(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3944154号
(P3944154)
(45)発行日　平成19年7月11日(2007.7.11)　(24)登録日　平成19年4月13日(2007.4.13)

(51)Int.Cl.　F I
G06F 9/48 (2006.01)　G06F 9/46 452G

請求項の数 18 (全 26 頁)

(21)出願番号　特願2003-396275(P2003-396275)
(22)出願日　平成15年11月26日(2003.11.26)
(65)公開番号　特開2004-213624(P2004-213624A)
(43)公開日　平成16年7月29日(2004.7.29)
審査請求日　平成15年11月26日(2003.11.26)
(31)優先権主張番号　10/334768
(32)優先日　平成14年12月31日(2002.12.31)
(33)優先権主張国　米国(US)

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100086243
弁理士　坂口　博
(74)代理人　100091568
弁理士　市位　嘉宏
(74)代理人　100108501
弁理士　上野　剛史
最終頁に続く

(54)【発明の名称】マルチスレッド・サーバ内のスレッド・プールを動的に調整する方法およびシステム

(57)【特許請求の範囲】
【請求項１】
マルチスレッド・サーバ内のスレッド・プールを動的に調整する方法であって、
マルチスレッド・サーバ上で動的に変更可能な作業負荷に関して基準線となる実行時統計を集めるステップであって、前記基準線となる実行時統計は前記動的に変更可能な作業負荷の第１の複数の要求を実行することに関係し、前記要求は複数のスレッド・プールによって処理されるものであるステップと、
タイマ期間経過に応答して前記複数のスレッド・プールについて一のスレッド・プールを追加または除去することにより、前記スレッド・プールをプログラムに基づいて変更するステップと、
前記動的に変更可能な作業負荷に関して新しい実行時統計を集めるステップであって、前記新しい実行時統計は前記動的に変更可能な作業負荷の第２の複数の要求を実行することに関係し、前記要求は前記プログラムに基づいて変更されたスレッド・プールによって処理されるものであるステップと、
前記スレッド・プールの変更後の前記タイマ期間経過後に取得した前記新しい実行時統計と前記基準線となる実行時統計とを比較して、前記プログラムに基づいた変更の結果、性能が低下したかまたは改善されなかったことを示す場合、前記プログラムに基づいた前記スレッド・プールの変更をプログラムに基づいて逆転させるステップとを含む方法。
【請求項２】
前記新しい実行時統計を集める前記ステップの前に、前記追加されたスレッド・プール

を反映するために、前記複数のスレッド・プールへの前記動的に変更可能な作業負荷の割振りをプログラムに基づいて再バランシングするステップをさらに含む、請求項１に記載の方法。

【請求項３】

前記プログラムに基づいて再バランシングするステップは、前記スレッド・プールによって処理される要求の実行時間の上限を計算するステップをさらに含む、請求項２に記載の方法。

【請求項４】

前記プログラムに基づいて逆転させるステップは、

前記スレッド・プールが追加された場合には、追加されたスレッド・プールをプログラムに基づいて除去するステップと、

前記スレッド・プールが除去された場合には、除去されたスレッド・プールをプログラムに基づいて追加するステップと、

前記除去または追加されたスレッド・プールを反映するために、前記複数のスレッド・プールへの前記動的に変更可能な作業負荷の割振りを再バランシングするステップとを、さらに含む、請求項１に記載の方法。

【請求項５】

マルチスレッド・サーバ内のスレッド・プールを動的に調整する方法であって、

マルチスレッド・サーバ上で動的に変更可能な作業負荷に関して基準線となる実行時統計を集めるステップであって、前記基準線となる実行時統計は前記動的に変更可能な作業負荷の第１の複数の要求を実行することに関係し、前記要求は複数のスレッド・プールによって処理されるものであるステップと、

前記複数のスレッド・プールのうちの選択された１つに割り当てられたスレッド数を増分または減分することにより、前記スレッド・プールをプログラムに基づいて変更するステップと、

前記動的に変更可能な作業負荷に関して新しい実行時統計を集めるステップであって、前記新しい実行時統計は前記動的に変更可能な作業負荷の第２の複数の要求を実行することに関係し、前記要求は前記プログラムに基づいて変更されたスレッド・プールによって処理されるものであるステップと、

前記複数のスレッド・プールの変更後のタイマ期間経過後に取得した前記新しい実行時統計と前記基準線となる実行時統計とを比較して、前記プログラムに基づいた変更の結果、性能が低下したかまたは改善されなかったことを示す場合に前記プログラムに基づいた前記スレッド・プールの変更をプログラムに基づいて逆転させるステップとを含む方法。

【請求項６】

前記プログラムに基づいて逆転させるステップは、前記スレッド・プールのうちの前記選択された１つに割り当てられた前記スレッド数を減分するステップをさらに含む、請求項５に記載の方法。

【請求項７】

前記プログラムに基づいて逆転させるステップは、前記スレッド・プールのうちの前記選択された１つに割り当てられた前記スレッド数を増分するステップをさらに含む、請求項６に記載の方法。

【請求項８】

前記基準線となる実行時統計および前記新しい実行時統計は、前記動的に変更可能な作業負荷を含む前記要求を処理するための平均実行時間および平均待ち行列時間である、請求項１または請求項５に記載の方法。

【請求項９】

前記平均実行時間および平均待ち行列時間はＵＲＩ名ごとに維持される、請求項８に記載の方法。

【請求項１０】

前記平均実行時間および平均待ち行列時間は処理宛先ごとに維持される、請求項８に記

載の方法。
【請求項１１】
前記基準線となる実行時統計および新しい実行時統計は、前記動的に変更可能な作業負荷を含む前記要求を処理するための、平均実行時間、実行時間の標準偏差、平均待ち行列時間、および待ち行列時間の標準偏差である、請求項１または請求項５に記載の方法。
【請求項１２】
マルチスレッド・サーバ内のスレッド・プールを動的に調整するシステムであって、
マルチスレッド・サーバ上で動的に変更可能な作業負荷に関して基準線となる実行時統計を集めるための手段であって、前記基準線となる実行時統計は前記動的に変更可能な作業負荷の第１の複数の要求を実行することに関係し、前記要求は複数のスレッド・プールによって処理されるものである手段と、
タイマ期間経過に応答して前記複数のスレッド・プールについて一のスレッド・プールを追加または除去することにより、前記スレッド・プールをプログラムに基づいて変更するための手段と、
前記動的に変更可能な作業負荷に関して新しい実行時統計を集めるための手段であって、前記新しい実行時統計は前記動的に変更可能な作業負荷の第２の複数の要求を実行することに関係し、前記要求は前記プログラムに基づいて変更されたスレッド・プールによって処理されるものである手段と、
前記複数のスレッド・プールの変更後の前記タイマ期間経過後の前記新しい実行時統計と前記基準線となる実行時統計とを比較して、前記プログラムに基づいた変更の結果、性能が低下したかまたは改善されなかったことを示す場合に前記プログラムに基づいた前記スレッド・プールの変更をプログラムに基づいて逆転させるための手段とを含むシステム。
【請求項１３】
前記追加されたスレッド・プールを反映するために、前記新しい実行時統計を集めるための手段のオペレーションの前に、前記複数のスレッド・プールによって処理される要求の実行時間の上限を計算することによって、前記複数のスレッド・プールへの前記動的に変更可能な作業負荷の割振りをプログラムに基づいて再バランシングするための手段をさらに含む、請求項１２に記載のシステム。
【請求項１４】
前記新しい実行時統計を集めるための手段のオペレーションの前に、前記除去されたスレッド・プールを反映するために、前記複数のスレッド・プールへの前記動的に変更可能な作業負荷の割振りをプログラムに基づいて再バランシングするための手段をさらに含む、請求項１３に記載のシステム。
【請求項１５】
前記プログラムに基づいて逆転させるための手段は、
他のスレッド・プールをプログラムに基づいて追加するための手段と、
前記追加されたスレッド・プールを反映するために、前記複数のスレッド・プールへの前記動的に変更可能な作業負荷の割振りを再バランシングするための手段とをさらに含む、請求項１３に記載のシステム。
【請求項１６】
前記基準線となる実行時統計および前記新しい実行時統計は、前記動的に変更可能な作業負荷を含む前記要求を処理するための平均実行時間および平均待ち行列時間である、請求項１２に記載のシステム。
【請求項１７】
前記平均実行時間および平均待ち行列時間はＵＲＩ名ごとに維持される、請求項１６に記載のシステム。
【請求項１８】
前記基準線となる実行時統計および前記新しい実行時統計は、前記動的に変更可能な作業負荷を含む前記要求を処理するための、平均実行時間、実行時間の標準偏差、平均待ち

行列時間、および待ち行列時間の標準偏差である、請求項１２に記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明はコンピュータ・ソフトウェアに関し、より詳細には、（たとえば、マルチスレッド・サーバ環境においてサーバの作業負荷のバランスを取るために）実行時にプログラムに基づいてスレッド・プールを調整することによる、方法およびシステムに関する。
【背景技術】
【０００２】
クライアント／サーバ・コンピューティングの普及が近年急速に進んでいるが、これは、公衆インターネットおよび「Ｗｏｒｌｄ　Ｗｉｄｅ　Ｗｅｂ」（または単に「Ｗｅｂ」）として知られるそのサブセットの、ビジネスおよび家庭での利用が増えたことに大きく起因している。企業イントラネットおよびエクストラネットなどの他のタイプのクライアント／サーバ・コンピューティング環境も、ますます普及しつつある。その解決策として、プロバイダは高度なＷｅｂベースのコンピューティングを配信することに焦点を当て、開発されるソリューションの多くは他のクライアント／サーバ・コンピューティング環境に適応可能なものである。したがって、本明細書でインターネットおよびＷｅｂに言及するのは、例示のためであって限定するためではない。（さらに本明細書では、「インターネット」、「Ｗｅｂ」、および「Ｗｏｒｌｄ　Ｗｉｄｅ　Ｗｅｂ」という用語は区別なく使用される。）
【０００３】
毎日、何百万人もの人々が、個人の楽しみやビジネスあるいはその両方に、インターネットを使用している。電子情報およびビジネス・サービスの消費者として、今や人々は地球規模のレベルでソースに簡単にアクセスすることができる。人間のユーザがインターネットを介してソフトウェア・アプリケーションと対話し、コンテンツを要求する場合、応答を返す際の遅延または効率の悪さは、ユーザの満足感にかなりの悪影響を与える可能性があり、ユーザが別のソースに切替えてしまうことすらある。したがって、要求されたコンテンツを即時に効率良く送達することは、ユーザを満足させる上で不可欠であり、ネットワークのサーバ側にあるシステムができる限り効率良く動作するよう保証することが重要である。
【０００４】
これまでの経験によれば、この種の環境で様々なクライアントに関する要求を処理するアプリケーション・サーバでは、受け取られる様々な要求全体にわたって最高のスループットおよび応答時間を提供するために、通常、リソースの使用量に制約を与えることが必要である。関心の対象である主なリソースの１つが実行スレッド（以下、同様の意味で単に「スレッド」と呼ぶ）である。スレッドを制約なしに作成、使用、および廃棄すると、当分野で知られた様々な理由で応答時間およびスループットの両方を損なう可能性がある。たとえば、スレッドをあまり多く作成しすぎると、スレッドを管理するためのシステム・オーバヘッドが許容できないほどに高くなることがあり、またこれらのスレッドに関するシステムの状態および他の情報を格納するために必要なメモリ量が多くなりすぎることがある。さらに、限られたリソースに対して多くのスレッドが待ち行列に入ると、通常はそれらのリソースにスラッシングを生じさせるため、共用リソースの競合が使用可能なスレッドの数を制限する主な理由である。ただし他方では、使用可能なスレッド数が少なすぎると、入ってくる要求がスレッドに割り当てられるまでに長い時間待機し、それによってエンド・ユーザへの応答時間が長くなる可能性がある。
【０００５】
したがって、システム内のスレッド数を調整することが有用である。本明細書では、作成されたが廃棄されていないスレッド・セットのことを「スレッド・プール」と呼ぶ。特定のクライアント／サーバ環境でスレッド・プールに対して作成されるスレッド数は、サーバを初期設定する際に、しばしばユーザ（たとえばシステム管理者）によって構成パラ

