(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5299869号
(P5299869)
(45)発行日　平成25年9月25日(2013.9.25)　(24)登録日　平成25年6月28日(2013.6.28)

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

請求項の数 4 (全 16 頁)

(21)出願番号	特願2009-516504 (P2009-516504)	(73)特許権者	508371792
(86)(22)出願日	平成19年6月6日 (2007.6.6)		コンデューシブ・テクノロジーズ・コーポレイション
(65)公表番号	特表2009-541852 (P2009-541852A)		CONDUSIV TECHNOLOGIES CORPORATION
(43)公表日	平成21年11月26日 (2009.11.26)		アメリカ合衆国、９１５０４ カリフォルニア州、バーバンク、ノース・グレノークス・ブールバード、７５９０
(86)国際出願番号	PCT/US2007/013452	(74)代理人	100064746
(87)国際公開番号	WO2007/149228		弁理士 深見 久郎
(87)国際公開日	平成19年12月27日 (2007.12.27)	(74)代理人	100085132
審査請求日	平成22年5月20日 (2010.5.20)		弁理士 森田 俊雄
(31)優先権主張番号	11/471,466	(74)代理人	100083703
(32)優先日	平成18年6月19日 (2006.6.19)		弁理士 仲村 義平
(33)優先権主張国	米国(US)		
(31)優先権主張番号	11/546,072		
(32)優先日	平成18年10月10日 (2006.10.10)		
(33)優先権主張国	米国(US)		

前置審査　最終頁に続く

(54)【発明の名称】コンピュータマイクロジョブ

(57)【特許請求の範囲】
【請求項１】
コンピュータジョブを所有し、実行のために前記コンピュータジョブが提供されるオペレーティングシステムとは別個のプログラムによって、前記コンピュータジョブを複数のマイクロジョブに分割するステップを備え、
前記プログラムは前記複数のマイクロジョブのうちの第１のマイクロジョブのサイズを、前記第１のマイクロジョブを実行するのに必要なリソースの使用のために前記オペレーティングシステムによって前記コンピュータジョブに割当てられたタイムスライスの持続期間に基づいて選択し、さらに、
前記複数のマイクロジョブの各マイクロジョブの実行を要求することによって、前記コンピュータジョブを実行させるステップを備え、
プロセッサを備えるシステムによって実行される、方法。
【請求項２】
前記複数のマイクロジョブの各マイクロジョブは、前記リソースの所有権が当該マイクロジョブに与えられる実際の割当時間以下の時間で実行される、請求項１に記載の方法。
【請求項３】
前記各マイクロジョブの実行は、当該マイクロジョブを動作させるのに必要な１つ以上のリソースが１つ以上の遊休基準に準拠しているのに応答して要求される、請求項１に記載の方法。
【請求項４】

１つ以上のプロセッサと、
１つ以上のコンピュータプロセッサに通信結合されたコンピュータ読取可能記憶媒体とを備え、
前記コンピュータ読取可能媒体は、前記１つ以上のプロセッサによって実行されると、前記１つ以上のプロセッサに請求項１～３の１つ以上に記載の方法を実行させる命令の１つ以上のシーケンスを格納している、システム。

【発明の詳細な説明】
【技術分野】
【０００１】
発明の分野
この発明は、コンピュータ環境でソフトウェアアプリケーションを実行することに関する。特に、この発明の実施例は、アプリケーションのコンピュータまたは入出力ジョブをマイクロジョブに分割すること、およびそのマイクロジョブを実行することに関する。

【背景技術】
【０００２】
背景
多くのマルチタスクオペレーティングシステムにおいて、プロセスはいくつかのスレッドに細かくされる。スレッドは、オペレーティングシステム（Ｏ／Ｓ）によって実行される１つのコードである。マルチスレッディングの概念は、１つのプロセス内のいくつかのコード（またはスレッド）を「同時に」動作可能にすることである。たとえば、ワードプロセッサが動作している場合、ユーザは「メニュー項目を探す」をクリックしてポップアップボックスを表示することができる。このポップアップは、メインのワードプロセッサウインドウから独立して移動および操作され得る。したがって、ポップアップのためにメインのワードプロセッサウインドウが非アクティブになることはない。これが、ワードプロセッサプロセス内で動作する２つの異なるスレッドの例である。

【発明の概要】
【発明が解決しようとする課題】
【０００３】
マルチタスキングの概念は、単一のコンピュータプロセッサ上で同時に実行されている複数のコードの見かけを与えるという点でマルチスレッディングと同様である。相違点は、マルチタスキングはコンピュータ上で動作する１つよりも多いプロセスを指し、マルチスレッディングは上記の例のように同一のプロセス内で動作する１つよりも多いスレッドを指すことである。
【０００４】
同時に動作する１つよりも多いプロセスまたはスレッドの見かけは、マルチタスクスケジューラが、スレッドを「量子」と称され得る非常に小さな時間増分で動作するようにスケジュールする結果である。量子は、スレッドに与えられ、かつその間はそのスレッドがＣＰＵリソースを所有するタイムスライスである。量子の長さは、最近のオペレーティングシステムでは約２０ミリ秒から約１２０ミリ秒の範囲内にある。正確な時間は、Ｏ／Ｓが動作しているハードウェアに依存して異なり得る。さらに、Ｏ／Ｓは、特定のスレッドに与えられる量子の値を変更することができる。たとえば、スレッドがその第１の量子中に完了しない場合、Ｏ／Ｓは、そのスレッドの実行がスケジュールされている次回に量子のサイズを増大または減少させ得る。
【０００５】
人間の時間感覚と比べて量子の長さが短いため、およびラウンドロビン方法でスレッドを実行することによって、スレッドは同時に動作するように見える。最近のマルチタスクＯ／Ｓスケジューラはスレッドに優先順位を加え、優先順位の低いスレッドの前に優先順位の高いスレッドを最適に動作させるためのさまざまなアルゴリズムが存在する。しかし、すべてのスレッドは即時実行のためにＯ／Ｓスケジューラに提示され、Ｏ／Ｓスケジューラはそれらの優先順位に基づいてできる限り速くすべてのスレッドに実行を完了させる

