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

特許第4349301号
(P4349301)

(45)発行日　平成21年10月21日(2009.10.21)　(24)登録日　平成21年7月31日(2009.7.31)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 545B
G06F 3/06 (2006.01)　G06F 12/00 520P
G06F 3/06 304F

請求項の数 95　(全 48 頁)

(21)出願番号　特願2005-59115(P2005-59115)
(22)出願日　平成17年3月3日(2005.3.3)
(65)公開番号　特開2006-164211(P2006-164211A)
(43)公開日　平成18年6月22日(2006.6.22)
審査請求日　平成17年11月15日(2005.11.15)
(31)優先権主張番号　特願2004-329209(P2004-329209)
(32)優先日　平成16年11月12日(2004.11.12)
(33)優先権主張国　日本国(JP)

前置審査

(73)特許権者　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(74)代理人　100109313
弁理士　机　昌彦
(74)代理人　100121290
弁理士　木村　明隆
(74)代理人　100160554
弁理士　浅井　俊雄
(72)発明者　鳥居　隆史
東京都港区芝五丁目７番１号　日本電気株式会社内
(72)発明者　山川　聡
東京都港区芝五丁目７番１号　日本電気株式会社内

最終頁に続く

(54)【発明の名称】ストレージ管理システムと方法並びにプログラム

(57)【特許請求の範囲】
【請求項１】
クライアントとファイルサーバとに接続されるコントローラであって、
前記クライアントから前記ファイルサーバのファイルへのアクセス要求が発行されたとき、前記ファイルへのアクセス要求を受けるパケット転送手段と、
前記ファイルが、前記ファイルサーバから他のファイルサーバに移動された実ファイルの位置情報を記録しており前記ファイルサーバに格納されているスタブファイルであるか否かを判断するスタブファイル判定手段と、
前記ファイルサーバに格納された前記スタブファイルと前記他のファイルサーバに格納された前記実ファイルとの対応関係を含む情報を記憶管理する手段と、
スタブファイルである場合、前記アクセス要求が、実ファイルへのアクセスを必要とする場合には、前記スタブファイルの情報に基づき、前記他のファイルサーバの実ファイルにアクセスし前記クライアントに応答を返す制御を行うサーバアクセス手段と、を備え、
前記スタブファイル内に、複数の分割ファイルの位置情報を格納しておき、一のファイルに対応した前記スタブファイルを介して、前記一のファイルの最大ファイルサイズの拡大を可能としてなることを特徴とするコントローラ。
【請求項２】
前記アクセス要求が、前記コントローラで保持する前記情報で対処できるものである場合、前記実ファイルにアクセスすることなく、前記コントローラから、前記クライアントに前記アクセス要求に対する応答を返す、ことを特徴とする請求項１記載のコントローラ。

【請求項３】
前記コントローラは、少なくとも１つのクライアントと複数のファイルサーバとの間に論理的に配置される中間装置に含まれている、ことを特徴とする請求項１又は２記載のコントローラ。
【請求項４】
前記コントローラは、前記ファイルサーバに含まれている、ことを特徴とする請求項１又は２記載のコントローラ。
【請求項５】
前記ファイルの属性情報に基づき、スタブファイルであるか否かを判断する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項６】
前記スタブファイルは固定長である、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項７】
前記スタブファイルの作成時刻属性欄に、スタブファイルであることを示す値を含む、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項８】
前記スタブファイルの変更時刻属性欄に、スタブファイルであることを示す値を含む、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項９】
前記スタブファイルは、前記中間装置がスタブファイルであることを判断することを可能とするためのマジックナンバーを含む、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項１０】
前記スタブファイルが、前記実ファイルの属性を含む、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項１１】
前記スタブファイルが、前記実ファイルへアクセスする識別子を含む、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項１２】
前記識別子が、パス名、ストレージの生成したＩＤ、ＵＲＬ（Uniform Resource Locator）のいずれかである、ことを特徴とする請求項１１記載のコントローラ。
【請求項１３】
前記スタブファイルが、前記識別子を複数含む、ことを特徴とする請求項１１記載のコントローラ。
【請求項１４】
前記クライアントから前記ファイルサーバへのファイルアクセスに基づき、スタブファイルへのアクセスであるか否かを判断する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項１５】
前記クライアントからのアクセスに含まれるファイルＩＤ又はファイルハンドルに基づき、スタブファイルへのアクセスであるか否かを判断する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項１６】
前記クライアントからのアクセス要求をファイルサーバに転送し、前記ファイルサーバからの応答に含まれる属性に基づき、スタブファイルであるかを判断する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項１７】
前記クライアントからのアクセス要求をファイルサーバに転送し、前記ファイルサーバからの応答に含まれる属性に基づき、スタブファイルである可能性があると判断した場合、

前記スタブファイルを前記１つのファイルサーバより読み出し、前記スタブファイルのマジックナンバーからスタブファイルであるか否かを判断する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項１８】
前記クライアントからのアクセス要求をそのままファイルサーバに転送するとファイルの属性が変更されることになる場合には、前記スタブファイルであるか否か判断する手段は、前記アクセスをファイルサーバに転送する前に、前記ファイルサーバにアクセスして、スタブファイルであるか否かを判断する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項１９】
前記クライアントから、前記ファイルサーバへのファイルアクセスを見て、スタブファイルへのアクセスであるかを判断して、制御動作を変える、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項２０】
前記スタブファイルが、前記コントローラの処理の記述を含み、前記スタブファイルでの記述内容で提示された動作を行う、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項２１】
前記クライアントからのアクセス要求の対象が、スタブファイルでない場合、アクセス要求を前記ファイルサーバにそのまま転送し、前記ファイルサーバからの応答をそのままクライアントに転送する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項２２】
スタブファイルである場合、前記クライアントに気づかせることなく、実ファイルへのアクセスに切り替える、ことを特徴とする請求項２１記載のコントローラ。
【請求項２３】
オープン時に、オープン対象のファイルがスタブファイルであれば、ファイルＩＤを記憶しておき、以後のファイルＩＤを使ったアクセス要求では、前記記憶されたファイルＩＤと比較することで、スタブファイルへのアクセスであるかを判定する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項２４】
ルックアップ（ＬＯＯＫＵＰ）時に、スタブファイルであれば、ファイルハンドルを記憶しておき、以後のファイルハンドルを使ったリクエストでは、記憶したファイルハンドルと比較することで、スタブファイルへのアクセスを判定する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項２５】
クライアントが、ファイルを、リード又はライトをするときに、スタブファイルへのアクセスであれば、実ファイルにアクセスするように変更する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項２６】
実ファイルをオープンし、前記実ファイルのファイルＩＤをスタブファイルのファイルＩＤとセットで記憶しておく、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項２７】
実ファイルをルックアップ（ＬＯＯＫＵＰ）し、前記実ファイルのファイルハンドルをスタブファイルのファイルハンドルとセットで記憶しておく、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項２８】
前記クライアントよりスタブファイルのファイルＩＤを用いたアクセス要求が入力されたときに、前記記憶されている実ファイルのファイルＩＤに付け替えて、前記実ファイルを

格納したファイルサーバに前記アクセス要求を転送する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項２９】
前記クライアントよりスタブファイルのファイルハンドルを用いたアクセス要求が入力された場合、前記記憶しておいた実ファイルのファイルハンドルに付け替え、前記実ファイルを格納したファイルサーバに前記アクセス要求を転送する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項３０】
前記クライアントより属性を変更するリクエストが入力されたとき、スタブファイル属性を書き換えられないように変更して、前記ファイルサーバに転送し、スタブファイル属性が書き換えられないように制御する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項３１】
前記スタブファイルの内容をキャッシュする記憶部を備え、
前記クライアントからのアクセスがあったとき、スタブファイル本体を前記１つのファイルから読み出さずに前記キャッシュの内容を用いて処理を行う、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項３２】
前記クライアントから、スタブファイルへの更新で属性が変わったとき、スタブファイル本体のファイル属性を変更するのではなく、前記コントローラでキャッシュしているスタブファイルデータの内容を更新し、前記クライアントからクローズ（ＣＬＯＳＥ）リクエストを受け取ったときに、スタブファイル本体に書き戻す、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項３３】
前記クライアントから、クローズ（ＣＬＯＳＥ）リクエストが発行されると、これを受け、スタブファイルと実ファイルの両方に対してクローズ処理する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項３４】
前記クライアントから、クローズ（ＣＬＯＳＥ）リクエストが発行されると、これを受け、記憶していたファイルＩＤのテーブルを消去する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項３５】
前記クライアントが明示的にクローズ（ＣＬＯＳＥ）を発行しないプロトコルの場合には、前記クライアントがファイルハンドルをキャッシュしておく時間を考慮し、それ以上の時間、前記クライアントからアクセスがない場合、記憶していたファイルハンドル変換テーブルを消去する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項３６】
前記ファイルサーバから前記クライアントへのファイルアクセス応答を受け、前記クライアントにはスタブファイルであることを気づかせないように、前記応答の情報の一部を変更し、前記クライアントに返す、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項３７】
クライアントがファイルをオープンするときに、ファイルの属性により、スタブファイルであるか否かを判断し、スタブファイル属性であれば、実ファイル属性に変更した応答を、前記クライアントに送信する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項３８】
前記クライアントからのファイルの属性取得コマンドに対する前記ファイルサーバからの応答において、前記ファイルがスタブファイルであれば、属性を実ファイル属性に変更し

た上で、変更した応答を、クライアントに送信する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項３９】
前記クライアントに影響を与えることなく、前記実ファイルをコピーして、スタブファイル化する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項４０】
前記１つのファイルサーバのファイルを他のファイルサーバにコピーして、クライアントアクセスを、一時保留して、前記１つのファイルサーバのファイルをスタブファイルに書き換え、その後に、クライアントによるアクセスを再開する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項４１】
前記クライアントに影響を与えることなく、前記スタブファイルを、実ファイルに戻す、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項４２】
前記実ファイルを、１つのファイルサーバのテンポラリ領域にコピーした後、クライアントのアクセスを一時保留して、スタブファイルに名前変更（ＲＥＮＡＭＥ）で入れ替え、その後でクライアントアクセスを再開する、ことを特徴とする請求項１乃至４のいずれか一に記載のコントローラ。
【請求項４３】
少なくとも１つのクライアントと、１次ストレージを有する第１のサーバと、２次ストレージを有する第２のサーバと、制御装置と、を備え、
前記１次ストレージは、前記１次ストレージから前記２次ストレージに移動された実ファイルの位置情報を記録したスタブファイルを備え、
前記制御装置は、前記クライアントから、前記第１のサーバの前記１次ストレージのファイルアクセス要求が発行されたとき、前記ファイルアクセス要求を受けるパケット転送手段と、アクセス対象のファイルがスタブファイルであるか否か判断するスタブファイル判定手段と、
前記第１ストレージに格納された前記スタブファイルと前記第２ストレージに格納された前記実ファイルとの対応関係を含む情報を記憶管理する手段と、
スタブファイルである場合、前記アクセス要求が、実ファイルへのアクセスを必要とする場合には、前記スタブファイルの情報に基づき前記２次ストレージの実ファイルにアクセスし前記クライアントに応答を返す制御を行い、前記クライアントからみて前記スタブファイルが実体であるかのように見せる制御を行うサーバアクセス手段、とを備え、
前記スタブファイル内に、複数の分割ファイルの位置情報を格納しておき、一のファイルに対応した前記スタブファイルを介して、前記一のファイルの最大ファイルサイズの拡大を可能としてなることを特徴とするストレージ管理システム。
【請求項４４】
前記制御装置は、前記１次ストレージに格納された前記スタブファイルと前記２次ストレージに格納された前記実ファイルの対応関係を含む情報を記憶管理する記憶手段を備え、
前記アクセス要求が、前記制御装置で保持する前記情報で対処できるものである場合、前記制御装置から、前記クライアントに前記アクセス要求に対する応答を返す、ことを特徴とする請求項４３記載のストレージ管理システム。
【請求項４５】
前記制御装置は、前記クライアントと前記第１、第２のサーバとの間に論理的に配置される中間装置に含まれている、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項４６】
前記制御装置は、前記第１のサーバに含まれている、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項４７】

前記制御装置は、前記ファイルの属性情報に基づき、スタブファイルであるか否かを判断する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項４８】
前記スタブファイルは固定長である、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項４９】
前記スタブファイルの作成時刻属性欄に、スタブファイルであることを示す値を含む、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項５０】
前記スタブファイルの変更時刻属性欄に、スタブファイルであることを示す値を含む、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項５１】
前記スタブファイルは、前記制御装置がスタブファイルであることを判断することを可能とするためのマジックナンバーを含む、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項５２】
前記スタブファイルが、前記実ファイルの属性を含む、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項５３】
前記スタブファイルが、前記実ファイルへアクセスする識別子を含む、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項５４】
前記識別子が、パス名、ストレージの生成したＩＤ（識別子）、ＵＲＬ（Uniform Resource Locator）のいずれかである、ことを特徴とする請求項４９記載のストレージ管理システム。
【請求項５５】
前記スタブファイルが、前記識別子を複数含む、ことを特徴とする請求項４９記載のストレージ管理システム。
【請求項５６】
前記クライアントから前記第１のサーバへのファイルアクセスに基づき、スタブファイルへのアクセスであるか否かを判断する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項５７】
前記制御装置は、前記クライアントからのアクセスに含まれるファイルＩＤ又はファイルハンドルに基づき、スタブファイルへのアクセスであるか否かを判断する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項５８】
前記制御装置は、前記クライアントからのアクセス要求を第１のサーバに転送し、前記第１のサーバからの応答に含まれる属性に基づき、スタブファイルであるかを判断する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項５９】
前記制御装置は、前記クライアントからのアクセス要求を前記第１のサーバに転送し、前記第１のサーバからの応答に含まれる属性に基づき、スタブファイルである可能性があると判断した場合、前記スタブファイルを前記第１のサーバより読み出し、前記スタブファイルのマジックナンバーからスタブファイルであるか否かを判断する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項６０】
前記制御装置は、前記クライアントからのアクセス要求をそのままサーバに転送するとファイルの属性が変更されることになる場合には、前記アクセスをサーバに転送する前に、前記第１のサーバにアクセスして、スタブファイルであるか否かを判断する、ことを特徴

とする請求項４３又は４４記載のストレージ管理システム。
【請求項６１】
前記制御装置は、前記クライアントから、前記第１のサーバへのファイルアクセスを見て、スタブファイルへのアクセスであるかを判断して、制御動作を変える、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項６２】
前記スタブファイルが、前記制御装置の処理の記述を含み、前記制御装置は、前記スタブファイルでの記述内容で提示された動作を行う、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項６３】
前記クライアントからのアクセス要求の対象が、スタブファイルでない場合、前記制御装置は、アクセス要求を前記第１のサーバにそのまま転送し、前記制御装置は、前記第１のサーバからの応答をそのままクライアントに転送する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項６４】
前記制御装置は、スタブファイルであれば前記クライアントに気づかせることなく実ファイルへのアクセスに切り替える、ことを特徴とする請求項５８記載のストレージ管理システム。
【請求項６５】
前記制御装置は、前記クライアントからのファイルのオープン要求を受け、オープン対象のファイルがスタブファイルであれば、ファイルＩＤを記憶しておき、以後のファイルＩＤを使ったアクセス要求では、前記記憶されたファイルＩＤと比較することで、スタブファイルへのアクセスであるかを判定する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項６６】
前記制御装置は、クライアントからのルックアップ（ＬＯＯＫＵＰ）コマンドを受け、スタブファイルであれば、ファイルハンドルを記憶しておき、以後のファイルハンドルを使ったリクエストでは、記憶したファイルハンドルと比較することで、スタブファイルへのアクセスを判定する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項６７】
前記制御装置は、クライアントが、ファイルを、リード又はライトをするときに、スタブファイルへのアクセスであれば、実ファイルにアクセスするように変更する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項６８】
前記制御装置は、前記第２のサーバにオープン要求を送信して、２次ストレージの実ファイルをオープンし、前記実ファイルのファイルＩＤをスタブファイルのファイルＩＤとセットで記憶しておく、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項６９】
前記制御装置は、２次ストレージの実ファイルをルックアップ（ＬＯＯＫＵＰ）し、前記実ファイルのファイルハンドルをスタブファイルのファイルハンドルとセットで記憶しておく、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項７０】
前記クライアントよりスタブファイルのファイルＩＤを用いたアクセス要求が入力されたときに、前記記憶されている実ファイルのファイルＩＤに付け替えて、前記実ファイルを格納した第１のサーバに前記アクセス要求を転送する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。
【請求項７１】
前記制御装置は、前記クライアントよりスタブファイルのファイルハンドルを用いたアク

セス要求が入力された場合、前記記憶しておいた実ファイルのファイルハンドルに付け替え、前記実ファイルを格納した第１のサーバに前記アクセス要求を転送する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項７２】

前記制御装置は、前記クライアントより属性を変更するリクエストが入力されたとき、スタブファイル属性を書き換えられないように変更して、前記第１のサーバに転送し、スタブファイル属性が書き換えられないように制御する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項７３】

前記制御装置は、前記スタブファイルの内容をキャッシュする記憶部を備え、

前記制御装置は、前記クライアントからのアクセスがあったとき、スタブファイル本体を前記１つのファイルから読み出さずに前記キャッシュの内容を用いて処理を行う、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項７４】

前記制御装置は、前記クライアントから、スタブファイルへの更新で属性が変わったとき、スタブファイル本体のファイル属性を変更するのではなく、キャッシュしているスタブファイルデータの内容を更新し、

前記制御装置は、前記クライアントからクローズ（ＣＬＯＳＥ）リクエストが来たときに、スタブファイル本体に書き戻す、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項７５】

前記制御装置は、前記クライアントから、クローズ（ＣＬＯＳＥ）リクエストが発行されると、これを受け、スタブファイルと実ファイルの両方に対してクローズ処理する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項７６】

前記制御装置は、前記クライアントから、クローズ（ＣＬＯＳＥ）リクエストが発行されると、これを受け、記憶していたファイルＩＤのテーブルを消去する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項７７】

前記クライアントが明示的にクローズ（ＣＬＯＳＥ）を発行しないプロトコルの場合には、前記制御装置は、前記クライアントがファイルハンドルをキャッシュしておく時間を考慮し、それ以上の時間、前記クライアントからアクセスがない場合、記憶していたファイルハンドル変換テーブルを消去する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項７８】

前記制御装置は、前記第２のサーバから前記クライアントへのファイルアクセス応答を受け、前記クライアントには、スタブファイルであることを気づかせないように、前記応答の情報の一部を変更し、前記クライアントに返す、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項７９】

前記制御装置は、前記クライアントがファイルをオープンするときに、オープン要求を受け、前記オープン要求のファイルの属性によりスタブファイルであるか否かを判断し、スタブファイル属性であれば、実ファイル属性に変更して、前記クライアントに送信する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項８０】

前記制御装置は、前記クライアントからのファイルの属性取得コマンドに対するサーバからの応答において、前記ファイルがスタブファイルであれば、属性を実ファイル属性に変更した上で、変更した応答を、クライアントに送信する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項８１】

前記制御装置は、前記クライアントに影響を与えることなく、実ファイルをコピーして、スタブファイル化する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項８２】

前記制御装置は、実ファイルを２次ストレージにコピーして、クライアントアクセスを、一時保留して、１次ストレージのファイルをスタブファイルに書き換え、その後に、クライアントによるアクセスを再開する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項８３】

前記制御装置は、前記クライアントに影響を与えることなく、前記スタブファイルを、実ファイルに戻す、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項８４】

前記制御装置は、前記２次ストレージの前記実ファイルを、前記１次ストレージのテンポラリ領域にコピーした後、前記クライアントのアクセスを一時保留して、スタブファイルに名前変更（ＲＥＮＡＭＥ）で入れ替え、その後、クライアントアクセスを再開する、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項８５】

前記制御装置は、前記１次ストレージのスタブファイルのファイル識別子と、前記２次ストレージの実ファイルのファイル識別子と、スタブ管理テーブルのポインタとを有するスタブファイルＩＤテーブルと、

スタブファイルにクライアントがアクセスしたときにスタブファイルの内容をキャッシュするテーブルであり、前記１次ストレージのファイルパス名と、前記２次ストレージのファイルパス名と、実ファイル属性を有するスタブ管理テーブルと、を備えている、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項８６】

前記制御装置は、アクセス対象のファイルがスタブファイルであるか否かを判定するスタブファイル判定部と、

スタブファイル判定のために１次ストレージのサーバにアクセスするほか、スタブファイル化や通常ファイル化のためのアクセスなどを制御するサーバアクセス部と、

前記スタブ管理テーブルを参照して、前記クライアントへの応答のファイル属性の変更を行う属性変更部と、

を含む、ことを特徴とする請求項４３又は４４記載のストレージ管理システム。

【請求項８７】

少なくとも１つのクライアントと、１次ストレージを有する第１のサーバと、２次ストレージを有する第２のサーバと、制御装置と、を含むシステムのストレージ管理方法であって、

