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

特許第5305626号
(P5305626)

(45)発行日　平成25年10月2日(2013.10.2)　(24)登録日　平成25年7月5日(2013.7.5)

(51)Int.Cl.　　F I
G06F 9/50 (2006.01)　　G06F 9/46 462Z
G06F 9/46 (2006.01)　　G06F 9/46 350

請求項の数 16　(全 19 頁)

(21)出願番号　特願2007-235546 (P2007-235546)
(22)出願日　平成19年9月11日 (2007.9.11)
(65)公開番号　特開2008-77652 (P2008-77652A)
(43)公開日　平成20年4月3日 (2008.4.3)
審査請求日　平成22年5月25日 (2010.5.25)
(31)優先権主張番号　11/533,558
(32)優先日　平成18年9月20日 (2006.9.20)
(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 MACHINES 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　100108501
弁理士　上野　剛史
(74)代理人　100112690
弁理士　太佐　種一
(74)代理人　100091568
弁理士　市位　嘉宏

最終頁に続く

(54)【発明の名称】論理的にパーティションされたコンピューティング環境の中央処理ユニットのリソースを、共有メモリをアクセスせずに管理する方法および装置

(57)【特許請求の範囲】
【請求項１】
１個もしくはそれ以上の物理的なマシンのリソースが論理パーティションのグループ間に割当てられる情報処理システムにおいて、
前記論理パーティションのグループ中のリクエストしているパーティションから、前記論理パーティションのグループ中の他のパーティションからの追加のリソースを求めるリソース・リクエスト・メッセージを受取るステップと、
前記リソース・リクエスト・メッセージを受取るのに応答して、前記リクエストしているパーティションが利用できるリソースを前記他のパーティションが有しているか前記他のパーティションがアセスするようにリクエストするアセスメント・リクエスト・メッセージを前記他のパーティションに送るステップと、
前記リクエストしているパーティションが利用できるリソースを前記他のパーティションが有しているか示すアセスメント応答メッセージを前記他のパーティションから受取るステップと、
前記アセスメント応答メッセージを受取るのに応答して、前記リクエストしているパーティションに他のパーティションからリソースが利用できることを知らせるメッセージを前記リクエストしているパーティションに送るステップと
を含み、
前記リクエストしているパーティションおよび前記他のパーティションが夫々関連する遅延対使用比（ＣＰＵ遅延サンプルのＣＰＵ使用サンプルに対する比であって、最新の１

分間に亘る　ＣＰＵ遅延／現在ＣＰＵ使用　として計算できる）に基づいたパフォーマンス・インデックスを備え、かつ前記リソース・リクエスト・メッセージおよびこれらのパーティションからのアセスメント応答メッセージが個々のパーティションの現在パフォーマンス・インデックスおよび個々のパーティションの投影されたパフォーマンス・インデックスを夫々特定し、かつ追加のリソースが再割当てされる方法として、

前記他のパーティションから前記リクエストしているパーティションに前記追加のリソースを再割当てすることから生じる、前記リクエストしているパーティションおよび前記他のパーティションのパフォーマンス・インデックスにおける正味の変化を調べるステップと、

前記リクエストしているパーティションのパフォーマンス・インデックスにおける正味の変化が前記他のパーティションのパフォーマンス・インデックスにおける正味の変化よりも大きい場合にのみ、前記他のパーティションから前記リクエストしているパーティションへのリソース再割当てを生じさせるステップと

を更に含む、パーティション相互間でリソースを再割当てする方法。

【請求項２】

前記論理パーティションのグループ中の複数個の他のパーティションにアセスメント・リクエスト・メッセージが送られ、かつ前記複数個の他のパーティションからアセスメント応答メッセージを受取る、請求項１に記載の方法。

【請求項３】

応答するパーティションからの複数個のアセスメント応答メッセージが、前記リクエストしているパーティションに利用できるリソースを前記応答するパーティションが有することを示す方法であって、

前記応答するパーティションの中からドナー・パーティションを選択するステップを更に含む、請求項２に記載の方法。

【請求項４】

前記他のパーティションからの前記アセスメント応答メッセージが、前記リクエストしているパーティションに利用できるリソースを有することを示す方法であって、

前記リクエストしているパーティションにリソースが利用できるようにするために前記他のパーティションがそれ自体を再構成することをリクエストする再構成リクエスト・メッセージを前記他のパーティションに送るステップと、

前記リクエストしているパーティションにリソースを利用できるようにするために、前記他のパーティションがそれ自体を成功裡に再構成したことを示す再構成応答メッセージを受取るステップと前記他のパーティションから受取るステップと

を更に含む、請求項１に記載の方法。

【請求項５】

前記リクエストしているパーティションにリソースを利用できるようにするために前記他のパーティションがそれ自体を成功裡に再構成したのでないことを前記再構成応答メッセージが示すなら、他のパーティションからは利用できるリソースがないことを、前記リクエストしているパーティションへの前記メッセージが知らせる、請求項４に記載の方法。

【請求項６】

前記リクエストしているパーティションにリソースを利用できるようにするために前記他のパーティションがそれ自体を成功裡に再構成したことを前記再構成応答メッセージが示すなら、前記他のパーティションによって利用できるようにされたリソースを受取るために前記リクエストしているパーティションがそれ自体を再構成することを前記リクエストしているパーティションへの前記メッセージがリクエストする、請求項４に記載の方法。

【請求項７】

前記リクエストしているパーティションに利用できるリソースがないことを前記アセスメント応答メッセージが示すなら、他のパーティションからは利用できるリソースがない

ことを、前記リクエストしているパーティションへの前記メッセージが前記リクエストしているパーティションに知らせる、請求項１に記載の方法。

【請求項８】

前記メッセージがネットワーク・プロトコルを用いて送られる、請求項１に記載の方法。

【請求項９】

前記リソースがＣＰＵリソースである、請求項１に記載の方法。

【請求項１０】

１個もしくはそれ以上の物理的なマシンのリソースが論理パーティションのグループ間に割当てられる情報処理システムにおいて、

前記論理パーティションのグループ中のリクエストしているパーティションからリソース・リクエスト・メッセージを受取るのに応答して、前記リクエストしているパーティションが利用できるリソースを前記他のパーティションが有しているか前記他のパーティションがアセスするようにリクエストするアセスメント・リクエスト・メッセージを他のパーティションに送るアセスメント・コレクタと、

前記リクエストしているパーティションが利用できるリソースを前記他のパーティションが有しているか示す、前記他のパーティションからのアセスメント応答メッセージを受取るのに応答して、前記リクエストしているパーティションに他のパーティションからリソースが利用できるかを知らせるメッセージを前記リクエストしているパーティションに送るアセスメント・アナライザと

を含み、

前記リクエストしているパーティションおよび前記他のパーティションが夫々関連する遅延対使用比（ＣＰＵ遅延サンプルのＣＰＵ使用サンプルに対する比であって、最新の１分間に亘る　ＣＰＵ遅延／現在ＣＰＵ使用　として計算できる）に基づいたパフォーマンス・インデックスを備え、かつ前記リソース・リクエスト・メッセージおよびこれらのパーティションからのアセスメント応答メッセージが個々のパーティションの現在パフォーマンス・インデックスおよび個々のパーティションの投影されたパフォーマンス・インデックスを夫々特定し、追加のリソースが再割当てされる装置として、前記アセスメント・アナライザが、

前記他のパーティションから前記リクエストしているパーティションに前記追加のリソースを再割当てすることから生じる、前記リクエストしているパーティションおよび前記他のパーティションのパフォーマンス・インデックスにおける正味の変化を調べ、

