

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第5005191号

(P5005191)

(45) 発行日 平成24年8月22日 (2012. 8. 22)

(24) 登録日 平成24年6月1日 (2012. 6. 1)

(51) Int. Cl.

F I

G 0 6 F 9/46 (2006. 01)

G 0 6 F 9/46 3 5 0

G 0 6 F 9/48 (2006. 01)

G 0 6 F 9/46 4 5 2 Z

請求項の数 9 (全 18 頁)

(21) 出願番号 特願2005-171365 (P2005-171365)
 (22) 出願日 平成17年6月10日 (2005. 6. 10)
 (65) 公開番号 特開2006-18820 (P2006-18820A)
 (43) 公開日 平成18年1月19日 (2006. 1. 19)
 審査請求日 平成20年5月8日 (2008. 5. 8)
 (31) 優先権主張番号 10/882, 979
 (32) 優先日 平成16年6月30日 (2004. 6. 30)
 (33) 優先権主張国 米国 (US)

(73) 特許権者 500046438
 マイクロソフト コーポレーション
 アメリカ合衆国 ワシントン州 9805
 2-6399 レッドモンド ワン マイ
 クロソフト ウェイ
 (74) 代理人 100077481
 弁理士 谷 義一
 (74) 復代理人 100115624
 弁理士 濱中 淳宏
 (74) 復代理人 100130524
 弁理士 藤田 英治
 (74) 代理人 100088915
 弁理士 阿部 和夫

最終頁に続く

(54) 【発明の名称】 仮想マシン環境においてオペレーティングシステムを実施するためのシステムおよび方法

(57) 【特許請求の範囲】

【請求項 1】

コンピュータ上の仮想マシン環境下で実行する複数のオペレーティングシステムによりオーバーヘッドを低減させるための方法であって、

前記コンピュータが、前記複数のオペレーティングシステムとホストオペレーティングシステムとの間で共有されるように構成されている記憶手段を備えており、

前記コンピュータは、

前記各オペレーティングシステムが仮想マシン上で実行しているかどうかを判定すること、

前記各オペレーティングシステムが仮想マシン上で実行している場合は、当該各オペレーティングシステムが前記仮想マシン上でより効率的に動作するように、前記ホストオペレーティングシステムに制御を渡すことなく前記各オペレーティングシステムが前記記憶手段を介して前記ホストオペレーティングシステムに対して当該動作に関する情報を送出することであって、前記動作に関する情報は、スレッドの優先順位、または、スレッドの予想実行時間に関する情報であること、および

前記ホストオペレーティングシステムが前記各オペレーティングシステムからの情報に基づいて前記オペレーティングシステム全体の制御を決定すること

を含むことを特徴とする方法。

【請求項 2】

前記動作に関する情報は、合成命令であることを特徴とする請求項 1 に記載の方法。

10

20

【請求項 3】

前記ホストオペレーティングシステムが、前記情報に基づいて前記コンピュータ内のリソースを効率的に割り当てる少なくとも 1 つのスケジューラを含むことを特徴とする請求項 1 に記載の方法。

【請求項 4】

仮想マシン環境下で実行するオペレーティングシステムによりオーバーヘッドを低減させるためのシステムであって、

前記システムが、前記複数のオペレーティングシステムとホストオペレーティングシステムとの間で共有されるように構成されている記憶手段を備えており、

前記システムは、

前記各オペレーティングシステムが前記仮想マシン上で動作しているかどうかを判定すること、

前記各オペレーティングシステムが前記仮想マシン上で実行している場合は、当該各オペレーティングシステムが前記仮想マシン上でより効率的に動作するために、前記ホストオペレーティングシステムに制御を渡すことなく前記オペレーティングシステムが前記記憶手段を介して前記ホストオペレーティングシステムに対して当該動作に関する情報を送出することであって、前記動作に関する情報は、スレッドの優先順位、または、スレッドの予想実行時間に関する情報であること、および

前記ホストオペレーティングシステムが前記各オペレーティングシステムからの情報に基づいて前記オペレーティングシステム全体の制御を決定すること

を含むことを特徴とするシステム。

【請求項 5】

前記動作に関する情報は、合成命令であることを特徴とする請求項 4 に記載のシステム。

【請求項 6】

前記ホストオペレーティングシステムが、前記情報に基づいて前記コンピュータ内のリソースを効率的に割り当てる少なくとも 1 つのスケジューラを含むことを特徴とする請求項 4 に記載のシステム。

