(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2010-529559
(P2010-529559A)
(43)公表日　平成22年8月26日(2010.8.26)

(51)Int.Cl.	F I	テーマコード（参考）
G06F 9/45 (2006.01)	G06F 9/44 322G	5B081

審査請求 未請求　予備審査請求 未請求　（全 28 頁）

(21)出願番号	特願2010-511260 (P2010-511260)
(86)(22)出願日	平成20年5月30日 (2008.5.30)
(85)翻訳文提出日	平成21年12月2日 (2009.12.2)
(86)国際出願番号	PCT/US2008/065363
(87)国際公開番号	WO2008/151046
(87)国際公開日	平成20年12月11日 (2008.12.11)
(31)優先権主張番号	11/810,111
(32)優先日	平成19年6月4日 (2007.6.4)
(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　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　ジョン　ジョセフ　ダフィー
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　インターナショナル　パテンツ内

最終頁に続く

(54)【発明の名称】トランザクションを用いるシーケンシャルフレームワークの並行化

(57)【要約】
シーケンシャルループを、トランザクショナルメモリシステムで使用するために、並列ループに変換するための各種の技術および手法を開示する。オープンエンドおよび／またはクローズドエンドのシーケンシャルループを、並列ループに変換することができる。例えば、当初のシーケンシャルループを含むコードのセクションを解析して、当初のシーケンシャルループについての繰り返し定数を決定する。当初のシーケンシャルループを、繰り返し定数までの数だけトランザクションを発生することのできる並列ループに変換する。別の例のように、オープンエンドシーケンシャルループは、それぞれの作業項目を含む個々のトランザクションを、各スペキュレーションパイプラインの繰り返し毎に発生する並列ループに変換することができる。その並列ループは次いで、異なるスレッド上で実行される個々のトランザクションのうち少なくともいくつかと共に、前記トランザクショナルメモリシステムを用いて実行される。

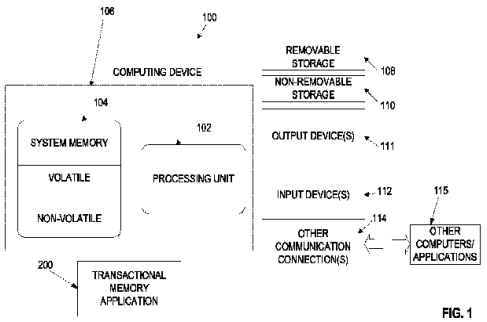


FIG. 1

【特許請求の範囲】
【請求項１】
クローズドエンドのシーケンシャルループを並列ループに変換する方法であって、
トランザクショナルメモリシステムを提供するステップと、
当初のシーケンシャルループを含むコードの第１セクションを解析して、前記当初のシーケンシャルループが実行する繰り返し定数を決定するステップと、
前記当初のシーケンシャルループを含むコードの前記第１セクションを、並列ループを含むコードの第２セクションに変換するステップであって、前記並列ループは前記繰り返し定数までの数の複数のトランザクションを生成するよう作動可能であり、前記トランザクションは前記並列ループの少なくとも一部を並列に実行可能にする、ステップと、
異なるスレッド上で実行する複数のトランザクションのうちの少なくともいくつかと共に、前記トランザクショナルメモリシステムを使用してコードの第２セクションを実行するステップと
を備えたことを特徴とする方法。
【請求項２】
前記繰り返し定数は、前記当初のシーケンシャルループと比較してループ終了を判定するための一定値を検索することにより決定されることを特徴とする請求項１に記載の方法。
【請求項３】
前記トランザクションの各々がそれぞれの誘導変数カウンタをコミット連続番号として使用し、該コミット連続番号は所定のコミット順処理を使用してトランザクション各々を正しい順序でコミットするのを保証することを特徴とする請求項１に記載の方法。
【請求項４】
請求項１に記載の各ステップをコンピュータに実行させるための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可能記憶媒体。
【請求項５】
コンピュータに実行させるためのコンピュータ実行可能命令を有するコンピュータ読取可能記憶媒体であって、該コンピュータ実行可能命令は、
トランザクショナルメモリシステムを提供するステップと、
オープンエンドシーケンシャルループを含むコードの前記第１セクションを、並列ループを含むコードの第２セクションに変換するステップであって、前記並列ループはスペキュレーションパイプラインの繰り返し毎にそれぞれの作業項目を含む個々のトランザクションを生成するように作動可能であるステップと、
異なるスレッド上で実行する前記個々のトランザクションのうち少なくともいくつかと共に、前記トランザクショナルメモリシステムを使用してコードの前記第２セクションを実行するステップと、
を含み、各ステップをコンピュータに実行させることを特徴とするコンピュータ読取可能記憶媒体。
【請求項６】
コードの前記第２セクションは、前記トランザクションのうち少なくともいくつかを並行して実行するように生成されることを特徴とする請求項５に記載のコンピュータ読取可能記憶媒体。
【請求項７】
コードの前記前記第２セクションは、前記オープンエンドシーケンシャルループのコンパイラ分析を実行することなく生成されることを特徴とする請求項５に記載のコンピュータ読取可能記憶媒体。
【請求項８】
当初の入力対出力のマッピングが、前記トランザクションを所定のコミット順でコミットすることにより維持されることを特徴とする請求項５に記載のコンピュータ読取可能記憶媒体。

【請求項９】
前記所定のコミット順が、前記オープンエンドシーケンシャルループの実行順と整合性が取れていることを特徴とする請求項８に記載のコンピュータ読取可能記憶媒体。
【請求項１０】
オープンエンドシーケンシャルループから生成される並列ループを実行する方法であって、
オープンエンドシーケンシャルループから生成される並列ループで実行する繰り返しの回数を見積もるスペキュレーションパイプラインを生成するステップと、
前記スペキュレーションパイプラインの各繰り返しをもとに、それぞれの作業項目を含む個々のトランザクションを生成するステップと、
異なるスレッドの上で個々のトランザクションのうち少なくともいくつかを実行するステップと、
それぞれの作業項目毎に終了条件を評価するステップと、
前記それぞれの作業項目のうちの特定の１つが、前記並列ループを終了する時期に到達したと判定するときに、前記それぞれの作業項目のうちの前記特定の１つの先行者をコミットし、後継者を廃棄するステップと、
を含むことを特徴とする方法。
【請求項１１】
それぞれの作業項目を実行しながら、現在の繰り返し値を読み出すことを特徴とする請求項１０に記載の方法。
【請求項１２】
それぞれの作業項目毎の前記現在の繰り返し値は、所定のコミット順処理において、コミット連続番号として使用されること、を特徴とする請求項１１に記載の方法。
【請求項１３】
前記現在の繰り返し値は、それぞれの作業項目毎にアクセス可能な値の極小の増分を実行することにより読み出されることを特徴とする請求項１１に記載の方法。
【請求項１４】
前記オープンエンドシーケンシャルループの当初の実行と整合性が取れているコミット順が達成されることを特徴とする請求項１０に記載の方法。
【請求項１５】
前記オープンエンドシーケンシャルループが、ｗｈｉｌｅループであることを特徴とする請求項１０に記載の方法。
【請求項１６】
前記オープンエンドシーケンシャルループが、ｄｏ　ｗｈｉｌｅループであることを特徴とする請求項１０に記載の方法。
【請求項１７】
前記オープンエンドシーケンシャルループが、ｆｏｒループであることを特徴とする請求項１０に記載の方法。
【請求項１８】
前記スペキュレーションパイプラインの初期値が、並列ループを実行するコンピュータ上で利用可能な処理装置の数に少なくとも部分的に基づいて計算されることを特徴とする請求項１０に記載の方法。
【請求項１９】
適応的統計データを用いて、前記並列ループの後の実行のために前記スペキュレーションパイプラインを調整することを特徴とする請求項１０に記載の方法。
【請求項２０】
請求項１０に記載の各ステップをコンピュータに実行させるための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可能記憶媒体。
【発明の詳細な説明】
【技術分野】

【０００１】
本発明は、トランザクションを用いるシーケンシャルフレームワークの並行化に関する。
【背景技術】
【０００２】
ソフトウェアトランザクショナルメモリ（ＳＴＭ）は、並行計算作業において共有メモリに対するアクセスを制御するためのデータベーストランザクションに類似した並行処理制御機構である。トランザクショナルメモリの文脈におけるトランザクションは、共有メモリに対する一連の読取書込を実行するコード１個である。ＳＴＭは、在来のロック機構の代替として使用される。プログラマが（例えば、アトミック（分割不可能な単位）で）宣言型注釈をコードブロックの周りに置いてそれらが必要とする安全属性を示すと、システムは、このブロックは他の保護コード領域に関してアトミックに実行（分割不可能な単位として実行）されることを、自動的に保証する。そのソフトウェアトランザクショナルメモリプログラム作成モデルは、ロックに基づく優先順位逆転およびデッドロック問題を防止する。
【０００３】
典型的ＳＴＭシステムには、多くの利点があるが、それでも、プログラマは、意図しないメモリアクセス順序付けを注意深く避けることを要求される。例えば、トランザクションをコミットする（即ち、コミット処理の）順序は、典型的ＳＴＭ環境においては拘束を受けない。トランザクションはコミットしようと互いに競争する。このことは、トランザクション１がトランザクション２の前にコミットするか後にコミットするかは、しばしばプログラムの動的スケジューリングの産物である（そしてしばしばプログラム特有のロジックの産物でもある）ことを意味する。その上、メモリの同一片に書込をしようとするように、２つのトランザクションが衝突するとき、それらのコミット順は、可能な多数の競合管理方針のうち１つに基づいて任意に決定することができる。これら筋書きの双方において、特定のコミット順は保証されない。したがって、自分のプログラムが何らかの順で正しく働くことを確認することはプログラマの双肩に掛かっている。これは並行プログラム作成を極めて困難にする。
【発明の概要】
【発明が解決しようとする課題】
【０００４】
実行の順序が重要であり、並行性が極めて魅力的である１つの筋書きは、ループの多重繰り返しを並行して実行するときである。下記のような古典的なｆｏｒ・・・ｅａｃｈループを取り上げる。
【０００５】
```
ＦｏｒＥａｃｈ（ｓｔｒｉｎｇ　ｓ　ｉｎ　Ｌｉｓｔ＜ｓｔｒｉｎｇ＞）
{
　Ｓ；
}
```
【０００６】
ループの各繰り返しの間、ループ本体におけるステートメントＳが実行される。このようなループは、二番目の繰り返しの始まる前にループの最初の繰り返しを終了させるなど、順次実行するために書かれていた。このようなシーケンシャルループを、起こりうる副次効果または順序依存性に対処する特別な注意を払うことなく並行して実行すると、予期しない結果を生じることがある。
【課題を解決するための手段】
【０００７】
トランザクショナルメモリシステム内のトランザクションに対して順序付けを適用するため、各種の技術および手法を開示する。トランザクショナルメモリシステムは、複数のトランザクションについて所定のコミット順を指定することができる機能を提供する。そ

の所定のコミット順は、実行時に、そのトランザクショナルメモリシステム内のそのトランザクションをコミットする順序を決定するのに使用される。１つの実施において、その所定のコミット順は全体の順序付けまたは部分的順序付けの何れかにすることができる。全体の順序付けの場合、そのトランザクションは、線形順序でコミットすることを強制される。部分的順序付けの場合、そのトランザクションは、許容可能な筋書きのうち１つでコミットすることができる。１つの実施において、コミットアービトレータが、次にコミットが許されるべきトランザクションをあらわす次回コミット値を追跡し続けており、特定のトランザクションがコミットする準備が整ったとき、そのコミット順番号がコミットアービトレータの次回コミット値と一致しているならば、コミットすることを許される。

【０００８】

第１トランザクションと第２トランザクションとの間に衝突が生じたとき、競合管理処理が呼び出される。その競合管理処理においては上記所定のコミット順を用いて、その第１トランザクションとその第２トランザクションの何れが衝突に勝って先に進むことを許されるかの決定を助ける。

【０００９】

トランザクショナルメモリシステムと共に使用するための、シーケンシャルループを並列ループに変換するための手法を開示する。トランザクショナルメモリに基づくシステムを提供する。当初のシーケンシャルループを含むコードの第１セクションを、トランザクションを用いて当初の入力対出力マッピングを維持する並列ループを含むコードの第２セクションに変換する。例えば、当初のシーケンシャルループは、当初のシーケンシャルループの各繰り返しをもとに所定のコミット順処理にしたがう個々のトランザクションを作成し、次いでそのトランザクションを、並行して実行されるように異なるスレッドに割り当てることにより、並列ループに変換することができる。並列ループを実行する間の特定のトランザクションの中から処理されない例外（unhandled exception）を発見すると、その特定のトランザクションおよび、もしあれば先行トランザクションをコミットすることにより、状態変更を行い、後続トランザクションの何れかにより行われる状態変更を廃棄する。さもなければ、すべてのトランザクションをコミットする。

