(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5705871号
(P5705871)
(45)発行日　平成27年4月22日(2015.4.22)　(24)登録日　平成27年3月6日(2015.3.6)

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

請求項の数 14 (全 24 頁)

(21)出願番号　特願2012-542028(P2012-542028)
(86)(22)出願日　平成22年11月5日(2010.11.5)
(65)公表番号　特表2013-513162(P2013-513162A)
(43)公表日　平成25年4月18日(2013.4.18)
(86)国際出願番号　PCT/US2010/055734
(87)国際公開番号　WO2011/068631
(87)国際公開日　平成23年6月9日(2011.6.9)
審査請求日　平成25年10月1日(2013.10.1)
(31)優先権主張番号　12/631,023
(32)優先日　平成21年12月4日(2009.12.4)
(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】分散された永続性インスタンスに対するロックの解決

(57)【特許請求の範囲】
【請求項１】
１つまたは複数のプロセッサおよびシステムメモリを含むコンピュータアーキテクチャにおいて、２つ以上の実行スレッド間のロック競合を解決する方法であって、前記コンピュータアーキテクチャは、アプリケーションホスト、永続化プロバイダ、およびインスタンスストアをさらに含み、前記アプリケーションホストは、インスタンスを前記インスタンスストアに永続化し、かつ前記インスタンスストアからのインスタンスにアクセスするためのコマンドを発行するように構成された複数の実行スレッドを含み、少なくとも前記コマンドのサブセットが、実行スレッドが実行中にインスタンスのロックを要求することを生じさせ、前記方法は、
前記永続化プロバイダが、第１のコマンドを、前記アプリケーションホストに含まれる第１の実行スレッドから受信する動作であって、前記第１のコマンドは、１つまたは複数の条件を満たす場合に第１のインスタンスに対するロックの取得を要求するように構成される、受信する動作と、
前記永続化プロバイダが、前記第１のコマンドをコマンドログに記録する動作と、
前記永続化プロバイダが、前記第１のコマンドを前記インスタンスストアにサブミットする動作と、
前記永続化プロバイダが、第２のコマンドを、前記アプリケーションホストに含まれる第２の実行スレッドから受信する動作であって、前記第２のコマンドは、前記第２のコマンドにより処理するために第２のインスタンスを前記第２の実行スレッドに対してロック

することを要求するように構成される、受信する動作と、
前記永続化プロバイダが、前記第２のコマンドを前記コマンドログに記録する動作と、
前記永続化プロバイダが、前記第２のコマンドを前記インスタンスストアにサブミットする動作と、
前記永続化プロバイダが、前記第２のコマンドに対するロック応答を、前記インスタンスストアから受信する動作であって、前記ロック応答は、前記アプリケーションホストが前記第２のインスタンスに対するロックの保持者であることを示し、前記ロック応答は、前記第１のコマンドをサブミットした後、かつ前記第１のコマンドが完了する前に受信される、受信する動作と、
前記永続化プロバイダが、前記コマンドログを参照し、前記第１のコマンドの現在の解決が、（ａ）前記第１のコマンドが前記第１のインスタンスに対するロックを取得したかどうか、ならびに（ｂ）前記第１のインスタンスおよび前記第２のインスタンスが同一のインスタンスであるかどうかを判定するには、十分な情報を提供していないと判定する動作であって、前記十分な情報を提供していないことは、前記第２のコマンドが要求したロックが、それ以前に前記第１のコマンドによって取得されたロックであったかどうかを曖昧にする、判定する動作と、
前記永続化プロバイダが、前記第１のコマンドのさらなる解決に到達するまで、前記第２のコマンドの処理を中断する動作であって、前記さらなる解決は、少なくとも、前記１つまたは複数の条件を満たすことに関する追加情報を提供する、中断する動作と、
前記永続化プロバイダが、前記第１のコマンドの解決に関する前記追加情報に基づき、前記第２のコマンドに関してどのように進めるかを判定する動作と
を含むことを特徴とする方法。
【請求項２】
前記永続化プロバイダが、前記追加情報に基づき、前記第１のコマンドは前記第２のコマンドが要求したロックを取得したと判定する動作をさらに含むことを特徴とする請求項１に記載の方法。
【請求項３】
前記永続化プロバイダが、前記第１のコマンドをコマンドログに記録する動作は、前記第１のコマンドを、第１のタイムスタンプと共に前記コマンドログに記録する動作であって、前記第１のタイムスタンプは、前記永続化プロバイダのコマンドクロックにより提供される、記録する動作を含み、
前記永続化プロバイダが、前記第２のコマンドを前記コマンドログに記録する動作は、前記第２のコマンドを第２のタイムスタンプと共に前記コマンドログに記録する動作であって、前記第２のタイムスタンプは、前記永続化プロバイダの前記コマンドクロックにより提供される、記録する動作を含む
ことを特徴とする請求項２に記載の方法。
【請求項４】
前記永続化プロバイダが、前記追加情報に基づき、前記第１のコマンドは前記第２のコマンドが要求したロックを取得したと判定する動作は、前記第１のタイムスタンプおよび前記第２のタイムスタンプに基づき、前記第１のコマンドを前記第２のコマンドより前に受信したと判定する動作を含むことを特徴とする請求項３に記載の方法。
【請求項５】
前記永続化プロバイダが、前記第２のコマンドに関してどのように進めるかを判定する動作は、前記第２のコマンドを失敗とする動作を含むことを特徴とする請求項２に記載の方法。
【請求項６】
前記永続化プロバイダが、前記第２のコマンドに関してどのように進めるかを判定する動作は、前記アプリケーションホストを前記第１のコマンドがロックを取得したインスタンスのコピーに導く動作を含むことを特徴とする請求項２に記載の方法。
【請求項７】

前記永続化プロバイダが、前記第１のコマンドは前記第２のコマンドが要求したロックを取得したと判定する動作は、
前記第１のコマンドは前記インスタンスのバージョンに対するロックを取得したと判定する動作と、
前記第２のコマンドに対する前記ロック応答は、前記インスタンスの同一のバージョンに対するロックに対するものであったと判定する動作と
を含むことを特徴とする請求項２に記載の方法。
【請求項８】
前記永続化プロバイダが、前記追加情報に基づき、前記第１のコマンドは前記第２のコマンドが要求したロックを取得しなかったと判定する動作をさらに含むことを特徴とする請求項１に記載の方法。
【請求項９】
前記永続化プロバイダが、第３のコマンドを、前記アプリケーションホストに含まれる第３の実行スレッドから受信する動作であって、前記第３のコマンドは、１つまたは複数の条件を満たす場合に第３のインスタンスに対するロックの取得を要求するように構成される、受信する動作と、
前記永続化プロバイダが、前記第３のコマンドをコマンドログに記録する動作と、
前記永続化プロバイダが、前記第３のコマンドを前記インスタンスストアにサブミットする動作と、
前記永続化プロバイダが、前記コマンドログを参照し、前記第３のコマンドの現在の解決が、（ａ）前記第３のコマンドが前記第３のインスタンスに対するロックを取得したかどうか、ならびに（ｂ）前記第３のインスタンスおよび前記第２のインスタンスが同一のインスタンスであるかどうかを判定するには、十分な情報を提供していないと判定する動作であって、前記十分な情報を提供していないことは、前記第２のコマンドが要求したロックが、それ以前に前記第３のコマンドによって取得されたロックであったかどうかを曖昧にする、判定する動作と、
前記永続化プロバイダが、前記第３のコマンドのさらなる解決に到達するまで、前記第２のコマンドの処理を中断する動作であって、前記さらなる解決は、少なくとも、前記１つまたは複数の条件を満たすことに関する追加情報を提供する、中断する動作と、
前記永続化プロバイダが、前記第３のコマンドの解決に関する前記追加情報に基づき、前記第２のコマンドに関してどのように進めるかを判定する動作と
をさらに含むことを特徴とする請求項８に記載の方法。
【請求項１０】
前記永続化プロバイダが、第２のコマンドを受信する動作は、前記永続化プロバイダが、前記第２のコマンドを、前記第１のコマンドを受信した後に受信する動作を含み、
前記永続化プロバイダが、前記第２のコマンドに対するロック応答を、前記インスタンスストアから受信する動作は、前記第２のコマンドに対するロック応答を、前記第１のコマンドに対応する応答を受信する前に受信する動作を含む
ことを特徴とする請求項１に記載の方法。
【請求項１１】
前記永続化プロバイダが、前記コマンドログを参照し、前記第１のコマンドの現在の解決が、十分な情報を提供していないと判定する動作は、前記第１のコマンド内のエイリアスがインスタンスに対していまだ解決されていないと判定する動作を含むことを特徴とする請求項１に記載の方法。
【請求項１２】
１つまたは複数のプロセッサおよびシステムメモリを含むコンピュータアーキテクチャにおいて、２つ以上の実行スレッド間のロック競合を解決する方法であって、前記コンピュータアーキテクチャは、１つまたは複数のアプリケーションホスト、永続化プロバイダ、およびインスタンスストアをさらに含み、前記１つまたは複数のアプリケーションホストのそれぞれは、インスタンスを前記インスタンスストアに永続化し、かつ前記インスタ

