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

特許第4464279号
(P4464279)

(45)発行日　平成22年5月19日(2010.5.19)　(24)登録日　平成22年2月26日(2010.2.26)

(51)Int.Cl.		F I		
G06F 12/00	(2006.01)	G06F	12/00	545Z
G06F 3/06	(2006.01)	G06F	12/00	514Z
		G06F	12/00	520Z
		G06F	12/00	531Z
		G06F	3/06	301J

請求項の数 10　(全 61 頁)

(21)出願番号　特願2004-553816 (P2004-553816)
(86)(22)出願日　平成15年11月14日 (2003.11.14)
(65)公表番号　特表2006-506741 (P2006-506741A)
(43)公表日　平成18年2月23日 (2006.2.23)
(86)国際出願番号　PCT/US2003/036699
(87)国際公開番号　WO2004/046971
(87)国際公開日　平成16年6月3日 (2004.6.3)
審査請求日　平成18年7月28日 (2006.7.28)
(31)優先権主張番号　60/426,464
(32)優先日　平成14年11月14日 (2002.11.14)
(33)優先権主張国　米国(US)

(73)特許権者　504041295
アイシロン・システムズ・インコーポレーテッド
アメリカ合衆国・ワシントン・９８１１９・シアトル・ウエスト・マーサー・ストリート・２２０・＃５０１
(74)代理人　100064908
弁理士　志賀　正武
(74)代理人　100089037
弁理士　渡邊　隆
(74)代理人　100108453
弁理士　村山　靖彦
(74)代理人　100110364
弁理士　実広　信哉

最終頁に続く

(54)【発明の名称】分散ファイルシステムにおけるファイルの再ストライピングのためのシステム及び方法

(57)【特許請求の範囲】
【請求項１】
相互に通信するように構成され、それぞれが、少なくとも一つの記憶装置と、少なくとも一つのプロセッサと、前記プロセッサによって実行される少なくとも一つの実行可能なソフトウェアモジュールとを有する複数の記憶ユニットを備え、
少なくともファイルの第１の部分が前記複数の記憶ユニットのうちの第１の記憶ユニットに格納されると共に少なくとも前記ファイルの第２の部分が前記複数の記憶ユニットのうちの第２の記憶ユニットに格納されるようにして、前記ファイルが当該分散ファイル記憶システムに格納され、
前記ファイルに関連するエラー訂正データが、前記ファイルを再構成するために利用可能なファイルデータと共に使用され得るようにして、前記複数の記憶ユニットのうちの１又は２以上の記憶ユニットに格納され、
当該分散ファイル記憶システムに格納された前記ファイルの各部分のアドレスロケーションと前記エラー訂正データのアドレスロケーションとを含むメタデータが、前記複数の記憶ユニットのうちの第３の記憶ユニットに格納され、
前記各記憶ユニットの少なくとも一つの実行可能なソフトウェアモジュールは、
前記複数の記憶ユニットのうちの前記第３の記憶ユニット上の前記メタデータをアクセスし、
前記複数の記憶ユニットのうちの２以上の記憶ユニットの間で前記エラー訂正データの少なくとも一部または前記ファイルの少なくとも一部を再ストライピングし、

前記複数の記憶ユニットのうちの前記第３の記憶ユニット上の前記メタデータを更新して、再ストライピングされた前記ファイルの各部についての新たなアドレスロケーションと、再ストライピングされた前記エラー訂正データのそれぞれについての新たなアドレスロケーションとを前記メタデータに含めるように構成され、

前記新たなアドレスロケーションは、別の記憶ユニット上のデータロケーションを含む分散ファイル記憶システム。

【請求項２】

前記再ストライピングは、前記エラー訂正データの少なくとも一部または前記ファイルの少なくとも一部の再ストライピング中のデータの移動を最小化することを含む１又は２以上のプリフェレンスまたは制約に従う請求項１記載の分散ファイル記憶システム。

【請求項３】

前記少なくとも一つの実行可能なソフトウェアモジュールは、第１のプロテクションスキームから別の第２のプロテクションスキームへ前記ファイルのプロテクションスキームを変更するために、指示に応答して前記エラー訂正データの少なくとも一部または前記ファイルの少なくとも一部を再ストライピングするように更に構成された請求項１記載の分散ファイル記憶システム。

【請求項４】

前記少なくとも一つの実行可能なソフトウェアモジュールは、当該分散ファイル記憶システムへの記憶ユニットの追加に対して、前記エラー訂正データの少なくとも一部または前記ファイルの少なくとも一部を再ストライピングするように更に構成された請求項１記載の分散ファイル記憶システム。

【請求項５】

前記少なくとも一つの実行可能なソフトウェアモジュールは、当該分散ファイル記憶システムにおける記憶ユニットの故障に対して、前記エラー訂正データの少なくとも一部または前記ファイルの少なくとも一部を再ストライピングするように更に構成された請求項１記載の分散ファイル記憶システム。

【請求項６】

前記少なくとも一つの実行可能なソフトウェアモジュールは、当該分散ファイル記憶システムからの記憶ユニットの除去に対して、前記エラー訂正データの少なくとも一部または前記ファイルの少なくとも一部を再ストライピングするように更に構成された請求項１記載の分散ファイル記憶システム。

【請求項７】

前記少なくとも一つの実行可能なソフトウェアモジュールは、前記エラー訂正データの少なくとも一部または前記ファイルの少なくとも一部が無事に再ストライピングされたかどうかを判断するように更に構成された請求項１記載の分散ファイル記憶システム。

【請求項８】

アドレスロケーションは、前記ファイルの一部が格納された装置のロケーションと、前記ファイルの一部についてのメモリブロックアドレスとを含む請求項１記載の分散ファイル記憶システム。

【請求項９】

前記エラー訂正データは、パリティデータブロック及びミラーされたデータブロックのうちの少なくとも一つを含む請求項１記載の分散ファイル記憶システム。

【請求項１０】

前記複数の記憶ユニットのうちの各記憶ユニットの前記少なくとも一つの実行可能なソフトウェアモジュールは、前記複数の記憶ユニットのうちの前記第１の記憶ユニットに格納された前記ファイルの少なくとも第１の部分と、前記複数の記憶ユニットのうちの前記第２の記憶ユニットに格納された前記ファイルの少なくとも第２の部分とに対するリード要求に応答するように更に構成された請求項１記載の分散ファイル記憶システム。

【発明の詳細な説明】

【技術分野】

【０００１】
本発明のシステム及び方法は、概して、分散ファイルストレージの分野に関し、特にインテリジェント分散ファイル管理システムに格納されたファイルの再ストライピングに関する。
【０００２】
本出願は、２００２年１１月１４日に出願された米国仮出願６０／４２６４６４の利益を主張するものであり、その出願は、参照することによりそっくりそのまま本明細書に組み込まれる。
【背景技術】
【０００３】
インターネットの爆発的な成長は、情報が常時交換され且つアクセスされる新規な領域において先導役を務めてきた。この成長に対応して、共有(share)されているデータのサイズが増加している。ユーザは、標準ＨＴＭＬドキュメント以上のものを要求しており、オーディオデータ、ビデオデータ、イメージデータおよびプログラミングデータのような種々のデータにアクセスすることを望んでいる。従って、大きなデータのセットに対応できると共に、そのデータに対する高速かつ信頼性のあるアクセスを同時に提供するデータストレージに対する要請がある。
【０００４】
一つの回答は、大量のデータを格納できるが高スループットレートの提供が困難な単一の記憶装置(storage device)を利用することであった。データ容量が増加するにつれ、データをアクセスするのに要する時間が同様に増加する。処理速度とパワーについては改善されたが、ディスクＩ／Ｏ（入力／出力）オペレーション性能は、特に大きなデータファイルについては、Ｉ／Ｏオペレーションを非効率にするのと同じ進度では改善されていない。
【０００５】
他の回答は、ストレージエリアネットワークソリューション(SAN; Storage Area Network solutions)のようなアーキテクチャを用いて複数のサーバが共有ディスクにアクセスすることを可能にしたが、このようなシステムは高価であり、しかもデータの完全性(integrity)を制御し且つセットアップするための複雑な技術を必要とする。更に、大量のデータ要求を取り扱うためには高速アダプタを必要とする。
【０００６】
従来のアプローチに関する一つの問題は、それらが拡張性(scalability)において制約されていることである。従って、データ量が増加するにつれてシステムが大きくなる必要があるが、拡大にはコストを要し、極めて秩序を乱す(disruptive)。
【０００７】
従来のアプローチに関する他の共通問題は、それらが、柔軟性(flexibility)において制約されていることである。システムは、しばしば、所定のエラー訂正制御を使用するように構成される。例えば、ＲＡＩＤシステムは、データが格納されるべき場所または使用されるべき冗長パラメータのタイプを決定することにおいて管理者に柔軟性をほとんど与えないか全く与えない物理ディスクレベルで、データファイルのミラーリング(mirroring)及び冗長性(redundancy)を提供するために使用される。
【発明の開示】
【課題を解決するための手段】
【０００８】
インテリジェント分散システムは、単一ファイルシステムとしてアクセスされるスマート記憶ユニット(smart storage unit)のセット間でのファイルデータの格納を有利に可能にする。このインテリジェント分散ファイルシステムは、メタデータ構造を利用して各ファイルについての詳細な情報を管理(manage)すると共に追跡(track)し、例えば、ファイルのデータブロックのブロックロケーション及び装置を備え、単一ファイルシステム内の冗長性(redundancy)及び／又は複製(replication)の異なるレベルを可能とし、冗長パラ

メータの変更を容易化し、メタデータのための高レベルのプロテクションを提供し、実時間でデータを移動すると共に複製する。等々。

【０００９】

また、インテリジェント分散ファイルシステムは、スマート記憶ユニットのセット間に分散されたファイルを再ストライピングするためのシステム及び方法を有利に備え、ここで、特定のストライプについてのデータブロックは一般に記憶ユニット上の何れかの特定のロケーションに位置される必要がなく、データは最小データ移動でスマート記憶ユニット間で再ストライピングされ、データは一般にシステム故障が再ストライピング処理中に発生しても回復可能であり且つ保護される。

【００１０】

本発明の一態様は、複数のインテリジェント記憶装置と通信する分散ファイルシステムに関し、この分散ファイルシステムは、メッセージングシステム(messaging system)を備え、且つメタデータが、メタデータのデータブロックのロケーション、コンテンツデータブロック、およびパリティデータブロックを含むように、分散ファイルシステム上に格納されたディレクトリおよびファイルについてのメタデータを格納し且つ管理するように構成され、且つ、この分散ファイルシステムは、更にデータファイルを再ストライピングするように構成される。

【００１１】

本発明の他の態様は、インテリジェント分散ファイルシステムにおける複数のスマート記憶装置に格納されたデータの再ストライピングの方法に関する。この方法は、既存のファイルからのデータのクラスタ(cluster)が格納される少なくとも一つのスマート記憶装置を識別(identify)するステップと、前記クラスタを前記識別されたスマート記憶装置上に格納するステップと、前記クラスタが無事に格納されたことを判断するステップと、前記データのクラスタに関連するメタデータを更新して新たな割り当て(assignment)を反映させるステップとを備える。

【００１２】

本発明の追加の態様は、データを再ストライピングするためのシステムに関する。このシステムは、複数のスマート装置(smart devices)であって該スマート装置間に分散されたストライプにデータブロックを格納するように構成されたスマート装置と、データブロックをスマート装置に割り当てるように構成された割当モジュール(assignment module)と、前記データブロックが割り当てられた後に前記データブロックを格納するためのスマート装置に命令を送信するように構成された要求モジュール(request module)と、前記データブロックが格納された後に前記データブロックに関連するメタデータを更新するように構成された更新モジュール(update module)とを備える。

【００１３】

本発明の他の態様は、データブロックからなるデータを再ストライピングする方法に関する。この方法は、データブロックを記憶ユニットに割り当てるステップと、前記データブロックが既に前記記憶ユニットに格納されているかどうかを判断するステップと、それが前記記憶ユニットにまだ存在していなければ、前記データブロックを前記記憶ユニットに格納するステップと、前記データブロックが格納されていれば、前記データブロックに関連するメタデータを更新するステップとを備える。

【００１４】

本発明の追加の態様は、記憶ユニットにデータブロックを割り当てる方法に関する。この方法は、利用可能な記憶ユニットを識別するステップと、各データブロックのための利用可能な記憶ユニットを選択するステップと、選択された各記憶ユニットについて関連アドレスロケーション(related address locations)を判断するステップとを備える。

【００１５】

本発明の他の態様は、プロテクショングループ(protection group)におけるデータのブロックを格納するための記憶ユニットのセットから記憶ユニットを選択する方法に関する。この方法は、前記記憶ユニットのセットからの何れの記憶ユニットが、プロテクション

グループに関して最適な記憶ユニットであるかを識別するステップと、前記最適な記憶ユニットを選択するステップとを備える。

【００１６】

本発明の追加の態様は、データのブロックのための記憶ユニットを選択する方法に関する。この方法は、前記ブロックデータが現在格納されている記憶ユニットを識別するステップと、第１記憶ユニット上の最適なロケーションを識別するステップと、最適な記憶ユニットのセットを判断するステップと、前記第１記憶ユニットが、前記データのブロックが現在格納されている前記記憶ユニットのうちの一つであれば、前記第１記憶ユニットを選択するステップと、前記第１の記憶ユニットが、前記データのブロックが現在格納されている前記記憶ユニットのうちの一つでなければ、前記最適な記憶ユニットのセットのうちの一つを選択するステップとを備える。

【００１７】

本発明の他の態様は、命令を含む記憶装置を備えたシステムに関し、この命令は、実行された場合に、既存ファイルからのデータのクラスタが格納される記憶装置を識別するステップと、前記クラスタを前記識別された記憶装置に格納するステップと、前記クラスタが無事に格納されたことを判断するステップと、前記データのクラスタに関連するメタデータを更新するステップの方法を前記システムに実行させる。

【００１８】

要約すると、本発明の或る態様、利点、及び新規な特徴が本明細書で述べられている。必ずしも、全てのこのような利点は本発明の何れかの特定の実施形態に従って達成されるものではないことが理解されるべきである。従って、例えば、当業者であれば、本明細書において示唆または教示されるような他の利点を必ずしも達成することなく、本明細書で教示されるような一つの利点または利点のグループを達成する方法で、本発明が実行または具現されることが分かる。

【００１９】

図１は、本発明の一実施形態のハイレベルブロック図である。

【００２０】

図２は、図１に図解される構成要素間でのデータの流れの一例を示す。

【００２１】

図３は、スマート記憶ユニットの一例のハイレベルブロック図である。

【００２２】

図４は、ファイルディレクトリの一例を図解する。

【００２３】

図５は、メタデータ構造の維持を図解する。

【００２４】

図６Ａは、データロケーションテーブル構造の一実施形態を図解する。

【００２５】

図６Ｂは、データロケーションテーブル構造の追加の実施形態を図解する。

【００２６】

図６Ｃは、データロケーションテーブル構造の追加の実施形態を図解する。

【００２７】

図６Ｄは、データロケーションテーブル構造の追加の実施形態を図解する。

【００２８】

図７Ａは、ディレクトリのためのメタデータ構造の一実施形態を図解する。

【００２９】

図７Ｂは、ファイルのためのメタデータ構造の一実施形態を図解する。

【００３０】

図８Ａは、データロケーションテーブルの一実施形態を図解する。

【００３１】

図８Ｂは、データロケーションテーブルの追加の実施形態を図解する。

【００３２】
図８Ｃは、データロケーションテーブルの追加の実施形態を図解する。
【００３３】
図９は、対応するサンプルデータを有するファイルのメタデータの一例を図解する。
【００３４】
図１０は、ファイルを読み出すためのフローチャートの一実施形態を図解する。
【００３５】
図１１は、名前解決(name resolution)を実施するためのフローチャートの一実施形態を図解する。
【００３６】
図１２は、ファイルを読み出すためのフローチャートの一実施形態を図解する。
【００３７】
図１３は、パリティ情報を生成するためのフローチャートの一実施形態を図解する。
【００３８】
図１４は、エラー訂正を実施するためのフローチャートの一実施形態を図解する。
【００３９】
図１５は、インテリジェント分散ファイルシステムにおいてデータを再ストライピングするためのフローチャートの一実施形態を図解する。
【００４０】
図１６は、スマート記憶ユニットにデータを割り当てるためのフローチャートの一実施形態を図解する。
【００４１】
図１７は、利用可能な記憶ユニットのセットの中から選択するためのフローチャートの一実施形態を図解する。
【発明を実施するための最良の形態】
【００４２】
本発明の種々の実施形態および適用例を表すシステムおよび方法が図面を参照して説明されるであろう。また他の実施形態を表すシステム及び方法に対する変形も述べられるであろう。
【００４３】
説明のため、いくつかの実施形態が、コンテンツ配信およびウェブホスティングとの関連において述べられるであろう。本発明者は、本システムおよび方法が使用される環境の形式に本発明が制限されず、本システムおよび方法が、例えばインターネット、ワールドワイドウェブ、病院の私設ネットワーク、政府機関のブロードキャストネットワーク、法人事業の内部ネットワーク、イントラネット、ローカルエリアネットワーク、ワイドエリアネットワーク、等々のような他の環境において使用してもよいことを意図している。しかしながら、図面および説明は、環境がインターネットコンテンツ配信およびウェブホスティングのものであるところの本発明の実施形態に関する。他の実施形態において、本システムおよび方法は、単一モジュールとして実施されてもよく、及び／又は、他のモジュールやその同等物と併せて実施されてもよい。さらに、本明細書で述べられる特定の実施は説明のためのものであって本発明を限定するものではない。本発明の範囲は添付の特許請求の範囲によって定められる。
【００４４】
これら及び他の特徴は上述の図面を参照して説明される。図面及び関連する説明は、本発明の実施形態を説明するために提供されるものであって、本発明の範囲を限定するものではない。図面の全体を通して、参照番号は、参照される要素間の対応を示すために再使用される。加えて、各参照番号の最初の桁は、概して要素が最初に現れる図面を示す。
【００４５】
Ｉ．概要
本発明のシステム及び方法は、インテリジェント分散ファイルシステムを提供し、この

インテリジェント分散ファイルシステムは、単一のファイルシステムとしてアクセスされるスマート記憶ユニットのセット間でのデータの格納を可能にする。本インテリジェント分散ファイルシステムは、各ファイルについての詳細なメタデータを追跡し且つ管理する。メタデータは、例えば、装置およびブロックロケーション情報の双方を含むファイルのデータブロックのロケーション、メタデータ及び／又はデータブロック（もしあれば）の冗長なコピーのロケーション、エラー訂正情報、アクセス情報、ファイルのネーム、ファイルのサイズ、ファイルのタイプ、ファイルのデータ及びプロテクション情報が格納されるスマート記憶ユニット、等々のようなファイルに関し及び／又は記述する任意のデータを含んでもよい。加えて、インテリジェント分散ファイルシステムは、本ファイルシステムによって管理されるところの異なるファイル及び／又はデータブロックについて、異なるレベルの複製及び／又は冗長を可能にし、システムがアクティブ中に冗長パラメータの変更を容易化し、システムがアクティブ中に欠損データの回復を容易化し、そしてメタデータ及びデータの実時間の複製及び移動を可能にする。更に、各スマート記憶ユニットは、スマート記憶ユニットからのファイルのデータを配置(locating)及び収集(collecting)することによりファイル要求に応答してもよい。

【００４６】

インテリジェント分散ファイルシステムは、有利には、特にライト(WRITE)要求の数に比例する多くのリード(READ)要求が存在する状況におけるデータへのアクセスを提供する。これは、スマート記憶ユニットのグループをロッキング(rocking)すること、またはライト要求に対する整合性を確保するためにスマート記憶ユニット上でジャーナル処理(journaling)を行うことの追加的複雑性のためである。さらに、インテリジェント分散ファイルシステムは、有利には、大きなデータのブロックに対する要求が一般的であるブロックトランザクションを取り扱う。

【００４７】

いくつかの実施形態のひとつの利点は、ファイル及びディレクトリのためのメタデータがインテリジェント分散ファイルシステムによって管理され且つアクセスされることである。このメタデータは、ディレクトリまたはファイルのためのメタデータが配置された場所、コンテンツデータが格納された場所、同様にシステムに関連する他のエラー訂正情報またはパリティが格納された場所を表してもよい。データロケーション情報は、例えば装置およびブロックロケーション情報を用いて格納されてもよい。従って、このインテリジェント分散ファイルシステムは、その両方がスマート記憶ユニット間に分散されて格納されるメタデータを用いて、要求されたコンテンツデータを配置し且つ読み出してもよい。加えて、インテリジェント分散ファイルシステムはメタデータへのアクセスを有するので、インテリジェント分散ファイルシステムは、データが格納されるべき場所を選択するために使用されてもよく、そしてスマート記憶ユニットのセットを混乱させることなくデータを移動(move)し、複製(replicate)し、回復(recover)し、及び／又は変更(change)するために使用されてもよい。

【００４８】

いくつかの実施形態の他の利点は、各ファイルについてのデータは、多くのスマート記憶ユニットにわたって格納されてもよく、且つタイミングを得た方法でアクセスされてもよい。各ファイルについてのデータブロックは、データアクセスタイムが減少されるようなスマート記憶ユニットのサブセット間で分散されてもよい。さらに、異なるファイルは、スマート記憶ユニットの異なるセットにわたるのと同様に、異なる数のスマート記憶ユニットにわたって格納されてもよい。このアーキテクチャは、インテリジェント分散ファイルシステムが、利用可能な記憶容量、ＣＰＵ稼働率、および各スマート記憶ユニットのネットワーク稼働率と同様に、ファイルのサイズ、重要性、予期されたアクセスレートに基づいてデータブロックを知的(intelligently)に格納することを可能にする。

【００４９】

いくつかの実施形態の追加的利点は、本システムおよび方法が、スマート記憶ユニット間に格納される異なるブロック又はファイルが異なるタイプのプロテクションを備えても

よいようなブロック又はファイルベースで、エラー訂正、冗長、およびミラーリングのような種々のプロテクションスキームを提供するために使用されてもよいということである。例えば、いくつかのディレクトリまたはファイルはミラーされてもよく、他は、種々のエラー又は欠損訂正スキームを用いてエラー及び／又は欠損訂正データで保護されてもよく、重要度の低い他のものは何らプロテクションスキームを使用しなくてもよい。

【００５０】

いくつかの実施形態の更なる利点は、本システムおよび方法が、進行中のデータ要求を混乱または中断することなく、スマート記憶ユニットの実時間の付加(addition)、削除(delete)、及び／又は修正(modification)を可能にするということである。従って、より多くの記憶が要求されるほど、追加のスマート記憶ユニットがスマート記憶ユニットのセットに加えられ、そしてファイル要求を中断することなく、または既存のスマート記憶ユニットをオフラインにする必要なく、実時間でインテリジェント分散ファイルシステムに組み込まれてもよい。既存のファイル又は新たなファイルのデータブロックが、今や新たなスマート記憶ユニットを備えるスマート記憶ユニットにわたってインテリジェント分散ファイルシステムにより分散されているので、既存のスマート記憶ユニットはファイルの要求を処理してもよい。

【００５１】

いくつかの実施形態の他の利点は、本システムおよび方法は、１または２以上のスマート記憶ユニット上のそれらのブロックを複製し従って任意の個々のデータブロックのためのアクセスの複数のポイントを生成することにより、データブロックの記憶に対する実時間の修正を実施してもよい。この複製は、ＣＰＵ稼働率、および、頻繁なアクセスパターンが観られるファイル又はファイルのグループのための個々のスマート記憶ユニットに対するネットワーク資源必要条件を低減させるのに役立つ。これらのアクセスパターンは、スマート記憶ユニットによりモニタされ、そしてインテリジェント分散ファイルシステムは、インテリジェント分散ファイルシステムの稼動中にこのようなデータ複製を作成するための柔軟性をスマート記憶ユニットに与える。

【００５２】

いくつかの実施形態の更なる利点は、本システムおよび方法が、１または２以上のスマート記憶ユニット上でファイルの再ストライピングを通じてデータファイルの実時間の再配置(relocation)を可能にする。いくつかの実施形態において、この再ストライピングは、有利には、システム故障が再ストライピング処理中に発生してもデータの完全性を保ち且つ回復可能性を確保する最小限のデータブロックの移動で実施される。この再ストライピングシステムは、有利には、特定のデータブロックがスマート記憶ユニット上の何れの特定のロケーションにも配置されることを必要とせず、格納されたデータブロックに対するプロテクションを提供することを継続し、たとえ１または２以上のスマート記憶ユニットが故障してもデータの回復可能性を可能にする。

【００５３】

II. 動作例

説明のため、インテリジェント分散ファイルシステムが動作において使用されるところのシナリオ例が述べられるであろう。このシナリオ例においては、インテリジェント分散ファイルシステムは、インターネットウェブサイトを介した映画のダウンロードを提供する会社によって使用される。この会社は、ウェブサイトを介して顧客によってアクセスされる映画の予告編、広告、および顧客情報のみならず、ダウンロード可能な映画のコピーを格納し且つ管理するためにインテリジェント分散ファイルシステムを使用してもよい。このデータは、種々のレベルのプロテクションを用いて格納されてもよく、且つ高速アクセスのために複数のスマート記憶ユニットにわたって格納されてもよい。