メータとして指定される。通常、アプリケーションが適度以上に酷使される環境では、所与のアプリケーション・セットに対するスレッド・プールの調整は反復オペレーションであり、その結果、スループットおよび応答時間を向上させる試みにおいてスレッド・プールがサイズ変更される。

【０００６】

均一の作業負荷では、要求の有する全システム応答時間がほぼ同じである場合が多く、スレッド・プールを繰り返しサイズ変更することが、システムの性能を向上させるために良好に働く。同様に、作業負荷に様々なタイプがミックスされた要求が含まれていても、その様々な要求が同様の応答時間を有する場合には、この種のオペレーションのサイズ変更もかなり良好に働く。ただし、様々な応答時間がミックスされている作業負荷の場合、問題はさらに複雑である。

【０００７】

様々な平均応答時間を有する要求タイプからなる作業負荷を伴う、スレッド数の制約された単一のスレッド・プールが使用される場合、要求が（平均して）適正な時間内に処理される、スレッド・プールの「最適サイズ」を見つけることが可能である。ただし、このようにミックスされた作業負荷に対して単一のスレッド・プールを使用することは、次善策となる傾向がある。具体的に言えば、この方法は、より短い実行時間を有する要求の応答時間を不均衡に長くする。

【０００８】

この現象が生じる理由は、上記のように、アプリケーション・サーバの単一のスレッド・プールに制約を与えることは、そのアプリケーション・サーバ内でのリソース利用率を制御するために極めて重要ではあるが、単一のスレッド・プールは、より長い実行時間を有する要求で一杯になる傾向もあり、したがって、より短い実行時間を有する要求は事実上締め出されることになる。実行時間の長い要求が爆発的に増えると、本質的に、実行時間の短い要求が単一の制約されたスレッド・プールからスレッドに割り当てられるのを妨げることになる。さらに、スレッドがスレッド・プールから割り当てられたときに、たとえそのスレッドが特定の要求をきわめて迅速に処理できるとしても、要求はスレッドが割り当てられるまでかなり長い間待機しなければならない可能性がある。したがって、こうした要求に関してエンド・ユーザ（または、より一般には要求者）が知覚する応答時間は、極端に長い可能性がある。

【０００９】

これら従来技術の問題を克服する技法が求められている。

【特許文献１】ＵＳ５，６６４，１０６

【特許文献２】ＵＳ６，４２７，１６１

【非特許文献１】「データベース・システムに関する目標指向の動的バッファ・プール管理」（"Goal-oriented dynamic buffer pool management for data basesystem"），IEEE、１９９５年８月　１９１～１９８、チェン・ヤオ・チャン等(Jen-Yao Chung, etal.)

【発明の開示】

【発明が解決しようとする課題】

【００１０】

本発明の一目的は、クライアント／サーバ・ネットワークでの性能を向上させることである。

【００１１】

本発明の他の目的は、マルチスレッド・サーバの性能を向上させることである。

【００１２】

本発明の他の目的は、スレッド・プールからスレッドへの要求をスケジューリングするための高度な技法を提供することである。

【００１３】

本発明の他の目的は、マルチスレッド・サーバ環境で動的に作業負荷のバランスをとるための技法を定義することである。

【００１４】
本発明の他の目的は、様々な平均応答時間を有する作業負荷に対するサーバの性能を向上させるために、スレッド・プールを動的に調整するための技法を定義することである。
【００１５】
本発明の他の目的は、実行のための待機時間を減らすために、要求が必要とする実行時間を短くすることのできる技法を定義することである。
【００１６】
本発明の他の目的は、実行時間の短い要求に関する応答時間を短くするために、実行リソース・セット全体にわたり、作業負荷をプログラムに基づいて分配するための技法を定義することである。
【課題を解決するための手段】
【００１７】
本発明の他の目的および利点は、一部は以下の説明および図面に記載され、一部は説明から明らかになるか、または本発明の実施によって習得することができる。
【００１８】
上記の目的を達成するため、および本明細書におおまかに記載された本発明の目的に従って、本発明は、マルチスレッド・サーバのスレッド・プールを動的に調整するための方法およびシステムを提供する。好ましい実施形態では、この技法には、基準線となる実行時統計は動的に変更可能な作業負荷の第１の複数の要求を実行することに関係し、当該要求は複数のスレッド・プールによって処理されるものである、マルチスレッド・サーバ上で動的に変更可能な作業負荷に関して基準線となる実行時統計を集めること、スレッド・プールをプログラムに基づいて変更すること、新しい実行時統計は動的に変更可能な作業負荷の第２の複数の要求を実行することに関係し、当該要求はプログラムに基づいて変更されたスレッド・プールによって処理されるものである、動的に変更可能な作業負荷に関して新しい実行時統計を集めること、および、新しい実行時統計と基準線となる実行時統計とを比較して、プログラムに基づいた変更の結果、性能が低下したことを示す場合、プログラムに基づいた変更をプログラムに基づいて逆転させること、が含まれる。（さらに、その結果、性能が改善されない場合は、プログラムに基づいて変更を逆転させることもできる。）
【００１９】
好ましくは、スレッド・プールは物理スレッド・プールの論理的に編成されたグループ化である。
【００２０】
プログラムに基づいた変更は、動的に変更可能な作業負荷を処理するために、追加のスレッド・プールを追加することを含むこともできる。この場合、調整は、新しい実行時統計を集める前に、追加されたスレッド・プールを反映するために、複数のスレッド・プールへの動的に変更可能な作業負荷の割振りをプログラムに基づいて再バランシングすることを、さらに含むことができる。このプログラムに基づいた再バランシングは、スレッド・プールによって処理される要求の実行時間の上限を計算することを、さらに含むことができる。プログラムに基づいた逆転は、追加されたスレッド・プールをプログラムに基づいて除去すること、および、除去されたスレッド・プールを反映するために、複数のスレッド・プールへの動的に変更可能な作業負荷の割振りを再バランシングすることを、さらに含むことができる。
【００２１】
好ましくは、基準線となる実行時統計および新しい実行時統計は、変更可能な作業負荷を含む要求を処理するための平均実行時間および平均待ち行列時間であり（および、これらの数字に関する標準偏差を含むこともできる）、さらに、要求タイプごと、要求タイプおよびパラメータ値ごと、要求タイプ、パラメータ名、およびパラメータ値ごと、メソッド名ごと、メソッド名およびパラメータ値ごと、メソッド名とパラメータ名および値ごと、Ｕｎｉｆｏｒｍ　Ｒｅｓｏｕｒｃｅ　Ｌｏｃａｔｏｒ（「ＵＲＬ」）のＵｎｉｆｏｒｍ

Ｒｅｓｏｕｒｃｅ　Ｉｄｅｎｔｉｆｉｅｒ（「ＵＲＩ」）名ごと、ＵＲＩ名およびパラメータ値ごと、ならびに／または、処理宛先ごとを含む（ただし、これらに限定されることのない）、様々な方法で維持することができる。

【００２２】

プログラムに基づいた変更は、動的に変更可能な作業負荷を処理する複数のスレッド・プールからスレッド・プールを除去することを、さらに含むことができる。この場合、調整は、新しい実行時統計を集める前に、除去されたスレッド・プールを反映するために、複数のスレッド・プールへの動的に変更可能な作業負荷の割振りをプログラムに基づいて再バランシングすることを、さらに含むことができる。プログラムに基づいた逆転は、他のスレッド・プールをプログラムに基づいて追加すること、および追加されたスレッド・プールを反映するために、複数のスレッド・プールへの動的に変更可能な作業負荷の割振りを再バランシングすることを、さらに含むことができる。

【００２３】

プログラムに基づいた変更は、新しい実行時統計を集める前に、スレッド・プールのうちの選択された１つに割り当てられたスレッド数を増分することをさらに含むことが可能であり、プログラムに基づいた逆転は、スレッド・プールのうちの選択された１つに割り当てられたスレッド数を減分することをさらに含むことが可能である。プログラムに基づいた変更は、新しい実行時統計を集める前に、スレッド・プールのうちの選択された１つに割り当てられたスレッド数を減分することをさらに含むことが可能であり、プログラムに基づいた逆転は、スレッド・プールのうちの選択された１つに割り当てられたスレッド数を増分することをさらに含むことが可能である。

【００２５】

次に、同じ参照番号が全体を通じて同じ要素を示す下記の図面を参照しながら、本発明について説明する。

【発明を実施するための最良の形態】

【００２６】

本発明は、クライアント／サーバ・ネットワーキング環境のマルチスレッド・サーバにおける実行リソースのセット全体にわたり、インバウンド要求を動的かつプログラムに基づいて分配する。好ましい実施形態では、実行リソースはスレッドであり、これらのスレッドは論理的には複数のスレッド・プールに編成される。（使用可能なスレッドのセットは、単一の物理プールから生じるものとみなすことが可能であり、これがさらに論理的に複数のプールに分けられる。以下の、図１の要素１３５、１４０、１４５、および１５０に関する考察も参照されたい。参照しやすくするために、本明細書では、論理的に編成されたプールを単にスレッド・プールと呼び、単一の物理プールを「グローバル」スレッド・プールと呼ぶ。）

【００２７】

好ましくは、プログラムに基づいた要求分配プロセスは、どのインバウンド要求をどのプールに割り当てるべきか（要求は、必要であればそのプールに関する待ち行列に入る）をプログラムに基づいて決定することを含み、オプションの拡張機能では、使用中のスレッド・プールの数および／またはサイズもプログラムに基づいて調整することができる。プログラムに基づいた分配プロセスの好ましい実施形態では、本発明は、要求が実行されるときに要求を追跡し、要求タイプごとの平均実行時間および待機時間を決定し、ならびに、（好ましくは、各プールで処理される要求の平均実行時間の上限を決定することにより）特定プールに対する要求の割振りを動的に調整する。スレッド・プールの数および／またはプール内のスレッドの数も動的に調整する好ましい実施形態では、一度にこれら変数（プールごとの平均実行時間の上限、スレッド・プール数、およびプール内のスレッド数）のうちの１つだけが調整され、その効果が肯定的であったか否定的であったかを判定するためにスナップショットが取られる。

【００２８】

従来技術に従って単一の制約されたスレッド・プールを使用することに関連付けられた

問題について、上記で説明した。当分野で知られるこれらの問題に対する１つの解決策は、少数のスレッド・プール（この数は静的に事前に定義される）をセットアップすること、および各プールに入ることのできる要求のタイプを手動で構成することである。その後、他のすべての要求は、「他のすべての」プールで処理することができる。この方法には、どのタイプの要求がこの機構から恩恵を受けることになるかを判別するためにシステムをプロファイリングすること、およびプールへの要求タイプのマッピングを記述したサイドテーブル情報を手動で構築することが必要である。

【００２９】

この方法には長所と欠点がある。長所の１つは、要求セットが、どの要求の待ち行列時間が異常に長くなっているか、およびどの要求の実行時間が他のすべての要求タイプに比べて十分短いかを識別できる場合、識別されたセットはおそらく、識別されたときに、この情報を適切に使用して、これらの要求を特定のスレッド・プールの方向へ向ける（すなわちこれに向けて送る）ことのできる実行時間に対して、より良好な応答時間を達成することになる、ということである。

