(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2005-539315
(P2005-539315A)
(43)公表日　平成17年12月22日(2005.12.22)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 12/00	G06F 12/00	546K	5B082
G06F 13/00	G06F 13/00	540B	

審査請求 未請求　予備審査請求 未請求　（全 39 頁）

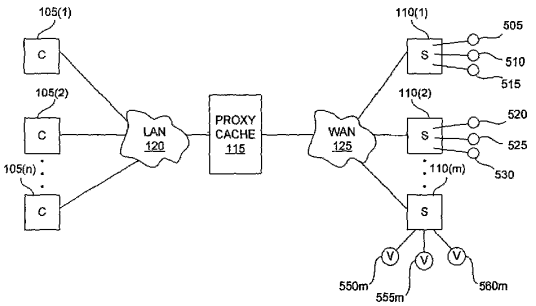
(21)出願番号	特願2004-536622 (P2004-536622)
(86)(22)出願日	平成15年9月16日 (2003.9.16)
(85)翻訳文提出日	平成17年4月28日 (2005.4.28)
(86)国際出願番号	PCT/US2003/029398
(87)国際公開番号	WO2004/025429
(87)国際公開日	平成16年3月25日 (2004.3.25)
(31)優先権主張番号	10/245,828
(32)優先日	平成14年9月16日 (2002.9.16)
(33)優先権主張国	米国 (US)

(71)出願人　500261341
ネットワーク・アプライアンス・インコーポレイテッド
アメリカ合衆国９４０８９カリフォルニア州サニーベイル、イースト・ジャーバ・ドライブ４９５番
(74)代理人　100087642
弁理士　古谷　聡
(74)代理人　100076680
弁理士　溝部　孝彦
(74)代理人　100121061
弁理士　西山　清春

最終頁に続く

(54)【発明の名称】プロキシ・キャッシュに関する装置および方法

(57)【要約】
一実施形態において本発明は、データをネットワーク内にキャッシュする装置を提供する。この装置は、クライアント（１０５）からオブジェクトの要求を受信し、サーバ（１１０）からデータブロックをフェッチするように構成されたプロキシ・キャッシュ（１１５）を含む。プロキシ・キャッシュは、データブロックをオブジェクト内に階層関係でキャッシュするように構成される。オブジェクトには、例えばデータファイルやディレクトリがある。プロキシ・キャッシュにキャッシュされたデータブロックは、クライアントからの要求に基づくアクティブデータセットを規定する。



【特許請求の範囲】
【請求項１】
データをネットワーク内にキャッシュする装置であって、
クライアントからオブジェクトの要求を受信し、サーバからデータブロックをフェッチするように構成されたプロキシ・キャッシュを含み、
前記プロキシ・キャッシュは、階層関係の前記データブロックをアクティブデータセットとしてオブジェクト内にキャッシュするように構成される、装置。
【請求項２】
前記データブロックがバッファ・ツリーに入れられ、ファイル内に前記階層関係が可能である、請求項１に記載の装置。
【請求項３】
クライアントからの要求に応答して、データブロックが前記アクティブデータセットに配置される、請求項１に記載の装置。
【請求項４】
前記アクティブデータセットは、少なくともサーバに格納されたデータの一部を含む、請求項１に記載の装置。
【請求項５】
前記アクティブデータセットは、特定の時間によって定義される、請求項１に記載の装置。
【請求項６】
前記プロキシ・キャッシュは、サーバによって要求されたファイルがデータブロックを欠いていた場合、クライアントからの要求に応答してサーバから前記データブロックをフェッチするように構成される、請求項１に記載の装置。
【請求項７】
前記プロキシ・キャッシュは、サーバによって要求されたファイルがプロキシ・キャッシュ内に存在しなかった場合、クライアントからの要求に応答してサーバから前記データブロックをフェッチするように構成される、請求項１に記載の装置。
【請求項８】
前記プロキシ・キャッシュと前記クライアントが、オープン・スタンダード・プロトコルを用いて通信する、請求項１に記載の装置。
【請求項９】
前記プロトコルは、インターネット・プロトコルを含む、請求項８に記載の装置。
【請求項１０】
前記プロトコルは、ネットワーク・ファイル・システム（ＮＦＳ）プロトコルを含む、請求項８に記載の装置。
【請求項１１】
前記プロトコルは、コモン・インターネット・ファイル・システム（ＣＩＦＳ）プロトコルを含む、請求項８に記載の装置。
【請求項１２】
前記プロキシ・キャッシュは、クライアントによって要求されたデータファイルにおけるデータブロックの有無を判定するための計算の一部として重複しない名前を計算する、請求項１に記載の装置。
【請求項１３】
置換ポリシーに基づいて、前記プロキシ・キャッシュから前記アクティブデータセットの一部を消去することができる、請求項１に記載の装置。
【請求項１４】
前記オブジェクトはデータファイルを含む、請求項１に記載の装置。
【請求項１５】
オブジェクトはファイルの一部を含む、請求項１に記載の装置。
【請求項１６】

データをネットワーク内にキャッシュする方法であって、
クライアントからオブジェクトの要求を受信するステップと、
要求された前記オブジェクト中の各データブロックがキャッシュ内にあれば、要求されたオブジェクトをクライアントに送信するステップと、
要求された前記オブジェクト中のデータブロックが欠けていた場合、該データブロックをサーバからフェッチし、該データブロックをアクティブデータセットの一部として階層関係に埋め込み、要求された前記オブジェクトをクライアントに送信するステップと、
からなる方法。

【請求項１７】
サーバによって要求されたファイルが存在しなかった場合に、クライアントからサーバへの要求に応答して、サーバからデータブロックをフェッチするステップをさらに含む、請求項１６に記載の方法。

【請求項１８】
前記データブロックがバッファ・ツリーに入れられ、ファイル内に階層関係が可能である、請求項１６に記載の方法。

【請求項１９】
前記アクティブデータセットは、少なくともサーバに格納されたデータの一部を含む、請求項１６に記載の方法。

【請求項２０】
前記アクティブデータセットは、特定の時間によって定義される、請求項１６に記載の方法。

【請求項２１】
前記オブジェクトの要求は、オープン・スタンダード・プロトコルを用いて送信される、請求項１６に記載の方法。

【請求項２２】
前記プロトコルは、インターネット・プロトコルを含む、請求項２１に記載の方法。

【請求項２３】
前記プロトコルは、ネットワーク・ファイル・システム（ＮＦＳ）プロトコルを含む、請求項２１に記載の方法。

【請求項２４】
前記プロトコルは、コモン・インターネット・ファイル・システム（ＣＩＦＳ）プロトコルを含む、請求項２１に記載の方法。

【請求項２５】
クライアントによって要求されたデータファイル中のデータブロックの有無を判定する計算の一部として重複しない名前を計算するステップをさらに含む、請求項１６に記載の方法。

【請求項２６】
置換ポリシーに基づいて前記アクティブデータセットの一部を消去するステップをさらに含む、請求項１６に記載の方法。

【請求項２７】
クライアントからオブジェクトの要求を受信する命令と、
要求された前記オブジェクトの各データブロックがキャッシュ内に存在する場合、要求された前記オブジェクトを前記クライアントに送信する命令と、
要求された前記オブジェクト中のデータブロックが欠けていた場合、該データブロックをサーバからフェッチし、該データブロックをアクティブデータセットとして階層関係に埋め込み、要求された前記オブジェクトをクライアントに送信する命令と、
が記録された機械読取可能媒体を含む製品。

【請求項２８】
データをネットワーク内にキャッシュする装置であって、
クライアントからオブジェクトの要求を受信するための受信手段と、

前記受信手段に接続され、要求された前記オブジェクトの各データブロックがキャッシュ内に存在する場合、要求された前記オブジェクトを前記クライアントに送信する手段と、

前記受信手段に接続され、要求された前記オブジェクト中のデータブロックが欠けていた場合、該データブロックをサーバからフェッチし、該データブロックをアクティブデータセットの一部として階層関係に埋め込み、要求された前記オブジェクトを前記クライアントに送信する手段と、

からなる装置。

【請求項２９】

データをネットワーク内にキャッシュする装置であって、

クライアントからオブジェクトの要求を受信し、サーバからデータブロックをフェッチするように構成されたプロキシ・キャッシュを含み、

前記プロキシ・キャッシュは、階層関係の前記前記データブロックをアクティブデータセットとしてオブジェクト内にキャッシュするように構成され、該アクティブデータセットが、前記クライアントからの要求に基づいて定義される、装置。

【請求項３０】

前記アクティブデータセットはデータ置換法を用いて変更される、請求項２９に記載の装置。

【請求項３１】

前記データ置換法は、先入れ先出し法、最長時間未使用法、データ重要度法、および、ユーザ入力を利用したデータ置換法のうちのいずれか１つである、請求項２９に記載の装置。

【請求項３２】

前記プロキシ・キャッシュに接続された全てのクライアントが前記アクティブデータセットを相互共有できるようにするために、前記アクティブデータセットは、前記プロキシ・キャッシュに接続された少なくともう１つのクライアントから利用することができる、請求項２９に記載の装置。

【請求項３３】

前記プロキシ・キャッシュは、ファイルハンドルを変換するように構成され、該ファイルハンドルが特定のサーバにマッピングされていることを示すようにする、請求項２９に記載の装置。

【請求項３４】

前記ファイルハンドルは、ファイルシステム識別子およびサーバ・インターネット・プロトコル・アドレスを利用してハッシュアルゴリズムを用いて変換される、請求項２９に記載の装置。

【請求項３５】

データをネットワーク内にキャッシュする方法であって、

クライアントからオブジェクトの要求を受信するステップと、

要求された前記オブジェクト中の少なくとも１つのデータブロックがキャッシュ内に存在するか否かを判定するステップと、

要求された前記オブジェクト中のデータブロックが欠けていた場合、該データブロックをサーバからフェッチし、該データブロックをキャッシュに割り当て、要求された前記オブジェクトを前記クライアントに送信するステップと、

からなる方法。

【請求項３６】

各データブロックが、オブジェクト内の階層関係に入れられる、請求項３５に記載の方法。

【請求項３７】

前記アクティブデータセットは、少なくとも前記サーバに格納されたデータの一部を含む、請求項３５に記載の方法。

【請求項３８】
前記アクティブデータセットは、特定の時間によって定義される、請求項３５に記載の方法。
【請求項３９】
前記オブジェクトの要求は、オープン・スタンダード・プロトコルを用いて送信される、請求項３５に記載の方法。
【請求項４０】
前記プロトコルは、インターネット・プロトコルを含む、請求項３９に記載の方法。
【請求項４１】
前記プロトコルは、ネットワーク・ファイル・システム（ＮＦＳ）プロトコルを含む、請求項４０に記載の方法。
【請求項４２】
前記プロトコルは、コモン・インターネット・ファイル・システム（ＣＩＦＳ）プロトコルを含む、請求項４０に記載の方法。
【請求項４３】
クライアントによって要求されたデータファイル中のデータブロックの有無を判定する計算の一部として重複しない名前を計算するステップをさらに含む、請求項３５に記載の方法。
【請求項４４】
置換ポリシーに基づいて前記アクティブデータセットの一部を消去するステップをさらに含む、請求項３５に記載の方法。
【請求項４５】
前記データ置換法は、先入れ先出し法、最長時間未使用法、データ重要度法、および、ユーザ入力を利用したデータ置換法のうちのいずれか１つである、請求項４４に記載の方法。
【請求項４６】
前記プロキシ・キャッシュに接続された全てのクライアントが前記アクティブデータセットを相互共有できるようにするために、前記アクティブデータセットは、前記プロキシ・キャッシュに接続された少なくともう１つのクライアントから利用することができる、請求項３５に記載の方法。
【請求項４７】
前記プロキシ・キャッシュは、ファイルハンドルを変換するように構成され、該ファイルハンドルが特定のサーバにマッピングされていることを示すようにする、請求項３５に記載の方法。
【請求項４８】
前記ファイルハンドルは、ファイルシステム識別子およびサーバ・インターネット・プロトコル・アドレスを利用してハッシュアルゴリズムを用いて変換される、請求項４７に記載の方法。
【請求項４９】
前記プロキシ・キャッシュは、クライアントからの要求に応答して重複しないファイル名を計算し、該重複しないファイル名に基づいて、要求された前記オブジェクトの各データブロックが前記プロキシ・キャッシュに格納されているか否かを判定する一致キャッシュファイル識別子を得る、請求項１に記載の装置。
【請求項５０】
前記一致ローカルキャッシュファイル識別子は、適当なｉｎｏｄｅレコードを指し示すｉｎｏｄｅファイルへの索引となる、請求項４９に記載の装置。
【請求項５１】
前記ｉｎｏｄｅレコードは、要求された前記オブジェクトのデータブロックが前記プロキシ・キャッシュに格納されているか否かを示すｉｎｏｄｅ番号を含む、請求項５０に記載の装置。

【請求項５２】
要求された前記オブジェクト中の関連データブロックが前記プロキシ・キャッシュ内に格納されていない場合、ｉｎｏｄｅ番号は特別な値Ｘをとり、該特別な値Ｘは、－１、－３、または、その他の値などの値のうちの１つである、請求項５１に記載の装置。
【請求項５３】
要求された前記オブジェクト中の関連データブロックが前記プロキシ・キャッシュ内に格納されている場合、ｉｎｏｄｅ番号は特別な値Ｘにならない、請求項５１に記載の装置。
【請求項５４】
前記ｉｎｏｄｅファイルは、オブジェクトに関するサーバ側属性を含む第２のｉｎｏｄｅレコードをさらに指し示す、請求項４９に記載の装置。
【請求項５５】
前記重複しないファイル名に基づく一致キャッシュファイル識別子が前記プロキシ・キャッシュ内にない場合、要求された前記オブジェクトのデータブロックは、前記プロキシ・キャッシュ内に格納されていない、請求項４９に記載の装置。
【請求項５６】
前記クライアントからの要求に応答して重複しないファイル名を計算するステップと、
前記重複しないファイル名に基づいて、要求された前記オブジェクトの各データブロックがプロキシ・キャッシュに格納されているか否かを判定する一致キャッシュファイル識別子を得るステップと、
をさらに含む、請求項１６に記載の方法。
【請求項５７】
前記一致ローカルキャッシュファイル識別子は、適当なｉｎｏｄｅレコードを指し示すｉｎｏｄｅファイルへの索引となる、請求項５６に記載の方法。
【請求項５８】
前記ｉｎｏｄｅレコードは、要求された前記オブジェクトのデータブロックが前記プロキシ・キャッシュに格納されているか否かを示すｉｎｏｄｅ番号を含む、請求項５７に記載の方法。
【請求項５９】
要求された前記オブジェクト中の関連データブロックが前記プロキシ・キャッシュ内に格納されていない場合、ｉｎｏｄｅ番号は特別な値Ｘをとり、該特別な値Ｘは、－１、－３、または、その他の値などの値のうちの１つである、請求項５８に記載の方法。
【請求項６０】
要求された前記オブジェクト中の関連データブロックが前記プロキシ・キャッシュ内に格納されている場合、ｉｎｏｄｅ番号は特別な値Ｘにならない、請求項５８に記載の方法。
【請求項６１】
前記ｉｎｏｄｅファイルは、オブジェクトに関するサーバ側属性を含む第２のｉｎｏｄｅレコードをさらに指し示す、請求項５６に記載の方法。
【請求項６２】
前記重複しないファイル名に基づく一致キャッシュファイル識別子が前記プロキシ・キャッシュ内にない場合、要求された前記オブジェクトのデータブロックは、前記プロキシ・キャッシュ内に格納されていない、請求項５６に記載の方法。
【請求項６３】
前記オブジェクトはディレクトリを含む、請求項１に記載の装置。

【発明の詳細な説明】
【技術分野】
【０００１】

本発明の実施形態は概してコンピュータ・ネットワークに関する。詳しくは、本発明の実施形態は概してファイルシステムおよび記憶装置に関する。

【背景技術】

【０００２】

大きな組織は地理的に分散配置されている。それらは、高速のローカル・エリア・ネットワーク（ＬＡＮ）を備えた少数の大型中央サイトを有し、ローカルデータを中央エリアまたは中央システムにまとめて整理して置くことができる。しかしながら、大きな組織は、小さな遠方の事務所、支社、および／または、他の末端位置をさらに有する場合があり、それらから中央集中化されたデータ保管場所までの接続の帯域幅は比較的狭く、遅延も比較的大きい。末端位置には、例えばサーバのバックアップや故障したハードウェアおよび／またはソフトウェアの交換といったローカルデータのニーズに携わるＩＴ（情報技術）スタッフが居ない場合がある。遠方の事務所に居るユーザは、中央サイトに居る相手方に比べて、企業ＩＴインフラストラクチャの能力、信頼性、および、サービスが非常に低いことを知っている。

