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

特許第4234008号
(P4234008)

(45)発行日　平成21年3月4日(2009.3.4)　(24)登録日　平成20年12月19日(2008.12.19)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 546K
G06F 13/00 (2006.01)　G06F 12/00 533J
G06F 13/00 540C

請求項の数 19　(全 13 頁)

(21)出願番号　特願2003-538936 (P2003-538936)
(86)(22)出願日　平成14年10月21日 (2002.10.21)
(65)公表番号　特表2005-507120 (P2005-507120A)
(43)公表日　平成17年3月10日 (2005.3.10)
(86)国際出願番号　PCT/US2002/033659
(87)国際公開番号　WO2003/036517
(87)国際公開日　平成15年5月1日 (2003.5.1)
審査請求日　平成16年4月26日 (2004.4.26)
(31)優先権主張番号　60/335,633
(32)優先日　平成13年10月25日 (2001.10.25)
(33)優先権主張国　米国(US)
(31)優先権主張番号　10/212,382
(32)優先日　平成14年8月5日 (2002.8.5)
(33)優先権主張国　米国(US)

前置審査

(73)特許権者　500105160
ビーイーエイ　システムズ，　インコーポレイテッド
ＢＥＡ　Ｓｙｓｔｅｍｓ，　Ｉｎｃ．
アメリカ合衆国　カリフォルニア　９５１３１，　サン　ノゼ，　ノース　ファースト　ストリート　２３１５
２３１５　Ｎｏｒｔｈ　Ｆｉｒｓｔ　Ｓｔｒｅｅｔ，　Ｓａｎ　Ｊｏｓｅ，　ＣＡＬＩＦＯＲＮＩＡ　９５１３１　Ｕ．Ｓ．Ａ．
(74)代理人　100082005
弁理士　熊倉　禎男
(74)代理人　100067013
弁理士　大塚　文昭

最終頁に続く

(54)【発明の名称】ビーン・キャッシュをフラッシュするためのシステムおよび方法

(57)【特許請求の範囲】

【請求項１】

データベースにおいてデータ項目のキャッシュされたコピーを更新する方法であって：

サーバ上にデータ項目のコピーを記憶するステップであって、前記データ項目はデータベースに記憶されるステップと；

前記コピーへの読み取りアクセスを提供するための読み取り専用ビーンを生成し、および前記読み取り専用ビーンに関連する識別情報を記憶するステップと；

前記データベースにおける前記データ項目への読み取りおよび書き込みアクセスを提供するために読み取り／書き込みビーンを生成し、および前記読み取り／書き込みビーンを用いて前記データ項目を更新するステップと；

前記データ項目のコピーを削除できるように、前記識別情報を用いて前記読み取り専用ビーンに無効リクエストを送信するステップと；

前記無効リクエスト内にバージョン識別子を含むステップと；

前記読み取り専用ビーンに関する無効リクエストを、前記バージョン識別子をチェックすることによって、怠ったかどうかを決定するステップと；および

前記読み取り専用ビーンが無効リクエストを怠っていた場合、無効メッセージが前記読み取り専用ビーンに再送信されることを要求するステップと

を含み、

前記読み取り専用ビーンに無効リクエストを送信することがさらに、前記識別情報によって識別された前記データ項目のコピーを記憶するあらゆる読み取り専用ビーンに前記リ

クエストをマルチキャストすることを伴うことを特徴とする方法。
【請求項２】
データ項目のコピーを記憶することが、第１のサーバ上のキャッシュに前記コピーを記憶することであることを特徴とする、請求項１に記載の方法。
【請求項３】
読み取り専用ビーンを生成することが、前記第１のサーバ上に前記読み取り専用ビーンを生成することであり；および
読み取り／書き込みビーンを生成することが、第２のサーバ上に前記読み取り／書き込みビーンを生成することである
ことを特徴とする、請求項２に記載の方法。
【請求項４】
無効リクエストを送信することが、前記読み取り／書き込みビーンを含む前記第２のサーバから、前記読み取り専用ビーンを含む前記第１のサーバへ前記リクエストを送信することであることを特徴とする、請求項３に記載の方法。
【請求項５】
読み取り専用ビーンを生成し、および識別情報を記憶することが、ＸＭＬドキュメントにおけるタグとして、前記識別情報を記憶することであることを特徴とする、請求項１に記載の方法。
【請求項６】
前記読み取り専用ビーンに無効リクエストを送信することがさらに、クラスタのあらゆるメンバに前記リクエストをマルチキャストすることを伴い、前記サーバは前記クラスタのメンバであることを特徴とする、請求項１に記載の方法。
【請求項７】
定期的に、前記無効リクエストを前記クラスタにマルチキャストすることをさらに含む、請求項６に記載の方法。
【請求項８】
前記読み取り専用ビーンに無効リクエストを送信することが、ポイント・ツウ・ポイント接続によって生じることを特徴とする、請求項１に記載の方法。
【請求項９】
無効リクエストを送信することは、バージョン番号を含むリクエストをマルチキャストすることを伴うことを特徴とする、請求項１に記載の方法。
【請求項１０】
最新の無効リクエストを記憶することをさらに含む、請求項１に記載の方法。
【請求項１１】
複数の更新に関する情報を収集し、および前記更新情報を一つの無効リクエストにまとめることをさらに含む、請求項１に記載の方法。
【請求項１２】
クラスタにおいてデータ項目のキャッシュされたコピーを更新するためのシステムであって：
前記クラスタにおける第１のサーバ上の前記データ項目の第１のキャッシュされたコピーと；
前記第１のキャッシュされたコピーへの読み取りアクセスを許可する第１の読み取り専用ビーンと；
前記クラスタにおける第２のサーバ上の前記データ項目の第２のキャッシュされたコピーと；
前記第２のキャッシュされたコピーへの読み取りアクセスを許可する第２の読み取り専用ビーンと；および
前記データ項目への書き込みアクセスを許可する読み取り／書き込みビーンであって、前記読み取り／書き込みビーンは、前記データ項目を更新するときに、前記第１および第２の読み取り専用ビーンへ無効リクエストを送信するようにプログラムされている読み取

