(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5501377号
(P5501377)
(45)発行日　平成26年5月21日(2014.5.21)　(24)登録日　平成26年3月20日(2014.3.20)

(51)Int.Cl.　F I
G06F 9/52 (2006.01)　G06F 9/46 472Z

請求項の数 10 (全 13 頁)

(21)出願番号	特願2011-546244 (P2011-546244)
(86)(22)出願日	平成21年12月4日 (2009.12.4)
(65)公表番号	特表2012-515393 (P2012-515393A)
(43)公表日	平成24年7月5日 (2012.7.5)
(86)国際出願番号	PCT/US2009/066889
(87)国際公開番号	WO2010/082983
(87)国際公開日	平成22年7月22日 (2010.7.22)
審査請求日	平成24年11月8日 (2012.11.8)
(31)優先権主張番号	12/353,905
(32)優先日	平成21年1月14日 (2009.1.14)
(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　100140109
弁理士　小野　新次郎
(74)代理人　100075270
弁理士　小林　泰
(74)代理人　100101373
弁理士　竹内　茂雄
(74)代理人　100118902
弁理士　山本　修
(74)代理人　100153028
弁理士　上田　忠

最終頁に続く

(54)【発明の名称】トランザクションメモリにおけるトランザクション処理

(57)【特許請求の範囲】
【請求項１】
共有メモリへアクセスできるアトミックトランザクションを制御する方法であって、
少なくとも１つのメモリ以外のリソースマネージャを参加させることと、
トランザクションメモリを管理するように構成された、メモリのリソースマネージャに参加させることと、
前記メモリ以外のリソースマネージャおよびメモリのリソースマネージャが前記アトミックトランザクションをコミットするために投票するかどうかを判断するように構成された、コミットプロトコルを呼び出すことと、
前記メモリ以外のリソースマネージャおよびメモリのリソースマネージャが前記アトミックトランザクションをコミットするために投票する場合、前記アトミックトランザクションをコミットすることと、
前記メモリのリソースマネージャが前記アトミックトランザクションをコミットするために投票しない場合、前記アトミックトランザクションを再実行することと、
前記メモリ以外のリソースマネージャの少なくとも１つが前記アトミックトランザクションをコミットするために投票せず、かつ前記メモリのリソースマネージャが前記アトミックトランザクションをコミットするために投票する場合、前記アトミックトランザクションを失敗（アボート）とすることであって、前記メモリのリソースマネージャは、前記アトミックトランザクション中にメモリ衝突が生じない場合には、コミットするために投票することと

を備えることを特徴とする方法。
【請求項２】
前記共有メモリは、前記メモリのリソースマネージャを参加させる前のオリジナル状態を含み、前記アトミックトランザクションを再実行することは、前記共有メモリを前記オリジナル状態へロールバックするこ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３】
前記アトミックトランザクションを再実行することは、前記アトミックトランザクションにおけるすべてのリソースマネージャがコミットするために投票するか、または前記アトミックトランザクションが失敗とされるまで、前記アトミックトランザクションを繰り返し再実行するこ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４】
前記アトミックトランザクションを再実行することは、前記メモリ以外のリソースマネージャが前記アトミックトランザクションをコミットするために投票し、且つ、少なくとも１つのメモリリのソースマネージャがアボートするために投票するときに生じることを特徴とする請求項１に記載の方法。
【請求項５】
前記コミットプロトコルは、準備のフェーズおよびコミットのフェーズを含むことを特徴とする請求項１に記載の方法。
【請求項６】
前記アトミックトランザクションは、入れ子アトミックトランザクションを含むことを特徴とする請求項１に記載の方法。
【請求項７】
複数のスレッドを含む並列プログラムのための処理動作を含むコンピュータシステムを制御するためにコンピュータ実行可能命令コンポーネントを記憶したコンピュータストレージ媒体であって、少なくとも１つのスレッドは、メモリ動作を有するアトミックトランザクションを含み、前記コンピュータ実行可能命令コンポーネントは、
前記アトミックトランザクションに参加するように構成された、少なくとも１つのメモリ以外のリソースマネージャと、
前記アトミックトランザクションに参加し、トランザクションメモリを管理するように構成された、メモリのリソースマネージャと、
前記アトミックトランザクションに参加した前記リソースマネージャの全てに結合されるトランザクションマネージャであって、前記トランザクションマネージャは、前記アトミックトランザクションに参加した全てのリソースマネージャから、前記アトミックトランザクションをコミットするかどうかについて、投票を受信するように構成され、前記アトミックトランザクションは、前記アトミックトランザクションに参加した全てのリソースマネージャからのコミット投票にコミットし、前記メモリのリソースマネージャは、コミットするかどうかについて投票する前にメモリ衝突が存在するか判断する、トランザクションマネージャと、
を備えることを特徴とするコンピュータストレージ媒体。
【請求項８】
前記トランザクションマネージャは、準備の指令およびコミットの指令を発するように構成されたことを特徴とする請求項７に記載のコンピュータストレージ媒体。
【請求項９】
前記アトミックトランザクションに参加した前記リソースマネージャは、前記準備の指令に応じて前記アトミックトランザクションをコミットするかどうかについて投票するように構成されたことを特徴とする請求項８に記載のコンピュータストレージ媒体。
【請求項１０】
コンピュータに請求項１～６のいずれか１項に記載の方法を実行させるためのプログラム。
【発明の詳細な説明】