【請求項 7】

コンピュータ上の仮想マシン環境下で実行する複数のオペレーティングシステムによりオーバーヘッドを低減させるためのコンピュータ可読命令を含むコンピュータ可読記憶媒体であって、

前記コンピュータが、前記複数のオペレーティングシステムとホストオペレーティングシステムとの間で共有されるように構成されている記憶手段を備えており、

前記コンピュータに、

前記各オペレーティングシステムが仮想マシン上で実行しているかどうかを判定すること、

前記各オペレーティングシステムが仮想マシン上で実行している場合は、当該各オペレーティングシステムが前記仮想マシン上でより効率的に動作するように、前記ホストオペレーティングシステムに制御を渡すことなく前記各オペレーティングシステムが前記記憶手段を介して前記ホストオペレーティングシステムに対して当該動作に関する情報を送出することであって、前記動作に関する情報は、スレッドの優先順位、または、スレッドの予想実行時間に関する情報であること、および

前記ホストオペレーティングシステムが前記各オペレーティングシステムからの情報に基づいて前記オペレーティングシステム全体の制御を決定すること

を実行させるためのコンピュータ可読命令を含むコンピュータ可読記憶媒体。

【請求項 8】

前記動作に関する情報は、合成命令であることを特徴とする請求項 7 に記載のコンピュータ可読記憶媒体。

【請求項 9】

10

20

30

40

50

前記ホストオペレーティングシステムが、前記情報に基づいて前記コンピュータ内のリソースを効率的に割り当てる少なくとも1つのスケジューラを含むことを特徴とする請求項7に記載のコンピュータ可読記憶媒体。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、一般に、仮想マシン（「プロセッサ仮想化」としても知られる）、および仮想マシン環境において実行されるソフトウェアの分野に関する。より具体的には、本発明は、自らが仮想マシン環境で実行されているかどうかを確認することができ、それに相応して自らの挙動を変更することができるオペレーティングシステムを実施するためのシステムおよび方法に関する。

10

【背景技術】

【0002】

コンピュータは、特定のシステム命令セットを実行するように設計されたCPU群（汎用中央処理装置群）を含む。類似したアーキテクチャまたは設計仕様を有する一群のプロセッサは、同一のプロセッサファミリのメンバであると見なすことができる。現在のプロセッサファミリの例には、アリゾナ州フェニックス所在のモトローラ株式会社によって製造されるMotorola 680X0、カリフォルニア州サニーバールのインテル株式会社によって製造されるIntel 80X86プロセッサファミリ、およびモトローラ株式会社によって製造され、カリフォルニア州キューパティーノ所在のアップルコンピュータ株式会社によって製造されるコンピュータにおいて使用されるPowerPCプロセッサファミリが含まれる。一群のプロセッサは、類似したアーキテクチャおよび設計上の配慮（design considerations）のために同一ファミリに入っている可能性があるが、プロセッサは、プロセッサのクロック速度、およびその他の性能パラメータに従ってファミリ内で大きく異なる可能性がある。

20

【0003】

各ファミリのマイクロプロセッサは、そのプロセッサファミリに固有である命令群を実行する。プロセッサ、またはプロセッサのファミリが実行することができる全体的な命令セットが、プロセッサの命令セットとして知られる。例として、Intel 80X86プロセッサファミリによって使用される命令セットは、PowerPCプロセッサファミリによって使用される命令セットと互換性がない。Intel 80X86命令セットは、CISC（Complex Instruction Set Computer）フォーマットに基づく。Motorola PowerPC命令セットは、RISC（Reduced Instruction Set Computer）フォーマットに基づく。CISCプロセッサ群は、一部はかなり複雑な機能を実行することができるが、実行するのに多くのクロックサイクルを一般に要する、多数の命令を使用する。RISCプロセッサ群は、より少数の利用可能な命令を使用して、はるかに高速で実行されるより単純な機能セットを実行する。