【００５４】

例えば、電子メールに高速にアクセスするために、会社は、インテリジェント分散ファイルシステムにおける多数のスマート記憶ユニットにわたって顧客調査(customer survey)電子メールを格納することを望んでもよい。しかしながら、会社は、全ての電子メール

のバックアップテープを保持してもよく、そして顧客調査の迅速な回復を可能にすることは重要ではないように感じるかもしれない。会社は、インテリジェント分散ファイルシステムに、顧客調査電子メールにエラー訂正またはミラーリングプロテクションを使用しないように指示してもよい。従って、もし１または２以上のスマート記憶ユニットがアクセス不能になれば、会社は、その電子メールがバックアップテープから回復できるまでは、それらのスマート記憶ユニット上の顧客調査電子メールが遅延されることを受け入れることができると感じるかもしれない。

【００５５】

広告については、会社は、もし１または２以上のスマート記憶ユニットが故障すれば、インテリジェント分散ファイルシステムが、広告の表示を中断することなくデータを回復できるような高度のエラー訂正パラメータを使用するように、インテリジェント分散ファイルシステムに指示してもよい。例えば、会社は、種々の障害許容性測定(fault tolerance measurement)を信頼して、どの程度のプロテクションが特定のファイルに与えられるべきかを判断することを支援してもよい。重要な情報については、会社はＸの障害許容性レベルを確保し、且つさほど重要でない情報については、会社はＹの障害許容性を確保することを望んでもよい。ここでＸ＞Ｙである。従って、会社は、その広告主に、たとえ１または２以上のスマート記憶ユニットが故障したとしても、広告が信頼できるベースで利用可能であることを保証してもよい。障害許容性に加えて又は障害許容性に代えて他の測定が使用されてもよいことが理解され、そして障害許容性は信頼性のひとつの測定を説明するために使用される。

【００５６】

人気のある映画のダウンロードについて、会社は、有利には、インテリジェント分散ファイルシステムをセットアップして自動的に複数の映画データのコピーを格納し、より多くのユーザがそのデータにアクセスすることを可能とし、もし１または２以上のスマート記憶ユニットが故障すれば、欠損したデータが再生成または他のロケーションから読み出される。更にまた、もし要求の数が増加し、またはもし１または２以上のスマート記憶ユニットがスマート記憶ユニット上に存在するデータの要求で氾濫し始めていれば、人気のある映画ダウンロードの追加コピーは、生成され且つスマート記憶ユニット間に格納されてもよい。

【００５７】

会社は、さほど人気のない他の映画を提供してもよく、そして少ない需要のために少ないコピーを格納するようにインテリジェント分散ファイルシステムに対し指示してもよい。更に、“人気のあるダウンロード映画”の人気がなくなるにつれ、会社は、有利には、映画が格納されたスマート記憶ユニットから映画の余分なコピーを削除し、そして“人気の少ない”映画を低速性能を有するスマート記憶ユニット（例えば、少ない利用可能ディスクスペースを有するそれらのスマート記憶ユニット）に移動するように、インテリジェント分散ファイルシステムをセットアップしてもよい。インテリジェント分散ファイルシステムは、自動的に、スマート記憶ユニットを用いてこれらのタスクを処理してもよい。

【００５８】

加えて、会社がより多くの映画を獲得するにつれ、会社は、追加のスマート記憶ユニットをインテリジェント分散ファイルシステムに加えてもよい。そして、会社は、新たなスマート記憶ユニットを使用して、より多くの映画を格納し、より多くの既存の映画のコピーを格納し、応答時間を改善するために既存の映画データを再分散し、及び／又は１または２以上のファイルのプロテクションスキームを変更してもよい。たとえインテリジェント分散ファイルシステムが複数の記憶ユニット間にデータを格納し且つ管理するとしても、インテリジェント分散ファイルシステムが単一のファイルシステムとして見えるように、追加のスマート記憶ユニットがインテリジェント分散ファイルシステムに組み込まれても良い。

【００５９】

もし、スマート記憶ユニットのひとつが故障すれば、インテリジェント分散ファイルシ

ステムは、故障したユニット上に格納された任意のファイルデータを再構築し、その情報を稼動中のユニットに格納し、そして適切なメタデータのデータ構造を更新してもよい。従って、もしユーザが、故障したユニットに部分的に格納されたファイルを要求すれば、そのユーザは、それでもユニットのひとつがダウンしていることを知らずにそのファイルのデータを受信してもよい。

【００６０】

この例では、インテリジェント分散ファイルシステムは、首位映画ダウンロードに対する確実且つ高速なアクセス、人気の少ない映画に対する高速アクセス、および顧客調査電子メールに対するアクセスを提供するための能力を会社に提供する。各ファイルについて、会社は、エラー及び／又は欠損訂正パラメータを設定してもよく、そしてどれだけ多くのファイルの追加的コピーが格納されるべきかを選択してもよい。いくつかの状況において、会社は、手動で、どれだけ多くのデータのコピーが格納されるべきかを選択し、そのデータを格納するための場所を決定してもよい。他の状況においては、会社は、どれだけ多くのデータのコピーが格納されるべきか、（もしあれば）使用されるべきエラー及び／又は欠損訂正スキームを選択し、及び／又はデータが格納されるべき場所を選択するために、本インテリジェント分散ファイルシステムの特徴を当てにしてもよい。従って、会社はその記憶スペースを効率的に使用してユーザの要求に一層良好に対処できる。記憶スペースは、まばらに要求されたファイルについては浪費されず、そして重要でないファイルについては、エラー訂正情報は生成されず且つ格納されない。

【００６１】

上述の例は、ダウンロード用の映画を提供する会社を含むが、この例は、インテリジェント分散ファイルシステムの一実施形態の特徴を例示するためだけに使用される。さらに、このインテリジェント分散ファイルシステムは、他の環境で使用されもよく、且つ、他とバ、サウンドファイル、オーディオファイル、グラフィックファイル、マルチメディアファイル、デジタル写真、実行ファイル等々を含む他のタイプのデータ及び／又はデータの組み合わせと共に使用してもよい。

【００６２】

III．インテリジェント分散ファイルシステム

図１は、ネットワークサーバ１２０と通信してリモートファイルアクセスを提供するインテリジェント分散ファイルシステムの一実施形態を示す。このインテリジェント分散ファイルシステム１１０は、例えばＮＦＳまたはＣＩＦＳのような種々のプロトコルを用いてネットワークサーバ１２０と通信してもよい。ユーザ１３０は、インターネット１４５のような通信媒体１４０を介してネットワークサーバ１２０と情報のやりとりを行って、インテリジェント分散ファイルシステム１１０によって管理されるファイルを要求する。代表的なインテリジェント分散ファイルシステム１１０は、スマート記憶ユニット１１４のセットおよびネットワークサーバ１２０と通信するスイッチコンポーネント１２５を使用する。このインテリジェント分散ファイルシステム１１０は、個々のファイルのブロックが複数のスマート記憶ユニット１１４にわたって分散されることを可能にする。このデータへのアクセスが、このデータが単一の装置上に格納された場合よりも高いスループットを提供するように、このデータが格納される。加えて、インテリジェント分散ファイルシステム１１０は、種々のプロテクションスキームを用いて格納された種々のデータファイルを格納するために使用されてもよい。

【００６３】

代表的なインテリジェント分散ファイルシステム１１０は、スマート記憶ユニット１１４のセット間にデータを格納する。このスマート記憶ユニット１１４についての詳細な説明に関しては、以下の“スマート記憶ユニット”と題された節を参照されたい。

【００６４】

代表的なインテリジェント分散ファイルシステムは、要求されているデータのタイプを処理することができるアプリケーションサーバに要求を方向付ける負荷バランシングスイッチのようなスイッチコンポーネント１２５を使用する。入来する要求は、遅延を最小化

し且つデータの完全性を確保するために、高速技術を用いて適切なアプリケーションサーバに転送される。

【００６５】

標準イーサネット（登録商標）スイッチまたは他の負荷バランシングスイッチと同様に、例えば１０００Base-T（銅）ギガビット負荷バランシングイーサネット（登録商標）スイッチ、エクストリームネットワークサミット(Extreme Networks Summit)７Ｉ、ファウンドリファストアイアン(Foundry Fast Iron)ＩＩ、ノーテルネットワークアルテオンＡＣＥスイッチ(Nortel Networks Alteon ACEswitch)１８０、Ｆ５ Ｂｉｇ－Ｉｐのような種々の負荷バランシングスイッチ１２５が使用されてもよいことが理解される。インテリジェント分散ファイルシステムは、例えば“ジャンボ”イーサネット（登録商標）フレームのような大きなフレームサイズをサポートするスイッチを使用する。加えて、負荷バランシングスイッチ１２５は、他の市販製品及び／又は有標製品のみならず、ファウンドリネットワークスのサーバアイアン(ServerIron)スイッチ、アサンテ(Asante)のインストラスイッチ(InstraSwith)６２００スイッチ、アサンテのホットスタック(HotStack)、シスコ(Cisco)のカタリスト(Catalyst)スイッチを用いて実施されてもよい。しかしながら、当業者であれば、多様なスイッチコンポーネント１２５が使用され、または他の技術が使用されてもよいことが分かる。さらに、種々のネットワークフレームサイズを伝送するようにスイッチコンポーネント１２５を構成してもよいことが分かる。

【００６６】

ディスク、マザーボード、ＣＰＵ、オペレーティングシステム、または１または２以上のスマート記憶ユニットへのアクセスを妨害する他のハードウェアまたはフトウェア故障の場合には、重要性の高いファイルは、データに高い回復率を提供する高いエラー訂正パラメータを用いて格納されてもよい。もし、何れかのデータが失われ又は損なわれていれば、スマート記憶ユニット１１４は、他のロケーションからデータを取得し又はデータを再生成するために、メタデータにおける冗長情報またはミラーリング情報を使用してもよい。高い需要があるファイルは、なお一層高いスループットレートを提供するために、追加のスマート記憶ユニット１１４にわたって実時間でミラーされてもよい。

【００６７】

インテリジェント分散ファイルシステム１１０の一実施形態において、メタデータ構造は、少なくとも、メタデータ構造に対応するディレクトリに由来するもの(descendant)を含んで参照するデータと同じプロテクションを有する。メタデータ構造なしでデータを読み出すことは困難であるので、メタデータ構造におけるデータの欠損(loss)は、インテリジェント分散ファイルシステム１１０を害する。インテリジェント分散ファイルシステム１１０では、メタデータ構造の代わりのコピーは、要求されたプロテクションを提供するために、必要に応じた多くのロケーションにおいてミラーされてもよい。従って、パリティプロテクションを有するファイルは、少なくとも同一又はより大きなパリティプロテクションで格納されたそのメタデータ構造を備えてもよく、２回ミラーされたファイルは、少なくとも２つのロケーションでミラーされたそのメタデータ構造を備えてもよい。

【００６８】

図１はインテリジェント分散ファイルシステム１１０の一実施形態を示すが、他の実施形態が用いられてもよいことが理解される。例えば、アプリケーションサーバのような追加のサーバがスイッチコンポーネント１２５と通信してもよい。これらアプリケーションサーバは、例えば、オーディオストリーミングサーバ、ビデオストリーミングサーバ、画像処理サーバ、データベースサーバなどを含んでも良い。さらに、スイッチコンポーネント１２５と通信するワークステーションのような追加の装置が存在していてもよい。加えて、図１は、４つのスマート記憶ユニット１１４と共に動作するインテリジェント分散ファイルシステム１１０を示すが、このインテリジェント分散ファイルシステム１１０は、異なる数のスマート記憶ユニット１１４と共に動作してもよいことが理解される。

【００６９】

用語“リモート”は、装置、コンポーネント、及び／又はローカルに格納されないモジ

ュールを含んでも良く、それは、ローカルバスを介してアクセスできない。従って、リモート装置は、物理的に同一の部屋に設置され且つスイッチ又はローカルエリアネットワークのような装置を介して接続された装置を含んでも良い。他の状況において、リモート装置は、また、例えば異なったロケーション、土地(country)などのような離れた地理上の区域に配置されてもよい。

【００７０】

また、インテリジェント分散ファイルシステム１１０を用いて種々のタイプのデータが格納されてもよいことも理解される。例えば、インテリジェント分散ファイルシステム１１０は、例えば、ビデオオンデマンド(video-on-demand)、オンラインミュージックシステム、ウェブサイトミラーリング、大規模データベース、大規模グラフィックファイル、ＣＡＤ／ＣＡＭ設計、ソフトウェア更新、会社のプレゼンテーション、保険金請求ファイル、医療画像ファイル、会社の文書記憶などのような大きなファイルのアプリケーションで使用されてもよい。

【００７１】

図２は、ウェブサイトユーザ１３０がオンデマンドデジタルビデオを観るための要求を送信した環境例を示す。イベントＡにおいて、ユーザ１３０は、インターネット１４５を介してウェブサイトに要求を送信し、映画movie.movieのコピーを観ることを要求する。この要求は、ウェブサイトサーバ１２０によって受信され、そしてサーバ１２０は、このファイルが、＼movies＼comedy＼mymovie.movieに配置されていることを判断する。イベントＢにおいて、インテリジェント分散ファイルシステムのスイッチコンポーネント１２５は、この要求を調べてインテリジェント分散ファイルシステム１１０に接続し、そして標準負荷バランシング技術を用いて、この要求を、スマート記憶ユニット０のような利用可能なスマート記憶ユニット１１４に転送する。イベントＣにおいて、スマート記憶ユニット０は、ファイル/DFSR/movies/comedy/mymovie.movieに対する要求を受信し、そしてそのルートメタデータのデータ構造（ルートディレクトリ/DFSRのための）から、サブディレクトリmoviesのメタデータのデータ構造がスマート記憶ユニット２で格納されていることを判断する。イベントＤにおいて、スマート記憶ユニット０は、サブディレクトリcomedyのためのメタデータ構造のロケーションを要求するスマート記憶ユニット２に要求を送信する。イベントＥにおいて、スマート記憶ユニット０は、サブディレクトリcomedyのメタデータ構造がスマート記憶ユニット３で格納されているという情報を受信する。イベントＦにおいて、スマート記憶ユニット０は、ファイルmymovie.movieのメタデータ構造のロケーションを要求しているスマート記憶ユニット３に要求を送信する。イベントＧにおいて、スマート記憶ユニット０は、ファイルmymovie.movieのメタデータのデータ構造がスマート記憶ユニット０で格納されているという情報を受信する。そして、スマート記憶ユニット０は、イベントＨにおいてファイルmymovie.movieのメタデータのデータ構造をローカル記憶から読み取る。このメタデータのデータ構造から、スマート記憶ユニット０は、ファイルにおけるデータの各ブロックのロケーションを格納しているmymovie.movieのデータロケーションテーブルを読み取る。そして、スマート記憶ユニット０は、データロケーションテーブル情報を使用して、ローカルに格納されたブロックの読み取りと、他のスマート記憶ユニットで格納されたデータに対する要求の送信とを開始する。

【００７２】

ファイルのデータ又はデータの一部が読み取られた後に、ファイルデータが、要求元のサーバ１２０に送信されて、要求元のユーザ１３０に転送される。一例において、ファイルデータは、データがいつどのようにユーザ１３０に送信されるかを定めるビデオストリーミングサーバにルーティングされてもよい。いくつかの実施形態において、要求の遅れ(latency)を低減するために、要求されたものよりも多くのデータを読み出す先見読み出し技術(read ahead techniques)を利用することが有利であるが理解される。

【００７３】

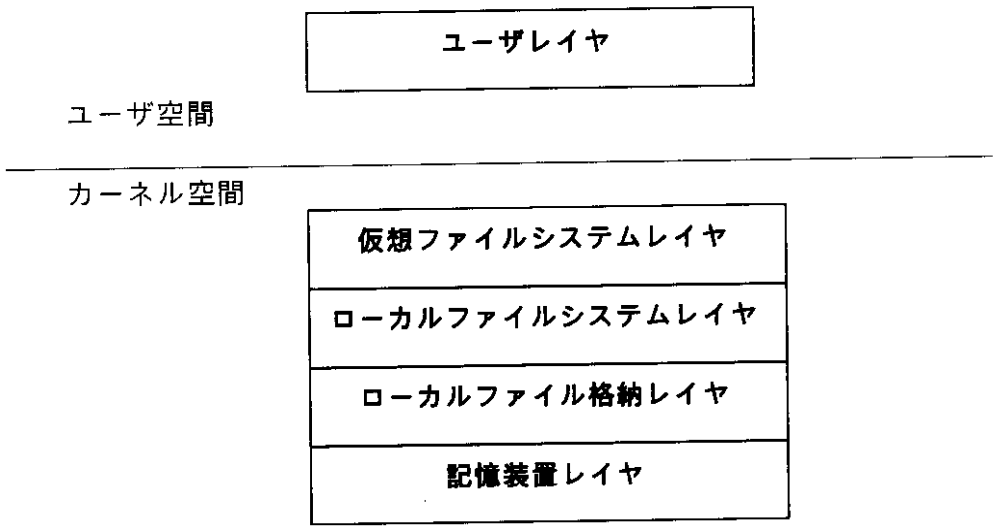
IV. インテリジェントファイルシステム構造

テーブルＩは、ファイルシステムレイヤのセット例の一実施形態を示し、そのファイル

システムレイヤを通して、物理的記憶装置をアクセスするためにファイル要求が処理される。この代表的ファイルシステムレイヤは、ユーザレイヤ(User Layer)、仮想ファイルレイヤ(Virtual File System Layer)、ローカルファイルシステムレイヤ(Local File System Layer)、ローカルファイルストアレイヤ(Local File Store Layer)、および記憶装置レイヤ(Storage Device Layer)である。

【００７４】

【表１】



【００７５】

ファイル要求の一つのタイプにおいて、例えば、マイクロソフトウィンドウズ（登録商標）ファイルシェアリングサーバプロトコルのバージョンを実施するユニックス上で使用されるＳＭＢＤ、ＦＴＰＤ、ＨＴＴＰＤ（アパッチウェブサーバ）のようなファイルシェアリングのためのユーザレベルプロトコルアプリケーションを介して、要求が受信される。ユーザレベルプロトコルアプリケーションは、カーネルレベルのオープン(open)、リード(read)、シーク(seek)、ライト(write)、または、例えば、Ｃランタイムライブラリlibcへのファンクションコールを行うことによるようなクローズシステムコール(close system call)を実行する。

【００７６】

システムコールは、バッファキャッシュを保持する仮想ファイルシステムレイヤ（“ＶＦＳ”）に受け渡される。このバッファキャッシュは、例えば、最も使用頻度が低いバッファのキャッシュ（“ＬＲＵ”）であってもよく、このバッファは、下層のファイルシステムレイヤから受信されたメタデータ構造またはデータを格納するために使用される。

【００７７】

次のレイヤは、ローカルファイルシステムレイヤであり、このローカルファイルシステムレイヤは、ファイルの階層的ネーミングシステムを保持すると共にディレクトリおよびファイルネーム要求を、下のレイヤであるローカルファイルストアレイヤに送信する。ローカルファイルシステムレイヤは、メタデータのデータ構造ルックアップ及びマネージメント(management)を処理する。例えば、いくつかのシステムにおいて、ユニックスベースのファイルシステムのように、メタデータのデータ構造は、ファイルアクセス許可、データブロックロケーション、およびレファレンスカウント(reference counts)についての情報を備えるファイルアブストラクション(file abstraction)である。一旦、そのネームによってファイルが開かれると、他のファイルオペレーションは、特定のファイルについてメタデータ構造を識別するユニークな識別子を介してファイルを参照する。このアプローチの利点は、単一のファイルが多くの異なるネームを有してもよく、単一のファイルが異なる複数のパスを介してアクセスされてもよく、そして、例えば‘ｍｖ’コマンドのような標準ＵＮＩＸ（登録商標）ユーザレベルユーティリティを介して実際のデータファイル

を上書きすることなく、ＶＦＳネームスペースにおける古いファイル上に新規なファイルがコピーされてもよい。現在のコンテンツサービスを混乱させることなく適切にコンテンツが更新されるので、これらの利点は、コンテンツ配信およびウェブホスティングのような環境において一層有利である。メタデータのデータ構造内のレファレンスカウントは、一旦、全てのオープンファイルハンドルが閉じられると、システムがデータブロックを無効とすることのみを可能とする。

【００７８】

４番目のレイヤは、ローカルファイルストアレイヤであり、このローカルファイルストアレイヤは、“バッファ要求からブロック要求へ”の変換(translation)およびデータバッファ要求マネージメントを処理する。例えば、ローカルファイルストアレイヤは、ブロック割り当てスキーム(block allocation scheme)を使用して、読み出し(reading)のためのブロック読み出しスキーム(block retrieval scheme)と同様に、ライトおよびリードについてのスループットを改善し且つ最小化する。

【００７９】

最後のレイヤは記憶装置レイヤであり、この記憶装置レイヤは、ファイルシステムによって使用される特定の一つのディスクハードウェア用の装置ドライバをホスト(host)する。例えば、もし物理的な記憶装置がＡＴＡディスクであれば、記憶装置レイヤはＡＴＡディスクドライバをホストする。

【００８０】

テーブル１はファイルシステムの一つのセット例を示すが、Ｈは、ファイルシステムレイヤが異なって定義されてもよく又は全く存在しなくてもよいような異なる環境におけるのと同様に種々の異なるレイヤでこのシステム及び方法が動作してもよいと認識される。

【００８１】

Ｖ．スマート記憶ユニット

一実施形態において、スマート記憶ユニット１１４は、プラグアンドプレイ(plug-and-play)の、高密度で、ラックマウント可能な装置であり、それは高スループットデータ配信に最適化されている。このスマート記憶ユニットは、単一の仮想ファイルシステムを提供するために、種々の他のスマート記憶ユニットと通信するように構成されてもよい。より多くの記憶スペースが必要とされるので、または、もし１または２以上のスマート記憶ユニットが故障すれば、追加のスマート記憶ユニットがシステム全体をダウンさせ又はサービスの中断を生じる必要なくインストールされてもよい。

【００８２】

本明細書で使用されるように、モジュールなる用語は、ハードウェアまたはファームウェアで実現されたロジックを称し、またはソフトウェア命令の集まりを称し、できれば入口点(entry point)または出口点(exit point)を有し、例えばＣまたはＣ＋＋のようなプログラミング言語で記述される。ソフトウェアモジュールは、コンパイルされると共に実行プログラムにリンクされて、動的リンクライブラリにインストールされてもよく、または、ＢＡＳＩＣ、Ｐｅｒｌ、Ｐｙｔｈｏｎのような翻訳プログラミング言語(interpreted programming language)で記述されてもよい。ソフトウェアモジュールは、他のモジュールまたは自身から読み出し可能であってもよく、及び／又はイベントまたはインタラプトに応答して呼び起こされてもよい。また、ソフトウェア命令は、ＥＰＲＯＭのようなファームウェアに組み込まれてもよいことが理解される。さらに、ハードウェアモジュールが、ゲートおよびフリップフロップのようなロジックユニットから構成されてもよく、及び／又はプログラマブルゲートアレイまたはプロセッサのようなプログラム可能なユニットから構成されてもよい。本明細書で述べられるモジュールは、好ましくは、ソフトウェアモジュールとして実施されるが、ハードウェア又はファームウェアで表現されてもよい。

【００８３】

図３は、マネージメントモジュール３２０、処理モジュール３３０、キャッシュ３４０、スタック３５０、および記憶装置３６０を備えるスマート記憶ユニット１１４の一実施形態を示す。代表的なスマート記憶ユニット１１４は、図１に示されるように、要求を送

信および受信するために、スイッチコンポーネント１２５と通信するように構成される。
【００８４】
Ａ．マネージメントモジュール
一実施形態において、スマート記憶ユニットは、例えば、インストール(installation)、パラメータ設定、インテリジェント分散ファイルシステムのモニタ、インテリジェント分散ファイルシステム１１０で発生するイベントのロギング(logging)、更新などの管理タスクを実行するためのマネージメントモジュール３２０を備える。
【００８５】
Ｂ．処理モジュール
代表的な処理モジュール３３０は、データファイルについての要求を受信し、要求されたデータファイルについてのローカル及び／又はリモートに格納されたメタデータを読み出し、そしてこの要求されたデータファイルのローカル及び／又はリモートに格納されたデータブロックを読み出すように構成されてもよい。加えて、処理モジュール３３０は、また、１または２以上の要求されたデータブロックが崩壊または失われたイベントにおいてエラー訂正およびデータ回復を実施してもよい。
【００８６】
一実施形態において、処理モジュール３３０は、ファイル要求に応答するための５つのモジュール、即ちブロック割り当てマネージャモジュール３３１、ブロックキャッシュモジュール３３３、ローカルブロックマネージャモジュール３３５、リモートブロックマネージャモジュール３３７、およびブロック装置モジュール３３９を備える。
【００８７】
１．ブロック割り当てマネージャモジュール
ブロック割り当てマネージャモジュール３３１は、リード要求に応答して、ブロックをどこに割り当て、このブロックをどこに配置するかを決定し、そして装置に故障回復を指示する。ブロックをどこに割り当てるかについての情報は、グラフィカルユーザインターフェイス又はシェルインターフェイス、またはこれらの組み合わせのようなツールを介してシステム管理者によって設定されたポリシー、デフォルトパラメータとして設定されたポリシーによって決定されてもよい。一実施形態において、ブロック割り当てマネージャモジュール３３１は、少なくともローカルファイルシステムレイヤに存在し、そしてバークレイソフトウェアデザインユニバーサルファイルシステム（“ＢＳＤ ＵＦＳ”）に代えて、及び／又はＴＣＰ／ＩＰおよびイーサネット（登録商標）のような標準ネットワーキングソフトウェアレイヤと共に動作する。
【００８８】
代表的なブロック割り当てマネージャモジュール３３１は、３つのサブモジュール、即ちブロック要求変換モジュール(block request translator module)、フォワード割り当てモジュール(forward allocator module)、および故障回復モジュール(failure recovery module)を備える。
【００８９】
ａ．ブロック要求変換モジュール
ブロック要求変換モジュールは、到来するリード要求を受信し、ネームルックアップを実施し、適切な装置を探し、そしてこの装置からデータを引き出して要求を満足させる。もしこのデータが直ちに利用可能であれば、ブロック要求変換モジュールは、データのブロックがローカル記憶装置に格納されているか又は他のスマート記憶ユニットの記憶装置に格納されているかによって、データ要求をローカルブロックマネージャモジュールまたはリモートブロックマネージャモジュールに送信する。
【００９０】
一実施形態において、ブロック要求変換モジュールは、“インテリジェント分散ファイルシステム処理－ネームルックアップ処理”と題された節において述べられる以下のネームルックアップ処理を含む。
【００９１】

