(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4263421号
(P4263421)
(45)発行日　平成21年5月13日(2009.5.13)　(24)登録日　平成21年2月20日(2009.2.20)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 545A
G06F 12/00 531D
G06F 12/00 535B

請求項の数 10　(全 28 頁)

(21)出願番号　特願2002-86870(P2002-86870)
(22)出願日　平成14年3月26日(2002.3.26)
(65)公開番号　特開2002-358226(P2002-358226A)
(43)公開日　平成14年12月13日(2002.12.13)
審査請求日　平成17年3月23日(2005.3.23)
(31)優先権主張番号　60/278,905
(32)優先日　平成13年3月26日(2001.3.26)
(33)優先権主張国　米国(US)
(31)優先権主張番号　10/005,629
(32)優先日　平成13年12月5日(2001.12.5)
(33)優先権主張国　米国(US)

(73)特許権者　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　アトゥル　アジャ
アメリカ合衆国　９８００７　ワシントン州　ベルビュー　ノースイースト　３５ストリート　１４５９０　ナンバーディー２０４

最終頁に続く

(54)【発明の名称】サーバレス分散ファイルシステム

(57)【特許請求の範囲】
【請求項１】
ファイルを記憶するのに使用する階層的名前空間のサブツリーを管理する１または複数のコンピュータからなるディレクトリグループを決定する方法であって、
第１のディリクトリを管理する第１のディレクトリグループのコンピュータが、前記第１のディレクトリのサブツリーの一部の管理を委任する第２のディレクトリグループのコンピュータを選択するステップと、
前記第１のディレクトリグループのコンピュータが、前記サブツリーの委任証明書を生成するステップであって、前記委任証明書は、
第３のディレクトリグループのコンピュータが、前記サブツリーの名前空間ルートの管理に責任を負うことを証明する、第１のディジタル署名された証明書と、
前記第２のディレクトリグループのコンピュータの権限を、前記名前空間ルートまでさかのぼって、前記第３のディレクトリグループのコンピュータの証明書と関連付けることを可能にする、第２のディジタル署名された証明書とを含む、ステップと、
前記第１のディレクトリグループのコンピュータが、前記委任証明書にディジタル署名するステップと、
前記第１のディレクトリグループのコンピュータが、前記第２のディレクトリグループのコンピュータに前記委任証明書を発行するステップと
を備えたことを特徴とする方法。
【請求項２】

前記委任証明書にディジタル署名することは、前記委任証明書に、複数のコンピュータがディジタル署名させることを備えたことを特徴とする請求項１に記載の方法。
【請求項３】
前記第２のディレクトリグループのコンピュータは、ビザンチンフォルトトレラントグループを構成することを特徴とする請求項１に記載の方法。
【請求項４】
前記第１および第２のディジタル署名された証明書は、
委任されるサブツリーのルートの識別と、
前記委任されるサブツリーの識別と、
前記サブツリーを管理するディレクトリグループのメンバの識別と
を含むことを特徴とする請求項１に記載の方法。
【請求項５】
前記第２のディレクトリグループのコンピュータは、前記第１のディレクトリグループのコンピュータに含まれていることを特徴とする請求項４に記載の方法。
【請求項６】
前記第１のディジタル署名された証明書は、認証局（ＣＡ）によってディジタル署名された証明書であることを特徴とする請求項１に記載の方法。
【請求項７】
前記委任証明書は、前記第２のディジタル署名された証明書から前記第１のディジタル署名された証明書への、証明書のチェーンの確立を可能にする、ディジタル署名された１または複数の追加の証明書を含むことを特徴とする請求項１に記載の方法。
【請求項８】
サーバレス分散ファイルシステムにおいて、ファイルを記憶するのに使用する階層的名前空間のサブツリーを管理する１または複数のコンピュータからなるディレクトリグループを決定する、第１のディリクトリを管理する第１のディレクトリグループのコンピュータであって、
プロセッサと、
該プロセッサに結合されたメモリとを備え、
該メモリに格納されたプログラム命令を、前記プロセッサが実行すると、
前記第１のディレクトリのサブツリーの一部の管理を委任する第２のディレクトリグループのコンピュータを識別するステップと、
前記サブツリーの委任証明書を生成するステップであって、前記委任証明書は、
第３のディレクトリグループのコンピュータが、前記サブツリーの名前空間ルートの管理に責任を負うことを証明する、第１のディジタル署名された証明書と、
前記第２のディレクトリグループのコンピュータの権限を、前記名前空間ルートまでさかのぼって、前記第３のディレクトリグループのコンピュータの証明書と関連付けることを可能にする、第２のディジタル署名された証明書とを含む、ステップと、
前記委任証明書にディジタル署名するステップと、
前記第２のディレクトリグループのコンピュータに前記委任証明書を発行するステップと
を実行することを特徴とするコンピュータ。
【請求項９】
前記第２のディレクトリグループのコンピュータは、ビザンチンフォルトトレラントグループを構成することを特徴とする請求項８に記載のコンピュータ。
【請求項１０】
前記第１および第２のディジタル署名された証明書は、
委任されるサブツリーのルートの識別と、
前記委任されるサブツリーの識別と、
前記サブツリーを管理するディレクトリグループのメンバの識別と
を含むことを特徴とする請求項８に記載のコンピュータ。

【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、コンピュータネットワークおよびファイルシステムに関し、より詳細には、サーバレス分散ファイルシステム（a serverless distributed file system）に関する。
【０００２】
【従来の技術】
ファイルシステムは、コンピュータシステム上に記憶されたファイルおよび他のデータオブジェクトを管理する。ファイルシステムは元来、コンピュータのオペレーティングシステム中に構築され、常駐する記憶媒体上にローカルに記憶されたファイルへのアクセスを容易にする。コンピュータがネットワーク化されると、いくつかのファイル記憶機能は、個々のユーザマシンから専用の記憶サーバへオフロードされ、これらのユーザマシンに代って専用記憶サーバが多数のファイルを記憶する。ファイルが必要になると、ユーザマシンは、単にサーバにファイルを要求した。このサーバベースのアーキテクチャにおいて、ネットワークを介して遠隔に、記憶サーバに記憶されたファイルの管理およびアクセスを容易にするようにファイルシステムが拡張された。
【０００３】
【発明が解決しようとする課題】
今日、ファイル記憶は、中央記憶サーバ上にファイルを記憶するよりも、ネットワーク化されたさまざまなコンピュータ上にファイルを記憶するモデルへと移行している。サーバのないアーキテクチャは、ファイルシステムに新たな課題を提起している。具体的な１つの課題は、多くの異なるコンピュータ上に分散したファイルを、コンピュータのいくつかを任意の時刻にアクセス不能にするにもかかわらず、高い信頼性で記憶し、アクセスできる方法で管理し、同時に、無許可ユーザによるファイルへのアクセスを防止することに関する。
【０００４】
本発明は、サーバレス分散ファイルシステムのこれらの課題に取り組み、有効な解決方法を提供する。
【０００５】
【課題を解決するための手段】
本明細書には、サーバレス分散ファイルシステムを開示する。
【０００６】
一態様によれば、ファイルとディレクトリは、サーバレス分散ファイルシステム内で異なる方法で管理される。ディレクトリは、ビザンチンフォルトトレラントグループ（Byzantine-fault-tolerant groups）を使用して管理され、一方、ファイルは、ビザンチンフォルトトレラントグループを使用せずに管理される。これは、対応するディレクトリエントリのコピーよりも少ない数のファイルコピーを記憶する結果、性能が向上する。
【０００７】
他の態様によれば、ファイルシステムは、階層的名前空間を使用してファイルを記憶する。ファイルは、複数のコンピュータを横断して分散され、これら複数のコンピュータの各々は、クライアントコンピュータおよびサーバコンピュータとして動作することができ、他のコンピュータを信用する必要がない。
【０００８】
他の態様によれば、ファイルシステム内の１または複数のディレクトリを管理する責任が、ディレクトリグループに割り当てられる。ディレクトリグループの各メンバは、システムに参加したコンピュータであり、ディレクトリグループは、各ディレクトリ中のオブジェクト（例えばファイルおよびディレクトリ）へのアクセスを制御する複数のロック（locks）を使用することができる。ロックは、オブジェクトのオープンを制御する第１のロックセット、およびオブジェクト中のデータへのアクセスを制御する第２のロックセットを含む。

【０００９】
【発明の実施の形態】
本明細書を通して、同じ構成要素および／または特徴を参照するのに同じ符号を使用した。
【００１０】
以下の議論は、ネットワーク化された複数のコンピュータ上で稼動し、中央サーバまたはサーバ群にファイルを記憶するのではなしに、これらのネットワーク化されたコンピュータ上にファイルを記憶する、共生（symbiotic）、サーバレス（serverless）、分散ファイルシステムを対象とする。共生とは、マシンが相互に協力はするが、完全には信用しないことを意味する。このファイルシステムは、ディスク記憶装置を直接には管理せず、オペレーティングシステムに組み込まれたファイルシステム（例えば、Windows（登録商標） NTのファイルシステム）などの、ローカルマシン上に存在するファイルシステムに依存する。
【００１１】
本明細書の議論では、公開鍵、暗号化およびディジタル署名に言及する。公開鍵暗号とは一般に、ひとまとめに鍵対と呼ばれる公開鍵と秘密鍵の使用を指す。実体（例えば、ユーザ、ソフトウェアアプリケーションなど）は、秘密鍵を秘密にしておき、公開鍵を他の実体に公開する。データは一般に平文と呼ばれ、暗号化アルゴリズムおよび公開鍵を使用して暗号化することができる。この場合、暗号化した結果（一般に暗号文と呼ばれる）は、対応する秘密鍵を知らなければ簡単には復号できないが、対応する秘密鍵を知っていれば比較的に簡単に復号することができる。同様に、暗号化アルゴリズムおよび秘密鍵を使用してデータにディジタル署名することができる。この場合、署名は、対応する公開鍵を使用して簡単に検証できるが、秘密鍵がなければ簡単に生成することができない。本明細書の議論は、読者が暗号法の基本的な理解を有することを前提とする。暗号法の基本については、Bruce Schneier著「Applied Cryptography :Protocols, algorithms, and Source Code in C」、John Wiley & Sons社刊、１９９４年（または第２版、１９９６年）を参照されたい。
【００１２】
（サーバレス分散ファイルシステム）
図１に、サーバレス分散ファイルシステムをサポートする例示的なネットワーク環境１００を示す。４台のクライアントコンピューティング装置１０２、１０４、１０６および１０８が、データ通信ネットワーク１１０を介して相互に結合されている。４台のコンピューティング装置を図示したが、ネットワーク環境１００には、さまざまな数（４台を越えるまたは４台未満）の装置を含めることができる。
【００１３】
ネットワーク１１０は、多種多様なデータ通信ネットワークのうちの１つを表す。ネットワーク１１０は、公的部分（例えばインターネット）、私的部分（例えば企業内ローカルエリアネットワーク（ＬＡＮ））、あるいは公的部分と私的部分との組合せを含むことができる。ネットワーク１１０は、有線および無線媒体を含む従来のさまざまな通信媒体のうち、１または複数の媒体を使用して実現することができる。公衆のおよび私有のプロトコルを含むさまざまな通信プロトコルのいずれかを使用して、ネットワーク１１０経由でデータを伝達することができる。このようなプロトコルの例にはＴＣＰ／ＩＰ、ＩＰＸ／ＳＰＸ、ＮｅｔＢＥＵＩなどがある。
【００１４】
コンピューティング装置１０２～１０８は、広範囲にわたるコンピューティング装置の１つを表し、互いに同じ装置であってもよいし、または異なる装置であってもよい。装置１０２～１０８は例えば、デスクトップコンピュータ、ラップトップコンピュータ、ハンドヘルドまたはポケットコンピュータ、パーソナルディジタルアシスタント（ＰＤＡ）、携帯電話、インターネット機器、家庭用電子装置、ゲーム機などである。
【００１５】

２台以上の装置１０２～１０８が動作して、サーバレス分散ファイルシステムを実現する。サーバレス分散ファイルシステムに参加する実際の装置は、その時々で変更することができる。すなわち、新たな装置をシステムに追加し、他の装置をシステムから除くことができる。分散ファイルシステムを実現する（分散ファイルシステムに参加する）各装置１０２～１０６は、ローカル記憶または分散記憶として使用するよう割り当てられた大容量記憶装置（例えば、ハードディスクドライブ）の部分を有する。ローカル記憶は、ユーザが、分散ファイルシステム構造にではなく、自分のローカルマシン上に記憶しておきたいデータに対して使用される。分散記憶部分は、その装置（または他の装置）のユーザが分散ファイルシステム構造に記憶しておきたいデータに対して使用される。