【技術分野】
【０００１】
本発明は、トランザクションメモリにおけるトランザクション処理に関する。
【背景技術】
【０００２】
共有メモリ・マルチプロセッサ・システムのための並列プログラミングは、マルチスレッドの性能を含み、同じデータへアクセスすることができる。マルチスレッドは、プロセッサ間で共有されるメモリに接続されている、マルチプロセッサ、マルチプロセッサコア、または、他の並列処理のクラスで実行する。共有メモリのモデルは、最も一般的に展開されているマルチスレッド伝達の方法である。それによって、マルチスレッドのプログラムは、順次プログラムとほぼ同じような方法で作成されることが可能である。共有メモリのモデルを実行するために、並列プログラミングは、競合およびこれと同等のものなどの望ましくない状況を引き起こしうる、共有データの並列アクセスおよび並列使用を避けるように注意する。
【０００３】
ロックは、共有データへの並列アクセスの問題を避けるための一般的な解決法である。ロックは、あるスレッドによってアクセスされた変数へ他のスレッドもアクセスを試みることができるが、その変数は同時には１つのスレッドによってのみ使用されうる、という前提に主として拠っている。ロックによって、１つのスレッドは、ロックが解除されるまで、変数の制御を行うこと、および、他のスレッドが変数を変更できないようにすることが可能である。ロックに基づいたプロトコルは、よく知られているが、しばしば使用が難しいと見なされる。粗い粒度でロックを使用することによって、比較的大きなデータを保護するが、通常は、粗い粒度でのロックの使用は効率的に処理をしない。スレッドは、それらが干渉しないときでさえ、互いを阻止し、このようなロックは、争いのもととなる。あるいはまた、より細かい粒度でロックを使用することによって、スケーラビリティの問題点を軽減することができるが、そのロックは、他の問題をもたらす。その理由は、正確さを保証し、およびデットロックを避けるロックの慣例的なやり方は、複雑でエラーが発生しやすくなるからである。
【０００４】
別の解決法は、トランザクションメモリを使用するアプリケーションを実装することである。トランザクションメモリ・システムは、スレッドの実行によって、スレッドのメモリアクセスを管理して、２または３以上のスレッドが衝突した形で同じメモリ位置へアクセスすることを試みる場合、スレッドの効果をロールバックまたは元に戻すようにする。トランザクションメモリ・システムは、ハードウエアおよび／またはソフトウエアのコンポーネントを使用して実装されうる。ソフトウエアによるトランザクションメモリ・システムによって、ソフトウエアのランタイムライブラリおよび／またはランタイム実行環境に意味をもたせ、および／またはコンパイラを使用する意味を持たせることができる。トランザクションメモリは、ある１つのスレッドによって読み出された変数は、他のスレッドによって修正されそうになく、ゆえに、その変数は、プログラムのスケーラビリティへの厳しい悪影響なく、共有できるという前提に基いて、共有メモリへのアクセスを制御するための、コンパイラレベルの並列の制御メカニズムとしてしばしば実装される。しかしながら、トランザクションメモリ・システムにおいてメモリアクセスを追跡することは、多分、プログラムの実行へオーバーヘッドを加えることになりうる。
【０００５】
粗いロックに基づいたプロトコルでのトランザクションメモリの１つの利点は、並列性が増加することである。トランザクションメモリにおいて、スレッドはデータへのアクセスを待つ必要がない。異なるスレッドは、安全かつ同時に、通常は同じロックのもとで保護されるデータ構造の別々の部分を修正することができる。失敗したトランザクションの再試行のオーバーヘッドにもかかわらず、多くの現実的な並列プログラムにおいて、競合がめったに起きないので、いくつかの個数のプロセッサまたはプロセッサコアから開始す

る、粗い粒度のロックに基づいたプロトコルを用いることによって性能が向上する。

【０００６】

トランザクションメモリの保証、および大規模な研究の主題であるにもかかわらず、広く使用され受け入れられるためには障害がある。例えば、プログラマーは、トランザクションメモリをよく知らず、実際的なユーザフレンドリーな実施が欠如しているので、トランザクションメモリを使用するのをためらう。

【発明の概要】

【０００７】

この概要は、簡単な形式にて、詳細な説明においてさらに後述される概念の抜粋を紹介するために提供される。この概要は、特許請求の範囲の主題の主要な特徴または本質的な特徴を特定することを意図しないし、特許請求の範囲の主題の範囲を限定するために使用されることも意図しない。

【０００８】

この開示は、コンパイラレベルでのトランザクションメモリの概念を、（データベーストランザクションなどの）上位レベルの従来のトランザクション処理へ統合することを指向している。コンパイラレベルでのアトミック（atomic）ブロックは、アトミックブロックトランザクションとして指定され、アトミック性および独立性の特徴を含むことができる。このアトミックブロックトランザクション内での動作は、利用可能なリソースマネージャのリポジトリからのリソースマネージャの参加（enlistment）を含む。そのリポジトリは、あらかじめプログラムされたメモリリソースマネージャを含んで、トランザクションメモリの管理をすることができる。従来のトランザクションにおいて見られるように、コミットプロトコルを用いて、動作が成功したかどうかを決定することができ、そして、その結果をトランザクションの外部に露出させることができる。しかしながら、従来のトランザクションと違って、このトランザクションは、動作のいくつかが有効でない場合に、必ずしも駄目になるわけではない。むしろ、メモリの衝突は、メモリを含むすべてのリソースマネージャのロールバック、およびアトミックブロックトランザクションの再実行を引き起こす。再実行は、メモリリソースマネージャの動作を含むすべての動作が成功裡に有効となるまで、必要な限り繰り返されることが可能である。

【図面の簡単な説明】

【０００９】

添付の図面は、実施形態のさらなる理解に供するために含まれ、この明細書に組み込まれ、および明細書の一部を構成する。これら図面は、実施形態を示し、および記述とともに実施形態の原理を説明するのに供される。他の実施形態、および実施形態の多くの意図した有利な点は、以下の詳細な説明を参照することでよりよく理解されるようになるにつれて、容易に認識されるであろう。これら図面の要素は、必ずしも互いに関連する縮尺比ではない。同様の参照番号は、対応する類似の部分を示す。

