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

特許第5174255号
(P5174255)

(45)発行日　平成25年4月3日(2013.4.3)　(24)登録日　平成25年1月11日(2013.1.11)

(51)Int.Cl.		F I		
G06F 12/00	(2006.01)	G06F	12/00	545B
G06F 13/10	(2006.01)	G06F	13/10	340A
G06F 3/06	(2006.01)	G06F	3/06	306K

請求項の数 14　(全 38 頁)

(21)出願番号　特願2012-40818(P2012-40818)
(22)出願日　平成24年2月28日(2012.2.28)
(62)分割の表示　特願2010-237828(P2010-237828)の分割
原出願日　平成22年10月22日(2010.10.22)
(65)公開番号　特開2012-108955(P2012-108955A)
(43)公開日　平成24年6月7日(2012.6.7)
審査請求日　平成24年3月29日(2012.3.29)

早期審査対象出願

(73)特許権者　391021710
株式会社インテック
富山県富山市牛島新町５番５号
(74)代理人　230104019
弁護士　大野　聖二
(74)代理人　100105038
弁理士　田中　久子
(74)代理人　100131451
弁理士　津田　理
(72)発明者　中川　郁夫
東京都江東区新砂１－３－３　株式会社インテック内

審査官　北村　学

最終頁に続く

(54)【発明の名称】ストレージサービス提供装置、システム、サービス提供方法、及びサービス提供プログラム

(57)【特許請求の範囲】

【請求項１】

複数のストレージ装置とネットワークを介して接続され、これらのストレージ装置を利用してファイルのストレージサービスを提供するストレージサービス提供装置であって、

前記ストレージサービスにおいて、前記ファイルは、複数の領域に分割され、分散して複数のストレージ装置に保存され、

前記複数のストレージ装置の各々は、オブジェクトを保存し、前記ストレージサービス提供装置からの要求に応じて、オブジェクト識別情報により識別されるオブジェクトを返送する手段を有するものであり、

前記複数のストレージ装置に保存される複数のオブジェクトは、前記ファイルの領域を構成するデータを有するブロックオブジェクト、及び、オブジェクト識別情報と該オブジェクト識別情報により識別されるオブジェクトが前記ファイル中のどの領域に対応するものであるかを示すオフセット情報との組を管理情報として有する管理情報オブジェクトを含み、

前記ストレージサービス提供装置は、

読み出すべきファイルに対応する第１のオブジェクト識別情報を求め、該第１のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することにより、該第１のオブジェクト識別情報により識別される管理情報オブジェクトを取得する手段と、

取得された管理情報オブジェクトに含まれている前記管理情報を参照して一つ以上の第

２のオブジェクト識別情報を求め、該第２のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することにより、該第２のオブジェクト識別情報により識別される一つ以上のオブジェクトであってブロックオブジェクトと更なる管理情報オブジェクトとのうち少なくとも一方のオブジェクトを取得する手段と、

取得されたオブジェクトが更なる管理情報オブジェクトである場合に、該更なる管理情報オブジェクトに含まれている前記管理情報を参照して一つ以上の第３のオブジェクト識別情報を求め、該第３のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することを、再帰的に行うことにより、該第３のオブジェクト識別情報により識別される一つ以上のブロックオブジェクトを取得する手段と、

取得された各ブロックオブジェクトが有するデータを、該ブロックオブジェクトが取得されるまでに参照された前記管理情報から求められる前記オフセット情報によって示される領域に配置することにより、前記ファイルを構築する手段とを備えることを特徴とするストレージサービス提供装置。

【請求項２】

少なくとも一つの更なる管理情報オブジェクトを含む少なくとも二つの管理情報オブジェクトが一つのファイルに属し、一つのファイルに属する複数の管理情報オブジェクトが複数のストレージ装置に分散して保存されていることを特徴とする請求項１に記載のストレージサービス提供装置。

【請求項３】

少なくとも二つのブロックオブジェクトが一つの更なる管理情報オブジェクトに属し、一つの更なる管理情報オブジェクトに属する複数のブロックオブジェクトが複数のストレージ装置に分散して保存されていることを特徴とする請求項１又は２に記載のストレージサービス提供装置。

【請求項４】

前記第１のオブジェクト識別情報により識別される管理情報オブジェクトは、二つ以上の第２のオブジェクト識別情報を含み、

前記二つ以上の第２のオブジェクト識別情報のうちある一つに基づいて特定されるストレージ装置に対して、該オブジェクト識別情報により識別されるある一つの更なる管理情報オブジェクトの返送を要求する処理と、前記二つ以上の第２のオブジェクト識別情報のうち別の一つに基づいて特定されるストレージ装置に対して、該オブジェクト識別情報により識別される別の一つの更なる管理情報オブジェクトの返送を要求する処理とを、並行して行う、請求項１～３のいずれか１項に記載のストレージサービス提供装置。

【請求項５】

書き込むべきファイルを複数のデータに分解して、複数のブロックオブジェクトを生成し、各ブロックオブジェクトにオブジェクト識別情報を付与する手段と、

前記ファイルを前記各ブロックオブジェクトのデータから構築するための情報を作成して、階層構造をなす複数の管理情報オブジェクトを生成し、各管理情報オブジェクトにオブジェクト識別情報を付与する手段と、

オブジェクト識別情報に基づいて前記複数のストレージ装置のうちの少なくとも一つを特定する手段と、

生成された前記各ブロックオブジェクト及び前記各管理情報オブジェクトを、それぞれのオブジェクト識別情報に基づいて特定されるそれぞれのストレージ装置へ送信して保存させる手段とを更に備える、請求項１～４のいずれか１項に記載のストレージサービス提供装置。

【請求項６】

前記ファイルに属する前記複数の管理情報オブジェクトのうちのある管理情報オブジェクトと別の管理情報オブジェクトとが、前記複数のストレージ装置のうちの別々のストレージ装置に保存されるように、ストレージ装置を特定するためのオブジェクト識別情報が割り当てられる、請求項５に記載のストレージサービス提供装置。

【請求項７】

前記オブジェクト識別情報は、前記複数のストレージ装置に保存される全ての管理情報オブジェクト及びブロックオブジェクトの間で一意にオブジェクトを識別可能な情報である、請求項１～６のいずれか１項に記載のストレージサービス提供装置。

【請求項８】

前記複数のストレージ装置のうち、ある一部のストレージ装置と他の一部のストレージ装置とが、異なるサービス提供事業者により提供されるストレージサービスに係るものであり、

前記ストレージサービス提供装置がストレージ装置に対して行うオブジェクトの要求は、該ストレージ装置をサーバとし、前記ストレージサービス提供装置をクライアントとして、該ストレージ装置のサービス提供事業者が定めるアクセスプロトコルにて行われるものである、請求項１～７のいずれか１項に記載のストレージサービス提供装置。

【請求項９】

前記オブジェクト識別情報に基づくストレージ装置の特定は、前記オブジェクト識別情報の値に対して所定の計算を行った結果を前記複数のストレージ装置の数で割り算して余りの値を求めることにより行われる、請求項１～８のいずれかに記載のストレージサービス提供装置。

【請求項１０】

前記オブジェクト識別情報に基づくストレージ装置の特定は、前記複数のストレージ装置のそれぞれに該装置の担当する値の範囲を割り振り、前記オブジェクト識別情報の値に対して所定の計算を行った結果と各装置の担当する値の範囲とを比較することにより行われる、請求項１～８のいずれかに記載のストレージサービス提供装置。

【請求項１１】

クライアント装置及び該クライアント装置にネットワークを介して接続された複数のストレージ装置を備え、該クライアント装置がユーザに対してファイルのストレージサービスを提供するシステムであって、

前記ストレージサービスにおいて、前記ファイルは、複数の領域に分割され、分散して複数のストレージ装置に保存され、

前記複数のストレージ装置の各々は、

オブジェクトを保存する手段と、

前記クライアント装置からの要求に応じて、オブジェクト識別情報により識別されるオブジェクトを返送する手段とを備え、

前記複数のストレージ装置に保存される複数のオブジェクトは、前記ファイルの領域を構成するデータを有するブロックオブジェクト、及び、オブジェクト識別情報と該オブジェクト識別情報により識別されるオブジェクトが前記ファイル中のどの領域に対応するものであるかを示すオフセット情報との組を管理情報として有する管理情報オブジェクトを含み、

前記クライアント装置は、

読み出すべきファイルに対応する第１のオブジェクト識別情報を求め、該第１のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することにより、該第１のオブジェクト識別情報により識別される管理情報オブジェクトを取得する手段と、

取得された管理情報オブジェクトに含まれている前記管理情報を参照して一つ以上の第２のオブジェクト識別情報を求め、該第２のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することにより、該第２のオブジェクト識別情報により識別される一つ以上のオブジェクトであってブロックオブジェクトと更なる管理情報オブジェクトとのうち少なくとも一方のオブジェクトを取得する手段と、

取得されたオブジェクトが更なる管理情報オブジェクトである場合に、該更なる管理情報オブジェクトに含まれている前記管理情報を参照して一つ以上の第３のオブジェクト識別情報を求め、該第３のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することを、再帰的に行うことにより、該第３のオブジェクト識

別情報により識別される一つ以上のブロックオブジェクトを取得する手段と、
取得された各ブロックオブジェクトが有するデータを、該ブロックオブジェクトが取得されるまでに参照された前記管理情報から求められる前記オフセット情報によって示される領域に配置することにより、前記ファイルを構築する手段とを備えることを特徴とするシステム。

【請求項１２】
少なくとも一つの更なる管理情報オブジェクトを含む少なくとも二つの管理情報オブジェクトが一つのファイルに属し、一つのファイルに属する複数の管理情報オブジェクトが複数のストレージ装置に分散して保存されており、
前記システムが、複数の前記クライアント装置を備え、
第１及び第２の前記クライアント装置が、それぞれのユーザのために同一のファイルを取得しようとする際、第１の前記クライアント装置が前記複数の管理情報オブジェクトのうちのある一つの返送を要求する処理と、第２の前記クライアント装置が前記複数の管理情報オブジェクトのうちの別の一つの返送を要求する処理とが、並行して行われる、請求項１１に記載のシステム。

【請求項１３】
複数のストレージ装置とネットワークを介して接続されたコンピュータにより、これらのストレージ装置を利用してファイルのストレージサービスを提供する方法であって、
前記ストレージサービスにおいて、前記ファイルは、複数の領域に分割され、分散して複数のストレージ装置に保存され、
前記複数のストレージ装置の各々は、オブジェクトを保存し、前記ストレージサービス提供装置からの要求に応じて、オブジェクト識別情報により識別されるオブジェクトを返送するものであり、
前記複数のストレージ装置に保存される複数のオブジェクトは、前記ファイルの領域を構成するデータを有するブロックオブジェクト、及び、オブジェクト識別情報と該オブジェクト識別情報により識別されるオブジェクトが前記ファイル中のどの領域に対応するものであるかを示すオフセット情報との組を管理情報として有する管理情報オブジェクトを含み、
前記方法は、
読み出すべきファイルに対応する第１のオブジェクト識別情報を求め、該第１のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することにより、該第１のオブジェクト識別情報により識別される管理情報オブジェクトを取得し、
取得された管理情報オブジェクトに含まれている前記管理情報を参照して一つ以上の第２のオブジェクト識別情報を求め、該第２のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することにより、該第２のオブジェクト識別情報により識別される一つ以上のオブジェクトであってブロックオブジェクトと更なる管理情報オブジェクトとのうち少なくとも一方のオブジェクトを取得し、
取得されたオブジェクトが更なる管理情報オブジェクトである場合に、該更なる管理情報オブジェクトに含まれている前記管理情報を参照して一つ以上の第３のオブジェクト識別情報を求め、該第３のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することを、再帰的に行うことにより、該第３のオブジェクト識別情報により識別される一つ以上のブロックオブジェクトを取得し、
取得された各ブロックオブジェクトが有するデータを、該ブロックオブジェクトが取得されるまでに参照された前記管理情報から求められる前記オフセット情報によって示される領域に配置することにより、前記ファイルを構築することを特徴とするサービス提供方法。

【請求項１４】
複数のストレージ装置とネットワークを介して接続されたコンピュータを、これらのストレージ装置を利用してファイルのストレージサービスを提供する装置として動作させる

ためのプログラムであって、
前記ストレージサービスにおいて、前記ファイルは、複数の領域に分割され、分散して複数のストレージ装置に保存され、
前記複数のストレージ装置の各々は、オブジェクトを保存し、前記ストレージサービス提供装置からの要求に応じて、オブジェクト識別情報により識別されるオブジェクトを返送するものであり、
前記複数のストレージ装置に保存される複数のオブジェクトは、前記ファイルの領域を構成するデータを有するブロックオブジェクト、及び、オブジェクト識別情報と該オブジェクト識別情報により識別されるオブジェクトが前記ファイル中のどの領域に対応するものであるかを示すオフセット情報との組を管理情報として有する管理情報オブジェクトを含み、
前記プログラムは、
読み出すべきファイルに対応する第１のオブジェクト識別情報を求め、該第１のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することにより、該第１のオブジェクト識別情報により識別される管理情報オブジェクトを取得するためのプログラムコードと、
取得された管理情報オブジェクトに含まれている前記管理情報を参照して一つ以上の第２のオブジェクト識別情報を求め、該第２のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することにより、該第２のオブジェクト識別情報により識別される一つ以上のオブジェクトであってブロックオブジェクトと更なる管理情報オブジェクトとのうち少なくとも一方のオブジェクトを取得するためのプログラムコードと、
取得されたオブジェクトが更なる管理情報オブジェクトである場合に、該更なる管理情報オブジェクトに含まれている前記管理情報を参照して一つ以上の第３のオブジェクト識別情報を求め、該第３のオブジェクト識別情報に基づいて特定されるストレージ装置に対してオブジェクトを要求することを、再帰的に行うことにより、該第３のオブジェクト識別情報により識別される一つ以上のブロックオブジェクトを取得するためのプログラムコードと、
取得された各ブロックオブジェクトが有するデータを、該ブロックオブジェクトが取得されるまでに参照された前記管理情報から求められる前記オフセット情報によって示される領域に配置することにより、前記ファイルを構築するためのプログラムコードとを備えることを特徴とするサービス提供プログラム。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、多数のコンピュータを用いて大量のファイルを保存することを可能にする分散ストレージ技術に関する。
【背景技術】
【０００２】
従来のコンピュータ利用は、ユーザである企業や個人等が、コンピュータのハードウェア、ソフトウェア、データ等を、自分自身で保有し、管理していたが、近年普及してきたクラウド・コンピューティングでは、ユーザは、自身の手元の機器が接続したネットワークの向こう側（データセンタ等）からサービスを受ける。このようなクラウド・サービスは、クラウド・サービス提供事業者から企業又は個人に対して提供されたり、企業内ネットワークにおいて社員等に対して提供されたりする。
【０００３】
クラウド・サービスのうち、ユーザのデータをネットワーク上のサーバに保存するスト

レージサービスとして、アマゾンＳ３（Ｓｉｍｐｌｅ　Ｓｔｏｒａｇｅ　Ｓｅｒｖｉｃｅ）や、マイクロソフトＷｉｎｄｏｗｓ（登録商標）Ａｚｕｒｅ等が知られている。特に、データを多数のストレージに分散して格納することができ、ＧＢ（ギガバイト）単位の大きなデータと小さなサイズのデータが大量に混在しても、これらを効率よく扱うことのできる分散ファイルシステムとしては、グーグルファイルシステム（ＧＦＳ）が知られている（例えば、非特許文献１を参照）。

【０００４】

ＧＦＳでは、ファイルを６４ＭＢ（メガバイト）のチャンクと呼ばれるブロックに分割して、複数のチャンクサーバに分散して配置することにより、一つのファイルの書き込みや読み出しを複数のサーバで並列に実行して、ファイルの入出力を高速化しており、多数のサーバがあれば、大きなファイルサイズを扱うことも可能である。また、全てのチャンクが３つ以上の複製を異なるサーバに有するように制御する仕組みによって、あるチャンクサーバに障害が発生しても別のチャンクサーバに保存された複製を使うことで耐故障性を高めるとともに、チャンクの複数の複製のうちの選択された一つにアクセスすることで負荷分散を実現することも可能にしている。

【先行技術文献】

【非特許文献】

【０００５】

【非特許文献１】西田　圭介、「Ｇｏｏｇｌｅを支える技術　巨大システムの内側の世界」、株式会社技術評論社、平成２０年８月２５日

【発明の概要】

【発明が解決しようとする課題】

【０００６】

上述したＧＦＳでは、一つのファイルを構成する複数のチャンクのそれぞれ（さらに、各チャンクにつき３つ以上の複製のそれぞれ）がどのチャンクサーバに保存されているかというマッピングを管理するための管理情報が、マスタサーバに保存される。よって、ファイルを読み出すときは、読み出すべきチャンクを特定し、特定されたチャンクを保存しているサーバのアドレスをマスタサーバに問い合せ、返答に示されたアドレスのチャンクサーバにアクセスするという処理が行われ、ファイルを書き込むときは、書き込むべきチャンクを特定し、特定されたチャンクを保存すべきサーバのアドレスをマスタサーバに問い合せ、返答に示されたアドレスのチャンクサーバにアクセスするという処理が行われる。障害発生時のチャンクサーバの切り替えや複製の再作成、チャンクサーバへのアクセスの負荷分散や複製の追加作成等も、管理情報を有するマスタサーバが全て指示することにより行われる。