【００１６】

図１に示した例では、ネットワーク１１０に接続されたある装置が、分散部分とローカル部分の両方を含む１または複数の大容量記憶装置を有する。分散またはローカル記憶に割り当てられる量は、装置ごとに異なる。例えば装置１０２では、ローカル部分１２２に比べて分散システム部分１２０に大きな割合が割り当てられており、装置１０４は、ローカル部分１２６とほぼ同じサイズの分散システム部分１２４を含み、装置１０６では、ローカル部分１３０に比べて分散システム部分１２８に小さな割合が割り当てられている。記憶域の複数部分の分離は、記憶装置単位で実施し（例えば、１台のハードドライブを分散システムで使用するように指定し、別のハードドライブをローカル使用専用に指定する）、および／または単一の記憶装置内で実施する（例えば、１台のハードドライブの一部分を分散システムで使用するように指定し、別の部分をローカル使用向けに指定する）ことができる。分散記憶またはローカル記憶に割り当てる量は、その時々で変更することができる。コンピューティング装置１０８などのネットワーク１１０に接続された他の装置が、分散ファイルシステムのいずれの部分をも実現せず、したがってその大容量記憶装置内に、分散システムが使用するように割り振られた部分を持たないことがある。したがって装置１０８は、ローカル部分１３２だけしか持たない。

【００１７】

分散ファイルシステム１５０は、異なるコンピューティング装置１０２～１０６に１または複数のファイルコピーを記憶するように動作する。コンピュータのユーザが新たなファイルを作成したとき、ユーザは、自分のコンピューティング装置のローカル部分にファイルを記憶するのか、または分散ファイルシステムにファイルを記憶するのかを選択することができる。分散ファイルシステム１５０に記憶する場合には、ファイルが、１台または数台の装置１０２～１０６の大容量記憶装置の分散システム部分に記憶される。ファイルを作成したユーザは一般に、どの装置１０２～１０６上にファイルを記憶するかを制御することができず、そのファイルがどの装置１０２～１０６上に記憶されたかも知ることができない。さらに、複製された複数のファイルコピーが保存され、たとえファイルが保存されているコンピューティング装置１０２～１０６のうちの１台が使用できない（例えば、パワーダウンまたは誤動作しているなど）場合であっても、ユーザが続いてファイルを検索することができる。

【００１８】

分散ファイルシステム１５０は、各装置１０２～１０６上の１または複数のコンポーネントによって実現され、これによってファイルシステムを調整する中央サーバの必要性が排除される。これらコンポーネントは、特定のファイルをどこに記憶するか、いくつのファイルコピーを作成して異なる装置上に記憶するかなどを決定するように動作する。どの装置にどのファイルが記憶されるかは多くの因子によって決まり、これには、分散ファイルシステム内の装置数、各装置からファイルシステムに割り振られた記憶空間、保存するファイルコピーの数、暗号法上安全な乱数、装置上にすでに記憶されているファイルの数などが含まれる。したがって、この分散ファイルシステムによってユーザは、正確にどのコンピューティング装置上にファイルが記憶されているのかを一切知らなくとも、ファイル（フォルダまたはディレクトリも同様に）を作成し、アクセスすることができる。

【００１９】

分散ファイルシステム１５０は、システム１５０内の多数のコンピュータをサポートするため、拡張が容易であるように設計される。システム１５０内の装置上のコンポーネントが使用するプロトコルおよびデータ構造は、システム内のコンピュータの数と釣り合うようには設計されず、これによって、多数のコンピュータに合わせて容易に拡張することができる。

【００２０】

ファイルシステムによって記憶されるファイルは、各装置１０２～１０６に分散され、暗号化された形態で記憶される。新しいファイルが作成されると、他の装置に伝達して記憶する前に、ファイルを作成した装置がファイルを暗号化する。新しいファイルのディレクトリエントリ（ファイル名を含んでいる）も、他の装置に伝達され記憶される。他の装置は、暗号化されたファイルを記憶する装置と同じである必要はない（一般には異なる装置に記憶される）。さらに、新しいフォルダまたはディレクトリが作成された場合も、ディレクトリエントリ（フォルダ名またはディレクトリ名を含んでいる）が他の装置に伝達され記憶される。本明細書においてディレクトリエントリは、ファイルシステムのディレクトリに追加することができ、ファイル名とディレクトリ名（またはフォルダ名）の両方を含む、任意のエントリを指す。

【００２１】

分散ファイルシステム１５０は、１台の装置１０２～１０６上に記憶されたデータを無許可ユーザが読めないように設計される。したがって、装置１０２が作成し、装置１０４に記憶されたファイルを、装置１０４のユーザは（許可を得ない限り）読むことができない。このようなセキュリティを実現するため、ファイルの内容、ディレクトリエントリ中の全てのファイル名およびディレクトリ名が暗号化され、許可されたユーザだけに復号鍵が与えられる。したがって、装置１０２が作成したファイルを装置１０４は記憶することができるが、装置１０４のユーザがそのファイルの許可されたユーザでない限り、ファイルの内容も、そのディレクトリエントリ中のファイル名も復号することができない（したがって読むことができない）。

【００２２】

分散ファイルシステム１５０は、１または複数の名前空間ルートを有し、各名前空間ルートのもとに複数のサブツリーを有する階層的記憶構造を使用する。異なるサブツリーの管理を異なるコンピュータ群に委任し、これによって名前空間ルートまたは特定のサブツリーを管理するコンピュータの負担が過大になるのを防ぐことができる。

【００２３】

分散ファイルシステム１５０はさらに、ファイルの記憶とファイルに対応するディレクトリエントリの記憶を異なる方法で管理する。システム１５０内に記憶されるファイルは複製され、システム内の異なる複数のコンピュータ上に保存される。さらに、そのファイルに対してディレクトリエントリが生成され、システム内の異なる複数のコンピュータ上に保存される。保存されるファイルコピーの数よりも多くのディレクトリエントリが保存される。一実施態様では、ディレクトリエントリが、ビザンチンフォルトトレラントグループの一部であるコンピュータ上に記憶される。これについては後に詳細に論じる。

【００２４】

分散ファイルシステム１５０はさらに、ディレクトリおよびファイルの読取りまたは書込みを制御することができるディレクトリ／ファイルロック機構を使用する。ビザンチングループ内のコンピュータとともに使用すると、使用されたロック機構は、ローカルに実行することができるオペレーションの数を増やすことによって、ディレクトリグループによるアクションを要求することなく、性能を向上させようとする。これについては後に詳細に論じる。

【００２５】

分散ファイルシステム１５０内の全てのコンピュータ１０２～１０６は、３つの機能を有することができる。すなわち、ローカルユーザのクライアントとなり、システム内に記憶された暗号化されたファイルコピーの保管場所となり、１または複数のディレクトリを維

持する一群のコンピュータのメンバとなることができる。

【００２６】

一般に、コンピュータ１０２～１０６のユーザが所与のディレクトリ内のファイルを開くと、そのコンピュータは、ビザンチンフォルトトレラントプロトコルを使用して、そのディレクトリを集合的に管理している一組のコンピュータ（「ビザンチングループ」または「ディレクトリグループ」と呼ばれる）にリクエストを送る。ビザンチングループは、そのコンピュータにファイルロックを許可し、そのファイルにローカル更新を実施し（書込みロックの場合）、続いてそれらの更新をビザンチングループへプッシュすることを可能にする。コンピュータが最近このファイルにアクセスしていた場合、このコンピュータはおそらく、暗号化されたファイル内容のコピーをローカルキャッシュ内に有し、そのためコンピュータは、キャッシュされているコピーを検索し、それを復号するだけでよく、その後、コンピュータはファイルの読取りまたは書込みを開始することができる。コンピュータが最近に、ファイルの現行バージョンにアクセスしていない場合、コンピュータは、そのファイルを記憶したコンピュータの１つから暗号化されたファイルのコピーを検索する。どのコンピュータが現行コピーを保持しているかについての情報は、ロック許可とともにビザンチングループによって提供される。ファイルを記憶した１台または数台のコンピュータがダウンしている場合には、別のコンピュータからファイルを検索する。ビザンチングループはさらに、取り込んだファイルを確認するためコンピュータが使用する、ファイル内容の暗号ハッシュを提供する。

【００２７】

（ファイルの暗号化）

ファイルは、「収束化暗号（convergent encryption）」として知られる技術を使用して暗号化される。収束化暗号は、以下の２つの特性を有する。第１に、暗号化可能な２またはそれ以上のオブジェクトが同一である場合、それらを暗号化して個々の暗号オブジェクトを提供するのに異なる暗号化鍵を使用する場合でも、暗号化鍵のいくつかにアクセスし、暗号化可能なオブジェクトが同一であることを、暗号オブジェクトを調べて決定する必要がない。第２に、２またはそれ以上の暗号化可能なオブジェクトが同一であり、それらを異なる暗号化鍵で暗号化する場合、全ての暗号オブジェクトを記憶するのに必要な総記憶空間は、暗号化可能な単一のオブジェクトを記憶するのに必要な空間に各々の異なった暗号化鍵に対する一定量の記憶域を加えたものに比例する。

【００２８】

コ一般に、ンバージェント暗号化によれば、ファイルＦ（または他の暗号化可能オブジェクトの任意のタイプ）は、一方向ハッシュ関数ｈ（例えばＳＨＡ、ＭＤ５など）を使用してハッシュされ、ハッシュ値ｈ（Ｆ）を生成する。次いで、対称暗号（例えばＲＣ４、ＲＣ２など）を使用し、ハッシュ値を鍵として、すなわち$E_{h(F)}$（Ｆ）としてファイルＦを暗号化する。次に、この暗号化されたファイルへの読取りアクセスが許可された各許可ユーザに対して、読取りアクセス制御エントリを作成する。書込みアクセス制御は、そのファイルのディレクトリエントリを記憶したディレクトリサーバによって統率される。読取りアクセス制御エントリは、任意の数の鍵K_1、K_2、．．．、K_mを用いて、ファイルのハッシュ値ｈ（Ｆ）を暗号化し、E_{K1}（ｈ（Ｆ））、E_{K2}（ｈ（Ｆ））、．．．、E_{Km}（ｈ（Ｆ））を得ることによって、形成される。一実施態様では、各々の鍵Ｋが、非対称暗号（例えばＲＳＡ）の公開／秘密鍵対のユーザの公開鍵である。

【００２９】

収束化暗号では、サーバレス分散ファイルシステム１５０の間で、ファイルの１つの暗号化バージョンが記憶され、複製される。ファイルの暗号化バージョンとともに、アクセスを有する許可ユーザ数に応じた１または複数のアクセス制御エントリが記憶される。したがって、分散ファイルシステム１５０内のファイルは以下の構造を有する。

$[E_{h(F)}(F), <E_{K1}(h(F))>, <E_{K2}(h(F))>, \ldots, <E_{Km}(h(F))>]$

収束化暗号の１つの利点は、暗号化されたファイルをファイルシステムが評価して、復号に訴えることなく（したがって暗号化鍵を知らなくても）、それが他のファイルと同一か

どうかを決定することができる点である。許可ユーザのアクセス制御エントリを残りのファイルに追加することによって、不要な複製ファイルを除去することができる。他の利点は、おそらくギガバイトに達する暗号化されたファイルに比べて、アクセス制御エントリのサイズが非常に小さいことである。その結果、各々のファイルに記憶されるオーバヘッド情報の量が小さくなる。これは、ファイルを記憶するために使用される総記憶空間が、暗号化された単一のファイルを記憶するのに必要な空間に、そのファイルの各追加の許可された読者に対する一定量の記憶域を加えたものに比例するという特性を可能にする。

【００３０】

収束化暗号の詳細については、Douceur他の名義で２０００年５月５日に出願され、Microsoft Corporationに譲渡された「Encryption Systems and Methods for Identifying and Coalescing Identical Objects Encrypted with different Keys」という名称の同時係属米国特許出願第０９／５６５８２１号を参照されたい。この出願は、参照によって本明細書に組み込まれる。