。
【０００６】
しかし、この態様のスケジューリングの問題点は、コンピュータ性能が予想され得るより悪いということである。しばしば、プロセスは一瞬止まったり、動かなくなったりさえする。たとえば、ユーザ入力に基づいて表示画面に現われるプロセスはしばしば、別のプロセスが消費しているプロセッサ時間が多過ぎるために、ユーザがデータを入力するように現れることができない。
【０００７】
この節で説明された方策は、追求され得る方策であるが、以前に着想または追求された方策であるとは限らない。したがって、特に明記しない限り、この節で説明された方策のいずれも、単にこの節に含まれているからといって先行技術と見なされると仮定されるべきではない。
【０００８】
この発明は、同様の参照番号が同様の要素を指す添付の図面において、限定的ではなく例示的に図示される。
【図面の簡単な説明】
【０００９】
【図１】この発明の実施例に従った、マイクロジョブを実行するためのアーキテクチャの図である。
【図２】この発明の実施例に従った、コンピュータジョブをマイクロジョブに分割するマイクロスケジューラの図である。
【図３】この発明の実施例に従った、コンピュータジョブをマイクロジョブに分割するプロセスのステップを図示するフローチャートである。
【図４】本発明の実施例が実現され得るコンピュータシステムを図示するブロック図である。
【発明を実施するための形態】
【００１０】
詳細な説明
以下の説明では、説明のために、多数の特定的な詳細が記載されてこの発明の完全な理解を与える。しかし、この発明はこれらの特定的な詳細がなくても実施され得ることが明らかになるであろう。他の場合では、この発明を不要に不明瞭にするのを避けるため、周知の構造および装置がブロック図形態で示される。
【００１１】
概要
大多数のコンピュータは、常に自身のリソース容量のすべてを利用しているとは限らない。これは典型的に、１日２４時間、週７日よく使用されているように見えるコンピュータ、および毎日わずかな間しか立ち上げられないコンピュータについても当てはまる。したがって、コンピュータ時間およびリソースが無駄である。たとえば、２４時間にわたって、かなり多用され、かつ活動時に短時間のスパイクを有し得るコンピュータシステムは、平均して約５から２０パーセントのリソースしか使用していない可能性がある。
【００１２】
コンピュータジョブをマイクロジョブに分割することによってこれらの使われていないコンピュータリソースを利用するための方法、システム、および装置が本明細書中に開示される。マイクロジョブは、非常に小さなコンピュータコード（たとえば比較的少数の命令）であり得る。１つの実施例では、マイクロジョブのサイズは、そのマイクロジョブが処理リソースを割当てられた時間内にそのマイクロジョブが実行を終了するようなものである。たとえば、１つの実施例では、マイクロジョブ内の演算の数は、マイクロジョブが量子内に実行を完了するようなものである。マイクロジョブの長さは量子よりも短いことがある。
【００１３】

１つの実施例では、コンピュータジョブ全体がマイクロジョブに分割され、コンピュータジョブはその後マイクロジョブごとに実行されて、コンピュータジョブ全体が完了する。現在のリソース制約を考慮してコンピュータジョブをできる限り迅速に動作させようと試みること、または他のジョブおよびアプリケーションに重大な影響を与えるのを避けるためにジョブを「時間外」に動作するようにスケジュールすることとは異なり、コンピュータジョブは継続的にコンピュータ上で動作し得るが、あまりに微小であるため、ユーザまたは他のコンピュータジョブには感知不可能であり得る。したがって、ジョブは、ユーザならびに他のジョブおよびアプリケーションにとって完全に透明であり得る。ユーザはジョブをスケジュールする必要はなく、この方法を用いれば、ジョブは性能臨界時間中を含むいつでも動作可能である。

【００１４】

１つの実施例では、マイクロジョブは、選択された基準に基づいて時々挿入されて実行される。１つの実施例では、マイクロのジョブの実行は、マイクロジョブどうしの間の間隔をあけるための何らかの時間間隔の判断に基づいて、ある時間にわたって分散される。時間間隔は、マイクロジョブの各々の間で同一である必要はない。１つの実施例では、マイクロジョブ実行の基準は、リソース可用性に基づく。たとえば、マイクロジョブを実行するために、そのマイクロジョブが使用する１つ以上のリソースが１つ以上の遊休基準に準拠しているかどうかについての判断が下される。遊休基準が満たされている場合、そのマイクロジョブが実行される。

【００１５】

マイクロジョブ

１つの実施例では、マイクロジョブのサイズは、特定のマイクロジョブが処理ジョブを実行するために使用されるリソースを所有する割当時間内にその特定のマイクロジョブが完了可能なサイズである。１つの実施例では、各マイクロジョブは、自身の割当時間内に完了するようなサイズである。しかし、マイクロジョブの中には、大きすぎて自身の割当時間内に実行を完了不可能なものもあり得る。

【００１６】

１つの実施例では、割当時間は量子である。上述のように、量子は、コンピュータコードの一部（たとえばスレッド）に与えられ、かつその間はそのコード部分がＣＰＵリソースを所有するタイムスライスである。これも上述のように、異なるオペレーティングシステムは異なる量子を使用する。さらに、特定のコード部分に割振られる量子は、実行時間中に状況に基づいて変化し得る。たとえば、Ｏ／Ｓは、スレッドに割当てられる量子のサイズを増大または減少させ得る。１つの実施例では、コンピュータジョブは、そのコンピュータジョブに割当てられると予想される量子のサイズに基づいてマイクロジョブに分割される。別の実施例では、コンピュータジョブは、そのコンピュータジョブに割当てられた量子のサイズに基づいてマイクロジョブに分割される。コンピュータジョブのどの部分をマイクロジョブとして分けるべきかについての判断は、実行時間前または実行時間中のいずれかに下され得る。

【００１７】

マイクロジョブは、１つの実施例に従って、次のマイクロジョブが実行されるまでの実行の一時休止を安全に見込んでおきつつ単一の単位として完了され得る、実質的により小さい（たとえば最小の）作業単位である。実行の一時休止を安全に見込んでおくというのは、特定のマイクロジョブの実行を、すべてのマイクロジョブの実行から生じる結果に影響を及ぼすことなく遅らせることができるという意味である。

【００１８】

マイクロジョブはスレッドの一部であり得る。たとえば、スレッドが複数のマイクロジョブに分割され得る。これらのマイクロジョブは、スレッドがスケジュールされる方法と同様にスケジュールされ得る。しかし、上述のように、１つの実施例では、マイクロジョブは、自身が処理リソースを所有する量子または他の期間にわたって実行可能であれば実行を完了することになる。