前記リクエストしているパーティションのパフォーマンス・インデックスにおける正味の変化が前記他のパーティションのパフォーマンス・インデックスにおける正味の変化よりも大きい場合にのみ、前記他のパーティションから前記リクエストしているパーティションへのリソース再割当てを生じさせる、パーティション相互間でリソースを再割当てする装置。

【請求項１１】

前記アセスメント・コレクタが、前記論理パーティションのグループ中の複数個の他のパーティションにアセスメント・リクエスト・メッセージを送り、かつ前記アセスメント・アナライザが、前記複数個の他のパーティションからアセスメント応答メッセージを受取る、請求項１０に記載の装置。

【請求項１２】

前記アセスメント・アナライザが、応答するパーティションからの複数個のアセスメント応答メッセージを受取り、前記リクエストしているパーティションに利用できるリソースを前記応答するパーティションが有することを示すのに応答して、前記応答するパーティションの中からドナー・パーティションを選択する、請求項１１に記載の装置。

【請求項１３】

前記アセスメント・アナライザが、前記他のパーティションからのアセスメント応答メッセージを受取り、前記リクエストしているパーティションに利用できるリソースを有す

ることを示すのに応答して、前記リクエストしているパーティションにリソースが利用できるようにするために前記他のパーティションがそれ自体を再構成することをリクエストする再構成リクエスト・メッセージを前記他のパーティションに送る、請求項１２に記載の装置。

【請求項１４】

前記リクエストしているパーティションにリソースを利用できるようにするために前記他のパーティションがそれ自体を成功裡に再構成したことを示す再構成応答メッセージを前記他のパーティションから受取り損ねたことに前記アセスメント・アナライザが応答して、他のパーティションからは利用できるリソースがないことを、前記リクエストしているパーティションに知らせるメッセージを送る、請求項１３に記載の装置。

【請求項１５】

前記リクエストしているパーティションにリソースが利用できるようにするために前記他のパーティションが成功裡にそれ自体を再構成したことを示す再構成応答メッセージを前記他のパーティションから受取るのに応答して、前記アセスメント・アナライザが、前記他のパーティションにより利用できるリソースを受取るために前記リクエストしているパーティションがそれ自体を再構成することをリクエストする再構成リクエスト・メッセージを前記リクエストしているパーティションに送る、請求項１３に記載の装置。

【請求項１６】

前記リクエストしているパーティションに利用できるリソースを他のパーティションが有することを示すアセスメント応答メッセージを受取り損ねるのに応答して、前記アセスメント・アナライザが、他のパーティションから利用できるリソースがないことを知らせるメッセージを前記リクエストしているパーティションに送って知らせる、請求項１０に記載の装置。

【発明の詳細な説明】

【技術分野】

【０００１】

本出願に関連するのは、同一出願人による同時米国特許出願で、Mathew S. Thoennesほかの発明者による「重層的な環境で単一のサーバーのパフォーマンス貢献度を測定するローカル・パフォーマンス・インデックスを予測するための方法および装置（METHOD AND APPARATUS FORESTIMATING A LOCAL PERFORMANCE INDEX TO MEASURE THE PERFORMANCE CONTRIBUTION OFA SINGLE SERVER IN A MULTI-TIERED ENVIRONMENT）」と題する米国特許出願番号１１／５３３５７４（出願人整理番号　ＰＯＵ９－２００６－００７５）である。

【０００２】

本発明は概略、コンピューティング・システム中のワークロードを管理することに関し、特に論理的にパーティション（区画）されたシステム中のワークロードを管理することに関する。

【背景技術】

【０００３】

単一の物理的なマシンが複数の論理的なマシンにパーティションされる論理的パーティション・システムは、単一の物理的なマシンもしくは中央プロセッサ・コンプレックス（ＣＰＣ）中に複数のシステム・イメージを確立するのを可能にする。各システム・イメージは、恰もそれが別個のコンピュータ・システムであるかのように動作することができる。即ち、各論理パーティションが独立にリセットされ、論理パーティションごとに異なってもよいオペレーティング・システム中で初期ロードされ、そして異なる入出力（Ｉ／Ｏ）装置を用い異なるソフトウエアで以って動作することができる。

【０００４】

論理的にパーティションされたコンピューティング・システムは、例えば「Partitioned Multiprocessor Programming System（パーティションされたマルチプロセッサ・プログラミング・システム）」と題する特許文献１、「LogicalResource Partitioning of a Data Processing System（データ処理システムの論理的リソース・パーティショニング）」

と題する特許文献２、および「Method and Apparatus for Providing a server Function in a LogicallyPartitioned Hardware Machine（論理的にパーティションされたハードウエア・マシンにおいてサーバー機能を提供するための方法および装置）」と題する特許文献３に記されている。論理的にパーティションされたシステムの市販例は、例えばＩＢＭ ｚＳｅｒｉｅｓおよびＳｙｓｔｅｍ ｚ９（総称して Ｓｙｓｔｅｍ ｚ）システムならびにＩＢＭ ＰＯＷＥＲ５ベース・システム（ＰＯＷＥＲ５、Ｓｙｓｔｅｍ ｚ および Ｓｙｓｔｅｍ ｚ９ は商標であり、またｚＳｅｒｉｅｓはＩＢＭ Ｃｏｒｐｏｒａｔｉｏｎの登録商標である。）
また特許文献１には、後述のソース同期インターフェイスの詳細が説明されている。
特許文献２および特許文献３には、ＩＢＭ Ｄｅｓｉｇｎ Ｄａｔａ Ｍａｎａｇｅｒ（ＤＤＭ）の詳細が説明されている。

【０００５】
論理的にパーティションされたシステムの一つの重要な側面は、そのシステムのパーティション内で実行するワークロードの管理である。産業界では各パーティション内で、あるいは複数のパーティションの間でワークロードを管理するのに使用される多くの例のワークロード・マネジャがある。一つのそのようなワークロード・マネジャは、ｚ／ＯＳオペレーティング・システムである。（ｚ／ＯＳはＩＢＭ Ｃｏｒｐｏｒａｒｔｉｏｎの登録商標である。）一般に、これらのワークロード・マネジャはシステムの物理的なリソースにワーク（作業）を移すことによって複数のパーティションのワークロードをバランスさせようとする。しかし、ワークロードを移すためには再配置されたワークにより必要とされるデータがその移された位置にあるように確保することが重要である。この必要性はワークを移すのをしばしば制限する。

【０００６】
ｚ／ＯＳワークロード・マネジャはこの能力を、論理パーティション間で中央処理装置（ＣＰＵ）容量を移す能力を提供することによって改良した。この能力については特許文献４に説明されている。しかし、このｚ／ＯＳワークロード・マネジャはピアツーピアの管理アプローチ（方法）、即ち各パーティション上でワークロード管理機能が、管理されている他の複数のパーティションの各々の上で実行するワークロードの全知識を持つというアプローチを用いる。これはパーティション間で共有されるデータを大量に必要とする。効率的にこのデータ共有をするには、そのパーティションに跨るメモリを共有するための機構にアクセスする、管理されている各パーティションにｚ／ＯＳワークロード・マネジャが依存する。そのようなメモリ共有能力は全ての論理的にパーティションされた環境で利用できるものでなく、ｚ／ＯＳアプローチの能力を制限してしまう。
また特許文献２、特許文献４、特許文献５、特許文献６には、ＩＢＭ ＤＤＭ ライブラリ・プロセス・マネジャについて詳細に説明している。
論理パーティションのグループについては、「（コンピューティング環境のパーティションのグループを管理するための方法、システムおよびプログラム製品）」と題する特許文献７にある。
非特許文献１には、後述のＡＰＩ（アプリケーション・プログラミング・インターフェース）が説明されている。