り／書き込みビーンと
前記データ項目のコピーをキャッシュする前記クラスタにおける各読み取り専用ビーンの識別性を含む無効ターゲットとを含み、
前記無効リクエストがバージョン識別子を含み、
前記第１および第２の読み取り専用ビーンの各々は、リクエストが受信されなかった場合に、前記無効リクエストの再送信を要求するように適応しており；
前記読み取り専用ビーンに関する無効リクエストを怠ったかどうかを決定するために、前記バージョン識別子がチェックされ；および
前記読み取り専用ビーンが無効リクエストを怠っていた場合、無効メッセージが前記読み取り専用ビーンに再送信されることが要求される
システム。
【請求項１３】
前記データ項目を含むように適応したデータベース
をさらに含む、請求項１２に記載のシステム。
【請求項１４】
前記第１の読み取り専用ビーンは、前記データ項目の第１のキャッシュされたコピーが削除されるようにするインターフェースを有し；および
前記第２の読み取り専用ビーンは、前記データ項目の第２のキャッシュされたコピーが削除されるようにするインターフェースを有すること
を特徴とする、請求項１２に記載のシステム。
【請求項１５】
前記第１および第２の読み取り専用ビーンの各々は、前記無効リクエストの再送信が得られない場合、前記データ項目を読み取るように適応していることを特徴とする、請求項１２に記載のシステム。
【請求項１６】
前記無効ターゲットはさらに、前記読み取り／書き込みビーンの識別性を含むことを特徴とする、請求項１２に記載のシステム。
【請求項１７】
前記読み取り／書き込みビーンは、前記データ項目のコピーをキャッシュする各読み取り専用ビーンへ無効リクエストを送信するために、前記無効ターゲットを検査するように適応していることを特徴とする、請求項１２に記載のシステム。
【請求項１８】
前記第１および第２の読み取り専用ビーンの各々はさらに、無効リクエストを処理した後に、前記データ項目の新しいコピーを読み取るように適応していることを特徴とする、請求項１２に記載のシステム。
【請求項１９】
ネットワーク・クラスタにおけるデータ項目のキャッシュされたコピーに関する、サーバ・コンピュータによる実行のためのコンピュータ・プログラムであって：該コンピュータ・プログラムが、前記サーバ・コンピュータに、
サーバ上にデータ項目のコピーを記憶することができる手順であって、前記データ項目はデータベースに記憶されている手順と；
前記コピーへの読み取りアクセスを提供するために読み取り専用ビーンを生成することができ、および前記読み取り専用ビーンに関連する識別情報を記憶する手順と；
前記データベースにおける前記データ項目への読み取りおよび書き込みアクセスを提供するために、読み取り／書き込みビーンを生成することができ、および前記読み取り／書き込みビーンを用いて前記データ項目を更新する手順と；および
前記データ項目のコピーを削除できるように、前記識別情報を用いて前記読み取り専用ビーンへの無効リクエストを送信することができる手順とを実行させ、
前記無効リクエストがバージョン識別子を含み、
前記読み取り専用ビーンへの無効リクエストを送信することが、前記識別情報によって

識別された前記データ項目のコピーを記憶するあらゆる読み取り専用ビーンに前記リクエストをマルチキャストし、

前記読み取り専用ビーンに関する無効リクエストを怠ったかどうかを決定するために、前記バージョン識別子がチェックされ；

前記読み取り専用ビーンが無効リクエストを怠っていた場合、無効メッセージが前記読み取り専用ビーンに再送信されることが要求される

コンピュータ・プログラム。

【発明の詳細な説明】

【技術分野】

【０００１】

優先権の主張

本出願は、ここに含まれている以下の出願に基づき優先権を主張する：

米国特許仮出願第６０／３３５，６３３号、２００１年１０月２５日出願、Ｄｅａｎ Ｂｅｒｎａｒｄ ＪａｃｏｂｓおよびＲｏｂ Ｗｏｏｌｌｅｎによる“ビーン・キャッシュをフラッシュするためのシステムおよび方法”。

米国特許出願第１０／２１２，３８２号、２００２年８月５日出願、Ｄｅａｎ Ｂｅｒｎａｒｄ Ｊａｃｏｂｓ、Ｒｏｂ ＷｏｏｌｌｅｎおよびＳｅｔｈ Ｗｈｉｔｅによる“ビーン・キャッシュをフラッシュするためのシステムおよび方法”。

【０００２】

著作権の通知

本特許明細書の開示の一部は、著作権保護を受ける資料を含んでいる。著作権所有者は、特許の開示の特許明細書の、いかなる者による複製にも異議を申し立てない。それは、特許商標局の特許ファイルまたは記録において見られるからであるが、そうでなければ、すべての著作権を保有する。

【０００３】

相互参照出願

以下の出願は、相互に参照され、また参考文献としてここに含まれる：

米国特許仮出願第６０／３０５，９８６号、Ｄｅａｎ Ｂｅｒｎａｒｄ Ｊａｃｏｂｓ，Ｒｅｔｏ Ｋｒａｍｅｒ，およびＡｎａｎｔｈａｎ Ｂａｌａ Ｓｒｉｎｖａｓａｎによる、２００１年7月１６日出願、“データ複製プロトコル”。