【００３１】

（ディレクトリエントリの暗号化）

ディレクトリエントリ中のファイル名およびディレクトリ名は、「排他的暗号（exclusive encryption）」と呼ばれるプロセスを使用して暗号化する。排他的暗号は、ディレクトリエントリ中のファイル名およびディレクトリ名を暗号化された形で記憶することを可能にし、これによって無許可ユーザが、ファイル名またはディレクトリ名に基づいて情報を不適切に獲得することを防ぐ。さらに、排他的暗号は、以下の３つの特性を有する。第１に、１つのディレクトリ内の暗号化されたエントリは、同じ名前に復号されることがない。第２に、１つのディレクトリ内の暗号化されたエントリが全て、構文上合法な名前に復号される。第３に、ディレクトリを維持するディレクトリグループが、エントリの平文名にアクセスすることができない。したがってファイルシステム１５０は、ディレクトリ内の２つのエントリが同じ名前の暗号化でないこと、およびディレクトリ内の全てのエントリが構文上合法な名前の暗号化であることを保証することができ、同時に、ディレクトリを維持する装置が、エントリの平文名にアクセスできないことを保証する。

【００３２】

排他的暗号によれば一般に、平文名（ディレクトリエントリ中のファイル名またはディレクトリ名）が新しい名前にマップされる。マップされた名前は、任意選択で、ディケシファイ（decasified）された（大文字と小文字の区別をしない）名前と対応する大文字小文字情報にディケシファイされ、大文字と小文字の区別をしない複製名前検出が可能になる。マップ（および任意選択でディケシファイ）された名前は、次いで符号化され、暗号化される。この暗号化された名前（および任意選択で付随する大文字小文字情報）は、ディレクトリエントリの管理に責任を負うディレクトリグループに（例えば後に詳細に論じるパス名に基づいて）転送される。

【００３３】

排他的暗号の詳細については、Douceur他の名義で２００１年１月１７日に出願され、Microsoft Corporationに譲渡された「Exclusive Encryption for a Secure Directory Service」という名称の同時係属米国特許出願第０９／７６４９６２号を参照されたい。この出願は、参照によって本明細書に組み込まれる。

【００３４】

（ファイルフォーマット）

図１のサーバレス分散ファイルシステム１５０のファイルフォーマットは、基本データストリームとメタデータストリームの２つの部分から成る。基本データストリームは、複数のブロックに分割されたファイルを含む。各ブロックは、対称暗号（例えばＲＣ４）および暗号化鍵としてブロックのハッシュを使用して暗号化される。メタデータストリームは、ヘッダ、基本データストリーム中の暗号化されたブロックに索引を付けるための構造、およびいくつかのユーザ情報を含む。

【００３５】

索引付けツリー構造は、各々のブロックのリーフノードを定義する。各リーフノードは、関連ブロックの復号に使用するアクセス値、および他のブロックとは独立に暗号化されたブロックを検証するのに使用する検証値から成る。一実施態様では、アクセス値は、ファイルブロックをハッシュし、得られたハッシュ値を、対称暗号およびランダムに生成された鍵を使用して暗号化することによって形成される。次いでこの鍵を、非対称暗号（例えばＲＳＡ）および暗号化鍵としてユーザの公開鍵を使用して暗号化する。検証値は、一方向ハッシュ関数（例えばＳＨＡ）を使用して、暗号化された関連ブロックをハッシュすることによって形成する。

【００３６】

ファイルのサイズに応じて、索引付け構造は、リーフノードをツリーブロックにグループ分けし、各ツリーブロックのハッシュ値を計算することによって形成された中間ノードを含むことができる。これらの中間ノードは、ブロックに再びセグメント化することができ、各ブロックをハッシュして次のノードを形成することができる。これを、ルートノードに到達するまで所望の回数繰り返すことができる。次いで、ルートノードをハッシュし、このハッシュ値を、メタデータヘッダおよびユーザ情報とともに使用して、ファイル全体の検証値を生み出す。一実施態様では、この全体ファイル検証値には、ユーザの署名が署名される。あるいは、このような署名なしでファイルを構築することもできる。

【００３７】

このファイルフォーマットは、ランダムに生成された鍵または任意のユーザ鍵についての知識なしで個々のファイルブロックを検証することをサポートする。ファイルのブロックを検証するため、ファイルシステムは任意選択で、全体ファイル検証値上の署名を評価し（存在する場合）、全体ファイル検証値がルートブロックのハッシュ、メタデータヘッダおよびユーザ情報と一致するかどうかをチェックし、次いで、検証する目的ブロックに関連した適当なリーフノードまでツリーを横断する。ファイルシステムが目的ブロックをハッシュし、ハッシュがリーフノードに含まれるアクセス値と一致する場合、そのブロックは本物である。

【００３８】

このファイルフォーマットはさらに、他のブロックを妨害することなく個々のブロックから読み取り、個々のブロックへ書き込むことをサポートする。このファイルフォーマットは、広大な非データ領域を有する粗なファイルに資する。

【００３９】

ファイルフォーマットの詳細については、Bolosky他の名義で２００１年３月２１日に出願され、Microsoft Corporationに譲渡された「On-Disk File Format for a Serverless Distributed File System」という名称の同時係属米国特許出願第０９／８１４２５９号を参照されたい。この出願は、参照によって本明細書に組み込まれる。

【００４０】

（コンピューティング装置のアーキテクチャ）

図２に、図１の分散ファイルシステム１５０に参加しているコンピューティング装置１０２～１０６のうちの１台を表す例示的なコンピューティング装置２００の論理的構成要素を示す。コンピューティング装置２００は、サーバコンポーネント２０２、クライアントコンポーネント２０４、メモリ２０６、大容量記憶装置２０８および分散ファイルシステムインタフェース２１０を含む。典型的には、コンピューティング装置２００は、追加の構成要素（例えばプロセッサ）を含むが、図面が乱雑にならないよう図２には、それらの追加構成要素を図示しない。さまざまなハードウェアおよびソフトウェアコンポーネントを有するより一般的なコンピュータアーキテクチャを後に図３を参照して説明する。

【００４１】

メモリ２０６は、ＲＡＭ、ＲＯＭ、フラッシュメモリなどの従来の任意の揮発性および／または不揮発性メモリとすることができる。大容量記憶装置２０８は、磁気ディスク、光ディスク、フラッシュメモリなどの従来の任意の不揮発性記憶装置とすることができる。大容量記憶装置２０８は、分散システム部分とローカル部分とに区画されている。図２に

は、大容量記憶装置２０８が１つしか示されていないが、コンピューティング装置２００は、（異なるタイプまたは全て同じタイプの）複数の記憶装置２０８を含むことができる。

【００４２】

コンピューティング装置２００は、サーバレス分散ファイルシステム内で使用することを目的としたものであり、これに沿って、サーバコンポーネント２０２とクライアントコンポーネント２０４とを含む。サーバコンポーネント２０２は、記憶装置２０８に記憶された（または記憶する）ファイルまたはディレクトリエントリに関係する要求に装置２００が応答しているときに、要求を処理し、クライアントコンポーネント２０４は、分散ファイルシステム内に記憶された（または記憶する）ファイルまたはディレクトリのために、装置２００による要求の発行を処理する。クライアントコンポーネント２０４とサーバコンポーネント２０２は互いに独立に動作する。したがって、クライアントコンポーネント２０４が記憶中のファイルを、サーバコンポーネント２０２が大容量記憶装置２０８に記憶するといった状況も起こる可能性がある。

【００４３】

クライアントコンポーネント２０４は、コンピューティング装置１５０に代って、ファイルおよびディレクトリの作成、記憶、検索、読取り、書込み、変更および検証目的のサーバレス分散ファイルシステム１５０へのアクセスを、インタフェース２１０とともに管理する記憶／検索制御モジュール２２０を含む。制御モジュール２２０は、ディレクトリグループ探索モジュール２２２を使用して、特定のファイルまたはディレクトリの管理に責任を負うディレクトリグループを識別し、ファイル暗号化モジュール２２６を使用してファイルを暗号化し、ディレクトリ暗号化モジュール２２８を使用して、ディレクトリエントリ中のファイル名およびディレクトリ名を暗号化する。これらのモジュールの動作については後に詳細に論じる。

【００４４】

サーバコンポーネント２０２は、分散システム制御モジュール２５０、複製識別子２５２およびサブツリー委任モジュール２５４を含む。分散システム制御モジュール２５０は、暗号化されたファイル２４０へのアクセスを管理する。分散システム制御モジュール２５０は、大容量記憶装置２０８と通信して、暗号化されたファイル２４０を記憶／検索する。分散システム制御モジュール２５０はさらに、コンピューティング装置２００（あるいはサーバレス分散ファイルシステム内の他の場所）に記憶されたディレクトリエントリ（図示せず）のレコードを、メモリ２０６および／または大容量記憶装置２０８内に維持する。サブツリー委任モジュール２５４は、サブツリーを他のディレクトリグループに委任するように動作する。これについては後に詳細に論じる。

【００４５】

複製識別子２５２は、分散ファイルシステム内の暗号化された同一のファイルの識別を助ける。フォルトトレラント目的の意図的な複製ではない複製を見つけると、複製識別子２５２は、制御モジュール２５０に通知し、制御モジュール２５０は、この複製ファイルを排除し、排除されたファイルへのアクセス制御エントリを残りのファイルに追加する。

【００４６】

図３に、分散ファイルシステムの実現に使用する、より一般的なコンピュータ環境３００を示す。コンピュータ環境３００は、コンピューティング環境の一例に過ぎず、コンピュータおよびネットワークアーキテクチャの使用または機能の範囲に関して何らかの制限を暗示しようとするものではない。コンポーネントの包含（または排除）、あるいは例示的なコンピュータ環境３００内に示したコンポーネントの結合または組合せに関して、コンピュータ環境３００が要件を有すると解釈してはならない。

【００４７】

コンピュータ環境３００は、コンピュータ３０２の形態の汎用コンピューティング装置を含む。コンピュータ３０２のコンポーネントには、１または複数のプロセッサまたは処理ユニット３０４、システムメモリ３０６、およびシステムメモリ３０６にプロセッサ３０

４を含むさまざまなシステムコンポーネントを結合するシステムバス３０８が含まれる。ただしこれらに限定されるわけではない。

【００４８】

システムバス３０８は、メモリバスまたはメモリコントローラ、周辺バス、アクセラレーテッドグラフィクスポート、およびさまざまなバスアーキテクチャを使用したプロセッサまたはローカルバスを含むいくつかのタイプのバス構造のうち、任意の１または複数のバス構造を表す。このようなアーキテクチャには例えば、Industry Standard Architecture（ＩＳＡ）バス、Micro Channel Architecture（ＭＣＡ）バス、Enhanced ISA（ＥＩＳＡ）バス、Video Electronics Standards Association（ＶＥＳＡ）ローカルバス、およびMezzanineバスとしても知られるPeripheral Component Interconnects（ＰＣＩ）バスが含まれる。

【００４９】

コンピュータ３０２は一般に、さまざまなコンピュータ読取り可能な媒体を含む。このような媒体は、コンピュータ３０２がアクセス可能な任意の使用可能媒体であり、揮発性および不揮発性媒体と、取外し可能および取外し不能媒体とを含む。

【００５０】

システムメモリ３０６は、ランダムアクセスメモリ（ＲＡＭ）３１０などの揮発性メモリ、および／またはリードオンリーメモリ（ＲＯＭ）３１２などの不揮発性メモリの形態のコンピュータ読取り可能な媒体を含む。ＲＯＭ３１２には、スタートアップ時などに、コンピュータ３０２の要素間で情報を転送するのを助ける基本ルーチンを含む基本入出力システム（ＢＩＯＳ）３１４が記憶されている。ＲＡＭ３１０は一般に、処理ユニット３０４が即座にアクセス可能なおよび／または現在操作中の、データおよび／またはプログラムモジュールを含む。

【００５１】

コンピュータ３０２はさらに、取外し可能／取外し不能の揮発性／不揮発性コンピュータ記憶媒体を含むことができる。例えば図３には、取外し不能な不揮発性磁気媒体（図示せず）の読取り／書込み用のハードディスクドライブ３１６、取外し可能な不揮発性磁気ディスク３２０（例えば「フロッピー（登録商標）ディスク）の読取り／書込み用の磁気ディスクドライブ３１８、およびＣＤ－ＲＯＭ、ＤＶＤ－ＲＯＭなどの取外し可能な不揮発性光ディスク３２４の読取り／書込み用の光ディスクドライブ３２２が示されている。ハードディスクドライブ３１６、磁気ディスクドライブ３１８および光ディスクドライブ３２２は、１または複数のデータ媒体インタフェース３６０によってシステムバス３０８に接続されている。あるいは、ハードディスクドライブ３１６、磁気ディスクドライブ３１８および光ディスクドライブ３２２を、１または複数のインタフェース（図示せず）によってシステムバス３０８に接続することもできる。