ブロック要求変換モジュールは、また、装置故障に応答してもよい。例えば、もし装置がダウンしていれば、ブロック要求変換モジュールは、使用中のデータ(data using)、例えばパリティ情報を再構成するために使用されるローカル及びリモートデータブロックを要求してもよい。従って、このデータは、たとえリードが実施されなくても生成される。加えて、ブロック要求変換モジュールは、故障回復モジュールがパリティ又は他のエラー又は欠損訂正データを用いてデータを再生成し、インテリジェント分散ファイルシステムにおける自由スペースにわたって欠損訂正データ(loss correction data)を再ストライピングするように、故障回復モジュールと通信してもよい。他の実施形態において、ブロック要求変換モジュールは、崩壊したデータ又は欠落したデータの鮮明なコピー(clean copy)を要求してもよい。

【００９２】

ｂ．フォワード割り当てモジュール

フォワード割り当てモジュールは、例えば冗長性、スペース、および性能のような要素に基づき、どの装置のブロックがライト要求に使用されるべきかを決定する。これらのパラメータは、システム管理者によって設定されてもよく、インテリジェント分散ファイルシステム１１０に組み込まれた情報から導き出されてもよく、インテリジェント分散ファイルシステム１１０にロジックとして組み込まれてもよく、これらの組み合わせによってもよい。フォワード割り当てモジュール１１０は、インテリジェント分散ファイルシステムを使用する他のスマート記憶ユニットから統計データ(statistics)を受信し、そしてこれらの統計データを使用して、新たに到来するデータを配置すべき最良のロケーションはどこなのかを決定する。集められたこの統計データは、例えば、ＣＰＵ稼働率、ネットワーク稼働率、およびディスク稼働率を含む。

【００９３】

フォワード割り当てモジュールは、また、リモートのスマート記憶ユニットの応答時間に基づき、このリモートのスマート記憶ユニットからの遅延情報(latency information)を受信してもよい。もし、中間装置の遅延が他のスマート記憶ユニットに比較して高いレベルに達すれば、割り当てスキームは、できれば冗長設定に基づき、遅いスマート記憶ユニットを十分に活用しない他のスマート記憶ユニットを助けるように調整されてもよい。一つの好都合な例において、インテリジェント分散ファイルシステムは、データのブロックを或るスマート記憶ユニットから他のスマート記憶ユニットに移動させ、対応するメタデータ構造を適宜更新してもよい。遅延状況(latency condition)は、ロギングシステムを通して記録されてもよく、そしてシステム管理者に報告されてもよい。遅いリンク状況(slow link condition)の原因は、例えば、不良のネットワークカード、不適切な双方向情報交換(duplex negotiation)、または比較的頻繁にリードまたはライトされる装置のデータであってもよい。

【００９４】

種々の方策が、データをどこに格納するかを決定するために使用される。これらの方策は、パラメータとの整合性、またはシステムの管理者により設定されるプリフェレンス(preference)、選択された冗長性レベルの一致、及び／又は性能改善のようなシステムの目標に基づき調整されてもよい。以下では、データを格納するためにフォワード割り当てモジュールによって採用される方策の例を少し提供する。以下に述べられることに加えて、またはそのことと共に、広範な種々の方策が使用されてもよいことが理解される。

【００９５】

フォワード割り当てモジュールは、複数のスマート記憶ユニットにわたってデータをストライピングするための割り当てスキームを備えてもよい。データのストライピングは、一般にはハイエンドのＲＡＩＤ記憶装置で使用される公知技術であるが、複数のディスクを備える単一ユーザワークステーションマシンに採用されてもよい。データのストライピングは、単に、ファイルのデータの異なる部分が、存続(live)し及び／又は異なる記憶装置またはディスクに格納されることを意味する。データのストライピングに対する利点は、リード要求が複数のディスクに配置されたブロックに及ぶときに、各ディスクがデータ

読み出しの総スループットに関与することである。代表的なシステムでは、データのストライピングはソフトウェア装置レイヤで行われる。即ち、ファイルシステムは、データのストライピングに関する情報を持たない。ファイルシステム下にあるソフトウェアレイヤのみがこの構造を理解する。いくつかの特化したハードウェアにおいては、このストライピングは、実際のハードウェアレイヤでソフトウェア装置レイヤの下でさえ行われる。インテリジェント分散ファイルシステム１１０においては、ファイルシステム自体は、データのストライピングを処理する。この実施は、より大きな柔軟性をストライピング構成に与える。例として、一般的なＲＡＩＤ技術は、全ディスクが同一のサイズでなければならず、且つ同一の性能特性を有していなければならないという制約がある。これらの制約は、データが異なる装置に均一に分散されることを保証するために必要である。ＲＡＩＤについての更なる詳細については、Paul Massigliaによる“The RAID Book”第６版（１９９７）を参照されたく、それは参照することにより本明細書に組み込まれる。

【００９６】

加えて、フォワード割り当てモジュールは、ユーザ要求（例えば、新しいスマート記憶ユニットのプロテクションスキームの付加など）に応答して、及び／又はシステム（例えば、故障したスマート記憶ユニットの検出）に応答してデータのストライピングを実施してもよい。しかしながら、再ストライピングは、ブロック割り当て管理モジュールにおける他のモジュールにより又は他のモジュールと共に実施されてもよく、またはスマート記憶ユニットの他の部分により又はスマート記憶ユニットの他の部分と共に実施されてもよい。

【００９７】

インテリジェント分散ファイルシステム１１０については、ディスクおよびディスクサイズを異なえることは、種々のスマート記憶ユニット１１４で使用され、そしてファイルストライピングに関与する。フォワード割り当てモジュールは、ディスク装置情報のためのルートメタデータ構造を調べ(look up)、そして、性能メトリックまたは前もってセットされた規則を用いて、ファイルデータが拡散されるべきスマート記憶ユニットの数を計算する。そして、フォワード割り当てモジュールは、ファイルのブロックをスマート記憶ユニットのセットに割り当ててもよい。

【００９８】

また、フォワード割り当てモジュールは、パリティ又は他のエラー又は欠損訂正プロテクションのための割り当てスキームを備える。ほとんどのＲＡＩＤシステムにおいては、ファイルストライピングが使用されるとき、ひとつを除いて全てのディスクがデータ記憶のために使用されるようにパリティプロテクションもまた使用される。最後のディスクは、純粋にパリティ情報のために使用される。このパリティ情報は、一般には、データディスクの全てにわたって各データのブロックのビット単位の排他的論理和（“ＸＯＲ”）をとることにより計算される。このパリティ情報は、ディスク故障が発生したときにデータ回復を実施するために使用される。失われたデータは、残りのディスクのデータブロックおよびパリティ情報のビット単位のＸＯＲをとることで再計算される。一般的なＲＡＩＤシステムでは、データは、失われたデータを再編するために交換用ディスクが配列(array)に挿入されるまで回復不能である。

【００９９】

インテリジェント分散ファイルシステム１１０では、パリティプロテクションがソフトウェア装置レイヤの代わりにファイルシステムレイヤで行われるので、失われたデータは、再計算されて、残りのスマート記憶ユニットの他の部分の自由スペースに再書き込みされてもよい。もし、そのデータを再書き込みするための十分な自由スペースが残っていなければ、パリティデータは、再計算されたデータで上書きされてもよく、そして冗長性がオリジナルのレベル以下に低下したという事実が記録され及び／又はシステム管理者に報告されてもよい。

【０１００】

また、フォワード割り当てモジュールは、データのミラーリングのための割り当てスキ

ームを備えてもよく、それは異なるスマート記憶ユニット上で利用可能なデータの複数のコピーを作成している。フォワード割り当てモジュールは、記憶スペース、ネットワーク稼働率、及び／又はＣＰＵ稼働率の観点から使用される度合いが最も少ないスマート記憶ユニットを用いて、それらスマート記憶ユニットにわたってデータのブロックのロケーションを負荷バランスするための割り当てスキームを使用してもよい。ミラーリングは、さらなる性能および更なる耐故障性を提供してもよい。もし、ミラーリングが、あるいくつかのコンテンツに対して要求されれば、フォワード割り当てモジュールは、ミラーされたデータと同様にオリジナルのデータにスペースを割り当てる。もし、１よりも大きな耐故障性レベルが要求されれば、フォワード割り当ては、耐故障性カウント(fault tolerance count)によって、スマート記憶ユニット、またはスマート記憶ユニットのサブセットを論理的に分割し、そしてストライピングされたデータを生成してもよい。例えば、もし、１０のスマート記憶ユニットがインテリジェント分散ファイルシステム１１０に存在し、且つ２の耐故障性が要求されれば、フォワード割り当ては、インテリジェント分散ファイルシステムを、それぞれが５つのスマート記憶ユニットの２つのセクションに分け、各セクションにおける４つのスマート記憶ユニットにわたってデータをストライピングし、そして各セクションから５番目のスマート記憶ユニットをパリティディスクとして使用してもよい。このスマート記憶ユニットの分割は、配列ミラー分割(array mirror split)と称される。配列ミラー分割の種々の実施は、記憶時にデータが重複(overlap)され又は歪曲(skew)される実施を含んで使用されてもよい。

【０１０１】

ｃ．故障回復モジュール

故障回復モジュールは、実時間でインテリジェント分散ファイルシステム１１０を再構成し、装置故障のためにもはや利用できないデータを回復する。故障回復モジュールは、性能を維持しながらサービスを中断することなく再構成を実施してもよく、そしてそのデータを短時間で所望の冗長レベルに戻してもよい。

【０１０２】

上述したように、リモートブロック管理モジュール３３７は、故障を検出して、このような故障の通知を故障回復モジュールに受け渡す。初期故障については、故障回復モジュールは、システム管理者により要求され又はインテリジェント分散ファイルシステム１１０によって設定されるような冗長パラメータに合致しない何れかのデータを配置する。

【０１０３】

最初に、パリティ情報から再生成され得るデータが再生成され、そして新たなデータにスペースを割り当てるために、要求がフォワード割り当てモジュールに送られる。フォワード割り当ては、ＣＰＵとネットワークの稼働率をモニタし、そしてＣＰＵおよびネットワーク稼働率が所定の目標に達するまで積極的に動作を始める。この所定の目標は、システム管理者によって設定されてもよく、あるいは、例えばコンピュータプロセッサのような要素によって予め設定されてもよい。一旦、その目標に達すると、故障回復モジュールは、スマート記憶ユニットの性能に与える影響を低減するために、その目標の時点で到達したレートで再計算を有利に実施してもよい。

【０１０４】

もし、最近故障した装置がオンラインに復帰すれば、故障回復モジュールはその回復した装置のリモートブロック管理モジュール３３７と通信して、データの完全性を確かめ、そして不一致があれば修正する。

【０１０５】

また、インテリジェント分散ファイルシステム１１０は、ホットスタンバイ装置の具備を支援してもよい。このホットスタンバイ装置は、現在のところは何のデータも処理していないが装置故障のときに使用される待機状態(idle)の記憶装置である。このような状況において、故障回復モジュールは、ホットスタンバイ装置のリモートブロック管理モジュール３３７と通信することにより、ホットスタンバイ装置を用いて、失われたデータを再構築してもよい。

【０１０６】
２．ブロックキャッシュモジュール
ブロックキャッシュモジュール３３３は、データブロックのキャッシング(caching)、ネームルックアップ、およびメタデータ構造を管理する。一実施形態において、ブロックキャッシュモジュール３３３は、ＢＳＤ仮想ファイルシステム(BSD Virtual File System)のバッファキャッシュの代わりに、またはＢＳＤ仮想ファイルシステムのバッファキャッシュと共に動作する。
【０１０７】
ブロックキャッシュモジュール３３３は、例えば頻度キャッシング(frequency caching)のような種々のキャッシングアルゴリズムを使用できるが、使用頻度最低(Least Recently Used)キャッシングアルゴリズムを用いてデータブロックとメタデータのデータブロックを格納(cache)してもよい。ブロックキャッシュモジュール３３３は、どちらが最善を遂行するかに応じて、どのブロックキャッシングアルゴリズムを使用するかを決定してもよく、他の実施形態では、アルゴリズムはデフォルトとして設定されてもよい。
【０１０８】
使用頻度最低キャッシング（“ＬＲＵ”）は、ほとんどのシステムで使用されている代表的なキャッシングスキームである。ＬＲＵは、一旦データがアクセスされると、そのデータは再びアクセスされる可能性が高いという原理に基づいている。従って、データは、最も長い時間アクセスされていないデータが捨てられるように、その最近の使用の順に格納される。
【０１０９】
頻度キャッシングは、最も頻繁にアクセスされているデータを格納する。ディスクライト(disk write)は、比較的時間集約的な動作であるため、追加的性能は、メタデータのデータ構造でアクセス頻度を追跡すること、およびアクセス頻度に基づきキャッシングすることにより得られる。
【０１１０】
加えて、ブロックキャッシュモジュール３３３は、より多くのデータが必要以上に要求される“先見読み出し(read ahead)”または“オンデマンド(on demand)”プロトコルを利用してもよい。ブロックキャッシュモジュール３３３は、データのセットに対する要求を送信してもよく、またデータのセットに先んじてその若干のデータを要求してもよい。例えば、ブロックキャッシュモジュール３３３は、１パケット先見読み出し、２パケット先見読み出し、１０パケット先見読み出し、１２パケット先見読み出し等々のような先見読み出しを実施してもよい。他の実施形態において、ブロックキャッシュモジュール３３３は、要求の遅延(latency)に基づいて先見読み出しを利用してもよい。例えばブロックキャッシュモジュール３３３は、Ｖパケット先見読み出しを実施してもよく、ここでＶはリンクの遅延(latency)および読み出しレート(read rate)を用いて計算される。ブロックキャッシュモジュール３３３は、また、ＣＰＵおよびネットワーク稼働率に基づいた他のアルゴリズムを使用して先見読み出しデータのサイズを決定してもよい。さらにまた、ブロックキャッシュモジュール３３３は、セットキャッシングプロトコルを使用してもよく、またはシステムの性能レベルに応じてキャッシングプロトコルを変えてもよい。
【０１１１】
キャッシュ３４０は、一般のマルチユーザのオペレーティングシステムに備わるデフォルトサイズを用いて実施されてもよく、またはシステム性能に大幅に影響を与えることのない範囲でキャッシュブロックサイズを異なる量に増加するように修正されてもよい。このような修正は、例えば、格納されるデータのタイプ、処理速度、インテリジェント分散ファイルシステムにおけるスマート記憶ユニットの数、及び使用されているプロテクションスキームのような要素に依存する種々の性能テストによって決定されてもよい。
【０１１２】
３．ローカルブロックマネージャモジュール
ローカルブロックマネージャモジュール３３５は、記憶装置３６０にローカルに格納さ

れたデータブロックの読み出し(retrieval)、記憶(storage)、割り当て(allocation)を管理する。ローカルブロックマネージャモジュール３３５は、ディスクから、例えばネットワークカードのような記憶装置３６０の他の部分にデータを移動させるためにゼロコピーファイル読み出し(zero copy file read)を実施し、これにより性能を改善してもよい。ローカルブロックマネージャモジュール３３５は、また、性能を向上させるために、使用されている記憶装置３６０に基づく修正を実施してもよい。一実施形態において、ローカルブロックマネージャモジュール３３５は、ローカルファイル格納レイヤ(Local File Store layer)に存在してもよく、ＦｒｅｅＢＳＤ高速ファイルシステム(FreeBSD Fast File System)の代わりに又はＦｒｅｅＢＳＤ高速ファイルシステムと共に稼動してもよい。

【０１１３】

一実施形態において、ローカルブロックマネージャモジュール３３５は、データを記憶装置３６０に格納するための要求を処理してもよい。一実施形態において、ローカルブロック管理モジュール３３５は、データが記憶装置３６０上のどこに格納されるかを決定する。例えば、ローカルブロック管理モジュールは、既に格納されているデータと関連するデータを受信したときに、新たなデータが、可能な限りその関連データの近くに格納されるように、関連データを隣接させて格納するように試みてもよい。しかしながら、種々の記憶プリフェレンス(storage preference)が使用され、且つ各スマート記憶ユニットが１または２以上の異なる記憶プリフェレンスを使用してもよいことが理解される。一実施形態において、インテリジェント分散ファイルシステムにおけるスマート記憶ユニットは同一の記憶プリフェレンスを使用してもよい。

【０１１４】

４．リモートブロックマネージャモジュール

リモートブロックマネージャモジュール３３７は、例えば、ブロック要求、ブロック応答、およびリモート装置故障の検出を含む中間装置通信を管理する。一実施形態において、リモートブロックマネージャモジュール３３７は、ローカルファイルシステムレイヤ(Local File System layer)に存在する。

【０１１５】

一実施形態において、スマート記憶ユニット１１４は、リモートブロックマネージャモジュール３３７を介してインテリジェント分散ファイルシステム１１０における他のスマート記憶ユニットと通信及び／又は接続してもよい。

【０１１６】

リモートブロックマネージャモジュール３３７は、スマート記憶ユニット１１４がＴＣＰのようなコネクションを介して互いにやり取り(talk)することを可能にしてもよい。一実施形態においては、各スマート記憶ユニットについて少なくとも２つのＴＣＰコネクションが存在し、ひとつはファイルデータトランスポテーションのためのものであり、もうひとつはコントロールメッセージトランスポテーションのためのものである。この二重チャンネル(dual channel)ＴＣＰ通信アーキテクチャの利点は、データがシステムの或る部分から他の部分にコピーされる必要なく、データブロックが複数のページサイズで送信される限りデータがＤＭＡ転送を介して直接的にネットワークインターフェイスカードからシステムメモリに送信されてもよく、且つＤＭＡ転送を介してシステムメモリからシステムの他の部分（おそらくは再びネットワークインターフェイスカード）に送信されてもよいことである。これは、データパケットが非データヘッダ(non-data header)を含まないときにＣＰＵがそのデータパケットを解析し、または情報がコントロールチャンネル上を伝送されるときにこの情報をＣＰＵが識別することに関与する必要がないためである。高性能のサーバ及びオペレーションシステムにおいては、システムメモリの或る部分から他の部分へのこれらのメモリコピーは、システム性能に関する厳しい制限になる。

【０１１７】

一実施形態において、リモートブロックマネージャモジュール３３７は、例えば、データブロックアクセスメッセージ（例えば、READ, READ_RESPONSE, WRITE, WRITE_RESPONSE）、メタデータアクセスメッセージ（例えば、GET_INODE, GET_INODE_RESPONSE, SET_ADD

RESS, GET_ADDRESS, INVALIDATE_INODE）、ディレクトリメッセージ（例えば、ADD_DIR, REMOVE_DIR）、ステータスメッセージ、同様の種々の他のタイプのメッセージのようなメッセージを利用するメッセージング通信を用いて通信する。

【０１１８】

ここでは二重チャンネルプロトコルについて述べたが、スマート記憶ユニット１１４間の通信を可能とするために他の通信プロトコルを使用してもよいことが理解される。

【０１１９】

５．ブロック装置モジュール

ブロック装置モジュール３３９は、ファイルシステムによって使用される特定の一部のディスクハードウェアのための装置ドライバをホスト(host)する。例えば、もし物理的記憶装置がＡＴＡディスクであれば、ブロック装置モジュール３３９はＡＴＡディスクドライバをホストする。

【０１２０】

Ｃ．キャッシュ

キャッシュメモリ又はキャッシュ３４０は、例えば、１ＧＲＡＭキャッシュのような本技術分野において公知の種々の製品を用いて実施されてもよい。図３に示されるキャッシュ３４０は、最近アクセスされたデータのブロック、または一定時間内にアクセスされるべきデータのブロックを格納してもよい。キャッシュ３４０は、スタティックＲＡＭ装置、ダイナミックＲＡＭ装置、内部キャッシュ、ディスクキャッシュ、同様の種々の他のタイプの装置のような高速記憶装置を用いて実施されてもよい。通常、データは、不揮発性記憶装置をアクセスするために要する時間よりも速くキャッシュ３４０からアクセスされる。もしスマート記憶ユニット１１４が記憶装置３６０からデータをアクセスする必要があれば、そのデータが既に読み出されたかを調べるためにキャッシュ３４０が最初にチェックされるようにキャッシュ３４０がデータを格納する。従って、キャッシュ３４０の使用は、データブロックの読み出しにおいてスマート記憶ユニットの性能を改善する。

【０１２１】

Ｄ．ネットワークスタック

一実施形態において、また、スマート記憶ユニット３１０は、例えばＴＣＰ／ＩＰのようなプロトコルを用いて、入ってくるメッセージトラヒックと、出て行くメッセージトラヒックとを処理するネットワークスタック３５０を備える。しかしながら、スタック３５０を実施するためには他のプロトコルまたはデータ構造が使用されてもよいことが理解される。

【０１２２】

Ｅ．記憶装置

記憶装置３６０は、１又は２以上の不揮発性メモリ装置のセットであり、それはデータブロックを格納するために使用されてもよい。記憶装置３６０は、例えば、４１．２５ＧＢ ＡＴＡ１００装置、ＳＣＳＩ装置等のような公知の種々の製品を用いて実施されてもよい。加えて、記憶装置３６０のサイズは、インテリジェント分散ファイルシステム１１０における全てのスマート記憶ユニット１１４について同一であってもよく、或いはそれは異なるスマート記憶ユニット１１４についてサイズを変えてもよい。

【０１２３】

Ｆ．システム情報

一実施形態において、スマート記憶ユニット１１４は、スマート記憶ユニット１１４が他のスマート記憶ユニット１１４と通信することを可能とするコンピュータ上で稼動する。このコンピュータは、例えば、ペンティアム（登録商標）プロセッサ、ペンティアム（登録商標）IIプロセッサ、ペンティアム（登録商標）Ｐｒｏプロセッサ、ペンティアム（登録商標）IVプロセッサ、ｘｘ８６プロセッサ、８０５１プロセッサ、ＭＩＰＳプロセッサ、パワーＰＣ（登録商標）プロセッサ、ＳＰＡＲＣ（登録商標）プロセッサ、アルファプロセッサ等のような１又は２以上マイクロプロセッサを用いた汎用のコンピュータであってもよい。

【０１２４】
一実施形態において、プロセッサユニットは、オープンソースのＦｒｅｅＢＳＤオペレーティングシステムを動作させ、そしてファイルのオープン、読み取り、書き込み、およびクローズのような標準オペレーティングシステムの機能を実施する。例えば、マイクロソフト（登録商標）ウィンドウズ（登録商標）３．Ｘ、マイクロソフト（登録商標）ウィンドウズ（登録商標）９８、マイクロソフト（登録商標）ウィンドウズ（登録商標）２０００、マイクロソフト（登録商標）ウィンドウズ（登録商標）ＮＴ、マイクロソフト（登録商標）ウィンドウズ（登録商標）ＣＥ、マイクロソフト（登録商標）ウィンドウズ（登録商標）ＭＥ、パームパイロットＯＳ、アップル（登録商標）ＭａｃＯＳ（登録商標）、ディスクオペレーティングシステム（ＤＯＳ）、ＵＮＩＸ（登録商標）、ＩＲＩＸ（登録商標）、Ｓｏｌａｒｉｓ（登録商標）、Ｓｕｎ（登録商標）ＯＳ、ＦｒｅｅＢＳＤ（登録商標）、Ｌｉｎｕｘ（登録商標）、ＩＢＭ（登録商標）ＯＳ／２（登録商標）オペレーティングシステムのような他のオペレーティングシステムが使用されてもよいことが理解される。

【０１２５】
一実施形態において、コンピュータは、例えば、イーサネット（登録商標）（ＩＥＥＥ８０２．３）、トークンリング（ＩＥＥＥ８０２．５）、ＦＤＤＩ(Fiber Distributed Datalink Interface)、またはＡＴＭ(Asynchronous Transfer Mode)のような従来のネットワーク接続性(network connectivity)を備える。さらに、コンピュータは、例えば、ＮＦＳ ｖ２／ｖ３overＵＤＰ／ＴＣＰ、マイクロソフト（登録商標）ＣＩＦＳ、ＨＴＴＰ １．０、ＨＴＴＰ １．１、ＤＡＦＳ、ＦＴＰ等のような種々のネットワークプロトコルを支援するように構成されてもよい。