30

【0004】

コンピュータシステム間におけるプロセッサファミリの固有性は、通常、コンピュータシステムのハードウェアアーキテクチャのその他の要素間における非互換性ももたらす。Intel 80X86プロセッサファミリからのプロセッサを使用して製造されたコンピュータシステムは、PowerPCプロセッサファミリからのプロセッサを使用して製造されたコンピュータシステムのハードウェアアーキテクチャとは異なるハードウェアアーキテクチャを有する。プロセッサ命令セットおよびコンピュータシステムのハードウェアアーキテクチャの固有性のため、アプリケーションソフトウェアプログラム群は、通常、特定のオペレーティングシステムを実行する特定のコンピュータシステム上で実行されるように書かれる。

40

【0005】

コンピュータ製造業者は、コンピュータ製造業者の製品系列に関連するマイクロプロセ

50

ッサファミリ上で、より少ないアプリケーションではなく、より多くのアプリケーションを実行させることにより、自らの市場占有率を最大化することを望む。コンピュータシステム上で実行することができるオペレーティングシステムおよびアプリケーションプログラムの数を拡張するため、ホストと呼ばれるあるタイプのCPUを有する所与のコンピュータが、ゲストと呼ばれる関連のないタイプのCPUの命令をホストコンピュータがエミュレートすることを可能にするエミュレータプログラムを含むようにする技術分野が開発されている。このため、ホストコンピュータは、所与のゲスト命令に応答して1つまたは複数のホスト命令が呼び出されるようにするアプリケーションを実行する。このため、ホストコンピュータは、ホストコンピュータ独自のハードウェアアーキテクチャ向けに設計されたソフトウェアと、関連のないハードウェアアーキテクチャを有するコンピュータ向けに書かれたソフトウェアをともに実行することができる。より具体的な例として、例えば、アップルコンピュータによって製造されたコンピュータシステムは、PCベースのコンピュータシステム向けに書かれたオペレーティングシステム群およびプログラムを実行することができる。また、エミュレータプログラムを使用して、単一のCPU上で互換性のないオペレーティングシステム群を同時に動作させることもできる可能性がある。この構成では、各オペレーティングシステムは、他方のオペレーティングシステムと互換性がないが、エミュレータプログラムが、2つのオペレーティングシステムの1つをホストし、そうでなければ互換性のないオペレーティングシステム群が、同一コンピュータシステム上で同時に実行されることを可能にすることができる。

10

【0006】

20

ゲストコンピュータシステムがホストコンピュータシステム上でエミュレートされる場合、ゲストコンピュータシステムは、「仮想マシン」といわれる。というのは、ゲストコンピュータシステムは、1つの特定のハードウェアアーキテクチャの動作の純粋なソフトウェア表現としてだけホストコンピュータ上に存在するからである。エミュレータ、仮想マシン、およびプロセッサエミュレーションという用語は、ときとして、互いに区別なく使用されて、コンピュータシステム全体のハードウェアアーキテクチャを模倣する、つまりエミュレートする能力を表す。例として、カリフォルニア州サンマテオ所在のコネクティクス（Connectix）株式会社によって作成されたVirtual PCソフトウェアは、Intel 80X86 Pentium（登録商標）プロセッサ、および様々なマザーボードコンポーネントおよびカードを含むコンピュータ全体をエミュレートする。それらのコンポーネントの動作は、ホストマシン上で実行される仮想マシンにおいてエミュレートされる。PowerPCプロセッサを有するコンピュータシステムなどのホストコンピュータのオペレーティングシステムソフトウェア上、およびハードウェアアーキテクチャ上で実行されるエミュレータプログラムが、ゲストコンピュータシステム全体の動作を模倣する。

30

【0007】

エミュレータプログラムは、ホストマシンのハードウェアアーキテクチャと、エミュレートされる環境内で実行されるソフトウェアによって伝送される命令の間のやりとりとして作用する。このエミュレータプログラムは、物理的なコンピュータハードウェア上で直接に実行されるオペレーティングシステムである、HOS（ホストオペレーティングシステム）であることが可能である。代替として、エミュレートされる環境も、ハードウェアの上で直接に実行され、VMM（仮想マシンモニタ）が仮想化しているハードウェアと同一であるインターフェース群を公開することにより、マシンのすべてのリソースを仮想化するソフトウェアレイヤである、VMMであることも可能である（これにより、VMMが、VMMの上で実行されているオペレーティングシステムレイヤ群に気付かれないことが可能になる）。ホストオペレーティングシステムとVMMは、同一の物理的ハードウェア上で並行に実行されることが可能である。