【００１９】
マイクロジョブは、どの時点においても非常に少量のリソース（たとえばＣＰＵ時間、メモリ割当）しか必要とし得ない。どの時点においてもそのように最小限のリソースしか使用しないため、見えないプロセスになり得る。マイクロジョブを小さく保つことによって、コンピュータジョブが１度に使用するコンピュータリソースが少量で済むようになる。したがって、この発明の１つの実施例に従って、マイクロジョブの実行は、コンピュータシステム内の他のアプリケーションの性能に重大な影響を与えないように十分少量のリソースを消費する。
【００２０】
コンピュータジョブのマイクロジョブへの分割
Ｉ）　アプリケーションが自身のコンピュータジョブをマイクロジョブに分割する
１つの実施例では、アプリケーションプログラムが自身のコンピュータジョブを複数のマイクロジョブに分割する。本明細書中で用いられているように、複数という用語は、１よりも大きいいずれかの数を意味する。図１は、この発明の実施例に従った、マイクロジョブを実行するためのアーキテクチャ１００の図である。各ＭＪＳ対応のアプリケーション１１５（１）～１１５（ｎ）が、自身のコンピュータジョブをマイクロジョブ１２５に分割して実行する。たとえば、アプリケーションプログラマがアプリケーションコード内の適切な場所で、マイクロジョブ１２５を実行するＭＪＳ１１０の許可を要求する呼出しを行うことができ、これが事実上コンピュータジョブをマイクロジョブ１２５に分割する。例として、コンピュータジョブは、バックアップ、索引付け、ソフトウェア更新、ウイルスおよび有害ソフトスキャン、ならびにフラグメンテーション解消などのメンテナンスを行い得る。しかし、ＭＪＳ対応のアプリケーション１１５（１）～１１５（ｎ）はメンテナンス以外のソフトウェアであってもよい。
【００２１】
引続き図１を参照して、マイクロジョブスケジューラ（ＭＪＳ）１１０は、いつマイクロジョブ１２５が実行可能であるかを判断する。ＭＪＳとともに作動可能なアプリケーションを、本明細書中ではＭＪＳ対応のアプリケーション１１５と称する。この実施例では、ＭＪＳ１１０は、特定のＭＪＳ対応のアプリケーション１１５（たとえば１１５（１））が、１つ以上のマイクロジョブ１２５を実行可能にするよう要求できるようにするためのアプリケーションプログラムインターフェイス（ＡＰＩ）１３０を有する。ＡＰＩ１３０はまた、以下により十分に説明されるように、ＭＪＳ対応のアプリケーション１１５が、マイクロジョブ１２５がどれだけ分散され得るかを指定できるようにする。例示的なＡＰＩが以下の本明細書中に含まれる。しかし、アーキテクチャ１００はこの例示的なＡＰＩに限定されない。
【００２２】
１つの実施例では、ＭＪＳ１１０は、次にどのマイクロジョブ１２５を実行可能にすべきかを判断できるように、マイクロジョブ待ち行列を維持する。マイクロジョブ１２５の実行は、他のアプリケーションに対する影響が無視できるものであるようにＭＪＳ１１０によってタイミングが取られる。
【００２３】
マイクロジョブ１２５の実行は、アプリケーションとＭＪＳとの間のＡＰＩの呼出しまたは他の通信方法において、ＭＪＳ対応のアプリケーション１１５によって指定され得る。ＭＪＳ１１０が、次のマイクロジョブ１２５が他のジョブに影響を与えることなく実行可能であると判断すると、ＭＪＳ１１０は、ＭＪＳ対応のアプリケーション１１５（１）にマイクロジョブ１２５を実行するよう命令することによって、ＭＪＳ対応のアプリケーション１１５に応答する。
【００２４】
１つの実施例では、コンピュータリソース利用が監視および分析されて、リソース利用が１つ以上の遊休基準に準拠しているかどうかを判断する。ＭＪＳは、１つ以上の遊休基準が満たされるとマイクロジョブ１２５を実行させる。１つの実施例では、特定のマイク

ロジョブ１２５を実行させるのに必要な時間は量子より小さいかそれに等しいため、そのマイクロジョブ１２５が使用するいずれのリソースも、別のジョブがそのリソースを必要とする前に放棄される。したがって、マイクロジョブ１２５によるリソース利用は気づかれないことがあり、マイクロジョブ１２５はそのアプリケーションの環境には見えないことがある。１つの実施例では、ＭＪＳは、時間間隔に基づいてマイクロジョブ１２５をスケジュールする。時間間隔に基づいたスケジューリングが以下に説明される。

【００２５】

ＭＪＳ１１０はまた、１つの実施例では、メモリマネージャ１４０も有する。ＭＪＳ１１０は初期化されると、オペレーティングシステムによってメモリが割当てられ、そのうちのいくらかは自身の目的のために使用し、そのうちのいくらかはＭＪＳ対応のアプリケーション１１５に割当てる。ＭＪＳ対応のアプリケーション１１５（１）は起動すると、ＭＪＳ１１０からのメモリ割当てを要求する。ＭＪＳ１１０は、すべてのプロセスによる現在のコンピュータシステムメモリ利用およびＭＪＳ対応のアプリケーション１１５（１）のニーズなどの要因に基づいて、どれくらいのメモリをＭＪＳ対応のアプリケーション１１５（１）に割当てるべきかを判断し得る。メモリ要件は、各ＭＪＳ対応のアプリケーション１１５に特定的であってもよいし、コンピュータソフトウェアプログラマによってＭＪＳ対応のアプリケーション１１５にプログラムされてもよい。

【００２６】

ＩＩ）　スケジューラがコンピュータジョブをマイクロジョブに分割する

１つの実施例では、ＭＪＳ１１０がコンピュータジョブをマイクロジョブ１２５に分割する。図２を参照して、ＭＪＳ１１０は、コンピュータジョブ２０５をマイクロジョブ１２５に分割するマイクロジョブ分割ロジック２１０を有する。ＭＪＳ１１０はまた、マイクロジョブ１２５を実行するためにスケジュールするマイクロジョブスケジューリングロジック２２０も有する。たとえば、ＭＪＳ１１０は、ＭＪＳ対応でないアプリケーションプログラムに巻付けられたシェルとして作動し得る。したがって、この例では、シェルＭＪＳ１１０は、それを通じて如何なる実行形式のファイルも動作させることが可能な完全なソフトウェアアプリケーションである。したがって、１つの実施例では、シェルＭＪＳ１１０は、コンピュータジョブを実行形式のファイルからマイクロジョブ１２５に自動的に分割する。換言すれば、この実施例では、アプリケーションプログラマはアプリケーションをマイクロジョブ１２５に分割する必要がない。

【００２７】

シェルＭＪＳ１１０は、１つの実施例では、リソース利用に基づいてコンピュータジョブを実行形式のファイルからマイクロジョブ１２５に分割する。シェルＭＪＳ１１０は、アプリケーションおよびそのアプリケーションがどのように動作するかを分析して、そのアプリケーションがどのリソースを使用するかを調べ得る。たとえば、シェルＭＪＳ１１０は、そのアプリケーションがどのリソースを使用するか、およびそのアプリケーションがどの程度までそのリソースを使用するかを分析する。たとえば、ディスクデフラグ用プログラムが動作すると、シェルＭＪＳ１１０は、アプリケーションがどのリソースを使用するか（たとえばＣＰＵ、ネットワーク、ディスクＩ／Ｏ）を判断することができる。シェルＭＪＳ１１０は、１つの実施例では、この分析に基づいてどのようにアプリケーションをマイクロジョブ１２５に分割するかを自動的に判断する。シェルＭＪＳ１１０はまた、この分析に基づいて、どのようにマイクロジョブ１２５をスケジュールするかも判断し得る。

【００２８】

シェルＭＪＳ１１０はさまざまなパラメータを使用して、どのようにコンピュータジョブをマイクロジョブ１２５に分割するかを判断し得、および／またはどのようにマイクロジョブ１２５を実行するためにスケジュールするかを判断し得る。これらのパラメータは、ユーザ入力に基づき得るか、またはシェルＭＪＳ１１０によって確立され得る。たとえば、ユーザは、特定のアプリケーションの優先順位が高いように指定し得る。