【０１２６】
一実施形態において、スマート記憶装置１１４は、ジャンボ９Ｋイーサネット（登録商標）フレームを支援する１０００／１００ネットワークインターフェイスカードと同様に、シングル又はデュアルＣＰＵ ２Ｕラック搭載可能な構成を備える。しかしながら、種々の構成が使用されてもよいことが理解される。

【０１２７】
上述したように、異なるモジュールがスマート記憶ユニットに関して述べられたが、そのタスクは異なるモジュールによって実行されてもよいことが理解される。加えて、１又は２以上のモジュールが組み合わされることができ、及び／又は、ある特定のモジュールが特定のタスクを実施することを要求されないように、１又は２以上の新たなモジュールが付加されてもよい。

【０１２８】
VI．インテリジェント分散ファイルシステムデータ構造
図４は、インテリジェント分散ファイルシステムで使用されるディレクトリ構造例を示す。この例において、ルートディレクトリは、“DFSR”と名づけられ、そしてサブディレクトリIMPORTANT、TEMP、USERを備える。サブディレクトリIMPORTANTは、サブディレクトリPASSWORDSおよびCREDITCARDを備える。ファイルUSER.TXTおよびADMIN.TXTは、サブディレクトリPASSWORDSに格納される。従ってファイルUSER.TXTについてのアドレスは、

/DFSR/IMPORTANT/PASSWORDS/USER.TXT

である。ディレクトリおよびファイルについての情報またはメタデータは、インテリジェント分散ファイルシステム１１０により格納され且つ保持される。

【０１２９】
Ａ．メタデータのデータ構造
図５は、メタデータを格納するためのデータ構造例を示す。代表的なデータ構造５１０は、次の情報を格納する。

【０１３０】

【表２】

フィールド	説明
モード	ファイルのモード（例えば、レギュラーファイル、ブロックスペシャル、キャラクタースペシャル、ディレクトリ、シンボリックリンク、ｆｉｆｏ、ソケット、ホワイトアウト、アンノウン）
オーナー	ファイルのオーナーシップを有するスマート記憶ユニットに関するアカウント
タイムスタンプ	最終変更ファイルのタイムスタンプ
サイズ	メタデータファイルのサイズ
パリティカウント	使用されたパリティ装置の数
ミラーカウント	使用されたミラー装置の数
バージョン	メタデータ構造のバージョン
タイプ	データロケーションテーブルのタイプ（例えば、タイプ０、タイプ１、タイプ２、タイプ３）
データロケーションテーブル	データロケーションテーブルまたは実際のデータロケーションテーブル情報のアドレス
リファレンスカウント	これを参照するメタデータ構造の数
フラグ	ファイルパーミッション（例えば標準ＵＮＩＸのパーミッション）
パリティマップポインタ	パリティブロック情報に対するポインタ

【０１３１】
データ構造５１０の例は、メタデータを格納するためのデータ構造５１０の実施形態を示し、且つ種々の実施が本発明に従って実施されてもよいことが理解される。例えば、データ構造５１０は異なるフィールドを備えてもよく、このフィールドは異なるタイプのものであってもよく、このフィールドはグループ化されて別々に格納されてもよい。等々。
【０１３２】
図６Ａ，６Ｂ，６Ｃ，６Ｄは、データロケーションテーブルのタイプ、即ちタイプ０、タイプ２、タイプ３についてのデータロケーションテーブル構造例をそれぞれ示す。図６Ａにおいて、タイプ０のデータロケーションテーブルは、２４の直接ブロックエントリを備え、これは、データが格納されるロケーションを示す装置／ブロック番号対をデータロケーションテーブルにおけるエントリが備えることを意味する。図６Ｂにおいて、タイプ１のデータロケーションテーブルは、１５の直接ブロックエントリと、３つのシングル間接エントリと、３つのダブル間接エントリと、３つのトリプル間接エントリとを備える。シングル間接エントリについてのエントリは、直接エントリの追加的データロケーションテーブルが格納されるロケーションを示す。ダブル間接エントリについてのエントリは、データロケーションテーブルが格納されるロケーションを示し、このデータロケーションテーブルはシングル間接エントリを備える。トリプル間接エントリについてのエントリは、データロケーションテーブルが格納されるロケーションを示し、このデータロケーションテーブルはダブル間接エントリを備える。
【０１３３】
任意数の装置にわたって任意のブロックがミラーされてもよいので、メタデータのデータ構造５１０は複数のロケーションを用いてブロックを表現するのに十分柔軟性があり、且つ一定のスペース内での直接指標付け(direct indexing)に由来する高速アクセスを依然として提供する。従って、使用されるべきデータロケーションテーブルのタイプを指示するために、タイプは、メタデータ構造５１０と有利に関連づけられてもよい。メタデータのデータ構造５１０の一実施形態において、例えば２４のポインタのような２４のデー

タエントリのための場所(room)があってもよい。

【０１３４】

タイプ０は、データファイルが小さいとき：データロケーションアドレスが直接エントリとして格納されるときに使用されてもよい。従って、タイプ０のメタデータのデータ構造は２４の直接エントリを備える。タイプ１は、より大きなファイルと２回までのミラー（ファイルの３つのコピー）を支援するために使用されてもよい。タイプ１は、１５の直接エントリと、３つのシングル間接エントリと、３つのダブル間接エントリと、３つのトリプル間接エントリとを使用する。タイプ２は、７回までのミラーリング（ファイルの８つのコピー）を支援するために使用されてもよく、そして８つのシングル間接エントリ、８つのダブル間接エントリ、８つのトリプル間接エントリを備えてもよい。タイプ３のデータロケーションテーブルは、ディスクアドレスの全てがトリプル間接エントリとして格納されるので更なるミラーリングを可能する。結果として、２４までの完全なファイルコピーが格納される。

【０１３５】

種々のデータロケーションテーブルが使用されてもよいことが理解され、図６Ａ，６Ｂ，６Ｃ，６Ｄは実施例を示す。他の実施形態において、例えば、データロケーションテーブルは、直接および間接エントリの異なる混合体を備えてもよい。さらに、他の実施形態において、データロケーションテーブルは、テーブルの各エントリについてエントリのタイプを指定するエントリフィールドを備えてもよい。タイプは、例えば、上述したもの（例えば、直接、シングル間接、ダブル間接、トリプル間接）を備えてもよく、他（例えば、クアドラプル間接など）も同様である。加えて、データロケーションテーブルは、Ｘレベルまでのデータロケーションテーブルのより深いネスティング(nesting)を備えてもよく、ここでＸは整数である。

【０１３６】

また、メタデータのデータ構造は、どのスマート記憶ユニットがファイルのコンテンツとプロテクションデータを備えるかについての情報を備える。加えて、メタデータのデータ構造は、ファイルのコンテンツデータのために使用される最後のブロックアドレスとファイルのプロテクションデータのために使用される最後のブロックアドレスとを追跡(tracking)する各スマート記憶ユニットのための情報を格納してもよい。例えば、メタデータのデータ構造は、MYFILE.TXTが、装置０、装置２、装置３、および装置５に格納されたそのデータを有することを記録してもよい。メタデータのデータ構造は、また、次のことを記録してもよい。

【０１３７】

【表３】

	コンテンツのための最後のブロックアドレス	パリティのための最後のブロックアドレス
装置０	３００	００１
装置２	３０７	２０３
装置３	２００	３０３
装置５	１０３	５０１

【０１３８】

１．ディレクトリメタデータ

図７Ａは、ディレクトリPASSWORDSについてのメタデータのセット例を示す。図７Ａにおいて、データ構造はディレクトリPASSWORDSについての情報を格納する。ディレクトリは２回ミラーされる（合計３つのコピー）。ディレクトリ構造は比較的小さいので（例えばブロックに収まる）、使用される３つの直接ポインタのみが存在し、ひとつは各コピーのためのものである。メタデータのセット例は、使用されないブロックエントリ７３０のセットと同様に装置／ブロック番号対を用いてデータブロックのロケーションを示す直接

エントリ７２０を有するデータロケーションテーブル７１０を備える。

【０１３９】

２．ファイルメタデータ

図７Ｂは、ファイルUSER.TXTについてのメタデータのセット例を示す。図７Ｂにおいて、データ構造は、ファイルUSER.TXTについての情報を格納する。ファイルデータUSER.TXTについてデータブロックのそれぞれの一つのコピーが存在し、そしてそのデータは３＋１パリティスキームを用いて保護される。USER.TXTについてのコンテンツデータは、サイズが４５Ｋであり、ブロックサイズが８Ｋであり、従って６番目のデータブロックが全く使用されない６つのデータブロックが存在する。データロケーションテーブル７１０は、６つのデータブロックのそれぞれが格納されるロケーションを示し、ここでデータブロックは、装置番号とブロック番号とによって参照され、最初のエントリが最初のデータブロックに対応する。さらに、コンテンツデータのためのパリティ情報のロケーションはパリティマップ７４０に格納され、そのパリティマップのロケーションは、“パリティマップポインタ”としてのデータ構造の最後のロケーションによって指定される。ファイルUSER.TXTは、３＋１パリティスキームを用いて格納され、従って３つのデータブロックに対しパリティデータのブロックが格納される。この３＋１パリティスキームには６つのブロックが存在するので、パリティデータの２つのブロックが存在する（６を３で割って、最も近い整数に丸める）。パリティマップは、両方のパリティデータのブロックが格納されたロケーションを示し、ここでパリティデータのブロックは、装置番号とブロック番号とによって参照され、最初のエントリがパリティデータの最初のブロックに対応する。

【０１４０】

Ｂ．データロケーションテーブルデータ構造

インテリジェント分散ファイルシステム１１０は、どのようにファイルが格納されるかについての柔軟性のみならず、多種多様なファイルのための格納を提供する。ファイルデータのミラーリングおよび冗長は、インテリジェント分散ファイルシステムが、異なるファイルについての変化する冗長パラメータを支援することを可能とするファイルシステムレベルで実施される。例えば、いくつかのディレクトリはミラーされ、パリティが保護され、または全く保護されない。

【０１４１】

図８Ａ，８Ｂ，８Ｃは、変化するプロテクションタイプとレベルのデータファイルのためのデータロケーション情報を格納するために使用されるデータロケーションテーブル例を示す。図８Ａ，８Ｂ，８Ｃは、さまざまなデータロケーションテーブルを図解するためのものであり、種々の異なるフォーマット及び／又は構造が使用されてもよい。

【０１４２】

図８Ａは、対応するファイルの各データブロックが格納される場所を示すデータロケーションテーブル８１０の例を図解する。図７Ｂにおけるように、ファイルについての対応するメタデータは示されていないが、データロケーションテーブル８１０がメタデータのセットに対応してもよいことが理解される。代表的なデータロケーションテーブル８１０は、直接エントリおよび間接エントリの両方を備える。

【０１４３】

直接エントリは装置ＩＤ／ブロック対を備える。装置ＩＤは、データが格納されたスマート記憶ユニットを示し、オフセットまたはブロックアドレスは、データが格納された記憶装置上のロケーションを示す。データロケーションテーブルにおける一つのエントリ例は、

エントリ	装置	ブロック
１	７	１２７

であり、これは、データのブロック１がブロック１２７の装置番号７上に格納されていることを示す。

【０１４４】

データロケーションテーブル８１０の例は、また、データロケーションテーブルがより

大きなデータのセットのためのデータロケーションを追跡することを可能とする追加のデータロケーションテーブルを指し示す間接エントリを備える。間接エントリのレベルは理論的に制限されないが、そのレベルは、スループットレートを改善するために有利に制限されてもよい。例えば、データロケーションテーブルは、せいぜいダブル間接エントリ、またはせいぜいトリプル間接エントリのみを許容するように制限されてもよい。代表的なデータロケーションテーブル８１０は２つのレベルの間接エントリを示す。

【０１４５】

さらに、データロケーションテーブルの最後のエントリは、パリティマップのアドレス（もしあれば）を格納するために予約されてもよい。他の実施形態において、パリティマップのアドレスは、例えば、メタデータのデータ構造におけるエントリのように、他のロケーションに格納されてもよい。もし、データのセットがパリティプロテクションを備えていなければ、アドレス値はＮＵＬＬのような標準的な値に設定されてもよい。

【０１４６】

図８Ｂは、２つの追加的なロケーションにおいてミラーされたデータのためのデータロケーションテーブルを示す。このデータロケーションテーブルは、データの各コピーについてのブロック又はオフセットアドレスおよび装置ＩＤを備える。代表的なデータロケーションテーブルにおいて、ミラーされたロケーションは、ブロック単位ベース(block-by-block basis)で選択されている。例えば、特定のスマート記憶ユニットをミラーするために１又は２以上の記憶ユニットを選択することのような他のスキームが使用されてもよいことが理解される。図８Ｂにおけるデータロケーションテーブルは直接エントリのみを備えているが、間接エントリが使用されてもよいことが理解される。

【０１４７】

一実施形態において、ファイルのためのミラーリング情報は、ファイルの対応するメタデータ構造に格納されてもよい。この情報は、例えば、各コピーのためのデータロケーションテーブルのロケーションのみならず、データのコピー数を備えてもよい。データロケーションテーブルは、シングルデータ構造として格納されてもよく、及び／又はデータロケーションテーブルの別個のコピーは異なるロケーションに格納されてもよいことが理解される。

【０１４８】

ミラーされたデータを有する図８Ｂのデータロケーションテーブル例は、パリティプロテクションを備えないが、このデータロケーションテーブルは、パリティ情報を備えてもよいことが理解される。

【０１４９】

図８Ｃは、パリティマップを有するデータロケーションテーブルを示す。代表的なデータロケーションテーブルにおいて、データは、３＋１パリティスキームを用いて保護され、即ちパリティデータのセットは３つのデータブロックの全てから生成されている。例えば、ビット単位(bit-by-bit)、バイト単位(byte-by-byte)、ブロック単位(block-by-block)ベースでデータブロックのＸＯＲを演算してパリティブロックを生成することによるような、データ生成のための技術分野における公知技術が使用されてもよい。

【０１５０】

代表的なデータロケーションテーブルは、２１個のデータブロック（ブロック０からブロック２０）からなるデータファイルに関する情報を提供する。パリティスキームが３＋１であるので、パリティブロックは、３つのデータブロックの各セットについて生成される。テーブル２は、図８Ｃに示されるいくつかのパリティブロックといくつかのデータブロックとの対応を示す。

【０１５１】

【表４】

テーブル２

データブロック			パリティブロック
0 装置５ ブロック１００	1 装置９ ブロック２００	2 装置７ ブロック３０６	0 装置０ ブロック００１
3 装置５ ブロック１０３	4 装置９ ブロック２０３	5 装置７ ブロック３０３	1 装置８ ブロック００１

【０１５２】

データロケーションテーブルの例は、パリティマップまたはパリティロケーションテーブルを備える。この代表的なパリティマップにおいて、データを生成するために使用されるブロックエントリのセットとパリティマップとの間には１対１のマッピングが存在する。他の実施形態において、パリティマッピングは、また、装置故障によりその直接ロケーション(direct locations)の何れにおいても利用可能でないイベントにおいて、何れのブロックがデータを再生成するために一緒にＸＯＲがとられたパリティであるかを、装置およびブロック番号により特定する可変サイズのエントリを備える。他の実施形態において、パリティ生成スキームは、パリティデータの対応(correspondence)およびロケーションが、データを再生成するために一緒にＸＯＲがとられるべきブロックを特定することなくインテリジェント分散ファイルシステム１１０によって決定されてもよいように、予め設定される。

【０１５３】

一実施形態において、パリティマップは、例えば、メタデータのデータ構造に備えられるよりは、むしろ、メタデータのデータ構造の最後のエントリにおけるようなメタデータのデータ構造によって指し示されてもよい。このマップは、故障したスマート記憶ユニット１１４のまれな場合においてその使用が要求されるだけなので、メタデータ構造に直接的に備えられる代わりに指し示されてもよい。また、スマート記憶ユニット１１４が、データを再構築しながら一回でパリティマップをトラバース(traverse)し且つトラバースされるときにパリティマップを解析することを可能とするパリティ再結合ブロックを表わすために、パリティマップは可変サイズのエントリを使用してもよい。或る状況では、エントリを取得して解析するための計算およびＩ／Ｏ時間はパリティ計算時間に比較して無視できる。

【０１５４】

パリティロケーション情報を有する図８Ｃのデータロケーションテーブル８１０の例は、ミラーリング情報または間接エントリを備えないが、一方または両方がパリティロケーション情報と共に使用されてもよいことが理解される。さらに、他のデータ構造が使用されてもよく、データロケーションテーブルデータ構造は本発明の一実施形態を説明することのみを意図したものであることが理解される。

【０１５５】

Ｃ．データ例

図９は、データロケーションテーブル９１０およびパリティマップ９２０の例と、そのデータが格納される対応装置を示す。図９の例は、装置上の変化するロケーションにデータがどのように格納されるかを示し、データの“ストライプ(stripe)”が各装置上の異なるオフセットアドレスにわたって格納され、そしてパリティデータが、同一ファイルからのデータについてさえ種々の装置に格納されてもよいことを示す。他の実施形態においては、データは各装置上の同一オフセットアドレスに格納されてもよい。

【０１５６】

例えば、最初のストライプのためのパリティデータは、ロケーション４００で装置３上

に格納され、ロケーション１００で装置０に格納されたデータブロック０と、ロケーション２００で装置１上に格納されたデータブロック１と、ロケーション３００で装置２上に格納されたデータブロック２とに関連する。２番目のストライプのためのパリティデータは、ロケーション６００で装置２上に格納され、そしてロケーション３００で装置０上に格納されたデータブロック３と、ロケーション８００で装置４上に格納されたデータブロック４と、ロケーション７００で装置１上に格納されたデータブロック５とに関連する。

【０１５７】

いくつかの実施形態において、個々のスマート記憶ユニットは、どこで及び／又はどのようにロケーションをディスク上の実際のロケーションにマッピングするかを決定する。例えば、もし装置０が４つの物理的なハードディスクを備え、且つ各ハードディスクが１００ブロックの記憶容量を備えていれば、装置０は、ロケーション０からロケーション３９９への格納を可能にする。ロケーションをディスク上のブロックにどのようにマッピングするかを決定するために使用されるガイドラインのセット例は、次のようである。

ディスク番号＝（ディスクごとのブロックの番号／ロケーション）のフロア

ディスク上のブロック＝ロケーションＭＯＤディスクごとのブロックの番号

【０１５８】

ＭＯＤは、除算の余りをとるモジュラス(modulus)演算子である。上述のガイドラインは、ロケーションをディスク及びディスクブロックにマッピングさせるために使用されるガイドラインの単なる一例を表し、多くの他のガイドラインまたはスキームを使用できることが理解される。例えば、一実施形態は、各ディスクを表すブロック範囲の関連したリスト(linked list)を利用し、そしてリストトラバーサル(list traversal)を実施してもよい。関連したリストは、複数のサイズ化されたディスクを可能にする利点を有する。

【０１５９】

データ及びパリティ情報の記憶の柔軟性のために、新たなスマート記憶ユニットが付け加えられるに従って、システムを混乱させることなく、既存のデータが新たなスマート記憶ユニットに移動されてもよく（例えば、既存ユニット上のデータを削除する前にコピーを作成することによって）、及び／又は、新たなデータが新たなスマート記憶ユニット上に格納されてもよい。加えて、データブロックまたはファイル全体が、大量の要求、ディスク故障、冗長またはパリティパラメータにおける変更などに応じて実時間でコピー(copy)され又は移動(move)されてもよい。

【０１６０】

VII.インテリジェント分散ファイルシステム処理

Ａ．データの読み取り

図１０は、データの読み取り（“データ読み取り処理”）のためのフローチャートの一実施形態を示す。例えばディレクトリメタデータ、ファイルメタデータ、コンテンツデータなどのような種々のデータタイプが読み取られる。

【０１６１】

スタートステートから始めると、データ読み取り処理は、データが格納されているロケーションを受信する（ステート１０１０）。一実施形態において、ロケーションは、スマート記憶ユニットＩＤおよびオフセットまたはブロックアドレスを用いて指定されてもよい。他の実施形態において、記憶装置のＩＤが使用されてもよいが、他の実施形態において、ＩＤを他のＩＤにマッピングするためにテーブルが使用されてもよい。等々。

【０１６２】

次に、データ読み取り処理は、データがローカルに格納されているかどうかを決定する（ステート１０２０）。もし、データがローカルに格納されていれば、データ読み取り処理は、そのデータをローカル記憶装置から読み取る（ステート１０３０）。一実施形態において、データ読み取り処理は、最初にキャッシュをチェックし、そしてもしデータが存在しなければ、記憶装置をチェックしてもよい。他の実施形態において、データ読み取り処理は記憶装置のみをチェックしてもよい。

【０１６３】

もし、データがローカルに格納されていなければ、データ読み取り処理は、データの要求を、そのデータが格納されたスマート記憶ユニットに送信する（ステート１０４０）。一実施形態において、この要求は、図１に示されるスイッチコンポーネント１２５を介して送信される。それから、データ読み取り処理は、要求されたデータを受信する（ステート１０５０）。

【０１６４】

データ読み取り処理は、要求されているデータを収集し、そしてそのデータを戻す（ステート１０６０）。いくつかの実施形態において、データは、データのセット全体が収集された後に戻される。他の実施形態において、データの一部またはセットは、そのデータがローカル記憶から読み出され又は他のスマート記憶ユニットから受信されるときに戻される。その一部は、ファイルロケーションテーブルに従う順番(sequential order)で戻されてもよく、またはそれらは読み出され又は受信されるときに戻されてもよい。データが戻された後、データ読み出し処理はエンドステートに進む。

【０１６５】

図１０はデータ読み出し処理の一実施形態を示しており、そして他の実施形態が使用されてもよいことが理解される。他の例において、例えば、並列処理、パイプライン、または非同期Ｉ／Ｏのような技術または技術の組み合わせを用いて、複数のデータ読み出し処理によりデータが並列に読み出されるように２以上のデータ読み出し処理が同時に使用されてもよい。

【０１６６】

Ｂ．ネームルックアップ処理

図１１は、ネームルックアップのための処理（“ネームルックアップ処理”）の一実施形態を示す。スタートステートから始めると、ネームルックアップ処理は、ファイルネームを受信し（ステート１１１０）、ルートディレクトリのメタデータを読み出し、そしてルートメタデータのロケーションをCURRENTとして設定する（ステート１１２０）。一実施形態において、ルートディレクトリのデータは、図５のデータ構造のようなデータ構造に格納されてもよいが、種々のデータ構造が、ルートディレクトリのメタデータを格納するために使用されてもよいことが理解される。さらにまた、いくつかの実施形態において、ルートディレクトリのメタデータは、各スマート記憶ユニット１１４がルートディレクトリのメタデータのコピーと同一または類似のコピーを備えるように各スマート記憶ユニット１１４で格納されてもよい。他の実施形態において、ルートディレクトリのメタデータは、インテリジェント分散ファイルシステム１１０における他のロケーションに格納され、またはファイル要求と共にスマート記憶ユニット１１４に送信されてもよい。例えば、ミューテックス(mutexes)及び／又はセマフォ(semaphores)などによってロックするような、複数のデータのコピーの完全性を確保するための公知技術が使用されてもよい。

【０１６７】

そして、ネームルックアップ処理は、ファイルネームの一部である次のトークン(token)を読み取ってもよい（ステート１１３０）。そして、ネームルックアップ処理は、CURRENTのデータを格納するスマート記憶ユニット１１４からトークンのメタデータのロケーションのアドレスを要求する（ステート１１４０）。この要求は、ローカルまたはリモートであってもよい。それから、ネームルックアップ処理は、戻されたアドレスをCURRENTとして設定し（ステート１１５０）、そして他のトークンが存在しないかどうかを判断してもよ（ステート１１６０）。ここでトークンはディレクトリ階層における単一のレベルを表す。もし、他のトークンが存在すれば、ネームルックアップ処理はブロック１１３０に戻る。もし、それ以上トークンが存在しなければ、ネームルックアップ処理は、参照値をCURRENTに戻し（ステート１１７０）、そしてエンドステートに進む。

【０１６８】

ネームルックアップ処理の他の実施が使用され得ることが理解される。例えば、ネームルックアップ処理はファイルのメタデータを読み出しても良い。加えて、一旦、要求されたデータのロケーションが発見されると、ネームルックアップ処理は、データがローカル

または他のスマート記憶ユニットで格納されているかどうかを判断してもよい。もし、データがローカルに格納されていれば、ネームルックアップ処理はリード要求をスマート記憶ユニット１１４のローカルブロックマネージャモジュール３３５に送信してもよく、もしデータが他のスマート記憶ユニットに格納されていれば、ネームルックアップ処理は、リード要求をリモートスマート記憶ユニット１１４のリモートブロックマネージャモジュール３３７に送信してもよい。

【０１６９】

Ｃ．ファイル要求処理

図１２は、ファイル要求のための処理（“ファイル要求処理”）のフローチャートの一実施形態を示す。スタートステートから始めると、ファイル要求処理は、ファイルを読み出すための要求を受信する（ステート１２１０）。一実施形態において、ファイルは、ロケーションおよびファイルネームを備えるファイルの完全なパスネームを用いて指定される。他の実施形態において、パスは、相対パスであってもよく、及び／又は、ファイルのアドレスについての情報が格納されるテーブルのような他のデータ構造であってもよい。次に、ファイル要求処理は、図１１に示されるようなネームルックアップ処理を実施して、ファイルのメタデータのデータ構造のロケーションを決定する。

