

(19) 日本国特許庁(JP)

(12) 公表特許公報(A)

(11) 特許出願公表番号

特表2007-509409

(P2007-509409A)

(43) 公表日 平成19年4月12日(2007.4.12)

(51) Int. Cl.	F I	テーマコード (参考)
G06F 12/00 (2006.01)	G06F 12/00 545A	5B082
	G06F 12/00 520P	
	G06F 12/00 518A	

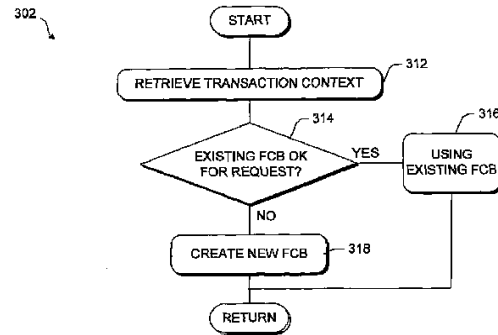
審査請求 未請求 予備審査請求 未請求 (全 60 頁)

(21) 出願番号 特願2006-536567 (P2006-536567) (86) (22) 出願日 平成16年7月22日 (2004.7.22) (85) 翻訳文提出日 平成17年5月16日 (2005.5.16) (86) 国際出願番号 PCT/US2004/023642 (87) 国際公開番号 W02005/045738 (87) 国際公開日 平成17年5月19日 (2005.5.19) (31) 優先権主張番号 10/693,658 (32) 優先日 平成15年10月24日 (2003.10.24) (33) 優先権主張国 米国 (US)	(71) 出願人 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 9805 2-6399 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 プラディーブ ヤナ マダバラブ アメリカ合衆国 98052 ワシントン 州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーシ ョン内 最終頁に続く
--	---

(54) 【発明の名称】 ネットワーク上のトランザクション処理ファイルオペレーションのための方法およびシステム

(57) 【要約】

ネットワーク上でトランザクション処理リモートファイルオペレーションを実行するシステムは、クライアントおよびサーバを含む。このクライアントとサーバは、それぞれトランザクションマネージャ (TM) とファイルシステム (FS) を含む。クライアントはリダイレクタ (RDR) も含むが、サーバはサーバアプリケーション (SRV) を含む。RDRは、リモートトランザクション処理ファイルオペレーションを求める要求を受信する。それに応答して、RDRはこの要求からトランザクションを検索する。RDRは、TMを使用してサーバに送信するためにこのトランザクションをマーシャリングすることができる。RDRは、ネットワーク上でトランザクションをサーバに対して送信する。SRVコンポーネントは、トランザクションを受信し、次いでサーバのTMおよびFSはこのトランザクションを使用してファイルオペレーションを実行する。次いで、サーバはネットワークを介してファイルオペレーション結果をクライアントに対して戻す。



【特許請求の範囲】

【請求項 1】

ローカルデバイスとリモートデバイスとの間でトランザクション処理リモートファイルオペレーションをサポートするシステムであって、

トランザクションマネージャと、

要求を受信して、前記リモートデバイス中に存在するファイルに対してファイルオペレーションを実行するリダイレクタであって、前記ローカルデバイスおよび前記リモートデバイスはネットワークに接続され、前記リダイレクタは、トランザクション内において前記ネットワーク上で前記リモートデバイスに前記要求を送信することと、
を備えることを特徴とするシステム。

10

【請求項 2】

前記トランザクションマネージャは、ファイルシステムに一体化されていないことを特徴とする請求項 1 に記載のシステム。

【請求項 3】

前記リダイレクタは、前記ファイルに関連付けられたファイルシステム識別子 (F i d) およびバージョン識別子を含む、前記リモートデバイスからの受信ファイル情報であることを特徴とする請求項 1 に記載のシステム。

【請求項 4】

前記リダイレクタは、前記ファイルに関連付けられた前記 F i d 情報および前記バージョン識別子情報を含む、前記ファイルに関連付けられたファイル制御ブロック (F C B) を選択的に作成することを特徴とする請求項 3 に記載のシステム。

20

【請求項 5】

前記リダイレクタは、既存の F C B を前記要求に対して使用することができるかどうかを判定することを特徴とする請求項 4 に記載のシステム。

【請求項 6】

前記リダイレクタは、コミットされていないトランザクションに関連付けられた既存の F C B を前記要求に対して使用することができるかどうかを判定することを特徴とする請求項 5 に記載のシステム。

【請求項 7】

前記リダイレクタは、既存の F C B を前記要求に対して使用することができるかどうかについて判定する際に、前記要求についてのパスネームおよびトランザクションコンテキストを前記既存の F C B に関連付けられたパスネームと比較することを特徴とする請求項 5 に記載のシステム。

30

【請求項 8】

前記トランザクションマネージャは、カーネルレベルトランザクションマネージャであることを特徴とする請求項 1 に記載のシステム。

【請求項 9】

前記リダイレクタは、前記リダイレクタにより要求されなかった前記ファイルについて実行されているファイルオペレーションに応答して、前記リモートデバイスが前記ローカルマシンに信号で伝えるべきことを、前記要求中に選択的に指示することを特徴とする請求項 1 に記載のシステム。

40

【請求項 10】

前記リダイレクタは、サーバメッセージブロック (S M B) プロトコルに基づいたプロトコルを使用して前記トランザクションを要求と共に送信することを特徴とする請求項 1 に記載のシステム。

【請求項 11】

前記プロトコルは、非トランザクション処理リモートファイルオペレーションをサポートすることを特徴とする請求項 10 に記載のシステム。

【請求項 12】

請求項 1 に記載のコンポーネントをその上に記憶していることを特徴とするコンピュー

50

タ読取り可能媒体。

【請求項 13】

ローカルデバイスとリモートデバイスとの間でトランザクション処理リモートファイルオペレーションをサポートするシステムであって、

前記ローカルデバイス中に存在するファイルについてトランザクション処理ファイルオペレーションを求める要求をリモートデバイスから受信するサーバコンポーネント（SRV）と、

トランザクションマネージャと、

前記SRVコンポーネントおよび前記トランザクションマネージャに応答して、前記ファイルについて前記要求されたファイルオペレーションを実行するファイルシステムであって、前記ローカルデバイスおよび前記リモートデバイスは、ネットワークに接続されることと、

を備えること特徴とするシステム。

【請求項 14】

前記SRVコンポーネントは、前記ファイルに関連付けられたファイルシステム識別子（Fid）およびバージョン識別子を前記リモートデバイスに提供することを特徴とする請求項 13 に記載のシステム。

【請求項 15】

前記トランザクションマネージャは、カーネルレベルトランザクションマネージャであることを特徴とする請求項 13 に記載のシステム。

【請求項 16】

前記ファイルシステムは、前記リモートデバイスにより要求されなかった前記ファイルについて実行されているファイルオペレーションに応答して、選択的に、前記SRVコンポーネントが前記リモートデバイスに信号で伝えるようにすることを特徴とする請求項 13 に記載のシステム。

【請求項 17】

前記SRVコンポーネントは、サーバメッセージブロック（SMB）プロトコルに基づいたプロトコルを使用して、前記トランザクションを要求と共に送信することを特徴とする請求項 13 に記載のシステム。

【請求項 18】

前記プロトコルは、非トランザクション処理リモートファイルオペレーションをサポートすることを特徴とする請求項 17 に記載のシステム。

【請求項 19】

請求項 12 に記載のコンポーネントをその上に記憶していることを特徴とするコンピュータ読取り可能媒体。

【請求項 20】

ローカルデバイス上でトランザクション処理リモートファイルオペレーションを実装する方法であって、

トランザクション処理リモートファイルオペレーションを求める要求を受信するステップと、

トランザクションを検索するステップと、

前記トランザクションをマーシャリングするステップと、

前記トランザクションを前記要求と共にネットワーク上でリモートデバイスに送信するステップと、

前記ファイルオペレーションからもたらされる情報を前記リモートデバイスから受信するステップと、

を備えることを特徴とする方法。

【請求項 21】

前記リモートデバイスから受信される情報は、ファイル識別子（Fid）およびバージョン識別子を含むことを特徴とする請求項 20 に記載の方法。

10

20

30

40

50

【請求項 2 2】

トランザクション処理リモートファイルオペレーションを求める要求を受信する前記ステップは、

前記要求がリモートデバイス上のファイルオペレーションを求めるものかどうかを判定するステップと、

前記要求がリモートデバイス上のファイルオペレーションを求めるものである場合に前記要求に対する名前を提供するステップと、

前記ファイルに関連付けられた前記 F i d 情報および前記バージョン識別子情報を含む、前記ファイルに関連付けられたファイル制御ブロック (F C B) を選択的に作成するステップと、

をさらに含むことを特徴とする請求項 2 1 に記載の方法。

【請求項 2 3】

F C B を選択的に作成する前記ステップは、既存の F C B を前記要求に対して使用することができるかどうかを判定するステップをさらに含むことを特徴とする請求項 2 2 に記載の方法。

【請求項 2 4】

既存の F C B を前記要求に対して使用することができるかどうかを判定する前記ステップは、前記既存の F C B がコミットされていないトランザクションに関連付けられているかどうかを判定するステップをさらに含むことを特徴とする請求項 2 3 に記載の方法。

【請求項 2 5】

既存の F C B が前記要求に対して使用することができるかどうかを判定する前記ステップは、前記要求に対するパスネームおよびトランザクションコンテキストを前記既存の F C B に関連付けられたパスネームと比較するステップをさらに含むことを特徴とする請求項 2 3 に記載の方法。

【請求項 2 6】

前記方法は、カーネルモードで実行されることを特徴とする請求項 2 0 に記載の方法。

【請求項 2 7】

前記ローカルデバイス以外のデバイスにより要求された前記ファイルについて実行されているファイルオペレーションに応答して、前記リモートデバイスが前記ローカルマシンに信号で伝えるべきことを前記要求中に選択的に指示するステップをさらに備えることを特徴とする請求項 2 0 に記載の方法。

【請求項 2 8】

前記トランザクションは、サーバメッセージブロック (S M B) プロトコルに基づいたプロトコルを使用して要求と共に送信されることを特徴とする請求項 2 0 に記載の方法。

【請求項 2 9】

前記プロトコルは、非トランザクション処理リモートファイルオペレーションをサポートすることを特徴とする請求項 2 8 に記載の方法。

【請求項 3 0】

トランザクション処理リモートファイルオペレーションを求める要求を受信するステップと、

トランザクションを検索するステップと、

前記トランザクションをマーシャリングするステップと、

前記トランザクションを前記要求と共にネットワーク上でリモートデバイスに送信するステップと、

前記ファイルオペレーションからもたらされる情報を前記リモートデバイスから受信するステップと、

を備えるオペレーションを実行するコンピュータ実行可能命令を有することを特徴とするコンピュータ読取り可能媒体。

【請求項 3 1】

前記リモートデバイスから受信される情報は、ファイル識別子 (F i d) およびバージ

10

20

30

40

50

ョン識別子を含むことを特徴とする請求項 30 に記載のコンピュータ読取り可能媒体。

【請求項 32】

トランザクション処理リモートファイルオペレーションを求める要求を受信する前記ステップは、

前記要求がリモートデバイス上のファイルオペレーションを求めるものかどうかを判定するステップと、

前記要求がリモートデバイス上のファイルオペレーションを求めるものである場合に前記要求に対する名前を提供するステップと、

前記ファイルに関連付けられた前記 F i d 情報および前記バージョン識別子情報を含む、前記ファイルに関連付けられたファイル制御ブロック (F C B) を選択的に作成するステップと、

をさらに含むことを特徴とする請求項 31 に記載のコンピュータ読取り可能媒体。

【請求項 33】

F C B を選択的に作成する前記ステップは、既存の F C B を前記要求に対して使用することができるかどうかを判定するステップをさらに含むことを特徴とする請求項 32 に記載のコンピュータ読取り可能媒体。

【請求項 34】

既存の F C B を前記要求について使用することができるかどうかを判定する前記ステップは、前記既存の F C B がコミットされていないトランザクションに関連付けられているかどうかを判定するステップをさらに含むことを特徴とする請求項 33 に記載のコンピュータ読取り可能媒体。

【請求項 35】

ローカルデバイスとリモートデバイスとの間でトランザクション処理リモートファイルオペレーションを実装する方法であって、

前記ローカルデバイス中に存在するファイルについてトランザクション処理ファイルオペレーションを求める要求をリモートデバイスから受信するステップであって、前記ローカルデバイスおよび前記リモートデバイスはネットワークに接続されていることと、

前記トランザクションの一部分となる前記要求されたファイルオペレーションを前記ファイルについて実行するステップと、

を備えることを特徴とする方法。

【請求項 36】

前記ファイルに関連付けられたファイルシステム識別子 (F i d) およびバージョン識別子を前記リモートデバイスに提供するステップをさらに備えることを特徴とする請求項 35 に記載の方法。

【請求項 37】

前記方法は、カーネルモードで実行されることを特徴とする請求項 35 に記載の方法。

【請求項 38】

前記リモートデバイスにより要求されなかった前記ファイルについて実行されるファイルオペレーションに応答して、前記リモートデバイスに選択的に信号を伝えるステップをさらに備えることを特徴とする請求項 35 に記載の方法。

【請求項 39】

前記ローカルデバイスと前記リモートデバイスは、サーバメッセージブロック (S M B) プロトコルに基づいたプロトコルを使用して、前記ネットワーク上で選択的に情報をやりとりすることを特徴とする請求項 35 に記載の方法。

【請求項 40】

前記プロトコルは、非トランザクション処理リモートファイルオペレーションをサポートすることを特徴とする請求項 39 に記載の方法。

【請求項 41】

ローカルデバイス中に存在するファイルについてトランザクション処理ファイルオペレーションを求める要求をリモートデバイスから受信するステップであって、前記ローカル

10

20

30

40

50

デバイスおよび前記リモートデバイスはネットワークに接続されていることと、

前記トランザクションの一部分となる前記要求されたファイルオペレーションを前記ファイルについて実行するステップと、

を備えるオペレーションを実行するコンピュータ実行可能命令を有することを特徴とするコンピュータ読取り可能媒体。

【請求項 4 2】

前記オペレーションは、前記ファイルに関連付けられたファイルシステム識別子 (F i d) およびバージョン識別子を前記リモートデバイスに提供するステップをさらに備えることを特徴とする請求項 4 1 に記載のコンピュータ読取り可能媒体。

【請求項 4 3】

前記オペレーションは、カーネルモードで実行されることを特徴とする請求項 4 1 に記載のコンピュータ読取り可能媒体。

【請求項 4 4】

前記オペレーションは、前記リモートデバイスにより要求されなかった前記ファイルについて実行されるファイルオペレーションに応答して、前記リモートデバイスに選択的に信号を伝えるステップをさらに備えることを特徴とする請求項 4 1 に記載のコンピュータ読取り可能媒体。

【請求項 4 5】

ネットワークに接続されたローカルデバイスとリモートデバイスの間でトランザクション処理リモートファイルオペレーションをサポートするシステムであって、

トランザクションを管理する手段と、

前記リモートデバイス中に存在するファイルについてファイルオペレーションを実行するためにトランザクションを伴う要求を前記リモートデバイスに送信する手段と、

を備えることを特徴とするシステム。

【請求項 4 6】

前記ファイルに関連付けられたファイルシステム識別子 (F i d) およびバージョン識別子を含むファイル情報を前記リモートデバイスから受信する手段をさらに備えることを特徴とする請求項 4 5 に記載のシステム。

【請求項 4 7】

前記ファイルに関連付けられた前記 F i d 情報および前記バージョン識別子情報を含む、前記ファイルに関連付けられたファイル制御ブロック (F C B) を選択的に作成する手段をさらに備えることを特徴とする請求項 4 6 に記載のシステム。

【請求項 4 8】

既存の F C B を前記要求に対して使用することができるかどうかを判定する手段をさらに備えることを特徴とする請求項 4 7 に記載のシステム。

【請求項 4 9】

コミットされていないトランザクションに関連付けられた既存の F C B を前記要求に対して使用することができるかどうかを判定する手段をさらに備えることを特徴とする請求項 4 8 に記載のシステム。

【請求項 5 0】

リダイレクタにより要求されなかった前記ファイルについて実行されているファイルオペレーションに応答して、前記リモートデバイスが前記ローカルマシンに信号を伝えるべきことを前記要求中に選択的に指示する手段をさらに備えることを特徴とする請求項 4 5 に記載のシステム。

【請求項 5 1】

前記トランザクションマネージャは、ファイルシステムに一体化されていないことを特徴とする請求項 1 3 に記載のシステム。

【請求項 5 2】

ネットワーク上でトランザクションファイルオペレーションを実行する際に使用する方法であって、

10

20

30

40

50

第1のコンピューティングプラットフォーム中に存在するトランザクションマネージャからプリプリペア通知を受信するステップと、

前記第1のコンピューティングプラットフォームから第2のコンピューティングプラットフォームに前記ネットワークを介してファイルオペレーションデータを提供するステップと、