前記１次ストレージに、前記１次ストレージから前記２次ストレージに移動された実ファイルの位置情報を記録したスタブファイルを配置し、

前記制御装置は、前記クライアントから、前記第１のサーバの前記１次ストレージのファイルアクセス要求が発行されたとき、パケット転送手段が前記ファイルアクセス要求を受けるステップと、スタブファイル判定手段がアクセス対象のファイルがスタブファイルであるか否か判定するステップと、

記憶管理する手段が前記第１ストレージに格納された前記スタブファイルと前記第２ストレージに格納された前記実ファイルとの対応関係を含む情報を記憶管理するステップと、

前記制御装置は、前記判定の結果、スタブファイルである場合、前記アクセス要求が、実ファイルへのアクセスを必要とする場合には、サーバアクセス手段が前記スタブファイルの情報に基づき前記２次ストレージの実ファイルにアクセスし前記クライアントに応答を返す制御を行うステップと、を含み、前記クライアントからは前記スタブファイルがあたかも実体であるかのように見えるようにし、

前記スタブファイル内に、複数の分割ファイルの位置情報を格納しておき、一のファイルに対応した前記スタブファイルを介して、前記一のファイルの最大ファイルサイズの拡大を可能としてなることを特徴とするストレージ管理方法。

【請求項８８】

前記制御装置が、前記１次ストレージに格納された前記スタブファイルと前記２次ストレージに格納された前記実ファイルとの対応関係を含む情報を記憶管理するステップと、

前記制御装置が、前記アクセス要求が、前記制御装置で保持する前記情報で対処できるものである場合、前記制御装置から、前記クライアントに前記アクセス要求の応答を返すステップと、

を含む、ことを特徴とする請求項８７記載のストレージ管理方法。

【請求項８９】

クライアントからファイルサーバのファイルへのアクセス要求が発行されたとき、パケット転送手段が前記ファイルへのアクセス要求を受ける処理を行う制御装置を構成するコンピュータに、

前記ファイルが、前記ファイルサーバから他のファイルサーバに移動された実ファイルの位置情報を記録しておりスタブファイル判定手段が前記ファイルサーバに格納されているスタブファイルであるか否か判断する処理と、

記憶管理する手段が前記ファイルサーバに格納された前記スタブファイルと前記他のファイルサーバに格納された前記実ファイルとの対応関係を含む情報を記憶管理する処理と、

スタブファイルである場合、前記アクセス要求が、実ファイルへのアクセスを必要とする場合には、前記スタブファイルの情報に基づき、サーバアクセス手段が前記２次ストレージの実ファイルにアクセスし前記クライアントに応答を返す制御を行う処理と、

前記スタブファイル内に、複数の分割ファイルの位置情報を格納しておき、一のファイルに対応した前記スタブファイルを介して、前記一のファイルの最大ファイルサイズの拡大を可能にしてなることを実行させるプログラム。

【請求項９０】

請求項８９記載のプログラムにおいて、

前記コンピュータは、前記ファイルサーバに格納された前記スタブファイルと前記他のファイルサーバに格納された前記実ファイルとの対応関係を含む情報を記憶手段で記憶管理する処理と、

前記アクセス要求が、前記コントローラで保持する前記情報で対処できるものである場合、前記実ファイルにアクセスすることなく、前記コントローラから、前記クライアントに前記アクセス要求に対する応答を返す処理と、

を前記コンピュータに実行させるプログラム。

【請求項９１】

前記スタブファイルが、１つ又は複数の実ファイルの位置情報を含む、ことを特徴とする請求項１記載のコントローラ。

【請求項９２】

前記スタブファイルが、１つ又は複数の実ファイルの位置情報を含む、ことを特徴とする請求項４３記載のストレージ管理システム。

【請求項９３】

前記スタブファイルが、１つ又は複数の実ファイルの位置情報を含む、ことを特徴とする請求項８７記載のストレージ管理方法。

【請求項９４】

前記スタブファイルが、１つ又は複数の実ファイルの位置情報を含む、ことを特徴とする請求項８９記載のプログラム。

【請求項９５】

クライアントからのファイルのアクセス要求が発行されたとき、前記ファイルへのアクセス要求を受けるパケット転送手段と、

前記ファイルが、１つのファイルシステム内又は前記１つのファイルシステムとは異なる他のファイルシステム内の少なくとも１つのファイルの位置情報を記録しているスタブファイルであるか否か判断するスタブファイル判定手段と、
前記ファイルシステムに格納された前記スタブファイルと前記他のファイルシステムに格納された前記実ファイルとの対応関係を含む情報を記憶管理する手段と、
スタブファイルである場合、前記アクセス要求が、前記スタブファイルに情報が格納されたファイルへのアクセスを必要とする場合には、前記スタブファイルに格納された情報に基づき、前記１つのファイルシステム又は他のファイルシステムのファイルにアクセスし前記クライアントに応答を返す制御を行うサーバアクセス手段と、を備え、
前記スタブファイル内に、複数の分割ファイルの位置情報を格納しておき、一のファイルに対応した前記スタブファイルを介して、前記一のファイルの最大ファイルサイズの拡大を可能としてなることを特徴とするコントローラ。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ストレージ管理技術に関し、特にクライアントとファイルサーバの間に論理的な中間装置を備え、階層間でファイルを配置し、クライアントには階層を隠蔽するシステム及び方法とコンピュータプログラムに関する。
【背景技術】
【０００２】
現在のコンピュータシステムでは、保存しておくデータは年々増えている。ファイル数が増えているだけでなく、ファイルのサイズもマルチメディア系の動画データなどで大きくなっている。そのため、一層の大容量、高速なストレージが必要とされている。企業やデータセンターではこの需要に応えるためにストレージ容量を増やしつづけている。しかしながら、ファイルに保存されているデータの全てがアクセスされるわけではなく、全体のうち、半分以上のデータは、最近１年以上まったくアクセスされないとも言われている。
【０００３】
この現象に注目し、よくアクセスされるデータは高速・高価・低容量なストレージに、ほとんどアクセスされないデータは、低速・安価・大容量なストレージに保管するようにするのが階層ストレージ管理方式（ＨＳＭ：Ｈｉｅｒａｒｃｈｙ　Ｓｔｏｒａｇｅ　Ｍａｎａｇｅｍｅｎｔ）である。
【０００４】
データの量が少なければ、管理者がデータを移動したりできるが、テラバイトレベルのストレージが普通になってきている現状では、管理者がデータの重要度を把握できる容量を超えている。
【０００５】
ＨＳＭでは、クライアントのアクセス頻度やファイルの情報から、自動的に移動するデータを選び出して、高速ストレージから低速ストレージへ移動するなどの処理を行う。
【０００６】
そして、データを低速ストレージに移動しても、クライアントには、移動前と同じように（ただし少し遅いと感じるかもしれないが）アクセスができるようにする。
【０００７】
従来のＨＳＭは、クライアントにＨＳＭソフトウエアを入れるホスト型、サーバ側でＨＳＭを行うサーバ型の大きく２つに大別される。
【０００８】
クライアントにＨＳＭソフトウエアを入れるものは、アプリケーションに特化したＨＳＭが多い。例として、ＨＳＭ対応メールソフトがあげられる。メールは、日付が古くなる

にしたがって重要度が下がるという単純なモデルが成立しやすいため、ＨＳＭ対応がやりやすいという特徴がある。日付の古いメールは、二次ストレージに自動的に移動され、一次ストレージのメールファイルはショートカットファイルに置き換えられる。ショートカットファイルの中には、実体ファイルの二次ストレージ上のパス名が格納されている。

【０００９】

クライアントが、このファイルを読み出すと、ＨＳＭ対応メールソフトがショートカットファイルの中身を読み出して、二次ストレージにアクセスする。そのため、クライアントは、二次ストレージにあることを意識しなくてよい。なお、「ショートカットファイル」は、Ｗｉｎｄｏｗｓ（登録商標）での称呼であり、Ｕｎｉｘ（登録商標）では「シンボリックリンク」という。

【００１０】

このタイプのものは、ショートカットファイルやシンボリックリンクではクライアントには、二次ストレージに移動していることが完全に隠蔽できない。

【００１１】

ＨＳＭ対応ソフトを通した操作であれば、二次ストレージに移動したショートカットファイルのサイズは実体ファイルの容量に変換されるが、例えばＷｉｎｄｏｗｓ（登録商標）のエクスプローラ（Ｅｘｐｌｏｒｅｒ）（登録商標）で見ると、ファイルサイズが１ＫＢのショートカットファイルに見えてしまう。

【００１２】

また、ファイルの削除をする場合、ＨＳＭ対応ソフトを通した操作では、二次ストレージの実体ファイルも削除されるが、エクスプローラ（登録商標）からでは、削除されるのはショートカットファイルのみであり、二次ストレージの実体ファイルは削除されない。

【００１３】

ところで、サーバ側でＨＳＭを行うタイプでは、アプリケーションに特化せず、ファイルシステムレベルで実現するために、スタブファイルを用いる方式がある。

【００１４】

スタブファイルとショートカットファイルとの違いは、ショートカットファイルは、ファイルシステムが認識するのに対して、スタブファイルは、ファイルシステム上、通常ファイルであり、認識するのは、ＨＳＭ対応ソフトである点である。

【００１５】

サーバ側で行うＨＳＭの問題点は、サーバをＨＳＭ対応のもので揃える必要があり、既存のサーバを入れ替える必要が出てくることである。

【００１６】

なお、クライアントから複数のファイルサーバを１つのファイルシステムとしてアクセス可能とする中間装置（スイッチ）として、特許文献１等が参照される。また、階層記憶装置として、特許文献２には、アクセス速度が異なる複数の記憶媒体（一次記憶媒体と二次記憶媒体）を用いて大量のデータを管理する階層記憶装置が、利用頻度の低下したデータの二次記憶媒体への格納等を自動的に行えるようにした構成が開示されている。また、特許文献３には、ＣＩＦＳ（Common Internet File System）プロトコルフィルタが、ＣＩＦＳクライアントから送られる書き込みＳＭＢ（Server Message Block）メッセージデータをストライピング処理し、複数のサーバに順次書き込み、データ読み出しのときは、複数のサーバから順次交互に読み出して復元する構成が開示されている。

【００１７】

【特許文献１】特開２００３－２０３０２９号公報
【特許文献２】特開２００１－２２２４５０号公報
【特許文献３】特開２００３－１５０４３５号公報

【発明の開示】

【発明が解決しようとする課題】

【００１８】

従来の階層ストレージ管理方式は、下記記載の問題点を有している。

【００１９】
第１の問題点は、従来の階層ストレージ管理方式がユーザにとっては導入しにくい、ということである。
【００２０】
クライアントに導入されるＨＳＭ対応ソフトウエアの場合、全てのクライアントにＨＳＭ対応ソフトを導入する手間が必要とされ、サーバ側のＨＳＭでは、ＨＳＭ対応の同一ベンダー装置で揃える必要があり、既存のストレージ装置は使えなかった。
【００２１】
第二の問題点は、二次ストレージへの移動によりクライアントへの見え方が変わってしまう、ということである。すなわち、ＨＳＭ方式によっては、二次ストレージへの移動をクライアントから完全に隠蔽できない。
【００２２】
したがって、本発明の目的は、導入を簡易化するストレージ管理システムと方法並びにプログラムを提供することにある。
【００２３】
本発明の他の目的は、二次ストレージへの移動をクライアントから完全に隠蔽できるストレージ管理システムと方法並びにプログラムを提供することにある。
本発明の他の目的は、最大ファイルサイズを拡大可能とするストレージ管理システムと方法並びにプログラムを提供することにある。
さらに本発明の他の目的は、分散された複数拠点間でスタブ化を可能とするストレージ管理システムと方法並びにプログラムを提供することにある。
さらにまた本発明の他の目的は、スタブ化／非スタブ化のトリガー制御を可能とするストレージ管理システムと方法並びにプログラムを提供することにある。
【課題を解決するための手段】
【００２４】
本願で開示される発明は、前記目的を達成するため、概略以下の構成とされる。
【００２５】
本発明の１つのアスペクト（側面）に係るコントローラは、クライアントからファイルサーバのファイルへのアクセス要求が発行されたとき、前記ファイルへのアクセス要求を受け、前記ファイルが、前記ファイルサーバから他のファイルサーバに移動された実ファイルの位置情報を記録しており前記ファイルサーバに格納されているスタブファイルであるか否か判断する手段と、スタブファイルである場合、前記アクセス要求が、実ファイルへのアクセスを必要とする場合には、前記スタブファイルの情報に基づき、前記他のファイルサーバの実ファイルにアクセスし前記クライアントに応答を返す制御を行う手段と、を備えている。本発明に係るコントローラは、前記ファイルサーバに格納された前記スタブファイルと前記他のファイルサーバに格納された前記実ファイルとの対応関係を含む情報を記憶管理する記憶手段を備え、前記アクセス要求が、前記コントローラで保持する前記情報で対処できるものである場合、前記実ファイルにアクセスすることなく、前記コントローラから、前記クライアントに前記アクセス要求に対する応答を返す構成としてもよい。
【００２６】
本発明の他のアスペクトに係るストレージ管理システムは、少なくとも１つのクライアントと、１次ストレージを有する第１のサーバと、２次ストレージを有する第２のサーバと、制御装置と、を備え、前記１次ストレージは、前記１次ストレージから前記２次ストレージに移動された実ファイルの位置情報を記録したスタブファイルを備え、前記制御装置は、前記クライアントから、前記第１のサーバの前記１次ストレージのファイルアクセス要求が発行されたとき、前記ファイルアクセス要求を受け、アクセス対象のファイルがスタブファイルであるか否か判断し、スタブファイルである場合、前記アクセス要求が、実ファイルへのアクセスを必要とする場合には、前記スタブファイルの情報を用いて前記２次ストレージの実ファイルにアクセスし前記クライアントに応答を返す制御を行い、前

記クライアントからみて前記スタブファイルが実体であるかのように見せる制御を行う。本発明に係るストレージ管理システムにおいて、前記制御装置は、前記１次ストレージに格納された前記スタブファイルと前記２次ストレージに格納された前記実ファイルの対応関係を含む情報を記憶管理する記憶手段を備え、前記アクセス要求が、前記制御装置で保持する前記情報で対処できるものである場合、前記制御装置から、前記クライアントに前記アクセス要求に対する応答を返す構成としてもよい。

【００２７】

本発明の他のアスペクトに係る方法は、少なくとも１つのクライアントと、１次ストレージを有する第１のサーバと、２次ストレージを有する第２のサーバと、制御装置と、を含むシステムのストレージ管理方法であって、

前記１次ストレージに、前記１次ストレージから前記２次ストレージに移動された実ファイルの位置情報を記録したスタブファイルを配置し、

前記制御装置は、前記クライアントから、前記第１のサーバの前記１次ストレージのファイルアクセス要求が発行されたとき、前記ファイルアクセス要求を受け、アクセス対象のファイルがスタブファイルであるか否か判定するステップと、

前記制御装置は、前記判定の結果、スタブファイルである場合、前記アクセス要求が、実ファイルへのアクセスを必要とする場合には、前記スタブファイルの情報に基づき前記２次ストレージの実ファイルにアクセスし前記クライアントに応答を返す制御を行うステップと、

を含み、前記クライアントからは前記スタブファイルがあたかも実体であるかのように見えるようにしたものである。本発明に係る方法において、前記制御装置が、前記１次ストレージに格納された前記スタブファイルと前記２次ストレージに格納された前記実ファイルとの対応関係を含む情報を記憶管理するステップと、

前記制御装置が、前記アクセス要求が、前記制御装置で保持する前記情報で対処できるものである場合、前記制御装置から、前記クライアントに前記アクセス要求の応答を返すステップと、を含むようにしてもよい。

【００２８】

本発明の他のアスペクトに係るコンピュータ・プログラムは、クライアントからファイルサーバのファイルへのアクセス要求が発行されたとき、前記ファイルへのアクセス要求を受ける制御装置を構成するコンピュータに、前記ファイルが、前記ファイルサーバから他のファイルサーバに移動された実ファイルの位置情報を記録しており前記ファイルサーバに格納されているスタブファイルであるか否か判断する処理と、スタブファイルである場合、前記アクセス要求が、実ファイルへのアクセスを必要とする場合には、前記スタブファイルの情報に基づき、前記２次ストレージの実ファイルにアクセスし前記クライアントに応答を返す制御を行う処理と、を実行させるプログラムよりなる。本発明に係るプログラムにおいて、前記コンピュータは、前記ファイルサーバに格納された前記スタブファイルと前記他のファイルサーバに格納された前記実ファイルとの対応関係を含む情報を記憶手段で記憶管理する処理と、前記アクセス要求が、前記コントローラで保持する前記情報で対処できるものである場合、前記実ファイルにアクセスすることなく、前記コントローラから、前記クライアントに前記アクセス要求に対する応答を返す処理と、を実行させるプログラムを含む構成としてもよい。

【００２９】

本発明に係る上記コントローラ、ストレージ管理システムと方法、コンピュータ・プログラムにおいて、以下のように構成してもよい。

【００３０】

前記ファイルの属性情報に基づき、スタブファイルであるか否かを判断する。

【００３１】

前記スタブファイルのファイルサイズを固定長とする。

【００３２】

前記スタブファイルの作成時刻属性欄に、スタブファイルであることを示す値を含む。

【００３３】
前記スタブファイルの変更時刻属性欄に、スタブファイルであることを示す値を含む。
【００３４】
前記スタブファイルは、前記コントローラ（制御装置）がスタブファイルであることを判断することを可能とするためのマジックナンバーを含む。
【００３５】
前記スタブファイルは、前記実ファイルの属性を含む。
【００３６】
前記スタブファイルは、前記実ファイルへアクセスする識別子を含む。
【００３７】
前記識別子が、パス名、ストレージの生成したＩＤ、ＵＲＬ（Uniform Resource Locator）のいずれかである。
【００３８】
前記スタブファイルは、前記識別子を複数含む。
【００３９】
前記クライアントから前記ファイルサーバへのファイルアクセスに基づき、スタブファイルへのアクセスであるか否かを判断する。
【００４０】
前記クライアントからのアクセスに含まれるファイルＩＤ又はファイルハンドルに基づき、スタブファイルへのアクセスであるか否かを判断する。
【００４１】
前記クライアントからのアクセス要求をファイルサーバに転送し、前記ファイルサーバからの応答に含まれる属性に基づき、スタブファイルであるかを判断する。
【００４２】
前記クライアントからのアクセス要求をファイルサーバに転送し、前記ファイルサーバからの応答に含まれる属性に基づき、スタブファイルである可能性があると判断した場合、前記スタブファイルを前記１つのファイルサーバより読み出し、前記スタブファイルのマジックナンバーからスタブファイルであるか否かを判断する。
【００４３】
前記クライアントからのアクセス要求をそのままファイルサーバに転送するとファイルの属性が変更されることになる場合には、前記スタブファイルであるか否か判断する手段は、前記アクセスをファイルサーバに転送する前に、前記ファイルサーバにアクセスして、スタブファイルであるか否かを判断する。
【００４４】
前記クライアントから、ファイルサーバへのファイルアクセスを見て、スタブファイルへのアクセスであるかを判断して、制御動作を変える。
【００４５】
前記スタブファイルが前記コントローラ（制御装置）の処理の記述を含み、前記スタブファイルでの記述内容で提示された動作を行う。
【００４６】
前記クライアントからのアクセス要求の対象が、スタブファイルでない場合、アクセス要求をサーバにそのまま転送し、ファイルサーバからの応答をそのままクライアントに転送する。
【００４７】
スタブファイルである場合、前記クライアントに気づかせることなく実ファイルへのアクセスに切り替える。
【００４８】
オープン時に、オープン対象のファイルがスタブファイルであれば、ファイルＩＤを記憶しておき、以後のファイルＩＤを使ったアクセス要求では、前記記憶されたファイルＩＤと比較することで、スタブファイルへのアクセスであるかを判定する。

