(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2017-123040
(P2017-123040A)
(43)公開日　平成29年7月13日(2017.7.13)

(51)Int.Cl.		F I		テーマコード（参考）
G06F 12/00	(2006.01)	G06F	12/00	545F
		G06F	12/00	514R
		G06F	12/00	520J

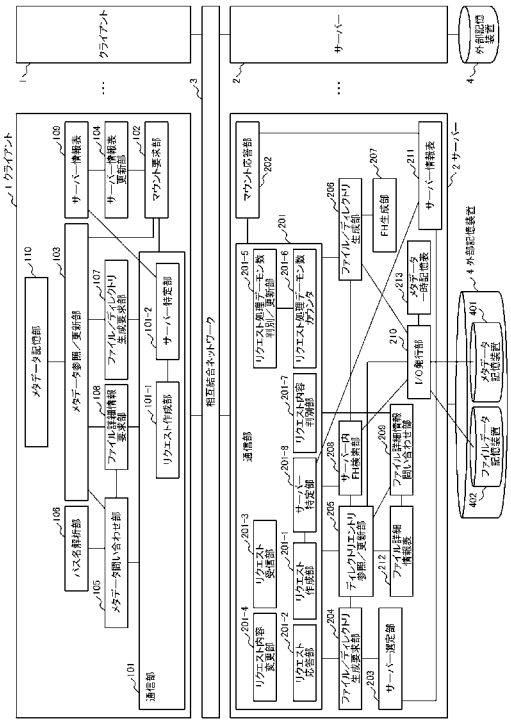
審査請求 未請求　請求項の数 10　ＯＬ　（全 33 頁）

(21)出願番号　特願2016-1571(P2016-1571)
(22)出願日　平成28年1月7日(2016.1.7)

(71)出願人　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　100124154
弁理士　下坂　直樹
(72)発明者　大谷　敦久
東京都港区芝五丁目７番１号
日本電気株式会社内

(54)【発明の名称】サーバー装置、分散ファイルシステム、分散ファイルシステム制御方法、および、プログラム

(57)【要約】
【課題】　ファイルデータもメタデータも複数のサーバー装置に分散配置されているシステムにおいて、クライアント装置による効率的なアクセス対象サーバー装置の特定と、多数のリクエストが特定のサーバー装置に集中した際の効率的なアクセスを可能とする。
【解決手段】　ファイルが存在するディレクトリを管理するサーバー装置は、オープン対象ファイルのファイルハンドルと、ファイル詳細情報のリクエストを１つにまとめて受信し、同時に処理する。この際、当該サーバー装置は、ファイル詳細情報をメモリ上にキャッシュしておく。当該サーバー装置は、所定閾値以上のリクエストを受信した場合、リクエストを１つにまとめて処理することを止める。この場合、クライアント装置が、個別にファイル詳細情報のリクエストを、ファイルを管理するサーバー装置に送信する。
【選択図】　図２



【特許請求の範囲】
【請求項１】
ファイルを記憶するファイルデータ記憶装置と、
ディレクトリ階層における直下のディレクトリ、または、ファイルの、ａ）識別子であって、当該ディレクトリ、または、当該ファイルを管理するサーバー装置の識別情報であるサーバーIDを含むファイルハンドル（以降、ＦＨと略記）、ｂ）名前、および、ｃ）ファイルまたはディレクトリの区別を示すファイルタイプを関連付けて記憶するディレクトリ、および、前記ファイルデータ記憶装置に記憶されているファイルのファイル詳細情報を記憶するメタデータ記憶装置と、に接続され、
ファイル詳細情報のキャッシュ用メモリ領域であるファイル詳細情報表と、
クライアント装置、若しくは、他の前記サーバー装置から処理リクエストを受信、または、他の前記サーバー装置にリクエストを送信するとともに、処理中または送信中のリクエスト数をカウントする通信部と、
ファイルのＦＨを入力されると、ＦＨが包含するサーバーIDの前記サーバー装置にファイル詳細情報を要求するリクエストを送信して、ファイル詳細情報を受信するファイル詳細情報問い合わせ部と、
クライアント装置から、ディレクトリ階層における直下のファイルの名前を含む、１）ファイルのＦＨ、および、ファイルタイプにたいする要求、並びに、２）ファイル詳細情報にたいする要求の２つの要求をまとめたリクエストが受信されたとき、ａ）前記メタデータ記憶装置に格納されているディレクトリから、入力された名前のファイルのＦＨとファイルタイプを取得し、ｂ１）前記通信部がカウントしたリクエスト数が所定閾値を超えておらず、ｃ１）受信された名前のファイルのファイル詳細情報が前記ファイル詳細情報表にキャッシュされていれば、キャッシュされているファイル詳細情報を取得し、ｃ２）前記ファイル詳細情報表にキャッシュされていなければ、取得したファイルのＦＨを前記ファイル詳細情報問い合わせ部に入力してファイル詳細情報を取得して、前記ファイル詳細情報表にキャッシュし、取得したファイルのＦＨ、ファイルタイプとファイル詳細情報とを前記クライアント装置に返信し、ｂ２）前記通信部がカウントしたリクエスト数が所定閾値を超えている場合は、入力されたリクエストを変更してファイル詳細情報の出力を中止し、取得したファイルのＦＨ、ファイルタイプを前記クライアント装置に返信し、
また、前記サーバー装置から返信されたファイルのＦＨを、改めてファイルを管理する前記サーバー装置に送信する前記クライアント装置から、または、他の前記サーバー装置の前記ファイル詳細情報問い合わせ部から、前記ファイルデータ記憶装置に記憶されているファイル、のＦＨを含むリクエストが受信されると、前記メタデータ記憶装置から、入力されたＦＨを識別子とするファイルのファイル詳細情報を取得して返信するディレクトリエントリ参照／更新部と、を備えたサーバー装置。
【請求項２】
前記サーバー装置は、さらに、
前記クライアント装置から、新たなディレクトリ、または、ファイルの第１の生成要求が送信されたとき、前記第１の生成要求に含まれる名前に依存しない方法で新たなディレクトリ、または、ファイルを管理する前記サーバー装置を選択するサーバー選定部と、
前記サーバー選定部が選択した前記サーバー装置に新たなディレクトリ、または、ファイルの第２の生成要求を送信して、生成された新たなディレクトリ、または、ファイルのＦＨを受信し、さらに、新たなファイルを生成した時はファイル詳細情報も受信する、ファイル／ディレクトリ生成要求部と、
前記第２の生成要求を受信すると、ａ）ディレクトリを生成して前記メタデータ記憶装置に格納、または、ｂ）ファイル詳細情報を生成して前記メタデータ記憶装置に格納すると共に前記第２の生成要求送信元に返信し、さらに、自装置のサーバーＩＤ及び自装置内一意の値から新たなディレクトリ、または、ファイルのＦＨを作成して、作成したＦＨを前記メタデータ記憶装置に格納すると共に前記第２の生成要求送信元に返信するファイル／ディレクトリ生成部とを、備え、

前記ディレクトリエントリ参照／更新部は、ファイル／ディレクトリ生成要求部が受信したＦＨ、並びに、前記第１の生成要求に含まれる名前、および、ファイルタイプを前記メタデータ記憶装置に格納されているディレクトリに記憶し、さらに、受信した場合はファイル詳細情報を前記ファイル詳細情報表に記憶する、請求項１のサーバー装置。

【請求項３】

前記ディレクトリエントリ参照／更新部は、前記クライアント装置から、ディレクトリ階層における直下のディレクトリの名前を入力されると、前記メタデータ記憶装置に格納されているディレクトリから、入力された名前のＦＨとファイルタイプを取得して出力する、請求項１乃至請求項２の何れか１項のサーバー装置と、

まず、１）上位のディレクトリを管理する前記サーバー装置の前記ディレクトリエントリ参照／更新部に、直下のディレクトリの名前を入力して、直下のディレクトリのＦＨとファイルタイプを取得することを、前記パス名に名前が含まれるディレクトリについて、ルートディレクトリを管理する予め定められた前記サーバー装置を起点に、上位から順に繰り返すディレクトリ探索を実行して、パス名中の最下位ディレクトリのＦＨとファイルタイプを取得し、次に、２）最下位ディレクトリのＦＨが包含するサーバーIDの前記サーバー装置の前記ディレクトリエントリ参照／更新部に、2-1）ファイルのＦＨ、および、ファイルタイプにたいする要求、並びに、2-2）ファイル詳細情報にたいする要求の２つの要求をまとめたリクエストを送信して、ａ）前記パス名のファイルのＦＨ、ファイルタイプ、および、ファイル詳細情報を取得する、または、ｂ）前記パス名のファイルのＦＨ、および、ファイルタイプを取得するメタデータ問い合わせ部と、

前記メタデータ問い合わせ部がファイル詳細情報を取得できなかったとき、前記メタデータ問い合わせ部が取得した前記パス名のファイルのＦＨを、改めてファイルを管理する前記サーバー装置に送信して、ファイル詳細情報を取得するファイル詳細情報要求部と、を備える前記クライアント装置と、を包含する分散ファイルシステム。

【請求項４】

前記クライアント装置は、

ディレクトリの、名前、ＦＨ，及び、ファイルタイプを関連付けてキャッシュするメタデータ記憶部を、さらに、備え、

前記メタデータ問い合わせ部は、前記ディレクトリ探索の過程で前記パス名から、ディレクトリの名前を取り出すと、ａ１）取り出した名前（以降、取得ディレクトリ名）が前記メタデータ記憶部にキャッシュされていなければ、前記取得ディレクトリ名のディレクトリの上位のディレクトリを管理する前記サーバー装置の前記ディレクトリエントリ参照／更新部に前記取得ディレクトリ名を送信して、前記取得ディレクトリ名のディレクトリのＦＨとファイルタイプを得るとともに、前記メタデータ記憶部に、前記取得ディレクトリ名、並びに、前記サーバー装置から取得したＦＨとファイルタイプをキャッシュし、ａ２）キャッシュされていれば前記メタデータ記憶部から前記取得ディレクトリ名のディレクトリのＦＨとファイルタイプを取得する、請求項２乃至請求項３の何れか１項の分散ファイルシステム。

【請求項５】

前記メタデータ記憶部は、ファイルの、名前、ＦＨ、ファイルタイプ、及び、ファイル詳細情報を関連付けて、さらに、キャッシュし、

前記メタデータ問い合わせ部は、前記パス名から、ファイルの名前を取り出すと、ａ１）取り出した名前（以降、取得ファイル名）が前記メタデータ記憶部にキャッシュされていなければ、前記サーバー装置から、前記取得ファイル名のファイルの、ＦＨ、ファイルタイプ、及び、ファイル詳細情報を得るとともに、前記メタデータ記憶部に、前記取得ファイル名、並びに、前記サーバー装置から取得したＦＨ、ファイルタイプ、及び、ファイル詳細情報をキャッシュし、ａ２）キャッシュされていれば前記メタデータ記憶部から前記取得ファイル名のファイルのＦＨ、ファイルタイプ、及び、ファイル詳細情報を取得する、請求項４の分散ファイルシステム。

【請求項６】

ファイルを記憶するファイルデータ記憶装置と、
ディレクトリ階層における直下のディレクトリ、または、ファイルの、ａ）識別子であって、当該ディレクトリ、または、当該ファイルを管理するサーバー装置の識別情報であるサーバーIDを含むファイルハンドル（以降、ＦＨと略記）、ｂ）名前、および、ｃ）ファイルまたはディレクトリの区別を示すファイルタイプを関連付けて記憶するディレクトリ、および、前記ファイルデータ記憶装置に記憶されているファイルのファイル詳細情報を記憶するメタデータ記憶装置と、に接続され、ファイル詳細情報のキャッシュ用メモリ領域であるファイル詳細情報表を備える前記サーバー装置が、
クライアント装置、若しくは、他の前記サーバー装置から処理リクエストを受信、または、他の前記サーバー装置にリクエストを送信するとともに、処理中または送信中のリクエスト数をカウントし、
クライアント装置から、ディレクトリ階層における直下のファイルの名前を含む、１）ファイルのＦＨ、および、ファイルタイプにたいする要求、並びに、２）ファイル詳細情報にたいする要求の２つの要求をまとめたリクエストを受信すると、ａ）前記メタデータ記憶装置に格納されているディレクトリから、入力された名前のファイルのＦＨとファイルタイプを取得し、ｂ１）カウントしたリクエスト数が所定閾値を超えておらず、ｃ１）受信された名前のファイルのファイル詳細情報が前記ファイル詳細情報表にキャッシュされていれば、キャッシュされているファイル詳細情報を取得し、ｃ２）前記ファイル詳細情報表にキャッシュされていなければ、取得したファイルのＦＨを含むリクエストを、当該ＦＨが包含するサーバーIDの前記サーバー装置に送信してファイル詳細情報を取得して、前記ファイル詳細情報表にキャッシュし、取得したファイルのＦＨ、ファイルタイプとファイル詳細情報とを前記クライアント装置に返信し、ｂ２）カウントしたリクエスト数が所定閾値を超えている場合は、入力されたリクエストを変更してファイル詳細情報の出力を中止し、取得したファイルのＦＨ、ファイルタイプを前記クライアント装置に返信し、
また、ファイルのＦＨを、改めてファイルを管理する前記サーバー装置に送信する前記クライアント装置から、または、他の前記サーバー装置から、前記ファイルデータ記憶装置に記憶されているファイル、のＦＨを含むリクエストを受信すると、前記メタデータ記憶装置から、入力されたＦＨを識別子とするファイルのファイル詳細情報を取得して返信する、分散ファイルシステム制御方法。
【請求項７】
前記サーバー装置が、さらに、
前記クライアント装置から、新たなディレクトリ、または、ファイルの第１の生成要求を受信すると、前記第１の生成要求に含まれる名前に依存しない方法で新たなディレクトリ、または、ファイルを管理する前記サーバー装置を選択し、
選択した前記サーバー装置に新たなディレクトリ、または、ファイルの第２の生成要求を送信して、生成された新たなディレクトリ、または、ファイルのＦＨを受信し、さらに、新たなファイルを生成した時はファイル詳細情報も受信し、
受信したＦＨ、並びに、前記第１の生成要求に含まれる名前、および、ファイルタイプを前記メタデータ記憶装置に格納されているディレクトリに記憶し、さらに、受信した場合はファイル詳細情報を前記ファイル詳細情報表に記憶し、
また、他の前記サーバー装置から前記第２の生成要求を受信すると、ａ）ディレクトリを生成して前記メタデータ記憶装置に格納、または、ｂ）ファイル詳細情報を生成して前記メタデータ記憶装置に格納すると共に前記第２の生成要求送信元に返信し、さらに、自装置のサーバーＩＤ及び自装置内一意の値から新たなディレクトリ、または、ファイルのＦＨを作成して、作成したＦＨを前記メタデータ記憶装置に格納すると共に前記第２の生成要求送信元に返信する、請求項６の分散ファイルシステム制御方法。
【請求項８】
前記サーバー装置が、前記クライアント装置から、ディレクトリ階層における直下のディレクトリの名前を受信すると、前記メタデータ記憶装置に格納されているディレクトリ