【０００３】

大型サイト間におけるデータの共有は厄介でもある。大型サイトは通常、専用の広帯域リンクによって相互接続される。しかしながら、サイト間の地理的距離は、多くの用途にとって許容されない遅延を発生させる。

【０００４】

現在の解決策は、各末端位置にファイルサーバ（ネットワーク・アプライアンス・インコーポレイテッドから市販されている「ファイラ」など）を設置し、サーバをワイド・エリア・ネットワーク（ＷＡＮ）を介して中央位置に自動的にバックアップすることである。サイト間で共有される読出し専用データは、同期ミラーリングによって複製される。ハイパーテキスト・トランスファ・プロトコル（ＨＴＴＰ）およびストリーミング・トラフィックが、例えばネットワーク・アプライアンス・インコーポレイテッドから市販されている「ＮｅｔＣａｃｈｅ」のようなネットキャッシュを用いてキャッシュされる。

【０００５】

末端位置におけるファイルサーバの管理は、コストおよび／またはリソースの点で高くつく可能性がある。ＷＡＮを介したデータのバックアップ作業は、慎重な計画と管理を必要とする。ＷＡＮを介して読出し専用ボリュームを複製することは、やり過ぎであることが多い。多くの場合、実際に日常的に使用されるのは、ボリュームのうちの小さな割合の部分（／ｕｓｒ／ｌｏｃａｌ／ｂｉｎなど）だけである。一般に、動的データセット（ユーザのホームディレクトリなど）を複製することはなく、動的データセットは各位置で個別に管理しなければならない。

【０００６】

従って、上記の製品および方法は、特定の能力および機能に制限され、数々の制約を受ける。

【発明の開示】

【課題を解決するための手段】

【０００７】

本発明は一実施形態において、データをネットワーク内にキャッシュする装置を提供する。この装置は、クライアントからオブジェクトの要求を受信し、サーバからデータブロックをフェッチするように構成されたプロキシ・キャッシュを含む。プロキシ・キャッシュは、データブロックをオブジェクト内に階層関係でキャッシュするように構成される。オブジェクトには、例えばデータファイル、ディレクトリ、データブロックのようなデータファイルの一部などがある。プロキシ・キャッシュにキャッシュされたデータブロックは、クライアントからの要求に基づいてアクティブデータセットを定義する。

【０００８】

それらの対策、並びに、説明の進行につれて当業者には明らかとなるであろう付随的な対策および特徴は、添付の図面に例として符号付きで示される装置、アセンブリ、本発明

の実施形態の方法、およびそれらの好ましい実施形態によって達成される。
【発明を実施するための最良の形態】
【０００９】
図面を参照し、本発明の種々の実施形態について説明する。特に断わりがない限り、全部図面を通じて、同じ符号は同じ部品を意味している。
【００１０】
本発明の実施形態を完全に理解してもらうために、本明細書には、構成要素および／または方法の例など、多数の具体的詳細を記載している。しかしながら、本発明が、それらの具体的詳細のうちの１以上が無くても実施できるものであり、他の装置、システム、方法、構成要素、材料、部品などを用いて実施することも可能であることは、当業者には明らかであろう。本発明の実施形態が不明確になることを避けるために、他の例では、周知の構造、材料、処理については図示も説明も詳しくしていない。
【００１１】
図１Ａは、クライアント装置１０５（１）～１０５（ｎ）、サーバ１１０（１）～１１０（ｍ）およびプロキシ・キャッシュ（すなわちプロキシ・アプライアンス）１１５を含むネットワーク１００を示すブロック図である。これらの構成要素は、本発明の一実施形態に従ってフォワード・プロキシ構成を形成している。以下で説明するように、本発明の他の実施形態は、リバース・プロキシ構成の少なくとも１つのプロキシ・キャッシュを含むネットワークを有する。本発明の実施形態は、下記の機能のうちの少なくとも一部を有する。すなわち、提供する。
（１）位置に非依存、および、名前空間の統合
（２）オン・デマンドでまばらな整合のとれたデータの複製
（３）ロード・バランシング
（４）遠隔切断アクセスおよびデータ変更
（５）プロトコル変換
これらの仮想化を適用することにより、従来の方法に関連する法外なコストを伴なうことなく、分散型ストレージの基盤を構築することが可能になる。
【００１２】
上記の変数ｎおよびｍは任意の適当な整数値である。従って、クライアント装置（本明細書では通常、クライアント１０５と呼ぶ）の数と、サーバ（本明細書では通常、サーバ１１０と呼ぶ）の数は、変更してもよい。例えば、ネットワーク１００は、たった１つのクライアント装置１０５、および／または、たった１つのサーバ１１０を用いて実施することもできる。クライアント装置１０５は、ローカル・エリア・ネットワーク（ＬＡＮ）１２０を介してプロキシ・キャッシュ１１５に接続され、サーバ１１０は、ワイド・エリア・ネットワーク（ＷＡＮ）１２５を介してプロキシ・キャッシュ１１５に接続される。
【００１３】
図１Ａに示すフォワード・プロキシ構成によれば、データをプロキシ・キャッシュ１１５にオン・デマンドでキャッシュすることができる。つまり、プロキシ・キャッシュ１１５によるオン・デマンドでまばらな整合のとれたデータの複製が可能となる。このオン・デマンド・キャッシュ処理は、従来の複製方法に比べて効率が高く、従来の複製方法で使用されているような特殊なソフトウェアを必要としないという利点がある。
【００１４】
一実施形態において、プロキシ・キャッシュ１１５とクライアント装置１０５は、ネットワーク・ファイル・システム（ＮＦＳ）プロトコルのようなオープン・スタンダード・プロトコル１３０を用い、ＬＡＮ１２０を介して互いに通信することができる。以下で説明するように、オープン・スタンダード・プロトコル１３０は、コモン・インターネット・ファイル・システム（ＣＩＦＳ）プロトコルのような他の適当なオープン・スタンダード・プロトコルであってもよい。プロキシ・キャッシュ１１５とサーバ１１０は、ＮＦＳのようなオープン・スタンダード・プロトコル１３５を用い、ワイド・エリア・ネットワーク（ＷＡＮ）１２５を介して互いに通信することができる。本発明の一実施形態は、Ｗ

ＡＮ１２５およびＬＡＮ１２０にオープン・スタンダード・プロトコルを使用しているので、従来の方法で必要とされる特殊なソフトウェアが不要であるという利点を有する。特に、ＮＦＳのようなオープン・スタンダード・プロトコルを使用することにより、クライアント１０５として異種のクライアントを使用することが可能となる。言い換えれば、「異種のクライアント」という用語は、異なる製造業者またはベンダーで製造された種々のクライアントを都合よく使用出来ることを意味している。クライアントは、サーバ１１０と通信するための特殊なソフトウェアを必要としない。

【００１５】

さらに、オープン・スタンダード・プロトコルを使用すると、プロキシ・キャッシュ１１５をネットワーク１００内に設置するときに通常ならば必要となるクライアント装置１０５の設定変更も不要になる。従って、プロキシ・キャッシュ１１５を使用すると、ネットワーク管理に関するオーバヘッドおよびコストを低減することができる。なお、プロキシ・キャッシュ１１５は、ネットワーク１００内のサーバ１１０やクライアント１０５から離れた場所から管理したり、それらから離れた場所に設置してもよい。

【００１６】

通常、プロキシ・キャッシュ１１５は、プロキシ・キャッシュ１１５に接続されたサーバ１１０およびクライアント１０５を識別するとともに、サーバ１１０に格納された特定のデータファイルを識別することができる。

【００１７】

例えば、クライアント１０５がデータを要求したときに、そのデータがプロキシ・キャッシュ１１５になければ、「キャッシュ・ミス」が発生する。一方、要求されたデータがプロキシ・キャッシュ１１５にあれば、「キャッシュ・ヒット」が発生する。これらの処理については、図１Ｂおよび図１Ｃを参照して後で詳しく説明する。

【００１８】

キャッシュ・ヒットが発生すると、プロキシ・キャッシュ１１５は、要求されたデータを要求元クライアント１０５に送信する。キャッシュ・ミスが発生すると、プロキシ・キャッシュ１１５は、データをサーバ１１０に要求し、次いでそのデータを要求元クライアント１０５に渡す。サーバ１１０から得られたデータは、アクティブデータセット１４０としてプロキシ・キャッシュ１１５内にキャッシュされ、他のクライアント１０５から直ぐに利用できるようにされる。アクティブデータセットは、クライアントおよびクライアントのアプリケーションの属性であり、所与の時間（すなわち時間窓）内に参照され、キャッシュされたデータである。従って、アクティブデータセットは、所与の時間に応じて異なる場合がある（例えば、所与の時間を２分としたときのアクティブデータセットは、所与の時間を１日または１週間としたときのアクティブデータセットとは異なる場合がある）。

【００１９】

プロキシ・キャッシュ１１５は、キャッシュした複製データを複数のクライアント１０５間で相互共有させることができる。この相互共有状態は、クライアント１０５からサーバ１１０に要求された特定のデータが、少なくとも１つの他のクライアント１０５によってさらに要求される可能性が高いことを前提としている。

【００２０】

一実施形態において、プロキシ・キャッシュ１１５はアクティブデータセット１４０をキャッシュする。アクティブデータセット１４０とは、最近要求されたデータセットまたは要求頻度の高いデータセットであって、プロキシ・キャッシュ１１５からまだ消去されていないデータセットである。アクティブデータセット１４０は、サーバ（複数の場合もあり）１１０に格納されているデータのサブセットである。クライアント１０５がサーバ１１０からデータファイルの特定のコピー１５０を読み出すために読出し要求１４５を送ると、その読出し要求１４５はプロキシ・キャッシュ１１５に受信され、そこでそのファイルまたはフォルダの特定部分（例えば、データブロックなど）が、プロキシ・キャッシュ１１５内にローカルにキャッシュされているか否かがチェックされる。要求された特定

のデータファイルがアクティブデータセット１４０内にあった場合、キャッシュ・ヒット状態が発生し、プロキシ・キャッシュ１１５は、要求されたデータをＬＡＮ１２０を介して要求元クライアント１０５に送信する。

【００２１】

一方、要求されたデータファイル中の或るデータブロックがアクティブデータセットの一部として格納されていなかった場合、キャッシュ・ミス状態が発生する。すると、プロキシ・キャッシュ１１５は、要求１５５をサーバ１１０に送り、要求されたデータファイル中の欠けているデータブロックのコピーを得る。

【００２２】

一実施形態において、キャッシュされたファイルは、アクティブデータセット１４０内でそのファイルが適当な置換方法（例えば、データの先入れ先出し管理や、最長時間未使用（ＬＲＵ）アルゴリズムなど）によって置換されるまで、アクティブデータセット１４０の一部として保持される。当然ながら、以下で説明するように、一実施形態において、アクティブデータセット１４０の管理に関しては、他のデータ置換方法を使用してもよい。例えば、あるファイルを永久記憶用として指定し、クライアント１０５のユーザがアクティブデータセット１４０からそのファイルを抹消（消去）するコマンドを発行するまで、そのファイルを保持するようにしてもよい。

【００２３】

図１Ｂは、本発明の一実施形態によるプロキシ・キャッシュ１１５を示すブロック図である。プロキシ・キャッシュ１１５は、ＩＰを利用したネットワーク・トラフィック（または、ファイバ・チャネルやストレージ・エリア・ネットワークなどに関する他のタイプのトラフィック）を構文解析するための、ネットワーク・インタフェース１６１、ネーミング仮想化層１６２、ＮＦＳサービス１６３、および／または、ＣＩＦＳサービス１６４、ローカル・アクセス層１６５、ＮＦＳフィルエンジン１６６、および／または、ＣＩＦＳフィルエンジン１６７、および／または、他の適当なタイプのフィルエンジン、リモート・フィル層１６８、ファイルシステム層１６９（例えば、Write-Anywhere-File-Layout、すなわちＷＡＦＬなど）、ストレージデバイス・マネージャ１７０（Redundant Array Of Independent（or Inexpensive）Disks層、すなわちＲＡＩＤ層など）、および、ストレージ・ディスク（複数の場合もあり）１７１を含む。

【００２４】

ネットワーク・インタフェース１６１は、クライアント１０５からストレージ関係のサービス要求を受信するためのコンポーネントを含む。

【００２５】

一般にファイルシステムは、情報をディレクトリとファイルの階層構造としてストレージデバイス（例えば、ディスクなど）上に論理編成することができる。（ディスク上の）各ファイルは、テキストなどの情報を格納するように構成されたディスクブロックの集合として実施されるのに対し、ディレクトリは、他のファイルやディレクトリに関する情報が格納された特殊形式のファイルとして実施される。

【００２６】

ストレージデバイス・マネージャ１７０は、ストレージシステム内のストレージデバイス１７１を管理する。ストレージデバイス・マネージャ１７０は、ファイルシステム１６９から読出しコマンドおよび書込みコマンドを受信し、それらのコマンドを処理することによって、ストレージシステムにアクセスする。ストレージデバイス・マネージャ１７０は、ファイルシステム１６９からブロックの論理アドレスを得て、その論理アドレスをストレージシステム内の１以上のストレージデバイス１７１における物理アドレスに変換する。一実施形態において、ストレージデバイス・マネージャ１７０は、ストレージデバイスをＲＡＩＤ（Redundant Array of Independent, or Inexpensive, Disks）に従って管理する。

【００２７】

一般に、ディスク・ストレージは大抵、記憶空間の全体的論理構成を定義している物理

ストレージディスクによって形成された１以上のストレージ「ボリューム」として実施される。各ボリュームは通常、そのボリューム独自のファイルシステムに関連しているので、「ボリューム」という用語と「ファイルシステム」という用語は、同じ意味で使用される。ボリューム内のディスクは通常、１以上のＲＡＩＤグループとして編成される。

【００２８】

図１Ｂに示す他のモジュールの機能については、図１Ｃを参照して説明する。

【００２９】

キャッシュ・ヒット

次に、図１Ｂおよび図１Ｃのブロック図を参照し、本発明の一実施形態によるプロキシ・キャッシュ１１５の動作について説明する。ファイルハンドルは、ファイルシステム内のファイル等のオブジェクトに名前を付けるのに使用される。ファイルハンドルについては、図４を参照して後で詳しく説明する。

【００３０】

プロキシ・キャッシュ１１５（図１Ｂ）が、クライアント１０５から、例えばＦＨ＝「ＦＩＬＥ１　ＯＮ　ＣＡＣＨＥ」の値を有するファイルハンドル１８０を含む読出し要求を受信したものと仮定する。すると、ネットワーク・インタフェース１６１はその要求をネーミング仮想化層１６２に転送し、仮想化層が「ＦＩＬＥ１」を、例えばＩＰ（インターネット・プロトコル）アドレス１０．５６．２０．３４を有するサーバ１１０上のサーバ・ファイルハンドルＦＨ＝「ＦＩＬＥ１　ＯＮ　ＳＥＲＶＥＲ１」にマッピングする。ローカル・アクセス層１６５は、ファイルサーバＩＤ（ＦＳｉｄ）値１８１およびサーバＩＰアドレス１０．５６．２０．３４に基づいて重複しない（一意の）名前１８２を計算する。一実施形態において、重複しない名前を計算するアルゴリズムはＭＤ５ハッシュアルゴリズムである。ＭＤ５ハッシュアルゴリズムはメッセージを受け取ると、それをメッセージ・ダイジェストと呼ばれる固定桁数の文字列に変換する。また、要求１７２がＮＦＳ要求であるかＣＩＦＳ要求であるかに応じて、ＮＦＳサービス層１６３またはＣＩＦＳサービス層１６４が、その要求１７２を構文解析する機能を提供する。

【００３１】

ファイルシステム層１６９は、重複しない名前１８２に基づいて検索機能を実施し、ローカル・キャッシュファイルＩＤ１８５を得る。一実施形態において、この検索機能は、ファイルシステム内のモジュール１８４によって実施される。一致するローカル・キャッシュファイルＩＤが無ければ、キャッシュ・ミスが発生する。キャッシュ・ミスについては、後で詳しく説明する。

