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

特許第4185103号
(P4185103)

(45)発行日　平成20年11月26日(2008.11.26)　(24)登録日　平成20年9月12日(2008.9.12)

(51)Int.Cl.　　F I
G06F 9/50 (2006.01)　　G06F 9/46 465Z
G06F 9/46 (2006.01)　　G06F 9/46 350

請求項の数 8　(全 12 頁)

(21)出願番号　特願2006-39467(P2006-39467)
(22)出願日　平成18年2月16日(2006.2.16)
(65)公開番号　特開2006-244479(P2006-244479A)
(43)公開日　平成18年9月14日(2006.9.14)
審査請求日　平成18年2月16日(2006.2.16)
(31)優先権主張番号　11/067852
(32)優先日　平成17年2月28日(2005.2.28)
(33)優先権主張国　米国(US)

(73)特許権者　503003854
ヒューレット－パッカード　デベロップメント　カンパニー　エル．ピー．
アメリカ合衆国　テキサス州　７７０７０　ヒューストン　２０５５５　ステイト　ハイウェイ　２４９
(74)代理人　110000039
特許業務法人アイ・ピー・エス
(72)発明者　スコット・エイ・ライン
アメリカ合衆国カリフォルニア州　パロアルト　ハノーバー・ストリート　３０００　ヒューレット・パッカード・カンパニー内

審査官　鈴木　修治

最終頁に続く

(54)【発明の名称】実行可能プログラムをスケジューリングするためのシステム及び方法

(57)【特許請求の範囲】
【請求項１】
複数のプロセッサ（１３１）と、
実行可能プログラムの複数のグループ（１４３）であって、各グループ（１４３）について、前記グループ（１４３）の実行可能プログラムをサポートするプロセッサ資源の量を表す各シェアパラメータ（１２２）が定義される、実行可能プログラムの複数のグループ（１４３）と、
前記シェアパラメータ（１２２）を使用して複数の重みを配分し、前記複数のプロセッサ（１３１）間における前記重みの配分を生成するソフトウェアルーチン（１２４）であって、前記配分が、各グループ（１４３）についてのプロセッサのサブセットと、前記グループ（１４３）の実行可能プログラムとをスケジューリングするための、前記サブセット内の各プロセッサの比率を定義するソフトウェアルーチン（１２４）と、
前記配分に従ったスケジューリング間隔の期間中に、前記複数のグループ（１４３）の各実行可能プログラムを前記複数のプロセッサ（１３１）の特定のプロセッサにスケジューリングするためのスケジューリングソフトウェアルーチン（１２５）と
を備え、
前記ソフトウェアルーチン（１２４）は、
複数の整数区画問題（ＩＰＰ；integer partition problem）アルゴリズムを使用して、前記複数のグループ（１４３）の少なくとも１つに割り当てられるシェアパラメータ（１２２）から生成された重みの個数を変化させることによって複数の配分を生成し、生成

された前記複数の配分をスケジューリングに使用する
コンピュータシステム。
【請求項２】
前記可変の個数の重みは、
デフォルトのグループに割り当てられるシェアパラメータ（１２２）から生成される
請求項１に記載のコンピュータシステム。
【請求項３】
前記スケジューリングソフトウェアルーチン（１２５）は、
前記複数のグループ（１４３）の間のスケジューリングの相違を補償するように前記複数の配分を交互に行う
請求項１に記載のコンピュータシステム。
【請求項４】
前記複数のグループ（１４３）の実行可能プログラムのスケジューリングパラメータを保持するためのソフトウェアルーチンであって、各スケジューリングパラメータが、グループの平均に関連した（relate）、各実行可能プログラムが受け取るプロセッサ資源の量を示すソフトウェアルーチン
をさらに備える
請求項１に記載のコンピュータシステム。
【請求項５】
前記スケジューリングソフトウェアルーチン（１２５）は、
前記スケジューリングパラメータの値に従って、前記複数のグループ（１４３）の実行可能プログラムのサブセットを、割り当て期間内にプロセッサ資源を受け取る付加的な機会を実行可能プログラムの前記サブセットに提供する１つ又は数個のプロセッサに割り当てる
請求項４に記載のコンピュータシステム。
【請求項６】
複数のグループの実行可能プログラムをスケジューリングするためのプロセッサ資源の量を表す複数のシェアパラメータを定義すること（２０２）と、
前記複数のシェアパラメータを使用して、整数区画問題（ＩＰＰ；integer partition problem）に従い複数の重みを生成すること（２０４）と、
複数のＩＰＰアルゴリズムを使用して、前記重みの個数を変化させることによって、複数のプロセッサ間における前記重みの配分を複数決定すること（２０６～２１０）と、
前記複数の配分を使用して、前記複数のプロセッサにグループの実行可能プログラムをスケジューリングすること（３０４）と
を含む方法。
【請求項７】
前記複数のグループの実行可能プログラムのスケジューリングパラメータを保持することであって、各スケジューリングパラメータが、グループの平均に関連した、各実行可能プログラムが受け取るプロセッサ資源の量を示すこと
をさらに含む請求項６に記載の方法。
【請求項８】
前記スケジューリングすることは、
１つ又は数個のプロセッサについて、前記スケジューリングパラメータの値に従って実行可能プログラムを選択することであって、前記１つ又は数個のプロセッサが、前記選択された実行可能プログラムに、スケジューリング間隔内でプロセッサ資源を受け取る付加的な機会を提供すること
を含む請求項７に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】