米国特許仮出願第６０／３１６，１８７号，Ｄｅａｎ Ｂｅｒｎａｒｄ ＪａｃｏｂｓおよびＲｏｂ Ｗｏｏｌｌｅｎによる、２００１年8月３０日出願、“同時チェックを伴うクラスタ・キャッシング”。

【０００４】

本発明は、一般的にネットワーク上にデータを記憶するためのシステムおよび方法に関する。

【背景技術】

【０００５】

データ項目が、ネットワークを介してアクセスできる単一のデータベースまたはデータ記憶装置に記憶される場合、複数のサーバまたはクライアントが、そのデータ項目(item)へのアクセスを要求することがしばしばある。従来、これは、データ項目がアクセスされるたびに、データベースへのヒットを必要とする。データベースへの各ヒットは、比較的にリソース集約的であり、かつ比較的に効率的ではない。

効率性およびスケーラビリティ(拡張性)の問題の一部を克服する一つの方法は、キャッシュ・メモリにデータ項目のローカル・コピーを記憶することである。サーバまたはクライアントは、該データ項目へのアクセスが将来必要になった場合、該ローカル・コピーを使用することができる。この処理は、決して変わることのないデータ項目には適切であり、および効率的であるかもしれないが、データ項目がデータベースにおいて更新されるときに問題が生じうる。

データベースにおけるデータ項目が更新される場合、ネットワーク上のローカル・キャ

ッシュに記憶されている該データ項目のコピーは、データベースにおける項目とは異なる。該キャッシュは自動的には更新を受信しないからである。ネットワーク上の複数のサーバおよび／またはクライアントにローカル・コピーがある場合、問題は大きくなる。これらのローカル・コピーの各々は異なる時間に作成されるので、ネットワーク上に複数のバージョンのデータ項目が存在しうる。ユーザが、データ項目を更新し、または見ようとすると、ユーザによってアクセスされたコピーは最新のものではないかもしれないし、また正しくないかもしれない。

【０００６】

データ待ち(レイテンシー)に関するそのような問題は、リアルタイムに近い正確さを必要とするアプリケーション、例えば“リアルタイム”株価を提供するウェブ・サイトでは深刻な問題を生じうる。そのようなアプリケーションは、テーブルに関する一次キーとして使用されうる株銘柄記号を含む１のコラムと、各株銘柄の最新の価格を含む１のコラムである、少なくとも２のコラムを有するデータベース・テーブルを利用してもよい。そのようなアプリケーションにおいては、アクティビティのほとんどは、ユーザが該サイトにアクセスして、最新の株価を読み取ることを含む。通常は、定期的に、例えば毎分１回、更新された株価とともに利用されるバック・エンド・アプリケーションまたはシステムを含むアクティビティも存在する。これらのバック・エンド・システムは、データを更新するために、データベースへの読み取り／書き込みアクセスを必要とする。

システムへのアクセスのほとんどは読み取り専用である。これら読み取り専用ユーザに関しては、該システムはより迅速なアクセスを提供するためにデータをキャッシュすることができる。該システムは定期的に、例えば１５分ごとにキャッシュされた情報を更新することができる。しかしながら、そのような“ほとんどが読み取り(read-mostly)”である状況では、ユーザに、最も新しいデータを提供することが好ましい場合がある。正確な情報を提供する際の１５分の遅れは、多くのアプリケーションには望ましくないかもしれない。通常、できるだけ正確な情報をユーザに提供することが望ましい。

【０００７】

ユーザが正確な情報、または少なくとも、データベースに記憶されたデータと同じ新しさを持つ情報を必ず得るための一つの方法は、キャッシュされたコピーを読み取る代わりに各リクエストのたびにデータベースから情報を引き出すことである。これは、多くのアプリケーションには大変高価なものになりうる。なぜなら、データベースへのヒットは、メモリから値を読み取るよりもずっと時間集中的でリソース集中的だからである。

データベースのデータを更新する人々にとって、性能を向上させるために、できるだけ多くの更新をバッチ・トランザクションへまとめることが望ましいかもしれない。更新を単一のトランザクションにまとめることは、すべての更新が生じるか、または更新がまったく生じさせないいずれかの状態であることを確実にする。しかしながら、トランザクションにおいて更新された各項目に対してキャッシュされたコピーをどのように更新するかにおいて問題が生じる。

【発明の開示】

【課題を解決するための手段】

【０００８】

ネットワーク・クラスタにおける少なくとも１のサーバにあるローカル・キャッシュに記憶されたデータ項目のコピーを更新するためのシステムおよび方法が含まれる。該クラスタにおけるサーバに記憶された読み取り／書き込みビーンに識別情報が供給される。該識別情報は、ローカル・キャッシュに読み取り専用ビーンおよびデータ項目のコピーを含むクラスタにおけるあらゆるサーバに関連している。読み取り専用ビーンは、データ項目のローカル・コピーへの読み取りアクセスを提供する。オリジナルのデータ項目はネットワーク・データベースに記憶され、および読み取り／書き込みビーンを用いて更新される。データ項目が、読み取り／書き込みビーンによって更新されると、無効リクエストが、読み取り／書き込みビーンを含むサーバからクラスタ全体へ送信されまたはマルチキャストされことができ、またはデータ項目のローカル・コピーを有する識別情報によって識別

されるあらゆるサーバまたは読み取り専用ビーンへと送信されることができる。データ項目のあらゆるローカル・コピーは、リクエストに応じて廃棄することができる。データ項目の最新のコピーはデータベースから読み取ることができ、またローカル・キャッシュに記憶することができる。

