(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5276094号
(P5276094)

(45)発行日　平成25年8月28日(2013.8.28)　(24)登録日　平成25年5月24日(2013.5.24)

(51)Int.Cl.　　F I
G06F 9/46 (2006.01)　　G06F 9/46 430
G06F 9/52 (2006.01)　　G06F 9/46 472B

請求項の数 17　(全 11 頁)

(21)出願番号　特願2010-510519(P2010-510519)
(86)(22)出願日　平成20年5月30日(2008.5.30)
(65)公表番号　特表2010-529540(P2010-529540A)
(43)公表日　平成22年8月26日(2010.8.26)
(86)国際出願番号　PCT/US2008/065312
(87)国際公開番号　WO2008/151013
(87)国際公開日　平成20年12月11日(2008.12.11)
審査請求日　平成23年5月18日(2011.5.18)
(31)優先権主張番号　11/809,514
(32)優先日　平成19年6月1日(2007.6.1)
(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)復代理人　100115624
弁理士　濱中　淳宏
(74)復代理人　100156971
弁理士　稲　綾子
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫

最終頁に続く

(54)【発明の名称】トランザクション・メモリ・システムにおけるトランザクション・コード・ブロックを効果的に検索する方法

(57)【特許請求の範囲】

【請求項１】

ソフトウェア・トランザクショナル・メモリを実装するソフトウェア・トランザクショナル・メモリ・アプリケーションを有するコンピューターに、

前記ソフトウェア・トランザクショナル・メモリ・アプリケーションの関数のうち、トランザクションセーフとしてマークされた各関数について、非トランザクションから呼び出される第１のバージョンのコードと、トランザクションから呼び出される第２のバージョンのコードを作成するステップと、

前記ソフトウェア・トランザクショナル・メモリ・アプリケーションの関数のうち、トランザクションセーフとしてマークされていない各関数について、非トランザクションから呼び出される第１のバージョンのコードを作成するステップと、

前記トランザクションセーフとしてマークされた各関数の前記第１のバージョンのコードのスタブポインターに前記第２のバージョンのコードのエントリポイントを書込み、前記第２のバージョンのコードのスタブポインターにエラールーチンのエントリポイントを書込むステップと

を含む処理を実行させるためのプログラムを記録したコンピューター読取り可能な記録媒体。

【請求項２】

前記第１のバージョンのコードの前記スタブポインターは、前記第１のバージョンのコードの本文の前に格納されることを特徴とする請求項１に記載のコンピューター読取り可

能な記録媒体。
【請求項３】
前記トランザクションセーフとしてマークされていない各関数の前記第１のバージョンのコードの前記スタブポインターは、エラールーチンを示すことを特徴とする請求項１に記載のコンピューター読取り可能な記録媒体。
【請求項４】
前記トランザクションセーフとしてマークされていない各関数の前記第１のバージョンのコードは、エラールーチンを示すスタブポインターを有することを特徴とする請求項１に記載のコンピューター読取り可能な記録媒体。
【請求項５】
コンピューターが、ソフトウェア・トランザクショナル・メモリにおけるトランザクショナル・コード・ブロックを検索する方法であって、前記コンピューターは、前記ソフトウェア・トランザクショナル・メモリを実装するソフトウェア・トランザクショナル・メモリ・アプリケーションを有し、前記方法は、
前記ソフトウェア・トランザクショナル・メモリ・アプリケーションの特定の関数がトランザクションセーフとしてマークされていると判断するステップと、
前記特定の関数について、非トランザクションにおいて使用される第１のバージョンの関数コードと、トランザクションにおいて使用される第２のバージョンの関数コードとを含む２つのバージョンの関数を作成するステップと、
前記第２のバージョンの関数コードを示すスタブポインターを前記第１のバージョンの関数コードに格納し、エラールーチンのエントリポイントを示すスタブポインターを前記第２のバージョンの関数コードに格納するステップと
を備えることを特徴とする方法。
【請求項６】
実行コンテキストに応じたバージョンの前記特定の関数を呼び出すステップをさらに備えたことを特徴とする請求項５に記載の方法。
【請求項７】
前記実行コンテキストが、前記特定の関数への直接呼び出しを用いる非トランザクショナルコンテキストであるとき、前記実行コンテキストに応じたバージョンの前記特定の関数を呼び出すステップは、前記第１のバージョンの関数コードを呼び出すことを含むことを特徴とする請求項６に記載の方法。
【請求項８】
前記実行コンテキストが、前記特定の関数への直接呼び出しを用いるトランザクショナルコンテキストであるとき、前記実行コンテキストに応じたバージョンの前記特定の関数を呼び出すステップは、前記第２のバージョンの関数コードを呼び出すことを含むことを特徴とする請求項６に記載の方法。
【請求項９】
前記実行コンテキストが、前記特定の関数へのポインター呼び出しを用いる非トランザクショナルコンテキストであるとき、前記実行コンテキストに応じたバージョンの前記特定の関数を呼び出すステップは、前記ポインター呼び出しにより、前記第１のバージョンの関数コードを呼び出すことを含むことを特徴とする請求項６に記載の方法。
【請求項１０】
前記実行コンテキストが、前記特定の関数への仮想呼び出しを用いる非トランザクショナルコンテキストであるとき、前記実行コンテキストに応じたバージョンの前記特定の関数を呼び出すステップは、前記仮想呼び出しにより前記第１のバージョンの関数コードを呼び出すことを含むことを特徴とする請求項６に記載の方法。
【請求項１１】
前記特定の関数がトランザクションセーフとしてマークされていると判断することは、前記特定の関数のソースコードの属性に基づいて行われることを特徴とする請求項５に記載の方法。