本出願は、包括的には、コンピュータ資源に対するアクセスのスケジューリングに関する。

【背景技術】

【０００２】

多くの企業は、その企業の組織内で使用されるコンピュータ及びアプリケーションの数の劇的な増加を経験してきた。企業の事業グループが新たなアプリケーションを配備する時、その新たなアプリケーションをホストする１つ又は複数の専用サーバプラットフォームを追加することが可能である。このタイプの環境は、時に、「ｏｎｅ－ａｐｐ－ｐｅｒ－ｂｏｘ（ボックスごとに１つのアプリケーション）」と呼ばれる。より多くの事業プロセスがデジタル化されるにつれて、「ｏｎｅ－ａｐｐ－ｐｅｒ－ｂｏｘ」環境では、サーバプラットフォームの個数が過度なものになる。その結果、サーバプラットフォームの管理運営コストは大幅に増加する。その上、サーバプラットフォームの資源が実際に使用される時間のパーセンテージ（利用率）は、かなり低くなる可能性がある。これらの問題に対処するために、多くの企業は、複数のアプリケーションを共通のサーバプラットフォームに一元化して、プラットフォームの個数を削減し、システムの利用率を向上させてきた。このような一元化が行われた場合、アプリケーション及び他の実行可能プログラムがいつプロセッサ資源にアクセスするかを決定するための或る機能が通常設けられる。このような機能は、通常、「スケジューリング」と呼ばれる。

【０００３】

さまざまな複雑さを有するいくつかのスケジューリングアルゴリズムが存在する。おそらく、最も簡単なスケジューリングは、先着順アルゴリズムである。優先順位に基づくアルゴリズムは、優先順位をプロセスに割り当て、最も高い優先順位を有するプロセスが、適切な時刻に選択されて実行される。プリエンプティブスケジューリングアルゴリズムは、高い優先順位のプロセスの実行の準備ができると、低い優先順位のプロセスをプロセッサから移す（remove）のに使用することができる。ラウンドロビンスケジューリングアルゴリズムは、プロセスが或る時間間隔の満了まで実行することを可能にし、次いで、別の実行可能プログラムが選択されて、各プロセッサで実行される。これに加えて、フェアシェアスケジューラは、パーセント又はシェアを定義し、定義したシェアに比例して、プロセッサ資源にアクセスする機会をプロセスに提供する。

【発明の開示】

【課題を解決するための手段】

【０００４】

一実施の形態では、コンピュータシステムは、複数のプロセッサと、実行可能プログラムの複数のグループであって、各グループについて、そのグループの実行可能プログラムをサポートするプロセッサ資源の量を表す各シェアパラメータが定義される、実行可能プログラムの複数のグループと、シェアパラメータを使用して複数の重みを生成し、且つ、複数のプロセッサ間における重みの配分を生成する、ソフトウェアルーチンであって、この配分が、各グループについてのプロセッサのサブセット、及び、そのグループの実行可能プログラムをスケジューリングするための、サブセット内の各プロセッサの比率を定義する、ソフトウェアルーチンと、配分に従ったスケジューリング間隔の期間中に、複数のグループの各実行可能プログラムを複数のプロセッサの特定のプロセッサにスケジューリングするためのスケジューリングソフトウェアルーチンと、を備える。

【０００５】

別の実施の形態では、方法は、複数のグループの実行可能プログラムをスケジューリングするためのプロセッサ資源の量を表す複数のシェアパラメータを定義すること、これらの複数のシェアパラメータを使用して、整数区画問題（ＩＰＰ）に従い複数の重みを生成すること、ＩＰＰアルゴリズムを使用して、複数のプロセッサ間における重みの配分を決定すること、及び、この配分を使用して、複数のプロセッサにグループの実行可能プログラムをスケジューリングすること、を含む。