から、入力された名前のＦＨとファイルタイプを取得して出力し、
前記クライアント装置が、まず、１）上位のディレクトリを管理する前記サーバー装置に、直下のディレクトリの名前を入力して、直下のディレクトリのＦＨとファイルタイプを取得することを、前記パス名に名前が含まれるディレクトリについて、ルートディレクトリを管理する予め定められた前記サーバー装置を起点に、上位から順に繰り返すディレクトリ探索を実行して、パス名中の最下位ディレクトリのＦＨとファイルタイプを取得し、次に、２）最下位ディレクトリのＦＨが包含するサーバーIDの前記サーバー装置に、2-1）ファイルのＦＨ、および、ファイルタイプにたいする要求、並びに、2-2）ファイル詳細情報にたいする要求の２つの要求をまとめたリクエストを送信して、ａ）前記パス名のファイルのＦＨ、ファイルタイプ、および、ファイル詳細情報を取得し、または、ｂ）前記パス名のファイルのＦＨ、および、ファイルタイプを取得し、
最下位ディレクトリのＦＨが包含するサーバーIDの前記サーバー装置から、ファイル詳細情報を取得できなかったとき、当該サーバー装置から取得した前記パス名のファイルのＦＨを、改めてファイルを管理する前記サーバー装置に送信して、ファイル詳細情報を取得する、請求項６乃至請求項７の何れか１項の分散ファイルシステム制御方法。
【請求項９】
前記クライアント装置が、
ディレクトリの、名前，ＦＨ，及び、ファイルタイプを関連付けてキャッシュするメタデータ記憶部を、さらに、備え、
前記ディレクトリ探索の過程で前記パス名から、ディレクトリの名前を取り出すと、ａ１）取り出した名前（以降、取得ディレクトリ名）が前記メタデータ記憶部にキャッシュされていなければ、前記取得ディレクトリ名のディレクトリの上位のディレクトリを管理する前記サーバー装置に前記取得ディレクトリ名を送信して、前記取得ディレクトリ名のディレクトリのＦＨとファイルタイプを得るとともに、前記メタデータ記憶部に、前記取得ディレクトリ名、並びに、前記サーバー装置から取得したＦＨとファイルタイプをキャッシュし、ａ２）キャッシュされていれば前記メタデータ記憶部から前記取得ディレクトリ名のディレクトリのＦＨとファイルタイプを取得する、請求項７乃至請求項８の何れか１項の分散ファイルシステム制御方法。
【請求項１０】
ファイルを記憶するファイルデータ記憶装置と、
ディレクトリ階層における直下のディレクトリ、または、ファイルの、ａ）識別子であって、当該ディレクトリ、または、当該ファイルを管理するコンピュータの識別情報であるサーバーIDを含むファイルハンドル（以降、ＦＨと略記）、ｂ）名前、および、ｃ）ファイルまたはディレクトリの区別を示すファイルタイプを関連付けて記憶するディレクトリ、および、前記ファイルデータ記憶装置に記憶されているファイルのファイル詳細情報を記憶するメタデータ記憶装置と、に接続され、ファイル詳細情報のキャッシュ用メモリ領域であるファイル詳細情報表を備える前記コンピュータに、
クライアント装置、若しくは、他の前記コンピュータから処理リクエストを受信、または、他の前記コンピュータにリクエストを送信するとともに、処理中または送信中のリクエスト数をカウントし、
クライアント装置から、ディレクトリ階層における直下のファイルの名前を含む、１）ファイルのＦＨ、および、ファイルタイプにたいする要求、並びに、２）ファイル詳細情報にたいする要求の２つの要求をまとめたリクエストを受信すると、ａ）前記メタデータ記憶装置に格納されているディレクトリから、入力された名前のファイルのＦＨとファイルタイプを取得し、ｂ１）カウントしたリクエスト数が所定閾値を超えておらず、ｃ１）受信された名前のファイルのファイル詳細情報が前記ファイル詳細情報表にキャッシュされていれば、キャッシュされているファイル詳細情報を取得し、ｃ２）前記ファイル詳細情報表にキャッシュされていなければ、取得したファイルのＦＨを含むリクエストを、当該ＦＨが包含するサーバーIDの前記コンピュータに送信してファイル詳細情報を取得して、前記ファイル詳細情報表にキャッシュし、取得したファイルのＦＨ、ファイルタイプと

ファイル詳細情報とを前記クライアント装置に返信し、ｂ２）カウントしたリクエスト数が所定閾値を超えている場合は、入力されたリクエストを変更してファイル詳細情報の出力を中止し、取得したファイルのＦＨ、ファイルタイプを前記クライアント装置に返信し、
また、ファイルのＦＨを、改めてファイルを管理する前記コンピュータに送信する前記クライアント装置から、または、他の前記コンピュータから、前記ファイルデータ記憶装置に記憶されているファイル、のＦＨを含むリクエストを受信すると、前記メタデータ記憶装置から、入力されたＦＨを識別子とするファイルのファイル詳細情報を取得して返信する、処理を実行させるプログラム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、サーバー装置、分散ファイルシステム、分散ファイルシステム制御方法、および、プログラム、特に、多数の計算ノードにより構成される並列計算機または分散処理環境において使用されるサーバー装置、分散ファイルシステム、分散ファイルシステム制御方法、および、プログラムに関する。

【背景技術】

【０００２】

特許文献１には、ネットワークを介して接続された複数のストレージサーバー装置から構成される分散ファイル管理システムにおいて、ファイルやディレクトリを分散して管理する方法が記載されている。

【０００３】

特許文献２には、階層ディレクトリ構造における各ディレクトリ、例えば、／, usr,bin, tmp,xxx, yyy, zzzの情報を、ディレクトリ名称(識別名)によって決定される計算機に分散して保存する分散ファイルシステムのディレクトリ管理方法が記載されている。本文献の段落００１９は、ディレクトリ情報を階層ディレクトリ構造と関係なくバラして各計算機に分散保存するから、アクセスが１つの計算機に集中せず負荷分散でき、システムの性能が低下しないとしている。

【０００４】

特許文献３には、サーバーの負荷に応じてI/O（Input / Output）処理プロセスを増減させる方法が記述されている。つまり、I/O要求が増加傾向か、減少傾向にあるかをその傾きから予測することによりプロセスの増減を実施する。

【０００５】

非特許文献１には、メタデータサーバーを持たない分散ファイルシステムが記載されている。この分散ファイルシステムの各サーバーへの分散方法は、概ね以下のように動作する。まず、分散ハッシュテーブルのハッシュ値の範囲（例：０～2^{32}－１）を各サーバーへ均等に割り当てる。次に、ファイルのオープンの際、与えられたファイルのパス名からハッシュ値を計算する。そして、そのハッシュ値を含む範囲に割り当てられたサーバーを目的のファイルが存在するサーバーであると決定する。

【先行技術文献】

【特許文献】

【０００６】

【特許文献１】特許第５３６７４７０号公報
【特許文献２】特開平５－２３３４１号公報
【特許文献３】特許第４０８９５０６号公報

【非特許文献】

【０００７】

【非特許文献１】http://gluster.readthedocs.org/en/latest/Quick-Start-Guide/Architecture/ http://www.gluster.org/community/documentation/index.php/GlusterFS_Concepts

【発明の概要】
【発明が解決しようとする課題】
【０００８】
特許文献１が開示するファイルやディレクトリの分散管理方法には、以下の問題がある。
【０００９】
第１に、キャッシュ蓄積部（メタデータキャッシュ）は、ストレージサーバー装置にしか存在しない（図２２）。このため、指定されたパス名の中のすべてのディレクトリ、ファイルのメタデータがキャッシュされている場合でも、ユーザ端末装置（クライアント）とストレージサーバー装置の間で通信が最低１回は発生する。
【００１０】
第２に、各ストレージサーバー装置は、各ユーザの利用できるディレクトリ配下についてそれぞれキャッシュを持つことになると考えられる。このため、ストレージサーバー装置のキャッシュ蓄積部には、多数のユーザが同時に利用するマルチユーザ環境において、ユーザ数を考慮した大容量の記憶域が必要になる。したがって、システム価格が増加する、あるいは、キャッシュの容量不足により期待した効果が得られない可能性が考えられる。
【００１１】
第３に、パス名中にディレクトリが複数ある場合、ユーザ端末装置が応答を得るまでに複数回のストレージサーバー装置間の通信が必要になり、OneHop方式（段落００２８）とは言えない。
【００１２】
第４に、段落００２１には、ユーザ利用ファイル名に分割されたデータファイルの番号が付加され、データファイルの番号に対応したデータファイルのファイル名を出力する旨、及び、ユーザ利用ファイルを複数のデータファイルに分割して複数のストレージサーバー装置において分散して記憶する旨が記載されている。この方法では、ユーザ利用ファイル名がユーザによって変更された場合、データファイルは各ストレージサーバー装置に分散されているため、新たな名前を反映させるために複数回のサーバー間通信が必要となると考えられる。
【００１３】
第５に、段落００８４以降の記載によると、read/writeのリクエストの処理の延長で、パス検索を行うと解釈できる。つまり、ファイルのオープンにおいて行われるパス検索の処理がread/write処理に含まれるため、その分I/O性能に影響する。また、read/writeシステムコールのインターフェースが規格と異なるため、POSIX（Portable Operating System Interface for UNIX）に準拠していないと考えられる。
【００１４】
最後に、ユーザ端末装置がリクエストを送信する際、どのストレージサーバー装置へ送信するかをどのように決定するのかが不明瞭である。このため、以前に検索を実行しキャッシュが存在するストレージサーバー装置があったとしても、そのストレージサーバー装置以外のストレージサーバー装置へリクエストを送信し、最初からパス検索することが必要になる可能性があり、有効にキャッシュが機能するとは限らない。
【００１５】
特許文献２が開示する分散ファイルシステムのディレクトリ管理方法は、ディレクトリ名称(識別名)によって保存先の計算機を決める。この方法では、段落００２０のようなハッシュや名前の代わりにID（Identification）を利用する方法などを加味しても、分散のされ方に偏りが出る可能性を否定できない。特に、段落００１５、００２１のような名称の最初の一文字で保存先の計算機を決める分散方法では、分散のされ方に偏りが出る可能性が高い。例えば、分散方法が、アルファベットａ～ｍで始まる名称を有するディレクトリは計算機１１ａに配置するという場合、abc1，abc2，…，abc1000という名称に同時にアクセスされると、すべて同一の計算機１１ａに要求が集中するという不都合が起きる。

【００１６】
また、ディレクトリ名称の変更において、不都合が生じたり、煩雑な処理が必要となったりすることが考えられる。
【００１７】
さらに、特許文献２の方法は、計算機が要求されたディレクトリ情報を記憶していない場合には、該ディレクトリ情報をサーバーに問い合わせる（段落００１３、図４）。この方法は、同一サーバーに処理が集中する可能性が有るだけではなく、ファイルシステムを構成するサーバーと計算機の合計台数が多くなりコスト高となる。
【００１８】
非特許文献１が開示するハッシュによる分散方法は、以下のような問題点がある。
【００１９】
まず、非特許文献１には、ファイル名が変更された場合、新たな名前に基づくハッシュ値から決まるサーバーにポインターファイルを置き、実際にファイルのデータが存在するサーバーを指し示すとある。つまり、ファイル名変更後は、該ファイルへのアクセス時に通常とは異なる処理が必要となり、その分余計なオーバーヘッドが生じることになる。ファイル名の変更は、エンドユーザが通常行う操作であり、処理効率低下、処理遅延が懸念される。
【００２０】
さらに、ハードリンク（異なる名前で同一ファイルにアクセスするためのリンク機構）への対応が困難であり、また多数のファイルの生成において、ある範囲のハッシュ値が多数出現する可能性もある。
【００２１】
特許文献３が開示するI/O処理プロセスの増減は、プロセスの生成、終了という比較的重いとされる処理を伴うため、この処理によりCPU(Central Processing Unit)資源を使用することで、本来のクライアントから要求された処理を邪魔してしまう可能性が有る。
【００２２】
上記の問題に対処するために、本発明にかかるファイルシステムは、ファイルのデータだけではなく、ファイルの位置情報などファイルシステムを管理するメタデータも複数のサーバー配下に分散させて配置することでメタデータサーバーを用いない構成とする。そのうえで、計算ノード（ファイルシステムのクライアント）が、複数のサーバーの中からアクセス対象のファイルが存在するサーバーを特定する為の手段を備える。
【００２３】
ただし、このシステムにはメタデータサーバーのようなサーバーが存在しないため、クライアントは特定のサーバーに問い合わせることはできない。このため、ファイルシステムの最上位のディレクトリを管理するルートサーバーを設定する。そして、例えばファイルのオープン処理（open システムコール）に際して、指定されたパス名に基づき、クライアント主導で、パス名中の各ディレクトリを管理するサーバーを、ルートサーバーから順に辿って特定することとした。
【００２４】
従って、本発明にかかるシステムにおいては、クライアントが、如何に効率的にアクセス対象のファイルやディレクトリが格納されているサーバーを特定できる手段を用意するかが最初の課題となる。特に、ディレクトリ名や、ファイル名を元にした（ハッシュなどの）方法によりサーバーを特定するのではなく、より偏りが起き難い方法で各サーバーへ分散させる場合であっても、効率的にアクセス対象のファイルやディレクトリが格納されているサーバーを特定することが課題となる。
【００２５】
一方、多数の計算ノードがあると、ファイルオープンの処理などにおいて、ほぼ同時にディレクトリ配下のファイルにアクセスが集中することで、特定のサーバーにリクエストが集中し、クライアントへの応答が遅延してTAT（Turn Around Time）が増大する可能性がある。したがって、多数のリクエストが特定のサーバーに集中した際の効率化がさらな

る課題となる。
【課題を解決するための手段】
【００２６】
本発明にかかる分散ファイルシステム等は、ファイルを多数のクライアント装置がほぼ同時にオープンする場合、１）対象ファイルが存在するディレクトリを管理するサーバー装置へのオープン対象ファイルのファイルハンドルなどのリクエストと、対象ファイルを管理するサーバーへのファイル詳細情報のリクエストを１つにまとめる。これにより、サーバー装置が、両リクエストを同時に処理できるようにする。この際、２）ファイル詳細情報を対象ファイルが存在するディレクトリを管理するサーバーのメモリ上にキャッシュすることで、サーバー装置間の通信を削減する。
【００２７】
さらに、同一ディレクトリ配下の異なるファイルのオープンの場合で、３）ほぼ同時に所定閾値以上のリクエストを受信した場合は、処理待ちが発生してＴＡＴが悪化することを防ぐため、上記１）のようにリクエストを１つにまとめて処理することを止める。この場合は、クライアント装置が、個別にファイル詳細情報のリクエストを、ファイルを管理するサーバー装置に送信することで効率化する。
【００２８】
以上、１）乃至３）を反映して、本発明の実施の形態のサーバー装置等は、以下のように構成される。
【００２９】
本発明の１実施の形態のサーバー装置は、
ファイルを記憶するファイルデータ記憶装置と、ディレクトリ階層における直下のディレクトリ、または、ファイルの、ａ）識別子であって、当該ディレクトリ、または、当該ファイルを管理するサーバー装置の識別情報であるサーバーIDを含むファイルハンドル（以降、ＦＨと略記）、ｂ）名前、および、ｃ）ファイルまたはディレクトリの区別を示すファイルタイプを関連付けて記憶するディレクトリ、および、前記ファイルデータ記憶装置に記憶されているファイルのファイル詳細情報を記憶するメタデータ記憶装置と、に接続され、
ファイル詳細情報のキャッシュ用メモリ領域であるファイル詳細情報表と、
クライアント装置、若しくは、他の前記サーバー装置から処理リクエストを受信、または、他の前記サーバー装置にリクエストを送信するとともに、処理中または送信中のリクエスト数をカウントする通信部と、
ファイルのＦＨを入力されると、ＦＨが包含するサーバーIDの前記サーバー装置にファイル詳細情報を要求するリクエストを送信して、ファイル詳細情報を受信するファイル詳細情報問い合わせ部と、
クライアント装置から、ディレクトリ階層における直下のファイルの名前を含む、１）ファイルのＦＨ、および、ファイルタイプにたいする要求、並びに、２）ファイル詳細情報にたいする要求の２つの要求をまとめたリクエストが受信されたとき、ａ）前記メタデータ記憶装置に格納されているディレクトリから、入力された名前のファイルのＦＨとファイルタイプを取得し、ｂ１）前記通信部がカウントしたリクエスト数が所定閾値を超えておらず、ｃ１）受信された名前のファイルのファイル詳細情報が前記ファイル詳細情報表にキャッシュされていれば、キャッシュされているファイル詳細情報を取得し、ｃ２）前記ファイル詳細情報表にキャッシュされていなければ、取得したファイルのＦＨを前記ファイル詳細情報問い合わせ部に入力してファイル詳細情報を取得して、前記ファイル詳細情報表にキャッシュし、取得したファイルのＦＨ、ファイルタイプとファイル詳細情報とを前記クライアント装置に返信し、ｂ２）前記通信部がカウントしたリクエスト数が所定閾値を超えている場合は、入力されたリクエストを変更してファイル詳細情報の出力を中止し、取得したファイルのＦＨ、ファイルタイプを前記クライアント装置に返信し、
また、前記サーバー装置から返信されたファイルのＦＨを、改めてファイルを管理する前記サーバー装置に送信する前記クライアント装置から、または、他の前記サーバー装置の