ンスストアからのインスタンスにアクセスするためのコマンドを発行するように構成された少なくとも１つの実行スレッドを含み、少なくとも前記コマンドのサブセットが、実行スレッドが実行中にインスタンスのロックを要求することを生じさせ、前記方法は、
前記永続化プロバイダが、第１のコマンドを、前記アプリケーションホストに含まれる第１の実行スレッドから受信する動作であって、前記第１のコマンドは、第１のインスタンスに対するロックの取得を要求するように構成される、受信する動作と、
前記永続化プロバイダが、前記第１のコマンドを前記インスタンスストアにサブミットする動作と、
前記永続化プロバイダが、前記第１のコマンドに対するロック応答を、前記インスタンスストアから受信する動作であって、前記ロック応答は、前記アプリケーションホストが前記第１のインスタンスに対するロックを取得したことを示し、前記第１のインスタンスに対するロックは、第１のインスタンスバージョンに対するものである、受信する動作と、
前記永続化プロバイダが、第２のコマンドを、前記アプリケーションホストに含まれる第２の実行スレッドから受信する動作であって、前記第２のコマンドは、前記第２のコマンドにより処理するために第２のインスタンスを前記第２の実行スレッドに対してロックすることを要求するように構成され、前記第２のコマンドは前記第１のコマンドより後に受信される、受信する動作と、
前記永続化プロバイダが、前記第２のコマンドを前記インスタンスストアにサブミットする動作と、
前記永続化プロバイダが、前記第２のコマンドに対するロック応答を前記インスタンスストアから受信する動作であって、前記ロック応答は、前記アプリケーションホストが前記第２のインスタンスに対するロックを取得したことを示し、前記第２のインスタンスに対するロックは、第２のインスタンスバージョンに対するものであり、前記第２のコマンドに対する前記ロック応答は前記第１のコマンドに対する前記ロック応答より後に受信され、前記第２のインスタンスバージョンは前記第１のインスタンスバージョンよりも新しい、受信する動作と、
前記永続化プロバイダが、前記第１のインスタンスおよび前記第２のインスタンスは同一のインスタンスであると判定する動作と、
前記永続化プロバイダが、前記第２のコマンドに対する前記ロック応答に対応する前記第２のインスタンスバージョンは前記第１のコマンドに対する前記ロック応答に対応する前記第１のインスタンスバージョンより新しいインスタンスバージョンであると判定する動作と、
前記第１のコマンドが前記第２のコマンドより前に受信されたとしても、前記第１のコマンドが旧インスタンスバージョンに対するロックを保持しているという前記判定に応じて、前記永続化プロバイダが、前記第１のコマンドを失敗とする動作と
を含むことを特徴とする方法。
【請求項１３】
前記インスタンスストアが前記第１のコマンドに対する前記ロック応答を前記永続化プロバイダに送った後、かつ前記インスタンスストアが前記第２のコマンドを前記永続化プロバイダから受信する前に、前記インスタンスストアが、１つまたは複数の他のコマンドを、１つまたは複数の他のアプリケーションホストから受信する動作と、
インスタンスクロックが、前記１つまたは複数の他のコマンドに応答し、バージョン更新規則に従って、前記インスタンスのバージョンを、前記第１のインスタンスバージョンから前記第２のインスタンスバージョンへと更新する動作と
をさらに含むことを特徴とする請求項１２に記載の方法。
【請求項１４】
永続性インスタンスに対するロック競合を解決するコンピュータシステムであって、
１つまたは複数のプロセッサと、
システムメモリと、

１つまたは複数のアプリケーションホスト、永続化プロバイダ、およびインスタンスストアを示すコンピュータ実行可能命令を格納する１つまたは複数のコンピュータ記憶媒体であって、前記１つまたは複数のアプリケーションホストのそれぞれは、少なくとも１つの実行スレッドを含み、
前記１つまたは複数のアプリケーションのそれぞれは、
インスタンスを前記インスタンスストアに永続化し、かつ前記インスタンスストアからのインスタンスにアクセスするためのコマンドを送信し、少なくとも前記コマンドのサブセットが、実行スレッドが実行中にインスタンスのロックを要求することを生じさせるように構成され、
前記インスタンスストアは、
前記インスタンスストアに格納されたインスタンスに対するロックの取得を要求するコマンドを、前記永続化プロバイダから受信し、
インスタンスクロックを用いて、インスタンスのバージョニングポリシーに従ってインスタンスのバージョンを保持し、
受信したコマンドに応答して、ロックがインスタンスに対していつ取得されたかを、ロックが取得された前記インスタンスのバージョンを示すことを含めて、前記永続化プロバイダに示し、
受信したコマンドに対するロックの取得の要求が、すでにロックされているインスタンスにいつ指示されたかを、前記永続化プロバイダに示す
ように構成され、
前記永続化プロバイダは、
インスタンスに対するロックの取得を要求するように構成されたコマンドを受信することを含め、前記１つまたは複数のアプリケーションホストに含まれる実行スレッドから前記コマンドを受信し、前記永続化プロバイダは、ロック競合を引き起こす可能性があるコマンドとロック競合を引き起こす可能性がないコマンドとを区別するように構成され、
コマンドを受信した論理時間を判定し、
コマンドを受信した論理時間を含むコマンド情報をコマンドログに記録し、前記永続化プロバイダは、ロック競合を引き起こす可能性があると判定される受信されたコマンドのコマンド情報のみを記録するようにさらに構成され、
受信したコマンドに応答して、ロックがインスタンスに対していつ取得されたかを示す、前記インスタンスストアからの指示を受信し、
受信したコマンドに対するロックの取得の要求が、すでにロックされているインスタンスにいつ指示されたかを示す、前記インスタンスストアからの指示を受信し、
前記コマンドログに記録された情報と共に、受信した指示に基づき、アプリケーションホスト間のロック競合を検出し、
前記コマンドログに記録された前記コマンド情報と共に、受信した指示に基づき、ロック競合を解決する
ように構成される、コンピュータ記憶媒体と
を備えたことを特徴とするコンピュータシステム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、分散された永続性インスタンスに対するロックの解決に関する。

【背景技術】

【０００２】

コンピュータシステムおよび関連技術は、社会の多くの側面に影響を与える。実際、コンピュータシステムの情報処理能力は、我々の生活や仕事の方法を変えてきた。コンピュータシステムの出現以前には手動で行われていた数多くのタスク（例えば、文書処理、スケジューリング、会計等）を、今では一般にコンピュータシステムが行っている。ごく最近では、コンピュータシステムを互いにまた他の電子機器と連結して有線および無線両方

のコンピュータネットワークを形成し、そのコンピュータネットワークを介して、コンピュータシステムおよび他の電子機器が電子データを転送することができるようになっている。従って、多くのコンピューティングタスクの実施が、多数の異なるコンピュータシステムおよび／または多数の異なるコンピューティング環境にわたって分散されている。

【０００３】

例えばワークフローアプリケーションといった長期運用のアプリケーションには、周期的にそのワークを永続化することにより、エラー、クラッシュ、または機器の故障後アプリケーションを回復させることを可能にすることが多くの場合効果的である。ワークの永続化により、アプリケーションを一時的にアイドル状態にすること、およびアプリケーションにリソースを再割り当てさせることが可能となる。ワークを永続化するために、アプリケーションホストは、永続化状態をランタイム状態と連携させ、一貫したチェックポイントが確実に作成されるようにする。例えば、永続化状態は、アプリケーショントランザクション、メッセージ通知、ロック、ローカル状態のキャッシュ等と連携する必要があり得る。

【０００４】

コンピュータシステムはまた、多くの長期運用のアプリケーションを同時に実行することを含めて、ワークを複数の計算スレッド、コア、およびプロセッサ間で分割して、実質的には並列化されている。従って、アプリケーションホストは、綿密なブックキーピングを用いて、複数の同時実行スレッドが、永続化状態およびランタイム状態と相互作用できるようにしなければならない。アプリケーションホストおよび状態永続化システムを、異なる機器上に配置することができるので、この連携を、更に分散システム内に埋め込むことができる。これにより、アプリケーションホストと状態永続化システムとの間の相互作用は、たとえアプリケーションホストにおいて実行スレッドに絶対的な順序付けがなされていても、再順序付けされる場合がある。さらに、２つの実行スレッドが同一の状態を参照しているということを曖昧にしながら、永続化状態を様々なエイリアスによって識別することができる。

【０００５】

これらの条件（ならびにこれらの条件の他の組み合わせ）が重なり合って、連続システムでは思いもよらない複雑さが生まれる可能性がある。例えば、アプリケーションホストは、共有する状態永続化システムにおいて永続化状態に競って修正を適用しようとするいくつかの同様に機能するプログラムの１つである場合がある。このようなシステムには本来備わっている競合が存在する場合があり、これらの修正の一部分は競合するかもしれない。さらに、これらの複雑さが原因で、また綿密で正しいブックキーピングを用いても、アプリケーションホストは、それが自らと競合すると判定する可能性がある。

【発明の概要】

【０００６】

本発明は、ロック競合を解決するための方法、システム、およびコンピュータプログラム製品に及ぶ。本発明の実施形態は、インスタンスストアに格納されたインスタンスをロックしようと試みる２つ以上の実行スレッド間のロック競合を解決する方法を含む。ある実施形態において、永続化プロバイダは、第１のコマンドを、アプリケーションホストに含まれる第１の実行スレッドから受け取る。第１のコマンドは、１つまたは複数の条件が満たされる場合に第１のインスタンスに対するロックの取得を要求するように構成される。永続化プロバイダは、第１のコマンドをコマンドログに記録する。永続化プロバイダは、第１のコマンドをインスタンスストアにサブミットする。

【０００７】

永続化プロバイダは、第２のコマンドを、アプリケーションホストに含まれる第２の実行スレッドから受け取る。第２のコマンドは、第２のコマンドにより処理する第２のインスタンスを第２の実行スレッドに対してロックすることを要求するように構成される。永続化プロバイダは、第２のコマンドをコマンドログに記録する。永続化プロバイダは、第２のコマンドをインスタンスストアにサブミットする。

【０００８】
永続化プロバイダは、第２のコマンドに対するロック応答を、インスタンスストアから受け取る。ロック応答は、アプリケーションホストが第２のインスタンスに対するロックの保持者であることを示す。ロック応答を、第１のコマンドをサブミットした後、かつ第１のコマンドが完了する前に受け取る。
【０００９】
永続化プロバイダは、コマンドログを参照し、第１のコマンドの現在の解決が、（ａ）第１のコマンドが第１のインスタンスに対するロックを取得したかどうか、ならびに（ｂ）第１のインスタンスおよび第２のインスタンスが同一のインスタンスであるかどうかを判定するには、十分な情報を提供していないと判定する。十分な情報ではないために、第２のコマンドが要求したロックがそれ以前に第１のコマンドによって取得されたロックであったかどうかに関して、曖昧となる。永続化プロバイダは、第１のコマンドのさらなる解決に到達するまで、第２のコマンドの処理を中断する。第１のコマンドのさらなる解決により、少なくとも、１つまたは複数の条件が満たされていることに関する追加情報を提供する。永続化プロバイダは、追加情報に基づき、第１のコマンドは第２のコマンドが要求したロックを取得したと判定する。永続化プロバイダは、第１のコマンドがそのロックをすでに取得したという判定に応じて、第２のコマンドを失敗とする。
【００１０】
他の実施形態においては、永続化プロバイダは、第１のコマンドを、アプリケーションホストに含まれる第１の実行スレッドから受け取る。第１のコマンドは、第１のインスタンスに対するロックの取得を要求するように構成される。永続化プロバイダは、第１のコマンドをインスタンスストアにサブミットする。永続化プロバイダは、第１のコマンドに対するロック応答を、インスタンスストアから受け取る。ロック応答は、アプリケーションホストが、第１のインスタンスに対するロックを取得したことを示す。第１のインスタンスに対するロックは、第１のインスタンスバージョンに対するものである。
【００１１】
永続化プロバイダは、第２のコマンドを、アプリケーションホストに含まれる第２の実行スレッドから受け取る。第２のコマンドは、第２のコマンドにより処理する第２のインスタンスを第２の実行スレッドに対してロックすることを要求するように構成される。永続化プロバイダは、第２のコマンドをインスタンスストアにサブミットする。永続化プロバイダは、第２のコマンドに対するロック応答を、インスタンスストアから受け取る。ロック応答は、アプリケーションホストが、第２のインスタンスに対するロックを取得したことを示す。第２のインスタンスに対するロックは、第２のインスタンスバージョンに対するものである。
【００１２】
永続化プロバイダは、第１のインスタンスおよび第２のインスタンスが同一のインスタンスであると判定する。永続化プロバイダは、第２のインスタンスバージョンは、第１のインスタンスバージョンより新しいインスタンスバージョンであると判定する。永続化プロバイダは、第１のコマンドが旧インスタンスバージョンに対するロックを保持しているという判定に応じて、第１のコマンドを失敗とする。
【００１３】
本「発明の概要」は、以下の「発明を実施するための形態」でさらに説明する概念から選択して、簡略化した形式で紹介するものである。本「発明の概要」は、特許請求する主題の主要な特徴または重要な特徴を特定しようとするものでもなく、特許請求する主題の範囲を決定する助けとして用いようとするものでもない。
【００１４】
本発明のさらなる特徴および利点は、以下に続く説明に記載されており、一部はその説明から明らかであろう、または、本発明を実践することにより理解されよう。本発明の特徴および利点は、添付請求項において特に指摘した機器およびその組み合わせによって実現および取得することができる。本発明のこれらおよび他の特徴は、以下の説明および添