前記第1のコンピューティングプラットフォーム中に存在する前記トランザクションマネージャからプリペア通知を受信するステップと、

前記プリペア通知を含む前記第2のコンピューティングプラットフォームに前記ネットワークを介してメッセージを提供するステップと、

を備えることを特徴とする方法。

10

【請求項53】

ネットワーク上でトランザクション処理ファイルオペレーションを実行する際に使用する方法であって、

トランザクション情報を受信するステップと、

ファイルもしくはディレクトリを作成もしくはオープンするために、前記ネットワークに接続されたコンピューティングプラットフォームに対して、前記トランザクション情報および要求を前記ネットワークを介して送信するステップであって、前記要求は第1の構造を含み、前記第1の構造はトランザクションコンテキストを保持するフィールド並びに前記トランザクション情報の長さを保持するフィールドを含むことと、

を備えることを特徴とする方法。

20

【請求項54】

前記要求に対する応答は、ファイルシステム識別子を保持するフィールド、並びに、バージョン識別子を保持するフィールドを含む第2の構造を備えることを特徴とする請求項53に記載の方法。

【請求項55】

前記ネットワークに接続された前記コンピューティングプラットフォームがトランザクション処理ファイルオペレーションをサポートするかどうかを示すフィールドを含む応答を受信するステップをさらに備えることを特徴とする請求項53に記載の方法。

【請求項56】

トランザクション情報、並びに、トランザクションが前記ファイルと共に使用されることを示すフィールドを含むファイルを見つける要求を、前記ネットワークを介して前記コンピューティングプラットフォームに送信するステップをさらに備えることを特徴とする請求項53に記載の方法。

30

【請求項57】

前記ネットワークに接続された前記コンピューティングプラットフォームからの、トランザクション状態変化に関する通知を提供するメッセージを前記ネットワークを介して受信するステップをさらに備えることを特徴とする請求項53に記載の方法。

【請求項58】

ネットワーク上でトランザクション処理ファイルオペレーションを実行する際に使用する方法であって、

40

第2のコンピューティングプラットフォームにおいて第1のコンピューティングプラットフォームからのプリプリペア通知を受信するステップと、

前記第2のコンピュータプラットフォーム中に存在するトランザクションマネージャに対して前記プリプリペア通知を提供するステップと、

前記第2のコンピューティングプラットフォームにおいて、前記ネットワークを介して、前記第1のコンピューティングプラットフォームからのファイルオペレーションデータを受信するステップと、

前記第2のコンピューティングプラットフォーム中に存在するファイルシステムに対して前記ファイルオペレーションデータを提供するステップと、

前記ネットワークを介して前記第1のコンピューティングプラットフォームからのプリ

50

ペア通知を受信するステップと、

前記第2のコンピューティングプラットフォーム中に存在する前記トランザクションマネージャに前記プリペア通知を提供するステップと、

を備えることを特徴とする方法。

【発明の詳細な説明】

【技術分野】

【0001】

以下に説明する様々な実施形態は、一般にネットワーク通信に関し、より詳細には、それだけには限定されないが、ネットワーク上のトランザクション処理ファイルオペレーションを可能にするための方法およびシステムに関する。

10

【背景技術】

【0002】

トランザクションは、長い間、データベースおよびトランザクション処理システムによって提供されてきている。トランザクションは、いくつかのオペレーションを1つの原始的オペレーションに、すなわち個々のオペレーションの結果がともに有効か、または共倒れとなるというオペレーションのグループと一緒にグループ分けすることによって、アプリケーションプログラマにとって望ましい簡略化された失敗モデルを提供する。

【発明の開示】

【発明が解決しようとする課題】

【0003】

20

もし、ただ1つのオペレーションが失敗する場合には、そのグループ中におけるすべてのオペレーションの効果は、そのトランザクションに関連するオペレーション数にかかわらず、「元に戻され(undone)」、あるいはロールバックされる。オペレーションの間のこの連帯(solidarity)は、失敗の数がいくつの場合においても適用され、結局のところ各トランザクション処理システムは、これらのオペレーションのすべてが適用されている、あるいは、これらのオペレーションのいずれも適用されていないという2つの状態の内の、どちらか一方の状態に至る。

【課題を解決するための手段】

【0004】

記載される様々な実施形態の態様によれば、ネットワーク上でファイルオペレーションをトランザクション処理する方法およびシステムが提供される。一態様によれば、コンピューティングプラットフォーム(すなわち、クライアント)は、ネットワークを介して、別のコンピューティングプラットフォーム(すなわち、サーバ)上のファイルに遠隔でアクセスすることができる。この態様によれば、これらクライアントおよびサーバはそれぞれ、TM(transaction managerトランザクションマネージャ)およびFS(file systemファイルシステム)を含んでいる。このクライアントは、RDR(redirectorリダイレクタ)も含んでいるが、このサーバは、SRV(server componentサーバコンポーネント)を含んでいる。

30

【0005】

動作中においては、RDRは、リモートトランザクション処理ファイルオペレーション(remote transacted file operation)の要求を受信する。この要求に応答して、RDRはこの要求からのトランザクションについて調べ、(例えば、一実施形態におけるこのTMによって)サーバへ送信するためにこのトランザクションをマーシャリングする。次いでRDRは、ネットワーク上でこのトランザクション情報(例えば、一実施形態においてはマーシャリングブロッブ(marshal blob))をサーバに送信する。SRVはトランザクション情報を受信し、次いでサーバのTMおよびFSは、トランザクション情報を使用してファイルオペレーションを実施する。次いで、サーバはネットワークを介してファイルオペレーションの結果をクライアントに返送する。

40

【0006】

50

他の態様においては、R D Rは、1つのファイルに対して複数のリモートファイルオペレーショントランザクションがオープンであるようにすることができる。R D Rがトランザクション処理リモートファイルオペレーションを求める新しい要求を受信するとき、R D Rは、リモートファイルの「ダーティ (dirty)」バージョン (すなわち、書き込まれているバージョン) が、クライアント上で知られているかどうかを判定する。次いで、R D Rはファイルについて元のバージョンの代わりに、新しい要求に対するダーティバージョンを使用する。一部の実施形態においては、R D Rは、1つのトランザクション処理されたライトオペレーションが所与のファイルについて一時にオープンであることを可能にするだけである。

【0007】

10

さらに他の態様においては、R D Rは、トランザクション処理されたリモートファイルオペレーションを求める新しい要求は、クライアント上ですでに知られているファイル情報を使用することができるかどうかを判定する。この同じファイル情報を使用することができる場合に、R D Rは、このファイル情報のもう1つのコピーを記憶する代わりにこの同じファイル情報を使用する。

【0008】

さらに他の態様においては、R D Rは、所与のリモートファイルについてのトランザクションに便宜的ロックを関連づけることができる。一実施形態においては、このロックは、ローカルサーバがファイルにアクセスすることを妨げないが、サーバに、ロックが解除されているというメッセージをクライアントに対して送信させる。次いで、トランザクション処理リモートファイルオペレーションを求める新しい要求が、クライアント中にすでにキャッシュされているファイル情報を使用することができるかどうかを判定するに際し、R D Rは所与のファイルについてロックが解除されたかどうかを検査することができる。

20

【0009】

非限定的で非網羅的な実施形態について、以下の図を参照しながら説明する。別に指定をしていない限り、様々な図面全体にわたって、同一の参照番号は同一の要素を示している。

【発明を実施するための最良の形態】

【0010】

30

実施例のネットワーク環境

前述のように、トランザクションは、データベースおよびトランザクション処理システムにおいて使用されてきているが、以下の実施形態においては、トランザクションは、リモートファイルオペレーションのために使用される。図1は、クライアントがネットワーク101を介してクライアント上でファイルオペレーションをトランザクション処理することができるシステム100を示している。図1の実施例のネットワーク環境においては、クライアント装置と呼ぶこともできる複数のクライアントコンピューティングデバイス105、110、115、および120が、ネットワーク101を介して少なくとも1つのサーバ装置125に結合されている。ネットワーク101は、有線ネットワークおよび/または無線ネットワークを含むことができる様々な従来のネットワークトポロジおよびタイプのいずれをも表すことが意図されている。ネットワーク101はさらに、パブリックプロトコルおよび/または独自開発プロトコルを含めて、様々な従来のネットワークプロトコルのうちのいずれをも利用することができる。ネットワーク101は、例えばインターネット、ならびに場合によっては1つまたは複数のLAN (local area network ローカルエリアネットワーク)、WAN (wide area network ワイドエリアネットワーク) などの少なくとも一部分を含むことができる。

40

【0011】

クライアント装置105は、それだけには限定されないが、デスクトップPC (personal computer パーソナルコンピュータ)、ワークステーション、メインフレームコンピュータ、インターネットアプライアンス、およびゲーミングコンソール (

50

gaming console)を含めて様々な従来のコンピューティングデバイスのうちのいずれも含むことができる。さらに、ネットワーク101に関連するクライアント装置は、有線リンクおよび／または無線リンクによってネットワーク101と情報をやりとりすることができるPDA (personal digital assistant 携帯型個人情報端末) 110、ラップトップコンピュータ115、およびセルラ電話120などを含むことができる。さらにまた、1つまたは複数のクライアント装置105、110、115、および120は、これら同じタイプのデバイス、または代わりの異なるタイプのデバイスを含むことができる。

【0012】

サーバ装置125は、コンピューティングデバイス105、110、115および120 10
0に対して、様々なデータおよび／または機能のうちのいずれも提供することができる。このデータは、パブリックに利用可能であり、あるいは代わりに制限されることもあり、例えばあるユーザだけに制限され、または適切な料金が支払われる場合にしか利用可能でないなどのこともある。

【0013】

サーバ装置125は、ネットワークサーバ、およびアプリケーションサーバ、またはその両者による組合せのうちの少なくとも1つである。サーバ装置125は、コンテンツの 20
ソースとなる任意のデバイスであり、クライアント装置105、110、115および120は、このようなコンテンツを受信する任意のデバイスを含んでいる。したがって、ピアツーピアネットワークにおいて、このコンテンツのソースとなるこのデバイスは、サーバ装置と呼ばれ、このコンテンツを受信するデバイスは、クライアント装置と呼ばれる。両方のタイプのデバイスは、本明細書中で説明される実施形態の例に従って、オペレーティングシステムおよびアプリケーションを含めて、ソフトウェアプログラムをロードし実行することが可能である。さらに、データおよび機能は、クライアント装置105、110、115および120の間で共有することができる。すなわち、サービス装置125だけが、各クライアント装置に対するデータおよび／または機能のソースであるというわけではない。

【0014】

データソース130または135において、オペレーティングシステムおよびアプリケーションを含めてソフトウェアプログラムが、実行するためにサーバ装置125またはク 30
ライアント装置105、110、115、および120のうちのどれか1つのために作成され、かつ／またはそのどれか1つに対して提供される。一貫性をもたせるために、以下の説明では、当技術分野において知られているように、単独または組合せのいずれでも、少なくともソフトウェアプログラム、オペレーティングシステム、およびアプリケーションのどれかを包含する「アプリケーション」について言及している。さらに、これらのアプリケーションは、データソース130からのものとしてオフラインで、または、データソース135からのものとしてオンラインで、サーバ装置125に対して配布される。さらにまた、これらのアプリケーションは、一般にサーバ装置125またはデータソース135からオンラインでクライアント装置105、110、115、および120に対して 40
配布される。これらのオフラインによる配布のための手段および方法についても同様に知られている。

【0015】

以下で説明する様々な実施形態によれば、共にデバイス105、110、115、120、125における、またはその間にあるデータおよび機能の少なくとも1つの配布は、トランザクションとして実装することができる。より詳細には、トランザクションは、図1の実施例のように、デバイス105、110、115、120、125のうちの1つの内部においてまたはネットワーク環境の中において、1つの原子的オペレーションとして同期してまたは非同期で実行されるオペレーションのグループである。このネットワーク上で実施されるトランザクション処理リモートファイルオペレーションの実施例について、図2～図7とともに以下にさらに詳細に説明する。

【0016】

トランザクション処理リモートファイルオペレーション

図2は、図1におけるシステム100中の（例えば、デバイス105、110、115、120、125から選択される）2つのデバイスのコンポーネントを示している。これらは、トランザクション処理リモートファイルオペレーションのためのクライアント202およびサーバ204として動作する。この実施形態においては、クライアント202もサーバ204も共に、あるバージョンのMicrosoft（登録商標）Windows（登録商標）オペレーティングシステムを使用している。他の実施形態においては、異なるオペレーティングシステムを使用することができる。

【0017】

この実施形態においては、クライアント202は、アプリケーション212、I/O（input/output入出力）マネージャ214、ファイルシステム（FS）216、リダイレクタセクタ（redirector selector）218、トランザクションマネージャ（TM）222、およびリダイレクタ（RDR）220を含んでいる。この実施形態において、サーバ204は、サーバコンポーネント（SRV）234、I/Oマネージャ214A、FS216AおよびTM222Aを含んでいる。この実施形態において、クライアント202およびサーバ204は、ネットワーク100（図1）を介して互いに情報をやりとりすることができる。一部の実施形態においては、これらのコンポーネントは、ソフトウェアにより実装される。

【0018】

この「Windows（登録商標）」の実施形態において、I/Oマネージャ214および214A、ならびにFS216および216Aは、NTFS（NT file system NTファイルシステム）によって実装され、リダイレクタセクタ218は、MUP（multiple UNC provider複数UNCプロバイダ）によって実装され、ここでUNCは、Uniform Naming Convention（汎用名前付け規則）の頭字語である。したがって、リダイレクタセクタ218は、本明細書中においてMUP218とも呼ばれる。さらに、Microsoft（登録商標）Windows（登録商標）オペレーティングシステムのRDRおよびSRVは、（追加機能とともに）それぞれRDR220およびSRV234を実装する。このMicrosoft（登録商標）Windows（登録商標）オペレーティングシステムのRDRおよびSRVに対する追加機能の一例については、以下で説明する。さらにまた、この実施形態において、TMM222およびTM222Aは、この実施形態の例においてはカーネルレベルトランザクションマネージャとして実装され、以下にさらに詳細に説明する。他の実施例においては、異なるI/Oマネージャ、ファイルシステム、リダイレクタセクタ、TM、および/またはRDRを使用することもできる。

【0019】

図3は、クライアント202とサーバ204の間のトランザクション処理リモートファイルオペレーション（図2）についての実施例の処理フローを示している。図2および図3を参照すると、トランザクション処理リモートファイルオペレーションは、一実施形態によれば、以下のように実施される。

【0020】

ブロック302において、RDR220は、サーバ204中に存在するファイルに対するトランザクション処理ファイルオペレーションを求める要求を受信する。典型的なファイルオペレーションは、新しいファイルを作成する工程、ファイルを読み取る工程、ファイルを書き込む工程、ファイルをコピーする工程、ファイルの名前を変更する工程などを含んでいる。この実施形態においては、トランザクション処理ファイルオペレーションを求める要求は、アプリケーション212によって生成される。このアプリケーションは、図2に示すようにユーザレベルのアプリケーションである。この要求は、トランザクションコンテキストについてのフィールドを含む構造を使用する。この要求は、I/Oマネージャ214によって受信され、このI/Oマネージャは、この要求がローカルファイルに対

10

20

30

40

50

するものか、それともリモートファイルに対するものかを判定する。この実施形態においては、I/Oマネージャ214は、このMicrosoft（登録商標）Windows（登録商標）オペレーティングシステムの標準コンポーネントである。例えば、アプリケーション212は、（\\server\share\subdirectory\filenameの形式の）このUNCネームを用いて、I/Oマネージャ214への呼び出しを介して、この要求を行うことができる。次いでI/Oマネージャ214は、この要求をMUP218に渡す。トランザクションについては複数のハンドルが存在し、所与のファイルについては複数のトランザクションが存在することができる。他方、この要求がこのクライアント上のファイルを求めるものである場合には、I/Oマネージャ214は、標準的な方法によって、この要求をNTFS216に渡すはずである。

10

【0021】

次いでMUP218は、この要求を実施するために必要とされるリダイレクタの位置を突き止める。このケースにおいては、このリダイレクタは、RDR220である。この実施形態において、MUP218は、このMicrosoft（登録商標）Windows（登録商標）オペレーティングシステムの標準コンポーネントである。この実施形態においては、RDR220は、このMicrosoft（登録商標）Windows（登録商標）オペレーティングシステムRDRの1つのバージョンであり、追加部分を用いることにより、このRDRは、TM222と対話してトランザクションを実施することができる。この追加部分は例えば、要求からトランザクション処理ファイルオペレーションについてのトランザクションコンテキストを検索し、トランザクション処理ファイルオペレーションについてのFCBを割り当て、ネットワーク上でトランザクションをリモートデバイスに送信し、（ファイル識別子およびバージョン識別子を含めて）これらのトランザクション処理ファイルオペレーションについての応答を受信し、TM222の指示の下にトランザクションオペレーションを実施し、RDR220がトランザクションの状態に関して通知され続けることができるようにするためのTM222をリソースマネージャとしてエンリストする（enlist）機能を含んでいる。一部の実施形態においては、RDR220は、同時係属の同一出願人による「Transactional File System」という名称の2000年3月30日に出願の、米国特許出願第09/539,233号明細書および米国出願番号[整理番号第MS1-1781US]に説明されているように実装される。リソースマネージャとしてのエンリスト（enlisting）については以下で説明する。RDR220は、このトランザクションをバッファリングするためのリソース、キャッシュマップ、FCB（file control blockファイル制御ブロック）、FOB（file object extensionファイルオブジェクト拡張部分）、およびこのトランザクションおよび要求を処理するために必要な他の構造を含んでいる。