【請求項１２】
前記特定の関数の前記２つのバージョンはコンパイラによって作成されることを特徴とする請求項５に記載の方法。
【請求項１３】
前記第２のバージョンの関数コードを示すスタブポインターは、前記第１のバージョンの関数コードのエントリポイントに格納されることを特徴とする請求項５に記載の方法。
【請求項１４】
コンピューターに請求項５に記載の方法を実行させるためのプログラムを記録したコンピューター読み取り可能な記録媒体。
【請求項１５】
ソフトウェア・トランザクショナル・メモリにおいて使用する関数について、トランザクションから呼び出される第２のバージョンのコードおよび非トランザクションから呼び出される第１のバージョンのコードを作成する方法であって、前記ソフトウェア・トランザクショナル・メモリを実装するソフトウェア・トランザクショナル・メモリ・アプリケーションを有するコンピューターが、
コンパイル時に、前記ソフトウェア・トランザクショナル・メモリ・アプリケーションの複数の関数の各関数にスタブポインターを割当てるステップと、
前記複数の関数のうちトランザクションセーフとしてマークされていない個々の関数毎に、トランザクションセーフとしてマークされていない個々の関数の第１のバージョンのコードを生成し、該第１のバージョンのコードの前記スタブポインターにランタイム・エラー・ルーチンのエントリポイントを書込むステップと、
前記複数の関数のうちトランザクションセーフとしてマークされた個々の関数毎に、
トランザクションセーフとしてマークされた個々の関数の第１のバージョンのコードおよび第２のバージョンのコードを作成するステップと、
前記第１のバージョンのコードの前記スタブポインターに、前記第２のバージョンのコードのエントリポイントを書込むステップと、
前記第２のバージョンのコードの前記スタブポインターに、ランタイム・エラー・ルーチンのエントリポイントを書込むステップと
を含むことを特徴とする方法。
【請求項１６】
前記ソフトウェア・トランザクショナル・メモリ・アプリケーションにおける各関数の各呼び出しサイトに対して、実行コンテキストに応じた関数の呼び出しを行うステップをさらに備えたことを特徴とする請求項１５に記載の方法。
【請求項１７】
コンピューターに、請求項１５に記載の方法を実行させるためのプログラムを記録したコンピューター読み取り可能な記録媒体。
【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・メモリ・システムに関し、より詳細には、トランザクション・メモリ・システムにおけるトランザクション・コード・ブロック（transactional code block）を効果的に検索する方法に関する。
【背景技術】
【０００２】
ソフトウェア・トランザクション・メモリ（ＳＴＭ）は、並列計算における共有メモリへの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似ている同時実行制御機構である。トランザクションメモリという状況におけるトランザクションは、共有メモリに対して一連の読取りかつ書込みを実行する１つのコードである。ＳＴＭを、従来のロック機構に対する代替手段として用いる。プログラマは、コードブロックの前後に宣言の注釈（例えば、アトミック）を記入して、コードブロックが他の保護されたコード領域に関

してアトミックに実行されることを、プログラマが必要としおよびシステムが自動的に保証する安全のプロパティを示す。ソフトウェア・トランザクション・メモリ・プログラミング・モデル（software transactional memory programming model）は、ロックベースの優先度逆転問題およびデッドロック問題を防ぐ。

【０００３】

ソフトウェア・トランザクション・メモリ（ＳＴＭ）システムは、あるシーケンシャルプログラムを取得し、シーケンシャルプログラムの部分に対して、トランザクションを使用して同時に（例えば、並列に）実行することを許可することができる。通常、プログラマは、種々の種類のプログラミング言語のうちの１つを使用して、シーケンシャルプログラムのソースコードを書く。通常、ソースコードは、後にコンピューターにより実行されるロジックを含む１つまたは複数の関数の中に埋め込まれる。用語「関数」を、関数、メソッド、プロシージャ、ステートメントブロック、および／またはコンピューターにより実行されるロジックの他の部分を含むこととして、本明細書において広く使用する。ソフトウェア・トランザクション・メモリ・システムによって、トランザクションコンテキストから呼び出しをすることができるすべての関数は、２つのバージョンを有するべきである。１つのバージョンは、トランザクションから呼び出しをすることができ、およびもう１つのバージョンは、トランザクションなしから呼び出しをすることができる。呼び出しをする関数についてどのバージョンを決定するかは、コンテキスト依存である。バージョンの決定は、実行時間において、仮想関数に対してまたは関数ポインターによって、呼び出しをするために行われるべきである。