【００１０】

１つの実施において、オープンエンドおよび／またはクローズドエンドのシーケンシャルループを並列ループに変換することができる。例えば、当初のシーケンシャルループを含むコードのセクションを解析して、その当初のシーケンシャルループのために繰り返しの定数を決定する。その当初のシーケンシャルループは、その繰り返しの定数までの個数のトランザクションを発生することのできる並列ループに変換される。別の例として、オープンエンドシーケンシャルループは、スペキュレーションパイプラインの各繰り返し当たりの作業項目それぞれを含む個々のトランザクションを発生する並列ループに変換することができる。これらのトランザクションには異なるスレッドを割り当てて、並列ループの少なくとも一部は並行して実行可能にする。その並列ループはこのとき、所定のコミット順の効能を有するトランザクショナルメモリシステムの保護の下で実行される。

【００１１】

１つの実施において、オープンエンドのシーケンシャルループから作成された並列ループを実行する方法を提供する。オープンエンドのシーケンシャルループから作成される並列ループにおいて実施すべき繰り返し数を見積もるスペキュレーションパイプラインを作成する。上記システムは、そのスペキュレーションパイプラインの各繰り返しをもとに、それぞれの作業項目を含む個々のトランザクションを作成する。これら個々のトランザクションは次いで、異なるスレッドに割り当てられ、結局それらが並行して実行される結果になる。それぞれの作業項目毎に終了条件を評価する。それぞれの作業項目のうちの特定の１つがその並列ループを終了する時期に達したと判定するとき、先行のものはコミットされ、後続のものは廃棄される。

【００１２】

本概要は、詳細な説明において下記に更に詳細に記述する概念の要点を簡略形で紹介す

るために提供されたものである。本概要は、請求する主題の重要特徴または不可欠特徴を特定することを目的とはせず、請求主題の範囲決定における補助として使用されることを目的としてもいない。

【図面の簡単な説明】

【００１３】

【図１】１つの実施のコンピュータシステムの概略図である。

【図２】図１のコンピュータシステム上で作動する１つの実施のトランザクショナルメモリアプリケーションの概略図である。

【図３】図１のシステムの１つの実施のためのハイレベル処理の流れ図である。

【図４】図１のシステムの１つの実施のための処理の流れ図であって、コミットアービトレータを使用して所定のコミット順を実施することを含むステージを示す流れ図である。

【図５】図１のシステムの１つの実施のための処理の流れ図であって、コミットアービトレータを使用して複数のトランザクションの全体の順序付けを実施することを含むステージを示す流れ図である。

【図６】図１のシステムの１つの実施のための処理の流れ図であって、コミットアービトレータを使用して複数のトランザクションの部分的順序付けを実施することを含むステージを示す流れ図である。

【図７】図１のシステムの１つの実施のための処理の流れ図であって、所定のコミット順の情報を使用して、競合を管理する競合管理処理を提供することを含むステージを示す流れ図である。

【図８】図１のシステムの１つの実施のための処理の流れ図であって、所定のコミット順の情報を使用して入れ子トランザクションとの競合を管理する競合管理処理を提供することを含むステージを示す流れ図である。

【図９】共通祖先を持つ最高順位の祖先による模範的な系図を示すロジック図である。

【図１０】共通祖先を持たない最高順位の祖先による模範的な系図を示すロジック図である。

【図１１】図１のシステムの１つの実施のための処理の流れ図であって、トランザクショナルメモリシステム内のコミットアービトレータを使用することにより、無駄な作業の量を減少することを含むステージを示す流れ図である。

【図１２】図１のシステムの１つの実施のための処理の流れ図であって、競合管理処理において系図全体を解析して適切な競合解消策を決定することを含むステージを示す流れ図である。

【図１３】図１のコンピュータシステムの上で働く１つの実施のトランザクショナルメモリアプリケーションの概略図である。

【図１４】図１のシステムの１つの実施のための処理の流れ図であって、当初のシーケンシャルループを並列ループに変換することを含むステージを示す流れ図である。

【図１５】図１のシステムの１つの実施のための処理の流れ図であって、所定のコミット順処理を使用して、並列ループ内のトランザクションは適切な順序でコミットされることを確認することを含むステージを示す流れ図である。

【図１６】図１のシステムの１つの実施のための処理の流れ図であって、コミットアービトレータを使用して、並列ループが実行している間に生じる競合を検知して対処することを含むステージを示す流れ図である。

【図１７】図１のシステムの１つの実施のための処理の流れ図であって、コミットアービトレータを使用して、並列ループを実行している間に生じる処理されない例外（unhandled exception）を検知して対処することを含むステージを示す流れ図である。

【図１８Ａ】当初のシーケンシャルループから並列ループへの模範的な変換のための仮想ソースコードを示す図である。

【図１８Ｂ】当初のシーケンシャルループから並列ループへの模範的な変換のための仮想ソースコードを示す図である。

【図１９】図１のシステムの１つの実施のための処理の流れ図であって、クローズドエン

ドシーケンシャルループを並列ループに変換することを含むステージを示す流れ図である。
【図２０】図１のシステムの１つの実施のための処理の流れ図であって、スペキュレーションパイプラインを使用して、オープンエンドシーケンシャルループを並列ループに変換することを含むステージを示す流れ図である。
【図２１】図１のシステムの１つの実施のための処理の流れ図であって、オープンエンドシーケンシャルループから作成された並列ループを実行することを含むステージを示す流れ図である。
【図２２】図１のシステムの１つの実施のための処理の流れ図であって、オープンエンドシーケンシャルループから作成された並列ループにおける各作業項目を適切な順序でコミットさせていることを確認することを含むステージを示す流れ図である。
【図２３】図１のシステムの１つの実施のための処理の流れ図であって、スペキュレーションパイプラインを計算して並列ループの中に幾つの繰り返しを含むかを決定することを含むステージを示す流れ図である。
【図２４Ａ】当初のオープンエンドシーケンシャルループから並列ループへの模範的な変換のための仮想ソースコードを示す図である。
【図２４Ｂ】当初のオープンエンドシーケンシャルループから並列ループへの模範的な変換のための仮想ソースコードを示す図である。
【発明を実施するための形態】
【００１４】
本発明の原理の理解を促進する目的のために、ここで図面に示す実施例を参照し、特定の言語を用いてそれを記述する。とは言っても、それによる範囲の限定は意図されていないことは理解されるであろう。記述する実施例における代替案および追加の変更、およびここに記述するような原理のさらなる応用は何れも、当業者にとっては普通に思い浮かぶはずのものである。
【００１５】
このシステムは、トランザクショナルメモリシステムとしての一般的文脈で記述するが、このシステムはこれらに加えて別の目的にも役立つ。１つの実施において、ここに記述する１つまたは複数の手法は、ＭＩＣＲＯＳＯＦＴ（登録商標）．ＮＥＴフレームワークのようなフレームワークプログラムの範囲内の特性として、またはソフトウェアアプリケーションを開発する開発者のためのプラットホームを提供する任意の別の型のプログラムまたはサービスが備える特性として実施することができる。別の実施において、ここに記述する手法の１つ以上が、同時実行環境において実行するアプリケーションの開発を扱う別のアプリケーションの伴う特性として実施される。
【００１６】
１つの実施においては、複数のトランザクションについて、所定のコミット順を指定することができる機能を、トランザクショナルメモリシステムの中に備える。所定のコミット順は、トランザクションをコミットする順序の決定を助けることに使用される。１つの実施例において、競合管理処理は、第１トランザクションと第２トランザクションとの間に衝突を生じるときに呼び出される。次いでその所定のコミット順を競合管理処理において使用して、第１トランザクションまたは第２トランザクションが競合に勝って先に進むことを許されるべきかどうかを決定するのを助ける。
【００１７】
別の実施において、当初のシーケンシャルループを並列ループに転換する機能を、トランザクショナルメモリシステムの中に提供する。当初のシーケンシャルループは、当初の入力対出力マッピングは維持されることを保証する方法で、並列ループに転換される。用語「当初の入力対出力マッピングは維持される」は、ここで使用するとき、並行化ループ実行の後のプログラムの状態は、代わりにシーケンシャルループが働いたときと全く同様であることを意味する。１つの実施において、当初のシーケンシャルループの各繰り返しをトランザクションの中に置き、次いでここに記述する所定のコミット順処理を用いてそ

のトランザクションは適切な順序でコミットすることを確認することにより、並列ループにおいて当初の入力対出力マッピングは維持される。

【００１８】

ここで明確にする例の多くは、ソフトウェアトランザクショナルメモリシステムの文脈で記述するけれども、他の実施において、ここに明確にする幾つかの、すべての、または、さらなる追加の機能および／または手法は、ソフトウェアトランザクショナルメモリシステムとは別個にまたは併せての何れでも、ハードソフトウェアトランザクショナルメモリシステムを用いて実施することができることは理解されるであろう。

【００１９】

図１に示すように、このシステムの１つまたは複数の部分を実施するために使用する模範的なコンピュータシステムは、計算装置（コンピューティングデバイス）１００のような計算装置を含む。その最も基本的な構成において、計算装置１００は典型的に、少なくとも１つの処理装置１０２およびメモリ１０４を含む。計算装置の正確な構成および型により、メモリ１０４は、（ＲＡＭのような）揮発性、（ＲＯＭ、フラッシュメモリなどのような）不揮発性、またはその２つの何らかの結合である。この最も基本的な構成を図１に破線１０６で示す。

【００２０】

加えて、計算装置１００は、さらなる特性／機能性をも有する。例えば、装置１００は、これらに限定はされないが磁気的または光学的ディスクまたはテープを含む追加の記憶装置（取り外し可能なおよび/または取り外しができない）をも含む。このような追加記憶装置は、取り外し可能な記憶装置１０８および取り外しができない記憶装置１１０により図１に示している。コンピュータ記憶媒体は、コンピュータ読取可能命令、データ構造体、プログラムモジュールまたは他のデータのような情報ストレージのため任意の方法または手法で実施される揮発性および不揮発性、取り外し可能なおよび取り外しができない、の媒体を含む。メモリ１０４、取り外し可能な記憶装置１０８および取り外しができない記憶装置１１０はすべて、コンピュータ記憶媒体の例である。コンピュータ記憶媒体は、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリまたは他のメモリ技術、ＣＤ－ＲＯＭ、デジタル多目的ディスク（ＤＶＤ）、または他の光学的記憶装置、磁気カセット、磁気テープ、磁気ディスク記憶装置または他の磁気記憶装置、若しくは所望の情報の記憶に使用することができて装置１００がアクセスすることのできる他の媒体の何れをも含むが、これらに限定はされない。このようなコンピュータ記憶媒体は装置１００の一部であることがある。

【００２１】

計算装置１００は、計算装置１００が他のコンピュータ／アプリケーション１１５と連絡可能にする１つまたは複数の通信接続１１４を備える。装置１００は、キーボード、マウス、ペン、音声入力装置、タッチ入力装置、などのような、入力装置１１２をも有する。ディスプレイ、スピーカ、プリンタ、などのような出力装置１１１も含むことができる。これらの装置は技術的によく知られているので、ここで長々と論じる必要はない。１つの実施において、計算装置１００は、トランザクショナルメモリアプリケーション２００を含む。トランザクショナルメモリアプリケーション２００は、図２においてさらに詳細に記述する。

【００２２】

ここで図１に対する参照を続けると共に、図２に転じて、計算装置１００の上で作動するトランザクショナルメモリアプリケーション２００を示す。トランザクショナルメモリアプリケーション２００は、計算装置１００の上に存在するアプリケーションプログラムのうちの１つである。しかしながら、トランザクショナルメモリアプリケーション２００は、代わりにまたは追加して１つまたは複数のコンピュータの上および／または図１に示すものとは異なる変換の中にコンピュータ実行可能命令として具体化されることは理解されるであろう。代わりにまたはさらに、トランザクショナルメモリアプリケーション２００の１つまたは複数の部分は、別のコンピュータおよび／またはアプリケーション１１５

、若しくはコンピュータソフトウェア技術に携わる人の心に浮かぶ通りの別のこのような変換の上で、システムメモリ１０４の部分とすることができる。

【００２３】

トランザクショナルメモリアプリケーション２００は、ここに記述する手法の幾つかまたは全部を実行することを担当するプログラムロジック２０４を含む。プログラムロジック２０４は、トランザクショナルメモリ（ＳＴＭ）システムを提供するロジック２０６、そのＳＴＭシステム内の複数のトランザクションについて、所定のコミット順を指定可能にするコミットアービトレータを動的または静的に作成するロジック２０８、そのコミットアービトレータがその所定のコミット順を実行時において使用して、そのトランザクショナルメモリシステム内の複数のトランザクションがコミットする順序を決定するのを助けるロジック２１０、第１トランザクションと第２トランザクションとの間に衝突が生じたときに呼び出される競合管理処理を提供するロジック２１２、その所定のコミット順をその競合管理処理において使用して、第１トランザクションまたは第２トランザクションの何れがその競合に勝って先に進むことを許されるかを決定する（例えば、同一トランザクショングループの２つのトランザクションのうちの何れが低いコミット順を有するかによって決める）のを助けるロジック２１４、そのコミットアービトレータを作動可能にして、その所定の順序付けに使用し１つまたは複数の順序付け値（例えば、全体の順序付けにおいては－複数のトランザクションのうちでコミットすることを許される次のトランザクションを表している次回コミットフィールド）を追跡するため使用可能にするため、および１つまたは複数の順序付け値を所与のトランザクションの特定コミット順番号に対して比較しそのトランザクションのコミットは実行されるべき順序付けを適切に与えられているか否かを調べるロジック２１６、および当該アプリケーションをオペレーション（運用）するための他のロジック２２０を含む。１つの実施において、プログラムロジック２０４は、プログラムロジック２０４のプロシージャに対する信号呼出を使用するなど、別のプログラムからプログラム的に呼び出される作動をすることができる。