20

30

【0022】

ブロック304において、RDR220は、TM222からこのトランザクションを検索し、このトランザクションをクライアント204に送信するためにマーシャリングする。一実施形態においては、RDR220は、TM222によって表示される（その実施形態について以下で説明している）APIを呼び出すことによってこのトランザクションを検索し、トランザクションをサポートするように拡張されている1つのバージョンのSMBプロトコルを使用して送信するためにこのトランザクション情報（例えば、マーシャリングプロブ）をフォーマットすることによってこのトランザクションをマーシャリングする。例示の一実施形態のSMB拡張部分については、表1～表3とともに以下で要約している。ブロック306において、RDR220は、矢印236で示すようにこのトランザクションおよびこの要求をサーバ204に対して送信する。ブロック308において、RDR220は、サーバ204からこのファイルオペレーションの結果を受信する。例えば、サーバ204は、前述のファイルおよびバージョン識別子を含むこの要求に対する応答を送信する。この実施形態においては、SRV234は、このMicrosoft（登録商標）Windows（登録商標）オペレーティングシステムSRVの1つのバージョンである。追加機能部分を用いることにより、このSRVは、トランザクション処理リモー

40

50

トファイルオペレーション中にファイルおよびバージョン識別子をクライアントに対して渡すことを含め、ネットワーク上でクライアントと対話してSMBに対する拡張部分を使用してトランザクションを実施することができる。

【0023】

【表1】

拡張部分	記述	
新しいコマンド: NT_TRANSACT_CREATE2を追加する	マーシャリングされたトランザクションを獲得し、2つの構造REQ_CREATE_WITH_EXTRA_OPTIONSおよびRESP_CREATE_WITH_EXTRA_OPTIONSをネットワーク上で送信する。これら2つの構造は、それぞれ表2および表3で定義され、SMB構造: REQ_CREATE_WITH_SD_OR_EAおよびRESP_EXTENDED_CREATE_WITH_SD_OR_EAの拡張部分である。	10
新しい機能ビット: CAP_TXFを追加する	CAP_TXFは、サーバによってセットまたはリセットされてこのサーバがトランザクションをサポートするかどうかを示す。CAP_TXFは、SMBネゴシエーション応答 (SMB Negotiate Response) の一部分である。この実施形態においては、CAP_TXFはサーバサポートトランザクションを示す0x20000として定義される。	
新しいフラグ: SMB_FIND_TRANSACTED_OPERATIONをSMB_FIND要求に追加する。このFIND要求 (REQ_FIND_FIRST2) 構造は、表4で定義され、その応答構造は表5で定義される。	SMB_FIND_TRANSACTED_OPERATIONは、トランザクションが使用されていることを示す。この実施形態においては、ファインドオペレーション (find operation) がハンドルベースではなくてパスベース (path based) であるので、このフラグが使用される。この実施形態においては、このフラグは、0x20として定義される。このフラグが設定される場合に、トランザクション情報は、FINDコマンドおよびECHOコマンドの終わりに送信される。	20
ECHOコマンドを拡張してトランザクション状態変化の通知を送信する。この要求/応答構造は、表6および7で定義される。	SMBECHOコマンドを拡張してトランザクションオペレーションのプリプリペア状態、プリペア状態、コミット状態、ロールバック状態の通知をサーバからクライアントへと提供する。	30

表 1

【0024】

【表 2】

REQ___CREATE___WITH___EXTRA___OPTIONS

フィールド	内容の記述	
_ULONG (Flags)	フラグNT_CREATE_XXXの作成	
_ULONG (	相対的なオープンについてのオプション的ディレク	
RootDirectoryFid)	トリ	
ACCESS_MASK DesiredAccess	望ましいアクセス (NTフォーマット)	
LARGE_INTEGER	バイト単位の 初期割り当て サイズ	
AllocationSize		10
_ULONG (FileAttributes)	ファイル属性	
_ULONG (ShareAccess)	共用アクセス	
_ULONG (	ファイルが存在するか否かを獲得するアクション	
CreateDisposition)		
_ULONG (CreateOptions)	新しいファイルを作成するためのオプション	
_ULONG (	バイト単位の SD 長	
SecurityDescriptorLength)		
_ULONG (EaLength)	バイト単位の EA (extended attribute 拡張属性) 長	
_ULONG (NameLength)	キャラクタによる名前の長さ	20
_ULONG (	セキュリティ QOS (Quality of Service サービス 品質	
ImpersonationLevel)	) 情報	
_UCHAR SecurityFlags	セキュリティ QOS 情報	
_ULONG (	バイト単位の マーシャリングされた トランザクショ	
TransactionLength)	ンコンテキスト 長	
_ULONG (EfsStreamLength)	バイト単位の \$EFS (encrypted file system 暗号化フ	
	ァイルシステム) ストリーム 長	
UCHAR Buffer[1]	データバッファ中で DWORD (4 バイト) 境界に位置合	
	わせされた、ファイルネーム用の バッファ	
UCHAR Name[]	(NUL で終了していない) ファイルネーム	30

表 2

【 0 0 2 5 】

【表 3 - 1】

RESP__CREATE__WITH__EXTRA__OPTIONS

フィールド	内容の記述	
UCHAR OplockLevel	許可された便宜的ロックレベル	
UCHAR ExtendedResponse	拡張された応答について1にセットする	
_USHORT(Fid)	ファイルID	
_ULONG(CreateAction)	獲得されたアクション	
_ULONG(EaErrorOffset)	EAエラーのオフセット	
TIME CreationTime	ファイルが作成された時刻	10
TIME LastAccessTime	ファイルがアクセスされた時刻	
TIME LastWriteTime	ファイルが最後に書き込まれた時刻	
TIME ChangeTime	ファイルが最後に変更された時刻	
_ULONG(FileAttributes)	ファイル属性	
LARGE_INTEGER AllocationSize	割り当てられたバイト数	
LARGE_INTEGER EndOfFile	ファイルオフセットの終了	
_USHORT(FileType)	ファイルのファイルタイプ	
_USHORT(DeviceState)	IPCデバイス（例えば、パイプ）の状態	20
BOOLEAN Directory	この構造がディレクトリである場合に真（TRUE）	
UCHAR VolumeGuid[16]	ボリュームGUID（Globally Unique IDグローバルに一意のID）	
UCHAR FileId[8]	ファイルID	
_ULONG(MaximalAccessRights)	セッション所有者に対するアクセス権	

【 0 0 2 6 】

30

【表 3 - 2】

フィールド	内容の記述	
_ULONG(GuestMaximalAccessRights)	ゲストに対する最大限のアクセス権	
LARGE_INTEGER FilesystemFid:	同じパスネーム（pathname）を有する異なるファイルを区別するサーバ上のNTFSFid。トランザクション（TxF）を使用しながら、同じパスネームで、2つの異なるファイルを示すことができる。	40
_ULONG(VersionNum):	オープンされているファイルのTxFバージョン#	

表 3

【 0 0 2 7 】

【表 4】

REQ__FIND__FIRST2

フィールド	内容の記述
_USHORT(SearchAttributes)	検索属性
_USHORT(SearchCount)	戻すべき最大エントリ数
_USHORT(Flags)	追加情報: 設定ビット 0 - この要求の後に検索をクローズ (close) する 1 - 末尾に到達した場合に、検索をクローズする 2 - レジュームキーを戻す
_USHORT(InformationLevel)	情報レベル
_ULONG(SearchStorageType)	検索ストレージタイプ
UCHAR Buffer[1]	ファイルネーム

10

20

表 4

【0028】

【表 5】

Rsp__find__first2

フィールド	内容の記述
_USHORT(Sid)	ハンドルを検索する
_USHORT(SearchCount)	戻されるエントリ数
_USHORT(EndOfSearch)	最後のエントリは戻されたか?
_USHORT(EaErrorOffset)	EAエラーの場合、EAリストにオフセットする
_ULONG(SearchStorageType)	検索ストレージタイプ
_USHORT(LastNameOffset)	サーバが検索を再開する必要がある場合、最後のエントリのファイルネームまでデータにオフセットし、そうでない場合には、0とする

30

40

表 5

【0029】

【表 6】

REQ__ECHO

フィールド	内容の記述
UCHAR WordCount	パラメータワードのカウント = 1
_USHORT(SearchCount)	戻されたエントリ数
_USHORT(EndOfSearch)	最後のエントリは戻されたか?
_USHORT(EaErrorOffset)	EAエラーの場合、EAリストにオフセットする
_USHORT(LastNameOffset)	サーバが検索を再開する必要がある場合、最後のエントリのファイルネームまでデータにオフセットし、そうでない場合には、0とする

10

表 6

【0030】

20

【表 7】

Rsp__echo

フィールド	内容の記述
UCHAR WordCount	パラメータワードのカウント = 1
_USHORT(SequenceNumber)	このエコーのシーケンス番号
_USHORT(ByteCount)	データバイトのカウント: min = 4
UCHAR Buffer[1]	エコーされたデータ

30

表 7

【0031】

図 3 A は、一実施形態におけるブロック 302 (図 3) をさらに詳細に示している。ブロック 312 において、RDR220 は、この要求されたファイルオペレーションについてのトランザクションコンテキストを検索する。トランザクション処理リモートファイルをオープンする際に、RDR220 は、トランザクションがこの要求にすでに関連づけられているかどうかを判定する。例えば、一実施形態において、トランザクションはそれをスレッド (thread) にアタッチすることによって要求に関連付けられるが、他の実施形態においては、異なる方法を使用してトランザクションを要求に関連づけることができる。一実施形態においては、RDR220 は、この要求がそれに関連づけられたトランザクションハンドルを有するかどうかを調べるために検査することによってこのオペレーションを実施する。関連づけられたトランザクションハンドルを有する場合には、この要求は、その既存のトランザクションに加え (join) られる。有していない場合には、RDR220 は、非トランザクション処理要求に対する標準的な方法でこの要求を処理する。

40

50

【0032】

次いで、RDR220は、この要求にFCBを割り当てる。前述のように、複数の要求を有する複数のトランザクションが、所与のファイルをオープンすることもある。したがって、ブロック302（図3）の一実施形態においては、RDR220が既存のFCBをこの要求に対しても使用できるかどうかについて判定するブロック314が、実行される。この実施形態において、RDR220は、検査して、この要求のファイル（すなわち、パスネーム）およびこの要求を作るスレッドに関連するこのトランザクションコンテキストが、既存のFCBのこれらとマッチングするかどうかを調べる。例えば、第2の要求の処理中に、この同じファイルの2つの書込みオペレーションが、同じトランザクション中で要求されたとする場合には、この第1の要求のために既にFCBが存在しているはずである。両方のオペレーションが共に書込みオペレーションであるので、この同じFCBを両者に対して使用することができる。

【0033】

ブロック314において、RDR220が、FCBが同じトランザクションコンテキストならびに同じファイル（すなわち、パスネーム）および同じバージョンと共に存在すると判定する場合には、ブロック316において、既存のFCBがこの要求に対して使用される。一部の実施形態において、RDR220は、最新バージョンを有するFCBを使用することになる。例えば、ファイルの読取りオペレーションがこの同じファイルのコミットされていない書込みオペレーションに続く場合、RDR220は、この書込みオペレーションが現在使用しているファイルのこのバージョンを使用することになる。このアプローチによって、キャッシングのさらに効率的な使用が可能になる。

【0034】

しかし、ブロック314において、この要求に対する既存のFCBを使用することができない場合には、ブロック318においてRDRは、この要求に対して新しいFCBを作成する。代替的な実施形態においては、新しいFCBが、要求ごとに作成される。

【0035】

図4は、この同じファイルに対するコミットされない複数のトランザクション処理要求の一例を示している。図4に示すように、オペレーション401は、ファイルの読取りオペレーションを求める要求に対応する。すなわち、このファイルオペレーションは、「読取りのためにオープンする」オペレーションである。オペレーション401は、ハンドルH1と、このハンドルに関連付けられたトランザクションT1を有する。RDR220（図2）が要求するこのファイルのバージョンは、バージョンAとして示される。これがこのファイルに対しての最初のコミットされないトランザクションであると想定すると、バージョンAは、サーバ204（図2）から読み出され、クライアント202（図2）中にキャッシュされる。

【0036】

後の時刻において、オペレーション402がこの同じファイルに対して要求される。この例においては、オペレーション402も読取りオペレーションであり、ハンドルH2およびトランザクションT2を有する。このトランザクションは、オペレーション401のトランザクションとは異なるので、RDR220は、この場合にもサーバ204からこのファイルのバージョンAが読み出される。

【0037】

この例においては、次いでオペレーション403が、オペレーション402と同じトランザクションにおいてこの同じファイルに対して要求される。したがって、オペレーション403はハンドルH3を有し、トランザクションT2に結合（join）される。しかし、この例においては、オペレーション403は書込みオペレーションであり、したがってRDR220は、このファイルのバージョンBをローカルに記憶している（例えば、キャッシュしている）。バージョンBは、時として「ダーティバージョン」と呼ばれることがある。

【0038】

10

20

30

40

50

次いでオペレーション404が、オペレーション402および403と同じトランザクションにおいて、この同じファイルに対して要求される。したがって、オペレーション404はハンドルH4を有し、これもまたトランザクションT2に結合される。この例においては、オペレーション404は読取りオペレーションである。この実施形態においては、ブロック314（図3A）の結果により、RDR220は、オペレーション404に対するバージョンBを記憶しており、おそらくキャッシュしていることになる。

【0039】

次いで、オペレーション405が、異なるトランザクションにおいてこの同じファイルに対して要求される。したがって、オペレーション405はハンドルH5を有し、新しいトランザクションT3に関連付けられる。このトランザクションは、以前のオペレーションのトランザクションとは異なるので、一実施形態において、RDR220は、この場合にもこのファイルのバージョンAをサーバ204から読み出す。他の実施形態においては、RDR220はサーバ204（図2）を調べなくても、依然としてバージョンAが現行バージョンであることを認識しており、この「ローカル」バージョンAを使用する。例えば、この代替的な実施形態では、便宜的ロックを使用してサーバ204中に存在するこのファイルのどのような新しいバージョンについても知ることができるようになる。すなわち、RDR220は、サーバ204上のファイルに対する書込みを妨げることをしない便宜的ロックを、このファイルに関連付けることができるが、サーバ204は、このロックが解除されたことをRDR220に信号で伝えるようにする。このようなケースにおいては、次いでRDR220は、バージョンAはもはや最新のバージョンではないことを知るはずである。さらに他の実施形態においては、RDR220は、サーバ204に照会してこのファイルの最新バージョンを決定し、次いでこの最新のバージョンに関連付けられた既存のFCBを再使用することができる。

【0040】

次いで、オペレーション406において、トランザクションT2がコミット（commit）される。これは、サーバ204上でこのバージョンを変更する効果を有する。サーバ204上に記憶されるこの新しいバージョンは、バージョンCとして示される。前述したように、RDR220は、すべてのリモートトランザクション中にリソースマネージャとしてエンリストするので、RDR220は、サーバ204がこのファイルの新しいバージョンを有することを学習する。

【0041】

次いで、オペレーション401と同じトランザクションにおいてこの同じファイルに対して、オペレーション407が要求される。したがって、オペレーション407はハンドルH6を有し、トランザクションT1に結合される。しかし、RDR220はサーバ204上のこのファイルのバージョンCを知っているので、RDR220は、このオペレーションに対するバージョンCを記憶しており、おそらくキャッシュしている。一部の実施形態においては、RDR220は、サーバ204からバージョンCを読み出す。

【0042】

同様に、オペレーション405と同じトランザクションによってこの同じファイルに対して、オペレーション408が要求されるとき、オペレーション408はハンドルH7を有し、トランザクションT3に結合される。この場合にも、RDR220がサーバ204上のこのファイルのバージョンCについて知っているので、RDR220は、このオペレーションに対するバージョンCを記憶しており、おそらくキャッシュしている。

【0043】

図5は、一実施形態に従って、クライアント202（図2）からのキャッシュされたデータがどのようにしてサーバ204（図2）にフラッシュされるの（すなわち、永続的に記憶される）かを示している。図2および図5を参照すると、クライアント202は、一実施形態によれば、以下で説明するようにサーバ204にデータをフラッシュする。

【0044】

ブロック502において、このデータを生成するアプリケーションは、呼出しを行うか

10

20

30

40

50

、あるいはこのトランザクションをコミットする要求を発行する。この呼出しまたは要求は、TM222に渡される。それに応答して、TM222は、（この実施例のトランザクションマネージャに関して以下で説明する）プリプリペア通知（Pre-prepare Notification）を生成する。