【００２９】

シェルＭＪＳ１１０は、１つの実施例では、時間間隔に基づいてマイクロジョブ１２５を実行するためにスケジュールする。

【００３０】

１つの実施例では、ＭＪＳ１１０はオペレーティングシステムの一部である。この実施例では、オペレーティングシステム内のＭＪＳ１１０がコンピュータジョブをマイクロジョブ１２５に分割し得る。

【００３１】

時間間隔に基づいたマイクロジョブのスケジューリング

１つの実施例では、マイクロジョブ１２５は時間間隔に基づいて実行される。たとえば、ＭＪＳ１１０は時間間隔に基づいてマイクロジョブ１２５をスケジュールする。たとえば、ＭＪＳ１１０はマイクロジョブ１２５の実行をある時間にわたって分散させる。たとえば、コンピュータジョブは、マイクロジョブ１２５に分割されていない場合、完了するのに１２分かかり得る。しかし、マイクロジョブ１２５に分割された場合、コンピュータジョブ全体の実行を２４時間にわたって分散させることができ、各特定のマイクロジョブ１２５は２，３秒に１度実行される。

【００３２】

特定の例として、量子が２０ミリ秒である場合、コンピュータジョブ全体は約３６，０００量子で完了し得る。したがって、コンピュータジョブは約３６，０００個のマイクロジョブ１２５に分割される。マイクロジョブ１２５のいくつかが量子よりも小さい場合、マイクロジョブ１２５の数は若干多くなり得る。２０ｍｓ量子であるとすると、２４時間にわたって４，３２０，０００量子が存在する。したがって、ＭＪＳ１１０は、１２０量子ごと、または２．４秒ごとに１つのマイクロジョブ１２５を実行し得る。この特定の例では、マイクロジョブ対応のアプリケーションは完了するのに２４時間かかることになるが、極端に繁忙のシステムでも検出不可能であり得る。

【００３３】

事象に基づいたマイクロジョブのスケジューリング

１つの実施例では、マイクロジョブ１２５は、事象に基づいて実行がスケジュールされる。たとえば、ＭＪＳ１１０は、発生するある数の演算に基づいてマイクロジョブ１２５をスケジュールする。別の例として、ＭＪＳ１１０は、発生するある数の量子に基づいてマイクロジョブ１２５をスケジュールする。異なる量子は異なるサイズであり得る。したがって、１つの実施例では、ＭＪＳ１１０は事象に基づいてマイクロジョブ１２５の実行を分散させる。

【００３４】

リソースに基づいたマイクロジョブのスケジューリング

１つの実施例では、マイクロジョブ１２５はリソース利用に基づいてスケジュールされる。たとえば、１つの実施例では、ＭＪＳ１１０はリソースに基づいたスケジューラである。たとえば、ＭＪＳ１１０は、マイクロジョブ１２５が遊休リソースしか利用しないようにマイクロジョブ１２５をスケジュールする。ＭＪＳ１１０は、リソース利用が１つ以上の遊休基準に準拠しているかどうかを判断してスケジューリング決定を下す。例として、ＭＪＳ１１０はディスク使用率を分析し得る。マイクロジョブ１２５を有するアプリケーション以外のアプリケーションがそのディスクを使用している場合、ＭＪＳ１１０は、この他のアプリケーションが終わるのを待ってからマイクロジョブ１２５をスケジュールする。ＭＪＳ１１０はディスクＩ／Ｏ利用を監視し続け、ディスクＩ／Ｏへのアクセスを求めている他のアプリケーションがなければ、別のマイクロジョブ１２５をスケジュールできるようにする。しかし、別のアプリケーションがディスクＩ／Ｏの利用を求めている場合は、ＭＪＳ１１０は別のマイクロジョブ１２５をスケジュールできないようにして、この他のアプリケーションがディスクＩ／Ｏを利用することができる。

【００３５】

別の例として、ＭＪＳ１１０はネットワーク使用率を分析し得る。ネットワークトラフィックが高すぎる場合、ＭＪＳ１１０はトラフィックが遅くなるまでマイクロジョブ１２

５を１つもスケジュールしないことになる。ネットワークトラフィックが十分低い場合は、ＭＪＳ１１０はマイクロジョブを実行するためにスケジュールする。ＭＪＳ１１０は、ネットワークトラフィックが十分低くあり続けていることを確認し続ける。ネットワークトラフィックが十分低くあり続けている場合、別のマイクロジョブ１２５がスケジュールされ得る。しかし、トラフィックが高くなりすぎると、さらに他のマイクロジョブ１２５は実行がスケジュールされない。

【００３６】

ＭＪＳ１１０は、いずれかの種類のコンピュータリソースおよびリソースのいずれかの組合せに基づいて、リソースに基づいたスケジューリング決定を下し得る。１つの実施例では、ＭＪＳ１１０は、実行される許可を待っているマイクロジョブ１２５の複数の待ち行列を有する。各待ち行列は特定のリソースに対応し得る。たとえば、ディスクＩ／Ｏを利用する必要があるマイクロジョブ１２５の待ち行列、ネットワークを利用する必要があるマイクロジョブ１２５の待ち行列、ＣＰＵを利用する必要があるマイクロジョブ１２５の待ち行列などがあり得る。リソースの組合せを利用するマイクロジョブ１２５の１つ以上の待ち行列もあり得る。ＭＪＳ１１０は、特定のリソースまたはリソースの組合せが利用可能であればマイクロジョブ１２５を配布する。特定のマイクロジョブ１２５は、２つのリソースの使用を必要とし得る。たとえば、特定のマイクロジョブ１２５は、ネットワークリソースおよびディスクリソースの使用を必要とし得る。しかし、この特定のマイクロジョブ１２５はＣＰＵリソースを必要としない。ＣＰＵリソース利用が現在高いとしても、この特定のマイクロジョブ１２５をスケジュールおよび実行することができる。

【００３７】

ＭＪＳ対応のアプリケーション１１５は、この発明の実施例に従って、ＭＪＳ１１０にパラメータを送ってリソース利用を制御する。リソース利用の制御は、ディスクＩ／Ｏ、ＣＰＵおよびネットワークを含むが、これらに限定されない。たとえば、ＭＪＳ対応のアプリケーション１１５は、上記の３つのリソースの閾値レベルのいずれかの組合せが得られるまでは、マイクロジョブ１２５の実行を要求することができる。さらに、ＭＪＳ対応のアプリケーション１１５は、異なるマイクロジョブ１２５に異なるリソース閾値レベルを指定することができる。たとえば、ＭＪＳ対応のアプリケーション１１５は、１つの実施例に従って、各マイクロジョブ１２５に異なるリソース閾値レベルを指定する。したがって、細分化されたリソース管理が可能である。ＭＪＳ１１０がリソース利用を計算する場合、この発明の１つの実施例に従うと、測定されるのは、ＭＪＳ対応のアプリケーション１１５以外のプロセスのリソース利用である。ＣＰＵ利用閾値が２０パーセントに設定される以下の例を用いて説明する。ＭＪＳ対応のアプリケーション１１５を実行可能にする前にＣＰＵ利用が２０パーセント未満である場合、ＣＰＵ利用は、マイクロジョブが実行されると２０パーセントより高く増加し得る。この２０パーセントを超える増加は、この例ではＣＰＵリソース利用違反とはみなされない。同様の原理がネットワークおよびディスクＩ／Ｏリソースに当てはまる。