前記ファイル詳細情報問い合わせ部から、前記ファイルデータ記憶装置に記憶されているファイル、のＦＨを含むリクエストが受信されると、前記メタデータ記憶装置から、入力されたＦＨを識別子とするファイルのファイル詳細情報を取得して返信するディレクトリエントリ参照／更新部と、を備える。

【００３０】

また、本発明の１実施の形態の分散ファイルシステムの制御方法は、

ファイルを記憶するファイルデータ記憶装置と、

ディレクトリ階層における直下のディレクトリ、または、ファイルの、ａ）識別子であって、当該ディレクトリ、または、当該ファイルを管理するサーバー装置の識別情報であるサーバーIDを含むファイルハンドル（以降、ＦＨと略記）、ｂ）名前、および、ｃ）ファイルまたはディレクトリの区別を示すファイルタイプを関連付けて記憶するディレクトリ、および、前記ファイルデータ記憶装置に記憶されているファイルのファイル詳細情報を記憶するメタデータ記憶装置と、に接続され、ファイル詳細情報のキャッシュ用メモリ領域であるファイル詳細情報表を備える前記サーバー装置が、

クライアント装置、若しくは、他の前記サーバー装置から処理リクエストを受信、または、他の前記サーバー装置にリクエストを送信するとともに、処理中または送信中のリクエスト数をカウントし、

クライアント装置から、ディレクトリ階層における直下のファイルの名前を含む、１）ファイルのＦＨ、および、ファイルタイプにたいする要求、並びに、２）ファイル詳細情報にたいする要求の２つの要求をまとめたリクエストを受信すると、ａ）前記メタデータ記憶装置に格納されているディレクトリから、入力された名前のファイルのＦＨとファイルタイプを取得し、ｂ１）カウントしたリクエスト数が所定閾値を超えておらず、ｃ１）受信された名前のファイルのファイル詳細情報が前記ファイル詳細情報表にキャッシュされていれば、キャッシュされているファイル詳細情報を取得し、ｃ２）前記ファイル詳細情報表にキャッシュされていなければ、取得したファイルのＦＨを含むリクエストを、当該ＦＨが包含するサーバーIDの前記サーバー装置に送信してファイル詳細情報を取得して、前記ファイル詳細情報表にキャッシュし、取得したファイルのＦＨ、ファイルタイプとファイル詳細情報とを前記クライアント装置に返信し、ｂ２）カウントしたリクエスト数が所定閾値を超えている場合は、入力されたリクエストを変更してファイル詳細情報の出力を中止し、取得したファイルのＦＨ、ファイルタイプを前記クライアント装置に返信し、

また、ファイルのＦＨを、改めてファイルを管理する前記サーバー装置に送信する前記クライアント装置から、または、他の前記サーバー装置から、前記ファイルデータ記憶装置に記憶されているファイル、のＦＨを含むリクエストを受信すると、前記メタデータ記憶装置から、入力されたＦＨを識別子とするファイルのファイル詳細情報を取得して返信する。

【００３１】

また、本発明の１実施の形態のプログラムは、

ファイルを記憶するファイルデータ記憶装置と、

ディレクトリ階層における直下のディレクトリ、または、ファイルの、ａ）識別子であって、当該ディレクトリ、または、当該ファイルを管理するコンピュータの識別情報であるサーバーIDを含むファイルハンドル（以降、ＦＨと略記）、ｂ）名前、および、ｃ）ファイルまたはディレクトリの区別を示すファイルタイプを関連付けて記憶するディレクトリ、および、前記ファイルデータ記憶装置に記憶されているファイルのファイル詳細情報を記憶するメタデータ記憶装置と、に接続され、ファイル詳細情報のキャッシュ用メモリ領域であるファイル詳細情報表を備える前記コンピュータに、

クライアント装置、若しくは、他の前記コンピュータから処理リクエストを受信、または、他の前記コンピュータにリクエストを送信するとともに、処理中または送信中のリクエスト数をカウントし、

クライアント装置から、ディレクトリ階層における直下のファイルの名前を含む、１）

ファイルのＦＨ、および、ファイルタイプにたいする要求、並びに、２）ファイル詳細情報にたいする要求の２つの要求をまとめたリクエストを受信すると、ａ）前記メタデータ記憶装置に格納されているディレクトリから、入力された名前のファイルのＦＨとファイルタイプを取得し、ｂ１）カウントしたリクエスト数が所定閾値を超えておらず、ｃ１）受信された名前のファイルのファイル詳細情報が前記ファイル詳細情報表にキャッシュされていれば、キャッシュされているファイル詳細情報を取得し、ｃ２）前記ファイル詳細情報表にキャッシュされていなければ、取得したファイルのＦＨを含むリクエストを、当該ＦＨが包含するサーバーIDの前記コンピュータに送信してファイル詳細情報を取得して、前記ファイル詳細情報表にキャッシュし、取得したファイルのＦＨ、ファイルタイプとファイル詳細情報とを前記クライアント装置に返信し、ｂ２）カウントしたリクエスト数が所定閾値を超えている場合は、入力されたリクエストを変更してファイル詳細情報の出力を中止し、取得したファイルのＦＨ、ファイルタイプを前記クライアント装置に返信し、
また、ファイルのＦＨを、改めてファイルを管理する前記コンピュータに送信する前記クライアント装置から、または、他の前記コンピュータから、前記ファイルデータ記憶装置に記憶されているファイル、のＦＨを含むリクエストを受信すると、前記メタデータ記憶装置から、入力されたＦＨを識別子とするファイルのファイル詳細情報を取得して返信する、処理を実行させる。

【発明の効果】

【００３２】

本発明にかかるサーバー装置は、メタデータも複数のサーバー装置に分散配置されている分散ファイルシステムにおいて、クライアント装置による効率的なアクセス対象サーバー装置の特定、および、多数のリクエストが特定のサーバー装置に集中した際の効率的なアクセスを可能とする。

【図面の簡単な説明】

【００３３】

【図１】図１は、第１の実施の形態にかかる分散ファイルシステム５の概要を示す構成図である。

【図２】図２は、第１の実施の形態にかかるクライアント装置、サーバー装置、および、外部記憶装置の内部構成を示す図である。

【図３】図３は、メタデータとサーバー装置との階層関係を表す概念図である。

【図４】図４は、ファイルのオープン、リードにおけるディレクトリ探索、リード要求の送信／応答の様子を示す図である（その１）。

【図５】図５は、ファイルのオープン、リードにおけるディレクトリ探索、リード要求の送信／応答の様子を示す図である（その２）。

【図６】図６は、ファイルのオープン、リードにおけるディレクトリ探索、リード要求の送信／応答の様子を示す図である（その３）。

【図７】図７は、クライアント装置側のファイルオープン処理の動作例のフローチャートである（その１）。

【図８】図８は、クライアント装置側のファイルオープン処理の動作例のフローチャートである（その２）。

【図９】図９は、サーバー装置側のファイルオープン処理の動作例のフローチャートである（その１）。

【図１０】図１０は、サーバー装置側のファイルオープン処理の動作例のフローチャートである（その２）。

【図１１】図１１は、サーバー装置側のファイルオープン処理の動作例のフローチャートである（その３）。

【図１２】図１３は、上位ディレクトリ管理サーバー装置側のファイル、ディレクトリ生成処理の動作例のフローチャートである。

【図１３】図１３は、生成サーバー装置側のファイル、ディレクトリ生成処理の動作例の

フローチャートである。
【図１４】図１４は、クライアント装置側のマウント処理の動作例のフローチャートである。
【図１５】図１５は、サーバー装置側のマウント処理の動作例のフローチャートである。
【図１６】図１６は、コンピュータ装置の構成図である。
【発明を実施するための形態】
【００３４】
＜第１の実施の形態＞
＜概要＞
図１は、本実施の形態にかかる分散ファイルシステム５の概要を示す構成図である。分散ファイルシステム５においては、図１が示すように、例えば複数の計算ノードであるクライアント装置１（以降、クライアント１と略記）が相互結合ネットワーク３に接続されており、ユーザのジョブを実行する。また、例えば複数のサーバー装置２（以降、サーバー２と略記）も同じ相互結合ネットワーク３に接続されており、任意のクライアント１とサーバー２の間、及び、任意のサーバー２同士間の通信が可能となっている。各サーバー２の配下には少なくとも１台の外部記憶装置４が接続される。
【００３５】
分散ファイルシステム５は、システム内のファイルを管理する為に、階層的なファイルディレクトリを用いる。
【００３６】
仮に、分散ファイルシステム５に、メタデータサーバーと呼ばれるような、システム全体のメタデータを管理するサーバー２があると、メタデータを更新する処理、例えばファイルやディレクトリの生成や削除、が同時に多数発生した場合、メタデータサーバーにアクセスが集中し、分散ファイルシステム５のボトルネックになる可能性がある。この問題を避けるため、分散ファイルシステム５は、ファイルのデータだけではなく、ファイルの位置情報などファイルを管理するメタデータも複数のサーバー２に分散させて配置する。すなわち、分散ファイルシステム５は、メタデータサーバーを用いない構成を採用している。
【００３７】
分散ファイルシステム５は、メタデータサーバーは用いないが、ファイルシステムのルートディレクトリを管理するサーバー２（ルートサーバーと呼称）を用いる。ルートディレクトリとは、マウント対象のファイルシステムの先頭のディレクトリを指す。分散ファイルシステム５のシステム構築時にシステム管理者が、ルートサーバーを決めておく。
【００３８】
なお、以降の説明において、あるディレクトリを管理するサーバー２は、当該ディレクトリを記憶し、クライアント１や他のサーバー２の要求に応じて当該ディレクトリに格納されている下位のディレクトリやファイルに関するデータを出力するサーバー２を指す。また、あるファイルを管理するサーバー２とは、当該ファイルおよび当該ファイルに関する詳細情報を記憶し、クライアント１や他のサーバー２の要求に応じて当該ファイルのデータを出力するサーバー２を指す。一台のサーバー２が、あるディレクトリを管理すると同時に、あるファイルを管理することも有る。
【００３９】
ファイル、ディレクトリの生成の際に、上位ディレクトリを管理するサーバー２が、新たに生成するファイル、ディレクトリを管理するサーバー２を、後述するサーバー情報表２１１に登録されているサーバー２から選定する。この際のサーバー２の選定は、ファイルやディレクトリの名前、またはそれらに付される一意のID（IDentification）などに依存しない方法を用いて行われる。この方法は、例えば、ラウンドロビンや一様乱数に基づく選定方法である。
【００４０】
選定されたサーバー２の識別子であるサーバーIDは、選定されたサーバー２から選定元

のサーバー２に返されるFH（File handle）に含まれる。FHは、生成されたファイル、ディレクトリの名称、ファイルタイプと共にメタデータとして、上位ディレクトリ毎に選定元のサーバー２が保持する。

【００４１】

ここで、FHは、ファイル、ディレクトリを分散ファイルシステム５内で一意に識別するための識別子であり、サーバーIDの他にサーバー２内で一意の数値も含む。また、ファイルタイプは、ファイルかディレクトリかの区別情報である。

【００４２】

この結果、各サーバー２は、自装置が管理する各ディレクトリ配下にあるファイル、ディレクトリの名とそれらの位置情報であるサーバーIDの一覧をディレクトリエントリとして持つことができる。

【００４３】

分散ファイルシステム５においては、各サーバー２がクライアント１からのリクエスト毎にそれぞれ独立してこのファイル、ディレクトリの生成処理を行うので、システム全体としての排他制御などは不要である。さらに、各サーバー２が、サーバー２の選定に際してラウンドロビンや一様乱数に基づく方法で行うので、ファイル、ディレクトリの名前、及びディレクトリ構造とは関係なく、ディレクトリ、ファイルをシステム内でほぼ均等に配置できる。例えば、サーバー２が３台ある場合、１番目のサーバー２は２、３、１番目、２番目のサーバー２は３、１、２番目、３番目のサーバー２は１、２、３番目のサーバー２の順で割り当てる方法により、ディレクトリ、ファイルをシステム内でほぼ均等に配置できる。

【００４４】

ファイル名を変更する場合は、その上位ディレクトリのディレクトリエントリ内の名前を書き換えるだけでよい。さらに、ハードリンクは、例えばディレクトリエントリ内に名前は異なるが同一のFH、ファイルタイプを持つエントリを作り、対象となるファイルを管理するサーバー２で該ファイルのファイル詳細情報のリンク数に１を加算することで実現可能となる。

【００４５】

運用中に新たに、分散ファイルシステム５にサーバー２を追加した場合でも、既存のファイル、ディレクトリの配置は変わらないので、特別な処理は不要である。

【００４６】

なお、ルートサーバーは、ファイルシステムの先頭のディレクトリのみを管理するのではなく、他のサーバー２同様にサーバー２選定の対象にもなるため、データ、メタデータの管理に関して他のサーバー２との違いはない。

【００４７】

ファイル、ディレクトリの生成に際して選定されたサーバー２は、生成したファイル、ディレクトリのFH毎のディレクトリエントリを設け、更新／参照時刻、パーミッションなどの詳細情報を保持する。さらにファイルを生成した場合は、ファイルのサイズ、チャンクサイズ、並列数などをファイル詳細情報に含めて記憶する。

【００４８】

これにより、分散ファイルシステム５の各サーバー２は、ルートサーバーを頂点とした、例えば、図３に示すような階層的なツリー構造を作る。

ａ）ファイルシステムの最上位のディレクトリを管理するルートサーバーは、自装置が記憶するルートディレクトリの中に、そのディレクトリ配下にある「子ディレクトリ」を管理するサーバー２を特定できる情報を記憶している。

ｂ）子ディレクトリを管理するサーバー２は、自装置が記憶する子ディレクトリの中に、そのディレクトリ配下にある「孫ディレクトリ」を管理するサーバー２を特定できる情報を記憶している。

ｃ）孫ディレクトリを管理するサーバー２は、自装置が記憶する孫ディレクトリの中に、そのディレクトリ配下にあるアクセス対象のファイルを管理するサーバー２を特定できる

情報を記憶している。

【００４９】

上記構造に基づき、クライアント１は、ファイルシステムをマウントする際、そのマウント先としてルートサーバーを指定し、ルートサーバーからルートディレクトリのFHとファイルシステムを構成している全サーバー２のサーバーIDとIPアドレスの対応表を受け取る。