【０００７】

このような機構では、管理情報を有するマスタサーバが、分散ストレージシステムにおける単一障害点となり、マスタサーバに障害が発生した場合は、システム全体が動作しなくなってしまうという問題がある。また、マスタサーバに負荷が集中するため、そこがボトルネックとなって、スケーラビリティや性能上の限界が出てきてしまうという問題もある。

【０００８】

ＧＦＳの場合は、マスタサーバの冗長化を行うための特別な仕組みを別途有しており、マスタサーバの障害時には、一定の操作を経てバックアップのサーバにその機能を引き継ぐ処理が行われるようにして、単一障害点を意識させないようにしているが、これは、ＧＦＳが、インターネットにおけるＷｅｂページの蓄積及び検索サービスという用途に特化しているからできることであると考えられる。分散ストレージ技術には、広く様々な用途があり、単一障害点のないシステムを実現して耐故障性をさらに向上させることが望ましい。

【０００９】

また、ＧＦＳでは、チャンクが固定長であることもあり、ファイルのサイズ（ファイル

のデータの長さ）が巨大になれば、各チャンクと各サーバとのマッピングを示す管理情報のサイズも膨大になる。そうすると、その大きな管理情報の中から、部分的にアクセスしたいチャンクを、高速に探し出すことが難しくなるため、ランダムアクセスに弱いという問題もある。

【００１０】

さらに、ＧＦＳにしろ、アマゾンＳ３にしろ、既存のストレージサービスでは、サービスを提供するための設備の全てを、一つのサービス提供事業者が管理し運用しなければならない。ストレージサービスの利用者からみれば、一つのサービス提供事業者を選択してサービスを受けるしかなく、ストレージサービスを利用して行いたい処理の全体が、選択した事業者の信頼性やサービスの質に依存することになってしまう。

【００１１】

本発明は、上記の事情に鑑み、多数のコンピュータを用いて様々なサイズの大量のファイルを保存可能にするとともに、システムにおいて単一障害点となる要素を低減することが可能な分散ストレージ技術を提供することを目的とする。本発明はまた、この分散ストレージ技術において、ファイルに対するアクセスの高速化、ランダムアクセス等の高性能化を可能にすることや、複数の事業者から提供されるストレージサービスを利用して一つのストレージサービスを構成可能にすることを目的とする。

【課題を解決するための手段】

【００１２】

本発明の原理に従う一つのストレージサービス提供装置は、複数のストレージ装置とネットワークを介して接続され、これらのストレージ装置を利用してファイルを保存するサービスを提供する。そして、書き込むべきファイルを一つ以上のデータに分解し、該ファイルを構成するデータをブロックオブジェクトとして、各ブロックオブジェクトにオブジェクト識別情報を付与する手段と、前記各ブロックオブジェクトのデータから前記ファイルを構築するための情報を作成し、該情報を管理情報オブジェクトとして、管理情報オブジェクトにオブジェクト識別情報を付与する手段と、オブジェクト識別情報に基づいて前記複数のストレージ装置のうちの少なくとも一つを特定する手段と、前記各ブロックオブジェクト及び前記管理情報オブジェクトを、それぞれのオブジェクト識別情報に基づいて特定されるそれぞれのストレージ装置へ送信して保存させる手段とを備える。

【００１３】

この構成により、複数のストレージ装置を利用した仮想ストレージが実現され、ストレージサービスが提供可能になる。この構成では、ファイルを構成するデータ（ブロックオブジェクト）だけでなく、分解されたデータからファイルを構築するための情報（管理情報オブジェクト）も、いずれもオブジェクトとして、複数のストレージ装置に分散して保存されるため、本ストレージサービス提供装置が、多数のファイルの全てのブロックについての管理情報を集中して保存することはなく、システムにおいて単一障害点となる要素を低減することが可能となる。

【００１４】

なお、本ストレージサービス提供装置は、例えば、クラウド・サービス提供事業者が、自身が管理する複数のストレージ装置（ストレージサーバでもストレージデバイスでもよい）もしくは他のサービス提供事業者が管理する複数のストレージ装置（ストレージサービスとして認識されてもよい）のフロントエンドに設置して、自身のサービスの利用者であるエンドユーザからのネットワークを介した要求に応じてファイルの読み書きを行うものとしてもよい。別の例として、本ストレージサービス提供装置を、企業内ネットワークの内部に設置し、同一企業内ネットワークにある複数のストレージ装置を利用してファイルの読み書きを行うものとしてもよいし、企業内ネットワークに設置した本ストレージサービス提供装置を、企業外のストレージサービス提供事業者のデータセンタ等に接続し、企業外の複数のストレージ装置を利用するファイルの読み書きを行うものとしてもよい。

【００１５】

上記のストレージサービス提供装置が、読み出すべきファイルに対応する先頭オブジェ

クト識別情報を求め、当該先頭オブジェクト識別情報に基づいて特定されるストレージ装置にアクセスして、前記管理情報オブジェクトを取得する手段と、取得された管理情報オブジェクトに含まれている前記ファイルを構築するための情報を用いて、前記ファイルを構成するデータを有するブロックオブジェクトのオブジェクト識別情報を求め、当該オブジェクト識別情報に基づいて特定されるストレージ装置にアクセスして、前記ブロックオブジェクトを取得する手段と、取得されたブロックオブジェクトが有するデータを、前記ファイルを構築するための情報に従って並べることにより、前記ファイルを取得する手段とをさらに備えるようにしてもよい。

【００１６】

これにより、複数のストレージ装置に分散して保存された管理情報オブジェクト及びブロックオブジェクトにアクセスして、元のファイルを取得することが可能になる。

【００１７】

上記のストレージサービス提供装置において、あるファイルに対応する管理情報オブジェクトと、別のファイルに対応する管理情報オブジェクトとが、前記複数のストレージ装置のうちの別々のストレージ装置に保存されるように、ストレージ装置を特定するためのオブジェクト識別情報が割り当てられるようにしてもよい。

【００１８】

これにより、オブジェクト識別情報の割り当てによって、管理情報を複数のストレージ装置に分散して保存することが可能になり、単一障害点となる要素の低減を効率的に実現することが可能となる。

【００１９】

上記のストレージサービス提供装置における前記オブジェクト識別情報に基づいて前記複数のストレージ装置のうちの少なくとも一つを特定する手段を、特定されたストレージ装置に適合するアクセス方法をも特定するものとし、前記特定されたアクセス方法に従い、前記特定されたストレージ装置に対して、前記オブジェクト識別情報の付与されたオブジェクトの保存又は取得を要求することにより、前記ネットワークを介して接続されたストレージ装置の利用が行われるようにしてもよい。

【００２０】

これにより、異なるアクセス方法が採用されているストレージ装置を混在させて、仮想ストレージを実現することが可能になる。例えば、多数の物理デバイスをストレージ装置と見立てて仮想ストレージを実現したい場合、ｉＳＣＳＩ、ＳＡＮ等の種々の方式があり得るが、これらの物理デバイスが混在しても、各物理デバイスに適合した方式でアクセスすることが可能になる。

【００２１】

また別の例として、ネットワーク上の多数のサーバをストレージ装置と見立てて仮想ストレージを実現したい場合は、各サーバへのアクセス方法としてｈｔｔｐ、ｎｆｓ、ｆｔｐ、ｃｉｆｓ、ｒｐｃ等の種々のプロトコルがあり得るが、これらのサーバが混在しても、各サーバに適合したプロトコルでアクセスすることが可能になる。

【００２２】

この場合、上記のストレージサービス提供装置と接続される前記複数のストレージ装置は、それぞれ、前記ストレージサービス提供装置をクライアントとし、任意のアクセスプロトコルにて動作することが可能な、ストレージサーバとすることができる。

【００２３】

さらに別の例として、クラウド・サービス提供事業者等が提供する様々なサービスをストレージ装置と見立てて仮想ストレージを実現したい場合は、各サービスへのアクセス方法として、Ｗｅｂサービスや、ＨＴＴＰを用いた独自プロトコル、あるいは、ＮＦＳ等の種々の体系があり得るが、これらのサービスが混在しても、各サービスに適合した体系でアクセスすることが可能になる。

【００２４】

この場合、上記のストレージサービス提供装置と接続される前記複数のストレージ装置

のうち、ある一部のストレージ装置と他の一部のストレージ装置とが、異なるサービス提供事業者により提供されるストレージサービスに係るものとすることができる。

【００２５】

これにより、複数の事業者から提供されるストレージサービスを利用して一つのストレージサービスを構成することが可能になる。

【００２６】

以上のように構成すると、本ストレージサービス提供装置が実現する仮想ストレージによるストレージサービスの利用者から見れば、複数の異なる物理デバイスや、複数の異なるサーバ、複数の異なるサービス等が混在して構成される仮想ストレージであっても、その内部構造はユーザインタフェースから隠蔽されるため、エンドユーザに対しては透過的に一つのファイルシステムに見せることが可能になる。

【００２７】

上記のストレージサービス提供装置が、前記複数のストレージ装置を利用して保存されるファイルに対する書き込み要求又は読み出し要求を、ユーザ端末から受信する手段をさらに備え、前記書き込み要求又は読み出し要求を、汎用のファイルシステムで用いられる形式の要求としてもよい。

【００２８】

これにより、本ストレージサービス提供装置が提供するストレージサービスの利用者のコンピュータ（ユーザ端末）に対しては、本ストレージサービス提供装置が実現する仮想ストレージを、ＮＦＳやｉＳＣＳＩ等のネイティブ・ファイルシステム（標準的に用いられている汎用ファイルシステム）として見せることが可能になる。

【００２９】

上記のストレージサービス提供装置において、前記管理情報オブジェクトを、前記ファイルのそれぞれ別の部分を構成するデータを有する複数のブロックオブジェクトのオブジェクト識別情報と、各ブロックオブジェクトのデータが前記ファイル中のどの部分に並ぶべきかを示すオフセット情報とを含むものとしてもよい。

【００３０】

このような管理情報を用いることにより、各オブジェクト識別情報に基づいてそれぞれ独立に複数のストレージ装置のいずれかにアクセスして取得したブロックオブジェクトから、オフセット情報に従ってファイルを構築することができるため、ファイルのデータを保存する一連のストレージ装置に順にアクセスするのではなく、複数のストレージ装置を並列でアクセスすることが可能になり、高性能なファイルアクセスが可能になる。

【００３１】

上記のストレージサービス提供装置において、前記管理情報オブジェクトに、前記ファイル内のある領域におけるそれぞれ別の部分を構成するデータを有する複数のブロックオブジェクトのオブジェクト識別情報と、各ブロックオブジェクトのデータが前記ある領域中のどの部分に並ぶべきかを示す領域内オフセット情報とを含む第１の管理情報オブジェクトと、前記第１の管理情報オブジェクトのオブジェクト識別情報と、該第１の管理情報オブジェクトが情報を有する前記ある領域が前記ファイル内でどこに位置するものかを示すファイル内オフセット情報とを含む第２の管理情報オブジェクトとがあるようにしてもよい。

【００３２】

これにより、ファイルに対応する先頭オブジェクト識別情報から、ファイルを構成するデータを有するブロックオブジェクトまでの間に、複数の管理情報オブジェクトを仮想的に配置して、２つ以上の階層を有する再帰構造をとらせることができる。ファイルのサイズが巨大になった場合、階層が一つしかないと、管理情報（本例では、各ブロックオブジェクトのオブジェクト識別情報とオフセット情報のリスト）の量が増加して、管理情報の全体を読み出すのにも、アクセスしたいブロックオブジェクトの情報を検索して選択的に読み出すのにも、時間がかかってしまうが、この膨大な管理情報を複数に分割して複数の管理情報オブジェクト（上記でいう第１の管理情報オブジェクト）を生成し、各管理情報

オブジェクトのオブジェクト識別情報とオフセット情報のリストを新たな管理情報オブジェクト（上記でいう第２の管理情報オブジェクト）を設ければ、これら複数の管理情報オブジェクトも複数のストレージ装置に分散して保存され、並列でアクセスすることが可能になる。

【００３３】

このように、管理情報を複数に分割して、２つ以上の階層を有する再帰構造をとらせることにより、単一障害点となる要素をさらに低減することができるとともに、より高性能なファイルアクセスが可能になる。そうすると、保存可能なファイルのサイズを、デバイスの物理容量や地理的空間の制限を超えて、論理的に上限なく大きくできるというだけでなく、どれだけ大容量のファイルであっても、実用上の問題が生じないようにできるため、真にスケーラブルな仮想ストレージを実現することが可能になる。

【００３４】

上記のストレージサービス提供装置において、前記管理情報オブジェクトを、再帰構造を有する複数の管理情報オブジェクトから構成されることが可能なものとし、前記ブロックオブジェクトの数が所定数より多い場合には、前記再帰構造の階層を増加させて、複数の管理情報オブジェクトを生成するようにしてもよい。

【００３５】

これにより、ブロックオブジェクトの数（ひいてはファイルのサイズ）に応じて、管理情報の再帰構造の階層を増加することができ、さらにスケーラビリティを向上することが可能になる。

【００３６】

上記のストレージサービス提供装置において、前記管理情報オブジェクトが、複数のオブジェクト識別情報を含み、前記複数のオブジェクト識別情報のうちある一つに基づいて特定されるストレージ装置に対して、該ある一つのオブジェクトの保存又は取得を要求する処理と、前記複数のオブジェクト識別情報のうち別の一つに基づいて特定されるストレージ装置に対して、該別の一つのオブジェクトの保存又は取得を要求する処理とを、並行して行うようにしてもよい。

【００３７】

これにより、並列処理が可能になるため、ファイルに対するアクセスを高速化することが可能となる。

【００３８】

上記のストレージサービス提供装置において、保存されているファイルのデータの一部を更新する場合、データが書き換わるブロックオブジェクトに新たなオブジェクト識別情報を付与し、当該ブロックオブジェクトのデータから前記ファイルを構築するための情報を含む管理情報オブジェクトにも新たなオブジェクト識別情報を付与し、前記管理情報オブジェクトの新たなオブジェクト識別情報が前記ファイルに対応する先頭オブジェクト識別情報として求められるように設定することにより、同一のオブジェクト識別情報を有するオブジェクトの内容を不変とする管理を行ってもよい。

【００３９】

これにより、ファイルの内容の更新は、先頭オブジェクト識別情報のアトミックな書き換え（途中の状態が存在しない不可分な書き換え処理）で確定することになり、一旦生成されオブジェクト識別情報が付与された各管理情報オブジェクト及び各ブロックオブジェクトの内容が書き換えられることは一切ないから、ファイルは常に完全な状態で保存されていると見ることができる。つまり、更新後のファイル内容を示す新たなブロックオブジェクトや新たな管理情報オブジェクトが生成されている間であっても、先頭オブジェクト識別情報が書き換わる直前までは、更新前のファイル内容が取得され、先頭オブジェクト識別情報が書き換わった直後から、更新後のファイル内容が取得されるようになる。

【００４０】

したがって、ファイルの書き込み中であっても、同じファイル（更新前の内容）の読み出しが自由にできるし、同一のオブジェクト識別情報を有するオブジェクトの内容は不変

であるから、各管理情報オブジェクト及び各ブロックオブジェクトの複製をそれぞれ独立に作成しても矛盾が起きることがなく、システム内に複製を準備することが容易に実現できる。また、更新前の先頭オブジェクト識別情報とファイルとの対応を履歴として蓄積しておくことにより、各時点のスナップショット（その時点の状態を保存したファイル）を提供することも容易に実現できる。

【００４１】

上記のストレージサービス提供装置におけるオブジェクト識別情報に基づいて前記複数のストレージ装置のうちの少なくとも一つを特定する手段を、二つ以上のストレージ装置を特定可能なものとし、前記各ブロックオブジェクト及び前記管理情報オブジェクトを、複製して、それぞれのオブジェクト識別情報に基づいて特定されるそれぞれの前記二つ以上のストレージ装置へ送信して保存させる手段をさらに備えるようにしてもよい。

【００４２】

これにより、ファイルの書き込み時に、各オブジェクトの複製が作成され、これらの複製が複数のストレージ装置に分散して保存された状態にすることが可能になる。

【００４３】

上記のストレージサービス提供装置が、読み出すべきファイルに対応する管理情報オブジェクト及び各ブロックオブジェクトのそれぞれのオブジェクト識別情報に基づいて、前記複数のストレージ装置のうち該当するオブジェクト又はその複製を保存している二つ以上を特定する手段と、特定された一つのストレージ装置にアクセスして応答がなかった場合に、特定された別のストレージ装置にアクセスしてオブジェクト又はその複製を取得する手段とをさらに備えるようにしてもよい。