【発明の概要】

【発明が解決しようとする課題】

【０００４】

トランザクション・メモリ・システムにおけるトランザクション・コード・ブロックの作成および／または検索のために、種々の技術および技法を開示する。例えばソフトウェア開発者等のユーザーは、識別子によって個別の関数をデコレートして、個別の関数がトランザクションセーフであることを示すことができる。ノーマルバージョンおよびトランザクションバージョンを、トランザクションセーフとしてマークされたソフトウェアアプリケーションの各関数に対して作成する。ノーマルバージョンを、トランザクションセーフとしてマークされていない各関数に対して作成する。トランザクションセーフとしてマークされた各関数のノーマルバージョンに対して、ノーマルバージョンの中のスタブポインター（stub pointer）よって、トランザクションバージョンを示す。関数に適切なバージョンを、アプリケーションコンテキストに依存して、呼び出しをする。

【課題を解決するための手段】

【０００５】

一実施例において、コンパイラは、トランザクション・メモリ・システムが使用するための関数についてのトランザクションバージョンおよびトランザクションなしのバージョンを生成する。コンパイルするときに、スタブポインターは、ソフトウェアアプリケーションの中の各関数に割当てられる。トランザクションセーフの関数としてマークされていない、それぞれの関数に対して、それぞれの関数についてのノーマルバージョンを生成し、およびランタイム・エラー・ルーチンのエントリポイントよってスタブポインターに書込む。トランザクションセーフの関数としてマークされた、それぞれの関数に対して、コードを、トランザクションセーフであるそれぞれの関数についてのノーマルバージョンおよびトランザクションバージョンとして作成する。ノーマルバージョンのスタブポインターにおいて、エントリポイントを、トランザクションバージョンに対して書込む。トランザクションバージョンのスタブポインターにおいて、エントリポイントを、ランタイム・エラー・ルーチンに対して書込む。

【０００６】

本発明の概念を提供して、発明を実施するための形態において、さらに、以下に記述される概念についての選択を、簡略化された形式において紹介した。本発明の概要は、主張

される主題についての重要な特徴または基本的な特徴を識別することを意図せず、主張される主題の範囲の決定における補助として用いられることを意図しない。

【図面の簡単な説明】

【０００７】

【図１】本発明の一実施例に係るコンピューターシステムを表す図である。

【図２】本発明の一実施例に係る図１のコンピューターシステムの演算についてのトランザクション・メモリ・アプリケーションを表す図である。

【図３】本発明の一実施例に係る図１のコンピューターシステムの高水準処理のフローチャートである。

【図４】本発明の一実施例に係る図１のコンピューターシステムについての、トランザクション関数とトランザクションなしの関数とのコードを生成し、およびコンパイル処理とリンク処理との少なくとも一方の一部としてスタブポインターに書込むコードジェネレーターの使用にかかわるステージを例示する処理のフローチャートである。

【図５】本発明の一実施例に係る図１のコンピューターシステムについての、関数に適切なバージョンを選択し、実行コンテキストに依存して呼び出しをするデシジョンツリーを例示する図である。

【図６】本発明の一実施例に係る、１つの関数がちょうどノーマルバージョンを有し、もう一方の関数がノーマルバージョンとトランザクションバージョンとを有する２つの仮想的な関数を例示する図である。

【発明を実施するための形態】

【０００８】

本発明の原理の理解を促進する目的のため、図面において例示する実施形態を参照し、および特定の言語を使用して同一のことを説明する。それにもかかわらず範囲の限定を意図しないことが理解されるだろう。説明される実施形態におけるあらゆる交替、さらなる変更、本明細書において説明する原理のあらゆるさらなる応用は、当事業者が普通は気づくであろうと同程度に予想される。

【０００９】

システムを、トランザクション・メモリ・システムとして一般的な文脈において説明するが、システムは、追加として他の目的に役立つ。一実施例において、本明細書にて説明する１つまたは複数の技法を、例えば本願発明の特許出願人のフレームワーク製品等のフレームワークプログラム内の機能として、または開発者にプラットホームを提供してソフトウェアアプリケーションを開発するプログラムもしくはサービスの他のあらゆる種類からの機能として、実装することができる。別の実施例において、本明細書にて説明する１つまたは複数の技法を、並列環境にて実行されるアプリケーションの開発を取り扱う他のアプリケーションによる機能として、実装することができる。