【００４９】
ルックアップ（ＬＯＯＫＵＰ）時に、スタブファイルであれば、ファイルハンドルを記憶しておき、以後のファイルハンドルを使ったリクエストでは、記憶したファイルハンドルと比較することで、スタブファイルへのアクセスを判定する。
【００５０】
クライアントが、ファイルを、リード又はライトをするときに、スタブファイルへのアクセスであれば、実ファイルにアクセスするように変更する。
【００５１】
実ファイルをオープンし、前記実ファイルのファイルＩＤをスタブファイルのファイルＩＤとセットで記憶しておく。
【００５２】
実ファイルをルックアップ（ＬＯＯＫＵＰ）し、前記実ファイルのファイルハンドルをスタブファイルのファイルハンドルとセットで記憶しておく。
【００５３】
前記クライアントよりスタブファイルのファイルＩＤを用いたアクセス要求が入力されたときに、前記記憶されている実ファイルのファイルＩＤに付け替えて、前記実ファイルを格納したファイルサーバに前記アクセス要求を転送する。
【００５４】
前記クライアントよりスタブファイルのファイルハンドルを用いたアクセス要求が入力された場合、前記記憶しておいた実ファイルのファイルハンドルに付け替え、前記実ファイルを格納したファイルサーバに前記アクセス要求を転送する。
【００５５】
前記クライアントより属性を変更するリクエストが入力されたとき、スタブファイル属性を書き換えられないように変更してファイルサーバに転送し、スタブファイル属性が書き換えられないように制御する。
【００５６】
前記スタブファイルの内容をキャッシュする記憶部を備え、前記クライアントからのアクセスがあったとき、スタブファイル本体を前記１つのファイルから読み出さずに前記キャッシュの内容を用いて処理を行う。
【００５７】
前記クライアントから、スタブファイルへの更新で属性が変わったとき、スタブファイル本体のファイル属性を変更するのではなく、キャッシュしているスタブファイルデータの内容を更新し、前記クライアントからクローズ（ＣＬＯＳＥ）リクエストが来たときに、スタブファイル本体に書き戻す。
【００５８】
前記クライアントから、クローズ（ＣＬＯＳＥ）リクエストが発行されると、これを受け、スタブファイルと実ファイルの両方に対してクローズ処理する。
【００５９】
前記クライアントから、クローズ（ＣＬＯＳＥ）リクエストが発行されると、これを受け、記憶していたファイルＩＤのテーブルを消去する。
【００６０】
前記クライアントが明示的にクローズ（ＣＬＯＳＥ）を発行しないプロトコルの場合には、前記クライアントがファイルハンドルをキャッシュしておく時間を考慮し、それ以上の時間、前記クライアントからアクセスがない場合、記憶していたファイルハンドル変換テーブルを消去する。
【００６１】
前記ファイルサーバから前記クライアントへのファイルアクセス応答を受け、前記クライアントにはスタブファイルであることを気づかせないように、前記応答の情報の一部を変更し、前記クライアントに返す。
【００６２】

クライアントがファイルをオープンするときに、ファイルの属性によりスタブファイルであるか否かを判断し、スタブファイル属性であれば、実ファイル属性に変更してクライアントに送信する。

【００６３】

前記クライアントからのファイルの属性取得コマンドに対するファイルサーバからの応答において、前記ファイルがスタブファイルであれば、属性を実ファイル属性に変更した上で、変更した応答を、クライアントに送信する。

【００６４】

前記クライアントに影響を与えることなく、実ファイルをコピーして、スタブファイル化する。

【００６５】

前記１つのファイルサーバのファイルを他のファイルサーバにコピーして、クライアントアクセスを、一時保留して、前記１つのファイルサーバのファイルをスタブファイルに書き換え、その後に、クライアントによるアクセスを再開する。

【００６６】

前記クライアントに影響を与えることなく、前記スタブファイルを、実ファイルに戻す。

【００６７】

前記実ファイルを、１つのファイルサーバのテンポラリ領域にコピーした後、クライアントのアクセスを一時保留して、スタブファイルに名前変更（ＲＥＮＡＭＥ）で入れ替え、その後でクライアントアクセスを再開する。

【００６８】

本発明においては、クライアントとサーバの間に、制御装置（「中間装置」ともいう）を導入し、中間装置は、クライアント・サーバ間の標準ファイルアクセスプロトコルのみを扱うため、クライアント・サーバともに特別な対応は一切必要がない。

【００６９】

また、本発明においては、制御装置は、二次ストレージへの移動をファイルアクセスプロトコルのレベルで完全に隠蔽するため、クライアントでは、二次ストレージへ移動されたことに全く気づくことなく、運用が可能になる。

本発明において、スタブファイル内に複数の分割ファイルのパス名を格納しておくことで最大ファイルサイズの拡大を可能としている。各分割ファイルは最大でリードライトサイズ分だけオーバーラップ領域を設けておき、１コマンドのアクセスで複数の分割ファイルにまたがることがないようにしている。

オーバーラップ領域の書き込みについては、２つの分割ファイルで同期をとる。

最大ファイルサイズ以下のある閾値になったら、あらたな分割ファイルとして、スタブファイルにパス名を登録する。

元のファイルを最初にスタブ化する場合、元ファイルをRENAMEすることで行う。そのため最初の分割ファイルは元のファイルと同一ファイルシステム内に配置される。これ以降に生成される分割ファイルは、異なるファイルシステムに分散配置されてもよい。

広域分散ネットワークファイルシステム（複数拠点）に、スタブ、実ファイルを分散配置し相互参照する構成としてもよい。

実ファイルが自拠点のファイルシステムに存在する場合、更新により属性が変更した場合、他拠点のスタブファイルを更新する。

実ファイルが他拠点のファイルシステムに存在する場合、自拠点のスタブファイル内の実ファイル情報に基づき、他拠点の実ファイルに転送する。

他拠点の実ファイルを中間装置がキャッシュし、自拠点のスタブファイルにキャッシュ情報を格納しておく。アクセスがあったら、自拠点のスタブファイルのキャッシュ情報と実ファイルの属性を比較し、キャッシュが有効であれば、キャッシュを用いて応答する。

より詳細には、１つの拠点のファイルシステムが、他の拠点の実ファイルをキャッシュするキャッシュ領域を備え、前記スタブファイルにキャッシュ情報を記録しておき、

前記中間装置は、キャッシュされている実ファイルに対応する前記スタブファイルのキャッシュ情報を格納する記憶部を備え、前記１つの拠点の前記中間装置は、前記クライアントからのアクセス要求を受け、スタブファイルのキャッシュ情報が、前記中間装置の記憶部に格納されているかチェックし、格納されている場合、前記キャッシュ領域の実ファイルを用いて、前記クライアントからのアクセス要求を処理する。

スタブ化のトリガーとして、ストレージ容量が閾値に達したら、スタブ化を行う。その際、アクセスがないファイルからスタブ化を行うようにしてもよい。

複数拠点からなる広域分散環境では、ユーザのアクセス情報に基づき、アクセスの多い拠点に実ファイルを置き、他拠点では、スタブ化する。

クオータリミットが設定されている場合、それより小さい閾値を設定し、該閾値に達したら、アクセスの古いファイルを下階層に移動する等の制御を行うようにしてもよい。

【発明の効果】

【００７０】

本発明によれば、既存のクライアント・サーバに変更を加えることなく階層ストレージ管理を実現でき、管理工数の大幅な削減ができる。

【００７１】

さらに、本発明によれば、二次ストレージへの移動をクライアントに完全に隠蔽できるため、クライアントは階層ストレージ管理を意識することなく、システムの運用を可能としている。

本発明によれば、スタブファイルに複数の分割ファイルのパス情報を備える構成としたことにより、最大ファイルサイズを拡大することができる。

また、本発明によれば、複数拠点間でのスタブ、実ファイルの相互参照を可能としている。アクセス頻度の高い拠点に実ファイルを置き、アクセス頻度の低い拠点をスタブ化等することで拠点間（クラスタ間）の通信の増大を抑止低減し、拠点間相互アクセスにおける性能低下を回避可能としている。

また本発明によれば、クオータ（Ｑｕｏｔａ）設定値に達するまえに、スタブ化を自動で行うことで、クオータエラーによるアクセスロック等の発生を事前に回避することができる。

【発明を実施するための最良の形態】

【００７２】

本発明の実施の形態について図面を用いて説明する。図１は、本発明に係る中間装置を用いた一実施形態のシステム構成を示す図である。図１を参照すると、本発明の一実施形態に係るシステムは、クライアント１と、本発明の制御装置を構成する中間装置３とは、ネットワーク２を介して接続されており、中間装置３とファイルサーバ５及びファイルサーバ６とはネットワーク４を介して接続されている。ファイルサーバ５は、相対的に高速・小容量な一次ストレージであり、ファイルサーバ６は、相対的に低速・大容量な二次ストレージである。ネットワーク２、４は、例えばＩＰ網が用いられる。クライアント１は、例えば標準ファイルアクセスプロトコル（ＮＦＳ（Network File System）やＣＩＦＳ（Common Internet File System）等）で、ファイルサーバ５（一次ストレージ）にアクセスする。

【００７３】

クライアント１は、ファイルはファイルサーバ５（一次ストレージ）にあるものとしてアクセスする。

【００７４】

ファイルサーバ６（二次ストレージ）に移動されたファイルは、中間装置３がアクセスをファイルサーバ６（二次ストレージ）にアクセス先を変更する処理を行う。

【００７５】

ファイルサーバ６（二次ストレージ）へのアクセスは、中間装置３が行うため、クライアント１は、ファイルサーバ６（二次ストレージ）にアクセスしていることは気づかず、完全に隠蔽される。なお、本発明の制御装置は、クライアントとファイルサーバのファイ

ルアクセスプロトコルの中間に位置付けされる中間装置３で構成されているが、かかる構成に制限されるものでなく、ファイルサーバ５（一次ストレージ）内に設ける構成としてもよい。また、本発明の制御装置は、ソフトウェアモジュールで構成してもよいことは勿論である。

【００７６】

図２は、本発明の一実施形態の動作原理を説明するための図である。図２を参照して、中間装置３がファイルサーバ５（一次ストレージ）からファイルサーバ６（二次ストレージ）にアクセスを変更する処理をするために用いる方法について説明する。

【００７７】

ファイルサーバ６（二次ストレージ）には、実ファイル８（ファイルサーバ５（一次ストレージ）から移動されたファイル）が置かれ、ファイルサーバ５（一次ストレージ）には、実ファイル８に対応したスタブファイル７が置かれている。そして、ファイルサーバ５（一次ストレージ）のスタブファイル７には、ファイルサーバ６（二次ストレージ）に移動された実ファイル８のパス名が格納されている。

【００７８】

中間装置３は、クライアント１がファイルサーバ５（一次ストレージ）にアクセスしたとき、これがスタブファイル７である場合、スタブファイル７に格納されているパス名を使って、ファイルサーバ６（二次ストレージ）の実ファイル８にアクセスして、ファイルサーバ６（二次ストレージ）からの応答を受け、クライアント１に返す。これにより、クライアント１は、あたかもファイルサーバ５（一次ストレージ）上に実ファイル８があるようにファイル操作ができる。

【００７９】

以下、中間装置３の処理を具体的に説明する。

【００８０】

まず、中間装置３は、クライアント１がアクセスしたファイルがスタブファイル７であるか否かを判定する必要がある。そのために、本実施の形態では、スタブファイル７は、以下のようなファイルとする。

【００８１】

・ファイルのタイムスタンプに、スタブファイルを示すＩＤを入れる（ＣＩＦＳでは作成時刻：Ｃｒｅａｔｅ　Ｔｉｍｅを使用。ＮＦＳではＭｔｉｍｅを使用）

【００８２】

・ファイルサイズを例えば１ＫＢ固定とする。ファイルサイズは、通常のファイルと重複しにくい値にするとよりよい。すなわち、ファイルサイズの値でスタブファイル又はその候補と判別できるためである。

【００８３】

・ファイルデータの先頭に、マジックナンバーを入れる。

【００８４】

このように、ファイル属性（ファイルタイムスタンプとファイルサイズ）でスタブファイルを判定できるため、高速な判定が可能である。

【００８５】

また、本実施形態によれば、ファイル先頭にマジックナンバーを入れておくことで、スタブファイルの誤認識を防止し、スタブファイルの確実な判別を可能としている。

【００８６】

図３は、本発明の一実施形態における、スタブファイル７のフォーマットを示す図である。図３を参照すると、スタブファイル７の先頭は、スタブ識別用マジックナンバー７Ａである。これは、スタブファイルであることを最終的に確認するためのもので、ある程度の長さのマジックナンバーとする。

【００８７】

実ファイル属性７Ｂは、実ファイル８の属性（ファイルサイズ、更新日時等）を格納しておく。こうすることで、属性のみを返すようなリクエストについては、ファイルサーバ

６（二次ストレージ）にアクセスすることなく、応答を返すことができる。

【００８８】

二次ストレージ実ファイルパス名７Ｃは、ファイルサーバ６（二次ストレージ）内の実ファイル８のパス名である。特に制限されないが、この例では、ファイルサーバ６がパス名で、ファイルアクセスするサーバであるため、二次ストレージ内の実ファイルパス名７Ｃとなる。ただし、パス名でなく、何らかのＩＤでファイルを特定するサーバであれば、ＩＤが用いられ、ブロックアドレスやＵＲＬ（Uniform Resource Locator）のようなものを用いてもよい。すなわち、二次ストレージ実ファイルパス名７Ｃは、これをもとに、中間装置３は実ファイル８にアクセスすることができるものであればよい。

【００８９】

リザーブ７Ｄは、前述のように、スタブファイル７を固定長とするために付加される空き領域である。

【００９０】

なお、スタブファイル７のフォーマットは、必ずしも、この順番である必要はないし、図３に示した要素以外にさらに別の要素を設ける構成としてもよいことは勿論である。

【００９１】

スタブファイル７を使う利点は、ファイルサーバ６（二次ストレージ）に移動したファイルのテーブルを、中間装置３が持たなくてよいことである。

【００９２】

例えばスタブファイル７を用いずに、中間装置３が全ての情報をもつ構成も考えられるが、その場合、クライアント１からの全てのリクエストについて、中間装置３内部で、テーブルエントリとの比較を行う必要がある。

【００９３】

これに対して、スタブファイル７を使うシステムでは、中間装置３は、クライアント１からのコマンドを、ファイルサーバ５（一次ストレージ）へ転送し、中間装置３が、ファイルサーバ５からの応答の中に含まれるファイル属性を見ることで、スタブファイルであることを判定することができる。このため、中間装置３で、全スタブファイルの情報をもつことは不要とされ、中間装置３の記憶容量によってスタブファイル７の数が制限されることがない。また、中間装置３の記憶容量の逼迫等により、転送速度が低下することもなく、高速転送が可能となる。

【００９４】

次に、中間装置３の内部で持つテーブルについて説明する。ＣＩＦＳプロトコルの場合、図４に示す２つのテーブルを持つ。スタブファイルＩＤテーブル１１は、クライアント１がスタブファイル７をオープンしている場合（クライアント１にとっては実ファイル８をオープンしているように見える）に作成されるテーブルである。スタブファイルＩＤテーブル１１は、一次ストレージＦＩＤ（１１Ａ）、二次ストレージＦＩＤ（１１Ｂ）、スタブ管理テーブルポインタ（１１Ｃ）を有する。

【００９５】

一次ストレージＦＩＤ（１１Ａ）は、ファイルサーバ５（一次ストレージ）のスタブファイル７のファイルＩＤ（ファイルＩＤについては後述）である。

【００９６】

二次ストレージＦＩＤ（１１Ｂ）はファイルサーバ６（二次ストレージ）の実ファイル８のファイルＩＤである。

【００９７】

スタブ管理テーブルポインタ（１１Ｃ）は、対応するスタブ管理テーブル１２へのポインタとなっている。

【００９８】

スタブ管理テーブル１２は、スタブファイル７にクライアント１がアクセスしたときにスタブファイル７の内容を、中間装置３でキャッシュするテーブルであり、一次ストレージファイルパス名（１２Ａ）、二次ストレージファイルパス名（１２Ｂ）、実ファイル属

性（１２Ｃ）を有する。
【００９９】
一次ストレージファイルパス名（１２Ａ）は、ファイルサーバ５（一次ストレージ）のスタブファイル７のパス名である。
【０１００】
二次ストレージファイルパス名（１２Ｂ）は、ファイルサーバ６（二次ストレージ）の実ファイル８のパス名である。これは、スタブファイル７内の二次ストレージ実ファイルパス名７Ｃと同じ内容が設定される。
【０１０１】
実ファイル属性（１２Ｃ）は、実ファイル８の属性であり、スタブファイル７内の実ファイル属性７Ｂと同じ内容が設定される。
【０１０２】
ただし、クライアント１から更新が入り、実ファイルの属性が変更された場合（例えばＷＲＩＴＥ（書き込み）してファイルサイズが大きくなった場合など）に、毎回スタブファイル７内の実ファイル属性７Ｂを書き換えるのではなく、中間装置３内のスタブ管理テーブル１２を書き換えるだけで対応するときがある。その場合、スタブファイル７内の実ファイル属性７Ｂとスタブ管理テーブル１２の実ファイル属性１２Ｃが異なる。
【０１０３】
次に、クライアント１が中間装置３を経由してアクセスするときの中間装置３の動作を説明する。
【０１０４】
特に制限されないが、以下では、ファイルアクセスプロトコルとして、Ｗｉｎｄｏｗｓ（登録商標）環境のデフォルトで使われているＣＩＦＳプロトコルの場合を例に説明する。他のファイルアクセスプロトコルでも、コマンド形式が多少異なるが、基本的な動作は同じであり、本発明を適用することができる。
【０１０５】
＜ＯＰＥＮ（対象をパス名で指定）＞
図５は、クライアント１からＯＰＥＮ（オープン）リクエストが発行され中間装置３で受けたときの処理手順を示す流れ図である。
【０１０６】
クライアント１がアクセスを開始するときには、最初に、パス名を指定してＯＰＥＮリクエスト（ＯＰＥＮＲｅｑ（ＰＡＴＨ））を発行する。ＯＰＥＮリクエストに対してファイルサーバ５、６は成功か失敗、成功ならファイルＩＤ（ＦＩＤ）とファイル属性を応答する。
【０１０７】
クライアント１は、今回のＯＰＥＮリクエスト以降のＣＬＯＳＥまでの間、ＯＰＥＮリクエストに対するファイルサーバからの応答で渡されるファイルＩＤを使ってＲＥＡＤ（読込み）やＷＲＩＴＥ（書込み）などのアクセスを行う。この仕組みは、ほとんどのファイルアクセスプロトコルに共通である。
【０１０８】
中間装置３は、クライアント１からのＯＰＥＮリクエスト（ＯＰＥＮＲｅｑ（ＰＡＴＨ））を受け取り、これをそのままファイルサーバ５（一次ストレージ）に転送する（ステップ１０１）。
【０１０９】
そして、中間装置３は、ファイルサーバ５（一次ストレージ）からの応答（ＯＰＥＮＲｅｓｐ（ＦＩＤ、属性））を受け取り、該応答に含まれるファイル属性をチェックする（ステップ１０２）。
【０１１０】
ファイル属性がスタブファイル属性でない場合には（ステップ１０２のＮＯ判定）、中間装置３は、ファイルサーバ５からの応答（ＯＰＥＮＲｅｓｐ（ＦＩＤ、属性））をその

まま、クライアント１に転送する（ステップ１０６）。スタブファイル属性とは、前述したように、ファイル属性のうちタイムスタンプとファイルサイズがスタブファイルであることを示す場合である。

【０１１１】

中間装置３は、スタブファイル属性であった場合には（ステップ１０２のＹＥＳ判定）、確認のためにファイル先頭をＲＥＡＤして（ＲＥＡＤＲｅｑ（ＦＩＤ））、ファイル先頭のマジックナンバーを確認する（ステップ１０３）。スタブファイル７でなければ（ステップ１０３のＮＯ判定）、中間装置３は、ファイルサーバ５からの応答（ＯＰＥＮＲｅｓｐ（ＦＩＤ、属性））をそのままクライアント１に転送する（ステップ１０６）。

【０１１２】

スタブファイル７であった場合（ステップ１０３のＹＥＳ判定）、中間装置３は、スタブファイルＩＤテーブル１１とスタブ管理テーブル１２を作成する（ステップ１０４）。ただし、中間装置３において、当該スタブファイル７のスタブ管理テーブル１２がすでに存在する場合には、新たに作成はしない。これは、他のクライアント１がすでにこのスタブファイルをＯＰＥＮしている場合等、あるいはＦＩＮＤ系コマンドなどＯＰＥＮしないでパス名で指定するコマンドが来たときに作成されている場合に相当する。

【０１１３】

次に、中間装置３は、ファイルサーバ５（１次ストレージ）からの応答（ＯＰＥＮＲｅｓｐ（ＦＩＤ、属性））内のファイル属性を、スタブ管理テーブル１２内の実ファイル属性（１２Ｃ）に書き換え（ステップ１０５）、応答（ＯＰＥＮＲｅｓｐ（ＦＩＤ、属性））をクライアントに転送する（ステップ１０６）。

【０１１４】

これにより、クライアント１は、実ファイル８の属性が渡されるため、スタブファイル７であることが隠蔽される。なお、上記した中間装置３の処理は、中間装置３を構成するコンピュータで実行されるプログラムによって実現してもよい。