【００４４】

これにより、いずれかのストレージ装置に障害が発生しても、そこから取得すべきオブジェクトの複製を他のストレージ装置から取得することができるため、本ストレージサービス提供装置が提供するサービスの利用者は、ストレージサービスを継続利用することができ、耐障害性を向上させることが可能となる。例えば、複数の事業者から提供されるストレージサービスを複数のストレージ装置と見立てて利用する場合、ある事業者のサービスがダウンしても、他の事業者のサービスを利用して、ストレージサービスを継続的に提供することが、自動的にできるようになる。

【００４５】

上記のストレージサービス提供装置が、読み出すべきファイルに対応する管理情報オブジェクト及び各ブロックオブジェクトのそれぞれのオブジェクト識別情報に基づいて、前記複数のストレージ装置のうち該当するオブジェクト又はその複製を保存している二つ以上を特定する手段と、特定された二つ以上のストレージ装置に並行してアクセスし、応答の早かったストレージ装置からオブジェクト又はその複製を取得する手段とさらに備えるようにしてもよい。

【００４６】

これにより、システムの冗長性を活用して、ファイルに対するアクセスをより高速化することが可能になる。

【００４７】

上記のストレージサービス提供装置において、保存されているファイルに対し部分的にデータを書き込む場合、書き込み対象のデータが前記ファイル中のどの部分に並ぶべきものかが指定され、前記ファイルに属する全てのブロックオブジェクト及び管理情報オブジェクトのうち前記指定された部分に関係するものを選択して又は新たにオブジェクトを生成して、この選択又は新たに生成された各オブジェクトのオブジェクト識別情報に基づいてそれぞれ特定されるストレージ装置にアクセスし、残りのオブジェクトについてはストレージ装置へのアクセスを行わないようにしてもよい。

【００４８】

これにより、ファイル中の任意の場所のデータを部分的に書き込むことが可能になり、ランダムアクセスが可能となる。

【００４９】
上記のストレージサービス提供装置において、保存されているファイルからデータを部分的に読み出す場合、読み出し対象のデータが前記ファイル中のどの部分に並んでいるものかが指定され、前記ファイルに属する全てのブロックオブジェクト及び管理情報オブジェクトのうち前記指定された部分に関係するものを選択して、この選択された各オブジェクトのオブジェクト識別情報に基づいてそれぞれ特定されるストレージ装置にアクセスし、残りのオブジェクトについてはストレージ装置へのアクセスを行わないようにしてもよい。
【００５０】
これにより、ファイル中の任意の場所のデータを部分的に読み出すことが可能になり、ランダムアクセスが可能となる。
【００５１】
上記のストレージサービス提供装置において、前記書き込むべきファイルが、ユーザにより元のファイルの全体が暗号化されたものであって、複数のデータに分割され、前記複数のデータのうちあるデータを有するブロックオブジェクトと、別のデータを有するブロックオブジェクトとが、前記複数のストレージ装置のうちの別々のストレージ装置に保存されるように、ストレージ装置を特定するためのオブジェクト識別情報が割り当てられるようにしてもよい。
【００５２】
これにより、ファイルを利用者から預かって保存するサービスを、高いセキュリティで提供することが可能になる。つまり、分割されたデータを暗号化して保存するのではなく、ファイル全体が暗号化されたデータを分割して保存するため、そのようなデータの一部だけを悪意者が取得しても、部分的な復号すら不能にすることができる。特に、複数の事業者から提供されるストレージサービスを複数のストレージ装置と見立てて利用する場合、事業者毎に設備も利用にあたっての認証等も異なることが多いから、一つの事業者のセキュリティが破られても他の事業者までは影響が及ばず、データの全体を悪意者に取得される可能性を極めて低くすることが可能である。
【００５３】
上記のストレージサービス提供装置において、前記書き込むべきファイルが、ユーザにより元のファイルの全体が暗号化されたものであって、複数のデータに分割され、前記複数のデータのうちあるデータを有するブロックオブジェクトと、前記管理情報オブジェクトとが、前記複数のストレージ装置のうちの別々のストレージ装置に保存されるように、ストレージ装置を特定するためのオブジェクト識別情報が割り当てられるものであるようにしてもよい。
【００５４】
これによっても、ファイルを利用者から預かって保存するサービスを、高いセキュリティで提供することが可能になる。悪意者が管理情報オブジェクトを取得できなければ、取得したブロックオブジェクトからファイルを構築することができず、同一ファイルに属する他のブロックオブジェクトがどれかも分からないため、復号不能となるからである。
【００５５】
上記のストレージサービス提供装置において、前記管理情報オブジェクトが、前記ファイルの部分を構成するデータを有するブロックオブジェクトのオブジェクト識別情報と、当該ブロックオブジェクトのデータが前記ファイル中のどの部分に並ぶべきかを示すオフセット情報とを含み、前記ファイル内にデータが存在しない部分がある場合、該データが存在しない部分に対応するブロックオブジェクトは生成せずに、存在するデータを有するブロックオブジェクト及び前記管理情報オブジェクトを保存させ、ファイルを読み出す際に、前記管理情報オブジェクトにより、前記ファイル中のある部分に対応するオブジェクト識別情報が存在しないことが示されると、該ある部分にＮＵＬデータを並べることにより、前記ファイルを取得するようにしてもよい。
【００５６】

これにより、ファイル中のデータのない部分についてはブロックの実体を保存しなくて済むため、ストレージ装置における保存に用いる容量は実際にデータがある分だけとすることができ、スパースファイルを容易に実現することが可能となる。ストレージサービスでは、使った容量に比例して課金されることがあるため、データのない分の容量まで料金を支払わずに済むことは、利用者にとっても好ましい。

【００５７】

このように、ファイルの構造を管理情報オブジェクトだけで表現できれば、例えば、実際に書き込むデータの容量に関わらず巨大なサイズのファイルを作成しておき、後からそのファイル中のデータのない部分にデータを書き込んで、その時に初めてブロックオブジェクトを生成するような使い方も可能になる。

【００５８】

上記のストレージサービス提供装置において、前記管理情報オブジェクトが、前記ファイルの部分を構成するデータを有するブロックオブジェクトのオブジェクト識別情報と、当該ブロックオブジェクトのデータが前記ファイル中のどの部分に並ぶべきかを示すオフセット情報とを含み、前記ファイル内にデータが存在する部分に続けてデータが存在しない部分がある場合、存在するデータと該データの長さの情報とを有するブロックオブジェクト及び前記管理情報オブジェクトを保存させ、ファイルを読み出す際に、前記管理情報オブジェクトが示す前記ファイル中のある部分に並ぶべきデータの長さが、該ある部分に対応するブロックオブジェクトが有する長さの情報が示すデータの長さより長いならば、長さが足りない分ＮＵＬデータを並べることにより、前記ファイルを取得するようにしてもよい。

【００５９】

このように、管理情報オブジェクト中のオフセット情報とブロックオブジェクト中のデータの長さ情報とを用いることによっても、ストレージ装置における保存に用いる容量は実際にデータがある分だけとすることができ、スパースファイル（データが存在しない部分はデータ書き込みを行わないファイル）を実現することが可能になる。

【００６０】

上記のストレージサービス提供装置において、読み出すべきファイルに対応して求められる先頭オブジェクト識別情報が付与された管理情報オブジェクトは、該ファイルの全体の長さの情報と、該長さを有する前記ファイル内のどの部分にどのオブジェクト識別情報が付与されたオブジェクトが配置されるかを示す情報とを含み、前記オブジェクト識別情報が付与されたオブジェクトが再び管理情報オブジェクトである場合には、当該管理情報オブジェクトは、当該オブジェクトが前記ファイル内で配置される領域の長さの情報と、該長さを有する前記領域中のどの部分にどのオブジェクト識別情報が付与されたオブジェクトが配置されるかを示す情報とを含み、前記オブジェクト識別情報が付与されたオブジェクトがブロックオブジェクトである場合には、当該ブロックオブジェクトは、前記ファイルを構成するデータと、該データの長さの情報とを有するようにしてもよい。

【００６１】

これにより、先頭オブジェクト識別情報から管理情報オブジェクトを辿ってブロックオブジェクトにアクセスする過程で、ファイル全体の長さ、ファイル内の各領域の長さ、各領域に入るブロックオブジェクトのデータの長さが分かるため、各ブロックのデータサイズを可変としても、分散させて保存させた複数のブロックから元のファイルを構築することが可能になる。このように、各ブロックのデータを固定長にしなくてよいことを、保存するファイルのサイズを大小様々にするために活用することも可能である。

【００６２】

上記のストレージサービス提供装置において、保存されているファイルのデータの一部を更新する場合、前記ファイルに属するブロックオブジェクト及び管理情報オブジェクトのうち、データが書き換わるブロックオブジェクト及び当該ブロックオブジェクトのオブジェクト識別情報を含む管理情報オブジェクトを、各オブジェクトを保存しているストレージ装置から取得し、取得された各オブジェクトの内容のうち、前記データの書き換えに

よって変更されない部分はそのまま残し、変更される部分にデータを書き込むことにより、新たな各オブジェクトを生成して、該新たな各オブジェクトのオブジェクト識別情報に基づいて特定されるストレージ装置に保存させるようにしてもよい。

【００６３】

これにより、ファイルの書き込みに係るオブジェクトの操作を、ｇｅｔ（取得）とｐｕｔ（保存）という基本命令によって実現することが可能になる。

【００６４】

上記のストレージサービス提供装置における前記オブジェクト識別情報を付与する手段を、前記複数のストレージ装置に保存される全てのブロックオブジェクト及び管理情報オブジェクトの間で一意になるように、新たなオブジェクト識別情報を割り当てるものとしてもよい。

【００６５】

これにより、オブジェクト識別情報として、例えば、ＵＵＩＤ（Ｕｎｉｖｅｒｓａｌ Ｕｎｉｑｕｅ Ｉｄｅｎｔｉｆｉｅｒ）を付与することができ、オブジェクトの内容が更新される毎に新たにオブジェクト識別情報を付与することによって、上述したアトミックな更新、ファイルの書き込み中の読み出し、複製及びスナップショットの提供等も可能になる。

【００６６】

上記のストレージサービス提供装置における前記オブジェクト識別情報に基づいて前記複数のストレージ装置のうちの少なくとも一つを特定する手段が、前記オブジェクト識別情報の値に対して所定の計算を行った結果を前記複数のストレージ装置の数で割り算して得られる余りの値に従って、前記複数のストレージ装置のうちの一つを特定することを含むようにしてもよい。

【００６７】

これにより、例えば、ストレージ装置（物理デバイス、ストレージサーバ、又はストレージサービス）の個数がＳ個であった場合、オブジェクト識別情報から計算される余りの値が０～（Ｓ－１）のいずれになるかによって、そのオブジェクトが保存されるストレージ装置を特定することができる。

【００６８】

このように、各オブジェクトがどのストレージ装置に保存されるかについては、オブジェクト識別情報に対して計算を行うだけで求めることができ、システム内に管理情報を保持する必要がない構成にすると、単一障害点となる要素をさらに低減することが可能になる。例えば、ストレージ装置によってアクセス方法が異なる場合は、ストレージ装置とアクセス方法との対応を示す情報をシステム内に保持することになるが、この情報の量は、ストレージ装置の数の分だけに限られ、上述した管理情報のようにファイルの数及びそれを構成するオブジェクトの数によって爆発的に増加するものではないため、単一障害点となる要素を極めて少なくできる。

【００６９】

別の例として、各ストレージ装置が担当する値の範囲を割り振っておき（ストレージ装置Ｂは３５以上４９未満、ストレージ装置Ｃは４９以上６０未満、…等）、オブジェクト識別情報から計算される値（例えば、ハッシュ値）が、どの範囲に入るかによって、そのオブジェクトが保存されるストレージ装置を特定するようにしてもよい。

【００７０】

この場合、上記のストレージサービス提供装置における前記オブジェクト識別情報に基づいて前記複数のストレージ装置のうちの少なくとも一つを特定する手段が、前記複数のストレージ装置のそれぞれに該装置の担当する値の範囲を割り振り、前記オブジェクト識別情報の値に対して所定の計算を行った結果と各装置の担当する値の範囲とを比較することにより、前記複数のストレージ装置のうちの一つを特定することを含むようにしてもよい。

【００７１】

これによっても、各オブジェクトがどのストレージ装置に保存されるかについては、オブジェクト識別情報に対して計算を行うだけで求めることができるため、単一障害点となる要素を低減することが可能になる。上述した余りの値を使う方法は、ストレージ装置の個数が固定的となるため、静的な方法であるが、こちらの担当範囲の値を使う方法は、担当範囲を動的に変更することができるため、ストレージ装置の動的な追加や削除に対応することができる。

【００７２】

その場合、すなわち、前記ネットワークを介して接続されるストレージ装置が追加又は削除される場合、上記のストレージサービス提供装置において、前記オブジェクト識別情報に基づいて前記複数のストレージ装置のうちの少なくとも一つを特定する手段を、複数のオブジェクト識別情報のうちの一部について追加されたストレージ装置が特定されるように、又は、いずれのオブジェクト識別情報についても削除されたストレージ装置が特定されないように、特定の処理方法を変更するものとしてもよい。

【００７３】

なお、上記のストレージサービス提供装置においては、ファイルと先頭オブジェクト識別情報との対応が、ファイルの数によって増加する管理情報になり得る。このファイルと先頭オブジェクト識別情報との対応をどこに保持するかについては、例えば、次の３つの方法があり得る。先頭オブジェクト識別情報は、例えば、最初にファイルを作成する時には、内容が空であることを示す特別なＩＤがどのファイルについても割り振られ、ファイルの内容が書き込まれると、システム全体で一意なオブジェクトＩＤが付与され、ファイルの内容がその後更新されると、新たに一意なオブジェクトＩＤに書き換えられる。なお、オブジェクトＩＤは、そのオブジェクトがテーブルであるかブロックであるかを示す情報を含んでもよい。

【００７４】

第１の方法は、ストレージサービス提供装置自身によって保持する方法であり、例えば、上述したストレージ装置とアクセス方法との対応を示す情報と共通の管理下で記憶しておくようにしてもよい。この場合、上記のストレージサービス提供装置が、読み出すべきファイルに対応する先頭オブジェクト識別情報を記憶する手段をさらに備え、前記記憶された先頭オブジェクト識別情報に基づいて、前記ファイルについて最初にアクセスするストレージ装置を特定するようにしてもよい。

【００７５】

第２の方法は、複数のストレージ装置のうち、そのファイルのファイル識別情報に基づいて特定されるストレージ装置に保存させる方法である。この場合、上記のストレージサービス提供装置が、読み出すべきファイルの識別情報に基づいて前記複数のストレージ装置のうちの少なくとも一つを特定する手段をさらに備え、ここで特定されるストレージ装置が、読み出すべきファイルに対応する先頭オブジェクト識別情報を、管理情報オブジェクトの一種として保存するようにしてもよい。

【００７６】

第３の方法は、ストレージサービス提供装置に接続されるデータベース等によって保持する方法であり、例えば、各ファイルの属性情報（所有者、作成日時、更新日時、タイトル、パスワード等）を保存するデータベースに、属性情報のうちの一つの要素として、そのファイルの先頭オブジェクト識別情報を記入しておいてもよい。この場合、上記のストレージサービス提供装置が、各ファイルに対応する先頭オブジェクト識別情報をファイルの属性情報とともに記憶する属性管理装置とネットワークを介して接続する手段をさらに備え、読み出すべきファイルに対応する先頭オブジェクト識別情報を前記属性管理装置から取得し、取得した先頭オブジェクト識別情報に基づいて、前記ファイルについて最初にアクセスするストレージ装置を特定するようにしてもよい。

【００７７】

本発明の原理に従う一つのストレージサービス提供システムは、クライアント装置及び該クライアント装置にネットワークを介して接続された複数のストレージ装置を備え、該

クライアント装置がユーザに対してファイルのストレージサービスを提供する。そして、前記複数のストレージ装置は、一つのファイルにつき、それぞれオブジェクト識別情報が付与された複数のブロックオブジェクト及び一つ以上の管理情報オブジェクトを保存する手段を備え、前記複数のブロックオブジェクトのそれぞれは、複数のデータに分解された前記ファイルを構成する各データを有し、前記管理情報オブジェクトは、前記各ブロックオブジェクトのデータから前記ファイルを構築するための情報を有するものであり、前記クライアント装置は、読み出すべきファイルに対応する先頭オブジェクト識別情報を求め、当該先頭オブジェクト識別情報に基づいて特定されるストレージ装置にアクセスして、前記管理情報オブジェクトを取得する手段と、取得された管理情報オブジェクトに含まれている前記ファイルを構築するための情報を用いて、前記ファイルを構成するデータを有するブロックオブジェクトのオブジェクト識別情報を求め、当該オブジェクト識別情報に基づいて特定されるストレージ装置にアクセスして、前記ブロックオブジェクトを取得する手段と、取得されたブロックオブジェクトが有するデータを、前記ファイルを構築するための情報に従って並べることにより、前記ファイルを取得する手段とを備える。