【００２４】

ここで図１～図２に対する参照を続けると共に、図３～図１０に転じて、トランザクショナルメモリアプリケーション２００の１つまたは複数の実施を実行するためのステージをさらに詳細に記述する。図３は、トランザクショナルメモリアプリケーション２００のためのハイレベル処理の流れ図である。１つの形として、図３の処理は、少なくとも部分的に計算装置１００の運用ロジックにおいて実施される。このプロシージャ（手続き）は、出発点２４０において始まり、トランザクショナルメモリシステム（例えば、ソフトウェアトランザクショナルメモリシステム）を提供すること（ステージ２４２）を伴う。複数のトランザクション（即ち、動的または静的に割り当てられた）について所定のコミット順（即ち、全体の順序付けまたは部分的順序付け）を指定可能にする機能を提供する（ステージ２４４）。用語「所定のコミット順」は、ここで使用するとき、関連トランザクションの特定のグループは、そのトランザクションのラン開始前の何れかの時点で決定した通りにコミットされる明確な順序を含むことを意味する。用語トランザクションの「グループ」は、ここで使用するとき、同一のコミットアービトレータにより管理されるトランザクションの特定のセット（例えば、複数のトランザクション）と同時に、これらトランザクションの入れ子チルドレン（nested children）を含む。

【００２５】

その所定のコミット順を実行時に使用して、そのトランザクショナルメモリシステム内の複数のトランザクションがコミットする順序を決定するのを助ける（ステージ２４６）。その所定のコミット順を使用して、その複数のトランザクションの2つまたはそれ以上の複数の間で生じる衝突を解決するのを助ける（ステージ２４８）。処理は終点２５０において終了する。

【００２６】

図４は、コミットアービトレータを使用して所定のコミット順を実施することを含むステージを示す。１つの形として、図４の処理は、計算装置１００のオペレーティングロジ

ックにおいて、少なくとも部分的に実施される。このプロシージャは、出発点２７０において始まり、トランザクショナルメモリシステム用の、そのコミットアービトレータは、複数のトランザクションについて所定のコミット順を指定させる１つまたは複数のコミットアービトレータを提供すること（ステージ２７２）を伴う。用語「コミットアービトレータ」は、ここで使用するとき、相互のために順序付けをしなければならない１つまたは複数のトランザクションのグループの管理を担当する任意の型のプログラム、特性、または処理を含むことを意味する。１つの実施において、１つまたは複数のコミットアービトレータは、任意の所与の時期にプログラムにおいて作動することができる。例えば、必要な数のコミットアービトレータを作成して、トランザクションの様々なグループを管理することができる。そのコミットアービトレータは、トランザクションの相互のために適切な順序付けを決定するのに使用する１つまたは複数の順序付け値を追跡して更新する（ステージ２７４）。全体の順序付けの場合は、複数のトランザクションのうち次にコミットすべき次のトランザクションをあらわず次回コミットフィールドを使用することができる（ステージ２７４）。部分的順序付けの場合は、可能な様々の順序の有向グラフ（directed graph）を、その順序付け値を使用して追跡する。必要に応じて、そのコミットアービトレータは、その所定のコミット順を使用して、複数のトランザクションそれぞれのためにコミット順番号を与える（ステージ２７６）。

【００２７】

複数のトランザクションのうちの特定のトランザクションがコミットの準備をするとき、１つまたは複数の順序付け値がそのコミットは適切であることを示していると、そのコミットアービトレータは、そのトランザクションがコミットするのを許可する（ステージ２７８）。全体の順序付けの場合、次回コミットフィールドと特定のトランザクションについてのコミット順番号とが同一番号を有するとき、この筋書きが生じる。このような筋書きにおいて、そのコミットアービトレータは、そのトランザクションがコミットすることを許し、コミットに成功するときは、次いで次回コミットフィールドを数列の次の番号（例えば、次に高い番号）に増加する（ステージ２７８）。複数のトランザクションのうちの特定のトランザクションがコミットの準備をするとき、その特定のトランザクションについてのコミット順番号は順序付け値に対して比較するとき、そのコミットは適切でないことが明らかになると、その特定のトランザクションは、先行トランザクションがコミットした後のある時期に解除されるまで保留モードに置く（ステージ２８０）。全体の順序付けの場合、この保留モードに入るのは、次回コミットフィールドと特定のトランザクションについてのコミット順番号とが同一の値を持っていないときである。

【００２８】

１つの実施において、システムは、その直前の先行者がコミットした後、トランザクションの活動を開始させる。この場合は、それは正しくコミットできる。代わりに、システムは、直前ではない先行者がコミットした後、その直前の先行者が未だコミットしていない場合に、トランザクションの保留を解除し活動開始を選ぶことができる。保留解除の後、システムは、現実にコミットすることがそのトランザクションについて適切であるか否かを点検する。適切ならばそのトランザクションはコミットする。その処理は、終点２８２において終了する。

【００２９】

図５は、コミットアービトレータを使用して複数のトランザクションの全体の順序付けを実施することを含むステージの１つの実施を示す。１つの形として、図５の処理は計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。このプロシージャは、出発点２９０において始まり、複数のトランザクションについて所定の全体の順序付けを指定すること（例えば、複数のトランザクションがコミットすべき正確な順序を指定すること）を許すよう動作可能な１つまたは複数のコミットアービトレータを提供すること（ステージ２９２）を伴う。複数のトランザクションのうちの特定のトランザクションがそのコミット時点に達するとき、そのコミット順を実施するため、その特定のトランザクションのコミット順をそのコミットアービトレータの次回コミットフィール

ドと比較する（ステージ２９６）。１つの実施において、このシステムが（明白に衝突が無いとの理由などで）全体の順序付けを実施する必要が無いと決定するときには、全体の順序付け要件は必要に応じて破棄することができ（ステージ２９４）、処理は終点３０２において終了する。

【００３０】

コミット順序付けを実施しようとするときで、しかも特定のトランザクションのコミット順がコミットアービトレータの次回コミットフィールドと同一値を有するときは（判定点２９６）、その特定のトランザクションをコミットし、コミットに成功すると、次回コミットフィールドを増加して、次の後継者があれば、それを解除（awake;起動）させる（ステージ２９８）。特定のトランザクションのコミット順がコミットアービトレータの次回コミットフィールドと同一値を有しないときは（判定点２９６）、その特定のトランザクションを、先行トランザクションがコミットした後のちょうどよい時点で解除するまで、保留／休眠モードに置く（ステージ３００）。１つの実施においては、その後の時点で、先行者と衝突が起こるときは、その特定のトランザクションは、以上終了し、ロールバックして先行者を前に進めるよう要求される。さもなければ、このような衝突が起こらない場合は、その特定のトランザクションは、ここに記述するコミット順要件に合致すると直ちにコミットすることができなければならない。この処理は終点３０２において終了する。

【００３１】

図６は、コミットアービトレータを使用して複数のトランザクションの部分的順序付けを実施することを含むステージの１つの実施を示す。１つの形として、図６の処理は計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。このプロシージャは出発点３１０において始まり、複数のトランザクションについて所定の部分的順序付けを指定すること（例えば、複数のトランザクションがコミットすべき複数の容認できる順序を（例えば、有向グラフの形で）指定すること）を許す作動をすることのできる１つまたは複数のコミットアービトレータを提供すること（ステージ３１２）を伴う。複数のトランザクションのうちの特定のトランザクションがそのコミット時点に達するとき、そのコミット順を実施するため、先行トランザクションの状態（例えば、１つまたは複数の順序付け値）を、（例えば、そのコミットアービトレータが追跡する通りの）特定のコミットしているトランザクションに関して調べる（ステージ３１４）。その特定のトランザクションに対する先行トランザクション全部がコミットし終えているときは（判定点３１６）、その特定のトランザクションはコミットされる（ステージ３１８）。コミットに成功すると、コミットアービトレータが追跡する１つまたは複数の値を必要に応じて更新し、もしあれば、可能な次の後継者全部を解除（活動開始）させる（３１８）。

【００３２】

その特定のトランザクションに対する先行トランザクション全部がコミットし終えていないときには（判定点３１６)、その特定のトランザクションを、先行トランザクションがコミットした後のちょうど良い時点で解除されるまで保留／休眠モードに置く（ステージ３２０）。処理は、終点３２２において終了する。

【００３３】

図７は、その所定のコミット順の情報を使用して競合を管理する競合管理処理を提供することを含むステージを示す１つの実施を示す。１つの形として、図７の処理は計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。このプロシージャは、出発点３４０において始まり、トランザクションの１つまたは複数のグループについての所定のコミット順をサポートするトランザクショナルメモリシステムを提供すること（ステージ３４２）を伴う。第１トランザクションと第２トランザクションとの間に競合を生じたときに呼び出される競合管理処理を提供する（ステージ３４４）。その競合管理処理においてその所定のコミット順を使用して、第１トランザクションまたは第２トランザクションのうちの何れが競合に勝って先に進むことを許されるべきかを決定するのを助ける（ステージ３４６）。第１トランザクションと第２トランザクションとが同一

トランザクショングループの一部でないときには（判定点３４８）、これら２つのトランザクションの間で、所定のコミット順は（存在しないので）実施しない（ステージ３５０）。このような筋書きにおいて、その２つのトランザクションは同一トランザクショングループの中にないので、競合を解決することに順序付け要因は使用されない（ステージ３５０）。

【００３４】

第１トランザクションと第２トランザクションとが同一トランザクショングループの一部であるときには（判定点３４８）、このシステムは、第１トランザクションの第１順序番号と第２トランザクションの第２順序番号とを比較する（ステージ３５２）。より低い順序番号（または別の適切な優先順序付け）を有するトランザクションが先へ進むことを許される（ステージ３５４）。処理は、終点３５６において終了する。

【００３５】

図８は、上記所定のコミット順の情報を使用して入れ子トランザクションとの競合を管理する競合管理処理を提供することを含むステージの１つの実施である。１つの形として、図８の処理は計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。１つの実施においては、特定のトランザクションをコミットする前に、トランザクション毎に、系図の全体(entire ancestor chain)を考慮に入れるので、その系図の中に順序付けがあればそれを実施する。このプロシージャは、出発点３７０において始まり、第１トランザクションと第２トランザクションとの間に競合が生じたときに呼び出される競合管理処理を提供すること（ステージ３７２）を伴う。その競合管理処理において所定のコミット順を使用して、第１トランザクションまたは第２トランザクションのうちの何れがその競合に勝って先に進むことを許されるかを決定するのを助ける（ステージ３７４）。第１トランザクションと第２トランザクションとが同一トランザクショングループの一部でないときには（判定点３７６）、これら２つのトランザクションの間で、所定のコミット順は（存在しないので）実施せず（ステージ３７８）、処理は終点３８８において終了する。その第１トランザクションと第２トランザクションとが同一トランザクショングループの一部であるときには（判定点３７６）、このシステムは、入れ子トランザクションを含むか否かを点検する（判定点３８０）。

【００３６】

入れ子トランザクションを含まないときには（判定点３８０）、第１トランザクションの順序番号（または他の順序付け表示）と第２トランザクションの順序番号（または他の順序付け表示）とを比較する（ステージ３８４）。より低い順序番号を有する（または他の適切な判断基準を用いることにより、順序では次であると判定された）トランザクションが先に進むことを許される（ステージ３８６）。

【００３７】

入れ子トランザクションを含むときには（判定点３８０）、第１トランザクションの最上位祖先の順序番号（または他の順序付け表示）と第２トランザクションの最上位祖先の順序番号（または他の順序付け表示）とを比較する（ステージ３８２）。用語「最上位祖先」は、ここで使用するとき、共通祖先が関与する場合は共通祖先の直接の子を含み、共通祖先が関与しない場合は各トランザクションの最高順位の祖先を意味する。共通および共通でない祖先が関与するこれらの筋書きをさらに詳しく図９および図１０に示す。より低い順序番号を有するトランザクション（例えば、より低い順序番号または適切な他の判断基準を有していた祖先に関係するトランザクション）が先に進むことを許される（ステージ３８６）。処理は終点３８８において終了する。

【００３８】

図９は、共通祖先を有する系図の一例を、最高順位の祖先と共に示すロジック図（論理図）である。図示の事例において、トランザクションＡはＤおよびＥの共通祖先である。ＤとＥとの間に生じる競合において、トランザクションＢとＣ（共通祖先Ａの直接の子）の順序番号を解析して、ＤまたはＥの何れが先に進むことを許されるべきかを決定する（図８のステージ３８２）。