【００５２】

これらのディスクドライブおよびその関連コンピュータ読取り可能な媒体は、コンピュータ３０２のコンピュータ読取り可能な命令、データ構造、プログラムモジュールおよびその他のデータの不揮発性記憶を提供する。この例には、ハードディスク３１６、取外し可能磁気ディスク３２０および取外し可能光ディスク３２４を示したが、コンピュータがアクセス可能なデータを記憶することができる、磁気カセットまたは他の磁気記憶装置、フラッシュメモリカード、ＣＤ－ＲＯＭ、ディジタルバーサタイルディスク（ＤＶＤ）または他の光記憶装置、ランダムアクセスメモリ（ＲＡＭ）、リードオンリーメモリ（ＲＯＭ）、電気的に消去可能なプログラマブルリードオンリーメモリ（ＥＥＰＲＯＭ）などの他のタイプのコンピュータ読取り可能な媒体を利用しても、この例示的なコンピューティングシステムおよび環境を実現できることを理解されたい。

【００５３】

ハードディスク３１６、磁気ディスク３２０、光ディスク３２４、ＲＯＭ３１２および／またはＲＡＭ３１０上には、例えば、オペレーティングシステム３２６、１または複数のアプリケーションプログラム３２８、他のプログラムモジュール３３０およびプログラム

データ３３２を含む、任意の数のプログラムモジュールを記憶することができる。オペレーティングシステム３２６、１または複数のアプリケーションプログラム３２８、他のプログラムモジュール３３０およびプログラムデータ３３２（またはこれらの組合せ）はそれぞれ、分散ファイルシステムをサポートする常駐コンポーネントの全部または一部を実現することができる。

【００５４】

ユーザは、キーボード３３４、ポインティングデバイス３３６（例えば「マウス」）などの入力装置を介して、コマンドおよび情報をコンピュータ３０２に入力することができる。他の入力装置３３８（具体的には図示せず）には、マイクロホン、ジョイスティック、ゲームパッド、衛星アンテナ、シリアルポートおよび／またはスキャナなどが含まれる。これらの入力装置およびその他の入力装置は、システムバス３０８に結合された入出力インタフェース３４０を介して処理ユニット３０４に接続されるが、パラレルポート、ゲームポート、ユニバーサルシリアルバス（ＵＳＢ）などの他のインタフェースおよびバス構造によって接続することもできる。

【００５５】

モニタ３４２または他のタイプのディスプレイ装置を、ビデオアダプタ３４４などのインタフェースを介してシステムバス３０８に接続することもできる。モニタ３４２の他に、出力周辺装置として、入出力インタフェース３４０を介してコンピュータ３０２に接続することができるスピーカ（図示せず）、プリンタ３４６などのコンポーネントを含めることができる。

【００５６】

コンピュータ３０２は、リモートコンピューティング装置３４８などの１台または数台のリモートコンピュータへの論理接続を使用した、ネットワーク化された環境で動作することができる。リモートコンピューティング装置３４８は例えば、パーソナルコンピュータ、ポータブルコンピュータ、サーバ、ルータ、ネットワークコンピュータ、ピア装置または他の一般的なネットワークノードなどである。リモートコンピューティング装置３４８は、コンピュータ３０２に関して本明細書で説明した要素および特徴の多くまたは全てを含むことができるポータブルコンピュータとして示されている。

【００５７】

コンピュータ３０２とリモートコンピュータ３４８の間の論理接続は、ローカルエリアネットワーク（ＬＡＮ）３５０および一般的なワイドエリアネットワーク（ＷＡＮ）３５２として図示されている。このようなネットワーキング環境は、事業所、企業内コンピュータネットワーク、イントラネットおよびインターネットでよく見られる。

【００５８】

ＬＡＮネットワーキング環境で実現するとき、コンピュータ３０２は、ネットワークインタフェースまたはネットワークアダプタ３５４を介してローカルネットワーク３５０に接続される。ＷＡＮネットワーキング環境で実現するとき、コンピュータ３０２は一般に、ワイドネットワーク３５２を介して通信を確立するためのモデム３５６またはその他の手段を含む。モデム３５６は、コンピュータ３０２の内部または外部に設置することができ、入出力インタフェース３４０または他の適当な機構を介してシステムバス３０８に接続することができる。図示のネットワーク接続は例示的なものであり、コンピュータ３０２と３４８の間に通信リンクを確立する他の手段を使用することもできることを理解されたい。

【００５９】

コンピューティング環境３００に示したようなネットワーク化された環境では、コンピュータ３０２に関して図示したプログラムモジュールまたはその一部分を、リモートメモリ記憶装置に記憶することができる。例えば、リモートコンピュータ３４８の記憶装置上に、リモートアプリケーションプログラム３５８が常駐する。図示の目的上、アプリケーションプログラムおよびオペレーティングシステムなどの他の実行可能プログラムコンポーネントを別個のブロックとして示したが、このようなプログラムおよびコンポーネントは

さまざまな時刻に、コンピューティング装置３０２のさまざまな記憶コンポーネントに存在し、コンピュータのデータプロセッサによって実行されることを理解されたい。

【００６０】

分散ファイルシステム１５０の一実施態様は、１台または数台のコンピュータあるいはその他の装置によって実行されるプログラムモジュールなどのコンピュータ実行可能命令の一般的な文脈で記述することができる。プログラムモジュールは一般に、特定のタスクを実行し、または特定の抽象データ型を実現するルーチン、プログラム、オブジェクト、コンポーネント、データ構造などを含む。典型的には、さまざまな実施形態で、プログラムモジュールの機能を希望に応じて結合し、または分散させることができる。

【００６１】

暗号化されたファイルのファイルフォーマットの一実施態様を、ある形態のコンピュータ読取り可能な媒体上に記憶し、またはある形態のコンピュータ読取り可能な媒体を横断して伝送することができる。コンピュータ読取り可能な媒体は、コンピュータがアクセス可能な任意の使用可能媒体とすることができる。コンピュータ読取り可能な媒体は例えば、「コンピュータ記憶媒体」および「通信媒体」を含む。ただしこれらに限定されるわけではない。

【００６２】

「コンピュータ記憶媒体」は、コンピュータ読取り可能な命令、データ構造、プログラムモジュール、その他のデータなどの情報を記憶するため任意の方法または技術で実現された、揮発性および不揮発性の取外し可能および取外し不能媒体を含む。コンピュータ記憶媒体には、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリまたは他のメモリ技術、ＣＤ－ＲＯＭ、ディジタルバーサタイルディスク（ＤＶＤ）または他の光記憶装置、磁気カセット、磁気テープ、磁気ディスク記憶装置または他の磁気記憶装置、あるいは所望の情報を記憶するのに使用することができ、コンピュータがアクセス可能な他の任意の媒体が含まれる。ただしこれらに限定されるわけではない。

【００６３】

「通信媒体」は一般に、搬送波などの被変調データ信号または他の移送機構中に、コンピュータ読取り可能な命令、データ構造、プログラムモジュールまたは他のデータを具体化する。通信媒体にはさらに任意の情報送達媒体が含まれる。用語「被変調データ信号」は、信号中に情報を符号化するような方法で、その１または複数の特徴が設定または変更された信号を意味する。通信媒体には例えば、有線ネットワーク、直接有線接続などの有線媒体、音響、ＲＦ、赤外線などの無線媒体が含まれる。ただしこれらに限定されるわけではない。上記の任意の媒体の組合せもコンピュータ読取り可能な媒体の範囲に含まれる。

【００６４】

（階層的記憶構造）

分散ファイルシステム１５０は、１または複数の名前空間ルートを含む階層的ファイル記憶構造を使用する。名前空間ルートはそれぞれ、１または複数のディレクトリまたはフォルダサブツリーをサポートすることができ、各サブツリーは、１または複数の追加のサブツリーをサポートすることができる。ディレクトリは、ゼロまたは１つ以上のファイル、および／またはゼロまたは１つ以上の他のディレクトリを保持することができる、シミュレートされたファイルフォルダと見ることができる。サブツリーは、１または複数のディレクトリを指し、ルートを含み（名前空間ルートを含むこともできる）、サブツリーのルートからサブツリーの全てのメンバへのパスがサブツリー自体の中にあるという特性を有する。図４に、ディレクトリＡ、Ｂ、Ｃ、Ｄ、Ｅ、Ｆ、Ｇ、Ｈ、Ｊ、Ｉ、Ｍ、Ｋ、Ｌを含む複数のサブツリーを有する名前空間ルートを含む、例示的な階層的名前空間４００を示す。名前空間ルートのサブツリーには一般に、これよりも多くのディレクトリが含まれるが、説明しやすいように図４には少数のディレクトリだけを示した。

【００６５】

各々のサブツリーは、１台または数台のコンピュータから成る、ディレクトリグループと呼ぶ一群のコンピュータによって管理される。本明細書では主に、サブツリーを管理する

ディレクトリグループとして論じるが、１または複数のディレクトリグループが、名前空間内の任意のディレクトリセットを管理することができる。図２の制御モジュール２５０など、コンピュータの１または複数のモジュールが、そのコンピュータが割り当てられたサブツリーを管理するディレクトリサービスを実現する責任を負う。一実施態様では、各ディレクトリグループが、後に詳細に論じるビザンチンフォルトトレラントグループ（単にビザンチングループとも呼ぶ）である。しかしディレクトリグループが、ビザンチンフォルトトレラントグループである必要はなく、他のグルーピングを使用することもできる。

【００６６】

図４の実線は、ディレクトリ間の関係を示し、どのディレクトリがどのディレクトリのサブディレクトリであるかを識別する。例えば、ディレクトリＣは、ディレクトリＢのサブディレクトリである。あるディレクトリを、そのサブディレクトリの「親」ディレクトリと呼ぶことができる。例えば、ディレクトリＢを、ディレクトリＣの親ディレクトリと呼ぶことができる。

【００６７】

図４の破線の囲みはそれぞれ、特定の破線の中に含まれるディレクトリを管理するディレクトリグループを示す。したがってこの例示的な名前空間４００では、ルート名前空間がディレクトリグループ４０２によって管理され、ディレクトリＡ、Ｂ、Ｃ、ＦおよびＧがディレクトリグループ４０４によって管理され、ディレクトリＤおよびＥがディレクトリグループ４０６によって管理され、ディレクトリＨおよびＪがディレクトリグループ４０８によって管理され、ディレクトリＫ、Ｉ、ＬおよびＭがディレクトリグループ４１０によって管理される。

【００６８】

特定のディレクトリまたは名前空間を管理するディレクトリグループは、そのディレクトリに記憶された各ファイルのディレクトリエントリ、ならびにそのディレクトリ内の各サブディレクトリのディレクトリエントリを維持する責任を負う。ファイルのディレクトリエントリの各々は、分散ファイルシステム１５０内の、そのファイルが記憶された１台または数台のコンピュータを識別する。サブディレクトリのディレクトリエントリの各々は、そサブディレクトリを管理する責任を負ったディレクトリグループを識別する。ディレクトリエントリは、作成、変更およびアクセスのタイムスタンプ、読取り／書込みアクセス制御リスト、複製位置のセット、ファイルサイズなどの追加の情報を含むことができる。

【００６９】

各ディレクトリグループは、名前空間ルートおよび／または名前空間内の１または複数のサブツリーを管理する責任を負う。各ディレクトリグループはさらに、１または複数の追加のサブツリーを識別し、それらの追加のサブツリーの管理責任を他のディレクトリグループに委任することができる。例えば、ディレクトリＤおよびＥは元々、ディレクトリグループ４０４によって管理されていたが、後に、ディレクトリグループ４０６に委任された。

【００７０】

ディレクトリグループはいつでも、サブツリーを別のディレクトリグループに委任することを決定することができる。一実施態様では、この決定が作業負荷に基づき、ディレクトリグループは、自身が過負荷になりつつあると判定したときにサブツリーの委任を決定する。ディレクトリグループは、自身が過負荷になりつつあるいことをさまざまな因子を使用して判定することができ、例示的な一実施態様では、各ディレクトリグループは、予想されるマシン１台あたりの平均ディレクトリ数（例えば１０，０００程度）にほぼ等しいサイズのサブツリーを管理しようとする。

【００７１】

サブツリーを委任する先のディレクトリグループは、さまざまな方法で決定することができる。一実施態様では、委任を実行する側のディレクトリグループが、分散ファイルシス

テム１５０内の知っているコンピュータの中からランダムに選択し、選択したコンピュータを、サブツリーを委任する先の新しいディレクトリグループとして使用する。他のさまざまな因子をこの選択プロセスに影響させることができる（例えば、可用性の低いコンピュータは選択しない、最近にサブツリーを委任したコンピュータは選択しないなど）。