【0045】

この実施形態において、RDR220は、ブロック504に示すようにTM222からこのプリプリペア通知を受信する。それに応答してRDR220は、このネットワークを介してこのデータをSRV234に対してフラッシュする。SRV234は、次にこのデータをNTFS216Aに渡す。これらのオペレーションは、ブロック506によって示される。一部の実施形態においては、サーバ204のTM222Aは、いつこのプリプリペアオペレーションが完了するかについてRDR220に信号で伝える。ブロック504および506は、（実施例のトランザクションマネージャに関して以下で説明する）プリペア（Prepare）オペレーションが実施される以前に、サーバ204に書き込むべきクライアント202からのデータが、サーバ204上に存在するようにするのを助ける。

10

【0046】

ブロック508において、RDR220は、（実施例のトランザクションマネージャに関して以下で説明する）プリペア通知をTM222から受信する。一実施形態においては、RDR220は、このプリペア通知に応答してプリペア通知メッセージをサーバ204に送信し、このプリペア通知は、TM222Aに渡される。次にTM222Aは、このプリペア通知をNTFS216Aに渡す。これらのオペレーションは、ブロック510および512によって示される。このプリペア通知は、クライアント202およびサーバ204が、このデータをコミットし、またはロールバックすることができるような方法で、このデータを記憶するようにする。一部の実施形態においては、サーバ204のTM222Aは、RDR220にいつこのプリペアオペレーションが完了するかを信号で伝える。次いで、このデータは、ブロック514で示すように、標準の2相コミットオペレーション（例えば、このトランザクションをコミットし、または、中止するオペレーション）を使用して処理される。

20

【0047】

以上では、トランザクションマネージメントは、別々のTMコンポーネント（すなわち、TM222およびTM222A）を使用して実施されるものとして説明しているが、他の実施形態においては、このトランザクションマネージメントインフラストラクチャは、このファイルシステムインフラストラクチャに一体化することができる。さらに、このような一体化された実施形態においては、これらのトランザクションメッセージ（例えば、以下で説明するようにプリプリペア（Pre-prepare）、プリペア（Prepare）、コミット（Commit）、アボート（Abort）など）は、この伝送チャネル上をファイルメッセージと共に流れる。

30

【0048】

実施例のトランザクションマネージャ

図6は、一実施形態による、トランザクションを実施する際に使用されるコンポーネントを示している。特定のトランザクションを構成するオペレーションのグループは、少なくとも当業者には「原子性（atomicity）」、「一貫性（consistency）」、「隔離性（isolation）」、および「永続性（durability）」を含む頭字語「ACID」として知られている特性を、全体として有している。より詳細には、トランザクションの各オペレーションからもたらされるデータの更新は、すべて不変であるか、あるいはどれも不変でなく（原子性）、トランザクションは、基礎をなすデータを一貫した状態にし（一貫性）、トランザクション更新の効果は、全体のトランザクションが不変となるまでは他の同時に実行されるオペレーションから見え（隔離性）、またトランザクションに対する結果が決定された後は、この結果は決して変化しないことが保証される（永続性）。

40

50

【0049】

図6のカーネルレベルのトランザクションマネージメントの実施例は、複数の装置に与する分散トランザクションの実施例を対象としており、トランザクションに期待される「ACID」特性を保持している。さらに、図6の実施例は、カーネルオブジェクトを参照しているが、この実施例は、決してカーネルオブジェクトによって実装されるトランザクションだけに限定されるものではない。より詳細には、本明細書中で説明するトランザクションは、カーネルオブジェクト以外のオブジェクトによっても実装することができ、また異なるタイプのトランザクションマネージャによっても実装することができる。

【0050】

図6において、クライアントアプリケーション600に対応するトランザクションは、少なくとも第1のデバイス上のトランザクションマネージャ605、ならびに第2のデバイス上のクライアントアプリケーション600Bおよびトランザクションマネージャ635を利用している。クライアントアプリケーション600Bは、クライアントアプリケーション600に関連付けられている。互いに情報をやりとりするトランザクションマネージャ605および635は、全体のトランザクションおよびリソースについての状態情報を保持し、さらにクライアントアプリケーションと関連するRM (resource manager リソースマネージャ) の間で相互作用またはプロトコルを調整することができるカーネルオブジェクトの集合体とすることができる。

【0051】

図6の実施例におけるRM625およびRM630を含めてリソースマネージャは、データを永続的状态に記憶することができる少なくとも1つの基礎をなすリソースの状態を保持する。このようなリソースの非排他的な実施例は、データベースおよびメッセージキュー (message queue) を含んでいる。図6の実施形態の例の第1のデバイスにおいて、RM625はリソース627に対応し、RM630はリソース632に対応し、第2のデバイスにおいて、RM655はリソース657に対応する。

【0052】

図6に示すように、第1のデバイス上のトランザクションマネージャ605は、以下のカーネルオブジェクト: TX (transaction object トランザクションオブジェクト) 610、RMO (resource manager object リソースマネージャオブジェクト) 615、およびEN (enlistment object エンリストメントオブジェクト) 620を含んでおり、第2のデバイス上のトランザクションマネージャ635は、以下のカーネルオブジェクト: TX640、RMO645、およびEN650を含んでいる。TXは特定のトランザクションを表し、このトランザクションに参加するプロセスによってオープンにすることができる。

【0053】

RMOは、特定のトランザクションに参加するリソースを表す。トランザクションへのRMOによる参加は、2相コミットメッセージを受信することを含んでいる。さらに、RMOは永続的であり、その結果、この対応するトランザクションマネージャは、どのトランザクションの結果が対応するRMに伝送されるかを知っている。代わりに、RMOが過渡的なこともあり、したがって、失敗を超えて永続的なRMOを管理することなく、クライアントアプリケーションがトランザクション通知のストリームに加入することができるようになる。

【0054】

ENは、トランザクションとリソースマネージャとの間の関係を表す。リソースマネージャは、EN上にエンリストメントを作成することによってENがトランザクションに参加することになることを示す。RMOが、特定のトランザクションに関して (Prepare や Commit などの) オペレーションを実施するように要求されているときには、RMOは、ENを使用して参加を示す。リソースマネージャは、特定のトランザクションに関して複数のENを有することもできる。

【0055】

10

20

30

40

50

以下のように、カーネル環境に対して、トランザクションが正常にすべての適切なファイルをアップデートするようにするために実装される2相コミットプロトコルが、図6および図7の実施例を参照しながら説明される。特に、クライアントアプリケーション600が、第1のデバイス上のトランザクションマネージャ605に対応するカーネルオブジェクトをオープンし、SRV234(図2)が、第2のデバイス上のトランザクションマネージャ635に対応するカーネルオブジェクトをオープンした後に、「プリペア(prepare)」フェーズ700が開始され、705において、このトランザクション中の各RMには、対応するトランザクションマネージャからの「プリペア」命令が送信される。このように警告を受けて、710において、RMはリソースデータを永続的状态にすることによってプリペアを行い、その結果、各リソース中のこのデータは、「コミットされ」または「ロールバックされる」ことが可能になる。プリペアが行われると、715において、RMはこの対応するトランザクションマネージャのTXに対して確認メッセージを送信する。 10

【0056】

「コミット」フェーズ720は、このトランザクションが解決すると実施され、それによって725においてこのトランザクションマネージャのTXは、各関連付けられたRMに対して「コミットされ」または「アボート／ロールバックされた」トランザクション結果を送信する。次いで、RMは、この結果を関連付けられたログに記録し、この基礎となるリソースデータは、このトランザクション結果に従ってコミットされ、あるいはロールバックされる。代替的な実施形態においては、このトランザクションについてのデータが永続的でない揮発性のエンリストメントを可能とすることができ、したがってこのデータは、ログ記録もされず、また復元もされない。 20

【0057】

カーネルレベル上のトランザクションマネージメントは、それだけには限定されないがMicrosoft(登録商標)Win32(登録商標)アプリケーションプログラミングインターフェースおよびMicrosoft(登録商標)Windows(登録商標)オペレーティングシステムを含めてシステムアーキテクチャに適用可能なAPI(application program interfaceアプリケーションプログラムインターフェース)を利用することによって実装することができる。本明細書中で説明しているAPIは、ハンドルベースのインターフェースを介して表示され、「ハンドル」は、APIを対象とするオブジェクトを参照する。さらに、非同期式のオペレーションが明示的に要求されない限り、各カーネルオブジェクト上、特にTXおよびRMO上のオペレーションは、同期式である。さらにまた、トランザクションの異なる実施形態に対応するオペレーションは、本明細書中で説明している1つまたは複数のAPIの様々な組合せによって実装することができる。すなわち、一部の実施形態では、本明細書中で説明しているAPIのすべてを使用することができるが、他の実施形態では、それらの様々な組合せを使用することができる。 30

【0058】

TXカーネルオブジェクト上でオペレーションを実装するAPI、およびこのAPIの機能の対応する記述が、以下に提供される(この関連するルーチンのさらに詳細な記述も、以下にさらに提供される)。 40

【0059】

・PreprepareEnlistment:「フェーズ0」の処理としても知られており、TXがすべての関連するRMに対してプリプリペアメッセージを送信することを要求する。

【0060】

・PrepareEnlistment:TXがすべてのエンリストされたRMに対してプリペア要求を送信することを要求する。

【0061】

・CreateTransaction:新しいTXをオープンする。 50

【0062】

・OpenTransaction：既存のTXをオープンする。

【0063】

・CommitTransaction：TXがコミットされることを要求する。

【0064】

・RollbackTransaction：TXがこのトランザクションをアボートまたはロールバックすることを要求する。

【0065】

・SavepointTransaction：TXがこのトランザクションの状態を保存することを要求する。

10

【0066】

・GetTransactionInfo：このTXについての情報を検索する。

【0067】

・SetTransactionInfo：このTXについての情報を設定する。

【0068】

RMOカーネルオブジェクト上でオペレーションを実装するために利用されるAPI、およびこのAPIの機能の対応する記述が、以下に提供される（この関連するルーチンのさらに詳細な記述も、以下にさらに提供される）。

【0069】

CreateResourceManager：リソースを表す新しいRMOを作成する。

20

【0070】

OpenResourceManager：既存のRMOをオープンする。

【0071】

DestroyResourceManager：RMOを破壊し、したがってRMOを非永続的なものにする。

【0072】

GetResourceManagerInfo：RMOについての情報を検索する。

【0073】

SetResourceManagerInfo：RMOについての情報を設定する。

30

【0074】

CreateEnlistment：RMOをトランザクションに参加させ、関連した通知を検索する。

【0075】

GetNotificationResourceManager：使用可能なRM通知を照会し戻す。

【0076】

トランザクションに参加した後、RMOカーネルオブジェクトによってTXカーネルオブジェクト上でオペレーションを実装するために利用されるAPI、およびこのAPIの機能の対応する記述が、以下に提供される（この関連するルーチンのさらに詳細な記述も、以下にさらに提供される）。

40

【0077】

・PrePrepareComplete：対応するトランザクションマネージャが要求するときに、RMがプリプリペアを完了していることを示す。

【0078】

・PrepareComplete：対応するトランザクションマネージャが要求するときに、RMがトランザクションのプリペアを完了していることを示す。

【0079】

50

・RollbackComplete：対応するトランザクションマネージャが要求するときに、RMが実行されたトランザクション作業のロールバックを完了していることを示す。

【0080】

・CommitComplete：対応するトランザクションマネージャが要求するときに、RMがトランザクション作業のコミットを完了していることを示す。

【0081】

残念ながら、トランザクションマネージメントを実装するために利用されるTX、RMO、およびENに関連するAPIは、様々なセキュリティ攻撃に対して1つまたは複数のカーネルオブジェクトをさらすこともある。例えば、悪意のあるまたは不正なRMが、トランザクション中にそれ自体をエンリストしてファンクション呼出しに決して応答しないことによってサービス拒否攻撃を引き起こし、あるいは、代わりにトランザクションアボートを強制することができる。したがって、図6をまた参照すると、さらなる例示の実施例は、安全なカーネルレベルの分散トランザクションを対象としている。

【0082】

図6の実施形態の一例はさらに、各カーネルオブジェクトの少なくとも1つに対してACL(access control listアクセスコントロールリスト)を含むことができるセキュリティデスク립タ(security descriptor)を適用することにより、ぜい弱なカーネルオブジェクトに対するセキュリティソリューションを提供する。

【0083】

第1のデバイスにおいては、ACL660はTX610に適用され、ACL665はRMO615に適用され、ACL670はEN620に適用される。第2のデバイスにおいては、ACL675はTX640に適用され、ACL680はRMO645に適用され、ACL685はEN650に適用される。

【0084】

ACLは、特定のユーザまたはユーザグループが特定のオブジェクト上で行使することを許可または拒否される「権利(right)」を定義する。より詳細には、図8の実施例ACL810に示すように、カーネルオブジェクトに適用またはアタッチされるACLは、対応するSID(security identifierセキュリティ識別子)および対応する1組の権利を含む少なくともACE(access control entryアクセス制御エントリ)を含んでいる。図8のACEエントリ1~12は、それぞれ対応するSID1~12、および対応するRIGHT1~12を含んでいる。

【0085】

SID1~12は、このACLが適用されるカーネルオブジェクト上でオペレーション、または一連のオペレーションを実装しようと試みることができるユーザまたはユーザグループを識別する。RIGHT1~12は、このSIDによって識別されるユーザまたはユーザグループがこの各カーネルオブジェクト上で実施することができるオペレーションまたは一連のオペレーションを指定し、さらにこの識別されたユーザまたはユーザグループへのこのようなオペレーションまたは複数のオペレーションのアクセス可能性を指定する。すなわち、RIGHT1~12は、識別されたユーザまたはユーザグループが、指定されたオペレーションを実施することが許可されること、あるいは、この指定されたユーザまたはユーザグループが、指定されたオペレーションを実施することが禁止されることを示すことができる。

【0086】

以下は、TXに対して適用されるACLにおいてRIGHT1~12が指定することができるオペレーションの例のリストであり、このオペレーションの機能の記述が後に続いている。RIGHT1~12はさらに、この対応するSIDによって識別されるユーザまたはユーザグループに対してTX上でこのオペレーションが許可または拒否されることを指定する。

10

20

30

40

50

【0087】

・TRANSACTION__QUERY__INFORMATION: TXについての情報を獲得する。

【0088】

・TRANSACTION__SET__INFORMATION: TXについての情報を設定する。

【0089】

・TRANSACTION__ENLIST: このトランザクション中のTX上にエンリストする。

【0090】

・TRANSACTION__COMMIT: TXに関連するすべてのデータアップデートを永続的にする。

【0091】

・TRANSACTION__ROLLBACK: TX上でこのオペレーションをアボート、すなわちロールバックする。

【0092】

・TRANSACTION__PROPAGATE: TXから別のオブジェクトへとデータを伝送する。

【0093】

・TRANSACTION__SAVEPOINT: このトランザクションの最新ポイントを保存する。

【0094】

・TRANSACTION__MARSHAL: このトランザクションに関するデータを別のデバイスへと伝送する。

【0095】

以下は、RMOに対して適用されるACLにおいてRIGHT 1～12が指定することができるオペレーションの例のリストであり、このオペレーションの機能の記述が後に続いている。RIGHT 1～12はさらに、この対応するSIDによって識別されるユーザまたはユーザグループに対してRMO上でこのオペレーションが許可または拒否されることを指定する。

【0096】

・RESOURCEMANAGER__QUERY__INFORMATION: RMOについての情報を獲得する。

【0097】

・RESOURCEMANAGER__SET__INFORMATION: RMOについての情報を設定する。

【0098】

・RESOURCEMANAGER__RECOVER: トランザクションの失敗の瞬間におけるトランザクションの状態を決定する。

【0099】

・RESOURCEMANAGER__ENLIST: トランザクションにおいてRMOをエンリストする。

【0100】

・RESOURCEMANAGER__GET__NOTIFICATION: トランザクションマネージャからのトランザクションの解決に応じて通知を受信する。

【0101】

・RESOURCEMANAGER__REGISTER__PROTOCOL: このトランザクションにおいてRMOがサポートするプロトコルを登録する。

【0102】

・RESOURCEMANAGER__COMPLETE__PROPAGATION: ト

10

20

30

40

50

ランザクションの解決に従ってリソースを設定する。

【0103】

以下は、ENに対して適用されるACLにおいてRIGHT 1～12が指定することができるオペレーションの例のリストであり、このオペレーションの機能の記述が後に続いている。RIGHT 1～12はさらに、この対応するSIDによって識別されるユーザまたはユーザグループに対してEN上でこのオペレーションが許可または拒否されることを指定する。

【0104】

・ENLISTMENT__QUERY__INFORMATION：ENについての情報を獲得する。

10

【0105】

・ENLISTMENT__SET__INFORMATION：ENについての情報を設定する。

【0106】

・ENLISTMENT__RECOVER：トランザクションの失敗の瞬間におけるエンリストメントの状態を決定する。

【0107】

・ENLISTMENT__REFERENCE：エンリストメントキーを取得し、参照（または逆参照）する。

【0108】

・ENLISTMENT__SUBORDINATE__RIGHTS：このトランザクションをロールバックし、通知に応答する。

20

【0109】

・ENLISTMENT__SUPERIOR__RIGHTS：トランザクション中でプリペアオペレーション、プリペアオペレーション、または上位ロールバックオペレーションを開始するなど、上位トランザクションマネージャが実施するはずのオペレーションを実施する。