【０００６】

別の実施の形態では、コンピュータシステムは、複数の資源デバイスと、複数の実行可能プログラムの複数のグループであって、各グループについて、そのグループの実行可能プログラムをサポートする複数の資源デバイスへのアクセス量を表す各シェアパラメータが定義される、複数の実行可能プログラムの複数のグループと、シェアパラメータを使用して複数の重みを生成し、且つ、複数の資源デバイス間における重みの配分を生成する、ソフトウェアルーチンであって、この配分が、各グループについての資源デバイスのサブセット、及び、そのグループの実行可能プログラムをスケジューリングするための、サブセット内の各資源デバイスの比率を定義する、ソフトウェアルーチンと、配分に従って、複数の資源デバイスの特定の資源デバイスに複数のグループの各実行可能プログラムをスケジューリングするためのスケジューリングソフトウェアルーチンと、を備える。

【０００７】

別の実施の形態では、コンピュータシステムは、整数区画問題（ＩＰＰ）アルゴリズムを使用して、コンピュータシステムの複数の資源デバイス間における重みの配分を生成するための手段であって、これらの重みが、それぞれが実行可能プログラムの各グループに提供される複数の資源デバイスへのアクセス量を表す複数のシェアパラメータから生成され、この配分が、各グループについての資源デバイスのサブセット、及び、そのグループの実行可能プログラムをスケジューリングするための、サブセット内の各資源デバイスの比率を定義する、生成するための手段と、この配分に従って、グループの各実行可能プログラムを資源デバイスにスケジューリングするための手段と、を備える。

【発明を実施するための最良の形態】

【０００８】

いくつかの代表的な実施の形態は、整数区画問題（ＩＰＰ）アルゴリズムを使用して、シェアに基づくワークロードグループのオペレーションのスケジューリングを実行する。各グループには、そのグループに割り当てられたシステム資源の「シェア」を表すパラメータの値が与えられる。ソフトウェアモジュールは、ＩＰＰアルゴリズムを使用して各グループを１つ又は数個のプロセッサにマッピングする。具体的には、グループのシェアは「重み」に区分（separate）され、これらの重みは、各プロセッサに関連した重みがほぼ等しくなるようにプロセッサ（「ビン（bin）」）に配分される。

【０００９】

シェアの重みへの区分は、ワークロードのいくつかをサポートするのに使用される複数の「仮想プロセッサ」を考慮することができる。たとえば、或るグループに４つの仮想ＣＰＵが割り当てられ、各仮想ＣＰＵが物理ＣＰＵの約７５パーセントの能力を有する場合、そのグループは、それぞれが７５の４つの個々の重みを生成する。各ＣＰＵを１００よりも多くのシェア又は少ないシェアでスケジューリングすることができるので、重みは、資源のパーセンテージに正確に対応するものではない。特定のＣＰＵの実際のスケジューリングパーセンテージは、そのＣＰＵで現在実行されているすべてのジョブの全重みを使用して求められる。

【００１０】

また、デフォルトのグループのシェアパラメータ又は最も低い優先順位のグループのシェアパラメータを複数の重みに区分することを可変に行って、プロセッサ間における重みの最適な配分を達成する確率を改善することができる。このデフォルトのグループは、具体的な重み又は優先順位を有しないすべての資源要求を保持するのに使用することができる。１つのインプリメンテーションでは、デフォルトのグループのすべてのメンバーは、他のグループにまだ割り当てられていない資源を均等に分け合う。

【００１１】

ＩＰＰアルゴリズムによって生成された配分は、各グループの物理ＣＰＵと、各グループがスケジューリング間隔内で受け取るそれらＣＰＵの比率とのリストを提供する。これに加えて、各ジョブが受け取るプロセッサ時間の量が、ジョブスケジューリングパラメータを使用して追跡される。時間サンプリング間隔でより多くのプロセッサチックを累積したジョブは、自身のパラメータを減らす。平均のプロセッサチックよりも少ないプロセッ

サチックを累積したジョブは、自身のパラメータをインクリメントする。新たな各スケジューリング間隔時には、最も高いパラメータの値を有するジョブが、より多くのプロセッサチックをこれらのジョブに提供する利用可能な物理ＣＰＵ（すなわち、全スケジューリング重みが最も低いＣＰＵ（複数可））に対して選択される。また、２つのＣＰＵのスケジューリング重みが等しい場合、履歴に基づく最も低い使用量が、より良いＣＰＵを選択するのに使用される。