【００７２】

ディレクトリグループは、１または複数のメンバによってディジタル署名された委任証明書を生成することによって、特定のサブツリーを委任することができる。複数のメンバが委任証明書に署名する状況では、署名プロセスをさまざまな形式で実施することができる。一実施態様では、各メンバは、委任証明書のコピーに署名する。他の実施態様では、委任証明書に繰り返し署名される（例えば、１つのメンバが証明書に署名し、次いで、ディジタル署名されたこの証明書に他のメンバが署名する）。ディジタル署名を検証するときに検証者がその順番を知っていさえすれば、異なるメンバが証明書に署名する順番は重要ではない（例えば、署名順を用いて検証者を前もってプログラムしておくこと、または順番を識別する情報を証明書に含めることができる）。以下に、４つの署名者によって繰り返し署名された例示的な証明書を示す。

$\sigma_{S4}(\sigma_{S3}(\sigma_{S2}(\sigma_{S1}(DC))))$

ここで、ＤＣは、ディジタル署名される委任証明書を表し、$\sigma_{Si}()$は、署名者ｉが（）の内容にディジタル署名したことを指示する。

【００７３】

一実施態様では、ディレクトリグループのメンバ（コンピュータ）の数が、設計者が許容できるようにしたいと考える障害コンピュータの数によって決まる。本明細書では障害コンピュータが、アクセス不能のコンピュータ（例えば電源が切られた、または誤動作しているコンピュータ）、または不正使用されたコンピュータ（例えば、悪意あるユーザまたはプログラムがコンピュータにアクセスし、適当な応答を与えるまたは不適切なデータを与えるなど、問合せに対して不適切に応答することができるコンピュータ）を指す。具体的な１つの例では、ｆ台の障害コンピュータを許容するために、ディレクトリグループは、３ｆ＋１台のコンピュータを含む。さらにこの例では、少なくともｆ＋１台のコンピュータが委任証明書にディジタル署名する。

【００７４】

各々の名前空間ルートには、認証局（ＣＡ）から取得した証明書が関連づけられている。認証局は、名前空間の作成を検証する信用機関である。サブツリーに関連づけられた各委任証明書は、カレントサブツリーから、ゼロまたは１つ以上の他のサブツリーを介してＣＡが署名した名前空間ルート証明書までさかのぼる証明書のチェーンを含む。したがって、各委任証明書には、そのサブツリーを管理することを許可されたディレクトリグループであることを（ＣＡが署名した証明書までさかのぼる証明書のチェーンを確立することによって）証明する複数の証明書が関連づけられている。

【００７５】

委任証明書は、さまざまな構成要素を含むことができ、一実施態様では委任証明書が、（１）委任を実行するディレクトリグループが管理しているサブツリーのルートよりも下にある、委任するパスの識別、（２）委任を実行するディレクトリグループに委任されたサブツリーのルートの識別、（３）委任するサブツリーの識別、および（４）サブツリーを委任する先のグループのメンバの識別を含む。サブツリーおよびパスメンバの識別はさまざまであり、実際のディレクトリ名（例えばディレクトリＡ、Ｂ、Ｃ、Ｄなどの名前）、あるいは識別番号（例えば、Globally Unique Identifier（ＧＵＩＤ））とすることができる。識別番号を使用すると、ディレクトリ名が変更された場合に、委任証明書を作成し直す必要性を回避することができる。

【００７６】

図４を参照して委任証明書の例を見てみる。ディレクトリグループ４０２は、グループ４０２が名前空間ルートを管理する権限を有することを証明する証明書をＣＡから取得する。この証明書は以下の形式をとる。

σ_{OurCA}（"Root", $GUID_{Root}$, DG_{402}）　（1）
ここで、σ_{OurCA}は、この証明書がＣＡ「ＯｕｒＣＡ」によって署名されたことを指示し、"Ｒｏｏｔ"は名前空間ルートの名前、$GUID_{Root}$は名前空間ルートのGlobally Unique Identifierであり、DG_{402}は、ディレクトリグループ４０２のメンバの名前（または他の識別子）を表す。

【００７７】
ディレクトリグループ４０２がディレクトリＡから始まるサブツリーをディレクトリグループ４０４に委任すると決定すると、ディレクトリグループ４０２は、ディレクトリグループ４０４のメンバに渡す委任証明書を生成する。この委任証明書は、先の証明書（１）ならびに以下の証明書を含む。

σ_{DG402}（$GUID_{Root}$／Ａ, $GUID_{A}$, DG_{404}）　（2）
ここで、σ_{DG402}は、ディレクトリグループ４０２のメンバがこの証明書に署名したことを指示し、$GUID_{Root}$／Ａは、ディレクトリグループ４０２に委任されたサブツリーのルートのＧＵＩＤ（$GUID_{Root}$）とディレクトリグループ４０４に委任されるパス（／Ａ）、$GUID_{A}$は、委任されるサブツリー（すなわちディレクトリＡから始まるサブツリー）のGlobally Unique Identifierであり、DG_{404}は、ディレクトリグループ４０４のメンバの名前（または他の識別子）を表す。

【００７８】
同様に、ディレクトリグループ４０４がディレクトリＤから始まるサブツリーをディレクトリグループ４０６に委任すると決定すると、ディレクトリグループ４０４は、ディレクトリグループ４０６のメンバに渡す委任証明書を生成する。この委任証明書は、先の証明書（１）および（２）、ならびに以下の証明書を含む。

σ_{DG404}（$GUID_{A}$／Ｂ／Ｃ／Ｄ, $GUID_{D}$, DG_{406}）　（3）
ここで、σ_{DG404}は、ディレクトリグループ４０４のメンバがこの証明書に署名したことを指示し、$GUID_{A}$／Ｂ／Ｃ／Ｄは、ディレクトリグループ４０４に委任されたサブツリーのルートのＧＵＩＤ（$GUID_{A}$）とディレクトリグループ４０６に委任されるパス（／Ｂ／Ｃ／Ｄ）、$GUID_{D}$は、委任されるサブツリー（すなわちディレクトリＤから始まるサブツリー）のGlobally Unique Identifierであり、DG_{406}は、ディレクトリグループ４０６のメンバの名前（または他の識別子）を表す。

【００７９】
図示の例では、委任証明書が、特定のサブツリー内の各ディレクトリに対して発行されるのではなく、委任ポイントで発行される。例えば、委任証明書は、Ａ（サブツリーの最上位ディレクトリ）に対して発行されるのであって、／Ａ／Ｂまたは／Ａ／Ｂ／Ｃに対して発行されるのではない。

【００８０】
図５は、サブツリーの管理責任を他のディレクトリグループに委任する例示的なプロセス５００を示す流れ図である。プロセス５００は、サブツリーの管理責任の委任を実行しているディレクトリグループ内のコンピュータのサブツリー委任モジュール２５４によって実行される。最初に、サブツリーを委任する先の一群のコンピュータを識別する（アクト５０２）。そのサブツリーに対する委任証明書を生成し（アクト５０４）、委任するグループの１または複数のメンバがディジタル署名する（アクト５０６）。次いで、そのサブツリーの管理責任を委任する先のコンピュータ群に、ディジタル署名された委任証明書を発行する（アクト５０８）。

【００８１】
図４に戻る。分散ファイルシステム１５０内の各コンピュータは、名前空間内のパス名のあるサブセットを、そのパス名を管理するディレクトリグループにマップするローカルキャッシュ（例えば図２のキャッシュ２６０）を維持する。例えば、ある特定のコンピュータのキャッシュが、パス名／Ａ、／Ａ／Ｂ、／Ａ／Ｂ／Ｃ、／Ａ／Ｆおよび／Ａ／Ｆ／Ｇを、ディレクトリグループ４０４にマップするマッピングを含む。コンピュータは、それらのキャッシュの中にさまざまなマッピングを有することができるが、典型的には、ディ

レクトリグループ（ディレクトリグループ４０２）を管理する名前空間ルートに、名前空間ルートをマッピングすることを少なくとも含む。

【００８２】

パス名から管理ディレクトリグループへのマッピングを維持することによって、コンピュータが、ディレクトリグループ探索プロセスの少なくとも一部分を、名前空間ルートを管理しているディレクトリグループ（およびおそらく他のディレクトリグループ）にアクセスする必要なしに、それ自体でローカルに実行することができる。例えば、あるコンピュータが、パス名／Ａ／Ｂ／ｆｏｏ．ｔｘｔを有する「ｆｏｏ．ｔｘｔ」と呼ばれるファイルにアクセスすることを希望しており、コンピュータがそのローカルキャッシュに、ディレクトリグループ４０４に対するパス名のマッピングを有すると仮定する。この例では、コンピュータが、ディレクトリＢの中のファイルを管理し、したがってファイルｆｏｏ．ｔｘｔを管理するディレクトリグループ４０４内のメンバを、自身のローカルキャッシュから容易に識別することができる。したがって、コンピュータは、ファイル「ｆｏｏ．ｔｘｔ」の位置を決定するのにどのコンピュータにアクセスすべきか（すなわち、パス名／Ａ／Ｂのディレクトリエントリをどのコンピュータが管理しているか）を、ディレクトリグループ４０２または４０４にアクセスする必要なしに、キャッシュの中の情報に基づいて決定する。

【００８３】

コンピュータがディレクトリグループへのパス名全体をマップするだけの十分な情報を、ローカルキャッシュ中に有していない場合、コンピュータは、そのキャッシュに存在するそのパス名の最も長いプリフィックスのマッピングを見つける。次いでコンピュータは、最も長いプリフィックスの最後のディレクトリを管理するディレクトリグループにアクセスして、パス名の残りの部分および委任証明書を、できるだけ多く管理するディレクトリグループを決定する。ディレクトリグループにアクセスし、委任証明書を取得するこのプロセスは、適当なマッピングが見つかるまで続けられる。

【００８４】

例えば、あるコンピュータが、パス名／Ａ／Ｂ／Ｃ／Ｄ／ｆｏｏ２．ｔｘｔを有する「ｆｏｏ２．ｔｘｔ」と呼ばれるファイルにアクセスすることを希望しており、コンピュータがそのローカルキャッシュに、ディレクトリグループ４０４に対するパス名のマッピングは持っているが、ディレクトリグループ４０６に対しては持っていないと仮定する。コンピュータはパス名を見て、そのキャッシュの中にあるパス名（／Ａ／Ｂ／Ｃ）における最も長いプリフィックスに対するマッピングを見つけ、ディレクトリの管理に責任を負うディレクトリグループ、すなわちディレクトリグループ４０４にアクセスする。コンピュータは、ディレクトリグループ４０４のメンバに、パス名／Ａ／Ｂ／Ｃ／Ｄ／ｆｏｏ２．ｔｘｔの関連サブツリーの委任証明書、すなわちディレクトリグループ４０６の委任証明書を問い合せる。ディレクトリグループ４０４のメンバは、問い合せたコンピュータにこの委任証明書を戻し、このコンピュータは、委任証明書を（例えば署名したコンピュータの公開鍵に基づいて）検証することができる。受け取られた委任証明書は、ディレクトリ／Ｄの管理に責任を負うディレクトリグループを識別し、そのためコンピュータは、ファイル「ｆｏｏ２．ｔｘｔ」がある場所を決定するために、そのディレクトリグループにアクセスすればよいことを知る。このように、ファイル「ｆｏｏ２．ｔｘｔ」の位置を決定するためにどのコンピュータにアクセスすべきかを決定することには、ディレクトリグループ４０４のメンバにアクセスすることが含まれるが、この決定のためにディレクトリグループ４０２のメンバへアクセスする必要はない。

【００８５】

図６は、特定のパス名の管理に責任を負うディレクトリグループを探索する例示的なプロセス６００を示す流れ図である。プロセス６００は、探索中のパス名にアクセスすることを希望するコンピュータのディレクトリグループ探索モジュール２２２（図２）によって実行される。最初に、ディレクトリグループへのマッピングのローカルキャッシュにアクセスし（アクト６０２）、このキャッシュにパス名の最も長いプリフィックスに対するマ