【００５０】

次に、クライアント１は、ファイルのオープン処理などにおいてディレクトリ探索を行う。すなわち、クライアント１は、指定されたパス名の中の最上位ディレクトリの直下にあるディレクトリを管理するサーバー２をルートサーバーに問い合わせ、さらにその下のディレクトリを管理するサーバー２を問い合わせることを繰り返す。クライアント１は、このディレクトリ探索により、指定されたパス名のファイルを管理するサーバー２のサーバーIDを取得する。

【００５１】

図４乃至図６は、ファイルのオープン、リードにおけるディレクトリ探索、リード要求の送信／応答の様子を示す。

【００５２】

図４において、ユーザがクライアント１で実行するAP（Application Program）が、マウントポイントが“/mnt”であって、パスが“/mnt/home/user1/file1”であるファイルをオープンするシステムコールを発行したとする。クライアント１は、ルートサーバーにマウント時に受け取ったルートディレクトリのFHと共に配下のディレクトリ“home”を「名前」として送信して問い合わせ、ディレクトリ“home” のFHを受け取る（図４中のａ）。さらに、クライアント１は、受け取ったFHに含まれるサーバーIDから“home”を管理するサーバー２を特定してそのサーバー２にディレクトリ“user1”を「名前」として送信して問い合わせ、ディレクトリ“user1” のFHを受け取る（図中のｂ）。最後に クライアント１は、同様に“file1” を問い合わせ、ファイル“file1”のFHを受け取る（図４中のｃ）。

【００５３】

この結果、クライアント１は、APが発行したファイル“file1”に対するリード要求を、当該ファイルを管理するサーバー＃４に送信することが出来る（図５中のｄ）。

【００５４】

なお、この操作は、漸化式$a_{i+1} = f(g(a_i), a_i, \text{名前})$で表すことができる。ここで、$a_i$はパス名中のi番目の名前に対応するFH、初期値$a_0$はマウント時にルートサーバーから返されたルートディレクトリのFHである。さらに、gはFHに対応するディレクトリを管理するサーバー２のサーバーIDを返す関数、fはサーバーID、FH、名前から、その名前に対応するFHを返す関数である。また、iは$i = 0,\cdots,n-1$ (nは、パス名中の名前の数)の整数である。

【００５５】

また、この過程において、クライアント１は、途中で受信したディレクトリを管理するサーバー２の情報をメタデータキャッシュ（図４乃至図６のクライアント１を参照）に保持する。このメタデータキャッシュは、後述するクライアント１のメタデータ記憶部１１０に格納される。そして後刻、上記のディレクトリ“user1”配下の他のファイル、例えば“file2”、のオープン処理において、既に問い合わせ済みのディレクトリ名がパス名中に含まれる場合は、このメタデータキャッシュを参照することでサーバー２との通信回数を削減する。すなわちクライアント１は、サーバー＃１、＃２との通信を省略し、サーバー＃３からファイル“file2”のFHを受け取ることが出来る（図６中のａ）。

【００５６】

なお、特許文献２は、頻繁にディレクトリ内容が書き換えられる場合、上記メタデータキャッシュについての不都合を指摘している（段落０００７、０００８）。しかし、ファイルシステムの上位のディレクトリは、システム管理者でなければ書き換えはできないこ

とが通常であり、またエンドユーザが書き換え可能なディレクトリであっても、上位のディレクトリほど書き換えの頻度は少ない。このため、少なくとも上位のディレクトリについてはメタデータキャッシュが有効に機能する。

【００５７】

また、ディレクトリ探索を実施する場合でも、その探索は比較的下位のディレクトリのみを対象とするため、ルートサーバーなどへ探索要求が集中することはなく、各サーバー２へ適度に負荷分散できる。

【００５８】

次に、多数のクライアント１が同一ディレクトリ配下のファイルをオープンする場合の２つのケースについて、分散ファイルシステム５におけるI/0の効率化方法を示す。

【００５９】

ケース１：

同一ファイルへの並列I/0などの目的で、多数のクライアント１が同一ファイルをオープンする場合

ケース２：

多数のクライアント１が同一ディレクトリ配下の異なるファイルをオープンする場合

ここで、効率化の前提となる事項について説明する。

【００６０】

ファイルのオープン処理において、ディレクトリ探索の最後の処理が、オープン対象のファイルのメタデータを取得することである。この際、クライアント１は、該ファイルのファイル詳細情報も必要としている。このファイル詳細情報は、例えば、ファイルのサイズ、更新／参照時刻、ファイルのチャンクサイズ、使用するサーバー数である。クライアント１は、この問い合わせを、該ファイルが存在するディレクトリを管理するサーバー２に対して行うが、これらファイル詳細情報は、もともと該ファイルを管理するサーバー２が保持している。

【００６１】

このため、ファイル詳細情報をクライアント１が得る手段として下記２つの方法がある。

【００６２】

(ア)該ファイルが存在するディレクトリを管理するサーバー２は、FHとファイルタイプのみをクライアント１へ返却する。その後、クライアント１は、返却されたFHから該ファイルを管理するサーバー２を特定し、そのサーバー２へファイル詳細情報を要求する。

【００６３】

(イ)該ファイルが存在するディレクトリを管理するサーバー２が、該ファイルを管理するサーバー２へファイル詳細情報を問い合わせるサーバー２間通信（例えば、図６のａ’）を実施し、その結果をファイルのFHなどと共にクライアント１へ返す。つまり、本来は（ア）のように２つの要求であるものを１つの要求にまとめる。

【００６４】

ここで、上述した２つのケースについて、I/0の効率化方法を説明する。

【００６５】

ケース１：

多数のクライアント１がほぼ同時期に上記（ア）によりファイル詳細情報を得る場合、クライアント１がサーバー２と通信する回数が、クライアント１毎に１回増える。クライアント１が５１２台あれば、クライアント１がサーバー２と通信する回数が５１２回となる。さらに、該ファイルを管理するサーバー２にもファイル詳細情報の要求が集中する可能性がある。ファイル詳細情報はデータ量が多少多くなるので、通信回数だけではなくサーバー２のI/0やCPU等の負荷の面からも好ましくない。

【００６６】

このため、サーバー２間通信を行う（イ）の方法を基にして、該ファイルを管理するサーバー２から受け取った（例えば、図６中のａ‘）ファイル詳細情報を、該ファイルが存

在するディレクトリを管理するサーバー２にキャッシュする。つまり、当該サーバー２は、該ファイルのFHと紐づくファイル詳細情報表２１２をキャッシュとしてメモリ上に持ち、ディレクトリエントリから参照可能とする。そして、当該サーバー２は、ファイル詳細情報表２１２にFHに対応するエントリがあれば、それを参照することで、ファイル詳細情報をクライアント１に出力する（例えば、図６中のａ）。これにより、分散ファイルシステム５は、サーバー２間の通信回数（例えば、図６中のサーバー＃３と＃５の間）を削減し、さらに該ファイルを管理するサーバー２（例えば、図６中のサーバー＃５）へのI/OやCPU等の負荷を減らす。

【００６７】

ケース２

対象ファイルがクライアント１毎にそれぞれ異なるため、（イ）の場合、サーバー２間通信は対象ファイルごとに必要になる。ただ、サーバー２ではケース１かケース２かの判別はできない。すなわち、次に来るリクエストが何であるかはそれを受け取るまでわからない。このため、サーバー２は、次に述べるデーモン数の範囲内では、ケース１と同様に（イ）の方法を選択する。

【００６８】

（イ）の方法を使った場合、サーバー２間通信の間、ファイルが存在するディレクトリを管理するサーバー２上（例えば、図６中のサーバー＃３）のデーモンが対象ファイルごとに１つブロックされる。ここで、デーモンとは、クライアント１からのリクエストを処理するプロセスである。

【００６９】

多数のクライアント１からのアクセスが集中し、デーモン数よりもクライアント１からのリクエスト数が多い場合は、デーモンが空くまでキューイングされて待たされることになり、その分TAT（Turn Around Time）が悪化することになる。ある特定のクライアント１のTATの悪化は、そのクライアント１を含むマルチノードジョブ全体のTATの悪化につながる。

【００７０】

このため、サーバー２がリクエストを受信した際、デーモンがすべて処理中である場合は、サーバー２は、クライアント１のファイル詳細情報取得方法を、（イ）から（ア）の方法に切り替える。つまり、ファイルが存在するディレクトリを管理するサーバー２は、自装置が保持しているディレクトリエントリ内の情報であるFH、ファイルタイプのみをクライアント１に返す。クライアント１は、返されたFHから、オープン対象のファイルを管理するサーバー２を特定し、特定したサーバー２へ直接ファイル詳細情報を要求する。オープン対象の各ファイルは、それぞれ各サーバー２に分散配置されているため、このリクエストの送信先もクライアント１毎に分散されることになる。これにより、TATの悪化を軽減することが可能となる。

【００７１】

＜構成＞

図２は、本実施の形態にかかるクライアント１、サーバー２、および、外部記憶装置４の内部構成を示す図である。

【００７２】

クライアント１は、通信部１０１、マウント要求部１０２、メタデータ参照／更新部１０３、サーバー情報表更新部１０４、メタデータ問い合わせ部１０５、パス名解析部１０６、ファイル／ディレクトリ生成要求部１０７、ファイル詳細情報要求部１０８、サーバー情報表１０９、および、メタデータ記憶部１１０を包含する。通信部１０１は、リクエスト作成部１０１－１、および、サーバー特定部１０１－２を包含する。

【００７３】

サーバー２は、通信部２０１、マウント応答部２０２、サーバー選定部２０３、ファイル／ディレクトリ生成要求部２０４、ディレクトリエントリ参照／更新部２０５、ファイル／ディレクトリ生成部２０６、ＦＨ生成部２０７、サーバー内ＦＨ検索部２０８、ファ

イル詳細情報問い合わせ部２０９、Ｉ／Ｏ発行部２１０、サーバー情報表２１１、ファイル詳細情報表２１２、および、メタデータ一時記憶表２１３を包含する。通信部２０１は、リクエスト作成部２０１－１、リクエスト応答部２０１－２、リクエスト受信部２０１－３、リクエスト内容変更部２０１－４、リクエスト処理デーモン数判別／更新部２０１－５、リクエスト処理デーモン数カウンタ２０１－６、リクエスト内容判別部２０１－７、および、サーバー特定部２０１－８を包含する。

【００７４】

外部記憶装置４は、メタデータ記憶装置４０１、および、ファイルデータ記憶装置４０２を包含する。

【００７５】

なお、図２中で各部を結ぶ矢印は、結ばれる両者間の主要な指示／情報の流れを示すものであるが、各部間の指示／情報の流れは、これらに限られるものではない。

【００７６】

図２中の各部は、それぞれ概略次のように動作する。最初に、クライアント１に含まれる各部の概略動作について説明する。

【００７７】

通信部１０１は、送信デーモン、受信デーモンを包含する。そして、通信部１０１は、サーバー２に対してリクエストを送信する際は、例えば、メタデータ問い合わせ部１０５から渡された要求を通信プロトコルにあった形式に変換して、リクエスト作成部１０１－１を用いてリクエストを作成する。また、この際に、通信部１０１は、FHを基に、サーバー特定部１０１－２を用いて宛先のサーバー２のIPアドレスを得る。

【００７８】

リクエスト作成部１０１－１は、TCP/IPなど使用する通信プロトコルに沿った形式のパケットを作成する。

【００７９】

FHにはこれに対応するファイル、ディレクトリを管理するサーバー２のサーバーIDが含まれている。サーバー特定部１０１－２は、FHを基にサーバーIDを抽出し、サーバー情報表１０９を参照して、宛先サーバー２のIPアドレスを得る。

【００８０】

マウント要求部１０２は、クライアント１がファイルシステムを利用可能にするための処理を行う。マウントに際して、あらかじめシステム管理者が、ルートサーバーのIPアドレス若しくはマシン名、及び、マウントポイントを指定しておく必要がある。

【００８１】

mountコマンドなどによりマウント処理が開始されると、クライアント１からルートサーバーにマウント要求が通信部１０１を介して送信される。その応答として、サーバー情報、および、ファイルシステムの先頭のディレクトリのFHがルートサーバーからクライアント１に返される。ここで、サーバー情報は、ファイルシステムを構成している全サーバー２のサーバーIDとそのIPアドレスの対応表、および、各サーバー２配下のデバイス情報を包含する。

【００８２】

サーバー情報は、クライアント１において、サーバー情報更新部１０４によりサーバー情報表１０９に記録され、FHはルートディレクトリのFHとして、メタデータ参照／更新部１０３によりメタデータ記憶部１１０に記録される。

【００８３】

メタデータ参照／更新部１０３は、メタデータ記憶部１１０の参照と更新を行う。

【００８４】

サーバー情報表更新部１０４は、サーバー情報表１０９更新を行う。

【００８５】

メタデータ問い合わせ部１０５は、最初に、パス名解析部１０６により、指定されたパス名中のより上位の名前を１つ抽出する。例えば、マウントポイントが“/mnt”、パス

名が“/mnt/home/user1/file1”である場合、まず“home”が抽出される。メタデータ問い合わせ部１０５は、それをメタデータ参照／更新部１０３により、ルートディレクトリ配下にこの名前がメタデータ記憶部１１０に既に登録されているかどうかを調べる。

【００８６】

記憶されていない場合、メタデータ問い合わせ部１０５は、マウント時にルートサーバーから取得したルートディレクトリのFHとその配下の名前（“home”）の組み合わせをルートサーバーに送信して、送信した名前に対応するFHを要求する。応答として、指定した名前に対応するFHとファイルタイプが返される。メタデータ問い合わせ部１０５は、これをメタデータ参照／更新部１０３により、メタデータ記憶部１１０に登録する。

【００８７】

その後、メタデータ問い合わせ部１０５は、パス名解析部１０６により、パス名中からその１つ下の名前（“user1”）を抽出し、その上位のディレクトリ（“home”）のFHからそれを管理するサーバー２を特定し、上記と同様にそのFHと名前（“user1”）に対応するFHを要求する。メタデータ問い合わせ部１０５は、応答として返された名前に対応するFHとファイルタイプをメタデータ参照／更新部１０３により、メタデータ記憶部１１０に登録する。メタデータ問い合わせ部１０５は、同様に処理を繰り返す。

【００８８】

なお、パス名の末端の名前（“file1”）は、オープンシステムコールから呼ばれている場合、ファイル名であるので、サーバー２からは名前に対応するFHと共にファイル詳細情報が返される（図４中のｃ）。

【００８９】

前述の図４は、メタデータ問い合わせ部１０５が関連モジュールを用いながら、ディレクトリ探索を経て、パス名に対応するファイルのFHとファイル詳細情報を取得する流れを示す。

【００９０】

パス名解析部１０６は、与えられたパス名から文字列操作により、名前を１つ抽出する。

【００９１】

ファイル／ディレクトリ生成要求部１０７は、与えられたパス名の最後の名前で、ファイルまたはディレクトリの作成を該当するサーバー２に要求する。与えられたパス名の最後の名前は、例えば、パス名が“/mnt/home/user1/file1”なら、名前“file1”である。ただし、作成するファイル、ディレクトリの上位ディレクトリのFHがメタデータ記憶部１１０に登録されていない場合は、先にメタデータ問い合わせ部１０５により上位ディレクトリのFHの問い合わせを行う。

