態では、データのバックアップを容易にするために2つのメディアータを使用し得る。例えば、第1のメディアータでは、第1の記録媒体上に格納されるデータファイルをファイル名と相互に関連付け得る。前述のように、第1のメディアータは、ファイル名を識別するユーザーまたはエンティティがデータファイルを記録媒体から取得するのを可能にするように構成される。

## 【 0 1 3 9 】

第2のメディアータを生成する、データ保護プロトコルが実行され得る。第2のメディアータは、時間T1における第1のメディアータの正確なコピーであろう。従って、T1において、第1のメディアータおよび第2のメディアータの両方は、第1の記録媒体上の同じLBAを指す。

## 【 0 1 4 0 】

時間T1の後、例えばT2において、ホストは、例えば、所与のセクターまたはセクタークラスタ上の、所与の位置内に格納されていると信じるファイルを更新しようとし得る。ホストは、第1の記憶アドレス（複数可）に格納されたデータを変更しない。NCM上の情報を上書きさせるのではなく、第1のメディアータは、更新されたファイルであるとホストが信じるものに対応する新しい相関関係エントリを生成し得る。NCM上に書き込まれるバッファユニットの全部ではないにしろ、ほとんどが一意のエントリであるので、メディアータ上の新しい相関関係は、元の相関関係におけるものとは異なるバッファユニットに対してのみ、元の相関関係と異なるであろう。従って、T0においてファイル（A）に対し、第1のメディアータは、次の本当のLBA：200、201、202、203、204、205、206を相互に関連付け得る。T1において、メディアータのコピーが作成され得る。T2において、ユーザーは、ファイル（A）を更新しようとし得る。第1のメディアータ上で、次の本当のLBA：200、201、310、203、204、205、206を指す、新しい相関関係が保存され得る。しかし、第2のメディアータは変更されないであろう。従って、それらは、それらがどこを指すかが、異なるであろう。以前に保存された相関関係は、第1のメディアータ上で非アクティブにされるか、または削除されるか、または上書きされ得る。

## 【 0 1 4 1 】

2つのメディアータのこの使用は、ファイルが、T1およびT2の両方において存在していたとおりに格納されていることを示すように、ホストに、そのファイルシステムを更新させる必要なく、データがT1において存在するとおりにそのデータのスナップショットを提供するのを可能にする。従って、スナップショットは、時間T1において格納されている全てのデータファイルをロックして、いずれも、それらの物理ファイルを通して削除または書込みが行われるのを防ぐ。しかし、ホストがそれらのファイルを修正したい場合、実際にはファイルの新しい部分のみが格納されて、新しいメディアータエントリが作成されるときに、ホストは、それを行っていると思い込んで動作できる。

## 【 0 1 4 2 】

上で示唆するように、この方法は、第1のメディアータ、第2のメディアータおよび非キャッシュ媒体を含むシステムによって実装され得る。第1のメディアータ、第2のメディアータおよび記録媒体の各々は、持続性媒体を含むか、基本的にそれから成るか、またはそれから成り得る、別個の装置上に格納され得るか、または別個の装置から形成され得る。加えて、システム内で、メディアータおよび記録媒体は、互いに、ならびに任意選択で、命令を格納する1つ以上のコンピュータまたはCPUに、動作可能に結合されて、それらに、それらの意図する機能を実行させ、ネットワークを経由して、1つ以上のホストへ1つ以上のポータルを通して通信させる。さらになお、この実施形態は2つのメディアータの使用と関連して説明されているが、2つの別個のメディアータではなく、同じメディアータの2つのセクションを使用するシステムを実装できる。

## 【 0 1 4 3 】

データをバックアップするための前述のシステムが、2つのメディアータのコンテキストで説明される。しかし、格納されたファイルの履歴またはファイルのバージョンを捕捉するために3つ以上のメディアータが使用できる。例えば、少なくとも3つ、少なくとも4つ、少なくとも5つ、または少なくとも10のメディアータなど、が使用され得る。追加として、ホストは、メディアータに、一定の間隔で（例えば、毎週、毎月、3か月ごともしくは毎年）または不規則な間隔で（例えば、要求に応じて）、スナップショットを取らせ得る。

## 【 0 1 4 4 】

データをバックアップするための別の方法によれば、非キャッシュ媒体のクローンが作成され得る。この方法では、第1のメディアータ内で、複数のファイル名を、非キャッシュ記憶媒体上に格納されている複数のデータ位置と相互に関連付ける。第1のメディアータは、特定のファイル名を識別するユーザーが、特定のファイル名に対応する第1の非キャッシュ記憶媒体からデータファイルを取得するのを可能にするように構成されている。特定ファイルの一部または全体が、第1のセクターまたはセクタークラスタ内に格納され得る。

## 【 0 1 4 5 】

複数のデータファイル（または第1の非キャッシュ記憶媒体の全てのデータファイル）のコピーを第2の非キャッシュ記憶媒体および第2のメディアータに対して作成し得る。第2のメディアータは、時間T1における第1のメディアータのコピーであり、第2の非キャッシュ記憶媒体に動作可能に結合されている。T1の後である、時間T2において、ユーザーは、システムに、第1の非キャッシュ記憶媒体上の前記第1のセクターまたはセクタークラスタ内に格納されているデータファイルに対する修正を保存するように指示し得る。新しいバッファユニット（またはハッシュ値とアクティブに関連付けられていないバッファユニット）のみが第1の非キャッシュ記憶媒体に追加され得、第1の非キャッシュ記憶媒体上のデータの上書きはないであろう。代わりに、新しい相関関係が第1のメディアータ上に書き込まれ得る。第2のメディアータまたは第2の非キャッシュ記憶媒体に対する変更は行われない。ユーザーがT2の後にファイルを要求すると、彼または彼女は、第1のメディアータを経由して、直前に格納されたファイルのバージョンを取得するであろう。しかし、システム管理者は、第2の非キャッシュ媒体上に格納されて、第2のメディアータを経由することにより取得し得た、前のバージョンにアクセスしていたであろう。