【発明を実施するための最良の形態】

【０００９】

ネットワークに分散された項目の間の一貫性を維持するために、本発明に従ったシステムは、ビーン、すなわちＪａｖａＢｅａｎｓ（Ｊａｖａは登録商標）を利用することができる。ビーンは基本的には、機能を拡張するためにサーバに加えることができるコンポーネントのためのフレームワークである。一つの実施形態は、２種類のビーン、すなわち“読み取り専用”エンティティ・ビーンおよび“読み取り／書き込み”エンティティ・ビーンを利用する。エンティティ・ビーンとは、持続的で、共有アクセスを可能にし、一次キーを持ち、および他のエンティティ・ビーンと関係することができるビーンである。各エンティティ・ビーンは、関連のあるデータベースに基礎となるテーブルを有することができ、ビーンの各インスタンスは該テーブルの列に対応している。

読み取り専用ビーンとは、サーバ上でキャッシュされることができるビーン、例えばネットワーク・クラスタに存在するエンタープライズＪａｖａＢｅａｎである。読み取り専用ビーンは、クラスタの内外のあらゆるクライアントと同様に、クラスタのあらゆるサーバへの読み取りアクセスを提供することができる。読み取り／書き込みビーンは、トランザクショナルであり、クラスタのサーバ上に存在し、およびネットワーク・データベースへの読み取り／書き込みアクセスをクラスタ・サーバに提供している。読み取り専用ビーンは、クラスタ・サーバ上のローカル・キャッシュにあるデータを取り扱う。読み取り／書き込みビーンは、データベースにある情報を取り扱う。

【００１０】

キャッシュおよびデータベースにおける情報の同時並行性(concurrency)を解決するための一つの方法は、タイムアウト値を各読み取り専用エンティティ・ビーンと結び付けることである。例えば、読み取り専用ビーンは、１０分のデフォルト・サイクルとともに活用することができる。１０分の各期間が経過した後、読み取り専用ビーンはデータベースに戻り、最新の値を読み取る。このアプローチは、一定のアプリケーション、例えば、定期的な間隔で変化する値を伴う該アプリケーションなどには有効に作用しうる。

しかしながら、ほとんど変化しないデータを有するアプリケーションがあるかもしれない。このデータが変化した時に、ユーザはできるだけ早くその変化を知りたがるかもしれない。データはそれほど頻繁には変化しないので、リソースを保存するために、長い読み取りサイクル時間を設定することが魅力的である。しかしながら、これはデータに関する待ち時間問題を作り出すという望ましくない影響を持ちうる。なぜならば、データ更新において生じる遅延は、更新が生じるサイクルの中の時点に依存して、ほぼサイクル時間と同じ長さとなりうるからである。そのようなアプリケーションに関して、読み取り専用ユーザがアクセスすることができるデータは、データベースにおけるデータが更新された後、できるだけ早く更新されることが望ましい。

【００１１】

本発明に従った一つのシステムは、読み取り専用ビーンによってさらされるインターフェースを供給する。該インターフェースによってユーザまたはアプリケーションは、ユーザがデータ項目を更新したとき、または更新に気づいたときに、キャッシュを廃棄する、すなわちキャッシュを“無効にする”ことをシステムに命じることができる。このインターフェースは、エンティティ・ビーンが通常は“ホーム”またはそれを作り出すファクトリを有するために、“ＣａｃｈｉｎｇＨｏｍｅ”と呼ばれる。ＣａｃｈｉｎｇＨｏｍｅはそれに関する３の方法を持つことができ、以下のようにコード化されうる：

```
package weblogic.ejb;
public interface CachingHome{
        public void invalidate(Object pk)throws RemoteException;
```

```
        public void invalidate(Collection pks)throws RemoteException;
        public void invalidateAll()throws RemoteException;
}
```

【００１２】

メソッドinvalidate(Object pk)は、ユーザに、データベースまたはデータ・テーブルにおける特定の一次キーと関連付けられたデータを無効にさせる。メソッドinvalidate(Collection pks)は、ユーザに、キーの集まりまたはグループに関するデータを無効にさせる。メソッドinvalidateAll()はユーザに、データベース・テーブルにあるすべてのキーに関するデータを無効にさせる。これらの無効メソッドは、値がローカル・キャッシュに記憶されることを、プログラマ、アプリケーション、ユーザまたはシステムがそうでないことを言わないまで、ユーザに対して、保証することを可能にする。

そのような方法３００が図５のフローチャートに示されている。データ項目のコピーは、ネットワーク・クラスタの少なくとも１のサーバに記憶される３０２。データ項目は、クラスタ・サーバの一つにある読み取り／書き込みビーンを用いてデータベースにおいて更新されうる３０４。無効リクエストは、サーバの一つにある読み取り専用ビーンのインターフェースを用いて開始することができ、該リクエストは、データ項目のローカル・コピーを含むあらゆるサーバに送られる３０６。データ項目のあらゆるコピーは、リクエストを受信するサーバから廃棄されることができる３０８。

【００１３】

図１に示されているような、ネットワーク・クラスタ１０４を伴うシステム１００において、データベース１０８に記憶された値１２２のコピーは、クラスタ１０４における各サーバ１１０、１１２、１１４でキャッシュされることが可能である。クライアント１０２がネットワーク１０６を通してサーバ１１０に接触し、例えばリクエスト“invalidate(Key)”を作ることによって、サーバ１１０が一定のキーを無効にすることを要求する場合、サーバ１１０にとっては、該キーに関連する値１２２または複数の値のキャッシュされたコピー１１６を廃棄することは容易である。しかしながら、サーバ１１２および１１４に、同様に、それらのキャッシュされたコピー１１８、１２０をどのように廃棄するか、に問題が存在する。