付請求項からより十分に明らかとなろう、または、以下に説明するように本発明を実践することにより理解されよう。

【図面の簡単な説明】

【００１５】

本発明の上述および他の利点ならびに特徴を得ることができる方法を説明するために、簡潔に上述した本発明のより詳しい説明を、添付図面に例示した本発明の具体的な実施形態を参照して提示する。これらの図面は、本発明の一般的な実施形態のみを示し、従って、本発明の範囲を制限するものとみなすべきではないという理解のもと、以下の添付図面を用いて、本発明をさらなる具体性および詳細をもって記述および説明する。

【００１６】

【図１】ロック競合の解決を促進するコンピュータアーキテクチャの例を示す図である。

【図２】ロック競合の解決を促進するコンピュータアーキテクチャの別の例を示す図である。

【図３】ロック競合を解決する方法の例の流れ図である。

【図４】ロック競合を解決する方法の別の例の流れ図である。

【発明を実施するための形態】

【００１７】

本発明は、ロック競合を解決する方法、システム、およびコンピュータプログラム製品に及ぶ。本発明の実施形態は、インスタンスストアに格納されたインスタンスをロックしようと試みる２つ以上の実行スレッド間のロック競合を解決する方法を含む。ある実施形態においては、永続化プロバイダは、第１のコマンドを、アプリケーションホストに含まれる第１の実行スレッドから受け取る。第１のコマンドは、１つまたは複数の条件が満たされる場合に第１のインスタンスに対するロックの取得を要求するように構成される。永続化プロバイダは、第１のコマンドをコマンドログに記録する。永続化プロバイダは、第１のコマンドをインスタンスストアにサブミットする。

【００１８】

永続化プロバイダは、第２のコマンドを、アプリケーションホストに含まれる第２の実行スレッドから受け取る。第２のコマンドは、第２のコマンドにより処理する第２のインスタンスが第２の実行スレッドに対してロックされることを要求するように構成される。永続化プロバイダは、第２のコマンドをコマンドログに記録する。永続化プロバイダは、第２のコマンドをインスタンスストアにサブミットする。

【００１９】

永続化プロバイダは、第２のコマンドに対するロック応答を、インスタンスストアから受け取る。ロック応答は、アプリケーションホストが第２のインスタンスに対するロックの保持者であることを示す。ロック応答を、第１のコマンドをサブミットした後、かつ第１のコマンドが完了する前に受け取る。

【００２０】

永続化プロバイダは、コマンドログを参照し、第１のコマンドの現在の解決が、（ａ）第１のコマンドが第１のインスタンスに対するロックを取得したかどうか、ならびに（ｂ）第１のインスタンスおよび第２のインスタンスが同一のインスタンスであるかどうかを判定するには、十分な情報を提供していないと判定する。十分な情報ではないために、第２のコマンドが要求したロックがそれ以前に第１のコマンドによって取得されたロックであったかどうかに関して、曖昧となる。永続化プロバイダは、第１のコマンドのさらなる解決に到達するまで、第２のコマンドの処理を中断する。第１のコマンドのさらなる解決により、少なくとも、１つまたは複数の条件が満たされていることに関する追加情報を提供する。永続化プロバイダは、追加情報に基づき、第１のコマンドは第２のコマンドが要求したロックを取得したと判定する。永続化プロバイダは、第１のコマンドがそのロックをすでに取得したという判定に応じて、第２のコマンドを失敗とする。

【００２１】

他の実施形態においては、永続化プロバイダは、第１のコマンドを、アプリケーション

ホストに含まれる第１の実行スレッドから受け取る。第１のコマンドは、第１のインスタンスに対するロックの取得を要求するように構成される。永続化プロバイダは、第１のコマンドをインスタンスストアにサブミットする。永続化プロバイダは、第１のコマンドに対するロック応答を、インスタンスストアから受け取る。ロック応答は、アプリケーションホストが、第１のインスタンスに対するロックを取得したことを示す。第１のインスタンスに対するロックは、第１のインスタンスバージョンに対するものである。

【００２２】

永続化プロバイダは、第２のコマンドを、アプリケーションホストに含まれる第２の実行スレッドから受け取る。第２のコマンドは、第２のコマンドにより処理する第２のインスタンスを第２の実行スレッドに対してロックすることを要求するように構成される。永続化プロバイダは、第２のコマンドをインスタンスストアにサブミットする。永続化プロバイダは、第２のコマンドに対するロック応答を、インスタンスストアから受け取る。ロック応答は、アプリケーションホストが、第２のインスタンスに対するロックを取得したことを示す。第２のインスタンスに対するロックは、第２のインスタンスバージョンに対するものである。

【００２３】

永続化プロバイダは、第１のインスタンスおよび第２のインスタンスが同一のインスタンスであると判定する。永続化プロバイダは、第２のインスタンスバージョンは、第１のインスタンスバージョンより新しいインスタンスバージョンであると判定する。永続化プロバイダは、第１のコマンドが旧インスタンスバージョンに対するロックを保持しているという判定に応じて、第１のコマンドを失敗とする。

【００２４】

本発明の実施形態は、例えば、以下でより詳細に論ずるように、１つまたは複数のプロセッサおよびシステムメモリなど、コンピュータハードウェアを含む専用もしくは汎用のコンピュータを備え、または利用することができる。本発明の範囲内にある実施形態はまた、コンピュータ実行可能命令および／もしくはデータ構造を搬送または格納する物理的および他のコンピュータ可読媒体を含む。このようなコンピュータ可読媒体は、汎用または専用のコンピュータシステムによりアクセス可能な任意の利用可能な媒体とすることができる。コンピュータ実行可能命令を格納するコンピュータ可読媒体は、物理的記憶媒体である。コンピュータ実行可能命令を搬送するコンピュータ可読媒体は、伝送媒体である。従って、本発明の実施形態は、少なくとも２つの明らかに異なる種類のコンピュータ可読媒体、すなわちコンピュータ記憶媒体およびコンピュータ伝送媒体を備えることができるが、これらは例であって限定するものではない。

【００２５】

コンピュータ記憶媒体には、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、ＣＤ－ＲＯＭもしくは他の光ディスクストレージ、磁気ディスクストレージもしくは他の磁気記憶装置、または、所望のプログラムコード手段をコンピュータ実行可能命令もしくはデータ構造の形式で記憶するために使用可能、かつ汎用もしくは専用コンピュータによりアクセス可能な他の任意の媒体を含む。

【００２６】

「ネットワーク」を、コンピュータシステムおよび／またはモジュールおよび／または他の電子装置間の電子データの移送を可能にする１つまたは複数のデータリンクとして、定義づける。情報が、ネットワークまたは別の通信接続（有線、無線、または有線および無線の組み合わせのいずれか）を介して、コンピュータに転送または与えられる場合、コンピュータはその接続を適切に伝送媒体とみなす。伝送媒体には、所望のプログラムコード手段をコンピュータ実行可能命令もしくはデータ構造の形式で搬送するために使用可能、かつ汎用もしくは専用コンピュータによりアクセス可能なネットワークおよび／またはデータリンクを含むことができる。上記の組み合わせもまた、コンピュータ可読媒体の範囲に含まれるべきである。

【００２７】

さらに、コンピュータ実行可能命令またはデータ構造の形式のプログラムコード手段が、様々なコンピュータシステムのコンポーネントに到達すると、それらを伝送媒体からコンピュータ記憶媒体（またはその逆も同様）に自動的に転送することができる。例えば、ネットワークまたはデータリンクを介して受け取ったコンピュータ実行可能命令またはデータ構造を、ネットワークインタフェースモジュール（例えば、「ＮＩＣ」）内のＲＡＭにバッファリングして、最終的にコンピュータシステムのＲＡＭ、および／またはコンピュータシステムにおける揮発性の低いコンピュータ記憶媒体に転送することができる。従って、当然のことながら、コンピュータ記憶媒体を、伝送媒体をも利用する（または主として利用する）コンピュータシステムのコンポーネントに含むことができる。

【００２８】

コンピュータ実行可能命令は、例えば、プロセッサにおいて実行される場合、汎用コンピュータ、専用コンピュータ、または専用の処理装置に、ある特定の機能もしくは機能群を実施させる命令およびデータを備える。コンピュータ実行可能命令は、例えば、バイナリ、アセンブリ言語といった中間フォーマットの命令、またはソースコードとすることができる。本主題を構造的特徴および／または方法論的動作に固有の言語で説明したが、当然のことながら、添付請求項で定義する主題は、必ずしも、上述の特徴または動作に限るものではない。むしろ、上述の特徴および動作は、請求項を実装する形式の例として開示されるものである。

【００２９】

本発明を、多くのタイプのコンピュータシステム構成を有するネットワークコンピューティング環境において実践することができることは、当業者には認識されよう。そのコンピュータシステム構成には、パーソナルコンピュータ、デスクトップコンピュータ、ラップトップコンピュータ、メッセージプロセッサ、ハンドヘルド装置、マルチプロセッサシステム、マイクロプロセッサベースまたはプログラム可能な家庭用電化製品、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、携帯電話、ＰＤＡ、ページャ、ルータ、スイッチ等が含まれる。本発明を、ネットワークを介してリンクされた（有線データリンク、無線データリンク、または有線データリンクおよび無線データリンクの組み合わせのいずれかによって）ローカルおよびリモートのコンピュータシステムの両方がタスクを実施する分散システム環境において実践することができる。分散システム環境において、プログラムモジュールを、ローカルおよびリモートのメモリ記憶装置両方に配置することができる。