【００３９】
図１０は、共通祖先を有しない系図の一例を最高順位の祖先と共に示すロジック図（論理図）である。図示の事例において、トランザクションＡはトランザクションＣの祖先である。トランザクションＤはトランザクションＦの祖先である。ＣとＦとの間に生じる競合において、このときはトランザクションＡとＤ（それぞれの最高順位の祖先）の順序番号を比較してトランザクションＣまたはＦの何れが先に進むことを許されるべきかを決定する（図８のステージ３８２）。
【００４０】
図１１は、トランザクショナルメモリシステム内のコミットアービトレータを使用することにより、無駄な作業の量を減少することを含むステージの１つの実施を示す。１つの形として、図１１の処理は計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。このプロシージャは、出発点４００において始まり、トランザクショナルメモリシステム用の、複数のトランザクションについて所定のコミット順を指定させる１つまたは複数のコミットアービトレータを提供すること（ステージ４０２）を伴う。そのコミットアービトレータは、トランザクションを保留／休眠モードに置いて、（例えば、所定のコミット順を解析して正しい順序を判定することにより）先行トランザクションが未だ実行中であるとき、そのトランザクションが再度実行されることを阻止するよう動作できる（ステージ４０４）。そのコミットアービトレータは、（例えば、再度所定のコミット順を解析して正しい順序を判定することにより）先行者が完了したとき、保留に置かれたトランザクションを解除するよう動作することもできる（ステージ４０６）。これら阻止と解除の機構を備えることにより、そのコミットアービトレータは、後に取り消さなければならない動作の実行を阻止して、それにより無駄になる作業の量を軽減するのを助ける（ステージ４０８）。この処理は終点４１０において終了する。
【００４１】
図１２は、競合管理処理において系図の全体を解析して適切な競合解消策を決定することを含むステージを示す。１つの形として、図１２の処理は計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。このプロシージャは、出発点４３０において始まり、第１トランザクションと第２トランザクションとの間に競合が生じるときに呼び出される競合管理処理を提供すること（ステージ４３２）を伴う。その競合管理処理において所定のコミット順を使用して、第１トランザクションまたは第２トランザクションのうちの何れがその競合に勝って先に進むことを許されるかを決定するのを助ける（ステージ４３４）。所定のコミット順の系図の全体を解析して、適切な競合管理を決定するのを助ける（ステージ４３６）。例えば、ＢはＡの中にネスト化（入れ子に）されており、ＤはＣの中にネスト化されているという、４つのトランザクション、２つの親と２つの子、があるとする。ＡはＣの前にコミットしなければならないというＡとＣとの間の順序付け関係があると仮定する。ＢとＤとの間に競合が生じると、競合管理処理はＢを支持するはずである。ＡはＣの前にコミットしなければならないことを考えると、Ｄを支持することは無駄だからである（ステージ４３６）。この処理は終点４３８において終了する。
【００４２】
ここで、図１に対する参照を続けると共に、図１３を参照すると、計算装置１００の上で働く並列ループサポートを有するトランザクショナルメモリアプリケーション５００が示されている。１つの実施において、並列ループサポートを有するトランザクショナルメモリアプリケーション５００は、計算装置１００の上に存在するアプリケーションプログラムのうちの１つである。しかしながら、並列ループサポートを有するトランザクショナルメモリアプリケーション５００は、代わりにまたはさらに１つまたは複数のコンピュータの上および／または図１に示すものとは異なる変換におけるコンピュータ実行可能命令として具体化されることは理解されるであろう。代わりにまたはさらに、並列ループサポートを有するトランザクショナルメモリアプリケーション５００の１つまたは複数の部分は、他のコンピュータおよび／またはアプリケーション、若しくはコンピュータソフトウ

ェア技術の当事者の心に浮かぶ通りの他の変換の上で、システムメモリ１０４の一部分とすることができる。

【００４３】

並列ループサポートを有するトランザクショナルメモリアプリケーション５００は、ここに記述する手法の幾つかまたはすべての実行を担当するプログラムロジック５０４を含む。プログラムロジック５０４は、トランザクショナルメモリシステムを提供するロジック５０６、当初のシーケンシャルループを含むコードの第１セクションを、トランザクションを使用して当初の入力対出力マッピングを維持し安全性を向上させる並列ループを含むコードの第２セクションに変換するロジック５０８、当初のシーケンシャルループの１つまたは複数の繰り返しを、並列ループ内のトランザクションのうちの個々の１つの中に置くロジック５１０、当初のシーケンシャルループの実行順と整合性が取れた所定のコミット順を用いてトランザクションをコミットすることにより、当初の入力対出力マッピングを維持するロジック５１２、当初のシーケンシャルループがデータを修正する動作を含むとき、コミットアービトレータを使用して並列ループにおける競合を検知し対処するロジック５１４、当初のシーケンシャルループのコンパイラ解析を実施することなくコードの上記第２セクションを作成するロジック５１５、（ヒューリスティックス（経験則）、コードの第１セクションにおけるユーザ定義の注釈、などを用いて）上記当初のシーケンシャルループは並べ替えを受け付けないと判断するときは当初のシーケンシャルループの実行順序に左右されない順序でトランザクションがコミットするのを許可する方法でコードのその第２セクションを作成するロジック５１６、少なくとも上記トランザクションのうち一部は並行して実行されるように、コードの上記第２セクションを生成するロジック５１７、異なるスレッド上で実行されているトランザクションのうち少なくともいくつかと共に、上記トランザクショナルメモリシステムを使用してコードの第２セクションを実行するロジック５１８、および当該アプリケーションをオペレーションするための他のロジック５２０を含む。１つの実施において、プログラムロジック５０４は、プログラムロジック５０４におけるプロシージャに対する単一呼出を用いるなど、別のコンピュータからプログラム的に呼び出されるよう動作することができる。

【００４４】

ここで、当初のシーケンシャルループを並列ループに変換することを含むハイレベルステージの１つの実施を示す図１４を参照する。１つの形として、図１４の処理は、計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。このプロシージャは、出発点５５０において始まり、シーケンシャルループの繰り返しの各々（または繰り返しの連続ストリップ）をもとに（例えば、それぞれの作業項目を含む）個々のトランザクションを作成することにより、当初のシーケンシャルループを、所定のコミット順にしたがって当初のシーケンシャルループの当初の実行と整合性が取れたコミット順を遵守する並列ループに変換すること（ステージ５５２）を伴う。別の実施においては、繰り返し毎に１つのトランザクションを作成することは余りに犠牲が大きい場合に、繰り返しの連続ストリップ（例えば、隣接するもの）を１つのトランザクションにまとめてグループ化することができる（ステージ５５２）。このシステムは、当初のシーケンシャルループのコンパイラ解析を実行することなく並列ループを生成する（ステージ５５４）。次いで、異なるスレッドに割り当てられている上記個別のトランザクションの少なくともいくつかを用いて、それらが並列に実行するように、上記並列ループを実行する（ステージ５５６）。この処理は終点５５８において終了する。

【００４５】

図１５は、所定のコミット順処理を使用して、並列ループ内のトランザクションは適切な順序でコミットされることを確認することを含むステージの１つの実施を示す。１つの形として、図１５の処理は、計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。この処理は、出発点５７０において始まり、当初のシーケンシャルループを、所定のコミット順処理にしたがう並列ループに変換すること（ステージ５７２）を伴う。このシステムは、並列ループ内のトランザクション毎にコミット順番号を

割り当てる（または、そのトランザクションをコミットする順序を追跡する別の適切な方法を用いる）（ステージ５７４）。並列ループを実行しているとき、このシステムは、その所定のコミット順処理を使用して、それぞれのトランザクションを、並列ループの先行の繰り返し（iteration）が首尾良くコミットした後でのみ、完了することができることを確実にする（例えば、そのトランザクションのコミット順が、そのトランザクションはコミットすることができると示すまで、そのトランザクションを待たせる）(ステージ５７６)。この処理は終点５７８において終了する。

【００４６】

図１６は、コミットアービトレータを使用して、並列ループが実行している間に生じる衝突を検知して対処することを含むステージの１つの実施を示す。１つの形として、図１６の処理は、計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。この処理は、出発点６００において始まり、当初のシーケンシャルループを、所定のコミット順処理を用いて正しい順序付けを確実にする並列ループに、変換すること（ステージ６０２）を伴う。このシステムは、その並列ループを実行する（ステージ６０４）。このシステムは次いで、同一データ要素を（例えば、スレッド安全性が欠けているとの理由で、順序付け要件の理由で、などで）修正しようとする個々のトランザクションの１つ以上（例えば、ループ繰り返し）を、上記並列ループが含むことを検知する（ステージ６０６）。コミットアービトレータを使用して、順番通りでない実行を検知し、先行トランザクションが完了次第に後続トランザクションの再実行の手続を取るなどにより、上記並列ループを実行する間に生じる衝突を検知して処理する（ステージ６０８）。この処理は終点６１０において終了する。

【００４７】

図１７は、コミットアービトレータを使用して、並列ループを実行している間に生じる処理されない例外を検知して対処することを含むステージの１つの実施を示す。１つの形として、図１７の処理は、計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。この処理は、出発点６３０において始まり、当初のシーケンシャルループを、トランザクションを用いて当初の入力対出力マッピングを維持し安全性を向上させる並列ループに変換すること（ステージ６３２）を伴う。このシステムはその並列ループを実行して（ステージ６３４）、並列ループを実行している間に特定のトランザクションに生じる処理されない例外を検知する（ステージ６３６）。その特定のトランザクションおよびその特定のトランザクションの先行トランザクションがあればそれにより行われた状態変更をコミットする（ステージ６３８）。後続トランザクションの何れかによりその特定のトランザクションに対し推論的に行われた状態変更を、それらのトランザクションをロールバック（後退）させることにより廃棄する（ステージ６４０）。処理は終点６４２において終了する。

【００４８】

図１８Ａ～図１８Ｂは、当初のシーケンシャルループから並列ループへの模範的な変換についての仮想のソースコードを示す。図１８Ａはｆｏｒ・・・ｅａｃｈループ６５２を含む当初のシーケンシャルループ６５０を示すが、別の形式のループ構造も使用することができることは理解されるであろう。ループにおける繰り返し毎に１つまたは複数のステートメント６５４が実行される。図１８Ｂは、ここに説明する手法のうち一部を使用して並列ループ６６０に変換した後、シーケンシャルループが呈する様相の仮想の例を示す。図示の事例においては、当初のシーケンシャルループ６６４の繰り返し毎に個々のトランザクションを創出することにより、並列ループを生成する。別の実施においては、繰り返し毎に１つのトランザクションを作ることは犠牲が多過ぎると思われる場合に、繰り返しの連続ストリップ（例えば、隣接するもの）をトランザクションの中にまとめてグループにすることができる。このとき個々のトランザクションの各々は、当初のループにおいてステートメント６６７として含まれていた作業を実行するため、新しい作業項目を作成する。個々のクラス６６２を使用して作業項目繰り返しを宣言することができる。その個々のトランザクションは次いで、異なるスレッドに割り当てられるのでそれらは並行して実

行することができる。

【００４９】

図１９は、クローズドエンドシーケンシャルループを並列ループに変換することを含むステージの１つの実施を示す。１つの形として、図１９の処理は、計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。この処理は、出発点６７０において始まり、トランザクショナルメモリシステムを提供すること（ステージ６７２）を伴う。このシステムは、当初のシーケンシャルループを含むコードの第１セクションを解析して、当初のシーケンシャルループが実行するだろう繰り返し定数を（例えば、ループ終了を判断するために使用する定数を検索することにより）決定する（ステージ６７４）。当初のシーケンシャルループを含むコードの第１セクションを、繰り返しの定数までトランザクションを生成することのできる並列ループを含むコードの第２セクションに変換する（ステージ６７６）。このシステムは、異なるスレッドに割り当てられているトランザクションのうち少なくともいくつかと共に、それらが並行して実行するように、上記トランザクショナルメモリシステムを使用してコードの第２セクションを実行する（ステージ６７８）。このシステムは、所定のコミット順処理を用いて、そのトランザクションを正しい順序でコミットする（例えば、各トランザクションはそれぞれの誘導変数カウンタをコミット連続番号として使用するところで）（ステージ６８０）。この処理は終点６８２において終了する。

【００５０】

１つの実施において、図１９に記述する変換処理は、誘導変数をループ本体自体には決して書き込まないループのためにのみ使用する。言い換えると、ループは、そのループ本体における誘導変数に書き込むことにより、若しくは誘導変数のアドレスを採用しそれを用いて書き込みに導く何か（ファンクションに渡す、それを擬似する、など）をすることにより、不適格となることがある。

【００５１】

図２０は、スペキュレーションパイプラインを使用して、オープンエンドシーケンシャルループを並列ループに変換することを含むステージの１つの実施を示す。１つの形として、図２０の処理は、計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。この処理は、出発点７００において始まり、トランザクショナルメモリシステムを提供すること（ステージ７０２）を伴う。このシステムは、当初のオープンエンドシーケンシャルループを含むコードの第１セクションを、（例えば、少なくともいくつかのトランザクションが並行して実行するように）、スペキュレーションパイプラインの繰り返し毎の作業項目それぞれを含む個々のトランザクションを生成する動作をすることのできる並列ループを含むコードの第２セクションに変換する（ステージ７０４）。コードのその第２セクションは、そのオープンエンドシーケンシャルループのコンパイラ解析を実行することなく、生成される（ステージ７０６）。このシステムは、異なるスレッドに割り当てられているトランザクションのうち少なくともいくつかと共に、それらが並行して実行するように、上記トランザクショナルメモリシステムを使用してコードの第２セクションを実行する（ステージ７０８）。そのトランザクションを（例えば、オープンエンドシーケンシャルループの実行順と整合性がとれた）所定のコミット順でコミットすることにより、当初の入力対出力マッピングは維持される（ステージ７１０）。この処理は、終点７１２において終了する。