【００１０】

【図１】本開示の特徴を実装するコンピューティング装置の多数の可能な例のうちの１つを示すブロック図である。

【図２】図１のコンピューティングシステム例において実行されるトランザクションメモリ処理システムの一例を示すブロック図である。

【図３】図２のトランザクションメモリ処理システムにおいて使用されるプロセスの一例を示すフロー図である。

【図４】時系列に関して、図２のコミットプロトコルのプロセスの一例を示す概略図である。

【発明を実施するための形態】

【００１１】

以下の詳細な説明において、添付の図面を参照する。これら添付の図面は、詳細な説明をなし、これら図面において、例示によって、本発明を実施することのできる特定の実施形態を示す。本発明の範囲から逸脱することなく、他の実施形態を用いることができ、お

よび構造上または論理上の変更を行うことができることが理解される。それゆえに、以下の詳細な説明は、限定的な意味にとられず、および本発明の範囲は、添付の特許請求の範囲によって定められる。本明細書に記載された種々の例示的な実施形態の特徴は、特に断わり書きがなされていない限り、互いに組み合わせることができることが理解される。

【００１２】

図１は、オペレーティング環境として採用されうる例示的なコンピュータシステムを示し、コンピューティング装置１００のようなコンピューティング装置を含んでいる。基本的な構成において、コンピューティング装置１００は、典型的には、少なくとも２つの処理装置（すなわち、プロセッサ１０２）、およびメモリ１０４を有するプロセッサアーキテクチャを含む。コンピューティング装置の正確な構成および種類に応じて、メモリ１０４は、（ＲＡＭ（random access memory）のような）揮発性、（ＲＯＭ（read only memory）、フラッシュメモリのような）不揮発性、またはこれら２つの組み合わせとすることができる。この基本的な構成は、図１で線１０６によって示されている。コンピューティング装置は、１または２以上のいくつかの形態をとることができる。このような形態は、パーソナルコンピュータ、サーバ、ハンドヘルド装置、（ビデオゲームコンソールのような）消費者向け電子装置、または他のものを含む。

【００１３】

コンピューティング装置１００は、追加の特徴／機能を有することもできる。例えば、コンピューティング装置１００は、着脱可能なストレージ１０８、および着脱不可能なストレージ１１０のような、磁気または光学ディスク、または半導体メモリ、またはフラッシュストレージ装置を含むが、これらに限られない、追加のストレージ（着脱可能および／または着脱不可能）を含むこともできる。コンピュータストレージ媒体は、コンピュータが読み取り可能な命令、データ構造、プログラムモジュール、または他のデータなどの情報のストレージのためのいかなる適切な方法または技術において実行される、揮発性および不揮発性、着脱可能および着脱不可能な媒体を含む。メモリ１０４、着脱可能なストレージ１０８および着脱不可能なストレージ１１０は、すべて、コンピュータストレージ媒体の例である。コンピュータストレージ媒体は、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリまたは他のメモリ技術、ＣＤ－ＲＯＭ、ＤＶＤ（digital versatile disc）、または他の光学ストレージ、磁気カセット、磁気テープ、磁気ディスクストレージまたは他の磁気ストレージ装置、ＵＳＢ（universal serial bus）フラッシュドライブ、フラッシュメモリカード、または他のフラッシュストレージ装置、または、所望の情報を格納するために使用でき、およびコンピューティング装置１００によってアクセスできるいかなる他の媒体も含むが、これらに限定されない。このようなコンピュータストレージ媒体のいかなるものも、コンピューティング装置１００の一部となりうる。

【００１４】

コンピューティング装置１００は、コンピューティング装置１００が、他のコンピュータ／アプリケーション／ユーザ１１５と通信できるようにする、１または２以上の通信接続１１４を含む。コンピューティング装置１００は、キーボード、ポインティング装置（たとえばマウス）、ペン、音声入力装置、タッチ入力装置などの入力装置（複数可）１１２も含むことができる。コンピューティング装置１００は、ディスプレイ、スピーカー、プリンタなどの出力装置（複数可）１１１も含むことができる。

【００１５】

コンピューティングシステム１００は、システムプラットフォームを構成する、オペレーティングシステム・ソフトウエアプログラム、および１または２以上のソフトウエアアプリケーションを実行するように構成することができる。一つの例において、コンピューティングシステム１００は、ランタイム環境と呼ばれるソフトウエアコンポーネントを含む。ランタイム環境は、オペレーティングシステムの一部として含まれることが可能であり、または、ソフトウエアのダウンロードとして後に含まれることが可能である。ランタイム環境は、典型的に、共通プログラミングの問題に対するあらかじめコード化された解決法を含んで、ソフトウエア開発者が、ランタイム環境で実行するためのアプリケーショ

ンなどのソフトウエアプログラムを作成することを支援する。ランタイム環境は、典型的に、バーチャルマシンも含み、それによって、ソフトウエアアプリケーションをランタイム環境で実行できるようにして、プログラマーは、特定のプロセッサ１０２の性能を考慮する必要がないようにする。ランタイム環境の例は、多くのものがあるがその中で、ワシントン州レッドモンドのマイクロソフト社の.NetＣＬＲ（Common Language Routine）、およびC++ルーティンを含む。