【００１２】

次に図面を参照して、図１は、代表的な一実施の形態によるシステム１００を示している。このシステム１００は、プラットフォームのハードウェアレイヤ１３０に対する低レベルのアクセスを制御するホストオペレーティングシステム１２０を含む。一実施の形態では、ホストオペレーティングシステム１２０は、一例としてそのカーネル内に仮想化レイヤ１２１を含む。仮想化レイヤ１２１は、ハードウェアレイヤ１３０の物理資源に対応するソフトウェア構成（software constructs）（論理デバイス）を作成する。ハードウェアレイヤ１３０は、ＣＰＵ１３１－１から１３１－Ｎ、メモリ１３２、ネットワークインターフェース１３３、入出力（Ｉ／Ｏ）インターフェース１３４等の任意の個数の物理資源を含むことができる。

【００１３】

一実施の形態では、仮想資源（たとえば、１又は数個の仮想ＣＰＵ、仮想メモリ、仮想ネットワークインターフェースカード、仮想Ｉ／Ｏインターフェース等）が、各仮想マシン１４１に割り当てられる。仮想ＣＰＵの個数は、物理ＣＰＵ１３１の個数を超えてもよい。各仮想マシン１４１は、自身に割り当てられた仮想資源に従ってオペレーティングシステム１２０上でプロセスとして実行される。ＣＰＵの仮想化は、各仮想マシン１４１がそれ自身のＣＰＵ又はそれ自身の１組のＣＰＵで実行するように見えるよう行うことができる。ＣＰＵの仮想化は、各仮想ＣＰＵについて、１組のレジスタ、変換索引バッファ、及び他の制御構造体を設けることによって実施することができる。したがって、各仮想マシン１４１は、他の仮想マシン１４１から隔離される。これに加えて、各仮想マシン１４１は、各ゲストオペレーティングシステム１４２を実行するのに使用される。仮想マシン１４１に割り当てられた仮想資源は、ゲストオペレーティングシステム１４２には、物理サーバのハードウェア資源として見える。次に、ゲストオペレーティングシステム１４２は、１つ又は数個のアプリケーション１４３を実行するのに使用することができる。

【００１４】

スケジューリングルーチン１２５は、仮想マシン１４１に関連したどの実行可能なスレッドが各プロセッサ１３１で実行されるかを決定する。実行可能なスレッドには、保証された時間の比率で、各プロセッサ１３１で実行される機会が与えられる。この比率は、所与のスケジューリング間隔については、部分的に、実行可能なスレッドのグループに従って定義される。たとえば、各仮想マシン１４１をグループに割り当てることができ、シェア１２２は、さまざまなグループについて定義される。各シェアパラメータは、各グループの仮想マシン１４１が平均して受け取るべきプロセッサ「チック」の最小量を表す。

【００１５】

シェアは、仮想マシングループの現在の要求と組み合わせられて、重み付き資源要求に変換される。ＩＰＰアルゴリズム１２４は、これらの重みを使用して、各グループを１組の物理ＣＰＵにマッピングする。このマッピングは、（要素１２３に記憶された）配分と呼ばれる。各グループについて、このマッピングは、ＣＰＵ、いくつのスレッドが各ＣＰＵで実行されるか、及び、時間の比率は何であるかのリストを含む。ＩＰＰアルゴリズム１２４によって生成された配分は、各ＣＰＵがサービス提供する全重みが可能な限り均一となるようにする。

【００１６】

所与のスケジューリング間隔内において、スケジューリングルーチン１２５は、各配分１２３及びスケジューリングパラメータ１２６を使用して、各グループ内のどの実行可能プログラムが各プロセッサ１３１で実行されるかを決定する。前述したように、選択され

た配分１２３は、各グループに利用可能な物理ＣＰＵを定義する。スケジューリングルーチン１２５は、スケジューリングパラメータ１２６を使用して、各グループからのどの特定のスレッドが、この間隔の間、そのリストのどのＣＰＵで実行されるかを決定する。スケジューリングパラメータ１２６は、さまざまな実行可能プログラムが受け取ったプロセッサチックの受け取り履歴を示す。最も高いパラメータの値を有する実行可能プログラムが、最良の利用可能な物理ＣＰＵに対して選択される。スケジューリング間隔の完了時に、累積したプロセッサチックが平均プロセッサチックよりも少ない実行可能プログラムは、自身のパラメータをインクリメントし、平均よりも多くのプロセッサチックを累積した実行可能プログラムは、自身のパラメータを減らす。