【００５２】

図２１は、オープンエンドシーケンシャルループから生成された並列ループを実行することを含むステージの１つの実施を示す。１つの形として、図２１の処理は、計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。この処理は、出発点７３０において始まり、オープンエンドシーケンシャルループ（例えば、ｗｈｉｌｅループ、ｄｏ　ｗｈｉｌｅループ、ｆｏｒループ、など）から生成される並列ループにおいて実行すべき繰り返しの回数を見積もるスペキュレーションパイプラインを生成すること（ステージ７３２）を伴う。１つの実施において、このシステムは、そのスペキュレ

ーションパイプラインの各繰り返しをもとに、それぞれの作業項目を含む個々のトランザクションを生成する（ステージ７３４）。別の実施例において、繰り返し毎に１つのトランザクションを作ることは犠牲が多過ぎると思われる場合などに、このシステムは、繰り返しの連続ストリップ（例えば、隣接するもの）をもとに、それらをトランザクションの中にまとめてグループ化する（ステージ７３４）。このシステムは、上記個々のトランザクションを、それらが並行して実行するように、異なるスレッドに割り当てる（７３５）。このシステムは、それぞれの作業項目毎に終了条件を評価する（ステージ７３６）。そのそれぞれの作業項目のうちの特定の１つが、上記並列ループを終了する時期に到達したと判定するときには、先行者をコミットして、後継者を廃棄する（ステージ７３８）。この処理は終点７４０において終了する。

【００５３】

図２２は、オープンエンドシーケンシャルループから生成された並列ループ内の各作業項目を適切な順序でコミットさせることを確実にすることを含むステージの１つの実施を示す。１つの形として、図２２の処理は、計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。この処理は、出発点７６０において始まり、それぞれのトランザクションにおけるそれぞれの作業項目各々を実行しながら、現在の繰り返し値を取り出すこと（ステージ７６２）を伴う。１つの実施において、現在の繰り返し値は、それぞれの作業項目各々がアクセスすることのできる値を、極小の増分（atomic increment）を実行することにより取り出す（ステージ７６２）。このシステムは、それぞれの作業項目各々の現在の繰り返し値を、所定のコミット順処理におけるコミット連続番号として使用する（ステージ７６４）。このシステムは、そのオープンエンドシーケンシャルループの当初の実行と整合性が取れたコミット順を達成する（ステージ７６６）。この処理は終点７６８において終了する。

【００５４】

図２３は、スペキュレーションパイプラインを計算してその並列ループの中に幾つの繰り返しを含むかを決定することを含むステージの１つの実施を示す。１つの形として、図２３の処理は、計算装置１００のオペレーティングロジックにおいて少なくとも部分的に実行される。この処理は、出発点７９０において始まり、このシステムが、並列ループを実行するコンピュータ上の利用可能な処理装置の数に少なくとも基づいて、スペキュレーションパイプラインの初期値を作成すること（ステージ７９２）を伴う。１つの実施において、ＣＰＵ束縛作業を行うのに費やす作業負荷を時間率で割った処理装置の数に基づいて、スペキュレーションパイプラインの初期値を生成する（ステージ７９２）。幾多の他の計算も利用可能である。その初期値を用いて、その並列ループの特定の実行のために作成される、上記並列ループの繰り返し数が幾つであるかを決定する（ステージ７９４）。このシステムは、適応的統計データを用いて、その並列ループの以後の実行のためにそのスペキュレーションパイプラインを調整することができる（例えば、経歴を用いてそのループの予想所要時間をさらに良く決定する、作業項目がブロックするとき柔軟に調整する、などで）（ステージ７９６）。この処理は終点７９８において終了する。

【００５５】

図２４Ａ～図２４Ｂは、当初のオープンエンドシーケンシャルループから並列ループへの模範的な変換のための仮想のソースコードを示す。用語「オープンエンドシーケンシャルループ」は、ここで使用するとき、その繰り返し数が未知であるシーケンシャルループを含むことを意味する。図２４Ａを参照すると、当初のオープンエンドシーケンシャルループ８１０が示されている。このループは、条件が真である間（例えば、図示の事例ではＰ＝ｔｒｕｅである間）一定のステートメントを実行するｗｈｉｌｅループである。図２４Ｂは、その当初のシーケンシャルループが並列ループ８２０に変換された後の様相を示す。図２４Ｂの仮想のコードに示すように、スペキュレーションパイプラインの繰り返し毎に、並行して実行する作業項目が生成される。１つの実施においては、このため標準ワークスチーリングキュー（standard work stealing queue）を使用することができる。ｃｕｒｒｅｎｔＩｔｅｒａｔｉｏｎと呼ばれる共有変数は、各作業項目にアクセスすること

ができる。各作業項目が実行するにつれ、標準ｃｏｍｐａｒｅ－ａｎｄ－ｓｗａｐハードウェア命令または別の機構のように、ｃｕｒｒｅｎｔＩｔｅｒａｔｉｏｎに極小の増分を実行して、それ自体の繰り返し値をフェッチ(fetch)する。これは、１つの繰り返しは何れも単一の作業者（worker）により処理されること、およびトランザクションがループの繰り返しのうち１つを実行し始める順序を決定することを保証する。これは次いで、トランザクションのコミット連続番号となり、繰り返しは先行者と後継者の間で正しい順序で直列化可能であることを保証する。各作業項目は、そのループ構造が命令する通りに、Ｐまたは作業の前または後に終了条件を適用することができるものは何でも、評価する（例えば、図２４Ｂに示す「ｗｈｉｌｅ」の場合は前であるが、ｄｏ－ｗｈｉｌｅの場合は「後」）。作業者の１つが、終了する時期であることを認識すると、先行者全部はコミットし、次いで後継者全部を廃棄しなければならない。

【００５６】

ここに説明する例は、各種の技術および手法を用いてコミット順序付けを実施することについて述べたけれども、トランザクションはコミットアービトレータを全く有しないことに注意しなければならない。トランザクションがコミットアービトレータを全く有しないこのような場合は、通常の無秩序なコミットが生じる。

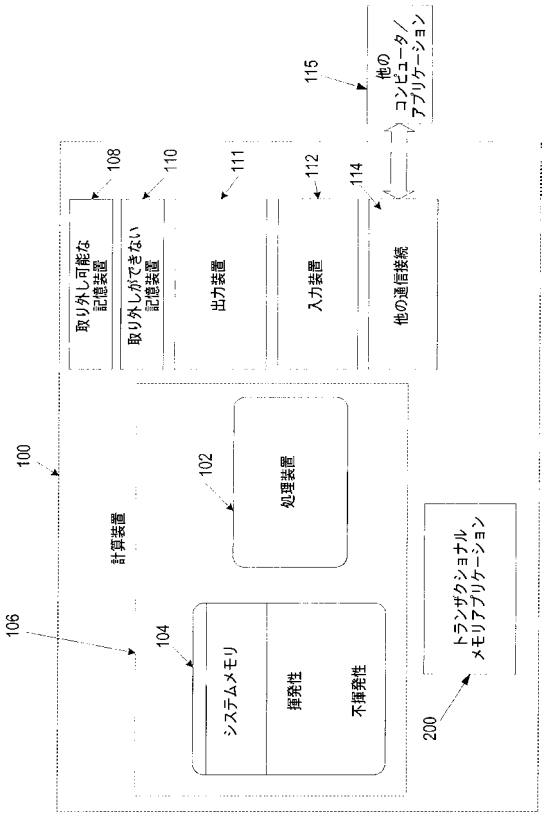
【００５７】

主題事項を、構造的特性および／または方法論的行動に特定の言語を用いて記述したけれども、添付の請求項に定義する主題事項は、上述の具体的な特性または行動に必ずしも限定されるものではないことは理解されなければならない。むしろ、上述の具体的な特性および行動は、請求項の実施の模範的な形態として開示されているものである。ここに記述する通りのおよび／または以下の請求項による実施の思想の範囲内に入る等価物（均等物）、変更および修正のすべては、保護されることが望まれる。

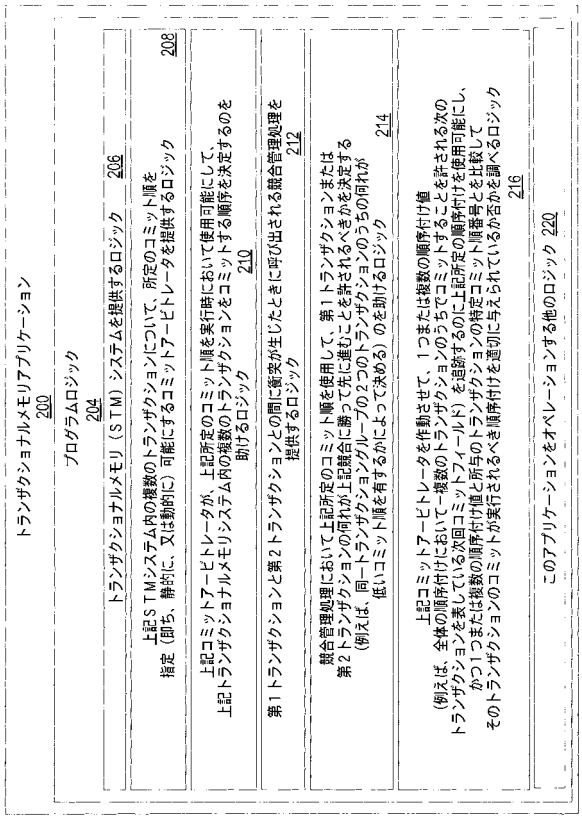
【００５８】

例えば、コンピュータソフトウェアの当業者は、ここに説明した例示において記述したような、クライアントおよび／またはサーバアレンジメント、ユーザインターフェイス画面コンテンツ、および／またはデータレイアウトは、１つまたは複数のコンピュータの上で異なって組織され、上記例示において描写したより少ないまたは多いオプションまたは機能を含むことができることを、認識するであろう。

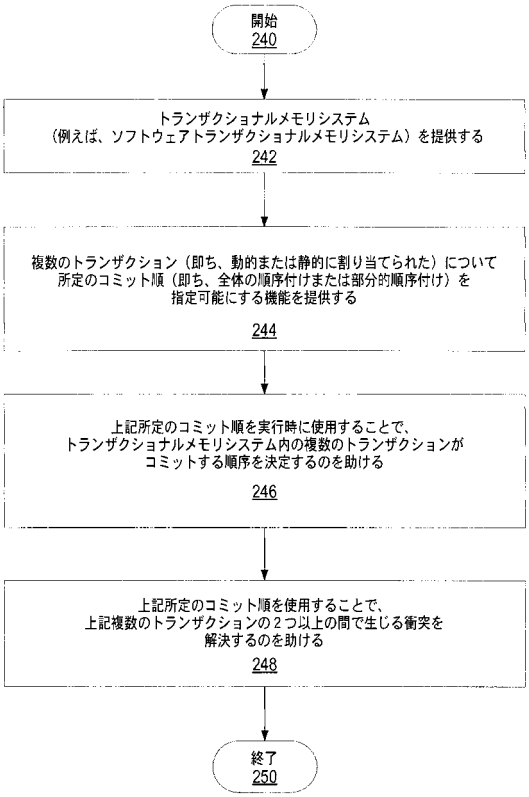
【図１】


他のコンピュータ／アプリケーション
115
取り外し可能な記憶装置
108
取り外しができない記憶装置
110
出力装置
111
入力装置
112
他の通信接続
114
100
計算装置
102
処理装置
106
104
システムメモリ
揮発性
不揮発性
トランザクショナルメモリアプリケーション
200