【００１６】

図２は、トランザクションメモリ処理システム２００の例を示している。トランザクションメモリ処理システム２００は、ランタイム環境において呼び出され、アトミックブロックトランザクション２０２をサポートすることができる。アトミックブロックトランザクション２０２は、スレッドにおけるソフトウエアコードのアトミックブロックとすることができる。アトミックブロックトランザクション２０２は、トランザクション境界２０１、２０３を含む。トランザクション境界２０１、２０３は、アトミックブロックを示し、トランザクションメモリ処理システム２００に関与するコードを特定する。アトミックブロックトランザクション２０２は、少なくともトランザクションメモリの動作２０４を含むが、第２の、すなわちメモリ以外の動作２０６も含むように示されている。トランザクションメモリの動作２０４は、トランザクションメモリのメモリオペレーションを提供する。変数に影響を及ぼす動作、副次効果を及ぼす動作などの他の動作を、アトミックブロックトランザクションに含むことができる。システム２００は、トランザクションメモリ動作２０４に対応するメモリリソースマネージャ２０８のように、少なくとも１つのメモリ動作に対応する少なくとも１つのメモリリソースマネージャを含む。システム２００は、この例におけるメモリ以外の第２の動作２０６に対応するトランザクション２０２へ参加するメモリ以外のリソースマネージャ２１０のように、少なくとも１つのメモリ以外の動作に対応する少なくとも１つのメモリ以外のリソースマネージャを含むことができる。リソースマネージャ２０８、２１０は、トランザクション２０２に関与する動作に適したリソース２１２、２１４をそれぞれ管理する。リソースマネージャ２０８、２１０の動作は、トランザクションマネージャ２１６と調整される。トランザクションマネージャ２１６は、リソースマネージャ２０８、２１０と協働して、トランザクション２０２のアトミック性および独立性を確保する。トランザクションマネージャ２１６は、コミットプロトコル２１８を実装する。リソースマネージャ２０８、２１０も、コミットプロトコル２１８に関与する。別の例において、アトミックブロックトランザクションは、複数のリソースマネージャおよびリソースに対応する、３以上の動作を含むことができる。

【００１７】

トランザクション２０２は、単一のスレッドにより実行される互いに結合された動作のシーケンスである。スレッドは、共有メモリにおいて、他のプロセッサ上で同時に実行している他のスレッドを考慮し、または考慮しないで、データの修正を完了する。いずれにせよ、トランザクションの完了後、トランザクションメモリは、他のスレッドがアクセスしたデータへ同時に変更を行わなかったことを検証する。変更が有効化され、そしてその有効化が成功の場合、有効化された変更は、コミットオペレーションで不変とされる。有効化が失敗の場合、変更は取り消しまたは“ロールバック”され、およびトランザクション２０２は、有効化が成功するまで再実行される。

【００１８】

トランザクション２０２は、アトミック性および独立性の特徴を保有する。トランザクションは、アトミックであり、瞬間的に論理的に行われる。１つの動作が失敗すると、全部のトランザクションが失敗する。さらに、また、トランザクションは他のスレッドから独立しているので、変数は中間状態において他のスレッドにさらされることはない。いくつかの実施形態において、独立性は、トランザクション内で実行されているスレッドにのみ提供されるが、トランザクションの保護無しでデータへアクセスしているスレッドは、同時のトランザクションの中間状態に遭うかもしれない。ブロックの終わりに到達すると、トランザクションは、コミットされ、失敗（アボート）またはロールバックされ、およ

び再実行される。したがって、コミットまたは失敗するユニットは、全部のプロセスよりむしろトランザクション２０２である。状態は、中間状態をさらすことよりむしろ元の形態に戻る。この点において、トランザクション２０２は、典型的には、コンパイラレベルで示され、トランザクションメモリを含むことを除いて、データベース技術のトランザクション、すなわちデータベーストランザクションと似ている。しかしながら、典型的な従来のトランザクション（例えば、データベーストランザクション）とは違って、トランザクション２０２は成功するまで再実行することができる。

【００１９】

リソースマネージャは、従来のトランザクション処理に存在し、トランザクションに関与するリソースを管理する。しかし、従来のトランザクション処理は、今までのところ、自動的に独立させ、およびメモリへの同時アクセスを同期させるために使用されていなかった。従来のトランザクション処理は、従来のトランザクションへ接続されていない、ロック、またはトランザクションメモリのメカニズムを用いる。本例の特徴は、トランザクションメモリの管理が、従来のトランザクションシステムの一部であり、共有メモリへのアクセスを制御しない従来のトランザクションとは違って、トランザクションメモリを従来の失敗のアトミック性と組み合わせているということである。トランザクションメモリの提案は、メモリおよび他のリソースにわたって、一般的な失敗のアトミック性を提供しない。トランザクションメモリの提案におけるエラー処理の複雑性は、アプリケーションが、部分的な失敗またはメモリ衝突の組み合わせから記録するのをしばしば妨げる。開発者は、いまだに、彼らが処置のとり方を知っている限られたセットのエラーの場合のための解決法を手動で作ることによって、回復機能を提供している。結果として、開発者は、生産性および品質の結果に苦しむ。加えて、手動によるエラー処理の解決法を作ることは、開発者にとって些細ではない作業をもたらしている。しかしながら、本開示の例は、トランザクションメモリを従来のトランザクション処理へ組み込むことで、失敗のアトミック性を提供する。ゆえに、すべてのサポートされた動作が実行されるか、あるいは、何も実行されなかったように見える。ランタイム環境は、メモリのためのあらかじめプログラムされたリソースマネージャとして、トランザクションメモリのメカニズムを使用することができる。他の適切な動作のためのあらかじめプログラムされたリソースマネージャは、そのようなリソースが使用されるときにランタイム環境において呼び出されうるライブラリまたはリポジトリに含まれることが可能である。