【０１７０】

そして、ファイル要求処理は、図１０に示され且つ上述したようなファイル読み出し処理を用いてファイルのメタデータを読み出すが（ステート１２３０）、他のファイル読み出し処理が使用されてもよい。一実施形態において、ファイルのメタデータは、インテリジェント分散ファイルシステムの全体にわたってファイルにおける各データのブロックが格納されているロケーションへのアクセスを提供するデータロケーションテーブルを備えてもよい。

【０１７１】

そして、ファイルにおける各データのブロックに対し（ステート１２４０，１２７０）、ファイル要求処理は、ファイルのメタデータにおいてそれを探し出すことによりデータブロックのロケーションを取得し（ステート１２５０）、そして図１０に示され且つ上述したようなファイル読み出し処理を用いることにより、そのデータブロックを読み出す（ステート１２６０）。ただし、他のファイル読み出し処理が使用されてもよい。

【０１７２】

それから、ファイル要求処理はファイルのデータを戻し（ステート１２８０）、そしてエンドステートに進む。いくつかの実施形態において、ファイルは、データのセット全体が収集された後に戻される。他の実施形態において、１又は２以上のデータのブロックは、データが読み出されるときに戻されてもよい。一部は、ファイルロケーションテーブルに従う順番で戻されてもよく、またはそれらは、それらが読み出され又は受信されるときに戻されてもよい。一実施形態において、ファイル要求処理は、順番にデータブロックを置いてもよく、及び／又はストリーミングサーバのような他のモジュールがデータブロックを順序づけてもよい。データが戻された後に、データ読み出し処理はエンドステートに進む。

【０１７３】

図１２はファイル要求処理の一実施形態を示し、そして他の実施形態が使用されてもよいことが理解される。例えば、ファイル要求処理は、図１１に示されるものとは異なるネームルックアップ処理を用いてファイルのロケーションを決定してもよい。他の例において、データブロックを読み出すために２以上のデータ読み出し処理が同時に使用されてもよく、例えば、並列処理、パイプライン、または非同期Ｉ／Ｏのような技術または技術の組み合わせを用いて、複数のデータ読み出し処理により並列にデータを読み出すことを可能にしてもよい。

【０１７４】

Ｄ．パリティ生成処理

図１３は、パリティ情報の生成（“パリティ生成処理”）のためのフローチャートの一

実施形態を示す。スタートステートから始めると、パリティ生成処理は、データのセットに関連するパリティスキーム情報を受信する（ステート１３１０）。データのセットは、ファイルデータ、ファイルメタデータ、ディレクトリメタデータ、ファイルデータのサブセットなどを表してもよい。パリティ生成処理は、データのセットに関連するデータロケーション情報を読み出す（ステート１３２０）。次に、各パリティのセットに対し（ステート１３３０，１３７０）、パリティ生成処理はデータのセットを読み出す（ステート１３４０）。例えば、もし、パリティが３＋１であれば、パリティ生成処理は、図１０に示されるようなデータ読み出し処理を用いて最初の３つのデータブロックを読み出す。次に、パリティ生成処理は、ビット単位、バイト単位、またはブロック単位ベースでデータのＸＯＲ演算を実施するようにして、データのセットについてのパリティデータを生成する（ステート１３５０）。それから、パリティ生成処理は、データをバッファに格納し、そしてデータのセットについてのパリティ情報が生成されるまでブロック１３３０に戻る。パリティ情報が生成された後に、パリティ生成処理は、そのパリティデータを格納すべき場所を決定する（ステート１３８０）。パリティ生成処理は、輪番方式のパリティスキーム(rotating parity scheme)を使用してもよい。ここで、ファイルデータの各連続したストライプについての各パリティブロックは輪番で次の装置上に格納される。パリティ生成処理は、パリティ情報がデータ情報と同時に失われることのない装置故障のイベントにおいて確保すべき現在のストライプのためのデータを保持しているどの装置とも異なる装置にパリティブロックを割り当てる。また、パリティ生成処理は、パリティ記憶のために考慮されることからいくつかの装置をふるい落とすために、記憶容量、ＣＰＵ稼働率、およびネットワーク稼働率のような他の要素を考慮に入れてもよい。そして、パリティ生成処理は、バッファされたデータを割り当てられたスペースに格納し（ステート１３９０）、パリティマップにパリティデータのロケーションを記録し（ステート１３９５）、そしてエンドステートに戻る。

【０１７５】

図１３はパリティ生成処理の一実施形態を示し、他の実施形態が使用されてもよいことが理解される。例えば、パリティ生成は、並列にデータのブロックを読み出し、そして並列にパリティ情報を生成してもよく、または公知のパイプラインあるいは非同期Ｉ／Ｏ技術を用いてもよい。さらに、パリティ生成処理は、一時的なバッファに書き込むことなくパリティ情報およびパリティ情報のロケーションを格納してもよく、または、パリティ生成処理は、パリティデータまたはパリティデータへのポインタを戻してもよい。

【０１７６】

Ｅ．データ回復処理

図１４は、欠損または崩壊したデータを回復するための処理（“データ回復処理”）の一実施形態を示す。スタートステートから始めると、データ回復処理は、使用されたパリティスキームに関する情報を受信する（ステート１４１０）。そして、データ回復処理は、故障または崩壊したディスクまたはデータについての情報を受信する（ステート１４２０）。次に、データ回復処理は、欠損または崩壊したデータが割り当てられたパリティブロックグループについてのアドレス情報を受信する（ステート１４３０）。そして、データ回復処理は、利用可能なスマート記憶ユニットからデータブロックを受信する（ステート１４４０）。データは、図１０のもののようなデータ読み取り処理を用いて読み取られてもよい。データ回復処理は、パリティスキームに従ってブロックのＸＯＲをとることのような誤り訂正を実施し（ステート１４５０）、そしてその結果をバッファに格納する（ステート１４６０）。バッファのデータは欠損したデータ(missing data)を表す。そして、データ回復処理は、バッファにあるデータを戻し（ステート１４７０）、そしてエンドステートに進む。

【０１７７】

図１４はデータ回復処理の一実施形態を示し、そして他の実施形態が使用されてもよいことが理解される。例えば、データ回復処理は回復されたデータを格納することなく、それを戻してもよい。

【０１７８】
VIII.分散ファイルシステムにおけるファイルの再ストライピング
いくつかの実施形態において、インテリジェント分散ファイルシステムは、スマート記憶ユニット間に分散されたファイルを再ストライピングするためのシステム及び方法を備える。既に分散されてインテリジェント分散ファイルシステムに格納されたファイルは、ファイルに対するユーザのアクセスを中断することなく且つシステムをオフラインにすることなく、再分散されてシステムに再格納されてもよい。加えて、データは、最小限のデータ移動でスマート記憶ユニット間に再ストライピングされてもよく、再ストライピング処理中にシステム故障が発生したとしても、通常は保護され且つ回復可能である。
【０１７９】
再ストライピング処理は、例えば、欠損データが再生成されてシステムに格納されるような或るタイプの故障にスマート記憶ユニットの一つが陥ったときに使用されてもよい。再ストライピング処理は、また、１又は２以上のスマート記憶ユニットが、データが新たなスマート記憶ユニットに付加され又は他のスマート記憶ユニットに再分散されるようなインテリジェント分散ファイルシステムに付加され又はこのインテリジェント分散ファイルシステムから削除されるときに使用されてもよい。加えて、再ストライピング処理は、ファイルのプロテクションスキームが変更されるときに使用されてもよい。例えば、もし、ファイルが３＋１パリティプロテクションから４＋１パリティプロテクションに移れば、再ストライピング処理は、新しいパリティプロテクションに合ったレイアウトでスマート記憶ユニットにデータを移動してもよく、新しいレイアウトが完成するまで、古いパリティスキームによってデータが保護されるような古いレイアウトの下でユーザがファイルにアクセスすることを可能とし続ける。
【０１８０】
一実施形態において、再ストライピング処理は、ブロック割り当てマネージャによって実施されるが、他の実施形態においては、再ストライピング処理は、インテリジェント分散ファイルシステムの他の部分によって実施されてもよい。
【０１８１】
上述の方法論を用いるスマート記憶ニットは、如何なる特定のストライピングもスマート記憶ユニット内の如何なる特定のロケーションに存在することを必要としないという利点を提供する。従って、“データストライプ”の抽出は、複数のスマート記憶ユニットにわたってブロックの何れの特定のセットにも関与する必要はないが、異なるユニットからの何れかの利用可能なブロックを有利に含む。
【０１８２】
Ａ．再ストライピング処理
図１５は、インテリジェント分散ファイルシステム内のデータの再ストライピング（“再ストライピング処理”）のための方法の一実施形態を示す。ファイルは、ファイルの所望のプロテクションスキームに基づき決定されるプロテクショングループのセットとして論理的に表される。例えば、もし、ファイルの所望のプロテクションスキームが３＋１パリティであれば、ファイルは、４つのクラスタまたはブロック、即ち３つのコンテンツブロックと１つのパリティブロックを有するプロテクショングループに分割される。もし、ファイルの所望のプロテクションスキームが、３ｘミラーされたプロテクションスキームであれば、ファイルは、３つのクラスタ、即ち３つの同一のコンテンツデータブロックを有するプロテクショングループに分割される。
【０１８３】
本明細書で述べられる再ストライピング処理は、システム内のデータを移動させるための方法論を有利に表すが、スマート記憶ユニット上の特定のロケーションにデータが存在することを必要としない。再ストライピング処理は、また、再ストライピング処理中のデータ回復を可能とし、且つデータのストライピングにおける実質的な変化を可能にする。加えて、プロテクションスキームの制約が満たされる。例えば、パリティデータおよびその関連するコンテンツデータは、異なるスマート記憶ユニットにそれぞれ格納される。ミ

ラーされたデータについては、各データのコピーは異なるスマート記憶ユニット上に格納される。

【０１８４】

一実施形態において、再ストライピング処理は、スマート記憶ユニット間にデータをどのように格納するかを決定するときに他の目的を可能とするためのプリフェレンス(preference)を用いて有利に再ストライピングを行ってもよい。

【０１８５】

一例において、上記のプリフェレンス(preference)とは、最小限の移動が、もしブロックが既にスマート記憶ユニットに格納されていれば、上記ブロックが異なるスマート記憶ユニット上にリストアされる必要がないように同一のスマート記憶ユニットがブロックに割り当てられるような優先事項(priority)であるということである。このプリフェレンスは、スマート記憶ユニットが故障した後にファイルを修復するときに使用できる。

【０１８６】

他のプリフェレンスは、最適なスマート記憶ユニットがレイアウトのために使用されることである。もし、ブロックが既にスマート記憶ユニット上に配置され且つそのスマート記憶ユニットが“最適な”ユニットの一つであれば、同一のスマート記憶ユニットがそのブロックに割り当てられる。従って、ブロックのかなりの移動が回避されるが、ブロックは、スマート記憶ユニットのセットにわたってファイルをバランスさせることが必要なときに移動され得る。このようなプリフェレンスは、例えば、プロテクションの設定が変更され、または新しいスマート記憶ユニットがインテリジェント分散ファイルシステムに付加されたときに使用され得る。

【０１８７】

さらにプリフェレンスは、データが既に格納された場所とは無関係に最適なレイアウトのためのものである。従って、再ストライピングは、既存のブロックのロケーションに何ら構うことなく行われる。代わりに、既存のブロックがファイルを最適にレイアウトするために移動されてもよい。しかしながら、いくつかのデータブロックは、適切なスマート記憶ユニットに既に格納されているので移動の必要がないことが理解される。いくつかの実施形態において、インテリジェント分散ファイルシステムは、記憶装置上のフラグメンテーション(fragmentation)を修復するため、最適なスマート記憶ユニット上にたまたま存在していたとしても、ブロックを再配置することを望んでも良い。このプリフェレンスは、例えばリード性能のための最適化又はデフラグメンテーション(defragmentation)のような動作を調整するために使用されてもよく、そして“必要に応じた(as-needed)”基準で、又は通常のファイルシステム動作中のバックグランドで使用されるかもしれない。

【０１８８】

“最適な”レイアウトの目標は、１又は２以上の要素に基づいている。例えば、その要素は、ファイルサイズ、リード性能、ライト性能、予想されるアクセス頻度、システムスループット、ネットワーク速度、利用可能な記憶装置スペースなどを含む。

【０１８９】

一実施形態において、再ストライピング処理は、もし再ストライピング処理中に１または２以上のスマート記憶ユニットが故障すれば、プロテクションデータを用いてファイルが再生成されることを可能とするステートにおいてファイルを残す(leave)ことを試みる。プリフェレンスができれば使用されるかもしれないが、プリフェレンスが適合しない状況が存在するかもしれないことが理解される。例えば、もしブロックＡ，Ｂ，Ｃが３ｘミラープロテクションスキームの一部であり、ブロックＡ，Ｂが装置０上にあり、且つブロックＣが装置１上にあれば、たとえ装置０上の残留ブロックＢが最小限のデータ移動のプリフェレンスを満足するとしても、プロテクションスキームの制約を満たすために、ブロックＡまたはＢの何れかが他の利用可能なスマート記憶ユニット上にリストア(restore)されなければならないであろう。このアルゴリズムの実行中に、プロテクションスキームの制約およびプリフェレンスが互いに対峙している周期的なインスタンス(instance)が存在すれば、いくつかの実施形態において、システムは、余分のデータ移動の追加費用でプ

ロテクションを維持することを選択する。他の実施形態において、１又は２以上のプリフェレンスは、特定の実施によってはプロテクションスキームに優先してもよい。

【０１９０】

ここで、再ストライピング処理の維持が、図１５を参照して述べられる。スタートステートから始めると、再ストライピング処理は次のステートに進み、ここで、各プロテクショングループに対し（ステート１５２０，１５４０）、再ストライピング処理は、スマート記憶ユニットをプロテクショングループのブロックに割り当てる（ステート１５３０）。割り当て処理の一実施形態は、図１６を参照して以下に述べられるが、他の割り当て処理が使用されてもよい。

【０１９１】

ブロックがスマート記憶ユニットに割り当てられた後、もしブロックが新たなスマート記憶ユニットに割り当てられれば（即ち、それは、割り当てられたスマート記憶ユニットに既に格納されていない）、再ストライピング処理は、ブロックを格納するため、割り当てられたスマート記憶ユニットに要求を送信する（ステート１５５０）。再ストライピング処理は、プロテクショングループのセットが割り当てられた後に、割り当てられたスマート記憶ユニットに要求を送信し、そして次のステートに進む前にプロテクショングループの全てが割り当てられる必要がないというような他の実施形態が使用されてもよいことが理解される。例えば、もし１又は２以上のスマート記憶ユニットがシステムに加えられるときに再ストライピング処理が使用されれば、または、もしスマート記憶ユニットの一つが故障すれば、再ストライピング処理は、各プロテクショングループが割り当てられた後、割り当てられたスマート記憶ユニットに要求を送信してもよい。

【０１９２】

次に、再ストライピング処理は、データが無事に格納されたことを確かめる（ステート１５６０）。代表的な実施形態において、割り当てられたスマート記憶ユニットに既に格納されたデータはこの問い合わせ(query)を満足する。もしデータが無事に格納されていれば、再ストライピング処理は、ファイルに関連するメタデータのデータ構造を更新してエンドステートに進む（ステート１５９０）。一旦、更新が完了すると、ファイルの新たな情報は、システムによって使用される。先に使用された何れのメモリも解放され、そしてメタデータはファイルの新しいステートを反映する。

【０１９３】

メタデータの更新は、例えば、コンテンツブロックの新しいアドレスロケーション、プロテクションブロックの新しいアドレスロケーション、新しいプロテクションスキーム、ファイルのデータを格納するために使用されるスマート記憶ユニットの新しいリスト、およびファイルのプロテクション情報を格納するために使用されるスマート記憶ユニットの新しいリストを含む。再ストライピング処理は、また、各スマート記憶ユニットについてファイルデータおよびパリティデータのための新たな“最後のブロックアドレス”を用いてメタデータのデータ構造を更新してもよい。プロテクショングループのセットが格納された後に、データが無事に格納されたことを確かめ（ステート１５６０）、そして次のステップトに進む前にブロックの全てが格納される必要がないというような他の実施形態が使用されてもよいことが理解される。このような実施形態において、再ストライピング処理は、新しいデータのプロテクション値および古いデータのプロテクション値の最小公倍数(the least common multiple)に基づきメタデータのデータ構造を更新する前に無事に格納されなければならないプロテクショングループの数を判断してもよい。このようなインスタンスにおいて、ミラーされたデータについてのプロテクション値は１である（例えば、３ｘミラーされたプロテクションスキームについて、プロテクション値は１である）。パリティ保護されたデータについてのプロテクション値は、各パリティグループにおけるコンテンツブロックの数である（例えば、４＋１パリティスキームの下でのデータについて、プロテクション値は４である）。

【０１９４】

もし、データが無事に格納されなかったならば（例えば、１又は２以上のスマート記憶

ユニットがデータを格納できなかったならば）、再ストライピング処理は、エラーを戻し（ステート１５８０）、そしてエンドステートに進む。従って、もし１又は２以上のスマート記憶ユニットがデータを格納できなければ、元のデータが維持され、そしてファイルがその元のプロテクションスキームによって依然として保護される。新たに格納された何れのデータも解放される。

【０１９５】

再ストライピング処理の他の実施形態が使用されてもよいことが理解される。例えば、割り当て処理は、アドレスおよびプロテクション設定に基づき最適なスマート記憶ユニットが推奨され、且つ各メタデータのデータ構造が調べられるようにブロック単位ベースで割り当てる。

【０１９６】

Ｂ．格納処理

一実施形態において、割り当てられたスマート記憶ユニットがデータを格納するための要求を受信したとき、割り当てられた各スマート記憶ユニットは、データが記憶装置上のどこに格納されるかを決定する。一実施形態において、割り当てられたスマート記憶ユニットは、新しいデータを受信したときにスマート記憶ユニットがファイルの関連データの最後のブロックアドレスを用いてその新しいデータを最初の利用可能なロケーションに格納するように、関連データを隣接させて格納することを試みる。しかしながら、種々の記憶のプリフェレンスが使用されてもよく、且つ各スマート記憶ユニットが異なるセットの記憶のプリフェレンスを使用してもよいことが理解される。他の実施形態において、インテリジェント分散ファイルシステムにおけるスマート記憶ユニットの全ては、同一の記憶プリフェレンスを使用してもよい。一旦、新しいデータが格納されると、スマート記憶ユニットは、新しいデータのアドレスロケーションを戻す。しかしながら、他の実施形態において、割り当てられたスマート記憶ユニットは、データの新しいアドレスを用いてメタデータのデータ構造を直接的に更新してもよい。

【０１９７】

Ｃ．割り当て処理

図１６はデータブロックおよびプロテクションブロックを利用可能なスマート記憶ユニットに割り当てるための方法（“割り当て処理”）の一実施形態を示す。割り当て処理は、再ストライピング中にブロックがどこに割り当てられるべきかを決定する。代表的な実施形態において、割り当ては、単一のプロテクショングループにおいて実施される。従って、割り当ては、データのストライプのために実施され、ここでストライプはデータブロックおよび任意の関連プロテクションデータを含む（例えば、パリティデータ、ミラーされたデータのコピー）。割り当て処理は、最も起こり得ないブロック再配置(relocation)を含むレイアウトを構築してもよく、また一方では、ブロックについて最適なロケーションを決定し、そして任意の指定されたレイアウト目標を満足する。

【０１９８】

スタートステートから始めると、割り当て処理は次のステートに進み、どのスマート記憶ユニットが利用可能であるかを識別する（ステート１６２０）。一実施形態において、利用可能なユニットは、利用可能な自由スペースによって格納された残りのスマート記憶ユニットと同様にファイルによって現在使用されるスマート記憶ユニットを含む。しかしながら、他の実施形態において、スマート記憶ユニットは、異なって順序づけられてもよく、または利用可能なスマート記憶ユニットを順序づけるために他のデータ構造が使用されてもよいことが理解される（例えば、ランク番号が各スマート記憶ユニットに与えられても良い）。スマート記憶ユニットが“利用可能”であるどうかは、例えば、スマート記憶ユニットが機能しているかどうか、スマート記憶ユニットのスループットレート、スマート記憶ユニットの記憶装置上の自由スペースの量などのような１又は２以上の要素に基づいてもよい。

【０１９９】

次に、割り当て処理は、割り当てを実施するための利用可能なスマート記憶ユニットが

十分存在するかどうかを決定する（ステート１６３０）。スマート記憶ユニットの好ましい数は、例えば、プロテクショングループにおけるデータブロックの数、使用されているプロテクションスキーム、システムにおけるスマート記憶ユニットの最大数、ファイルの所望のスループットレートなどのような種々の要素に依存してもよい。例えば、３＋１パリティプロテクションを有するファイルは、少なくとも４つのスマート記憶ユニットを必要とする。もし、利用可能なスマート記憶ユニットが十分になければ、割り当て処理は、エラーを戻し（ステート１６４０）、そしてエンドステートに進む（ステート１６７０）。

【０２００】

もし、利用可能なスマート記憶ユニットが十分あれば、割り当て処理は、プロテクショングループにおける各ブロックについてスマート記憶ユニットを選択する（ステート１６５０）。スマート記憶ユニットを選択するための処理の更なる詳細な説明は、図１７を参照して以下に説明されるが、他の処理が使用されてもよい。

【０２０１】

いくつかの実施形態において、スマート記憶ユニットが選択された後、選択された各スマート記憶ユニットについて、割り当て処理は、現在のファイルに関連するデータブロックの最後のロケーション、及び／又は、現在のデータに関連するプロテクションブロックの最後のロケーションを決定し（ステート１６６０）、そしてエンドステートに進む。これは、新しいデータブロック及び／又は新しいプロテクションブロックがファイルからの他のデータブロック及びプロテクションブロックの近くに格納されることを可能にする。他の実施形態において、他の記憶のプリフェレンスが使用されてもよいことが理解される。例えば、データは、隣接して又は隣接しないで格納されてもよく、データは、先頭よりはむしろ最後尾に格納されてもよい。等々。

【０２０２】

Ｄ．選択処理

図１７は、単一プロテクショングループのクラスタを格納するスマート記憶ユニットを選択するための方法（“選択処理”）の一実施形態を示す。代表的な実施形態において、選択処理は、装置をファイルのプロテクショングループに割り当てるために使用される。上述のように、プロテクションスキームの制約は、プロテクショングループにおけるブロックが異なるスマート記憶ユニット上にそれぞれ格納されることを要求する。従って、ミラーされたファイル、ブロックの各コピーは異なるスマート記憶ユニット上に格納されるべきであり、且つパリティ保護されたファイルについては、コンテンツデータの各ブロック及びその関連パリティデータは異なるスマート記憶ユニット上に格納されるべきである。代表的な選択処理は、スマート記憶ユニットを単一のプロテクショングループに割り当てるために使用されるが、他の実施形態において、選択処理は、スマート記憶ユニットを、より小さな及び／又はより大きなデータのセットに割り当てるために使用されてもよい。例えば、選択処理は、ブロック上、プロテクショングループのセット上等で、単一ブロックと連動してもよい。

【０２０３】

１．実施例

ここでは装置として参照されるスマート記憶ユニットのセット全体、インテリジェント分散ファイルシステムにおいて利用可能な装置は、Ｔとして表現される。動作している装置のセットは、インテリジェント分散ファイルシステムにおいて稼動している装置の全てのセットであり、Ｇとして表現される。Ｇは、或るタイプの故障（例えば、システム故障、ネットワーク故障、記憶装置故障など）により“ダウン”していたインテリジェント分散ファイルシステムにおける装置を除く。加えて、Ｗは、ファイルのコンテンツデータ及びプロテクションデータが現在格納されている装置のセット（即ち、ファイルによって占められている装置）を表現する。もし、ファイルがインテリジェント分散ファイルシステムに決して格納されることがないとすれば、Ｗは空(empty)である。

【０２０４】

ステートステートから始めると、選択処理は次のステートに進み、そして好ましい装置のセットを識別する（ステート１７１５）。

【０２０５】

代表的な実施形態において、ファイルが分散される装置の数は、選択されたプロテクションスキームの制約を満たすのに十分に大きくあるべきである。例えば、ｍ＋ｎのパリティプロテクションを達成するために、少なくともｍ＋ｎの装置が必要とされる。ｋ回のミラーリングを達成するためには、ｋ個の装置が必要とされる。

【０２０６】

加えて、ファイルが分散される装置の数は、ファイルが損傷を受けやすくしないように十分に小さくあるべきである。特定の装置が故障すればファイルが影響を受ける可能性が増加するため、ファイルは、より多くの装置に拡散されるときに損傷を受けやすくなるかもしれない。従って、システムは、ファイルのデータが占められる装置の最大数ｍａｘとして上限を選択してもよい。