【００３０】

ただし、この方法では長所よりも欠点の方が多い。その１つが、この方法は要求ストリームの（Ｗｅｂアプリケーションで遭遇するような）変化する性質を考慮しないことである。要求タイプのマッピングを含むハードコードされたテーブルが最初から悪いか、あるいは、当初測定されたコードの特徴が変化したかまたは新しいコードがシステムの力学を変化させたかのいずれかにより、経時的に変える必要があるかのいずれかである可能性が高い。いずれの場合も、たとえ最初から悪いわけではなくとも、情報は即時に時代遅れとなる可能性が高いのは明らかであろう。

【００３１】

本発明の適切な解決策、および好ましい実施形態によって使用される解決策は、所与のタイプの要求に関する平均実行時間および待機時間を追跡し、その後、それらの実行時間に従って、各タイプの要求をスレッド・プールに割り当てることである。

【００３２】

アプリケーション・サーバによって処理される様々な要求タイプに関する平均実行時間を追跡することによって、これらの要求タイプをいくつかの同じ実行時間カテゴリのグループに分けることができる。１つの方法では、実行時間帯域の周波数分布を構築することによって、カテゴリを決定することができる。好ましい実施形態で使用される単純な実施は、使用可能なスレッド・プールの（所定の）数を入手し、集められた実行時間を使用して要求タイプをこの数のスレッド・プールに分けることによって、帯域を計算することができる。これは、統計タイルの計算とも呼ばれ、タイル値（すなわち各特定帯域の上限）は各プールに可能な最大実行時間となる。タイル値を計算し、その後、どの実行時間が各タイルに対応するかを識別するマッピング・テーブル（または同様の関連）を作成するプロセスを、本明細書では「分配計算」または「プール・ターゲット計算」と呼ぶ。この情報を使用して、新しく到着したインバウンド要求がどこに向かって送られるべきかを決定するプロセスを、本明細書では要求の「分類」と呼ぶ。実行時間の上限を決定する際に使用される集められたデータを、本明細書では一般に「分類データ」と呼び、各要求タイプの平均実行時間ならびに以下でより詳細に説明する他の情報が含まれる。

【００３３】

たとえば、分類データの実行時間が１０，０００時間であり、１０のスレッド・プールが使用される場合、１，０００時間の最短実行時間をもつ要求タイプは、第１のプールからのスレッドによって処理され、１，０００時間の最長実行時間をもつ要求タイプは、最後（１０番目）のプールによって処理されることができる。本発明の発明者は、分配計算は、各実行時間帯域内にある要求の履歴周波数を自動的に考慮に入れるため、この方法が既知のプール・サイズに対して良好に働くと判断した。分類、またはスレッド・プールへの要求の実行時方向付け（ｖｅｃｔｏｒｉｎｇ）を、好ましい実施形態に従ってどのように実行することができるかについての詳細は、図１（以下で論じる）を参照されたい。

【００３４】
実験を通じて、複数の論理スレッド・プールの使用可能性が全体の応答時間およびスループットを助けることがわかった。前述のように、従来技術の方法は、要求を複数のプールに静的に割り振る。静的割当ての欠点についてはこれまでに述べたが、本発明の動的分類および分配計算技法はこれらの欠点を回避する。本明細書で開示された技法は、所与の作業負荷に対してプール・サイズおよび／またはプール数を動的に調整できるようにするものでもある。（この調整がどのように実施できるかについての詳細な考察を、図５～８を参照しながら以下で行う。）

【００３５】
スレッド・プール数を動的に調整するための技法が使用されない場合、好ましい実施形態のオペレーション中に使用するプールの数は、好ましくは所定の数（構成可能な値であってよい）に割り振られる。プールの数がシステムの挙動を調べることによって動的に決定される場合、本明細書に記載されるように、好ましくは反復方法が使用される。後者の場合、最初に使用されるプールの数（ならびにプール・サイズ）は、以前の知識（たとえば、動的に決定された可能性のある、最も新しく使用された値に関する保存された状態情報）によって、または初期のデフォルト構成から始めることによって、決定できる。

【００３６】
プールへの要求タイプの分配を動的に再計算することに加えて、プールの数およびプールのサイズを動的に調整する場合、このタイプの実行時調整には、３次元的な問題として取り組むことができる。１つの次元は、要求タイプの関連付けられた実行時間についてタイルを計算すること、もう１つの次元はプールの数を調整すること、およびもう１つの次元はプールのサイズを調整することである。このプロセスは、これらの調整面の間の相互作用により、さらに複雑になる。（プログラムに基づいて要求を分配するためにタイル値を計算することは、それ自体が調整の１タイプであるため、本明細書で使用される「調整」という用語は、参照のコンテキストがそれ以外のものを示す場合を除き、プール数およびプール・サイズの調整プロセスのみを指すことを意図するものではない。）たとえばプール・サイズの変更またはプール数の変更は、プールに向けて送られる要求の実行時間に影響を与えることが多い。次にこれが、様々な要求タイプを異なる帯域に押し込むことがある。これについて例示するために、４つの帯域を有する構成があり、第３の帯域（次に最も長い実行要求が処理される）がいくつかのスレッド「Ｔ」を有すると想定する。さらに、この数のスレッドを使用して、帯域３内の要求が、なんらかの低帯域「ＬＢ（３）」からなんらかの高帯域「ＵＢ（３）」までの範囲の時間間隔で、それらの実行を完了すると想定する。帯域３のスレッド数Ｔが変更されると、帯域３に向けて送られるいくつかの要求タイプの実行時間が範囲［ＬＢ（３）．．．ＵＢ（３）］から外れる可能性がある。これは、本明細書で使用される方法によれば、これらの要求タイプがもはや帯域３内に属さないことを意味する。ただし、これらを他の帯域に移動させると、それによってその帯域内の要求の実行時間が変更される、という連鎖反応を起こす可能性がある（追加の要求タイプが他の帯域に移動する必要がある可能性があることを示す）。この反応を制御するために、好ましい実施形態は、プール・サイズ調整プロセス中に、実行時間（および、したがってその時間内に実行される傾向がある要求タイプ）を特定の実行帯域に拘束し、後続の分配計算（すなわち、後続の実行データ分析およびそこから導出されたマッピングの改訂）が発生するときにのみそれらの拘束を解く。分配計算は、一般に、プール・サイズ調整と同時に発生すること（および、好ましくはその間には実行時間が拘束されていない、プール・カウント調整とも同時に発生すること）に留意されたい。好ましい実施形態では、この拘束は要求タイプの分類データ内のフラグを使用することで実行され、拘束フラグはプール・サイズ調整が完了した後に消去される。

【００３７】
マルチスレッド・サーバ環境では、以下のようないくつかの動的要素が観察できる。
ＤＥ１：所与の要求タイプの実行時間は、それが使用し、コード・パスが取得されるリソースに依存して変更可能である。

ＤＥ２：新しい要求がシステムに入り、それらの実行時間に従って分類されなければならない。
ＤＥ３：プール・サイズは変更可能であり、プールに向けられた要求の実行時間に影響を与えることになる。
ＤＥ４：実行時間の分配は再計算可能であり、その結果、様々な要求タイプが帯域を変更する可能性がある。
ＤＥ５：帯域の数および対応するプールの数は変更可能である。
【００３８】
上記に列挙した動的要素を参照し、ＤＥ１およびＤＥ２は調整プロセスから独立していることに留意されたい。実際、それらが動的調整機能を必要とする主な理由である。そうでなければ、要求タイプの帯域への分配は１回計算してそのままにしておくことができる。（同様に、ＤＥ１およびＤＥ２がなければ、プール・サイズまたはプール数を動的に調整することによって得られる利点はわずかであるか、またはまったくない可能性がある。）また、動的要素ＤＥ３、ＤＥ４、およびＤＥ５は、調整プロセスに直接起因することにも留意されたい。これらの観察は、調整プロセスを首尾よく推進するために、本明細書に開示された調整技法によって使用される。
【００３９】
したがって、本明細書に開示された動的分配および調整技法は、動的要素のセット全体にわたってバランスがとれるように設計される。好ましくは、発生するオーバヘッドをできる限り少なくするために、調整の経路長さ、競合、および周波数は最低限にされる。調整時の相互作用および連鎖反応を避けるために、好ましい実施形態では、１つを変更し、システムが何らかの時間間隔についてこの状態で実行できるようにし、変更の効果を測定する。（この方法は、変更の肯定的または否定的な影響を分離するためには有益であるが、調整プロセス全体の持続時間を長引かせる可能性がある。したがって、絶対に必要でない限り、帯域の最大数を、はじめに比較的少ない数に設定することが有益である。好ましくは、システムが大きくなるほど、この最大数が大きくなる。）
【００４０】
本発明の調整方法には、以下のように３つの主要な目標がある。
Ｇ１：定常作業負荷ミックスの準定常状態に、できる限り迅速に達するように試行する。
Ｇ２：過度の制御によって生じる可能性のある、実行時間の激しい変動を避ける。
Ｇ３：入ってくる要求の使用可能なスレッド・リソースと実行時間とのバランスがとれるようにする。
【００４１】
これらの目標を達成するために、システムによって処理された要求に関して実行時間および待機または待ち行列時間の履歴統計を保管する必要がある。したがって本発明は、上記で述べたようにこの情報を追跡する。調整中に行った決定は、これらの要素に関して判明した現在の値と過去に判明した値との比較に基づいて、その後評価することができる。（好ましくは、実行した変更の値の制限は、新しいタイプの要求、およびレート０を含む現在の要求のレート変更に関して、システムの動的性質も考慮に入れる。新しく遭遇した要求タイプは、図２～３を参照しながら以下で説明するように、本発明の分類プロセスの実施形態によって自動的に処理される。特定の要求タイプに関する着信レートの変更には分配計算が自動的に対処し、実行時間対プール・マッピングの上限が修正される場合がある。着信レートの変更は、結果的に、それらの要求タイプを処理するスレッド・プール・サイズをプログラムに基づいて調整することにもなる。）
【００４２】
動的な作業負荷の分配およびプールの調整を考えるときに、それ自体を直接表す難題の１つが、ほとんどの実際のアプリケーション・サーバが閉じたシステムでないという事実である。すなわち、要求によって提示される実行時間および待ち行列時間は、実行される作業のタイプに応じて、ダウンストリームおよびアップストリームの力に影響されることが多い。たとえば、特定の要求タイプが（データベース呼出しなどの）リモート呼出しを

実行し、その実行時間がそれら呼び出されたリソースの使用可能性、またはそれらの競合に応じて、変動する場合がある。

【００４３】

この影響に対処するために、好ましくは本発明の実施形態は、複雑なフィードバック・システムを構築しようと試みる代わりに、フィルタリング機構を適用し、このフィルタリング機構には、プールごとの実行時間上限、プール数、およびプールごとのスレッド数を変更することなしに、実行時間および待機時間の２つまたはそれ以上のスナップショットを取ることが含まれる。これらのスナップショットから集められたデータは、調整とは無関係に変動する要求、すなわち通常の変動をする要求を検出する試みにおいて比較される。要求が変動しない場合、このプロセスで「フィルタリング」することができる。（すなわち、要求がその標準偏差内にある場合は、この要求に関して性能を向上させるためにシステムを調整しようとする試みは有用でない可能性がある。）好ましくは、フィルタリング機構は統計的方法を適用し、要求タイプが変動したかどうかを判別する基準として、要求タイプごとの標準偏差を使用する。その後、調整変更なしで標準偏差が比較的大きい実行時間パターンを有するいずれの要求タイプも、通常変動要求タイプとして処理することができる。たとえば、１つまたは複数の要求タイプに関する実行時間データは、調整変更が実施されなかったサンプリング間隔に続いて分析することができる。これらの各要求タイプに関する標準偏差は、この「変更なし」間隔に関して計算することができる。その間隔中に調整変更がアクティブであった他の間隔中に集められたデータは、調整変更が存在する要求タイプに関して標準偏差を決定する場合と同じ方法で分析することができる。変化なし間隔中の特定要求タイプに関する標準偏差と、調整変更間隔中のその標準偏差とを比較することによって、この要求タイプの実行時間に与える調整変更の影響に関して予測を立てることができる。（このプロセスでは、通常の変動ではないと判定された要求タイプに焦点を当てることが有用な場合がある。）