【００２０】

適切なリソースマネージャが、リソースマネージャのあらかじめプログラムされたライブラリに存在しない多くの場合、プログラム開発者は、使用するリソースマネージャをプログラムに書き込むか、または、それをライブラリへ加えることができる。これは、従来のトランザクション処理で使用される標準的な方法においてなされることが可能であり、これは、トランザクション処理システム２００無しに、手動でエラー処理の解決法を作るよりも信頼性が高い。

【００２１】

さらに、従来のトランザクションにおいて使用されるリソースマネージャは、アトミック性（atomicity）、一貫性（consistency）、独立性（isolation）、および永続性（durability）（ＡＣＩＤ）の特徴を保有する。一方、トランザクションメモリにおけるトランザクション処理は、一貫性および永続性の特徴を必要としない。結果として、リソースマネージャ２０８、２１０の例は、永続性リソースマネージャよりむしろ揮発性リソースマネージャとして実装することができる。揮発性リソースマネージャは、それらの状態を揮発性メモリに格納し、トランザクション状態の回復をサポートしない。したがって、揮発性リソースマネージャの開発は、永続性リソースマネージャの開発よりも著しく容易である。トランザクション処理システム２００の場合、揮発性リソースマネージャは、永続性リソースマネージャより少ないシステムリソースを使用する。複数の揮発性リソースマネージャと共によく動作するトランザクションマネージャ２１６の例は、入手可能な軽量トランザクションマネージャである。このトランザクションマネージャは、より永続的な

トランザクションマネージャから生じるオーバーヘッドを著しく減らすことができる。他の例は、永続性トランザクションマネージャまたはリソースマネージャを含むことができる。

【００２２】

リソースマネージャ２０８、２１０は、トランザクション２０２に自動的に参加し、トランザクションの結果にしたがって、それらの状態になされる変更へコミットまたはロールバックする。ランタイム環境は、トランザクションへの参加、およびトランザクションリソース２１２、２１４に関するトランザクションの管理を自動化することができる。トランザクション２０２におけるリソース２１２、２１４の参加にあたり、リソースは、トランザクションマネージャへ、動作２０４、２０６がリソースに反するトランザクションの働きを実行したいことを伝える。次に、動作２０４、２０６は、リソース２１２、２１４に反する働きを実行する。機能的なエラーまたは衝突が起きない場合、トランザクションマネージャ２１６は、コミットプロトコル２１８を適用して、リソースマネージャ２０８、２１０を通じてリソース２１２、２１４に対して、その状態にされた変更をコミットするように要求する。リソースマネージャ２０８、２１０のいずれか一方がエラーに遭遇した場合、トランザクションマネージャ２１６は、動作２０４、２０６によりトランザクションの内部でなされるすべての変更のロールバックを惹起させる。そうでなければ、トランザクションマネージャ２１６は、トランザクションをコミットさせるであろう。いずれの場合にも、トランザクションマネージャ２１６は、リソースマネージャ２０８、２１０へ決定を伝えることができる。

【００２３】

一つの例において、別のリソースマネージャがトランザクション２０２に参加する前に、メモリリソースマネージャ２０８は、初めはトランザクションマネージャとして作動する。他のリソースの動作が発生しない場合、リソースマネージャ２０８は、上位レベルのトランザクションマネージャ２１６無しで自ら実行することができる。いったん、メモリ以外のリソースマネージャ２１０などの別のリソースマネージャが仮にも参加するとすると、トランザクション処理システム２００は、トランザクション２０２を、“メモリのみのトランザクション”から、トランザクションマネージャ２１６を採用するトランザクションへ進める。メモリリソース２０８は、リソースマネージャになる。

【００２４】

図３は、トランザクションメモリと組み合わせられるトランザクション処理で使用するプロセス例を参照番号３００にて示す。参照番号３０２において、トランザクション２０２は、トランザクションメモリ処理システム２００の例において実行する。トランザクションにおける動作（単数または複数）は、リソースを呼び出す。参照番号３０４において、プロセスは、適切なリソースマネージャが、リソースとともに動作するように自動的に参加することを要請する。そのリソースマネージャは、それがすでにトランザクションにおいて要請されているかどうかをチェックして判断する。もし要請されていなかった場合、リソースマネージャは、対応する動作が行われる前に、参加する。リソースマネージャ２１０は、トランザクションメモリの制御のもとで影響を及ぼされるメモリ変更を管理する。次に参照番号３０６において、動作は、リソースに適用されうる。参照番号３０６において、そのトランザクションにおける他の動作もまた、適切に参加したリソースマネージャを通じて動作する。

【００２５】

参照番号３０８において、トランザクションの終わり、または間近にて、トランザクションマネージャ２１６は、コミットプロトコル２１８を呼び出して、トランザクションが他のスレッドと衝突する場合にロールバックされるべきか、変更がコミットされるかを判断する。ある例におけるコミットプロトコル２１８は、１つのリソースマネージャのみがトランザクションへ参加される場合のように、１フェーズのコミットプロトコルとすることができる。別の例におけるコミットプロトコル２１８は、準備フェーズ３１０とコミットフェーズ３１４とを含む２フェーズのコミットプロトコルとすることができる。