【特許文献１】米国特許４，５６４，９０３号（Guyette et al.）
【特許文献２】米国特許４，８４３，５４１号（Bean et al.）
【特許文献３】米国特許５，５６４，０４０号（Kubala）
【特許文献４】米国特許６，５８７，９３８号
【特許文献５】米国特許５，４７３，７７３号（Aman et al.）
【特許文献６】米国特許５，６７５，７３９号（Eilert et al.）
【特許文献７】米国特許７，００７，２７６号（Kubala et al.）
【非特許文献１】ＩＢＭ刊行物AIX 5L Version 5.3Technical Reference: Base Operating System and Extensions, Volume 1, SC23-4913-03, Fourth Edition (July 2006)

【発明の開示】

【発明が解決しようとする課題】
【０００７】
本発明は、例えば特許文献４に説明される論理的にパーティションされたコンピュータ・システムのパーティションに跨ってＣＰＵリソースを管理するための機能を提供しようとするものである。両方のアプローチはこれらのパーティション上で実行している異なるワークロードのパフォーマンス目標を達成するためにＣＰＵリソースを管理する。しかし、本発明はパーティション相互間で大量のデータを共有する必要がなく、従ってパーティション相互間での共有メモリ能力のための同程度の必要性を持たない。
【課題を解決するための手段】
【０００８】
もっと具体的には、一つもしくはそれ以上の物理的なマシンが、一つのグループの論理パーティション相互間に割当てられる情報処理システムにおいてパーティション相互間でリソースを再割当てするための方法、装置およびコンピュータ・プログラムを本発明は企図している。本発明によれば、アセスメント・コレクタが、リクエストしているパーティションからリソース・リクエスト・メッセージを受取るのに応答して、アセスメント・リクエスト・メッセージをそのグループの中の他のパーティションにアセスメント・リクエスト・メッセージを送り、これによって他のパーティションがそのリクエストしているパーティションのために利用できるリソースを有するかどうかを他のパーティションがアセス（評価）する。アセスメント・アナライザが他のパーティションからのアセスメント応答メッセージであって、他のパーティションがそのリクエストしているパーティションのために利用できるリソースを有しているかを示す前記アセスメント応答メッセージを受取るのに応答して、そのリクエストしているパーティションに、他のパーティションから利用できるリソースがあるかを知らせるメッセージを送る。
【０００９】
好適には、アセスメント・コレクタがそのグループ中の他の複数のパーティションにアセスメント・リクエスト・メッセージを送り、そしてそこからアセスメント・アナライザがアセスメント応答メッセージを受取る。そのような場合、アセスメント・アナライザは複数の肯定的アセスメント応答メッセージを受取るのに応答して、その応答しているパーティションの中から一つのドナー・パーティションを選択する。
【００１０】
他のパーティションから肯定的アセスメント応答メッセージを受取るのに応答して（そしてもし複数の肯定応答パーティションがあるなら一つのドナー・パーティションを選択した後に）、アセスメント・アナライザはそのドナー・パーティションに再構成リクエスト・メッセージを送る。このメッセージはそのリクエストしているパーティションがリソースを利用できるようにするためにそのドナー・パーティションがそれ自体を再構成するようにリクエストするメッセージである。肯定的再構成応答メッセージというのは他のパーティションがそれを成功裡に再構成したことを示すのであるが、もしアセスメント・アナライザが他のパーティションから肯定的再構成応答メッセージを受取らないならば、それは他のパーティションから利用できるリソースがないことを、そのリクエストしているパーティションに知らせるメッセージを送る。
【００１１】
もしアセスメント・アナライザが他のパーティションから肯定的再構成応答メッセージを受取るならば、それは他のパーティションにより利用できるようにされたリソースを受取るようにそれがそれ自体を再構成することをリクエストするメッセージをそのリクエストしているパーティションに送る。
【００１２】
もし、他のパーティションがそのリクエストしているパーティションで利用できるリソースを有しているという肯定的アセスメント応答メッセージをアセスメント・アナライザが受取らなければ、それは他のパーティションから利用可能という応答がないことを、リクエストしているパーティションに知らせるためのメッセージを送る。

【００１３】
好適には、リクエストしているパーティションおよび他のパーティションが夫々協働するパフォーマンス・インデックスを有し、これらのパーティションからのリソース・リクエスト・メッセージおよびアセスメント応答メッセージが夫々、その各々のパーティションの現在パフォーマンス・インデックスおよび追加のリソースを再割当てされた各々のパーティションの投影パフォーマンス・インデックスを特定する。そのような場合、アセスメント・アナライザがそのリクエストしているパーティションのパフォーマンス・インデックス、およびそのドナー・パーティションからそのリクエストしているパーティションに追加のリソースを再割当てすることから生じるドナー・パーティション
の正味の変化を決定する。そのリクエストしているパーティションのパフォーマンス・インデックスの正味の変化がドナー・パーティションのパフォーマンス・インデックスの正味の変化よりも大きい場合にのみ、アセスメント・アナライザはドナー・パーティションからそのリクエストしているパーティションにリソース再割当てを生じる。
【００１４】
特許文献４に記載された実装例のように、本発明は管理されているパーティション上で実行しているワークロードをサービス・クラスの中にオーガナイズする。その際、各サービス・クラスはそれと関連するパフォーマンス目標（ゴール）を有する。本発明は、ワークロードに基づき、複数のパーティションに跨るＣＰＵリソースの管理を実装するために特許文献４に記されたような実装例の基本的なタスクをも実行する。これらのタスクは以下のとおりである。
１．管理されているパーティションのうちの一つにＣＰＵリソースがないことから、サービス・クラスがその目標をミスしていることを決定する。
２．このパーティションへのＣＰＵリソースの割当てを増加することによってどれだけのサービス・クラスがヘルプされるかを投影する。
３．ヘルプを必要とするパーティションにＣＰＵリソースを物理的に譲渡することができる他のパーティションを見つける。
４．これらのドナー・パーティションからＣＰＵリソースを除去することによって引き起こされるであろうドナー・パーティション上で実行するサービス・クラスのパフォーマンスへのインパクトを投影する。
５．ドナーおよびレシーバのためのパフォーマンス投影に基づき、ヘルプを必要とするパーティションにドナー・パーティションの一つからＣＰＵリソースを移すことが、ワークロードの目標を全体としてもっと緊密に達成する結果となる。
【００１５】
特許文献４に説明される実例では、前述のタスクの全てが、ヘルプを必要とするサービス・クラスを持つパーティション上のワークロード管理機能によって実行される。従来例ではパーティション相互間で共有する大量のパフォーマンス・データが必要な、異なるパーティション上で実行するワークロードについてパフォーマンス投影を行うことが要求される。パーティション相互間で共有しなければならないデータの量を大きく減じるために、本発明はこれらのステップを各パーティション上のワークロード管理機能とドメイン・マネジャと呼ばれる中央管理サーバー機能との間に割り振る。本発明では所与のパーティション上で実行するワークフローのための容量の変化の投影がパーティションのローカル・ワークロード管理機能によって行われる。従って低レベルのパフォーマンス・データはローカル・パーティション上で必要とされるだけである。ドメイン・マネジャは、ＣＰＵリソースが跨ってバランスし得るようなそしてローカル・パーティション上で為される投影に基づいて最適のリソース再割当てを決定するようなパーティションのトポロジを理解するために使用される。
【００１６】
本発明は、従来からある相互接続ハードウエアを用い、従来からあるコンピュータ・ハードウエア上で実行されるソフトウエアないしプログラムで実施されることが望ましい。しかし、一般には本発明が、（マイクロコードを含む）ハードウエア、ソフトウエア、また

はその２つの組合せで実施されてもよい。ソフトウエアまたはマイクロコードで実施されるとき、その実施は本発明の方法を実施するためのコンピュータ使用可能媒体上などにある命令からなるコンピュータ読み取り可能なプログラムの形式をとってもよい。そのようなソフトウエアまたはマイクロコードの実施例では、装置（手段のエレメントを含むこともある）の請求項に記載された種々のエレメントが、課題の諸ステップを実行するためのプログラム命令に対応する。

