(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3853592号
(P3853592)
(45)発行日　平成18年12月6日(2006.12.6)　(24)登録日　平成18年9月15日(2006.9.15)

(51)Int.Cl.　F I
G06F 9/50 (2006.01)　G06F 9/46 465Z

請求項の数 27 (全 28 頁)

(21)出願番号　特願2000-519523(P2000-519523)
(86)(22)出願日　平成10年10月26日(1998.10.26)
(65)公表番号　特表2001-522113(P2001-522113A)
(43)公表日　平成13年11月13日(2001.11.13)
(86)国際出願番号　PCT/US1998/022656
(87)国際公開番号　WO1999/023784
(87)国際公開日　平成11年5月14日(1999.5.14)
審査請求日　平成15年10月27日(2003.10.27)
(31)優先権主張番号　08/962,485
(32)優先日　平成9年10月31日(1997.10.31)
(33)優先権主張国　米国(US)

(73)特許権者　502303739
オラクル・インターナショナル・コーポレイション
アメリカ合衆国、９４０６５　カリフォルニア州、レッドウッド・ショアーズ、オラクル・パークウェイ、５００
(74)代理人　100064746
弁理士　深見　久郎
(74)代理人　100085132
弁理士　森田　俊雄
(74)代理人　100083703
弁理士　仲村　義平
(74)代理人　100096781
弁理士　堀井　豊

最終頁に続く

(54)【発明の名称】分散ウェブアプリケーションサーバ

(57)【特許請求の範囲】

【請求項１】

オペレーションを実行するための方法であって、ディスパッチャ（２１４）を第１のマシン上で実行するステップを含み、前記ディスパッチャ（２１４）は、クライアントからリクエストを受取り、かつ前記リクエストを実行することが可能であるカートリッジに前記リクエストを分散するよう構成されるエンティティであり、リソースマネージャ（２５４）を第２のマシン上で実行し、前記第１のマシンとは異なるマシン上で実行しているクライアント（２０２）から前記ディスパッチャ（２１４）においてリクエストを受取り、前記方法は

前記ディスパッチャ（２１４）が前記リクエストを受取ったことに応答して、第１のメッセージをディスパッチャ（２１４）からリソースマネージャ（２５４）へ送ることを特徴とし、第１のメッセージは前記オペレーションを行なうことのできるある特定のカートリッジタイプを識別し、さらに

第２のメッセージをリソースマネージャ（２５４）からディスパッチャ（２１４）へ送ることを特徴とし、第２のメッセージはある特定のカートリッジのある特定のインスタンスを識別し、ある特定のインスタンスは第３のマシン上で実行されており、さらに

第３のメッセージをディスパッチャ（２１４）からある特定のインスタンスに送り、ある特定のインスタンスにオペレーションを実行させることを特徴とし、

第１のマシン、第２のマシンおよび第３のマシンのうちの少なくとも２つは別個のマシンである、方法。

【請求項２】
ブラウザがブラウザリクエストをウェブリスナ（２１０）に送るステップと、
ウェブリスナ（２１０）がブラウザリクエストをディスパッチャ（２１４）に渡すステップとをさらに含み、
ディスパッチャ（２１４）はウェブリスナ（２１０）からブラウザリクエストを受取ったことに応答して第１のメッセージを送る、請求項１に記載の方法。
【請求項３】
第１のマシンおよび第２のマシンは別個のマシンであり、
第１のメッセージをディスパッチャ（２１４）からリソースマネージャ（２５４）へ送る前記ステップは、第１のメッセージをディスパッチャ（２１４）からリソースマネージャ（２５４）へオブジェクトリクエストブローカを介して送ることによって行なわれ、
第２のメッセージをリソースマネージャ（２５４）からディスパッチャ（２１４）へ送る前記ステップは、第２のメッセージをリソースマネージャ（２５４）からディスパッチャ（２１４）へオブジェクトリクエストブローカを介して送ることによって行なわれる、請求項１に記載の方法。
【請求項４】
第２のマシンおよび第３のマシンは別個のマシンであり、
リソースマネージャ（２５４）が第１のメッセージを受取ったことに応答してリソースマネージャ（２５４）がある特定のインスタンスに第３のマシン上での実行を開始させるステップを含む、請求項１に記載の方法。
【請求項５】
リソースマネージャ（２５４）は、第１のメッセージを受取ったことに応答して以下のステップを行なうことによってどのある特定のインスタンスがオペレーションを行なうべきであるかを定め、前記以下のステップは
ある特定のカートリッジ（２６０）のインスタンスで現在実行されており割当てされていないものがあるかどうかを判定するステップと、
ある特定のカートリッジ（２６０)のインスタンスのどれかが現在実行されており割当てされていない場合、現在実行されており割当てされていないインスタンスを前記第２のメッセージにおいて特定するステップと、
第１のメッセージにおいて特定されるある特定のカートリッジ（２６０）のインスタンスで現在実行されており割当てされていないものがない場合、ある特定のカートリッジの新しいインスタンスをイニシエートして前記新しいインスタンスを前記第２のメッセージにおいて特定するステップとを含む、請求項１に記載の方法。
【請求項６】
新しいインスタンスをイニシエートする前に、リソースマネージャ（２５４）は
メタデータ（２５６）を調べて、前記ある特定のカートリッジを実行することのできる、前記第３のマシンを含む１組のマシンを定めるステップと、
前記第３のマシンを選択して前記ある特定のカートリッジの前記新しいインスタンスを実行するステップとを行なう、請求項５に記載の方法。
【請求項７】
前記ディスパッチャ（２１４）に割当てられたインスタンスのリストを前記ディスパッチャに維持させるステップと、
前記第２のメッセージを受取ったことに応答して、前記ディスパッチャ（２１４）にインスタンスの前記リストを更新させて前記ある特定のインスタンスに対するエントリを含むようにさせるステップとをさらに含む、請求項１に記載の方法。
【請求項８】
インスタンスのリストは、前記リストにおけるインスタンスが現在ビジーであるかどうかを示すデータを含み、
前記第２のメッセージを送ることに応答して、ディスパッチャ（２１４）は前記ある特定のインスタンスに対する前記エントリにおいて前記ある特定のインスタンスがビジーであ

ることを示し、
前記方法はさらに、ディスパッチャ（２１４）が前記第３のメッセージに応答して前記ある特定のインスタンスから第４のメッセージを受取るステップを含み、
前記第４のメッセージを受取ったことに応答して、ディスパッチャ（２１４）は前記エントリを更新して前記ある特定のインスタンスが現在ビジーでないことを示す、請求項７に記載の方法。
【請求項９】
方法であって、
前記ディスパッチャ（２１４）においてリクエストを受取る前記ステップは、ディスパッチャ（２１４）が前記カートリッジによって行なうべき第１のオペレーションを特定する第１のブラウザリクエストを受取るステップを含み、前記方法はさらに
ディスパッチャ（２１４）が、第２のカートリッジによって行なうべき第２のオペレーションを特定する第２のブラウザリクエストを受取るステップと、
ディスパッチャ（２１４）がインスタンスの前記リストを調べてリストが現在ビジーではない前記第２のカートリッジのインスタンスに対するエントリを含むかどうかを判定するステップと、
リストが現在ビジーではない前記第２のカートリッジのインスタンスに対するエントリを含む場合、
ディスパッチャ（２１４）が前記第２のブラウザリクエストに応答して前記ある特定のインスタンスへ第５のメッセージを送り、前記ある特定のインスタンスに前記第２のオペレーションを行なわせるステップと、
前記エントリを更新して前記ある特定のインスタンスが現在ビジーであることを示すステップとを行なうステップと、
リストが現在ビジーではない前記第２のカートリッジのインスタンスに対するエントリを含んでいない場合、第６のメッセージを前記リソースマネージャ（２５４）へ送り、前記リソースマネージャ（２５４）に前記第２のオペレーションを行なうための前記第２のカートリッジのインスタンスを示すようにさせるステップとをさらに含む、請求項８に記載の方法。
【請求項１０】
前記ある特定のインスタンスが予め定められた長さの時間より長い間アイドル状態であった場合これを判定するステップと、
前記ある特定のインスタンスが予め定められた長さの時間より長い間アイドル状態であった場合に、前記ディスパッチャ（２１４）が前記ある特定のインスタンスを前記リソースマネージャ（２５４）に解放するステップとをさらに含む、請求項８に記載の方法。
【請求項１１】
オペレーションを実行するためのプログラムを記録したコンピュータ可読媒体であって、
前記プログラムは、第１のコンピュータにディスパッチャ（２１４）の機能を実現させ、前記ディスパッチャ（２１４）は、クライアントからリクエストを受取り、かつ前記リクエストを実行することが可能であるカートリッジに前記リクエストを分配するよう構成されたエンティティであり、
前記プログラムは、さらに第２のコンピュータに、リソースマネージャ（２５４）の機能を実現させ、
前記ディスパッチャ（２１４）は、前記第１のコンピュータとは異なるコンピュータ上で実行しているクライアント（２０２）からリクエストを受ける機能と、
前記リクエストを受取ったことに応答して、第１のメッセージをリソースマネージャ（２５４）へ送る機能とを含み、当該第１のメッセージは前記オペレーションを行なうことのできるある特定のカートリッジタイプを特定し、
前記リソースマネージャは、第２のメッセージをディスパッチャ（２１４）へ送る機能を含み、当該第２のメッセージはある特定のカートリッジのある特定のインスタンスを特

定し、当該ある特定のインスタンスは前記第３のコンピュータ上で実行しており、
前記ディスパッチャは、第３のメッセージを前記ある特定のインスタンスへ送り、前記ある特定のインスタンスにオペレーションを実行させる機能を含み、
前記第１のコンピュータ、前記第２のコンピュータおよび前記第３のコンピュータのうちの少なくとも２つは別個のコンピュータである、コンピュータ可読媒体。
【請求項１２】
前記プログラムは、さらに第４のコンピュータに、ウェブリスナ（２１０）の機能を実現させ、
前記ウェブリスナ（２１０）は、ブラウザから受取ったブラウザリクエストを前記ディスパッチャ（２１４）へ渡す機能を含み、
前記ディスパッチャ（２１４）は、前記ウェブリスナ（２１０）から前記ブラウザリクエストを受取ったことに応答して前記第１のメッセージを送る機能を含む、請求項１１に記載のコンピュータ可読媒体。
【請求項１３】
前記第１のコンピュータおよび前記第２のコンピュータは別個のコンピュータであり、
前記ディスパッチャ（２１４）による第１のメッセージを前記リソースマネージャ（２５４）へ送る前記機能は、前記第１のメッセージを前記リソースマネージャ（２５４）へオブジェクトリクエストブローカを介して送ることによって行なわれ、
前記リソースマネージャ（２５４）による第２のメッセージを前記ディスパッチャ（２１４）へ送る前記機能は、前記第２のメッセージを前記ディスパッチャ（２１４）へ前記オブジェクトリクエストブローカを介して送ることによって行なわれる、請求項１１に記載のコンピュータ可読媒体。
【請求項１４】
前記第２のコンピュータおよび前記第３のコンピュータは別個のコンピュータであり、
前記リソースマネージャは、前記第１のメッセージを受取ったことに応答して前記ある特定のインスタンスに前記第３のコンピュータ上で実行を開始させる機能を含む、請求項１１に記載のコンピュータ可読媒体。
【請求項１５】
前記リソースマネージャ（２５４）は、前記第１のメッセージを受取ったことに応答して以下の機能を行なうことによってどのある特定のインスタンスがオペレーションを行なうべきであるかを判定し、前記以下の機能は
前記ある特定のカートリッジ（２６０）のインスタンスで現在実行されており割当てされていないものがあるかどうかを判定する機能と、
前記ある特定のカートリッジ（２６０）のインスタンスで現在実行されており割当てされていないものがある場合、前記第２のメッセージにおいて、現在実行されており割当てされていないインスタンスを特定する機能と、
前記第１のメッセージにおいて特定される、前記ある特定のカートリッジ（２６０）のインスタンスで現在実行されており割当てされていないものがない場合、前記ある特定のカートリッジの新しいインスタンスをイニシエートして前記新しいインスタンスを前記第２のメッセージにおいて特定する機能とを含む、請求項１１に記載のコンピュータ可読媒体。
【請求項１６】
前記リソースマネージャ（２５４）は、前記新しいインスタンスをイニシエートする前に、メタデータ（２５６）を調べて、前記ある特定のカートリッジを実行することのできる、前記第３のコンピュータを含む１組のコンピュータを定める機能と、
前記ある特定のカートリッジの前記新しいインスタンスを実行するために前記第３のコンピュータを選択する機能とを含む、請求項１５に記載のコンピュータ可読媒体。
【請求項１７】
前記ディスパッチャ（２１４）は、前記ディスパッチャに割当てられたインスタンスのリストを維持する機能と、