【００４４】

変動は内部競合ならびに外部待機時間（幅広く変動する可能性がある）によって生じる場合があることに留意されたい。多くの場合、これらの問題を提示する要求に関してスレッド・プール・サイズに制約を与えることで、全体のスループットを助けることができる。したがって、標準偏差は、要求タイプの実行時間が変動しているかどうかの指示として使用するだけでなく、プール・サイズまたはプール数の変更の有効性に関する指示としても使用することができる。

【００４５】

これらすべての要素を考慮することにより、実行時間および待機／待ち行列時間の追跡に基づいて、複数のスレッド・プールに作業を効果的に分配するための、自己調整型および高スループットの機構を構築することができる。本明細書に開示された技法は、作業負荷全体にわたって動的にバランスをとることができるようにし、さらに、作業負荷が経時的に特徴を変更しながら、依然として最適なスループットおよび応答時間を達成し続けることもできるようにするものである。

【００４６】

次に図１を参照すると、本明細書に開示されたように動作するシステム１００の抽象図が示されている。作業要求１０５（たとえばインバウンド・クライアント要求）がシステムに入ると、好ましい実施形態に従って、こうした要求それぞれについて待機待ち行列要素（「ＷＱＥ」）１１０が作成される。ＷＱＥは、要求が処理されるときにシステムを「通過するフロー」としてみなすことが可能であり、要求およびその現在の処理に関係する情報を集めるために使用される。オブジェクト指向のプログラミング条件では、ＷＱＥは好ましくはインバウンド要求のための「ラッパー（ｗｒａｐｐｅｒ）」として実施される。各ラッパーは、要求のタイプなどの、それに関連付けられた要求を識別するための情報を含む。この識別情報は、インバウンド要求を特定のスレッド・プールに向けるためにそれらを分類する際に使用される、あらかじめ格納された履歴統計の位置を突き止めるために使用することができることから、「分類キー」とも呼ばれる場合がある。この分類キー

に加えて、ＷＱＥは、要求の現在の実行時間および現在の待ち行列時間も格納する。好ましいことに、ＷＱＥは、この要求のタイプに関する分類データへの参照も保持しており、その結果、ＷＱＥに格納された分類キーを使用して分類データを取り出すことができる。好ましい実施形態では、この分類データは、好ましくは移動平均として格納される実行時間、好ましくは移動平均として格納される待ち行列時間、およびオプションで、要求タイプの履歴実行時間および待ち行列時間の値に関する標準偏差値を含む。好ましくは、これらの標準偏差値も移動値である。ＷＱＥと共に分類データを格納することによって、（要素１５５および１６０を参照してより詳細に説明するように）要求の現在の実行時間および待機時間が履歴統計に繰り込まれると、統計計算スレッドのより効率的なオペレーションが可能になる。移動平均および移動標準偏差値、すなわちそれぞれの新しい実行で更新される値を使用することによって、本発明の実施形態は、実行時間および／または待ち行列時間における過去の異常の影響を弱める。

【００４７】

インバウンド要求（そのＷＱＥと共に）は、分類オペレーション１１５に入力される。この分類には、この要求をどのスレッド・プールに割り当てるべきかを決定することが含まれる。好ましい実施形態では、この要求タイプに関するあらかじめ計算された分類データ（すなわち履歴統計）を使用して、このタイプ（あるいは、ワイルドカードまたは同様の突合せ方法がサポートされている場合は同様のタイプ）を有する要求の平均実行時間が決定される。平均実行時間を使用して、この要求に似た要求、またはこの要求と挙動が同様の要求を処理するプールを識別することができる。したがって、実行時間の長い要求が、普通であれば実行時間の短い要求をブロックするという、従来技術の問題が回避される。

【００４８】

代替の実施形態では、要求タイプを分類キーとして使用するのではなく、追加または異なる情報をインデックスとして使用して、分類オペレーション１１５中に適切な履歴統計の位置を突き止めることができる。たとえば、要求タイプは、それらのパラメータの入力値（およびオプションでパラメータ名）を使用することによってさらに制限することが可能であり、この要求タイプとパラメータとの組合せを、（好ましくは、このより粗いレベルでも記録される）分類データのインデックス付けに使用することができる。ＵＲＬのＵＲＩ部分が使用可能であり、ＵＲＩと共にパラメータ名／値も使用可能である。また、Ｗｅｂ環境では「要求タイプ」が要求の有用なカテゴリ分類であるが、他の環境では他の情報が適切な可能性のあることにも留意されたい。たとえば、エンタプライズＪａｖａＢｅａｎｓ（登録商標）環境では、要求名の代わりにメソッド名（それらのクラスまたは配置名を含む）を使用することができる。メソッド名と共に、所望であれば、他の修飾子としてパラメータ名／値を使用することができる。（「エンタプライズＪａｖａＢｅａｎｓ」は、Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ，Ｉｎｃの登録商標である。）したがって本明細書では、限定ではなく例示のために、「要求タイプ」に言及している。

【００４９】

次に図２を参照し、分類オペレーション１１５についてより詳細に説明する。ブロック２００では、新しいインバウンド要求が入力待ち行列から受け取られる。ブロック２０５はこの要求を分析して、その分類キー（またはより一般的には、その識別情報）を決定する。本発明の特定の実施が、インバウンド要求の中で分類キーの位置を突き止める方法は、要求タイプ、または要求タイプにパラメータ値などを加えたものが、その特定の環境において要求を分類する際に関心の対象であるかどうかに依存する。分類キーが決定されると、これがマッピング・テーブルまたは履歴統計が記録された他のリポジトリのインデックスとして使用される（ブロック２１０）。

【００５０】

ブロック２１５は、この分類キーについて、以前に記録された分類データ（および具体的には、平均実行時間）の位置が突き止められたかどうかをチェックする。突き止められなかった場合、このインバウンド要求は「新しく着信した」要求タイプであるとみなされ

る。（要求タイプはこのシステムによって以前に処理されたが、最後の処理に関する統計データがすでに古くなってしまった可能性があることは、明らかであろう。好ましくは、実施特有の「妥当な」レベルで、履歴統計データに消費された記憶域量を保持するために、「ＬＲＵ（Ｌｅａｓｔ－Ｒｅｃｅｎｔｌｙ－Ｕｓｅｄ）」方法が使用される。）ブロック２２０および２２５は、新しく着信した要求タイプに対して追加の処理を実行する。ブロック２２０の処理には、この新しい要求タイプの統計を格納するために、分類データ内に新しいエントリを作成することが含まれ、この新しいエントリには現在の要求の分類キーでインデックス付けされる。次にブロック２２５は、好ましくは、平均実行時間をマイナス１（「－１」）などの特殊値に設定することによって、この新しいエントリを初期設定する。この特殊値は、図３を参照しながらさらに論じるように、プール割当てプロセスで検出される。（あるいは、値が単にゼロに設定される場合がある。）

【００５１】

要求の統計データの位置を突き止めた後（すなわち、ブロック２１５が肯定的な結果を有する場合）、または新しい実行時統計エントリを作成および初期設定した後（ブロック２１５が否定的な結果を有する場合）、制御はブロック２３０に達し、現在のインバウンド要求をどのプールに割り当てるべきかを決定し、その割当てを実行する、プール割当てプロセスを呼び出す。このプロセスは、図３に詳細に記載されている。現在のインバウンド要求の処理を完了すると、制御をブロック２００に戻すことによって、図２の論理は後続の要求に対して繰り返し実行される。

【００５２】

図３は、好ましい実施形態が、図２のブロック２３０から呼び出されたプール割当てプロセスをどのように実施できるかについて、さらに詳細に示した図である。このプロセスは、図中で「ｐｏｏｌＮｄｘ」と呼ばれるプール・カウンタまたはインデックスを初期設定することによって始まる（ブロック３００）。このｐｏｏｌＮｄｘ値は、プール・セットの中をループして、現在のインバウンド要求が割り当てられるべきプールをチェックするために使用される。プールは、段々に実行時間がより長い作業へと受け入れていくため、最も実行時間の長い作業が最後のプールに割り当てられる。

【００５３】

ブロック３０５は、この現在のインバウンド要求の平均実行時間が、ｐｏｏｌＮｄｘの値でインデックス付けされたプールのターゲット上限より少ないかまたは等しいかどうかを調べるためにテストする。好ましいことに、要求の平均実行時間は、この要求のタイプに関連付けられた履歴統計から取得されるか、またはこの要求タイプの履歴統計が使用できない場合は、マイナス１に初期設定されている可能性がある（図２のブロック２２５）。後者の場合、ブロック３０５のテストは、図３の論理を介した第１の反復で真となり、好ましい実施形態は、新しく着信した要求タイプを、最も短い実行時間を有する要求を処理するプールに割り当てる。（新しく着信した要求タイプを処理するためのプールを選択する他の方法を、代替の実施形態で使用することができる。）

【００５４】

ブロック３０５のテストが肯定的な結果となった場合、この要求を処理するためのプールが見つかっている。したがって、制御はブロック３２０に移り、ここで作業要素はｐｏｏｌＮｄｘとインデックス付けされたプールに向けられる（すなわち、実行用に割り当てられる）。次に図３の処理は完了し、制御は図２の論理呼出しに戻る。そうでない場合、ブロック３０５のテストが否定的な結果になると、ブロック３１０でプール・インデックスは増分され、ブロック３１５は、ターゲット上限がチェック可能なプールがまだほかにもあるかどうかをチェックする。このチェック・プロセスは、ｐｏｏｌＮｄｘ内の現在のプール・インデックス値とプールの合計数よりも１少ない値とを比較する。（変数「＃ｐｏｏｌｓ」は、現在使用中のプール数を格納する。）この方法では、結果として常に、実行時間が最後のタイル値よりも長い要求タイプを、最後のプールに割り当てることになる。チェック可能なプールがまだある場合、制御はブロック３０５に戻り、ない場合、処理はブロック３２０に進んで、現在のインバウンド要求が現在の（この場合は最終の）プー

ルに向けられることになる。

【００５５】

図３がどのように動作するかの一例として、実施が３つのプールを使用しており、これらのプールのタイル値（すなわち実行時間の上限）が１０時間単位および２０時間単位に設定されるものと想定する。図３に示された方法を使用すると、１０またはそれより少ない時間単位を使用して実行するいずれの要求タイプも、新しく着信した要求タイプと共に第１のプールに向けられ、１０より多く２０を超えない時間単位を必要とする要求タイプは第２のプールに向けられる。２０より多くの時間単位を必要とする要求タイプは、第３のプールに向けられる。インバウンド要求のタイプに関する履歴統計が、このタイプが平均して５０の実行時間単位を必要とすることを示すものと想定する。ｐｏｏｌＮｄｘ値が０の場合、ブロック３０５で５０が１０と比較され、このテストは否定的な結果となるため、ブロック３１０はｐｏｏｌＮｄｘを１に増分する。ブロック３０５を通過する次の反復では、５０が２０と比較されることになる。このテストも否定的な結果となり、ブロック３１０はｐｏｏｌＮｄｘを２に増分する。次にブロック３１５のテストでは、２（ｐｏｏｌＮｄｘ値）を２（プール数よりも１少ない値）と比較することになり、テストは否定的な結果となるため、要求は第３のプール（すなわち、ゼロ・ベースのインデックス付けを使用してインデックス値２を有するプール）に向けられる。