【発明を実施するための最良の形態】

【００１７】

本発明は、「Apparatus and method for Managing a Data Processing System WorkloadAccording to Two or More Distinct Processing Goals（２個もしくはそれ以上の区別できる処理目的に従ったデータ処理システム・ワークロードを管理するための装置および方法）」と題する特許文献５ならびに「Apparatus and Method for managing a Distributed data ProcessingSystem Workload According to a Plurality of Distinct Processing Goal types（複数個の区別できる処理目的のタイプに従ったデータ処理システム・ワークロードを管理するための装置および方法）」と題する特許文献６に記載されたワークロード・マネジャによるオブジェクト指向リソース管理の概念を確立する。これらの特許はコンピュータ・システムのワークロードをサービス・クラスにオーガナイズ（編成）するという概念を記述する。各サービス・クラスと関連するのはパフォーマンス目標である。各サービス・クラスと関連するのはまた、他のワークロードのサービス・クラス目標に比較されるそのサービス・クラス目標を達成するためにそれがどれだけ重要かを示す重要度レベルである。

【００１８】

図１は、１個もしくはそれ以上のパーティションで仕切られたコンピュータ・システム１００が管理される本発明を組み込んだデータ処理システムを示す。本発明は特定のアーキテクチャに限定されないが、そのようなコンピュータ・システム１００の例はＩＢＭ POWER５アーキテクチャに基づくコンピュータ・システムである。 この分野でよく知られた態様で、各コンピュータ・システム１００は、ＰＯＷＥＲ５もしくは他の命令セット・アーキテクチャを組み込んだ１個もしくはそれ以上の中央処理装置（ＣＰＵ）（とりたてて示さず）を含む。各コンピュータ・システムはまた論理パーティション・マネジャ即ちハイパーバイザ１０１を含む。その役割はコンピュータ・システム１００の論理パーティションを制御することである。例示のためそのうちの３個を示す。論理パーティションに割当てられた物理的なＣＰＵリソースのシェアに対する変更はハイパーバイザ１０１を介して為される。

【００１９】

管理される複数のパーティションの夫々で実行しているのはオペレーティング・システム（ＯＳ）および典型的には１個もしくはそれ以上のアプリケーションである。これらは従来のエレメントであるので、取り立てて示さない。本発明はどれか特定のオペレーティング・システムに限定されないが、 図示の実施例ではオペレーティング・システムがＩＢＭ ＡＩＸオペレーティング・システムと想定する（ＡＩＸはＩＢＭ Corporationの登録商標）。また管理されている各パーティションで実行しているのはワークロード管理（ＷＬＭ）コンポーネント１０２である。ＷＬＭコンポーネント１０２の役割は、そのパーティションで実行するワークロードのパフォーマンス（性能）についてのデータを収集することと、ハイパーバイザ１０１により提供されるアプリケーション・プログラム・インターフェイス（ＡＰＩ）を介してそのパーティションに割当てられたＣＰＵリソースの量を変更させることとである。

【００２０】

コンピュータ・システム１００上の論理パーティションは幾つかのグループに編成されることができる。グループは、ＣＰＵリソースが同じグループ中のパーティション間で移動されるだけでもよいようにＣＰＵリソースの範囲を制限する。論理パーティションのグループの説明は「（計算環境のパーティションのグループを管理する方法。システムおよび

プログラム製品）」と題する特許文献７にある。

【００２１】

ドメイン・マネジャ１０３は、管理されている全てのコンピュータ・システム１００にまたがるグローバル・コントロール・ポイントである。ドメイン・マネジャ１０３の役割は、パーティションのトポロジを、そのパーティションに亘ってリソースをバランスすることができるように理解することと、管理されているパーティションのための最適のＣＰＵリソース割当てを決定することである。ドメイン・マネジャ１０３は管理されているコンピュータ・システム１００へのネットワーク・アクセスを有する任意のコンピュータ・システム（複数のコンピュータ・システム１００のうちの一つを含む）で実行させることができる別個のアプリケーションである。

【００２２】

ドメイン・マネジャ１０３は幾つかのサブコンポーネントを備えている。抽象化マネジャ（ＶＭ）１０４は論理パーティションＣＰＵリソースのトップレベル・マネジャとして機能する。ＶＭ１０４は、論理パーティション・グループのパーティションが実行するコンピュータ・システム１００に対するその論理パーティション・グループの関係を理解する。それはまた各ＷＬＭコンポーネント１０２とそれが実行する論理パーティションとの間の関係を理解する。ＶＭサブコンポーネント１０４は特定のパーティション・グループの管理を、以下で説明する抽象グループ（ＶＧ）サブコンポーネント１０５に委譲する。ＶＭサブコンポーネント１０４は抽象化サーバー管理に関する全てのメッセージを受取り、それらを適切なＶＧサブコンポーネント１０５に差し向ける。

【００２３】

ＶＧサブコンポーネント１０５はドメイン・マネジャ１０３への論理パーティション・グループを表す。それは特定の論理パーティション・グループのために論理パーティション管理機能を提供する。そのＶＧサブコンポーネント１０５はその論理パーティション・グループの一部である論理パーティションのセットを理解する。それはそのような論理パーティション・グループ中のパーティション相互間でＣＰＵリソース割当てリクエストをアービトレートする役割を有する。

【００２４】

各論理パーティションで実行するＷＬＭコンポーネント１０２はまた論理パーティションＣＰＵリソース管理をサポートするために２つのサブコンポーネントを有する。抽象化エージェント（ＶＡ）コンポーネント１０７というサブコンポーネントはその論理パーティションのためのハイパーバイザ１０１に接続する役割がある。ＶＡコンポーネント１０７はハイパーバイザ１０１により提供されるＡＰＩを介しそのパーティションにより使用されるＣＰＵリソースについてのデータを収集する。そのようなＡＰＩの一例はＩＢＭ　ＡＩＸオペレーティング・システムにより提供されるlpar_get_info ＡＰＩ である。このＡＰＩは、オンラインで入手でき、またここで引用により組み込まれる非特許文献１に記述されている。このＶＡコンポーネント１０７は、そのパーティションに割当てられるＣＰＵリソースの量を変更するためにハイパーバイザＡＰＩを呼び出す役割もある。そのようなＡＰＩの一例は、ＩＢＭ　ＡＩＸ　オペレーティング・システムにより提供されるlpar_set_resource ＡＰＩである。これは同じＩＢＭ　ＡＩＸ刊行物（非特許文献１）に記述されている。

【００２５】

リソース・オプティマイザ（ＲＯ）１０６は、その論理パーティション上で実行するサービス・クラスのパフォーマンスを理解する役割がある。それはまたそのパーティションにＣＰＵリソースを加えるかまたはそのパーティションからＣＰＵリソースを除去するかのどちらかのサービス・クラスにパフォーマンスのインパクトを投影する役割もある。

【００２６】

図２は、管理されている各パーティション上のＷＬＭコンポーネント１０２中での処理を示す。この処理は、例えば１０秒間隔といったように間欠的に行われる。ステップ２０１では、ＷＬＭコンポーネント１０２がパーティション上で実行する各サービス・クラスの

ためにパフォーマンス・データを収集する。更に、「ＣＰＵ使用中」および「ＣＰＵ遅延」のサンプルが各サービス・クラスのために収拾される。パフォーマンス・データおよびＣＰＵ使用中ならびにＣＰＵ遅延のサンプルを収集するこれらのステップは特許文献５に説明されている。このデータ収集プロセスにおける追加のステップは、そのパーティションにより使用される全ＣＰＵリソースについての情報を収集するハイパーバイザＡＰＩを呼び出すＶＡコンポーネント１０７のためである。そのようなハイパーバイザＡＰＩの例は、ＩＢＭ　ＡＩＸオペレーティング・システムにより提供される前述のlpar_get_infoＡＰＩである。