【００３２】

ローカル・キャッシュファイルＩＤ１８３は、適当なｉｎｏｄｅレコード１８６を指し示す、ｉｎｏｄｅファイル１８５への索引である。ｉｎｏｄｅファイル１０５のｉｎｏｄｅレコードは、所与のファイルシステムに関連するｉｎｏｄｅファイルを表わす情報を有する。一般に、ｉｎｏｄｅレコードは、ファイルに関するメタデータ（属性）などの情報の格納に使用されるデータ構造であり、ファイル・データブロックは、そのファイルの実際のデータの格納に使用される構造である。ｉｎｏｄｅレコードに格納される情報には、例えば、ファイルの所有者、ファイルのアクセス・パーミッション、ファイルのサイズ、ファイルタイプ、ファイルのデータブロックのディスク上の位置に対する基準などがある。ｉｎｏｄｅファイル１８５のｉｎｏｄｅレコード１８６は、まとめて符号１８８で示すような、ファイルシステム・データブロック（例えば、ＷＡＦＬデータブロックなど）へのポインタを有している。ファイル・データブロックは、ファイルシステムによって処理されるデータの最小アドレス指定可能量として定義される。ファイル・データブロックは、例えば４キロバイト（ＫＢ）のデータを格納する容量を有する。ｉｎｏｄｅレコード１８６は間接ブロックを指し示すことがあり、間接ブロックは他のファイル・データブロックまたは他の間接ブロックをさらに指し示すことがある。例えば、間接ブロック１８８ｄはブロック１８９ａ、１８９ｂ、１８９ｃを指し示す場合があり、それらのブロックはそれぞれ、ファイル・データブロックまたは間接ブロックである。

【００３３】
具体的には、ｉｎｏｄｅレコード１８６内のｉｎｏｄｅ番号１８７は、ファイル・データブロックまたは間接ブロックを指し示す。また、ｉｎｏｄｅレコード１８６は、プロキシ・キャッシュ１１５にローカルに格納されたファイルの属性１９０をさらに含む。さらに、一実施形態において、ローカル・キャッシュファイルＩＤ１８３は、ファイルのサーバ側属性を含む適当なｉｎｏｄｅレコード１９３を指し示す、第２のｉｎｏｄｅファイル１９２への索引でもある。サーバ側属性には、例えば、ファイルシステムＩＤ、ファイルＩＤ、ブロックサイズ、ハードリンクの数、ファイルシステム上の空き空間などがある。当然ながら、第２のｉｎｏｄｅファイル１９２は、ｉｎｏｄｅファイル１８５に結合させてもよい。第２のｉｎｏｄｅファイル１９２を作成する利点は、図１Ｃを参照して説明した機能の一部を実施するための第１のｉｎｏｄｅファイル１８５の変更が、不要になることである。

【００３４】
また、ｉｎｏｄｅレコードは、ファイルシステム内のディスク上のファイルの表現に使用されるデータ構造であるファイルバッファ・ツリーを含む。図１Ｃでは、ブロック番号１８７およびブロック１８８，１８９により、ファイルバッファ・ツリー１９１が形成されている。

【００３５】
次に、ＩＯベクトル１９４を形成する。ＩＯベクトルは、ファイルバッファ・ツリー１９１のブロック番号１８７のリストである。ブロック番号１８７は、特定ファイルについてファイル・データブロックの有無を示す。一実施形態において、モジュール１９５はＩＯベクトル１９４を形成することができ、ファイル・データブロックの有無を判定する。一実施形態において、ブロック番号１８７は、ブロック番号１８７ａに示すような特殊な値Ｘ（Ｘは例えば－１または－３をとりうる）を有する。この特殊な値Ｘは、プロキシ・キャッシュ１１５に格納されたローカルキャッシュファイルが、要求されたデータブロックを有していないことを示す値である。

【００３６】
キャッシュ・ヒット状態になると、ブロック番号は特殊な値Ｘではなくなり、ＩＯベクトルがストレージデバイス・マネージャ１７０（例えば、ＲＡＩＤ層など）に送信される。ストレージデバイス・マネージャ１７０は、適当なストレージ・デバイス（複数の場合もあり）へのポインタを生成し、要求されたデータブロックをファイルシステムに送信するとともに、ＮＦＳサービス（ＮＦＳ要求の場合）に送信する。そしてＮＦＳサービスは、ＮＦＳ応答１７３を要求元クライアント１０５に対して生成する。従って、要求元クライアント１０５は、要求したファイルを受信することができる。上記の方法は、ディレクトリ要求の処理に使用することもできる。

【００３７】
ブロック番号リスト中に特殊な値Ｘを使用すると、要求されたファイルの欠けているデータブロックを有するプロキシ・キャッシュ（のストレージデバイス）におけるバッファの追跡が可能になり、従って散在するデータの追跡が可能になる。データブロックまたはキャッシュファイルはプロキシ・キャッシュ１１５から消去される場合があるので、（重複しない名前１８２を計算することによる）間接レベルによれば、特定の時刻に応じてファイルハンドルをｉｎｏｄｅファイル内の異なるスロットを指し示すものにすることができる。

【００３８】
キャッシュ・ミス
ＩＯベクトル１９４を作成した場合、ファイルバッファ・ツリー１９１のブロック番号が、要求されたファイル・データブロックが（ストレージデバイス１７１内の）バッファ内に無いことを意味する特殊な値Ｘであれば、キャッシュ・ミス状態が発生する。あるいは、重複しない名前１８２を計算した後、ファイルシステム１６９がテーブル検索機能を実施したときに一致するファイルＩＤ１８３が見付からなければ、キャッシュ・ミス状態

が発生する。

【００３９】

ＮＦＳ要求１９６のためのＮＦＳフィルエンジン１６６（またはＣＩＦＳ要求のためのＣＩＦＳフィルエンジン）は、不在のファイル・データブロックをそのデータブロックを有するサーバ１１０に要求する。図１Ｃの例では、不在のファイル・データブロックが破線ボックス１１８ｃとして図示されている。上記の例では、サーバ・ファイルハンドル「ＦＩＬＥ１　ＯＮ　ＳＥＲＶＥＲ１」を用いてその要求が送信される。

【００４０】

ＮＦＳフィルエンジン１６６がサーバ１１０からファイル・データブロックをフェッチ（１９７）すると、そのデータブロックは、リモート・フィル層１６８、ファイルシステム１６９およびストレージデバイス・マネージャ１７０によってストレージ・ディスク１７１に入れられる。そしてバッファ・ツリー１９１のブロック番号が更新され、ファイル・データブロックが割り当てられる。次に、それらのファイル・データブロックが要求元クライアント１０５に送信され、要求されたファイルがそのクライアントに与えられる。上記の方法は、ディレクトリ要求の処理にも使用することができる。

【００４１】

また、図１Ｃに示す実施形態によれば、バッファ・ツリー内のファイル・データブロックを１つのファイル内にまばらにキャッシュすることができる。この方法は、ファイルの途中における丸め処理や書込み処理のような、部分的なファイル処理を実施することが可能になるという有利がある。

【００４２】

通常、クライアント１０５がＮＦＳプロトコルで要求を送信する場合、各データブロックについて個別の要求が作成される。また、ＣＩＦＳプロトコルの場合にも、各データブロックについて個別の要求が作成される。例えば、第１のクライアント１０５（例えば、図１のクライアント１０５（１））がデータブロック１８８ａおよび１８８ｂを求めるデータ要求を送信するものと仮定する場合、それらのデータブロックは両方とも図１Ｃの例におけるプロキシ・キャッシュ１１５にキャッシュされている。クライアントからのデータ要求の数は変更することもできる点に注意して欲しい。データブロック１８８ａおよび１８８ｂがプロキシ・キャッシュ１１５にキャッシュされているので、プロキシ・キャッシュ１１５がクライアント要求を受信するとキャッシュ・ヒット状態が発生し、要求されたデータブロック１８８ａおよび１８８ｂが、プロキシ・キャッシュ１１５から要求元クライアント１０５（１）に送信される。

【００４３】

キャッシュ・ミス状態の例として、第２のクライアント１０５（例えば、図１のクライアント１０５（２）など）が、データブロック１８８ａ、１８８ｂおよび１８８ｃのデータを求める要求を送信する場合を仮定する。図１Ｃの例では、プロキシ・キャッシュ１１５がクライアント要求を受信したとき、データブロック１８８ａおよび１８８ｂはプロキシ・キャッシュ１１５にキャッシュされているが、データブロック１８８ｃはプロキシ・キャッシュ１１５内に存在しない。プロキシ・キャッシュ１１５はこのキャッシュ・ミス状態に応答して、上記と同様にサーバ１１０からデータブロック１１８ｃをフェッチする。フェッチされたデータブロック１８８ｃは、プロキシ・キャッシュ１１５に割り当てられた後、プロキシ・キャッシュ１１５から要求元クライアント１０５（２）に送信される。

【００４４】

さらに他の例として、他のクライアント１０５（例えば、図１のクライアント１０５（ｎ））がデータブロック１８８ｂおよび１８８ｃを求めるデータ要求を送信する場合を仮定する。データブロック１８８ｃはサーバ１１０からプロキシ・キャッシュ１１５に前もってフェッチされ、プロキシ・キャッシュ１１５に割り当てられているので、プロキシ・キャッシュ１１５がクライアント要求を受信したときに、プロキシ・キャッシュ１１５はデータブロック１８８ｂおよび１８８ｃを有している。従って、キャッシュ・ヒット状態

が発生し、要求されたデータブロック１８８ｂおよび１８８ｃが、プロキシ・キャッシュ１１５から要求元クライアント１０５（１）に送信される。クライアント１０５がプロキシ・キャッシュ１１５内に存在しないデータブロック１８８を要求した場合、プロキシ・キャッシュ１１５は、その存在しないデータブロック１８８をサーバ１１０からフェッチし、キャッシュしてから、そのフェッチしたデータ１８８を要求元クライアント１０５に送信する。

【００４５】

このように、プロキシ・キャッシュ１１５は、クライアント要求に応答して部分オブジェクトの複製を行なう。部分オブジェクトは、ファイルの１以上のデータブロックとして定義される。上記の例では、クライアント要求に応答してデータブロック１８８ｃをプロキシ・キャッシュ１１５に複製した後、それを他の要求元クライアント１５０から使用できるようにしていた。以下で説明するように、適当なデータ置換ポリシーを用いて、（プロキシ・キャッシュ１１５から）データブロック１８８、または複数のデータブロック１８８によって定義されるファイルを消去することができる。これに対し、従来の複製技術は、全てのファイルを複製している（例えば、種々のミラーリング技術は、全てのボリュームまたは全てのファイルセットを複製している）。

【００４６】

置換ポリシー

プロキシ・キャッシュ１１４においてアクティブデータセット１４０内のデータファイルを更新、維持、または消去するための置換ポリシーには、様々なものがある。図２は、アクティブデータセット１４０内にキャッシュされたデータの重要度を、そのキャッシュされたデータに対するクライアント（複数の場合もあり）からのアクセスすなわち要求の回数に基づいて増加させる方法を示すグラフ２００である。従ってプロキシ・キャッシュ１１５は、アクティブデータセット１４０内にキャッシュされたデータに対し、そのキャッシュされたデータに対するクライアント１０５からのアクセスまたは要求の回数に基づいて「重要度」値を動的に割り当て、調節するように構成される。プロキシ・キャッシュ１１５は、アクセス数すなわち要求回数が増えるのに従って、キャッシュされたデータの重要度を増加させる。例えば、プロキシ・キャッシュ１１５は、アクティブデータセット１４０内にキャッシュされたデータが重要度値Ｙ（図２のグラフに点２０５で示す）に達した場合、プロキシ・キャッシュ１１５のストレージユニットから、そのキャッシュ・データが消去されないようにする。

【００４７】

図３は、本発明の一実施形態による、アクティブデータセット１４０を管理する他の方法３００を示すブロック図である。上記のように、アクティブデータセット１４０は、先入れ先出し（ＦＩＦＯ）法３０５などの適当な方法により、置換、またはプロキシ・キャッシュ１１５から消去することができる。ＦＩＦＯ法３０５では、プロキシ・キャッシュ１１５のストレージユニットから消去する際に、アクティブデータセット１４０を循環させる。最長時間未使用（ＬＲＵ）法などの他の適当な置換ポリシーを使用することもできる。ＬＲＵによる置換方法は、一般にデータベース管理システムで使用される方法で、最も長時間使用されなかったブロックを最初に置換すべきブロックとする方法である。

【００４８】

代替または追加として、アクティブデータセット１５０内のファイルは、ユーザがクライアント１０５からプロキシ・キャッシュ１１５へロックコマンドを送信することにより、ロック（３１０）することもできる。ロックコマンドは、プロキシ・キャッシュからそのファイルが消去されないようにするためのものである。ユーザは、例えば、サーバ１１０までのリンク接続が壊れているかもしれないとか、サーバ１１０が故障しているかも知れないといったことが心配である場合に、ロックコマンドを送信する場合がある。代替として、オブジェクト（例えば、ファイルや、ファイルのデータブロックなど）の属性（メタデータ）は、そのオブジェクトがプロキシ・キャッシュ内にどのくらいの期間キャッシュされているのかを示すことができる。

【００４９】
代替または追加として、ユーザは、ヒントまたはインジケータをプロキシ・キャッシュ１１５に与え（３１５）、特定のキャッシュ・ファイルを重要なものとして指定することもできる。その結果、プロキシ・キャッシュ１１５のストレージユニット（複数の場合もあり）から指定された重要ファイルが消去されることが、インジケータによって防止される。
【００５０】
プロキシ・キャッシュ１１５を使用すると、データを様々な遠くの場所に分散および複製することができ、データ複製のための従来のミラーリング技術の使用を回避できるという利点がある。従来のミラーリング技術を使用するためには、所定の時刻または所定の時間間隔（例えば毎晩）で全てのデータセットをミラーリングしなければならず、全てのデータセットを入れるディスク空間が必要となる。これに対し、プロキシ・キャッシュ１１５は、アクティブデータセット１４０内のデータをオン・デマンドで複製する。アクティブデータセット１４０は、従来のデータ複製方法に関する大きなディスクの必要性を無くすことができるという利点を有する。さらに、アクティブデータセット１４０はキャッシュ・ミスに応答して自動的に更新または消去されるので、従来のデータ複製方法に関する特殊なソフトウェアの必要性も無くすことができる。
【００５１】
マウントポイントの統合とファイルハンドルの書換え／変形の方法
ＮＦＳネットワーク・ファイルシステム・プロトコルによって処理を実行するために、クライアントは、ＮＦＳ要求を下記の（１）～（３）とともにＮＦＳサーバに送信する。
（１）処理対象を指定するためのＮＦＳファイルハンドル
（２）処理（検索、読出し、書込み、パーミッション変更など）
（３）要求を送信する代表となるユーザ
ＮＦＳクライアントがリモートのファイルシステムに最初にアクセスしようとする場合、クライアントはまず、ボリューム（上で述べたように、「ボリューム」という用語と「ファイルシステム」という用語は、同じ意味で使用される）のエントリ・ポイントのファイルハンドルであるルート・ファイルハンドルを入手しなければならない。その目的のために、クライアントホストは、マウント要求をサーバのマウントデーモンに送信する。ここで、マウント要求とは、ルート・ファイルハンドルにアクセスするためのプロトコルの一部であり、デーモンとは、コンピュータシステムが受信する可能性のある定期的なサービス要求を処理するために、絶え間なく動作し続け、存在するプログラムである。デーモンプログラムは、必要に応じてその要求を他のプログラム（またはプロセス）に転送する。サーバのマウントデーモンは、要求されたファイルシステムにアクセスするためのパーミッションをクライアントが有していることを確認する。マウントデーモンは、アクセスを許可する場合、ファイル（ディレクトリ）ハンドルをそのＮＦＳクライアントに返送する。通常、各ファイルハンドルは、３２バイトの不可視の識別データである。ファイル名が変更されても、そのファイルハンドルは、リネーム後のファイルのファイルハンドルとしてそのまま維持される。従って、上記のマウント処理は、ルート・ファイルハンドルにアクセスするためのプロトコルである。
【００５２】
２つの異なるサーバが同じファイルハンドルを使用することもありうるので、ＮＦＳのファイルハンドルは、世界的に／いかなる場合にも、重複しないものではない。従来は、この条件が問題となることはなかった。その理由は、各サーバにどのファイルハンドルが関連しているのかを、クライアントが常に管理しているからである。プロキシ・キャッシュ１１５を使用して複数のマウントポイントを統合する場合、プロキシ・キャッシュ１１５は、多数の異なるサーバから得たボリュームをエクスポートする場合がある。それらのファイルハンドルは必ずしも重複していないとは限らないので、プロキシ・キャッシュ１１５は、どのファイルハンドルがどのサーバから得られたものであるのかを判断できない場合があり、第１のサーバにおける第１のファイルと第２のサーバにおける第２のファイ