【００２６】
特に、コミットするための要求をアプリケーションから受信すると、トランザクションマネージャ２１６は、２フェーズのコミットプロトコル２１８を使用することができる。２フェーズのコミットプロトコル２１８は、参照番号３１０において、トランザクション上で対応する投票（vote）を取得するために、参加した各リソースマネージャ２０８、２１０上で方法を呼び出すことによって、すべての参加要請された参加者の準備フェーズを開始する。リソースマネージャ２０８、２１０の各々は、それぞれ準備方法またはロールバック方法を呼び出すことによって、コミットまたはロールバックのいずれかに投票する。例えば、メモリ動作のリソースマネージャ２０８は、メモリ衝突が無い場合、コミットするために投票できる。コミットプロトコル２１８の第２フェーズにおいて、トランザクションマネージャは、すべてのリソースマネージャからいずれかの投票を受信してコミットするか、または、少なくとも１つの投票を受信してロールバックするかに応じて、適切に応答する。参照番号３１２において、トランザクションマネージャが、すべてのリソースマネージャからすべて一致したコミットする投票を受信した場合、参照番号３１４において、トランザクションマネージャ２１６は、各リソースマネージャのためのコミット方法を起動する。このトランザクションマネージャは、準備方法をすべて起動している。次に、リソースマネージャは、変更を永続的にして、コミットを完了する。次に、トランザクション２０２の解除後、参照番号３１６においてアプリケーションを進める。参照番号３１２において、いずれかのリソースマネージャが準備フェーズにおいてコミットしない投票をする場合、処理は失敗のときに次のように動作する。

【００２７】
メモリリソースマネージャ２０８により識別されるようにメモリ衝突により引き起こされた失敗は、トランザクションに参加したメモリ以外のリソースマネージャにより識別される失敗と区別できる。例えば、メモリ衝突により引き起こされた失敗によって、参照番号３１８において、トランザクションの自動的な再実行を行うことができる。一般に、トランザクション２０２を含むアプリケーションは、再実行が最終的に成功するときに、再実行に気が付かない。いくつかの実施形態において、メモリ以外のリソースマネージャ２１０などの他のリソースマネージャにより識別される失敗によって、トランザクションを失敗させて、再実行させない。例えば、１または２以上のメモリ以外のリソースマネージャがトランザクションのコミットに反対投票し、およびメモリリソースマネージャは、トランザクションをコミットする投票をする場合、トランザクションは再実行しない。参照番号３１８において、メモリリソースマネージャ２０８が、トランザクションのコミットに反対投票をする場合、メモリ以外のリソースマネージャのいくつか、またはすべてがコミットのために投票するにもかかわらず、参照番号３１８において３２０を介して３０２へのロールバックが生じる。この例では、再実行が生じる。その理由は、メモリ以外のリソースマネージャでの失敗は、メモリ衝突の結果であった可能性があるからである。すべてのリソースマネージャがトランザクションをコミットするために投票する場合、参照番号３１４において、トランザクションはコミットする。

【００２８】
図４は、２フェーズのコミットプロトコルのプロセス４００の例を示す。このコミットプロトコルは、トランザクション２０２の終わり、または間近にて、実装することができる。このコミットプロトコルは、上述の要素３０８および３１０として実行することができる。メモリリソースマネージャ２０８およびトランザクションマネージャ２１６の動作は、同時並列の時系列４０２、４０４を参照して示される。メモリリソースマネージャ２０８（および、マネージャ２１０または他の該当するものなどの他のリソースマネージャ）により、特定のトランザクションにおいて投票することに関心があるという参加の通知４０６をトランザクションマネージャ２１６へ送る。トランザクションを備えるコードが終了するとき、通知４０６は、トランザクションマネージャ２１６が、コミットプロトコル２１８を開始する準備ができていることをトランザクションマネージャ２１６に通知する。参照番号４０８において、トランザクションマネージャは、準備の指令を、メモリリ

ソースマネージャ２０８、およびトランザクションに参加するいずれか他のリソースマネージャへ発する。

【００２９】

参照番号４０８において、準備の指令を受信した後、参照番号４１０において、リソースマネージャ２０８、２１０は、それぞれのオブジェクトを有効にする処理を開始する。メモリリソースマネージャの場合、これは、メモリリソースマネージャが、他のスレッドとの衝突があったかどうかを判断するということを意味することができる。参照番号４１２において、リソースマネージャは、トランザクションマネージャ２１６へ提供される投票において検証結果を、投票として報告する。衝突がない場合、メモリリソースマネージャ２０８はコミットのために投票する。別な方法では、メモリリソースマネージャ２０８は、コミットしない投票をするか、他のリソースマネージャからのコミットのためのいずれかの投票を拒否する。

【００３０】

リソースマネージャのすべてがコミットのために投票する場合、参照番号４１４において、トランザクションマネージャ２１６は、メモリリソースマネージャ２０８およびリソースマネージャ２１０のような、トランザクションに参加するリソースマネージャのすべてへコミットの指令を送る。アトミックブロックトランザクションの結果としていかなるメモリの変更も、共有状態をコミットされ、他のスレッドにさらされる。メモリリソースマネージャの場合、トランザクションメモリは、コミットの指令を受信するまで、トランザクションの独立した作業のいずれもさらさない。このようなコミットの作業は、ロックの解除、バッファメモリ変更の書き戻し（ライトバック）、または暫定的なメモリ変更の取り消しを含む。参照番号４１６において、トランザクションの動作がいったんコミットすると、アプリケーションは次へ進む。参照番号４１８において、プロセス４００は、キューで待機しているアプリケーションの他のコミット要求を処理することができる。

【００３１】

ある例において、２フェーズ・コミットプロトコルの動作は、プロセス４００の順序で生じる。これらの例において、トランザクションマネージャのコミット順序を、準備の指令、およびリソースマネージャからの投票の受信と同期させる。別の例において、トランザクションメモリ処理システムは、悲観的ロックへ変換すること、およびプロセス４００での参加を避けることが可能である。あるいはまた、トランザクションに関与するメモリを保証することができるトランザクションメモリ処理システムは、順番通り以外で動作を行うことができるトランザクションメモリアトミックブロックに関与する、すべての他のメモリから取り外される。