【００３０】

図１は、ロック競合の解決を促進するコンピュータアーキテクチャ１００の例を示す。図１を参照すると、コンピュータアーキテクチャ１００は、アプリケーションホスト１０１、永続化プロバイダ１０３、およびインスタンスストア１０７を含む。図示したコンピュータシステムのそれぞれは、例えば、ＬＡＮ（Ｌｏｃａｌ Ａｒｅａ Ｎｅｔｗｏｒｋ）、ＷＡＮ（Ｗｉｄｅ Ａｒｅａ Ｎｅｔｗｏｒｋ）、およびインターネットといったネットワークを介して互いに接続されている（または、ネットワークの一部である）。従って、図示したコンピュータシステムのそれぞれ、ならびに他の任意の接続されたシステムおよびそのコンポーネントは、メッセージ関連データを作成し、メッセージ関連データ（例えば、ＩＰ（Ｉｎｔｅｒｎｅｔ Ｐｒｏｔｏｃａｌ）データグラム、およびＩＰデータグラムを利用する、ＴＣＰ（Ｔｒａｎｓｍｉｓｓｉｏｎ Ｃｏｎｔｒｏｌ Ｐｒｏｔｏｃｏｌ）、ＨＴＴＰ（Ｈｙｐｅｒｔｅｘｔ Ｔｒａｎｓｆｅｒ Ｐｒｏｔｏｃｏｌ）、ＳＭＴＰ（Ｓｉｍｐｌｅ Ｍａｉｌ Ｔｒａｎｓｆｅｒ Ｐｒｏｔｏｃｏｌ）等といった他のより高層のプロトコル）を、ネットワークを介して交換することができる。

【００３１】

アプリケーションホスト１０１は、実行スレッド１０２Ａおよび１０２Ｂを含む複数の実行スレッドを備える。一般に、実行スレッドは、インスタンスストア１０７内のインスタンスと相互作用するためのコマンド（例えば、永続化コマンド）を発行するように構成される。アプリケーションホスト１０１は、例えばプログラムまたは管理ツールといった

エンティティとすることができ、実行スレッドは、インスタンスストア１０７に格納されたインスタンスを操作するアプリケーションホスト１０１内のモジュールとすることができる。

【００３２】

アプリケーションホスト１０１内で、実行スレッド１０２Ａ、１０２Ｂ等は、長期運用のアプリケーションのプログラムシーケンスに対応することができる。しかしながら、実行スレッドが、オペレーティングシステムにより与えられた計算スレッドに直接対応しない場合がある。例えば、連続ポイントからの非同期の実行または再開を支えているアプリケーションホストのランタイムにおいては、実行スレッドの計算スレッドへのバインディングは動的かつ過渡的である場合がある。従って、アプリケーションホストの実行スレッドの一部または全てが、１つまたは複数の計算スレッド上で、同時に実行している場合がある。

【００３３】

実行スレッド１０２Ａおよび１０２Ｂを含むアプリケーションホスト１０１の各実行スレッドは、アプリケーションホスト１０１の機能の多くを実施するように構成される。例えば、実行スレッドは、永続化プロバイダからのインスタンスストアに対するインスタンスハンドルを要求し、インスタンス永続化コマンドプロセッサにコマンドをサブミットし、サブミットしたコマンドに応じた永続化したアプリケーションの状態を受け取るよう構成される。従って、一般にアプリケーションホストによって実施されている機能に言及する場合、アプリケーションホストの実行スレッドによって実施される機能を含むことができる。

【００３４】

インスタンスストア１０７は、例えばインスタンス１０８Ａおよび１０８Ｂといったインスタンスに、永続性ストレージを提供する。インスタンスストア１０７は、インスタンスと相互作用するコマンド（例えば、永続化コマンド）を受け取り、処理するように構成される。インスタンスストア１０７は、永続化コマンドに応じて、アプリケーションの状態を永続化する、ならびに要求しているエンティティ（例えば、実行スレッド）に永続化状態を戻すことができる。

【００３５】

一般に、永続化プロバイダ１０３は、実行スレッドから受け取った永続化コマンドを、インスタンスストア１０７に適合するコマンドに翻訳するように構成される。例えば、永続化プロバイダ１０３は、実行スレッド１０２Ａまたは１０２Ｂからの永続化コマンドを、インスタンスストア１０７に適合する永続化コマンドに翻訳することができる。

【００３６】

図示したように、永続化プロバイダ１０３は、インスタンス永続化コマンドプロセッサ１０４、インスタンスストアドライバ１０６、コマンドクロック１０８、およびコマンドログ１０９を含む。インスタンス永続化コマンドプロセッサ１０４は、アプリケーションホスト１０１とインスタンスストア１０７との間の契約を定義する。そのため、インスタンス永続化コマンドプロセッサ１０４は、アプリケーションホスト１０１がインスタンスストア１０７を修正または検査するコマンドを与えることを可能にする、アプリケーションホスト１０１とインスタンスストア１０７との間のインタフェースである。例えば、ＳＱＬサーバを用いて実装されたインスタンス永続化コマンドプロセッサ１０４およびインスタンスストア１０７の組み合わせは、ＳＱＬ永続化プロバイダと呼ばれる得る。永続化プロバイダは、インスタンス永続化コマンドプロセッサ１０４によって定義された１組の許容できる状態修正を用いたホストのコマンドに従って、インスタンスストアの状態を修正する。

【００３７】

一般に、コマンドログ１０９は、アプリケーションホストがサブミットしたコマンドで、コマンドの結果および／または効果がまだアプリケーションホストに表示されていないコマンドを追跡するように構成される。コマンドログ１０９で追跡されたコマンドは、「

インフライト」コマンドと呼ぶことができる。

【００３８】

コマンドクロック１０８は、アプリケーションホストの実行スレッドによるコマンドの提出および／またはコマンドの結果の受け取りの間で、部分的に因果関係のある順序を作成するために用いられる、単調に増えていくカウンタである。

【００３９】

従って、コマンドクロック１０８は、永続化プロバイダ１０３に対して論理時間を保持するように構成される。必要に応じて、永続化プロバイダ１０３で受け取ったコマンドに、コマンドクロック１０８による時刻を用いてタイムスタンプを付して、コマンドログ１０９に記録することができる。そのため、コマンドクロック１０８は、コマンドを受け取った順序の時間的理解を促進する。インスタンス永続化コマンドプロセッサ１０４は、ロック競合（例えば、アプリケーションホスト１０１の競合する実行スレッド間の）を解決する場合、この時間的理解を（他の情報と共に）用いることができる。

【００４０】

コマンドには、一部をコマンドログ１０９に記録するものもあるが、全てを記録するわけではないかもしれない。コマンドをコマンドログ１０９に記録するか否かを判定する場合、永続化プロバイダ１０３は、ロック競合を引き起こす可能性のあるコマンドとロック競合を引き起こす可能性のないコマンドを見分けることができる。ロック競合を引き起こす可能性のあるコマンドは、コマンドログ１０９に記録される。一方、ロック競合を引き起こす可能性のないコマンドは、記録せずに実行させることができる。

【００４１】

一部のコマンドには、それが明らかにロックの取得を何ら要求しないタイプのコマンドであることに起因して、ロック競合を引き起こす可能性がないと判定できるものがある。例えば、インスタンスへの非連携の読取りアクセスを要求するコマンドは、他のコマンドと競合する可能性がたとえあったとしても少なく、従って、（これらのコマンドは、ロックの取得を要求しないので）記録せずに実行させることができる。

【００４２】

一方、ロック競合を引き起こす可能性があると判定されるコマンドは、コマンドログ１０９に記録することができる。コマンドを受け取った際、インスタンス永続化コマンドプロセッサ１０４は、そのコマンドがすでにロックされているインスタンスに対してロックの取得を要求するものであるかどうか判定するには十分な情報を有していない場合がある。例えば、インスタンスは、様々な異なるエイリアスを用いて識別可能であるかもしれない。したがって、インスタンスのエイリアスを含むコマンドを受け取ると、そのインスタンスのエイリアスがどのインスタンスを指すのか、簡単にはわからないかもしれない。そのため、インスタンス永続化コマンドプロセッサ１０４は、さらなる情報（インスタンスハンドルに対するエイリアスの解決）を取得するまで、そのコマンドを、ロック競合を引き起こす可能性がないコマンドとして分類することはできない。

【００４３】

コマンドがロックの取得を要求するか否かは、コマンドに関連する様々な条件が満たされているかどうかによることができる。

【００４４】

永続化プロバイダ１０３の内部および外部（例えば、時刻、日付等）両方の情報を利用し、関連条件が満たされているかどうかを判定することができる。さらに、永続化プロバイダ１０３は、インスタンス永続化コマンドプロセッサ１０４においてコマンドを受け取る前または受け取った後を含む様々な時点で、異なる関連条件に関する情報を認識することができる。従って、追加情報が利用可能になると、検出されたロック競合を引き起こす可能性を取り除くことができる。例えば、水曜日ならばロックの取得を要求するコマンドを、最初に、コマンドログ１０９に記録することができる。しかし、永続化プロバイダ１０３は、その日が木曜日であることを認識すると、コマンドをコマンドログ１０９から取り除くことができる。