【００１０】

一実施例において、トランザクションセーフとしてマークされた各関数の２つのバージョンを、詳細には関数のノーマルバージョンおよび関数のトランザクションバージョンをプログラム的に作成するトランザクション・メモリ・システムを提供する。トランザクションセーフとしてマークされていない関数に対して、ノーマルバージョンのみを作成し、およびノーマルバージョンのエントリポイントにおけるスタブポインターを、エラールーチンに向ける。トランザクションセーフとしてマークされた関数に対して、ノーマルバージョンのエントリポイントのスタブポインターを、関数のトランザクションバージョンに向ける。トランザクションバージョンのエントリポイントにおけるスタブポインターを、エラールーチンに向ける。トランザクションセーフとしてマークされていない関数に対して、ノーマルバージョンのエントリポイントにおけるスタブポインターは、エラールーチンを示す。関数に適切なバージョンを、実行コンテキストに依存して、呼び出しをする。

【００１１】

図１を参照すると、１つまたは複数のシステムのパーツを実装するために使用する例示的コンピューターシステムは、例えばコンピューティングデバイス１００等のコンピュー

ティングデバイスを含む。通常、最も基本的な構成において、コンピューティングデバイス１００は、少なくとも１つの処理装置１０２およびメモリ１０４を含む。コンピューティングデバイスの正確な構成および種類に依存して、メモリ１０４は、揮発性メモリ（例えば、ＲＡＭ等）、不揮発性メモリ（例えば、ＲＯＭ、フラッシュメモリ等）または２つの何れかの組合せとすることができる。図１において、最も基本的な構成を、破線１０６によって例示する。

【００１２】

加えて、デバイス１００は、追加の機能／機能性を有することができる。例えば、デバイス１００は、限定されないが、磁気もしくは光ディスクまたは磁気もしくは光テープを含んでいる、追加の（取外し可能および／または固定の）記憶装置を含むことができる。図１において、追加の記憶装置を、取外し可能記憶装置１０８および固定記憶装置１１０によって例示する。コンピューター記憶メディアは、例えばコンピューター読取り可能な命令、データ構造、プログラムモジュール、または他のデータ等の情報を格納するあらゆる方法または技術に実装される揮発性および不揮発性の、取外し可能および固定のメディアを含む。メモリ１０４、取外し可能記憶装置１０８、および固定記憶装置１１０は、すべてコンピューター記憶メディアの例である。コンピューター記憶メディアは、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もしくは他の磁気記憶装置、または望ましい情報の格納に使用することができる他のあらゆるメディアおよびデバイス１００によってアクセスすることができる他のあらゆるメディアを含む。

【００１３】

コンピューティングデバイス１００は、コンピューティングデバイス１００に他のコンピューター／アプリケーション１１５との通信を可能にする、１つまたは複数の通信接続１１４を含む。デバイス１００は、例えば、キーボード、マウス、ペン、音声入力デバイス、タッチ入力デバイス等、（複数の）入力デバイス１１２を有することができる。（複数の）出力デバイス１１１を、例えば、ディスプレイ、スピーカー、プリンター等を含むことができる。上記デバイスは、本技術分野においてよく知られており、本明細書において詳細に論ずる必要がない。一実施例において、コンピューティングデバイス１００は、トランザクション・メモリ・アプリケーション（transactional memory application）２００を含む。図２において、トランザクション・メモリ・アプリケーション２００を、さらに詳細に説明することにする。

【００１４】

図２を参照して、図１の参照を続けつつ、コンピューティングデバイス１００上において動作するトランザクション・メモリ・アプリケーション２００を、説明する。トランザクション・メモリ・アプリケーション２００は、コンピューティングデバイス１００に属するアプリケーションプログラムの１つである。しかしながら、代替としてまたは追加として、トランザクション・メモリ・アプリケーション２００を、１つまたは複数のコンピューター上に、および／または図１に示す種々のバリエーションにおいて、コンピューター実行可能な命令として含むことができる。代替としてまたは追加として、トランザクション・メモリ・アプリケーション２００の１つまたは複数のパーツを、他のコンピューター／アプリケーション１１５上の、またはコンピューターソフトウェア技術者が気づくような他のバリエーション上のシステムメモリ１０４のパーツであるとすることができる。

【００１５】

トランザクション・メモリ・アプリケーション２００は、プログラムロジック２０４を含む。プログラムロジック２０４は、本明細書において説明される技法のい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２０４は、トランザクション・メモリ・システムを提供するロジック２０６、ユーザー／開発者がソフトウェアアプリケーションの関数を属性または他の識別子によってデコレートすることができるようにして、関数がトランザクションセーフであることを示すロジック２０８、トランザクションセーフとしてマ