【0110】

したがって、各カーネルオブジェクトTX、RMO、およびENはそれぞれそれに適用されるACLを有することができる。このように、APIがこれらのカーネルオブジェクトのそれぞれ1つの上にあるオペレーションを開始しようと試みるときに、このAPIが起源とするユーザまたはユーザグループに対して、このオペレーションが許可または拒否されるかどうかを判定して、このACLに従う必要がある。

30

【0111】

より詳細には、オペレーションを実施するためのハンドルがオープンされるとき、このAPIに対応するユーザまたはユーザグループはこのACL中のSIDに対して検査され、許可されたオペレーションのリストが生成され、このAPIによって指定されるこのオペレーションは所与のハンドル上のこのSIDに対して許可されたオペレーションに対して検査される。

【0112】

カーネルオブジェクトの間のトランザクションマネージメントをセキュリティ保護し、セキュリティパラメータを強要する代替的な実施形態は、このMicrosoft（登録商標）Windows（登録商標）オペレーティングシステム用のセキュリティモデルによるトランザクションに参加することができるカーネルオブジェクトにセキュリティデスク립タを適用する工程を含んでいる。

40

【0113】

上述のように、これらのAPIは、ハンドルベースのインターフェースとして表示され、このインターフェースを利用してこのセキュリティモデルを実装する。以上でリストアップしたTXカーネルオブジェクト上でオペレーションを実装するこれらのAPIのより詳細な記述が、以下に含まれている。これらの記述は、このルーチン、対応する引数、お

50

よびリターン値の記述を含んでいる。

【0114】

PreprepareEnlistment

(IN PHANDLE TransactionHandle;
IN PHANDLE ResourceManagerHandle)。

【0115】

・このルーチンは、すべての関連するRMに対してプリプリペア要求 (Pre-Prepare request) を発行することによってトランザクションが「プリプリペアされる」ことを要求する。プリプリペアは、このトランザクションが、下流のRMがもはや変更を受け入れることができないというプリペア状態 (Prepared state) に入る以前に、キャッシュに似た特性を有するRMにそのキャッシュを、可能であれば他のRMに対してフラッシュする機会を許可する。 10

【0116】

・このルーチンが呼び出されず、トランザクション参加者がフェーズ0の処理を要求している場合、プリペア (Prepare) が受信されるときに要求されるように、プリプリペア要求が発行される。しかし、PreprepareEnlistmentが存在しない場合には、キャッシュに似たRMを含む一部の構成では、分散シナリオにおいて、不必要なトランザクションロールバックが引き起こされることもある。

【0117】

・引数：

TransactionHandle：プリプリペアすべきトランザクションを示すハンドルを与える。 20

【0118】

ResourceManagerHandle：このトランザクションをプリプリペアしている上位-TM/CRMに対してハンドルを与える。この上位-TM/CRMだけが、このトランザクション上でPrepareEnlistment、SuperiorCommitTransaction、およびSuperiorRollbackTransactionを呼び出すことが可能になる。

【0119】

・リターン値：

【0120】

【表8】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_TM_TOO_LATE 40

【0121】

PrepareEnlistment

(IN PHANDLE TransactionHandle,
IN PHANDLE ResourceManagerHandle)；

・このルーチンは、すべてのその関連付けられたResourceManagerに対してプリペア要求 (Prepare request) を発行することによって、トランザクションが「プリペアされる」ことを要求する。この要求は、2相コミットプロトコルを開始する。

【0122】

・ `PrepareEnlistment` を発行するトランザクション参加者は、トランザクションオブジェクトをシステムまたはアプリケーションがクラッシュしても生き残ることになる永続的状态にする。このような参加者は、どのようなタイプの失敗の後にもこのトランザクション上でリカバリ (`recovery`) を実施して、結果を送付する。この要件を満たすのに失敗すると、リソースのリーク、ならびに矛盾したトランザクションの結果がもたらされることもある。

【0123】

・ 引数：

`TransactionHandle`：プリペアすべきトランザクションについてのハンドルを与える。

10

【0124】

`ResourceManagerHandle`：このトランザクションをプリペアしている TM に対して、ハンドルを与える。このトランザクションが (`PrepareEnlistment` に対するコールを介して) プリプリペアされている場合には、`ResourceManagerHandle` は、`PrepareEnlistment` に対する呼出し中で使用された上位-TM/CRM とマッチングする。さらに、この API をコールする上位-TM/CRM だけが、このトランザクション上で `SuperiorCommitTransaction` および `SuperiorRollbackTransaction` を呼び出すことが許可されることになる。

【0125】

20

・ リターン値：

【0126】

【表 9】

`STATUS_SUCCESS`

`STATUS_ACCESS_DENIED`

`STATUS_INVALID_HANDLE`

`STATUS_INSUFFICIENT_RESOURCES`

30

`STATUS_TM_TOO_LATE`

`STATUS_RM_NOT_RECOVERABLE`

【0127】

`CreateTransaction`

(OUT PHANDLE

IN ULONG

OPTIONAL；

IN POBJECT_ATTRIBUTES

OPTIONAL；

IN ULONG

OPTIONAL；

IN PHANDLE

OPTIONAL；

IN NOTIFICATION_MASK

OPTIONAL；

IN LPVOID

OPTIONAL)。

`TransactionHandle`、
`DesiredAccess`

`ObjectAttributes`

`CreateOptions`

`ResourceManagerHandle`

`NotificationMask`

`TransactionKey`

40

【0128】

50

・このルーチンは、新しいトランザクションオブジェクトを作成し、この新しいオブジェクトにハンドルを戻す。

【0129】

・代わりに (ResourceManagerHandle パラメータが指定される場合)、このルーチンは、それが正常に作成された後に、このトランザクション上で「ジョイン (Join)」オペレーションを実施する。

【0130】

クライアントは、CloseHandle API を使用してこのトランザクションハンドルをクローズする。この最後のトランザクションハンドルが、いずれかがこのトランザクション上で CommitTransaction を呼び出すことなくクローズする場合には、このトランザクションは、暗黙的にロールバックされる。 10

【0131】

・引数：

TransactionHandle：この新しいトランザクションに対するハンドルを受信することになるロケーションにポインタを与える。

【0132】

DesiredAccess：この所望のアクセスレベルを指定するマスクを与える。この有効なアクセスマスクの選択肢には、以下のものがある。：

SYNCHRONIZE このハンドル上で同期オペレーションを実施することができる。 20

【0133】

TRANSACTION_COMMIT このハンドルを使用してトランザクションをコミットすることができる。

【0134】

TRANSACTION_PREPARE このハンドルを使用してトランザクションをコミットすることができる。

【0135】

TRANSACTION_ROLLBACK このハンドルを使用してトランザクションをアボートすることができる。

【0136】

TRANSACTION_SAVEPOINT このハンドルを使用してこのトランザクションについてのセーブポイントを作成することができる。 30

【0137】

TRANSACTION_JOIN このハンドルを使用してこのトランザクションをRMとして加えることができる。

【0138】

TRANSACTION_READ_ATTRIBUTES トランザクションに関連する属性を読み取ることができる。

【0139】

TRANSACTION_WRITE_ATTRIBUTES トランザクションに関連する属性を書き込むことができる。 40

【0140】

ObjectAttributes：オプションのオブジェクト属性構造に、ポインタを与える。

【0141】

CreateOptions オプションのトランザクションフラグを与える。有効な作成フラグ選択肢は以下を含んでいる。

【0142】

TRANSACTION_CREATE_PRESUMED_NOHING 「何も想定しない (presumed nothing)」トランザクションを作成する 50

。

【0143】

ResourceManagerHandle：指定されたトランザクションについての通知を受信する**ResourceManager**にハンドルを与える。

【0144】

NotificationMask：この**ResourceManager**がこのトランザクションに関して受信したい通知を指定する。

【0145】

TransactionKey：このRMがこのトランザクションについての任意の通知とともに受信をしたい不明瞭なポインタ値を指定する。このRMは、このポインタ値を使用して通知をこのRMのアドレス空間中のあるオブジェクトに関連づけることができ、それによって通知が行われるたびにルックアップを実行する必要をなくすることができる。

【0146】

・リターン値：

【0147】

【表10】

STATUS_SUCCESS

STATUS_INVALID_PARAMETER

STATUS_OBJECT_NAME_COLLISION

STATUS_OBJECT_NAME_INVALID

STATUS_PRIVILEGE_NOT_HELD

STATUS_INSUFFICIENT_RESOURCES

【0148】

OpenTransaction

(OUT PHANDLE	TransactionHandle、
IN ACCESS_MASK	DesiredAccess、
IN POBJECT_ATTRIBUTES	ObjectAttributes、
IN PHANDLE	ResourceManagerHandle
optional、	
IN NOTIFICATION_MASK	NotificationMask
optional、	
IN LPVOID	TransactionKey
optional) ;	

・このルーチンは、既存のトランザクションオブジェクトを調べ、ハンドルをこのトランザクションに戻す。この呼出し元は、**ObjectAttributes**の**ObjectName**フィールド中において**GUID**のストリング表現を指定する。

【0149】

・代わりに（この**ResourceManagerHandle**パラメータが指定される場合）、このルーチンはまた、それがオープンされた後にこのトランザクション上で「ジョイン」オペレーションを実施する。

【0150】

・クライアントは、**CloseHandleAPI**を使用してこのトランザクションハンドルをクローズする。この最後のトランザクションハンドルが、どれかがこのトランザクション上で**CommitTransaction**をコールすることなくクローズする場合

合には、このトランザクションは、暗黙的にロールバックされたトランザクションである。

【0151】

・引数：

`TransactionHandle`：このオープンオペレーションが成功する場合に、このトランザクションに対するハンドルを受信することになるロケーションにポインタを与える。

【0152】

`DesiredAccess`：この所望のアクセスレベルを指定するマスクを与える。

10

【0153】

`ObjectAttributes`：オプションのオブジェクト属性構造にポインタを与える。

【0154】

`ResourceManagerHandle`：この指定されたトランザクションについての通知を受信する `ResourceManager` にハンドルを与える。

【0155】

`NotificationMask`：この `ResourceManager` がこのトランザクションに関して受信することができる通知を指定する。

【0156】

`TransactionKey`：このRMがこのトランザクションについての任意の通知とともに受信したい不明瞭なポインタ値をオプション的に指定する。このRMは、このポインタ値を使用して通知をこのRMのアドレス空間中のあるオブジェクトに関連づけることができ、それによって通知が行われるたびにルックアップを実行する必要をなくすることができる。

20

【0157】

・リターン値：

【0158】

【表11】

`STATUS_SUCCESS`

30

`STATUS_INVALID_PARAMETER`

`STATUS_OBJECT_NAME_INVALID`

`STATUS_OBJECT_NAME_NOT_FOUND`

`STATUS_OBJECT_PATH_SYNTAX_BAD`

`STATUS_PRIVILEGE_NOT_HELD`

40

`STATUS_INSUFFICIENT_RESOURCES`

【0159】

`CommitTransaction`

(IN PHANDLE `TransactionHandle`
IN ULONG `CommitOptions` Optional) ;

・このルーチンは、`TransactionHandle`に関連するこのトランザクションがコミットされることを要求する。オープンまたは作成されている任意のトランザクションハンドルは、`TransactionCommitDesiredAccess`を用いてコミットすることができる。トランザクションの作成者だけがそれをコミッ

50

トすることが可能であることを提示する制約は、存在していないからである。

【0160】

・問題のトランザクションが、以前に `PrepareEnlistment` 要求を発行していない場合に、2相コミットプロトコルは、すべてのエンリストされたRM上で開始することができる。この呼出しは、このクライアントによって発行されている1相コミット要求と考えることができる。

【0161】

・このトランザクションが以前に `PrepareEnlistment` を介してプリペアされている場合には、このルーティング (`routing`) は、呼び出されない。`PrepareEnlistment` を呼び出したRMだけが、このルーチン `SuperiorCommitTransaction` を使用して、このトランザクション状態を解決することができる。

10

【0162】

・引数：

`TransactionHandle`：コミットすべきトランザクションを示すハンドルを与える。

【0163】

`CommitOptions`：`COMMIT_RETAINING` トランザクションがコミットされることになる。

【0164】

・リターン値：

20

【0165】

【表12】

`STATUS_SUCCESS`

`STATUS_ACCESS_DENIED`

`STATUS_INVALID_HANDLE`

`STATUS_INSUFFICIENT_RESOURCES`

`STATUS_TM_TRANSACTION_ABORTED`

30

【0166】

`RollbackTransaction`

(IN PHANDLE `TransactionHandle`、
IN SAVEPOINT `SavePoint` Optional、
IN ROLLBACK_REASON `RollbackReason` Optional)；

・このルーチンは、`TransactionHandle`に関連するこのトランザクションをロールバックすることを要求する。このオプションの `SavePoint` が指定され、有効なセーブポイントである場合、このロールバックは、部分的なロールバックでもよい。`NULL` の `SavePoint` 引数は、このトランザクションが完全にロールバックまたはアボートされるべきことを示す。オプションの `RollbackReason` 構造を与えることができる。これは、トランザクションオブジェクト中に保持されることになり、`GetInformationTransaction` に対する呼出しを介して関心のあるトランザクション参加者は検索することができる。

40

【0167】

・引数：

`TransactionHandle`：ロールバックすべきトランザクションを示すハンドルを与える。

【0168】

50

SavePoint : どこまでトランザクションの状態をロールバックすべきかを示す**SavePoint**ネームを与える。

【0169】

RollbackReason : ロールバックリズン (**rollback reason**) を与える。

【0170】

・リターン値 :

【0171】

【表13】

STATUS_SUCCESS

10

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_TM_TRANSACTION_COMMITTED

【0172】

SavePointTransaction

20

(IN PHANDLE TransactionHandle、
IN ULONG SavePointFlags Optional、
OUT LPSAVEPOINT SavePoint) ;

・このルーチンは、「セーブポイント」が**TransactionHandle**に関連付けられたトランザクションについて生成されることを要求する。このセーブポイントは、後続のロールバック要求に対するターゲットとして使用される。

【0173】

・引数 :

TransactionHandle : セーブポイントを確立すべき対象のトランザクションを示すハンドルを与える。

30

【0174】

SavePointFlags : このセーブポイントの生成に影響を及ぼす1組のフラグをオプション的に与える。

【0175】

SavePoint : セーブポイント識別子が記憶されるロケーションにポインタを与える。

【0176】

・リターン値 :

【0177】

【表 1 4】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_TM_TRANSACTION_COMMITTED

10

STATUS_TM_TRANSACTION_ABORTED

【0 1 7 8】

QueryInformationTransaction

(IN HANDLE TransactionHandle、

IN TRANSACTION_INFORMATION_CLASS

TransactionInformationClass、

OUT PVOID

TransactionInformation、

IN ULONG

20

TransactionInformationLength、

OUT PULONG

ReturnLength Optional);

・このルーチンは、TransactionHandleによって表されるトランザクションオブジェクトについての要求された情報を戻す。

【0 1 7 9】

・引数：

TransactionHandle：情報が要求されているトランザクションを示すハンドルを与える。

【0 1 8 0】

TransactionInformationClass：このトランザクションオブジェクトについてのどのクラスの情報が要求されているかを示す。 30

【0 1 8 1】

TransactionInformation：要求されているこのトランザクション情報が記憶されるバッファに対してポインタを与える。

【0 1 8 2】

TransactionInformationLength：TransactionInformationによってポイントされるバッファの長さを示す。

【0 1 8 3】

ReturnLength：このTransactionInformationバッファに書き込まれる情報の長さを受信することになるロケーションにポインタを与える 40

【0 1 8 4】

・リターン値：

【0 1 8 5】

【表 1 5】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_INVALID_INFO_CLASS

10

STATUS_INFO_LENGTH_MISMATCH

【0 1 8 6】

SetInformationTransaction

(IN HANDLE TransactionHandle、

IN TRANSACTION_INFORMATION_CLASS

TransactionInformationClass、 IN PVOID

TransactionInformation、

IN ULONG

TransactionInformationLength) ;

20

・このルーチンはTransactionHandleによって表されるこのトランザクションオブジェクトについてこの要求された情報を設定する。

【0 1 8 7】

・引数：

TransactionHandle：どの情報が修正されることになるかをこのトランザクションに示すハンドルを与える。

【0 1 8 8】

TransactionInformationClass：このトランザクションオブジェクトについてのどのクラスの情報が要求されているかを示す。

【0 1 8 9】

TransactionInformation：要求されるこのトランザクション情報が記憶されるバッファにポインタを与える。

30

【0 1 9 0】

TransactionInformationLength：TransactionInformationによってポイントされるバッファの長さを示す。

【0 1 9 1】

ReturnLength：このTransactionInformationバッファに書き込まれる情報の長さを受信することになるロケーションにポインタを与える。

【0 1 9 2】

40

・リターン値：

【0 1 9 3】

【表 1 6】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_INVALID_INFO_CLASS

STATUS_INFO_LENGTH_MISMATCH

10

【0 1 9 4】

以下は、以上でリストアップした、RMOカーネルオブジェクト上でオペレーションを実装するAPIのより詳細な記述を含んでいる。これらの記述は、このルーチン、対応する引数、およびリターン値の記述を含んでいる。