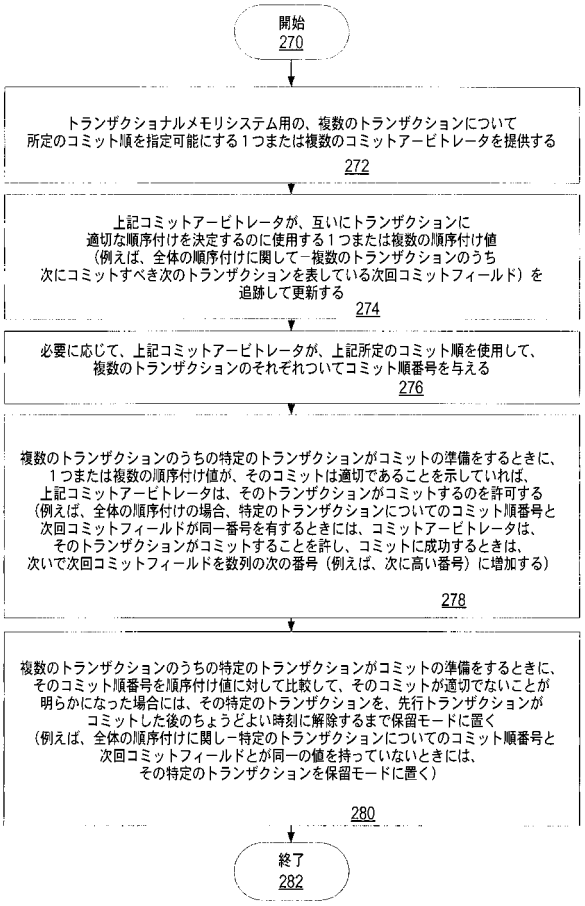
【図２】


トランザクショナルメモリアプリケーション
200
プログラムロジック
204
トランザクショナルメモリ（STM）システムを提供するロジック 206
上記STMシステム内の複数のトランザクションについて、所定のコミット順を指定（即ち、静的に、又は動的に）可能にするコミットアービトレータを提供するロジック 208
上記コミットアービトレータが、上記所定のコミット順を実行時において使用可能にして、上記トランザクショナルメモリシステム内の複数のトランザクションをコミットする順序を決定するのを助けるロジック 210
第１トランザクションと第２トランザクションとの間に衝突が生じたときに呼び出される競合管理処理を提供するロジック 212
競合管理処理において上記所定のコミット順を使用して、第１トランザクションまたは第２トランザクションの何れが上記競合に勝って先に進むことを許されるべきかを決定する（例えば、同一トランザクショングループの２つのトランザクションのうちの何れが低いコミット順を有するかによって決める）のを助けるロジック 214
上記コミットアービトレータを作動させて、１つまたは複数の順序付け値（例えば、全体の順序付けにおいて－複数のトランザクションのうちでコミットすることを許される次のトランザクションを表している次回コミットフィールド）を追跡するのに上記所定の順序付けを使用可能にし、かつ１つまたは複数の順序付け値と所与のトランザクションの特定コミット順番号とを比較してそのトランザクションのコミットが実行されるべき順序付けを適切に与えられているか否かを調べるロジック 216
このアプリケーションをオペレーションする他のロジック 220

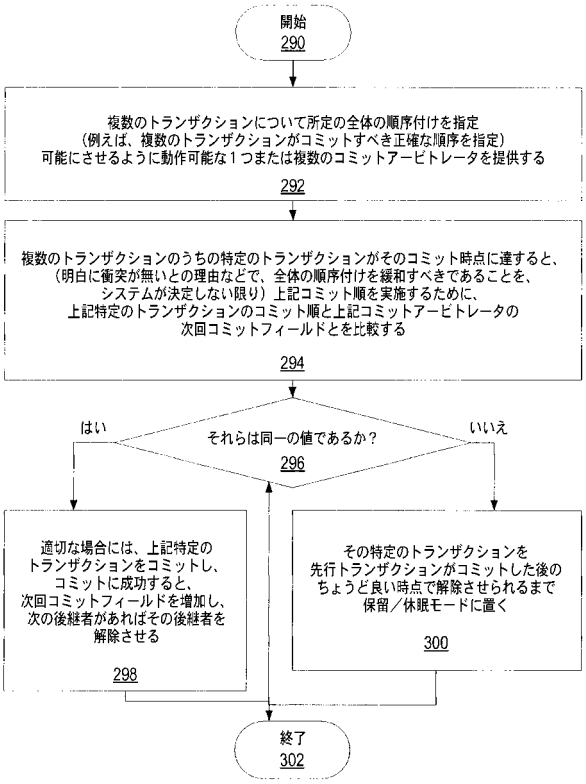
【図３】


開始
240
トランザクショナルメモリシステム（例えば、ソフトウェアトランザクショナルメモリシステム）を提供する
242
複数のトランザクション（即ち、動的または静的に割り当てられた）について所定のコミット順（即ち、全体の順序付けまたは部分的順序付け）を指定可能にする機能を提供する
244
上記所定のコミット順を実行時に使用することで、トランザクショナルメモリシステム内の複数のトランザクションがコミットする順序を決定するのを助ける
246
上記所定のコミット順を使用することで、上記複数のトランザクションの２つ以上の間で生じる衝突を解決するのを助ける
248
終了
250

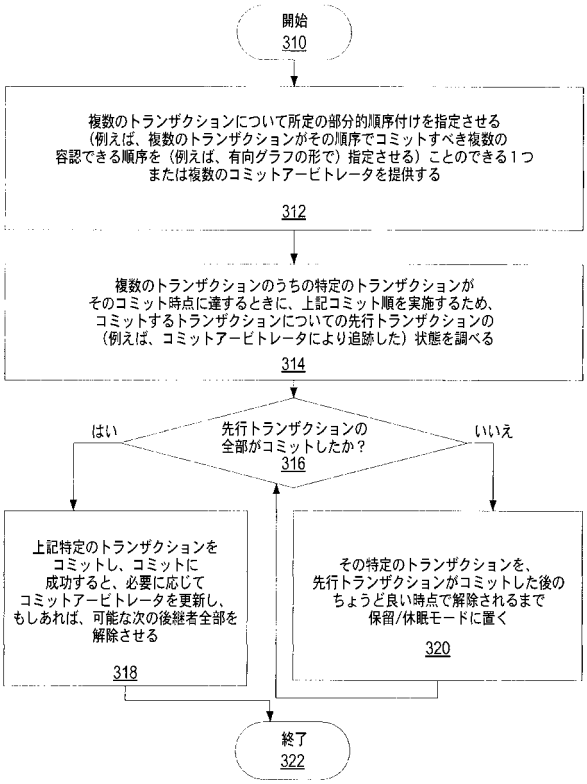
【図４】


開始
270
トランザクショナルメモリシステム用の、複数のトランザクションについて所定のコミット順を指定可能にする１つまたは複数のコミットアービトレータを提供する
272
上記コミットアービトレータが、互いにトランザクションに適切な順序付けを決定するのに使用する１つまたは複数の順序付け値（例えば、全体の順序付けに関して－複数のトランザクションのうち次にコミットすべき次のトランザクションを表している次回コミットフィールド）を追跡して更新する
274
必要に応じて、上記コミットアービトレータが、上記所定のコミット順を使用して、複数のトランザクションのそれぞれついてコミット順番号を与える
276
複数のトランザクションのうちの特定のトランザクションがコミットの準備をするときに、１つまたは複数の順序付け値が、そのコミットは適切であることを示していれば、上記コミットアービトレータは、そのトランザクションがコミットするのを許可する（例えば、全体の順序付けの場合、特定のトランザクションについてのコミット順番号と次回コミットフィールドが同一番号を有するときには、コミットアービトレータは、そのトランザクションがコミットすることを許し、コミットに成功するときは、次いで次回コミットフィールドを数列の次の番号（例えば、次に高い番号）に増加する）
278
複数のトランザクションのうちの特定のトランザクションがコミットの準備をするときに、そのコミット順番号を順序付け値に対して比較して、そのコミットが適切でないことが明らかになった場合には、その特定のトランザクションを、先行トランザクションがコミットした後のちょうどよい時刻に解除するまで保留モードに置く（例えば、全体の順序付けに関し－特定のトランザクションについてのコミット順番号と次回コミットフィールドとが同一の値を持っていないときには、その特定のトランザクションを保留モードに置く）
280
終了
282

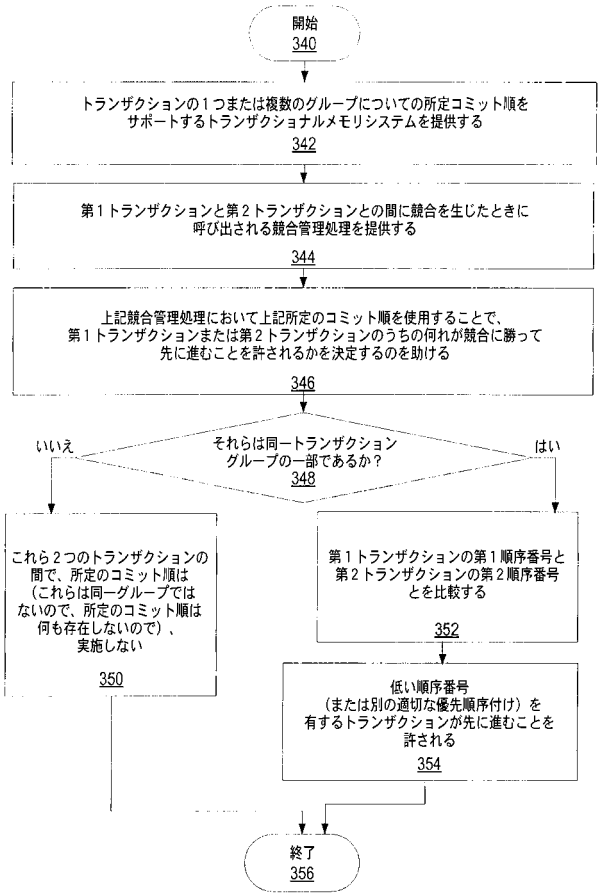
【図５】


開始
290
複数のトランザクションについて所定の全体の順序付けを指定
（例えば、複数のトランザクションがコミットすべき正確な順序を指定）
可能にさせるように動作可能な１つまたは複数のコミットアービトレータを提供する
292
複数のトランザクションのうちの特定のトランザクションがそのコミット時点に達すると、
（明白に衝突が無いとの理由などで、全体の順序付けを緩和すべきであることを、
システムが決定しない限り）上記コミット順を実施するために、
上記特定のトランザクションのコミット順と上記コミットアービトレータの
次回コミットフィールドとを比較する
294
はい
それらは同一の値であるか？
296
いいえ
適切な場合には、上記特定の
トランザクションをコミットし、
コミットに成功すると、
次回コミットフィールドを増加し、
次の後継者があればその後継者を
解除させる
298
その特定のトランザクションを
先行トランザクションがコミットした後の
ちょうど良い時点で解除させられるまで
保留／休眠モードに置く
300
終了
302

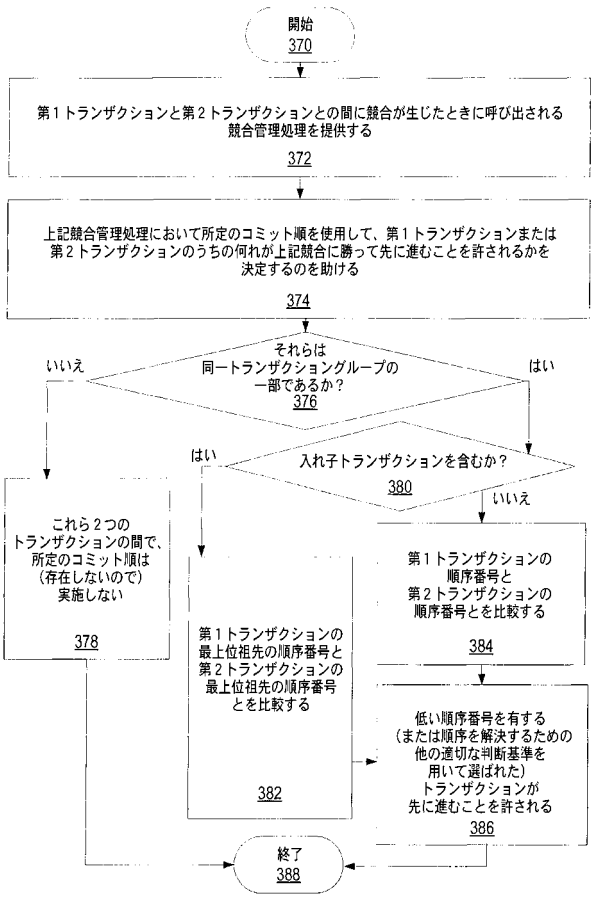
【図６】


開始
310
複数のトランザクションについて所定の部分的順序付けを指定させる
（例えば、複数のトランザクションがその順序でコミットすべき複数の
容認できる順序を（例えば、有向グラフの形で）指定させる）ことのできる１つ
または複数のコミットアービトレータを提供する
312
複数のトランザクションのうちの特定のトランザクションが
そのコミット時点に達するときに、上記コミット順を実施するため、
コミットするトランザクションについての先行トランザクションの
（例えば、コミットアービトレータにより追跡した）状態を調べる
314
はい
先行トランザクションの
全部がコミットしたか？
316
いいえ
上記特定のトランザクションを
コミットし、コミットに
成功すると、必要に応じて
コミットアービトレータを更新し、
もしあれば、可能な次の後継者全部を
解除させる
318
その特定のトランザクションを、
先行トランザクションがコミットした後の
ちょうど良い時点で解除されるまで
保留/休眠モードに置く
320
終了
322

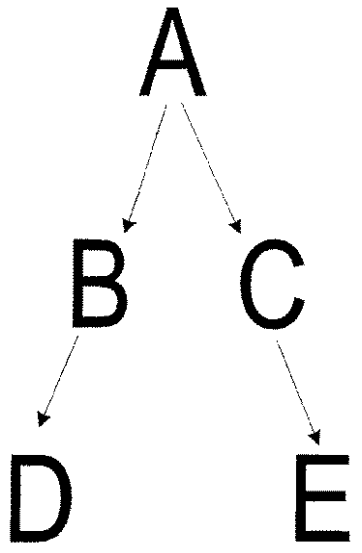
【図７】


開始
340
トランザクションの１つまたは複数のグループについての所定コミット順を
サポートするトランザクショナルメモリシステムを提供する
342
第１トランザクションと第２トランザクションとの間に競合を生じたときに
呼び出される競合管理処理を提供する
344
上記競合管理処理において上記所定のコミット順を使用することで、
第１トランザクションまたは第２トランザクションのうちの何れが競合に勝って
先に進むことを許されるかを決定するのを助ける
346
いいえ
それらは同一トランザクション
グループの一部であるか？
348
はい
これら２つのトランザクションの
間で、所定のコミット順は
（これらは同一グループでは
ないので、所定のコミット順は
何も存在しないので）、
実施しない
350
第１トランザクションの第１順序番号と
第２トランザクションの第２順序番号
とを比較する
352
低い順序番号
（または別の適切な優先順序付け）を
有するトランザクションが先に進むことを
許される
354
終了
356