【００１７】

仮想プロセッサに関連したマッピング及びスケジューリングが考察されたが、他の代表的な実施の形態は、あらゆる適切なマルチプロセッサコンピュータシステムのあらゆるタイプの実行可能プログラムをスケジューリングするのに使用することができる。これに加えて、マッピング及びスケジューリングは、コンピュータのタイムスライスされたあらゆるタイプの資源（たとえば、ネットワーク接続カード、ディスクＩ／Ｏチャネル、暗号化デバイス等）について行うことができる。

【００１８】

図２は、代表的な一実施の形態による、ソフトウェアジョブのグループのプロセッサへのマッピングを生成するためのフローチャートを示している。図２は、適切なコンピュータ可読媒体から取り出されたソフトウェアコード又はソフトウェア命令を使用して実施される。ステップ２０１では、複数のジョブをサポートする複数のグループが定義される。一実施の形態では、各ジョブは、各仮想マシンによってサポートされる。各仮想マシンは、１つ又は数個の仮想プロセッサを備える。ステップ２０２では、それらグループのシェアが定義される。これらのシェアは、各グループが平均して受け取る機会を有するプロセッサ資源の量を定義する。一実施の形態では、シェアは、１００チックが１秒の時間内における単一の物理プロセッサの全能力を表す、プロセッサ「チック」を符号化する。いくつかの実施の形態では、他のグループに明示的に割り当てられていないチックのすべてを受け取る最も優先順位の低いグループ又はデフォルトのグループが定義される。たとえば、或るコンピュータシステムが６つのジョブ（Ａ、Ｂ、Ｃ、Ｄ、及びＥ）をサポートし、２つのプロセッサ（２００の全シェアが利用可能である）を有するものと仮定する。ジョブＡは、「高い」優先順位のグループに割り当てられ、８０のシェアを受け取る。ジョブＢは、「中程度」の優先順位のグループに割り当てられ、４５のシェアを受け取る。ジョブＣ、Ｄ、及びＥは、デフォルトのグループに割り当てられ、このデフォルトのグループには、残りの７５のシェアが割り当てられ、各グループは、平均して約２５のシェアを受け取る。実行可能プログラムをグループに割り当てることによって、整数区画問題の組み合わせ複雑度が明らかに削減される。

【００１９】

ステップ２０３では、変数（Ｎ）が、（ｉ）スケジューリングの目的のためにコンピュータシステムで利用可能な物理プロセッサの個数、及び、（ｉｉ）デフォルトのグループのアクティブなジョブの個数、のうちの最小値に等しくなるように設定される。

【００２０】

ステップ２０４では、グループのシェアパラメータが、異なる重みに区分される。いくつかの実施の形態では、デフォルトのグループのシェアパラメータは、Ｎ個の異なる等しい重み（又は、丸め誤差を考慮してほぼ等しい重み）に分割される。前の２つのプロセッサの例を使用すると、最初の繰り返しにおいて、デフォルトのグループのシェアパラメータ（７５）は、３７の第１の重み及び３８の第２の重みに分割することができる。いくつかの実施の形態では、グループのシェアは、これに加えて、マルチスレッド化されたジョブを考慮した異なる重みに区分される。たとえば、ジョブＡが、２つの仮想プロセッサを有する仮想マシンを使用して実施されるものと仮定する。高い優先度のグループの８０のシェアは、２つの仮想プロセッサのスレッドをサポートするために、４０の２つの重みに

分割することができる。或るグループ（デフォルトのグループ以外）がマルチスレッド化されたジョブを含まない場合、そのグループには、そのシェアパラメータと等しい単一の重みが生成される。

【００２１】

ステップ２０５では、プロセッサ間での重みの配分を制限するための制約（constraints）が定義される。これらの制約は、あらかじめ定義された１組のルール又は条件に従って自動的に生成することができる。たとえば、マルチスレッド化されたジョブについて複数の資源要求の重みが生成される場合、これらの重みが同じプロセッサに割り当てられることを防止するための制約が定義される。また、特定のシステムについて、たとえば、高可用性アプリケーションに使用される冗長なソフトウェアモジュールを分離する（separate）ための制約を手動で定義することもできる。