40

【0008】

現在の仮想マシンソフトウェア（マイクロソフト株式会社によって販売されるVirtual ServerやVirtual PCなどの）は、前述した仮想化を可能にする

50

が、仮想化を可能にすることに関連する相当なパフォーマンスオーバーヘッドが存在する。パフォーマンスオーバーヘッドは、特に、重いI/O作業負荷（多量のディスクアクセスまたはネットワーク通信で）を有するソフトウェアアプリケーション群において、高い場合には70%のレベルに達する可能性がある。このレベルのオーバーヘッドは、最大限のプロセッサ速度を要するアプリケーション群において許容できない。必要とされているのは、仮想マシン環境においてプロセッサオーバーヘッドを低減する方法である。

【発明の開示】

【発明が解決しようとする課題】

【0009】

慣用のOS（オペレーティングシステム）では、一部のOS活動は、オペレーティングシステムが専用の物理的ハードウェア上で実行されているという前提で実行される。仮想マシン環境では、それらの活動は、同一の物理的ハードウェア上で同時に実行されているその他のゲストOS群に有害である可能性がある。それらの有害な活動は、オペレーティングシステムリソース群（仮想環境においてではなく、専用の物理的ハードウェア上で実行されるように設計された）を拘束する（tie up）可能性がある。というのは、オペレーティングシステムは、ハードウェアがオペレーティングシステムに専用であると想定し、そのリソース群を使用している、または使用するのを待っている他のオペレーティングシステム群についてまったく知らないからである。必要とされているのは、ゲストOSの挙動を変更して、その挙動が、仮想マシン環境において実行されている他のゲストOS群に有害でないようにする方法である。

【課題を解決するための手段】

【0010】

本発明は、自らが仮想マシン環境で動作しているかどうかを確認することができ、仮想マシン環境においてより効率的に動作するように自らの挙動を変更することがさらにできるオペレーティングシステムを実施するためのシステム、および実施する方法を含む。

【0011】

本発明の諸実施形態は、自らが仮想マシン環境において動作していることを認識し、この認識の結果、仮想マシン環境において従来、問題となってきたパフォーマンスオーバーヘッドのいくらかを低減することができるオペレーティングシステムを提供するためのシステム、および提供する方法を対象とする。ホストオペレーティングシステムとゲストオペレーティングシステム群の間における共有通信領域の導入により、ホストとゲストの間でコンピュータの制御を受け渡しすることなしに、ゲストとホストの間で通信のメカニズムが提供される。ゲストオペレーティングシステム群とホストの間で受け渡しされるタイプの通信の一例は、スレッドスケジューリングに関連する。ゲストがVM環境において動作していることを認識し、および共有通信領域が導入されて、ゲストオペレーティングシステム群は、追加の情報（実行優先順位などの）をホストオペレーティングシステムに送り、これにより、ホストオペレーティングシステムが、より効率的なスレッドスケジューリング決定を行うことが可能になる。というのは、ホストが、プロセッサ時間の全体的な要求に関してさらなる情報を有するからである。

【0012】

以上の概要、および好ましい諸実施形態の以下の詳細な説明は、添付の図面と併せて読むことで、よりよく理解される。本発明を例示する目的で、図面には、本発明の典型的な構成が示されているが、本発明は、開示する特定の方法および手段（instrumentalities）に限定されない。

【発明を実施するための最良の形態】

【0013】

本発明の対象物を、法的要件を満たすように具体的に説明する。ただし、説明自体は、本特許の範囲を限定するものではない。むしろ、本発明者は、請求の対象物が他の形で実施されて、他の現在の技術、または将来の技術と連携して、本明細書で説明するステップと異なるステップ、または類似するステップの組合せを含むことも可能であることを企図

している。さらに、「ステップ」という用語は、使用する方法の異なる要素を表すように本明細書で使用される可能性があるが、個々のステップの順序を明示的に説明しない限り、またそうする場合を除き、この用語が、本明細書で開示する様々なステップの間における特定の順序を暗示しているものと解釈してはならない。