【００５６】

図４は、好ましい実施形態が、異種の動的に変更可能な作業負荷の実行時間および待機時間の特徴を分析することによって、タイル値またはプールごとの実行時間の上限をどのように決定するかを示す論理を提供するものである。このプロセスは、本明細書では、分配計算またはプール・ターゲット計算プロセスと呼ばれる。好ましくは、図４の論理は、図１の要素１６０を参照しながら以下で説明するように、上限を改定するために定期的に呼び出される。

【００５７】

ブロック４００は、現在の分類の集まり（すなわち履歴統計の集まり）を、それらの平均実行時間によってソートする。前述のように、好ましくはこの平均実行時間値は、いくつかの最近の間隔全体にわたる移動平均を表す。この方法では、過去に発生した問題状態または他の異常（結果的に過度に長い実行時間を生じさせたタイムアウト状況、または異常に短い実行時間を発生させた例外条件など）が、将来の決定を曲げることはない。好ましい実施形態は、実行時間を配列内にコピーし、この配列をソートする。（このアレイの記憶域がいったん割り振られると、好ましいことに、アレイ・サイズを増やすために再割振りが必要とならない限り、図４の後続の反復についてもこれが維持される。）

【００５８】

ブロック４０５は、値「ｅｔａｓ」（「実行時間アレイ・サイズ（ｅｘｅｃｕｔｉｏｎ　ｔｉｍｅ　ａｒｒａｙ　ｓｉｚｅ）」を表す）を、このソートされた実行時間の集まりのサイズに設定する。次にこの「ｅａｔｓ」値は、プールのセット間で実行時間が適切に分配されるように、どれだけの実行時間があるかのカウンタとして機能する。ブロック４１０は、プール・インデックス値ｐｏｏｌＮｄｘをゼロに初期設定する。

【００５９】

ブロック４１５では、ターゲット上限を割り当てる必要のあるプールがまだほかにもあるかどうかを調べるためにテストが実行される。このチェック・プロセスは、ｐｏｏｌＮｄｘ内の現在のプール・インデックス値とプールの合計数よりも１少ない値とを比較する（この合計数は変数「＃ｐｏｏｌｓ」に格納されている）。この方法は、結果的に、実行時間が最後の上限よりも長いすべての要求タイプが最後のプールに向けられるように（図３を参照しながら論じたように）、そこにあるプールよりも１つ少ない上限を割り当てることになる。ブロック４１５のテストが肯定的な結果になった場合、処理はブロック４２０に進み、否定的な結果になった場合は、割り当てられる上限がないため、制御は論理呼出しに戻る。

【００６０】

ブロック４２０では、現在のプール（すなわち、ｐｏｏｌＮｄｘとインデックス付けされたプール）に向けられる実行時間の上限が計算されて割り当てられる。好ましい実施形態では、これには、使用可能なプール全体にわたって実行時間統計（および、それらの関連付けられた要求タイプ）の合計数を等しく分配することが含まれる。したがって、ソートされた実行時間から「Ｎ番目」の要素が突き止められ、その要素からの実行時間が、現在のプールの上限（図中では「ターゲット」属性と呼ばれる）として割り当てられる。Ｎ番目の要素は、ブロック４２０に示されるように、第１に、ソートされたアレイ内にある要素のカウント（値「ｅｔａｓ」で表される）を使用可能なプール数（＃ｐｏｏｌｓ内にある）で割り、次にこれにｐｏｏｌＮｄｘ値＋１を掛けて、最後にその値から１を引くことで決定される。

【００６１】

上限を設定した後、ブロック４２５はプール・インデックス値を増分し、割り当てられる上限がほかにもあるかどうかを判別するために、制御をブロック４１５に戻す。

【００６２】

図４の論理がどのように動作するかの一例として、現在の集まりに１２の分類があるものと想定する。（実際には、何百、何千という分類のある可能性があることは明らかであろう。）さらに、４つの使用可能なプールがあると想定する。第１の反復では、ブロック４２０での処理により、プール（０）の上限が計算される。この例で「ｅｔａｓ」の値は１２であるため、（ｅｔａｓ／＃ｐｏｏｌｓ）は１２／４、すなわち３である。この値に１を掛けると３となり、さらに１を引くと、プール（０）の上限は、ｓｏｒｔｅｄＥｘｅｃＴｉｍｅｓ［２］から取得した実行時間である。その後の反復では、プール（１）の上限がｓｏｒｔｅｄＥｘｅｃＴｉｍｅｓ［５］からの実行時間にセットされる、という具合になる。

【００６３】

図４に示された方法では、結果として上限を、最近観察された実行時間の分配に基づいた値に設定することになる。上限を割り当てなければならないプール数は、プール数が調整されるときに動的に変更可能であることに留意されたい。（プール数の調整方法に関する詳細な情報は、図５～７の考察を参照されたい。）図４に示された論理は、プール数の変化に自動的に適合する。

【００６４】

次に、図１に示された処理全体の考察に戻ると、各インバウンド要求が１１５で分類され、次にこれが適切なスレッド・プール（すなわち、図２および３の処理を使用して識別されたスレッド・プール）に向けて送られる。要求は待機待ち行列に入り、スレッドが使用可能になるまで待機しなければならない。したがって、図１では、要求は「Ｎ」個の待機待ち行列１２０、１２５、１３０のうちの１つに向けて送られるように示されており、ここでは各待機待ち行列が、現在システム１００で使用されているＮ個の論理スレッド・プール１３５、１４０、１４５のうちの１つに対応する。（システム１００内のプール数が、その後の何らかの地点で増えるかまたは減った場合、それに応じて待機待ち行列の数も調整しなければならないことは明らかであろう。また、待機待ち行列のサイズは待ち行列に入っている要素の数と共に変化し、本明細書で開示された動的な調整は、待ち行列サイズを修正するためのものではない。）

【００６５】

何らかの地点で、待ち行列に入った要求は、そのために待ち行列に入ったスレッド・プールからのスレッドに割り当てられることになる。好ましい実施形態によれば、要求が待機待ち行列内で費やした時間は、そのＷＱＥ内に記録される。（要求が分類オペレーションを完了したときにスレッドが使用可能な場合、要求は待機待ち行列を迂回することも起こり得る。その場合、待機待ち行列はゼロとして記録される。ただし、こうした要求は実際には待ち行列に提出される可能性があり、待ち行列に入った状態でかなり短い時間を費やすだけとなる。本明細書の以下の考察では、実施は、分類されたすべての要求を待ち行列に送るものと想定する。）

【００６６】
図１は、待機待ち行列から「ラン可能（ｒｕｎｎａｂｌｅ）プール」１３５、１４０、１４５へと横切る、インバウンド要求を示す図である。これらのラン可能プールは、本明細書に記載された論理スレッド・プールに対応し、さらに図１に示されるように、これら論理プール内のスレッドは実際にはグローバル・スレッド・プール１５０内で定義される。好ましい実施形態では、ラン可能プールはいくつかの限定数の実行可能ラッパー・オブジェクトを含み（本発明をオブジェクト指向言語で実施する場合）、各ラッパー・オブジェクトは、論理スレッド・プールに割り当てられたスレッドのうちの１つを表す。（実行可能ラッパー・オブジェクトは、スレッドにインターフェース機構も提供する。）したがって、スレッド数と同様に、実行可能ラッパー・オブジェクトの数も１つのプール１３５、１４０、１４５から他のプールまでいろいろ変化する可能性がある。（特定のラン可能プール内では、実行可能ラッパー・オブジェクトの数は、本明細書に開示されたプール・サイズ調整オペレーションの実行により、変化する可能性がある。）したがって、インバウンド要求およびそのＷＱＥは、その待機待ち行列に関連付けられたラン可能プール内の実行可能ラッパー・オブジェクトのうちの１つが使用可能になるまで、待機待ち行列内に残る。（実行可能ラッパー・オブジェクトの使用可能性は、定義上、スレッドが使用可能であることを暗に示す。）この方法では、実行可能ラッパー・オブジェクトは、プールごとのスレッド数に制限を課すための効率的かつ信頼できる方法を提供するが、依然としてスレッドが実際に定義された単一のグローバル・スレッド・プールを使用する。（単一のグローバル・スレッド・プールを使用すると、結果的に、別の物理スレッド・プールを維持するよりもかなりオーバヘッドが少なくなる。これは、プールあたりのスレッド数が変更される調整オペレーション中に特にあてはまり、好ましい実施形態は、別の物理スレッド・プールが使用された場合に実行されるように、スレッドを作成および破壊するのではなく、論理プール内で使用できる実行可能ラッパー・オブジェクトの数を変更することによって、単にある程度のスレッドを論理プールに割り振る。）
【００６７】
好ましくは、各スレッド・プール１３５、１４０、１４５のサイズは、そのプールに向けて送られるタイプの作業がどの程度同時に実行されるべきかに応じて変化する。オプションのプール・サイズ調整が実施される場合（図５、６、および８を参照しながら以下で説明するように）、プールのサイズは自己調整となる。たとえば、１つのプールがデータベース・アクセスを必要とする要求を処理している場合、および要求を最適に実行するためにデータベース・システムへの接続数が制約されていなければならない場合、そのプールのサイズは、その要求の性能を低下させることになるサイズを超えないように、それ自体で調整する傾向がある。
【００６８】
各要求は、ある程度の実行時間を費やし、その実行が完了すると、要求のスレッドがそのラン可能プールに戻され（すなわち、実行可能なラッパー・オブジェクトを戻すことまたは解放することによって）、要求のＷＱＥはその実行時間を記録するために更新される。従来技術を使用して、クライアントによって要求されたコンテンツが戻される（図１には図示せず）。ＷＱＥは、統計処理のために待ち行列に入り（統計待ち行列１５５を参照）、その後、最終的に統計計算スレッドは、要素１６０で示されるように、ＷＱＥを待ち行列から外してそのデータを処理する。
【００６９】
本発明では、１６０で実行される処理が、待ち行列に入れられたＷＱＥからの待機時間および実行時間を処理すること、最後に使用された（「ＬＲＵ」）トリミング・プロセスを実行すること、および／またはプール・ターゲット計算を実行することを含む。統計計算スレッドは、好ましくは背景プロセスとして実行するように実施され、好ましくはタイマ駆動方式で呼び出される。好ましい実施形態では、時間間隔は構成可能であり、システム性能を低下させないような最低値（２０秒など）を有する。
【００７０】

好ましい実施形態によれば、統計計算スレッドが実行中の場合、統計待ち行列上にエントリがあれば、それらのエントリは待ち行列から外されて処理される。他のオペレーションも、発生したタイムアウトに応じて実行可能である。好ましくは、統計計算スレッドを１回呼び出すごとにタイムアウトが１つだけ処理され、待ち行列１５５からの統計処理に基本設定が与えられる。（この待ち行列に入ってくる統計が分類を更新するため、分類が時宜を得た方法で実際の状態を表すように、即時に処理されなければならない。ＬＲＵおよびプール・ターゲット計算は集合体情報に依存するため、たびたび実行する必要はない。）