【００２７】

ステップ２０２では、ステップ２０１で収集されたパフォーマンス・データが、そのパーティションにより消費されるＣＰＵリソースとサービス・クラスのパフォーマンスとの関係をプロットする、サービス・クラスごとのグラフを更新するのに使用される。このグラフは図9に示し、下記で説明する。

【００２８】

ステップ２０３では、ＲＯコンポーネント１０６は前述の特許文献５に説明されるように、各サービス・クラスのためにパフォーマンス・インデックス（ＰＩ）を計算する。（各階層(tier)に複数のサーバーを備えた複数階層システム(multi-tier system)では、この方法は前述した関連するアプリケーションに説明されたように修正されてもよい。）

【００２９】

ステップ２０４では、いずれかのサービス・クラスがそのパフォーマンス・ゴールを見失っているかいないかをＲＯコンポーネント１０６が調べる。もしどのサービス・クラスもそのゴールを見失っていなければ、その処理はステップ２０５で次のインターバルになるまで待機し、そしてその処理を繰り返す。

【００３０】

もし少なくとも一つのサービス・クラスがゴールを見失っていれば、処理はステップ206に進み、そこでそのゴールを見失っている最も重要なサービス・クラスが選択される。もし同じ重要度レベルにある複数のサービス・クラスがそのゴールを見失っていれば、最も高いパフォーマンス・インデックスを持つサービス・クラスが選択される。この選択されたサービス・クラスはレシーバ・サービス・クラスと呼ばれる。ステップ２０７において、このレシーバ・サービス・クラスはそれがＣＰＵ遅延サンプルをすでに収集しているか調べるためにチェックされる。もしそのレシーバ・サービス・クラスがまだＣＰＵ遅延サンプルを収集していなければ、そのパーティションに与えられるＣＰＵリソースを増加することによってはそのパフォーマンスの助けにはならない。この場合、処理はステップ２０４および２０６に戻り、そこでそのゴールを見失っている他のサービス・クラスが選択される。ゴールを見失っている全てのサービス・クラスが選択されてしまうと、処理はステップ２０５に進み、次のインターバルまで待機する。

【００３１】

ステップ２０８では、パーティションに与えられるＣＰＵリソースを増加することによってレシーバ・サービス・クラスのパフォーマンスがどれだけ改良されるかを知るために投影が行われる。ステップ２０９ではＣＰＵリソースの再割当てを正当化するに足りるかテストする。好適な実施例では、このテストはレシーバ・サービス・クラスのために投影されるパフォーマンス・インデックスの改良が（１）レシーバ・サービス・クラスの現在パフォーマンス・インデックスと１．０との差の１０％よりも大きく、かつ（２）０．０５よりも大きいかというテストである。

【００３２】

もし充足値に対するこのテストに通らなければ、処理はステップ２０４に戻り、他の潜在的なレシーバ・サービス・クラスを探す。

【００３３】

もし充足値であれば、ステップ２１０では、「ヘルプの申し立て」メッセージがドメイン・マネジャ１０３に送られる。このメッセージはリクエストされたＣＰＵリソース、レシ

ーバ・サービス・クラスの名称、レシーバ・サービス・クラスの現在のパフォーマンス・インデックスおよびレシーバ・サービス・クラスの投影されたパフォーマンス・インデックスの全体を含む。

【００３４】

各サービス・クラスと関連するのはパーティションに与えられるＣＰＵリソースの変更のパフォーマンス・インパクトを投影するのに利用されるグラフである。この投影の結果はパフォーマンス・インデックスの投影された変更である。パフォーマンス投影のためにグラフを使用することは、文献５に示されている。

【００３５】

図９は使用されるグラフの一例である。ｙ軸は、ある期間、例えばこの実施例では１分間に亘るＣＰＵ遅延サンプルのＣＰＵ使用サンプルに対する比である。この比は遅延対使用比と呼ばれる。このグラフのｘ軸は同じ期間に亘るパーティションにより使用されるＣＰＵリソース（平均的に使用されるＣＰＵリソース）の量である。前述のステップ２０２では最新のデータがそのグラフに加えられる。

【００３６】

パフォーマンス・インパクトを投影するための第１のステップは、最新の１分間に亘りパーティションにより使用されるＣＰＵリソースに対応する遅延対使用比をグラフから読み出すことである。その次のステップは、追加のＣＰＵリソースがパーティションに与えられた後のＣＰＵリソースに対応する投影された遅延対使用比を読み出すことである。リソースの変更がＣＰＵ遅延サンプルに大きなインパクトを与えることと、ＣＰＵ使用サンプルが一定であることとを仮定する。従って、遅延対使用比からは、ＣＰＵ遅延サンプルが以下の式１のように計算できる。

【数１】

ＣＰＵ遅延＝遅延対使用比×現在ＣＰＵ使用

$$CPUDelay = DelayToUSINGRatio \times CurrentCPU\,USING$$

【００３７】

ＣＰＵ遅延サンプルに於ける投影された変更は以下の式２で計算できる。

【数２】

△ＣＰＵ遅延＝現在ＣＰＵ遅延－投影されたＣＰＵ遅延

$$\Delta CPUDelay = CurrentCPUDelay - \mathrm{Pr}\,ojectedCPUDelay$$

【００３８】

サービス・クラスのために見られる合計サンプルがサービス・クラスの平均応答時間にほぼ比例する。この関係は以下の式３で表される。

【数３】

△ＣＰＵ遅延／合計サンプル＝△応答時間／現在応答時間

$$\frac{\Delta CPUDelay}{TotalSamples} = \frac{\Delta \mathrm{Re}\,sponseTime}{Current\,\mathrm{Re}\,sponseTime}$$

【００３９】

従って、式４は以下のとおり。

【数４】

△応答時間＝現在応答時間＊△ＣＰＵ遅延／合計サンプル

$$\Delta \mathrm{Re}\,sponseTime = \frac{Current\,\mathrm{Re}\,sponseTime * \Delta CPUDelay}{TotalSamples}$$