【0014】

コンピュータ環境

本発明の多数の実施形態が、コンピュータ上で実行されることが可能である。図1、および以下の説明は、本発明を実施することができる適切なコンピューティング環境の簡単な一般的説明を提供することを目的とする。必須ではないが、本発明は、クライアントワークステーションまたはサーバなどのコンピュータによって実行される、プログラムモジュールなどの、コンピュータ実行可能命令の一般的な文脈で説明する。一般に、プログラムモジュールには、特定のタスクを実行する、または特定の抽象データ型を実装するルーチン、プログラム、オブジェクト、コンポーネント、データ構造などが含まれる。さらに、本発明は、ハンドヘルドデバイス、マルチプロセッサシステム、マイクロプロセッサベースの家庭用電化製品またはプログラマブル家庭用電化製品、ネットワークPC、ミニコンピュータ、メインフレームコンピュータなどを含む、他のコンピュータシステム構成を使用して実施してもよいことが当業者には認められよう。また、本発明は、通信網を介してリンクされたりリモート処理装置によってタスクが実行される分散コンピューティング環境において実施してもよい。分散コンピューティング環境では、プログラムモジュール群は、ローカルメモリ記憶装置とリモートメモリ記憶装置の両方の中に配置することができる。

【0015】

図1に示すとおり、典型的な汎用コンピューティングシステムは、処理装置21、システムメモリ22、ならびにシステムメモリから処理装置21までを含む様々なシステムコンポーネントを結合するシステムバス23を含む、慣用のパーソナルコンピュータ20などを含む。システムバス23は、様々なバスアーキテクチャのいずれかを使用するメモリバスまたはメモリコントローラ、周辺バス、およびローカルバスを含め、いくつかのタイプのバス構造のいずれであることも可能である。システムメモリは、ROM（読み取り専用メモリ）24およびRAM（ランダムアクセスメモリ）25を含む。始動中などにパーソナルコンピュータ20内部の要素間で情報を転送するのを助ける基本ルーチンを含むBIOS（基本入出力システム）26が、ROM24の中に格納される。パーソナルコンピュータ20は、図示していないハードディスクに対して読み取りおよび書き込みを行うためのハードディスクドライブ27、リムーバブルな磁気ディスク29に対して読み取りまたは書き込みを行うための磁気ディスクドライブ28、およびCD-ROMまたは他の光媒体などのリムーバブルな光ディスク31に対して読み取りまたは書き込みを行うための光ディスクドライブ30をさらに含むことが可能である。ハードディスクドライブ27、磁気ディスクドライブ28、および光ディスクドライブ30は、それぞれ、ハードディスクドライブインターフェース32、磁気ディスクドライブインターフェース33、および光ドライブインターフェース34でシステムバス23に接続される。以上のドライブ、および関連するコンピュータ可読媒体により、コンピュータ可読命令、データ構造、プログラムモジュール、およびその他のデータの不揮発性ストレージがパーソナルコンピュータ20に提供される。本明細書で説明する典型的な環境は、ハードディスク、リムーバブルな磁気ディスク29、およびリムーバブルな光ディスク31を使用するが、磁気カセット、フラッシュメモリカード、デジタルビデオディスク、ベルヌーイカートリッジ、RAM（ランダムアクセスメモリ）、ROM（読み取り専用メモリ）などの、データを格納することができ、コンピュータがアクセス可能な他のタイプのコンピュータ可読媒体も、典型的な動作環境において使用できることが当業者には認められよう。

【0016】

オペレーティングシステム35、1つまたは複数のアプリケーションプログラム36、その他のプログラムモジュール群37、およびプログラムデータ38を含め、いくつかの

プログラムモジュールが、ハードディスク、磁気ディスク 29、光ディスク 31、ROM 24、または RAM 25 に格納されることが可能である。ユーザは、キーボード 40 やポインティングデバイス 42 などの入力デバイス群を介して、コマンドおよび情報をパーソナルコンピュータ 20 に入力することができる。その他の入力デバイス群（図示せず）には、マイク、ジョイスティック、ゲームパッド、サテライトディスク、スキャナなどが含まれることが可能である。以上、およびその他の入力デバイス群は、しばしば、システムバスに結合されたシリアルポートインターフェース 46 を介して処理装置 21 に接続されるが、パラレルポート、ゲームポート、または USB（ユニバーサルシリアルバス）などの他のインターフェース群で接続してもよい。モニタ 47、またはその他のディスプレイデバイスも、ビデオアダプタ 48 などのインターフェースを介してシステムバス 23 に接続される。モニタ 47 に加え、パーソナルコンピュータは、通常、スピーカやプリンタなどの他の周辺出力デバイス（図示せず）も含む。図 1 の典型的なシステムは、ホストアダプタ 55、SCSI（スモールコンピュータシステムインターフェース）バス 56、ならびに SCSI バス 56 に接続された外部記憶装置 62 も含む。