【００９２】

メタデータ問い合わせ部１０５の処理において、オープンシステムコールによりファイルをオープンする際、与えられたパス名の末端の名前に対応したFHの問い合わせでファイル詳細情報が返されず、FHとファイルタイプのみが返される場合が有る。すなわち、図４中のｃにおいて、ファイル詳細情報が返されない場合が有る。

【００９３】

ファイル詳細情報要求部１０８は、この場合、このFHに基づいて該当するサーバー２、例えば、図４のサーバー＃４、へ直接ファイル詳細情報を要求し、それをメタデータ参照／更新部１０３によりメタデータ記憶部１１０に登録する。

【００９４】

サーバー情報表１０９は、ファイルシステムを構成している全サーバー２のサーバーIDとそのIPアドレスの対応表と各サーバー２配下のデバイス情報をクライアント１に記憶するメモリ領域である。

【００９５】

メタデータ記憶部１１０は、メタデータをクライアント１にキャッシュするためのメモリ領域である。メタデータ記憶部１１０は、オープンシステムコールなどで指定されるパ

ス名中の各ディレクトリ名、またはファイル名をファイルディレクトリの上位から順に記憶し、それに対応するディレクトリ、ファイルのFHとファイルタイプを記憶する。さらに、ファイルの場合は、メタデータ記憶部１１０はファイル詳細情報も記憶する。

【００９６】

クライアント装置１内の各部の機能分担は、適宜変更して実装しても良い。例えば、メタデータ問い合わせ部１０５は、パス名解析部１０６、および、メタデータ参照／更新部１０３の機能を包含しても良い。

【００９７】

次に、サーバー２に含まれる各部の概略動作について説明する。

【００９８】

通信部２０１は、受信デーモンと送信デーモンとして動作し、リクエストの受信とその応答及びサーバー２間通信の際のリクエストの作成を行う。

【００９９】

さらに、通信部２０１は、動作中／待ち状態のリクエスト処理デーモンの数を管理している。クライアント１からファイル詳細情報を要求された際、待ち状態のリクエスト処理デーモン数が１個以下の場合、通信部２０１はファイル詳細情報を返却せずに、当該ファイルのFHとファイルタイプを返却するようにリクエストの内容を変更する。なお、通信部２０１は、動作中／待ち状態のリクエスト処理デーモンの数を管理する代わりに、ファイル詳細情報取得のため入出力について、現在進行中の入出力数と発行可能な最大多重度と現在進行中の入出力数との差分、を管理しても良い。

【０１００】

リクエスト作成部２０１－１は、サーバー２間通信の際に宛先のサーバー２を指定し、必要なパラメータの設定などをしてリクエストを作成し、送信デーモンに渡す。

【０１０１】

リクエスト応答部２０１－２は、クライアント１、またはサーバー２間通信でのリクエストに対する応答に必要なデータを設定し、送信デーモンに渡す。

【０１０２】

リクエスト受信部２０１－３は、受信デーモンが受信したリクエストをリクエスト処理デーモンに渡す。なお、この際、リクエストがFH、ファイルタイプ、ファイル詳細情報を要求し、かつ、待ち状態のリクエスト処理デーモン数が１個以下であれば、リクエスト内容変更部２０１－４によりリクエストをFHとファイルタイプのみのリクエストに置き換える。

【０１０３】

リクエスト内容変更部２０１－４は、リクエスト内容の書き換えを行う。

【０１０４】

リクエスト処理デーモン数判別／更新部２０１－５は、処理中のリクエスト処理デーモンの数をリクエスト処理デーモン数カウンタ２０１－６に保持する。すなわち、リクエスト処理デーモン数判別／更新部２０１－５は、リクエスト処理デーモンの処理の開始／終了時に、リクエスト処理デーモン数カウンタ２０１－６のカウント値に１を加算／減算する。また、リクエスト処理デーモン数判別／更新部２０１－５は、受信デーモンがファイル詳細情報のリクエストを受信した際にリクエスト処理デーモン数カウンタ２０１－６のカウント値を参照する。

【０１０５】

リクエスト処理デーモン数カウンタ２０１－６は、処理中のリクエスト処理デーモンの数を持つカウンタであり、メモリ上にその領域を確保される。

【０１０６】

リクエスト内容判別部２０１－７は、クライアント１または他のサーバー２から受信したリクエストの種別を調べる。

【０１０７】

FHには、これに対応するファイル、ディレクトリを管理するサーバー２のサーバーIDが

含まれている。サーバー特定部２０１－８は、FHからサーバーIDを抽出し、サーバー情報表２１１を参照して宛先サーバー２のIPアドレスを特定する。

【０１０８】

マウント応答部２０２は、クライアント１から、ファイルシステムのマウントを要求するリクエストを受信する。そして、マウント応答部２０２は、同クライアント１に対し、サーバー情報表２１１に記録されている該ファイルシステムのルートディレクトリのFHと、該ファイルシステムを構成するサーバー２のサーバーIDとIPアドレスを返却する。さらに、マウント応答部２０２は、各サーバー２配下のデバイスに関する情報を返す。なお、同リクエストを受信したサーバー２がルートサーバーではない場合は、マウント応答部２０２は、同クライアント１へエラーを返す。

【０１０９】

サーバー選定部２０３は、ファイル、ディレクトリの生成を要求された際、サーバー情報表２１１を参照して、ファイル、ディレクトリの名前に依存しない方法、例えば、ラウンドロビンや一様乱数に基づき、サーバー２を一つ選定する。

【０１１０】

ファイル／ディレクトリ生成要求部２０４は、サーバー選定部２０３が選定したサーバー２に対してファイル、ディレクトリの生成を要求する。

【０１１１】

ディレクトリエントリ参照／更新部２０５は、ファイル、ディレクトリを生成した際、その上位ディレクトリを管理するサーバー２において、メタデータ記憶装置４０１内の上位ディレクトリのディレクトリエントリに、生成されたファイル、ディレクトリのエントリを追加する。また、ディレクトリエントリ参照／更新部２０５は、参照要求を受けて、ディレクトリエントリ内の指定された名前に一致するエントリ、あるいはディレクトリエントリの各エントリのFHとファイルタイプを返す。

【０１１２】

ファイル／ディレクトリ生成部２０６は、ファイル、ディレクトリの生成要求をサーバー２間通信により受け取って、メタデータ記憶装置４０１内に該当するエントリが有るかどうかをチェックする。無ければ、ファイル／ディレクトリ生成部２０６は、ＦＨ生成部２０７によりFHを新たに生成し、ファイルの場合はファイルの構成情報や、ファイルタイプと共に新たなエントリを記録する。

【０１１３】

ＦＨ生成部２０７は、ファイル、ディレクトリを生成する際に起動されて、当該サーバー２のサーバーIDとサーバー２内でユニークな数値を組み合わせて、ファイルシステム全体でユニークなFHを生成する。

【０１１４】

サーバー内ＦＨ検索部２０８は、Ｉ／Ｏ発行部２１０を介して、メタデータ記憶装置４０１内を検索し、指定されたFHを検索する。

【０１１５】

ファイル詳細情報問い合わせ部２０９は、クライアント１がオープンしようとする、自装置が管理するディレクトリ配下のファイルのファイル詳細情報を、当該ファイルを管理するサーバー２に問い合わせる。

【０１１６】

Ｉ／Ｏ発行部２１０は、メタデータ記憶装置４０１、及びファイルデータ記憶装置４０２に対して、指定されたメモリ上のデータを書き込む、あるいは、指定されたデータをメモリ上に読み込む。この際、Ｉ／Ｏ発行部２１０は、メタデータの場合は同時にメタデータ一時記憶表２１３にも登録し、さらに読み込みに際してはメタデータ一時記憶表２１３に当該メタデータがあればその値を返す。

【０１１７】

サーバー情報表２１１は、ファイルシステムを構成するサーバー２のサーバーIDとIPアドレスの対応、及びルートディレクトリのFHを格納するメモリ上に確保された領域である

。また、メタデータ記憶装置４０１には、サーバー情報表２１１と同じデータが格納されており、サーバー２を起動させた際にサーバー情報表２１１に読み込まれる。

【０１１８】

ファイル詳細情報表２１２は、ファイル詳細情報を格納するためのメモリ上に確保された領域である。

【０１１９】

メタデータ一時記憶表２１３は、メタデータをキャッシュするためのメモリ上に確保された領域である。

【０１２０】

サーバー装置１内の各部の機能分担は、適宜変更して実装しても良い。例えば、ディレクトリエントリ参照／更新部２０５は、サーバー内ＦＨ検索部２０８の機能を包含しても良いし、ファイル／ディレクトリ生成部２０６は、ＦＨ生成部２０７の機能を包含しても良い。

【０１２１】

最後に、外部記憶装置４に含まれる各部が記憶する情報について説明する。

【０１２２】

メタデータ記憶装置４０１は、ディレクトリエントリ、ファイル、ディレクトリの詳細情報、及びサーバー情報表２１１と同じサーバー２の構成情報などのファイルシステムの管理情報、いわゆるメタデータを記憶する記憶媒体である。

【０１２３】

ファイルデータ記憶装置４０２は、ファイルのデータを記憶する記憶媒体である。なお、ファイルデータ記憶装置４０２とメタデータ記憶装置４０１は、物理的に別の媒体でも同一の媒体でも良い。

【０１２４】

なお、クライアント１やサーバー２は、ファイル、ディレクトリの削除、属性情報の取得などのファイルシステムが通常持っている他の機能も備えているが、他の処理から容易に想到し得るため、ここでは説明を割愛する。

【０１２５】

ここで、クライアント１における、通信部１０１、マウント要求部１０２、メタデータ参照／更新部１０３、サーバー情報表更新部１０４、メタデータ問い合わせ部１０５、パス名解析部１０６、ファイル／ディレクトリ生成要求部１０７、および、ファイル詳細情報要求部１０８は、論理回路で構成される。各部は、適宜、クライアント１が備える図示されない半導体メモリにアクセスする。サーバー情報表１０９、およびメタデータ記憶部１１０は、クライアント１が備える図示されない半導体メモリ上に設けられる。

【０１２６】

また、サーバー２における、通信部２０１、マウント応答部２０２、サーバー選定部２０３、ファイル／ディレクトリ生成要求部２０４、ディレクトリエントリ参照／更新部２０５、ファイル／ディレクトリ生成部２０６、ＦＨ生成部２０７、サーバー内ＦＨ検索部２０８、ファイル詳細情報問い合わせ部２０９、および、Ｉ／Ｏ発行部２１０は、論理回路で構成される。各部は、適宜、サーバー２が備える図示されない半導体メモリにアクセスする。サーバー情報表２１１、ファイル詳細情報表２１２、およびメタデータ一時記憶表２１３は、サーバー２が備える図示されない半導体メモリに記憶される。

【０１２７】

外部記憶装置４は、例えば、ＨＤＤ（Hard-Disk Drive）やＳＳＤ(Solid State Drive)である。

【０１２８】

また、クライアント１、およびサーバー２は、それぞれ、プログラム４３を備えるコンピュータ装置４０で実現することも出来る。

【０１２９】

図１６は、コンピュータ装置４０の構成図である。コンピュータ装置４０は、バス４５

で相互に接続されたプロセッサ４１、主記憶部４２、外部記憶装置４４を備える。ここで、例えば、主記憶部４２は半導体記憶装置、外部記憶装置４４はＨＤＤやＳＤＤである。主記憶部４２はプログラム４３を記憶している。

【０１３０】

クライアント１が記憶しているプログラム４３は、クライアント１として用いられるコンピュータ装置４０において、プロセッサ４１で実行されることにより、プロセッサ４１を通信部１０１、マウント要求部１０２、メタデータ参照／更新部１０３、サーバー情報表更新部１０４、メタデータ問い合わせ部１０５、パス名解析部１０６、ファイル／ディレクトリ生成要求部１０７、および、ファイル詳細情報要求部１０８として機能させる。主記憶部４２は、サーバー情報表１０９、および、メタデータ記憶部１１０を格納する。

【０１３１】

さらに、サーバー２が記憶しているプログラム４３は、サーバー２として用いられるコンピュータ装置４０において、プロセッサ４１で実行されることにより、プロセッサ４１を通信部２０１、マウント応答部２０２、サーバー選定部２０３、ファイル／ディレクトリ生成要求部２０４、ディレクトリエントリ参照／更新部２０５、ファイル／ディレクトリ生成部２０６、ＦＨ生成部２０７、サーバー内ＦＨ検索部２０８、ファイル詳細情報問い合わせ部２０９、および、Ｉ／Ｏ発行部２１０として機能させる。主記憶部４２は、サーバー情報表２１１、ファイル詳細情報表２１２、およびメタデータ一時記憶表２１３を格納する。

【０１３２】

サーバー２において、外部記憶装置４４または主記憶部４２は、メタデータ記憶装置４０１、および、ファイルデータ記憶装置４０２として機能する。

【０１３３】

＜動作＞

次に、本発明の実施例の動作について詳細に説明する。なお、説明にあたって、クライアント１、サーバー２は、コンピュータ装置４０を用いて実現されていると仮定する。クライアント１、サーバー２のオペレーティングシステムは、UNIX（登録商標）や Linux（登録商標）であると仮定する。また、クライアント１、サーバー２は、EtherNet（登録商標）、InfiniBand（登録商標）など一般に利用可能なネットワークインターフェースを持ち、相互結合ネットワーク３を介して接続される。これにより、クライアント１は任意のサーバー２と通信可能であり、サーバー２はそのサーバー２以外の任意のサーバー２と通信可能である。

【０１３４】

１.ファイルのオープン処理

図７乃至図１１は、ファイルオープン処理の動作例のフローチャートである。図７及び図８はクライアント１側、図９乃至図１１はサーバー２側の動作フローチャートである。なお、これらのフローチャートは、クライアント１が、ファイルディレクトリを検索して、オープン対象のファイルのFHとファイル詳細情報を取得するまでの流れを示しており、その後のオープン処理については割愛されている。その後のオープン処理は、公知技術で実現できる。

【０１３５】

１．ａ.ファイルのオープン処理（クライアント１側）

クライアント１の動作は、メタデータ問い合わせ部１０５が中心となる。クライアント１上で動作するＡＰから、分散ファイルシステム５内のファイルに対してオープンシステムコールが呼び出されると、オペレーシングシステムのカーネルからメタデータ問い合わせ部１０５が起動されて、図７の処理が開始される。

【０１３６】

最初に、メタデータ問い合わせ部１０５は、パス名解析部１０６により、指定されたパス名中の上位の名前を１つ抽出する（ステップ１）。パス名の末端、すなわち文字列としての終端、もしくは、作成対象の名前を検出した場合（ステップ２でＹＥＳ）、メタデー

タ問い合わせ部１０５は、図８の処理に進む。

【０１３７】

パス名の末端、もしくは作成対象の名前以外を検出した場合（ステップ２のＮＯ）、メタデータ問い合わせ部１０５は、抽出した名前がメタデータ記憶部１１０に既に登録されているかどうかを、メタデータ参照／更新部１０３により調べる（ステップ３）。登録されている場合は（ステップ４でＹＥＳ）、メタデータ問い合わせ部１０５はステップ１に戻る。

【０１３８】

登録されていない場合は（ステップ４でＮＯ）、メタデータ問い合わせ部１０５は、まず、上位ディレクトリのFHから問い合わせ先のサーバー２を特定し（ステップ５）、該サーバー２へ上位ディレクトリのFH、問い合わせ対象の名前を送信する（ステップ６）。メタデータ問い合わせ部１０５は、問い合わせ先のサーバー２からFHとファイルタイプを受信すると、それらを名前と共にメタデータ参照／更新部１０３によりメタデータ記憶部１１０に登録する（ステップ７）。