一つの実施形態はサーバ１１０に、図２に示されたように、クライアント１０２から無効リクエスト１２４を受信するときに、ローカル・キャッシュにおいてコピー１１６を廃棄させる。ローカル・メモリからコピー１１６を廃棄した後、サーバ１１０は、ローカル・キャッシュにある値のコピーを廃棄するために、他のサーバ１１８、１２０にマルチキャストでメッセージ１２６をまたはクラスタ１０４内で読み取り専用ビーンを送信することができる。マルチキャストは、１つのサーバ、または発信源から、不要なパケット複製をすることなく他のサーバへ情報のパケットまたはメッセージを送信するための技術であり、１つのパケットがソースから送信され、必要に応じてネットワークで複製される。このアプローチは各サーバに、データベースにおける値が更新されるときに、キャッシュされた値を廃棄させる。

【００１４】

図６は、ネットワーク４０４との通信をする複数のクライアント４０２および複数のデータベース４１６を含む別のシステムを示す。データ項目４１８がデータベース４１６のうちの１に記憶される。データ項目のコピー４２０は、クラスタ４０８のクラスタ・サーバ４０６に記憶される。クラスタ４１２のクラスタ・サーバ４２２に記憶された２つのコピー４２２、及びどのサーバ・クラスタにも含まれていない、サーバ４１４上のコピー４２４がある。このシステムは、データ項目のコピー４１８を含むサーバ４０６、４１０、４１４のうちの１つが、それがクライアントから無効リクエストを受信するときにローカル・キャッシュにおけるコピーを廃棄することができ、またローカル・キャッシュにおける値のあらゆるコピーを廃棄するため、ネットワーク上の他のサーバにマルチキャストでメッセージを送信するという点で、図１および２に記載のシステムと同じように作動する。

マルチキャスト・メッセージは発信源によって一度しか送信されず、また他のサーバによる受信の確認を待たないという事実により、別の問題が存在する。クラスタにおけるサーバは、例えば一時的なオフラインの場合、無効リクエストを入手しないかもしれない。本発明に従ったシステムは、バージョン番号または連続番号でそのような各メッセージまたはリクエストにタグを付けることによって、より信頼できるマルチキャストを供給することができる。このように、リクエストを受信するサーバは、そのリクエストのバージョンとサーバが受信した前のリクエストのバージョンを知り、サーバがメッセージを逃したかどうかを知るようにする。サーバが、メッセージを逃したと決定すると、メッセージが、それに従って更新できるように再度送信されることを要求することができる。

【００１５】

しかしながら、このアプローチの問題は、サーバが、別の更新が送信されるまで更新を怠ったことを知らない、ということである。あるアプリケーション、例えば第１，２、および３の週の、ウィークリー・スペシャルを知らせるオンライン店舗においては、正しい情報を入手するために、次の更新まで待つというのは認容できないかもしれない。該店舗は、第３週の更新まで、第２週のスペシャルを逃したことを気づかないだろう。第１週スペシャルは、第２週の間そのままであり、その時間の間にシステムにアクセスしているユーザに、誤った情報を表示する。システムが第２週の更新を怠ったことに気づくとき、すでに第３週になっている。サーバは結局、ユーザに表示されることなく、第２週の情報を単に廃棄して終わるだろう。

本発明に従ったシステムは、クラスタにおけるサーバに、定期的に情報を“ハートビートする”ことによって、この問題を避けることができる。サーバは、ネットワークまたはクラスタを介して定期的にメッセージを送信することによって、情報のパケットまたはメッセージをハートビートする。ハートビート・メッセージは、最新バージョン番号、前のバージョン番号などの情報、または更新情報が十分に小さく実用的である場合には、実際の更新情報そのものを含むことができる。サーバが、最新バージョン番号を含むハートビート・メッセージを受信し、かつサーバがそのバージョンのデータに基づいてはいなく、または最新の無効リクエストを受信しなかった場合、サーバは、サーバから無効メッセージを要求し、または引き出すことができる。

【００１６】

マルチキャストおよび／またはハートビートを送信するサーバでもよい、最初に無効リクエストを送信する開始サーバは、一定の時間、最新のリクエストを記憶することができ、または一定数の最新のリクエストを記憶することができる。クラスタ・サーバが、開始サーバがまだ記憶している無効メッセージを要求する場合、開始サーバは、例えばマルチキャストやポイント・ツウ・ポイント接続などの方法によって、クラスタ・サーバに該メッセージを単純に送信することができる。開始サーバがもはやメッセージを持っていない場合、開始サーバが、クラスタ・サーバに、単純にそのキャッシュ全体を廃棄するように、命じることができるが、その理由は、どのキーが変わったのかをクラスタ・サーバに伝えることができないからである。クラスタ・サーバは、データベースから、新しいおよび／または最新の情報を読み取ることができる。これは一時的に性能を下げることがあるが、新しくキャッシュされた情報は、少なくともデータベースにある情報と同じくらい新しい。

動作上、クライアントまたはアプリケーションは、読み取り／書き込みエンティティ・ビーンを通してデータ項目を更新することができる。更新、すなわち複数の更新を含むトランザクションは、データベースへコミットする。無効メッセージは、クラスタにおけるサーバへと送信されることができ、該メッセージは、例えばデータ項目を更新するクライアントまたはサーバによって引き起こされる。クラスタ・サーバはそれぞれ、ローカル・キャッシュに中のあるあらゆるコピーを廃棄することができ、およびすぐに、後に、またはリクエストを出す必要があるときに、データベースから新しい値を読み取ることができる。通常は、コミットされていないデータを読み取ることはできないので、最初にデータがコミットされ、そしてメッセージがクラスタへマルチキャストされる２段階処理を用い