【００４０】

PIに投影される変更は以下の式５のように計算される。

但し、式５中、ゴール(Goal)はサービス・クラスのための応答時間の目標である。
【数５】
投影されたＰＩ＝（ＰＩ＊ゴール＋△ローカル応答時間）／ゴール

$$ProjectedPI = \frac{PI * Goal + \Delta Local\,ResponseTime}{Goal}$$

【００４１】
図３は、管理されているパーティション上のＷＬＭコンポーネント１０２とドメイン・マネジャ１０３との間のデータ・フローを示す。このメッセージは任意の利用可能なネットワーク・プロトコル、例えばＴＣＰ／ＩＰ上で流れる。図示のとおり、ドメイン・マネジャ１０３のＶＭサブコンポーネント１０４がメッセージを最初に受取り、それからそのメッセージを送ったパーティションを管理するＶＧサブコンポーネント１０５に委譲する。
【００４２】
図４はドメイン・マネジャ１０３が「ヘルプの申し立て」を受けるときの処理を示す。この処理はアセスメント・コレクタ機能（ステップ４０１－４０３）とアセスメント・アナライザ機能（４０４－４１２）とに都合よく分割されることができ、前者では他のパーティションからアセスメント（評価）が収集され、後者ではアセスメントが分析される。
【００４３】
ステップ４０１では、ドメイン・マネジャ１０３が「ヘルプの申し立て」メッセージを受取り、そしてステップ４０２でそのメッセージを適正なＶＧサブコンポーネント１０５に委譲する（もしくは進める）。「ヘルプの申し立て」メッセージを送ったパーティションがレシーバ・パーティションで知られる。ステップ４０３では、ＶＧサブコンポーネント１０５が「ヘルプの申し立て」メッセージをそのパーティション・グループ中、そのレシーバ・パーティションを除く全ての各パーティションに送る。このメッセージは、そのパーティション上で実行するワークロード上のＣＰＵリソースを譲渡するインパクトを投影するためにこれらのパーティションの各々でＷＬＭコンポーネント１０２を問い合わせる。このメッセージはレシーバ・パーティションにより必要とされる量のＣＰＵリソースを含む。図５は「アセスメント・リクエスト」メッセージのためのメッセージ・フローと夫々のＷＬＭコンポーネント１０２からの応答とを示す。
【００４４】
図６は．アセスメント・リクエスト・メッセージを受取るパーティションのＷＬＭコンポーネント１０２中の処理を示す。
【００４５】
ステップ６０１で、アセスメント・リクエスト・メッセージを受取る。ステップ６０２で、そのメッセージでリクエストされる量のＣＰＵリソースをローカル・パーティションが譲渡するのを許容されるか調べるためにテストが行われる。パーティションの定義の部分はそのパーティションのためのＣＰＵリソースの最小限許容される量を含んでもよい。「アセスメント・リクエスト」メッセージに含まれるＣＰＵリソースの量を譲渡する行為がそのような最小量を下回るローカル・パーティションを取るならば、そのパーティションはリクエストされたリソースを譲渡することは許されない。この場合ステップ６０８で、そのパーティションがそのリクエストされたＣＰＵリソースを提供できないことを示す「アセスメント応答」メッセージが送られる。
【００４６】
もしもパーティションがリクエストされたＣＰＵリソースを部分的に譲渡することができるなら、ステップ６０３に処理が続く。そこではそのリクエストされたＣＰＵリソースを譲渡するローカル・パーティションによって生じさせられるであろうサービス・クラスごとのパフォーマンス・インパクトから投影が構成される。このパフォーマンス投影は前述のように、サービス・クラスごとに投影されたパフォーマンス・インデックスの形をとる。ステップ６０４では、いずれかのサービス・クラスがその目標をミスするように投影されるか調べるテストが行われる。もしそれらの目標をミスするように投影されるサービス

・クラスがあるなら、ステップ６０６で、それらの目標が選択されることをミスしている最高の重要度レベルのサービス・クラスの組が選択される。ステップ６０７では、選択されたサービス・クラス、それらの現在パフォーマンス・インデックスおよびそれらの投影されたパフォーマンス・インデックスからなるこのリストを含むように「アセスメント応答」メッセージが打ち立てられる。このメッセージはドメイン・マネジャ１０３に送られる。

【００４７】

もしその目標をミスするようにサービス・クラスが投影されないなら、ステップ６０５では、全てのサービス・クラス、それらの現在のパフォーマンス・インデックス、それらの投影されたパフォーマンス・インデックスを含むように「アセスメント応答」メッセージが打ち立てられる。この「アセスメント応答」メッセージがドメイン・マネジャ１０３に送られる。

【００４８】

図４に示す処理に戻ると、アセスメント分析段階では、ステップ４０４でドメイン・マネジャ１０３のＶＧサブコンポーネント１０５がそのグループ中のそのレシーバ・パーティションを除く各パーティションに「アセスメント・リクエスト」を送った。

【００４９】

全ての「アセスメント応答」を受取ってしまうと、ステップ４０５でいずれかのパーティションがリクエストされたＣＰＵリソースを提供することができるかを調べるためのテストが行われる。このテストを実施するために、潜在的ドナー・パーティション（ＣＰＵリソースを提供できる可能性のあるパーティション）のリストが作成される。まず、このリストはレシーバ・パーティションを除くそのグループ中全てのパーティションを含む。次に、リクエストされたＣＰＵリソースを提供することができないことを示す「アセスメント応答」メッセージをもつパーティションをそのリストから除外する。もしそのリストにまだパーティションが残っていれば、その潜在的ドナーの各々がレシーバ・サービス・クラスに比したそのサービス・クラスのパフォーマンス・インデックスの変化とその相対的な重要度とに基づいて評価される。例えば、レシーバ・サービス・クラスよりも重要な潜在的ドナーのサービス・クラスが目標をミスするように投影されるなら、そのドナー・パーティションを使用すべきではない。その完全な評価を以下の表１に擬似コードで記述する。

【表１】

```
1   receiver_change_in_pi = receiver_projected_pi - receiver_current_pi
2   do for each service class in donor response
3   donor_change_in_pi = donor_projected_pi - donor_current_pi
4   if donor service class projected to missing goals
5   if donor service class more important than receiver service class
6   Indicate donor partition cannot be used
7   Exit
8   end if
9   if receiver service class meeting goals
10  Indicate donor partition cannot be used
11  Exit
12  end if
13  if donor service class importance equal to receiver service class importance
14  if donor_change_in_pi > receiver_change_in_pi
15  Indicate donor partition cannot be used
16  Exit
17  end if
18  If (donor_projected_pi - receiver_projected_pi) > (donor_current_pi - receiver_current_pi)
19  /* The above test to see if the donor and receiver PIs are further apart after the change*/
20  Indicate donor partition cannot be used
21  Exit
22  end if
23  end if
24  Else /* donor service class projected to meet goals */
25  if receiver service class projected to meet goals
26  If donor_change_in_pi > receiver_change_in_pi
27  Indicate donor partition cannot be used
28  Exit
29  end if
30  end if
31  end do for each donor service class
32  Indicate donor partition can be used
33  Exit
```

【００５０】

この評価がドナーとして使用できないことを示す各パーティションは潜在的ドナー・リストから除外される。もしそのリストに潜在的ドナーが残っていなければ、ステップ４０６で、選択されたレシーバ・サービス・クラスを助けるドネーション（ＣＰＵパーティションの提供）をこの時点でなし得ないことレシーバ・パーティションに知らせるようにそれにメッセージ（図４の失敗メッセージ）が送られる。このレシーバ・サービス・クラスは所定の期間、この実施例では３分間はレシーバとして選ばれる適格性がないであろう。この不適格な期間というのは、その同じレシーバ・サービス・クラスが、そのヘルプできない全ての期間に選択されるのを回避する。

【００５１】

もしその潜在的ドナー・パーティション・リストが空でなければ、処理がステップ４０８に進み、そこで最良のドナー・パーティションが選択される。潜在的ドナー・パーティション・リスト上に、目標をミスするように投影されるサービス・クラスがないパーティションがあるなら、それが最良のドナーとして選択される。もし潜在的ドナー・リスト上の全てのパーティションが目標をミスするサービス・クラスを有するなら、目標をミスするサービス・クラスが最低の重要度レベルにあるパーティションが最良のドナーとして選択される。

【００５２】

ステップ４０９では、「構成変更リクエスト」メッセージがステップ４０８で選択されたドナー・パーティションに送られる。このメッセージはそのドナー・パーティションが譲渡すべきＣＰＵ・リソースの量を含む。図７は、「構成変更リクエスト」メッセージのためのメッセージの流れを示す。ドメイン・マネジャ１０３のＶＭサブコンポーネント１０４はそのドナー・パーティション上のＷＬＭコンポーネント１０２にメッセージを送る。このメッセージはＶＡコンポーネント１０７によって処理される。このメッセージの処理について図８に示す。