前記第２のメッセージを受取ったことに応答して、インスタンスの前記リストを更新して前記ある特定のインスタンスに対するエントリを含むようにする機能とを含む、請求項１１に記載のコンピュータ可読媒体。

【請求項１８】

インスタンスのリストは、前記リスト内のインスタンスが現在ビジーであるかどうかを示すデータを含んでおり、

前記ディスパッチャ（２１４）は、前記第２のメッセージを送ることに応答して、前記ある特定のインスタンスに対する前記エントリにおいて前記ある特定のインスタンスがビジーであることを指示する機能と、

前記第３のメッセージに応答して前記ある特定のインスタンスから第４のメッセージを受取る機能と、

前記第４のメッセージを受取ったことに応答して、前記エントリを更新して前記ある特定のインスタンスが現在ビジーではないことを指示する機能とを含む、請求項１７に記載のコンピュータ可読媒体。

【請求項１９】

前記ディスパッチャ（２１４）によってリクエストを受取る前記機能は、前記カートリッジによって行なうべき第１のオペレーションを特定する第１のブラウザリクエストを受取る機能を含み、

前記ディスパッチャ（２１４）は、

第２のカートリッジによって行なうべき第２のオペレーションを特定する第２のブラウザリクエストを受取る機能と、

インスタンスの前記リストを調べて、リストが現在ビジーではない前記第２のカートリッジのインスタンスに対するエントリを含んでいるかどうかを判定する機能とを含み、

前記ディスパッチャ（２１４）は、

リストが現在ビジーではない前記第２のカートリッジのインスタンスに対するエントリを含んでいる場合、前記第２のブラウザリクエストに応答して前記ある特定のインスタンスへ第５のメッセージを送り、前記ある特定のインスタンスに前記第２のオペレーションを行なわせ、前記エントリを更新して前記ある特定のインスタンスが現在ビジーであることを指示する機能と、

リストが現在ビジーではない前記第２のカートリッジのインスタンスに対するエントリを含んでいない場合、第６のメッセージを前記リソースマネージャ（２５４）に送り、前記リソースマネージャ（２５４）に前記第２のオペレーションを行なうべき前記第２のカートリッジのインスタンスを指示させる機能とを含む、請求項１８に記載のコンピュータ可読媒体。

【請求項２０】

前記ディスパッチャ（２１４）は、前記ある特定のインスタンスが予め定められた長さの時間より長い間アイドル状態であった場合これを判定する機能と、

前記ある特定のインスタンスが予め定められた長さの時間より長い間アイドル状態であった場合に、前記ある特定のインスタンスを前記リソースマネージャ（２５４）に解放する機能とを含む、請求項１８に記載のコンピュータ可読媒体。

【請求項２１】

ブラウザリクエストに関連のあるオペレーションを行なうためのシステムであって、複数のウェブリスナに結合される複数のディスパッチャを含み、前記複数のディスパッチャの各ディスパッチャは前記複数のウェブリスナの対応するウェブリスナから前記対応するウェブリスナが受取ったブラウザリクエストを受取り、前記システムは

マシン間通信機構を介して前記複数のディスパッチャに結合されるリソースマネージャ（２５４）を特徴とし、前記マシン間通信機構は前記複数のディスパッチャが存在するマシンにかかわらず前記リソースマネージャ（２５４）が前記複数のディスパッチャと通信できるようにし、前記リソースマネージャ（２５４）は、前記ディスパッチャ（２１４）からインスタンスに対するリクエストを受取ったことに応答して前記複数のディスパッチャ

の各ディスパッチャに前記複数のカートリッジのうちのあるカートリッジ（２６０）のインスタンスを割当てるよう構成されており、リクエストはある特定のオペレーションを行なうことができるある特定のカートリッジタイプを識別しており、
前記複数のディスパッチャは、前記リソースマネージャ（２５４）によって前記ディスパッチャに割当てられるインスタンスへメッセージを前記マシン間通信機構を介して送るよう構成されており、前記マシン間通信機構により、前記インスタンスが存在するマシンにかかわらず前記複数のディスパッチャが前記インスタンスと通信することができ、前記メッセージは前記インスタンスに前記ブラウザリクエストと関連するオペレーションを行なわせる、システム。
【請求項２２】
前記マシン間通信機構を介して前記複数のディスパッチャに結合される仮想パスマネージャ（２５０）をさらに含み、前記仮想パスマネージャ（２５０）は複数のカートリッジのどれが前記ブラウザリクエストに関連付けられているかを前記ディスパッチャに示す、請求項２１に記載のシステム。
【請求項２３】
前記マシン間通信機構はオブジェクトリクエストブローカである、請求項２１に記載のシステム。
【請求項２４】
リソースマネージャ（２５４）はさらに、前記ディスパッチャからのメッセージに応答してカートリッジの新しいインスタンスをイニシエートするよう構成されている、請求項２１に記載のシステム。
【請求項２５】
各カートリッジに対して、前記リソースマネージャ（２５４）は、実行しているインスタンスの数を予め定められた最小数と予め定められた最大数との間で維持するよう構成される、請求項２４に記載のシステム。
【請求項２６】
前記複数のディスパッチャの各ディスパッチャは、前記リソースマネージャ（２５４）によって前記ディスパッチャ（２１４）に割当てられているインスタンスのリストを維持するよう構成される、請求項２１に記載のシステム。
【請求項２７】
前記複数のディスパッチャのうちのあるディスパッチャ（２１４）がある特定のカートリッジに関連付けられるブラウザリクエストを受取り、かつそのディスパッチャ（２１４）が維持するインスタンスのリストが、そのある特定のカートリッジ（２６０）のインスタンスが現在利用可能であることを示している場合、ディスパッチャ（２１４）はリソースマネージャ（２５４）からのブラウザリクエストに対するインスタンスをリクエストすることなくインスタンスにメッセージを送る、請求項２１に記載のシステム。
【発明の詳細な説明】
【０００１】
【発明の分野】
この発明はネットワーク接続されたコンピュータシステムにおけるサーバアーキテクチャに関し、より特定的には、種々のマシンを通じたユーザリクエストに対する動的処理を可能にするための分散アーキテクチャに関する。
【０００２】
【発明の背景】
ワールド・ワイド・ウェブ（World Wide Web）はインターネット（Internet）上のサーバのネットワークを含み、その各々は１つ以上のＨＴＭＬ（Hypertext Markup Language：ハイパテキスト・マークアップ言語）ページに関連付けられている。サーバに関連付けられているＨＴＭＬページは、情報と、そのサーバおよび（通常は）他のサーバ上の他の文書へのハイパテキストリンクとを提供する。サーバはハイパテキスト・トランスファー・プロトコル（ＨＴＴＰ：Hypertext Transfer Protocol）を用いてクライアントと通信す

る。サーバはクライアントからのＨＴＭＬページに関するリクエストを聞くことから、多くの場合、「リスナ」と称される。

【０００３】

ワールド・ワイド・ウェブのユーザはブラウザと呼ばれるクライアントプログラムを用いてリスナから情報をリクエストし、デコードし、表示する。ブラウザのユーザがあるＨＴＭＬページ上のリンクを選択すると、そのページを表示しているブラウザはそのリンク内で特定されるユニバーサル・リソース・ロケータ（ＵＲＬ：Universal Resource Locator）と関連付けられるリスナにインターネットを通じてリクエストを送る。そのリクエストに応答して、リスナはそのリクエストを発行したブラウザにリクエストされた情報を送信する。ブラウザは情報を受取り、受取った情報をユーザに示し、次のユーザリクエストを待つ。

【０００４】

従来、リスナに蔵置される情報は静的ＨＴＭＬページの形態である。静的ＨＴＭＬページは、ウェブブラウザからのリクエストに先立ち作成されリスナに蔵置される。リクエストに応答して、静的ＨＴＭＬページが単に記憶から読出され、リクエストを行なっているブラウザに送信される。現在、ブラウザリクエストに応答して動的オペレーションを行なうリスナを開発しようとする趨勢にある。たとえば、リスナはリクエストに応答する際にデータベースにクエリーを発行し、そのクエリーの結果を含むウェブページを動的に構成し、動的に構成されたＨＴＭＬページをリクエストを行なっているブラウザに送信してもよい。動的オペレーションを行なうために、リスナの機能性を向上または増強させなければならない。動的オペレーションをサポートするようリスナを拡張するためにさまざまなアプローチが開発されてきた。

【０００５】

ウェブブラウザからのリクエストに応答して動的オペレーションをもたらす１つのアプローチでは共通ゲートウェイ・インターフェイス（ＣＧＩ）を用いる。ＣＧＩはリスナとＣＧＩプログラムとの間で情報を転送するための仕様である。ＣＧＩプログラムはＣＧＩ仕様に従った、データを受け、かつ返すよう設計されるプログラムであればどのようなものであってもよい。そのプログラムはＣ、PerlまたはVisual Basicを含むどんなプログラム言語で書かれたものであってもよい。

【０００６】

ＣＧＩのアプローチでは、サーバが特定されたリクエストを受取るたびに別個のプロセス（ＣＧＩプログラムの別個のインスタンス）がイニシエートされるという不利な点がある。さらに、ＣＧＩプログラムはブラウザリクエストを受取ったリスナと同じマシン上で実行される。したがって、千ものこうしたリクエストを異なったユーザから受信すると、リスナを実行しているマシンで千ものプロセスがイニシエートされることとなり、サーバ上の使用可能なリソースを使い果たすこととなる。

【０００７】

ＣＧＩアプローチの第２の不利な点は、各リクエストに対して別個のプロセスがイニシエートされ実行され終了されることである。すなわち、第１の組の１０のリクエストの後に第２の組の１０のリクエストが続く場合、その第１の組のリクエストに対して第１の組の１０のプロセスがイニシエートされ終了され、その第２の組のリクエストに対して第２の組の１０のプロセスがイニシエートされ終了されることとなる。ＣＧＩではプロセスのイニシエートに関連するオーバーヘッドを避けるため、第１の１０のリクエストに対して用いられた１０のプロセスと同じものを第２の１０のリクエストを処理するのに用いることが許されていない。

【０００８】

リクエストに対する動的な応答を行なうための代替的なアプローチは「プラグイン」拡張を用いることである。プラグイン拡張はサーバに送られたメッセージをさまざまな段階で代行受信して特定のユーザリクエストに対してアプリケーションに特有の処理を行なう。サーバ側のプラグインはリスナおよび他のすべてのサーバ側プラグインと同じアドレス空

間において実行される。よって、プラグインを設計するアプリケーション開発者はリスナの下位レベルのオペレーションの詳細に精通していなければならない。さらに、リスナと同じアドレス空間でプラグインを実行することによりリスナを安全性および安定性の面で危険にさらすこととなる。障害のあるプラグインにより、他のプラグインまたはリスナそのものがクラッシュしたり、または予測できない態様で動作してしまうこともある。

【０００９】

プラグインアプローチの第２の問題は、ＣＧＩアプローチと同様に、すべてのプラグインオペレーションがリスナを実行しているマシンと同じマシン上で実行されることである。プラグイン拡張によって実行されるタスクは他のマシンに負荷を転嫁することができないため、プラグインによるアプローチのスケーラビリティ（scalability）は大幅に制限される。