【００３８】

ＭＪＳ１１０がオペレーティングシステムの外部で実行される場合、ＭＪＳ１１０は、１つの実施例では、自身のリソース利用において自己制限する。たとえば、ＭＪＳ１１０は自身のリソース利用を監視し、自身のリソース利用が高くなりすぎると、ＭＪＳ１１０は、ある期間の間だけＭＪＳ１１０のスケジューリングを停止するようオペレーティングシステムに要求を出す。

【００３９】

プロセスフロー

図３は、この発明の実施例に従った、コンピュータジョブをマイクロジョブ１２５に分割することによってコンピュータジョブを実行するためのプロセス３００のステップを図示するフローチャートである。ステップ３０２において、コンピュータジョブが開始される。コンピュータジョブは、ＭＪＳ対応のアプリケーション１１５からのものであり得る。しかし、コンピュータジョブは、ＭＪＳ対応のアプリケーション１１５に関連付けられ

なくてもよい。

【００４０】

例として、ステップ３０２において、１つの実施例では、コンピュータシステムが立ち上がるとＭＪＳ対応のアプリケーション１１５プログラムが起動される。ＭＪＳ対応のアプリケーション１１５が行うべきコンピュータジョブを有していない場合、ＭＪＳ対応のアプリケーション１１５は、動作すべきコンピュータジョブを有するまで遊休状態にあり続ける。この遊休状態では、ＭＪＳ対応のアプリケーション１１５は、時折の監視などのいくつかの機能を果たし得る。ある点で、ＭＪＳ対応のアプリケーション１１５は、記憶媒体のデフラグ、またはウイルススキャンなどの行うべきコンピュータジョブを有する。コンピュータジョブは、単一のディスクまたはそれに記憶されるファイルをデフラグすることであり得、ＭＪＳ対応のアプリケーション１１５は継続的にそのディスクをデフラグする。

【００４１】

ＭＪＳ対応のアプリケーション１１５は、起動されると少量のメモリが割当てられ得る。ＭＪＳ対応のアプリケーション１１５は典型的に、１度に単一のマイクロジョブ１２５しか実行しようとしないため、割当てられる量は非常に少量であり得る。しかし、場合によっては、ＭＪＳ対応のアプリケーション１１５は、実行すべき他のプロセスを待たずに複数のマイクロジョブ１２５を実行しようとし得る。たとえば、必要なコンピュータシステムリソースが遊休状態にあるとＭＪＳ１１０が判断した場合、ＭＪＳ１１０は、複数のマイクロジョブ１２５が利用するリソースを利用している別のプロセスがなければ、ＭＪＳ対応のアプリケーション１１５がその複数のマイクロジョブ１２５を続けて実行できるようにし得る。

【００４２】

ステップ３０４において、コンピュータジョブはマイクロジョブ１２５に分割される。特定のマイクロジョブ１２５のサイズは、１つの実施例では、その特定のマイクロジョブ１２５が処理ジョブを実行するために使用されるリソースを所有する割当時間内にその特定のマイクロジョブ１２５が完了可能なサイズである。マイクロジョブ１２５は、マイクロジョブ１２５の実行が、コンピュータシステム内の他のジョブの性能に重大な影響を与えないよう十分少量のリソースを利用するようなサイズであり得る。１つの実施例では、コンピュータジョブをマイクロジョブ１２５に分割することは、各マイクロジョブ１２５が、そのコンピュータジョブにそのマイクロジョブ１２５を実行するために使用されるリソースの所有権が与えられる割当時間内に実行を完了できるように、マイクロジョブ１２５のサイズを選択することを備える。

【００４３】

１つの実施例では、コンピュータジョブは、そのコンピュータジョブを所有するアプリケーションによってマイクロジョブ１２５に分割される。コンピュータジョブのマイクロジョブ１２５への分割は、ＭＪＳ対応のアプリケーション１１５内の命令によって達成され得る。一般に、これらの命令は、ＭＪＳ対応のアプリケーション１１５内の決定点である。たとえば、命令は、マイクロジョブ１２５を実行する許可を要求する、ＭＪＳ１１０へのＡＰＩの呼出しであり得る。しかし、ＭＪＳ１１０はＭＪＳ対応のアプリケーション１１５に統合されてもよく、その場合命令は、ＭＪＳ対応のアプリケーション１１５内のスケジューリング機能への呼出しであり得る。他の技術を用いてコンピュータジョブをマイクロジョブ１２５に分割してもよい。たとえば、１つの実施例では、コンピュータジョブは、シェルＭＪＳ１１０内のマイクロジョブ分割ロジックによってマイクロジョブ１２５に分割される。

【００４４】

ステップ３０６はマイクロジョブ１２５の実行である。１つの実施例では、コンピュータジョブ全体がマイクロジョブ１２５に分割され、コンピュータジョブはその後マイクロジョブ１２５ごとに実行されて、コンピュータジョブ全体が完了する。たとえば、デフラグジョブ全体がマイクロジョブ１２５に分割され、それらが１度に１つずつ実行されて、

デフラグジョブ全体が完了する。特定のマイクロジョブ１２５は、１つの実施例では、割当時間より短い時間で、または割当時間と等しい時間で実行を完了する。

【００４５】

マイクロジョブ１２５ごとの実行は、順次の１つずつの実行に限定されないが、順次の１つずつの実行を含む。複数のマイクロジョブ１２５が同時に実行され得る。たとえば、複数のＣＰＵがある場合、異なるマイクロジョブ１２５が異なるＣＰＵで同時に実行され得る。

【００４６】

１つの実施例では、マイクロジョブ１２５のスケジューリングはリソース利用に基づく。この実施例では、マイクロジョブ１２５の各々について、特定のマイクロジョブ１２５が使用するコンピュータシステムの１つ以上のリソースの利用が１つ以上の遊休基準を満たすかどうかについての判断が下される。特定のリソースの遊休基準は１つ以上の要因に基づき得る。たとえば、１つの実施例では、ＣＰＵリソース利用の遊休基準としてＣＰＵ利用が使用される。したがって、マイクロジョブ１２５が実行されるのは、マイクロジョブ１２５が必要とするコンピュータシステムのリソースが十分遊休状態にあるときのみである。遊休基準は、１つの実施例では、リソース閾値に基づく。たとえば、ＭＪＳ対応のアプリケーション１１５のマイクロジョブ１２５が、他のプロセスによるリソース利用がそのＭＪＳ対応のアプリケーション１１５が指定する閾値未満である場合にのみ実行されるようなリソース閾値が使用され得る。以下に説明される例示的なＡＰＩは、いくつかのリソース閾値パラメータの一例を含む。しかし、プロセス３００はこれらのリソース閾値パラメータに限定されない。