ッピングを見つけ出す（アクト６０４）。次いで、パス名全体がマップされているかどうかに基づいて処理を続行する（アクト６０６）。パス名全体がマップされている場合、ディレクトリグループ探索プロセスは完了となる（アクト６０８）。しかし、パス名全体がマップされていない場合には、パス名の最後にマップされたプリフィックスを管理するグループのメンバから、関連サブツリーの委任証明書を取得する（アクト６１０）。次いで、受け取った委任証明書を検証する（アクト６１２）。委任証明書が正しく検証されないか、または委任証明書を取得することができない場合には、アクト６１０へ戻り、グループの別のメンバを選択して問い合せる。最後にマップされたプリフィックスを管理しているグループの少なくとも１つのメンバが正常に機能している限り、このプロセスは最後には成功する。最後にマップされたプリフィックスを管理しているグループのメンバが１つも正常に機能していない場合、プロセスは、有効なより短いプリフィックスを探すか、または名前空間ルートへ戻る。委任証明書のチェーンが検証されたら、証明書からのパス名マッピング情報をローカルキャッシュに追加する（アクト６１４）。次いで、プロセスはアクト６０６へ戻る。このときの最も長いプリフィックスは、以前の最も長いプリフィックスに新しい関連サブツリー情報を連結したものになる（例えば、パス名が／Ａ／Ｂ／Ｃ／Ｄ／Ｅ／Ｆ、以前の最も長いプリフィックスが／Ａ／Ｂ、新しい関連サブツリーが／Ｃ／Ｄである場合、新しい最も長いプリフィックスは／Ａ／Ｂ／Ｃ／Ｄとなる）。次いで、パス名全体がマップされるまで、アクト６０６、６１０、６１２および６１４を繰り返す。異なるディレクトリの管理を異なるディレクトリグループ上へ分離することによって、管理責任は、異なる複数のコンピュータに分散する。これによって、特定のコンピュータ、特に名前空間ルートのディレクトリグループおよび名前空間ルートに最も近いディレクトリグループのコンピュータにかかる管理負担は低減する。例えば、特定のパス名をルートノードから解析する必要はなく、そのローカルキャッシュを介して、そのパス名の途中までは拾い上げることができる。

【００８６】

（ディレクトリおよびファイルの複製および記憶）

図１の分散ファイルシステム１５０は、ディレクトリエントリの記憶とディレクトリエントリに対応するファイルの記憶を異なる方法で管理する。システム１５０内に記憶するファイルは複製され、システム１５０内の異なる複数のコンピュータに保存される。さらに、そのファイルに対するディレクトリエントリが生成され、ビザンチンフォルトトレラントグループの一部分であるシステム１５０内の異なる複数のコンピュータに保存される。ディレクトリエントリは、後に詳細に論じるように、ファイルが保存されるよりも多くのコンピュータに保存される。

【００８７】

ファイルおよびディレクトリエントリを記憶する本明細書に記載のさまざまな処理を、先に論じた階層的記憶構造とともに使用することができる。しかし、ファイルおよびディレクトリエントリを記憶する本明細書に記載のさまざまな処理を、階層的記憶構造を使用しないシステムで使用することもできる。

【００８８】

ビザンチンフォルトトレラントグループは、たとえそのうちのいくつかのコンピュータに障害（故障または他の理由で使用できない）があっても、情報を記憶し、かつ／または他のアクションを実行するのに使用することができる一群のコンピュータである。コンピュータは、コンピュータを操作する悪意のユーザ、コンピュータ上で走る悪意のプログラムなど、さまざまな方法で危険にさらされる。これらのコンピュータからは、リクエストへの応答を拒否する、リクエストに対して不正確なまたは不要な情報で故意に応答するなど、任意のタイプの振舞いを観察することができる。このようなコンピュータの存在にもかかわらず、ビザンチンフォルトトレラントグループは、情報を正確に記憶し、かつ／または他のアクションを実行することができる。ビザンチングループは当業者に周知であり、したがって、本発明に関する場合を除き、さらに論じることはしない。

【００８９】

当業者には周知のとおり、あるタイプの計算では、ｆ台の故障コンピュータの存在にもかかわらず（故障コンピュータは危険にさらされ、またはパワーダウンなどの他の理由で使用不能である）、正確に動作できるようにするためには、ビザンチンフォルトトレラントグループが、少なくとも３ｆ＋１台のコンピュータを含んでいなければならない。分散ファイルシステム１５０では、ディレクトリエントリが、ビザンチンフォルトトレラントグループの３ｆ＋１台のコンピュータに記憶され、ファイル自体は、ｆ＋１台のコンピュータ（ディレクトリエントリが記憶された同じコンピュータのうちの１台または数台とすることができる）に記憶される。

【００９０】

図７に、ファイルおよび対応するディレクトリエントリをサーバレス分散ファイルシステム内に記憶する例示的な記憶を示す。ファイルシステム７００（例えば図１のサーバレス分散ファイルシステム１５０）は、１２台のコンピュータ７０２、７０４、７０６、７０８、７１０、７１２、７１４、７１６、７１８、７２０、７２２および７２４を含む。システム７００の設計者が、２台のコンピュータの故障を許容できるようにしたいと考えているとすると、ビザンチンフォルトトレラントグループは、少なくとも７台（（３×２）＋１）のコンピュータを含まなければならない。コンピュータ７０２～７１４を含むビザンチングループ７２６が示されている。

【００９１】

ファイル７２８をファイルシステム７００に記憶するときには、対応するディレクトリエントリ７３０が、適当なディレクトリグループ（ファイル７２８のパス名に基づいてファイルが記憶されたディレクトリの管理に責任を負うディレクトリグループ）内のコンピュータによって記憶される。ディレクトリエントリ７３０に対する図７のディレクトリグループは、ビザンチングループ７２６であり、そのため、ディレクトリエントリ７３０は、ビザンチングループ７２６の正常に機能している各々のコンピュータ７０２～７１４上に記憶される。したがってディレクトリエントリ７３０は、最大７台の異なるコンピュータ上に記憶される。一方、ファイル７２８は複製され、３台のコンピュータ（コンピュータ７１６、７２０および７２４）にそれぞれ記憶される。図示のとおり、ファイル７２８を記憶したコンピュータが、ビザンチングループ７２６のコンピュータである必要はなく、一般に、ビザンチングループ７２６のコンピュータではない（ただし、任意選択で、ファイル７２８が記憶された１台または数台のコンピュータがビザンチングループ７２６のコンピュータであってもよい）。

【００９２】

各ディレクトリエントリは、対応するファイルの名前、ファイルが記憶されたコンピュータの識別、およびファイルの内容がディレクトリエントリに対応するかどうかを検証することを可能にするファイル検証データを含む。ファイル検証データは、異なるさまざまな形式をとることができ、一実施態様では、ファイル検証データが、ＭＤ５（Message Digest5）、ＳＨＡ－１（Secure Hash Algorithm-1）などの暗号法上安全なハッシュ関数をファイルに適用することによって生成されたハッシュ値である。記憶装置からファイルを検索するとき、検索するコンピュータは、ハッシュ値を再生成し、その値をディレクトリエントリ中のハッシュ値と比較して、コンピュータが正確なファイルを受け取ったことを検証することができる。他の実施態様では、ファイル検証データが、ファイル識別番号（例えばファイルの一意識別子）、ファイルのバージョン番号、およびファイル上に署名があるユーザの名前を結合したものである。

【００９３】

図８は、サーバレス分散ファイルシステムにファイルを記憶する例示的なプロセスを示す流れ図である。最初に、クライアントコンピューティング装置が新しいファイル記憶要求を受け取る（アクト８０２）。クライアントは、ファイルおよびファイル名を暗号化し、ファイル内容のハッシュを生成する（アクト８０４）。クライアントは、暗号化されたファイル名およびファイル内容ハッシュを、ディレクトリエントリを作成するよう求める要求とともに、適当なビザンチンフォルトトレラントディレクトリグループに送る（アクト

８０６）。ディレクトリグループは、要求の有効性を、例えば、ファイル名が既存の名前と競合しないこと、およびクライアントが要求の内容を実行する許可を有することを検証することによって、調べる（アクト８０８）。要求が有効と認められなかった場合、要求は失敗に終わる（アクト８１０）。しかし、要求が有効と認められた場合には、ディレクトリグループが、この新しいファイルのディレクトリエントリを生成する（アクト８１２）。ディレクトリグループはさらに、この新しいファイルの複製セットを決定し、その複製セットを、新しく生成されたディレクトリエントリに追加する（アクト８１４）。ファイルの複製も生成し（アクト８１６）、ファイルシステム内の複数のコンピュータに保存する（アクト８１８）。

【００９４】

ビザンチングループにディレクトリエントリを記憶し、このエントリにファイル検証データを含めることによって、フォルトトレランス（最大ｆ故障まで）が維持される。しかし、ファイルをディレクトリとは別に記憶し、ファイルへのアクセスにビザンチン操作を使用しないことによって、記憶空間の必要量およびビザンチン操作が低減される。例えば、ディレクトリエントリは、１００バイト程度であるのに対して、ファイル自体は、数千または数十億バイトにもなる。

【００９５】

（ディレクトリおよびファイルのロック機構）

図１の分散ファイルシステム１５０の各オブジェクト（例えばディレクトリおよびファイル）は、一組の専用ロックに関連づけられている。これらのロックは、アプリケーションが実行したいオペレーションのタイプに基づいて、そのオペレーションを実行するためにアプリケーションがディレクトリまたはファイルを開くことができるか否かを決定するのに使用される。ロックは、ロックのタイプおよび競合のレベルによって決まる特定の時間範囲を有するリース（lease）と見ることができる。例えば、書込みロックの時間範囲は数分であり、一方、読取りロックの時間範囲は数日にもなる。アプリケーションが、オブジェクトに対してオペレーションを実行したいとき、このアプリケーションを実行中のクライアントコンピュータは、そのオペレーションを実行するのに必要なロックをすでに有しているかどうかを調べる。必要なロックを持っていない場合には、そのオブジェクトの管理に責任を負うディレクトリグループから適当なロックを要求する。所望のオペレーションを実行し終えると、アプリケーションは任意選択で、取得したロックを解放し、あるいはロックが自動的に期限切れになるか、または管理ディレクトリグループによってロックがリコールされるまで、ロックを保持する。

【００９６】

特定のディレクトリに対して、そのディレクトリを実装するビザンチンフォルトトレラントグループが、そのディレクトリ内の全てのファイル、そのディレクトリのサブディレクトリの名前、およびそのディレクトリ自体を削除する権利に対するロックを制御する。ロック機構は、適当なファイルおよびディレクトリの幅広い（細分性の粗い）ロックを、要求しているクライアントコンピュータに許可しようとし、クライアントコンピュータは、ロックを取得するのに複数のビザンチンメッセージを要求するのではなく、１回のビザンチンロックの取得で多くの読取りおよび／または更新を処理することができる。

【００９７】

図示の例では、ロック機構が１０個の異なるロック、すなわち読取り、書込みク、オープン読取り（Open Read）、オープン書込み（Open Write）、オープン削除（Open Delete）、非共用読取り（Not Shared Read）、非共用書込み（Not Shared Write）、非共用削除（Not Shared Delete）、挿入、および排他のロックを使用する。読取りおよび書込みロックは、オブジェクト中のデータ（例えばファイルの内容）へのアクセスを制御するのに使用される。オープン読取り、オープン書込み、オープン削除、非共用読取り、非共用書込み、および非共用削除ロックは、オブジェクトのオープンを制御するのに使用される。挿入および排他ロックは特殊用途のロックである。これらの１０個のロックについては後にさらに詳細に論じる。アプリケーションは、実行したいオペレーションに応じて、これ

らのロックの中から適当なものを要求する。

【００９８】

［読取りロック］読取りロックは、アプリケーションが関連ファイルを読み取ることができるようにアプリケーションが要求する。読取りロックは、書込みロックとともに、ディレクトリグループがファイル中のデータを矛盾なしに保つことを可能にする。

【００９９】

［書込みロック］書込みロックは、アプリケーションが関連ファイルに書き込む（ファイルを更新するとも言う）ことができるようにアプリケーションが要求する。書込みロックは、読取りロックとともに、ディレクトリグループがファイル中のデータを矛盾なしに保つことを可能にする。

【０１００】

アプリケーションがオブジェクトを開きたいとき、ディレクトリグループは、２つのチェックを実行する。（１）アプリケーションが求めているモードが、すでにそのオブジェクトを開いている他のアプリケーションと競合するかどうか、および（２）アプリケーションがオブジェクトを共用する気があるオペレーションが、他のアプリケーションがそのオブジェクトをすでに開き、そのオブジェクトを共用する気があることを指示したオペレーションと競合するかどうかをチェックする。１０個のロックのうち、オープン読取り、オープン書込み、オープン削除、オープン非共用読取り、オープン非共用書込み、およびオープン非共用削除の６つのロックが、このチェックをサポートすることに向けられる。これらのロックは、オブジェクトを開く能力をアプリケーションに許可するのに使用されるが、そのオブジェクトのデータを取得できることを保証するとは限らない（データにアクセスするためには、（アプリケーションが実行したいオペレーションのタイプに応じて）読取りロックまたは書込みロックを取得する）。