背景および、この発明の実施例により解決される課題についてのさらなる詳細は、ナイジェル・エドワーズ（Nigel Edwards）およびオーエン・リーズ（Owen Rees）による「安全性の高いウェブサーバおよびゲートウェイ」（“High security Web servers and gateways” COMPUTER NETWORKS AND ISDN SYSTEMS, 29, (1997) 927-938.）と題された論文に記載されている。

【００１０】

【発明の概要】

分散ウェブアプリケーションサーバによりブラウザリクエストを処理するための方法およびシステムを提供する。分散環境により、ブラウザリクエスト内で特定されるオペレーションを実行するプロセス（「カートリッジインスタンス」）が、そのリクエストを受取るリスナおよびそのリクエストを発行するブラウザとは異なるマシン上で実行できる。カートリッジインスタンスがリスナとは異なるマシン上にあるため、リスナは障害のあるカートリッジインスタンスからよりよく隔離されることとなり、システムの信頼性および安全性が向上する。さらに、リスナが実行されるマシンと同じマシンでなく、数多くのマシン間でカートリッジインスタンスを実行する処理の負担を分散することによってシステムのスケーラビリティが大幅に増加する。複数のマシンにわたってカートリッジインスタンスの実行を分散できることから、いつどこで新しいカートリッジインスタンスを生じさせるかを決定するのに数多くのタイプの負荷の最適配分の技法を用いることができる。

【００１１】

この発明の一局面によれば、第１のマシン上で実行されるディスパッチャと第２のマシン上で実行されるリソースマネージャとを用いてオペレーションが実行される。第１のメッセージがディスパッチャからリソースマネージャへ送られる。この第１のメッセージはオペレーションを行なうことのできるある特定のカートリッジを識別している。第２のメッセージがリソースマネージャからディスパッチャへ送られる。この第２のメッセージはそのある特定のカートリッジのある特定のインスタンスを特定している。このある特定のインスタンスは第３のマシン上で実行されている。

【００１２】

ディスパッチャからの第３のメッセージはそのある特定のインスタンスに送られ、そのある特定のインスタンスにオペレーションを行なわせる。第１のマシン、第２のマシンおよび第３のマシンのうちの少なくとも２つは別個のマシンである。

【００１３】

この発明の別の局面によれば、ブラウザリクエストに関連するオペレーションを行なうためのシステムが提供される。システムは複数のウェブリスナに結合される複数のディスパッチャを含む。その複数のディスパッチャの各ディスパッチャはその複数のウェブリスナの対応するウェブリスナからその対応するウェブリスナが受取ったブラウザリクエストを受取る。

【００１４】

システムはさらに、仮想パスマネージャおよびリソースマネージャを含む。仮想パスマネージャはマシン間通信機構を介して複数のディスパッチャに結合される。仮想パスマネー

ジャは、複数のカートリッジのうちのどれがそのブラウザリクエストと関連付けられているのかをディスパッチャに示す。リソースマネージャはマシン間通信機構を介して複数のディスパッチャに結合される。リソースマネージャは、ディスパッチャからインスタンスに対するリクエストを受取ったことに応答して、複数のディスパッチャの各々のディスパッチャに複数のカートリッジのうちのあるカートリッジのインスタンスを割当てるよう構成される。

【００１５】

複数のディスパッチャは、リソースマネージャによってディスパッチャに割当てられるインスタンスへメッセージをマシン間通信機構を介して送るよう構成される。そのメッセージにより、インスタンスがブラウザリクエストに関連するオペレーションを行なうことになる。

【００１６】

この発明は限定によってではなく例によって、添付の図面において例示され、添付の図面において類似の参照番号は類似の要素を指す。

【００１７】

【好ましい実施例の詳細な説明】

ネットワークにわたってオペレーションを行なうための方法および装置を説明する。以下において説明のため、この発明の完全な理解をもたらすように数多くの特定の詳細事項を挙げている。しかしながら、当業者にはこの発明がこれらの特定の詳細事項がなくても実施できることが明らかになるであろう。場合によっては不必要にこの発明を曖昧にすることを避けるため周知の構造および装置をブロック図の形態で示している。

【００１８】

ハードウェアの概観

図１は、この発明の実施例が実装され得るコンピュータシステム１００を示すブロック図である。コンピュータシステム１００は、バス１０２または情報を通信するための他の通信機構と、バス１０２に結合され情報を処理するためのプロセッサ１０４とを含む。コンピュータシステム１００はまた、バス１０２に結合され情報およびプロセッサ１０４が実行すべき命令を蔵置するための、ランダムアクセスメモリ（ＲＡＭ）または他の動的記憶装置などのメインメモリ１０６を含む。メインメモリ１０６はまた、プロセッサ１０４が実行すべき命令の実行の間に一時変数または他の中間情報を蔵置するのに用いられてもよい。コンピュータシステム１００はさらに、バス１０２に結合され静的情報およびプロセッサ１０４に対する命令を蔵置するための読取専用メモリ（ＲＯＭ）１０８または他の静的記憶装置を含む。磁気ディスクまたは光ディスクなどの記憶装置１１０が設けられ、バス１０２に結合されて情報および命令を蔵置する。

【００１９】

コンピュータシステム１００はバス１０２を介して、陰極線管（ＣＲＴ）などの表示装置１１２に結合されコンピュータユーザに対して情報を表示してもよい。英数字および他のキーを含む入力装置１１４がバス１０２に結合され、情報およびコマンド選択をプロセッサ１０４に伝える。別のタイプのユーザ入力装置は、プロセッサ１０４に方向情報およびコマンド選択を伝え、かつ表示装置１１２上のカーソル移動を制御するためのマウス、トラックボールまたはカーソル方向キーなどのカーソル制御装置１１６である。この入力装置は典型的に２つの軸、すなわち第１の軸（たとえばｘ）および第２の軸（たとえばｙ）において２つの自由度を有し、これは装置が平面上の位置を特定できるようにする。

【００２０】

この発明はコンピュータシステム１００を用いてブラウザからのメッセージに応答して特定のオペレーションを行なうことに関する。この発明の一実施例によれば、プロセッサ１０４がメインメモリ１０６に含まれる１つ以上の命令の１つ以上のシーケンスを実行することに応答してコンピュータシステム１００がオペレーションを行なう。このような命令は、記憶装置１１０などの別のコンピュータ可読媒体からメインメモリ１０６に読込んでもよい。メインメモリ１０６に含まれる命令のシーケンスを実行することにより、プロセ

ッサ１０４がここに説明するプロセスステップを実行することとなる。代替の実施例では、ソフトウェア命令の代わりにまたはソフトウェア命令と併せて、結線回路を用いてこの発明を実現してもよい。すなわち、この発明の実施例はハードウェア回路およびソフトウェアの如何なる特定の組合せにも限定されない。

【００２１】

「コンピュータ可読媒体」という用語はここでは、実行のためにプロセッサ１０４に命令を提供するのにかかわるすべての媒体を指して用いる。このような媒体は、不揮発性媒体、揮発性媒体および伝送媒体を含むが、これらに限定されない数多くの形態を取り得る。不揮発性媒体には、たとえば、記憶装置１１０などの光ディスクまたは磁気ディスクが含まれる。揮発性媒体には、メインメモリ１０６などのダイナミックメモリが含まれる。伝送媒体には、バス１０２を構成するワイヤを含む、同軸ケーブル、導線および光ファイバが含まれる。伝送媒体はまた、電波および赤外線データ通信において生成されるもののような音波または光波の形態を取ることもある。

【００２２】

コンピュータ可読媒体の一般的な形態にはたとえば、フロッピーディスク、フレキシブルディスク、ハードディスク、磁気テープまたは何らかの他の磁気媒体、ＣＤ－ＲＯＭおよび何らかの他の光媒体、パンチカード、せん孔テープおよび孔のパターンを有する何らかの他の物理的媒体、ＲＡＭ、ＰＲＯＭ、ＥＰＲＯＭ、ＦＬＡＳＨ－ＥＰＲＯＭおよび何らかの他のメモリチップまたはカートリッジ、以下に説明するような搬送波、またはコンピュータが読むことのできるものであればどのような他の媒体でも含まれる。

【００２３】

実行のために１つ以上の命令の１つ以上のシーケンスをプロセッサ１０４に与える上でさまざまな形態のコンピュータ可読媒体がかかわり得る。たとえば、命令は初めに遠隔コンピュータの磁気ディスク上に担持されてもよい。遠隔コンピュータは命令をそのダイナミックメモリにロードし、その命令をモデムを用いて電話回線を介して送ることができる。コンピュータシステム１００がローカルに有するモデムが電話回線上でデータを受取り、赤外線送信機を用いてそのデータを赤外線信号に変換できる。バス１０２に結合される赤外線検出器は赤外線信号で運ばれるデータを受取り、そのデータをバス１０２上に出力することができる。バス１０２はそのデータをメインメモリ１０６へ運び、そこからプロセッサ１０４は命令を取出して実行する。メインメモリ１０６が受取った命令は場合により、プロセッサ１０４による実行の前または後に記憶装置１１０に蔵置されてもよい。

【００２４】

コンピュータシステム１００はまた、バス１０２に結合される通信インターフェイス１１８を含む。通信インターフェイス１１８はローカルネットワーク１２２に接続されるネットワークリンク１２０への両方向データ通信結合を提供する。たとえば、通信インターフェイス１１８は、対応するタイプの電話回線に対してデータ通信接続を提供するサービス統合デジタル網（ＩＳＤＮ）カードまたはモデムであってもよい。別の例としては、通信インターフェイス１１８は、互換性のあるローカル・エリア・ネットワーク（ＬＡＮ）に対するデータ通信接続を提供するローカル・エリア・ネットワーク（ＬＡＮ）カードであってもよい。また、無線リンクを実現してもよい。このような実現例のいずれにおいても、通信インターフェイス１１８はさまざまなタイプの情報を表わすデジタルデータストリームを運ぶ電気信号、電磁気信号または光信号を送受信する。

【００２５】

ネットワークリンク１２０は典型的に１つ以上のネットワークを介して他のデータ装置へのデータ通信を提供する。たとえば、ネットワークリンク１２０はローカルネットワーク１２２を介してホストコンピュータ１２４または、インターネットサービスプロバイダ（ＩＳＰ：Internet Service Provider）１２６により運用されるデータ装置に対する接続を提供してもよい。ＩＳＰ１２６はこれを受けて、現在一般的に「インターネット」１２８と称される世界規模のパケットデータ通信網を介するデータ通信サービスを提供する。ローカルネットワーク１２２およびインターネット１２８はともに、デジタルデータスト

リームを運ぶ電気信号、電磁気信号または光信号を用いる。コンピュータシステム１００へまたはコンピュータシステム１００からデジタルデータを運ぶ、さまざまなネットワークを介する信号およびネットワークリンク１２０上の、通信インターネット１１８を介する信号は情報を運ぶ搬送波の典型的な形態である。

【００２６】

コンピュータシステム１００はネットワーク、ネットワークリンク１２０および通信インターネット１１８を介して、メッセージを送り、プログラムコードを含むデータを受取ることができる。インターネットの例では、サーバ１３０はインターネット１２８、ＩＳＰ１２６、ローカルネットワーク１２２および通信インターネット１１８を介してアプリケーションプログラムに対するリクエストされたコードを送信するかもしれない。

【００２７】

受取られたコードはそれが受取られると同時にプロセッサ１０４によって実行され、および／または後に実行するために記憶装置１１０または他の不揮発性記憶に蔵置されてもよい。このように、コンピュータシステム１００は搬送波の形態でアプリケーションコードを得ることもある。

【００２８】

アプリケーションサーバ機能の概要

図２は、この発明の一実施例に従って設計されたシステム２００のブロック図である。システム２００は、ＨＴＴＰプロトコルに従ってインターネット２０８にわたって複数のリスナ２１０、２１６および２２２と通信する複数のブラウザ２０２、２０４および２０６を含む。ブラウザからのリクエストに応答して、リスナはウェブアプリケーションサーバ２８０にここでカートリッジと称するソフトウェアモジュールを呼出させる。例示の実施例では、ウェブアプリケーションサーバ２８０は３つのカートリッジ２３０、２３４および２３８の実行を開始している。