るのが好ましいかもしれない。

【００１７】

上述のアプローチに関する１つの問題は、無効リクエストをクライアントに強制的に開始させることであり、それは該クライアントとってさらにやや複雑なものを伴いうる。クライアントが、誤って無効方法を用いたり、間違いを犯す可能性もある。よって、システムはそれを自動的に行えることが好ましいかもしれない。

本発明に従ったシステムは、“無効ターゲット”を用いることによってこの問題を解決することができる。無効ターゲットは、読み取り専用および読み取り／書き込みビーンがデータベースにおける同じデータを指し、データを読み取る人は、読み取り専用ビーンを使用し、データを更新する人は読み取り／書き込みビーンを使用するという考えに基づいている。この考えとは、読み取り／書き込みビーンが更新され、または変更される時読み取り専用ビーンを無効にすることである。

【００１８】

エンティティ・ビーンすなわちエンタープライズＪａｖａＢｅａｎを配置するとき、通常は実際のエンティティ・ビーンに関するメタ・データを記憶するために使用される配置記述子がある。配置記述子は、例えば、クライアントが利用できなければならないビーンによって供給されるサービスに関する情報を供給するための使用されるＸＭＬドキュメントでもよい。該情報は、クライアントに幅広い情報、例えばリクエスト・ルーティング情報を、Ｊａｖａクラスを支持するための方法およびクラスの詳細と同様に供給することができる。以前に“無効ターゲット”と称された、配置記述子にタグを加えることができる。読み取り／書き込みビーンに関する無効ターゲットは、あらゆる関連する読み取り専用ビーンの識別性を含むことができる。

一つの実施形態において、サーバが読み取り／書き込みビーンから情報を要求するとき、または該リクエストが読み取り専用ビーンを生成するとき、無効ターゲットが更新されうる。読み取り／書き込みビーンがデータベースまたはデータ記憶装置からの情報を、要求サーバに転送するとき、読み取り／書き込みビーンは無効ターゲットも更新することができる。読み取り／書き込みビーンを含むサーバに記憶されたＸＭＬファイルは、情報を要求する、またはビーンを生成するサーバの識別性を含むために更新されうる。

【００１９】

図３はまた、図１に記載のシステム１００を示しているが、この実施形態においては、システムが無効ターゲットを利用するように示されている点が異なる。読み取り／書き込みビーン１２８がデータ項目１２２を更新するために使用されるときはいつも、システムは読み取り／書き込みビーン１２８と関連する無効ターゲットに注意を向けることができ、および読み取り／書き込みビーンと関連する各読み取り専用ビーン１３０へと無効リクエストを送信することができる。無効リクエストは、サーバ１１０によって直接影響を受けることがあり、それは読み取り／書き込みビーン１２８および読み取り専用ビーン１３０の両方を含む。無効リクエストは、適切なプロトコルによって、読み取り専用ビーン１３０を含む他のサーバにも送られることができ、それは無効ターゲットの中にある。１つのアプローチにおいて、サーバ１１０は、ポイント・ツウ・ポイント接続１３４によって直接サーバ１１２と接触し、サーバ１１２上の読み取り専用ビーン１３０に、サーバ１１２上のキャッシュされたコピー１１８を廃棄するよう命じる。別のアプローチでは、サーバ１１０は、ネットワーク１０６を介して、例えばサーバ１１２など、無効ターゲットの範囲内の読み取り専用ビーン１３０を含むあらゆるサーバへとマルチキャスト・メッセージ１３２を送信することができる。

【００２０】

無効化された読み取り専用ビーンへのその後の呼び出しは、ejbLoadなどのメソッドを呼び出させることができ、そのメソッドはデータベースからキャッシュへの最新の情報を読み取ることができる。例えば、コンテナ管理持続性（ＣＭＰ）ビーン、すなわち状態が自動的にデータベースと同期するエンティティ・ビーンは、読み取り／書き込みビーンが変更されたときに無効にされるべき読み取り専用エンティティ・ビーンを特定するために

、ejb-jar.xmlなどのＸＭＬファイルにある無効ターゲット・エレメントを使用することができる。この実施例におけるコンテナは、例えばトランザクションが完了した後など、無効ターゲットを自動的に無効にすることができる。

そのような方法２００が図４のフローチャートに示されている。データ項目のコピーは、ネットワーク・クラスタにおける少なくとも１のサーバに記憶される２０２。識別情報、すなわち無効ターゲットは、クラスタにおけるサーバ上の読み取り／書き込みビーンに供給され、該情報は、読み取り専用ビーンおよびデータ項目のローカル・コピーを含むクラスタにおけるあらゆるサーバに関連する２０４。データ項目は、読み取り／書き込みビーンを用いて、データベースにおいて更新されうる２０６。無効リクエストは、読み取り／書き込みビーンを含むサーバから、識別情報によって識別されるあらゆるサーバへと送信されうる２０８。データ項目のコピーはそれから、リクエストを受信する各サーバによって廃棄されうる２１０。

【００２１】

このようにして、カスタマまたはクライアントは、項目を無効にするために追加のコードを書く必要が無い。本発明の実施形態に従って、無効ターゲットのみが、読み取り／書き込みおよび読み取り専用ビーンをコヒーレントに維持するために特定されなければならない。該ビーンは、同じサーバに共存することができ、読み取り専用ビーンが、ローカル・キャッシュから項目を読み取り、および読み取り／書き込みビーンは、データベースから読み取り、かつデータベースに書き込む。

性能を向上させるため、本発明に従ったシステムは、各更新のために個別のメッセージを送信する代わりに、更新全体のトランザクションまたは一連の更新がデータベースまたはデータ・テーブルにコミットされるまで待機することができる。サーバ、例えば更新を開始するサーバは、トランザクションの間にどのキーが更新されたかを追跡することができ、また更新されたすべての一次キーに関する情報を含む単一のメッセージをマルチキャストすることができる。そのようにメッセージをまとめることで、そのようなシステムの全体の性能を向上することができ、およびエラーや不一致の機会を低減することができる。