【０１０１】

［オープン読取りロック］オープン読取りロックは、読取りのためアプリケーションが関連オブジェクトを開くことができるよう、アプリケーションが要求する。

【０１０２】

［オープン書込みロック］オープン書込みロックは、書込みのためアプリケーションが関連オブジェクトを開くことができるよう、アプリケーションが要求する。

【０１０３】

［オープン削除ロック］オープン削除ロックは、削除のため、アプリケーションが関連オブジェクトを開くことができるよう、アプリケーションが要求する。

【０１０４】

［オープン非共用読取りロック］オープン非共用読取りロックは、アプリケーションがオブジェクトを読み取る能力を他のアプリケーションと共用したくないときにアプリケーションが要求する。

【０１０５】

［オープン非共用書込みロック］オープン非共用書込みロックは、アプリケーションがオブジェクトに書き込む能力を他のアプリケーションと共用したくないときにアプリケーションが要求する。

【０１０６】

［オープン非共用削除ロック］オープン非共用削除ロックは、アプリケーションがオブジェクトを削除する能力を他のアプリケーションと共用したくないときにアプリケーションが要求する。

【０１０７】

サポートされる他の２つのロックが挿入ロックおよび排他ロックである。

【０１０８】

［挿入ロック］挿入ロックは、ディレクトリ内のオブジェクトに対して特定の名前を作成するためにアプリケーションが要求する。挿入ロックの許可は、特定の名前を有するオブジェクトを作成する許可をアプリケーションに与える。挿入ロックは、同じオブジェクト

名を有する他の挿入ロック、およびディレクトリの排他ロックと競合する。

【０１０９】

［排他ロック］排他ロックは、上述した９つのロック全てを取得するためにアプリケーションが要求し、ディレクトリ内に存在することができる（しかしまだ存在しない）各々の可能な名前の挿入ロックを含む。ディレクトリの排他ロックは、そのディレクトリのファイルまたはサブディレクトリの排他ロックを意味せず、ディレクトリの名前空間だけを意味する。排他ロックは、先に論じた９つのロックのそれぞれと競合する。

【０１１０】

異なるさまざまなロック間には、さまざまな競合が存在する。表１は、例示的な一実施態様におけるロック間の競合を示す競合マトリックスである。表１では以下の省略形を使用した：Ｉｎｓ（挿入）、Ｅｘｃｌ（排他）、Ｏ－Ｒ（オープン読取り）、Ｏ－Ｗ（オープン書込み）、Ｏ－Ｄ（オープン削除）、Ｏ－！Ｒ（オープン非共用読取り）、Ｏ－！Ｗ（オープン非共用書込み）およびＯ－！Ｄ（オープン非共用削除）。表１の欄の中の「Ｘ」は、対応する２つのロック間の競合を示す。例えば、オープン読取りは、オープン非共用読取りと競合するが、オープン非共用書込みとは競合しない。

【０１１１】

【表１】

表１

	Ins	Read	Write	Excl	O-R	O-W	O-D	O-!R	O-!W	O-!D
Ins	X	X	X	X						
Read	X		X	X						
Write	X	X	X	X						
Excl	X	X	X	X	X	X	X	X	X	X
O-R				X				X		
O-W				X					X	
O-D				X						X
O-!R				X	X					
O-!W				X		X				
O-!D				X			X			

【０１１２】

図９は、特定のオブジェクトを開くことを許可するかどうか決定する例示的なプロセスを示す流れ図である。図９のプロセスは、その特定のオブジェクトの管理に責任を負うディレクトリグループによって実現される。図９のプロセスでは、特定のオブジェクトを開くことを要求しているクライアントが、所望のオブジェクトを開くのに必要なロックをまだ持っていないと仮定する。最初に、識別された特定のロックを有するオブジェクトにアクセスするリクエストを受け取る（アクト９０２）。選択されたロックが意味するモードが、別のクライアントに許可されたロックと競合するかどうかに関するチェックが、ディレクトリグループによって実施される（アクト９０４）。例えば、このリクエストが、読取りのためオブジェクトを開くリクエストであり、他のアプリケーションが、非共用読取りロックを有するこのオブジェクトをすでに開いている場合、選択されたロックが意味するモード（オープン読取り）は、そのオブジェクトをすでに開いている他のアプリケーションと競合する。ディレクトリグループは、競合するロックをクライアントに発行したかどうかを知っているだけで、そのクライアントが現在、オブジェクトへのアプリケーションのアクセスを可能にするためにそのロックを使用しているかどうかを知らないので、いくつかのケースでは、アクト９０４のチェックを実施するのに、ロックを現在保持している

クライアントがそれを放棄する気があるかどうかを尋ねる必要がある。

【０１１３】

アクト９０４のチェックで競合が識別されなかった場合、要求されたロックが許可され、選択したロックを有するファイルをアプリケーションが開くことができるようになり（アクト９０６）、アクト９０２のリクエストが許可される。次いで、これらのロックが許可されたこと、およびそれら許可されたクライアントが、ディレクトリグループによって保存され（アクト９０８）、後続のリクエストに対して競合を決定し、必要に応じてロックのリコールを試みることができる。

【０１１４】

しかし、アクト９０４のチェックで競合が識別された場合には、競合するロックを保持しているクライアントに、それらを戻すよう求めるリクエストが発行される（アクト９１０）。次いで、要求されたロックが全て戻されたかどうかに関するチェックが実施される（アクト９１２）。要求されたロックが全て戻された場合には、要求のロックが許可され、選択したロックを有するファイルをアプリケーションが開くことができるようになり（アクト９０６）、これらのロックが記録される（アクト９０８）。一方、要求されたロックのうち戻されないロックがある場合には、ディレクトリグループは、このオープンリクエストを拒絶する（アクト９１４）。

【０１１５】

１台だけのクライアントコンピュータが名前空間のある領域にアクセスするときに性能を向上させる試みにおいて、ファイルシステム１５０は、アプリケーション（またはクライアント）が近い将来に、追加の関連ロックを要求する可能性が高いとの前提のもとに、クライアント上で実行中のアプリケーションが要求するよりも広い範囲のロックを発行することができる。例えば、アプリケーションがファイル／Ａ／Ｂ／Ｃ／ｆｏｏ．ｔｘｔを開く場合、クライアントは、このファイルに対するロックを要求する。ディレクトリグループがそのロックを許可する場合、ディレクトリグループはそのロックを、／Ａ／Ｂ／Ｃのディレクトリロックにアップグレードすることができる（例えば、過去の実施に基づいて、ディレクトリ上の競合がまれであるとディレクトリグループが判定した場合）。次いでアプリケーションが、同じディレクトリ内の別のファイルを開く場合、クライアントは、ディレクトリグループから他のロックを要求する必要なしにそのファイルを開くことができる。

【０１１６】

クライアントのロックリクエストが、他のクライアントに許可された既存のロックと競合した場合、ディレクトリグループは、（例えば、アクト９１４でリクエストを拒絶するのではなく）アクト９１０で、新しいリクエストと競合しないロックに、以前に発行されたロックをダウングレードすることを試みることができる。ロックのアップグレードの結果、クライアントは、要求していないロックを保持するので、ロックのダウングレードが成功する可能性は一般に低くない。このロックリコールに失敗した場合、リクエストは拒絶される。

【０１１７】

ファイルシステム内のオブジェクトにさまざまなオペレーションを実行することができる。表２に、より一般的ないくつかのオペレーション、およびオペレーションを実行するためにアプリケーションが要求するロックを記載する。

【０１１８】

【表２】

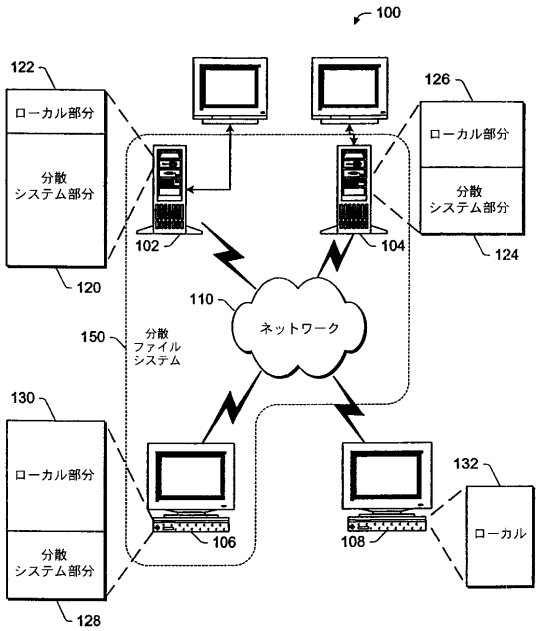
表２

オペレーション	説明
オブジェクト読取り	ディレクトリまたはファイルを読み取るためのリクエスト。オブジェクトのオープン読取りロックおよびそれに続く読取りロックを必要とする。アプリケーションは任意選択で、任意のオープン非共有ロックを要求することができる。
オブジェクト書込み／更新	ファイルに書き込むためのリクエスト。オブジェクトのオープン書込みロックおよびそれに続く書込みロックを必要とする。要求されたならば、アプリケーションは任意選択で、任意のオープン非共有ロックを要求することができる。
ファイル削除	ファイルを削除するためのリクエスト。オープン削除および書込みロックを必要とする。通常、アプリケーションはさらに、全てのオープン非共有ロックを要求する。
ディレクトリ削除	ディレクトリを削除するためのリクエスト。ディレクトリに対する排他ロックを必要とする。ディレクトリを削除できるのはディレクトリが空のときだけである。
ディレクトリ名変更	ディレクトリ名を変更するためのリクエスト。親ディレクトリの排他ロック（名前を変えられているディレクトリ名が変更されるディレクトリはサブディレクトリである）、およびあて先ディレクトリ中の新しいディレクトリ名に対する挿入ロックを必要とする。名前変更が複数のディレクトリにまたがる場合、新しい親ディレクトリに対する挿入ロックが必要となる。
ファイル名変更	ディレクトリ中のファイルの名前を変更するためのリクエスト。ファイルの書込みロック、およびディレクトリ（名前変更が複数のディレクトリにまたがる場合には別のディレクトリとすることができる）中の新しい名前に対する挿入ロックを必要とする。
オブジェクト作成	新しいファイルまたはディレクトリを作成するためのリクエスト。新しい名前に対する挿入ロックを必要とする。

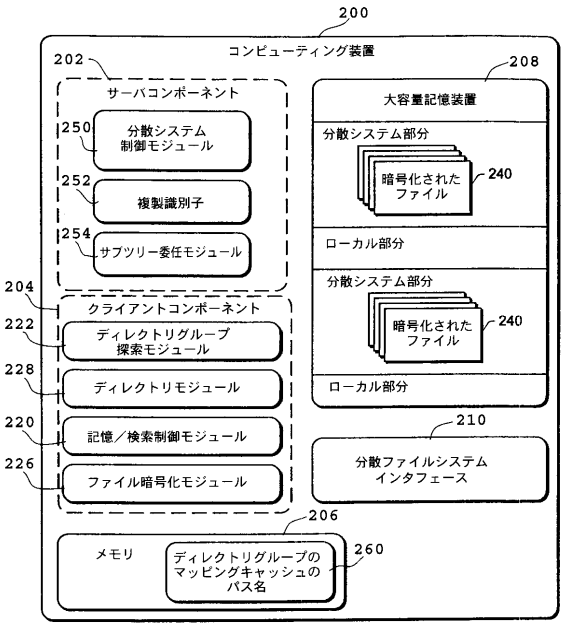
【０１１９】
ファイルへの変更は、そのコンピュータによってローカルに実施され、次いでファイルは（暗号化された後に）は、そのファイルの管理に対して責任を負うディレクトリグループにプッシュバックされる。この情報は、ディレクトリグループ内のさまざまなコンピュータに記憶され、更新されたファイルは、適当なコンピュータに記憶される。
【０１２０】
（結語）
以上の説明では、構造上の特徴および／または方法論的行為に特有の言葉を使用したが、添付の請求項に定義した発明は、記載した特定の特徴または行為に限定されないことを理解されたい。これらの特定の特徴および行為は、本発明を実現する例示的な形態として開示したものである。
【図面の簡単な説明】
【図１】サーバレス分散ファイルシステムをサポートする例示的なネットワーク環境を示