【０１３９】

図７のステップ２でＹＥＳとなり、図８の処理に進んだ場合、メタデータ問い合わせ部１０５は、ファイル詳細情報要求部１０８により、オープン対象のファイルの名前のファイル詳細情報、及びFHを、そのファイルの上位ディレクトリを管理するサーバー２へ問い合わせる。

【０１４０】

図８において、メタデータ問い合わせ部１０５は、メタデータ参照／更新部１０３により、オープン対象のファイルの名前が既にメタデータ記憶部１１０に登録されているかを調べ（ステップ１１）、既に登録されている場合（ステップ１２でＹＥＳ）、動作を終了する。登録されていない場合（ステップ１２でＮＯ）、メタデータ問い合わせ部１０５は、上位ディレクトリのFHより問い合わせ先のサーバー２を特定し（ステップ１３）、オープン対象ファイルの名前に相当するFH、及びファイル詳細情報を含む問い合わせを該サーバー２へ送信する（ステップ１４）。なお、この問い合わせは、１）上位ディレクトリを管理するサーバー２が保持するオープン対象ファイルの名前に相当するFH、及び、ファイルタイプの問い合わせと、２）オープン対象ファイルを管理するサーバー２が保持するファイル詳細情報の問い合わせの２つの問い合わせをまとめた問い合わせである。

【０１４１】

FH、ファイルタイプと共にファイル詳細情報を受信した場合（ステップ１５でＹＥＳ）、メタデータ問い合わせ部１０５は、ステップ９の実施後動作を終了する。ファイル詳細情報は返されず、FHとファイルタイプが返された場合（ステップ１５でＮＯ）、メタデータ問い合わせ部１０５は、返されたFHとファイルタイプをメタデータ参照／更新部１０３により一旦メタデータ記憶部１１０に登録する（ステップ１６）。さらに、メタデータ問い合わせ部１０５は、ファイル詳細情報要求部１０８により、返されたFHを基にオープン対象ファイルを管理するサーバー２を特定し（ステップ１７）、該サーバー２へFHを指定してオープン対象のファイルのファイル詳細情報を問い合わせる（ステップ１８）。その後、メタデータ問い合わせ部１０５は、ファイル詳細情報要求部１０８が受信した該ファイル詳細情報を、FH、名前と共にメタデータ参照／更新部１０３によりメタデータ記憶部１１０に登録する（ステップ１９）。

【０１４２】

メタデータ問い合わせ部１０５による図８の動作が終了すると、クライアント１上では、メタデータ記憶部１１０上のファイルの名前、FH、ファイル詳細情報を用いて、オープンシステムコールの処理が続行される。

【０１４３】

１．ｂ.ファイルのオープン処理（サーバー２側）

図９乃至１１は、図７のステップ６、図８のステップ１４、およびステップ１８においてサーバー２への問い合わせをクライアント１が行った際のサーバー２側の処理の例を示

す。クライアント１からの当合わせはサーバー２において、通信部２０１により受信される。

【０１４４】

図９において、通信部２０１は、受信されたクライアント１からのリクエスト（問い合わせ）を、リクエスト内容判別部２０１－７により何のリクエストかを調べる。そのリクエストがFHで示されたディレクトリ配下のファイルのファイル詳細情報とFH、ファイルタイプを要求するものであった場合（ステップ２１でＹＥＳ）、通信部２０１は、リクエスト処理デーモン数判別／更新部２０１－５によりリクエスト処理デーモン数カウンタ２０１－６を参照し、待ち状態の同デーモンの数を確認する（ステップ２２）。

【０１４５】

同デーモン数が２個以上である場合（ステップ２３でＮＯ）、通信部２０１は、図１１のファイルのFH、ファイルタイプ、及び、ファイル詳細情報取得処理に進む。１個以下である場合（ステップ２３でＹＥＳ）、通信部２０１は、リクエスト内容変更部２０１－４により、リクエストの内容を該ファイルのFHとファイルタイプの問い合わせのみに書き換える（ステップ２４）。次に、通信部２０１は、そのリクエストをリクエスト処理デーモンに渡し（ステップ２５）、図１０のファイルのFHとファイルタイプ取得処理に進む。

【０１４６】

なお、受信したリクエストがFHで示されたディレクトリ配下のファイルのFHとファイルタイプを要求するものであった場合（ステップ２１でＮＯ）、通信部２０１は、そのリクエストをリクエスト処理デーモンに渡し（ステップ２５）、図１０のファイルのFHとファイルタイプ取得処理に進む。

【０１４７】

図９でステップ２５を実施した場合は、図１０において、ディレクトリエントリ参照／更新部２０５は、サーバー内ＦＨ検索部２０８により、受信したFHをキーとして該当するディレクトリを特定する（ステップ３１）。ディレクトリエントリ参照／更新部２０５は、さらにそのディレクトリから、名前をキーとして該当するエントリを特定することで、名前に対応するFHとファイルタイプを抽出し、要求元のクライアント１へ送信し（ステップ３２）、処理を終了する。

【０１４８】

図９で、リクエスト処理デーモン数が２個以上である場合（ステップ２３でＮＯ）、図１１において、ディレクトリエントリ参照／更新部２０５は、サーバー内ＦＨ検索部２０８により、受信したFHをキーとして該当するディレクトリを特定する（ステップ４１）。ディレクトリエントリ参照／更新部２０５は、さらにそのディレクトリから、名前をキーとして該当するエントリを特定することで、名前に対応するFHとファイルタイプを抽出する（ステップ４２）。

【０１４９】

続いてディレクトリエントリ参照／更新部２０５は、ファイル詳細情報表２１２を検索し、該ファイルのファイル詳細情報が登録されているかどうかを確認する（ステップ４３）。

【０１５０】

登録されていた場合（ステップ４３でＹＥＳ）、ディレクトリエントリ参照／更新部２０５は、FHとファイルタイプと共に該ファイル詳細情報を要求元のクライアント１へ返却し（ステップ４４）、処理を終了する。登録されていない場合（ステップ４３でＮＯ）、該ファイルのファイル詳細情報をファイル詳細情報問い合わせ部２０９が、該ファイルを管理するサーバー２へ問い合わせる（ステップ４５）。このとき、サーバー特定部２０１－８が、該ファイルのFHからサーバー２を特定する。

【０１５１】

問い合わせ先の該サーバー２、すなわち、該ファイルを管理するサーバー２からファイル詳細情報を受信すると、ディレクトリエントリ参照／更新部２０５は、これをファイル詳細情報表２１２へ登録し（ステップ４６）、要求元のクライアント１へFH、ファイルタ

イプと共に該ファイル詳細情報を送信し（ステップ４７）、処理を終了する。

【０１５２】

なお、問い合わせ先のサーバー２では、ディレクトリエントリ参照／更新部２０５が問い合わせ元のサーバー２（該ファイルが存在するディレクトリを管理するサーバー２）が送信したFH受信する。そして、ディレクトリエントリ参照／更新部２０５が、サーバー内ＦＨ検索部２０８により、メタデータ一時記憶表２１３またはメタデータ記憶装置４０１を検索し、受信したFHに対応するファイル詳細情報を読み出して、問い合わせ元のサーバー２へ返却する。

【０１５３】

２．ファイル、ディレクトリの生成処理

図１２および図１３は、サーバー２におけるファイル、ディレクトリ生成処理の動作例のフローチャートである。

【０１５４】

クライアント１上で動作するＡＰから、create/openシステムコール、または、mkdirシステムコールが呼び出されると、オペレーシングシステムのカーネルからメタデータ問い合わせ部１０５が起動されて、当該システムコールにおいて指定されたパス名を基に、生成するファイル、ディレクトリを作成するディレクトリのFHを取得する。その後、ファイル／ディレクトリ生成要求部１０７が起動され、該当するサーバー２へその生成を要求する。

【０１５５】

図１２は、ファイル、ディレクトリを作成するディレクトリを管理するサーバー２側、図１３は、ファイル、ディレクトリを生成するサーバー２側の動作フローチャートである。ここで、ファイル、ディレクトリを作成するディレクトリとは、新たに生成されたファイル、ディレクトリの上位ディレクトリとなるディレクトリである。

【０１５６】

２．ａ.ファイル、ディレクトリの生成処理（上位ディレクトリを管理するサーバー２側）

図１２において、クライアント１から生成要求を受信すると、ファイル／ディレクトリ生成要求部２０４が起動される。ファイル／ディレクトリ生成要求部２０４は、サーバー選定部２０３により新たに生成するファイル、ディレクトリを管理するサーバー２を選定し（ステップ５１）、選定したサーバー２へファイル、ディレクトリの作成要求としてファイルタイプと名前、ファイル、ディレクトリのパーミッションに関する情報を送信する（ステップ５２）。

【０１５７】

ファイル／ディレクトリ生成要求部２０４は、作成要求先のサーバー２から、生成したファイル、ディレクトリのFHを、ファイルの場合はファイル詳細情報も、受信する。ディレクトリエントリ参照／更新部２０５は、受信したFHとファイルタイプ及び名前をＩ／Ｏ発行部２１０を経てメタデータ一時記憶表２１３、及びメタデータ記憶装置４０１へ新たなディレクトリエントリとして登録する（ステップ５３）。

【０１５８】

また、ファイルを生成した場合は（ステップ５４でＹＥＳ）、ディレクトリエントリ参照／更新部２０５は、さらに、受信したファイル詳細情報をファイル詳細情報表２１２に登録する（ステップ５５）。この後、通信部２０１が、受信したファイル、ディレクトリのFHを、ファイルを生成した場合はそのファイルのファイル詳細情報も、要求元のクライアント１へ送信して（ステップ５６）、処理を終了する。

【０１５９】

２．ｂ.ファイル、ディレクトリの生成処理（作成するサーバー２側）

図１２のステップ５２でファイル、ディレクトリの生成要求を受信したサーバー２は、図１３のフローチャートのように動作する。

【０１６０】

生成要求を受信したファイル／ディレクトリ生成部２０６は、ＦＨ生成部２０７により自サーバー２のサーバーIDと自サーバー２内で一意な番号を組み合わせることで、ファイルシステム内で一意なFHを生成する（ステップ６１）。次に、ファイル／ディレクトリ生成部２０６は、このFHと共にファイル、ディレクトリの詳細情報としてファイルのパーミッション、生成時刻などをＩ／Ｏ発行部２１０を経て、メタデータ一時記憶表２１３及びメタデータ記憶装置４０１に登録する（ステップ６２）。

【０１６１】

この後、ファイル／ディレクトリ生成部２０６は、要求元のサーバー２へこれら情報を返して（ステップ６３）、処理を終了する。

【０１６２】

３．マウント処理

図１４および図１５は、マウント処理の動作例のフローチャートである。図１４はクライアント１側、図１５はサーバー２側の動作フローチャートである。

【０１６３】

３．ａ.マウント処理（クライアント１側）

図１４において、クライアント１のユーザがmountコマンドを入力、または、クライアント１上のＡＰがマウントシステムコールを呼び出すと、マウント要求部１０２がオペレーティングシステムから起動される。mountコマンド、または、mountシステムコールは、ルートサーバーであるサーバー２を指定しており、マウント要求部１０２は、指定されているサーバー２へマウント要求を送信する（ステップ７１）。

【０１６４】

マウント要求部１０２は、該サーバー２から、ルートディレクトリのFH、ファイルシステムを構成する全サーバー２のサーバーIDとIPアドレスの対応表を含む応答を受信し（ステップ７２）、同対応表を、サーバー情報更新部１０４によりサーバー情報表１０９に登録する（ステップ７３）。マウント要求部１０２は、ルートディレクトリのFHをメタデータ参照／更新部１０３によりメタデータ記憶部１１０に登録して（ステップ７４）、処理を終了する。

【０１６５】

３．ｂ.マウント処理（サーバ２側）

図１５において、マウント応答部２０２は、サーバー情報表２１１を参照し、ファイルシステムを構成する全サーバー２のサーバーIDとIPアドレスの対応表、及びルートディレクトリのFHの値を読み出し（ステップ８１）、要求元のクライアント１へ送信して（ステップ８２）、処理を終了する。

【０１６６】

なお、サーバー２での各リクエストの処理において、リクエスト処理デーモン数判別／更新部２０１－５は、リクエスト処理デーモン数カウンタ２０１－６の値をリクエストの処理開始時に１増加させ、リクエストの処理終了時に１減少させる。

【０１６７】

＜効果＞

本実施の形態にかかるファイル分散システム５の効果は以下の通りである。

【０１６８】

第１の効果は、多数のクライアント１での同一ファイルへのオープン処理により、そのリクエストが対象ファイルを管理するサーバー２とその上位ディレクトリを管理するサーバー２に集中した場合において、サーバー２間の通信回数を削減できることである。

【０１６９】

その理由は、上位ディレクトリを管理するサーバー２が、当該ファイルの詳細情報を、ファイルを管理するサーバー２から最初に取得した際に、ファイル詳細情報表２１２に保持するからである。このため、２度目以降のリクエストにおいては、上位ディレクトリを管理するサーバー２とファイルを管理するサーバー２との間の通信が不要となる。

【０１７０】

第２の効果は、多数のクライアント１での同一ディレクトリ配下の異なるファイルへのオープン処理により、そのリクエストが当該ディレクトリを管理するサーバー２に集中した場合において、当該サーバー２がボトルネックになるのを防止できることである。

【０１７１】

その理由は、当該ディレクトリを管理するサーバー２のI/O多重度が高くなり、デーモンがすべて使用中になった場合、該サーバー２がディレクトリエントリ内に保持しているFH、及びファイルタイプのみをクライアント１へ返すからである。その後そのFHを基にクライアント１が、オープン対象のファイルを管理するサーバー２を特定し、そのサーバー２に問い合わせを行う。オープン対象の各ファイルは、それぞれ各サーバー２に分散配置されているため、このリクエストの送信先もクライアント１毎に分散されることになる。このため、アクセスが集中するディレクトリを管理するサーバー２のデーモンがすべて使用中であったとしても待ちが発生せず、クライアント１に対するTATが悪化しない。

【０１７２】

第３の効果は、ファイルへのread/write処理だけではなく、ファイルのオープン処理などのメタデータアクセス処理も複数のサーバー２で分散処理できることである。つまり、多数のクライアント１が同一ファイルシステムにアクセスする場合でも、read/write処理、メタデータ処理の両面において負荷分散できる。

【０１７３】

その理由は、ファイルのデータだけではなく、ファイル／ディレクトリの詳細情報やディレクトリエントリ等のメタデータもファイル、ディレクトリ毎に複数のサーバー２に分散して配置することが可能だからである。

【０１７４】

第４の効果は、ファイル、ディレクトリの生成の際に行われる、生成対象ファイル、ディレクトリを管理するサーバー２の選定において、システム全体として管理する情報の更新や排他制御により、システム全体のスループットを妨げないことである。

【０１７５】

その理由は、その上位ディレクトリを管理するサーバー２が、生成対象ファイル、ディレクトリを管理するサーバー２を、サーバー情報表２１１から選定するからである。このため、サーバー２の選定に際して、システム全体として管理する情報の更新や排他制御が不要となる。

【０１７６】

第５の効果は、同一ディレクトリ配下に多数のファイル、ディレクトリを生成しても、特定のサーバー２配下にこれらファイルデータ、メタデータが偏って配置されることはなく、容量、負荷の両面において適正に分散させることが可能なことである。

【０１７７】

その理由は、上位ディレクトリを管理するサーバー２が、新たに生成するファイル、ディレクトリを管理するサーバー２をサーバー情報表２１１から選定する際、名前に依存しない、例えば、ラウンドロビンや一様乱数などに基づく方法を取るからである。