【０１１５】

＜ＲＥＡＤ／ＷＲＩＴＥ（対象をファイルＩＤで指定）＞

図６は、ＯＰＥＮ後に、クライアント１からのＲＥＡＤ／ＷＲＩＴＥリクエストが発行されたときの処理手順を示す流れ図である。なお、図６では、ＲＥＡＤリクエスト発行時の処理手順が示されている。

【０１１６】

クライアント１は、ＲＥＡＤリクエスト（ＲＥＡＤＲｅｑ（ＦＩＤ））において、ファイルＩＤ（ＦＩＤ）を指定して来る（ステップ１１１）。

【０１１７】

中間装置３は、クライアント１からのＲＥＡＤリクエスト（ＲＥＡＤＲｅｑ（ＦＩＤ））を受け取り、ファイルＩＤをスタブファイルＩＤテーブル１１から検索する（ステップ１１２）。ステップ１１２での検索の結果、当該ファイルＩＤがスタブファイルＩＤテーブル１１に存在しなければ（ステップ１１３のＮＯ分岐）、スタブファイルではないことがわかる。このため、中間装置３は、ＲＥＡＤリクエスト（ＲＥＡＤＲｅｑ（ＦＩＤ））を、通常どおり、そのままファイルサーバ５（一次ストレージ）へ転送する。そして、中間装置３は、ファイルサーバ５（一次ストレージ）からの応答もそのままクライアント１へ返す。

【０１１８】

一方、中間装置３において、ファイルＩＤがスタブファイルＩＤテーブル１１に存在する場合（ステップ１１３のＹＥＳ分岐）、中間装置３は、スタブファイルＩＤテーブル１１の二次ストレージＦＩＤ（１１Ｂ）が有効であるか否かをチェックする（ステップ１１４）。二次ストレージＦＩＤ（１１Ｂ）が有効であるか否かは、スタブファイルＩＤテーブルの作成時に、二次ストレージＦＩＤ（１１Ｂ）をＮＵＬＬ（ヌル）に設定しておき、判定の際に、ＮＵＬＬ以外であれば有効と判断できる。

【０１１９】

ステップ１１４の判定の結果、二次ストレージＦＩＤ（１１Ｂ）が有効である場合（ステップ１１４のＹＥＳ分岐）、中間装置３は、ファイルサーバ６（二次ストレージ）に、二次ストレージＦＩＤ（１１Ｂ）を使ってＲＥＡＤ要求を転送し（１１６）、ファイルサーバ６（二次ストレージ）からの応答をクライアント１へ転送する（ステップ１１７）。

【０１２０】

一方、ステップ１１４の判定の結果、二次ストレージＦＩＤ（１１Ｂ）が無効の場合（ステップ１１４のＮＯ分岐）、中間装置３は、ＯＰＥＮ後の最初のＲＥＡＤやＷＲＩＴＥであるため、ファイルサーバ６（二次ストレージ）の実ファイル８はＯＰＥＮされていない。そのため、中間装置３は、ファイルサーバ６（二次ストレージ）の実ファイル８をスタブ管理テーブル１２内の二次ストレージファイルパス名（１２Ｂ）を用いてＯＰＥＮし、ファイルサーバ６（二次ストレージ）の応答のファイルＩＤを、スタブファイルＩＤテーブル１１の二次ストレージＦＩＤ（１１Ｂ）に登録する（ステップ１１５）。すなわち、中間装置３は、ファイルサーバ６（二次ストレージ）にＯＰＥＮリクエスト（ＯＰＥＮＲｅｑ（ＰＡＴＨ２））を送信し、ファイルサーバ６（二次ストレージ）からの応答（ＯＰＥＮＲｅｓｐ（ＦＩＤ２、属性）から二次ストレージのファイルＩＤ（ＦＩＤ２）をスタブファイルＩＤテーブル１１の二次ストレージＦＩＤ（１１Ｂ）に格納する。これにより、二次ストレージＦＩＤ（１１Ｂ）は初期設定された値ＮＵＬＬでなくなる。

【０１２１】

その後、中間装置３は、二次ストレージＦＩＤ（１１Ｂ）を使ってＲＥＡＤ要求（ＲＥＡＤＲｅｑ（ＦＩＤ２））をファイルサーバ６（二次ストレージ）に転送し（ステップ１１６）、中間装置３は、ファイルサーバ６（二次ストレージ）からの応答（ＲｅａｄＲｅｓｐ）を受け、該応答を、クライアント１へ転送する（ステップ１１７）。

【０１２２】

クライアント１からのアクセス要求がＷＲＩＴＥリクエストの場合、書き込みの結果、ファイルサイズが変わる場合がある。ファイルサーバ５（一次ストレージ）のスタブファイル７内に実ファイル属性７Ｂを備えているため、変更する必要があるが、本実施形態では、実ファイルのファイルサイズの変更があったとき、中間装置３でキャッシュしているスタブ管理テーブル１２内の実ファイル属性１２Ｃを変更するのみにしておき、ＣＬＯＳＥ時に、スタブファイル７に書き戻すようにしている。

【０１２３】

これにより、毎回ファイルサーバ５（一次ストレージ）を書き換える作業がなくなり、高速転送が可能となる。なお、上記した中間装置３の処理は、中間装置３を構成するコンピュータで実行されるプログラムによって実現してもよい。

【０１２４】

＜属性ＧＥＴ系コマンド（対象をファイルＩＤで指定）＞

図７は、属性ＧＥＴ系で対象をファイルＩＤで指定するコマンドが発行された場合の処理手順を示す流れ図である。中間装置３では、クライアント１から来た属性ＧＥＴリクエスト（ＧＥＴＲｅｑ（ＦＩＤ））をファイルサーバ５（一次ストレージ）に転送する（ステップ１２１）。

【０１２５】

中間装置３は、属性ＧＥＴリクエスト（ＧＥＴＲｅｑ（ＦＩＤ））のファイルＩＤ（ＦＩＤ）でスタブファイルＩＤテーブル１１を検索し（ステップ１２２）、スタブファイルか否かを判別する（ステップ１２３）。

【０１２６】

中間装置３は、ステップ１２３の判別の結果、スタブファイルでなければ（ステップ１２３のＮＯ分岐）、ファイルサーバ５（一次ストレージ）からの応答（ＧＥＴＲｅｓｐ）をそのままクライアント１へ転送する（ステップ１２５）。

【０１２７】

一方、中間装置３は、ステップ１２３の判別の結果、スタブファイルであれば（ステップ１２３のＹＥＳ分岐）、応答（ＧＥＴＲｅｓｐ（属性））に含まれるファイル属性をス

タブ管理テーブル１２内の実ファイル属性１２Ｃに書き換えた上（ステップ１２４）、応答（ＧＥＴＲｅｓｐ（属性））をクライアントに転送する（ステップ１２５）。

【０１２８】

図７において、属性ＧＥＴリクエスト（ＧＥＴＲｅｑ（ＦＩＤ））に含まれるファイルＩＤから検索すれば、中間装置３がファイルサーバ５（一次ストレージ）に属性ＧＥＴリクエスト（ＧＥＴＲｅｑ（ＦＩＤ））を転送することは必要ないように見えるが、クライアント１のアクセス権チェック等の処理をファイルサーバ５（一次ストレージ）で行うため、中間装置３は、必ず、属性ＧＥＴリクエスト（ＧＥＴＲｅｑ（ＦＩＤ））をファイルサーバ５（一次ストレージ）へ転送し、もし、ファイルサーバ５（一次ストレージ）からの応答がエラーであれば、エラーのままクライアントへ返却する必要がある。

【０１２９】

＜属性ＧＥＴ系コマンド（対象をパスで指定）＞

図８は、属性ＧＥＴ系で対象がパス名で指定されるコマンドが発行された場合の処理手順を示す流れ図である。

【０１３０】

中間装置３は、クライアント１から来た属性ＧＥＴリクエスト（ＧＥＴＲｅｑ（ＰＡＴＨ））をファイルサーバ５（一次ストレージ）へ転送する（ステップ１３１）。

【０１３１】

ファイルサーバ５（一次ストレージ）からの応答（ＧＥＴＲｅｓｐ（属性））には、ファイル属性が含まれるため、中間装置３は、該ファイル属性がスタブファイル属性か判別し（ステップ１３２）、スタブファイルでなければ（ステップ１３２のＮＯ分岐）、クライアント１へ応答（ＧＥＴＲｅｓｐ）をそのまま転送する（ステップ１３７）。

【０１３２】

中間装置３は、属性がスタブファイル属性である場合（ステップ１３２のＹＥＳ分岐）、ファイルサーバ５（一次ストレージ）のスタブファイル７をオープンして、その中身をＲＥＡＤしてスタブファイル７をクローズする（ステップ１３３）。

【０１３３】

中間装置３は、読み出した先頭のマジックナンバー（図３参照）を確認し、スタブファイルか否かを判別する（ステップ１３４）。スタブファイルでなければ（ステップ１３４のＮＯ分岐）、中間装置３は、応答（ＧＥＴＲｅｓｐ）をそのままクライアントへ転送する。

【０１３４】

スタブファイルであれば（ステップ１３４のＹＥＳ分岐）、中間装置３は、スタブ管理テーブル１２を作成し（ステップ１３５）、応答（ＧＥＴＲｅｓｐ（属性））のファイル属性を、スタブファイル７の実ファイル属性７Ｂに入れ替えて（ステップ１３６）、クライアント１に転送する（ステップ１３７）。

【０１３５】

ステップ１３５におけるスタブ管理テーブルの作成処理は、以後も同一ファイルがアクセスされるときのキャッシュとして作成しておくものであるため、実行しないことも可能である。

【０１３６】

＜属性ＳＥＴ系コマンド（対象をファイルＩＤで指定）＞

図９は、属性ＳＥＴ系で対象をファイルＩＤで指定するコマンドが発行された場合の処理手順を示す流れ図である。属性ＳＥＴ系は、ＧＥＴ系と異なり、そのままファイルサーバ５（一次ストレージ）へ転送してしまうと、スタブファイルであった場合に、スタブファイル属性を書き換えてしまい、この結果、中間装置３がスタブファイルを認識できなくなる。そこで、本実施形態では、必ず、スタブファイル判定を行ってから転送を行う。

【０１３７】

まず、中間装置３が、クライアント１から属性ＳＥＴリクエスト（ＳＥＴＲｅｑ（ＦＩＤ））を受けると（ステップ１４１）、ＲＥＡＤ／ＷＲＩＴＥリクエストと同様に、ファ

イルＩＤでスタブファイルＩＤテーブル１１を検索し（ステップ１４２）、スタブファイルでない場合には（ステップ１４３のＮＯ分岐）、ファイルサーバ５（一次ストレージ）に、該属性ＳＥＴリクエストをそのまま転送する。

【０１３８】

スタブファイルであった場合には（ステップ１４３のＹＥＳ分岐）、中間装置３は、スタブファイルＩＤテーブル１１の二次ストレージＦＩＤ（１１Ｂ）が有効であるか否か（二次ファイルＩＤがあるか否か）をチェックし（ステップ１４４）、有効である場合（ステップ１４４のＹＥＳ分岐）、中間装置３は、属性を二次ストレージＦＩＤ（１１Ｂ）に入れ替えた属性ＳＥＴリクエスト（ＳＥＴＲｅｑ（ＦＩＤ２））を、ファイルサーバ６（二次ストレージ）に転送し、ファイルサーバ６（二次ストレージ）からの応答（ＳＥＴＲｅｓｐ）を受け取る（ステップ１４７）。

【０１３９】

応答（ＳＥＴＲｅｓｐ）が、属性ＳＥＴの成功である場合、スタブ管理テーブル１２内の実ファイル属性１２Ｃを書き換え（ステップ１４８）、応答（ＳＥＴＲｅｓｐ）をクライアントに転送する（ステップ１４９）。

【０１４０】

一方、二次ストレージＦＩＤ（１１Ｂ）が無効である場合には（ステップ１４４のＮＯ分岐）、中間装置３は、ファイルサーバ６（二次ストレージ）にＯＰＥＮリクエスト（ＯＰＥＮＲｅｑ（ＰＡＴＨ２））を送信して、二次ストレージのファイルパス名（１２Ｂ）をＯＰＥＮし（ステップ１４５）、ファイルサーバ６（二次ストレージ）からの応答（ＯＰＥＮＲｅｓｐ（ＦＩＤ２，属性））のファイルＩＤ（ＦＩＤ２）を、スタブファイルＩＤテーブルの二次ストレージＦＩＤ（１１Ｂ）に登録する（ステップ１４６）。これ以降、二次ストレージＦＩＤ（１１Ｂ）が有効であった場合と同様、属性ＳＥＴリクエスト（ＳＥＴＲｅｑ（ＦＩＤ２））を、ファイルサーバ６（二次ストレージ）に転送し（ステップ１４７）、スタブ管理テーブル１２内の実ファイル属性１２Ｃを書き換えて（ステップ１４８）、クライアントに転送する（ステップ１４９）。

【０１４１】

＜属性ＳＥＴ系コマンド（対象をパス名で指定）＞

図１０は、属性ＳＥＴ系で対象をパス名で指定するコマンドが発行された場合の処理手順を示す流れ図である。この場合も、図９を参照して説明した手順と同様、転送する前に、スタブファイル判定を実行する必要がある。図９と異なるのは、スタブファイルＩＤテーブル１１が作成されていないため、実際にファイルサーバ５（一次ストレージ）にアクセスして判定する点である。

【０１４２】

まず、中間装置３が、クライアント１から属性ＳＥＴリクエスト（ＳＥＴＲｅｑ（ＰＡＴＨ、属性））を受け取ると、ファイルサーバ５（一次ストレージ）へそのパス名を使って属性ＧＥＴリクエスト（ＧＥＴＲｅｑ（属性））を送る（ステップ１５１）。中間装置３は、ファイルサーバ５（一次ストレージ）からの応答（ＧＥＴＲｅｓｐ（属性））の属性がスタブファイル属性かを判定し（ステップ１５２）、スタブファイルでない場合（ステップ１５２のＮＯ分岐）、属性ＳＥＴリクエスト（ＳＥＴＲｅｑ（ＰＡＴＨ、属性））を、ファイルサーバ５（一次ストレージ）へ転送し、ファイルサーバ５（一次ストレージ）からの応答（ＳＥＴＲｅｓｐ）をクライアント１へ転送する（ステップ１５９）。

【０１４３】

スタブファイル属性であった場合には（ステップ１５２のＹＥＳ分岐）、中間装置３は、スタブ識別用マジックナンバー７Ａを得るために、ファイルサーバ５（一次ストレージ）を、そのパス名でＯＰＥＮし、ファイル先頭をＲＥＡＤし、ＣＬＯＳＥする（ステップ１５３）。

【０１４４】

スタブ識別用マジックナンバー７Ａでなければ（ステップ１５４のＮＯ分岐）、スタブファイルではないので、中間装置３は、属性ＳＥＴリクエスト（ＳＥＴＲｅｑ（ＰＡＴＨ

、属性））をファイルサーバ５（一次ストレージ）へ転送し、ファイルサーバ５（一次ストレージ）からの応答（ＳＥＴＲｅｓｐ）をクライアント１へ転送する（ステップ１５９）。

【０１４５】

スタブファイルと確認できた場合には（ステップ１５４のＹＥＳ分岐）、中間装置３は、まず、ファイルサーバ５（一次ストレージ）へ属性ＳＥＴリクエスト（ＳＥＴＲｅｑ（ＰＡＴＨ、属性＆ｓｔｕｂ））を転送する（ステップ１５５）。その際に、ＳＥＴする属性に、スタブファイル属性ｓｔｕｂを埋め込んで転送する。こうすることで、スタブファイル属性を失わないようにする。

【０１４６】

その後、スタブファイル７内の実ファイル属性を書き換えるために、中間装置３は、スタブファイル７をＯＰＥＮし、実ファイル属性をスタブファイル７にＷＲＩＴＥした後、ＣＬＯＳＥする（ステップ１５６）。

【０１４７】

対象をファイルＩＤで指定する属性ＳＥＴ系コマンドの場合は、すでに、クライアント１からＯＰＥＮされているため、中間装置３において、スタブ管理テーブル１２が作成されており、属性の変更は、スタブ管理テーブル１２にのみ反映し、クライアント１からＣＬＯＳＥが来たときに反映すればよいが、対象をパス名で指定する属性ＳＥＴ系コマンドの場合は、ＯＰＥＮしていないので、スタブ管理テーブルが存在せず、スタブファイル７内の実ファイル属性７Ｂに反映する必要がある。

【０１４８】

その後、中間装置３は、ファイルサーバ６（二次ストレージ）に、属性ＳＥＴリクエスト（ＳＥＴＲｅｑ（ＰＡＴＨ、属性））を転送し（ステップ１５７）、ファイルサーバ６（二次ストレージ）からの応答（ＳＥＴＲｅｓｐ）を受け取って、クライアント１へ転送する（ステップ１５８）。

【０１４９】

＜ＲＥＡＤＤＩＲ、ＦＩＮＤ系コマンド（対象をパス名で指定）＞

図１１は、ＲＥＡＤＤＩＲ・ＦＩＮＤ系コマンドの場合の処理手順を示す流れ図である。

【０１５０】

まず、中間装置３が、クライアント１からＦＩＮＤ系リクエスト（ＦＩＮＤＲｅｑ（ＰＡＴＨ））を受け取ると、ファイルサーバ５（一次ストレージ）に転送する（ステップ１６１）。中間装置３は、ファイルサーバ５（一次ストレージ）からの応答（ＦＩＮＤＲｅｓｐ（属性））に含まれる属性がスタブファイル属性であるか否かの判定を行い（ステップ１６２）、スタブファイルでなければ（ステップ１６２のＮＯ分岐）、そのまま応答（ＦＩＮＤＲｅｓｐ（属性））をクライアント１に転送する（ステップ１６８）。

【０１５１】

一方、スタブファイル属性の場合には（ステップ１６２のＹＥＳ分岐）、中間装置３は、スタブ管理テーブル１２を検索し（ステップ１６３）、検索の結果、存在する場合（ステップ１６４のＹＥＳ分岐）、応答（ＦＩＮＤＲｅｓｐ（属性））の属性を、実ファイル属性１２Ｃに書き換えて（ステップ１６７）、該属性を書き換えた応答をクライアント１に転送する（ステップ１６８）。

【０１５２】

ステップ１６３におけるスタブ管理テーブル１２の検索の結果、見つからない場合（ステップ１６４のＮＯ分岐）、中間装置３は、スタブ識別用マジックナンバー７Ａを確認するため、ファイルサーバ５（一次ストレージ）に対して、ＯＰＥＮ、ＲＥＡＤ、ＣＬＯＳＥを行う（ステップ１６５）。

【０１５３】

その結果（読み出した結果）、スタブファイル７であれば（ステップ１６６のＹＥＳ分岐）、中間装置３は、属性を書き換え（ステップ１６７）、スタブファイル７でなければ

（ステップ１６６のＮＯ分岐）、クライアント１への応答転送の処理（ステップ１６８）に進む。この際、今後も、クライアントからＦＩＮＤ系アクセスが来るのに備えて、スタブ管理テーブル１２を作成しておいてもよい。

【０１５４】

＜ＣＬＯＳＥ（対象をファイルＩＤで指定）＞

図１２は、ＣＬＯＳＥコマンドが発行された場合の処理手順を示す流れ図である。

【０１５５】

中間装置３は、クライアント１からＣＬＯＳＥリクエスト（ＣＬＯＳＥＲｅｑ（ＦＩＤ））を受け取ると、ファイルＩＤを取り出してスタブファイルＩＤテーブル１１を検索する（ステップ１７１）。

【０１５６】

ファイルＩＤ（ＦＩＤ）がスタブファイルＩＤテーブル１１に存在しなければ（ステップ１７２のＮＯ分岐）、スタブファイルではないので、中間装置３は、ＣＬＯＳＥリクエスト（ＣＬＯＳＥＲｅｑ（ＦＩＤ））を、ファイルサーバ５（一次ストレージ）に転送し、応答をクライアント１に転送する。

【０１５７】

一方、スタブファイルＩＤテーブル１１が存在する場合（ステップ１７２のＹＥＳ分岐）、スタブファイルであることがわかるので、中間装置３は、まず、スタブファイル７をＣＬＯＳＥするためファイルサーバ５（一次ストレージ）に、ＣＬＯＳＥリクエスト（ＣＬＯＳＥＲｅｑ（ＦＩＤ））を転送する（ステップ１７３）。

【０１５８】

スタブファイルＩＤ管理テーブル１１の二次ストレージＦＩＤ（１１Ｂ）が有効であれば（ステップ１７４のＹＥＳ分岐）、実ファイル８がＯＰＥＮされているので、中間装置３は、ファイルサーバ６（二次ストレージ）の実ファイル８をＣＬＯＳＥリクエスト（ＣＬＯＳＥＲｅｑ（ＦＩＤ２））を送信する（ステップ１７５）。