【００２２】

ステップ２０６では、ＩＰＰアルゴリズムが使用されて、プロセッサ間における重みの最適なバランスを達成するように、プロセッサのリスト全体にわたって重みの配分が生成される。この生成された配分は、後の解析（ステップ２１０を参照）のために一時的に記憶される。「貪欲（greedy）」方法等の既知のＩＰＰアルゴリズムを使用することができる。この貪欲方法では、すべての重みが割り当てられるまで、前に割り当てられた全重みが最も低い「ビン」に、残っている最も高い重みが割り当てられる。代替的に、「差分」方法を使用することもできる。この「差分」方法では、異なるサブセットの最大数を配置するとともに、それらの差分を新たな数として挿入することによって、割り当てが行われる。これらの数のすべてがこのようにして割り当てられた後、元の重みの配分が、後ろ向き再帰（backward recursion）によって求められる。ＩＰＰアルゴリズムのインプリメンテーションに関する詳細は、いくつかの出典から入手可能である。たとえば、ＩＰＰアルゴリズムの概要は、２００２年１２月１４日のHaikun Zhu著の論文「On the Integer Partitioning Problem: Examples, Intuition and Beyond」に与えられている。

【００２３】

いくつかの実施の形態によれば、解は、まず、高速の貪欲方法を使用して計算される。この解が完全でない場合、Ｎ次元差分方法が使用され、最も高い正確度を有する解が選択される。プロセッサビンへの重みの配分は、各グループに利用可能なＣＰＵの選択肢を決定する。ＣＰＵの全重みによって除算された、特定のジョブに関連した重みは、最終的に提供されるそのＣＰＵの部分を決定する。特定のスレッドのＣＰＵへの明示的なマッピングはこの段階では行われていないことに留意すべきである。その代わり、スレッドのグループのみが１組のＣＰＵにマッピングされている。これに加えて、個々の配分が生成された後、その配分が妥当であるかどうか（たとえば、制約が満たされるかどうか）を判断するための論理比較（フローチャートには図示せず）が行われる。配分が妥当でない場合、その特定の配分のさらなる使用を省くことができる。代替的に、プロセッサビンへの重みの割り当て中に、ＩＰＰアルゴリズムのビン割り当てロジックの変更によって、制約に対処することができる。

【００２４】

ステップ２０７では、生成された配分が完全である（たとえば、各プロセッサに、計画された作業の同じ全重みが割り当てられている）かどうかを判断するための論理比較が行われる。生成された配分が完全である場合、生成された配分は記憶され、その配分は、その後のスケジューリングオペレーションで利用可能になる（ステップ２１１）。また、前に計算された完全でない配分は、完全な配分の生成時に消去することができる。

【００２５】

生成された配分が完全でない場合、プロセスフローは、ステップ２０８に進み、ステップ２０８において、変数Ｎが１に等しいかどうかを判断するための別の論理比較が行われる。変数Ｎが１に等しくない場合、プロセスフローはステップ２０９に進み、ステップ２０９において、Ｎがデクリメントされて、プロセスフローはステップ２０４に戻る。したがって、デフォルトのグループに関連した重みの個数は変更され、それらの重みの値は変

更される。このように整数区画問題を変更して問題を再度解くことにより、配分の正確度を改善することができ、正確な配分を得る確率が増大する。

【００２６】

ステップ２０８の論理比較において、Ｎが１に等しい場合、プロセスフローはステップ２１０に進む。ステップ２１０では、記憶された配分が調べられて、Ｍ個の最良な配分（すなわち、各プロセッサに割り当てられた重み間の差分を最小にする配分）が特定される。ステップ２１１では、特定された配分が記憶されて、それらの配分がその後のスケジューリングオペレーションで利用可能になる。

【００２７】

いくつかの代表的な実施の形態では、図２のプロセスフローは、スケジューリングオペレーションの観点から、比較的低い頻度で実行される。具体的には、利用可能なプロセッサの個数が変化しない限り、又は、グループへのシェアの割り当てが変化しない限り、このプロセスフローの結果は変わらない。したがって、図２のプロセスフローは、大きなオーバーヘッドを課すことはなく、ワークロードの性能を低減させない。