ルが同じファイルハンドルを有することもあり、その状態は衝突問題を引き起こす可能性がある。この問題を悪化させる原因は、ファイルハンドルが不可視なものとして定義されていることにある。ここで、不可視とは、ファイルハンドルの内容を知ることができず、ファイルハンドルの名前だけしか得られないことを意味している。

【００５３】

本発明の一実施形態では、ファイルハンドルをクライアント１０５に送信する前に、ファイルハンドルを変形することにより、プロキシ・キャッシュ１１５がファイルハンドルを解釈してファイルハンドルの宛先サーバ１０５を判断できるようにする。従ってクライアント１０５は、データの入手元であるサーバを明確に知らなくても、プロキシ・キャッシュ１１５を介してデータを得ることができる。

【００５４】

図４に示すように、ファイルハンドル４００は、FS_{id}（ファイルシステム識別子）フィールド４０５、ファイル識別子（ＩＤ）フィールド４１０、生成番号フィールド４１５、および、その他データ用フィールド４２０などの種々のフィールドを有する。通常、FS_{id}フィールド４０５は約４～８バイトである。

【００５５】

本発明の一実施形態によれば、サーバ側ファイルハンドルの仮想化が可能になる。クライアント１０５とサーバ１１０との間にプロキシ・キャッシュ１１５を導入すると、ファイルハンドルを変形（変換）し、世界的に重複しない名前空間を構成することが可能になる。クライアントに返送されるファイルハンドルは、それらが種々の入手元サーバまたは種々のエクスポートオプションセットにマッピングされること、すなわち、複数のマウントポイントを交差させるものであることを示すものに変更される。クライアントに送信されるファイルハンドルと、入手元サーバ上のオブジェクトのアクセスに使用されるファイルハンドルとの間に間接層を追加することにより、クライアント１０５に何も影響を与えることなく、入手元サーバ１１０に対する変更を行なうことができる。

【００５６】

図５および図６は、本発明の一実施形態による、マウントポイントを統合する方法およびファイルハンドルを書換え、すなわち変換する方法を示す図である。図５に示すように、各サーバ１１０は通常、ファイルシステムに相当するボリューム（例えば、ボリューム／Ｖｏｌ１、ボリューム／Ｖｏｌ２、ボリューム／Ｖｏｌ３など）を含む。つまり、サーバ１１０（１）はボリューム５０５～５１５を含み、サーバ１１０（２）はボリューム５２０～５３０を含む場合がある。

【００５７】

本発明の一実施形態では、図６に示すように、特定のサーバ１１０の各ボリュームについてFS_{id}値のマッピング６００を作成し、そのマッピング６００をプロキシ・キャッシュ１１５に格納する。あらゆるFS_{id}値を変換することで、キャッシュ・ミスが発生したときに、あるファイルがどのサーバに格納されているかを判定することが可能になるという利点がある。

【００５８】

変換テーブル６０５は、ハッシュ値Ｎ（例えば、Ｎ１、Ｎ２、Ｎ３、・・・など）を有し、FS_{id}テーブル６１０は、サーバ１１０の各ボリュームについてFS_{id}値を有する。ファイルハンドル中に特定のFS_{id}値を有するサーバ１１０（例えば、図６の例の場合、FS_{id}＝値１を有するサーバ１１０（１）のボリューム５０５など）からトラフィック６１５が受信されると、そのFS_{id}＝値１は、プロキシ・キャッシュ１１５のテーブル６０５のFS_{id}値Ｎ１に変換される。サーバ１１０から受信された他のファイルハンドル中の他のFS_{id}値は、テーブル６１０に格納され、テーブル６０５内の様々な値に変換、すなわち変形される。従って、プロキシ・キャッシュ１１５は、クライアント１０５からファイルハンドル６２０を受信したとき（クライアントのデータ要求の際）にキャッシュ・ミスが発生すると、テーブル６１０にＮの値およびFS_{id}の値を有するマッピング６００を用いて、そのファイルハンドル６２０を送信すべき特定のサーバ１１０

およびファイル／ボリュームを判定する。プロキシ・キャッシュ１１５は、ファイルハンドル６２０を宛先サーバ１１０に送信する前に、ファイルハンドル６２０の変換後のＦＳ$_{id}$値Ｎをテーブル６１０中の適当なＦＳ$_{id}$値に変換する。

【００５９】

テーブル６００における値Ｎは、ボリュームおよび／または他の識別子を有するサーバ１１０のアドレスから何らかのハッシュを利用して計算される。各値Ｎに関するハッシュは重複しないので、上記のような名前衝突問題は起こらない。値Ｎは通常、８バイトの数値であり、サーバ名、ボリューム名およびファイル名（例えば、ファイラ１/ボリューム０/ファイル０など）によって決まる設定値を有する。

【００６０】

あるいは、値Ｎは、上で述べたように、サーバのインターネット・プロトコル（ＩＰ）アドレスおよびサーバのファイルハンドル値からＭＤ５アルゴリズムを用いて計算することもできる。ファイルハンドル値は種々のサーバ間で重複しない値であるとは限らないので、サーバＩＰアドレスを用いて作成すれば、ファイルに関する各Ｎの値をサーバ間で異なる値にすることができる。

【００６１】

図７に示すように、マッピング６００は様々なプロキシ・キャッシュ（例えば、プロキシ・キャッシュ１１５と７０５）に複製することができる。そのため、あるプロキシ・キャッシュが故障した場合でも、他のプロキシ・キャッシュを利用することができる。例えば、装置故障によりクライアント１０５がプロキシ・キャッシュ１１５にアクセスできなくなった場合、プロキシ・キャッシュ７０５へのフェイルオーバー７１０を実施することができ、その結果、クライアント１０５はファイルハンドル４００の書換え用のマッピング６００を用いてプロキシ・キャッシュ７０５にアクセスできるようになる。

【００６２】

マッピング６００は複数のプロキシ・キャッシュに複製することができるので、代替または追加として、クライアント１０５がボリュームをアンマウント・リマウントしたり、他の変更をクライアント１０５に施したりしなくても、プロキシ・キャッシュ（例えば、プロキシ・キャッシュ１１５など）は、新たなプロキシ・キャッシュまたは他のプロキシ・キャッシュ（例えば、プロキシ・キャッシュ７０５など）に交換することができる。

【００６３】

世界的に重複しない仮想名前空間を作成して使用する方法

本発明の一実施形態によれば、上記のようなマウントポイント間の統合が可能になるだけでなく、世界的に重複しない名前空間の作成も可能になる。従来は、ＮＦＳのマウントポイントをネストさせること（すなわち、クライアントがＮＦＳボリュームをマウントするときに、そのボリュームに他のマウントポイントを含めること）が出来なかった。この制約は、ネットワークのユニフォーム・ビューの作成を非常に難しいものにしていた。

【００６４】

本発明の一実施形態によるファイルハンドルの仮想化を使用すると、管理者は、ＮＦＳによってエクスポートされたボリューム内の任意のディレクトリをマウントポイント（ファイルシステムに対するアクセスポイント）として設定することが可能になる。これは、クライアント１０５が１つのボリュームをマウントするだけで、誰かがそのボリュームにアクセスしたときに、そのクライアント要求がプロキシ・キャッシュによって特定の適当なボリュームに適当に宛先変更されることを意味する。従って、プロキシ・キャッシュ１１５は、単一のマウントポイントを用いて、各サーバ１１０（または新たに追加されたサーバ）についてあらゆるクライアント１０５に通知することができる。オブジェクトをリネームすることにより、ファイルの仮想グローバル・ビュー（すなわち、仮想グローバル名前空間）が作成される。その結果、各クライアント１０５は、ファイルの仮想グローバル・ビューを用いて、各サーバ１１０に関する情報および各サーバ１１０の各ファイルに関する情報を得ることができる。仮想グローバル名前空間を使用すると、クライアント１０５の管理が簡単になる。

【００６５】
図８Ａに示すように、（プロキシ・キャッシュ内の）ボリューム名８０５は、複数のクライアント１０５からアクセスすることができる。ボリューム名の例は、例えば「／ｇｌｏｂａｌ」である。ボリューム名８０５は、本発明の一実施形態による仮想名前空間を可能にする。クライアント１０５は、ボリューム名８０５、並びにサーバ１１０（１）、１１０（２）、１１０（ｍ）にそれぞれ関連するビュー・フォルダ８１０、８１５、８２０ｍをマウントすることができる。各フォルダ８１０、８１５、８２０ｍには、特定サーバに割り当てられたボリューム名８０５が格納される。従ってボリューム名８０５は、サーバ１１０（１）の／ｖｏｌ５０５、サーバ１１０（１）の／ｖｏｌ５１０、サーバ１１０（１）の／ｖｏｌ５１５、サーバ１１０（２）の／ｖｏｌ５２０、サーバ１１０（２）の／ｖｏｌ５２５、サーバ１１０（２）の／ｖｏｌ５３０、およびサーバ１１０の他の既知のボリュームを指し示す。従って、フォルダ８０５～８２０ｍは、ボリューム名８０５によってマウントポイントに仮想的にマッピングされる。クライアントは、１つのマウントポイント（例えば、／ｇｌｏｂａｌなど）を知っているだけで、適当なサーバ１１０にアクセスすることができる。マッピング６００（図６を参照）を用いると、ＮＦＳプロトコルにおけるファイルハンドルの書換えにより、ファイルハンドル４００をサーバ１００のボリューム内の仮想アクセスポイントとして使用することが可能になる。上記のように、マウント処理は、サーバと通信し、ファイルシステムのルート・ファイルハンドルを得ることを含む。このルート・ファイルハンドルは、後に検索リモート・プロシージャ・コール（ＲＰＣ）に渡され、リモートファイルシステムのディレクトリ階層中の他のファイルハンドルを見付けるのに使用される。

【００６６】
単一マウントポイントによって得られる利点は、クライアント１０５があらゆる単一マウントポイントを知っている必要はない点にある。従来の方法では、各クライアントが通常、フォルダ内の既知のリモートファイルシステムが全てリストされたＦＳｔａｂファイル（ファイルシステム・テーブルファイル）を有している（すなわち、ＦＳｔａｂは、クライアントがドライバ、サーバ名およびボリューム名にアクセスできる場合の、クライアントにおけるローカル・ディレクトリまたはローカル・フォルダのリストである）。サーバ１１０に変化が生じ、その変化が名前空間に影響を与えた場合、各クライアント１０５を再構成し、クライアントがサーバにアクセスできるようにしなければならない。そのため、従来の方法におけるマウントポイント管理作業は、複雑で時間を要するものとなっている。

【００６７】
本発明の一実施形態の場合、各クライアントはディレクトリ８０５をマウントするだけでよく、この１つに統合されたマウントポイントによって、マウントポイントの管理を単純化している。特に、このマウントポイントは、プロキシ・キャッシュ１１５に対してアンマウントおよびマウントされる。そのため、ボリューム８０５をクライアント１０５に対してアンマウントおよびリマウントする必要はない。従って、ディレクトリ８０５はファイルシステムテーブルファイルの機能を提供し、クライアントがドライバおよびサーバにアクセスできるようにする。ディレクトリ８０５は、リモート・ファイルシステムへのアクセスポイント、および、リモートサーバ（複数の場合もあり）１１０に格納されているような属性のリストを提供する。

【００６８】
複数のマウントポイントを作成し、属性集合をマウントポイントに関連付ける方法
同一オブジェクトについて、ファイルハンドルは様々な方法で書き換えることができる。一実施形態において、ファイルハンドルは、プロキシ・キャッシュ上の関連ローカルマウントポイントの属性を利用して様々な方法で書き換えることができる。

【００６９】
次に、図８Ｂを参照する。マウントポイントは、ローカルボリューム名として定義される。例えば、ローカルボリューム名「ｕｓｅｒ／ｒｅａｄ－ｏｎｌｙ／ｅｍｐｌｏｙｅｅ

１」（ブロック８５０を参照）とローカルボリューム名「ｕｓｅｒ／ｅｍｐｌｏｙｅｅ１」（ブロック８５５を参照）はそれぞれ、２つの異なるファイルハンドルを生成する。その理由は、各ローカルボリューム名についてマウントポイントが異なるからである。具体的には、「ｕｓｅｒ／ｒｅａｄ－ｏｎｌｙ／ｅｍｐｌｏｙｅｅ１」８５０のマウントポイントは「ｕｓｅｒ／ｒｅａｄ－ｏｎｌｙ」である一方、「ｕｓｅｒ／ｅｍｐｌｏｙｅｅ１」８５５のマウントポイントは「ｕｓｅｒ」である。

【００７０】

ローカルボリューム名は、関連する属性またはメタデータを有することができる。例えば、アクセス制限、格納するための資格、またはキャッシュ機能、セキュリティ属性、認証、プロキシ・キャッシュ装置の認証レベル、サーバの読出し専用属性または書込み専用属性などである。データを共有する他のボリュームに関連するマウントポイントにアクセスすることにより、そのデータに関連する属性を変更することができる。つまり、個々の属性は特定のマウントポイントまたはアクセスポイントに関連付けることができ、それをハッシュすると、重複しない識別子が形成される。一実施形態において、この重複しない識別子は、ローカルボリューム名、ファイルシステムＩＤおよびサーバＩＰアドレスをＭＤ５のような適当なハッシュアルゴリズムを用いてハッシュしたものである。図８Ｂの例では、属性（複数の場合もあり）８５６がマウントポイント８５０に関連付けられ、属性（複数の場合もあり）８５７がマウントポイント８５５に関連付けられている。

【００７１】

なお、プロキシ・キャッシュ１１５のディスク上の同じ物理オブジェクトを異なるマウントポイントからアクセスすることも可能である。

【００７２】

キャッシュ・データの整合性をとる方法

最初に、プロキシ・キャッシュがＮＦＳクライアントに対して保証するデータ整合性モデルを定義する。このモデルは、ＮＦＳサーバによって保証されるものと同じである。次に、サーバおよびプロキシ・キャッシュがどのようにしてこのＮＦＳデータ整合性モデルを保証するのかについて説明する。次に、入手元サーバにアクセスし、キャッシュとディレクトリの両方を通してデータを変更しているクライアント群に対し、このデータ整合性モデルがどのように作用するかについて説明する。最後に、キャッシュ最大寿命（すなわち、入手元サーバを用いて新しさが確認される前にヒットが満たされる際の最大量の時間）を使用する方法について説明し、キャッシュ最大寿命がデータの新しさおよび整合性にどのように影響を与えるかについて説明する。以下に記載する例はＮＦＳプロトコルに関するものになっているが、キャッシュ・データの整合性をとるためのこの方法は、他のプロトコルでも使用することができる。

【００７３】

ＮＦＳクライアントデータ整合性

クライアントの視点から見ると、ＮＦＳ整合性モデルは、番号順のファイル・トランザクションを保証するものである。あるトランザクションが完了し、特定バージョンのファイルが返された後、そのファイルに対して開始されるトランザクションはいずれも、少なくとも新しいバージョン、すなわちもっと新しいバージョンを返すことが保証される。我々はこれを連続整合性と呼んでいる。ＮＦＳにおける整合性の理解が難しい理由は、例えば、種々の時間にわたるトランザクションがパイプライン化され、信頼性のないトランスポートに依存することがあるからである。従って、下記のように少数のクライアントの定義から始めることが有用である。

【００７４】

トランザクションの開始

ＮＦＳトランザクションの開始は、クライアントが最初の要求ＲＰＣ（リモート・プロシージャ・コール）を送信した時刻として定義される。以後のＤＵＰ要求（複製要求）がトランザクションの開始をリセットすることはない。ＤＵＰ要求は、ＮＦＳプロトコルによって以前に行なわれた要求の複製である。ファイルサーバ上のＤＵＰキャッシュは、ク

ライアントからの要求に対する応答だけでなく、トランザクション識別子（ＸＩＤ）も格納している。各ＮＦＳ要求は、重複しないＸＩＤを有する（要求が重複しないＸＩＤを有するのに対し、応答は要求に対応するＸＩＤを有する）。複製要求は、もとの要求と同じＸＩＤを有する。