【0 1 9 5】

CreateResourceMnager

(OUT PHANDLE	ResourceManagerHandle、	20
IN ACCESS_MASK	DesiredAccess	
Optional、		
IN POBJECT_ATTRIBUTES	ObjectAttributes、	
IN ULONG	CreateOptions	
Optional、		
IN RM_NOTIFICATION_ROUTINE	NotificationRoutine	
Optional) ;		

・このルーチンは、リソースを表す新しいResourceManagerオブジェクトを作成する。

【0 1 9 6】

・ResourceManagerオブジェクトは、このRMが参加したという、トランザクションに関するTM通知についてのエンドポイントとしての役割も果たす。RMは、GetNotificationResourceManagerを呼び出すことによってこれらの通知を要求する。

30

【0 1 9 7】

・ResourceManagerは、通常は永続的なオブジェクトであり、すなわちこのオブジェクトは、（システムまたはRMの）あらゆる失敗の後に再オープンされ、リカバリを実施する必要がある。ResourceManagerオブジェクトの過渡的なバージョンは、このオプションRESOURCEMANAGER_NO_RECOVERYを指定することによって作成することができる。過渡的なRMは、リカバリを実施するように義務づけられておらず、また許可されてもいない。このリカバリ不可能なRMオプションにより、この十分複雑なログ記録をとるプリペアを実装する必要なしに、またリカバリを実施することもなしに、アプリケーションまたはRMはトランザクションの進行（例えば、PREPREPARE、PREPARE、COMMIT）についての通知を受信できるようになる。

40

【0 1 9 8】

・引数：

ResourceManagerHandle：この新しいResourceManagerに対するハンドルを受信することになるロケーションにポインタを与える。

【0 1 9 9】

50

`DesiredAccess`：所望のアクセスレベルを指定するマスクを与える。有効なアクセスマスクの選択肢を以下に示す。：

`SYNCHRONIZE`：ハンドル上でオペレーションを同期させる。

【0200】

`RESOURCE MANAGER_DESTROY`：このリソースマネージャを破壊する。

【0201】

`RESOURCE MANAGER_READ_ATTRIBUTES`：リソースマネージャに関連する属性を読み取る。

【0202】

`RESOURCE MANAGER_WRITE_ATTRIBUTES`：リソースマネージャに関連する属性を書き込む。

【0203】

`ObjectAttributes`：この新しいRMオブジェクトについての属性を指定する。この属性は、RMネームを含んでいる。

【0204】

`CreateOptions`：この作成されたオブジェクトについてのオプションを指定する。

【0205】

`RESOURCEMANAGER_NO_RECOVERY`：`ResourceManager`オブジェクトは、非永続的であり、リカバリを実施しない。 20

【0206】

`RESOURCEMANAGER_COMMUNICATION`：`ResourceManager`は、他のコンピュータに対してどのように通信を行うかを知っている。`ResourceManager`を使用してトランザクションをマーシャリングまたはアンマーシャリングする (`unmarshall`) ことができる。

【0207】

`RESOURCEMANAGER_CLUSTER_RECOVERY`：`ResourceManager`は、失敗している可能性のあるファイルをログ記録した結果をどのようにして読み取り、このクラスタ中の他のノードに送付するべきかを知っている。`ResourceManager`を使用してクラスタ中のトランザクションをリカバリすることができる。 30

【0208】

`NotificationRoutine`：通知がこの `ResourceManager` にとって使用可能であるときに呼び出すべき通知ルーチンを指定する。

【0209】

・リターン値：

【0210】

【表17】

`STATUS_SUCCESS`

40

`STATUS_INVALID_PARAMETER`

`STATUS_OBJECT_NAME_COLLISION`

`STATUS_OBJECT_NAME_INVALID`

`STATUS_PRIVILEGE_NOT_HELD`

`STATUS_INSUFFICIENT_RESOURCES`

50

【0211】

OpenResourceManager

(OUT PHANDLE	ResourceManagerHandle、
IN ACCESS_MASK	DesiredAccess
Optional、	
IN POBJECT_ATTRIBUTES	ObjectAttributes、
IN ULONG	OpenOptions
Optional、	
IN RM_NOTIFICATION_ROUTINE	NotificationRoutine、
Optional)。	

10

【0212】

・このルーチンは、名前によって既存の Resource Manager オブジェクトをオープンする。ターゲットの Resource Manager オブジェクトが永続的であるが、現在オープンされていない場合、このオブジェクトは最初に「リカバリする」状態にあり、リカバリされる必要がある。リカバリが完了した後に、RecoveryCompleteResourceManager を呼び出す必要がある。

【0213】

・引数：

ResourceManagerHandle：既存の Resource Manager オブジェクトに対するハンドルを受信することになるロケーションにポインタを与える。 20

【0214】

DesiredAccess：このオブジェクトに対する所望のアクセスを指定するマスクを与える。

【0215】

ObjectAttributes：この新しい RM オブジェクトについての属性を指定する。

【0216】

OpenOptions：このオブジェクトについてのオプションを指定する。有効なオプションは、以下を含んでいる。 30

【0217】

RESOURCE__MANAGER__DETAILED__RECOVERY__NOTIFICATIONS：このリソースマネージャは、通常のリカバリ通知の代わりに、（通信エンドポイントについての追加の情報を伴う）詳細なリカバリ通知を受信する。

【0218】

NotificationRoutine：通知が、この Resource Manager にとって使用可能であるときに、呼び出されることになる通知ルーチンを指定する。

【0219】

・リターン値：

40

【0220】

【表 18】

STATUS_SUCCESS

STATUS_INVALID_PARAMETER

STATUS_OBJECT_NAME_INVALID

STATUS_OBJECT_NAME_NOT_FOUND

STATUS_OBJECT_PATH_SYNTAX_BAD

STATUS_PRIVILEGE_NOT_HELD

STATUS_INSUFFICIENT_RESOURCES.

10

【0221】

DestroyResourceManager

(IN PHANDLE ResourceManagerHandle) ;

・このルーチンは、ResourceManagerオブジェクトを破壊し、それがもはや永続的でないようにする。

【0222】

・引数：

ResourceManagerHandle：破壊すべきResourceManagerオブジェクトを指示するハンドルを与える。

20

【0223】

・リターン値：

【0224】

【表 19】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_TM_NEEDS_RECOVERY.

30

【0225】

QueryInformationResourceManager

(IN HANDLE ResourceManagerHandle、

IN RESOURCEMANAGER_INFORMATION_CLASS

ResourceManagerInformationClass、

OUT PVOID

ResourceManagerInformation、

IN ULONG

ResourceManagerInformationLength、

OUT PULONG

ReturnLength Optional)。

40

【0226】

・このルーチンは、ResourceManagerHandleによって表されるRMOについてのこの要求された情報を戻す。

【0227】

50

・引数：

`ResourceManagerHandle`：情報が要求されているこの `ResourceManager` を示すハンドルを与える。

【0228】

`ResourceManagerInformationClass`：この `ResourceManager` オブジェクトについてのどのクラスの情報が要求されているかを示す。

【0229】

`ResourceManagerInformation`：要求されるこの `ResourceManager` 情報が記憶されることになるバッファにポインタを与える。

10

【0230】

`ResourceManagerInformationLength`：`ResourceManagerInformation` によってポイントされるバッファの長さを示す。

【0231】

`ReturnLength`：`ResourceManagerInformation` バッファに書き込まれる情報の長さを受信するロケーションにポインタを与える。

【0232】

・リターン値：

【0233】

20

【表20】

`STATUS_SUCCESS`

`STATUS_ACCESS_DENIED`

`STATUS_INVALID_HANDLE`

`STATUS_INSUFFICIENT_RESOURCES`

`STATUS_INVALID_INFO_CLASS`

30

`STATUS_INFO_LENGTH_MISMATCH`

【0234】

`SetInformationResourceManager`

(`IN HANDLE` `ResourceManagerHandle`、

`IN RESOURCEMANAGER_INFORMATION_CLASS`

`ResourceManagerInformationClass`、

`IN PVOID` `ResourceManagerInformation`、

`IN ULONG` `ResourceManagerInformationLength`) ；

・このルーチンは `ResourceManagerHandle` によって表される `RM` `O` についてのこの要求された情報を設定する。

40

【0235】

・引数：

`ResourceManagerHandle`：情報が修正されているこの `ResourceManager` を示すハンドルを与える。

【0236】

`ResourceManagerInformationClass`：この `ResourceManager` オブジェクトについてのどのクラスの情報が要求されているかを示す。

【0237】

50

`ResourceManagerInformation`: 要求されるこの `ResourceManager` 情報が記憶されるバッファにポインタを与える。

【0238】

`ResourceManagerInformationLength`: `ResourceManagerInformation` によってポイントされるバッファの長さを示す。

【0239】

・リターン値:

【0240】

【表21】

10

`STATUS_SUCCESS`

`STATUS_ACCESS_DENIED`

`STATUS_INVALID_HANDLE`

`STATUS_INSUFFICIENT_RESOURCES`

`STATUS_INVALID_INFO_CLASS`

`STATUS_INFO_LENGTH_MISMATCH.`

20

【0241】

`CreateEnlistment`

(IN PHANDLE ResourceManagerHandle,
IN PHANDLE TransactionHandle,
IN NOTIFICATION_MASK NotificationMask,
IN LPVOID TransactionKey Optional);

・このルーチンは、RMOを特定のトランザクションに結合 (`joint`) させ、それに関連した通知を受信する。

【0242】

30

・この `CreateEnlistment` 呼出しは、べき等 (`idempotent`) であり、RMはこのルーチンを複数回呼び出して、その `NotificationMask` または `TransactionKey` を変更することができる。 `CreateEnlistment` に対する後続の呼出しは、このトランザクション上で新しいエンリストメントを作成せずに通知マスクおよびトランザクションキーを置換する。

【0243】

・ `NotificationMask` を使用して通知が複数回受信されることを要求することができる。例えば、`PREPREPARE` 通知を受信するRMは、`JoinTransaction` を呼び出し、`PREPREPARE` フラグを指定することによって別の通知を要求することができる。したがって、RMは、複数の `PREPREPARE` 要求を受信することができる。このような要求は拒絶されることもあり、このトランザクションが、この要求された通知が受信されているはずのポイントを以前に進んでいる場合には、これは望ましいこともある。例えば、あるRMが既に `PARE` に通知されているときに、`PREPREPARE` を要求することは認められない。

40

【0244】

・引数:

`ResourceManagerHandle`: RMにハンドルを供給してこの指定されたトランザクションについての通知を受信する。

【0245】

`TransactionHandle`: このRMが参加することを望むトランザクシ

50

ョンに対してハンドルを与える。

【0246】

NotificationMask: RMがこのトランザクションに関して受信したいと思う通知を指定する。有効なマスクは、以下のようになっており、これらは一緒にORをとることもできる。

【0247】

TRANSACTION__NOTIFY__MASK__RM: RM (PREPARE、COMMIT、ROLLBACK、SAVEPOINT) が望む一般の通知。

【0248】

TRANSACTION__NOTIFY__MASK__CRM: CRMまたは上位TM (PrePrepareComplete、PrepareComplete、CommitComplete、RollbackComplete、SavebackComplete) が望む一般の通知。 10

【0249】

TRANSACTION__NOTIFY__PREPREPARE: PrePrepareに対する通知。

【0250】

TRANSACTION__NOTIFY__PREPARE: PREPAREに対する通知。

【0251】

TRANSACTION__NOTIFY__COMMIT: COMMITに対する通知。 20

【0252】

TRANSACTION__NOTIFY__ROLLBACK: ROLLBACKに対する通知。

【0253】

TRANSACTION__NOTIFY__PREPREPARE__COMPLETE: PREPREPAREが完了しているという通知。

【0254】

TRANSACTION__NOTIFY__PREPARE__COMPLETE: PREPAREが完了しているという通知。 30

【0255】

TRANSACTION__NOTIFY__COMMIT__COMPLETE: COMMITが完了しているという通知。

【0256】

TRANSACTION__NOTIFY__ROLLBACK__COMPLETE: ROLLBACKが完了しているという通知。

【0257】

TRANSACTION__NOTIFY__SAVEPOINT__COMPLETE: SAVEPOINTが完了しているという通知。 40

【0258】

TransactionKey: RMがこのトランザクションについての任意の通知とともに受信したいと思う不明瞭なポインタ値を指定する。このRMは、このポインタ値を使用して通知をこのRMアドレス空間中のあるオブジェクトに関連づけ、それによって通知が行われるたびにルックアップを実施する必要をなくすることができる。

【0259】

・リターン値:

【0260】

【表 2 2】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_PARAMETER

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_TM_TOO_LATE.

10

【0 2 6 1】

GetNotificationResourceManager

(IN PHANDLE ResourceManagerHandle,
 IN PTRANSACTION_NOTIFICATION TransactionNotification,
 IN PLARGE_INTEGER Timeout Optional);

- ・ どれかが使用可能な場合、このルーチンは、RM通知を照会し、戻す。

【0 2 6 2】

- ・ 引数：

ResourceManagerHandle：通知を戻す対象のResourceManagerを示すハンドルを与える。

20

【0 2 6 3】

TransactionNotification：この第1の使用可能な通知で満たすべきTRANSACTION_NOTIFICATION構造にポインタを与える。

【0 2 6 4】

Timeout：使用可能になる通知を待ちながら、それ以降、呼出し側はブロックしたいと考える時刻を与える。このタイムアウトが満了するときに何も使用可能でない場合には、この呼出し側は、STATUS_TIMEOUTと共に戻る。

【0 2 6 5】

- ・ リターン値：

【0 2 6 6】

【表 2 3】

STATUS_SUCCESS

STATUS_TIMEOUT

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES.

40

【0 2 6 7】

以上でリストアップした、トランザクションに参加した後にRMOカーネルオブジェクトによってTXカーネル上でオペレーションを実装するこれらのAPIのより詳細な記述が、以下に含まれている。この記述は、このルーチン、対応する引数、およびリターン値を含んでいる。

【0 2 6 8】

PrePrepareComplete

(IN PHANDLE EnlistmentHandle) ;

50

・このルーチンは、このK T Mが要求するときに、R Mがトランザクションのプリプリペア処理（別名「フェーズ0」）を完了していることを示す。

【0 2 6 9】

・引数：

T r a n s a c t i o n H a n d l e：このプリプリペアオペレーションが完了しているトランザクションを示すハンドルを与える。

【0 2 7 0】

・リターン値：

【0 2 7 1】

【表 2 4】

10

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_TM_NOT_REQUESTED

【0 2 7 2】

20

PrepareComplete

(IN PHANDLE EnlistmentHandle)；

・このルーチンは、K T Mが要求するときに、このR Mがトランザクションのプリペアを完了していることを示す。

【0 2 7 3】

・引数：

T r a n s a c t i o n H a n d l e：このプリペアオペレーションが完了しているトランザクションを示すハンドルを与える。

【0 2 7 4】

・リターン値：

【0 2 7 5】

30

【表 2 5】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

STATUS_TM_NOT_REQUESTED

40

【0 2 7 6】

RollbackComplete

(IN PHANDLE EnlistmentHandle)；

・このルーチンは、要求されたように、R Mがトランザクションによって実施される作業のロールバックを正常に完了していることを示す。R Mが、要求されたようにこのトランザクションを正常にロールバックすることができない場合には、R Mは、R o l l b a c k T r a n s a c t i o nを介して完全なロールバックを求める要求を発行すべきである。

【0 2 7 7】

50

・引数：

`TransactionHandle`：このロールバックオペレーションが完了しているトランザクションを示すハンドルを与える。

【0278】

・リターン値：

【0279】

【表26】

`STATUS_SUCCESS`

`STATUS_ACCESS_DENIED`

10

`STATUS_INVALID_HANDLE`

`STATUS_INSUFFICIENT_RESOURCES`

`STATUS_TM_NOT_REQUESTED`

【0280】

`CommitComplete`

(`IN PHANDLE` `EnlistmentHandle`)；

・このルーチンは、要求されたように、RMが、トランザクションによって実施される作業のコミットを完了していることを示す。 20

【0281】

・引数：

`TransactionHandle`：このコミットオペレーションが完了しているトランザクションを示すハンドルを与える。

【0282】

・リターン値：

【0283】

【表27】

`STATUS_SUCCESS`

30

`STATUS_ACCESS_DENIED`

`STATUS_INVALID_HANDLE`

`STATUS_INSUFFICIENT_RESOURCES`

`STATUS_TM_NOT_REQUESTED.`

【0284】

さらに、伝搬ルーチン (`propagation routine`) をこのカーネルオブジェクトについて提供することができる。このようなルーチンの実施例を以下に示す。 40

【0285】

`RegisterProtocolAddressInformation`

(`IN HANDLE` `ResourceManager`、

`IN PROTOCOL_ID` `ProtocolId`、

`IN ULONG` `ProtocolInformationSize`、

`IN PVOID` `ProtocolInformation` `Optional`)。

【0286】

・このルーチンは、リソースマネージャを特定のプロトコルに対する通信リソースマネージャとして登録する。このルーチンはまた、情報のブロブをこのプロトコルに関連づけ 50