【０１５９】

ＯＰＥＮ中にＷＲＩＴＥがあってファイルサイズや属性が変わっている場合（ステップ１７６のＹＥＳ分岐）があることから、その場合、中間装置３は、スタブファイル７をＯＰＥＮして、内部の実ファイル属性７Ｂを書き換えておく（ステップ１７７のＯＰＥＮ－ＷＲＩＴＥ－ＣＬＯＳＥ）。

【０１６０】

最後に、中間装置３は、クライアント１に応答（ＣＬＯＳＥＲｅｓｐ）を返す（ステップ１７８）。なお、図７乃至図１３を参照して説明した中間装置３の処理は、中間装置３を構成するコンピュータで実行されるプログラムによって実現してもよい。

【０１６１】

次に、通常ファイルをスタブファイルにする時の動作について説明する。図１３は、ファイルサーバ５（一次ストレージ）からファイルサーバ６（二次ストレージ）へファイルを移動し、スタブファイル化するフローを説明するための図である。

【０１６２】

まず、中間装置３がファイルサーバ５（一次ストレージ）からファイルサーバ６（二次ストレージ）にファイルをコピーする（ステップ１）。この間にクライアントから対象ファイルに対してアクセスがあった場合はコピーを中止し、一定時間後に、再度コピーを再開するなどの制御を行う。

【０１６３】

コピーが終了したら、中間装置３では、対象ファイルへのクライアント１からのアクセスを停止する（ステップ２）。アクセスの停止の仕方は、

・リクエストをドロップする方式や、

・リクエストを一時的にキューイングする方式

等を用いることができる。いずれにしても、ファイルサーバ５の対象ファイルに対してクライアント１のアクセスが来ないようにする。アクセスを止めている間に、中間装置３

がファイルサーバ５（一次ストレージ）の対象ファイルをスタブファイルに書き換える（ステップ３）。

【０１６４】

その後、停止していたアクセスを再開する（ステップ４）。アクセスを停止する期間は中間装置３がスタブファイルに書き換える時間となるが、これはスタブファイル７が小さいサイズなのでほとんどの場合は１回のＷＲＩＴＥと属性ＳＥＴで終了するため、非常に短い時間で済む。

【０１６５】

次に、スタブファイルから通常ファイルへ戻す時の動作について説明する。図１４は、ファイルサーバ６（二次ストレージ）からファイルサーバ５（一次ストレージ）へ実ファイルを戻し、スタブファイルをなくすフローを説明するための図である。

【０１６６】

ファイルサーバ５（一次ストレージ）のスタブファイルにファイルサーバ６（二次ストレージ）から実ファイルを上書きコピーしてしまうと、スタブファイルでなくなってしまい、その間にクライアント１からアクセスがあった場合にスタブファイルと判定できない。そのため、コピー中には中途半端なファイルをクライアント１に見せてしまうことになるという問題がある。

【０１６７】

この問題を避けるため、クライアント１の対象ファイルへのアクセスをコピー中は止めておく方式がある。ただし、この方式では、ファイルサイズが大きい場合などでは、クライアント１のアクセスを停止する期間が非常に長くなり、クライアント１に影響を与えてしまう。

【０１６８】

この問題を回避してクライアントに影響を与えない書き戻しを実現するため、ファイルサーバ６（二次ストレージ）からファイルサーバ５（一次ストレージ）のテンポラリ領域に実ファイルのコピーを行う（ステップ１）。

【０１６９】

この間にクライアントから対象ファイルにアクセスがあったら、スタブファイル７がそのまま残っているので、今まで説明した方式で中間装置３が転送を行う。

【０１７０】

テンポラリ領域へのコピーが終了したら、中間装置３で対象ファイルへのアクセスを停止する（ステップ２）。

【０１７１】

その後、中間装置３でテンポラリ領域の実ファイルをＲＥＮＡＭＥコマンドでスタブファイル７と置き換える（ステップ３）。

【０１７２】

その後、中間装置３でクライアント１の対象ファイルへのアクセスを再開する（ステップ４）。

【０１７３】

この方式では、クライアント１のアクセスを停止する期間はＲＥＮＡＭＥコマンドの時間であり、同一ファイルシステム内のＲＥＮＡＭＥコマンドは通常はｉノードテーブルの付け替えだけであり非常に短い時間で済む。そのためクライアント１にほとんど影響を与えることなく書き戻しをすることができる。なお、上記した中間装置３の処理は、中間装置３を構成するコンピュータで実行されるプログラムによって実現してもよい。

【０１７４】

次に、ＮＦＳプロトコルの場合の実施形態について説明する。ＣＩＦＳプロトコルと違い、ＮＦＳプロトコルはネットワーク上にＯＰＥＮやＣＬＯＳＥが出ないプロトコルである。このようなプロトコルの場合、前述のＯＰＥＮをきっかけにしてテーブルを作成するフローは使えない。ＣＩＦＳの場合と同様に、スタブファイルを用いて実現する。スタブファイルの中身も同じである。

【０１７５】
ＣＩＦＳでは、ファイルＩＤはクライアント１のＯＰＥＮからＣＬＯＳＥまでの間だけ用いられるテンポラリのＩＤであるが、ＮＦＳでは、全てのファイルを区別できるファイルハンドルというＩＤが用いられる。
【０１７６】
クライアント１は、ＬＯＯＫＵＰコマンドに、親ディレクトリのファイルハンドルと探したいファイル名を入れてサーバに送り、応答として、ファイルのファイルハンドルを獲得する。
【０１７７】
以後、そのファイルハンドルを用いてＲＥＡＤやＷＲＩＴＥを行う。明示的にＣＬＯＳＥをすることはない。
【０１７８】
そのため、中間装置３では、ファイルＩＤの代わりに、ファイルハンドルをテーブルで持ち、クライアントのＲＥＡＤ・ＷＲＩＴＥコマンドに含まれるファイルハンドルがスタブファイルかを判別する。
【０１７９】
全てのスタブファイルのファイルハンドルをテーブルで持つと、テーブルが大きくなりすぎるので、ＬＯＯＫＵＰをきっかけにテーブル作成し、ある程度の時間アクセスがなければ、テーブルから消す処理を行う。
【０１８０】
図１５は、ＮＦＳの場合の中間装置３でもつテーブルの一例を示す図である。スタブ管理テーブル１２は、ＣＩＦＳの場合と同じであるが、スタブファイルＩＤテーブル１１の代わりに、スタブファイルハンドルテーブル１３を持つ。内容は、一次ストレージファイルＩＤ１１Ａと二次ストレージファイルＩＤ１１Ｂの代わりに、一次ストレージファイルハンドル１３Ａと、二次ストレージファイルハンドル１３Ｂが入る。
【０１８１】
図１６を用いて、本実施例の動作を説明する。クライアント１から中間装置３にＬＯＯＫＵＰリクエスト（ＬＯＯＫＵＰＲｅｑ(ファイル名)）が来ると、中間装置３はファイルサーバ５（一次ストレージ）へ転送し（１８１）、応答に含まれるファイル属性を監視し、ファイルがスタブファイル７かを判断する。
【０１８２】
スタブファイル属性（変更時刻、ファイルサイズで判断する）であれば、実際にファイル先頭をＲＥＡＤしマジックナンバーを確認する。スタブファイルであれば、中間装置内にスタブファイルハンドルテーブル１３とスタブ管理テーブル１２を作成する（ステップ１８２）。
【０１８３】
クライアント１へは、ファイルサーバ５（１次ストレージ）からの応答の属性を実ファイル属性１２Ｃに置き換えてクライアントに転送する。スタブファイルハンドルテーブル１３はファイルハンドルをハッシュして検索できるようにしておくと検索が高速にできる。
【０１８４】
以後、クライアントからＲＥＡＤやＷＲＩＴＥが中間装置に来たら（ステップ１８３）、含まれるファイルハンドルがスタブファイルハンドルテーブル１３にあるかを検索する（ステップ１８４）。
【０１８５】
スタブファイルであった場合、ファイルサーバ６（二次ストレージ）に転送する（ステップ１８５）。なお、上記した中間装置３の処理は、該中間装置３を構成するコンピュータで実行されるプログラムによって実現してもよい。
【０１８６】
最初のＲＥＡＤやＷＲＩＴＥが来たときにスタブファイル７の二次ストレージ実ファイ

ルパス名７Ｃに基づき、ＬＯＯＫＵＰを行って得たファイルハンドルをスタブファイルハンドルテーブル１３の二次ストレージファイルハンドル１３Ｂに入れておく。

【０１８７】

以後のＲＥＡＤやＷＲＩＴＥではスタブファイルハンドルテーブル１３内の二次ストレージファイルハンドル１３Ｂに付け替えてファイルサーバ６（二次ストレージ）に転送する。

【０１８８】

ＧＥＴＡＴＴＲのような、実データは触らずに属性のみを獲得するようなコマンドの場合はファイルサーバ６（二次ストレージ）にアクセスせず、ファイルサーバ５（一次ストレージ）に転送し、応答の属性をスタブ管理テーブル１２に基づいて変更して転送する。

【０１８９】

ファイルＩＤ、もしくはファイルハンドルに、スタブファイルであることを示す識別子を入れる構成としてもよい。

【０１９０】

ＣＩＦＳプロトコルの場合、ＯＰＥＮ時にスタブファイルであることを判別し、スタブファイルであれば、クライアントに返すファイルＩＤの中にスタブファイル識別子を入れる。こうすることで、後続のＲＥＡＤやＷＲＩＴＥ時にスタブ管理テーブルを検索するのはスタブファイル識別子が入ったファイルＩＤのみとなり、全てをスタブ管理テーブルと比較する必要がなくなる。

【０１９１】

ＮＦＳプロトコルの場合、ＬＯＯＫＵＰ応答のファイルハンドルにスタブ識別子を入れておく。こうすることで、後続のＲＥＡＤやＷＲＩＴＥ時に、スタブ識別子が入ったリクエストのみスタブ管理テーブルを検索し、他のリクエストではスタブ管理テーブルを検索する必要がなくなる。これにより、中間装置の処理が軽くなり、高速転送が可能になる。この場合、スタブファイル作成やスタブファイル削除時に、すでにクライアントにスタブ識別子が入っていないファイルＩＤやファイルハンドルを渡している場合に問題となる。これに対処するため、スタブファイル作成や削除の前後では全てのリクエストに対してスタブ管理テーブルの検索が必要になる。処理を簡単にするために、クライアントを識別して、すでにスタブ識別子の入っていないファイルＩＤやファイルハンドルを渡しているクライアントが識別できる場合は、これ以外のクライアントからのリクエストではスタブ管理テーブルを検索しないという処理も可能である。

【０１９２】

スタブファイルに二次ストレージのパス名だけでなく、以下のような情報をいれておく形態がある。

【０１９３】

・複数の二次ストレージパス名を用いた世代管理スタブファイルの内部に複数の二次ストレージパス名を入れておき、ファイルのバージョン管理をすることができる。ある時点で二次ストレージの実ファイルをコピーし、コピー先のパス名をスタブファイルに入れておく。クライアントからアクセスが来たら、スタブファイル内の最新の実ファイルをアクセスする。こうすることで、スナップショット機能などの特殊な機能のないファイルサーバでも世代管理を容易に行うことができる。単にファイルをコピーして保存するのとは異なり、クライアントからは世代管理を隠蔽でき、前世代のファイルを不当に更新できないシステムであり、改ざんが行われていないことが保障できるというメリットがある。

【０１９４】

・中間装置３で付加的に行う処理を記述し、中間装置３でファイルごとに違う処理をすることができる。すなわち、スタブファイルに中間装置３で行う処理を記述しておき、クライアントがアクセスしてきたときに、スタブファイル内の記述に従って中間装置３が処理を行う。従来の構成では、ファイルごとに異なる処理をするためには、ファイルと処理の関係をテーブルにして、他に記憶保持しておく必要があった。これに対して、本発明によれば、スタブファイルを用いることで、ファイルシステム内部に埋め込むことができ、

別のテーブルを持たなくてよい。そのため、アクセスごとにテーブルを検索する必要もなく、処理の高速化、簡素化ができる。

【０１９５】

図１７は、前記した中間装置３の内部構成の一例を示した図である。図１７に示すように、中間装置３は、パケット転送部１４と、スタブファイル判定部１５と、サーバアクセス部１６と、属性変更部１７と、図４を参照して説明したスタブファイルＩＤテーブル１１とスタブ管理テーブル１２を有するテーブル管理部１８を備えている。クライアント１、ファイルサーバ５、６との通信は、パケット転送部１４を経由して行われる。パケット転送部１４は、クライアント１とファイルサーバ５、６間の通信パケットの状態を管理し、必要に応じてスタブファイル判定部１５、サーバアクセス部１６、属性変更部１７との間で処理の受け渡しを行う。スタブファイル判定部１５は、パケット転送部１４からパケットを受け、その処理がスタブファイルに対するものかを判定する。スタブファイル判定部１５は、テーブル管理部１８に問い合わせたり、サーバアクセス部１６に依頼してファイルサーバ５にアクセスする制御を行う。サーバアクセス部１６は、スタブファイル判定のためにファイルサーバ５にアクセスするほか、スタブファイル化や通常ファイル化のためのアクセスなどを制御する。スタブファイル判定部１５によるスタブファイルであるか否かの判定は、前記実施例で説明した手法（属性情報、マジックナンバー、その他）に従う。属性変更部１７は、パケット転送部１４からパケットを受け、テーブル管理部１８に問い合わせてパケット内のファイル属性の変更を行う。

【０１９６】

上記した実施例においては、スタブファイルに対して１つの実ファイルが存在する階層ストレージシステムにおいて、スタブファイルに実ファイルの情報（パス名等）を格納しておく例を説明したが、本発明は、１つのスタブファイルと１つの実ファイルとの対応（１：１対応）に制限されるものでないことは勿論である。以下では、１つのスタブファイルに複数の実ファイルの情報等を格納する構成としたシステムの実施例について説明する。１つのスタブファイルに複数の実ファイルの情報を格納することで、階層ストレージシステムとは別の機能、あらたな付加価値を実現している。

【０１９７】

さらに、上記した実施例では、中間装置は、配下の複数のファイルサーバ（ファイルシステム層、記憶装置）をあたかも１つのファイルシステムであるかのようなファイルサービスをクライアントに提供していたが、以下では、複数の拠点にファイルサーバを配し、１つの拠点のクライアントをして複数拠点のファイルサーバに中間装置を介してアクセス可能としたシステムの実施例についても説明する。複数の拠点に分散してファイルサーバを配し、各拠点に対応して中間装置を配し、異なる拠点の中間装置同士は、例えば広域ネットワークを介して相互に通信し、１つの拠点のクライアントをして１つの拠点のファイルサーバのみならず、他の拠点の中間装置を介して他の拠点のファイルサーバにもアクセス可能としたものであり、広域分散した複数のファイルシステムを仮想化しクライアントに対してファイルの存在する拠点（すなわちファイルの在り処）を意識させないですむファイルサービスを可能としている。

【０１９８】

まず、本発明の別の実施例として、スタブファイルを用いた最大ファイルサイズの拡大について説明する。ＮＡＳ等では、実装されているファイルシステム層によって、最大ファイルシステムサイズ、最大ファイルサイズの制限がある。HPC（High Performance Computing）等では、上記制限以上のファイルの作成の要求がある。そこで、本発明は、スタブファイルを用いてファイルの最大サイズを拡大可能としている。

【０１９９】

図１８は、本発明の別の実施例を説明するための図である。図１８を参照すると、１つのファイルは、複数の分割ファイル（図１８では、２つの分割ファイル＃１、＃２）からなる。２つの分割ファイル＃１、＃２はオーバーラップ領域を有している。図１８に示す例では、分割ファイルはＮＡＳの最大ファイルサイズである２TB（Teraバイト）とされ、

２つの分割ファイル＃１、＃２からなるファイルのファイルサイズはほぼ４TB程度となる。ここで、リード／ライトアクセスの最大サイズを16MB(Megaバイト)とすると、分割ファイル＃１の２TBのアドレス空間の後から先頭側に向けた16MB、分割ファイル＃２の先頭から始まる16MBが互いに重なっており、オーバーラップ領域とされる。この場合、リードライトアクセスアドレスが２TB-16MBより小の場合、分割ファイル＃１へのアクセスとなり、リードライトアクセスアドレスが２TB-16MBより大の場合、分割ファイル＃２へのアクセスとなる。

【０２００】

オーバーラップ領域を設けない場合、例えばリードアクセスアドレスが２TB-16MBより大であり、分割ファイル＃１をアクセスした場合、１回のリードアクセスが16MBであることから、該リードアクセスの実行時、次の分割ファイル＃２の先頭の領域のデータを読み出すことになる。一方、本実施例によれば、例えばリードアクセスアドレス（先頭アドレス）が２TB-16MBより大であれば、分割ファイル＃２の該当アドレスから16MB分読み出すことになる。このように、相隣る分割ファイル間に互いに重なるオーバーラップ領域を設けることで、１回のリード／ライトアクセスの実行時、複数の分割ファイルをまたぐことがないようにしている。

【０２０１】

分割ファイル＃１と＃２のオーバーラップ領域へのデータの書き込みは、Ｄｕａｌ　Ｃａｌｌ（書き込みの二回呼出）か、分割ファイル＃１、＃２の一方のオーバーラップ領域へのデータを書き込んだ後、コピーをして、同期させる。これにより、分割ファイル＃１、＃２のオーバーラップ領域のデータの整合が行われる。同期の分、たしかにオーバーヘッドはあるものの、最大ファイルサイズ２TBと比べて、最大リード/ライトサイズ16MBは極めて小であるため、オーバーヘッドの影響は少ない。

【０２０２】

次に、本発明の一実施例により、スタブファイルを用いて最大ファイルサイズを拡大する手法について説明する。図１９は、本発明の一実施例を説明するための図である。前述したスタブファイルに格納される情報として、実ファイルのパス名に、複数の分割ファイル＃１～＃３のパス名を格納しておく。なお、図１９に示す例においても、相隣る分割ファイル＃１、＃２にはオーバーラップ領域が設けられており、相隣る分割ファイル＃２、＃３にも、オーバーラップ領域が設けられている。

【０２０３】

図２０は、本発明の一実施例を説明するための図である。元ファイル（分割ファイルではない）のファイルサイズが予め定められた閾値を超えた場合、スタブ化して分割ファイル＃１をつくる。本実施例では、分割ファイルの作成は、好ましくは”RENAME”（名前の変更）で行う。分割ファイル＃１は、スタブファイルと同一ファイルシステム内に設けられる。スタブファイル化する時点で、ファイルサイズはTB（テラバイト）を超えており、別ファイルシステムにコピーする場合、膨大な時間を要する。ファイルの”RENAME”でスタブ化する場合、ファイルデータのコピーは不要とされる。分割ファイル＃２以降は、別ファイルシステム内に設けるようにしてもよい。

【０２０４】

前述した実施例のシステムでは、スタブファイルは、１つの拠点、１つの中間装置内で用いられていた。以下では、複数拠点、複数の中間装置間でスタブファイルを相互に参照するシステム構成（「相互参照モデル」という）について説明する。

【０２０５】

図２１は、本発明の別の実施例の構成を説明するための図であり、相互参照モデルを説明するための図である。クライアント１とＮＡＳスイッチ（中間装置）３とＮＡＳ（ファイルサーバ）５とからなる拠点が、広域ネットワーク２を介して接続されている。拠点ＡのＮＡＳのファイルシステムのスタブファイルＢが拠点ＢのＮＡＳのファイルシステムの実ファイルＢを参照し、拠点ＢのＮＡＳのファイルシステムのスタブファイルＡが拠点ＡのＮＡＳのファイルシステムの実ファイルＡを参照している。