【００２９】

ウェブアプリケーションサーバ２８０は、トランスポート・アダプタ２１２、２１８および２２４、ディスパッチャ２１４、２２０および２２６、認証サーバ２５２、仮想パスマネージャ２５０、リソースマネージャ２５４、コンフィギュレーションプロバイダ２５６および複数のカートリッジ実行エンジン２２８、２３２および２３６を含む、数多くの構成要素からなる。ウェブアプリケーションサーバ２８０のさまざまな構成要素について以下により詳しく説明する。

【００３０】

重要なことであるが、ウェブアプリケーションサーバ２８０の数多くの構成要素はオブジェクトリクエストブローカ（Object Request Broker）２８２などのマシン間通信機構を介して通信する。マシン間通信機構を用いることにより、ブラウザリクエストにおいて特定されるオペレーションを実行するカートリッジインスタンスは、リクエストを受取るリスナおよびリクエストを発行するブラウザとは異なるマシン上で実行することができる。カートリッジインスタンスがリスナとは異なるマシン上にあるため、リスナは障害のあるカートリッジインスタンスからよりよく隔離されることとなり、システムの信頼性および安全性が向上する。さらに、リスナが実行されるマシンと同じマシンでなく、数多くのマシン間でカートリッジインスタンスを実行する処理の負担を分散することによってシステムのスケーラビリティが大幅に増加する。複数のマシンにわたってカートリッジインスタンスの実行を分散できることから、いつどこで新しいカートリッジインスタンスを生じさせるかを決定するのに数多くのタイプの負荷の最適配分の技法を用いることができる。

【００３１】

システム２００内の典型的なオペレーションには一般的に以下の段階が含まれる。

【００３２】

ブラウザがインターネット２０８にわたってリクエストを送信する。

リスナがそのリクエストを受取ってトランスポートアダプタを介してディスパッチャに渡す。

【００３３】
ディスパッチャは仮想パスマネージャ２５０と通信してそのリクエストを扱うのに適切なカートリッジを判定する。
【００３４】
この時点で、ディスパッチャは２つの事柄のうち１つを行なう。ディスパッチャがそのカートリッジに関する使用されていないインスタンスのことを知っている場合、ディスパッチャはそのインスタンスにリクエストを送る。そのカートリッジに関して使用されていないカートリッジインスタンスがない場合、ディスパッチャはリソースマネージャ２５４に新しいカートリッジインスタンスを作るよう依頼する。そのインスタンスがうまく起動した後に、カートリッジはその存在をリソースマネージャに知らせる。リソースマネージャ２５４はそこで、ディスパッチャに新しいインスタンスのことを知らせる。ディスパッチャはブラウザリクエストに基づいて変更されたリクエストを作り、その変更されたリクエストを新しいインスタンスに送る。
【００３５】
カートリッジインスタンスはその変更されたリクエストを処理して、ディスパッチャに応答を送る。
【００３６】
ディスパッチャはその応答をリスナを介してクライアントに戻す。
これらの段階について以下により詳しく説明する。
【００３７】
カートリッジ
カートリッジは特定のアプリケーションまたはシステム機能を行なうためのコードのモジュールである。カートリッジはシステム２００における分散の基本単位を形成する。この発明の一実施例によれば、カートリッジはユニバーサル・リソース・ロケータ（ＵＲＬ）を用いて名付けられる。すなわち、カートリッジ名には２つの部分、すなわちカートリッジが存在するサーバのＩＰアドレスと、コンパイルされたカートリッジコードのサーバディレクトリ構造における仮想パスとがある。カートリッジはＵＲＬを用いて名付けられるため、カートリッジ名空間はグローバルであり、文書などの他のウェブリソースにアクセスするのに用いられるのと同じメッセージ通信技術を用いてカートリッジにアクセスすることができる。
【００３８】
この発明の一実施例によれば、各カートリッジはすべてのカートリッジに対して共通の全体構造をもたらす標準インターフェイスを有する。この標準インターフェイスは、特定の条件の下でウェブアプリケーションサーバ２８０により呼出されるルーチンのインターフェイスを規定する。この発明の一実施例によれば、抽象カートリッジインターフェイスは以下のとおりである。
【００３９】
【表１】

```
interface Cartridge
{
boolean init();
boolean authenticate(in Principal user_passwd);
boolean exec(in Request req_obj, out Response resp_obj);
boolean shutdown();
}
```

【００４０】

init()ルーチンはカートリッジインスタンスを初期化する役割を果たす。このことには、いくつかのサブオブジェクトのコンストラクタを呼出すことと、スレッドを予め分岐させることと、他のすべての必要となる共用リソースを獲得することとが含まれ得る。

【００４１】

shutdown()ルーチンはすべてのリソースを片付けカートリッジインスタンスをシャットダウンする役割を果たす。カートリッジインスタンスに対して一度shutdown()ルーチンが呼出されると、直ちにカートリッジインスタンスは後続のリクエストを処理するのに利用不可能となる。

【００４２】

authenticate()ルーチンではカートリッジのサービスをリクエストしているクライアントがそのサービスを用いることを許可されているかどうかを検証する。

【００４３】

exec()ルーチンはカートリッジに対してすべてのサービスリクエストをディスパッチする包括的なやり方である。

【００４４】

例示的カートリッジ

各カートリッジは、明確に定義された機能を行なうカートリッジとして構成されるか、またはあるアプリケーションのためのインタプリタまたはルーチン環境として作用するプログラム可能なカートリッジとして構成される。プログラム可能なカートリッジの一例はＰＬ／ＳＱＬランタイムであり、これは構造的問合せ言語（Structured Query Language）を用いるオラクルベースのプログラム言語（Oracle-based Programming Language）（ＰＬ／ＳＱＬ）に従ってデータベースクエリーを処理するよう構成される。ＰＬ／ＳＱＬランタイムはデータベースクエリーを有するブラウザリクエストを実行する。ＰＬ／ＳＱＬランタイムは、たとえば、データリンクを介してカートリッジインスタンスと通信しているデータベースサーバにアクセスすることによってリクエストを処理する。

【００４５】

プログラム可能なカートリッジの別の例はＪＡＶＡランタイムインタプリタである。ＪＡＶＡランタイムインタプリタカートリッジのおかげで、ウェブアプリケーション開発者らはブラウザリクエストを処理するためのサーバ側のＪＡＶＡアプリケーションを書くことができる。同様に、たとえば第三者のサーバによって実行されるプロセスにアクセスするなどの動的オペレーションをもたらすために、カスタムサーバをカートリッジとして構成してもよい。

【００４６】

ディスパッチャ

ディスパッチャは、リスナが受取ったリクエストを適当なカートリッジへ経路選択するよう構成されるソフトウェアモジュールである。この発明の一実施例によれば、ディスパッチャはサーバ側のプログラム拡張（すなわち、「プラグイン」）として実現される。そのため、ディスパッチャはそれらが属するリスナと同じアドレス空間にロードされそこで実行される。ディスパッチャはコンパイル時にリスナコードとリンクされてもよいし、実行時に動的にロードされてもよい。

【００４７】

例示される実施例では、ディスパッチャ２１４、２２０および２２６はそれぞれ、リスナ２１０、２１６および２２２に関連付けられる。ディスパッチャ２１４、２２０および２２６は、リスナ２１０、２１６および２２２が受取ったブラウザリクエストをカートリッジへ選択的に経路制御する。

【００４８】

たとえば、リスナ２１０がユニフォーム・リソース・ロケータ（ＵＲＬ：Uniform Resource Locator）の形態で運ばれるブラウザリクエストをインターネット２０８から受取ると仮定する。このブラウザリクエストは、たとえばＨＴＭＬページまたは実行すべきオペレーションであるウェブオブジェクトに対する識別子の役割を果たす。リスナ２１０はその

ブラウザリクエストの解釈を全く試みることなくディスパッチャ２１４に受渡す。ブラウザリクエストを受取ると、ディスパッチャ２１４は、
（１） 仮想パスマネージャ２５０と通信してブラウザリクエストによって選択されるカートリッジを識別し、かつそのカートリッジが認証を要するかどうかを判定し、
（２） そのカートリッジが認証を必要とする場合、認証サーバ２５２と通信してブラウザがその選択されたカートリッジにアクセスすることが許されるかどうかを判定し、
（３） アクセスが許可された場合に、リソースマネージャと通信してブラウザリクエストを送るべき選択されたカートリッジの特定のインスタンスを定め、
（４） カートリッジの特定されたインスタンスによる実行のため、変更されたブラウザリクエストを作りかつディスパッチする。

【００４９】
変更されたブラウザリクエストは元のブラウザリクエストで受取った情報を再パッケージする。変更されたブラウザリクエストはたとえば、カートリッジの正しいオペレーションに必要なデータを含むコンテキストオブジェクトを含んでいてもよい。カートリッジの正しいオペレーションに必要なデータには、たとえば、そのブラウザリクエストが関連付けられるトランザクションを識別するトランザクションＩＤが含まれる。

【００５０】
カートリッジがリクエストに返答すると、そのカートリッジはディスパッチャにその返答を送り、ディスパッチャはこの返答をリスナに渡してそのリクエストを開始したブラウザへ送信されるようにする。

【００５１】
コンフィギュレーションプロバイダ
この発明の一実施例によれば、ウェブアプリケーションサーバ２８０とともに用いるべきカートリッジはまずウェブアプリケーションサーバ２８０に登録される。登録プロセスの間、そのカートリッジに関する情報がコンフィギュレーションプロバイダ２５６に供給される。コンフィギュレーションプロバイダ２５６はその情報をメタデータ２５８として蔵置してウェブアプリケーションサーバ２８０の構成要素が後からアクセスできるようにする。

【００５２】
メタデータ２５８はたとえば、
（１） カートリッジ名、
（２） 必要となるインスタンスの最小数、
（３） インスタンスの最大数、
（４） カートリッジを実装するコードの場所、
（５） コールバック機能（初期化、リクエストハンドラ、シャットダウン）を実行するためカートリッジ実行エンジンが用いるプログラムに依存した関数名、
（６） カートリッジを実行するためのマシンのリスト、
（７） カートリッジのためのアイドルタイム（カートリッジのインスタンスがシャットダウンされる前にアイドル状態でいることができる時間量）、
（８） オブジェクト識別子および
（９） もしあれば、カートリッジとともに用いるべき、認証サービスのタイプを示すデータ
を含んでいてもよい。

【００５３】
オブジェクト識別子は、対応するカートリッジによるオペレーションの実行をリクエストするためにブラウザリクエストが供給しなければならないデータを特定する。オブジェクトタイプは特定のワードまたはＵＲＬであってもよく、または「/java」などの仮想パスを含んでいてもよい。

【００５４】
一度コンフィギュレーションプロバイダ２５６が特定のカートリッジに対するコンフィギ

ュレーション情報をメタデータ２５８内に蔵置すると、そのカートリッジはウェブアプリケーションサーバ２８０が起動される際に自動的に登録される。

【００５５】

カートリッジがウェブアプリケーションサーバ２８０に登録された後、リソースマネージャ２５４はカートリッジに対する最小インスタンスをイニシエートする。一度最小数のインスタンスがイニシエートされると、ウェブアプリケーションサーバ２８０はブラウザリクエストを処理する準備が整う。

【００５６】

仮想パスマネージャ

上に述べたように、ディスパッチャは仮想パスマネージャ２５０と通信して各々の変更されたブラウザリクエストをどこに経路選択するかを定める。特定的には、各ブラウザリクエストは典型的にＵＲＬを含む。ブラウザリクエストを受取ると、ディスパッチャはそのリクエスト内のＵＲＬを仮想パスマネージャ２５０に送る。仮想パスマネージャ２５０はこれに応答して、もしあれば、ＵＲＬに関連付けられるカートリッジを識別するデータをディスパッチャに送る。

【００５７】

必要とされる情報をディスパッチャに供給するため、仮想パスマネージャ２５０はＵＲＬをカートリッジにマッピングするメタデータ２５８を調べる。ブラウザリクエストを受取ったことに応答して、仮想パスマネージャ２５０はマッピングデータを用いて、もしあれば、ブラウザリクエストに含まれるＵＲＬに対応するカートリッジを判定する。