【００７５】

トランザクションの終了

ＮＦＳトランザクションの終了は、クライアントが、発生しうる複数のＤＵＰ応答のうちの最初のものを受信したときに発生する。

【００７６】

連続トランザクション

一方のトランザクションが完了する前に他方のトランザクションが開始された場合、それら２つのトランザクションは連続であるものと定義される。

【００７７】

ファイル

ＮＦＳサーバ上では、ファイルは、そのファイルのアクセスに使用されるプロトコル・バージョンやエクスポートパスに応じて、種々のファイルハンドルによって参照される。プロキシ・キャッシュの一実施形態において、ファイルはファイルハンドルによって一意に識別される。これは、異なるエクスポートパスまたは異なるプロトコル・バージョンを用いて複数のマウントポイントにインポートされた実際には同一のファイルが、複数の別個のファイルとみなされることを意味している。

【００７８】

整合性

サーバの視点から見ると、サーバは通常、クライアント側でトランザクションが何時開始され、何時終了するのかを正確には知らない。サーバは応答を送信した後、それがクライアントに直ぐに受信されることを想定している。サーバが要求を受信した場合、サーバは、以前に送信した応答がすべてクライアントに受信された後で、その要求をクライアントが開始したものと想定する。サーバの視点から見れば、トランザクションは最初の要求を受信した時に開始され、最初の応答を返送した時に終了する。

【００７９】

サーバ上で実行しなければならない変更処理は、一度だけであることが望ましい。例えば、ＲＥＭＯＶＥ処理について考えてみる。ＲＥＭＯＶＥ処理が最初にサーバに受信されたときは、ファイルシステムから適当なファイルが消去され、その記録がＤＵＰキャッシュに作成される。以後の複製要求に対する応答はすべて、このＤＵＰキャッシュを用いて行なわれる。これによって、以前の応答の中にクライアントに届くまでの間に失われたものがあっても、複製要求の失敗を防止することができる。

【００８０】

ＮＦＳプロキシ・キャッシュにおける整合性

プロキシ・キャッシュがクライアントに応答を送信した場合、プロキシ・キャッシュは、それ以降に受信した全ての要求に対する応答が、整合性のとれた返答になることを保証しなければならない。これは、特定バージョンのファイルをクライアントに教えた後、そのファイルに関する以後のトランザクションは全て、少なくともその特定バージョンと同程度に新しいバージョンを返さなければならないことを意味している。

【００８１】

プロキシ・キャッシュの視点からすれば、これは次の２つを保証することを意味する。

（１）プロキシ・キャッシュは、応答をクライアントに送信するときに、以後のキャッシュ・ヒットに対して整合性のとれた応答を返すように準備することが望ましい。これは、従来は、キャッシュ・ミス時に応答をクライアントに送信する前に、データが満杯であることを確認することによって行なわれている。

（２）プロキシ・キャッシュにデータを入れる場合、プロキシ・キャッシュは、クライアントに既に返送されたそれよりも新しいバージョンのファイルが無いことが確認できた

場合にのみ、そのデータを以後のキャッシュ・ヒット時に使用することができる。我々は、これを整合性充填と呼んでいる。整合性充填については以下の実施形態で説明する。

【００８２】

キャッシュ・リブートの扱い

プロキシ・キャッシュが起動すると、プロキシ・キャッシュはメモリ内オブジェクトキャッシュを再構築しながらプロキシ専用モードで動作する。一実施形態において、充填はロギングされず、不適切なシャットダウンは、キャッシュを不整合状態にすることがある。そのため、プロキシ・キャッシュは起動時に、全てのキャッシュ・オブジェクトを、キャッシュ・ヒット時に返送する前に確認が必要なものとしてマークする。プロキシ・キャッシュは、クライアント要求に応答してオブジェクトをクライアントに渡す前に、そのオブジェクトの属性をチェックすることにより、そのオブジェクトが最新のバージョンであることをサーバを用いて確認する。

【００８３】

リブート時には、プロキシ・キャッシュは全てのトランザクションを中止し、トランザクションがクライアントによって再起動される。サーバから配送中のキャッシュ充填要求はプロキシ・キャッシュによって捨てられる（破棄される）。なぜなら、それらのキャッシュ充填要求のＸＩＤ（トランザクション識別子）は通常、解釈できないものになるからである（リブートの際には通常、後続の要求のＸＩＤについて新たなＸＩＤマッピングがプロキシ・キャッシュ内に作成されるので、新たなＸＩＤマッピングはリブート発生前のＸＩＤを解釈することができない）。リブートの後、到来するクライアント要求はすべて、新たなトランザクションの最初の要求として扱われる。起動時に全てのトランザクションを再起動させれば、プロキシ・キャッシュは、すべてのアクティブ・トランザクションについて開始時刻を記録することができる。これによって、整合性充填に関する本発明の一実施形態が可能になる。

【００８４】

整合性キャッシュ充填

キャッシュ充填には次の２つがある。

（１）読出し専用処理によって返送されたデータの格納

（２）キャッシュ・データに対する変更処理の再現

変更処理は通常、プロキシ・キャッシュで常に見られるものではないので、あるいは、それらを再現することは非常に難しいので、プロキシ・キャッシュからファイルを投げなければならない場合がしばしばある。「投げる」とは、特定タイプの充填を簡略化したものである。また、性能上の理由から、データの充填が有用な場合もあり、そのデータは新しい場合もあれば、キャッシュファイルを不整合状態にする場合もある（ＮＦＳ応答によっては、参照される全てのファイルについてファイル属性を持たないものもある）。本発明の一実施形態では、データを充填することによってこの状態を処理する。ただし、ファイルは、後続のヒット時にキャッシュから返送される前に確認が必要なものとしてマークされる。

【００８５】

整合性のとれたデータがクライアントに常に送信されることを保証するためには、全てのファイルを次の２つのうちの一方に分類しなければならない。

（１）整合性があるもの

（２）不整合の可能性があるもの

不整合の可能性がある充填は、以前にクライアントに送信されたデータに比べて、新しいデータを有する場合がある。ＮＦＳバージョン２では、応答が新しいデータを有しているか否かを常に知ることができるとは限らない。この種の応答は、不整合な充填の候補となる。応答に関連するファイル（複数の場合もあり）に変更があった時にサーバから再要求された応答も、候補となる。プロキシ・キャッシュのアプローチは通常、堅実なものでなければならないので、確実なことが言えない場合は「新しい」と推測する。不整合な充填があると、プロキシ・キャッシュは、後続のヒット時に問題のファイルを提供する前に

、その新しさを入手元サーバを用いて確認しなければならない。これは、直後に簡単なキャッシュ・ヒットが続く可能性がある整合性のある充填とは対照的である。

【００８６】

一実施形態では、全てのトランザクションが起動時に再起動され、キャッシュされた内容はすべて、確認が必要なものとしてマークされる。これによって、全てのキャッシュデータが、整合性のあるものとして開始されることが保証される。ファイルＦに対する不整合な充填は、Ｆに対する最後に記録された変更よりも前にキャッシュによって開始されたトランザクションの結果得られるものとして定義される。最後に記録された変更は、変更の可能性を示すサーバ応答がプロキシ・キャッシュによって受信された時刻として定義される。これは、その変更がプロキシ・キャッシュのクライアントから見れるようになった時である。

【００８７】

ファイルに対する最後に記録された変更を追跡する場合、プロキシ・キャッシュは非常に堅実なものであることが望ましい。プロキシ・キャッシュは、サーバからの応答を処理するときに、例えば下記の事項のうちのいずかが真であれば、トランザクションに関連するファイルに対する最後に記録された変更を更新する。

・応答が、ファイルに対する変更があったことを知っている処理に関するものである。

・応答が、ファイルの処理後属性を含まない（処理後属性とは、ファイルに対して処理を実施した後のファイルの属性である）。

・応答が、ファイルの処理後属性を含み、それらの属性が、現在キャッシュされているものとは異なる。

・ファイルの属性が何もキャッシュされていない。

・キャッシュが再構築段階にある。

【００８８】

各ファイルについて最後に記録された変更を追跡する本方法は、ハッシュテーブル（下記の説明を参照）を用いて重複しないファイル名（例えば、ＭＤ５アルゴリズムによって得られるものなど）を定めることによる。ファイルは通常、ＬＲＵに返還要求される。トランザクションがこのＬＲＵにおける最も古い要素よりも古い場合、そのトランザクションに関する最後に記録された変更は返還要求された可能性があるものと仮定され、最後に記録された変更は不整合な充填として扱われる。

【００８９】

分散ＮＦＳ整合性

ファイルシステムを求める全てのクライアントが特定のプロキシ・キャッシュを通過するとすれば、それらのクライアントは、データの最新のコピーを得ることが保証される。その理由は、変更処理はすべて、プロキシ・キャッシュで実施されるか、または、新たなキャッシュデータが投げられるかのいずれかとなるからである。具体的には、全ての変更処理がプロキシ・キャッシュを通過するとすれば、たとえ読出し専用処理をルーティングしたとしても、このことが言える。

【００９０】

クライアントが様々な経路を通じてファイルシステムにアクセスする場合（一部がプロキシ・キャッシュを通り、一部が入手元サーバに直接アクセスする場合など）、複雑さが生じる。プロキシ・キャッシュは、変更処理を入手元サーバに直接渡した後、新しいデータをクライアントに提供することができる。プロキシ・キャッシュのクライアントが新しいデータを得ることができ、且つ、整合性のとれたデータを常に受け取れることは重要である。クライアントによって様々な新しさの許容範囲が認められるためには、プロキシ・キャッシュは、様々なデータセットについて最大寿命を指定することが出来なければならない。この最大寿命は、入手元サーバを用いて新しさが確認される前にヒットが満たされる際の最大量の時間である。

【００９１】

図８Ｃは、プロキシ・キャッシュがサーバに対して前回、要求Ｒ０を行い、その要求Ｒ

０が時刻Ａ０でサーバに受信された場合の図である。プロキシ・キャッシュが次の要求Ｒ１をサーバに対して行うと、その要求は時刻Ａ１でサーバに受信される。ただし、要求Ｒ０に応答するデータＤ０は、要求Ｒ１に応答するデータＤ１をプロキシ・キャッシュが受信した後で、プロキシ・キャッシュに受信される。従って、後続の受信データＤ０は、ファイルＤの最新バージョンではないかもしれない。

【００９２】

図８Ｄは、プロキシ・キャッシュがサーバに対して前回、要求Ｒ０を行い、その要求Ｒ０が時刻Ａ０でサーバに受信された場合の図である。プロキシ・キャッシュが次の要求Ｒ１をサーバに対して行なうと、その要求は時刻Ａ１でサーバに受信される。要求Ｒ０に応答するデータＤ０は、要求Ｒ１に応答するデータＤ１をプロキシ・キャッシュが受信する前に、プロキシ・キャッシュに受信される。

【００９３】

図８Ｅに示すように、本発明の一実施形態では、ハッシュテーブル８７０をプロキシ・キャッシュに格納する。ハッシュテーブル８７０はエントリ８７２を有する。各エントリは、オブジェクト８７３（例えば、ファイルやデータブロックなど）および最後に記録された変更値８７４を含む。例えば、返却データＤ０は、その最後に記録された変更を示す属性ＬＲＣ０を有する場合がある（そのオブジェクトが最後に変更されたものである場合）。同様に、返却データＤ１は、その最後に記録された変更を示す属性ＬＲＣ１を有する場合がある。本発明の一実施形態において、データＤ０のＬＲＣ０値が、所定の時間（例えば、３０秒など）内に要求Ｒ１が成された時間よりも大きい場合、プロキシ・キャッシュはＤ１の属性を確かめるための要求を作成し、Ｄ１がデータの最新バージョンであるか否かを確認する。一実施形態において、プロキシ・キャッシュ１１５のフィルエンジン１６６は、例えば「ＧＥＴＡＴＴＲ」コマンドをサーバに送信し、データＤの属性を得る。この方法によれば、プロキシ・キャッシュに格納されたデータのデータ整合性を達成することが可能になる。

【００９４】

プリフェッチによる属性の事前確認

ＨＴＴＰプロトコルの場合、サーバは通常、クライアントがサーバにおけるファイル更新をチェックしなければならない時を示すインジケータをクライアントに与える。これに対し、ＮＦＳプロトコルでは、サーバは、サーバファイルの更新をクライアントに通知することができない。以下で説明する種々の事前確認方法は、プロキシ・キャッシュがサーバに格納されたデータの属性を確認できるようにするためのものである。

【００９５】

プロキシ・キャッシュ１１５を用いると、サーバ１１０とクライアント１０５が離れている場合に（例えば、サーバ１１０とクライアント１０５が異なる都市にある場合に）、ＷＡＮ１２５を介してデータを統合することができる。プロキシ・キャッシュ１１５は、クライアント１０５がサーバ１１０に格納されているデータを要求する際の遅延を減らすことができるので、サーバ１１０はクライアント１０５にとってローカルであるかのようになる。遅延を減らすために、本発明の実施形態は、ＮＦＳのようなオープン・プロトコルをＷＡＮ１２５を介して確実に伝送できるようにする解決策を提供する。ＮＦＳを使用するクライアントは通常、フォルダ内の各ファイルに関する情報を順番に要求する。以下で説明するような本発明の一実施形態によるプリフェッチ方法は、ＷＡＮを介したＮＦＳの遅延を減らすことができるという利点がある。図９に示すように、クライアント１０５がデータの要求９０５を送信し、キャッシュ・ミスが発生した場合、プロキシ・キャッシュ１１５は、サーバ９１０内のデータの属性をプリフェッチ９１０し、サーバはその属性を返す９１５。そのため、プロキシ・キャッシュ１１５は、データのいかなる変化も判定することができる。その後、プロキシ・キャッシュ１１５は、要求されたデータをクライアントに送信することができる。

【００９６】

本発明の一実施形態では、サーバ（複数の場合もあり）１１０内のフォルダ、サブフォ

ルダ、および／または、ファイルの属性をチェックすることにより、プロキシ・キャッシュ１１５に格納されているデータに関する更新や変更を事前に確認することができる。オブジェクト（ディレクトリやファイルなど）の属性をＮＦＳで得るためには、「ＧＥＴＡＴＴＲ」コマンドが使用される（もちろん、他の適当な処理を用いてもよい）。各オブジェクトについて「ＧＥＴＡＴＴＲ」コマンドを使用し、オブジェクトに変更があったか否かを示す属性をチェックする。一実施形態において、ＮＦＳトラフィックはメタデータの事前確認を行い、トラフィックを加速することができる。

【００９７】

事前確認アクセスパターンには、例えば、シーケンシャル・アクセス、ランダム・アクセス、履歴を利用した隣接オブジェクトのリンクのアクセスなど、様々なパターンが使用される。シーケンシャル・アクセスによる確認とは、複数のオブジェクトの属性を順番に確認することを意味する。クライアント事前確認は、履歴によるリンクを作成し、クライアントは、構築された履歴によるリンクから利点を得る。履歴を利用した隣接オブジェクトのリンクについては、後で説明する。クライアントによっては、作成されたリンクに重みを加えてもよい。また、リンクを破壊する選択枝を追加することもできる。

【００９８】

図１０は、本発明の一実施形態による、事前確認アクセスを実施するためのプロキシ・キャッシュ１１５の構成要素を示すブロック図である。オブジェクトＥ０は不可視のファイルハンドルを有する場合があるので、オブジェクトＥ０の親ディレクトリを見つける方法は無い場合がある。プロキシ・キャッシュ１１５のメモリに格納される要素ハッシュテーブル１００５およびディレクトリ・ハッシュテーブル１０１０は、ディレクトリＤに関する情報を有する。最初の４つの要素（例えば、Ｅ０～Ｅ３）は、要素ハッシュテーブル１０１０にロードされる。この方法は、ディレクトリＤへのリンクを作成することができ、シーケンスＥ０～Ｅ３のキャプチャを短時間で行なうことができる。最初の４つのオブジェクトＥ０～Ｅ３は、オブジェクトＥ０ファイルのファイルハンドルに基づいてテーブル１０１０内にハッシュされ、所定の隣接オブジェクトＥ１～Ｅ３およびＥ０は、要素ハッシュテーブル１００５内のディレクトリＤを指すことになる（１０１５）。クライアント１０５はオブジェクトＥ０の代わりにオブジェクトＥ１またはＥ２を要求する場合があるので、オブジェクトＥ０～Ｅ４がロードされる。オブジェクトＥ０エントリは、ハッシュテーブル１００５にある親ディレクトリＤに関する十分な情報を有する。オブジェクトＥ０がロードされると、次のポインタがマークされ、次のオブジェクトがＥ１になる。このポインタは、アクセスパターンに関するシーケンシャル・リンクを作成する。図１のリンク方法を使用すると、ディレクトリ・クッキー順位の判定が不要になる。従って、このリンク方法は、クライアント１０５ごとに処理順序を管理することができる。テーブル１０１０にロードされるファイルは、「隠し」ファイルではない方が望ましい。隠しファイルとは、それを見るために特殊な方法を知っていなければならないファイルである。