【０１７８】

第６の効果は、クライアント１が、２度目以降の同一ディレクトリ配下へのファイルのオープン処理などメタデータアクセスを伴う処理において、サーバー２との通信回数を削減できることである。

【０１７９】

その理由は、ファイルを管理するサーバー２を特定する際、クライアント１上にメタデータをキャッシュするからである。例えば、オープン処理で同じディレクトリ配下の複数のファイルを同一クライアント１が続けてアクセスすることは、決して少なくない。メタデータのキャッシュにより、ファイルアクセスの都度、ルートサーバーからファイルを管理するサーバー２を辿る処理は不要になる。このため、処理効率が大幅に向上する。

【０１８０】

第７の効果は、名前に関する問題が軽減され、エンドユーザが通常行うファイル名変更

やハードリンク利用の操作の処理が煩雑にならないことである。

【０１８１】

その理由は、ファイル名を変更する場合は、その上位ディレクトリのディレクトリエントリ内の名前を書き換えるだけでよいからである。また、ハードリンクは、例えばディレクトリエントリ内に名前は異なるが同一のFH、ファイルタイプを持つエントリを作り、対象となるファイルを管理するサーバー２で該ファイルのファイル詳細情報のリンク数をカウントアップすることで実現可能となる。このため、本実施の形態にかかるファイル分散システム５は、特許文献１、２、または、非特許文献１のシステム等において発生するような問題を生じない。

【０１８２】

以上、実施形態を参照して本願発明を説明したが、本願発明は上記実施形態に限定されるものではない。本願発明の構成や詳細には、本願発明のスコープ内で当業者が理解し得る様々な変更をすることができる。

【産業上の利用可能性】

【０１８３】

本発明は、HPC (High Performance Computing) やビッグデータ解析のような分野において利用できる。特に、単一のノード、単一のCPUでは現実的な時間での計算や解析が不可能な多量のデータや計算量の処理を、多数のノードに分割して行うような並列計算機システムまたは分散処理基盤におけるデータストアに利用できる。

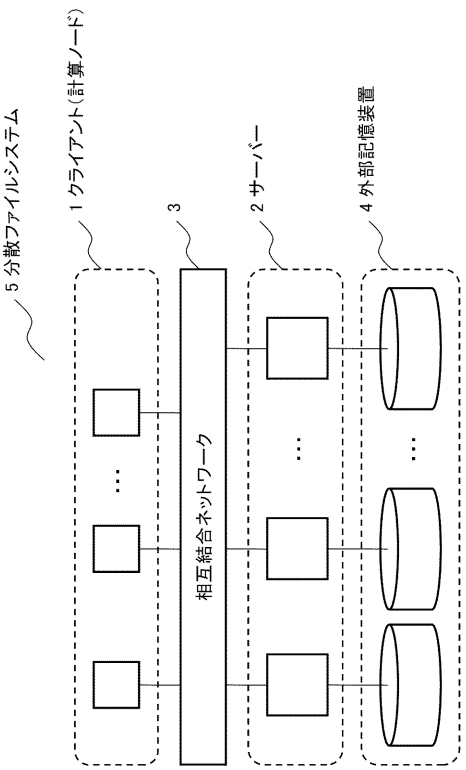
【符号の説明】

【０１８４】

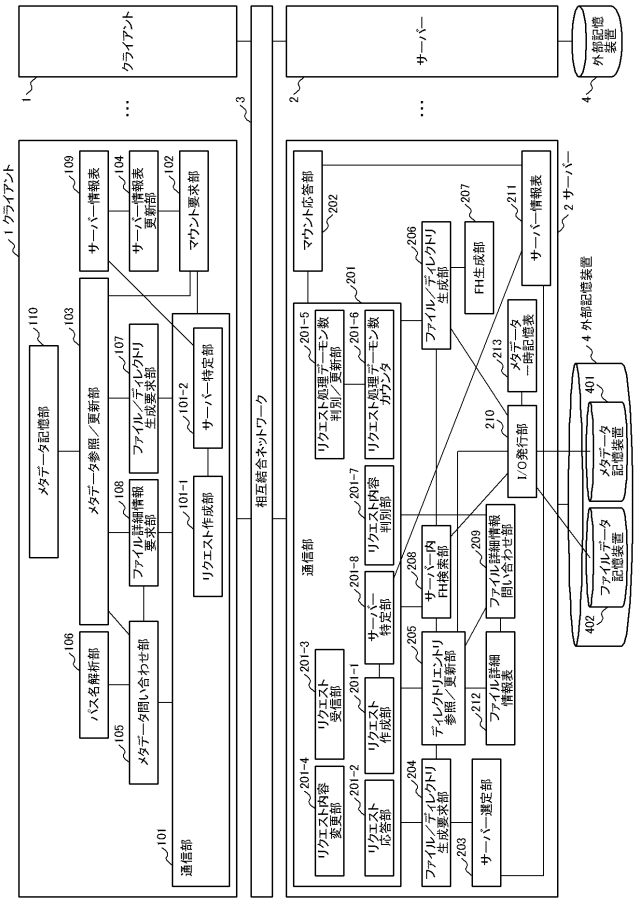
１　クライアント
１　クライアント装置
２　サーバー
２　サーバー装置
３　相互結合ネットワーク
４　外部記憶装置
５　分散ファイルシステム
４０　コンピュータ装置
４１　プロセッサ
４２　主記憶部
４３　プログラム
４４　外部記憶装置
４５　バス
１０１　通信部
１０１－１　リクエスト作成部
１０１－２　サーバー特定部
１０２　マウント要求部
１０３　メタデータ参照／更新部
１０４　サーバー情報表更新部
１０５　メタデータ問い合わせ部
１０６　パス名解析部
１０７　ファイル／ディレクトリ生成要求部
１０８　ファイル詳細情報要求部
１０９　サーバー情報表
１１０　メタデータ記憶部
２０１　通信部
２０１－１　リクエスト作成部
２０１－２　リクエスト応答部
２０１－３　リクエスト受信部

２０１－４　　リクエスト内容変更部
２０１－５　　リクエスト処理デーモン数判別／更新部
２０１－６　　リクエスト処理デーモン数カウンタ
２０１－７　　リクエスト内容判別部
２０１－８　　サーバー特定部
２０２　　マウント応答部
２０３　　サーバー選定部
２０４　　ファイル／ディレクトリ生成要求部
２０５　　ディレクトリエントリ参照／更新部
２０６　　ファイル／ディレクトリ生成部
２０７　　ＦＨ生成部
２０８　　サーバー内ＦＨ検索部
２０９　　ファイル詳細情報問い合わせ部
２１０　　Ｉ／Ｏ発行部
２１１　　サーバー情報表
２１２　　ファイル詳細情報表
２１３　　メタデータ一時記憶表
４０１　　メタデータ記憶装置
４０２　　ファイルデータ記憶装置

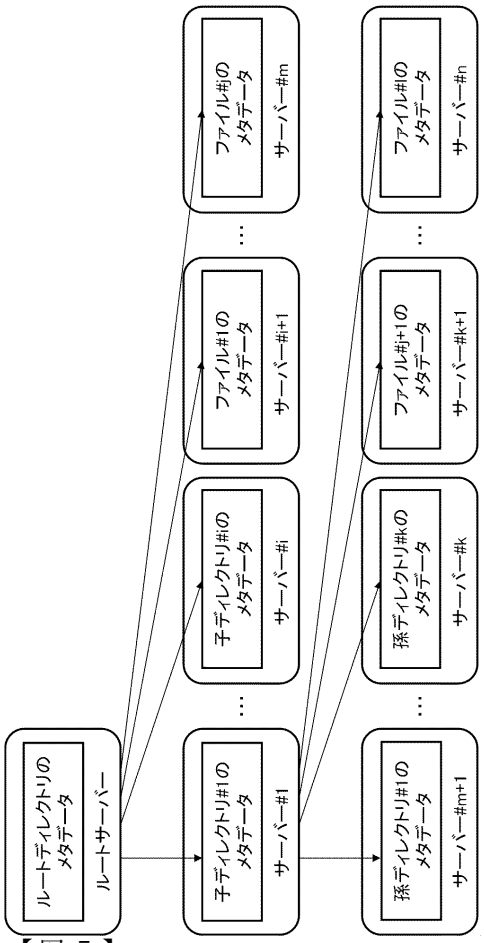
【図１】



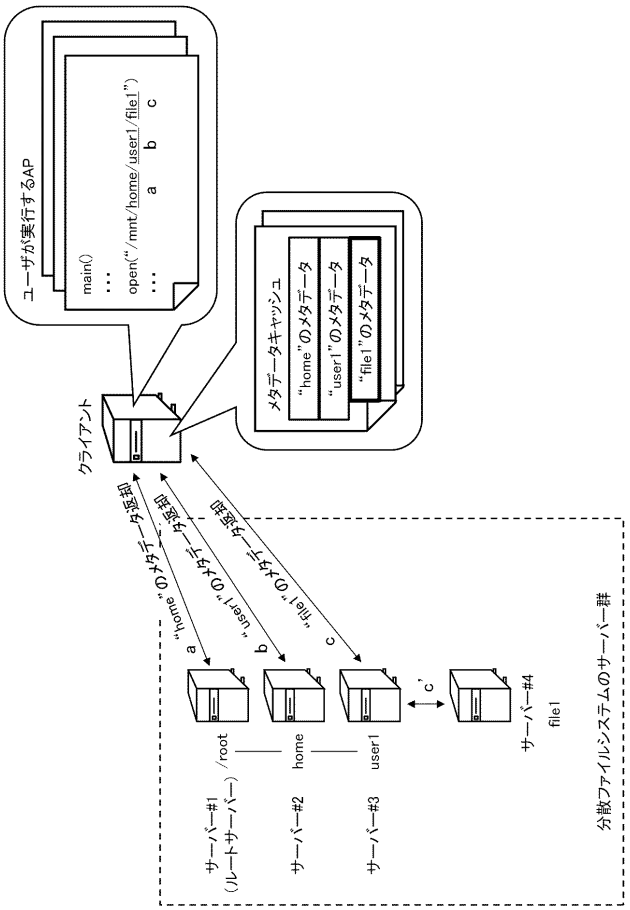
【図２】



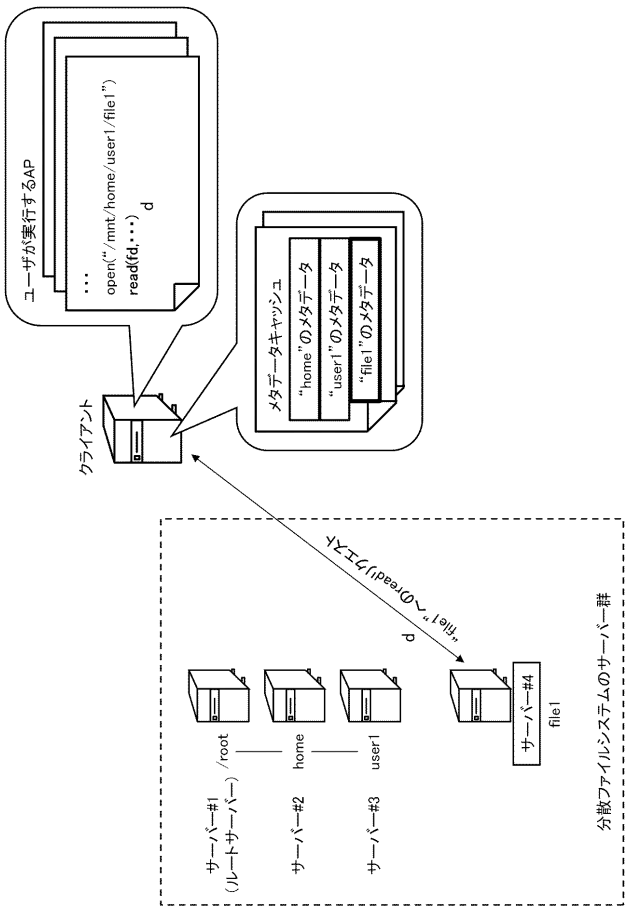
【図３】


ルートディレクトリのメタデータ
ルートサーバー
子ディレクトリ#1のメタデータ
サーバー#1
孫ディレクトリ#1のメタデータ
サーバー#n+1
子ディレクトリ#iのメタデータ
サーバー#i
孫ディレクトリ#kのメタデータ
サーバー#k
ファイル#1のメタデータ
サーバー#i+1
ファイル#j+1のメタデータ
サーバー#k+1
ファイル#jのメタデータ
サーバー#m
ファイル#lのメタデータ
サーバー#n

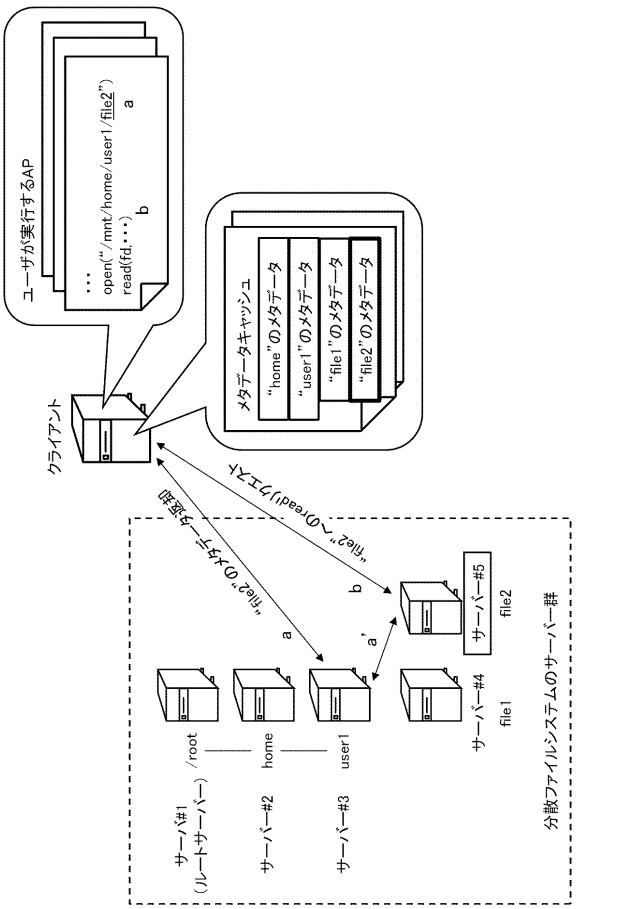
【図４】


ユーザが実行するAP
main()
…
open(“/mnt/home/user1/file1”)
…
a b c
クライアント
メタデータキャッシュ
“home”のメタデータ
“user1”のメタデータ
“file1”のメタデータ
“home”のメタデータ問合せ
“user1”のメタデータ問合せ
“file1”のメタデータ問合せ
分散ファイルシステムのサーバー群
サーバー#1（ルートサーバー）
/root
サーバー#2
home
サーバー#3
user1
サーバー#4
file1