10

【0017】

パーソナルコンピュータ 20 は、リモートコンピュータ 49 のような 1 つまたは複数のリモートコンピュータに対する論理接続を使用するネットワーク化された環境において動作することができる。リモートコンピュータ 49 は、別のパーソナルコンピュータ、サーバ、ルータ、ネットワーク PC、ピアデバイス、または他の一般的なネットワークノードであることが可能であり、通常、パーソナルコンピュータ 20 に関連して前述した諸要素の多く、またはすべてを含むが、メモリ記憶装置 50 だけを図 1 に示す。図 1 に示す論理接続は、LAN（ローカルエリアネットワーク）51 および WAN（ワイドエリアネットワーク）52 を含む。そのようなネットワーキング環境は、オフィス、企業全体のコンピュータ網、イントラネット、およびインターネットで一般的である。

20

【0018】

LAN ネットワーキング環境で使用される場合、パーソナルコンピュータ 20 は、ネットワークインターフェースまたはネットワークアダプタ 53 を介して LAN 51 に接続される。WAN ネットワーキング環境で使用される場合、パーソナルコンピュータ 20 は、通常、インターネットなどのワイドエリアネットワーク 52 を介して通信を確立するためのモデム 54、または他の手段を含む。内部にあることも、外部にあることも可能なモデム 54 は、シリアルポートインターフェース 46 を介してシステムバス 23 に接続される。ネットワーク化された環境では、パーソナルコンピュータ 20 に関連して示したプログラムモジュール群、またはプログラムモジュール群の諸部分は、リモートメモリ記憶装置の中に格納することができる。図示したネットワーク接続は、典型的であり、コンピュータ間で通信リンクを確立する他の手段も使用できることが認められよう。さらに、本発明の多数の実施形態は、コンピュータ化されたシステムに特によく適していることが構想されているが、本発明をそのような諸実施形態に限定することを意図するものはまったく本明細書に存在しない。

30

【0019】**仮想マシン**

40

概念的に見ると、コンピュータシステムは、一般に、基礎的なハードウェアレイヤ上で実行される 1 つまたは複数のソフトウェアレイヤを含む。この階層化（layering）は、抽象化のために行われる。所与のソフトウェアレイヤに関するインターフェースを定義することにより、そのレイヤより上位の他のレイヤ群とは異なる形でそのレイヤが実施されることが可能である。うまく設計されたコンピュータシステムでは、各レイヤは、そのレイヤの 1 つ下のレイヤについてだけ知っている（またそのようなレイヤだけに依拠する）。これにより、レイヤ、または「スタック」（複数の隣接するレイヤ）を、前記レイヤまたは前記スタックより上位のレイヤ群に悪影響を与えることなしに、置き換えることが可能になる。例えば、ソフトウェアアプリケーション群（上位レイヤ群）は、通常、オペレーティングシステムの下位レベル（下位レイヤ）に依拠して、何らかの形態の永久

50

ストレージにファイルを書き込み、これらのアプリケーションは、フロッピー（登録商標）ディスク、ハードドライブ、またはネットワークフォルダにデータを書き込むことの間の違いを理解する必要がない。その下位レイヤが、ファイルを書き込むための新たなオペレーティングシステムコンポーネント群で置き換えられた場合、上位レイヤのソフトウェアアプリケーション群の動作は、影響を受けないままである。

【0020】

階層化されたソフトウェアの柔軟性により、VM（仮想マシン）が、実際には別のソフトウェアレイヤである仮想ハードウェアレイヤを提示することが可能になる。このようにして、VMは、VMより上位のソフトウェアレイヤ群に、前記ソフトウェアレイヤ群が独自の専用コンピュータシステム上で実行されているという錯覚をもたらすことができ、このため、VMは、複数の「ゲストシステム」が単一の「ホストシステム」上で同時に実行されることを可能にすることができる。