【００４５】
ある実施形態においては、コマンド自体が、ロックの取得を要求する可能性を示す情報を含むことができる。そのため、インスタンス永続化コマンドプロセッサ１０４は、そのコマンドがロックの取得を要求している可能性があるかどうかを、コマンドに問い合わせることができる。他の実施形態においては、明らかにロックの取得を要求しないコマンドのリストを保持する。コマンドを受け取ると、インスタンス永続化コマンドプロセッサ１０４は、そのリストを参照することができる。
【００４６】
一般に、インスタンスストアドライバ１０６は、インスタンスストア１０７に適合する通信に必要な場合、コマンドを分解するように構成される。例えば、アプリケーションホストのコマンドセットが、インスタンスストアのコマンドセットのある特定のコマンドを欠いている場合がある。しかしながら、インスタンスストアのコマンドセットの２つ以上のコマンドの組み合わせを用いて、アプリケーションホストのコマンドセットのコマンドを実現することができるかもしれない。従って、永続化プロバイダ１０３は、受け取った永続化コマンドがインスタンスストアのコマンドセットに含まれていないことを検出した場合、インスタンスストアドライバ１０６を参照して、そのコマンドを他の適合するコマンドに分解することができる。
【００４７】
時々、実行スレッド１０２Ａ、１０２Ｂ等は、永続化プロバイダ１０３にコマンドをサブミットすることができる。必要に応じて、受け取ったコマンドには、コマンドクロック１０８によってタイムスタンプを付し、コマンドログ１０９に格納することができる。
【００４８】
図３は、２つ以上の実行スレッド間のロック競合を解決する方法３００の例の流れ図を示す。方法３００を、コンピュータアーキテクチャ１００のコンポーネントおよびデータに関して説明する。
【００４９】
方法３００は、永続化プロバイダ（例えば、永続化プロバイダ１０３）が第１のコマンドをアプリケーションホストに含まれる第１の実行スレッドから受け取る動作であって、第１のコマンドは、１つまたは複数の条件が満たされる場合に第１のインスタンスに対するロックの取得を要求するように構成される動作（動作３０１）を含む。例えば、インスタンス永続化コマンドプロセッサ１０４が、条件１３３を含むコマンド１１１を実行スレッド１０２Ａから受け取ることができる。コマンド１１１は、条件１３３が満たされる場合にインスタンス１０８Ａに対するロックの取得を要求するように構成される。条件１３３は、例えば、「今日が水曜日ならば、ロックをかけよ。」というような条件文を表すことができる。
【００５０】
方法３００は、永続化プロバイダが第１のコマンドをコマンドログに記録する動作（動作３０２）を含む。例えば、コマンド１１１を受け取った際、インスタンス永続化コマンドプロセッサ１０４が、コマンド１１１がロック競合を引き起こす可能性がないと確信を持って判定するには十分な情報を有していない場合がある（例えば、インスタンス永続化コマンドプロセッサ１０４は、今日が水曜日かどうか知らない場合がある）。そのため、インスタンス永続化コマンドプロセッサ１０４は、コマンド１１１をコマンドログ１０９に記録する。ある実施形態においては、コマンドログ１０９への入力は、コマンドおよび、コマンドクロック１０８からのタイムスタンプを含む。例えば、入力１３１は、コマンド１１１および時刻１２１を含む。
【００５１】
方法３００は、永続化プロバイダが第１のコマンドをインスタンスストアにサブミットする動作（動作３０３）を含む。例えば、インスタンスストアドライバ１０６が、コマンド１１１をインスタンスストア１０７にサブミットすることができる。
【００５２】

方法３００は、永続化プロバイダが第２のコマンドをアプリケーションホストに含まれる第２の実行スレッドから受け取る動作であって、第２のコマンドは、第２のコマンドにより処理する第２のインスタンスを第２の実行スレッドに対してロックすることを要求するように構成される動作（動作３０４）を含む。例えば、インスタンス永続化コマンドプロセッサ１０４が、コマンド１１２を実行スレッド１０２Ｂから受け取る。コマンド１１２は、コマンド１１２により処理する第２のインスタンスを実行スレッド１０２Ｂに対してロックすることを要求するように構成される。

【００５３】

方法３００は、永続化プロバイダが第２のコマンドをコマンドログに記録する動作（動作３０５）を含む。例えば、コマンド１１２を受け取った際、インスタンス永続化コマンドプロセッサ１０４が、コマンド１１２がロック競合を引き起こす可能性がないと確信を持って判定するには十分な情報を有していない場合がある。例えば、インスタンス永続化コマンドプロセッサ１０４は、コマンド１１２のエイリアスがロックの取得がすでに要求されているインスタンスを指すかどうか判定するための情報を欠いている場合がある。そのため、インスタンス永続化コマンドプロセッサ１０４は、コマンド１１２および時刻１２２（時刻１２１より後の時刻）を含む入力１３２を、コマンドログ１０９に記録する。

【００５４】

方法３００は、永続化プロバイダが第２のコマンドをインスタンスストアにサブミットする動作（動作３０６）を含む。例えば、インスタンスストアドライバ１０６が、コマンド１１２をインスタンスストア１０７にサブミットすることができる。

【００５５】

方法３００は、永続化プロバイダが第２のコマンドに対するロック応答をインスタンスストアから受け取る動作であって、ロック応答は、アプリケーションホストが第２のインスタンスに対するロックの保持者であることを示し、ロック応答は、第１のコマンドをサブミットした後、かつ第１のコマンドが完了する前に受け取られる動作（動作３０７）を含む。例えば、インスタンスストアドライバ１０６は、ロック応答１１４をインスタンスストア１０７から受け取ることができる。ロック応答１１４は、アプリケーションホスト１０１がインスタンス１０８Ａに対するロックの保持者であることを示すことができる。ロック応答１１４を、コマンド１１１をサブミットした後、かつコマンド１１１を完了する前に（例えば、インスタンスストアドライバ１０６は、ロック応答１１３をまだ受け取っていないかもしれない）受け取ることができる。

【００５６】

方法３００は、永続化プロバイダがコマンドログを参照し、第１のコマンドの現在の解決が、（ａ）第１のコマンドが第１のインスタンスに対するロックを取得したかどうか、ならびに（ｂ）第１のインスタンスおよび第２のインスタンスが同一のインスタンスであるかどうかを判定するには、十分な情報を提供していないと判定する動作であって、十分な情報ではないために、第２のコマンドが要求したロックがそれ以前に第１のコマンドによって取得されたロックであったかどうかに関して、曖昧となる動作（動作３０８）を含む。例えば、永続化プロバイダ１０３が、コマンド１１１がロックを取得したかどうか、および／またはコマンド１１１及び１１２が同一のインスタンスを指しているかどうかを判定するには十分な情報を有していない場合がある。十分な情報ではないために、コマンド１１２が要求したロックがそれ以前にコマンド１１１によって取得されたかどうかに関して、曖昧となる。

【００５７】

永続化プロバイダは、コマンドが明らかにインスタンス１０８Ａではないインスタンスのみを指していることにより、ロック競合を引き起こす可能性はないと判定できるコマンドログ内のコマンドを、除外することができる。例えば、インスタンス永続化コマンドプロセッサ１０４は、コマンドログ１０９内のコマンドが、インスタンス１０８Ａに対するエイリアスの何れでもないとわかる未解決のインスタンスのエイリアスを有していると判定することができる。

【００５８】
方法３００は、永続化プロバイダが、第１のコマンドのさらなる解決に到達するまで、第２のコマンドの処理を中断する動作であって、さらなる解決により、少なくとも、１つまたは複数の条件が満たされていることに関する追加情報が提供される動作（動作３０９）を含む。例えば、永続化プロバイダ１０３は、コマンド１１１のさらなる解決に到達するまで、コマンド１１２の処理を中断することができる。コマンド１１１のさらなる解決により、条件１３３が満たされていることに関する情報が提供されることが可能である。例えば、コマンド１１１が未解決のインスタンスのエイリアスに対するロックを要求する場合、永続化プロバイダ１０３は、インスタンスのエイリアスがある特定のインスタンスに対して解決されるまで、コマンド１１２の処理を中断することができる。さらに、永続化プロバイダ１０３は、コマンド１１１がインスタンス１０８Ａに対するロックを取得したこと（および従って、コマンド１１１のいずれのエイリアスもインスタンス１０８Ａを指すものであったこと）を示すロック応答１１３を受け取るかもしれない。

【００５９】
方法３００は、永続化プロバイダが、第１のコマンドの解決に関する追加情報に基づき、第２のコマンドに関してどのように進めるかを判定する動作を含む（動作３１０）。ある実施形態においては、永続化プロバイダが、追加情報に基づき、第１のコマンドは第２のコマンドが要求したロックを取得したと判定する。例えば、インスタンス永続化コマンドプロセッサ１０４は、コマンド１１１はインスタンス１０８Ａに対するロックを取得したと判定することができる。インスタンス永続化コマンドプロセッサ１０４はまた、コマンド１１２もまたインスタンス１０８Ａのロックを要求したので（しかしインスタンス１０８Ａのロックがすでにコマンド１１１によって取得された後に）、コマンド１１２はロック応答１１４を受け取ったと判定することができる。必要に応じて、インスタンス永続化コマンドプロセッサ１０４は、コマンドログ１０９を参照し、コマンド１１１をコマンド１１２より前に受け取ったと判定することができる。これらの実施形態においては、永続化プロバイダは、第１のコマンドがロックを取得したという判定に応じて、第２のコマンドを失敗とする。例えば、永続化プロバイダ１０３が、コマンド１１１がインスタンス１０８Ａに対するロックを取得したという判定に応じて、コマンド１１２を失敗とすることができる。

【００６０】
他の実施形態においては、永続化プロバイダは、追加情報に基づき、第１のコマンドは第２のコマンドが要求したロックを取得しなかったと判定する。例えば、インスタンス永続化コマンドプロセッサ１０４は、コマンド１１１はインスタンス１０８Ａに対するロックを取得しなかったと判定することができる。これらの他の実施形態においては、アプリケーションホストに、第２のコマンドに対して受け取ったロック応答が示すロックを無効にすることを試みるよう指示することができる。例えば、アプリケーションホスト１０１に、コマンド１１２に対するロック応答１１４に示されたロックを無効にすることを試みるよう指示することができる。

【００６１】
ロックを無効にする１つの理由は、ロックが誤ったロックであると判定される場合があるためである。例えば、残されたロックが、アプリケーションホストによって忘れさられた、以前の計算から残されたものである場合がある。したがって、第２のコマンドをインスタンスストアが受け取った時点で、そのロックは存在したが、アプリケーションホストは第２のコマンドの応答を受け取った際、そのロックの記録を有していない。第２のコマンドと競合する可能性のあるそれ以前に発行したコマンドの全てが、そのロックを取得していなかったことがわかった場合、誤ったロックが検出される可能性がある。

【００６２】
あるいは、永続化プロバイダ１０３は、アプリケーションホスト１０１に、すでにコマンド１１１に対してロックされているインスタンス１０８Ａのコピーを用いるよう指示することができる。