る。ただ1つのリソースマネージャしか、所与のマシン上のあるプロトコルに対して登録をすることができない。

【0287】

・引数：

ResourceManager：登録中のリソースマネージャにハンドルを与える

。

【0288】

ProtocolId：このプロトコルを識別するGUID。

【0289】

ProtocolInformationSize：ProtocolInformationのサイズ。 10

【0290】

ProtocolInformation：このプロトコルに関連するオプションのプロブ。

【0291】

・リターン値：

【0292】

【表28】

STATUS_SUCCESS

20

STATUS_INVALID_HANDLE

【0293】

MarshallTransaction

(IN PHANDLE TransactionHandle、
IN ULONG NumberOfProtocols、
IN PPROTOCOL_ID ProtocolArray、
IN ULONG BufferLength、
IN PVOID Buffer、
OUT PULONG BufferUsed Optional)。

30

【0294】

・このルーチンは、TransactionHandleに対応するこのトランザクションの表現がバッファ中に直列に配列されることを要求する。

【0295】

・引数：

TransactionHandle：このコミットオペレーションが完了しているトランザクションを示すハンドルを与える。

【0296】

NumberOfProtocols：このプロトコルアレイのサイズを示す。

【0297】

ProtocolArray：このトランザクションをマーシャリングするために使用することができるプロトコルを指定するPROTOCOL_ID (GUID) のアレイ。このアレイは、プリファランス (preference) によって順序づけすべきである (このアレイ中の第1のプロトコルは、優先のプロトコルであり、第2のプロトコルは、第2最優先のプロトコルであるなど)。

40

【0298】

BufferLength：使用可能なバッファの長さを与える。

【0299】

Buffer：このトランザクションの直列配列を記憶すべきバッファにポインタを与える。

50

【0300】

BufferUsed: バッファ中に書き込まれる実際のバイトを記憶すべきロケーションにポインタを与える。

【0301】

・リターン値:

【0302】

【表29】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

10

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

【0303】

GetProtocolAddressInformation

(IN ULONG AddressBufferSize,
OUT PVOID AddressBuffer,
OUT PULONG AddressBufferUsed Optional)。

20

【0304】

・このルーチンは、このマシン上ですべての登録済みのプロトコルを表現する情報が、AddressBuffer中で、直列に配列されることを要求する。次いでこの情報を別のマシンに渡し、これをPushTransactionに対する引数として使用してAddressInformationが生成されるマシンに対してトランザクションを転送(push)することができる。

【0305】

・引数:

AddressBufferSize: 使用可能なバッファの長さを与える。

【0306】

AddressBuffer: 使用可能なバッファの長さを与える。

30

【0307】

AddressBufferUsed: このトランザクションの直列配列が記憶されるバッファのロケーションにポインタを与える。

【0308】

・リターン値:

【0309】

【表30】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

40

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

【0310】

PullTransaction

(OUT PHANDLE TransactionHandle,
IN ULONG NumberOfProtocols,

50

IN PCRM_PROTOCOL_ID ProtocolArray、
 IN ULONG BufferLength、
 IN PVOID Buffer)。

【0311】

・このルーチンは、バッファ中の直列配列によって表現されるトランザクションが、トランザクションマネージャにより使用可能になることを要求する。このトランザクションが、登録済みのリソースマネージャのうちの1つによって正常に伝搬された後に、この新しいトランザクションオブジェクトに対するハンドルが戻される。

【0312】

・引数：

TransactionHandle：この新しいトランザクションを表すハンドルが記憶されるべきところにポインタを与える。

【0313】

NumberOfProtocols：このプロトコルアレイのサイズを示す。

【0314】

ProtocolArray：このトランザクションをマーシャリングするために使用することができるプロトコルを指定するPROTOCOL_ID (GUID) のアレイ。このアレイは、プリファランスによって順序づけすべきである（このアレイ中の第1のプロトコルは、優先のプロトコルであり、第2のプロトコルは、第2最優先のプロトコルであるなど）。

【0315】

BufferLength：使用可能なバッファの長さを与える。

【0316】

Buffer：このトランザクションの直列配列が記憶されるバッファにポインタを与える。

【0317】

・リターン値：

【0318】

【表31】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

【0319】

PushTransaction

(IN HANDLE TransactionHandle、
 IN ULONG NumberOfProtocols、
 IN PCRM_PROTOCOL_ID ProtocolArray、
 IN ULONG DestinationInfoLength、
 IN PVOID DestinationInfo、
 IN ULONG ResponseBufferLength、
 OUT PVOID ResponseBuffer、
 OUT PULONG ResponseBufferUsed Optional、
 OUT PULONG PushCookie Optional)。

【0320】

・このルーチンは、このトランザクションが、プッシュスタイルの伝搬を使用して宛先

10

20

30

40

50

マシンに伝搬されることを要求する。プロトコルは、`ProtocolArray`中にリストされる順序で、成功するまで使用されることになる。この宛先マシンへ伝搬する際にどのプロトコルでもうまくいかない場合には、このルーチンは`failure`を戻すことになる。

【0321】

・引数：

`TransactionHandle`：このリモートマシンに伝搬すべきこのトランザクションオブジェクトにポインタを与える。

【0322】

`DestinationInfoLength`：使用可能な`DestinationInfoLength`の長さを与える。 10

【0323】

`DestinationInfo`：この宛先についての「エンドポイント」情報が記憶されるバッファに対してポインタを与える。これは、この宛先マシン上の`GetProtocolAddressInformation`に対する呼出から受信される出力とすることができる。

【0324】

`ResponseBufferLength`：使用可能な`ResponseBuffer`の長さを与える。

【0325】

`ResponseBuffer`：このトランザクションの直列配列が記憶されるバッファにポインタを与える。 20

【0326】

`PushCookie`：このプッシュ要求を表すクッキーが記憶されることになるバッファにポインタを与える。

【0327】

・リターン値：

【0328】

【表32】

STATUS_SUCCESS

30

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

【0329】

`GetPushTransactionBuffer`

(IN HANDLE TransactionHandle、
IN ULONG PushCookie、
IN ULONG ResponseBufferLength、
OUT PVOID ResponseBuffer、
OUT PULONG ResponseBufferUsed Optional)。

40

【0330】

・`PushTransaction`に対する初期呼出しが、`STATUS_BUFFER_TOO_SMALL`リターンコードを受信した場合に、この呼出しを使用して`PushTransaction`に対する呼出しの出力を検索する。この場合には、呼出し側は、`GetPushTransactionBuffer`をコールし、十分なバッファサイズで渡すことになる。

50

【0331】

・引数：

`TransactionHandle`：この新しいトランザクションを表すハンドルが記憶されるロケーションにポインタを与える。

【0332】

`BufferLength`：使用可能なバッファの長さを与える。

【0333】

`Buffer`：このトランザクションの直列配列が記憶されるバッファにポインタを与える。

【0334】

・リターン値：

【0335】

【表33】

`STATUS_SUCCESS`

`STATUS_ACCESS_DENIED`

`STATUS_INVALID_HANDLE`

`STATUS_INSUFFICIENT_RESOURCES`

10

20

【0336】

`PropagationComplete`

(IN HANDLE `EnlistmentHandle`、
IN ULONG `RequestCookie`、
IN ULONG `BufferLength`、
IN PVOID `Buffer`)。

【0337】

・トランザクションの伝搬を正常に完了した後に、このルーチンは、CRMによって呼出される。

30

【0338】

・引数：

`TransactionHandle`：この新しいトランザクションを表すハンドルを記憶すべきロケーションにポインタを与える。

【0339】

`RequestCookie`：どの要求が完了しているかを示すために、この元のPROPAGATE通知の引数中で受信されたこの`RequestCookie`を与える。

【0340】

`BufferLength`：使用可能なこのバッファの長さを与える。

【0341】

`Buffer`：このトランザクションの直列配列が記憶されるバッファにポインタを与える。

40

【0342】

・リターン値：

【0343】

【表 3 4】

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES

【0 3 4 4】

10

PropagationFailed

(IN HANDLE ResourceManagerHandle、
 IN ULONG RequestCookie、
 IN STATUS PropStatus)。

【0 3 4 5】

・CRMは、このルーチンを使用して要求されたときにこのトランザクションを伝搬することに失敗していることを示す。

【0 3 4 6】

・引数：

TransactionHandle：この新しいトランザクションを表すハンドルを記憶すべきロケーションにポインタを与える。 20

【0 3 4 7】

BufferLength：使用可能なこのバッファの長さを与える。

【0 3 4 8】

Buffer：このトランザクションの直列配列が記憶されるバッファにポインタを与える。

【0 3 4 9】

・リターン値：

【0 3 5 0】

【表 3 5】

30

STATUS_SUCCESS

STATUS_ACCESS_DENIED

STATUS_INVALID_HANDLE

STATUS_INSUFFICIENT_RESOURCES.

【0 3 5 1】

40

実施例のコンピューティング環境

図 9 は、一般的なコンピュータ環境 900 を示しており、このコンピュータ環境を使用して本明細書中で説明している技術を実装することができる。このコンピュータ環境 900 は、コンピューティング環境の一実施例にすぎず、コンピュータおよびネットワークアーキテクチャの利用範囲あるいは機能について、どのような限定も示唆するものではない。また、このコンピュータ環境 900 は、この実施例のコンピュータ環境 900 において示されたコンポーネントのうちのどの 1 つまたは組合せに関してもどのような依存性または要件を有するものとも解釈すべきではない。

【0 3 5 2】

コンピュータ環境 900 は、コンピュータ 902 の形態の汎用コンピューティングデバイスを含んでいる。コンピュータ 902 のコンポーネントは、それだけには限定されない 50

が、1つまたは複数のプロセッサまたは処理装置904と、システムメモリ906と、プロセッサ904を含めて様々なシステムコンポーネントをシステムメモリ906に結合するシステムバス908とを含むことが可能である。

【0353】

システムバス908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ックスポート (accelerated graphics port)、および様々なバスアーキテクチャのうちのどれかを使用したプロセッサバスまたはローカルバスを含めて、いくつかのタイプのバス構造のうちの1つまたは複数のいずれかを表す。実施例として、このようなアーキテクチャは、ISAバス、MCAバス、EISA (Enhanced ISA拡張ISA) バス、VESA (Video Electronics Standard Associationビデオエレクトロニクス規格協会) ローカルバス、メザンバスとしても知られているPCIバス、PCIエクスプレス (Express) バス、USB、SD (Secure Digitalセキュアデジタル) バス、またはIEEE1394バス、すなわちファイヤワイヤ (FireWire) を含んでいる。

10

【0354】

コンピュータ902は、様々なコンピュータ読取り可能媒体を含むこともできる。このような媒体は、コンピュータ902によってアクセス可能な任意の入手可能な媒体とすることが可能であり、揮発性媒体も不揮発性媒体も、着脱可能媒体も着脱不能媒体も含んでいる。

20

【0355】

システムメモリ906は、RAM (random access memoryランダムアクセスメモリ) 910などの揮発性メモリ、および／またはROM (read only memory読取り専用メモリ) 912やフラッシュRAM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媒体を含んでいる。起動中などにコンピュータ902内の要素間で情報を転送する助けをする基本ルーチンを含むBIOS (Basic input/output system基本入出力システム) 914は、ROM912またはフラッシュRAMに記憶される。RAM910は、一般に処理装置904によって直ちにアクセス可能な、かつ／または現在操作されているデータおよび／またはプログラムモジュールを含んでいる。

30

【0356】

コンピュータ902は、他の着脱可能／着脱不能の揮発性／不揮発性のコンピュータストレージ媒体を含むこともできる。実施例として、図9は、着脱不能の不揮発性の磁気媒体 (図示せず) から読み取りそれ書き込むためのハードディスクドライブ916、着脱可能な不揮発性の磁気ディスク920 (例えば、「フロッピー (登録商標) ディスク」) から読み取りそれ書き込むための磁気ディスクドライブ918、およびCD-ROM、DVD-ROM、他の光媒体など着脱可能な不揮発性の光ディスク924から読み取りまたはそれ書き込みあるいはその両方を行うための光ディスクドライブ922を示している。ハードディスクドライブ916、磁気ディスクドライブ918、および光ディスクドライブ922は、それぞれ1つまたは複数のデータ媒体インターフェース925によってシステムバス908に接続されている。代わりに、ハードディスクドライブ916、磁気ディスクドライブ918、および光ディスクドライブ922は、1つまたは複数のインターフェース (図示せず) によってシステムバス908に接続することもできる。

40

【0357】

これらのディスクドライブおよびこれらの関連するコンピュータ読取り可能媒体は、コンピュータ902用のコンピュータ読取り可能命令、データ構造、プログラムモジュール、および他のデータの揮発性ストレージを提供する。この実施例では、ハードディスク916、着脱可能磁気ディスク920、および着脱可能光ディスク924が示されているが、磁気カセットまたは他の磁気ストレージデバイス、フラッシュメモリカード、CD-ROM、DVD (digital versatile diskデジタル多用途ディス

50

ク) または他の光ストレージ、ランダムアクセスメモリ (RAM)、読取り専用メモリ (ROM)、EEPROM (electrically erasable programmable read-only memory 電氣的消去可能プログラム可能読取り専用メモリ) など、コンピュータによってアクセス可能なデータを記憶することができる他のタイプのコンピュータ読取り可能媒体を利用してこの実施例のコンピューティングシステムおよびコンピューティング環境を実装することもできることが理解されよう。

【0358】

実施例としてオペレーティングシステム926、1つまたは複数のアプリケーションプログラム928、他のプログラムモジュール930、およびプログラムデータ932を含めて、任意数のプログラムモジュールをハードディスク916、磁気ディスク920、光ディスク924、ROM912、および/またはRAM910に記憶することができる。このようなオペレーティングシステム926、1つまたは複数のアプリケーションプログラム928、他のプログラムモジュール930、およびプログラムデータ932 (またはそれらの何らかの組合せ) のそれぞれは、前述の実施形態の例に従ってトランザクションを実行して、分散ファイルシステムをサポートする存在するコンポーネントのすべてまたは一部分を実装することができる。

【0359】

ユーザは、キーボード934やポインティングデバイス936 (例えば、「マウス」) などの入力デバイスを介してコマンドおよび情報をコンピュータ902に入力することができる。他の入力デバイス938 (詳細には示さず) は、マイクロフォン、ジョイスティック、ゲームパッド、サテライトディッシュ、シリアルポート、および/またはスキャナなどを含むことができる。これらおよび他の入力デバイスは、システムバス908に結合された入出力インターフェース940を介して処理装置904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) など他のインター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。

【0360】

モニター942または他のタイプのディスプレイデバイスも、ビデオアダプタ944などのインターフェースを介してシステムバス908に接続することができる。モニター942に加えて、他の出力周辺デバイスには、I/Oインターフェース940を介してコンピュータ902に接続することができるスピーカ (図示せず) やプリンタ946などのコンポーネントも含めることが可能である。

【0361】

コンピュータ902は、リモートコンピューティングデバイス948など1台または複数台のリモートコンピュータに対する論理接続を使用してネットワーク環境で動作することが可能である。実施例として、リモートコンピューティングデバイス948は、PC、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、ピアデバイス、または他の共通ネットワークノードなどとするのが可能である。リモートコンピューティングデバイス948は、コンピュータ902に対して本明細書中で説明しているエレメントおよび特徴の多くまたはすべ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。代わりに、コンピュータ902は、非ネットワーク環境でも同様に動作することができる。

【0362】

コンピュータ902とリモートコンピュータ948の間の論理接続は、ローカルエリアネットワーク (LAN) 950および一般的なワイドエリアネットワーク (WAN) 952として示されている。このようなネットワーキング環境はオフィス、企業規模の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において一般的なものである。

【0363】

LANネットワーキング環境中で実装されるとき、コンピュータ902は、ネットワークインターフェースまたはネットワークアダプタ954を介してローカルネットワーク9

10

20

30

40

50

50に接続される。WANネットワーキング環境中で実装されるときには、コンピュータ902は、一般にワイドネットワーク952上で通信を確立するためのモデム956または他の手段を含んでいる。コンピュータ902に内蔵または外付けにすることができるモデム956は、I/Oインターフェース940または他の適切なメカニズムを介してシステムバス908に接続することができる。ここに示すネットワーク接続は実施例であり、コンピュータ902と948との間で少なくとも1つの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。

【0364】

コンピューティング環境900と共に示すネットワーク環境などのネットワーク環境において、コンピュータ902に関連して示すプログラムモジュールまたはその一部分は、リモートメモリストレージデバイスに記憶することができる。実施例として、リモートアプリケーションプログラム958は、リモートコンピュータ948のメモリデバイス上に存在する。例示のために、アプリケーションまたはプログラム、およびこのオペレーティングシステムなど他の実行可能なプログラムコンポーネントは、本明細書中では個別のブロックとして示されているが、かかるプログラムおよびコンポーネントは、コンピューティングデバイス902の異なるストレージコンポーネント中にいろいろな時刻に存在し、このコンピュータの少なくとも1つのデータプロセッサによって実行されることが理解されよう。