【００７８】

このシステムにおけるクライアント装置は、上述したストレージサービス提供装置のうちのファイルの読み出しに係る機能を有するものであるが、ファイルの書き込みに係る機能を付加したものとしてもよい。

【００７９】

上記のストレージサービス提供システムにおいて、複数の前記クライアント装置を備えるようにし、前記管理情報オブジェクトが、複数のオブジェクト識別情報を含み、前記複数のクライアント装置のそれぞれが、前記読み出すべきファイルに対応する先頭オブジェクト識別情報を求めることができるように設定され、前記先頭オブジェクト識別情報に基づいて前記管理情報オブジェクトの取得を要求する処理及び前記複数のオブジェクト識別情報に基づいて各オブジェクトの取得を要求する処理を、他のクライアント装置から独立して行うようにしてもよい。

【００８０】

これにより、特定のファイルに多数のアクセスが集中するような場合、エンドユーザからのアクセスを受け付けるクライアント装置を複数設けて、フロントエンドでアクセス処理を分散することができ、バックエンドの複数のストレージ装置にファイルが分散して保存されているという利点を、さらに効果的に引き出すことが可能になる。

【００８１】

上述したストレージサービス提供装置の各発明は、ストレージサービス提供システムの発明としても成立し、上述したストレージサービス提供システムの各発明は、そこでクライアント装置として動作するストレージサービス提供装置の発明としても成立するものである。

【００８２】

さらに、上述したストレージサービス提供装置又はシステムの各発明は、ストレージサービス提供装置が行う方法の発明としても、システム全体が行う方法の発明としても、汎用のコンピュータを本ストレージサービス提供装置として動作させるためのプログラム（又はそのプログラムを記録した記録媒体）の発明としても、汎用のコンピュータシステムを本システムとして動作させるためのプログラム（又はそのプログラムを記録した記録媒体）の発明としても、勿論成立するものである。

【００８３】

例えば、本発明の原理に従う一つのストレージサービス提供方法（ファイルの書き込みに係る）は、複数のストレージ装置とネットワークを介して接続されたコンピュータにより、これらのストレージ装置を利用してファイルを保存するサービスを提供する方法であって、書き込むべきファイルを一つ以上のデータに分解し、該ファイルを構成するデータをブロックオブジェクトとして、各ブロックオブジェクトにオブジェクト識別情報を付与し、前記各ブロックオブジェクトのデータから前記ファイルを構築するための情報を作成

し、該情報を管理情報オブジェクトとして、管理情報オブジェクトにオブジェクト識別情報を付与し、前記各ブロックオブジェクト及び前記管理情報オブジェクトを、前記複数のストレージ装置のうち、それぞれのオブジェクト識別情報に基づいて特定されるそれぞれのストレージ装置へ、送信して保存させる。

【００８４】

本発明の原理に従う別のストレージサービス提供方法（ファイルの読み出しに係る）は、複数のストレージ装置とネットワークを介して接続されたコンピュータにより、これらのストレージ装置を利用して保存されているファイルを取得するサービスを提供する方法であって、一つのファイルにつき、それぞれオブジェクト識別情報が付与された複数のブロックオブジェクト及び一つ以上の管理情報オブジェクトが保存されており、前記複数のブロックオブジェクトのそれぞれは、複数のデータに分解された前記ファイルを構成する各データを有し、前記管理情報オブジェクトは、前記各ブロックオブジェクトのデータから前記ファイルを構築するための情報を有するものであり、読み出すべきファイルに対応する先頭オブジェクト識別情報を求め、当該先頭オブジェクト識別情報に基づいて特定されるストレージ装置にアクセスして、前記管理情報オブジェクトを取得し、取得された管理情報オブジェクトに含まれている前記ファイルを構築するための情報を用いて、前記ファイルを構成するデータを有するブロックオブジェクトのオブジェクト識別情報を求め、当該オブジェクト識別情報に基づいて特定されるストレージ装置にアクセスして、前記ブロックオブジェクトを取得し、取得されたブロックオブジェクトが有するデータを、前記ファイルを構築するための情報に従って並べることにより、前記ファイルを取得する。

【００８５】

また例えば、本発明の原理に従う一つのストレージサービス提供プログラム（ファイルの書き込みに係る）は、複数のストレージ装置とネットワークを介して接続されたコンピュータを、これらのストレージ装置を利用してファイルを保存するサービスを提供する装置として動作させるためのプログラムであって、書き込むべきファイルを一つ以上のデータに分解し、該ファイルを構成するデータをブロックオブジェクトとして、各ブロックオブジェクトにオブジェクト識別情報を付与するためのプログラムコードと、前記各ブロックオブジェクトのデータから前記ファイルを構築するための情報を作成し、該情報を管理情報オブジェクトとして、管理情報オブジェクトにオブジェクト識別情報を付与するためのプログラムコードと、前記各ブロックオブジェクト及び前記管理情報オブジェクトを、前記複数のストレージ装置のうち、それぞれのオブジェクト識別情報に基づいて特定されるそれぞれのストレージ装置へ、送信して保存させるためのプログラムコードとを備える。

【００８６】

本発明の原理に従う別のストレージサービス提供プログラム（ファイルの読み出しに係る）は、複数のストレージ装置とネットワークを介して接続されたコンピュータを、これらのストレージ装置を利用して保存されているファイルを取得するサービスを提供する装置として動作させるためのプログラムであって、一つのファイルにつき、それぞれオブジェクト識別情報が付与された複数のブロックオブジェクト及び一つ以上の管理情報オブジェクトが保存されており、前記複数のブロックオブジェクトのそれぞれは、複数のデータに分解された前記ファイルを構成する各データを有し、前記管理情報オブジェクトは、前記各ブロックオブジェクトのデータから前記ファイルを構築するための情報を有するものであり、読み出すべきファイルに対応する先頭オブジェクト識別情報を求め、当該先頭オブジェクト識別情報に基づいて特定されるストレージ装置にアクセスして、前記管理情報オブジェクトを取得するためのプログラムコードと、取得された管理情報オブジェクトに含まれている前記ファイルを構築するための情報を用いて、前記ファイルを構成するデータを有するブロックオブジェクトのオブジェクト識別情報を求め、当該オブジェクト識別情報に基づいて特定されるストレージ装置にアクセスして、前記ブロックオブジェクトを取得するためのプログラムコードと、取得されたブロックオブジェクトが有するデータを、前記ファイルを構築するための情報に従って並べることにより、前記ファイルを取得す

るためのプログラムコードとを備える。
【発明の効果】
【００８７】
以上のとおり、本発明によれば、多数のストレージ装置を利用し様々なサイズの大量のファイルを保存して提供するストレージサービスを、システムにおいて単一障害点となる要素を低減しスケーラビリティを向上させた仮想ストレージとして実現することが可能になる。
【図面の簡単な説明】
【００８８】
【図１】本発明の実施の形態における分散型ストレージサービス提供システム（本システム）の説明図
【図２】本システムのクライアント装置の構成を示すブロック図
【図３】本システムのストレージ装置の構成を示すブロック図
【図４】本システムの機能の説明図
【図５】本システムのファイル構造の概念図
【図６】本システムのファイル構造の概念図
【図７】本システムのファイル構造を説明するための具体例を示す図
【図８】本システムの仕組みの説明図
【図９】本システムにおけるファイル読み出しアクセスの例を示す図
【図１０】本システムにおけるファイル書き込みアクセスの例を示す図
【図１１】本システムにおけるファイル書き込みアクセスの例を示す図
【図１２】本システムにおけるファイル書き込みアクセスの例を示す図
【図１３】本システムにおけるファイル書き込みアクセスの例を示す図
【図１４】本システムにおけるファイル書き込みアクセスの例を示す図
【図１５】本システムにおける静的なサービステーブルの例を示す図
【図１６】本システムにおける動的なサービステーブルの例を示す図
【図１７】本システムにおいて動的なサービステーブルを実現する手法の例を説明するための図
【図１８】本システムにおけるクライアント装置の分散処理の説明図
【図１９】本システムにおけるクラス情報の共有の説明図
【図２０】本システムにおける並列分散処理によるファイル書き込み処理の説明図
【図２１】本システムの特徴の説明図
【発明を実施するための形態】
【００８９】
以下、本発明の実施の形態のシステムについて、図面を用いて説明する。本実施の形態では、例えば、ストレージサービス（ユーザのデータをネットワーク上の多数のストレージ装置に保存するサービス）等のクラウドサービスに用いられる分散型ストレージサービス提供システムの場合を例示する。
【００９０】
まず、本実施の形態の分散型ストレージサービス提供システム（本システム）の構成を、図面を参照して説明する。図１は、本システムの構成を示す説明図である。図１に示すように、本システム１は、エンドユーザが使用するユーザ端末２と、本システム１の利用者（サービス提供者）が使用するクライアント装置３と、クライアント装置３にネットワークを介して接続された複数のストレージ装置４で構成されている。本システムでは、クライアント装置３を介してストレージサービスがエンドユーザに提供される。したがって、クライアント装置３は、ストレージサービス提供装置と呼ぶこともできる。ここで、ユーザ端末２やクライアント装置３は、例えばコンピュータなどであり、ストレージ装置４は、例えば事業者やデータセンターなどに設置されたサーバなどである。
【００９１】
図１（ａ）は、複数の事業者（例えば、事業者Ａ～Ｃ）のストレージサービスを仮想デ

バイスに見立てた場合の説明図である。また、図１（ｂ）は、複数のサーバ（サーバＡ～Ｅ）のストレージ機能を仮想デバイスに見立てた場合の説明図である。図１（ａ）および（ｂ）に示すように、本システムでは、仮想デバイスを多数たばねることにより、論理的に容量が無制限の仮想的なファイルとして扱うことができる。この場合、エンドユーザが使用するユーザ端末からは、ネイティブファイルシステムとして見えるようになる。

【００９２】

図２は、本システムのクライアント装置３の構成を示すブロック図である。図２に示すように、クライアント装置３は、ストレージ装置４との通信を行う通信部５と、ユーザ端末２からの要求に応じてファイルの書き込み処理や読み出し処理を実行する書き込み・読み出し処理部６と、ストレージ装置４へのアクセス方法を定めるためのサービステーブル記憶部７を備えている。

【００９３】

図３は、本システムのストレージ装置４の構成を示すブロック図である。図２に示すように、ストレージ装置４は、クライアント装置３との通信を行う通信部８と、ファイルやオブジェクトが保存されるファイル・オブジェクト保存部９と、保存されているファイルやオブジェクトの管理を行うファイル・オブジェクト管理部１０を備えている。

【００９４】

次に、本システムの機能について説明する。図４は、本システムの機能の説明図である。上述のように、本システムは、エンドユーザが使用するユーザ端末からは、標準的なＮＦＳ（ネットワークファイルシステム）サーバ（例えば、ＲＦＣ１８１３サーバ）などとして使うことができる。そのため、本システムは、ファイルへの操作に関して、以下のような種々の機能を備えている。

【００９５】

まず、図４（ａ）に示すように、本システムは、ファイルを作成するための「CREATE」という機能と、ファイルを削除するための「DELETE」という機能を備えている。「CREATE」の機能では、特にパラメータなどは用いられない。「DELETE」の機能では、「file-id」がパラメータとして用いられる。ここで、「file-id」は、ファイルを識別するための識別情報である。

【００９６】

また、図４（ｂ）に示すように、本システムは、ファイルからデータを読み出すための「READ」という機能と、ファイルへデータを書き込むための「WRITE」という機能を備えている。「READ」の機能では、「file-id」、「offset」、「data」などがパラメータとして用いられる。「WRITE」の機能では、「file-id」、「offset」、「データ領域」などがパラメータとして用いられる。ここで、「offset」は、データがデータ領域内でどの部分に配置されるべきかを示す情報である。「data」は、文字列や数値などのデータの情報であり、データの長さ（length）の情報も含まれる。「データ領域」は、データが書き込まれる領域の情報であり、その領域に書き込むことができるデータの長さ（length）の情報も含まれる。

【００９７】

次に、本システムのファイル構造について説明する。図４および図５は、本システムのファイル構造の概念図である。本システムでは、図４に示すようなデータ構造でファイルが表現される。ユーザ端末からアクセスされるファイルは、「ファイルオブジェクト（単にファイルともいう）」、「テーブルオブジェクト（単にテーブルともいう）」、「ブロックオブジェクト（単にブロックともいう）」により構成される階層構造を有している。この場合、「ファイルオブジェクト」の中に「テーブルオブジェクト」が含まれ、「テーブルオブジェクト」の中に「ブロックオブジェクト」が含まれる。図４に示すように、「テーブルオブジェクト」の中に「テーブルオブジェクト」が含まれてもよい。なお、この概念図は、本システムのファイル構造を抽象的に表現したものである（データの実体やデータの保存場所については後述する）。

【００９８】

この場合、ユーザ端末からのファイルへのアクセスは、利用するプロトコルに依存する。例えば、ファイルシステムでは、ファイルへのアクセスに「パス名」が用いられる。また、ＮＦＳの場合には「パス名」から対応する「file handle」を検索し、それ以降は「file handle」を用いてファイルへのアクセスが行われる。また、ＨＴＴＰの場合には「ＵＲＬ」がファイルへのアクセスに用いられる。

【００９９】

本システムでは、ファイルは「file-id」によって表現される。上述のように「file-id」は、ファイルに対して一意なＩＤ（識別情報）である。本システムでは、ファイルの実体はテーブルであり、テーブルは、ファイルの長さ（length）やデータの配置（offset）などの情報を持っている。なお、上述のように、テーブルは、再帰的に配置が可能なテーブルを表現することができ、テーブルの中にテーブルを配置することが可能である。テーブルは、テーブルを構成する要素のオブジェクト（テーブルまたはブロック）のリスト（<offset,object-id>のリスト）を取得するインターフェースを備えている。本システムでは、ブロックがデータの実体を表現している。ブロックは、実データ（仮想デバイス上に存在するデータ）にアクセスするためのインターフェースを備えている。

【０１００】

図６では、本システムのファイル構造が「木構造」として表現されている。本システムでは、ファイルを表現するためのデータ構造は、再帰構造をとることが可能である。したがって、本システムのファイル構造は、図６に示すように木構造（ツリー構造）として表現することもでき、ブロックは、木構造のリーフに位置することになる。この場合、オブジェクト（テーブルまたはブロック）は、「object-id（table-idまたはblock-id）」と「length」の情報を持つ「箱」として表現することができる。

【０１０１】

図７は、本システムのファイル構造を説明するための具体例を示した図である。図７の例では、ファイルオブジェクト（file-id）により特定されたテーブル（先頭テーブル）は、「table-1」というテーブルＩＤのテーブルであり、「950」という「length」の情報と、「<0,block-20>、<100,table-3>、<600,block-12>、<750,table-10>」というリストの情報を持っている。

【０１０２】

この「table-1」のテーブルを「readAt」すると、オフセット０の位置に配置された１００バイトの長さの「block-20」のデータと、オフセット１００の位置に配置された「table-3」のテーブル（リスト）と、オフセット６００の位置に配置された１５０バイトの長さの「block-12」のデータと、オフセット７５０の位置に配置された「table-10」のテーブル（リスト）が読み出される。なお、「readAt」は、データの読み出しをするための命令（インターフェース）である。なお、上述のように、「オフセット」は、ファイル内での位置（そのファイルの先頭からの位置）を表す情報である。

【０１０３】

「table-3」と「table-10」のテーブル（リスト）は、さらに「readAt」できる。この場合、「table-3」のテーブルは、「500」という「length」の情報と、「<0,block-120>、<300,block-130>」というリストの情報を持っている。したがって、この「table-3」のテーブルを「readAt」すると、領域オフセット０の位置に配置された１００バイトの長さの「block-120」のデータと、領域オフセット３００の位置に配置された２００バイトの長さの「block-130」のデータが読み出される。また、「table-10」のテーブルは、「200」という「length」の情報と、「<0,block-800>、<50,block-900>」というリストの情報を持っている。したがって、この「table-10」のテーブルを「readAt」すると、領域オフセット０の位置に配置された５０バイトの長さの「block-800」のデータと、領域オフセット５０の位置に配置された１５０バイトの長さの「block-900」のデータが読み出される。なお、「領域オフセット」は、そのテーブルが示す領域内でのオフセット（そのテーブルが示す領域の先頭からの相対位置）を表す情報である。