【００９９】

図１１は、本発明の一実施形態による、確認後のオブジェクトを履歴的にリンクする方法を示すブロック図である。オブジェクトは、隣接オブジェクトのクライアントによる確認が済んでいれば、隣接オブジェクトの集合と対にされる（リンクされる）。例えば、クライアント１０５からの確認要求「ＧＥＴＡＴＴＲ　Ｄ」が、オブジェクトＤを確認する。オブジェクトＥ０およびＥ１は隣接オブジェクトであり、オブジェクトＥ０およびＥ１は事前に確認されているので、オブジェクトＤにリンクされる。

【０１００】

オブジェクトの事前確認が特に有用である理由は、ＮＦＳプロトコルにおける「ＧＥＴＡＴＴＲ」トラフィックが通常、ネットワークのボトルネックとなるからである。オブジェクトを事前確認することにより、ＷＡＮを介したＮＦＳトラフィックを最適化することができる。これに対し、従来の方法でＷＡＮを介したＮＦＳトラフィックを最適化するためには、ＮＦＳトラフィックを適当なプロトコルに変換しなければならない。

【０１０１】

図１２は、オブジェクトの属性の変化によってリンクが破壊されたところを示すブロック図である。例えば、クライアント１０５が「ＧＥＴＡＴＴＲ　Ｅ０」コマンドを送信した後、オブジェクトＥ０が事前確認され、オブジェクトＥ０の属性の変化が検出された場合、プロキシ・キャッシュ１１５は、サーバ１１０からオブジェクトＥ０を読出し、オブジェクトＥ０をキャッシュすることにより処理を進める。

【０１０２】

プロトコル変換（変換要求）

図１３は、クライアント１０５からのＣＩＦＳトラフィックに応答するキャッシュ・ヒット状態を説明するための、クライアント１０５、プロキシ・キャッシュ１１５、および、サーバ１１０を含むネットワークシステムを示す図である。クライアント１０５とプロキシ・キャッシュ１１５の間で使用されるプロトコルはＣＩＦＳプロトコルのようなオープン・プロトコルであり、プロキシ・キャッシュ１１５とサーバ１１０の間で使用されるプロトコルはＮＦＳプロトコルであるものと仮定する。

【０１０３】

一実施形態においてプロキシ・キャッシュ１１５は、プロトコルレベルの変換を実施することなく、ファイルシステムレベルで、クライアント１０５からのＣＩＦＳトラフィックをサーバ１１０に対するＮＦＳトラフィックに変換することができる（さらに、サーバ１１０からのＮＦＳトラフィックをクライアント１０５に対するＣＩＦＳトラフィックに変換することもできる）。従って、プロキシ・キャッシュ１１５を用いると、ＣＩＦＳクライアントは、キャッシュ・ミスがあっても、ＮＦＳサーバのデータにアクセスすることができる。一般に、ＣＩＦＳプロトコルは、逐次処理であり、ＣＩＦＳプロトコルに比べて処理回数が多いので、ＷＡＮを介して使用することは実際的でない。さらに、ＣＩＦＳプロトコルはあまり十分に定義されていないので、ＣＩＦＳプロトコルをＷＡＮを介して使用するように最適化することは、プロトコル違反となる場合もある。これに対し、プロキシ・キャッシュ１１５は、クライアント１０５からのＣＩＦＳトラフィックをＮＦＳトラフィックに変換することができるので、キャッシュ・ミスがあっても、サーバ１１０に対するアクセスが可能になる。

【０１０４】

図１３は、クライアント１０５からのＣＩＦＳ要求１３０５に関するキャッシュ・ヒット状態を示す図である。スタック１３１０は、要求１０５に応答してファイルシステム１３１５（例えば、Ｗｒｉｔｅ－Ａｎｙｗｈｅｒｅファイルレイアウト、すなわちＷＡＦＬなど）に質問を送り、要求されたファイルをファイルシステム１３１５が有しているか否かを確認する。すると、ファイルシステム１３１５は、要求されたデータ１３２０をスタック１３１０に渡す。次に、スタック１３１０は、要求されたデータ１３２０を要求元クライアント１０５に送信する。同様の処理は、プロキシ・キャッシュ１１５がクライアントからＮＦＳ要求を受信し、キャッシュ・ヒット状態が発生した場合にも行なわれる。

【０１０５】

図１４は、プロキシ・キャッシュ１１５においてキャッシュ・ミス状態を引き起こすＣＩＦＳ要求の処理を示すブロック図である。スタック１３１０は、要求に応じて、要求されたデータに関する質問をファイルシステム１３１５に送る。ファイルシステム１３１５は、要求されたデータがファイルシステム１３１５内に存在しないので、ＮＦＳクライアントモジュール１３２５（このモジュールは通常、プログラムとして実施され、クライアント上で実行される）に質問を送る。すると、ＮＦＳクライアントモジュール１３２５はＮＦＳ要求１３３０をＷＡＮを介してサーバ１１０に送信する。サーバ１１０は、プロキシ・キャッシュ１１５からのＮＦＳ要求１３３０に応答して、要求されたデータをＮＦＳプロトコルによりＷＡＮを介してＮＦＳクライアントモジュール１３２５に送信する（１３３５）。次に、ＮＦＳクライアントモジュール１３２５は要求されたデータをファイルシステム１３１５に送信し、ファイルシステム１３１５はその要求されたデータをスタック１３１０に渡す。次にスタック１３１０は、要求されたデータ１３４０を要求元クライアント１０５に送信する。

【０１０６】
図１５は、プロキシ・キャッシュ１１５にキャッシュ・ミス状態を引き起こすＮＦＳ要求１５００の処理を示すブロック図である。スタック１５１０は、要求されたデータがファイルシステム１５１５内にあるか否かをチェックする。プロキシ・キャッシュ１１５は、キャッシュ・ミスに応答して、ＮＦＳ要求１５３０をネットワーク・インタフェース１５０５からサーバ１１０に送信する。フィルエンジン１５２０は、サーバ応答１５３５を構文解析し、クライアント１０５とサーバ１１０の間の通信を可能にするデータ構造、および、不可視の特徴をもつファイルハンドルを使用する。要求されたデータ１５２５は、プロキシ・キャッシュ１１５に割り当てられた後、プロキシ・キャッシュ１１５からクライアント１０５に送信される。

【０１０７】
遅延書込み
図１６は、遅延書込み処理を説明するための、本発明の一実施形態によるネットワークを示すブロック図である。プロキシ・キャッシュ１１５は、クライアント１０５からの書込み処理１６０５に応答し、遅延スパース書込み処理１６１０をサーバ１１０に対して実施するように構成される。例えば、クライアント１０５がプロキシ・キャッシュ１１５に対して書込み処理１６０５を実施する場合、プロキシ・キャッシュ１１５は、そのデータをサーバ１１０に書込み、サーバ１１０内のファイルを変更することができる。書込み処理１６１０は、プロキシ・キャッシュ１１５のバックグラウンド処理として実施してもよいし、後で実施してもよい。従って、書込みデータは最初にプロキシ・キャッシュ内のローカルディスクに送信され、リモートサーバに対する書込み処理は遅れる場合がある。

【０１０８】
プロキシ・キャッシュ１１５は、書込み処理１６１０を実施する前の準備領域としての働きを持ち、書込みデータをバッファリングしておくことができる。書込み処理１６１０は、書込み処理１６０５の速度よりも遅い速度で実施することができる。代替または追加として、書込み処理１６１０は、書込み処理１６０５よりも遅い時間に、すなわち書込み処理１６０５よりも後の時刻に実施してもよい。従って、プロキシ・キャッシュ１１５とサーバ１１０とを接続しているＷＡＮが故障した場合や、プロキシ・キャッシュ１１５とサーバ１１０との間に計画的切断が発生した場合、プロキシ・キャッシュ１１５は、ＷＡＮが復旧した時点でサーバ１１０に対する書込み処理１６１０を実施することができる。この遅延書込み機能によれば、クライアント１０５の動作を低下させたり、クライアント１０５の動作に悪影響を及ぼすことなく、最終的には書込みデータをサーバ１１０に確実に格納することができる。従って、本発明の一実施形態によれば、サーバ１１０に対する書込み処理１６１０は通常自動的に実施されるので、データをコピーするための従来のソフトウェアは不要になる。

【０１０９】
書込み処理１６０５は、プロキシ・キャッシュ１１５に格納された読出し専用以外のデータを無効することができる。さらに、プロキシ・キャッシュ１１５にキャッシュされた書込みデータは、消去処理を必要としない。なぜなら、書込みデータはその後、例えば上記のようなＦＩＦＯ処理を用いて、プロキシ・キャッシュ１１５から消去されるからである。

【０１１０】
プロキシ・キャッシュ１１５からサーバ１１０への遅延書込み処理は、次のように実施される。（１）即座に、（２）スケジュールに従って、（３）ネットワークのアイドル時間に、および／または、（４）帯域幅制限に応じて（例えば、書込み処理は、最大ネットワーク帯域幅が得られる時間に実施される）。プロキシ・キャッシュ１１５に対するトラフィックの帯域幅を制限するために、予約速度制御を行なってもよい。具体的には、帯域幅は、様々な要素に基づいて優先順位付けをすることによって制限または管理することができる。そうした要素には、例えば、（１）クライアントＩＰ情報（レイヤ４　ＴＣＰ／ＩＰ層情報）、（２）ＮＦＳ情報（レイヤ７）（例えば、ユーザＩＤ、読出し／書込み等

の処理タイプなど）、（３）特定ファイル（例えば、特定のディレクトリの優先順位を他のディレクトリの優先順位よりも高くする場合がある）などがある。従って、帯域幅は、要求を送信しているクライアントのタイプや、上記のようなその他の要素に基づいて優先順位を付けることができる。

【０１１１】

Ｄｅｌｅｇａｔｉｏｎ（委譲）を用いた方法（Ｄｅｌｅｇａｔｉｏｎは、サーバ内のオブジェクトを変更するパーミッションすなわち権利だけをプロキシ・キャッシュに与える）については、後で図１８のところで説明する。Ｄｅｌｅｇａｔｉｏｎ技術は、ＣＩＦＳおよびＮＦＳバージョン４でサポートされている。

【０１１２】

図１７は、サーバ１１０内のファイルシステム１７０５のうちの下位部分１７０３を含むアクティブデータセットをキャッシュするように構成されたプロキシ・キャッシュ１１５の一実施形態を示す。下位部分は、例えば「ルート」１７０７および「ユーザ１」の下にあるサブツリーである。サブツリー１７０３は、プロキシ・キャッシュ１１５とサーバ１１０の間の計画的切断に先立って、サーバ１１０からプロキシ・キャッシュ１１５にフェッチされる。従って、クライアント１０５は、プロキシ・キャッシュ１１５にアクセスすることにより、必須データまたは特定データを常に利用することができる。例えば、プロキシ・キャッシュ１１５は、計画的切断処理に先立って、サーバ１１０のサブツリー１７０６をフェッチするように構成される。従って、プロキシ・キャッシュは、従来の方法のミラーリング技術を使用せずにデータを複製する方法を提供することができるという利点を有する。

【０１１３】

プロキシ・キャッシュ１１５内にもともとキャッシュされていたデータのコピーは、サーバ１１０から全てのデータを取得するまで、サーバ１１０からの新たな更新データで置換されることはない。プロキシ・キャッシュ１１５におけるデータの置換は、ディレクトリレベルで実施してもよいし、ファイルレベルで実施してもよい。

【０１１４】

上で述べた名前空間の仮想化によれば、サーバ１１０内のオブジェクトの事前確認が可能になる。

【０１１５】

Ｄｅｌｅｇａｔｉｏｎ

図１８は、本発明の一実施形態によるＤｅｌｅｇａｔｉｏｎを使用するシステムを示すブロック図である。サーバ１１０（１）は、ディレクトリまたはファイル（複数の場合もあり）のコピー１８１０を有し、プロキシ・キャッシュ１１５は、論理コピー１８１０におけるデータのキャッシュされたコピー１８０５を有するものと仮定する。この例においてキャッシュされたコピー１８０５は、プロキシ・キャッシュ１１５内のアクティブデータセットに含まれることが好ましい。サーバ１０５は、論理コピー１８１０をプロキシ・キャッシュ１１５に「ロック」することができ、それによってプロキシ・キャッシュ１１５は、キャッシュ・ヒット時に、サーバ１１５の論理コピー１８１０にファイルの更新があったか否かをチェックすることなく、役目を果たすことができるようになる。サーバ１１０（１）は、サーバ１１０（１）の論理コピー１８１０に関するファイル更新をいずれも、ファイル更新ロックの所有者に通知する。プロキシ・キャッシュ１１５がサーバ内のファイルに対するロックを有している場合、そのファイルに対する書込み要求は通常いずれもプロキシ・キャッシュ１１５を通過し、その後プロキシ・キャッシュにより、そのファイルに対する遅延書込み処理が実施される。その後、プロキシ・キャッシュ１１５が所有するロックは、サーバによって無効にされる。

【０１１６】

クラスタリング

図１９は、本発明の一実施形態によるクラスタ構成の複数のプロキシ・キャッシュ１１５ａ～１１５ｃを含むブロック図である。プロキシ・キャッシュの数は、図１９のものと

は違っていてもよい。プロキシ・キャッシュのクラスタを用いると、一次プロキシ・キャッシュが故障したときに、クライアント要求トラフィックおよびデータトラフィックに関する代替経路の方法が可能になる。これらのプロキシ・キャッシュ１１５は、例えば、世界または地域の様々な部分に配置することができる。例えば、プロキシ・キャッシュ１１５ａは一次キャッシュであり、その後これが故障した場合、他のプロキシ・キャッシュを一次プロキシ・キャッシュとして選択することにより、ネットワーク・トラフィックの代替経路を形成することができる。例えば、プロキシ・キャッシュ１１５ｂおよび１１５ｃが、一次プロキシ・キャッシュとして選択される。新たなプロキシ・キャッシュを選択する際の基準には例えば、サイズ（大きなサイズのプロキシ・キャッシュはキャッシュ・ヒット率を向上させる）、低減されるトラフィック待ち時間、位置（例えば、１つの「ピア」プロキシ・キャッシュにするか、近接する複数のプロキシ・キャッシュにするかなど）、および／または、要求されているデータなどがある。クライアント１０５は、グローバル名前空間機能およびファイルハンドル変換機能により、他のプロキシ・キャッシュの代わりに、１つのプロキシ・キャッシュにアクセスすることができる。種々のプロキシ・キャッシュはお互いを知らないので、ファイルハンドルの書換えを利用する。クライアントは、ファイルハンドルの書換えにより、自分の要求を特定のプロキシ・キャッシュに適切に宛先変更することができる。そして種々のプロキシ・キャッシュがそのメッセージを解釈する。これに対し、ＨＴＴＰプロトコルでは、オブジェクトの名前がサーバの名前に既に埋め込まれている。

【０１１７】

図２０は、本発明の一実施形態による、リバース・プロキシ構成に構成されたクライアント装置１０５（１）～１０５（ｎ）、プロキシ・キャッシュ２２０５（１）～２２０５（ｐ）およびサーバ２２１０を含むネットワーク２００を示すブロック図である。プロキシ・キャッシュ（図２２では、まとめて２２０５と呼ぶ）を用いると、サーバ２２１０に対する双方向トラフィックのスケーリングまたは分散が可能になる。プロキシ・キャッシュ２２０５は、ＬＡＮ２２１５を介してサーバ２２１０にローカル接続される。変数ｐは任意の適当な整数値である。つまり、プロキシ・キャッシュ装置（図２２では、まとめて２２０５と呼ぶ）の数は変更してもよい。例えば、サーバ２２１０のトラフィック処理速度が遅い場合、ネットワーク２２００は、プロキシ・キャッシュ２２０５だけで実施することもできる。クライアント装置１０５は、ＷＡＮ１２５を介してロードバランス調節スイッチのようなロード・バランサ２２２０に接続される。ロードバランス調節スイッチは、トラフィックをクライアント１０５とプロキシ・キャッシュ２２０５に分散させる。