【０２０７】

従って、ファイルの幅に関する制約は、次のように表現される。

ｍ＋ｎを用いたパリティ保護：　｛ｍ＋ｎ｝＜＝ファイルの幅＜＝ｍａｘ

ｋ回のミラーリング：　｛ｋ｝＜＝ファイルの幅＜＝ｍａｘ

【０２０８】

好ましい装置Ｐのセットを選択するため、システムは、ｍａｘを使用してＰのサイズを設定する。ＧおよびＷの共通集合(intersection)からの装置はＰに加えられ、そしてもしＰが依然として最大のサイズでなければ、Ｇ及びＷ！の共通集合からの装置は、Ｐが最大サイズに到達するまでＰに加えられる。Ｐに加えられないＧ及びＷ！の共通集合からの残りの装置はＳに加えられる（ステート１７２０）。

【０２０９】

従って、もし|“ｍａｘ”|＝＝|Ｗ＆Ｇ|であれば、Ｐはデータが既に存在するＵＰ装置のみを含み、且つＳは、データが既に存在しないＵＰ装置のみを含むであろう。もし|“ｍａｘ”|＜|Ｗ＆Ｇ|であれば、Ｐは、データが既に存在するＵＰ装置のいくつかを含み、且つＳは、データが既に存在しないＵＰ装置のいくつかと同様のデータが既に存在するＵＰ装置のいくつかを含むであろう。もし|“ｍａｘ”|＞|Ｗ＆Ｇ|であれば、Ｐは、データが既に存在するＵＰ装置と、データが既に存在しないＵＰ装置のいくつかとを含むであろう。Ｓは、データが既に存在しないＵＰ装置のいくつかを含むであろう。

【０２１０】

好ましい装置を選択する他の方法もまた使用されてもよいことが理解される。

【０２１１】

次に、選択処理は、順序づけられた最適な装置のセットを識別し、それは現在のプロテクショングループについての“最適なプロテクショングループ”である（ステート１７２５）。最適なプロテクショングループは、Ｏとして表現されてもよい。代表的な実施形態において、最適なファイルレイアウトは、初回に好ましい装置のセット全体にファイルを書き込む場合に使用されるレイアウトとして定義される。Ｐにおける最後の装置が使用されるとき、使用される次の装置は、ブロックがＰにおける装置によって“包み込まれる(wrapped around)”ことを可能にするＰにおける最初の装置である。従って、最初のプロテクショングループの最初のブロックは、Ｐにおける最初の装置に格納され、最初のプロテクショングループの２番目のブロックは、Ｐにおける２番目の装置に格納され、最初のプロテクショングループにおける３番目のブロックは、Ｐにおける３番目の装置に格納され、最初のプロテクショングループにおける残りのブロック等について同様である。そして、２番目のプロテクショングループの最初のブロックは、先のプロテクショングループの最後のブロックが格納された装置に格納される。２番目のプロテクショングループの２番目のブロックは次の装置に格納される。等々。この実施形態は、Ｐにおける装置間でプロテクショングループにおける一つのブロックの重複(overlap)を可能とするが、ゼロを含む他の重複サイズが使用されてもよいことが理解される。この実施形態において、“最適

なプロテクショングループ”は、初回に好ましい装置にファイルを書き込む場合に特定のプロテクショングループが格納される装置の順序づけされたセットとして定義され得る。“最適なファイルレイアウト”および“最適なプロテクショングループ”を選択するための他の定義が使用されてもよい。

【０２１２】

次に、選択処理は非最適装置のセットを識別し、それはＮとして表現される（ステート１７３０）。この実施形態において、Ｎは、もし初回にファイルが書き込まれればプロテクショングループが格納されない好ましい装置内の装置のセットである。

【０２１３】

次に、選択処理は、プロテクショングループにおけるブロックのそれぞれが現在格納される装置の配列(array)または行列(matrix)を生成し（ステート１７３５）、それは、Ｃとして参照される。Ｃは、新しいプロテクショングループに存在するブロックと同数の列(column)と、現在格納されるようなファイルによって使用されるミラーの数に対応する各列における行(row)の数とを有し、行の最小数は１とされる。Ｃにおける各列のエントリは、ブロックが既に格納された種々の装置を表す。例えば、もしファイルが現在３ｘファイルとして格納されおり、それが５ｘファイルに変更されていれば、新しいプロテクショングループはサイズ５であり、そして現在格納されるようなファイルによって使用されるミラーの数は３である。従って、Ｃは５列を有し、且つ各列は３行を有する。もしファイルが、現在、３＋１パリティ保護され、そして５＋１パリティプロテクションに変化していれば、Ｃは６列を有し、そして各列は１行を有する。もしファイルがまだシステムに格納されていなければ、Ｃにおける各エントリが、現在システムにブロックが格納されていないことを表すゼロであるように、Ｃは新しいプロテクショングループにおける各ブロックのための列を有すると共に１行を有する。ゼロエントリは、ブロックが現在システムに格納されていないことを表し、例えば、パリティが前に使用されていなければパリティブロックがシステムにまだ格納されていないように、パリティプロテクションがファイルに加えられているときに使用されてもよい。

【０２１４】

そして、選択処理は、このファイルについて何れのプリフェレンスが選択されたかを決定する（ステート１７４０）。代表的な実施形態において、３つのプリフェレンスが存在する。１番目のプリフェレンス（“修復(repair)”）は、データが既に存在するスマート記憶ユニットにブロックを割り当てることによってデータの移動を最小化することを支持する。２番目のプリフェレンス（“再バランス(rebalance)”）は、“最適なリスト”における装置上に既に存在するブロックがその装置に残るようにブロックを割り当てることを支持し、そして他のブロックが“最適なリスト”における残りの装置に移動される。３番目のプリフェレンス（“再チューン(retune)”）は、“最適なリスト”における順序づけられた値に全てのブロックを割り当てることを支持する。プリフェレンスは、選択されたプロテクションスキームに抵触すれば、必ずしも満足されるとは限らないことが理解される。

【０２１５】

上述したプリフェレンスは代表的なものであり、種々のプリフェレンスが使用されてもよいことが理解される。さらにまた、１又は２以上のプリフェレンスがシステムの目標に最も適合するように組み合わされてもよい。加えて、いくつかの一実施形態は、プリフェレンスを使用しなくてもよいが、装置を選択するためにプロテクションスキームの制約を使用することだけはよい。他の実施形態において、プリフェレンスは、プロテクションスキームの制約に優先してもよい。

【０２１６】

もし修復(REPAIR)が上記プリフェレンスであれば、選択処理は、Ｃにおける各列を通してトラバース(traverse)し、列における装置の一つがＯにおける装置の一つと一致するかどうかを調べるためにチェックする（ステート１７４５）。もし一致すれば、選択処理は、最後の割り当てリスティング(final assignment listing)において装置ＩＤを記録する

ことにより、一致した装置をそのブロックに割り当て、Ｏからその装置を削除し、そしてＣにおおける次の列に移動する。もし一致がなければ、選択処理は、Ｃにおける次の列に移動する。一旦、Ｃにおける各列がトラバースされると、選択処理は次のステートに進む。

【０２１７】

このステートでは、割り当てられていない全てのブロックについて、選択処理がＣにおける対応列を通してトラバースし、そして列における装置の一つがＮからの装置の一つと一致するかどうかを調べるためにチェックする（ステート１７５０）。もし一致すれば、選択処理は、最後の割り当てリスティングにおいて装置ＩＤを記録することにより、一致した装置をそのブロックに割り当て、Ｎからその装置を削除し、そして割り当てられていない次のブロックに移動する。もし一致しなければ、選択処理は、割り当てられていない次のブロックに移動する。一旦、割り当てられていないブロックがトラバースされると、選択処理は次のステートに進む。

【０２１８】

このステートでは、割り当てられていない全てのブロックについて、選択処理は、Ｃにおける対応列を通してトラバースし、そして列における装置の一つがＳからの装置の一つと一致するかどうかを調べるためにチェックする（ステート１７５５）。もし一致すれば、選択処理は、最後の割り当てリスティングにおいて装置ＩＤを記録することにより、一致した装置をそのブロックに割り当て、Ｓからその装置を削除し、そして割り当てられていない次のブロックに移動する。もし一致しなければ、選択処理は、割り当てられていない次のブロックに移動する。一旦、割り当てられていない全てのブロックがトラバースされると、選択処理は次のステートに移動する。

【０２１９】

このステートでは、まだ割り当てられていない全てのブロックについて、選択処理はＯからの装置を割り当てる（ステート１７６０）。一旦、各ブロックが割り当てられると、選択処理はエンドステートに進む。

【０２２０】

もし再バランス(REBALANCE)がプリフェレンスであれば、選択処理は、Ｃにおける各列を通してトラバースし、そして列における装置の一つがＯにおける装置の一つと一致するかどうかを調べるためにチェックする（ステート１７６５０）。もし一致すれば、選択処理は、最後の割り当てリスティングにおいて装置ＩＤを記録することにより、一致した装置をそのブロックに割り当て、Ｏからその装置を削除し、そしてＣにおける次の列に移動する。もし一致しなければ、選択処理は、Ｃにおける次の列に移動する。一旦、Ｃにおける各列がトラバースされると、選択処理は次のステートに進む。

【０２２１】

このステートでは、まだ割り当てられていない全てのブロックについて、選択処理は、Ｏからの装置を割り当て（ステート１７７０）、そして最後の割り当てリスティングにおいてその割り当てを記録する。一旦、各ブロックが割り当てられると、選択処理は、エンドステートに進む。

【０２２２】

もし再チューン(RETUNE)がプリフェレンスであれば、選択処理は、最初のブロックから始めて、Ｏにおける最初の装置を割り当て、２番目のブロックは、Ｏにおける２番目の装置に割り当てられ、装置は順序づけられたＯリストを用いて割り当てられる。等々。一旦、各ブロックが割り当てられると、選択処理はエンドステートに進む。

【０２２３】

従って、選択処理は、プロテクショングループにおける各ブロックについて装置を選択する。

【０２２４】

図１７は、選択処理の一実施形態を示し、他の実施形態が使用されてもよいことが理解される。例えば、ファイルを最初に書き出すために使用される方法は異なってもよい。上

述したように、代表的な実施形態は、好ましい装置間にブロックを分散させるために、スキュー(skew)、および一つのブロックによるプロテクショングループ重複(protection group overlap)のようなものを使用する。加えて、システムはスキュー値の使用を排除できる。このようなケースでは、各プロテクショングループは、プロテクショングループへのユニットの割り当ての後またはその度に、同一のユニットを用いてそのデータのその割り当てを開始することができ、好ましいユニットのリストは、種々のパラメータに基づき再順序づけされてもよい。さらにまた、他の実施形態において、選択処理は、Ｃにおける次の列に移動する前に、セットＯ、Ｎ、Ｓのそれぞれを通してトラバースしてもよい。

【０２２５】

加えて、好ましい装置及び最適な装置のセットが、追加の及び／又は他の基準を用いて選択されてもよい。このような基準は、例えば、どのスマート記憶ユニットが動作しているか、スマート記憶ユニットの記憶装置上の利用可能なスペースの量、スマート記憶ユニットのスループットレート、プロテクショングループからのデータが既に格納されたスマート記憶ユニットなどを含んでもよい。異なるシステムが異なる目標を有してもよいことが理解される。

【０２２６】

２．基礎の概要

以下では、上述した数学的構成のいくつかの概要を提供する。

【０２２７】

ａ．オペレータ

以下のものは、図１７に関して説明される代表的な実施形態のための基礎を提供するために使用される演算子である。

＆　＝　交差部(Intersection)

｜　＝　ユニオン(Union)

！　＝　反対(Inverse)

－　＝　減算(Subtraction)

【０２２８】

ｂ．セット／リスト

以下のものは、図１７に関して説明される代表的な実施形態についての基礎を提供するために使用されるセット及び順序づけされたリストである。

Ｔ　＝　インテリジェント分散ファイルシステムにおける利用可能な装置のセット

Ｇ　＝　インテリジェント分散ファイルシステムにおける動作中の装置の全てのセット

Ｗ　＝　ファイルによって占められた装置のセット

Ｓ　＝　予備装置のセット

Ｐ　＝　好ましい装置の順序づけされたセット

Ｏ[size]　＝　プロテクショングループにおける各ブロックへの装置の最適な割り当てであり、“size”はプロテクショングループのサイズである。

Ｎ　＝　非最適セット　＝　Ｐ－Ｏ

Ｆ[size]　＝　プロテクショングループにおける各ブロックについての装置の最後の割り当て

Ｃ[size,k]　＝　各ブロックに割り当てられた現在の装置であり、“ｋ”は現在格納されるようなファイルによって使用されるミラーの数である。

【０２２９】

３．例

装置の選択例が説明される。しかしながら、この例は、本発明の範囲を制限することを意図したものではなく、種々の実施形態についての詳細を提供するものに過ぎない。

【０２３０】

ａ．修復(repair)

我々は、ちょうどノード３上の単一の装置故障にであった４ノード配列上の３ｘファイルを修復することを試みると仮定する。

W == (1,2,3)
G == (1,2,4)
dev(X) 3 ; widthpolicy(x) == 8
R == Repair
group_index = 0

【０２３１】

ここで、|Ｇ|＞＝ｄｅｖ（Ｘ）とすれば、

P = G & W == (1,2)
S = G & !W == (4)

を得る。

【０２３２】

ここで、我々は、Ｓから引いてＰに追加すると、

P = (1,2,4)
S = ( )

となる。

【０２３３】

ｃｙｃｌｅ＿ｏｆｆｓｅｔは、ｇｒｏｕｐ＿ｉｎｄｅｘ％|Ｐ|，又は０％３＝０である。これから、

O = (1,2,4)
N = ( )

を得る。

【０２３４】

Ｆにおける全ての値にわたって繰り返す。

【０２３５】

Ｆ[0]について、Ｃ[0]＆Ｏは（１，２）である。これら二つのミラーは可能な選択だからである。装置１および２の両方はＦ[0]のミラーを備える。Ｆ[0]＝１に選択する。Ｏから１を削除すると、Ｏは（２，４）になる。

【０２３６】

さて、Ｆ[1]について、Ｃ[1]＆Ｏは（２）である。Ｆ[1]＝２に選択する。Ｏから２を削除する。Ｏは（４）となる。

【０２３７】

Ｆ[2]について、Ｃ[2]＆Ｏは空(empty)のセットである。我々は、最後のステップに進み、Ｏの残りの値をＦ[2]に割り当てる。そこでＦ[2]＝４となる。最終的なＦは、

F = (1,2,4)

となる。

【０２３８】

修復の目標が達成された。一つのクラスタのみが再構成されなければならない。他の二つのクラスタは残される。

【０２３９】

ｂ．再バランス(Rebalance)

３ノード配列上に２＋１ファイルが与えられると、再バランス目標は、４番目のノードが配列に加えられるときに３＋１にファイルを再ストライピングするために使用されてもよい。

W == (1,2,3)
G == (1,2,3,4)
dev(X) == 4; widthpolicy(X) == 8
R == Rebalance
group_index = 30

【０２４０】

ここで|Ｇ|＞＝ｄｅｖ（Ｘ）であり、

P = (1,2,3)
S = (4)
を得る。
【０２４１】
Ｓから引いてＰに追加すると、
P = (1,2,3,4)
S = ( )
を得る。
【０２４２】
ｇｒｏｕｐ＿ｉｎｄｅｘは３０であり、ｃｙｃｌｅ＿ｏｆｆｓｅｔは３０％４＝２である、従って、
O = (3,4,1,2)
N = ( )
である。
【０２４３】
ファイルはミラーされず、従って各ソースクラスタのただ一つのデータコピーしか存在しないので、全てのC[i]はせいぜい単一要素を持つことに注意されたい。この例のために、つぎの事柄を仮定する。
C = ((3),(4),(2),(3))
従って、
C[0] = (3), C[1] = (4), 等
【０２４４】
我々は、F[0]から始める。このケースではC[i]＆Oは３を提供し、従ってF[0]＝３である。同様に、F[1]＝４であり、且つF[2]＝２である。各ステップで、我々は、Oが（１）のみを含んで残されるように、Oから削除する。
【０２４５】
F[3]について、C[3]＆Oは空(empty)である。我々は、Oの最後のメンバーをF[3]に割り当てる最終ステップに進み、そして従ってF[3]＝１である。最終的なFは、
F = (3,4,2,1)
である。
【０２４６】
ここで、二つのブロックが移動され、そしてFの全てのメンバーはOにある。
【０２４７】
ｃ．再チューン(Retune)
代表的なインテリジェント分散ファイルシステムは、８つの装置Ａ，Ｂ，Ｃ，Ｄ，Ｅ，Ｆ，Ｇ，Ｈを備え、そしてファイルは、１２個のコンテンツブロックｂ１，ｂ２，ｂ３，ｂ４，ｂ５，ｂ６，ｂ７，ｂ８，ｂ９，ｂ１０，ｂ１１，ｂ１２から構成される。現在、ファイルは装置Ａ，Ｂ，Ｃ，Ｄに格納されており、そして２回ミラーされているが、パリティプロテクションは有していない。以下に、ファイルの各ブロックが現在格納されている装置を示す。
【０２４８】
【表５】

b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12
A	B	C	D	A	B	C	D	A	B	C	D
B	C	D	A	B	C	D	A	B	C	D	A

【０２４９】
もし新しいデータレイアウトが３＋１パリティプロテクションを備えるが、ミラーされたデータを備えていなければ、ファイルは４つのプロテクショングループを有し、それぞ

れは、４つのブロック、即ち３つのコンテンツブロックと１つのパリティブロックを有する。ファイルが分散される装置の最大数ｍａｘは４に設定される。

【０２５０】

従って、

T = {A,B,C,D,E,F,G,H}

W = {A,B,C,D}

である。

【０２５１】

次に、装置Ｆ，Ｇ，Ｈがダウンしたとする。このことは、

G = {A,B,C,D,E}

を意味する。

【０２５２】

Ｐを構築するため、セットWおよびＧの共通集合からの装置がＰに加えられる。もしＰがまだ装置の“ｍａｘ”数を有していなければ、セットＧおよびＷ！の共通集合からの装置が、“ｍａｘ”数に到達するまでＰに加えられる。セットＧおよびＷ！の共通集合における残りの装置はＳに配置される。一実施形態において、ＧおよびＷ！の共通集合における装置は、前部(front)により多くの自由スペースを持つもので順序づけされる。

P = {A,B,C,D}

S = {F}

【０２５３】

上述したスキューを用いて、最適なファイルレイアウトは、次のようになる。

【０２５４】

【表６】

	A	B	C	D	E
パリティグループ１	b1	b2	b3	p1	--
パリティグループ２	b6	p2	--	b4	b5
パリティグループ３	--	b7	b8	b9	p3
パリティグループ４	b11	b12	p4	--	b10

【０２５５】

従って、パリティグループ１：

O = {A,B,C,D}

N = {E}

C = {{A,B},{B,C},{C,D},{0,0}}

【０２５６】

パリティグループ２について：

O = {D,E,A,B}

N = {C}

C = {{D,A},{A,B},{B,C},{0,0}}

【０２５７】

パリティグループ３について：

O = {B,C,D,E}

N = {A}

C = {{C,D],{D,A},{A,B},{0,0}}

【０２５８】

パリティグループ４について：

O = {E,A,B,C}

N = {D}
C = {{B,C},{C,D},{D,A},{0,0}}

【０２５９】

ここで、パリティグループ１は、リターン(RETURN)プリフェレンスを用いて装置を選択するための例として使用される。

0 = {A,B,C,D}

を呼び出す。

【０２６０】

従って、Ｆにおける最初のブロックはＡに割り当てられ、Ｆにおける２番目のブロックはＢに割り当てられ、Ｆにおける３番目のブロックはＣに割り当てられ、そしてＦにおける４番目のブロックはＤに割り当てられる。

【０２６１】

選択処理は、プロテクショングループのそれぞれに呼び出され、つぎの割り当てが得られる。

F_1 = {A,B,C,D}
F_2 = {D,E,A,B}
F_3 = {B,C,D,E}
F_4 = {E,A,B,C}

【０２６２】

従って、ブロックは、ファイルがあれば最初に書き込まれたであろう場所と同じロケーション－上述した最適なファイルレイアウトと同様のレイアウト－に格納される。

【０２６３】

４．擬似コード実例(Sample Pseudocode)

以下に、選択処理の実施例についての擬似コードの例を提供する。しかしながら、この例は、本発明の範囲を制限することを意図したものではなく、特定の実施形態のための詳細を提供するものに過ぎない。

【０２６４】

[著作権２００３　アイシロン・システムズ・インコーポレーテッド]

プロテクショングループについての装置選択の実施

‘＆’は共通集合を取ること(intersect)を意味し、‘｜’はユニオン(union)を意味し、‘！’は反転(inverse)を意味する。

本アルゴリズムは、プロテクショングループについての装置選択のセットであるＦを提供することを試みる。Ｆは配列であり、Ｆ[i]のようにインデックスが付けられる。ＦＥＣ保護された（パリティ保護された）ファイルについて、Ｆは、ファイルの全体を通してオフセット順にプロテクショングループにおけるクラスタのセットのためのデヴィッド(devid)選択を記述する。ミラーされたファイルについては、Ｆは、プロテクショングループにおける全てのミラーのためのデヴィッド(devid)選択を記述する。

本アルゴリズムは多くの入力を考慮する：

group_indexは、プロテクショングループのインデックスである。どのファイルも多くの互いに素(disjoint)のプロテクショングループに分割され、インデックスが割り当てられる。

Ｇは、全ての‘アップ(up)’の装置のセットであり、即ち利用可能であり且つ故障状態に陥っていない装置である。

Ｗは、既にファイルによって占められている装置のセットである。それは、順序づけされたリスト(an ordered list)である。

Ｘは、ファイルがどのようにレイアウトされるかを表すポリシーである。どのＸについても、Ｘによってファイルを保護するために必要とされる装置の数を示す値dev(X)が存在する。これは、また、プロテクショングループにおけるエレメントの数であり、従って、|F|=dev(X)である。Ｘは、また、ファイルの所望の幅widthpolicy(X)を提供し、それは、

もし|G|>=widthpolicy(X)であれば、理想条件下でファイルが占めるべき装置の数である。

Cは、既存のレイアウトを表す。それは、もしあれば、各F[i]に必要とされるデータのコピーが発見される装置のセットを各F[i]について記述する。このような装置のセットをC[i]とする。|C|==|F|である。

Rは、本アルゴリズムがめざす目的物(goal)である。それは、アルゴリズムが、次のようであるかどうか表す。

R＝＝修復(repair)：Fを取得すると、ブロック移動を最小化することを試みる。

R＝＝再バランス(rebalance)：Fを取得しながら|W|==widthpolicy(X)を達成することを試みる。

1．Gが、サブミットされたXに適合するかを検証することから始める。もし、|G|<dev(X)であれば、アルゴリズムは継続できず、エラーを発呼側(caller)に戻し、別のXを用いて再試行することを可能にする。

我々は、好ましい装置Pの順序づけされた配列を確立することを望み、それからFの選択がなされる。次のように、順序づけされたWのサブセットとしてPを確立する。

P == G & W

最初に、自由スペースの量によって順序付けられた、WにないGの全てのメンバーのように、予備配列Sをセットアップする。

S == G & !W

PおよびSは、両方とも順序づけされる。|P| == wpになるまで、SからメンバーをPに追加する：

```
while(|P|<widthpolicy(X) and |S|>0) {
        x=S.pop() // take from front
        P.append(x) // add to end
}
```

|P|は、ファイルによる占有のための装置の理想的なリストを表す。それは、Gが与えられると、widthpolicy(X)に近いサイズとされるべきであるが、もちろん|G|<widthpolicy(X)である。

2．ミラー及びパリティのローテイション(rotation)を可能にするためにプロテクショングループインデックスに基づきサイクルオフセットを計算する。Pにおけるサイクルオフセットから初めて、装置をカウントし、そしてそれらを最適なセットOに加える：

```
cycle_offset=group_index % |P|
for (i=0; i<dev(X); i++)
        O.append(P[(cycle_offset + i) % |P|])
```

Nを、あまり最適でない装置(the less-than-optimal device)として定義する：

```
N = P - O
```

注記：

-N,O,Sは互いに素(disjoint)である

-N|O == P

3．既存の現在の装置にわたって繰り返し、Rが達成されるように各F[i]を選択することを試みる。R == Repairについて、我々は、ブロック移動を最小化したい。R == Rebalanceについて、我々は、Fを0に近づけたい。

R == Repairについて、選択セットの順序づけされた配列(array)を次のように定義する：

```
A = (O, N, S)
```

R == Rebalanceについて、Aを次のように簡単に定義する：

```
A = O
```

RによるAの適切な選択の後、我々は、次のようにFを完結する：

```
for SET in A {
        for (i=0; i<|F|; i++) {
```

```
                x = C[i] & SET       // 既存の装置と所望のものとの共通集合をとる
                if (|x|>0) {
                        y = x.first()          // これらのうちの一つを取得
                        SET = SET - y          // 再考されることを防止
                        F[i] = y       // Fに入れる
                }
        }
}
```

４．我々は、依然として、C[i]からの適切な選択が存在しない任意のF[i]についてFにおけるデヴィッド(devids)を選択しなければならない。

```
for (i=0; i<|F|; i++) {
        if (F[i] unassigned) {
                F = 0.pop();
        }
}
```

【０２６５】

Ｅ．擬似コードの例(Example Pseudocode)

以下では、再ストライピング処理の実施例についての擬似コードの例を提供する。しかしながら、この例は、本発明の範囲を制限することを意図したものではなく、一実施形態の詳細を提供するものに過ぎない。