【００２８】

図４は、代表的な一実施の形態による、図２のフローチャートに従って生成できる配分４００を示している。或るシステムが８つのジョブ（Ａ～Ｇ）をサポートするものと仮定する。このシステムは、３つのプロセッサを含み、したがって、３００のシェアが利用可能である（３＊１００）。また、ジョブＡが８０のシェア値を割り当てられ、２つの仮想プロセッサの仮想マシンに関連付けられるものと仮定する。さらに、ジョブＢ、Ｃ、及びＤがそれぞれ６０のシェア値を割り当てられるものと仮定する。ジョブＡ～Ｄは、単一のジョブグループ（Ｉ～ＩＶ）に割り当てられる。ジョブＥ～Ｇは、デフォルトのグループ（グループＶ）に割り当てられる。このデフォルトのグループは、４０のシェア値、すなわち、他のグループに割り当てられていないシェア量（３００－８０－６０－６０－６０）を受け取る。ジョブＡを含むグループＩのシェア値は、２つの仮想プロセッサをサポートする２つの重みに分割される。これらの重みが同じ物理プロセッサに割り当てられることを防止するための制約も定義される。

【００２９】

これらの重み及び制約についてのＩＰＰ解法プロセスは、図４に示すような配分４００という結果になる場合がある。グループＩの第１の重みはプロセッサ１に割り当てられ、グループＩの第２の重みはプロセッサ２に割り当てられる。グループＩＩの重み、グループＩＩＩの重み、及びグループＩＶの重みは、それぞれ、プロセッサ１、２、及び３に割り当てられる。この場合、グループＶのシェア値は、複数の重みに分割されず、グループＶの単一の重みがプロセッサＶに割り当てられる。次に、グループＡ～Ｇの実行可能プログラムのスケジューリングが、配分４００で特定されたプロセッサに行われる。

【００３０】

図３は、代表的な一実施の形態による、特定の物理ＣＰＵへの個々のジョブのスケジューリングを実行するためのフローチャートを示している。図３は、適切なコンピュータ可読媒体から取り出されたソフトウェアコード又はソフトウェア命令を使用して実施される。たとえば、システム割り込みに応答して呼び出された、オペレーティングシステムのスケジューリングソフトウェアルーチンを、図３のフローチャートを実施するのに使用することができる。

【００３１】

ステップ３０１では、ジョブスケジューリングパラメータが、ジョブによるプロセッサチックの受け取りに従って更新される。時間サンプリング間隔中に受け取るジョブがグループの平均よりも少ない場合、それらジョブのパラメータはインクリメントされる。受け取るジョブがグループの平均よりも少ない場合、それらジョブのパラメータはディクメントされる。アイドル状態にあるか、又は、要求の少ないジョブに関連したパラメータの値は、次第に０に減衰させることが可能な場合がある。

【００３２】

ステップ３０２では、グループの１つ又は複数の誤差が計算される（もしあれば）。ステップ３０３では、累積したあらゆるグループの誤差を補正する配分が選択される。具体的には、正確な配分が特定されなかったことから、複数の配分が生成されている場合、プロセスフローのさまざまな繰り返しにおいて、配分を交互に行う（alternation）ことができる。たとえば、配分Ａがグループ１に有利であり、配分Ｂがグループ２に有利である場合、それら２つの配分を交互に行うことによって、より正確な方法でジョブ間のスケジューリングを行うことが可能になる。正確な配分が特定された場合、その正確な配分が使用される。

【００３３】

ステップ３０４では、各グループのジョブが、選択された配分に従い、且つ、各ジョブスケジューリングパラメータを使用してスケジューリングされる。具体的には、各グループについて、そのグループのジョブが、それらジョブの各ジョブスケジューリングパラメータによって順序付けられる。配分によって定義されたそのグループのＣＰＵのリストが「望ましさ」によって順序付けられる。具体的には、全スケジューリング重みの低いＣＰＵほど、より大きな望ましさを所有する。その理由は、このようなＣＰＵの処理能力は、異なるグループの実行可能プログラム用に比較的大きなセグメント又は部分に分割されるからである。複数のＣＰＵの全スケジューリング重みが等しい場合、ＣＰＵの履歴に基づく使用量を使用して、相対的な望ましさを求めることができる。具体的には、或るＣＰＵが、履歴に基づく使用量が少ないことを示している場合、或るジョブが、当該ジョブにスケジューリングされた処理能力の部分を使用しない確率は高くなり、このような能力は別のジョブが使用することができる。