【００４７】

１つの実施例では、マイクロジョブ１２５のスケジューリングは時間間隔に基づく。１つの実施例では、コンピュータジョブの実行はある期間にわたって分散される。たとえば、コンピュータジョブは数時間にわたって分散され得る。コンピュータジョブがいくつのマイクロジョブ１２５に分割されるかに基づいて、どのようにマイクロジョブ１２５をある時間にわたって分散させるかについての判断が下され得る。連続するマイクロジョブ１２５どうしの間の時間は均一である必要はない。

【００４８】

１つの実施例では、コンピュータジョブを開始したアプリケーションプログラムは継続的に動作し、コンピュータシステムが立ち上がっているかぎり動作し続けるが、行わなければならない作業がない間は遊休状態にあり続ける。たとえば、ディスクデフラグ用プログラムからウイルス検出アプリケーションプログラムへと動作し続ける。したがって、コンピュータジョブが完了したとしても、アプリケーションプログラムは作業の次のセグメントを待ちながら遊休モードで動作し続ける。したがって、アプリケーションプログラムは、行うべき別のコンピュータジョブがあるときに再起動される必要がない。この結果、アプリケーションプログラムは、アプリケーションプログラムの起動に典型的な付加的なリソースを消費しない。アプリケーションプログラムが、行うべき別のコンピュータジョブを有すると判断した場合、コンピュータジョブはマイクロジョブ１２５に分割され、マイクロジョブ１２５はある時間にわたって実行される。たとえば、ディスクデフラグ用プログラムのアプリケーションプログラムは、コンピュータの記憶媒体に対する変更に基づいて付加的な記憶媒体デフラグがなされるべきであると判断し得る。

【００４９】

例示的なＡＰＩ

この発明の実施例は、ＭＪＳ対応のアプリケーション１１５がＭＪＳ１１０とインターフェイス接続できるようにするためのＡＰＩである。例示的なＡＰＩは、ＣＰＵ、ディスク、およびネットワークについて以下のリソース閾値パラメータを有する。

【００５０】

・ＣＰＵ利用閾値

・保留ディスクＩ／Ｏカウント閾値

・ネットワーク利用閾値
各マイクロジョブ１２５について上記のパラメータが指定され得る。換言すれば、異なるマイクロジョブ１２５には異なるリソース閾値パラメータが割振られ得る。たとえば、ネットワークを使用するマイクロジョブ１２５については、ネットワーク閾値が使用され得る。しかし、ネットワークを使用しないマイクロジョブ１２５については、ネットワーク閾値はゼロであり得る。したがって、この発明の実施例に従って、細分化されたリソース管理が提供される。

【００５１】
特定の例として、ＭＪＳ対応のアプリケーション１１５は、ＣＰＵ利用が５０％未満であり、Ｉ／Ｏディスク利用が４０％未満であり、ネットワークトラフィックが６０％未満である場合にのみ特定のマイクロジョブ１２５の実行を要求し得る。全く何もなしも含めて、リソース閾値因子のいずれかの組合せが用いられ得る。ＣＰＵ利用閾値は、この発明の実施例に従って、いずれかの他のジョブの利用閾値とは対照的に、ＭＪＳのＣＰＵの使用ごとに異なる。

【００５２】
以下の２つのパラメータを用いて、リソース利用をどれほど頻繁に測定すべきかを指定する。

【００５３】
・ＣＰＵ利用ウィンドウ
・ネットワーク利用ウィンドウ
ＣＰＵ利用ウィンドウパラメータは、それにわたるＣＰＵ利用が計算されるタイムウィンドウを規定する。たとえば、最新のｎミリ秒にわたるＣＰＵ利用が平均化される。ネットワーク利用ウィンドウは、それにわたるネットワーク利用が計算されるタイムウィンドウを規定する。これらのパラメータはＭＪＳ１１０に内在し得る。しかし、ＭＪＳ対応のアプリケーション１１５はこれらのパラメータをオーバーライドし得る。保留ディスクＩ／Ｏはどの時点においても絶対であり、したがって計算する必要はない。

【００５４】
必須の遊休時間パラメータがＭＪＳ対応のアプリケーション１１５からＭＪＳに渡されて、どのようにマイクロジョブ１２５がある時間にわたって分散されるかを制御し得る。必須の遊休時間パラメータは任意である。さらに、使用時には、必須の遊休パラメータの値はゼロであり得る。

【００５５】
・必須の遊休時間
ＭＪＳ１１０は、すべてのマイクロジョブ１２５が実行された後のシステム遊休時間であると規定される「遊休時間」を追跡する。上述のように、ＭＪＳ対応のアプリケーション１１５は、ＭＪＳ１１０を用いてマイクロジョブ１２５を待ち行列に入れることができる。ＭＪＳ待ち行列にマイクロジョブ１２５がない場合、ＭＪＳ１１０は指定された必須の遊休時間だけ待ってから起きて、ＭＪＳ対応のアプリケーション１１５が付加的な作業を行なうことを認可する。たとえば、ＭＪＳ対応のデフラグ用プログラムは、まずある数のマイクロジョブ１２５を実行してディスクドライブをデフラグし、その後ＭＪＳ１１０によって休止され得る。指定された必須の遊休時間の後、ＭＪＳ１１０は、ＭＪＳ対応のデフラグ用プログラムを呼出して付加的な作業を認可する。たとえば、ＭＪＳ対応のデフラグ用プログラムは、メモリの解放などのクリーンアップジョブを実行し得る。必須の遊休時間は、ＭＪＳ対応のアプリケーション１１５によって調整され得るデフォルトパラメータであり得る。

【００５６】
以下のパラメータは、リソース利用が閾値を上回るときにマイクロジョブ１２５の実行を待つことに関する。

【００５７】
・待ち時間

・最大待ち時間
リソース利用が現在高すぎてマイクロジョブを実行できないとＭＪＳ１１０が判断した場合、ＭＪＳ１１０は、指定された待ち時間だけ待ってから、リソース利用を再確認する。待ち時間パラメータは、ＭＪＳ１１０がリソース利用が高すぎると判断するたびに増加され得る。たとえば、ＭＪＳ１１０は、最大待ち時間に達するまで待ち時間パラメータを増加させることができる。これらのパラメータは、ＭＪＳ対応のアプリケーション１１５が最初に始動されるときにアプリケーション１１５によって指定され得る。ＭＪＳ対応のアプリケーション１１５は、実行時間中にこれらのパラメータを調整することができる。