【００７１】

統計待ち行列を処理する場合、統計計算スレッドはエントリを待ち行列から外し、現在の実行時間および待機時間情報を含むように履歴統計を改定する。前述のように、好ましくは履歴統計への参照は、値が効率的な方法で容易に使用できるように、待ち行列に入ったＷＱＥ内で保持される。後者の場合、統計計算スレッドは、第１に、待ち行列から外されたエントリから識別情報（たとえば要求タイプ、および代替実施形態ではパラメータ値などの追加または異なる情報）を取得し、識別情報を、事前に計算された履歴統計にアクセスするための分類キーとして使用する。次に、履歴統計で維持されている待機時間および実行データは、待ち行列から外されたエントリからの情報を反映するように改訂され、標準偏差情報は、この情報が特定の実施で使用される場合には改訂することもできる。好ましくは、統計計算スレッドはイベント駆動型であり、入ってきた統計が受け取られると起動される。好ましくは、ウェイクアップ・イベントもタイムアウト時に起動される。（好ましくは、ＬＲＵおよびプール・ターゲット計算は、それらの関連付けられたタイマが満了すると無条件で実行される。）

【００７２】

ＬＲＵトリミング・プロセスおよびプール・ターゲット計算の処理を起動するために、異なるタイマ間隔を使用することができる。ＬＲＵトリミング・プロセスが起動されると、好ましくは最近使用されていなかった分類データが廃棄され、好ましくはそのデータ用に使用された記憶リソースが解放される。（たとえば、エンド・ユーザは異なるＷｅｂページに移動し、事前に受け取られたある要求タイプをサーバの現在の作業負荷とは無関係にすることができる。この場合、それらの要求タイプの統計を考慮することは、もはや有用ではない。さらに、統計はエンド・ユーザの集まりに関する集合情報を表し、これらユーザの一部は自分のセッションを終了している可能性がある。この場合、集められたデータの一部は、もはやシステムの現在の動作状態に関係していない可能性がある。）プール・ターゲット計算が起動されると、図４の処理（上記で説明）が呼び出される。これまでに説明した異なるタイプの処理に別々のスレッドを使用するのではなく、１６０で複数の目的に単一のスレッドを使用することで、オーバヘッドが最小限となり、分類データなどの共用リソースに関する競合も減少する。

【００７３】

情報が待ち行列から外されたＷＱＥから抽出され、１６０で処理された後、ＷＱＥはその後使用するために解放リスト１６５に戻すことができる。（または、ＷＱＥに使用した記憶域を解放することができる。好ましい実施形態は、オーバヘッドを減らす試みで、ＷＱＥを再使用する。）図１の処理は、インバウンド要求ごとにこの方法で繰り返される。

【００７４】

図５は、本明細書で開示されたプール調整プロセスの実施形態で使用することのできる、状態の移行を示す状態図である。これらの移行は、図６～８に示された論理と組み合わせて、プールの数および各プールのサイズを調整する場合の変更を分離するために使用することができる。

【００７５】

図５に示されるように、初期状態「Ｓ０」では、プール数またはプール・サイズには何の変更も行われていない。次に、プール数は次の状態「Ｓ１」で調整される。最後に、プール・サイズは状態「Ｓ２」で個々に調整することができる。好ましくは、１つの状態か

ら別の状態への移行はタイマ駆動であり、その結果、システムはある程度の期間特定の状態のままであり、実行に与える状態の影響を記録および分析することができる。好ましい実施形態では、図６の論理（ブロック６２０が、次の状態移行が可能になる前に「スリープ」オペレーションを実施する）で示されるように、タイマは調整プロセスに組み込まれる。代替実施形態は、移行が、イベント駆動方式を使用するなどの他の方法で起動されるように設計することができる。（たとえばこの代替方法では、図８がすべてのプールについてプール・サイズの調整を完了すると、状態Ｓ２から状態Ｓ０への移行を起動することができる。）

【００７６】

代替実施形態では、状態Ｓ１およびＳ２の順序を反対にすることができる。他の実施形態では、プール数調整プロセスおよびプール・サイズ調整プロセスを呼び出すための技法が、状態移行図によって起動される必要はない。

【００７７】

図６～８は、プール・サイズおよび／またはプール数を所与の作業負荷に調整するために使用することのできる論理の流れ図である。図６は、次の調整状態を（図５に示された状態図を参照しながら）取得することによって始まる（ブロック６００）。次の状態が「変化なし」の場合（ブロック６１５）、制御はブロック６２０に移り、そうでない場合、処理はブロック６２５へ進む。

【００７８】

処理がブロック６２０に達すると、好ましくは「ＴＵＮＩＮＧ＿ＳＥＴＴＬＥ＿ＴＩＭＥ」として図中に示されている構成済みインターバル、すなわち、システムを定常状態に落ち着かせることができるだけの十分な時間に対して、スリープまたは遅延が実行される。次にブロック６０５は、現在の統計を取得し、それらを使用して基準線を設定した後、ブロック６００に戻ることによって次の調整オペレーションが実行される。

【００７９】

制御がブロック６２５に移ると、次の調整状態がプール数を調整するためのものであるかどうかを調べるテストが行われる。そうであれば、図７でより詳細に示されるように、ブロック６３０でプール数調整プロセスが実行される。そのプロセスが完了すると、次に制御はブロック６２０に移る。

【００８０】

ブロック６２５のテストが否定的な結果になると、次にブロック６３５は、次の調整状態がプール・サイズを調整するためのものであるかどうかをチェックする。そうであれば、図８でより詳細に示されるように、ブロック６４０で、プール・サイズ調整プロセスが実行される。そのプロセスが完了すると、次に制御はブロック６２０に移る。

【００８１】

ブロック６３５のテストが否定的な結果になると、これはエラーである。このエラーは、ブロック６１０に示されるように、図６の調整プロセスをＴＵＮＩＮＧ＿ＳＥＴＴＬＥ＿ＴＩＭＥインターバルの間スリープ状態にできるようにすることで対処することが可能であり、その後制御はブロック６００に戻る。あるいは、調整プロセスが停止される場合がある（および、好ましくはこの場合、システム管理者にエラー・メッセージが表示される）。

【００８２】

次に図６の論理が繰り返され、プール調整プロセスは反復的に実行される。

【００８３】

図７の論理は、プール数が調整されるときに図６のブロック６３０から呼び出される。ブロック７００は、明白な変更が必要かどうかをチェックする。たとえば、分類データ内にある別個のエントリよりも多くのプールがあるものと想定する。この場合、様々なタイルの上限を２倍にすることができるか、または単に、マッピング要求タイプよりも多くのプールがあるようにすることができる。これは、首尾よく調整され適切に実行しているシステムでは発生するべきではないが、このタイプのエラーが発生する場合がある。一般的

な意味では、ブロック７００は、このタイプの明らかな問題の「すべての状況に対応できるもの」とみなすことができる。したがって、このテストが肯定的な値になる場合は、制御は、プール数調整の計算を実行する代わりに直接ブロック７０５に移り、変更が実行される。次に制御は図６の論理呼出しに戻る（システムは、この変更に続いて定常状態に落ち着くための時間を有することになる）。

【００８４】

明白な変更が必要でない場合、ブロック７１０で、システムの現在の作業負荷によって実行されている要求タイプの現在の統計が獲得される。好ましくは、これには、実行時間および待機時間の情報（および、オプションで標準偏差情報）が更新された、分類データの現在のスナップショットを入手することが含まれる。追加のプールが追加される（ブロック７１５）。

【００８５】

次にブロック７２０は、「ＳＥＴＴＬＥ＿ＴＩＭＥ」と呼ばれる時間間隔中、スリープまたは待機プロセスを実施し、このプール数の変更がインバウンド要求の実行時間および待機時間（ならびに、オプションで標準偏差）にどのように影響を与えたかに関する情報を反映するように、分類データを更新できるようにする。好ましくは、このＳＥＴＴＬＥ＿ＴＩＭＥ値は構成可能であり、図６で使用されるＴＵＮＩＮＧ＿ＳＥＴＴＬＥ＿ＴＩＭＥ値と同一かまたは異なる場合がある。好ましくは、ＳＥＴＴＬＥ＿ＴＩＭＥ値は、図４の分配計算プロセスがスリープ状態である時間間隔よりも長く、その結果、分配計算はＳＥＴＴＬＥ＿ＴＩＭＥスリープ中に自動的に起動されることになり、それによって、ブロック７１５で追加されたプールを反映するように、それ全体にわたって実行時間が分配されるプールの数が増える。あるいは分配計算を、ブロック７２０のスリープの前に明示的に呼び出すことができる。分配計算が動作すると、プールの上限値は改訂される。したがって、たとえばシステムが以前は４つのプールを使用していたが、現在５つのプールを使用しようとしている場合、分配計算は、実行時間を４つではなく５つのグループに分けることになる。

【００８６】

待機が完了すると、ブロック７２５は分類データのスナップショットを獲得し、ブロック７３０は、ブロック７２５からのスナップショットの統計がブロック７１０からのものよりも良いかどうか、すなわち、変更して良かったかどうかを調べるためのテストを行う。良かった場合、次にこの追加されたプールが保持され、制御は論理呼出し（ブロック７３５）に戻る。そうでなければ、追加されたプールがインバウンド要求の実行時間および待機時間を改善しなかった場合に、プールは除去される（ブロック７７５）。さらに好ましくは、各プールの上限がこの少ない方のプール数に基づくように、実行時間の分配が再度計算される。（ブロック７１５および７２０を参照しながら上記で述べたように、別法として、スリープ・オペレーションを実行する前に分配計算を明示的に呼び出してもよく、あるいは、スリープ・インターバルが分配計算インターバルよりも長い場合、新しい分配計算がスリープ中に自動的に発生することになる。他の別法として、ブロック７１５でプール数を変更する前に上限を保存してもよく、ブロック７７５に達したときに、これらの保存した値を単に復元することができる。）

【００８７】

ブロック７７０で開始される処理は、プール数を（増やすのではなく）減らすことが、実行時間および待機時間を改善するのかどうかを調べるように設計されている。ブロック７７０は、ＳＥＴＴＬＥ＿ＴＩＭＥインターバル中スリープを実施し、これによってシステムは、追加されたプールの除去に続いて定常状態に戻ることができる。次にブロック７６５は、現在実行中の要求についての統計のスナップショットを獲得する。次にブロック７６０はプール数を減分し、好ましくは、減らされたプール数を反映するようにプールあたりの実行時間の上限が再計算される。次にブロック７５０で、他のスリープが実行される。（ブロック７７５を参照しながら述べたように、上限は、ブロック７５０のスリープ中に適切なスリープ・インターバルを選択するか、またはスリープの前の明示的な呼出し

によって、再計算することができる。）このスリープ・インターバルが満了すると、ブロック７４０は新しいスナップショットを獲得し、ブロック７４５はこのスナップショットとブロック７６５で取得したものとを比較する。新しい実行時統計の方が良い場合、システムは論理呼出しへ戻る（ブロック７３５）ことによって、減らされたプール数で続行する。そうでなければ、プールを除去した後、インバウンド要求の実行時間および待機時間が良くなかった場合、プールが再度追加され（ブロック７５５）、論理呼出しに戻る前に、より多いプール数を使用するためにスレッド・プールへの実行時間の分配が復元（または再計算）される。