【０１０４】

結果として、「table-1」のテーブルからは、オフセット０の位置に配置された１００バイトの長さの「block-20」のデータと、オフセット１００（領域オフセット０）の位置に配置された１００バイトの長さの「block-120」のデータと、オフセット４００（領域オフセット３００）の位置に配置された２００バイトの長さの「block-130」のデータと、オフセット６００の位置に配置された１５０バイトの長さの「block-12」のデータと、オフセット７５０（領域オフセット０）の位置に配置された５０バイトの長さの「block-800」のデータと、オフセット８００（領域オフセット５０）の位置に配置された１５０バイトの長さの「block-900」のデータが、読み出されることになる。なお、オフセット２００から４００までの間のデータがない領域は「nul」で埋められている。

【０１０５】

次に、図８を参照しながら、本システムの仕組みについて説明する。図８に示すように、本システムでは、テーブルやブロックに関連する情報を、抽象化されたオブジェクトとして扱い、その内容をネットワーク上の仮想デバイス（サーバやサービス）上に持つようにしている。そして、本システムでは、ネットワーク上の仮想デバイス（サーバやサービス）上に必要な情報を分散配置することによって、特定のサーバや特定の領域に管理情報を保存する必要性をなくすようにしている。

【０１０６】

続いて、ファイルに関するデータ構造について説明する。ファイルは、そのファイルを示す一意な「file-id」を持っている。また、ファイルは、そのファイルの内容を表現するためのテーブル（先頭テーブル）の「table-id」に対応付けされている。本システムにおいて、ファイルの内容に関する情報は、先頭テーブルに書き込まれた「table-id」のみであり、詳細な内容は「table-id」から再帰的にデータを取得することにより得られるようになっている。なお、実装上は、「file-id」と先頭テーブルの「table-id」の対応表もネットワーク上にオブジェクトとして保存されていてよい。ファイルの長さ（length）は、先頭テーブルに対して「getLength」して取得できるの長さ（length）と同じである。すなわち、「getLength(ファイルオブジェクト) = getLength(該当ファイルの先頭テーブル)」である。なお、ファイルは、属性情報（所有者、アクセス権限、更新日時）や管理情報を持っていてもよい。

【０１０７】

次に、本システムにおけるオブジェクトの定義について説明する。オブジェクトは、そのオブジェクトを識別するために一意性を有する識別子（object-id）と、データの長さ（length）の情報を持っている。「object-id」としては、例えば、６４ビットの整数、ＵＵＩＤ、任意の文字列などが使用される。また、「length」は、負でない整数であり、例えば６４ビットの整数などで表現される。オブジェクトの「length」は、「getLength」で取得することができる。本システムのオブジェクトには、ブロックとデータが含まれる。

【０１０８】

ブロックは、オブジェクトの一種であり、「block-id」と「length」の情報を持っている。ブロックは、コンテンツと呼ぶこともできる。ブロック（コンテンツ）のデータは、「getContent」で取得することができる。また、ブロックは、「putContent(block-id, content, length)」で「block-id」と「content」と「length」を与えることにより、新規に生成することもできる。なお、ブロックの上書きは不可である。

【０１０９】

テーブルも、オブジェクトの一種であり、「table-id」と「length」の情報を持っている。また、テーブルは、「<offset,object-id>のリスト」を持っている。ここで、「object-id」は、「block-id」または「table-id」である。テーブルのデータは、「getTable」で取得することができる。また、テーブルは、「putTable(table-id,<offset,object-id>)」で「table-id」と「<offset,object-id>」を与えることにより、新規に生成することもできる。なお、テーブルの上書きは不可である。また、<offset0,object0>というエントリについては、「offset0 ≧ 0（「offset」の値が０以上であること）」、「object

0.getLength() ＞ 0（そのオブジェクトの「length」の値が０より大きいこと）」、「offset0 + object0.getlength() ≦ table.getLength()（「offset」の値にそのオブジェクトの「length」の値を加えた値が、テーブルの「length」の値以下であること）」が、すべて成り立つことが条件とされる。

【０１１０】

本システムにおいて、「block-id」は、仮想デバイス上のデータに対応しており、ブロックオブジェクトへのアクセスでは、「サービステーブル（後述する）」を用いて仮想デバイスへのアクセスに必要な情報を求め、「block-id」に対応付けられているデータ（と長さの情報）を取得する。また、本システムにおいて、「table-id」は、仮想デバイス上の情報に対応しており、テーブルオブジェクトへのアクセスでは、ブロックの場合と同様にして、仮想デバイスから情報を取得することができる。ただし、この場合に取得できる情報の中身は、テーブルの情報（すなわち、テーブルが表現するファイル内の領域の長さ（length）と<offset,object-id>のリスト）である。

【０１１１】

上記データのアクセスで利用されるサービステーブルは、「一意の（ユニークな）ＩＤ番号」と「アクセス手段」の情報を含んでいる。事業者が提供するサービスの場合のアクセス手段は、例えば「http://jigyousya.com/storage/%s」などである。また、ネットワークサーバへのアクセス手段は、例えば「10.0.50.11:/users/isi（ＮＦＳの場合）」や「samba://10.0.60.1/public（ＣＩＦＳの場合）」などであり、物理デバイスへのアクセス手段は、例えば「/dev/sd0a」などである。なお、サービステーブルの詳細については後述する。

【０１１２】

ここで、ブロック（コンテンツ）の保存や取得をする場合に用いられる命令（インターフェース）について説明する。ブロックの保存（新規作成）をする場合には「putContent(block-id, length, content)」が用いられる。この場合、「block-id」から仮想デバイスが特定され、その「block-id」に対応するデータを保存する。このとき、「ＨＴＴＰ ＰＵＴ」や「ＮＦＳ ＷＲＩＴＥ」など、仮想デバイスに応じたデータ保存の仕組みを利用してもよい。なお、上述のとおり、本システムでは、ブロックの新規保存のみ可能であり、ブロックの上書きは不可である。

【０１１３】

ブロックの取得（データの取得）をする場合には「getContent(block-id)」が用いられる。この場合、「block-id」から仮想デバイスが特定され、その「block-id」に対応するデータを取得する。このとき、「ＨＴＴＰ ＧＥＴ」や「ＮＦＳ ＲＥＡＤ」など、仮想デバイスに応じたデータ保存の仕組みを利用してもよい。なお、その「block-id」に対応するデータがない場合には「エラー」となる。

【０１１４】

なお、上記の説明では、ブロックの保存や取得のためのインターフェースのみを定義したが、仮想デバイス（サービスやサーバなど）によって、実際の通信手法は、「block-id」に対応するデータが保存または取得できるものであれば、如何なる手法を用いてもよい。例えば、一般的なウェブの場合には「ｈｔｔｐ」を用いてもよく、ウェブサービスの場合には「ｒｅｓｔ」、「ｘｍｌ」、「ｘｍｌ－ｒｐｃ」などを用いてもよい。また、ネットワークサーバの場合には「ｎｓｆ」、「ｗｅｂｄａｖ」、「ｃｉｆｓ」、「ｆｔｐ」などを用いてもよく、物理デバイスの場合には、「ｉＳＣＳＩ」やその他の通常のストレージを用いてもよい。

【０１１５】

次に、テーブルの保存や取得をする場合に用いられる命令（インターフェース）について説明する。テーブルの保存（新規作成）をする場合には「putTable(id, length, list of <offset,object-id>)」が用いられる。これにより、「length」の情報と<offset,object-id>のリストがエンコードされる。ここで、「object-id」は、「block-id」または「table-id」である。また、最も単純なエンコード手法は、文字列としてこの内容を書き出

す方法である。エンコード手法の例としては、（１）可読文字列で表現する手法（例えば、「１０」を、文字列の「１０」で表現する手法）や、（２）バイト列で保存する手法（例えば、「int」は４バイト、「long」は８バイトの列で保存する手法）や、（３）「tuple（タイプ、長さ、データ）」の列としてデータを保存する手法などがある。

【０１１６】

この場合、「table-id」から仮想デバイスが特定され、その「table-id」にエンコードしたデータを保存する。このとき、「ＨＴＴＰ　ＰＵＴ」や「ＮＦＳ　ＷＲＩＴＥ」など、仮想デバイスに応じたデータ保存の仕組みを利用してもよい。なお、実装上は、ブロックの保存のときに用いられる「putContent」を利用してもよい。また、上述のとおり、本システムでは、テーブルの新規保存のみ可能であり、テーブルの上書きは不可である。

【０１１７】

テーブルの情報を取得する場合には「getTable(table-id)」が用いられる。この場合、「table-id」から仮想デバイスが特定され、その「table-id」に対応するデータを取得する。このとき、「ＨＴＴＰ　ＧＥＴ」や「ＮＦＳ　ＲＥＡＤ」など、仮想デバイスに応じたデータ保存の仕組みを利用してもよい。なお、実装上は、ブロックの取得のときに用いられる「getContent」を利用してもよい。取得したテーブルの情報をデコードすることによって、「length（長さ）」と「list of <offset,object-id>（<offset,object-id>のリスト）」を取得することができる。

【０１１８】

なお、この場合、「各「object-id」から取得した「(sub) length」は、次の「offset」までに収まっていること」が条件とされる。また、「最終の「offset」に位置する「object-id」の「length」については、「テーブルの「length」≧ 最終の「offset」+ (sub) length（最終の「offset」の値に「(sub) length」の値を加えた値が、テーブルの「length」の値以下であること）」が、成り立つことが条件とされる。

【０１１９】

続いて、ファイル、テーブル、ブロックのすべてに共通するインターフェース（共通インターフェース）について説明する。書き込み用の共通インターフェースは「writeAt」である。「writeAt」は、「offset」と「length」と「データのバイト列」を受け取る。「writeAt (object-id, object_offset, buffer, bufoff, buflen)」は、「object-id」と「object_offset」と「buffer」と「bufoff」と「buflen」を与えると、更新されたデータを持った新しい「object-id」を返す。読み出し用の共通インターフェースは「readAt」である。「readAt」は、「offset」と「length」と「読み出すデータ領域の情報」を受け取る。「readAt (object-id, object_offset, buffer, bufoff, buflen)」は、「object-id」と「object_offset」と「buffer」と「bufoff」と「buflen」を与えると、該当の領域のデータを読み出した内容を、「buffer」、「bufferoff」、「bufferlen」で示される領域にコピーし、読み出せたデータの長さ「length」を返す。

【０１２０】

この場合、「object-id」は、「file-id」または「table-id」または「block-id」である。また、「object-offset」は、そのオブジェクト内での相対位置（offset）を表現する情報である。また、「buffer」と「bufoff」と「buflen」は、「バッファ」を表現する情報である。読み出しや書き込みの要求は、実際のデータの受け渡しを「buffer」というポインタで始まるデータ領域の「bufoff」で示される相対位置から、「buflen」という長さのバイト分のデータという形で表現する。「writeAt」で書き込みをするときには、上記のデータ領域（バッファ領域）から書き込むべきデータを取得して、データを書き込む。「readAt」で読み出しをするときには、上記のデータ領域（バッファ領域）に読み出したデータをコピーすることにより、データが読み出される。

【０１２１】

なお、上記の説明では、ファイル、テーブル、ブロックのすべてに共通するインターフェースのみを定義したが、実際の動作は、ファイル、テーブル、ブロックのそれぞれで定義してよい。

【０１２２】
次に、本システムの機能の実装例について説明する。ファイルの新規作成（CREATE）の機能は、「create」により実装される。この「create」は、新規に長さがゼロのファイルを作るインターフェースである。「create」の内部処理は、まず「file-id」を新規に割り当て、その「file-id」の先頭テーブルを「EMPTY_TABLE」にセットし、その「file-id」を返すというものである。ここで、「EMPTY_TABLE」は、長さがゼロであって、リストの要素数がゼロである特別なテーブルである。これは、実データを持たないため、仮想デバイス上に実態を持つ必要がなく、特別なオブジェクトＩＤ（空テーブルを示す全ファイルに共通のＩＤ）だけが存在する。「create」は、「file-id」を割り当てるだけのものであるともいえる。

【０１２３】
ファイル（ファイルオブジェクト）への書き込み（WRITE-1）の機能は、「writeAt (object-id, object_offset, buffer, bufoff, buflen)」により実装される。この場合、「object-id」は「file-id」である。このファイルへの書き込みでは、「object_offset」が十分に大きい場合、例えば、現在の階層の深さｄに対して「object_offset」が「４ＭＢ×１０００^{d}」以上である場合に、階層構造をひとつ増やす（木構造を一段深くする）処理が行われる。このとき、「現在のlength」と<0,先頭テーブルのtable-id>を持ったテーブルが新規に作成され、そのファイルの先頭テーブルの「tabale-id」が書き換えられる。また、ファイルに対応づけられている「table-id」に対して「writeAt (table-id, object_offset, buffer, bufoff, buflen)」を実行すると、新しく割り当てられた「table-id」が先頭テーブルとして登録される。これにより、特別なオブジェクトＩＤ又は以前の「table-id」が、新たな「table-id」に書き換えられる。「writeAt」は、更新されたオブジェクトのＩＤを返す必要があるが、ファイルへの書き込みでは、先頭テーブルだけを書き換えて、自身の「file-id」を返すようになっている。

【０１２４】
テーブル（テーブルオブジェクト）への書き込み（WRITE-2）の機能は、「writeAt (object-id, object_offset, buffer, bufoff, buflen)」により実装される。この場合、「object-id」は「table-id」である。このテーブルへの書き込みでは、「buflen」＞０の間、以下の（１）～（４）の処理が繰り返される。

【０１２５】
（１）の処理では、リストから「object_offset」に該当するサブ（下位の）オブジェクト（テーブルまたはブロック）を検索する。例えば、「object_offset + bufoff」がテーブルの各要素の「offset + getLength()」に入る場合を検索する。

【０１２６】
（２）の処理では、該当するオブジェクトがなかった場合、「createId()」により「newId」を作成し、「putContent(newId, writelen, (buffer, bufoff, writelen))」により、その「newId」を「object_offset」に位置するサブオブジェクトとしてリストを更新する。ここで、「writelen」は、「Min(buflen,（次のオブジェクトのoffset － object_offset)）」である。

【０１２７】
（３）の処理では、該当オブジェクトの「child-id」に対して、「writeAt(child-id, object_offset － child-offset, buffer, bufoff, writelen)」を行うことにより、「newId」を「child-offset」に対応するサブオブジェクトとしてリストを更新する。ここで、「child-id」は、該当のサブオブジェクトを示すＩＤである。また、「child-length」は、そのサブオブジェクトの長さ（すなわち、getLength(child-id)で取得される長さ）であり、「child-offset」は、そのサブオブジェクトのoffsetの値である。また、「writelen」は、「Min(buflen, child-length)」である。

【０１２８】
（４）の処理では、該当するオブジェクトがあった場合でも、該当するオブジェクトがなかった場合でも、「object_offset」を「object_offset + writelen」とし、「buflen

」を「buflen － writelen」として、パラメータの更新を実行する。

【０１２９】

このテーブルへの書き込みでは、現状の長さの「length」に対して、「length ＜ object_offset + buflen」であれば、「length」を「object_offset + buflen」とする。また、最終的に出来上がったリストである「list」と、上記の「length」に対して、「createId()」により「newId」を作成し、「putTable(newId, length, list)」を実行したうえで、「newId」を返すようにされている。

【０１３０】

ブロック（ブロックオブジェクト）への書き込み（WRITE-3）の機能は、「writeAt (object-id, object_offset, buffer, bufoff, buflen)」により実装される。この場合、「object-id」は「block-id」である。このブロックへの書き込みでは、該当ブロックの現在のデータを読み出す処理が行われる。例えば、「getContent(object-id)」により「currentData」が読み出される。「currentLength」は、この「currentData」のバイト列（実データ）の長さである。ブロックへの書き込みでは、「currentData」のバイト列について、「object_offset」から始まる部分に対して、「(buffer, bufoff, min(buflen, currentLength))」のデータを上書きする。そして、最終的にできあがったデータと、上記の「currentLength」に対して、「createId()」により「newId」を作成し、「putContent(newId, currentLength, currentData)」を実行したうえで、「newId」を返すようにされている。

【０１３１】

ファイル（ファイルオブジェクト）からの読み出し（READ-1）の機能は、「readAt (object-id, object_offset, buffer, bufoff, buflen)」により実装される。この場合、「object-id」は「file-id」である。このファイルからの読み出しにおいて、「object_offset」が現在の長さ「length」以上の場合には、読み込むべきデータがないので、「END-OF-FILE」を返す。また、ファイルの先頭テーブルである「table-id」に対しては、「readAt (table-id, object_offset, buffer, bufoff, min(buflen, length － object_offset))」を実行する。

【０１３２】

テーブル（テーブルオブジェクト）からの読み出し（READ-2）の機能は、「readAt (object-id, object_offset, buffer, bufoff, buflen)」により実装される。この場合、「object-id」は「table-id」である。このテーブルからの読み出しでは、「buflen」＞０の間、以下の（１）～（４）の処理が繰り返される。

【０１３３】

（１）の処理では、リストから「object_offset」に該当するサブ（下位の）オブジェクト（テーブルまたはブロック）を検索する。

【０１３４】

（２）の処理では、該当するオブジェクトがなかった場合、「readlen = Min(buflen, (次のオブジェクトの offset － object_offset))」に対して、「(buffer, bufoff, readlen)」の領域をnul文字で埋める。