【００３２】

トランザクションメモリ処理システム２００は、トランザクション内のトランザクション、つまり入れ子（ネステッド）トランザクションの使用を提供することもできる。このような場合、コミットプロトコル２１８は、データベーストランザクションの場合のように、入れ子をフラット（flat）または入れ子のままにする。フラットでは、入れ子トランザクションによるトランザクションは、１つのアトミックブロックのように見える。トランザクションがロールバックされる場合、入れ子トランザクションを含む全部のトランザクションを、再実行する。あるいはまた、トランザクションメモリ処理システムは、部分的なロールバックをサポートすることができ、ここで入れ子トランザクションは、コミットするまで再実行される。

【００３３】

トランザクション処理をトランザクションメモリと組み合わせることにより、生産性に利益をもたらす。開発者は、エラー処理および失敗のアトミック性の達成のために、特殊なコードを書くよりも、むしろ、リソースマネージャのあらかじめプログラムされたコードに依拠することができる。すべてのメモリアクセスは、トランザクションメモリの実装により、アトミックブロックトランザクションの範囲の中に自動的に含まれうる。その理由は、アトミックブロックおよびトランザクションの語彙的範囲は、同じだからである。

これにより、開発者は、ある動作が成功し、他の動作が成功しない場合のすべての組み合わせのアドレスにコードを書くことを省くこともできる。

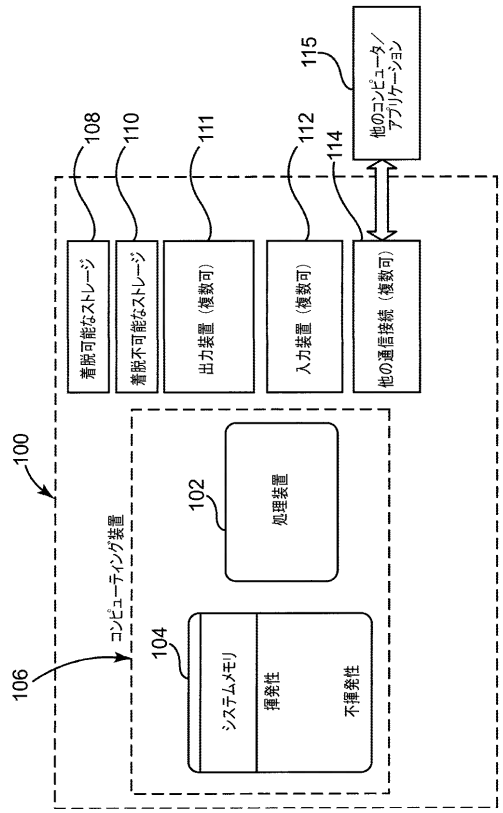
【００３４】

トランザクションリソースは、ワシントン州レッドモンドのマイクロソフト社によってサポートされるプラットフォーム上でサポートされる。このプラットフォームは、マイクロソフトＳＱＬサーバという商品名のもとで販売されるもの、トランザクションメッセージキュー（ＭＳＭＱ）、および、ウインドウズビスタ・トランザクショナルファイルシステムおよびトランザクショナルレジストリという商品名のもとで市販されているものなどのデータベースを含む。トランザクションは、ウインドウズオペレーティングシステムのバージョンＮＴ４から利用可能なＤＴＣ（Distributed Transaction Coordinator）、ウインドウズＸＰより前のウインドウズのバージョンでのＭＴＳ、またはウインドウズビスタで初めて利用可能なＫＴＭ（Kernel Transaction Manager）という商品名のもとでマイクロソフト社により提供されるものなどのトランザクションマネージメント製品を使用して、作成および管理されうる。マイクロソフトから市販されている.NETなどの管理コードにおいて、システムトランザクションの特徴は、ＤＴＣを使用し、および自己のＬＴＭ（lightweight transaction manager）を提供する、管理アプリケーションプログラムのインターフェースを提供する。ＬＴＭは、メモリリソースマネージャなどの揮発性リソースマネージャの作成もする、メカニズムを提供する。かかる揮発性リソースマネージャは、データベースＳＱＬなどの他のリソースマネージャとともにトランザクションに参加する。

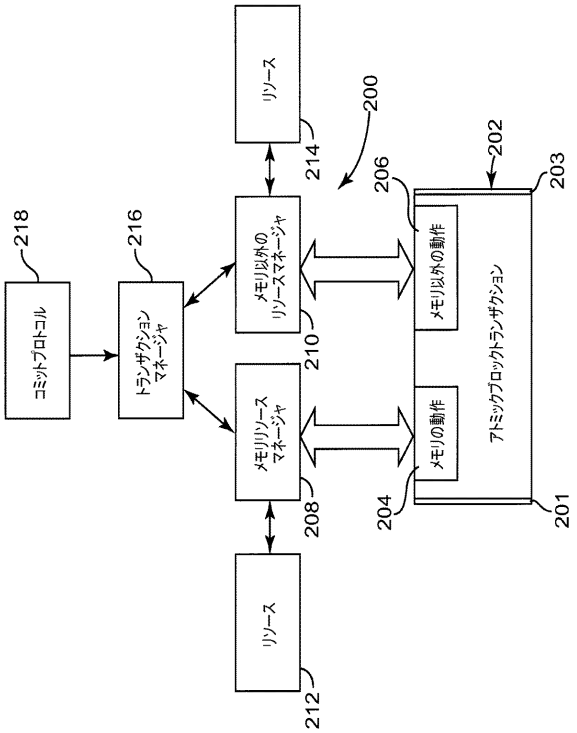
【００３５】

特定の実施形態が、本明細書に示され、および説明されているけれども、当業者にとって、本発明の範囲から逸脱することなく、種々の代替および／または同等な実施形態を、ここに示され、および説明された特定の実施形態と置き換えることが可能であることが理解されるであろう。この出願は、本明細書で議論された特定の実施形態のいかなる改変または変形も包含することを意図する。それゆえ、本発明は、請求項およびそれらの均等物によってのみ限定されることを意図する。