【０２０６】
スタブファイルと実ファイルの配置の仕方として、アクセス頻度の履歴情報等、アクセスコンテキストに基づき、比較的多くアクセスされるデータを、自分のファイルシステムに配置（ローカライズ）するようにしている。
【０２０７】
拠点Ａ、Ｂのどちらからもアクセスされるファイルの場合、実ファイルが配置されるのは、アクセス頻度の高い拠点とされる。図２１に示す例では、実ファイルＡは、拠点Ａからアクセスされる頻度が、拠点Ｂからアクセスされる頻度よりも高いため、拠点Ａに配置されている。実ファイルＢは、拠点Ｂからアクセスされる頻度が、拠点Ａからアクセスされる頻度よりも高いため、拠点Ｂに配置されている。
【０２０８】
スタブファイルと実ファイルについて、実データは常に最新の状態に保つ。なお、スタブファイルの情報は、非同期に更新してもよいし、常に最新の状態に保つようにしてもよい。
【０２０９】
次に、複数ノードで実行されることによる、整合性の管理について説明する。ファイルデータの整合性として、標準プロトコルのロック機構を用いて整合性を保つ。スタブファイル側からアクセスが行われた場合、スタブファイルと実ファイルをロックする。一方、実ファイル側からアクセスが行われた場合、実ファイルのみをロックする。
【０２１０】
本実施例において、スタブファイル側から、更新が発生した場合、実データとの整合性を保つ。実データ側から更新が発生した場合には、スタブファイルを、非同期で更新する。
【０２１１】
図２２は、図２１に示した複数拠点の分散システムにおける、リード、ライト系の処理フローを示す図であり、スタブ側からの更新が発生した場合の手順を示すシーケンスダイアグラムである。以下では、特に制限されないが、中間装置として、ＮＡＳ（Ｎｅｔｗｏｒｋ　Ａｔｔａｃｈｅｄ　Ｓｔｏｒａｇｅ）とクライアント（Ｃｌｉｅｎｔ）間に配されるＮＡＳを例に説明する。ここでは、スタブが自拠点（例えば図２１の拠点Ａ）のＮＡＳ、実データが他拠点（例えば図２１の拠点Ｂ）のＮＡＳにあるものとする。
【０２１２】
図２２を参照すると、クライアント（Ｃｌｉｅｎｔ）が、ＯＰＥＮリクエストを自拠点のＮＡＳスイッチに発行し、自拠点のＮＡＳスイッチは、配下のＮＡＳの該当ファイルがスタブファイルであるか否かを、前記実施例で説明チェック方法（属性、固定長のファイルサイズ）にしたがってチェックする（スタブ？）。スタブファイルの場合、自拠点のＮＡＳスイッチは、当該ファイルを配下のＮＡＳから読み出して、前述したように、例えばファイル先頭のマジックナンバー等に基づき、スタブファイルであることを確かめる。スタブファイルであることが確定すると（スタブ確定）、他拠点のＮＡＳスイッチにＯＰＥＮ要求を送信する。他拠点のＮＡＳの実ファイルのＯＰＥＮが成功すると、他拠点のＮＡＳスイッチは、自拠点のＮＡＳスイッチに応答（Ｒｅｓｐ）を返し、また、他拠点のＮＡＳの実ファイルをロック（ＬＯＣＫ）する。
【０２１３】
自拠点のＮＡＳスイッチからＯＰＥＮ要求の成功応答（Ｒｅｓｐ）を受信したクライアントは、ＲＥＡＤ要求を自拠点のＮＡＳスイッチに送信し、自拠点のＮＡＳスイッチは広帯域ネットワークを介して、他拠点のＮＡＳスイッチにＲＥＡＤ要求を送信し、他拠点のＮＡＳスイッチは配下のＮＡＳから読み出したデータ（ＤＡＴＡ）を自拠点のＮＡＳスイッチに送信する。自拠点のＮＡＳスイッチは、読出しデータをクライアントに送信する。
【０２１４】
読出しデータを受け取ったクライアントは、ＣＬＯＳＥ要求を自拠点のＮＡＳスイッチに送信し、自拠点のＮＡＳスイッチは、ＣＬＯＳＥ要求を広帯域ネットワークを介して他

拠点のＮＡＳスイッチに送信し、他拠点のＮＡＳスイッチは、配下のＮＡＳの実ファイルのＵＮＬＯＣＫ処理を行う。そして、他拠点のＮＡＳスイッチは、ＣＬＯＳＥ要求の応答（Ｒｅｓｐ）を自拠点のＮＡＳスイッチに返却し、自拠点のＮＡＳスイッチは、自拠点のＮＡＳのスタブをＣＬＯＳＥ処理し、その応答（ＣＬＯＳＥ　Ｒｅｓｐ）をクライアントに返す。他拠点ＮＡＳスイッチから自拠点のＮＡＳスイッチへの各応答の転送は、拠点間の同期転送とされる。

【０２１５】

図２３は、リード、ライト系の処理フローを示す図であり、実データ側からの更新が発生した場合の手順を示すシーケンスダイアグラムである。実データが自拠点（図２１の拠点Ａ）のＮＡＳ、スタブが他拠点（図２１の拠点Ｂ）のＮＡＳにあるものとする。

【０２１６】

クライアントがＯＰＥＮリクエストを自拠点のＮＡＳスイッチに発行し、自拠点のＮＡＳスイッチは、自拠点のＮＡＳのファイル（実ファイル）をＯＰＥＮし（ＬＯＣＫ）、その応答（ＯＰＥＮ　Ｒｅｓｐ）をクライアントに返す。

【０２１７】

クライアントが、実データへのデータの書き込みを行う。すなわちＷＲＩＴＥ要求をＮＡＳスイッチを介して拠点内のＮＡＳに送信する。書き込みが終了すると、クライアントはＣＬＯＳＥ要求を発行し、ＮＡＳスイッチを介して、配下のＮＡＳに送信され、実ファイルのＣＬＯＳＥ処理が行われる。そして、その応答（ＣＬＯＳＥ　Ｒｅｓｐ）がクライアントに返送される。自拠点のＮＡＳスイッチは、上記ＷＲＩＴＥ処理による実ファイルの属性変更（ファイルサイズの変更）に対して、スタブファイルを格納した他拠点のＮＡＳに対応するＮＡＳスイッチに対して、サイズ更新情報を送信する。自拠点のＮＡＳスイッチにおいて、他拠点のＮＡＳスイッチへのサイズ更新情報の送信は、ＣＬＯＳＥ処理のあとに行われ、拠点間の非同期転送により行われる。

【０２１８】

他拠点のＮＡＳスイッチは、配下のＮＡＳのファイルシステム内のスタブファイルに格納される実ファイルのファイルサイズ情報を更新し、その応答を、自拠点ＮＡＳスイッチに返す。なお、リードアクセスの場合、実ファイルのサイズ情報の変更等はないため、サイズ情報更新要求の送信は行われない。

【０２１９】

図２４は、本実施例におけるファイルシステム更新（ファイルの削除、作成）処理の例を示す図であり、スタブ側からの更新が発生した場合の手順を示すシーケンスダイアグラムである。この例では、スタブは自拠点のＮＡＳ、実ファイルは他拠点のＮＡＳにあるものとする。

【０２２０】

クライアントが削除（ＤＥＬＥＴＥ）要求を自拠点のＮＡＳスイッチに発行し、自拠点のＮＡＳスイッチは、スタブであるかチェックし、スタブである場合、他拠点ＮＡＳスイッチにＤＥＬＥＴＥ要求を送信する。他拠点のＮＡＳスイッチは配下のＮＡＳのファイルシステムに当該実ファイル（スタブファイルでパス名が指定される）を削除するように、ＤＥＬＥＴＥ要求を送信する。他拠点のＮＡＳスイッチは、実ファイルの削除の応答を自拠点のＮＡＳスイッチに返送し、自拠点のＮＡＳスイッチは、該削除（ＤＥＬＥＴＥ）処理が成功した場合に、自拠点のＮＡＳのファイルシステム内のスタブを削除するように指示する。そして、ＤＥＬＥＴＥ要求の応答（ＤＥＬＥＴＥ　Ｒｅｓｐ）をクライアントに返す。

【０２２１】

自拠点のＮＡＳスイッチから他拠点ＮＡＳスイッチへの実ファイルのＤＥＬＥＴＥ要求の送信と、他拠点ＮＡＳスイッチから自拠点のＮＡＳスイッチへの応答の送信、及び、自拠点のＮＡＳへのＤＥＬＥＴＥ要求と、その応答の返送は、削除の応答をクライアントに返す前に行われる（拠点間同期転送）。また、ディレクトリの下にファイルを作成する、あるいは、ディレクトリを作成する等の操作については、あらかじめスタブ側か、実デー

タ側かを決めておく必要がある。

【０２２２】

図２５は、本実施例において、ファイルシステム更新（ファイルの削除、作成）処理の別の例を示す図であり、実データ側からの更新が発生した場合の手順を示すシーケンスダイアグラムである。スタブが他拠点のＮＡＳ、実データが自拠点のＮＡＳにあるものとする。

【０２２３】

クライアントがＤＥＬＥＴＥ要求を自拠点のＮＡＳスイッチに発行し、自拠点のＮＡＳスイッチは、配下のＮＡＳに実データの削除を行うように指示し、削除が成功すると、クライアントにＤＥＬＥＴＥ要求の応答（ＤＥＬＥＴＥ　Ｒｅｓｐ）を返す。つづいて、他拠点のＮＡＳスイッチにスタブの削除要求を非同期で転送する。スタブの削除の応答は他拠点ＮＡＳスイッチから、自拠点ＮＡＳスイッチに送信される。ディレクトリの下にファイルを作成する、あるいは、ディレクトリを作成する等の操作については、あらかじめスタブ側か、実データ側かを決めておく必要がある。

【０２２４】

次に、拠点がＮ個（Ｎ≧２）のシステム構成について説明する。整合性管理のため、全ての実データの格納された拠点からの確認を必要とする。リード／ライト系処理の場合、実データの拠点からの許可（ＬＯＣＫ処理成功）、スタブの拠点からのスタブファイルの許可がそろって初めて、リード／ライトアクセスの実行が行われる。

【０２２５】

ファイルシステム更新系処理の場合、常に、実データの拠点で処理が成功したことを確認して処理結果を、他拠点に非同期で伝播させる。

【０２２６】

本発明さらに別の実施例について説明する。図２６は、スタブファイルをキャッシュするシステムの構成を模式的に示す図である。図２６に示した構成は、図２１に示した構成において、少なくとも１つの拠点に実ファイルをキャッシュする構成としたものである。各拠点が、クライアント１、ＮＡＳスイッチ３、ＮＡＳ５を備え、異なる拠点のＮＡＳスイッチ３は広域ネットワーク２で通信接続し、クライアント１には、該クライアントが属する拠点であるか他の拠点へのファイルアクセスであるかを意識させないファイルアクセスサービスが提供される。

【０２２７】

本実施例において、スタブ側からの読み出し、更新要求があった場合、スタブ側の拠点にデータをキャッシュすることにより、スタブ側のアクセス性能を向上させる。本実施例において、ＲＥＡＤ、ＷＲＩＴＥキャッシュをそれぞれ備えてもよい。

【０２２８】

本実施例では、スタブファイルに、キャッシュのポインタとなる識別子を含む。例えば図３に示したスタブファイルのフォーマットにおいて、二次ストレージ実ファイルパス名７Ｃと、リザーブ７Ｄの間に、キャッシュのポインタとなる識別子（不図示）が追加される。スタブ化されたデータへのアクセス時、読み出されたデータを、自拠点のＮＡＳのキャッシュ領域５１に格納する。キャッシュのポインタ３１となる識別子を、ＮＡＳスイッチ（中間装置）の記憶部へテーブル化して格納する。例えば図４のスタブファイルIDテーブルのスタブ管理テーブルには、キャッシュ領域５１にキャッシュされている実ファイルの位置情報を格納するようにしてもよい。なお、ＮＡＳ５のサービス領域５２は、ファイルデータを格納する記憶領域である。

【０２２９】

次回、キャッシュされたスタブファイルにアクセスする時、スタブファイルに含まれるキャッシュポインタの識別子が、ＮＡＳスイッチ３にテーブルとして格納されていた場合、キャッシュヒットであり、ＮＡＳスイッチ（中間装置）配下のローカルな（自拠点の）ＮＡＳへのデータアクセスが行われる。なお、オープンとロック（ＬＯＣＫ）はスタブファイル側と実ファイル側ともに実行する。

【０２３０】
図２７は、本実施例における、ＷＲＩＴＥキャッシュの動作を説明するシーケンス図である。図２７を参照して、本実施例のＷＲＩＴＥキャッシュ動作を説明する。クライアントがＯＰＥＮ要求を自拠点のＮＡＳスイッチに送信し、自拠点のＮＡＳスイッチは、配下（自拠点）のＮＡＳの当該ファイルがスタブファイルであるかチェックし、スタブファイルの場合、当該ファイルを読み出し、ファイル先頭のマジックナンバー等をチェックすることで、スタブファイルであるか確認する。
【０２３１】
スタブファイルであることが確認された場合、当該スタブファイルに含まれるキャッシュポインタの識別子が、ＮＡＳスイッチの記憶部（図２６の３１）に格納されている場合（キャッシュヒット時）、キャッシュされているスタブファイルの内容に基づき、他拠点の実ファイルをＯＰＥＮ処理する。他拠点のＮＡＳスイッチは、当該実ファイルをロック（ＬＯＣＫ）し、ＯＰＥＮ要求の応答を自拠点のＮＡＳスイッチに送信する。ＯＰＥＮ処理が成功し、属性変更がないため、自拠点のＮＡＳスイッチは、ＯＰＥＮ要求の応答をクライアントに返す。なお、属性に差異がある場合、スタブファイルをキャッシュから追い出す（キャッシュアウト）。
【０２３２】
クライアントは、自拠点のＮＡＳスイッチからのＯＰＥＮ要求の応答（ＯＰＥＮ　Ｒｅｓｐ）を受け、ＷＲＩＴＥ要求を自拠点のＮＡＳスイッチに送信する。自拠点のＮＡＳスイッチは、キャッシュ領域５１にキャッシュされている実ファイルへの書き込みを行い、また、ＮＡＳスイッチにキャッシュされているスタブファイルの変更（ファイルサイズ等の属性変更）を行う。そして、ＷＲＩＴＥ要求の応答（ＷＲＩＴＥ　Ｒｅｓｐ）をクライアントに返す。
【０２３３】
クライアントがＣＬＯＳＥ要求を自拠点のＮＡＳスイッチに送信すると、自拠点のＮＡＳスイッチは、自拠点のＮＡＳのキャッシュ領域５１にキャッシュされている実ファイルのＣＬＯＳＥ処理を行い、その応答（ＣＬＯＳＥ　Ｒｅｓｐ）をクライアントに返す。そして、自拠点のＮＡＳスイッチは、他拠点のＮＡＳスイッチに対して、ＷＲＩＴＥ要求を行い、他拠点のＮＡＳスイッチは、該ＷＲＩＴＥ要求を受け配下のＮＡＳの実ファイルへの書き込みを行うことで、データの整合性（ある拠点にキャッシュされているデータと他の拠点の実ファイルのデータ）を保つ。他拠点のＮＡＳスイッチは、ＷＲＩＴＥ処理の応答を、自拠点のＮＡＳスイッチに送信する。自拠点のＮＡＳスイッチは、他拠点ＮＡＳスイッチにＣＬＯＳＥ要求を送り、他拠点ＮＡＳスイッチは、実ファイルをＵＮＬＯＣＫする。
【０２３４】
図２８は、ＲＥＡＤキャッシュの動作を説明するシーケンス図である。図２８を参照して、本実施例のＲＥＡＤキャッシュ動作を説明する。クライアントがＯＰＥＮ要求を自拠点のＮＡＳスイッチに送信し、ＮＡＳスイッチは配下（自拠点）のＮＡＳの当該ファイルがスタブファイルであるかチェックし、スタブファイルの場合、当該ファイルを読み出し、先頭のマジックナンバー等をチェックすることで、スタブファイルであるか確認する。スタブファイルと確認され、スタブファイルに含まれるキャッシュポインタの識別子が、ＮＡＳスイッチのテーブルに格納されていた場合（キャッシュヒット時）、キャッシュされているスタブファイルの内容に基づき、他拠点の実ファイルをＯＰＥＮ処理する。
【０２３５】
他拠点のＮＡＳスイッチは、当該実ファイルをロック（ＬＯＣＫ）し、ＯＰＥＮ要求の応答を自拠点ＮＡＳスイッチに送信する。ＯＰＥＮ処理が成功し、属性変更がないため、自拠点ＮＡＳスイッチは、ＯＰＥＮ要求の応答（ＯＰＥＮ　Ｒｅｓｐ）をクライアントに返す。なお、属性に差がある場合、スタブファイルをキャッシュから追い出す。
【０２３６】
クライアントは、ＯＰＥＮ要求（ＯＰＥＮ　Ｒｅｓｐ）の応答を受け、ＲＥＡＤ要求を

、自拠点のＮＡＳスイッチに送信する。自拠点のＮＡＳスイッチは、キャッシュされているスタブファイルの内容に基づき、自拠点のＮＡＳのキャッシュ領域５１（実ファイルデータがキャッシュされている）からの読み出しを行い、その応答（読み出しデータ）をクライアントに返す。

【０２３７】

クライアントがＣＬＯＳＥ要求を自拠点のＮＡＳスイッチに送信し、自拠点のＮＡＳスイッチは、自拠点のＮＡＳの実ファイルのＣＬＯＳＥ処理を行い、その応答（ＣＬＯＳＥ Ｒｅｓｐ）をクライアントに返す。

【０２３８】

その後、自拠点のＮＡＳスイッチは、他拠点のＮＡＳスイッチに対してＣＬＯＳＥ要求を送信し、他拠点のＮＡＳスイッチは、配下のＮＡＳの実ファイルをＵＮＬＯＣＫする。このように、ＯＰＥＮ時に、スタブファイルで参照される他の拠点の実ファイルを自拠点にキャッシュすることで、アクセスの高速化を図ることができる。なお、ある拠点のＮＡＳスイッチの記憶部（図２６の３１）に、キャッシュポインタの識別子が格納されていない場合（ミスヒット時）、他の拠点の実ファイルを、ある拠点のキャッシュ領域５１に転送する処理が行われ、キャッシュポインタの識別子がＮＡＳスイッチの記憶部（テーブル）に配置され、上記したキャッシュへのアクセス処理が行われる。

【０２３９】

なお、上記では、他拠点の実ファイルを、スタブファイルが置かれる自拠点のキャッシュ領域にキャッシュする例に即して説明したが、本発明はかかる構成に制限されるものでないことは勿論である。例えば他拠点の実ファイルを、スタブファイルのある自拠点に近接する拠点にキャッシュしてもよいし、あるいは、該他拠点に実ファイルを置いてアクセスする形態よりも、通信コスト、通信状態、ストレージ容量等の条件でより良い拠点があれば、該拠点のキャッシュ領域に実ファイルをキャッシュするようにしてもよい。スタブファイルには、キャッシュされている実ファイル、及び他拠点の実ファイルへのアクセスパスの情報が格納され、スタブファイルからキャッシュ領域の実ファイルへのアクセス、他拠点の実ファイルへのアクセスを行うことができる。また、実ファイルをキャッシュ領域にキャッシュするシステムは、図２６に示した複数拠点よりなるシステムにのみ制限されるものでなく、例えば、クライアント、中間装置、複数のファイルシステムを備えたシステム等に対しても適用できることは勿論である。

【０２４０】

次に、本発明のさらに別の実施例について説明する。ファイルのスタブ化／非スタブ化のトリガーとして、ストレージ容量の閾値、クオータ（Ｑｕｏｔａ）の設定値、アクセス頻度等のアクセスコンテキストの分析の結果に基づき指示等を用いてもよい。

【０２４１】

以下、Ｑｕｏｔａの閾値を用いた例に即して説明する。ファイルサーバには、一般に、ユーザ、グループ、ディレクトリを管理単位として装置に付随するストレージ資源の使用量（使用量の割り当てを「Ｑｕｏｔａ」という）を制限するＱｕｏｔａシステムが実装されており、管理者により、予め設定されたストレージ使用制限量を超えて、データの書き込みが行えないような制御を行うことが可能とされる。ユーザ管理型Ｑｕｏｔａは、例えばユーザ識別子（ＵＮＩＸ（登録商標）のＵＩＤ）を元に管理され、Ｑｕｏｔａは、ユーザが所有するファイルやブロック数に対して設定される。また、グループ管理型Ｑｕｏｔａは、グループ識別子（ＵＮＩＸ（登録商標）のＧＩＤ）を元に管理され、Ｑｕｏｔａは、グループが所有するファイルやブロック数に対して設定される。そして、ディレクトリ管理型Ｑｕｏｔａは、ディレクトリ下のファイルやブロック数に対して設定される。また、よく知られいるように、Ｑｕｏｔａには、ハードＱｕｏｔａ（ハードリミット：リミットを越えて書き込もうとするとエラーとなる）と、ソフトＱｕｏｔａ（ソフトリミット：一定の猶予期間であれば超過して書き込むことができる。このリミットを越えると、管理者又はユーザに警告が通知される）がある。ユーザ管理のＱｕｏｔａ構造体（レコード）は例えばユーザ識別子に対応して設けられ、ブロックのハードとソフト制限、ユーザが現

在割り当てられているブロックとファイル数、ソフト制限がハード制限として取り締まられるまでに残っているユーザへの警告の回数等の情報を含む。Ｑｕｏｔａ管理機能は、ストレージ使用量を制限する対象と、その制限量の設定機能や、設定した対象でのストレージ使用量（設定した制限量を上限とする使用率）に関する情報を取得する機能等が挙げられる。Ｑｕｏｔａの設定管理は、既存のＮＡＳやファイルサーバに備えられた管理インタフェースを用いることができる。