【００３４】

ＩＰＰアルゴリズムを使用して実行可能プログラムのグループをプロセッサへマッピングすること、及び、実行可能プログラムによる処理資源の受け取りを監視することによって、各グループ内の各ジョブは、同じ量のプロセッサ能力を経験することが可能になる。したがって、いくつかの代表的な実施の形態は、他の既知のマルチプロセッサスケジューリングアルゴリズムよりもはるかに「公平な」スケジューリングアルゴリズムを提供する。これに加えて、不完全な配分及び要求が少ないジョブは、限られた個数の間隔の間しかジョブに影響を与えない。具体的には、ジョブスケジューリングパラメータを使用して個々のジョブを特定のプロセッサにマッピングすることによって、このような問題を、ジョブのサブセットの損害につながる、スケジューリングオペレーションの恒久的なゆがみから防止する。グループ間の不完全性は、ＩＰＰアルゴリズムによって生成された複数の配分を交互に繰り返すことを使用して対処することができる。これに加えて、グループマッピングを実行可能プログラムの割り当てから分離することによって、いくつかの代表的な実施の形態が課すオーバーヘッドは比較的低くなり、それによって、アプリケーションからスケジューリングオペレーションへのプロセッサ資源の転換が省かれる。

【図面の簡単な説明】

【００３５】

【図１】代表的な一実施の形態による、複数の物理プロセッサに仮想プロセッサをスケジューリングするシステムを示す図である。

【図２】代表的な一実施の形態による、１組のＣＰＵに実行可能プログラムの各グループをマッピングする１つ又は数個の配分を生成して、スケジューリングオペレーションをサポートするＩＰＰアルゴリズムを含むフローチャートである。

【図３】代表的な一実施の形態による、特定の物理ＣＰＵへの個々のジョブのスケジューリングを行うためのフローチャートである。

【図４】実行可能プログラムのグループと複数のプロセッサとの間のマッピングを定義する配分を示す図である。

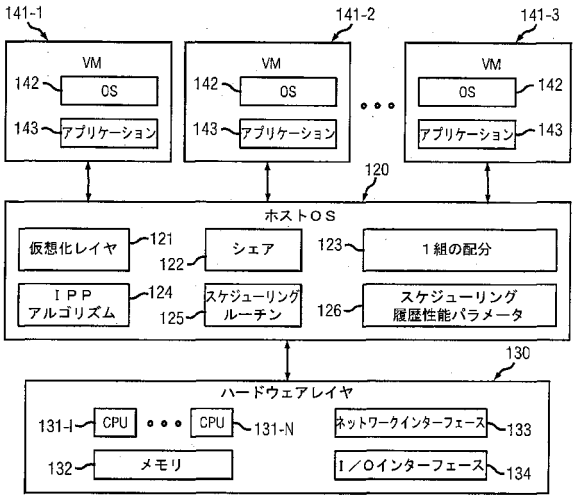
【符号の説明】

【００３６】

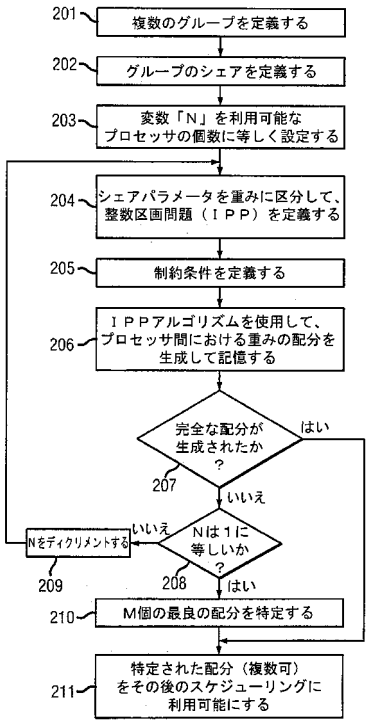
１２０・・・ホストＯＳ

１２１・・・仮想化レイヤ
１２２・・・シェア
１２３・・・１組の配分
１２４・・・ＩＰＰアルゴリズム
１２５・・・スケジューリングルーチン
１２６・・・スケジューリング履歴性能パラメータ
１３０・・・ハードウェアレイヤ
１３２・・・メモリ
１３３・・・ネットワークインターフェース
１３４・・・Ｉ／Ｏインターフェース
１４３・・・アプリケーション

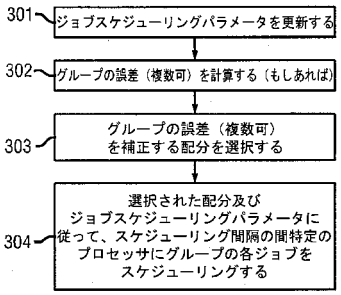
【図１】



【図２】



【図３】



【図４】

400

	プロセッサ		
	1	2	3
グループ及び重み	II (60)	III (60)	IV (60)
	I (40)	I (40)	V (40)

フロントページの続き

(56)参考文献 特開２００２－２０２９５９（ＪＰ，Ａ）
特開２００３－１５７１７７（ＪＰ，Ａ）

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