【００５８】

たとえば、ブラウザリクエストが仮想パス「/java」で始まるＵＲＬリクエストである場合、マッピングにより、ＪＡＶＡインタプリタカートリッジが仮想パス「/java」を有するリクエストを取扱うよう構成されていることが示されるかもしれない。

【００５９】

この発明の一実施例によれば、仮想パスマネージャ２５０はまた、ＵＲＬに関連付けられたカートリッジが認証を必要とするかどうかを判定する。カートリッジが認証を要する場合、仮想パスマネージャ２５０はディスパッチャに送る応答において、認証が必要であることを示す。認証が必要でない場合、ディスパッチャは認証サーバ２５２を呼出すことなく、カートリッジのインスタンスに対する変更されたブラウザリクエストを作って送る。認証が必要である場合、変更されたリクエストをカートリッジのインスタンスに送出してもよいことを認証サーバが示した後に初めてディスパッチャはそのカートリッジのインスタンスに変更されたリクエストを送る。

【００６０】

リソースマネージャ

ウェブアプリケーションサーバ２８０のリソースマネージャ２５４は、カートリッジに対して予め定められた最小数のインスタンスをイニシエートし、各カートリッジのインスタンスの間で負荷の最適配分を行ない、所与のカートリッジの予め定められた最大数のインスタンスまで必要に応じてカートリッジの新しいインスタンスをイニシエートすることによってカートリッジの各々の実行を管理する。

【００６１】

たとえば、ある特定のカートリッジ（Ｃ１）のためのメタデータが以下の情報を含んでいると仮定する。

【００６２】

名称＝Ｃ１

最小インスタンス＝１０

最大インスタンス＝５０

ホストマシン＝Ｍ１、Ｍ２、Ｍ３

アイドルタイム＝３０秒

このメタデータに基づいて、カートリッジＣ１が初めに登録される際に、リソースマネー

ジャ２５４はＣ１の１０のインスタンスをイニシエートさせる。リソースマネージャ２５４はラベルＭ１、Ｍ２およびＭ３に関連付けられるマシン上の１０のインスタンスをイニシエートすることになる。

【００６３】

Ｃ１をアクセスするリクエストをディスパッチャから受取った際、リソースマネージャ２５４はＣ１のいずれかの既存のインスタンスが利用可能であるかどうかを判定する。リクエストを受取った際にＣ１のどのインスタンスも利用可能でない場合、リソースマネージャ２５４はＣ１の最大数のインスタンスが既に実行されているかどうかを判定する。Ｃ１の最大数のインスタンスが既に実行されていない場合、リソースマネージャ２５４は可能なホストマシンの１つの上でＣ１の新しいインスタンスをイニシエートし、リクエストを発行したディスパッチャにこの新しいインスタンスを識別するメッセージを送信する。Ｃ１の最大数のインスタンスが既に実行されている場合、リソースマネージャ２５４はリクエストを発行したディスパッチャに対し、その時点でそのリクエストを取扱うことができないことを示すメッセージを送る。

【００６４】

負荷の最適配分

この発明の一実施例によれば、リソースマネージャ２５４は１組の負荷の最適配分の規則を適用して、２つ以上の可能なホストマシンがある中、カートリッジのインスタンスをどこでイニシエートするかを定める。すなわち、上の例では、Ｍ１、Ｍ２およびＭ３はすべてカートリッジＣ１のインスタンスを実行することが可能である。Ｍ１、Ｍ２およびＭ３が同じ処理能力を有する場合、インスタンスを３つのマシンにわたって均等に分散するのが望ましいであろう。しかしながら、Ｍ１がＭ２およびＭ３の１０倍の処理力を有するなら、Ｃ１のインスタンスのすべてをある時点まではＭ１でイニシエートし、それからさらなるインスタンスをＭ１、Ｍ２およびＭ３の間で均等に分散させるのが望ましいであろう。

【００６５】

可能なマシンの間で負荷の最適配分を如何に行なうかを定める上でリソースマネージャ２５４を助けるため、各カートリッジに対して蔵置されるメタデータはさらなる詳細を含んでいてもよい。たとえば、メタデータは各マシンに対してインスタンスの別個の最小および最大数を特定してもよい。その場合リソースマネージャ２５４は、最大インスタンスに対する実際のインスタンスの比が最も小さなマシンはどれかに基づいて、マシン間で新しいインスタンスを分配することができる。

【００６６】

メタデータはまた、カートリッジを実行できるマシンのための順序を特定してもよい。その順序においてＮ＋１の位置にあるマシンは、その順序においてＮ番目の位置にあるマシンが既にその最大数のインスタンスを実行している場合にだけカートリッジのインスタンスを実行するのに用いられる。

【００６７】

カートリッジインスタンスステータスの管理

この発明の一実施例によれば、リソースマネージャ２５４は作られたカートリッジインスタンスを管理するためステート情報を維持する。ステート情報には、インスタンスを識別し、そのインスタンスを実行するマシンを識別し、そのインスタンスが割当てられたリスナを識別するデータが含まれる。

【００６８】

図５には、このステート情報を蔵置するためリソースマネージャ２５４が維持し得る表５００が示される。表５００には、インスタンス列５０２、カートリッジ列５０４、リスナ列５０６およびマシン列５０８が含まれる。表５００の各行は別個のカートリッジインスタンスに対応する。所与のカートリッジインスタンスに対する行内で、カートリッジ列５０４はカートリッジインスタンスに関連づけられるカートリッジを識別し、インスタンス列５０２はそのカートリッジインスタンスのインスタンス番号を示す。たとえば、行５１

０はカートリッジＣ１のインスタンスに対応する。このため、行５１０のカートリッジ列５０４はカートリッジＣ１を示す。行５１０のインスタンス列５０２は行５１０に関連づけられるカートリッジインスタンスがカートリッジＣ１のインスタンス１であることを示す。

【００６９】

リスナ列５０６は、ある行に関連づけられるカートリッジインスタンスが割当てられたリスナを示す。マシン列５０８は、ある行に関連づけられるカートリッジインスタンスが実行されているマシンを示す。たとえば、行５１０に関連づけられるカートリッジインスタンスはリスナ２１０に割当てられており、マシンＭ１上で実行されている。

【００７０】

リソースマネージャ２５４と同様に、各ディスパッチャはそのディスパッチャが設けられているリスナに割当てられたカートリッジインスタンスに対するステート情報を維持する。このようなステート情報はたとえば、図４に示されるような表４００において維持されてもよい。表５００と同様に、表４００にはそれぞれインスタンス番号およびカートリッジ識別子を保持するインスタンス列４０２およびカートリッジ列４０４が含まれる。しかしながら、表５００ではリソースマネージャ２５４によって割当てられるすべてのカートリッジインスタンスに対して１つずつエントリが含まれるのに対し、表４００では特定のリスナに割当てられたカートリッジインスタンスに対するエントリしか含まれない。たとえば、表４００は、表５００に列挙されたカートリッジインスタンスのうちリスナ２００に割当てられたもののみに対するエントリを含む。

【００７１】

インスタンス列４０２およびカートリッジ列４０４に加えて、表４００にはステータス列４０６が含まれる。各行に対し、ステータス列４０６はその行に関連づけられるインスタンスのステータスを示す値を保持する。たとえば、行４０８のステータス列４０６はカートリッジＣ１にインスタンス１が現在ビジーであることを示している。例示される実施例では、ステータス列４０６はカートリッジインスタンスがＢＵＳＹまたはＦＲＥＥのいずれかであることを示すフラグを保持する。カートリッジステータスの重要性について、リソースマネージャ２５４とディスパッチャ２１４および２２０とのオペレーションに関連して以下に説明する。

【００７２】

ディスパッチャとリソースマネージャとの相互作用

上述のように、ディスパッチャは特定のカートリッジに変更されたブラウザリクエストを送る必要がある際にリソースマネージャ２５４と通信する。この発明の一実施例によれば、ディスパッチャはまず、適当なカートリッジのインスタンスが（１）これに既に割当てられているかどうか、また（２）新しい変更されたブラウザリクエストを処理するのに利用可能であるかどうかを判定する。適当なカートリッジインスタンスがディスパッチャに既に割当てられており新しい変更されたブラウザリクエストを処理するのに現在利用可能である場合、ディスパッチャはリソースマネージャ２５４とさらに通信することなく、変更されたブラウザリクエストをカートリッジインスタンスへ転送する。

【００７３】

たとえば、仮想パスマネージャ２５０によればカートリッジＣ１が処理しなければならないブラウザリクエストをリスナ２１０が受取ったと仮定する。また、表４００がリスナ２１０に割当てられたカートリッジインスタンスの現在のリストおよびステータスを反映すると仮定する。リスナ２１０からブラウザリクエストを受取ると、ディスパッチャ２１４は表４００を調べてカートリッジＣ１のＦＲＥＥインスタンスがあるか探す。例示される表４００では、行４１０がカートリッジＣ１のインスタンス３が現在ＦＲＥＥであることを示している。したがって、ディスパッチャ２１４はさらにリソースマネージャ２５４と通信することなく変更されたブラウザリクエストを直接カートリッジＣ１のインスタンス３に転送する。変更されたブラウザリクエストを送ることに応答してディスパッチャ２１４は行４１０のステータス列４０６のステータス値をＢＵＳＹに変える。

【００７４】
リスナに現在利用可能である適当なカートリッジインスタンスが既に割当てられていない場合、カートリッジに関連づけられるディスパッチャはリソースマネージャ２５４からのカートリッジインスタンスをリクエストする。必要とされるカートリッジのインスタンスが利用可能でなく、かつ必要とされるカートリッジの既存のインスタンスの数が最大数より下であることをリソースマネージャ２５４が判定した場合、リソースマネージャ２５４は新しいカートリッジをイニシエートする。新しいカートリッジをイニシエートする際に、リソースマネージャ２５４は表５００に新しいカートリッジインスタンスのためのエントリを挿入する。
【００７５】
たとえば、カートリッジＣ３が処理しなければならないブラウザリクエストをリスナ２１０が受取ったと仮定する。また、カートリッジＣ３のインスタンス３がまだイニシエートされていないものと仮定する。こうした条件の下では、ディスパッチャ２１４はリソースマネージャ２５４にカートリッジＣ３のインスタンスに対するハンドルを求めるリクエストを送る。このリクエストに応答して、リソースマネージャ２５４はマシンＭ３上でカートリッジＣ３のインスタンス３をイニシエートする。さらに、リソースマネージャ２５４は行５１２に見られるエントリを表５００に挿入する。
【００７６】
表５００にカートリッジＣ３のインスタンス３のための行５１２を挿入した後、リソースマネージャ２５４はディスパッチャ２１４に新しく作られたインスタンスに対するハンドルを送り返す。このハンドルを受取ったことに応答して、ディスパッチャ２１４はそのステータス表４００に新しいインスタンスのためのエントリ（行４１２）を挿入する。ディスパッチャ２１４はそこで、変更されたブラウザリクエストをカートリッジＣ３のインスタンス３に送信する。
【００７７】
カートリッジインスタンスの解放
この発明の一実施例によれば、リスナはカートリッジインスタンスが未処理のブラウザリクエストに応答し終わった際にカートリッジインスタンスの所有権を自動的に解放することはない。たとえば、カートリッジＣ３のインスタンス３が変更されたブラウザリクエストを受取り、その変更されたブラウザリクエストを処理して、ディスパッチャ２１４に応答を送り返すと想定する。ディスパッチャ２１４はその応答をリスナ２１０に渡し、これがブラウザリクエストを発行したブラウザに送り返されるようにする。
【００７８】
この時点で、リスナ２１０はカートリッジＣ３のインスタンス３の所有権をもはや必要としない。しかしながら、カートリッジＣ３のインスタンス３の所有権をリソースマネージャ２５４に戻す代わりに、ディスパッチャ２１４は単に行４１２のステータス列４０６をＢＵＳＹからＦＲＥＥに変える。
【００７９】
行４１２のステータス列４０６の値をＦＲＥＥに変えることによって、カートリッジＣ３のインスタンス３がもはやリクエストを処理しておらず、よって後続のリクエストを処理する準備ができていることが示される。しかしながら、カートリッジＣ３のインスタンス３が利用可能であることを示す表４００はディスパッチャ２１４によって局所的に維持されているため、カートリッジＣ３のインスタンス３はリスナ２１０に到達する後続のブラウザリクエストに対してだけ利用可能である。リソースマネージャ２５４が維持する表５００の行５１２はカートリッジＣ３のインスタンス３をリスナ２１０が所有していることを引続き示す。
【００８０】
リスナはリクエストが処理されるたびにカートリッジインスタンスを自動的に解放するわけではないため、リソースマネージャ２５４とさまざまなディスパッチャとの間での通信に関連するオーバーヘッドは大幅に減少される。たとえば、カートリッジＣ３に伝えなけ