【００６３】
本発明の実施形態はまた、さらなるコマンドを含むロック競合の解決を含む。例えば、永続化プロバイダは、第３のコマンドを、アプリケーションホストに含まれる第３の実行スレッドから受け取ることができる。第３のコマンドは、１つまたは複数の条件を満たす場合、第３のインスタンスに対するロックの取得を要求するように構成可能である。永続化プロバイダは、第３のコマンドをコマンドログに記録することができる。
【００６４】
永続化プロバイダは、第３のコマンドをインスタンスストアにサブミットする。永続化プロバイダは、コマンドログを参照し、第３のコマンドの現在の解決が、（ａ）第３のコマンドが第３のインスタンスに対するロックを取得したかどうか、ならびに（ｂ）第３のインスタンスおよび第２のインスタンスが同一のインスタンスであるかどうかを判定するには、十分な情報を提供していないと判定する。十分な情報ではないために、第２のコマンド（例えば、コマンド１１２）が要求したロックが第３のコマンドによってそれ以前に取得されたロックであったかどうかに関して、曖昧となる。
【００６５】
永続化プロバイダは、第３のコマンドのさらなる解決に到達するまで、第２のコマンドの処理を中断する。さらなる解決により、少なくとも、１つまたは複数の条件を満たしていることに関する追加情報を提供する。永続化プロバイダは、第３のコマンドの解決に関する追加情報に基づき、第２のコマンドに関してどのように進めるかを判定する。
【００６６】
図２は、ロック競合の解決を促進するコンピュータアーキテクチャ２００の例を示す。図２を参照すると、コンピュータアーキテクチャ２００は、アプリケーションホスト２０１、永続化プロバイダ２０３、およびインスタンスストア２０７を含む。コンピュータアーキテクチャ１００と同様に、図示したコンピュータシステムのそれぞれは、例えば、ＬＡＮ（Ｌｏｃａｌ Ａｒｅａ Ｎｅｔｗｏｒｋ）、ＷＡＮ（Ｗｉｄｅ Ａｒｅａ Ｎｅｔｗｏｒｋ）、およびインターネットといったネットワークを介して互いに接続されている（または、ネットワークの一部である）。従って、図示したコンピュータシステムのそれぞれ、ならびに他の任意の接続されたシステムおよびそのコンポーネントは、メッセージ関連データを作成し、メッセージ関連データ（例えば、ＩＰ（Ｉｎｔｅｒｎｅｔ Ｐｒｏｔｏｃａｌ）データグラム、および、ＩＰデータグラムを利用する、ＴＣＰ（Ｔｒａｎｓｍｉｓｓｉｏｎ Ｃｏｎｔｒｏｌ Ｐｒｏｔｏｃｏｌ）、ＨＴＴＰ（Ｈｙｐｅｒｔｅｘｔ Ｔｒａｎｓｆｅｒ Ｐｒｏｔｏｃｏｌ）、ＳＭＴＰ（Ｓｉｍｐｌｅ Ｍａｉｌ Ｔｒａｎｓｆｅｒ Ｐｒｏｔｏｃｏｌ）等といった他のより高層のプロトコル）を、ネットワークを介して交換することができる。
【００６７】
図２において、図１と同様にラベル付けされたコンポーネントは、同様の機能を含む。例えば、アプリケーションホスト２０１は、実行スレッド２０２Ａ、および２０２Ｂを含む複数の実行スレッドを備える。一般に、実行スレッドは、インスタンスストア２０７内のインスタンスと相互作用するためのコマンド（例えば、永続化コマンド）を発行するように構成される。アプリケーションホスト２０１は、例えばプログラムまたは管理ツールといったエンティティとすることができ、実行スレッドは、インスタンスストア２０７に格納されたインスタンスを操作するアプリケーションホスト２０１内のモジュールとすることができる。
【００６８】
インスタンスストア２０７は、例えばインスタンス２０８といったインスタンスに、永続性ストレージを提供する。インスタンスストア２０７は、インスタンスと相互作用するためのコマンド（例えば、永続化コマンド）を受け取り、処理するように構成される。インスタンスストア２０７は、永続化コマンドに応じて、アプリケーションの状態を永続化する、ならびに要求しているエンティティ（例えば、実行スレッド）に永続化状態を戻すことができる。

【００６９】
一般に、永続化プロバイダ２０３は、実行スレッドから受け取った永続化コマンドを、インスタンスストア２０７に適合するコマンドに翻訳するように構成される。例えば、永続化プロバイダ２０３は、実行スレッド２０２Ａまたは２０２Ｂからの永続化コマンドを、インスタンスストア２０７に適合する永続化コマンドに翻訳することができる。
【００７０】
図示したように、永続化プロバイダ２０３は、インスタンス永続化コマンドプロセッサ２０４、インスタンスストアドライバ２０６、およびコマンドログ２０９を含む。インスタンス永続化コマンドプロセッサ２０４は、アプリケーションホスト２０１とインスタンスストア２０７との間の契約を定義する。そのため、インスタンス永続化コマンドプロセッサ２０４は、アプリケーションホスト２０１がインスタンスストア２０７を修正または検査するコマンドを与えることを可能にする、アプリケーションホスト２０１とインスタンスストア２０７との間のインタフェースである。例えば、ＳＱＬサーバを用いて実装されたインスタンス永続化コマンドプロセッサ２０４およびインスタンスストア２０７の組み合わせは、ＳＱＬ永続化プロバイダと呼ばれ得る。永続化プロバイダは、インスタンス永続化コマンドプロセッサ２０４によって定義された１組の許容できる状態修正を用いたホストのコマンドに従って、インスタンスストアの状態を修正する。
【００７１】
永続化プロバイダ１０３における機能と同様に、コマンドには、一部をコマンドログ２０９に記録するものもあるが、全てを記録するわけではないかもしれない。コマンドをコマンドログ２０９に記録するかどうかを判定する場合、永続化プロバイダ２０３は、ロック競合を引き起こす可能性のあるコマンドとロック競合を引き起こす可能性のないコマンドを見分けることができる。ロック競合を引き起こす可能性のあるコマンドは、コマンドログ２０９に記録される。一方、ロック競合を引き起こす可能性のないコマンドは、記録せずに実行させることができる。インスタンス永続化コマンドプロセッサ２０４は、コマンドプロセッサ１０４がコマンドをコマンドログ１０９に記録することに関して判定を下す方法と同様に、これらの判定を下すことができる。
【００７２】
一般に、インスタンスストアドライバ２０６は、インスタンスストア２０７に適合する通信に必要な場合、コマンドを分解するように構成される。例えば、アプリケーションホストのコマンドセットが、インスタンスストアのコマンドセットのある特定のコマンドを欠いている場合がある。しかしながら、インスタンスストアのコマンドセットの２つ以上のコマンドの組み合わせを用いて、アプリケーションホストのコマンドセットのコマンドを実現することができるかもしれない。従って、永続化プロバイダ２０３は、受け取った永続化コマンドがインスタンスストアのコマンドセットに含まれていないことを検出した場合、インスタンスストアドライバ２０６を参照して、そのコマンドを他の適合するコマンドに分解することができる。
【００７３】
図示したように、インスタンスストア２０７は、インスタンスクロック２８１を含む。インスタンスクロック２８１は、インスタンスストア２０７に格納されたインスタンスに対するバージョンを保持するように構成される。インスタンスクロック２８１は、バージョンをいつ更新すべきかを定義するバージョン更新規則に従って、バージョンを保持することができる。例えば、バージョン更新規則は、排他的なロックがインスタンスにかかった場合、排他的なロックがインスタンスに対して解除される都度、インスタンスに関連する永続化状態が修正された場合等に、インスタンスのバージョンを更新する（増やす）べきと、指示することができる。インスタンスのバージョンを保持するステップには、バージョンが更新されるべき時、カウンタを増やすステップを含むことができる。例えば、現在はバージョン３であるインスタンスからのデータを修正すると、インスタンスをバージョン４に増やすことができる。永続化プロバイダ２０３は、ロック競合を解決する際、インスタンスのバージョンを用いることができる。

【００７４】
時々、実行スレッド２０２Ａ、２０２Ｂ等は、コマンドを永続化プロバイダ２０３にサブミットすることができる。これもまた図示するように、他のコマンド／結果２１７を、インスタンスストア２０７および他のアプリケーションホスト間で（おそらく、他の中間永続化プロバイダを介して）伝達し合うことができる。従って、アプリケーションホスト２０１ならびに他のアプリケーションホストが、インスタンスストア２０７と相互作用することができ、インスタンスのバージョンを変更させる可能性がある。例えば、（アプリケーションホスト２０１および／または他のアプリケーションホストから）受け取ったコマンドに応じて、インスタンス２０８は、バージョン２６１（より初期のバーション）からバージョン２６３（より最近のバージョン）に遷移することができる。

【００７５】
図４は、２つ以上の実行スレッド間のロック競合を解決する方法４００の例の流れ図を示す。方法４００を、コンピュータアーキテクチャ２００のコンポートネントおよびデータに関して説明する。

【００７６】
方法４００は、永続化プロバイダが第１のコマンドをアプリケーションホストに含まれる第１の実行スレッドから受け取る動作であって、第１のコマンドは、第１のインスタンスに対するロックの取得を要求するように構成される動作（動作４０１）を含む。例えば、インスタンス永続化コマンドプロセッサ２０４が、コマンド２１１を実行スレッド２０２Ａから受け取ることができる。コマンド２１１を、インスタンス２０８に対するロックの取得を要求するように構成することができる。

【００７７】
方法４００は、永続化プロバイダが第１のコマンドをインスタンスストアにサブミットする動作（動作４０２）を含む。例えば、インスタンスストアドライバ２０６が、コマンド２１１をインスタンスストア２０７にサブミットすることができる。

【００７８】
方法４００は、永続化プロバイダが第１のコマンドに対するロック応答をインスタンスストアから受け取る動作であって、ロック応答は、アプリケーションホストが第１のインスタンスに対するロックを取得したことを示し、第１のインスタンスに対するロックは、第１のインスタンスバージョンに対するものである動作（動作４０３）を含む。例えば、インスタンスストアドライバ２０６は、ロック応答２１３をインスタンスストア２０７から受け取ることができる。ロック応答２１３は、コマンド２１１がインスタンス２０８のバージョン２６１をロックしたことを示す。インスタンス永続化プロバイダ２０４は、入力２３１をコマンドログ２０９に記録することができる。図示するように、入力２３１は、コマンド２１１をロック応答２１３に関連付ける。

【００７９】
方法４００は、永続化プロバイダが第２のコマンドをアプリケーションホストに含まれる第２の実行スレッドから受け取る動作であって、第２のコマンドは、第２のコマンドにより処理する第２のインスタンスが第２の実行スレッドに対してロックされることを要求するように構成される動作（動作４０４）を含む。例えば、インスタンス永続化コマンドプロセッサ２０４は、コマンド２１２を実行スレッド２０２Ｂから受け取ることができる。コマンド２１２を、インスタンス２０８に対するロックの取得を要求するように構成することができる。

【００８０】
方法４００は、永続化プロバイダが第２のコマンドをインスタンスストアにサブミットする動作（動作４０５）を含む。例えば、インスタンスストアドライバ２０６は、コマンド２１２をインスタンスストア２０７にサブミットすることができる。

【００８１】
方法４００は、永続化プロバイダが第２のコマンドに対するロック応答をインスタンスストアから受け取る動作であって、ロック応答は、アプリケーションホストが第２のイン

スタンスに対するロックを取得したことを示し、第２のインスタンスに対するロックは、第２のインスタンスバージョンに対するものである動作（動作４０６）を含む。例えば、インスタンスストアドライバ２０６は、ロック応答２１４をインスタンスストア２０７から受け取ることができる。ロック応答２１４は、コマンド２１２がインスタンス２０８のバージョン２６３をロックしたことを示す。