【００２２】

本発明に従って使用されうるシステムの一つの実施形態は、各記号と関連する情報、例えば価格や量と同様に、株銘柄記号のテーブルを含む。Ｊａｖａサーバ・ページは、ユーザに株価の最新価格を要求させるために使用されうる。Ｊａｖａサーバ・ページは、読み取り専用エンティティ・エンタープライズＪａｖａビーンから情報を読み取ることができる。Ｊａｖａ　Ｍｅｓｓａｇｅ　Ｓｅｒｖｉｃｅ（ＪＭＳ）待ち行列は、株価の更新を伴うメッセージを受信することができる。メッセージ駆動型ビーンは、これらのメッセージを待機解除することができ、および関連するＣＭＰエンティティ・ビーンを更新することができる。この変更が生じると、コンテナは、関連する読み取り専用ビーンを無効にすることができる。

本発明の好ましい実施形態の前述の説明は、例示と説明の目的で示されてきた。それは網羅的なものを意図しておらず、また本発明を、開示された詳細な形式に限定することを意図していない。多くの変更や変形が、通常の当業者には明らかである。実施形態は、本発明の原理およびその実践的応用を最も良く説明するために選択され、また説明されたのであり、それによって他の当業者が本発明の様々な実施形態について理解することができ、特定の使用に適した様々な変更が予期される。本発明の範囲は、以下の特許請求の範囲およびその均等物によって定められることが意図される。