【００５８】
ハードウェア概要
図４は、本発明の実施例が実現され得るコンピュータシステム４００を図示するブロック図である。プロセス３００のステップは、システム４００の１つ以上のコンピュータ読取可能媒体の命令として記憶され、コンピュータシステム４００のプロセッサ上で実行される。コンピュータシステム４００は、情報を通信するためのバス４０２または他の通信メカニズムと、バス４０２に結合される、情報を処理するためのプロセッサ４０４とを含む。コンピュータシステム４００はまた、バス４０２に結合される、プロセッサ４０４によって実行されるべき情報および命令を記憶するための、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４０６も含む。メインメモリ４０６はまた、プロセッサ４０４によって実行されるべき命令の実行中に、一時変数または他の中間情報を記憶するために用いられ得る。コンピュータシステム４００はさらに、バス４０２に結合される、プロセッサ４０４のための静的情報および命令を記憶するための読出専用メモリ（ＲＯＭ）４０８または他の静的記憶装置を含む。情報および命令を記憶するための、磁気ディスクまたは光学ディスクなどの記憶装置４１０が設けられ、バス４０２に結合される。コンピュータシステム４００は、プロセッサ４００をいくつでも有してよい。たとえば、１つの実施例では、コンピュータシステム４００はマルチプロセッサシステムである。プロセッサ４０４は、コアをいくつでも有してよい。１つの実施例では、プロセッサ４０４はマルチコアプロセッサ４０４である。コンピュータシステム４００は、ハイパースレッドマシン内で用いられ得る。

【００５９】
コンピュータシステム４００は、コンピュータユーザに情報を表示するための、陰極線管（ＣＲＴ）などのディスプレイ４１２にバス４０２を介して結合され得る。プロセッサ４０４に情報および指令選択を通信するための、英数字および他のキーを含む入力装置４１４がバス４０２に結合される。別の種類のユーザ入力装置は、プロセッサ４０４に方向情報および指令選択を通信するため、ならびにディスプレイ４１２上のカーソルの動き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部４１６である。この入力装置は典型的に、第１の軸（たとえばｘ）および第２の軸（たとえばｙ）の２本の軸における２自由度を有し、これにより、この装置が平面内の位置を指定することできる。

【００６０】
本発明は、本明細書中において説明される技術を実現するためのコンピュータシステム４００の使用に関する。本発明の１つの実施例によると、それらの技術は、プロセッサ４０４がメインメモリ４０６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のに応答して、コンピュータシステム４００によって行われる。そのような命令は、記憶装置４１０などの別の機械読取可能媒体からメインメモリ４０６に読込まれ得る。メインメモリ４０６に含まれる命令のシーケンスの実行により、プロセッサ４００が、本明細書中において説明されるプロセスステップを実行することになる。代替的な実施例では、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組合せて、配線による回路構成を用いて本発明を実現してもよい。したがって、本発明の実施例は、ハードウェア回路構成およびソフトウェアのいずれの特定的な組合せにも限定されない。

【００６１】

本明細書中で用いられるような「機械読取可能媒体」という用語は、機械に特定的な態様で操作させるデータを与えるのに関与するいずれかの媒体を指す。コンピュータシステム４００を用いて実現される実施例では、たとえばプロセッサ４０４に実行用の命令を与える際に、さまざまな機械読取可能媒体が含まれる。そのような媒体は、不揮発性媒体、揮発性媒体、および送信媒体を含む多くの形態を取り得るがこれらに限定されない。不揮発性媒体は、たとえば、記憶装置４１０などの光学または磁気ディスクを含む。揮発性媒体は、メインメモリ４０６などの動的メモリを含む。送信媒体は、バス４０２を備える配線を含む、同軸ケーブル、銅線および光ファイバを含む。送信媒体はまた、電波および赤外線データ通信中に発生されるような音波または光波の形態も取り得る。すべてのそのような媒体は、媒体によって搬送される命令が、機械に命令を読込む物理メカニズムによって検出され得るように具体的である必要がある。

【００６２】

機械読取可能媒体の一般的な形態は、たとえば、フロッピー（登録商標）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、もしくはいずれかの他の磁気媒体、ＣＤ－ＲＯＭ、いずれかの他の光学媒体、パンチカード、紙テープ、孔のパターンを有するいずれかの他の物理媒体、ＲＡＭ、ＰＲＯＭ、ＥＰＲＯＭ、ＦＬＡＳＨ－ＥＰＲＯＭ、いずれかの他のメモリチップもしくはカートリッジ、後で説明されるような搬送波、またはそこからコンピュータが読込可能ないずれかの他の媒体を含む。

【００６３】

実行用の１つ以上の命令の１つ以上のシーケンスをプロセッサ４０４に搬送する際、さまざまな形態の機械読取可能媒体が含まれ得る。たとえば、命令は当初、リモートコンピュータの磁気ディスクで搬送され得る。リモートコンピュータは、自身の動的メモリに命令をロードし、モデムを用いて電話線で命令を送ることができる。コンピュータシステム４００にローカルなモデムが電話線上のデータを受け、赤外線送信機を用いてそのデータを赤外線信号に変換することができる。赤外線検出器が、赤外線信号内に搬送されるデータを受信することができ、適切な回路構成がそのデータをバス４０２に乗せることができる。バス４０２は、データをメインメモリ４０６に搬送し、そこからプロセッサ４０４が命令を取出して実行する。メインメモリ４０６が受けた命令は、プロセッサ４０４による実行の前または後のいずれかに、記憶装置４１０に任意で記憶され得る。

【００６４】

コンピュータシステム４００はまた、バス４０２に結合される通信インターフェイス４１８も含む。通信インターフェイス４１８は、ローカルネットワーク４２２に接続されるネットワークリンク４２０に、双方向データ通信結合を与える。たとえば、通信インターフェイス４１８は、対応する種類の電話線にデータ通信接続を与えるための統合サービスデジタル網（ＩＳＤＮ）カードまたはモデムであり得る。別の例として、通信インターフェイス４１８は、互換性のあるＬＡＮにデータ通信接続を与えるためのローカルエリアネットワーク（ＬＡＮ）カードであり得る。無線リンクも実現され得る。いずれのそのような実現例においても、通信インターフェイス４１８は、さまざまな種類の情報を表わすデジタルデータストリームを搬送する電気、電磁または光学信号を送受信する。

【００６５】

ネットワークリンク４２０は典型的に、１つ以上のネットワークを通じて他のデータ装置にデータ通信を与える。たとえば、ネットワークリンク４２０は、ローカルネットワーク４２２を通じてホストコンピュータ４２４に、またはインターネットサービスプロバイダ（ＩＳＰ）４２６によって操作されるデータ機器に接続を与え得る。ＩＳＰ４２６は次に、現在一般的に「インターネット」４２８と称される広域パケットデータ通信網を通じてデータ通信サービスを与える。ローカルネットワーク４２２およびインターネット４２８は両方とも、デジタルデータストリームを搬送する電気、電磁または光学信号を用いる。さまざまなネットワークを通る信号、ならびにコンピュータシステム４００との間でデジタルデータを搬送し合うネットワークリンク４２０上のおよび通信インターフェイス４１８を通る信号は、情報を移送する搬送波の例示的な形態である。