【００８２】

前述のように、他のアプリケーションホストは、（他のコマンド／結果２１７で示すように）インスタンスストア２０７と相互作用することができる。従って、コマンド２１１および２１２を、インスタンスストア２０７で受け取った（他のアプリケーションホストからの）他のコマンドに組み入れることができる。様々な要因の内の何れかによって、インスタンスストア２０７は、アプリケーションホストが以前に取得したロックを解除する可能性がある。例えば、通信の失敗により、インスタンスストア２０７は、もはや自らが実行スレッド２０２Ａ（および／または永続化プロバイダ２０３）と通信していないと検出する可能性がある。結果として、インスタンスストア２０７は、実行スレッド２０２Ａが以前取得したロックを解除する可能性がある。しかしながら、これもまた通信の失敗の結果として、アプリケーションホスト２０１および／または永続化プロバイダ２０３には、インスタンスストア２０７がロックを解除したことがわからない場合もある。従って、例えば、実行スレッド２０２Ａは、ロックが実際にはインスタンスストア２０７において既に解除されているにもかかわらず、ロック応答２１３に示されたロックをあたかも取得しているかのように継続する可能性がある。

【００８３】

ロックを解除した後に、別のアプリケーションホストが同一のインスタンスに対するロックを取得し、バージョンを更新させる可能性がある。例えば、他のコマンド／結果２１７におけるコマンドを介して、インスタンス２０８は、バージョン２６１（より初期のバージョン）からバージョン２６３（より最近のバージョン）に更新される可能性がある。さらに、実行スレッド２０２Ａがロック応答２１３に基づき継続しているが、実際にはインスタンス２０８に対してロックを有していない間に、この事態は起こる可能性がある。

【００８４】

方法４００は、永続化プロバイダが第１のインスタンスおよび第２のインスタンスは同一のインスタンスであると判定する動作（動作４０７）を含む。例えば、インスタンス永続化コマンドプロセッサ２０４は、ロック応答２１３および２１４は両方ともインスタンス２０８に対するロックの取得を要求したものであると判定することができる。

【００８５】

方法４００は、永続化プロバイダが、第２のインスタンスバージョンは第１のインスタンスバージョンより新しいインスタンスバージョンであると判定する動作（動作４０８）を含む。例えば、インスタンス永続化プロバイダ２０４は、バージョン２６１およびバージョン２６３を比較することができる。その比較から、インスタンス永続化コマンドプロセッサ２０４は、インスタンス２０８のバージョン２６３がインスタンス２０８のバージョン２６１より新しく、従ってバージョン２６１が旧式ものであると判定することができる。

【００８６】

方法４００は、第１のコマンドが旧式のインスタンスバージョンに対するロックを保持しているという判定に応じて、永続化プロバイダが第１のコマンドを失敗とする動作（動作４０９）を含む。例えば、インスタンス永続化コマンドプロセッサ２０４は、コマンド２１１がバージョン２６１に対するロックを保持しているという判定に応じて、コマンド２１１を失敗とすることができる。

【００８７】

コマンドを失敗とするステップは、そのコマンドが成功裏に完了したことを以前に示した場合に、実行スレッドまたはアプリケーションホストに通知するステップを含むことができる。例えば、ロック競合を検出する前に、インスタンス永続化コマンドプロセッサ２

０４は、コマンド２１１が正常に完了したことを実行スレッド２０２Ａに示すことができる。ロック競合を検出した後で、インスタンス永続化コマンドプロセッサ２０４は、コマンド２１１が失敗となったことを実行スレッド２０２Ａまたはアプリケーションホスト２０１に示すことができる。実行スレッド２０２Ａまたはアプリケーションホスト２０１は、コマンド２１１が失敗となったという通知に基づき、修正動作を行うことができる（例えば、実行スレッド２０２Ａがインスタンス２０８に対して実施している計算を停止することによって）。

【００８８】

上述のように、アプリケーションホストは、例えば、ロックを誤りであると判定する場合、ロックを無効にしようと試みることができる。ロックを無効にするステップは、ロックの取得の試みを繰り返すステップ、既存のロックをロック競合によって示された特定のロックのバージョンと共にアプリケーションホストが保持している場合、その既存のロックを潜在的に無視するステップを含むことができる。ロックは、第２のコマンドをインスタンスストアが受け取った後に解除することができるので、その後別のアプリケーションホスト、または同一のアプリケーションでさえ、ロック競合の解決中にそのインスタンスをロックしていることがあり得る。従って、ロックのバージョニングは、ロックの取得を試みている間に、第２のロック競合におけるロックが以前に第１のロック競合中に示されたロックとは異なることを検出する機構を提供する。

【００８９】

そのため、第２のロック競合は、（方法３００及び４００に関して説明したように）何れかのタイプのロック競合であり得る。さらにいずれかのタイプのロック競合が、第１のロック競合について知らずに、取り扱われる可能性がある。従って、連続的な試みは独立している。例えば、第２のコマンドを再試行する前に、そのインスタンスを再びロックした新しいコマンドへの応答を受け取ることさえあり得る。従って、新しいバージョンにおけるロックがすでに受け取られ、ロックの取得は、インスタンスストアに戻る必要なく失敗とすべきと判定される可能性がある。

【００９０】

本発明の実施形態はまた、方法３００及び４００において説明したものと同様の技術を同時に実装することによって、３つ以上のコマンド間のロック競合を解決するステップを含む。例えば、３つのコマンド間のロック競合を、ロックのバージョンの組み合わせを用いて、いつコマンドを受け取ったかを判定して、解決することができる。

【００９１】

従って、本発明の実施形態は、各永続化状態格納場所に対して、状態永続化システムによって保持された論理ロッククロックを採用することができる。ロック時間を、インスタンス永続化コマンドプロセッサによって実施されるブックキーピングに組み込んで、インスタンスが以前の論理時間にアプリケーションホストによってロックされている場合と、インスタンスがアプリケーションホストによって異なる名前で同時にロックされている場合を区別することができる。インスタンス永続化コマンドプロセッサはさらに、発行されたコマンドのどれがロックをかける可能性があるかを判定するために内観するとともに、アプリケーションホストが状態永続化システムに発行したコマンドに対して論理コマンドクロックを保持することができる。次いで、インスタンス永続化コマンドプロセッサは、ロック競合する可能性のあるコマンドの影響が明らかになるまでコマンドの実行を中断し、ロック時間を調べて永続化状態格納場所の複数のコピーの中から見分けることによって、競合を解決することができる。

【００９２】

例えば、本発明の実施形態を用いて、並行して行われた要求を再順序付けすることによって引き起こされたロック競合を解決することができる。インスタンス永続化コマンドプロセッサは、第１の実行スレッドから、インスタンスへ配信するための第１のアプリケーションメッセージを受け取ることができる。しかしながら、第１のアプリケーションメッセージの内容によっては、宛先のインスタンスが明確でない場合がある。例えば、第１の

アプリケーションメッセージが、インスタンスそれ自体に対する固有の識別子ではなく、インスタンスのデータの一部であるビジネス情報（注文番号等）を含んでいる場合がある。アプリケーションホストでさえ、第１のアプリケーションメッセージを正しい宛先のインスタンスへと解決できない場合がある。例えば、固有のインスタンス識別子を求めて第１のアプリケーションメッセージを調べ、またメッセージ内のビジネス情報に対応するデータを求めて以前にロードされたインスタンスを探しても、アプリケーションホストが適切なインスタンスを見つけられない場合がある。インスタンスストアは、メッセージ内の情報をあるインスタンスのデータと相関させることによって、第１のアプリケーションメッセージをある特定のインスタンスへと解決することができる場合がある。従って、アプリケーションホストは、インスタンスストアに適切なインスタンスをロードするように（または、適切なインスタンスがまだ存在しない場合には新しいインスタンスを作成するように）依頼する場合がある。

【００９３】

原則的に同時に、アプリケーションホストが、第２のアプリケーションメッセージを、第２の並行実行スレッドにおいて受け取る。第２のアプリケーションメッセージも同様に、宛先のインスタンスを一意的に特定していない場合がある。さらに、第２のアプリケーションメッセージが異なるビジネス情報を含んでいる場合がある。例えば、第２のアプリケーションメッセージは、注文番号を含むのではなく、顧客の配送先を含んでいる場合がある。従って、２つのアプリケーションメッセージは、同じ宛先のインスタンスを指すことがあるが、アプリケーションホストはこの関係性を検出することができない場合がある。そのため、第２の実行スレッドは、続いて、インスタンスストアに適切なインスタンスをロードするように（または、適切なインスタンスがまだ存在しない場合には新しいインスタンスを作成するように）依頼する。

【００９４】

再順序付けのため、第２の実行スレッドへの応答は、アプリケーションホストがすでに関連インスタンスをロックしているというものである可能性がある。しかしながら、アプリケーションホストは、さらにロックの通知を受け取る必要があるかもしれない。したがって、インスタンス永続化コマンドプロセッサがコマンドログを参照し、競合を解決することができる。

【００９５】

他の連続する事象もインスタンスストアによる同様の応答へつながる場合がある。例えば、アプリケーションホストは、実行スレッドをすでに停止しているか、以前にクラッシュしたことがあるため、インスタンスをロックしたという自らの記録を失っているものの、インスタンスストアは、アプリケーションホストがコピーを有しているといまだ認識している場合がある。同様の競合が、アプリケーションホストが１つの実行スレッドにおいてインスタンスを保存およびアンロードする一方で、別の実行スレッドが同時に同一のインスタンスをロードする場合に、発生する可能性がある。ロック競合を解決するには、アプリケーションホストおよびインスタンスストアの状態を常に調整するために、これらの様々な場合を見分けなければならない。本発明の実施形態を用いて、これら、ならびに他のさらなる連続する事象から起こるロック競合を解決することができる。

【００９６】

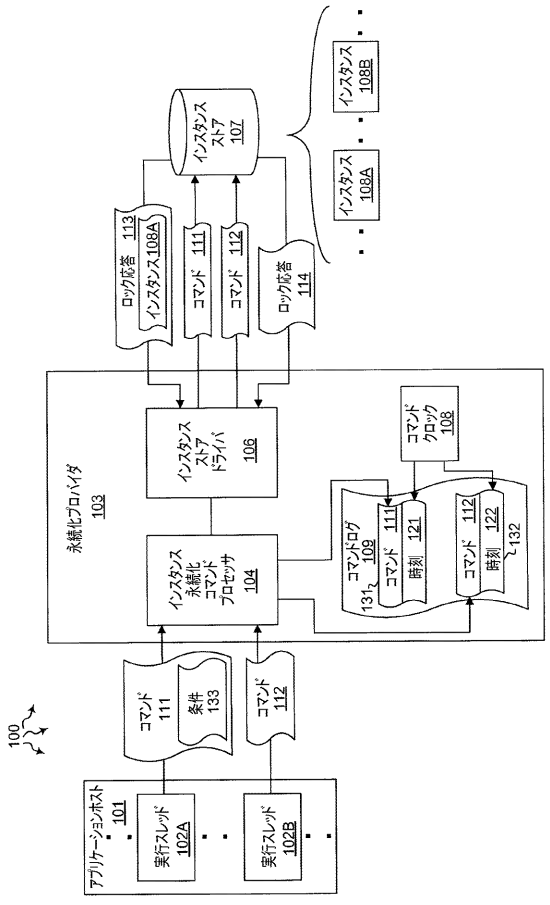
本発明の実施形態にはまた、１つまたは複数の永続化プロバイダを有するコンピュータアーキテクチャを含む。各永続化プロバイダは、複数のアプリケーションホストを設け、そのそれぞれが１つまたは複数の実行スレッドを備え、インスタンスストアに格納されたインスタンスにアクセスする。これらの実施形態においては、各インスタンス永続化プロバイダはコマンドクロックを含むことができ、インスタンスストアはインスタンスクロックを含むことができる。コマンドクロックから、およびインスタンスクロックから得られる情報を用いて、時間的な順序付けおよびバージョニングの両方に基づき、アプリケーションホスト間のロック競合を解決することができる。従って、本発明の実施形態は、コマンドが異なる時刻に受け取られ、インスタンスの異なるバージョンを指す場合の、複数の