【００５３】

ステップ４１０では「構成変更応答」が処理される。もしドナー・パーティションがそのリクエストされたＣＰＵリソースを譲渡することができたことをその応答が示すなら、レシーバ・パーティションはそのドナー・パーティションが譲渡したＣＰＵリソースを得るように通知されることができる。これはステップ４１１で為されるが、それはドナー・パーティションが譲渡した量だけそのＣＰＵリソースを増やすように「構成変更リクエスト」メッセージをレシーバ・パーティションに送ることによって為される。「構成変更リクエスト」メッセージの処理については図８に示す。レシーバ・パーティションがその「構成変更リクエスト」メッセージを一旦処理すると、そのレシーバ・パーティションからの元の「ヘルプの申し立て」メッセージの処理が成功裡に完了し、そしてそのリクエストされたリソースがドナー・パーティションからレシーバ・パーティションに移されたことになる。

【００５４】

図８は、管理されているパーティション上のＷＬＭコンポーネント１０２により「構成変更リクエスト」メッセージの処理を示す。この「構成変更リクエスト」メッセージは、そのパーティションがドナー・パーティションである場合にＣＰＵリソースを譲渡するためのリクエストかまたはそのパーティションがレシーバ・パーティションである場合にもっと多くのＣＰＵリソースを得るためのリクエストかのいずれであってもよい。ステップ８０１では、そのメッセージをＶＡコンポーネント１０７が受取る。そのリクエストを処理するには、そのパーティションに割当てられたＣＰＵリソースにそのリクエストされた変更を生じさせるようにハイパーバイザ１０１によって提供されるＡＰＩをＶＡコンポーネント１０７が呼び出す。そのようなハイパーバイザＡＰＩの一例は、ＩＢＭ　ＡＩＸオペレーティング・システムによって提供される前述のlpar_set_resource　ＡＰＩである。このＡＰＩ呼び出しは、そのパーティションに割当てられたＣＰＵリソースを変更させようとする試みが成功であったか失敗であったかの表示を返す。その変更が成功だったならば、ステップ８０４で「構成変更応答」メッセージがドメイン・マネジャ１０３に送られて、そのリソース割当て変更が成功したことを示す。もしその変更が成功でなかったなら、ステップ８０５で「構成変更応答」メッセージがドメイン・マネジャ１０３に送られて、その応答割当て変更が失敗したことを示す。

【００５５】

特定の実施例を示し説明したが、種々の変更が当業者には明らかである。

【図面の簡単な説明】

【００５６】

【図１】本発明を組み込んだデータ処理システムを示す図である。

【図２】各パーティションが管理されているワークロード・マネジャ（ＷＬＭ）コンポーネント中の処理を示す図である。

【図３】管理されているパーティション上のＷＬＭコンポーネントとドメイン・マネジャとの間での「ヘルプの申し立て」メッセージのデータ・フローを示す図である。

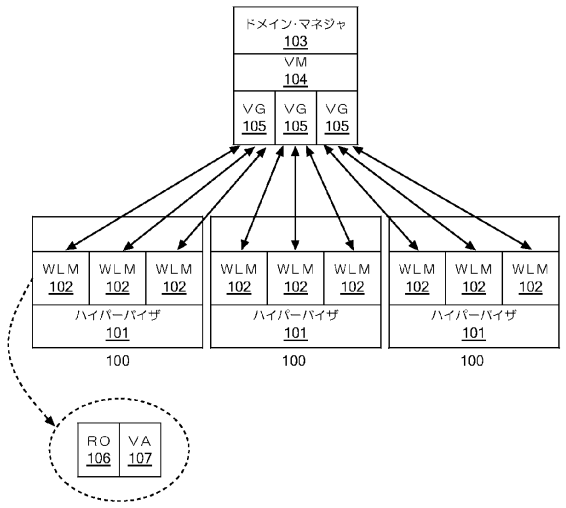
【図４】ドメイン・マネジャが「ヘルプの申し立て」を受けるときの処理を示す図である。

【図５】「アセスメント・リクエスト」メッセージおよび各ＷＬＭコンポーネントからの応答のためのメッセージ・フローを示す図である。

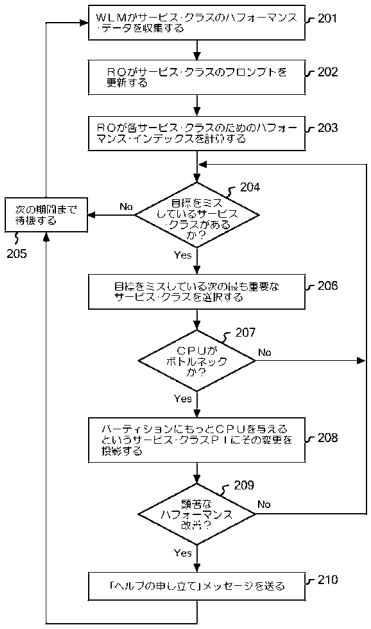
【図６】アセスメント・リクエスト・メッセージを受け取るパーティションのＷＬＭコン

ポーネント内の処理を示す図である。
【図７】「構成変更リクエスト」メッセージのためのメッセージ・フローを示す図である。
【図８】ＶＡサブコンポーネントによる「構成変更リクエスト」メッセージの処理を示す図である。
【図９】パーティションに割当てられたＣＰＵリソース（資源）中の変更のパフォーマンス・インパクトを投影するのに使用されるグラフを示す図である。
【符号の説明】
【００５７】
１００　コンピュータ・システム
１０１　ハイパーバイザ
１０２　ＷＬＭ（ワークロード管理）コンポーネント
１０３　ドメイン・マネジャ
１０４　ＶＭ（抽象化マネジャ）サブコンポーネント
１０５　ＶＧ（抽象化グループ）サブコンポーネント
１０６　ＲＯ（リソース・オプティマイザ）コンポーネント
１０７　ＶＡ（抽象化エージェント）コンポーネント

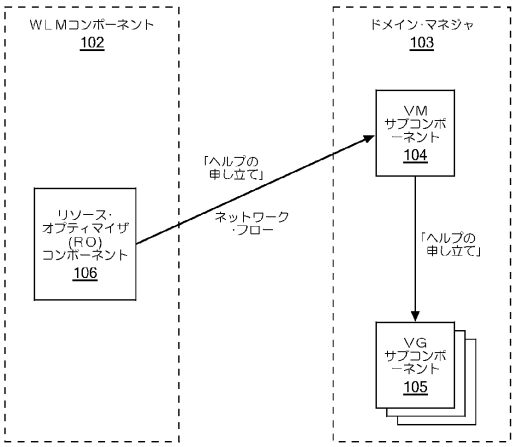
【図１】



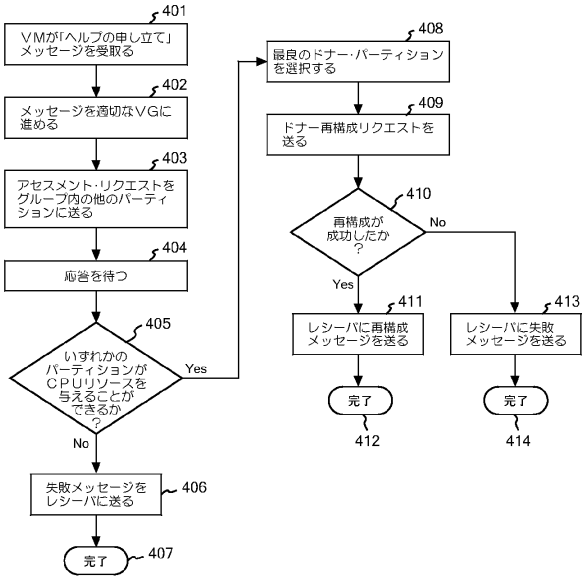
【図２】



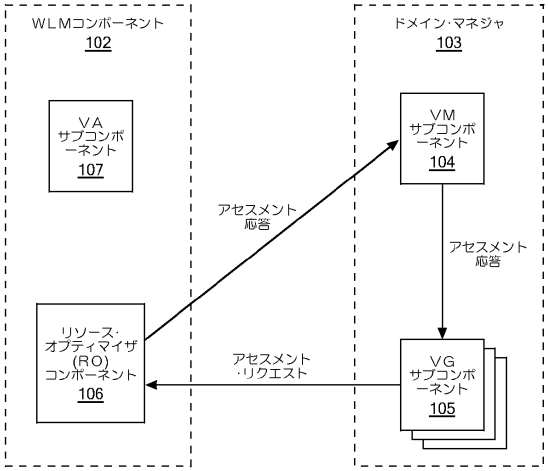
【図３】


WLMコンポーネント
102
ドメイン・マネジャ
103
VM
サブコンポ
ーネント
104
「ヘルプの
申し立て」
ネットワーク
・フロー
リソース・
オプティマイザ
(RO)
コンポーネント
106
「ヘルプの
申し立て」
VG
サブコンポ
ーネント
105