ークされた各関数に対して２つのバージョン（例えば、ノーマルバージョンおよびトランザクションバージョン）を作成するロジック２１０、トランザクションセーフとしてマークされていない各関数に対して１つのバージョン（例えば、ノーマルバージョン）を作成するロジック２１２、各関数に対してスタブポインターを（例えば、コードセクションの冒頭または他の場所の前に）割当てるロジック２１４、スタブポインターが正しい場所（例えば、トランザクションバージョンまたはランタイム・エラー・ルーチン）を示すようにさせるロジック２１６、およびアプリケーションを動作させる他のロジック２２０を含む。一実施例において、プログラムロジック２０４を、例えば、プログラムロジック２０４においてプロシージャに対して単一の呼び出しを使用すること等、別のプログラムからプログラム的に呼び出しをするのに使用可能である。

【００１６】

図３～６を参照して、図１～２の参照を続けつつ、トランザクション・メモリ・アプリケーション２００の１つまたは複数の実施例を実施するステージを、より詳細に説明する。図３は、トランザクション・メモリ・アプリケーション２００の高水準処理のフローチャートである。１つのフォームにおいて、図３の処理を、コンピューティングデバイス１００の動作ロジックに少なくとも部分的に実装する。処理は、開始点２４０において始まり、トランザクション・メモリ・システム（例えば、ソフトウェア・トランザクション・メモリ・システム）を提供する（ステージ２４２）。システムは、ユーザー／開発者が個別の関数を属性または他の識別子によってデコレートすることができるようにして、関数がトランザクションセーフであることを示す（ステージ２４４）。コンパイラおよびリンカーの少なくとも一方は、個別の関数についての２つのバージョンを作成する。１つは、トランザクションなしに使用するバージョンであり（例えば、ノーマルバージョン）、および１つは、トランザクションありに使用するバージョンである（例えば、トランザクションバージョン）（ステージ２４６）。ノーマルバージョンにおいて、システムは、個別の関数についてのトランザクションバージョンを示すスタブポインターを格納する（ステージ２４８）。一実施例において、本明細書において説明される（複数の）スタブポインターは、それぞれコンパイルされる関数本文の前に格納される。他の実施例において、スタブポインターがそれぞれの関数のバージョンに関連付けられていれば、スタブポインターを、他の位置に格納することができる。トランザクションバージョンにおいて、システムは、ランタイム・エラー・ルーチンを示すスタブポインターを格納する（ステージ２４９）。関数に適切なバージョンを、実行コンテキストに依存して、呼び出しをする（ステージ２５０）。処理は、終了点２５２において終了する。

【００１７】

図４は、トランザクション関数とトランザクションなしの関数とのコードを生成し、およびコンパイル処理とリンク処理との少なくとも一方の一部としてスタブポインターに書込むコードジェネレーターの使用にかかわるステージの一実施例を例示する。１つのフォームにおいて、図４の処理を、コンピューティングデバイス１００の動作ロジックに少なくとも部分的に実装する。処理は、開始点２７０において始まり、コードジェネレーターが、コンパイルするときに、各関数に対して追加のポインター（例えば、スタブポインター）を割当てる（ステージ２７２）。各関数に対して（すなわち、関数がある間）（判断ポイント２７４）、種々のタスクを実行する。例えば、関数がトランザクションセーフの関数としてマークされていないと（判断ポイント２７６）、システムは、関数に対してノーマルバージョン（すなわち、通常のコード）を生成し、およびランタイム・エラー・ルーチンのエントリポイントによってノーマルバージョンのスタブポインターに書込む（ステージ２８６）。

【００１８】

関数がトランザクションセーフの関数としてマークされると（判断ポイント２７６）、システムは、ノーマルバージョンＦおよびトランザクションバージョンＦＴに対して、コードを作成する（ステージ２７８）。システムは、関数Ｆのトランザクションバージョンおよびノーマルバージョンに対してスタブポインターを割当てる（ステージ２８０）。ノ

ーマルバージョンに対するスタブポインターに、トランザクションバージョンのエントリポイントを書込む（ステージ２８２）。トランザクションバージョンに対するスタブポインターに、ランタイム・エラー・ルーチンのエントリポイントを書込む（ステージ２８４）。上記ステージを、各関数に対して適切に繰返す。各呼び出しサイトに対して、システムは、コンテキストに依存して、適切な呼び出しをする（ステージ２８８）。図５において、呼び出しサイトの判断処理を、さらに詳細に説明する。処理は、終了点２９０において終了する。