【０１３５】

（３）の処理では、該当するオブジェクトがあった場合、その「child-id」に対して、「readAt(child-id, object_offset － child-offset, buffer, bufoff, readlen)」を実行する。ここで、「child-id」は、該当のサブオブジェクトを示すＩＤである。また、「child-length」は、そのサブオブジェクトの長さ（すなわち、getLength(child-id)で取得される長さ）であり、「child-offset」は、そのサブオブジェクトのoffsetの値である。また、「readlen」は、「Min(buflen, child-length)」である。

【０１３６】

（４）の処理では、該当するオブジェクトがあった場合でも、該当するオブジェクトがなかった場合でも、「object_offset」を「object_offset + readlen」とし、「buflen」を「buflen － readlen」とし、「bufoff」を「bufoff + readlen」として、パラメータ

の更新を実行する。

【０１３７】

ブロック（ブロックオブジェクト）からの読み出し（READ-3）の機能は、「readAt (object-id, object_offset, buffer, bufoff, buflen)」により実装される。この場合、「object-id」は「object-id」である。このブロックからの読み出しでは、該当ブロックの現在のデータを読み出す処理が行われる。例えば、「getContent(object-id)」により「currentData」が読み出される。「currentLength」は、この「currentData」のバイト列（実データ）の長さである。ブロックからの読み出しでは、「currentData」のバイト列に対して、「object_offset」から始まる「readlen」バイト分のデータを、「(buffer, bufoff, readlen)」にロードして、「readlen」を返す。ここで、「readlen」は、「min(buflen, currentLength) 」である。

【０１３８】

次に、ファイルの読み出しと書き込みの具体例について説明する。図９には、本システムにおけるファイルの読み出しアクセスの一例が示されている。図９に示すように、ユーザ端末からファイルを読み出すアクセスがあると、まず、ファイルオブジェクトを特定するための「file-id」が算出される。次に、「file-id」と「table-id」の対応表を用いて、「file-id」に対応する「table-id」を取得する。ここでは、例えば「table-1」というテーブルＩＤが取得される。なお、上述のように、実装上は、「file-id」と「table-id」の対応表も一つのオブジェクトとしてネットワーク上に保存されていてもよい。

【０１３９】

そして、「table-1」の内容を、仮想デバイスの選択アルゴリズム（後述する）を用いて、事業者Ａから取得する。ここでは、例えば「650,<0,table-2>,<300,table-3>」という内容が取得される。以下、「table-2」と「table-3」の内容が再帰的に取得される。図９の例では、「table-2」の内容が「300,<0,block-a>,<100,block-b>,<200,block-c>」であり、「table-3」の内容が「200,<0,block-d>,<100,block-e>」である。そうすると、これらのテーブルの内容に従って、オフセット０の位置に配置された１００バイトの長さの「block-a」のデータと、オフセット１００の位置に配置された１００バイトの長さの「block-b」のデータと、オフセット２００の位置に配置された１００バイトの長さの「block-c」のデータと、オフセット３００（領域オフセット０）の位置に配置された２５０バイトの長さの「block-d」のデータと、オフセット５５０（領域オフセット２５０）の位置に配置された１００バイトの長さの「block-e」のデータが読み出される。

【０１４０】

図１０～図１４には、本システムにおけるファイルの書き込みアクセスの一例が示されている。ここでは、まず、新規ファイルの作成について説明する。図１０に示すように、ユーザ端末からファイルを書き込むアクセスがあると、まず、「CREATE」の機能により、新規ファイルが作成される。このとき、先頭テーブルに「EMPTY_TABLE」がセットされる。

【０１４１】

次に、先頭から１００バイトのデータ（data）を書き込む処理について説明する。この場合、「WRITE-1」の機能により、ファイルへの書き込みが実行される。例えば、先頭テーブルの「EMPTY_TABLE」に対して、「writeAt(EMPTY_TABLE, 0, data, 0, 100)」が実行される。その後、「WRITE-2」の機能により、テーブルへの書き込みが実行される。この場合、「offset」に該当するブロックが存在しないため、ブロックを作成する。例えば、「block-a = createId()」、「putContent(block-a, 0 － 0, data, 0, 100)」により「block-a」のブロックを作成して、リストを更新する。次いで、「writeAt」内のメインループを抜けて、「length」を「１００」に更新した後、「table-x = createId()」、「putTable(table-x, 100, <0, block-a>)」を実行し、「table-x」を返す。そして、上記「table-x」を先頭テーブルにセットし、処理を終了する。

【０１４２】

続いて、図１１および図１２を参照しながら、「offset」が「５０から１５０バイト」

のデータ（data）を書き込む処理について説明する。図１１に示すように、まず、「WRITE-1」の機能により、ファイルへの書き込みが実行される。この場合、先頭テーブルの「table-x」に対し、「writeAt(table-x, 50, data, 0, 150) 」が実行される。次に、「WRITE-2」の機能により、テーブルへの書き込みが行われる。この場合、第１のループ「loop-1」として、「offset」が「５０」であるブロック「block-a」に対して、「wiriteAt(block-a, 50 － 0, data, 0, 50) 」が実行される。続いて、「WRITE-3」の機能により、「block-a」の現在のデータである１００バイト分の「orig」をロードし、「orig」の「offset=50」に「(data, 0, 50) 」を上書きする。その後、「block-b = createId()」、「putContent(block-b, 100, orig)」で新規ブロックを作成する。そして、上記の「block-b」を、新たなブロックとして、リストを更新する。その結果、新しいリストは<0,block-b>になる。その後、「object_offset 」に「50 + 50 = 100」をセットし、「buflen」に「150 － 50 = 100」をセットし、「bufoff」に「0 + 50 = 50」をセットして、次のループへ進む。

【０１４３】

図１２に示すように、第２のループ「loop-2」では、「offset」が「１００」に該当するブロックがないため、「block-c = createId()」、「putContent(block-c, 100, (data, 50, 100))」を実行し、新規ブロックを作成する。これにより、テーブルのリストが更新され、新しいリストは「<0,block-b>,<100,block-c>」になる。このとき、「object_offset」に「100 + 100 = 200」をセットし、「buflen」に「100 － 100」をセットし、「bufoff」に「50 + 100 = 150」をセットする。この場合、「buflen」がゼロになるので、ループを終了する。そして、「length」を「２００」にセットしたうえで、「table-y = createId()」、「putTable(table-y, 200, <0,block-b>,<100,block-c>)」を実行して新規テーブルを作成し、「table-y」を返す。その後、ファイルの先頭テーブルを「table-y」にセットして処理を終了する。

【０１４４】

次に、図１３および図１４を参照して、「object_offset」が「１００００から２００バイト」のデータ（data2）を書き込む処理について説明する。図１３に示すように、「WRITE-1」の機能により、ファイルへの書き込みが実行される。この場合、「object_offset」が十分に大きいと判断され、階層を増やす処理が実行される。つまり、現状の「table-y」を単一要素とする、新テーブルが作成される。なお、このとき、テーブルの深さだけが変わるだけで、「getLength()」で取得できる値は「table-y」と同じである。新しく作られたテーブルについては、「table-α = createId()」、「putTable(table-α, 200, <0,table-v>)」が実行される。そして、この新しく作られたテーブル「table-α」に対して「writeAt(table-α, 10000, data2, 0, 200)」が実行される。

【０１４５】

続いて、「WRITE-2」の機能により、「table-α」に対する書き込みが行われる。この場合、「object_offset = 10000」に既存の要素が存在しないので、「block-d = createId()」、「putContent(block-d, 200, (data2, 0, 200))」を実行し、新規ブロックを作成する。その後、各種パラメータを更新する。この場合、「object_offset」を「10000 + 200 = 10200」にセットし、「buflen」を「200 - 200」にセットし、「bufoff」を「0 + 200= 200」にセットして、ループを終了する。

【０１４６】

図１４に示すように、この場合、「length ＜object_offset + len」であるので、「length」に「10200」をセットして、長さを更新する。さらに、テーブルのリストを更新する。新しいテーブルは、「table-z = createId()」、「putTable(table-z, 10200, <0,table-y>,<10000,block-d>)」によって与えられる。最後に、ファイルの先頭テーブルを「table-z」にセットして、処理を終了する。

【０１４７】

ここで、本システムにおける新規ＩＤの割当について簡単に説明する。本システムでは、「createId」で新規のオブジェクトＩＤを作る。このとき、オブジェクトＩＤの一意性

を保証することが必要である。オブジェクトＩＤの一意性は、サービステーブルで利用される。例えば、ＵＵＩＤを用いる場合には、ＲＦＣに記述されている手法で、一意な１２８ビットの値が生成される。また、６４ビットの整数を用いる場合には、「64 bit long generator (original)」を利用してもよい。

【０１４８】

次に、本システムで利用する「サービステーブル」について説明する。本システムにおいて、オブジェクトＩＤ（テーブルＩＤ、ブロックＩＤなど）から、その内容を取得する際には、「内容を保存してあるサービス（もしくはサーバ）」を特定するアルゴリズムが用いられる。したがって、本システムでは、データベースや膨大な管理テーブルを準備する必要がない。すなわち、本システムでは、サービスを特定しようとするときに、アルゴリズムで「計算」できるため、特別な管理サーバや管理情報のための領域が不要になる。

【０１４９】

サービステーブルは、オブジェクトＩＤからオブジェクトを取得する方法を示すテーブルである。本システムでは、上述したように「アルゴリズム」によって、オブジェクトＩＤからサービス（もしくはサーバ）を特定できるようになっており、サービステーブルは、各サービス（もしくはサーバ）への「アクセス手段」についての情報を得るために用いられる。サービステーブルには、実装と運用が簡単な「静的」なものと、サービステーブルの更新（サービスの追加・変更・削除）が容易な「動的」なものがある。

【０１５０】

静的なサービステーブルでは、Ｎ個のサービスがあるとして、オブジェクトＩＤから「hash code」を計算し、その計算結果をＮで割った余りでサービスを特定できるようになっている。動的なサービステーブルでは、各サービス名称（一意な名称）がハッシュ値に変換されて、オブジェクトＩＤがそのハッシュ値に一番近いサービスが特定されるようになっている。

【０１５１】

本システムでは、このようなサービステーブル（静的なサービステーブルまたは動的なサービステーブル）を利用することにより、特別な管理用テーブルが不要になり、システムにおける単一障害点をなくすことが可能になり、バックアップが容易になる。また、オブジェクトＩＤごとに異なるサービスを割り当てることによって、分散処理が可能となる。また、サービスを特定するためのアルゴリズムは計算が早い（例えば、処理時間が「0(1)」のオーダーである）。

【０１５２】

サービステーブルに含まれる情報には、サービス提供事業者の情報、ＵＲＩなどの「location」の情報、サービスを利用するための認証情報、その他、サービスに依存する付加情報がある。サービス提供事業者の情報は、自ネットワークの場合には「local」の情報であってよい。「location」の情報には、「ｈｔｔｐ，ｎｆｓ，ｃｉｆｓ，ｆｔｐ，ｗｅｂｄａｖ」などのプロトコルの情報や、「%o（オブジェクトＩＤに置き換え）」や「%u（ユーザＩＤに置き換え）」などのパス情報（必要に応じてパラメータ変換を行うための情報）が含まれる。また、サービスを利用するための認証情報には、ユーザＩＤ、パスワード、必要に応じて認証鍵情報などが含まれる。また、サービスに依存する付加情報には、ウェイト、暗号化手法、各種のパラメータ（ブロックサイズ、並列度、待ち行列のサイズなど）が含まれる。

【０１５３】

図１５は、静的なサービステーブルの一例を示す図である。この場合、Ｓ個のサービスに対して、０～（Ｓ－１）の番号を持つサービステーブルを作成する。オブジェクトＩＤからサービスを決定する場合には、まず、オブジェクトＩＤ（例えば、０ａ１２ｃｄ－０５２０１ａ－・・・－ａｂ００ｆａ）からハッシュ値（例えば、１３５６２）を計算する。このハッシュ値をサービス数Ｓで割った余りからサービスを特定する。例えば、サービス数が３（つまり、Ｓ＝３）である場合には、１３５６２÷３＝２となり、サービス提供者は、「事業者Ｃ」に決定される。そして、アクセス方法について前述のパラメータ変換

を行った後に、この事業者Ｃのサービスにアクセスする。

【０１５４】

図１６は、動的なサービステーブルの一例を示す図である。この場合、サービスごとにＳＨＡ１ハッシュを計算し、サービステーブルを作成する。オブジェクトＩＤからサービスを決定する場合には、まず、オブジェクトＩＤ（例えば、０ａ１２ｃｄ－０５２０１ａ－・・・－ａｂ００ｆａ）からＳＨＡ１ハッシュ値（例えば、６０ａｂ）を計算する。そして、このＳＨＡ１ハッシュ値が担当範囲として割り当てられているサービスが選択される。そして、図１５の場合と同様に、アクセス方法について前述のパラメータ変換を行った後に、そのサービスにアクセスする。

【０１５５】

本システムでは、サービスを動的に追加、削除するための一つの手法として、ＤＨＴ（Distributed Hash Table）を利用することができる。例えば、図１７に示すように、サービスを提供する８つのサーバ（サーバＡ～Ｈ）にそれぞれキー情報（ＩＤ）が割り当てられており、それぞれのサーバの担当範囲がＩＤに基づいて決められていたとする。各サーバのＩＤは、そのサーバの一意な名称から計算したＳＨＡ１ハッシュ値の一部（例えば、先頭の二桁）をとって決めてもよい。例えば、サーバＡの担当範囲が「０８～３４」、サーバＢの担当範囲が「３５～４８」・・・に設定されていたとする。そうすると、上記のオブジェクトＩＤについては、ＳＨＡ１ハッシュ値の一部（例えば、先頭の二桁）の「６０」から、サービスを提供するサーバは、「サーバＤ」（担当範囲「６０～９０」）に決定される。

【０１５６】

以下、本システムの特徴について説明する。本システムの特徴の一つとして、ランダムアクセス（Random Access I/O）が可能であることが挙げられる。本システムでは、ツリー構造によるデータアクセスが可能であり、任意の場所のデータを「O(log N)」のオーダーの処理時間で更新、追加、削除可能である（Ｎ＝ファイルサイズ）。なお、固定配列にすると、保存できるファイルのサイズに制限ができてしまい、リスト構造にすると「O(N)」のオーダーの処理時間が必要になってしまう。そして、本システムでは、書き込み時に、オフセットと領域を指定して更新を行う。そのため、更新が必要なブロック、及び、上位のテーブルのみ更新すればよいという利点がある。また、本システムでは、読み込み時には、オフセットと領域を指定して読み出しを行う。そのため、読み出しが必要なテーブルとブロックだけ読み出せばよいという利点がある。

【０１５７】

また、本システムには、ファイルサイズが事実上無限であるという特徴がある。すなわち、固定長配列をもたないためファイルサイズは事実上無制限であり、例えば、ブロックサイズ４ＭＢ、テーブルあたりの要素数が１０２４の場合には、深さ４で「４ＭＢ×１０２４×１０２４×１０２４×１０２４＝４ＥＢ」を表現できる。また、データ構造がツリー型の再帰構造をとるため、ファイルサイズＮに対し、データの検索、追加、削除の処理時間は「O(log N)」のオーダーであり、実用的な速度（十分な速度）でアクセスできる。

【０１５８】

また、本システムは、必要最小限の管理情報が「file-id」と「table-id」の対のみであるという点に特徴がある。つまり、本システムでは、ファイルを示す「file-id」とその先頭の「table-id」のみが必要であり、実装上は、「file-id」と「table-id」の対もネットワーク上のオブジェクトとして保存することができる。例えば、「file-id」をキーに単一の値を持つオブジェクトとして分散配置することができる。なお、「file-id」と「table-id」の対は、テーブルやブロックとは異なり、上書き可能なオブジェクトである。また、サービステーブルは、別途管理すればよく、ファイルやデータの「量に依存しない」参照情報のみを含めるようにすることができる。

【０１５９】

また、本システムの特徴には、スパースファイル（sparse file）を容易に実現できることがある。スパースファイルでは、使ってないところが「nul埋め」される。したがっ

て、例えば、「readAt」を実行した場合、データがないところは「nul」データとして読み出される。これにより、物理容量を使わずに論理的なファイルサイズが大きくできる。スパースファイルの第１のメリットとして、論理的には事実上無限サイズのファイルを作成可能であることが挙げられる。ファイルの構造は、テーブルだけで表現することが可能であり、データのないところは「nul」データとして読み出される。スパースファイルの第２のメリットとして、実際にデータを書き込んだときに初めてブロックが作成されることが挙げられる。データを書き込まない部分は、必要な容量は最小限（テーブル情報のみ）で済む。