【図面の簡単な説明】

【００２３】

【図１】図１は、本発明に従った一つの実施形態に従ったシステムの図である。

【図２】図２は、図１の実施形態に従ったシステムの図である。

【図３】図３は、図１のシステムの代替的実施形態の図である。

【図４】図４は、本発明の一つの実施形態に従った方法のフローチャートである。

【図５】図５は、本発明の別の実施形態に従った方法のフローチャートである。
【図６】図６は、本発明に従った別の実施形態に従ったシステムの図である。

【図１】

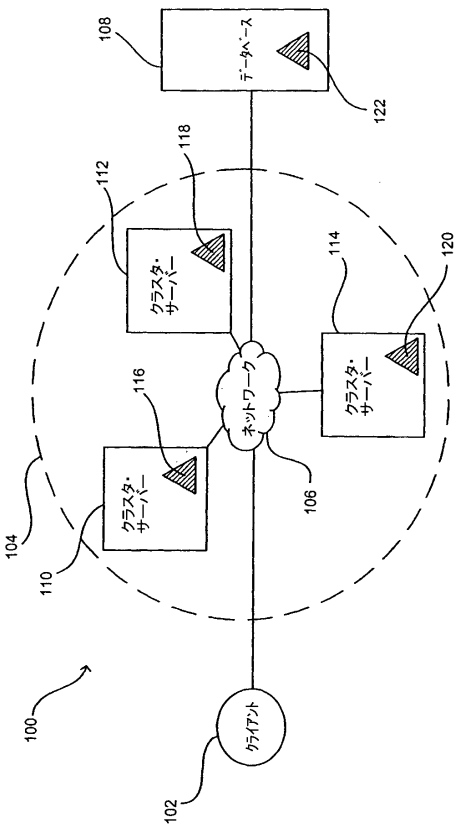

Figure 1

【図２】

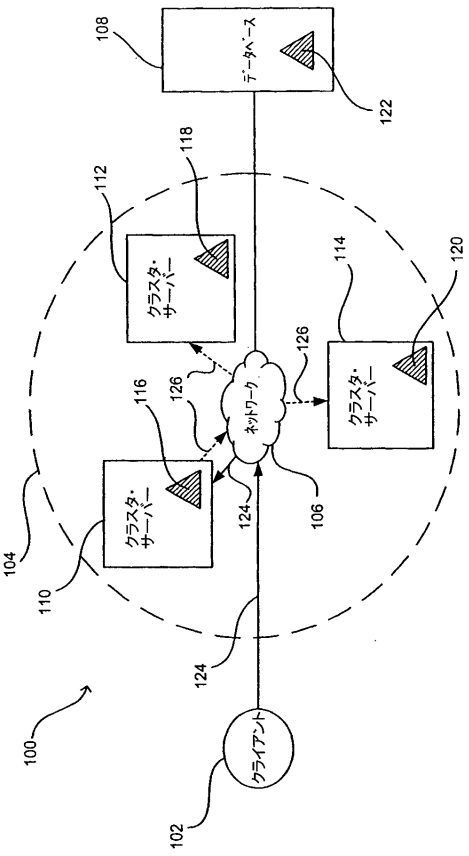

Figure 2

【図３】

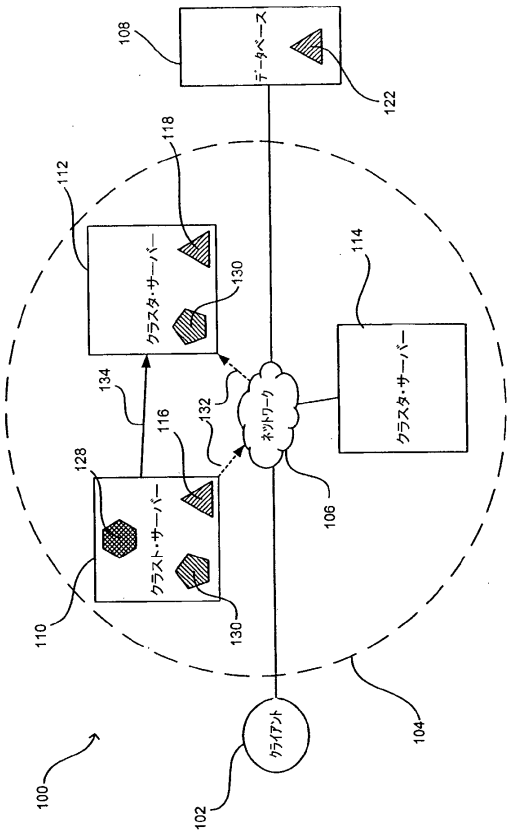

Figure 3

【図４】

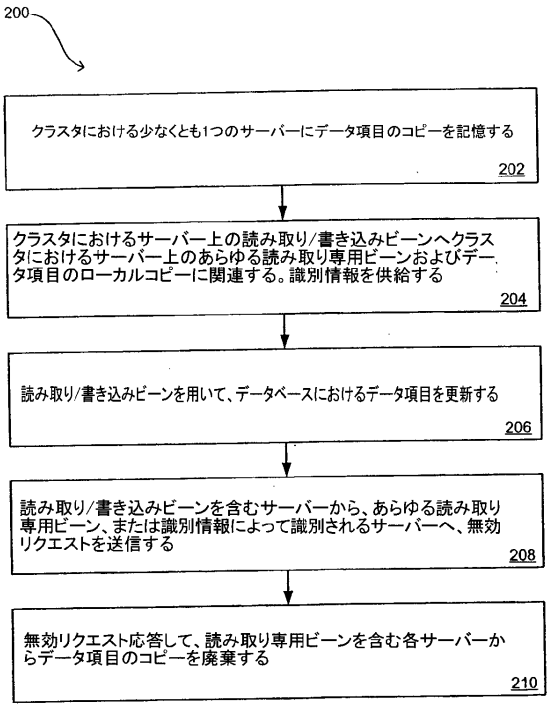

Figure 4

【図５】

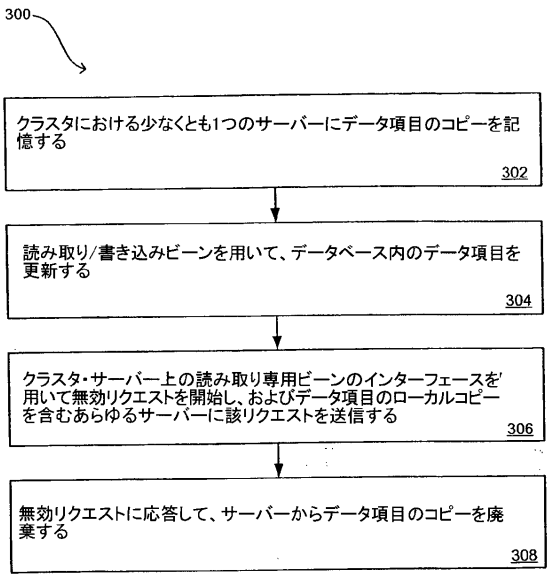

Figure 5

【図６】

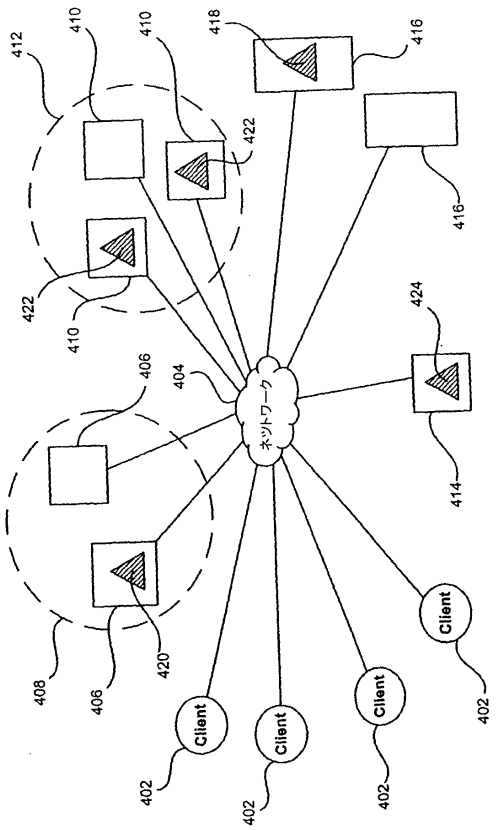

Figure 6

フロントページの続き

(74)代理人 100086771
弁理士 西島 孝喜
(72)発明者 ジェイコブズ ディーン バーナード
アメリカ合衆国 カリフォルニア州 ９４７０７ バークリー マデラ ストリート １７４７
(72)発明者 ウーレン ロブ
アメリカ合衆国 カリフォルニア州 ９４１２７ サン フランシスコ フォーティーンス アベニュー ２５３１
(72)発明者 ホワイト セス
アメリカ合衆国 カリフォルニア州 ９４１１６ サン フランシスコ リヴァーラ ストリート １０４５ アパートメント ビー

審査官 上嶋 裕樹

(56)参考文献 荒井文吉, データベースアクセス用サーバサイド ＪａｖａＢｅａｎｓコンポーネントの制作, ＪＡＶＡ ＰＲＥＳＳ, 日本, （株）技術評論社, ２００１年 4月１０日, 第１７巻, 第１１０－１１８頁
実森仁志, 注目のインターネット製品 Ｏｒａｃｌｅ９ｉ 製品統合しプラットフォーム色を強化 キャッシュ技術で高速化を図る, 日経インターネットテクノロジー, 日本, 日経ＢＰ社, ２００１年 1月２２日, 第４３号, 第１３６－１３９頁

(58)調査した分野(Int.Cl., ＤＢ名)
G06F 12/00
G06F 13/00