【００１９】

図５を参照すると、図１のシステムの一実施例に係る図３００は、関数に適切なバージョンを選択し、実行コンテキストに依存して呼び出しをするのに適しているデシジョンツリーを例示する。１つのフォームにおいて、図５の判断処理を、コンピューティングデバイス１００の動作ロジックに少なくとも部分的に実装する。トランザクションなしのコンテキスト３０２における直接呼び出し３０６に対して、ノーマルバージョンのエントリポイントを、関数３１２に対し直接的に呼び出しをする。トランザクションコンテキスト３０４における直接呼び出し３０６に対して、トランザクションバージョンのエントリポイントを、関数３１４に対し呼び出しをする。トランザクションなしのコンテキスト３０２におけるポインター呼び出し３０８に対して、呼び出しは、関数３１６に対し同様である。トランザクションコンテキスト３０４におけるポインター呼び出し３０８に対して、スタブ関数を示すワードのアドレスを、関数ポインターの値の逆参照から計算し、およびアドレスを、呼び出し３１８において使用する。スタブ関数は、トランザクションバージョンのエントリポイントへのポインターによって格納されるので、正確なバージョンが実行される。トランザクションなしのコンテキスト３０２における仮想呼び出しまたはインターフェース呼び出し３１０は、影響を受けず、および呼び出し３２０も同様に影響を受けない。トランザクションコンテキスト３０４における仮想呼び出しまたはインターフェース呼び出し３１０は、ｖテーブル検索を実行し、および関数３２２のトランザクションバージョンについてのスタブ関数のエントリポイントを計算する。計算されたエントリポイントを、呼び出しをするのに使用する。

【００２０】

図６は、１つの関数がちょうどノーマルバージョンを有し、およびもう一方の関数がノーマルバージョンとトランザクションバージョンとを有する２つの仮想的な関数を例示する一実施例の論理的な図４００である。「ＢＡＲ」関数４０２は、ソースコードにおいてトランザクションセーフとしてマークされていなかった関数である。上述のように、関数４０２に対するエントリポインター４１２は、エラールーチン４１０を示す。「ＦＯＯ」関数４１４は、元のソースコードにおいてトランザクションセーフとしてマークされていた関数であり、結果として、普通にコンパイルされるバージョン（ノーマルバージョン）を、トランザクションバージョン４０８とともに、関数４０６に提供する。ＦＯＯ関数４０６のノーマルバージョンについてのエントリポイント４１４は、ＦＯＯ関数４０８のトランザクションバージョンを示す。ＦＯＯ関数４０８のトランザクションバージョンについてのエントリポイント４１６は、エラールーチン４１０を示す。一実施例において、上記エラールーチンを上記関数のスタブ（エントリポイント）に埋め込み、エラーを、不適切な使用に直面するときのランタイム時に見つけることを可能にする。

【００２１】

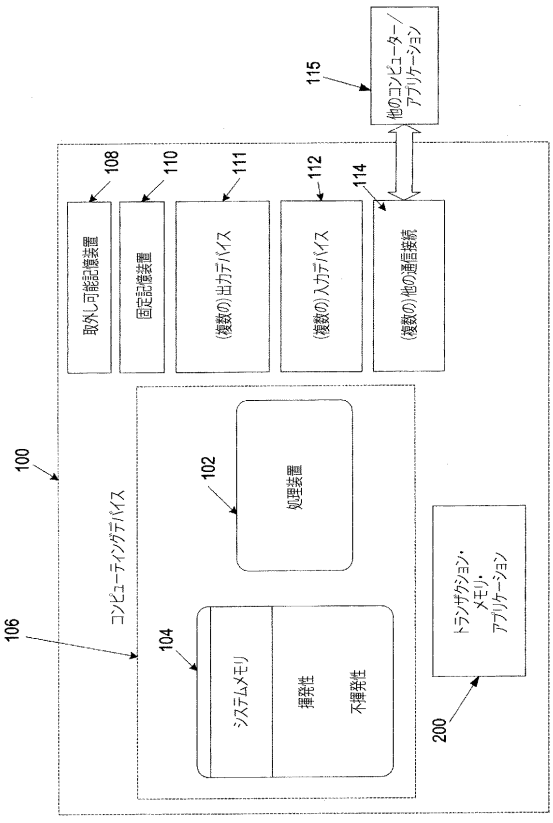
主題を、構造的な機能および／または方法論的な動作に固有の言語において説明してきたが、添付のクレームにおいて明確にされる主題は、上記の特定の機能または動作を必ずしも限定されないことが理解される。適切に言えば、上記の特定の機能および動作を、クレームを実施する形式の例として開示する。本明細書において説明される実施例の精神のおよび／または次に述べるクレームによる範囲内にある、すべての等価、修正、および変更は、保護されることを望む。