【０１６０】

また、本システムは、暗号化との組み合わせで安全なデータ書き込みを実現できる点に特徴がある。事前にユーザがファイル全体を暗号化することにより、安全なデータ書き込みが実現される。暗号化方式は任意である。本システムでは、暗号化されたファイルを多数のブロックに分割し、複数の仮想デバイス（例えば、サービス事業者）に分散して保存する。ファイルの復号化は、仮想デバイスから各断片を集めてきてファイルを再構成し、ファイルを再構成後、ユーザ側で復号化すればよい。この特徴により、暗号化されたデータを、より安全に保存することができる。この場合、各仮想デバイスは「暗号化されたファイル」の断片のみを持つことになるので、仮想デバイスがサービス事業者である場合、特定のサービス事業者のデータだけでは、元ファイルを生成することはできない。

【０１６１】

また、本システムの特徴の一つには、アトミックな更新（途中の状態が存在しない不可分な更新処理）が可能であることが挙げられる。ファイルの更新は、先頭テーブルの「table-id」のアトミックな書き換えで確定する。ファイルを構成する「object-id」は、毎回新規作成され、「table-id」と「block-id」は、いずれも「更新」するたびに新しいIDが割り当てられる。すなわち、「table-id」が決められると、それ以降、その中身は同一であり変更されることがない。このような特徴を備えることにより、本システムでは「書き込み中の読み込みが可能である」というメリットが得られる。ファイルは常に「完全な状態」であるため、先頭テーブルの「table-id」が書き換えられるまでは、以前のファイルのままであり、先頭テーブルの「table-id」が書き換えられると、即座に、更新されたファイルの内容になる。したがって、ファイルのロックをしなくても「書き込み中に読み込みも実行できる」ようになる。また、本システムでは「コピー、スナップショットが容易である」というメリットも得られる。コピーやスナップショットの方法としては、例えば、新しい「file-id」に対して、既存ファイルの「table-id」をコピーする方法がある。これにより、同一の「table-id」の内容の同一性が保証される。

【０１６２】

また、本システムは、クライアントの分散処理が容易であるという特徴を備えている。図１８において破線で示すように、例えば、人気のＶＯＤ（ビデオ・オン・デマンド）やイベント時のホームページなど、または、大容量のデータの分析を「分散処理」で行う場合など、一つのクライアント装置にアクセスが集中してしまうことがある。そのような場合に、集中アクセスの分散して処理することが望まれる。また、非同期に多数のアクセスがある場合にも、負荷分散をすることが望まれる。

【０１６３】

本システムでは、仮想ストレージ（上述した「クライアント装置」）を複数（多数）並べて、必要なファイルの情報（file-idとtable-idの対）を各仮想ストレージで取得するようにしている。したがって、図１８において実線で示すように、各仮想ストレージが独立に仮想デバイスにアクセスすることで分散処理を実現することができる。例えば、仮想デバイスがサービス事業者である場合には、事業者間の負荷分散が可能になる。また、仮想デバイスがサーバである場合には、サーバ間での負荷分散が可能になる。いずれの場合にも、フロントエンドの個々の仮想ストレージはお互いに干渉しない。また、本システムでは、特に大容量ファイルの場合に負荷分散が容易である。その場合、構成するテーブル、ブロック数が多くなるため、統計的にアクセスが分散することになる。結果として、本

システムでは、極めてスケーラビリティが高い「アクセス分散」の仕組みが実現できる。

【０１６４】

例えば、本システムで保存管理するファイルは、オブジェクト指向プログラムのクラス情報であってもよく、本システムの利用者が指定したプログラムをサーバ上で実行できるようにしてもよい。その場合、本システムの利用者（クライアント装置）とサーバで、プログラムの定義であるクラス情報を共有してもよい。クラス情報を共有する場合には、例えば、図１９に示すように、利用者は、本システムのサーバにクラス情報を登録し、各サーバにファイルＩＤを教えてから、プログラムの実行を指示する。各サーバでは、ファイルＩＤを用いて本システムからプログラムの実行に必要なクラス情報を取得することができる。

【０１６５】

また、本システムの特徴には、並列分散処理により高スループットが得られるという点がある。図２０は、本システムにおける並列分散処理の説明図である。図２０に示すように、例えば、テーブルへの書き込み時には、テーブルが持つリスト内の各オブジェクト（テーブルまたはブロック）毎の書き込み処理の並列分散処理が可能である。また、テーブルからの読み出し時には、テーブルが持つリスト内の各オブジェクト（テーブルまたはブロック）毎の読み出し処理の並列分散処理が可能である。このように、例えば、ひとつのテーブルの書き込み／読み出しのいずれの場合にも、該テーブルの処理において、テーブル内のリストの要素数Ｎ（例えば、１０２４）に対して並列分散処理が可能である。結果として、本システムでは、サイズの大きなファイルほど、並列分散処理が有効に機能することになり、高スループットを実現できる。

【０１６６】

また、本システムは、複製や冗長化の実現の仕方に特徴がある。本システムにおける冗長化の方式は「複数の仮想デバイスに複製を作成する」というものである。つまり、書き込み時には、オブジェクト（テーブルまたはブロック）を複数の仮想デバイスにコピーし、読み込み時には、オブジェクトを上記のいずれかの仮想デバイスから読み出すようにしている。複製を作成する仮想デバイスの選択では、サービステーブルが利用できる。本システムでは、本来、オブジェクトを保存すべき仮想デバイスの「隣の仮想デバイス」が選択される。この場合、冗長度（＝多重度＝複製の数）に応じて、複数の仮想デバイスが選択される。静的なサービステーブルの場合には、Ｎで割った余りをもとに「隣の仮想デバイス」を決定する。例えば、「（ｎ÷Ｎ）の余り、（（ｎ－１）÷Ｎ）の余り、（（ｎ－２）÷Ｎ）の余り、・・・」が、該当の仮想デバイスとして決定される。動的なサービステーブルの場合には、該当の仮想デバイスの一つ前のサーバが、該当の仮想デバイスとして決定される。なお、書き込み時の処理は、該当の仮想デバイスすべてに対して同じオブジェクトのデータを書き込む。また、読み出し時の処理は、該当の仮想デバイスに対して順次読み込みを行い、最初にレスポンスが返ってきた時点で、それを読み込みデータとして用いる、または、該当の仮想デバイスに対して並列読み込みを行い、最もレスポンスが早かったものを読み込みデータとして用いる。

【０１６７】

図２１は、本システムの特徴の説明図である。図２１に示すように、本システムでは、ファイルを多数のブロックとして保存し、並列にアクセスすることで、ストレージ装置の台数に比例したアクセス性能とストレージ容量を実現できる。この場合、クライアント装置（アクセスノードと呼ぶこともできる）は、アクセス性能に応じて台数を増やすことができる。また、ストレージ装置（コアノードと呼ぶこともできる）は、容量に応じてスケーラブルに追加が可能である。つまり、本システムは、高度な分散コンピューティング技術を応用することにより、性能／容量とも台数に比例し、容易に拡張（スケールアウト）が可能であるともいえる。

【０１６８】

以上、本発明の実施形態について説明したが、上述の実施形態を本発明の範囲内で当業者が種々に変形、応用して実施できることは勿論である。

【産業上の利用可能性】

【０１６９】

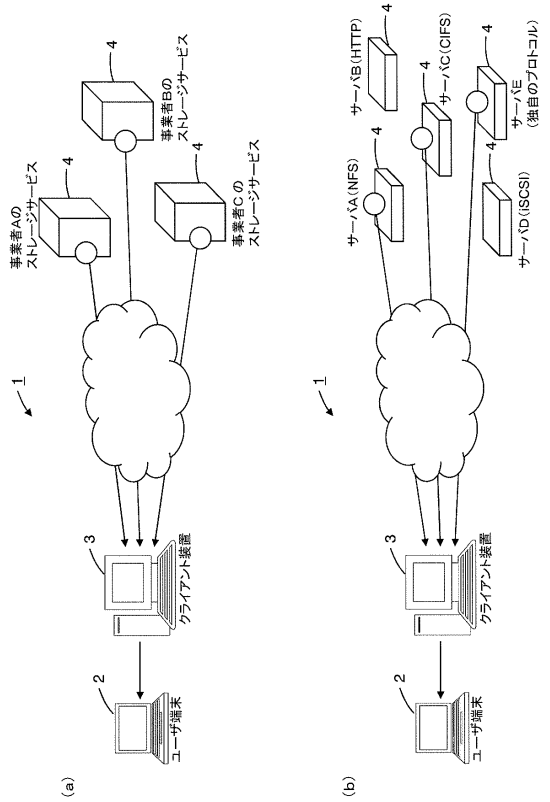
以上のように、本発明にかかるストレージサービス提供装置は、例えば、ストレージサービス等のクラウドサービスに利用することができ、有用である。

【符号の説明】

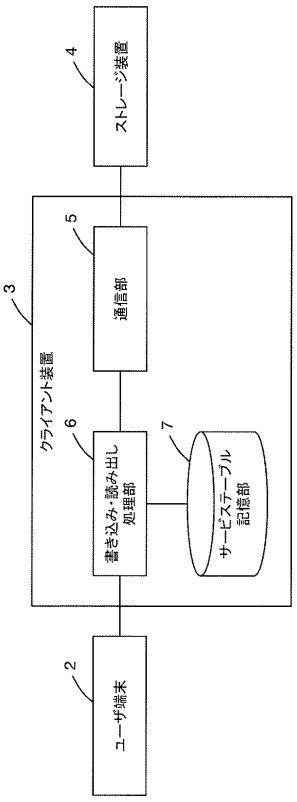
【０１７０】

１　分散型ストレージサービス提供システム（本システム）
２　ユーザ端末
３　クライアント装置（ストレージサービス提供装置）
４　ストレージ装置
５　通信部
６　書き込み・読み出し処理部
７　サービステーブル記憶部
８　通信部
９　ファイル・オブジェクト保存部
１０　ファイル・オブジェクト管理部

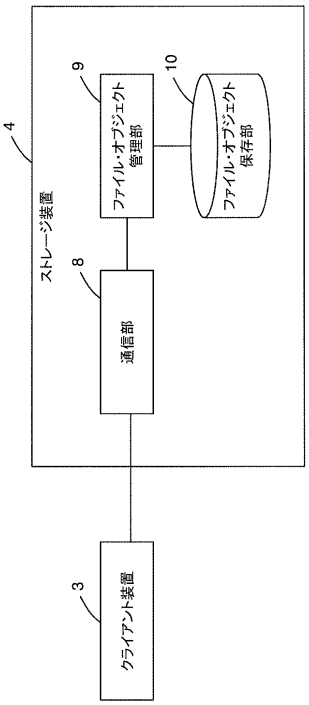
【図１】



【図２】



【図３】



【図４】

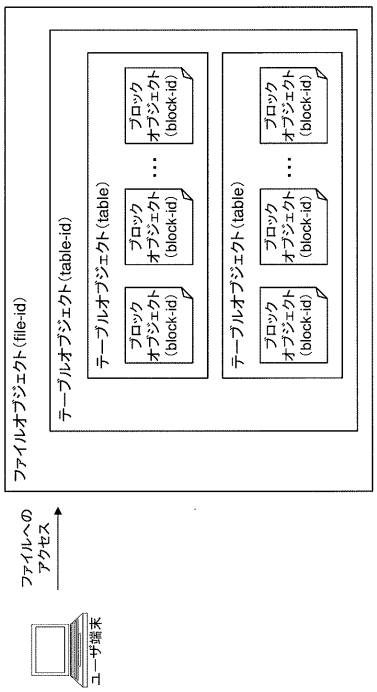
(a) ファイルの作成と削除

機能名	概要	パラメータ等
CREATE	ファイルの作成	なし
DELETE	ファイルの削除	file-id

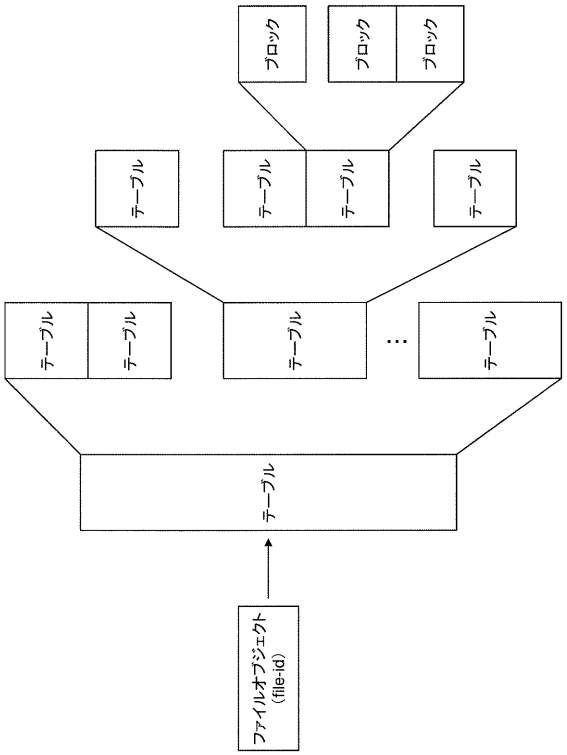
(b) データの読み書き

機能名	概要	パラメータ等
READ	ファイルからのデータの読み出し	file-id, offset, data(lengthを含む)
WRITE	ファイルへのデータの書き込み	file-id, offset, データ領域(lengthを含む)

【図５】



【図６】



【図７】

table-1（先頭テーブル）

950	このテーブルに対して getLength(table-id) して取得できる長さ
<0,block-20>	offset = 0 から block-20 が始まる、長さは getLength(block-20) = 100 バイト
<100,table-3>	offset = 100 から table-3 が始まる、長さは getLength(table-3) = 500 バイト
<600,block-12>	offset = 600 から block-12 が始まる、長さは getLength(block-12) = 150 バイト
<750,table-10>	offset = 750 から table-10 が始まる、長さは getLength(table-10) = 200 バイト

table-1をreadAtした結果

```
>   0----------------------------------
>       block-20のデータ
> 100----------------------------------
>       table-3のテーブル（リスト）
> 600----------------------------------
>       block-12のデータ
> 750----------------------------------
>       table-10のテーブル（リスト）
> 950----------------------------------
```

table-3

500	getLength(table-3) で取得できる長さ
<0, block-120>	offset=0 から block-120 が始まる。長さは getLength(block-120) = 100 とする。
<300, block-130>	offset=300 から block-130 が始まる、長さは getLength(block-130) = 200 とする。

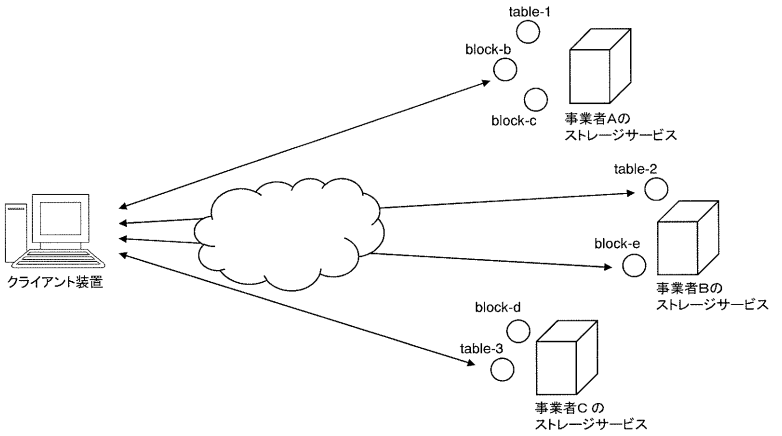
table-10

200	getLength(table-10) で取得できる長さ
<0, block-800>	offset=0 から block-800 が始まる。長さは getLength(block-800) = 50 とする。
<50, block-900>	offset=50 から block-900 が始まる。長さは getLength(block-900) = 150 とする。

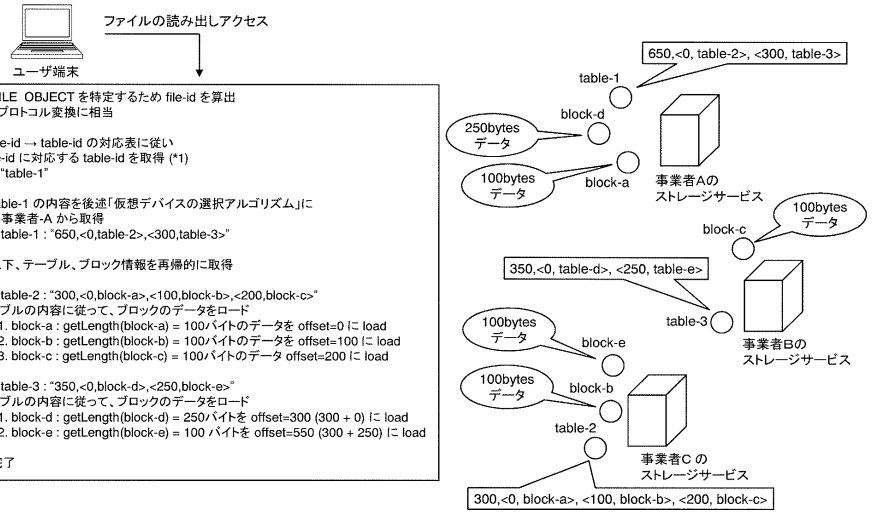
table-1で表現されるデータ

```
offset = 0   ------------------------------------------------------------------ offset = 0
                  block-20 (length = 100)
offset = 100 ------------------------------------------------------------------ offset = 100
                  table-3                          block-120 (length=100)
                                                   ---------------------------- offset = 200
                                                   <brank == nul (character code = 0) で埋められる>
                                                   ---------------------------- offset = 400
                                                   block-130 (length=200)
offset = 600 ------------------------------------------------------------------ offset = 600
                  block-12 (length=150)
offset = 750 ------------------------------------------------------------------ offset =750
                  table-10                         block-800 (length=50)
                                                   ---------------------------- offset = 800
                                                   block-900 (length=150)
offset = 950 ------------------------------------------------------------------ offset =950
```