10

【0365】

様々なモジュールおよび技法は、1台または複数台のコンピュータまたは他のデバイス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状況において本明細書中において説明することができる。一般に、プログラムモジュールは、特定のタスクを実施し、または特定の抽象データ型を実装するルーチン、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んでいる。これらのプログラムモジュールなどは、仮想マシンや他のジャストインタイムコンパイル実行環境などの中で、ネイティブコード(native code)として実行し、またはダウンロードし実行することができる。一般に、プログラムモジュールの機能は、様々な実施形態において必要に応じて組み合わせ、または分散させることができる。

20

【0366】

これらのモジュールおよび技法の実装は、コンピュータ読取り可能媒体の何らかの形態上に記憶し、またはその形態に渡って伝送することができる。コンピュータ読取り可能媒体は、コンピュータによりアクセスすることが可能な任意の使用可能な媒体とすることが可能である。実施例として、限定するものではないが、コンピュータ読取り可能媒体は、「コンピュータストレージ媒体」と「通信媒体」を含むことができる。

30

【0367】

「コンピュータストレージ媒体」は、コンピュータ読取り可能命令、データ構造、プログラムモジュール、または他のデータなどの情報の記憶のための任意の方法または技術により実装される揮発性および不揮発性の、着脱可能および着脱不能の媒体を含んでいる。コンピュータストレージ媒体は、それだけには限定されないが、RAM、ROM、EEPROM、フラッシュメモリもしくは他のメモリ技術、CD-ROM、デジタル多用途ディスク(DVD)もしくは他の光ストレージ、磁気カセット、磁気テープ、磁気ディスクストレージ、もしくは他の磁気ストレージデバイス、または所望の情報を記憶するために使用することができコンピュータによりアクセスすることが可能な他の任意の媒体を、含んでいる。

40

【0368】

「通信媒体」は一般に、搬送波や他の搬送メカニズムなどの被変調データ信号の形で、コンピュータ読取り可能命令、データ構造、プログラムモジュール、または他のデータを実施する。通信媒体は、任意の情報配信媒体も含んでいる。用語「被変調データ信号」は、信号中の情報を符号化するようにしてその信号中の1つまたは複数のその特性が設定あるいは変更された信号を意味する。非限定的な実施例にすぎないが、通信媒体には、有線

50

ネットワークや直接配線接続などの有線媒体、並びに、音響、RF、赤外線、他の無線媒体などの無線媒体が含まれる。以上のうちの任意の組合せもまた、コンピュータ読取り可能媒体の範囲内に含まれる。

【0369】

本明細書全体を通して、「一実施形態 (one embodiment)」、「一実施形態 (an embodiment)」または「実施形態の一例 (an example embodiment)」と言及しているが、これらは、説明している特定の機能、構成、または特徴が本発明の少なくとも一実施形態中に含まれることを意味する。したがって、このようなフレーズの使用によって、ただ1つの実施形態を超えたものについて言及することができる。さらに、この説明される機能、構成、または特徴は、1つまたは複数の実施形態において適切な任意の方法で組み合わせることができる。 10

【0370】

しかし、本発明は、1つまたは複数の特定の細部によらなくても、あるいは、他の方法、リソース、材料などを用いても実行することができることは当業者に理解されるだろう。他の事例においては、本発明の態様を単にあいまいにしないように、よく知られている構成、リソース、またはオペレーションを詳細に示さず、または説明していない。

【0371】

本発明の実施形態の例および用途について例示し説明してきたが、本発明は上述の正確な構成およびリソースに限定されるものでないことを理解されたい。本明細書中に開示している本発明の方法およびシステムの構成、オペレーション、および細部において、請求している本発明の範囲を逸脱することなく、当業者にとって明らかな様々な修正、変更、および変形を行うことができる。 20

【図面の簡単な説明】

【0372】

【図1】 トランザクション処理リモートファイルオペレーションを使用することができるシステムの一実施例を示す図である。

【図2】 図1のシステムのクライアントおよびサーバのコンポーネントの一実施例を示す図である。

【図3】 図2のクライアントとサーバの間のトランザクション処理リモートファイルオペレーションについての実施例の処理フローを示す図である。 30

【図3A】 図2のクライアントとサーバの間のトランザクション処理リモートファイルオペレーションについての実施例の処理フローを示す図である。

【図4】 図2のクライアントおよびサーバによる複数のリモートファイルアクセスの一実施例を示す図である。

【図5】 ネットワーク上のトランザクションの2相コミットを実施するための一実施例の処理フローを示す図である。

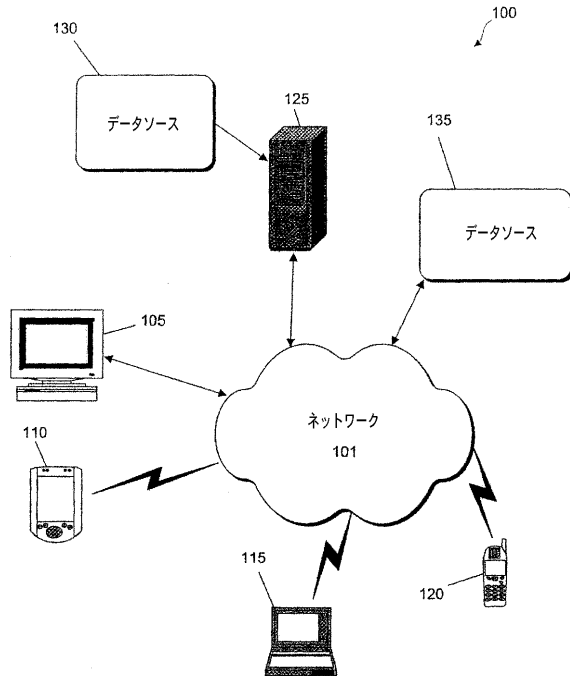
【図6】 トランザクション管理を実装するためのコンポーネントの一実施例を示す図である。

【図7】 カーネルレベルトランザクションについての一実施例処理フローを示す図である。 40

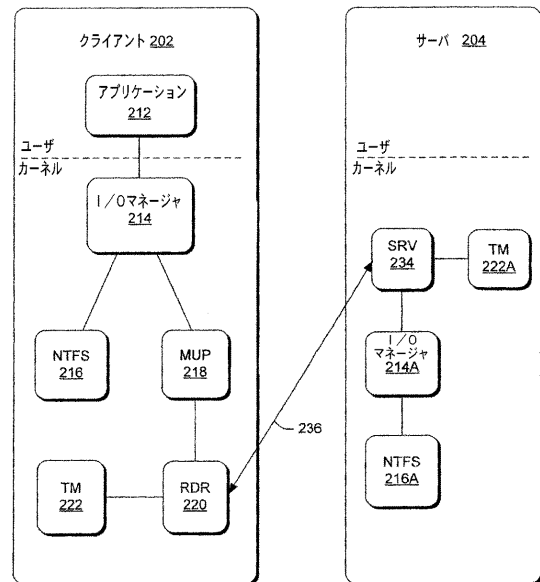
【図8】 セキュリティフィーチャの一実施例を示す図である。

【図9】 様々な実施形態による、本明細書中で説明される技法を実装するために使用することができる一般的なコンピュータ環境を示す図である

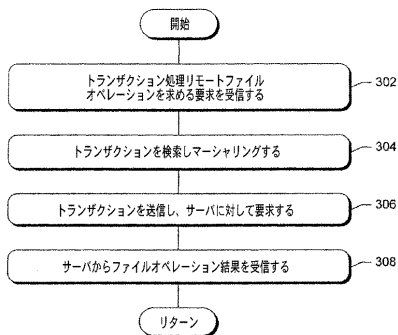
【図 1】



【図 2】



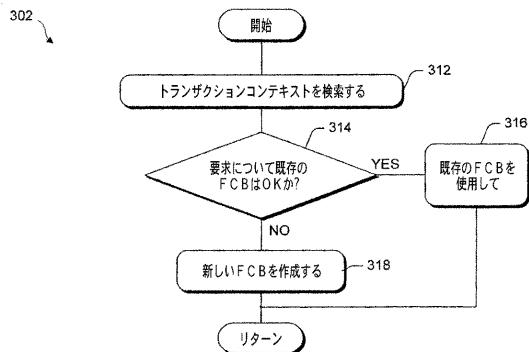
【図 3】



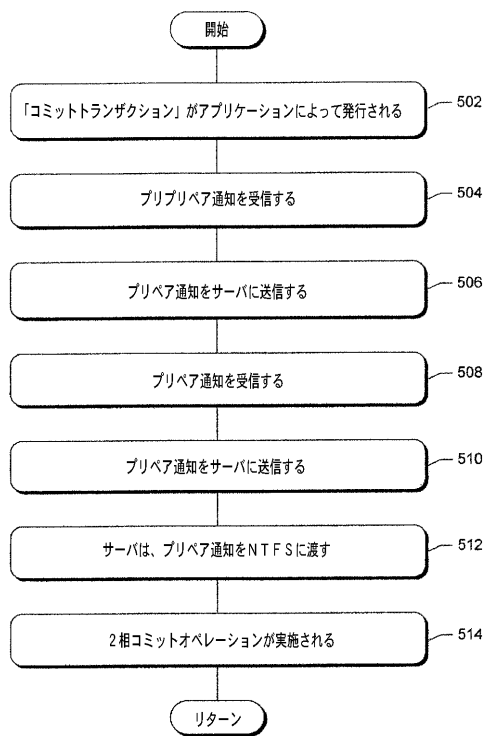
【図 4】

	オペレーション	戻される ハンドル	トランザクション	見られる バージョン
401	読取りのため オープンする	H1	T1	A
402	読取りのため オープンする	H2	T2	A
403	書き込みのため オープンする	H3	T2	B
404	読取りのため オープンする	H4	T2	B
405	読取りのため オープンする	H5	T3	A
406	T2を コミットする		T2	
407	読取りのため オープンする	H6	T1	C
408	読取りのため オープンする	H7	T3	C

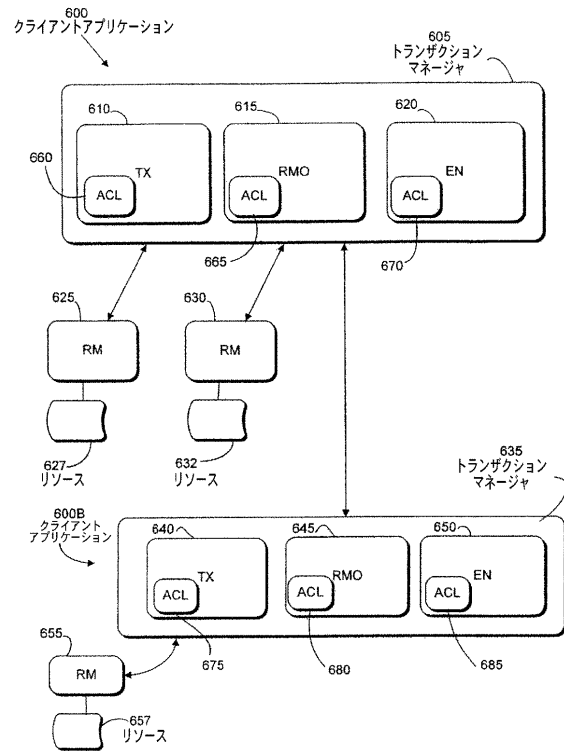
【図 3 A】



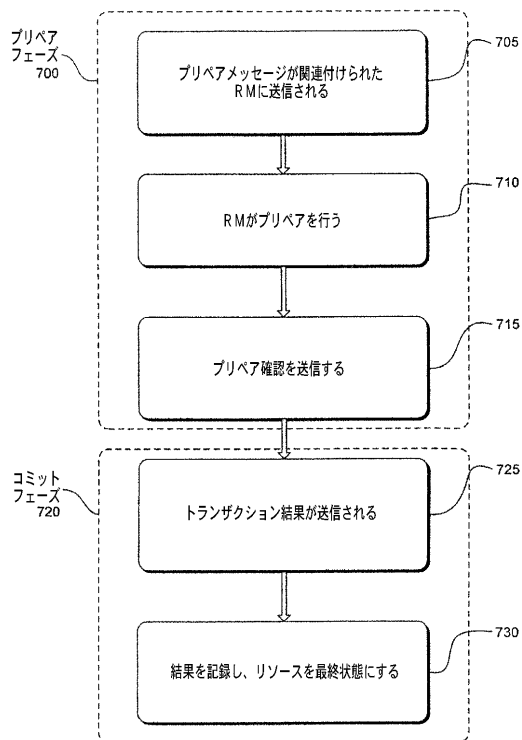
【図 5】



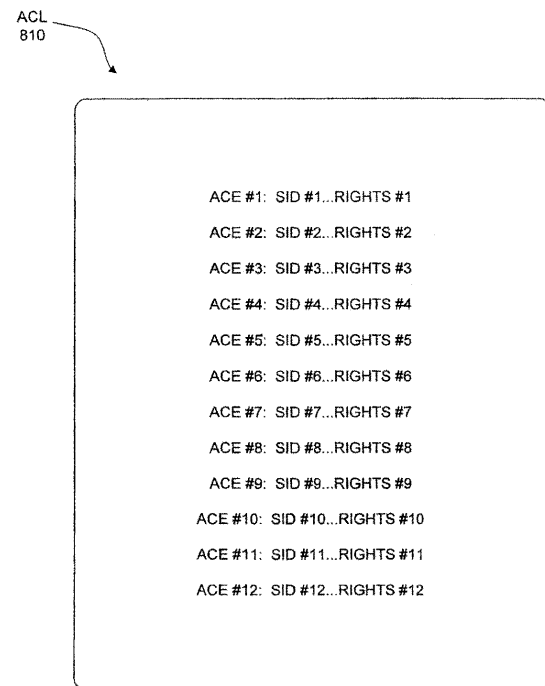
【図 6】



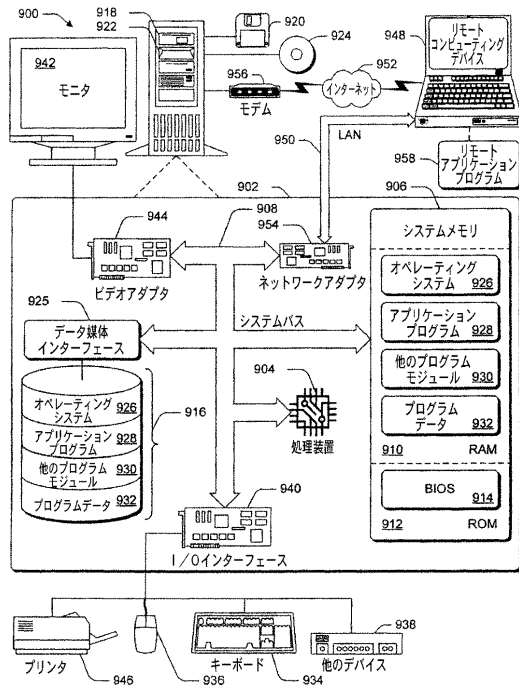
【図 7】



【図 8】



【図 9】



【国際調査報告】

INTERNATIONAL SEARCH REPORT		International application No. PCT/US04/23642
A. CLASSIFICATION OF SUBJECT MATTER IPC(7) : G06F 15/00, 17/00, 17/30 US CL : 707/1, 205; 712/30; 715/539 According to International Patent Classification (IPC) or to both national classification and IPC		
B. FIELDS SEARCHED		
Minimum documentation searched (classification system followed by classification symbols) U.S. : 707/1, 205; 712/30; 715/539		
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched NONE		
Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) West US, JPO, Derwent, IBM.....Search		
C. DOCUMENTS CONSIDERED TO BE RELEVANT		
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6,314,437 A (Starek et al) 6 November 2001, col. 2-6, lines 1-30	1-58
A	US 20020156713 A1 (Gaini) 24 October 2002, Paragraph [0040]-[0047]	1-58
☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.		
* Special categories of cited documents: "A" document defining the general state of the art which is not considered to be of particular relevance "B" earlier application or patent published on or after the international filing date "L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) "O" document referring to an oral disclosure, use, exhibition or other means "P" document published prior to the international filing date but later than the priority date claimed "T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention "X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone "Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art "&" document member of the same patent family		
Date of the actual completion of the international search 17 October 2005 (17.10.2005)		Date of mailing of the international search report 16 NOV 2005
Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450 Alexandria, Virginia 22313-1450 Facsimile No. (571) 273-8300		Authorized officer Safet Metjahic Telephone No. (571) 272-2100

フロントページの続き

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

(72)発明者 シシール パディカ

アメリカ合衆国 98052 ワシントン州 レッドモンド ワン マイクロソフト ウェイ
マイクロソフト コーポレーション内

(72)発明者 バラン セシュ ラマン

アメリカ合衆国 98052 ワシントン州 レッドモンド ワン マイクロソフト ウェイ
マイクロソフト コーポレーション内

(72)発明者 スレンドラ ヴェルマ

アメリカ合衆国 98052 ワシントン州 レッドモンド ワン マイクロソフト ウェイ
マイクロソフト コーポレーション内

(72)発明者 ジョン カーギル

アメリカ合衆国 98052 ワシントン州 レッドモンド ワン マイクロソフト ウェイ
マイクロソフト コーポレーション内

(72)発明者 ヤコブ ラクチュール

アメリカ合衆国 98052 ワシントン州 レッドモンド ワン マイクロソフト ウェイ
マイクロソフト コーポレーション内

Fターム(参考) 5B082 GB02 HA05