【００２２】

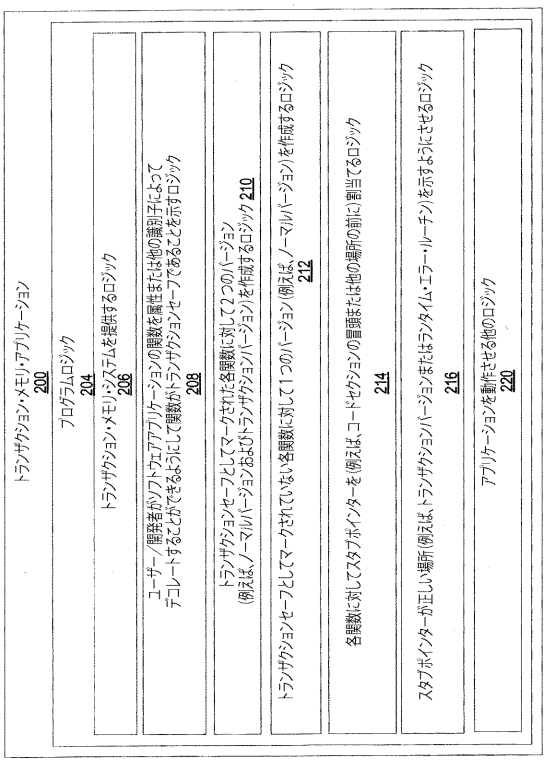
例えば、コンピューターソフトウェア分野の当事業者は、本明細書において論じた実施

例にて説明されたのと同程度に、クライアント配置および／またはサーバ配置を、ユーザーインターフェースの画面コンテンツおよび／またはデータ配置を、１つまたは複数のコンピューターに異なった構成をして、実施例において表されたのよりも、より少ないオプションもしくは追加のオプションまたはより少ない機能もしくは追加の機能を含むことができるだろう。