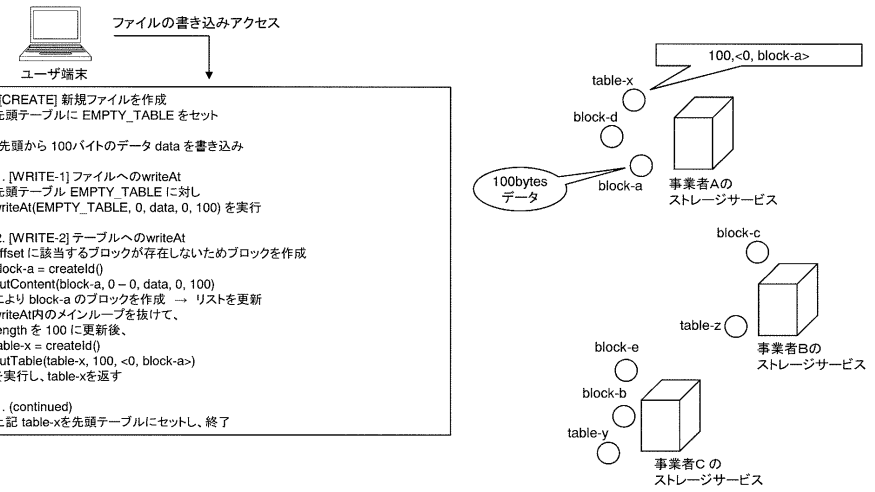
【図８】



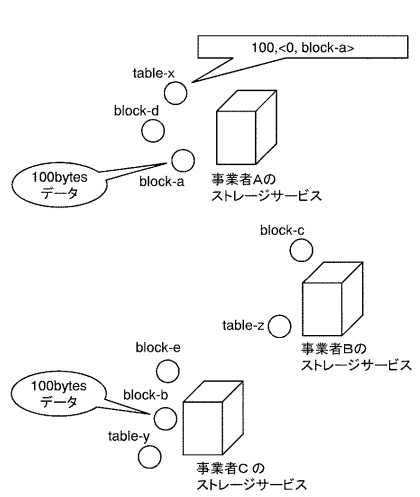
【図９】



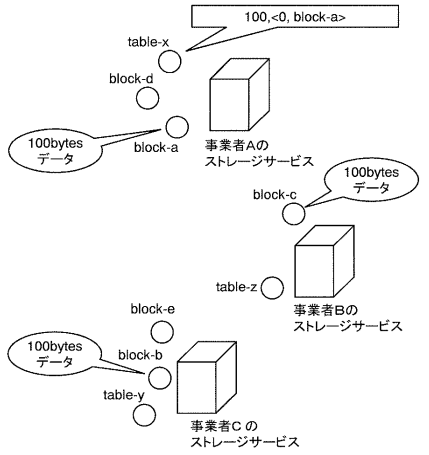
【図１０】



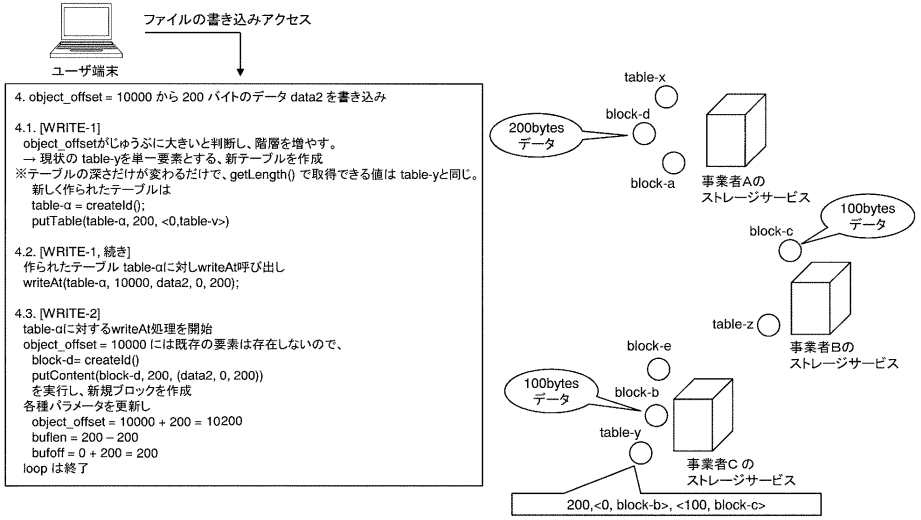
【図１１】



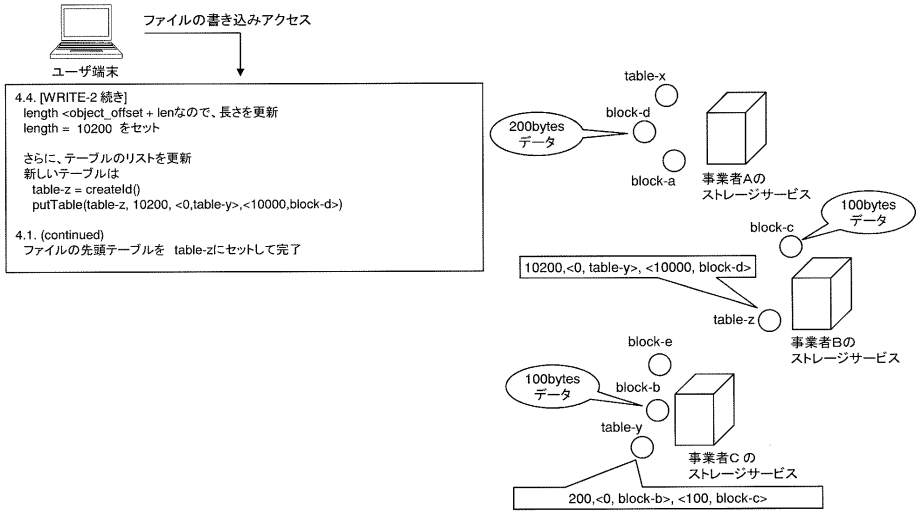
【図１２】



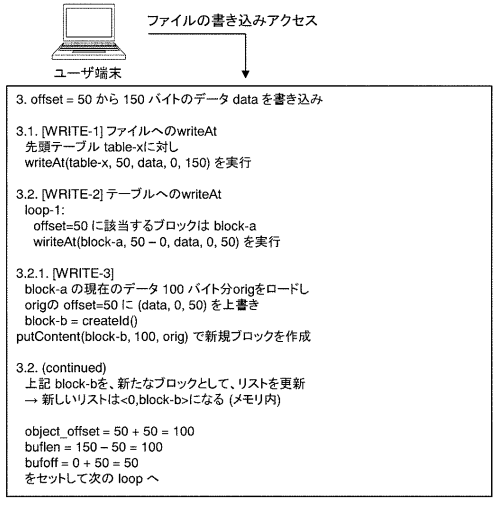
【図１３】



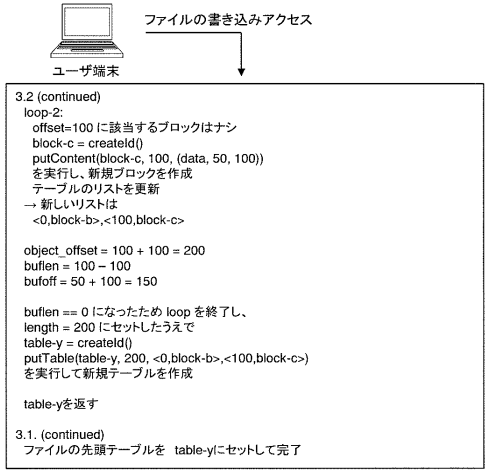
【図１４】



【図１５】



【図１６】



【図１５】

静的(static)なサービステーブル

番号	提供者	アクセス方法	認証
0	事業者A	http://a.com/storage/%o	user-1/pass-1
1	事業者B	http://b.com/disk/%u/%o	user01/pass-2
2	事業者C	nfs://c.com/home/%u/disk/%o	user001/pass-3
…	…	…	…

パラメータ変換前

nfs://c.com/home/%u/disk/%o

→

パラメータ変換後

nfs://c.com/home/user-id/disk/0a12cd-05201a-…-ab00fa

【図１６】

動的(dynamic)なサービステーブル

番号	提供者	アクセス方法	認証
01a3	事業者A	http://a.com/storage/%o	user-1/pass-1
25a0	事業者B	http://b.com/disk/%u/%o	user01/pass-2
5c8a	事業者C	nfs://c.com/home/%u/disk/%o	user001/pass-3
…	…	…	…

パラメータ変換前

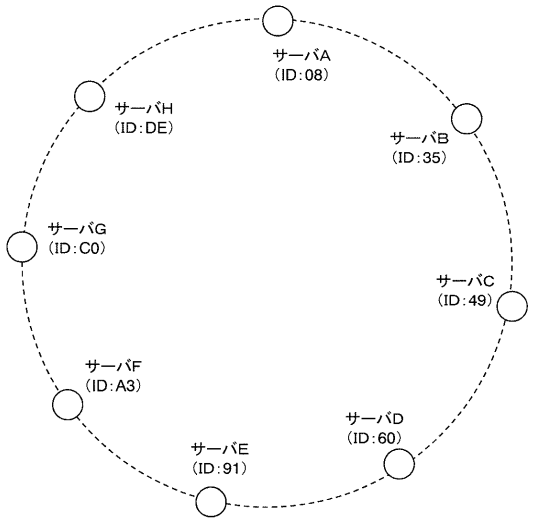
nfs://c.com/home/%u/disk/%o

→

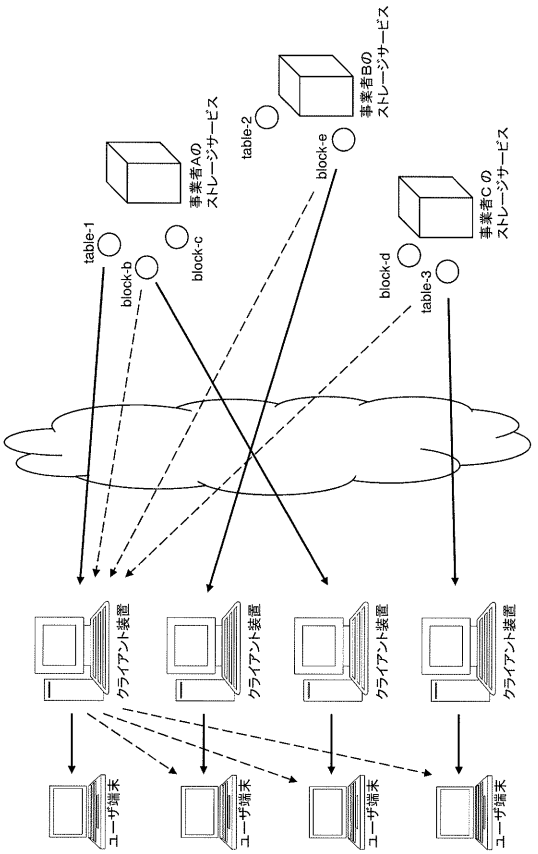
パラメータ変換後

nfs://c.com/home/user-id/disk/0a12cd-05201a-…-ab00fa

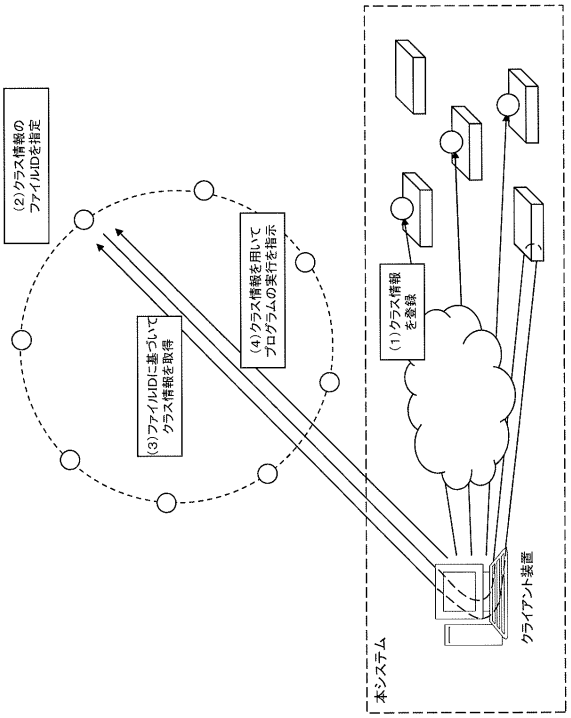
【図１７】



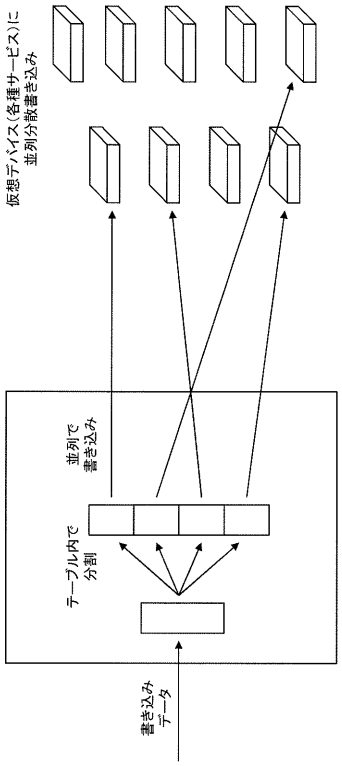
【図１８】



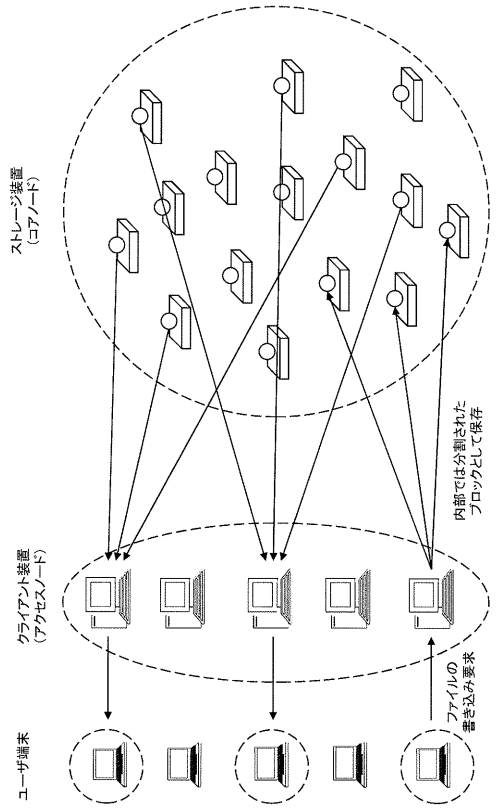
【図１９】


(2)クラス情報の
ファイルIDを指定
(3)ファイルIDに基づいて
クラス情報を取得
(4)クラス情報を用いて
プログラムの実行を指示
(1)クラス情報
を登録
クライアント装置
本システム

【図２０】


仮想デバイス(各種サービス)に
並列分散書き込み
並列で
書き込み
テーブル内で
分割
書き込み
データ

【図２１】


ストレージ装置
(コアノード)
クライアント装置
(アクセスノード)
ユーザ端末
内部では分割された
ブロックとして保存
ファイルの
書き込み要求

フロントページの続き

(56)参考文献 特開平０８－１５３１１１（ＪＰ，Ａ）
特開平０９－２５９０２８（ＪＰ，Ａ）
特開平１０－１４３４０３（ＪＰ，Ａ）
特開平１０－２６０８９０（ＪＰ，Ａ）
特開２００２－２２９８４２（ＪＰ，Ａ）
特開２００２－３５１７０５（ＪＰ，Ａ）
特開２００６－３０９３９８（ＪＰ，Ａ）
特表２００８－５００６０８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　３／０６
Ｇ０６Ｆ　１２／００
Ｇ０６Ｆ　１３／００
Ｇ０６Ｆ　１３／１０
Ｇ０６Ｆ　１７／２１
ＪＳＴＰｌｕｓ（ＪＤｒｅａｍＩＩ）