【０２４２】

特に制限されないが、以下の実施例では、各ファイルシステムの最新のストレージ使用量の取得、各ファイルシステム（ファイルサーバ）におけるＱｕｏｔａ設定値の記憶管理、ストレージ使用量とＱｕｏｔａ設定値との関係から、ファイルシステムにおけるスタブ化の決定、スタブ化の実行制御は、ファイルサーバと通信接続するＮＡＳスイッチ（中間装置）にて行う。

【０２４３】

図２９は、本発明の一実施例を説明するための図である。ＮＡＳスイッチのＱｕａｒｔｅｒ設定値が１００ＧＢ、ハードリミットが１００ＧＢ、ソフトリミット１００ＧＢ、ファイルシステムＡがプライマリ、ファイルシステムＢがセカンダリのシステム構成において、各ファイルシステムへのＱｕｏｔａの設定は固定（ハードリミット＝ソフトリミット）としている。特に制限されないが、ＮＡＳスイッチは、定期的に、配下のファイルシステムＡ、ファイルシステムＢのクオータ管理情報を取得し、プライマリファイルシステムＡの使用量が、そのクオータ設定値の７０～９０％以上に達した場合、一定容量を、スタブ化する。

【０２４４】

図２９（Ａ）に示すように、ファイルシステムＡのＱｕｏｔａ設定値は、３０ＧＢ＋α（ただし、αはあそび）に設定され、ファイルシステムＢのＱｕｏｔａ設定値は、７０ＧＢ＋αに設定されている。一回目の調査の結果、ファイルシステムＡの使用量は１５ＧＢであった。なお、ＮＡＳスイッチ等の中間装置（例えば図２の３）において、ファイルサーバにおけるストレージ使用量の監視は、中間装置からファイルサーバに対して定期的に行われるポーリング等で行ってストレージ使用量を取得してもよく、また、ファイルサーバ側から中間装置への割り込み等でストレージ使用量を通知する構成としてもよい。

【０２４５】

図２９（Ｂ）に示すように、２回目のストレージ使用量調査の結果、ファイルシステムＡの使用量（２５ＧＢ）がＱｕｏｔａ設定値（３０ＧＢ＋α；αはあそび）の８０％以上に達した場合、ファイルシステムＡの１０ＧＢをスタブ化する。スタブ化は、例えば図１３等を参照して説明した手順で行われる。

【０２４６】

その結果、図２９（Ｃ）に示すように、１０ＧＢをスタブ化した後のファイルシステムＡの使用量は１５ＧＢとなり、ファイルシステムＢには、実ファイルが格納され、使用量は１０ＧＢとなる。

【０２４７】

なお、本実施例において、ファイルシステムのＱｕｏｔａ設定値とストレージ使用量の監視結果に基づき、ＮＡＳスイッチ（中間装置）側でファイルのスタブ化の決定（スタブ化の候補の選択等）、及び、スタブ化実行（プライマリファイルシステムＡからセカンダリファイルシステムＢへのデータの移行）の制御を自動で行っている。

【０２４８】

すなわち、本実施例において、スタブ化による実ファイルセカンダリファイルシステムへの移動等のデータ移行（ｄａｔａ　ｍｉｇｒａｔｉｏｎ）は、ＮＡＳスイッチ（中間装置）を介して、クライアントには、隠蔽されている。なお、データ移行の隠蔽技術の詳細は、例えば、特許文献１等の記載が参照される。また、本実施例において、好ましくは、ＮＡＳスイッチ（中間装置）は、複数のファイルシステムをクライアントに対して、あたかも１つのファイルシステムであるかのようなファイルサービスを提供している。クライ

アントに複数のファイルシステムを１つの擬似ファイルシステム（ディレクトリーツリーによる管理）としてのファイルアクセスサービスを提供する機能の実現については、例えば特許文献１等の記載が参照される。

【０２４９】

上記したストレージ使用量に基づくスタブ化の制御は、図２１を参照して説明した複数拠点の相互参照モデルに対しても適用できることは勿論である。この場合、スタブ化において、アクセスコンテキスト情報に基づき、アクセスの多い拠点に、実ファイルを置き、他の拠点では、スタブファイルを配置するようにしてもよい。かかる複数拠点の構成においても、クライアントには、スタブ化によるデータ移行は隠蔽されている。

【０２５０】

以上、本発明を上記実施例に即して説明したが、本発明は、上記実施例の構成にのみ限定されるものでなく、本発明の範囲内で当業者であればなし得るであろう各種変形、修正を含むことは勿論である。

【図面の簡単な説明】

【０２５１】

【図１】本発明の一実施例のシステム構成を示す図である。

【図２】本発明の一実施例のスタブファイルを用いた処理概要を示す図である。

【図３】本発明の一実施例のスタブファイルフォーマットを示す図である。

【図４】本発明の一実施例の中間装置でもつテーブルを示す図である。

【図５】本発明の一実施例の動作をフローで示した図である。

【図６】本発明の一実施例の動作をフローで示した図である。

【図７】本発明の一実施例の動作をフローで示した図である。

【図８】本発明の一実施例の動作をフローで示した図である。

【図９】本発明の一実施例の動作をフローで示した図である。

【図１０】本発明の一実施例の動作をフローで示した図である。

【図１１】本発明の一実施例の動作をフローで示した図である。

【図１２】本発明の一実施例の動作をフローで示した図である。

【図１３】本発明の一実施例のスタブファイルを用いた処理概要を示す図である。

【図１４】本発明の一実施例のスタブファイルを用いた処理概要を示す図である。

【図１５】本発明の一実施例の中間装置でもつテーブルを示す図である。

【図１６】本発明の一実施例の動作をフローで示した図である。

【図１７】本発明の一実施例の中間装置の構成の一例を示す図である。

【図１８】本発明の他の実施例における最大ファイルサイズの拡大を説明するための図である。

【図１９】本発明の他の実施例においてスタブファイルを用いた最大ファイルサイズの拡大を説明するための図である。

【図２０】本発明の他の実施例におけるスタブファイルと分割ファイルの対応を説明するための図である。

【図２１】本発明の他の実施例の広域分散環境における相互参照モデルを説明するための図である。

【図２２】広域分散環境におけるREAD/WRITE系処理（スタブ側）を示すシーケンス図である。

【図２３】広域分散環境におけるREAD/WRITE系処理（実データ側）を示すシーケンス図である。

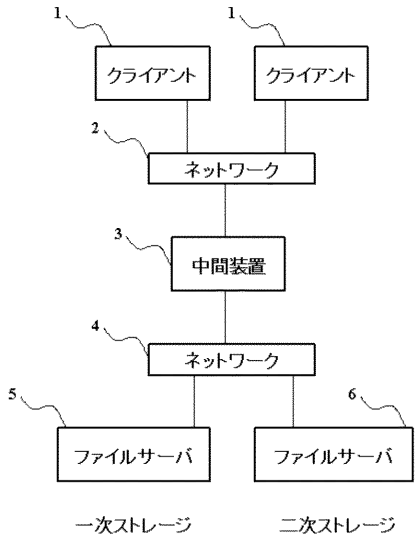
【図２４】広域分散環境におけるファイルシステム更新処理（スタブ側）を示すシーケンス図である。

【図２５】広域分散環境におけるファイルシステム更新処理（実データ側）を示すシーケンス図である。

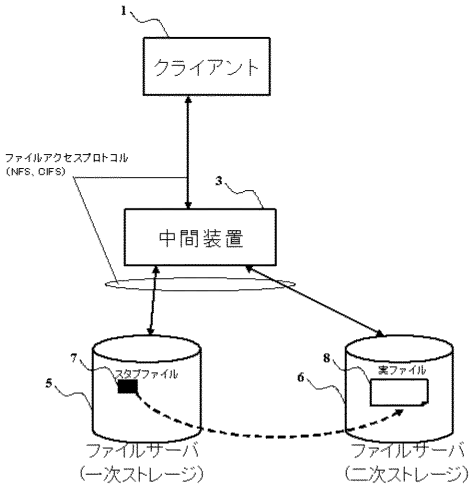
【図２６】本発明の他の実施例におけるスタブファイルのキャッシュを説明するための図である。

【図２７】図２６のシステムにおけるREADキャッシュ（スタブ側）を示すシーケンス図である。
【図２８】図２６のシステムにおけるWRITEキャッシュ（スタブ側）を示すシーケンス図である。
【図２９】本発明の他の実施例のＱｕｏｔａ設定値とストレージ使用量閾値によるスタブ化を説明するための図である。
【符号の説明】
【０２５２】
１　クライアント
２　ネットワーク
３　中間装置（ＮＡＳスイッチ）
４　ネットワーク
５　ファイルサーバ（一次ストレージ）、ＮＡＳ
６　ファイルサーバ（二次ストレージ）
７　スタブファイル
７Ａ　スタブ識別用マジックナンバー
７Ｂ　実ファイル属性
７Ｃ　二次ストレージ実ファイルパス名
７Ｄ　リザーブ
８　実ファイル
１１　スタブファイルＩＤテーブル
１１Ａ　一次ストレージＦＩＤ
１１Ｂ　二次ストレージＦＩＤ
１１Ｃ　スタブ管理テーブルポインタ
１２　スタブ管理テーブル
１２Ａ　一次ストレージファイルパス名
１２Ｂ　二次ストレージファイルパス名
１２Ｃ　実ファイル属性
１３　スタブファイルハンドルテーブル
１３Ａ　一次ストレージファイルハンドル
１３Ｂ　二次ストレージファイルハンドル
１３Ｃ　スタブ管理テーブルポインタ
１４　パケット転送部
１５　スタブファイル判定部
１６　サーバアクセス部
１７　属性変更部
１８　テーブル管理部
３１　ポインタ
５１　キャッシュ領域
５２　サービス領域
１０１～１０６　ＯＰＥＮ時の動作フロー
１１１～１１７　ＲＥＡＤ時の動作フロー
１２１～１２５　属性ＧＥＴ（対象をファイルＩＤで指定）の動作フロー
１３１～１３７　属性ＧＥＴ（対象をＰＡＴＨで指定）の動作フロー
１４１～１４９　属性ＳＥＴ（対象をファイルＩＤで指定）の動作フロー
１５１～１５９　属性ＳＥＴ（対象をＰＡＴＨで指定）の動作フロー
１６１～１６８　ＦＩＮＤ系（対象をＰＡＴＨで指定）の動作フロー
１７１～１７８　ＣＬＯＳＥ時の動作フロー
１８１～１８５　ＮＦＳプロトコルの場合の動作フロー

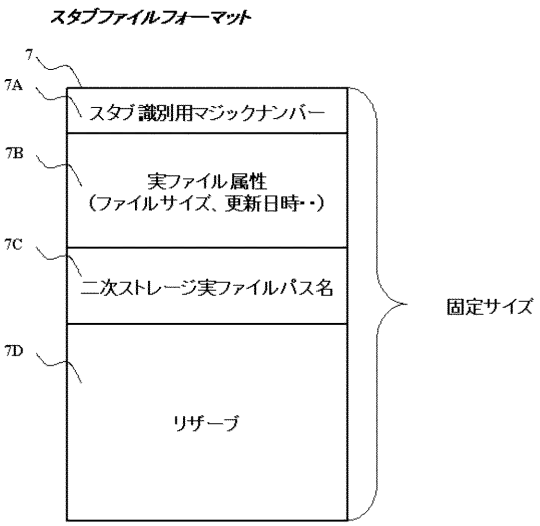
【図１】


1
1
クライアント
クライアント
2
ネットワーク
3
中間装置
4
ネットワーク
5
6
ファイルサーバ
ファイルサーバ
一次ストレージ
二次ストレージ

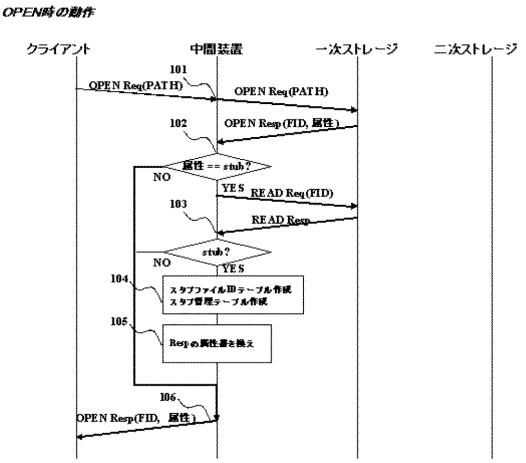
【図２】


1
クライアント
ファイルアクセスプロトコル
(NFS、CIFS)
3
中間装置
7
スタブファイル
8
実ファイル
5
6
ファイルサーバ
(一次ストレージ)
ファイルサーバ
(二次ストレージ)

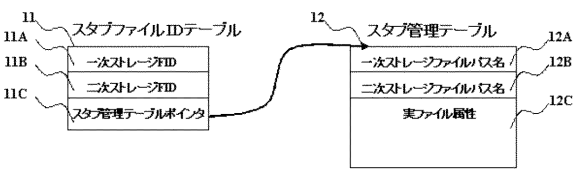
【図３】


スタブファイルフォーマット
7
7A
スタブ識別用マジックナンバー
7B
実ファイル属性
(ファイルサイズ、更新日時・・)
7C
二次ストレージ実ファイルパス名
7D
リザーブ
固定サイズ

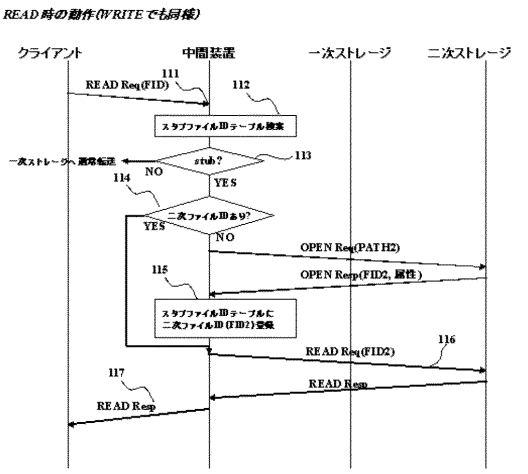
【図５】


OPEN時の動作
クライアント
中間装置
一次ストレージ
二次ストレージ
101
OPEN Req(PATH)
OPEN Req(PATH)
102
OPEN Resp(FID, 属性)
属性 == stub?
NO
YES
READ Req(FID)
103
READ Resp
stub?
NO
YES
104
スタブファイルID テーブル作成
スタブ管理テーブル作成
105
Respの属性値を換え
106
OPEN Resp(FID, 属性)

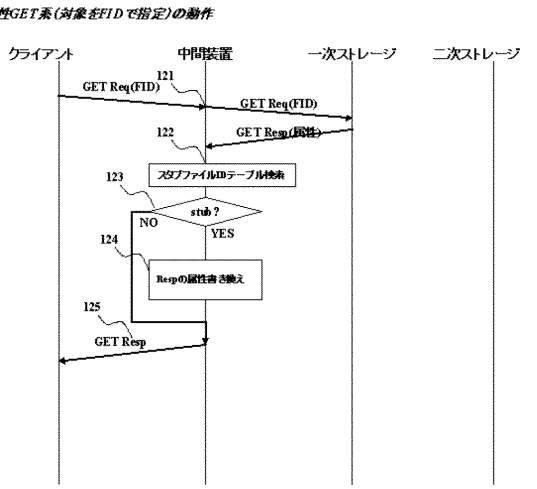
【図４】


スタブ管理テーブル・スタブファイルID 管理テーブル
11
スタブファイルIDテーブル
11A
一次ストレージFID
11B
二次ストレージFID
11C
スタブ管理テーブルポインタ
12
スタブ管理テーブル
12A
一次ストレージファイルパス名
12B
二次ストレージファイルパス名
12C
実ファイル属性

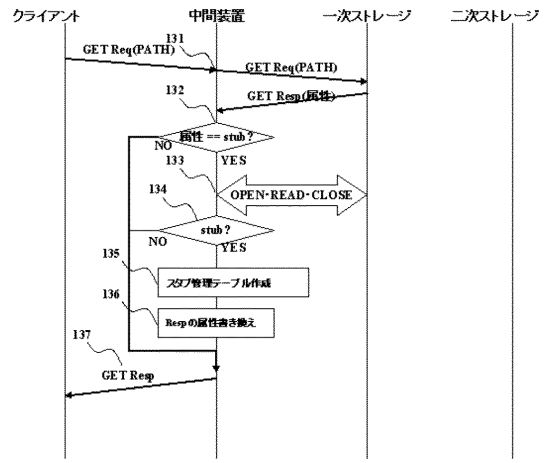
【図６】


READ時の動作(WRITEでも同様)
クライアント
中間装置
一次ストレージ
二次ストレージ
111
READ Req(FID)
112
スタブファイルIDテーブル検索
一次ストレージへ通常転送
stub?
113
NO
YES
114
二次ファイルIDあり?
YES
NO
OPEN Req(PATH2)
OPEN Resp(FID2, 属性)
115
スタブファイルIDテーブルに
二次ファイルID(FID2)登録
READ Req(FID2)
116
READ Resp
117
READ Resp

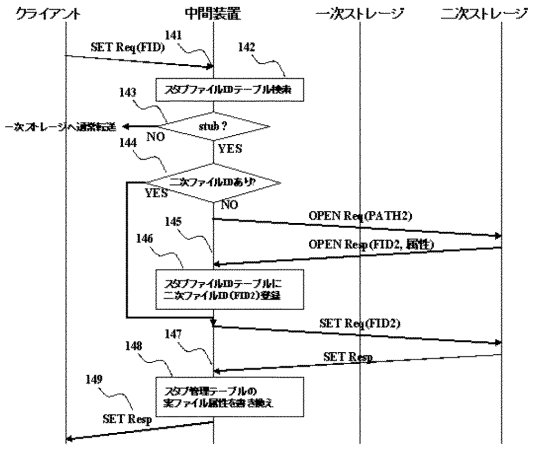
【図７】


属性GET系(対象をFIDで指定)の動作
クライアント
中間装置
一次ストレージ
二次ストレージ
121
GET Req(FID)
GET Req(FID)
122
GET Resp(属性)
123
スタブファイルIDテーブル検索
stub?
NO
YES
124
Respの属性書き換え
125
GET Resp

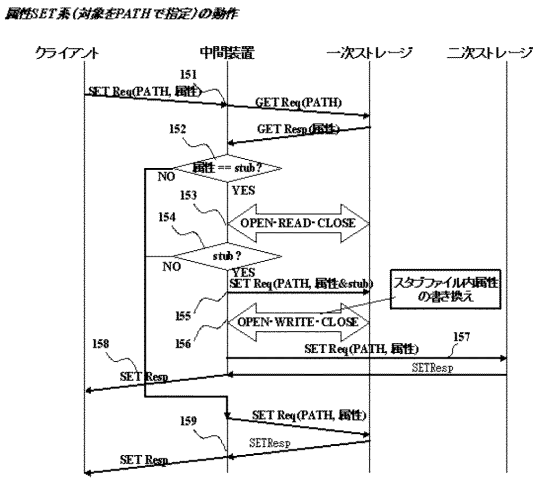
【図８】


属性GET系(対象をPATHで指定)の動作
クライアント
中間装置
一次ストレージ
二次ストレージ
131
GET Req(PATH)
GET Req(PATH)
132
GET Resp(属性)
属性 == stub?
NO
YES
133
OPEN・READ・CLOSE
134
stub?
NO
YES
135
スタブ管理テーブル作成
136
Respの属性書き換え
137
GET Resp

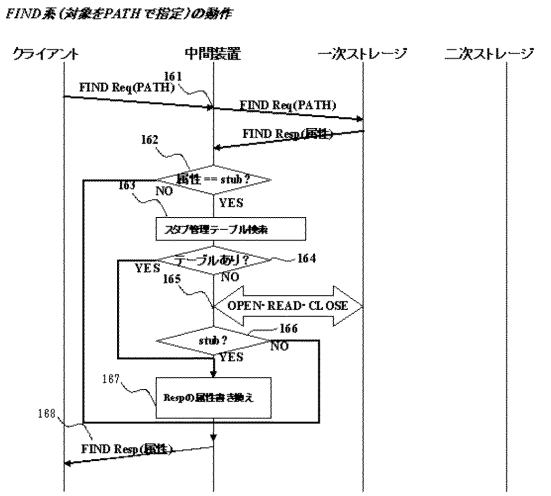
【図９】


属性SET系(対象をFIDで指定)の動作
クライアント
中間装置
一次ストレージ
二次ストレージ
141
SET Req(FID)
142
スタブファイルIDテーブル検索
143
一次ストレージへ通常転送
stub?
NO
YES
144
二次ファイルIDあり
YES
NO
OPEN Req(PATH2)
145
OPEN Resp(FID2, 属性)
146
スタブファイルIDテーブルに
二次ファイルID(FID2)登録
SET Req(FID2)
147
SET Resp
148
スタブ管理テーブルの
実ファイル属性を書き換え
149
SET Resp