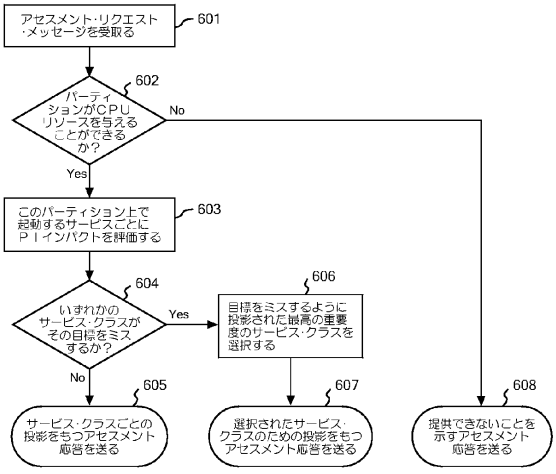
【図４】


401
VMが「ヘルプの申し立て」
メッセージを受取る
402
メッセージを適切なVGに
進める
403
アセスメント・リクエストを
グループ内の他のパーティ
ションに送る
404
応答を待つ
405
いずれかの
パーティションが
CPUリソースを
与えることが
できるか
?
Yes
No
406
失敗メッセージを
レシーバに送る
407
完了
408
最良のドナー・パーティション
を選択する
409
ドナー再構成リクエストを
送る
410
再構成が
成功したか
?
No
Yes
411
レシーバに再構成
メッセージを送る
412
完了
413
レシーバに失敗
メッセージを送る
414
完了

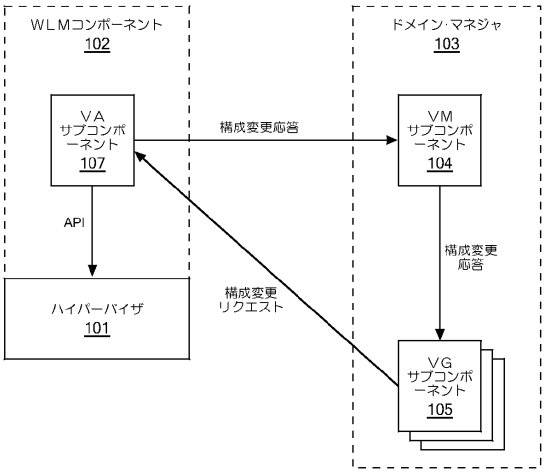
【図５】


WLMコンポーネント
102
ドメイン・マネジャ
103
VA
サブコンポ
ーネント
107
VM
サブコンポ
ーネント
104
アセスメント
応答
アセスメント
応答
リソース・
オプティマイザ
(RO)
コンポーネント
106
アセスメント
・リクエスト
VG
サブコンポ
ーネント
105

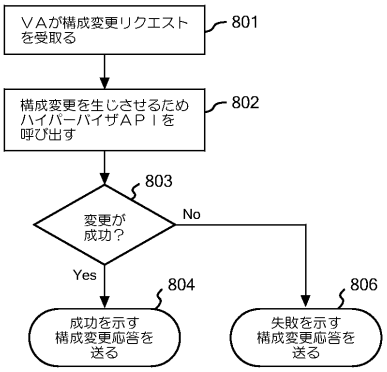
【図６】


アセスメント・リクエスト
・メッセージを受取る
601
パーティ
ションがCPU
リソースを与える
ことができる
か?
602
No
Yes
このパーティション上で
起動するサービスごとに
PIインパクトを評価する
603
いずれかの
サービス・クラスが
その目標をミス
するか?
604
Yes
No
目標をミスするように
投影された最高の重要
度のサービス・クラスを
選択する
606
サービス・クラスごとの
投影をもつアセスメント
応答を送る
605
選択されたサービス・
クラスのための投影をもつ
アセスメント応答を送る
607
提供できないことを
示すアセスメント
応答を送る
608

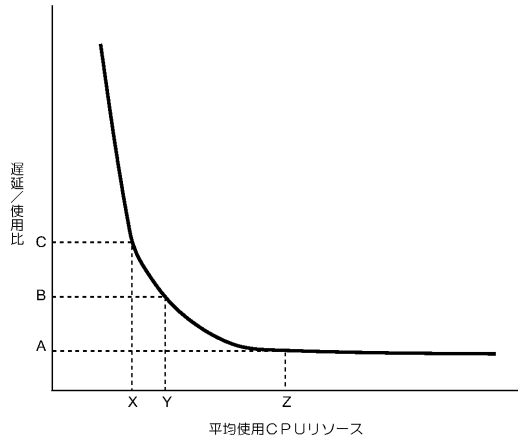
【図７】


WLMコンポーネント
102
ドメイン・マネジャ
103
VA
サブコンポーネント
107
構成変更応答
VM
サブコンポーネント
104
API
構成変更
応答
ハイパーバイザ
101
構成変更
リクエスト
VG
サブコンポーネント
105

【図８】


VAが構成変更リクエスト
を受取る
801
構成変更を生じさせるため
ハイパーバイザAPIを
呼び出す
802
803
変更が
成功？
No
Yes
804
806
成功を示す
構成変更応答を
送る
失敗を示す
構成変更応答を
送る

【図９】


遅延／使用比
C
B
A
X
Y
Z
平均使用CPUリソース

フロントページの続き

(74)代理人 100086243
弁理士 坂口 博
(72)発明者 ピーター・ビー・ヨーコム
アメリカ合衆国 １２５４０ ニューヨーク州ラグランゲビル、ストリングハム ロード １２９ ユニット＃２９
(72)発明者 ジェフェリー・ディー・アマン
アメリカ合衆国 １２６０３ ニューヨーク州ポキプシー、プレストウイック・コート ４０
(72)発明者 ヒレン・アール・シャー
アメリカ合衆国 １２５２８ ニューヨーク州ハイランド ナイン・ハドソン・レーン
(72)発明者 ユクセル・グナル
アメリカ合衆国 １０１２８ ニューヨーク州ニューヨーク、イースト・ナインティシクスス・ストリート ５３、アパートメント＃４Ａ
(72)発明者 ユック・エル・チャン
アメリカ合衆国 １２６０３ ニューヨーク州ポキプシ、ジャックマン・ドライブ ６９Ｂ
(72)発明者 マシュー・エス・ソーンズ
アメリカ合衆国 １０６０４ ニューヨーク州ウエスト・ハリソン、ウッドサイド・アベニュー １８３

審査官 井出 和水

(56)参考文献 特表２００３－５１４２７０（ＪＰ，Ａ）
特開２００４－３２６７７４（ＪＰ，Ａ）
特開平０５－０１２２２４（ＪＰ，Ａ）
特開平０７－２８１９０８（ＪＰ，Ａ）
特開２００６－０８５７１２（ＪＰ，Ａ）
特開２００４－２４０９７０（ＪＰ，Ａ）
特開２０００－１３２５３０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ ９／４６ － Ｇ０６Ｆ ９／５０