ればならない１０の連続したリクエストをリスナ２１０が受取るものと仮定する。１０のリクエストの各々に対してリソースマネージャ２５４と通信するのではなく、ディスパッチャ２１４は最初のリクエストに応答してリソースマネージャ２５４と通信してもよい。ディスパッチャ２１４は後続の９つのリクエストをリソースマネージャ２５４と通信することなく処理することができるが、これはディスパッチャ２１４が最初のリクエストを処理するＣ３の同じインスタンスを用いて９つの後続のリクエストを処理するためである。

【００８１】

各リクエストを処理する際にカートリッジインスタンスのリスナ所有権を自動的に解放しないことでウェブアプリケーションサーバ２８０の効率が向上されるとはいえ、リスナは無期限にカートリッジインスタンスの所有権を維持することはできない。たとえば、長い期間にわたって用いられなかったインスタンスはリソースマネージャ２５４に戻され、リソースを空けるためにその割当ての解除ができるようにすべきである。さらに、他のリスナがそのカートリッジのインスタンスを必要としているところに、１つのリスナが比較的長い時間使っていなかったカートリッジのインスタンスの所有権を維持することは効率的ではない。

【００８２】

したがって、リソースマネージャ２５４は、リスナに渡される各カートリッジインスタンスのための最大アイドルタイムを各リスナに伝える。最大アイドルタイムとは、リスナがカートリッジインスタンスの所有権を解放しなければならなくなる前にそのカートリッジインスタンスが使用されない状態でいられる最大の時間を示すものである。たとえば、カートリッジＣ３のインスタンス３のための最大量のアイドルタイムが１０分であることをリソースマネージャ２５４がリスナ２１０に示すものと仮定する。この情報に基づいて、リスナ２１０は、カートリッジＣ３のインスタンス３が１０分より長い間アイドル状態またはＦＲＥＥとならない限りにおいてカートリッジＣ３のインスタンス３を続けて用いてカートリッジＣ３に対するブラウザリクエストを処理することができる。

【００８３】

カートリッジＣ３のインスタンス３が１０分より長い間アイドル状態である場合、ディスパッチャ２１４は表４００から行４１２を取除き、リスナ２１０がカートリッジＣ３のインスタンス３の所有権を解放することをリソースマネージャ２５４にメッセージを送って知らせる。このメッセージに応答して、リソースマネージャ２５４は行５１２を更新して、カートリッジＣ３のインスタンス３がいずれのリスナによっても所有されておらず、よって別のリスナに再割当てされるか終了され得ることを示す。

【００８４】

代替の実施例では、カートリッジインスタンスのためのアイドルタイムが満了した際にもディスパッチャはカートリッジインスタンスを自動的に解放しない。代わりに、ディスパッチャはリソースマネージャ２５４にメッセージを送り、満了となったインスタンスを解放することを申し出る。リソースマネージャ２５４はこの申し出に応答して、リスナがカートリッジインスタンスを解放するようリクエストするか、またはリスナがその満了となったカートリッジインスタンスの所有権を持ち続けることを許可してもよい。

【００８５】

この発明の一実施例によれば、リソースマネージャ２５４は直ちに処理することのできないリクエストのキューを維持する。キュー内のリクエストを処理することが可能となると、そのリクエストはキューから取除かれ処理される。

【００８６】

たとえば、カートリッジＣ１が処理しなければならないブラウザリクエストをリスナ２２２が受取り、そのリスナ２２２にカートリッジＣ１のいずれのインスタンスも割当てられていなかったと仮定する。ディスパッチャ２２６はリソースマネージャ２５４にＣ１のインスタンスに対するリクエストを送る。さらに、Ｃ１の最大５０のインスタンスが許容され、Ｃ１の５０のインスタンスがリスナ２１０に割当てられたと仮定する。こうした条件の下では、リソースマネージャ２５４はリスナ２２２からのリクエストを処理することが

できない。このため、リソースマネージャ２５４はそのリクエストをキューにおく。リスナ２１０がＣ１のインスタンスを解放すると、リソースマネージャ２５４はリスナ２２２にＣ１のインスタンスが利用可能であることを伝える。

【００８７】

ある特定の条件の下で、リソースマネージャ２５４は強制的にリスナにカートリッジインスタンスを解放させることがある。たとえば、リソースマネージャ２５４はシステムオーバロード状況を検出し、これに応答して１組のカートリッジインスタンスを終了することがあり、これはそのカートリッジインスタンスが終了されることになることをそのカートリッジインスタンスが現在割当てられているリスナに知らせる前または知らせた後に行なわれる。

【００８８】

また、リソースマネージャ２５４はリスナの間での公平性の方針を実現するために強制的にリスナにカートリッジインスタンスを解放させることがある。たとえば、別のリスナが予め定められたしきい値より長い時間にわたってカートリッジのインスタンスを待っている場合、リソースマネージャ２５４は所与のカートリッジの最も多い数のインスタンスを保持するリスナにそのカートリッジのあるインスタンスを解放させてもよい。たとえば、リスナ２１０にカートリッジＣ１の５０のインスタンスが割当てられ、Ｃ１は最大で５０のインスタンスを有する場合、リソースマネージャ２５４は別のリスナからＣ１のインスタンスに対するリクエストを受取ってから１０秒後にリスナ２１０にＣ１のあるインスタンスを解放させてもよい。

【００８９】

カートリッジ実行エンジン

この発明の一実施例によれば、各カートリッジインスタンスはカートリッジ実行エンジンおよびカートリッジから構成される。カートリッジ実行エンジンは、ウェブアプリケーションサーバ２８０およびマシン間通信機構の複雑さからカートリッジを隔離するコードモジュールである。カートリッジは、関数テーブルにカートリッジ関数に対するポインタを蔵置することによってカートリッジ実行エンジンに利用可能となる。一実施例によれば、すべてのカートリッジが、上述の例示のカートリッジインターフェイスにおいて特定された関数を提供する。すべてのカートリッジが同じインターフェイスをサポートするようにさせることにより、単一の標準カートリッジ実行エンジンをすべてのカートリッジに対して用いることができる。

【００９０】

この発明の一実施例によれば、カートリッジは共用ライブラリとして実装され、カートリッジ実行エンジンは標準カートリッジインターフェイスを用いて共用ライブラリ内のルーチンを呼出す実行可能なプログラムである。カートリッジ実行エンジンはカートリッジとディスパッチャとの間でインターフェイスをもたらし、カートリッジの制御のフローを指示し、カートリッジが用いるサービスを提供する。

【００９１】

リソースマネージャ２５４が新しいカートリッジインスタンスの生成を必要とする場合、リソースマネージャ２５４はカートリッジ実行エンジンのインスタンスを生成させる。これに対し、このようにして作られたカートリッジ実行エンジンのインスタンスは適当なカートリッジのインスタンスを生成させる。リソースマネージャ２５４はたとえば、カートリッジが実行されるマシン上にある「カートリッジ実行エンジンファクトリ」と呼出すことによってカートリッジ実行エンジンのインスタンスを生成させることができる。カートリッジ実行エンジンのインスタンスはたとえば、そのカートリッジを構成する共用ライブラリ内のルーチンのうちの１つに呼出を行なうことによってカートリッジのインスタンスを生成させることができる。

【００９２】

図２に示されるように、ウェブアプリケーションサーバ２８０はカートリッジ２３０、２３４および２３８の各々に対しカートリッジ実行エンジン２２８、２３２および２３６を

含む。カートリッジ実行エンジンは標準のカートリッジインターフェイスを介してカートリッジへの呼出を行なうことにより対応するカートリッジのインスタンスの実行を制御する。カートリッジ実行エンジンとカートリッジとの間で基本的なコールバック機能を確立することにより、そのカートリッジをコールバック機能に応答するよう構成し、次にそのカートリッジをコンフィギュレーションプロバイダ２５６に登録することによってどんなカートリッジもウェブアプリケーションサーバ２８０に統合することができる。この事は以下に説明する。

【００９３】

このように、ＰＬ／ＳＱＬランタイムカートリッジがリクエストを処理するのに適当なカートリッジであるとディスパッチャ２１４が判定すると、ディスパッチャ２１４はそのＰＬ／ＳＱＬランタイムカートリッジに関連づけられるカートリッジ実行エンジンを含むカートリッジインスタンスにそのリクエストをディスパッチする。新しいインスタンスをイニシエートする必要がある場合、リソースマネージャ２５４は別個のアドレス空間においてＰＬ／ＳＱＬランタイムカートリッジの新しいインスタンスを作り、その新しいインスタンスのカートリッジ実行エンジン２２８へそのリクエストをディスパッチする。プログラムのインスタンスを実行するのに用いられるアドレス空間は、ウェブアプリケーションサーバ２８０の構成要素の１つ以上が実行されているコンピュータシステムのメモリ内にあるか、または別のコンピュータシステム上にあってもよい。

【００９４】

ディスパッチャからのメッセージに応答して、カートリッジ実行エンジンはカートリッジにリクエストハンドラコールバック機能を発行し、カートリッジにそのリクエストを処理させる。そのリクエストを処理するカートリッジはその結果をカートリッジ実行エンジンに返し、カートリッジ実行エンジンはその結果をディスパッチャに転送する。ウェブアプリケーションサーバ２８０がオペレーションにおいて障害を検出した場合にはカートリッジ実行エンジンはカートリッジのシャットダウン関数を発行する。

【００９５】

このように、カートリッジ実行エンジンは、実行すべき予め定められたオペレーションを特定するウェブアプリケーションサーバ２８０へのアプリケーションプログラミングインターフェイスを提供する。標準のカートリッジインターフェイスを使用することにより、カートリッジのプログラマが、カートリッジがともに用いられることになる特定のウェブリスナが用いるプロトコルとは無関係に、各カートリッジをウェブアプリケーションサーバ２８０へ高レベルで統合化するよう構成することができる。

【００９６】

トランスポートアダプタ

リスナは、サーバ側のプラグインが使用するためのプログラミングインターフェイスおよびプロトコルを提供することによってそのようなプラグインの使用を可能にする。残念なことに、リスナが提供するプログラミングインターフェイスおよびプロトコルはリスナごとに異なる。たとえば、ネットスケープ・サーバ・アプリケーション・プログラミング・インターフェイス（ＮＳＡＰＩ：Netscape Server Application Programming Interface）、インターネット・サーバ・アプリケーション・プログラミング・インターフェイス（ＩＳＡＰＩ：Internet Server Application Programming Interface）およびアプリケーション開発インターフェイス（ＡＤＩ：Application Development Interface）は現在リスナによりもたらされる別個のプログラミングインターフェイスの３つの例である。

【００９７】

トランスポートアダプタはウェブリスナが用いる私的プロトコルおよびインターフェイスからディスパッチャを隔離する。特定的には、各トランスポートアダプタはさまざまなリスナのプロトコルを認識し、リスナから受取ったブラウザリクエストを変換してリスナのプロトコルとは無関係の標準ディスパッチャプロトコルを有する変換されたブラウザリクエストを生じるよう構成される。同様に、トランスポートアダプタはディスパッチャからの返答をリスナのトランスポートプロトコルに変換する。