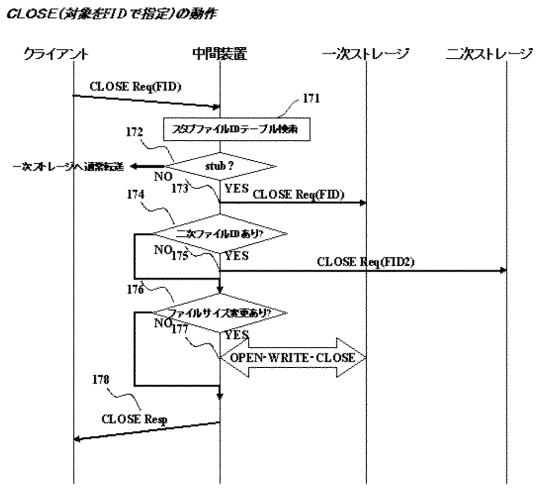
【図１０】



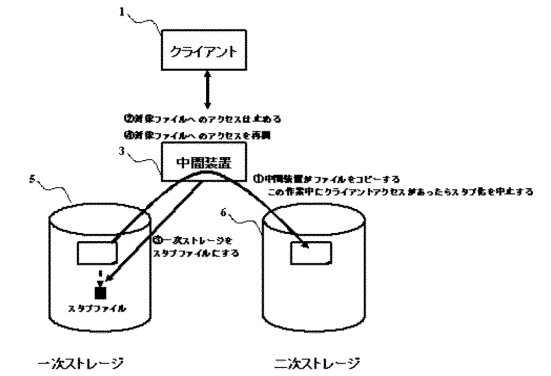
【図１１】



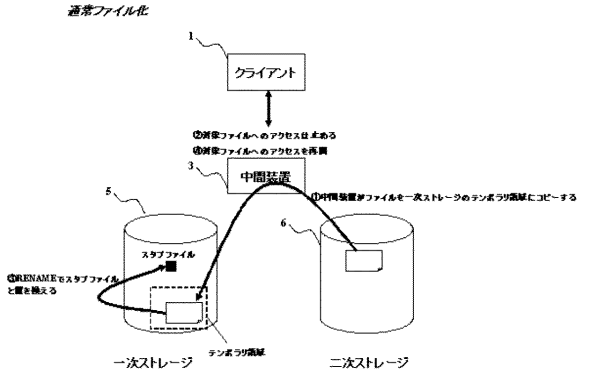
【図１２】



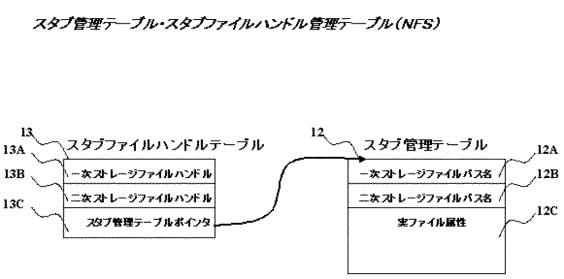
【図１３】



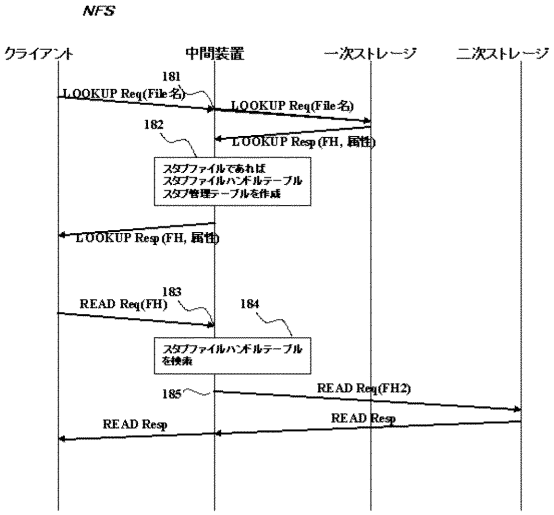
【図１４】



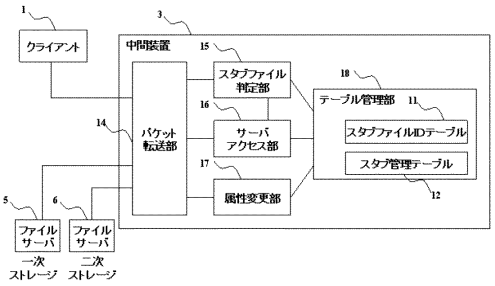
【図１５】



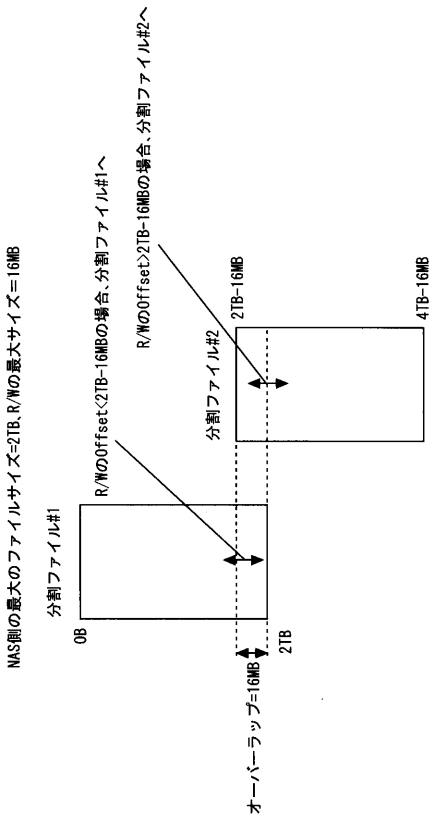
【図１６】



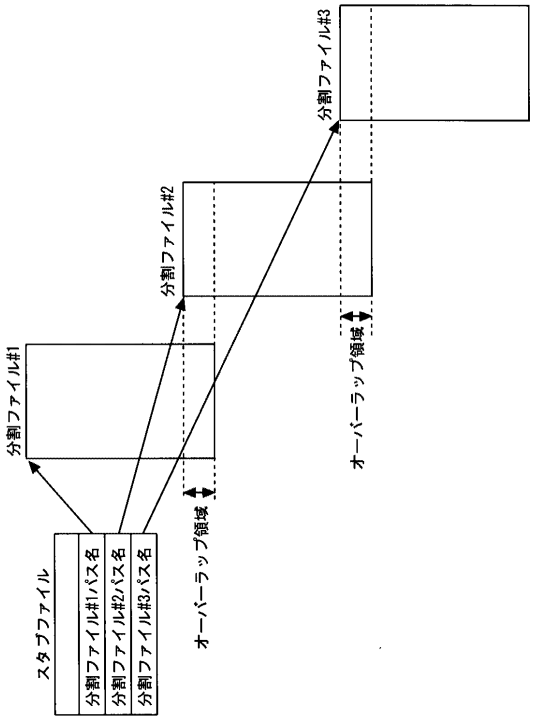
【図１７】



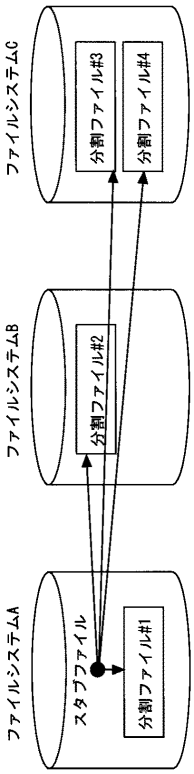
【図１８】



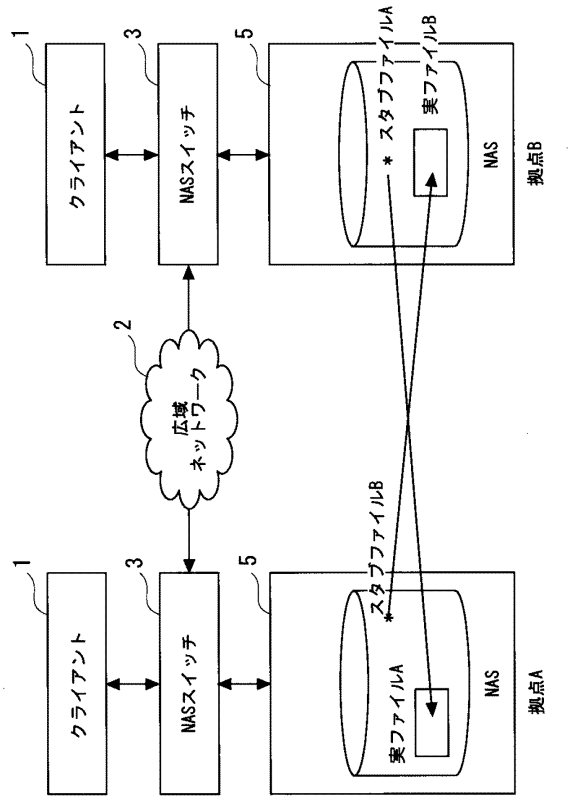
【図１９】



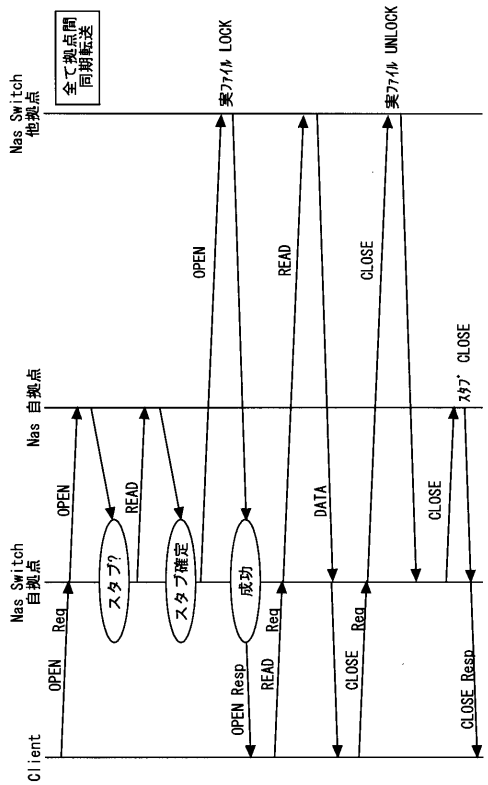
【図２０】



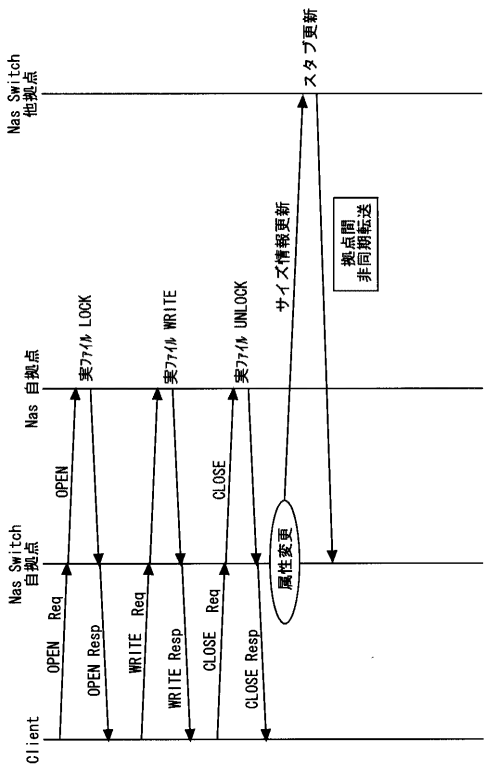
【図２１】



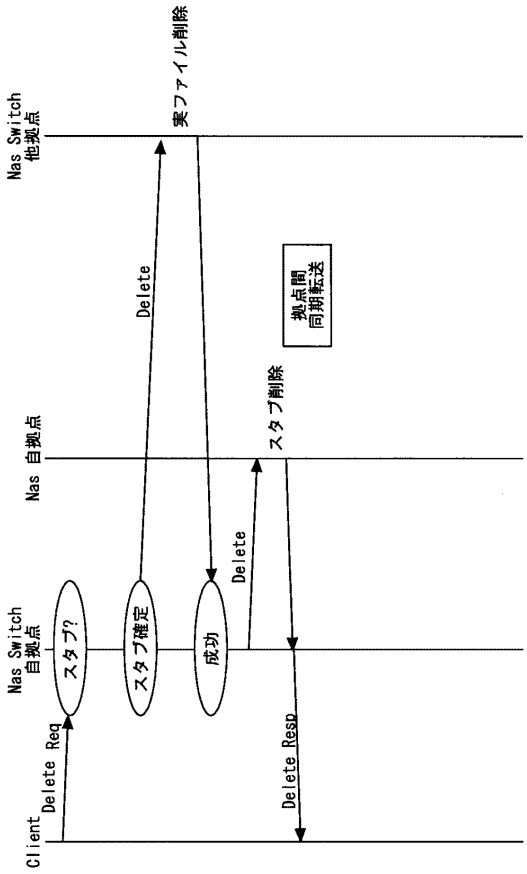
【図２２】



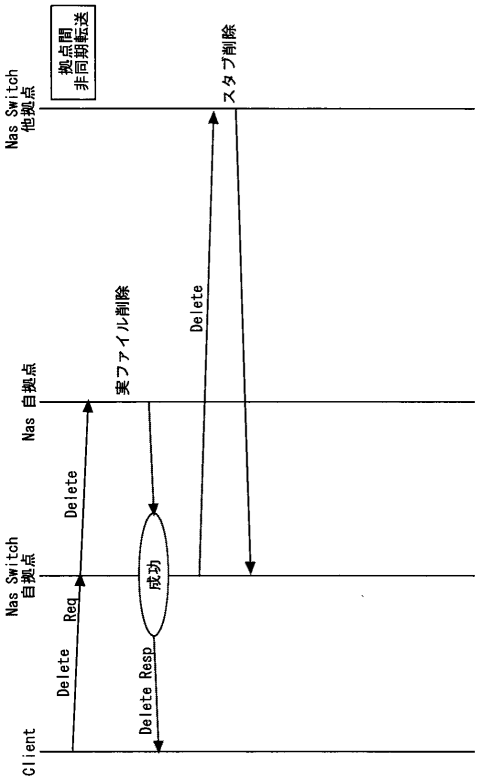
【図２３】



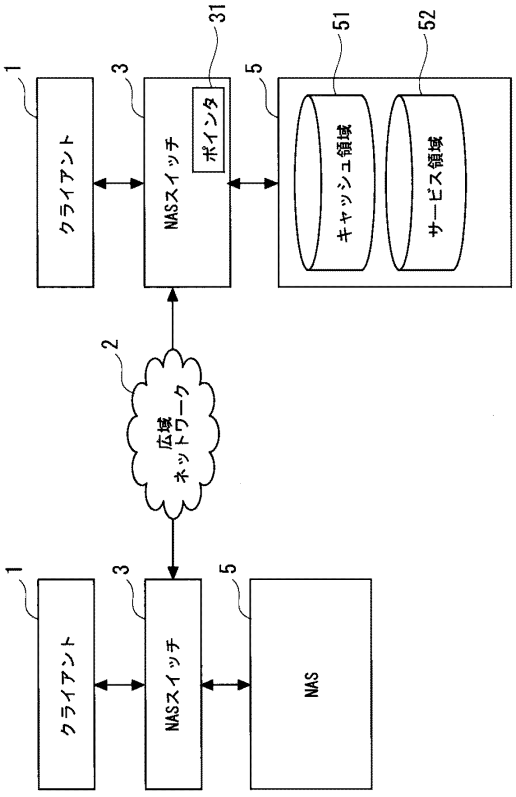
【図２４】



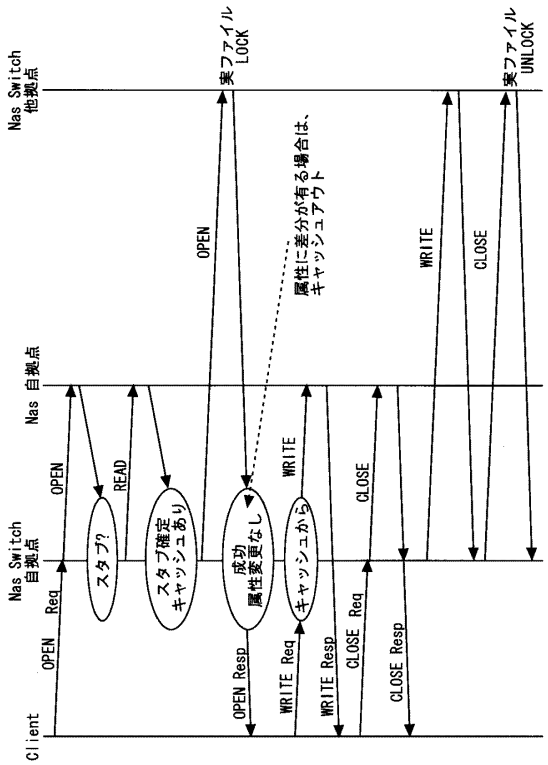
【図２５】



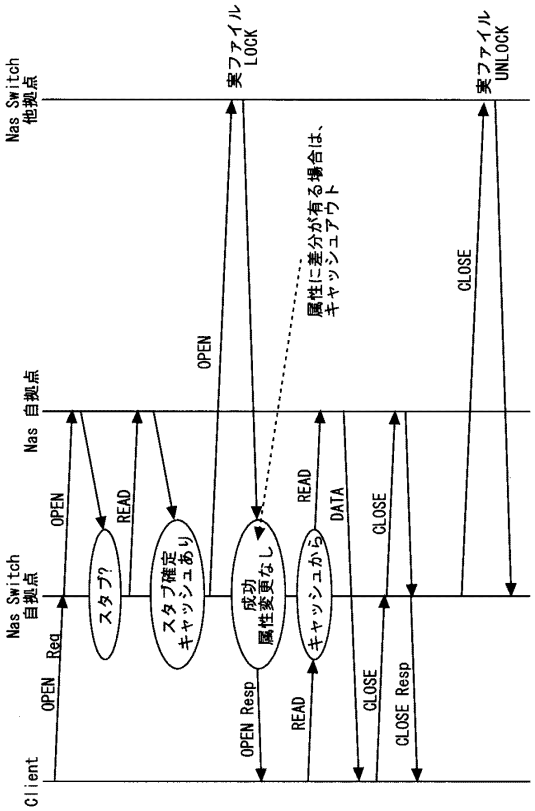
【図２６】



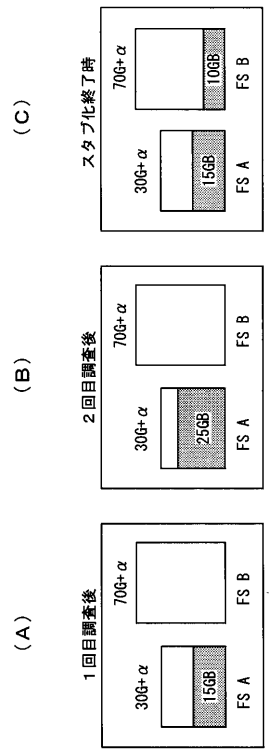
【図２７】



【図２８】


Nas Switch
他拠点
実ファイル
LOCK
実ファイル
UNLOCK
Nas 自拠点
OPEN
属性に差分が有る場合は、
キャッシュアウト
CLOSE
Nas Switch
自拠点
OPEN
READ
READ
DATA
CLOSE
Req
スタブ?
スタブ確定
キャッシュあり
成功
属性変更なし
キャッシュから
Client
OPEN
OPEN Resp
READ
CLOSE
CLOSE Resp

【図２９】


（A）
1回目調査後
70G+α
30G+α
15GB
FS A
FS B
（B）
2回目調査後
70G+α
30G+α
25GB
FS A
FS B
（C）
スタブ化終了時
70G+α
30G+α
15GB
10GB
FS A
FS B

フロントページの続き

(72)発明者　梶木　善裕
東京都港区芝五丁目７番１号　日本電気株式会社内

審査官　田川　泰宏

(56)参考文献　特開２００４－２９５４５７（ＪＰ，Ａ）
桂島　航，ＣＩＦＳサーバ仮想化方式の設計及び評価，電子情報通信学会技術研究報告，日本，社団法人電子情報通信学会，２００３年　7月２９日，Ｖｏｌ．１０３、Ｎｏ．２４８，p.73-78
山川　聡，ＮＡＳスイッチ：ＮＦＳサーバの仮想化統合技術の開発，電子情報通信学会技術研究報告，日本，社団法人電子情報通信学会，２００２年　8月１５日，Ｖｏｌ．１０２、Ｎｏ．２７５，p.13-18

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