【０１１８】

図２２のリバース・プロキシ構成によれば、データをプロキシ・キャッシュ２２０５にオン・デマンドでキャッシュすることが可能になる。このオン・デマンドのキャッシュ処理は、標準的な複製技術に比べて効率が高く、標準複製技術の場合のように特殊なソフトウェアを必要としない。複数のプロキシ・キャッシュ２２０５を使用することで、サーバ２２１０に宛てられた要求を分散させることが可能になる。複数のプロキシ・キャッシュ２２０５を用いると、クライアント１０５のうちの１つからの要求に応答するキャッシュ・ヒットの可能性も全体的に上げることができる。なぜなら、各プロキシ・キャッシュ２２０５のアクティブデータセットが、要求されたデータを含む場合があるからである。従って、複数のプロキシ・キャッシュ２２５０を用いると、クライアント１０５からサーバ２２１０に送信される要求の量を減らすことができる。

【０１１９】

利点として、サーバ２２１０の処理速度が遅い場合は、複数のプロキシ・キャッシュ２２０５を使用することで、キャッシュ・ヒットの可能性を向上させることができ、クライアント１０５に対してオン・デマンドのデータキャッシュ機能を提供することができる。従って、サーバ２２１０の処理速度が遅いことが、クライアント要求のためのボトルネックにはならなくなる。図２２のリバース・プロキシ構成は、読出しトラフィックに（例えば、動画のグラフィック・レンダリングなどの用途に）関して非常に有用である。また、

リバース・プロキシ構成の場合、ファイルハンドルの変形は一般に必要とされない。さらに、プロキシ・キャッシュの使用はファイルサーバの使用に比べて安価であり、また、プロキシ・キャッシュにおけるデータの複製は特殊なソフトウェアを必要とせずに自動的に実施することができる。

【０１２０】

ロード・バランサ２２２０は、キャッシュ・ヒットからの要求およびデータを分散させる。キャッシュ・ミスがあった場合（すなわち、要求されたデータをアクティブデータセット内に有しているプロキシ・キャッシュ２０５が無い場合）、プロキシ・キャッシュ２０５のうちの１つは、要求されたデータを求める要求をサーバ２１０に送り、そのデータのキャッシュされたコピーを要求元サーバに返送する。

【０１２１】

図２１は、単一のプロキシ・キャッシュ２２０５（１）がサーバ２２１０にリバース・プロキシ構成で接続されたネットワーク２３００を示すブロック図である。この例では、サーバ２２１０の処理速度が十分に遅いものになっている点に注意して欲しい。従って、ネットワーク２３００で実施される単一のプロキシ・キャッシュ２２０５は、クライアント要求を受信し、キャッシュ・ヒットがあった場合はデータを要求元クライアント１０５に返送し、キャッシュ・ミスがあった場合はサーバ２２１０にアクセスする。

【０１２２】

図２２は、本発明の他の実施形態によるネットワーク２４００を示すブロック図である。プロキシ・キャッシュ２２０５（１）～２２０５（ｐ）は、例えばネットワーク・スイッチ２４０５を介して、複数のサーバ２２１０（１）～２２１０（ｍ）と通信することができる。変数ｍは、任意の適当な整数値である。通常、ネットワーク・スイッチ２４０５はＬＡＮ２２１５内で実施される。従って、プロキシ・キャッシュ２２０５は、複数のサーバ２２１０に対する双方向のトラフィックをスケーリングまたは分散させることができ、サーバ２２１０に対する双方向のトラフィックを加速させることができる。図２２のリバース・プロキシ構成は、特にサーバの処理能力が低い場合に、サーバ２２１０がクライアント・トラフィックに対するボトルネックとして作用することを防止する。

【０１２３】

図２３は、本発明のさらに他の実施形態によるネットワーク２４５０を示すブロック図である。プロキシ・キャッシュ２５０５は、プロキシ・キャッシュ２２０５（１）～２２５０（ｐ）と一緒にクラスタ化され、クラスタ化プロキシ・キャッシュについて上に記載した種々の機能を実施することができる。

【０１２４】

本発明の実施形態は、全ての要求についてクライアントがサーバにアクセスしなければならない従来の方法とは違い、クライアントの待ち時間を低減することができるという利点を有する。本発明の実施形態は、クライアントに特殊なソフトウェアを必要とせず、遠くの場所からサーバを簡単に管理することができるので、クライアントおよび／またはサーバの管理の容易さを向上させるという利点を有する。本発明の実施形態は、従来の方法のミラーリングステップを不要にすることにより、データ複製の容易さを向上させることができるという利点を有する。具体的には、本発明の実施形態は、所与の時間間隔でデータのミラーリングを行なうことが可能な管理者を必要としない。本発明の実施形態は、アクティブデータセット全体のサイズがサーバ内のデータセット全体のサイズよりも小さいので、必要とされるディスク要件も少なくて済むという利点を有する。ディスク要件が少ないことから、コストも少なくて済み、故障問題も少なくなる。本発明の実施形態は、多数のクライアントがリバース・プロキシ構成のネットワークを飽和させることなく、多数のデータ要求を行なえるという利点を有する。

【０１２５】

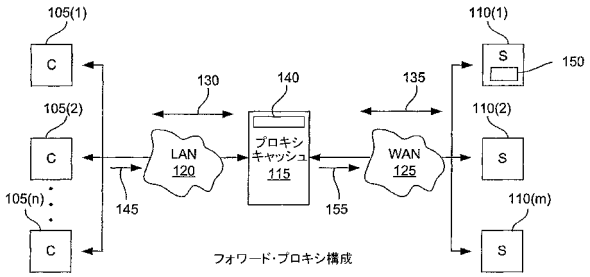
また、本明細書に記載する種々のエンジンまたはモジュールは、例えば、ソフトウェア、コマンド、データファイル、プログラム、コード、モジュール、命令などであってもよく、適当なメカニズムをさらに含むものであってもよい。

【０１２６】
本明細書全体を通じて「１つの実施形態」、「一実施形態」または「特定の実施形態」と呼ばれるものは、本発明の少なくとも１つの実施形態に含まれる実施形態に関連して説明された特定の特徴、構造または特性を意味している。従って、本明細書全体を通じて種々の場所に記載されている「１つの実施形態において」「一実施形態において」または「特定の実施形態において」のような語句は、すべて同じ実施形態を指しているとは必ずしも限らない。さらに、特定の特徴、構造または特性は、任意の適当な方法で１以上の実施形態と組み合わせることもできる。
【０１２７】
上に記載された教示を参照すれば、上記の実施形態および方法に対する変形および変更は他にも可能である。また、本発明の一実施形態に関する少なくとも一部の構成要素は、相互接続された構成要素および回路のネットワークを用いて、特定用途向け集積回路、プログラマブル・ロジック・デバイスまたはフィールド・プログラマブル・ゲート・アレイを用いて、プログラムされた汎用デジタルコンピュータを用いて実施される。接続は、有線でも、無線でも、モデムによるものなどであってもよい。
【０１２８】
また、図面／図に描かれた要素のうちの１以上は、特定の用途にとって有用となるように、さらに離した態様で実施することも、さらに一体化した態様で実施することもでき、場合によっては、除去したり、動作しないようにすることもできる。
【０１２９】
さらに、機械読取可能媒体に格納可能なプログラムまたはコードを実施し、上記の方法のうちのいずれかをコンピュータで実施することも、本発明の範囲内である。
【０１３０】
さらに、図／図面中の信号矢印は、例として解釈すべきものであり、特にことわりがない限り、限定の意味で解釈してはならない。さらに、本明細書の開示に使用される「または」という用語は通常、特にことわりがない限り、「および／または」の意味で使用している。また、構成要素またはステップの組み合わせも考えられる。その場合、分離または結合する能力の解釈が不明瞭であるので、専門用語は予測される。
【０１３１】
本明細書の説明および特許請求の範囲の記載に使用されている「ａ」、「ａｎ」および「ｔｈｅ」は、文脈から明らかにそうでないと言えない限り、複数の場合も含む。また、本明細書の説明および特許請求の範囲の記載に使用されている「ｉｎ」は、文脈から巻きらかにそうでないと言えない限り、「ｉｎ」の意味と「ｏｎ」の意味とを含む。
【０１３２】
さらに　本発明の態様は、ハードウェア、ソフトウェア、ファームウェア、および／または、それらの組み合わせで実施することができる。
【０１３３】
要約書に記載したものを含む本発明の例示的な実施形態に関する上記の説明は、本発明を網羅的に記載しようとするものでもなければ、本発明を開示した形態に厳密に制限しようとするものでもない。本明細書には、例示の目的で本発明の具体的実施形態や例が記載されているが、本発明の範囲内で種々の均等な変更が可能であることは、当業者にとって明らかであろう。
【０１３４】
上記の詳細な説明を参照すれば、本発明に対してそれらの変更を行なうことができる。特許請求の範囲に使用されている用語は、本発明を明細書および特許請求の範囲に開示されている特定の実施形態に制限するものとして解釈してはならない。そうではなく、本発明の範囲は特許請求の範囲によって完全に決定され、特許請求の範囲は、特許請求の範囲に関する確立された解釈の原則に従って解釈しなければならない。
【図面の簡単な説明】
【０１３５】

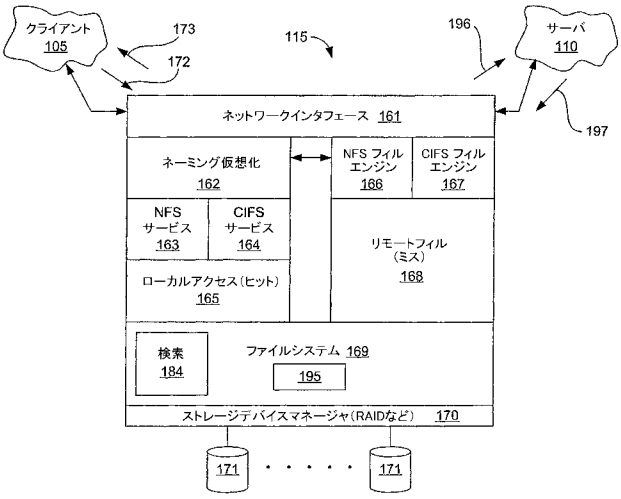
【図１Ａ】本発明の一実施形態による装置を示すブロック図である。
【図１Ｂ】本発明の一実施形態によるプロキシ・キャッシュを示すブロック図である。
【図１Ｃ】キャッシュ・ヒット処理およびキャッシュ・ミス処理を示すブロック図である。
【図２】本発明の一実施形態による加齢方式または重要度方式を用いてアクティブデータセットを管理する方法を示すブロック図である。
【図３】本発明の実施形態によりアクティブデータセットを管理する他の方法を示すブロック図である。
【図４】ファイルハンドルを示すブロック図である。
【図５】各サーバ内のボリュームの例を示すブロック図である。
【図６】本発明の一実施形態による、特定サーバにおける各ボリュームのＦＳ$_{id}$値のマッピングを示すブロック図である。
【図７】本発明の一実施形態による、同一にマッピングされたファイルハンドルを有する複数のプロキシ・キャッシュを備えたネットワークシステムを示すブロック図である。
【図８Ａ】本発明の一実施形態による、仮想名前空間を可能にするためのローカルディレクトリを示すブロック図である。
【図８Ｂ】１つのマウントポイントに一連の属性が関連付けられた複数のマウントポイントを作成する方法を示すブロック図である。
【図８Ｃ】キャッシュデータの整合性をとる方法を示すブロック図である。
【図８Ｄ】キャッシュデータの整合性をとる方法を示すブロック図である。
【図８Ｅ】キャッシュデータの整合性をとる方法を示すブロック図である。
【図９】本発明の一実施形態による、サーバ内のデータの属性のプリフェッチまたは事前確認を示すブロック図である。
【図１０】本発明の一実施形態による事前確認アクセスを実施するための、プロキシ・キャッシュ内の構成要素を示すブロック図である。
【図１１】本発明の一実施形態による、確認されたオブジェクトを履歴によりリンクさせる方法を示すブロック図である。
【図１２】本発明の一実施形態による、オブジェクトの属性の変化に応じてリンクが破壊されたところを示すブロック図である。
【図１３】本発明の一実施形態による、クライアントからのＣＩＦＳトラフィックに応答するキャッシュヒット状態を説明するための、クライアント、プロキシ・キャッシュおよびサーバを含むネットワークシステムを示す図である。
【図１４】本発明の一実施形態による、プロキシ・キャッシュにキャッシュ・ミス状態を引き起こすＣＩＦＳ要求の処理を示すブロック図である。
【図１５】本発明の一実施形態による、プロキシ・キャッシュにキャッシュ・ミス状態を引き起こすＮＦＳ要求の処理を示すブロック図である。
【図１６】遅延書込みを説明するための、本発明の一実施形態によるネットワークを示すブロック図である。
【図１７】サーバの所定の記憶領域にあるデータの特定の論理コピーのキャッシュされたコピーを含むアクティブデータセットをキャッシュするように構成されたプロキシ・キャッシュの一実施形態を示す図である。
【図１８】本発明の一実施形態によるデータコヒーレンシを提供するシステムのブロック図である。
【図１９】本発明の一実施形態による、クラスタ構成の複数のプロキシ・キャッシュを含むブロック図である。
【図２０】本発明の一実施形態による、リバース・プロキシ構成のクライアント、プロキシ・キャッシュおよびサーバを含むネットワークを示すブロック図である。
【図２１】単一のプロキシキャッシュ２０５（１）がサーバ２１０にリバース・プロキシ構成で接続されたネットワーク３００を示すブロック図である。
【図２２】本発明の他の実施形態によるネットワーク２４００を示すブロック図である。