【００６６】
コンピュータシステム４００は、ネットワーク、ネットワークリンク４２０および通信インターフェイス４１８を通じて、メッセージを送り、プログラムコードを含むデータを受信することができる。インターネットの例では、サーバ４３０が、インターネット４２８、ＩＳＰ４２６、ローカルネットワーク４２２および通信インターフェイス４１８を通じてアプリケーションプログラムについての要求コードを送信し得る。

【００６７】
受けたコードは受信時にプロセッサ４０４によって実行され得、および／または後で実行するために記憶装置４１０もしくは他の不揮発性記憶装置に記憶される。この態様で、コンピュータシステム４００は、搬送波の形態のアプリケーションコードを得ることができる。

【００６８】
上記の明細書では、本発明の実施例が、実現例ごとに異なり得る多数の特定的な詳細を参照して説明された。したがって、発明であるもの、および出願人によって発明であると意図されるものを唯一および独占的に示すものは、いずれの後の訂正も含む、この出願から生じる一組の請求項であり、そのような請求項が生じる特定的な形態におけるものである。そのような請求項に含まれる用語について本明細書中において明示的に記載されるいずれの定義も、請求項において用いられるようなそのような用語の意味を決定するものとする。したがって、請求項において明示的に列挙されていない限定、要素、性質、特徴、利点または属性は、そのような請求項の範囲を如何なる意味でも限定するものではない。明細書および図面はしたがって、限定的ではなく例示的に認識されるべきである。

【図１】

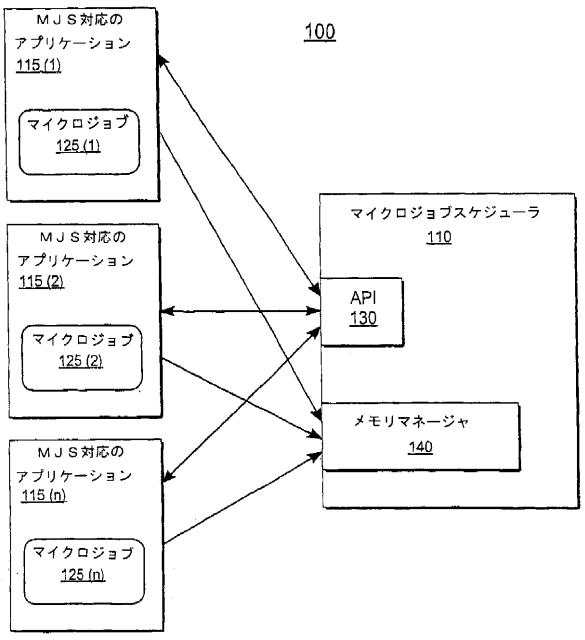

Fig. 1

【図２】

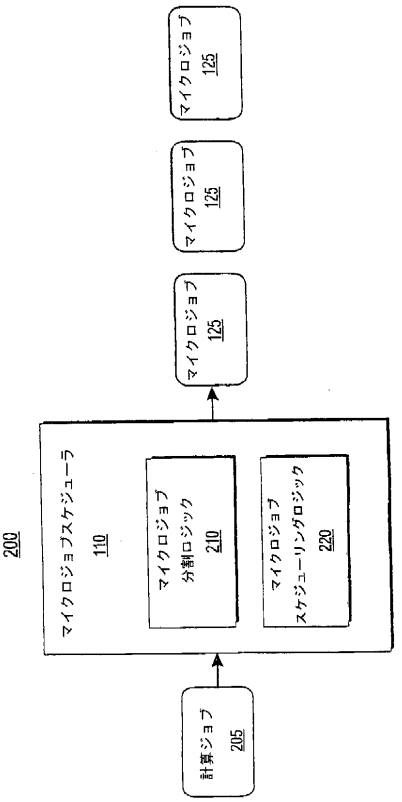

Fig. 2

【図３】

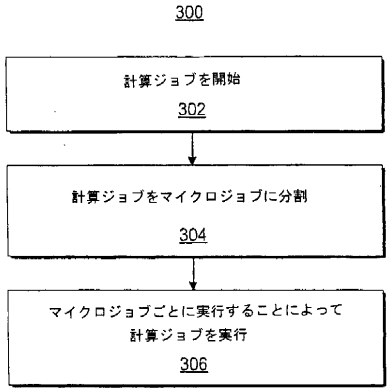

Fig. 3

【図４】

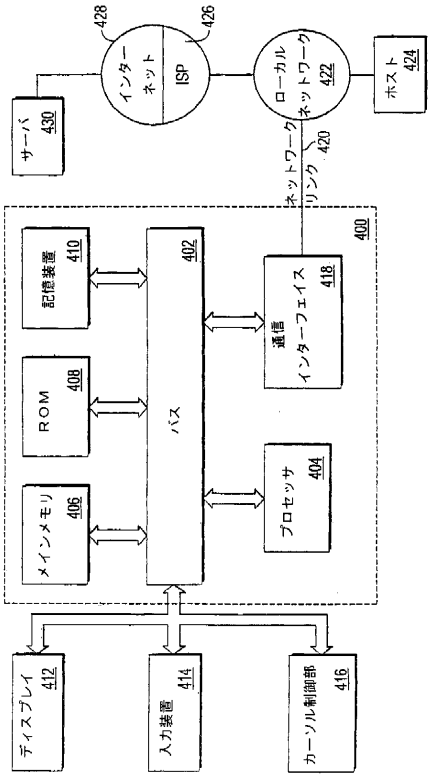

Fig. 4

フロントページの続き

(74)代理人 100096781
弁理士 堀井 豊
(74)代理人 100098316
弁理士 野田 久登
(74)代理人 100111246
弁理士 荒川 伸夫
(72)発明者 ジェンセン, クレイグ
アメリカ合衆国、３３７５６ フロリダ州、クリアウォーター、エス・フォート・ハリソン・アベニュ、６１１、ナンバー・３５７
(72)発明者 スタファー, アンドリュー
アメリカ合衆国、９１３４２ カリフォルニア州、シルマー、アルタ・ビスタ・ウェイ、１３２７０
(72)発明者 トマス, バジル
アメリカ合衆国、９１３４２ カリフォルニア州、シルマー、ウィロウグリーン・レーン、１４５３７
(72)発明者 キャドルビ, リチャード
アメリカ合衆国、９３０６５ カリフォルニア州、シミ・バレー、ソナタ・ウェイ、５３０、ナンバー・ディ

審査官 田中 幸雄

(56)参考文献 特開平０３－０８５６４２（ＪＰ，Ａ）
Tiago Ferreto et al, Scheduling Divisible Workloads Using the Adaptive Time Factoring Algorithm, Lecture Notes in Computer Science, ドイツ, Springer-Verlag, ２００５年, Vol. 3719, pages232-239, Distributed and Parallel Computing

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ ９／４８