【００９８】
このように、トランスポートアダプタにより、ウェブアプリケーションサーバ２８０を異なるベンダのリスナとともに用いることが可能となる。さらに、トランスポートアダプタは異なるサーバアーキテクチャおよびオペレーティングシステムに対応するよう構成されてもよい。
【００９９】
ウェブアプリケーションサーバのオペレーション
図３Ａおよび図３Ｂはこの発明の一実施例によるブラウザリクエストに応答する方法を示すフロー図である。ステップ３５０においてリスナがブラウザリクエストを受取る。説明のために、このブラウザリクエストはブラウザ２０２により発行されリスナ２１０により受取られるものと仮定する。
【０１００】
ブラウザリクエストを受取ると、リスナ２１０はステップ３５２においてウェブアプリケーションサーバ２８０にそのリクエストを転送する。特定的には、リスナ２１０はリスナ２１０の私的プログラミングインターフェイスを用いてそのリクエストをトランスポートアダプタ２１２に渡す。トランスポートアダプタ２１２は、必要に応じてそのリクエストを標準ディスパッチャプログラミングインターフェイスを用いて変換しそのリクエストをディスパッチャ２１４に渡す。
【０１０１】
ディスパッチャ２１４は、仮想パスマネージャ２５０と通信することによって、ブラウザリクエストにより特定される仮想パスに基づいて、ステップ３５４においてブラウザリクエストに対応するリクエストオブジェクトタイプを識別する。ディスパッチャ２１４はステップ３５６において、そのリクエストオブジェクトタイプが識別可能なカートリッジに対応するかどうかを判定する。そのリクエストオブジェクトタイプが識別可能なカートリッジに対応していない場合、そのリクエストはステップ３５８においてリスナ２１０に戻される（図３Ｂを参照）。ステップ３５８においてそのリクエストが静的ＨＴＭＬページに対するリクエストであることをリスナ２１０が認識すると、リスナはその静的ＨＴＭＬページにアクセスし、そのＨＴＭＬページをステップ３６０においてブラウザ２０２に送る。ブラウザリクエストがリスナ２１０によって認識されない場合、ステップ３６０において返答がブラウザ２０２に送られそのリクエストが認識不可能であったことを示す。
【０１０２】
ステップ３５６において、（１）リクエストをカートリッジに送らなければならないことと、（２）リスナ２１０にそのカートリッジの現在ＦＲＥＥであるインスタンスが１つも割当てられていないこととをディスパッチャ２１４が判定すると、ディスパッチャ２１４は、そのブラウザリクエストを送ることのできるカートリッジ２３０のインスタンスが割当てられることになるリソースマネージャ２５４と通信する。図３Ｂに示されるステップ３６２では、リソースマネージャ２５４は、識別されたカートリッジのインスタンスが既存のインスタンスの数の中で利用可能である（所有されていない）かどうかを判定する。説明のため、このリクエストはカートリッジ２３０に関連づけられるものであり、カートリッジ２３０はＰＬ／ＳＱＬランタイムカートリッジであると仮定する。
【０１０３】
ステップ３６２においてリソースマネージャが利用可能なインスタンス、たとえばＰＬ／ＳＱＬランライム２３０のインスタンス２６０を識別すると、リソースマネージャ２５４はリクエストをインスタンス２６０に送るべきであることをディスパッチャ２１４に知らせる。ディスパッチャ２１４はそこで、変更されたブラウザリクエストを作りステップ３６８においてインスタンス２６０のカートリッジ実行エンジン２２８にその変更されたブラウザリクエストを送って、以下に説明するように利用可能なインスタンス２６０にそのリクエストを処理させる。
【０１０４】
しかしながら、ステップ３６２においてカートリッジ２３０のインスタンスが１つも利用

可能でない場合、リソースマネージャ２５４はステップ３６４において、既存のインスタンスの数が最大所定数を超えるかどうかを判定する。ステップ３６４において既存のインスタンスの数が最大所定数を超える場合、リソースマネージャ２５４はその時点ではそのリクエストを処理できないことをディスパッチャ２１４に示す。これに応答して、ディスパッチャ２１４はステップ３５８においてそのリクエストをリスナ２１０に返し、その後に、ウェブリスナ２１０はステップ３６０においてネットワークを介してブラウザ２０２へ返答を送り、そのリクエストが処理されなかったことを示す。

【０１０５】

代わりに、カートリッジインスタンスがリクエストを処理するためにその現時点で利用可能でない場合、リスナ２１０はそのカートリッジインスタンスに対する待ちリストにそのリクエストを入れてもよい。カートリッジインスタンスが利用可能となると、変更されたブラウザリクエストが待ちリストから取除かれカートリッジインスタンスに転送される。変更されたブラウザリクエストが予め定められた時間より長い間待ちリストに残っている場合、リスナ２１０はそのリクエストを待ちリストから取除いてブラウザ２０２にメッセージを送りそのリクエストが処理できなかったことを示すようにしてもよい。

【０１０６】

ステップ３６４において既存のインスタンスの数が最大所定数を超えない場合、リソースマネージャ２５４は識別されたプログラムの新しいインスタンスをイニシエートし、ブラウザリクエストに基づく変更されたブラウザリクエストを新しいインスタンスに送るべきであることをディスパッチャ２１４に知らせる。そこでディスパッチャ２１４は変更されたブラウザリクエストをその新しいインスタンスのカートリッジ実行エンジンへディスパッチする。

【０１０７】

たとえば、リソースマネージャ２５４がブラウザリクエストに応答してインスタンス２６０をイニシエートしたと仮定する。その初期化の間、ＰＬ／ＳＱＬランタイムのための蔵置された命令シーケンスにアクセスが行なわれ、ディスパッチャ２１４が実行しているアドレス空間とは別個のアドレス空間においてカートリッジ２３０の新しいインスタンス２６０が作られる。一実施例によれば、初期化はカートリッジ実行エンジン２２８をロードし、そのカートリッジ実行エンジンにカートリッジ２３０内の初期化ルーチンを呼出させることによって行なわれる。

【０１０８】

一度新しいインスタンス２６０が実行されると、ディスパッチャ２１４はステップ３６８においてその新しいインスタンス２６０に関連づけられているカートリッジ実行エンジン２２８にリクエストをディスパッチする。カートリッジ実行エンジン２２８は新しいインスタンス２６０にコールバックメッセージを送りそのリクエストの実行を要求する。そのコールバックメッセージにおいて、カートリッジ実行エンジン２２８はそのリクエストを処理する上でインスタンス２６０が必要とする全てのパラメータを渡す。このようなパラメータにはたとえば、パスワード、データベース探索キー、またはインスタンス２６０が実行する動的オペレーションのための他のどんな引き数を含んでいてもよい。

【０１０９】

次にインスタンス２６０はリクエストを実行する。ステップ３６８におけるインスタンスによるリクエストの実行の間、ディスパッチャ２１４はステップ３７０においてインスタンスを監視して障害が発生するかどうかを判定する。ステップ３７０においてディスパッチャ２１４が障害を検出すると、ディスパッチャ２１４はステップ３７２において対応するカートリッジ実行エンジン２２８を呼出し、その障害を有するインスタンス２６０を打ち切る。これに対し対応するカートリッジ実行エンジン２２８はＡＰＩを通して障害のあるインスタンスに対してシャットダウンコマンドを発行する。そのインスタンスはカートリッジ実行エンジン２２８によるシャットダウンコマンドに応答して、他のどんなアドレス空間における他のどんなプロセスにも影響を与えることなくシャットダウンする。

【０１１０】

ステップ３７０において障害が検出されない場合、ディスパッチャ２１４はステップ３７４において実行の完了の際にインスタンス２６０から返答を受取る。ステップ３７６においてディスパッチャ２１４はその返答をリスナ２１０に転送し、リスナ２１０は実行されたインスタンス２６０からの返答をもってブラウザに応答する。インスタンス２６０を実行した後、ステップ３７８においてディスパッチャ２１４はステップ３７８に示されるようにメモリ内にインスタンスを維持し、後続のリクエストの実行を可能にする。

【０１１１】

ウェブサーバの分散アーキテクチャ

重要なことであるが、ウェブアプリケーションサーバ２８０のさまざまな構成要素は、それら構成要素が同じアドレス空間において実行される必要がなく同じマシン上で実行されることさえ必要としない通信機構を用いて互いに通信する。例示される実施例では、ウェブアプリケーションサーバ２８０の構成要素はオブジェクトリクエストブローカ（ＯＲＢ：Object Request Broker）２８２を介して通信するよう構成される。オブジェクトリクエストブローカは「共通オブジェクトリクエストブローカ：アーキテクチャおよび仕様（ＣＯＲＢＡ）」（"Common Object Request Broker: Architecture and Specification (CORBA)"）に詳しく記載されている。ＣＯＲＢＡに関連するこのおよび他の文献はワールド・ワイド・ウェブのhttp://www.omg.orgにおいて見つけることができる。

【０１１２】

この発明の実施例はＣＯＲＢＡに従ったＯＲＢを介する通信に関連して説明するが、他のクロスプラットフォーム通信機構を用いてもよい。たとえば、ウェブアプリケーションサーバ２８０の構成要素は代わりに、遠隔手続き呼出（ＲＰＣ：Remote Procedure Calls）、ＵＮＩＸパイプ、Microsoft COMを用いて互いに通信してもよい。

【０１１３】

ウェブアプリケーションサーバ２８０のさまざまな構成要素がマシンから独立した通信機構を用いて互いに通信するため、構成要素が互いに対してどこに存在するかということに関して固有の制約はない。たとえば、リスナ２１０、２１６および２２２は同じマシン上で実行していてもよく、またはそれぞれがこのようなオペレーティングシステムを有する３つの完全に異なったマシン上で実行してもよい。同様に、認証サーバ２５２、仮想パスマネージャ２５０、リソースマネージャ２５４およびコンフィギュレーションプロバイダ２５６は同じマシン上で実行していてもよく、または４つの異なるマシン上で実行していてもよい。さらに、これらの４つの異なるマシンはリスナ２１０、２１６および２２２を実行する３つのマシンと少しも重複していなくてもよい。

【０１１４】

カートリッジ実行エンジン２２８、２３２および２３６は、オブジェクトリクエストブローカ２８２を介してウェブアプリケーションサーバ２８０の他の構成要素と通信するのに必要な論理のすべてを組込んでいる。したがって、カートリッジインスタンス自体の場所は通信機構によって本質的に制約されるものではない。すなわち、インスタンス２６０はこれがリクエストを受取るディスパッチャとは全く異なるマシンおよびオペレーティングシステムにおいて実行されてもよい。同様に、インスタンス２６０は、リソースマネージャ２５４、または同じウェブアプリケーションサーバ２８０によって管理される他のカートリッジのインスタンスを含む、ウェブアプリケーションサーバ２８０の他の構成要素のいずれとも異なるマシンおよびオペレーティングシステム上にあってもよい。

【０１１５】

重要なことだが、ウェブアプリケーションサーバ２８０が用いるカートリッジが享受する場所の独立性は、カートリッジそのものにおける何らかのカスタムプログラミングを介してではなくカートリッジ実行エンジンの通信論理を介して達成される。したがって、分散アプリケーションサーバ環境における実行のために特別にカートリッジを設計する必要がない。このように、カートリッジ設計者らは分散システムの複雑さから隔離され、カートリッジが作られる目的であるタスクと関連する論理に労力を集中させることができる。

【０１１６】

上述のように、異なるカートリッジの複数のインスタンスを管理してさまざまなユーザリクエストを処理するウェブアプリケーションサーバ２８０が提供される。各カートリッジインスタンスはリスナとは別個のメモリ空間において実行され、このためサーバ側のプラグインに関連のある安全性の問題を回避できる。さらに、リスナが受取ったブラウザリクエストを処理するのに用いられるカートリッジインスタンスはリスナとは全く異なるマシン上で実行されてもよい。このため、システムのスケーラビリティが向上すると同時に、いずれかの特定のマシンの負担が軽減される。

【０１１７】

さらに、ウェブアプリケーションサーバ２８０はまた、所与のカートリッジの各々に対するインスタンスの数を制御する。よって、サーバ側のリソースが制御され、インスタンスが制御不可能なほど生成されることによって多数のリクエストがいずれかのマシンを圧倒することがないことが確実となる。