【図２３】本発明の他の実施形態によるネットワーク２５００を示すブロック図である。

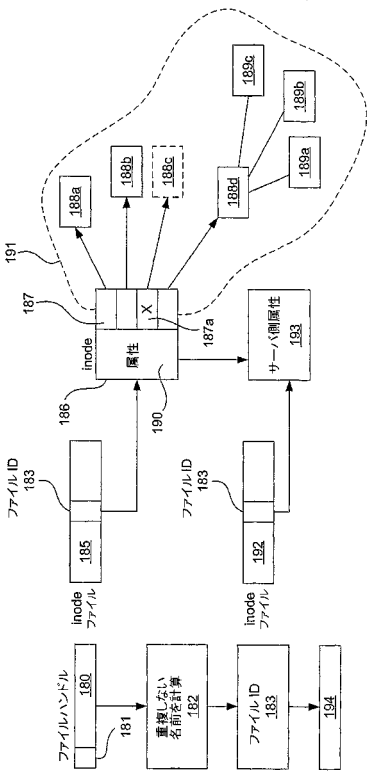
【図１Ａ】



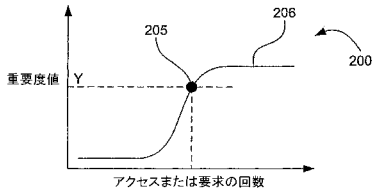
【図１Ｂ】



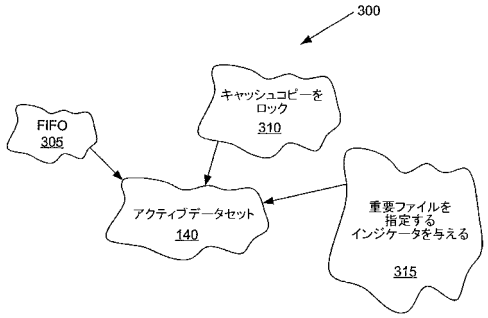
【図１Ｃ】



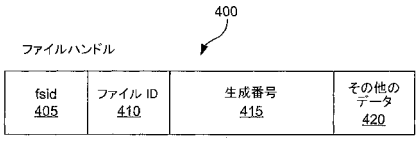
【図２】



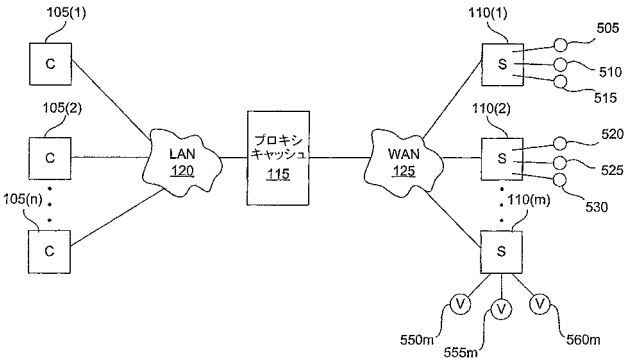
【図３】



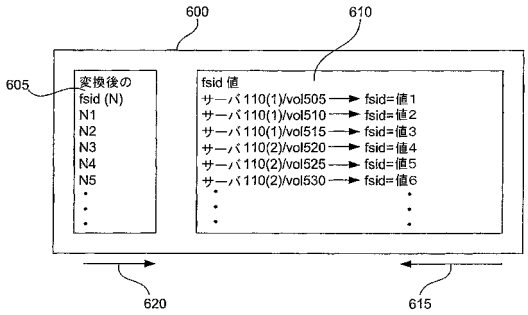
【図４】



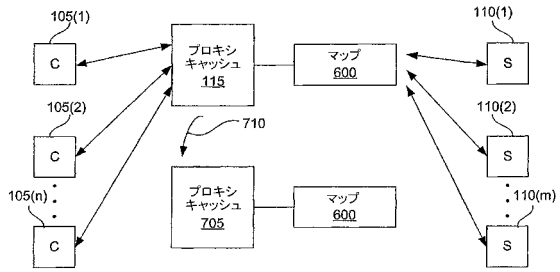
【図５】



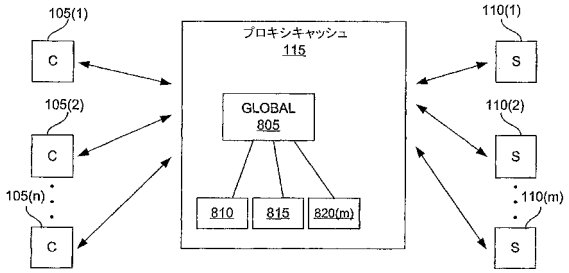
【図６】



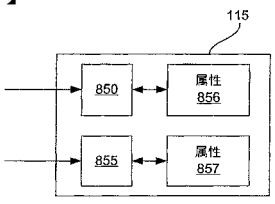
【図７】



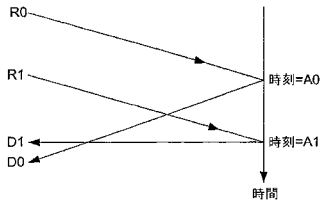
【図８Ａ】



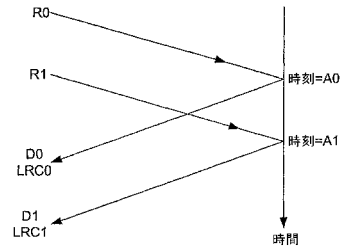
【図８Ｂ】



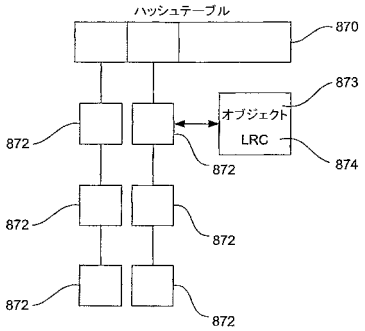
【図８Ｃ】



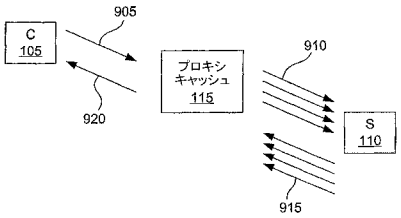
【図８Ｄ】



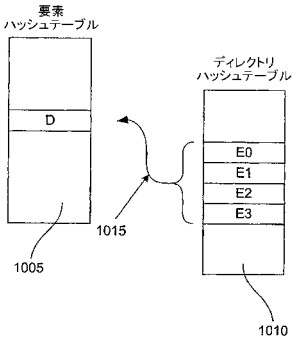
【図８Ｅ】



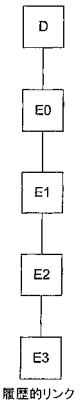
【図９】



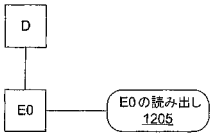
【図１０】



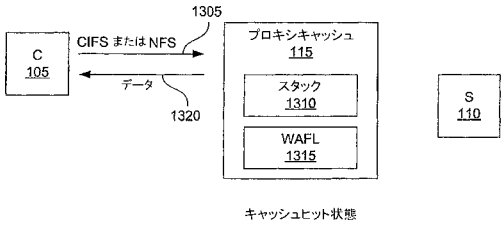
【図１１】



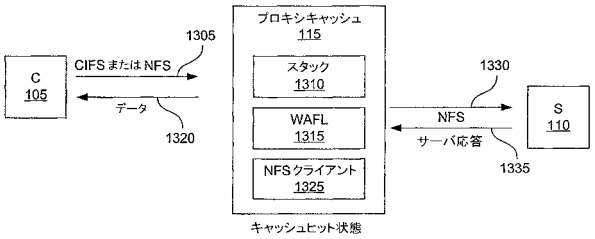
【図１２】



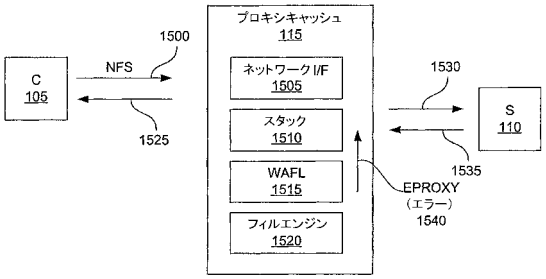
【図１３】



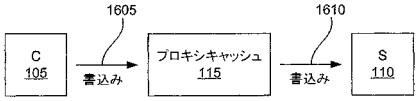
【図１４】



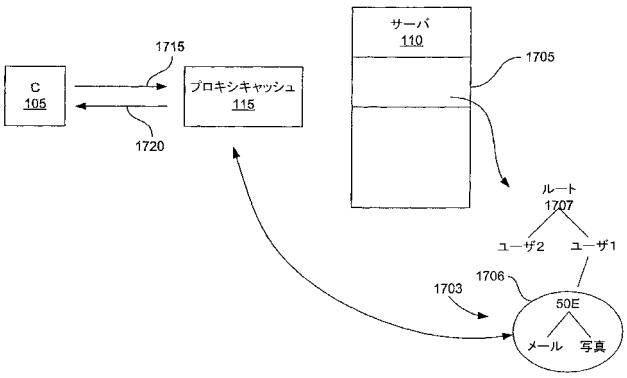
【図１５】



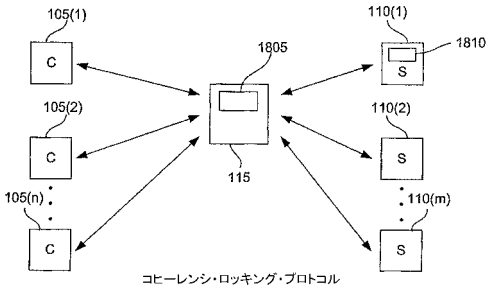
【図１６】



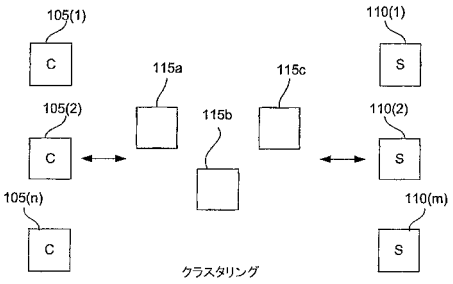
【図１７】



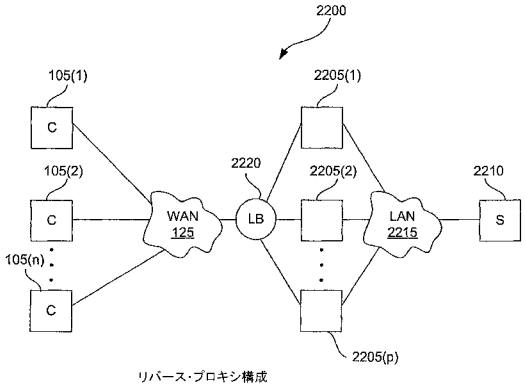
【図１８】



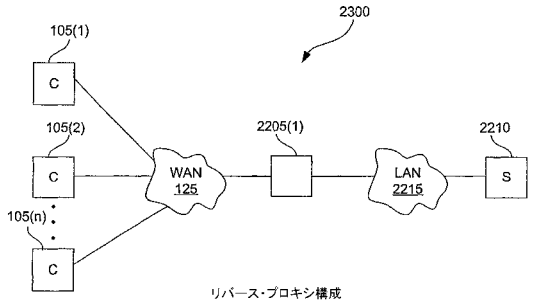
【図１９】



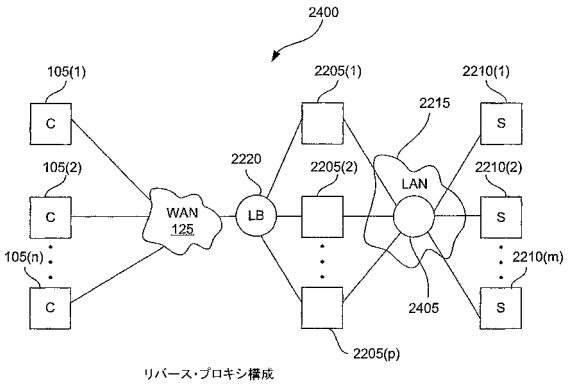
【図２０】



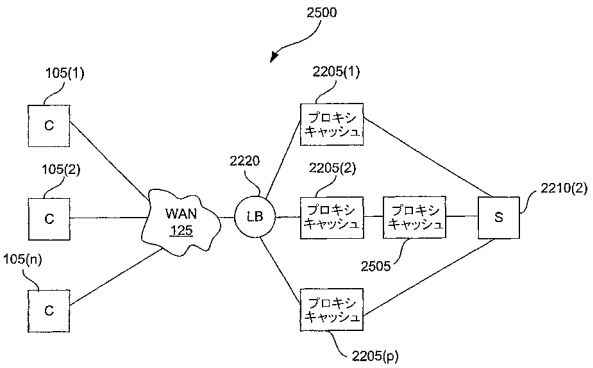
【図２１】



【図２２】



【図２３】



【手続補正書】
【提出日】平成17年7月11日(2005.7.11)
【手続補正１】
【補正対象書類名】特許請求の範囲
【補正対象項目名】全文
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】
データをネットワーク内にキャッシュする装置であって、
クライアントからオブジェクトの要求を受信し、サーバからデータブロックをフェッチするように構成されたプロキシ・キャッシュを含み、
前記プロキシ・キャッシュは、階層構造の前記データブロックをアクティブデータセットとして前記オブジェクト内にキャッシュするように構成される、装置。
【請求項２】
前記データブロックはバッファ・ツリーに入れられ、ファイル内に前記階層関係が可能である、請求項１に記載の装置。
【請求項３】
クライアントからの要求に応答して、前記アクティブデータセット中にデータブロックが配置される、請求項１に記載の装置。
【請求項４】
前記アクティブデータセットは、少なくとも前記サーバに格納されたデータの一部を含む、請求項１に記載の装置。
【請求項５】
前記アクティブデータセットは、特定の時間によって定義される、請求項１に記載の装置。
【請求項６】
前記プロキシ・キャッシュは、前記サーバによって要求されたファイルのデータブロックが欠けていた場合、前記クライアントからの要求に応答して該データブロックを前記サーバからフェッチするように構成される、請求項１に記載の装置。
【請求項７】
前記プロキシ・キャッシュは、前記サーバによって要求されたファイルが前記プロキシ・キャッシュに存在しない場合、前記クライアントから前記サーバへの要求に応答して前記サーバからデータブロックをフェッチするように構成される、請求項１に記載の装置。
【請求項８】
前記プロキシ・キャッシュは、クライアントによって要求されたデータファイル中のデータブロックの有無を判定する計算の一部として重複しない名前を計算する、請求項１に記載の装置。
【請求項９】
前記オブジェクトはデータファイルを含む、請求項１に記載の装置。
【請求項１０】
前記オブジェクトはファイルの一部を含む、請求項１に記載の装置。
【請求項１１】
前記オブジェクトはディレクトリを含む、請求項１に記載の装置。
【請求項１２】
データをネットワーク内にキャッシュする方法であって、
クライアントからプロキシ・キャッシュへのオブジェクトの要求を受信するステップと、
要求された前記オブジェクトが前記プロキシ・キャッシュ内にある場合、要求された前記オブジェクトを前記クライアントに送信するステップと、

要求された前記オブジェクト中のデータブロックが欠けていた場合、該データブロックをサーバからフェッチし、該データブロックをアクティブデータセットとして階層関係に入れ、要求された前記オブジェクトを前記クライアントに送信するステップと、
からなる方法。
【請求項１３】
前記サーバによって要求されたファイルが存在しない場合、前記クライアントから前記サーバへの要求に応答して前記サーバからデータブロックをフェッチするステップをさらに含む、請求項１２に記載の方法。
【請求項１４】
前記データブロックがバッファ・ツリーに入れられ、ファイル内に前記階層関係が可能である、請求項１２に記載の方法。
【請求項１５】
前記アクティブデータセットは、少なくとも前記サーバに格納されたデータの一部を含む、請求項１２に記載の方法。
【請求項１６】
前記アクティブデータセットは、特定の時間によって定義される、請求項１２に記載の方法。
【請求項１７】
クライアントによって要求されたデータファイル中のデータブロックの有無を判定する計算の一部として重複しない名前を計算するステップを含む、請求項１２に記載の方法。
【請求項１８】
置換ポリシーに基づいて前記アクティブデータセットの一部を消去するステップをさらに含む、請求項１２に記載の方法。
【請求項１９】
オブジェクトは、ファイルの少なくとも一部を含む、請求項１２に記載の方法。
【請求項２０】
データをネットワーク内にキャッシュする方法であって、
クライアントからプロキシ・キャッシュへのオブジェクトの要求を受信するステップと、
要求された前記オブジェクトを前記クライアントに送信するステップと、
前記データブロックをアクティブデータセットの一部として階層関係に入れ、要求された前記オブジェクトを前記クライアントに送信するステップと、
からなる方法。

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US03/29398

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 15/167, 173
US CL : 709/225, 216
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 709/225, 216

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EAST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6,311,216 B1(SMITH et al.) 30 October 2001, col. 1, line 10 - col. 21, line 13.	1-63
Y	US 6,438,652 B1 (JORDAN et al) 20 August 2002, col. 1, line 13 to col. 9, line 58.	1-63
Y	US 6,442,651 B2 (CROW et al.) 27 August 2002, col. 1, line 13 to col. 5, line 55.	1-63
A	US 6,178,461 B1 (CHAN et al) 23 January 2001, col. 1, line 15 to col. 11, line 63.	1-63
A, P	US 6,553,411 B1 (DIAS et al) 22 April 2003, col. 1, line 13 to col. 6, line 15	1-63
A	US 2002/0026560 A1 (JORDAN et al) 28 February 2002, [0002-0041].	1-63

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
07 April 2004 (07.04.2004)	19 APR 2004
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450 Alexandria, Virginia 22313-1450 Facsimile No. (703) 305-3230	Authorized officer David A. Wiley Telephone No. 703-305-3900

Form PCT/ISA/210 (second sheet) (July 1998)

フロントページの続き

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HU,IE,IT,LU,MC,NL,PT,RO,SE,SI,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RU,SC,SD,SE,SG,SK,SL,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,UZ,VC,VN,YU,ZA,ZM,ZW

(72)発明者 アカウィー，エマニュエル
アメリカ合衆国カリフォルニア州９４１０７，サンフランシスコ，コネチカット・ストリート・５７５
(72)発明者 プラカシュ，アシシュ
アメリカ合衆国ノースカロライナ州２７５６０，モリスヴィル，ラ・ジョラ・レーン・１０６
(72)発明者 アムダー，マシュー
アメリカ合衆国カリフォルニア州９４１０７，サンフランシスコ，ミシシッピー・ストリート・５８０
(72)発明者 アイヤー，カーティク
アメリカ合衆国カリフォルニア州９５０５１，サンタクララ，ホメステッド・ロード・３１３１，アパートメント・ナンバー２３エイチ

Ｆターム(参考) 5B082 FA12 HA02 HA05 HA08