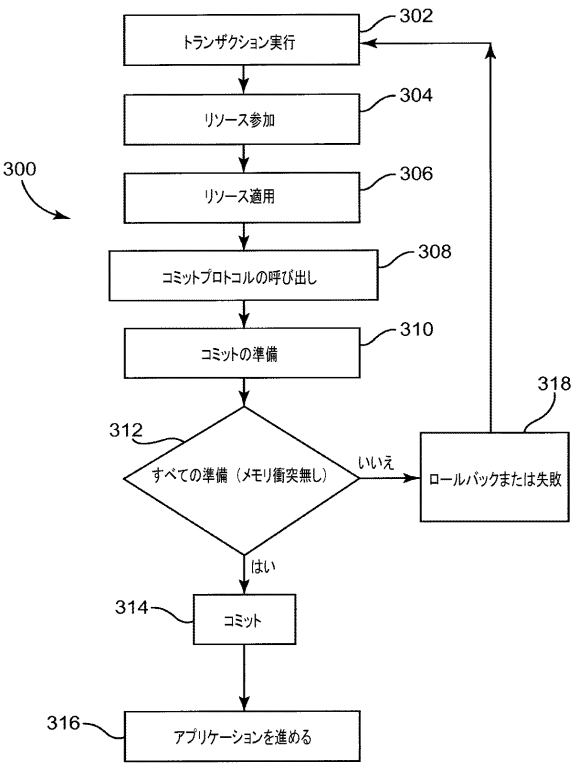
【図１】


100
102
処理装置
104
106
コンピューティング装置
システムメモリ
揮発性
不揮発性
108
着脱可能なストレージ
110
着脱不可能なストレージ
111
出力装置（複数可）
112
入力装置（複数可）
114
他の通信接続（複数可）
115
他のコンピュータ／アプリケーション

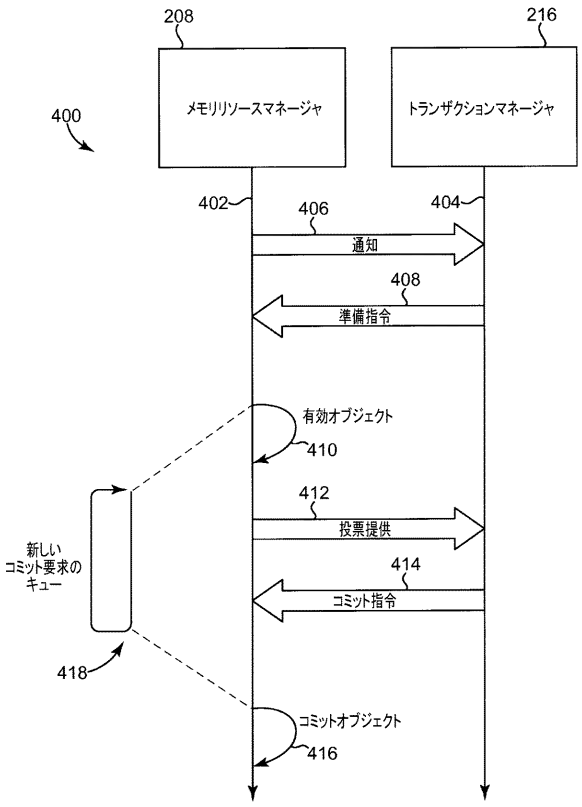
【図２】


200
202
203
リソース
214
206
メモリ以外の動作
メモリ以外のリソースマネージャ
218
216
コミットプロトコル
トランザクションマネージャ
210
メモリリソースマネージャ
メモリの動作
アトミックブロックトランザクション
204
208
201
リソース
212

【図３】


300
302
トランザクション実行
304
リソース参加
306
リソース適用
308
コミットプロトコルの呼び出し
310
コミットの準備
312
すべての準備（メモリ衝突無し）
いいえ
318
ロールバックまたは失敗
はい
314
コミット
316
アプリケーションを進める

【図４】


400
208
メモリリソースマネージャ
216
トランザクションマネージャ
402
406
404
通知
408
準備指令
有効オブジェクト
410
412
投票提供
新しい
コミット要求の
キュー
414
コミット指令
418
コミットオブジェクト
416

フロントページの続き

(74)代理人 100120112
弁理士 中西 基晴
(74)代理人 100147991
弁理士 鳥居 健一
(74)代理人 100119781
弁理士 中村 彰吾
(74)代理人 100162846
弁理士 大牧 綾子
(74)代理人 100173565
弁理士 末松 亮太
(74)代理人 100138759
弁理士 大房 直樹
(74)代理人 100091063
弁理士 田中 英夫
(72)発明者 アレクサンダー ダディオモブ
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 エルシーエー－インターナショナル パテンツ内
(72)発明者 ダナ グロフ
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 エルシーエー－インターナショナル パテンツ内
(72)発明者 ヨセフ レバノニ
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 エルシーエー－インターナショナル パテンツ内
(72)発明者 ジェイムズ イー. ジョンソン
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 エルシーエー－インターナショナル パテンツ内

審査官 漆原 孝治

(56)参考文献 特表２００３－５３０６４６（ＪＰ，Ａ）
特開２００２－０４９５１９（ＪＰ，Ａ）
特開２０００－０２０３６４（ＪＰ，Ａ）
特開２００１－１８８６９６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ ９／５２