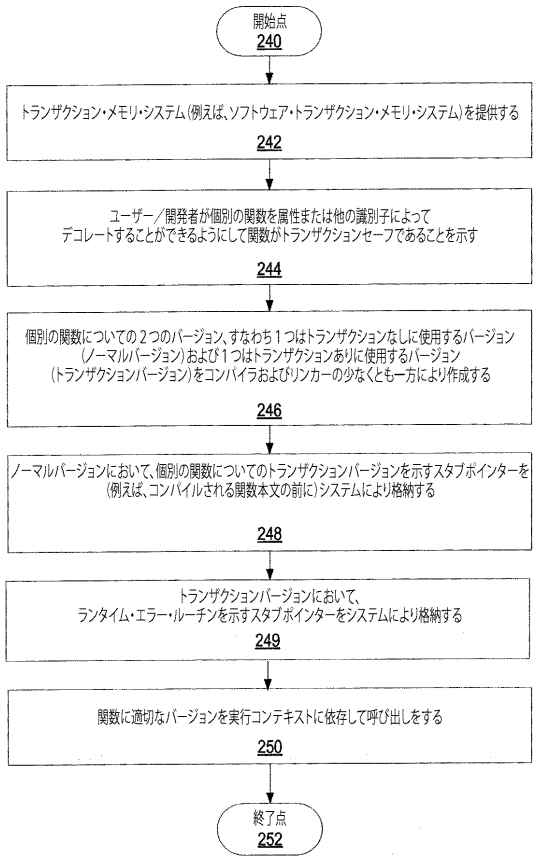
【図１】



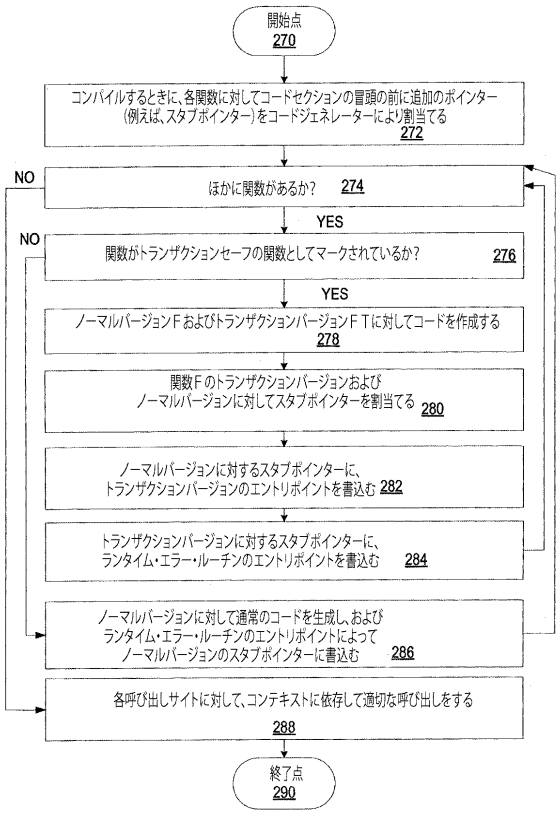
【図２】



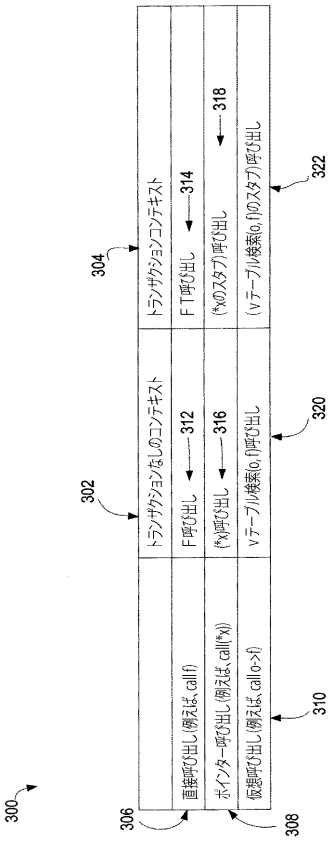
【図３】


開始点
240
トランザクション・メモリ・システム(例えば、ソフトウェア・トランザクション・メモリ・システム)を提供する
242
ユーザー／開発者が個別の関数を属性または他の識別子によって
デコレートすることができるようにして関数がトランザクションセーフであることを示す
244
個別の関数についての２つのバージョン、すなわち１つはトランザクションなしに使用するバージョン
(ノーマルバージョン)および１つはトランザクションありに使用するバージョン
(トランザクションバージョン)をコンパイラおよびリンカーの少なくとも一方により作成する
246
ノーマルバージョンにおいて、個別の関数についてのトランザクションバージョンを示すスタブポインターを
(例えば、コンパイルされる関数本文の前に)システムにより格納する
248
トランザクションバージョンにおいて、
ランタイム・エラー・ルーチンを示すスタブポインターをシステムにより格納する
249
関数に適切なバージョンを実行コンテキストに依存して呼び出しをする
250
終了点
252

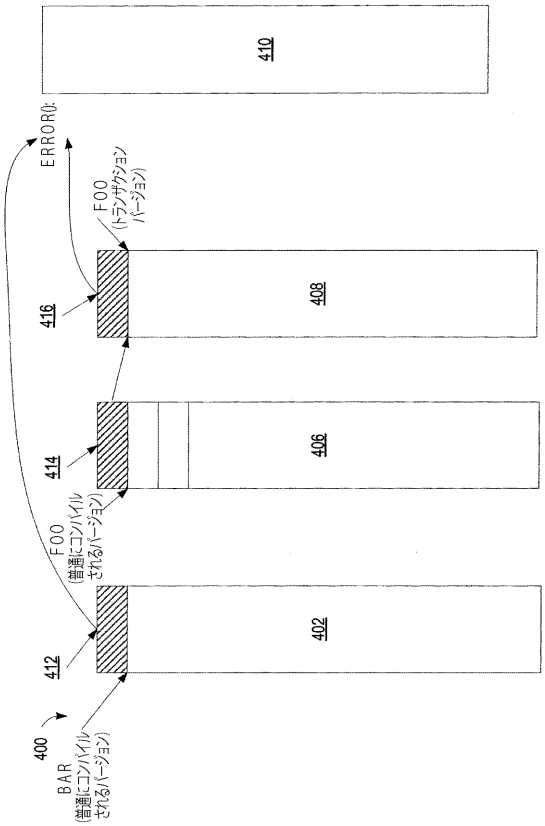
【図４】


開始点
270
コンパイルするときに、各関数に対してコードセクションの冒頭の前に追加のポインター
(例えば、スタブポインター)をコードジェネレーターにより割当てる
272
NO
ほかに関数があるか？ 274
YES
NO
関数がトランザクションセーフの関数としてマークされているか？ 276
YES
ノーマルバージョンFおよびトランザクションバージョンFTに対してコードを作成する
278
関数Fのトランザクションバージョンおよび
ノーマルバージョンに対してスタブポインターを割当てる 280
ノーマルバージョンに対するスタブポインターに、
トランザクションバージョンのエントリポイントを書込む 282
トランザクションバージョンに対するスタブポインターに、
ランタイム・エラー・ルーチンのエントリポイントを書込む 284
ノーマルバージョンに対して通常のコードを生成し、および
ランタイム・エラー・ルーチンのエントリポイントによって
ノーマルバージョンのスタブポインターに書込む 286
各呼び出しサイトに対して、コンテキストに依存して適切な呼び出しをする
288
終了点
290

【図５】


300
302
304
306
308
310
312
314
316
318
320
322
トランザクションなしのコンテキスト
トランザクションコンテキスト
F呼び出し
FT呼び出し
(*x)呼び出し
(*xのスタブ)呼び出し
直接呼び出し(例えば、call f)
ポインター呼び出し(例えば、call(*x))
仮想呼び出し(例えば、call o->f)
vテーブル検索(o, f)呼び出し
(vテーブル検索(o, f)のスタブ)呼び出し

【図６】


410
ERROR!!
FOO
(トランザクション
バージョン)
416
408
406
414
FOO
(普通にコンパイル
されるバージョン)
402
412
400
BAR
(普通にコンパイル
されるバージョン)

フロントページの続き

(72)発明者　デイヴィッド　キャラハン
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　インターナショナル　パテンツ内
(72)発明者　ビノッド　ケー．グローバー
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

審査官　田中　幸雄

(56)参考文献　米国特許出願公開第２００８／０１２０５９０（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／４６
Ｇ０６Ｆ　　９／５２