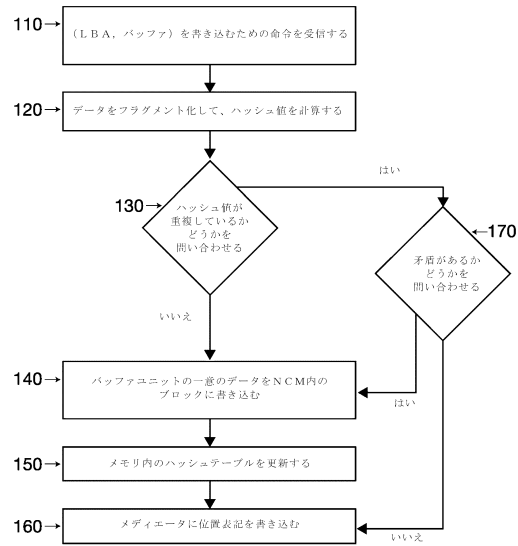
## 【 0 1 4 6 】

この方法は、第1のメディアータ、第2のメディアータ、第1の非キャッシュ記憶媒体および第2の非キャッシュ記憶媒体を含む、システムによって実装され得る。第1のメディアータ、第2のメディアータならびにデータファイルを格納するための第1および第2の記録媒体の各々は、持続性媒体を含むか、基本的にそれから成るか、またはそれから成り得る、別個の装置上に格納され得る。いくつかの実施形態では、第1の非キャッシュ媒体内に格納されている、最新のファイルは、従来のファイルが第2の非キャッシュ媒体内に有するのと同じLUNを有する。

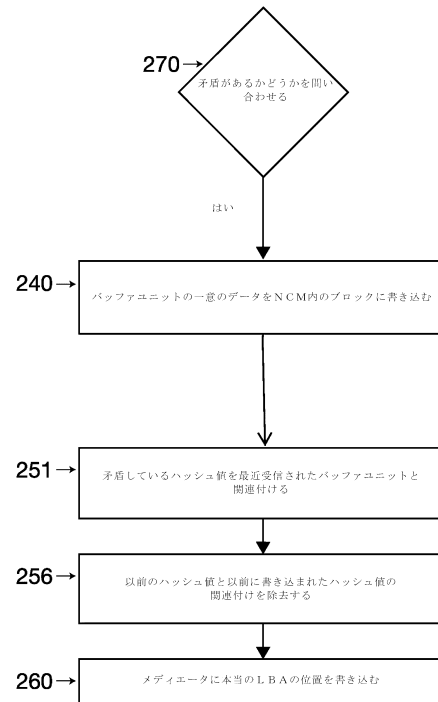
## 【 0 1 4 7 】

本明細書で説明する様々な実施形態の特徴のいずれも、特別の定めのない限り、開示する任意の他の実施形態に関連して説明する特徴と併用できる。従って、様々な、または特定の実施形態に関連して説明する特徴は、かかる排他性が明記されていないか、またはコンテキストから暗黙的でない限り、本明細書で説明する他の実施形態に関連して適切でないと解釈されるべきでない。

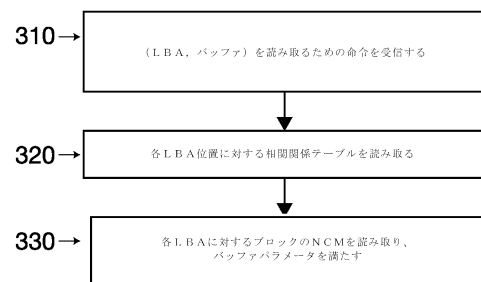
【図 1】



【図 2】



【図 3】



---

フロントページの続き

(31)優先権主張番号 13/756,921

(32)優先日 平成25年2月1日(2013.2.1)

(33)優先権主張国 米国(US)

(72)発明者 イグノミレロ, ブライアン

アメリカ合衆国, ニュージャージー州 07722, コルツ ネック, 40 マナー ロード

(72)発明者 リャン, スイホン

アメリカ合衆国, ニュージャージー州 08836, マーティンズビル, 64 ヒルクレスト ロード

審査官 田中 啓介

(56)参考文献 特開2010-072740(JP, A)

特開2011-203842(JP, A)

特開2008-165293(JP, A)

特開2007-272826(JP, A)

特開2009-251725(JP, A)

小池 到 Itaru KOIKE 他, Dedup技術の有効性の分析 Analyse validity of Dedup technology, 情報処理学会研究報告 2011(平成23)年度 6 [DVD-ROM], 日本, 一般社団法人情報処理学会, 2012年 5月 1日, Vol.2012-EVA-37, No.1, 1-6ページ

(58)調査した分野(Int.Cl., DB名)

G06F3/06-3/08、12/00

G06F13/10-13/14、17/30