【０１１８】

さらに、実行する準備の整ったインスタンスの最小数を維持することによって実行のスループットもまた向上する。単一の実行の後にインスタンスを終了して、それから後続のリクエストの実行のためにインスタンスを再び作るためにメモリにカートリッジを再ロードするのではなく、さらにインスタンスをイニシエートしてメモリ内に維持して後続のリクエストを実行するようにしてもよい。

【０１１９】

前述の明細書において、この発明をその特定の実施例に関連して説明した。しかしながら、この発明のより幅の広い範囲から逸脱することなくさまざまな修正および変更を行なうことができることが明らかになるであろう。したがって、明細書および図面は限定的な意味ではなく例示的な意味で考慮されるべきである。

【図面の簡単な説明】

【図１】　この発明の一実施例が実装され得るコンピュータシステムのブロック図である。

【図２】　この発明の一実施例による分散アプリケーションサーバのブロック図である。

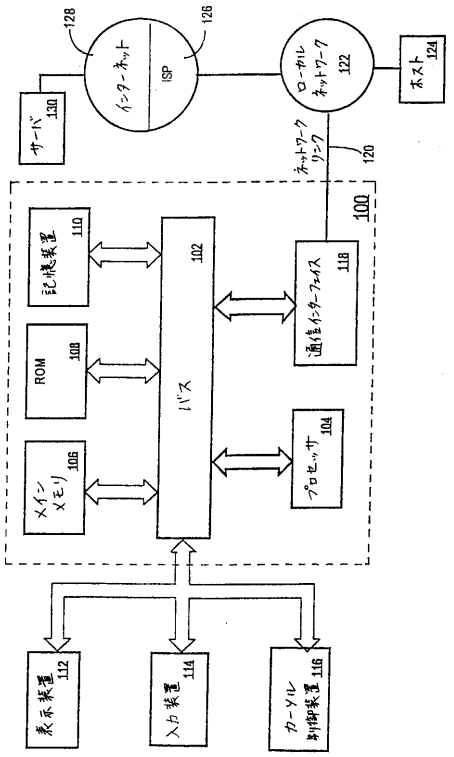
【図３Ａ】　この発明の一実施例によるブラウザリクエストを処理するためのステップを示すフローチャートの一部の図である。

【図３Ｂ】　この発明の一実施例によるブラウザリクエストを処理するためのステップを示すフローチャートの別の部分の図である。

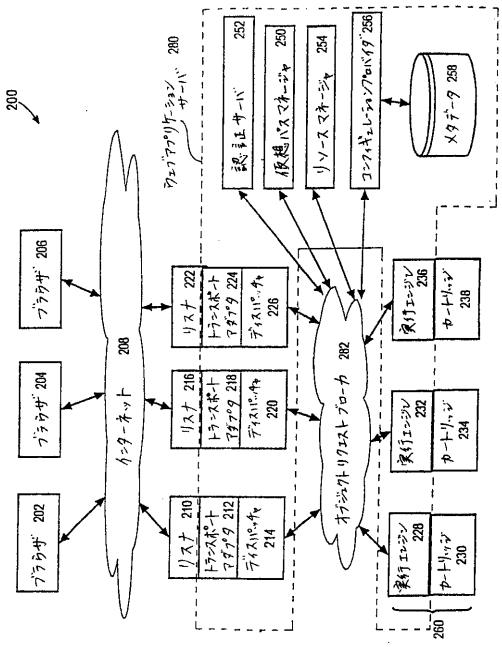
【図４】　この発明の一実施例によるディスパッチャが維持する情報を含む表のブロック図である。

【図５】　この発明の一実施例によるリソースマネージャが維持する情報を含む表のブロック図である。

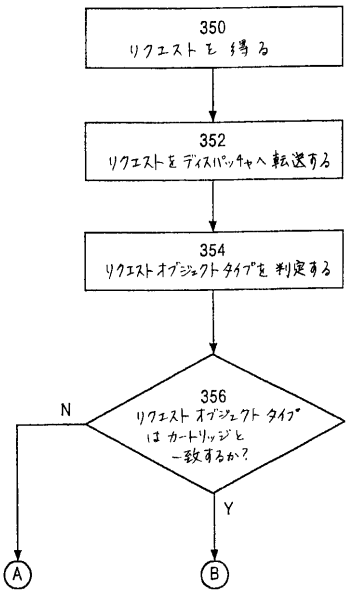
【図１】


インターネット
ISP
ローカルネットワーク 122
ホスト 124
サーバ 130
ネットワークリンク 120
100
記憶装置 110
ROM 108
メインメモリ 106
バス 102
通信インターフェイス 118
プロセッサ 104
表示装置 112
入力装置 114
カーソル制御装置 116

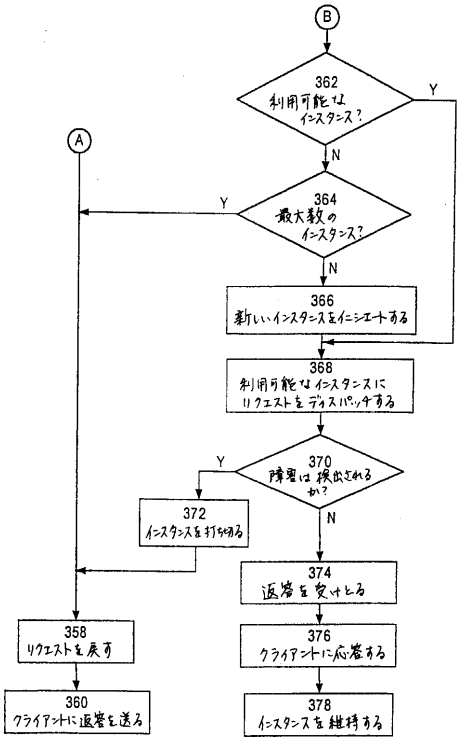
【図２】


200
アプリケーションサーバ 200
認証サーバ 252
仮想バスマネージャ 250
リソースマネージャ 254
コンフィギュレーションプロパティ 256
メタデータ 258
ブラウザ 206
ブラウザ 204
ブラウザ 202
ネットワーク 208
リスナ 222
トランスポートアダプタ 224
ディスパッチャ 226
リスナ 216
トランスポートアダプタ 218
ディスパッチャ 220
リスナ 210
トランスポートアダプタ 212
ディスパッチャ 214
オブジェクトリクエストブローカ 282
実行エンジン 236
カートリッジ 238
実行エンジン 232
カートリッジ 234
実行エンジン 228
カートリッジ 230
260

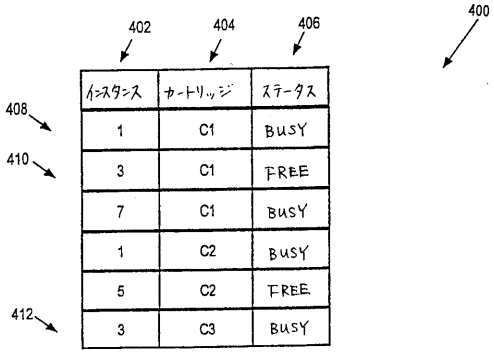
【図３Ａ】


350
リクエストを得る
352
リクエストをディスパッチャへ転送する
354
リクエストオブジェクトタイプを判定する
356
リクエストオブジェクトタイプはカートリッジと一致するか?
N
Y
A
B

【図３Ｂ】

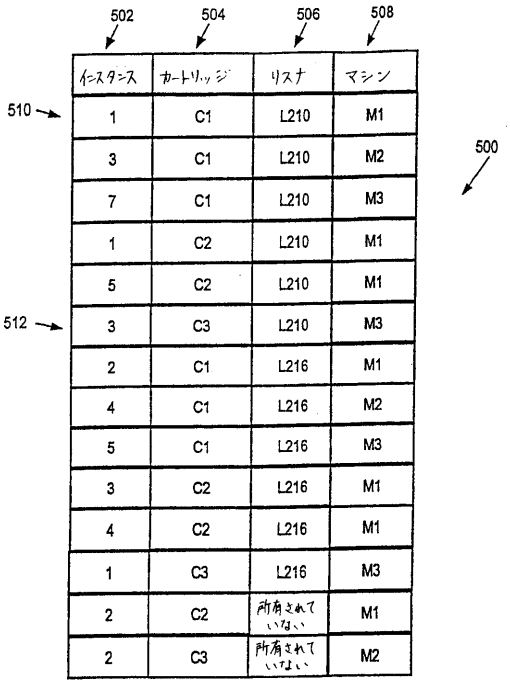

B
362
利用可能なインスタンス?
Y
N
A
364
最大数のインスタンス?
Y
N
366
新しいインスタンスをイニシエートする
368
利用可能なインスタンスにリクエストをディスパッチする
370
障害は検出されるか?
Y
N
372
インスタンスを打ち切る
374
返答を受け取る
358
リクエストを戻す
376
クライアントに応答する
360
クライアントに返答を送る
378
インスタンスを維持する

【図４】



インスタンス	カートリッジ	ステータス
1	C1	BUSY
3	C1	FREE
7	C1	BUSY
1	C2	BUSY
5	C2	FREE
3	C3	BUSY

【図５】



インスタンス	カートリッジ	リスナ	マシン
1	C1	L210	M1
3	C1	L210	M2
7	C1	L210	M3
1	C2	L210	M1
5	C2	L210	M1
3	C3	L210	M3
2	C1	L216	M1
4	C1	L216	M2
5	C1	L216	M3
3	C2	L216	M1
4	C2	L216	M1
1	C3	L216	M3
2	C2	所有されていない	M1
2	C3	所有されていない	M2

フロントページの続き

(74)代理人 100098316
弁理士 野田 久登
(74)代理人 100109162
弁理士 酒井 將行
(72)発明者 チョウ, ツン－ジェン
アメリカ合衆国、９４０２５ カリフォルニア州、メンロ・パーク、ラ・ロマ・ドライブ、１５
(72)発明者 アドゥヌトゥラ, セシュ
アメリカ合衆国、９４５５５ カリフォルニア州、フレモント、フェリックス・テラス、３４５４３
(72)発明者 アナンド, マラ
アメリカ合衆国、９４３０１ カリフォルニア州、パロ・アルト、フォレスト・アベニュ、５０１
(72)発明者 シャルマ, アンクール
アメリカ合衆国、９４００２ カリフォルニア州、ベルモント、オールド・カントリー・ロード、９５１－2、ナンバー・３６５
(72)発明者 チェン, エレイン
アメリカ合衆国、９４０６３ カリフォルニア州、レッドウッド・シティ、ブラッドフォード・ストリート、７８０、ナンバー・１５
(72)発明者 ナコダ, シェザード
アメリカ合衆国、９４３０６ カリフォルニア州、パロ・アルト、グラント・アベニュ、４５５、ナンバー・１４

審査官 殿川 雅也

(56)参考文献 国際公開第９７／０４０４５７（WO, A1）
国際公開第９６／００２８８２（WO, A1）
Ron-Chung Hu, et al., Navigation Server:A Highly Parallel DBMS on Open Systems, Data Engineering, 1995. Proceedings of the Eleventh International Conference on, IEEE, １９９５年 3月１０日, pp. 184 - 185
木村 春生, これだけは知っておきたいインターネット／イントラネット最新一旬一最強用語集 データベース, ＰＣ ＷＯＲＫ！, 日本, 株式会社毎日コミュニケーションズ, １９９７年５月１８日, 第4巻、第5号, 第４９頁乃至第５１頁
山本 美桜, ＮＣＡ、うぶ声をあげた新しいシステムパラダイム, ｎｅｔ ＰＣ, 日本, 株式会社アスキー, １９９７年 5月, 第2巻、第5号, 第１０８頁乃至第１１３頁
一志 達也, 新時代を迎えるインターネット第一回 Oracle Web Application Server 3.0, Ｗｉｎｄｏｗｓ ＮＴ ＰＯＷＥＲＳ, 日本, 株式会社ビー・エヌ・エヌ, １９９７年１０月 9日, 第1巻、第6号, 第２０２頁乃至第２０５頁

(58)調査した分野(Int.Cl., ＤＢ名)
G06F 9/46 - 9/54