【００８８】

図８の論理は、プール・サイズ調整が実行されているときに呼び出される。ブロック８００は、変数「ＰｏｏｌＣｔｒ」（すなわち、プール・カウンタ・インデックス）を、図中では「ＮｕｍＰｏｏｌｓ」と示されている、現在アクティブな論理スレッド・プール数に設定する。第１にブロック８０５は、ゼロ・ベースのインデックス付けを使用するために、このプール・カウンタ・インデックスを減分し、次にブロック８１０は、インデックスがゼロより大きいかまたは等しいことを調べる。そうでなければ、プール・サイズ調整オペレーションはすべてのプールについて試行したことになり、図６の論理呼出しに戻る（ブロック８１５）。

【００８９】

代替の実施形態では、各プール・サイズの調整を試みる必要はない。たとえば、現在いくつのプールを使用しているかに関係なく、反復カウンタを使用して、図８の論理を通る反復回数を制限することができる。他の代替実施形態では、プール・サイズ調整を、最上位の番号ではなく最下位の番号が付けられたプールから開始することができる。当分野の通常の技術者であれば、代替方法を提供するために図８の論理がどのように修正できるか、およびこうした方法が本発明の範囲内であることが明らかになろう。

【００９０】

プール・カウンタ・インデックス値が、評価するプールがまだほかにもあることを示す場合、制御はブロック８２０に達し、プール内のスレッド数を減分するために「現在の調整量」が負の値に設定される。好ましくは、構成可能値（図中では「ＰＯＯＬ＿ＤＥＬＴＡ」と示される）が調整量として使用される。この値は、パーセンテージ、または無名数（単にスレッド数を１つずつ変更していくなど）として表すことができる。特定の実施でどの方法が最も有利であるかは、スレッド・プールの相対的サイズなどの要素に依存し、どちらの方法も本発明の範囲内である。

【００９１】

図８に示された方法は、最も長い実行要求を有するプールを第１に調整し、次に、最も短い実行要求を有するプールに向かって後進することに留意されたい。通常は、最も長い実行要求を有するプールが、最も多くの特典をプール・サイズ調整から得るものと考えられる。

【００９２】

ブロック８２５は、このプール（すなわち、プール・カウンタによってインデックス付けされたプール）からのスレッドによって現在処理されている要求タイプについて、実行統計のスナップショットが取得されることを示す。こうしたスナップショットに関する詳細は、上記のブロック７１０の考察を参照されたい。（ブロック８２５では、好ましい実施形態では統計のサブセットが取得され、現在のプールからのスレッドによって処理されている要求タイプのみが関心の対象であることに留意されたい。あるいは、作業負荷全体の統計を使用することができる。）ブロック８３０は、これらの要求タイプが、プール・サイズ調整手続きの持続期間中、このプールに結び付けられていることを示す。次にブロック８３５は、このプールのサイズを調整し、ＰＯＯＬ＿ＤＥＬＴＡにパーセンテージを使用する場合は、そのプールに割り当てられたスレッド数をＰＯＯＬ＿ＤＥＬＴＡパーセント値だけ減らすことが含まれる。（プール内のスレッド数を減らすかまたは増やす場合、プールあたりの実行時間に使用される上限値を改訂する必要はないことに留意されたい

。ただしこれは、背景で実行するために分配計算を続行する場合は、必然的に発生する可能性がある。新しい要求タイプおよびＬＲＵ処理によって廃棄されている要求タイプは、プール・サイズ調整インターバル中に、ターゲット時間に影響を与える可能性がある。）

【００９３】

次にブロック８４０は、「ＰＯＯＬ＿ＡＤＪＵＳＴＭＥＮＴ＿ＩＮＴＥＲＶＡＬ」と呼ばれるタイマ・インターバルに対してスリープまたは待機プロセスを実施し、これによってシステムは、一定時間、現在のプールの新しく変更されたサイズの下で動作できるようになる。好ましくは、このＰＯＯＬ＿ＡＤＪＵＳＴＭＥＮＴ＿ＩＮＴＥＲＶＡＬ値は構成可能であり、図７で使用されたＳＥＴＴＬＥ＿ＴＩＭＥ値と同一かまたは異なる場合がある。待機が完了すると、ブロック８２５を参照しながら上記で述べたように、ブロック８４５はこのプールのスレッドによって実行されている要求タイプのスナップショットを獲得し、ブロック８５０は、ブロック８４５からの統計がブロック８２５からのものよりも良いかどうか、すなわち、変更して良かったかどうかを調べるためのテストを行う。良かった場合、次にこの減分されたプール・サイズが維持され、調整するプールがほかにあるかどうかを判定するために制御はブロック８０５に戻る。

【００９４】

そうでなければ、減分されたプール・サイズがインバウンド要求の実行時間および待機時間を改善しなかった（すなわち、ブロック８５０の結果が否定的であった）場合に、プール・サイズはその前のサイズに復元される（ブロック８５５）。ブロック８６０から始まる処理は、このプールのサイズを減らすのではなく増やすことによって、実行時間および待機時間が改善されるのかどうかを調べるように設計されている。ブロック８６０は、ＰＯＯＬ＿ＡＤＪＵＳＴＭＥＮＴ＿ＩＮＴＥＲＶＡＬ中にスリープを実施し、これによってシステムは、プール・サイズの復元に続いて定常状態に戻ることができる。

【００９５】

スリープ・インターバルが満了した後、ブロック８６５は、プール・サイズをＰＯＯＬ＿ＤＥＬＴＡパーセンテージだけ増やすように現在の調整量を設定する。次にブロック８７０は、このスレッド・プール内のスレッドによって処理されている現在実行中の要求についての統計のスナップショットを獲得する。ブロック８７５に示されているように、要求タイプはこの手続きに関してこのプールに結び付けられているため、追加のスレッドがこれらの要求の処理に与える影響を評価することができる。次にブロック８８０は、このプールのプール・サイズを（正の）調整量で調整し、その結果プールは、より多くのスレッドを有することになる。（代替の実施形態では、パーセンテージを増やすのではなく、単なる増分方法を使用することができる。この代替方法では、好ましくは、スレッド・プール・サイズを調整する場合にスレッド数が１つだけ増やされるが、本発明の範囲を逸脱することなく他の増分も使用することができる。）

【００９６】

次に、ブロック８８５で他のスリープが実行される。このスリープ・インターバルが満了すると、ブロック８９０は新しいスナップショットを獲得し、ブロック８９５はこのスナップショットとブロック８７０で入手したものとを比較する。新しい実行時統計の方が良ければ、システムは増やされたスレッド数で続行して制御をブロック８０５に戻し、ここで、調整するスレッド・プールがほかにもあるかどうかを調べるテストが行われる。そうでなければ、プール・サイズを増やした後でインバウンド要求の実行時間および待機時間が良くなかった（すなわち、ブロック８９５で否定的な結果）場合、プール・サイズは復元される（ブロック９００）。次に他のスリープが開始され（ブロック９０５）、その後制御はブロック８０５に戻る。

【００９７】

従来技術の技法を使用して、何らかの理由でスレッド・プールに制約を与えることが、システム性能にとって有益となるかどうかを実験的に判別することができる。たとえば、データベース・アクセスを必要とする要求を参照すると、一度にオープンすることのできる最大数のデータベース接続が存在する場合がある。スレッドが制限されていなければ、

データベースにアクセスしようとする要求でプールは「閉塞状態」となる可能性がある。このシナリオでスレッド・プールを制限すれば目的は果たせるが、望ましくない副作用は、どんなデータベース・アクセスもまったく必要としない可能性のあるシステム内の他の要求にも影響を与えてしまうことである。本明細書に開示された自己調整型の動的プロセスは、システム管理者の介入なしに、この種の影響を自動的に取り除く。

【００９８】

これまで実証してきたように、本発明は、作業負荷の応答時間特徴（および具体的には、その応答時間の実行時間および待機時間構成要素）に基づいて、実行リソースのセット全体にわたって作業負荷をプログラムに基づいて分配するための、有利な技法を提供する。これにより、マルチスレッド・サーバ全体の性能が向上し、エンド・ユーザは、多くの要求の応答時間が減少することによって恩恵を受けることができる。このプログラムに基づいた分配は、本明細書で開示されたプール調整技法とは無関係に使用することが可能であり、プール数および／またはそれらプールのサイズがたとえ固定されたまま（少なくとも一時的に）であっても、性能の向上を実現することができる。あるいは、いずれかまたは両方のプール調整技法が同様に実施可能であり、他の性能の向上を提供すると予測される。開示された技法は、入ってくる様々な作業負荷タイプで有利に使用することができる。好ましい実施形態について、特定のデータ・タイプを使用したインバウンド要求の分類を参照しながら説明してきたが、これは例示を目的とするものであって、限定的なものではない。一般に、メッセージ待ち行列作業負荷は、メッセージ・タイプおよび／またはメッセージに含まれるデータ（メッセージの処理宛先を含むがこれに限定されるものではなく、たとえば処理宛先は、インバウンド作業の第１のレベルのハンドラを定義し、パラメータは、待ち行列に入れられたメッセージによって使用されるコード・パスおよびリソースが異なる追加の論理を駆動することができる）によって分類することができる。

【図面の簡単な説明】

【０１０７】

【図１】本発明に従って動作するシステムの抽象図を示す図である。

【図２】本発明の好ましい実施形態がどの要求タイプをどのスレッド・プールに割り当てるべきかを決定できる方法を示す、流れ図である。

【図３】本発明の好ましい実施形態がどの要求タイプをどのスレッド・プールに割り当てるべきかを決定できる方法を示す、流れ図である。

【図４】本発明の好ましい実施形態がプールごとの実行時間の上限をどのように決定するかを示す、流れ図である。

【図５】本発明の実施形態で使用可能な状態の移行を示す状態図であり、これらの移行は、図６～８に示された論理と組み合わせて使用することが可能である。

【図６】好ましい実施形態に従ってプールの数および各プールのサイズを調整する場合の変更を分離するための論理を示す図である。

【図７】好ましい実施形態に従ってプールの数を調整するための論理を示す図である。

【図８】好ましい実施形態に従って各プールのサイズを調整するための論理を示す図である。

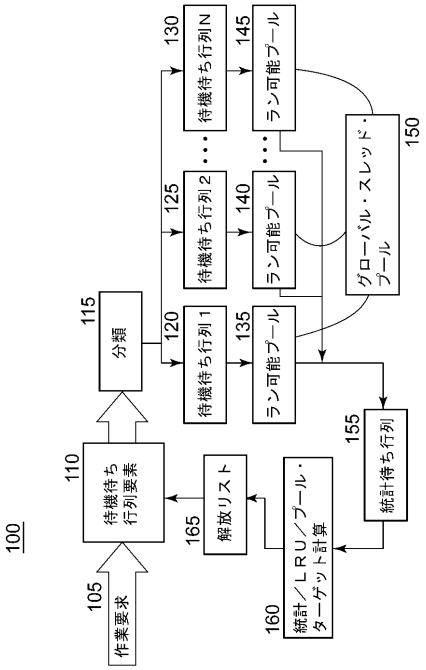
【符号の説明】

【０１０８】

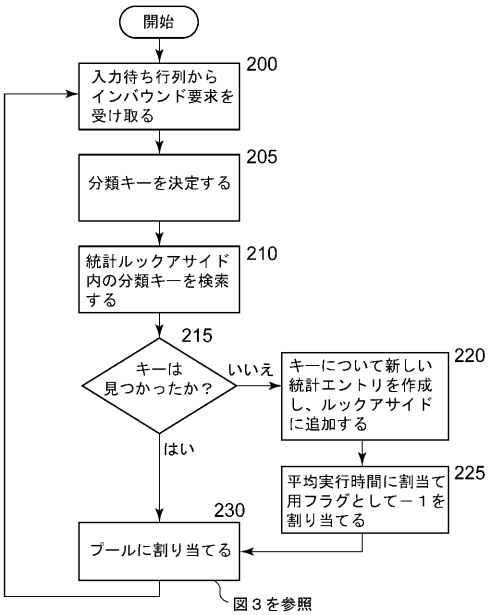
１０５　作業要求
　１１０　待機待ち行列要素
　１１５　分類
　１２０　待機待ち行列１
　１２５　待機待ち行列２
　１３０　待機待ち行列Ｎ
　１３５　ラン可能プール
　１４０　ラン可能プール
　１４５　ラン可能プール