す図である。
【図２】分散ファイルシステムに参加している図１の装置のうちの１台を表す例示的なコンピューティング装置の論理的構成要素を示す図である。
【図３】図１の分散ファイルシステムを実現するのに使用される、より一般的なコンピュータ環境を示す図である。
【図４】複数のサブツリーを有する名前空間ルートを含む、例示的な階層的名前空間を示す図である。
【図５】サブツリーの管理責任を他のディレクトリグループに委任する例示的なプロセスを示す流れ図である。
【図６】特定のパス名の管理に責任を負うディレクトリグループを探索する例示的なプロセスを示す流れ図である。
【図７】サーバレス分散ファイルシステム内でのファイルおよび対応するディレクトリエントリの例示的な記憶を示す図である。
【図８】サーバレス分散ファイルシステム内にファイルを記憶する例示的なプロセスを示す流れ図である。
【図９】特定のオブジェクトのオープンを許可するか否かを決定する例示的なプロセスを示す流れ図である。
【符号の説明】
１００　ネットワーク環境
１０２，１０４，１０６，１０８　クライアントコンピューティング装置
１１０　データ通信ネットワーク
１２０，１２４，１２８　分散システム部分
１２２，１２６，１３０，１３２　ローカル部分
１５０　分散ファイルシステム
７００　ファイルシステム
７０２，７０４，７０６，７０８，７１０，７１２，７１４，７１６，７１８，７２０，７２２，７２４　コンピュータ
７２６　ビザンチングループ
７２８　ファイル
７３０　ディレクトリエントリ

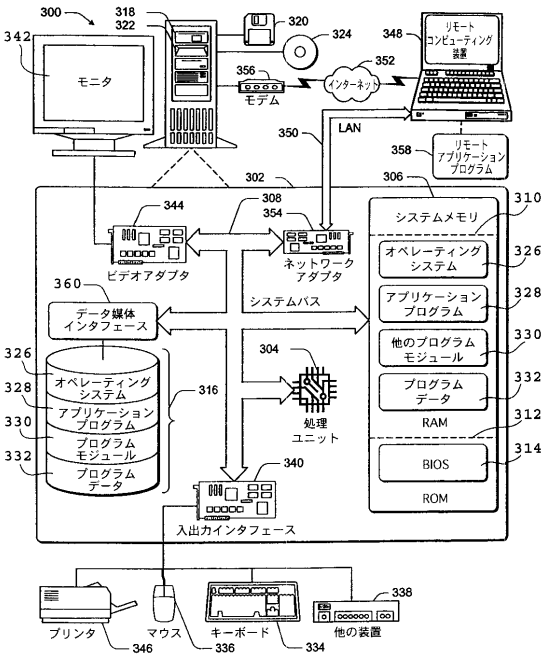
【図１】


100
122
ローカル部分
分散
システム部分
120
126
ローカル部分
分散
システム部分
124
102
104
110
ネットワーク
150
分散
ファイル
システム
130
ローカル部分
分散
システム部分
128
106
108
132
ローカル

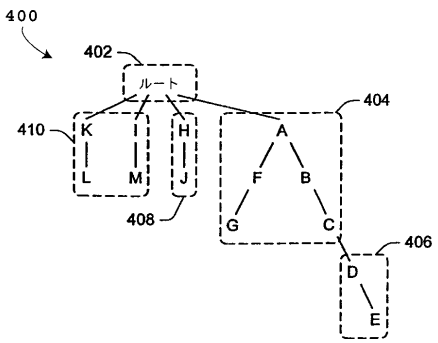
【図２】


200
コンピューティング装置
202
サーバコンポーネント
250
分散システム
制御モジュール
252
複製識別子
254
サブツリー委任モジュール
204
クライアントコンポーネント
222
ディレクトリグループ
探索モジュール
228
ディレクトリモジュール
220
記憶／検索制御モジュール
226
ファイル暗号化モジュール
206
メモリ
260
ディレクトリグループの
マッピングキャッシュの
パス名
208
大容量記憶装置
分散システム部分
240
暗号化された
ファイル
ローカル部分
分散システム部分
240
暗号化された
ファイル
ローカル部分
210
分散ファイルシステム
インタフェース

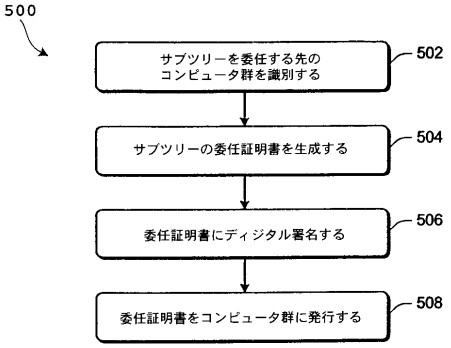
【図３】


300
318
322
320
324
348
リモート
コンピューティング
装置
342
モニタ
356
モデム
352
インターネット
350
LAN
358
リモート
アプリケーション
プログラム
302
306
344
308
354
システムメモリ
310
ビデオアダプタ
ネットワーク
アダプタ
オペレーティング
システム
326
360
データ媒体
インタフェース
システムバス
アプリケーション
プログラム
328
他のプログラム
モジュール
330
304
プログラム
データ
332
326
328
330
332
オペレーティング
システム
アプリケーション
プログラム
プログラム
モジュール
プログラム
データ
316
処理
ユニット
RAM
312
BIOS
314
ROM
340
入出力インタフェース
338
プリンタ
マウス
キーボード
他の装置
346
336
334

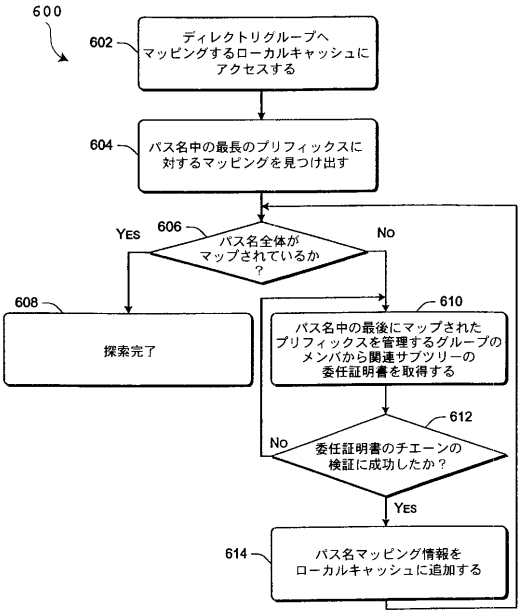
【図４】


400
402
ルート
404
410
K
L
M
H
J
A
F
B
G
C
408
406
D
E

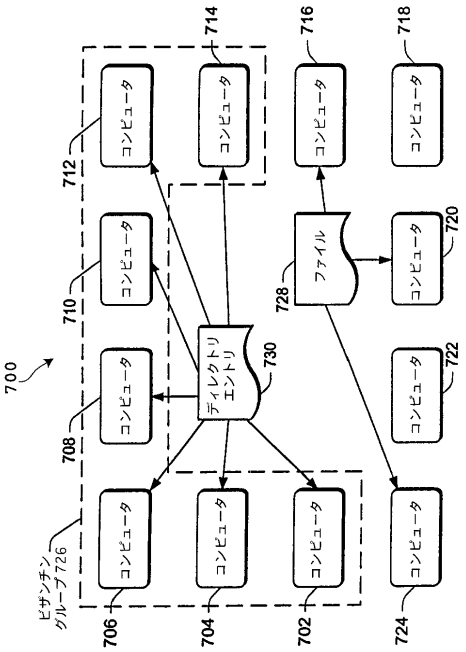
【図５】


500
サブツリーを委任する先の
コンピュータ群を識別する
502
サブツリーの委任証明書を生成する
504
委任証明書にディジタル署名する
506
委任証明書をコンピュータ群に発行する
508

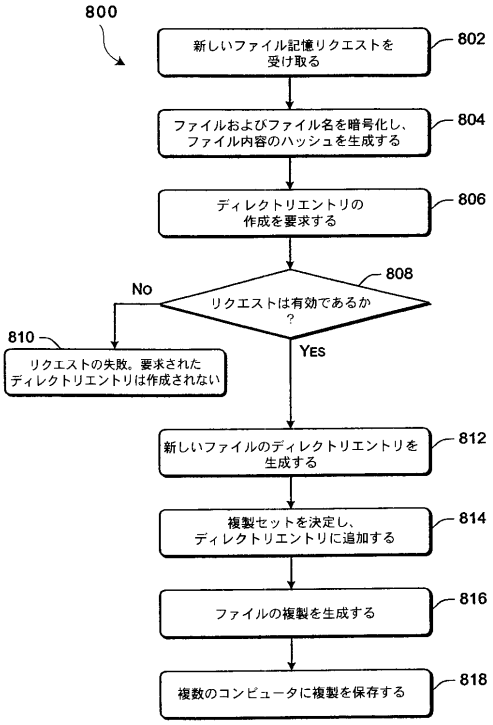
【図６】


600
602
ディレクトリグループへ
マッピングするローカルキャッシュに
アクセスする
604
パス名中の最長のプリフィックスに
対するマッピングを見つけ出す
606
パス名全体が
マップされているか
？
YES
No
608
探索完了
610
パス名中の最後にマップされた
プリフィックスを管理するグループの
メンバから関連サブツリーの
委任証明書を取得する
612
委任証明書のチエーンの
検証に成功したか？
No
YES
614
パス名マッピング情報を
ローカルキャッシュに追加する

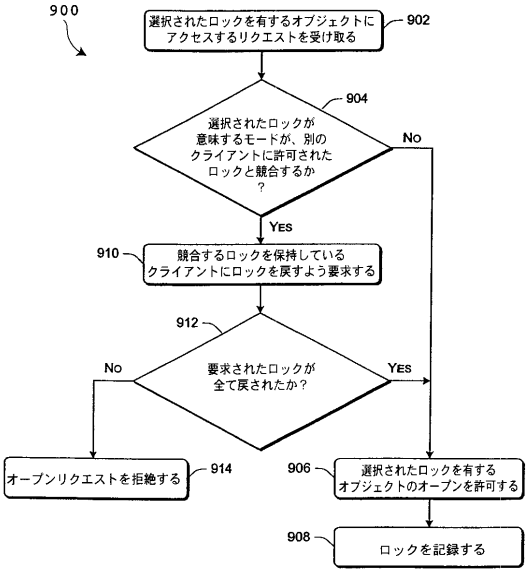
【図７】


700
コンピュータ
ディレクトリ
エントリ
730
ファイル
728
ビザンチン
グループ726
712
714
716
718
710
720
708
722
706
704
702
724

【図８】


800
新しいファイル記憶リクエストを
受け取る
802
ファイルおよびファイル名を暗号化し、
ファイル内容のハッシュを生成する
804
ディレクトリエントリの
作成を要求する
806
リクエストは有効であるか
？
808
No
YES
810
リクエストの失敗。要求された
ディレクトリエントリは作成されない
新しいファイルのディレクトリエントリを
生成する
812
複製セットを決定し、
ディレクトリエントリに追加する
814
ファイルの複製を生成する
816
複数のコンピュータに複製を保存する
818

【図９】


900
選択されたロックを有するオブジェクトに
アクセスするリクエストを受け取る
902
選択されたロックが
意味するモードが、別の
クライアントに許可された
ロックと競合するか
？
904
No
YES
競合するロックを保持している
クライアントにロックを戻すよう要求する
910
要求されたロックが
全て戻されたか？
912
No
YES
オープンリクエストを拒絶する
914
選択されたロックを有する
オブジェクトのオープンを許可する
906
ロックを記録する
908

フロントページの続き

(72)発明者 ウィリアム ジェイ. ボロスキー
アメリカ合衆国 ９８０２７ ワシントン州 イサコア サウスイースト ミラーモント ドライブ ２４６２２
(72)発明者 ジェラルド サーマック
アメリカ合衆国 ９８０１１ ワシントン州 ボズエル ノースイースト １４１ ストリート ８９０８
(72)発明者 ジョン アール. デューサー
アメリカ合衆国 ９８００７ ワシントン州 ベルビュー ノースイースト １６ ストリート １４７０５
(72)発明者 マービン エム. セイマー
アメリカ合衆国 ９８００６ ワシントン州 ベルビュー １３７ アベニュー サウスイースト ４４４０
(72)発明者 ロバート ピー. ワッテンホファー
スイス シーエイチ－８００６ チューリッヒ クラウジウスシュトラーセ ３５

審査官 工藤 嘉晃

(56)参考文献 GEORGE COULOURIS 他 著／水野 忠則 他 訳, 分散システム, 日本, 株式会社電気書院, １９９７年１０月３１日, 第2版, p.785-p.827

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