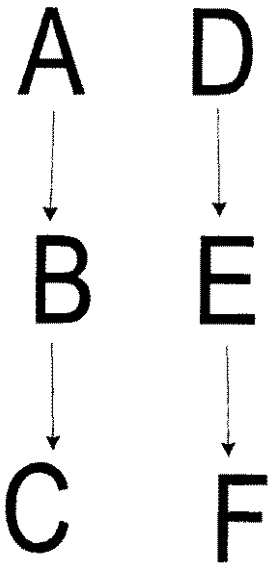
【図８】


開始
370
第１トランザクションと第２トランザクションとの間に競合が生じたときに呼び出される
競合管理処理を提供する
372
上記競合管理処理において所定のコミット順を使用して、第１トランザクションまたは
第２トランザクションのうちの何れが上記競合に勝って先に進むことを許されるかを
決定するのを助ける
374
いいえ
それらは
同一トランザクショングループの
一部であるか？
376
はい
はい
入れ子トランザクションを含むか？
380
いいえ
これら２つの
トランザクションの間で、
所定のコミット順は
（存在しないので）
実施しない
378
第１トランザクションの
最上位祖先の順序番号と
第２トランザクションの
最上位祖先の順序番号
とを比較する
382
第１トランザクションの
順序番号と
第２トランザクションの
順序番号とを比較する
384
低い順序番号を有する
（または順序を解決するための
他の適切な判断基準を
用いて選ばれた）
トランザクションが
先に進むことを許される
386
終了
388