【０２６６】

＜著作権２００３　アイシロン・システムズ・インコーポレーテッド＞

このセクションでは、（ＢＡＭにおいて）どのようにブロックが装置レベルで割り当てられるかを決定するファイルシステムのコンポーネントを述べる。ディスクおよびブロックレベルでの割り当ては、ＬＢＭ割り当てコードによって処理される。以下に述べられるモジュールは、書き込み及び読み出し処理の間に新しいブロックが割り当てられるべき場所を見つけ出すために、主として、ＢＡＭの書き込み及び読み出しコンポーネントによって使用されるであろう。しかしながら、ＡＰＩは、通常、必要に応じて他のコンポーネントによって十分に使用され得る。

ファンクションはbam_layout.cに存在し、そして最初は新たな書き込み及び読み出しコードによって使用される。ＤＦＭのような他のモジュールは、必要に応じて将来使用するために修正されてもよい：

```
int bam_layout_protection_group
                                (const struct gmp_group_info *gi,
                                long *free_cluster_counts,
                                struct inode *ip,
                                enum layout_goal_t goal,
                                struct protection_level *protection,
                                int width_policy,
                                ifs_lbn_t start_lbn,
                                int num_clusters,
                                int curr_devid_depth,
                                ifs_devid_t **curr_devids,
                                u_int8_t *alloc_counts,
                                ifs_devid_t *result_devids,
                                ifs_baddr_t *result_pbas);
```

＜概要＞

このファンクションは、特定のプロテクション設定下で単一のプロテクショングループのためのレイアウト情報を計算する。このファンクションは、（上記特定のプロテクション設定に基づき）完全なプロテクショングループのために呼び出される。レイアウトリコ

メンデーション(Layout recommendation)は、グループにおける各クラスタのための単一の装置ｉｄの形式で、アウトパラメータ(result_devids)で戻されるであろう。他のアウトパラメータ(result_pbas)は、割り当てにおいて使用される各クラスタについての前のブロックアドレスを備えるであろう。

＜パラメータ＞

Gi：現在のクラスタグループ情報。

free_cluster_counts：グループにおける各ノードについてのフリークラスタカウント(free cluster count)。これは、その現在の自由スペースに基づき、どの装置を使用するかをレイアウトエンジンが決定することを可能にする。この配列は、まさに、長さが‘dset_size(&gi->group)’であるべきである。

Ip：inodeである。これは、主として、width_device情報と前のブロックアドレス情報とをアクセスするために使用される。

Goal：このレイアウト動作の目標を特定するフラグ。このフラグは、ブロックのレイアウトの仕方を決定する時を参照するための明確な目標(objective)をレイアウトエンジンに与える。現在３つの目標(goal)が存在する：

LAYOUT_REPAIR：これは、レイアウトエンジンに、最小のブロック移動が最優先であることを知らせる。レイアウトエンジンは、新たなデヴィッドを、我々のプロテクション制約（ミラーは異なる装置上にあるべきである、など）を維持する必要のある場所にのみ割り当てる。既存のブロックは、絶対必要なときにのみ移動される。これは、装置またはノード故障の後にファイルを修復するときに使用され得る目標である。割り当てられないクラスタが、このシナリオの下に最適に依然として完全に配置されているので、これは、書き込みパス(bam_new_write)によって使用され得る目標である。それは、少なくとも、他の二つの目標と同様に全てのシナリオにおいて高速であり、多くの場合、‘とても(much)’高速である。

LAYOUT_REBALANCE：これは、レイアウトエンジンに、最適な装置がレイアウトのために使用されるべきであることを知らせる。しかしながら、それらの装置は、既存のブロックの移動を避けるためにシャッフルされてもよい。この方法では、システムは、ブロックの移動を極力避けるが、正確な数のノードにわたってファイルをバランスする必要があるときにブロックを移動するであろう。多くの場合、これは、LAYOUT_REPAIRと同様に高速であるが、しかしながら、より最適且つより良好なバランスされたファイルレイアウトをもたらす。この目標は、新たなノードの付加によりプロテクション設定または再バランシングを変更するときに、再ストライパー(restriper)によって使用され得る。

LAYOUT_RETUNE - これは、レイアウトエンジンに、最適なレイアウトが最優先であることを知らせる。既存のブロックには何の注意も払われず、そして任意の又は既存のブロックは、ファイルを完全にレイアウトするために移動されてもよい。もし我々が、相当に良好なレイアウトファイルを手がければ、ブロックは、適切な装置上に図らずも既に存在するので、移動される必要がないが、しかし、これを成し遂げるための試みはなされていない。実際、発呼側は、たとえ、ディスクにおけるフラグメンテーションを修復するために図らずも最適な装置に位置していても、ブロックを再割当てすることを望んでも良い。この目標は、大部分のシナリオにおいて明らかに最も遅く、且つ決して、LAYOUT_REPAIRまたはLAYOUT_REBALANCEよりも速くはない。発呼側は、この目標が、リード性能についての最適化またはデフラグメンテーションのような‘チューニング’動作であることを選択し得る。それらの処理は、通常のファイルシステム動作中にバックグランドまたは必要に応じて稼動してもよいであろう。

Protection：所望のプロテクションの設定。これは、inodeにおけるプロテクション設定と一致しない（例えば、もし発呼側が２ｘでのパリティファイルを考慮していれば、パリティは可能ではない）。レイアウトコードは、ブロックを置く場所を決定するときに、このプロテクション設定を使用する。それは、inode自体におけるプロテクション設定を無視するであろう。

width_policy：ファイルについての所望の幅ポリシー。これは、ファイル全体が配置さ

れるべきノードの目標数である。これは、多くの状況において達成できなくてもよいが、しかしながら、レイアウトエンジンは、このポリシーを満足するように努める。

start_lbn：領域(region)における最初のlbn。このファンクションは操作すべきプロテクショングループの全数を必要とするので、これはプロテクショングループの始まりにある。Start_lbnは、スキューを計算するための容易な手法を提供し、クラスタの周辺のパリティ情報およびデータのローテイションを可能にする。これらのlbnは、また、結果として得られた構造における特定のブロックを識別するために使用される。

num_clusters：配置されるべきプロテクショングループにおけるクラスタの全数。例えば、３＋１プロテクション設定においては、各プロテクショングループは、４つのクラスタを備える一方、３ｘプロテクショングループは３つのクラスタを備える。Num_clustersは、まさにプロテクショングループと一致する。もしそうでなければ、EINVALが戻される。num_clusterパラメータは、３つの配列パラメータ、即ちcurr_devids，alloc_counts，result_devidsの長さを規定することに注意されたい。

curr_devids：各クラスタにおいて最初に割り当てられたブロックのデヴィッド(devid)。これは、まさに長さが‘num_clusters’であるべき２次元配列である。第２の次元は、発呼側が、配列されるべき各クラスタについて代わりの既存のロケーションを提供することを可能にする。このパラメータの更なる詳細な説明については、本文書の‘Curr_devids’のセクションを参照されたい。この配列の多くのエレメントは０に設定されてもよく、このことは、その特定のクラスタには現在ブロックが割り当てられていないことを示す。もしレイアウト制約がそのように行うことを妨げないのであれば、レイアウトエンジンは、既存のブロックが割り当てられた場所に新しいブロックを置くことを試みる。他の配列パラメータと同様にパリティクラスタがこの最後にリストされるべきであることに注意されたい。

alloc_counts：各クラスタについて、発呼側が割り当てようとするブロックの数。これは、レイアウトエンジンが、完全に割り当てられたクラスタをそのままにしておき、且つ、部分的に割り当てられているが現在の装置上で完結されていないクラスタを移動することを可能にする（なぜなら、その装置は、自由スペースがなくなるので）。このパラメータは、１または２以上の装置が容量に近づいているときに参照されるのみである。このパラメータは、まさに、長さが‘rum_clusters’である。

result_devids：このアウトパラメータは、発呼側によって割り当てられ、まさに長さが‘num_clusters’である。それは、プロテクショングループにおける各クラスタについて推奨されたデヴィッド(devid)を備える。その上、将来において、レイアウトエンジンは推奨された特定のドライブに拡大されてもよい。発呼側は、各クラスタについて推奨されたデヴィッドを調べる。もしそのデヴィッドが、そのブロックに既に割り当てられたブロックのデヴィッドに等しくなければ、これらの既存のブロックは、レイアウト制約を満たすために移動される。新たなブロックが、推奨されたデヴィッドに割り当てられる。再び、任意のパリティクラスタは、これと他の配列パラメータの最後にリストされ、そして、ミラーは昇順にリストされる。

result_pbas：このアウトパラメータは、発呼側によって割り当てられ、そして長さが‘num_clusters’のサイズである。それは、各クラスタについての前のブロックアドレス（ｐｂａ）を備え、最適な連続した割り当てについてのブロック割り当て要求と共に送信される。もしそのノード上に既に割り当てられたパリティブロック又は前のデータが存在しなければ、これらｐｂａのいくつかはゼロであってもよい。他の配列パラメータのように、パリティクラスタはこの配列において最下位(last)であり、ミラーは昇順にリストされる。

＜Curr devids フォーマット＞

上述したcurr_devidsの説明は、発呼側が、プロテクショングループにおいて現在割り当てられたブロックのロケーションについてのレイアウトエンジン情報に伝えることを可能にする。レイアウトエンジンは、この情報を参照して、要求された‘目標’を依然として満足しながら、可能な限り少ないブロックを移動するように試みる。

他の配列パラメータのように、curr_devids配列は、長さが‘num_clusters’のサイズ（まさに１つのプロテクショングループのサイズ）である。パリティの場合には、パリティクラスタは、プロテクショングループにおいて最下位にリストされ、そしてミラーは、ミラーインデックスの昇順にリストされる。

このパラメータが他の配列と異なる点は、これが２次元配列であることである。この配列は、新たなプロテクショングループにおいて各クラスタについて１列(column)のデヴィッドを含むであろう（それは、長さが‘num_clusters’である）。各列は、既存のデータブロックのミラーを備える（従って、この配列の行の数は、古いレイアウトのデータミラーカウントにほとんど常に等しい）。このようにして、発呼側は、レイアウトエンジンに、既存のクラスタが装置Ａ，Ｂ，Ｃ上にミラーを有することを知らせることができる。そして、レイアウトエンジンは、もしそれが、そのクラスタについて、Ａ、ＢまたはＣを選択すれば、発呼側が最近割り当てられたどのブロックも移動しないかもしれないことを知る。

少しの例がこのことを更に明確にするであろう。システムが３ｘから３＋１にファイルを再ストライピングすると仮定する。システムは、レイアウトエンジンに、１つの３＋１プロテクショングループ（ファイルにおける最初の３つのクラスタを含む）を配置することを要求する。システムは、３行４列の配列を割り当てることにより、curr_devidsを構成する。４列については、新たなプロテクショングループが４つのクラスタを有するためであり、３行については、各既存のブロックが３つのミラーを有するためである：

```
curr_devids[4][3]：
        [1][2][3][0]
        [2][3][4][0]
        [3][4][5][0]
```

４番目の列は、パリティブロックが現在のところ割り当てられていないので、空(empty)であるが、各データクラスタは３つの既存のミラーを有している。これは、レイアウトエンジンに、それが現在のファイルレイアウトについて知る必要のあることの全てを知らせる。

他の例。システムが３＋１プロテクションから３ｘプロテクションでにファイルを再ストライピングしていると仮定する。システムは、長さが３クラスタ（我々の新たな３ｘプロテクショングループが３のクラスタを有するため）であり、且つ深さが１クラスタ（古いデータはミラーされていないため）であるcurr_devidsリストを作成する：

```
curr_devids[3][1]：
        [1][2][3]
```

このことは、現在のクラスタがミラーされれば余剰の行が必要とされるだけであり、且つ、システムが、既存のミラーの場所をレイアウトエンジンに知らせることを欲することを示している。前述したように、２番目の次元は、ほとんど常に、ファイルの古いデータミラーカウント(old data mirror count)に設定されるであろう。もし、古いファイルがパリティプロテクションであれば、これは１であろう。

この構造の実際の能力(power)は、あるミラー設定から他の設定に再ストライピングするときに現れる。例えば、システムが５ｘから２ｘに再ストライピングしていることを想定する。システムは、２の長さ（我々の新たなプロテクショングループは２クラスタをそなえるため）を有すると共に５の深さ（各既存のデータクラスタは５ミラーを備えるため）を有するcurr_devids配列を生成するであろう。レイアウトエンジンは、既存のミラーのロケーションを知る必要があり、従ってそれはベスト２を選択して保存する：

```
curr_devids[2][5]：
        [1][1]
        [2][2]
        [3][3]
        [4][4]
        [5][5]
```

このケースでは、情報のいくつかは冗長である。なぜなら、システムは、１データクラスタについて語るだけであるから（それは、５回ミラーされる）。しかしながら、この構造は、レイアウトエンジンが容易に理解する。エンジンが気をつけることの全ては、それが最初のミラーについて最初の列の任意のメンバーを選択できることであり、そして、結果として得られるレイアウトは全くデータブロックの移動を必要としないであろう。このフォーマットは、レイアウトエンジンに、それが効率的なレイアウトの決定をすることを必要としているという情報を与える。

大体の場合、通常のファイルを書き込むとき、この配列は極めてシンプルである。余分な複雑性は、再ストライピング、または装置が利用不能のときに回復された書き込みを実施するときに必要とされるのみである。例えば、もし３番目のクラスタを３＋１ファイルに書き込む場合、curr_devids配列は次のように単純になる：

curr_devids[4][1]：
[1][2][0][4]

同様に、３ｘファイルの３番目のクラスタを書き込む場合、配列は次のように単純にできる：

curr_devids[3][1]：
[1][2][0]

これらのケースでは、既存のプロテクション設定は、まさに新たな設定と同一であり、従ってシステムは、フラットな配列で既存のレイアウト情報を伝えることができる。

＜リターン値＞

０：　成功(success)。アウトパラメータは、割り当て又は移動を必要とするブロックについての装置ＩＤを含み、且つ移動の必要のないブロックについての結果は含まない。割り当てを必要とするクラスタは、どれも、実際の割り当てコールで用いる関連の前のブロックアドレス（ｐｂａ）を有するであろう。

ＥＲＯＦＳ：　レイアウト制約に違反することなく、要求されたブロックを配置するための利用可能な装置が十分に存在しない。発呼側は、このエラーをその発呼側に戻すか、または他の所望のプロテクション設定を用いてbam_layout_protection_group()を呼び出すことの何れかを行うことができる。

ＥＩＮＶＡＬ：　無効パラメータ(Invalid parameter)。もし特定された‘num_clusters’が、ひとつの完全なプロテクショングループと厳密に等しくなければ、このエラーは戻されるであろう。デバッグビルド(debug builds)において、また、不良の発呼側を見つけ出すことを手助けすることは、レイアウトモジュールにおけるアサーション故障(assertion failure)を伴う。このエラーコードは、通常の動作では予期されず、且つコーディングエラーを示さない。

＜擬似コード＞

この文書では、ＢＡＭレベルレイアウトモジュールの基本的実施について述べる。

フェーズ１：グループ情報(group info)、幅要件(width requirement)、および装置ごとの自由ブロック統計に基づき、使用するのに十分な自由スペースを有する利用可能な装置のリストが形成される。システムのスペースが尽きるか、または多くの装置がダウンして所望のプロテクションレベルで書き込みできないケースについてここでは理解され、そしてエラーは発呼側に戻されるであろう。

フェーズ２：別のデヴィッドが装置のリストからプロテクショングループにおける各クラスタに割り当てられる。もし特定の装置に割り当てられたブロックが既に存在すれば、システムは、既存のブロックが移動される必要のないようにデヴィッドを割り当てることを試みる。これは、既存のブロックレイアウトおよび現在利用可能な装置によっては、可能であるかもしれないし、可能でないかもしれない。目標のパラメータが、ここでは大きな役割を演じるであろうことに注意されたい。もし目標がLAYOUT_REPAIRであれば、システムはブロックを移動することを避けるが、もしファイルを適切にバランスすることが必要とされていれば、そのようにする。最終に、もし目標がLAYOUT_REBALANCEであれば、既存のブロックがある場所とは無関係に、最適な配置が計算される。

フェーズ３：どのクラスタにもデヴィッドを割り当てると、システムは、割り当て目的で各クラスタにｐｂａを割り当てる。これは、各装置上に割り当てられた最後のブロックのブロックアドレスを与えるstruct inodeにおけるｐｂａリストを参照することにより行われる。ここでの唯一の希薄な点は、システムがｐｂａを使用するためのデータブロックと、パリティｐｂａを使用するためのパリティブロックとを欲することである。それらは、struct inodeに別々に格納され、従ってリストが参照されるべきである。

＜問題＞

１）ディスクスペースの不足：このアルゴリズムひとつの未解決の問題は、いくつかの又は全てのノードが容量に近づいたときにすべきことである。もし可能であれば、システムは、割り当てに関してＥＮＯＳＰＣエラーをもたらすであろう発呼側に装置を推奨することは避けたい。それらのエラーは、もし配列ステートが変更されていなければ、シスコールリスタート(syscall restart)を起動せず、したがってユーザにまで遡って伝搬される。この問題は、我々が実際に装置を結果リストに割り当てるときに、フェーズ２において最もよく取り扱われる。そのときに、我々は、その装置上の利用可能なディスクスペースを調べることができる（我々がアクセスすることができない予約スペースを捕らえることを記憶している）。情報が各シナリオにどのように影響を及ぼすかは次のようである：

LAYOUT_RETUNE：振る舞いに変化がない。このシナリオは、どんなことがあっても最適なレイアウトを選ぶ。もしシステムが最適でない配置を与えれば、チャンスは、それが既に利用可能であることと同然であることである。

LAYOUT_REPAIRE/LAYOUT_REBALANCE：もしシステムが、廃されたcurrent_devidと符合するデヴィッドを割り当てていれば、システムは、自由スペースについて心配する必要はない。システムは、既に、そこで割り当てられたブロックを割り当てており、従ってシステムは、その何れも移動していないか、または少しばかりを割り当てているかのどちらかである。システムは、これらのシナリオでは、可能な限り少ないブロックを移動することを欲し、従ってそれらの装置をそれまでのままにしておく。

システムが空のスロットにデヴィッドをまさに割り当てようとしているときに（そのクラスタについてはcurrent_devidが存在しないことを意味する）、その装置について自由スペースを参照する。もしそれがほとんど満杯であれば、より多くの自由スペースを有する他の好ましい装置を探すことを試み、そしてそれを代わりに使用する。もし好ましい装置の全てがほとんど満杯であれば、好ましい装置と予備装置のリストから最大量の自由スペースを有する装置を割り当てる。

ＥＮＯＳＰＣは、レイアウトエンジンＡＰＩから戻されないことに注意されたい。自由スペース情報は、それらのカウントをリモード装置から読み出すのにある程度の時間を要するので、少しばかり期限を過ぎている。スペースが厳しいとき、システムは、既存のカウントに基づき最も見込みのある装置を推奨する。最悪の場合、発呼側は、割り当てが試みられるときにＥＮＯＳＰＣを取得するであろう。

＜アルゴリズム擬似コード：＞

```
  int
bam_layout_protection_group(gi, free_clusters, ip, goal, protection,
        width_policy, start_lbn, num_clusters, curr_devid_depth, curr_devids,
        alloc_counts, *result_devids, *result_pbas)
{
        /* dinodeから現在幅装置リスト(current width device list)を取得する。*/
        /* リストからｎのダウン装置を削除する。*/
        /* max_width(MAX_WIDTH - n)を計算する。*/
        /* リストに残りの全てのアップ装置を加える。*/
        while (width device list size < UP device list size) {
                /* アップ装置に最も自由なクラスタを加える。*/
        }
        /*
```

```
        * 長さがmax_widthにリストを切り詰める。このことは、dinode width device
        * limitをオーバフローすることを防止する。
        */
       /*
        * width_policy及び所望プロテクションパラメータを用いて、preferred_width
          * 及びmin_widthを計算する。
        */
       /* 我々は利用可能な装置を十分に備えていることを検証する。*/
       if (not)
              return EROFS;
       /*
        * この点で、我々は、おそらく我々が使用することができるオンライン装置の全
         * てのリストを有する。リストの始まりには、このファイルによって現在使用
           * される装置があり、この後には、利用可能な自由スペースによってソート
             * (sort)された残りの全ての装置が続く。
        */
       /*
        * リストを２つの部分に分割する。第１のサブリストはまさに長さにおいて好ま
         * しい幅(preferred_width)であり、我々のマスターリストの先頭から始まる。
          * 第２のサブリストはマスターリストの残りを含む。我々は、第１のリストを
我          *々の‘好ましい装置(preferred_devices)’と呼び、そして、第２のリスト
を                * 我々の‘予備の装置(spare_devices)’と呼ぶ。第２のリストは空
(empt)であ                * ってもよい。
        */
       /* ここで、プロテクショングループのレイアウトを開始する。*/
       /* num_clusters及びalloc_countsの正常チェック(sanity check)。*/
       If (insane)
       /*
        * 我々は、全数のプロテクショングループを必要とし、そして、alloc_counts
              * は０と１６の間である。
       Return EINVAL;
       /* start_lbnを計算する（プロテクショングループにおける最初のｌｂｍ）。*/
       /* start_lbnを用いてgroup_numberを計算する。*/
       /* ダウン(DOWN)であるcurrent_devidsのエレメントをクリアする。*/
       /* current_devidsにおける複製装置(duplicate devices)をクリアする。*/
       /*
        * preferred_deviceにおけるstart_offsetを計算する（恐らく(group_number %
          * list size)にちょうどなる）。これはスキューを受け持つ。
        */
       /* preferred_listにおけるstart_offsetから始めて、我々が割り付ける各クラ
        /* スタについてのリストを通したステップを進め、そして各デヴィッドを新た
な        /* リストにコピーする（必要であれば、好ましいリストの先頭に包み込む）
。こ              /* の新規なサブリストは最適装置リスト(optimal_device list)と
呼ばれる。
        */
       /* 発呼側特定の目標に基づきデヴィッドを割り当てる。*/
       switch (goal) {
              case LAYOUT_REPAIR:
                     /* ブロック移動を避ける。*/
                     error = bam_layout_repair();
```

```
                        break;
                case LAYOUT_REBALANCE:
                        /* バランスされたレイアウトを確保する。*/
                        error = bam_layout_rebalance();
                        break;
                case LAYOUT_RETUNE:
                        /* 最適なレイアウト。*/
                        error = bam_layout_retune();
                        break;
                default:
                        ASSERT(0);
        };
        /*
         * もうすぐ終わり。全クラスタはデヴィッドを割り当てられるべき。
         */
        /* ここで、我々は、前のブロックアドレスを選択しなければならない・・・。*/
        for (each cluster in the protection group) {
                /* ｐｂａをzero_baddrに初期化する。*/
                /* これがパリティであるのかデータクラスタであるのかを決定する。*/
                /* struct inodeから適切なｐｂａリストを取得する。*/
                /* 使用可能なｐｂａを探す。*/
                for (each item in the pba list) {
                        /*
                         * もし、デヴィッドがこのクラスタについての我々の推奨デヴ
                          * ィッドと一致すれば、このｐｂａを使用し、そしてブレーク
                          * する。
                         */
                }
        }
        /* ここで、本当に終わり！*/
}
int
bam_layout_repair()
{
        /* 可能な限り既存のデヴィッドを割り当てる。*/
        /* 最適なリストからのデヴィッドで空(empty)のスロットを満たす。*/
}
int
bam_layout_rebalance()
{
        /* 最適なリストに存在しない既存のデヴィッドはない。*/
        /*
         * もし既存のブロックの移動を最小化することが必要であれば、最適なリストを
          * 再シャッフルし、そして出力に割り当てる。
         */
}
int
bam_layout_retune()
{
        /* 最適なリストを出力に割り当て、何も質問なし。*/
```

【０２６７】
IX.著作権情報
この特許文書の開示の一部は、著作権保護の対象となっている。著作権所有者は、特許商標局の特許ファイルまたは記録にあるので、特許文書または特許開示のいずれによるファクシミリ複製に対しても異議をもたないが、そうでなければ、どんなものであれ、すべての著作権を留保する。
【０２６８】
X.結論
本発明のいくつかの実施形態を説明したが、これらの実施形態は単なる一例を表したものであり、本発明の範囲を制限することを意図したものではない。従って、本発明の広がりと範囲は、添付の特許請求の範囲及びそれらの均等物によって定められるべきである。
【図面の簡単な説明】
【０２６９】
【図１】本発明の一実施形態のハイレベルブロック図である。
【図２】図１に図解される構成要素間でのデータの流れの一例を示す図である。
【図３】スマート記憶ユニットの一例のハイレベルブロック図である。
【図４】ファイルディレクトリの一例を示す図である。
【図５】メタデータ構造の維持を説明するための図である。
【図６Ａ】データロケーションテーブル構造の一実施形態を示す図である。
【図６Ｂ】データロケーションテーブル構造の追加の実施形態を示す図である。
【図６Ｃ】データロケーションテーブル構造の追加の実施形態を示す図である。
【図６Ｄ】データロケーションテーブル構造の追加の実施形態を示す図である。
【図７Ａ】ディレクトリのためのメタデータ構造の一実施形態を示す図である。
【図７Ｂ】ファイルのためのメタデータ構造の一実施形態を示す図である。
【図８Ａ】データロケーションテーブルの一実施形態を示す図である。
【図８Ｂ】データロケーションテーブルの追加の実施形態を示す図である。
【図８Ｃ】データロケーションテーブルの追加の実施形態を示す図である。
【図９】対応するサンプルデータを有するファイルのメタデータの一例を示す図である。
【図１０】ファイルを読み出すためのフローチャートの一実施形態を示す図である。
【図１１】名前解決(name resolution)を実施するためのフローチャートの一実施形態を示す図である。
【図１２】ファイルを読み出すためのフローチャートの一実施形態を示す図である。
【図１３】パリティ情報を生成するためのフローチャートの一実施形態を示す図である。
【図１４】エラー訂正を実施するためのフローチャートの一実施形態を示す図である。
【図１５】インテリジェント分散ファイルシステムにおいてデータを再ストライピングするためのフローチャートの一実施形態を示す図である。
【図１６】スマート記憶ユニットにデータを割り当てるためのフローチャートの一実施形態を示す図である。
【図１７】利用可能な記憶ユニットのセットの中から選択するためのフローチャートの一実施形態を示す図である。
【符号の説明】
【０２７０】
１１０　インテリジェント分散ファイルシステム
１１４　スマート記憶ユニット
１２０　サーバ
１２５　スイッチ
１４０　通信媒体
１４５　インターネット
１３０　ユーザ