アプリケーションホスト間のロック競合を解決することができる。

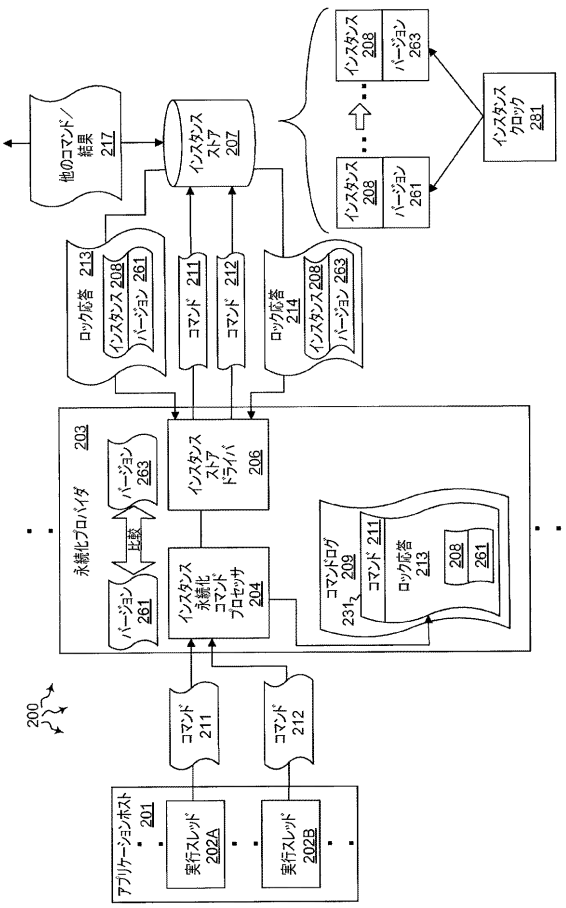
【００９７】

本発明を、本発明の趣旨または本質的特徴から逸脱することなく、他の特定の形式で具現化することができる。説明した実施形態は、あらゆる点で、単に例示であって制限するものではないとみなされるべきである。従って、本発明の範囲は、前述の説明によってではなく、添付請求項によって示すものである。請求項と同等の意味および範囲内にあるあらゆる変更は、請求項の範囲内に包含すべきものである。

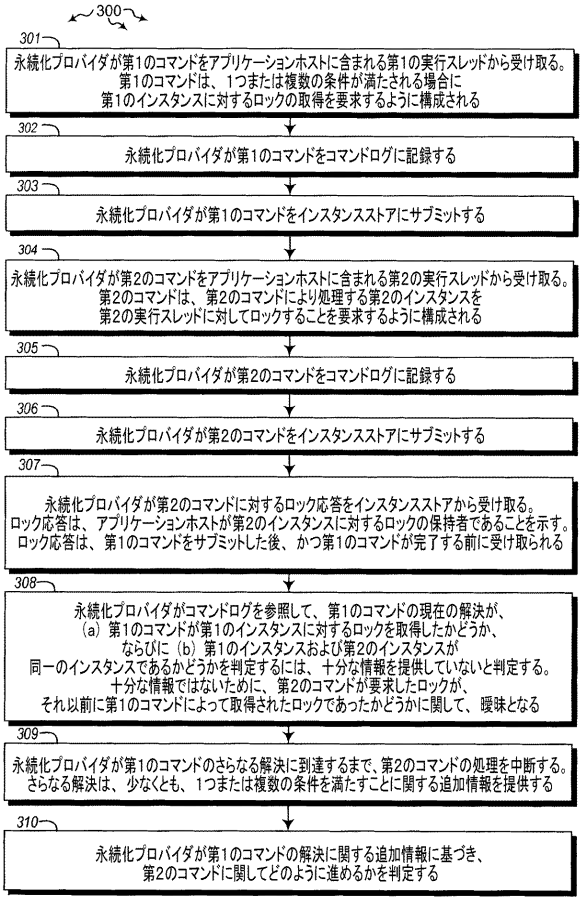
【図１】



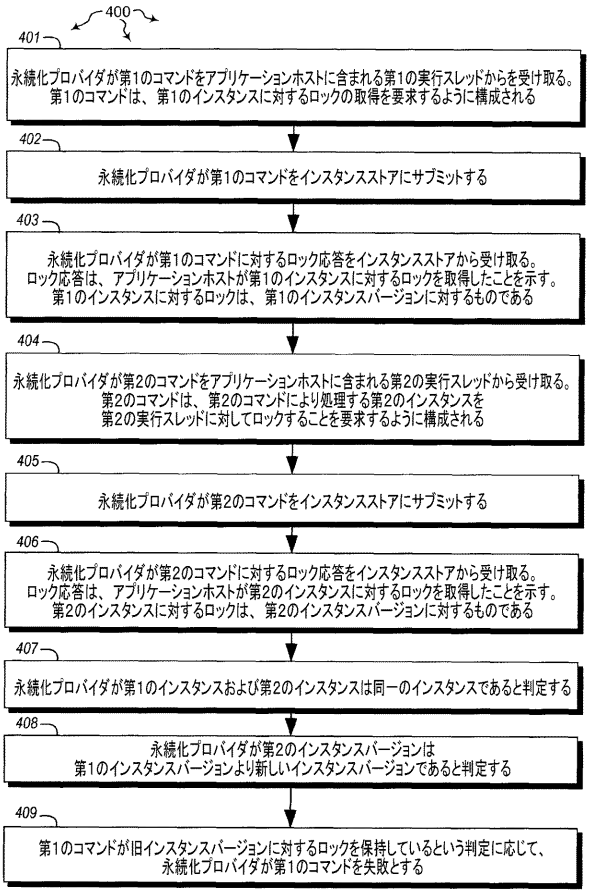
【図２】



【図３】


300
301
永続化プロバイダが第1のコマンドをアプリケーションホストに含まれる第1の実行スレッドから受け取る。
第1のコマンドは、1つまたは複数の条件が満たされる場合に
第1のインスタンスに対するロックの取得を要求するように構成される
302
永続化プロバイダが第1のコマンドをコマンドログに記録する
303
永続化プロバイダが第1のコマンドをインスタンスストアにサブミットする
304
永続化プロバイダが第2のコマンドをアプリケーションホストに含まれる第2の実行スレッドから受け取る。
第2のコマンドは、第2のコマンドにより処理する第2のインスタンスを
第2の実行スレッドに対してロックすることを要求するように構成される
305
永続化プロバイダが第2のコマンドをコマンドログに記録する
306
永続化プロバイダが第2のコマンドをインスタンスストアにサブミットする
307
永続化プロバイダが第2のコマンドに対するロック応答をインスタンスストアから受け取る。
ロック応答は、アプリケーションホストが第2のインスタンスに対するロックの保持者であることを示す。
ロック応答は、第1のコマンドをサブミットした後、かつ第1のコマンドが完了する前に受け取られる
308
永続化プロバイダがコマンドログを参照して、第1のコマンドの現在の解決が、
(a) 第1のコマンドが第1のインスタンスに対するロックを取得したかどうか、
ならびに (b) 第1のインスタンスおよび第2のインスタンスが
同一のインスタンスであるかどうかを判定するには、十分な情報を提供していないと判定する。
十分な情報ではないために、第2のコマンドが要求したロックが、
それ以前に第1のコマンドによって取得されたロックであったかどうかに関して、曖昧となる
309
永続化プロバイダが第1のコマンドのさらなる解決に到達するまで、第2のコマンドの処理を中断する。
さらなる解決は、少なくとも、1つまたは複数の条件を満たすことに関する追加情報を提供する
310
永続化プロバイダが第1のコマンドの解決に関する追加情報に基づき、
第2のコマンドに関してどのように進めるかを判定する

【図４】


400
401
永続化プロバイダが第1のコマンドをアプリケーションホストに含まれる第1の実行スレッドからを受け取る。
第1のコマンドは、第1のインスタンスに対するロックの取得を要求するように構成される
402
永続化プロバイダが第1のコマンドをインスタンスストアにサブミットする
403
永続化プロバイダが第1のコマンドに対するロック応答をインスタンスストアから受け取る。
ロック応答は、アプリケーションホストが第1のインスタンスに対するロックを取得したことを示す。
第1のインスタンスに対するロックは、第1のインスタンスバージョンに対するものである
404
永続化プロバイダが第2のコマンドをアプリケーションホストに含まれる第2の実行スレッドから受け取る。
第2のコマンドは、第2のコマンドにより処理する第2のインスタンスを
第2の実行スレッドに対してロックすることを要求するように構成される
405
永続化プロバイダが第2のコマンドをインスタンスストアにサブミットする
406
永続化プロバイダが第2のコマンドに対するロック応答をインスタンスストアから受け取る。
ロック応答は、アプリケーションホストが第2のインスタンスに対するロックを取得したことを示す。
第2のインスタンスに対するロックは、第2のインスタンスバージョンに対するものである
407
永続化プロバイダが第1のインスタンスおよび第2のインスタンスは同一のインスタンスであると判定する
408
永続化プロバイダが第2のインスタンスバージョンは
第1のインスタンスバージョンより新しいインスタンスバージョンであると判定する
409
第1のコマンドが旧インスタンスバージョンに対するロックを保持しているという判定に応じて、
永続化プロバイダが第1のコマンドを失敗とする

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 ニコラス エー．アレン
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
(72)発明者 ジャスティン ディー．ブラウン
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 篠塚 隆

(56)参考文献 特開２００１－６７２３８（ＪＰ，Ａ）
特開平１１－６５８６３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ９／４６
９／４８
９／５０－９／５２
９／５４
１７／３０