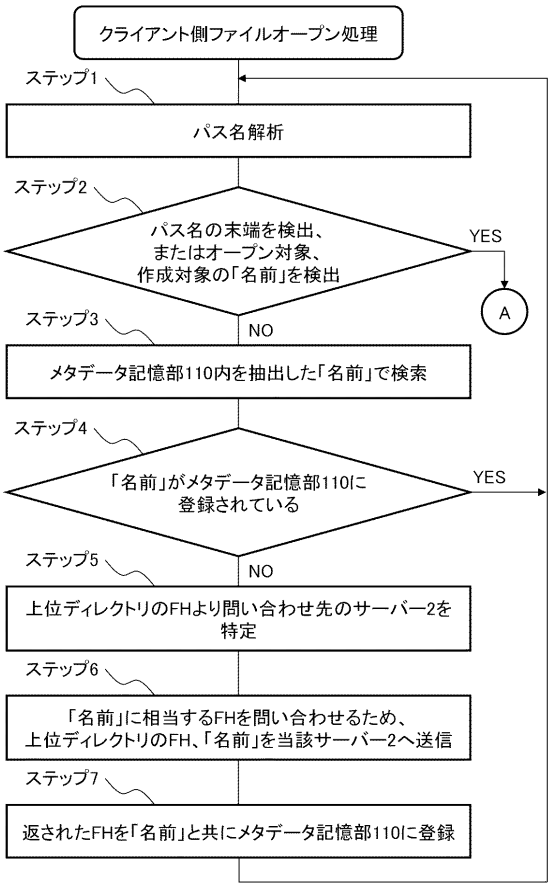
【図５】


ユーザが実行するAP
…
open(“/mnt/home/user1/file1”)
read(fd,・・・)
d
クライアント
メタデータキャッシュ
“home”のメタデータ
“user1”のメタデータ
“file1”のメタデータ
“file1”へのreadリクエスト
分散ファイルシステムのサーバー群
サーバー#1（ルートサーバー）
/root
サーバー#2
home
サーバー#3
user1
サーバー#4
file1

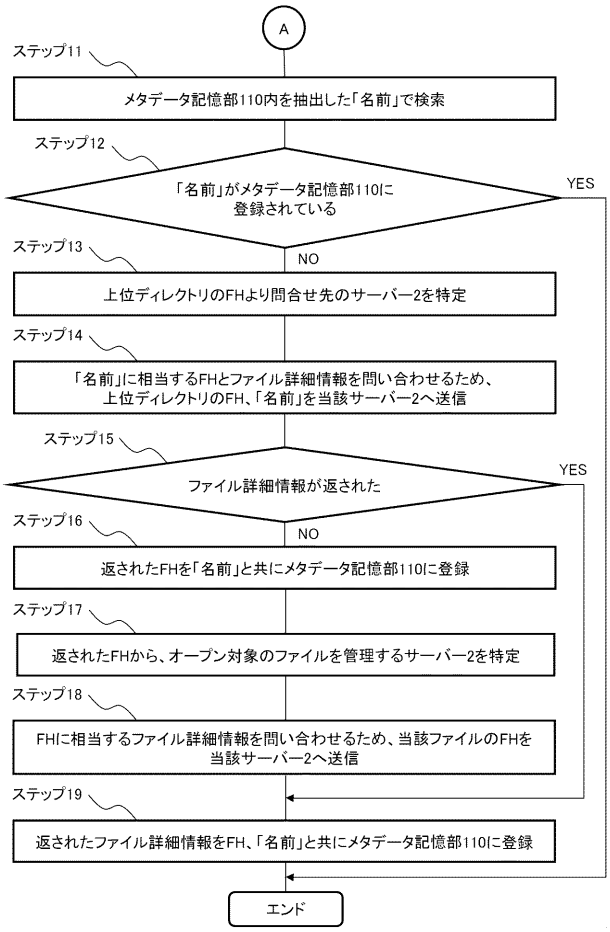
【図６】


ユーザが実行するAP
…
open(“/mnt/home/user1/file2”)
read(fd,・・・)
a
b
クライアント
メタデータキャッシュ
“home”のメタデータ
“user1”のメタデータ
“file1”のメタデータ
“file2”のメタデータ
“file2”のメタデータ問合せ
“file2”へのreadリクエスト
分散ファイルシステムのサーバー群
サーバー#1（ルートサーバー）
/root
サーバー#2
home
サーバー#3
user1
サーバー#4
file1
サーバー#5
file2

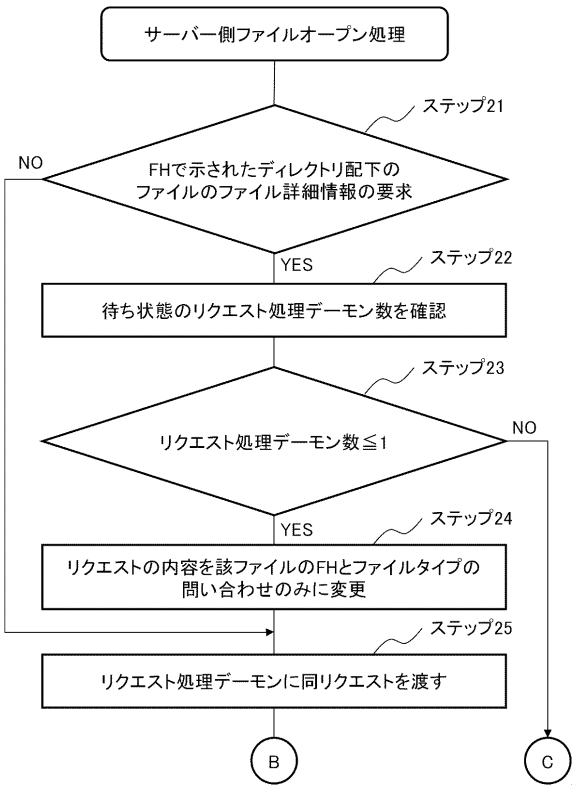
【図７】


クライアント側ファイルオープン処理
ステップ1
パス名解析
ステップ2
パス名の末端を検出、
またはオープン対象、
作成対象の「名前」を検出
YES
A
ステップ3
NO
メタデータ記憶部110内を抽出した「名前」で検索
ステップ4
「名前」がメタデータ記憶部110に
登録されている
YES
ステップ5
NO
上位ディレクトリのFHより問い合わせ先のサーバー2を
特定
ステップ6
「名前」に相当するFHを問い合わせるため、
上位ディレクトリのFH、「名前」を当該サーバー2へ送信
ステップ7
返されたFHを「名前」と共にメタデータ記憶部110に登録

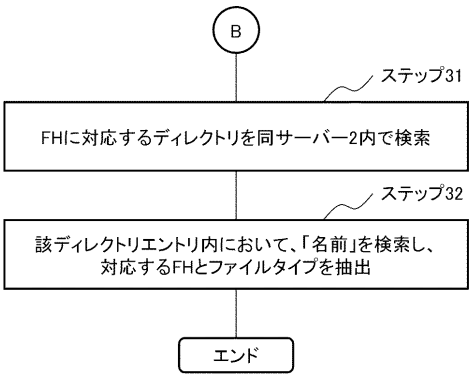
【図８】


A
ステップ11
メタデータ記憶部110内を抽出した「名前」で検索
ステップ12
「名前」がメタデータ記憶部110に
登録されている
YES
ステップ13
NO
上位ディレクトリのFHより問合せ先のサーバー2を特定
ステップ14
「名前」に相当するFHとファイル詳細情報を問い合わせるため、
上位ディレクトリのFH、「名前」を当該サーバー2へ送信
ステップ15
ファイル詳細情報が返された
YES
ステップ16
NO
返されたFHを「名前」と共にメタデータ記憶部110に登録
ステップ17
返されたFHから、オープン対象のファイルを管理するサーバー2を特定
ステップ18
FHに相当するファイル詳細情報を問い合わせるため、当該ファイルのFHを
当該サーバー2へ送信
ステップ19
返されたファイル詳細情報をFH、「名前」と共にメタデータ記憶部110に登録
エンド

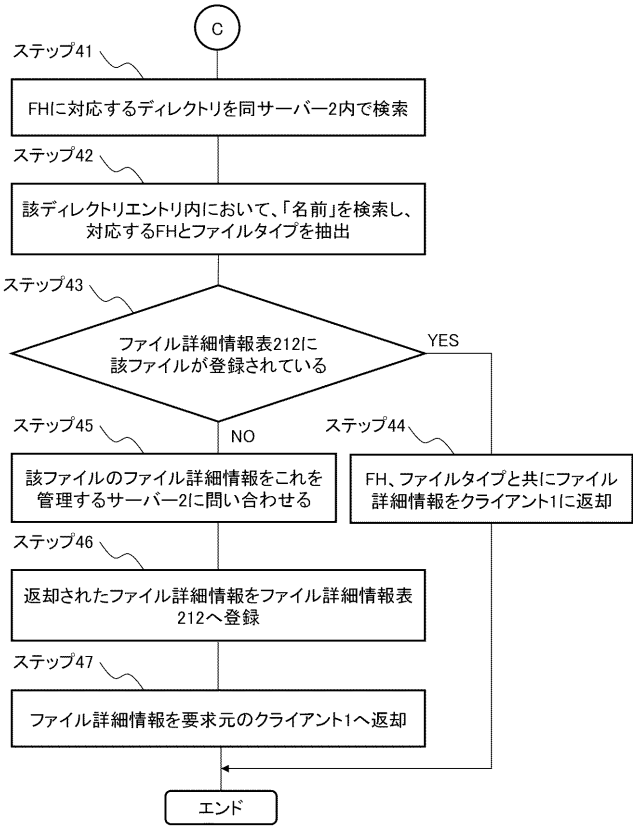
【図９】


サーバー側ファイルオープン処理
ステップ21
NO
FHで示されたディレクトリ配下の
ファイルのファイル詳細情報の要求
YES
ステップ22
待ち状態のリクエスト処理デーモン数を確認
ステップ23
リクエスト処理デーモン数≦1
NO
YES
ステップ24
リクエストの内容を該ファイルのFHとファイルタイプの
問い合わせのみに変更
ステップ25
リクエスト処理デーモンに同リクエストを渡す
B
C

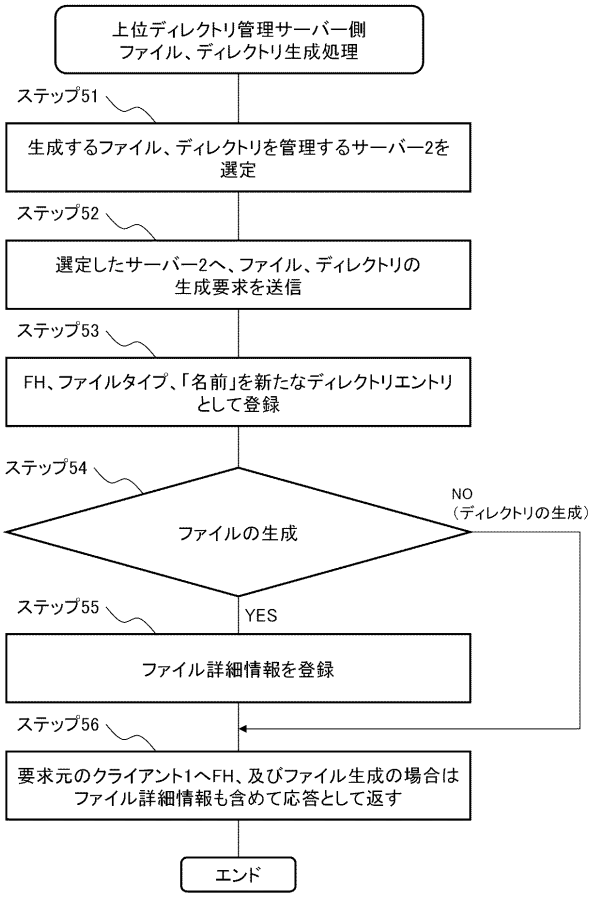
【図１０】


B
ステップ31
FHに対応するディレクトリを同サーバー2内で検索
ステップ32
該ディレクトリエントリ内において、「名前」を検索し、
対応するFHとファイルタイプを抽出
エンド

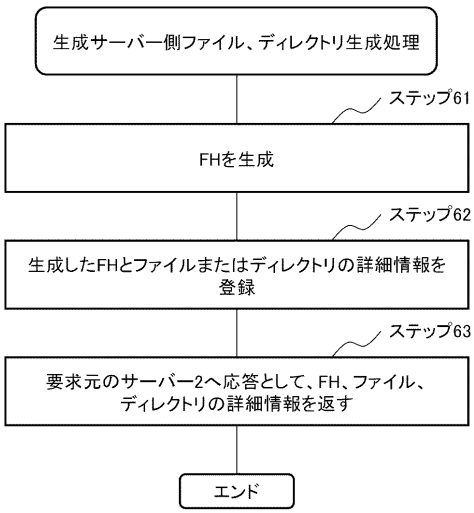
【図１１】



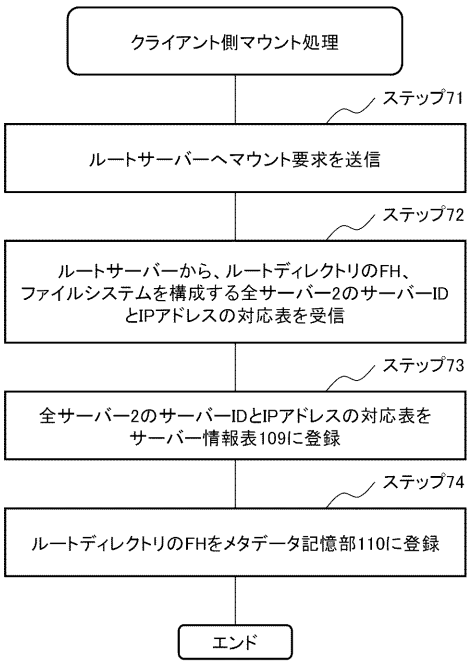
【図１２】



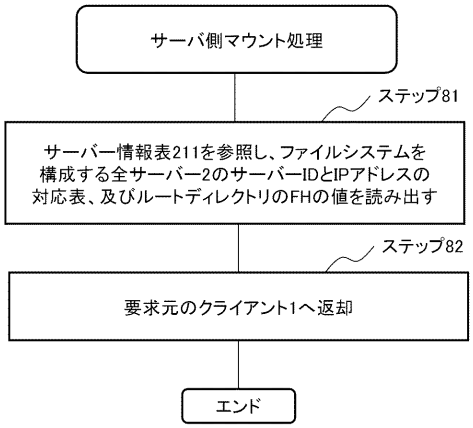
【図１３】



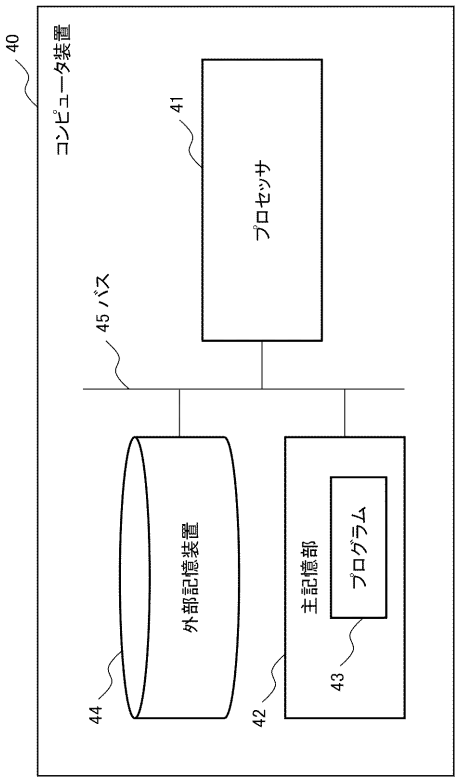
【図１４】



【図１５】


サーバ側マウント処理
ステップ81
サーバー情報表211を参照し、ファイルシステムを
構成する全サーバー2のサーバーIDとIPアドレスの
対応表、及びルートディレクトリのFHの値を読み出す
ステップ82
要求元のクライアント1へ返却
エンド

【図１６】


40
コンピュータ装置
41
プロセッサ
45 バス
外部記憶装置
主記憶部
プログラム
44
42
43