【図１】

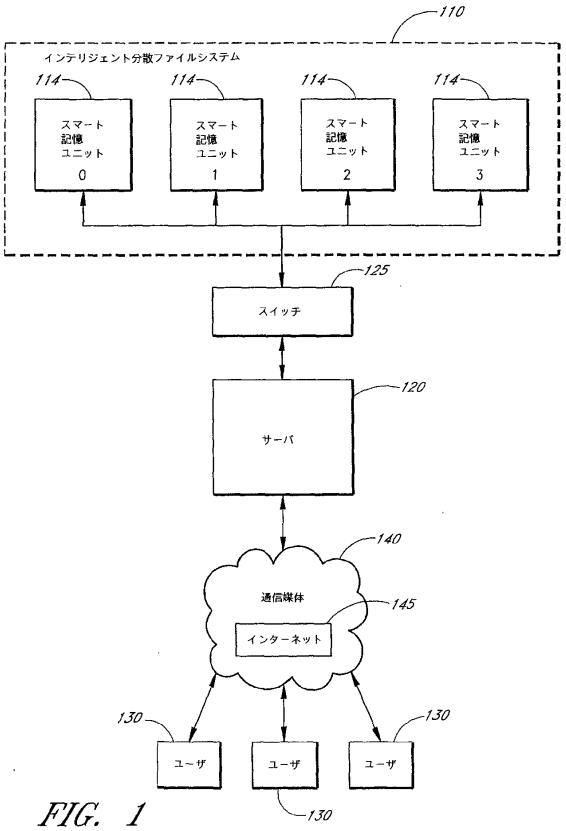


FIG. 1

【図２】

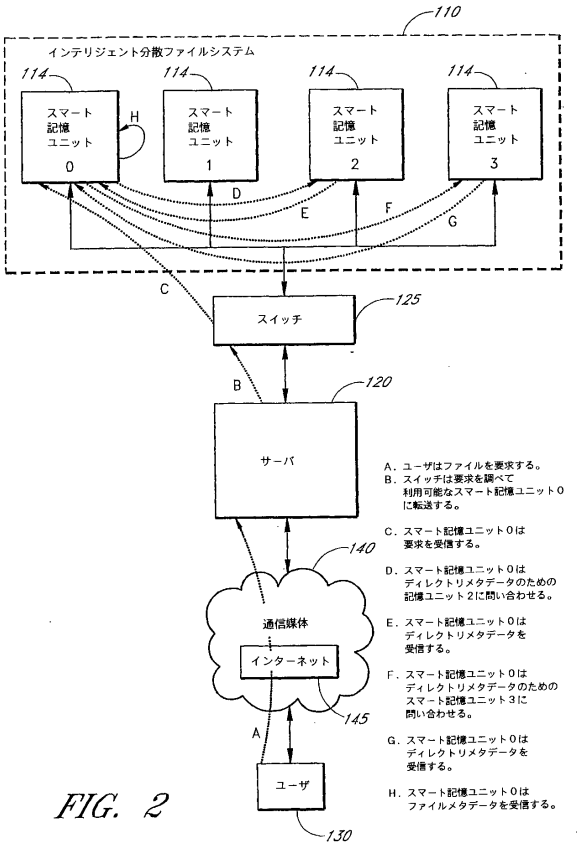


FIG. 2

【図３】

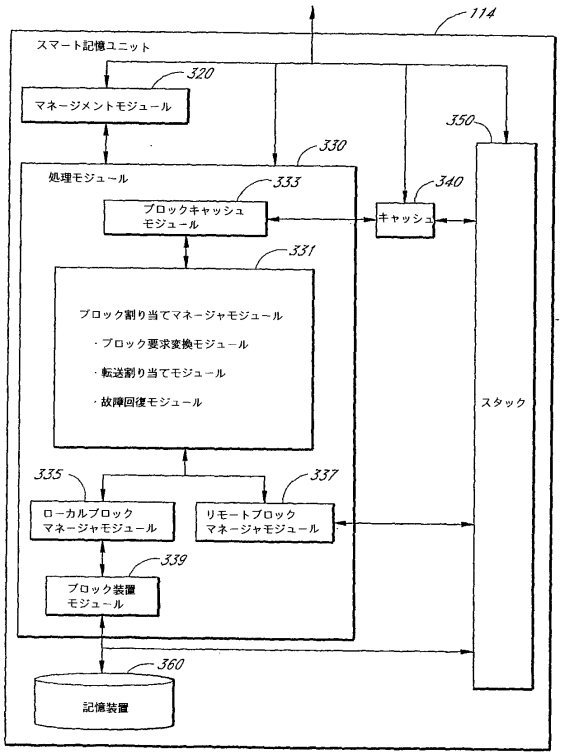


FIG. 3

【図４】

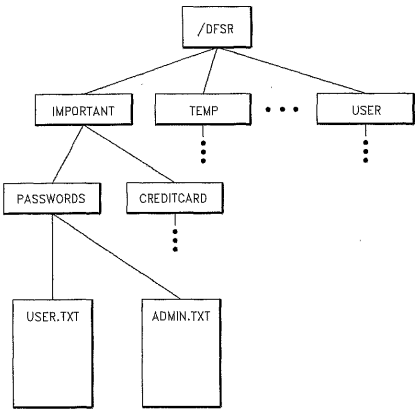


FIG. 4

【図５】

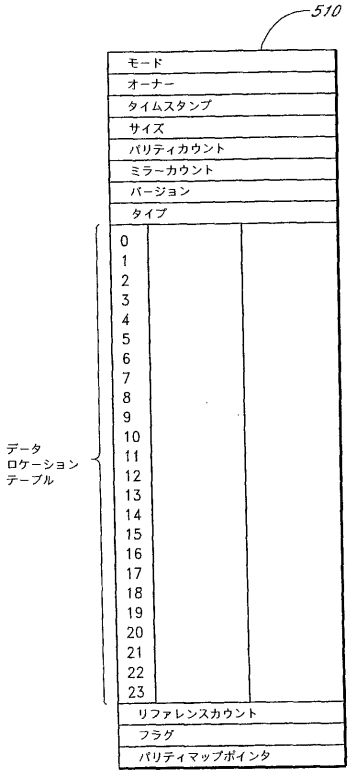


FIG. 5

【図６Ａ】

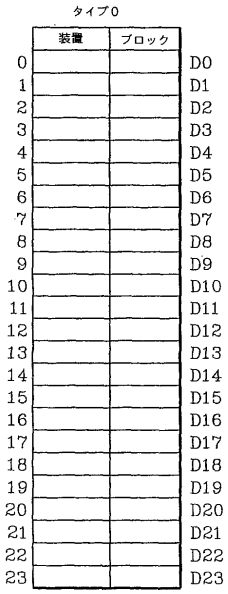


FIG. 6A

【図６Ｂ】

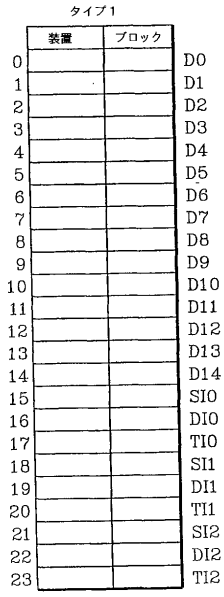


FIG. 6B

【図６Ｃ】

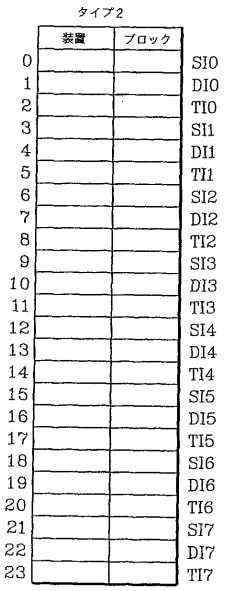


FIG. 6C

【図６Ｄ】

タイプ3

	装置	ブロック	
0			TI0
1			TI1
2			TI2
3			TI3
4			TI4
5			TI5
6			TI6
7			TI7
8			TI8
9			TI9
10			TI10
11			TI11
12			TI12
13			TI13
14			TI14
15			TI15
16			TI16
17			TI17
18			TI18
19			TI19
20			TI20
21			TI21
22			TI22
23			TI23

FIG. 6D

【図７Ａ】

モード	ディレクトリ
オーナー	ルート
タイムスタンプ	65536
サイズ	345
パリティカウント	0
ミラーカウント	3
バージョン	1
タイプ	1
装置1	ブロック11
装置2	ブロック21
装置3	ブロック31
リファレンスカウント	1
フラグ	777
パリティマップポインタ	ゼロ(NULL)

710, 720, 730

FIG. 7A

【図７Ｂ】

モード	レギュラーファイル
オーナー	ルート
タイムスタンプ	65892
サイズ	45897
パリティカウント	3+1
ミラーカウント	1
バージョン	1
タイプ	1
装置1	ブロック11
装置2	ブロック21
装置3	ブロック31
装置2	ブロック100
装置3	ブロック343
装置4	ブロック34
リファレンスカウント	1
フラグ	777
パリティマップポインタ	0x12

装置4	ブロック45
装置1	ブロック87

710, 720, 730, 740

FIG. 7B

【図８Ａ】

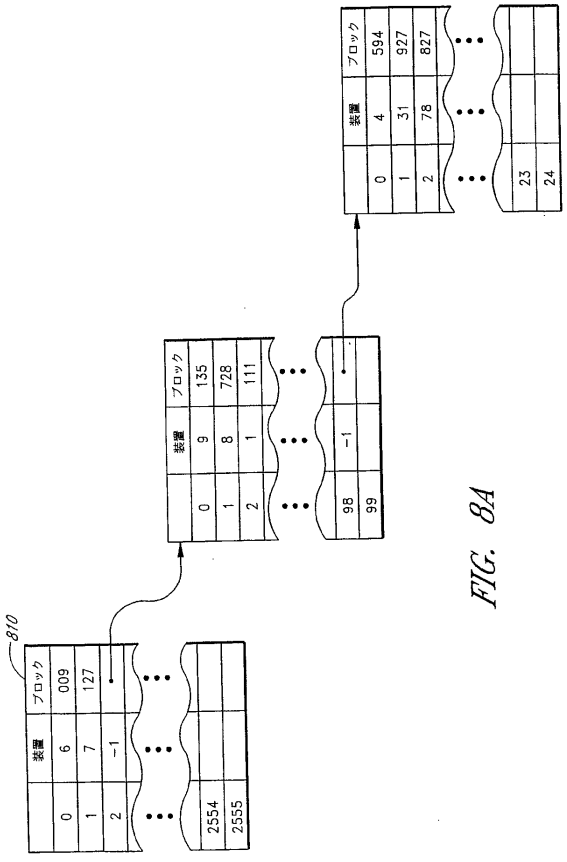


FIG. 8A

【図８Ｂ】

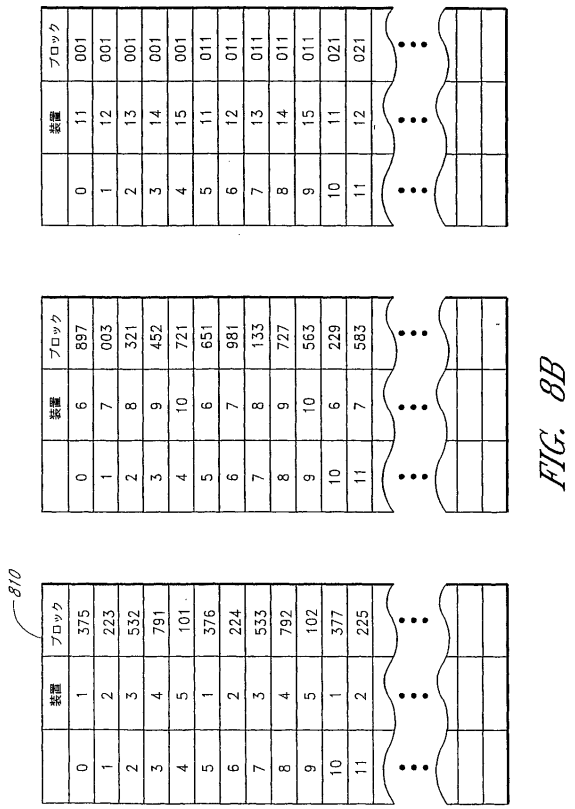


FIG. 8B

【図８Ｃ】

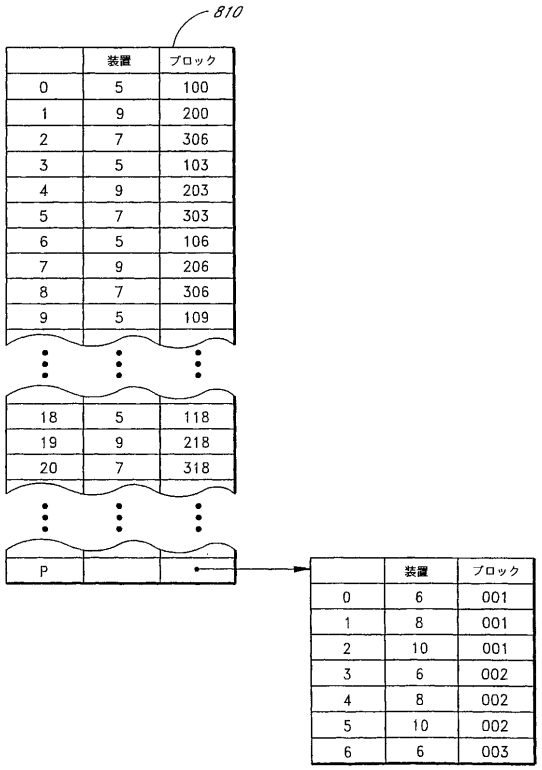


FIG. 8C

【図９】

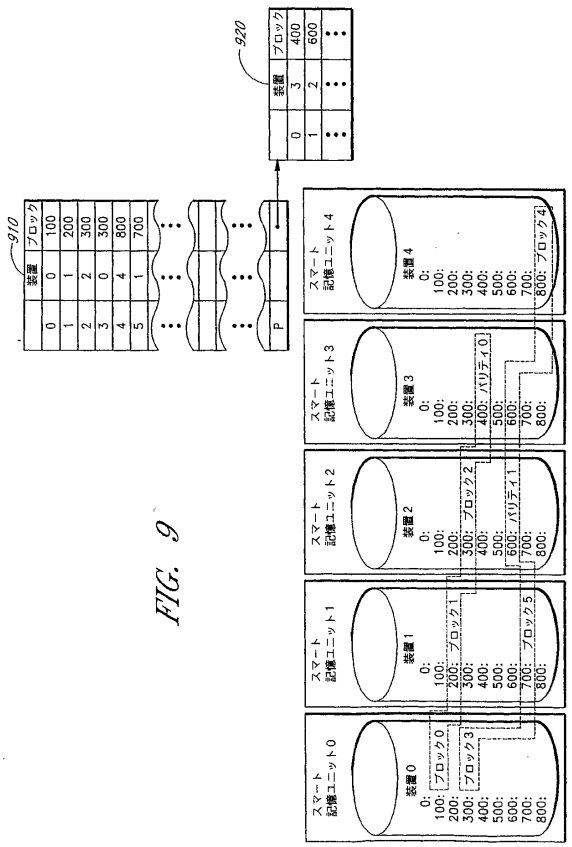


FIG. 9

【図１０】

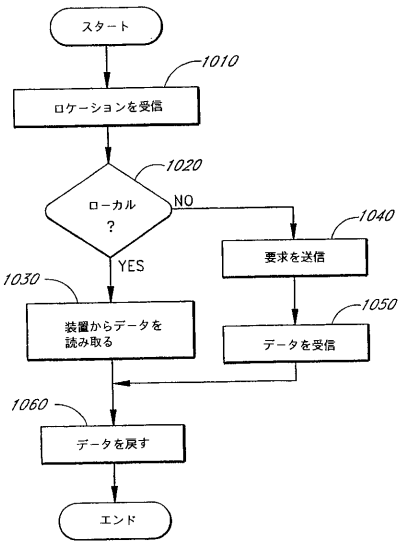


FIG. 10

【図１１】

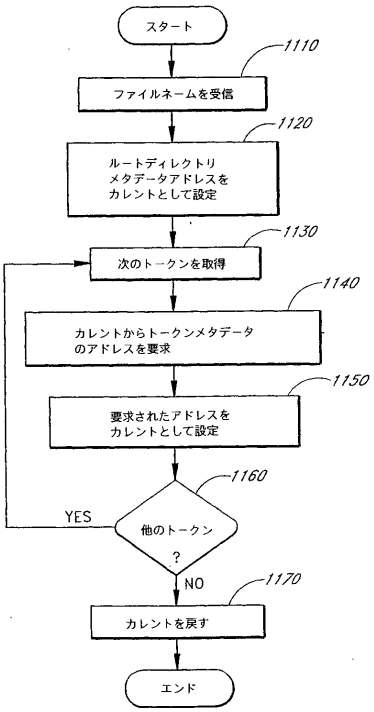


FIG. 11

【図１２】

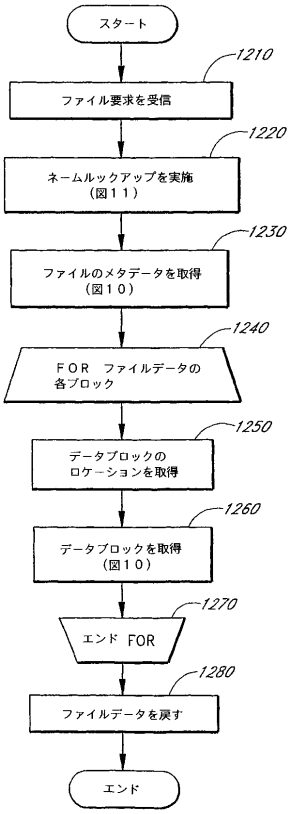


FIG. 12

【図１３】

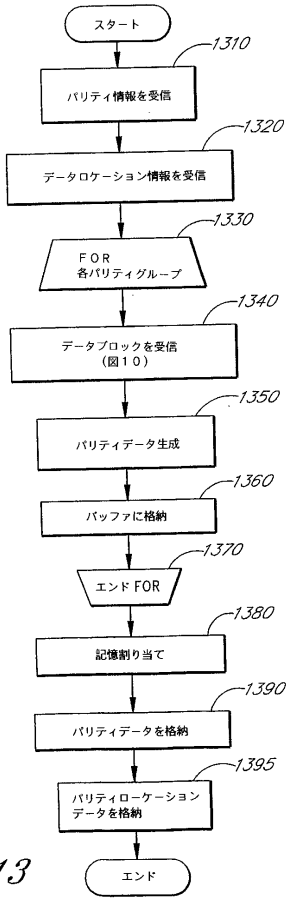


FIG. 13

【図１４】

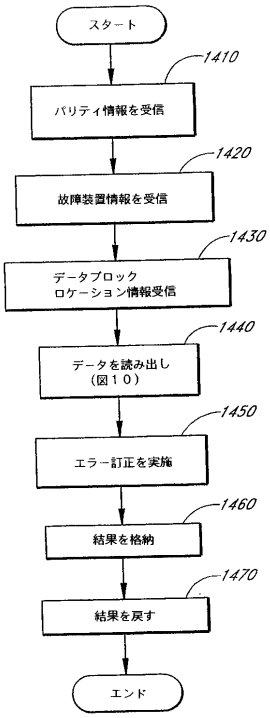


FIG. 14

【図１５】

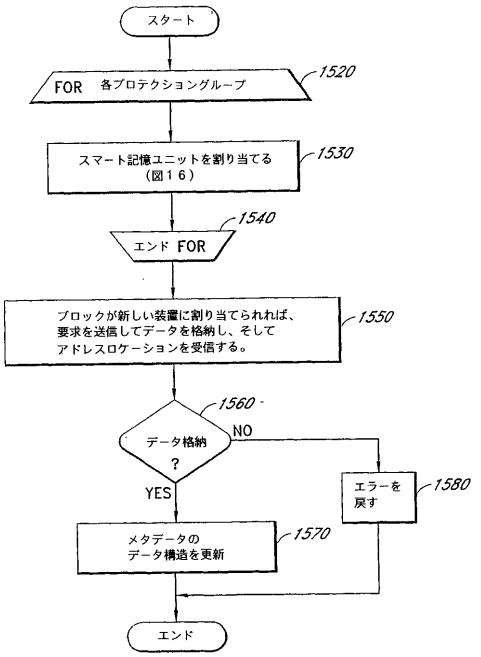


FIG. 15

【図１６】

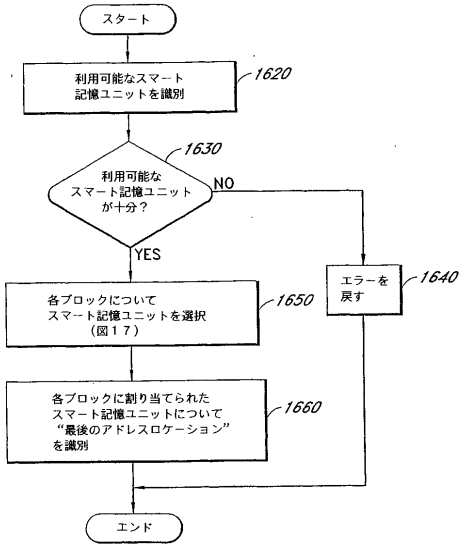


FIG. 16

【図１７】

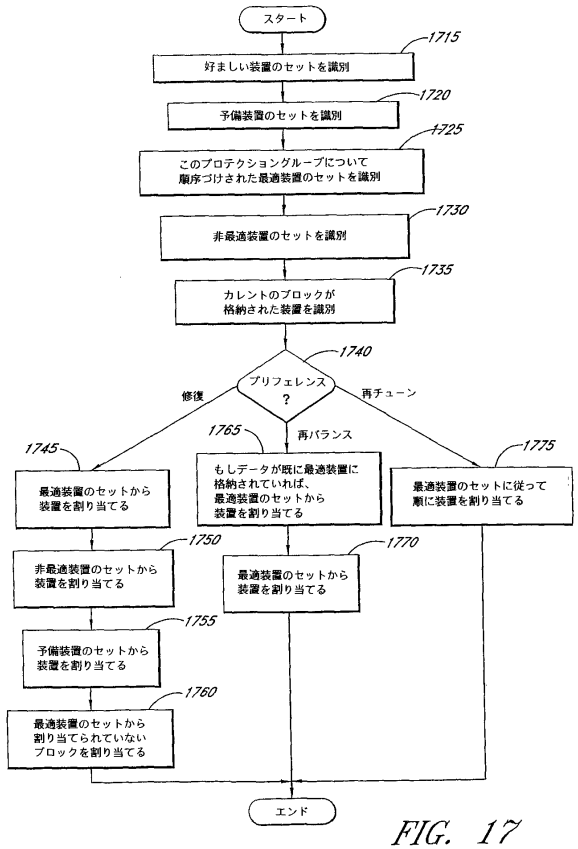


FIG. 17

フロントページの続き

(72)発明者　ポール・エー・マイクセル
アメリカ合衆国・ワシントン・９８１２１・シアトル・フォース・アヴェニュ・＃１７１・２４００
(72)発明者　ロブ・アンダーソン
アメリカ合衆国・ワシントン・９８１０６・シアトル・フォーティーンス・アヴェニュ・ノースイースト・６３３０
(72)発明者　アーロン・ジェイ・パッセイ
アメリカ合衆国・ワシントン・９８１０７・シアトル・セヴェンティーンス・アヴェニュ・ノースイースト・５４０２
(72)発明者　ピーター・ジョン・ゴッドマン
アメリカ合衆国・ワシントン・９８１１６・シアトル・エイキンス・アヴェニュ・サウスウェスト・４２２１
(72)発明者　ハッサン・エフ・カーン
アメリカ合衆国・ワシントン・９８１２１・シアトル・フォース・アヴェニュ・２５１４・アパートメント・８０１
(72)発明者　ダレン・ピー・シャック
アメリカ合衆国・ワシントン・９８１１９・シアトル・ウェスト・レパブリック・ストリート・＃４４０・４１５

審査官　工藤　嘉晃

(56)参考文献　特開２０００－９９２８２（ＪＰ，Ａ）

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