１５０　　グローバル・スレッド・プール
１５５　　統計待ち行列
１６０　　統計／ＬＲＵ／プール・ターゲット計算
１６５　　解放リスト

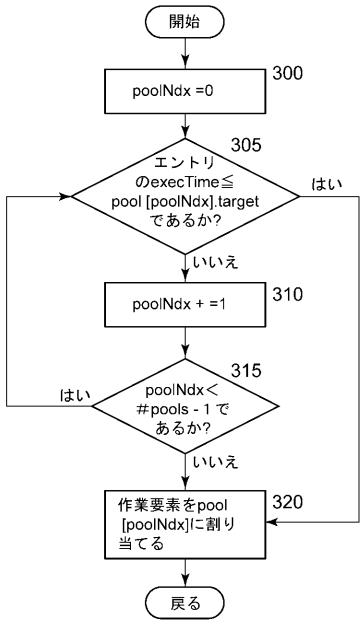
【図１】



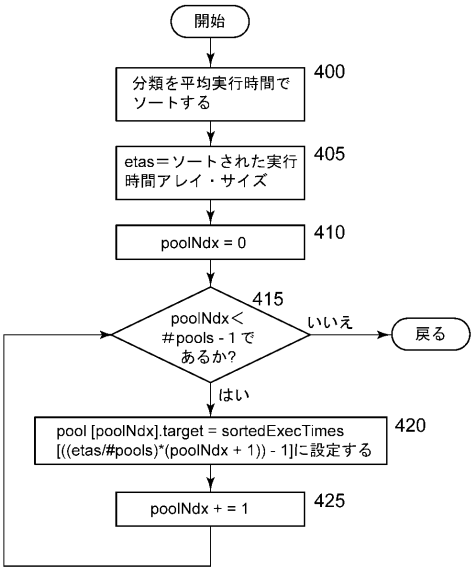
【図２】



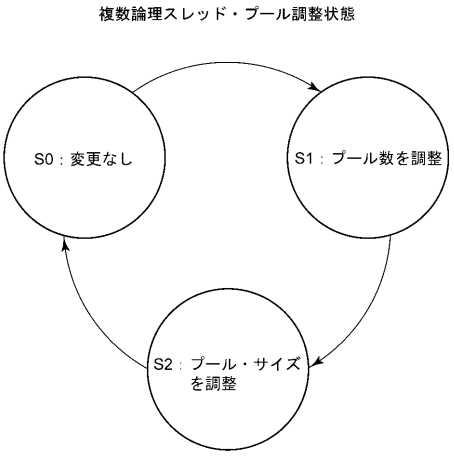
【図３】


開始
poolNdx =0
300
エントリのexecTime≦pool [poolNdx].targetであるか?
305
はい
いいえ
poolNdx + =1
310
poolNdx＜#pools - 1であるか?
315
はい
いいえ
作業要素をpool [poolNdx]に割り当てる
320
戻る

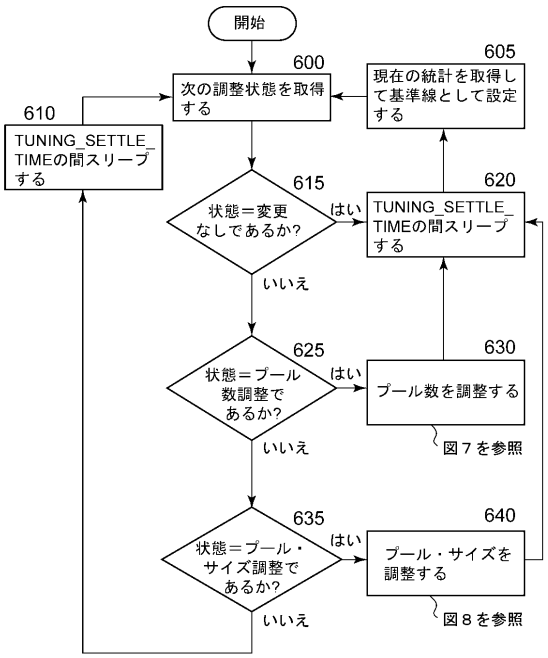
【図４】


開始
分類を平均実行時間でソートする
400
etas＝ソートされた実行時間アレイ・サイズ
405
poolNdx = 0
410
poolNdx＜#pools - 1であるか?
415
いいえ
戻る
はい
pool [poolNdx].target = sortedExecTimes [((etas/#pools)*(poolNdx + 1)) - 1]に設定する
420
poolNdx + = 1
425

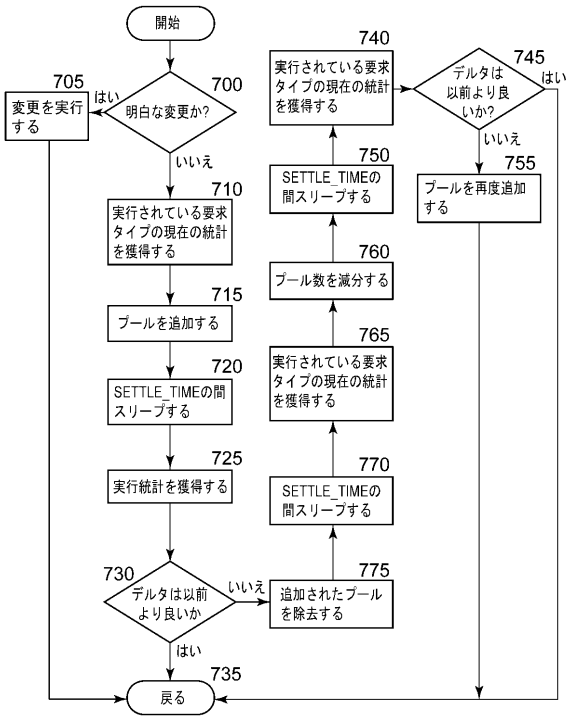
【図５】


複数論理スレッド・プール調整状態
S0：変更なし
S1：プール数を調整
S2：プール・サイズを調整

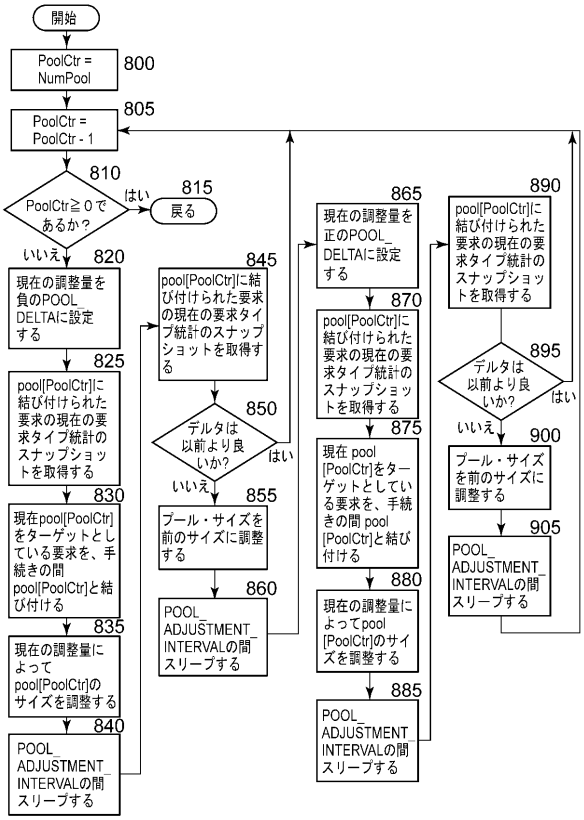
【図６】


開始
600
次の調整状態を取得する
605
現在の統計を取得して基準線として設定する
610
TUNING_SETTLE_TIMEの間スリープする
615
状態＝変更なしであるか?
はい
620
TUNING_SETTLE_TIMEの間スリープする
いいえ
625
状態＝プール数調整であるか?
はい
630
プール数を調整する
図７を参照
いいえ
635
状態＝プール・サイズ調整であるか?
はい
640
プール・サイズを調整する
図８を参照
いいえ

【図７】


開始
705
変更を実行する
はい
700
明白な変更か?
いいえ
710
実行されている要求タイプの現在の統計を獲得する
715
プールを追加する
720
SETTLE_TIMEの間スリープする
725
実行統計を獲得する
730
デルタは以前より良いか
いいえ
はい
735
戻る
775
追加されたプールを除去する
770
SETTLE_TIMEの間スリープする
765
実行されている要求タイプの現在の統計を獲得する
760
プール数を減分する
750
SETTLE_TIMEの間スリープする
740
実行されている要求タイプの現在の統計を獲得する
745
デルタは以前より良いか?
はい
いいえ
755
プールを再度追加する

【図８】


開始
PoolCtr = NumPool
800
PoolCtr = PoolCtr - 1
805
810
PoolCtr≧0であるか?
はい
815
戻る
いいえ
820
現在の調整量を負のPOOL_DELTAに設定する
825
pool[PoolCtr]に結び付けられた要求の現在の要求タイプ統計のスナップショットを取得する
830
現在pool[PoolCtr]をターゲットとしている要求を、手続きの間pool[PoolCtr]と結び付ける
835
現在の調整量によってpool[PoolCtr]のサイズを調整する
840
POOL_ADJUSTMENT_INTERVALの間スリープする
845
pool[PoolCtr]に結び付けられた要求の現在の要求タイプ統計のスナップショットを取得する
850
デルタは以前より良いか?
はい
いいえ
855
プール・サイズを前のサイズに調整する
860
POOL_ADJUSTMENT_INTERVALの間スリープする
865
現在の調整量を正のPOOL_DELTAに設定する
870
pool[PoolCtr]に結び付けられた要求の現在の要求タイプ統計のスナップショットを取得する
875
現在 pool[PoolCtr]をターゲットとしている要求を、手続きの間 pool[PoolCtr]と結び付ける
880
現在の調整量によってpool[PoolCtr]のサイズを調整する
885
POOL_ADJUSTMENT_INTERVALの間スリープする
890
pool[PoolCtr]に結び付けられた要求の現在の要求タイプ統計のスナップショットを取得する
895
デルタは以前より良いか?
はい
いいえ
900
プール・サイズを前のサイズに調整する
905
POOL_ADJUSTMENT_INTERVALの間スリープする

フロントページの続き

(72)発明者　クリストファー・ジェイムズ・ブライス
アメリカ合衆国２７５６０　ノースカロライナ州モリスビル　マーチン・タバーン・ロード　１０２６
(72)発明者　ジェンナーロ・エー・クオモ
アメリカ合衆国２７５０２　ノースカロライナ州エーペックス　ディア・バレイ・ドライブ　１１２
(72)発明者　エリック・エー・ドートレイ
アメリカ合衆国２７７１３　ノースカロライナ州ダラムランドレス・コート　２１３
(72)発明者　マット・アール・ホグストロム
アメリカ合衆国２７５０２　ノースカロライナ州エーペックス　ディア・バレイ・ドライブ　１０１

審査官　鈴木　修治

(56)参考文献　特開平０７－１５２５９０（ＪＰ，Ａ）
特開平０７－１５２７００（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　９／４６－９／５４
Ｇ０６Ｆ　１５／１６－１５／１７７