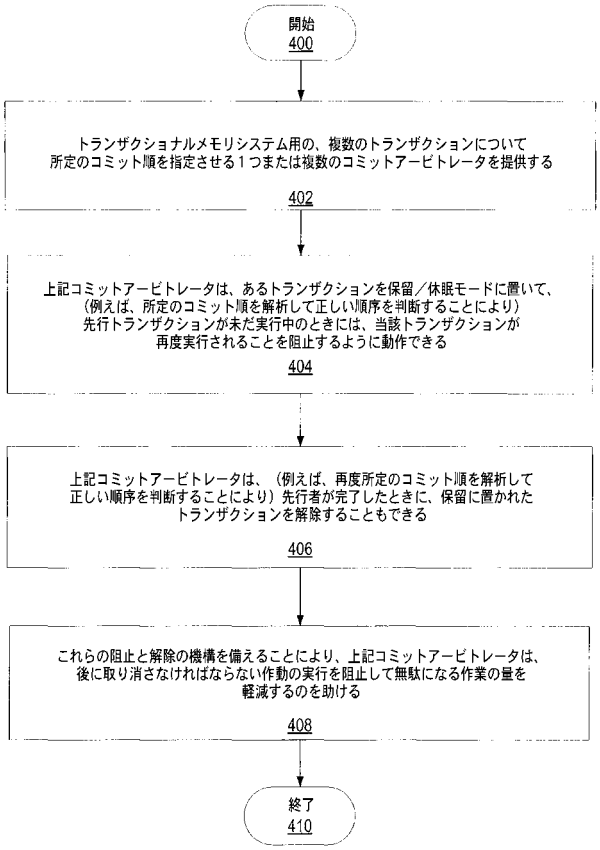
【図９】



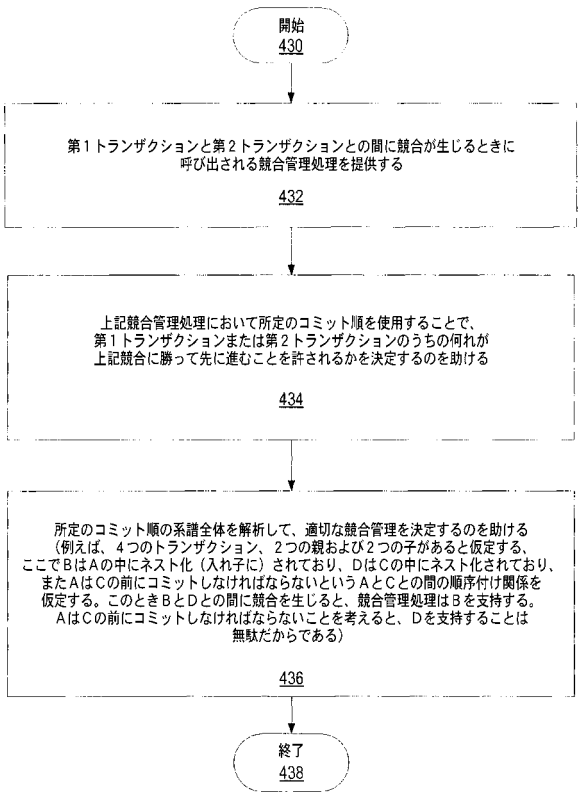
【図１０】



【図１１】

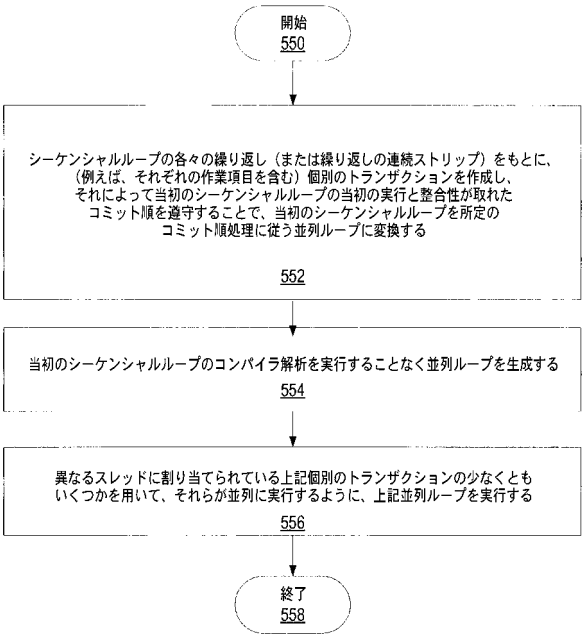


【図１２】

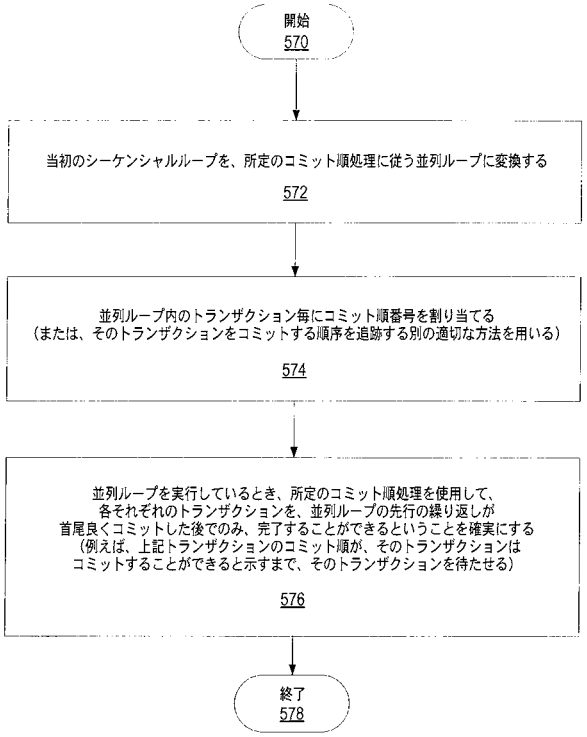


【図１３】

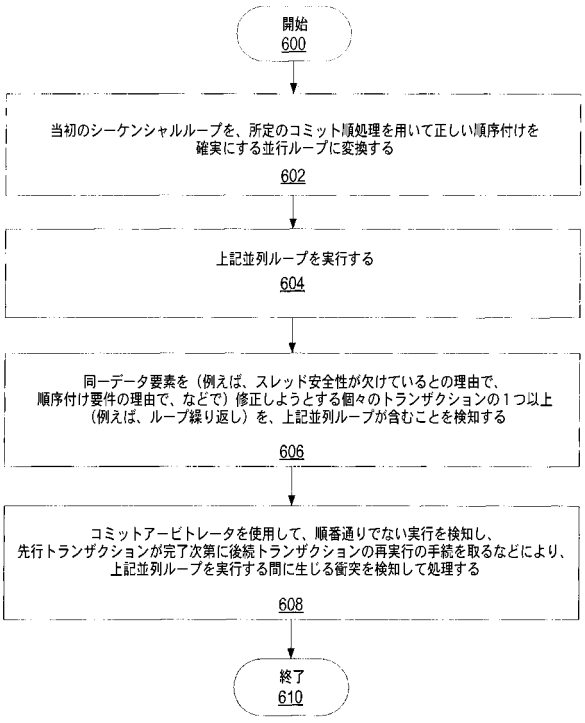
【図１４】



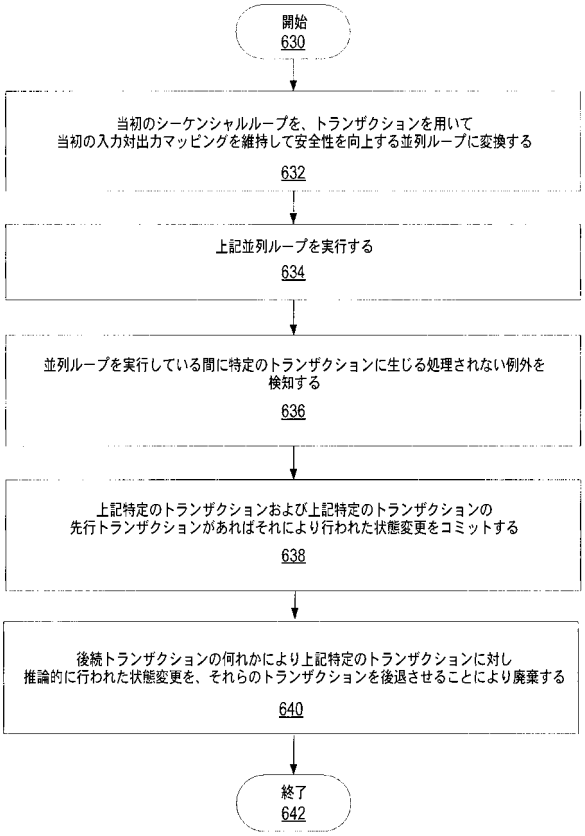
【図１５】



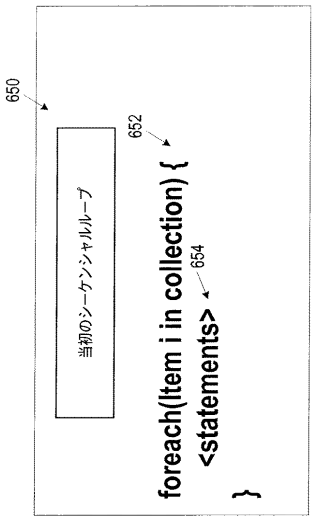
【図１６】



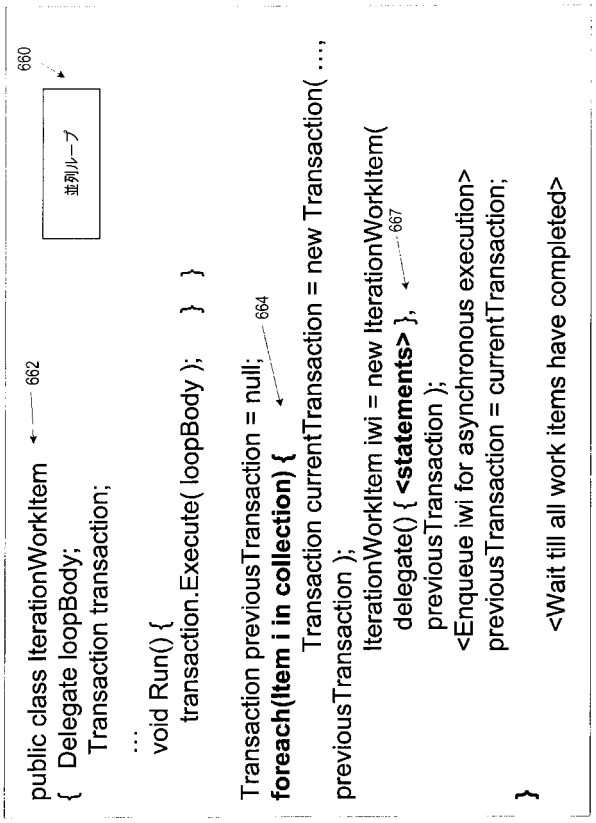
【図１７】



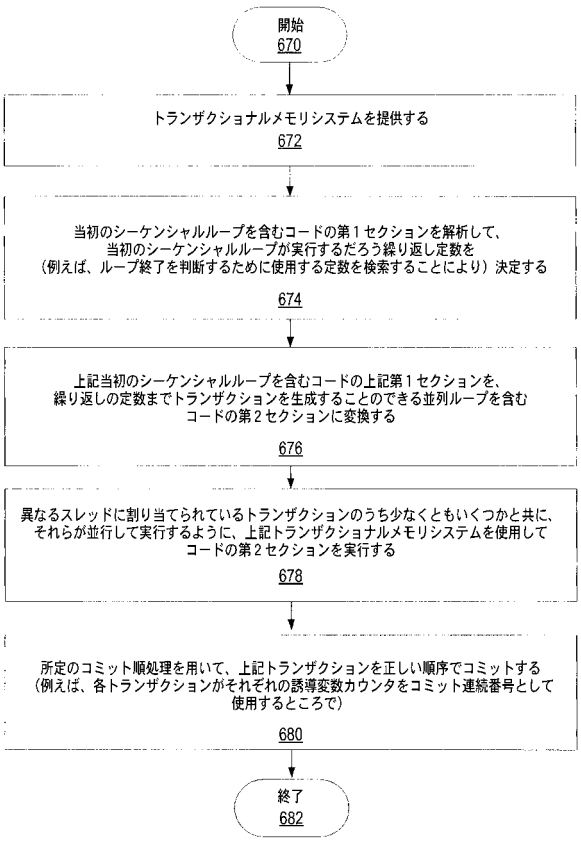
【図１８Ａ】



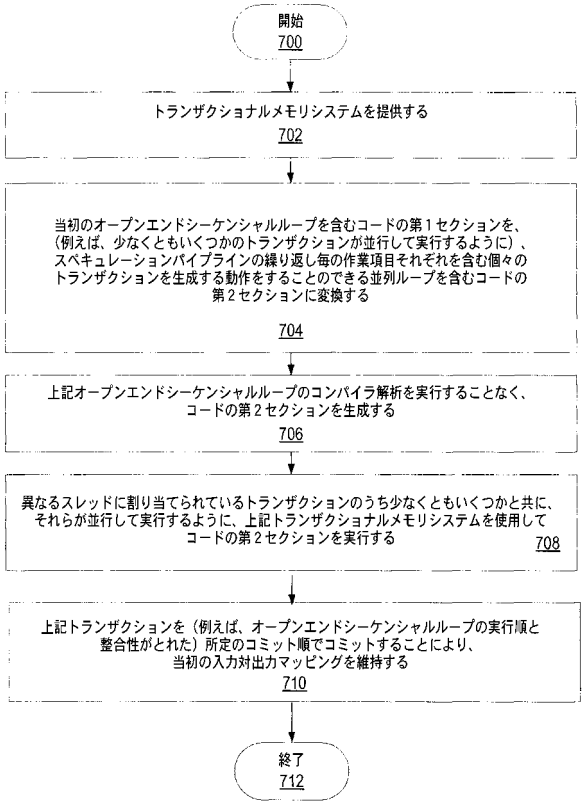
【図１８Ｂ】



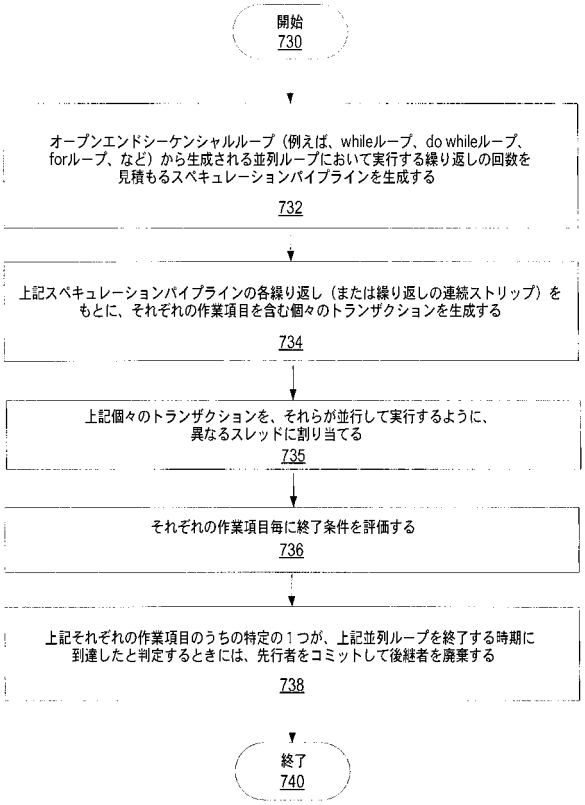
【図１９】



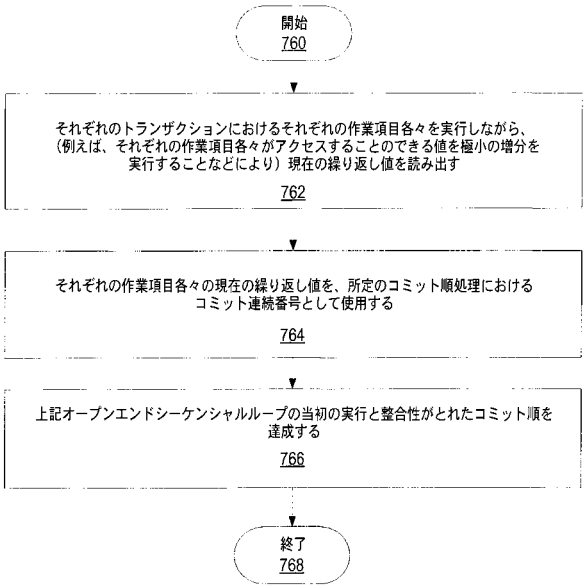
【図２０】



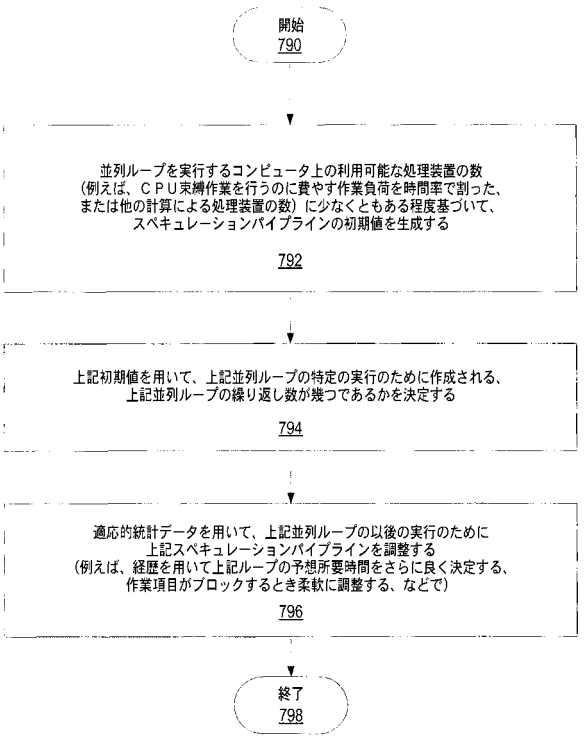
【図２１】



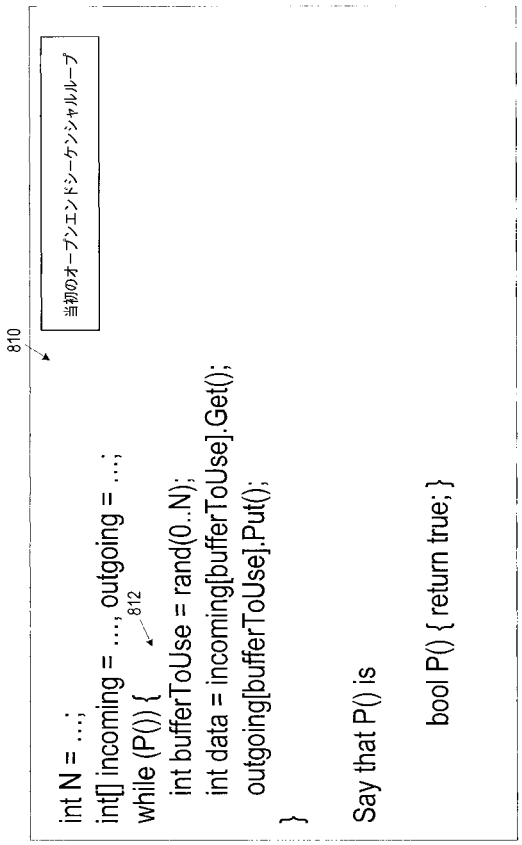
【図２２】



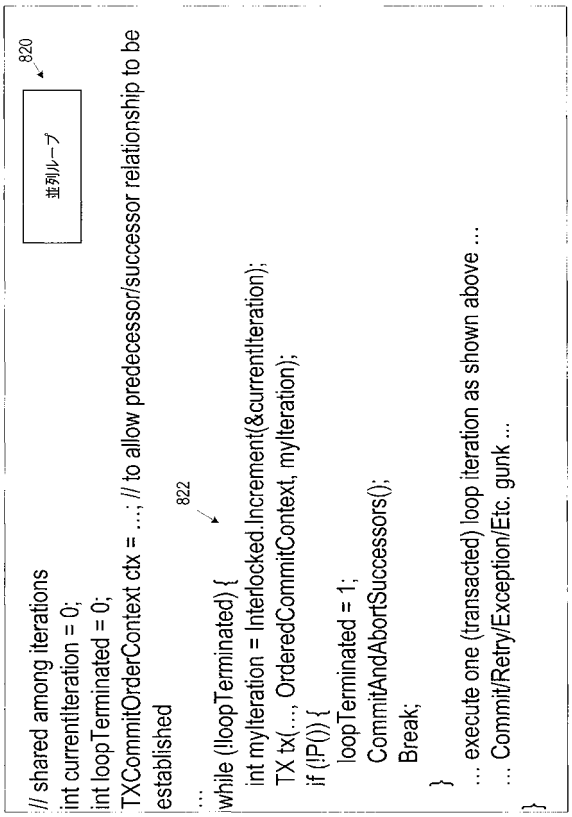
【図２３】



【図２４Ａ】


810
当初のオープンエンドシーケンシャルループ
int N = ...;
int[] incoming = ..., outgoing = ...;
812
while (P()) {
int bufferToUse = rand(0..N);
int data = incoming[bufferToUse].Get();
outgoing[bufferToUse].Put();
}
Say that P() is
bool P() { return true; }

【図２４Ｂ】


820
並列ループ
// shared among iterations
int currentIteration = 0;
int loopTerminated = 0;
TXCommitOrderContext ctx = ...; // to allow predecessor/successor relationship to be established
822
...
while (!loopTerminated) {
int myIteration = Interlocked.Increment(&currentIteration);
TX tx(..., OrderedCommitContext, myIteration);
if (!P()) {
loopTerminated = 1;
CommitAndAbortSuccessors();
Break;
}
... execute one (transacted) loop iteration as shown above ...
... Commit/Retry/Exception/Etc. gunk ...
}

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2008/065363

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/00(2006.01)i

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)
IPC 8 : G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean Utility models and applications for Utility models since 1975
Japanese Utility models and application for Utility models since 1975

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKIPASS(KIPO internal) "memory" "loop" "execute"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2004/0148150 A1 (PRANAV ASHAR et al.) 29 July 2004 See page 4, [0039] - page 17, [0211].	1-20
A	US 2005/0283769 A1 (ALEXANDRE E. EICHENBERGER et al.) 22 December 2005 See page 2, [0014] - page 10, [0174].	1-20
A	US 6014741 A (RUPAKA MAHALINGAIAH) 3 June 1997 See column 2, line 66 - column 32, line 54.	1-20
A	US 6016399 A (POHUA CHANG) 18 January 2000 See column 2, line 29 - column 13, line 24.	1-20

☐ 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
"E" earlier application or patent but 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 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	Date of mailing of the international search report
29 OCTOBER 2008 (29.10.2008)	**29 OCTOBER 2008 (29.10.2008)**

Name and mailing address of the ISA/KR
Korean Intellectual Property Office
Government Complex-Daejeon, 139 Seonsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea
Facsimile No. 82-42-472-7140

Authorized officer
KWON, Oh Seong
Telephone No. 82-42-481-8526

Form PCT/ISA/210 (second sheet) (July 2008)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2008/065363

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004-0148150 A1	29.07.2004	US 7383166 B	03.06.2008
US 2005-0283769 A1	22.12.2005	US 7367026 B US 7395531 B	29.04.2008 01.07.2008
US 6014741 A	11.01.2000	US 5898865 A	27.04.1999
US 6016399 A	18.01.2000	None	

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,HR,HU,IE,IS,IT,LT,LU,LV,MC,MT,NL,NO,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,AO,AT,AU,AZ,BA,BB,BG,BH,BR,BW,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IS,JP,KE,KG,KM,KN,KP,KR,KZ,LA,LC,LK,LR,LS,LT,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY,MZ,NA,NG,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RS,RU,SC,SD,SE,SG,SK,SL,SM,SV,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,ZA,ZM,ZW

(72)発明者 ジョン ジョセフ グレイ
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 インターナショナル パテンツ内
(72)発明者 ヨセフ レバノーニ
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 インターナショナル パテンツ内

Ｆターム(参考) 5B081 CC30 CC32