10

【0021】

図2は、コンピュータシステムにおけるエミュレートされた動作環境に関するハードウェアおよびソフトウェアのアーキテクチャの論理的な階層化を示す図である。エミュレーションプログラム94が、ホストオペレーティングシステムおよび／またはハードウェアアーキテクチャ92上で実行される。エミュレーションプログラム94は、ゲストハードウェアアーキテクチャ96およびゲストオペレーティングシステム98をエミュレートする。ソフトウェアアプリケーション100が、ゲストオペレーティングシステム98上で実行される。図2のエミュレートされた動作環境では、エミュレーションプログラム94が動作しているため、ソフトウェアアプリケーション100は、ホストオペレーティングシステムおよびハードウェアアーキテクチャ92と全体的に互換性のないオペレーティングシステム上で実行されるように設計されているものの、コンピュータシステム90上で実行されることが可能である。

20

【0022】

図3Aは、物理的コンピュータハードウェア102のすぐ上で実行されるホストオペレーティングシステムソフトウェアレイヤ104を含む、仮想化されたコンピューティングシステムを示し、ホストOS（ホストオペレーティングシステム）104は、ホストOSが仮想化しているハードウェアと同一であるインターフェース群を公開することにより、マシンのすべてのリソースを仮想化する（これにより、ホストOSが、ホストOS上で実行されているオペレーティングシステムレイヤ群に気付かれないことが可能になる）。

30

【0023】

代替として、仮想マシンモニタソフトウェアレイヤ、つまりVMMソフトウェアレイヤ104'が、ホストOS104"の代わりに、または並行に実行されていることも可能であり、並行に実行されているオプションを図3Bに示す。簡明にするため、以降のすべての説明（具体的には、ホストOS104に関する）は、図3Aに示した実施形態を対象とするが、そのような説明のすべての態様は、図3BのVMM104'が、以下に説明する図3AのホストOS104の役割に、機能レベルで基本的にとって代わる、図3Bの実施形態にも同様に適用されることになる。

【0024】

40

図3に関して、VM A108およびVM B110は、ソフトウェア構成体としてだけ存在する仮想化されたコンピュータハードウェア表現であり、この表現は、ゲストOS A112およびゲストOS B114にそれぞれ、VM A108およびVM B110を提示するだけでなく、ゲストOS A112およびゲストOS B114に必要なソフトウェアステップのすべてを実行して、現実の物理的コンピュータハードウェア102と間接的に対話することも行う専用ソフトウェアコードが存在しているために可能になることに留意することが重要である。この完全な機能は、全体的に、VMM（仮想マシンモニタ）（図3Bだけに示す）と呼ぶことができ、一部の実施形態（図3Aに示した実施形態のような）に関して、VMMは、ホストオペレーティングシステム104の一部を含む。しかし、他の諸実施形態（図示せず）では、VMMは、ホストオペレーティングシステ

50

ム104の上で実行され、前記ホストオペレーティングシステム104を介してだけコンピュータハードウェアと対話するアプリケーションであることが可能である。さらに別の諸実施形態（図3Bに示したような）では、VMMは、一部のレベルでは、ホストオペレーティングシステム104を介してコンピュータハードウェア102と間接的に対話する部分的に独立したソフトウェアシステムを含むことが可能であるが、別のレベルでは、VMMは、コンピュータハードウェア102と直接に対話する（ホストオペレーティングシステムがコンピュータハードウェアと直接に対話するのと類似した形で）。さらに別の諸実施形態（図3Bに示した実施形態に類似する）では、VMMは、ホストオペレーティングシステム104を利用することなしに（コンピュータハードウェア102の使用を調整し、競合などを回避する限りにおいて、前記ホストオペレーティングシステム104と、依然として、対話するが）、すべてのレベルでコンピュータハードウェア102と直接に対話する（ホストオペレーティングシステムがコンピュータハードウェアと直接に対話するのと類似した形で）完全に独立したソフトウェアシステムを含むことが可能である。

10

【0025】

VMMを実施するためのこれら変種のすべては、本明細書で説明する本発明の代替の諸実施形態を形成するものと予期され、本発明を何らかの特定のVMM構成に限定するものと解釈されるべきものはまったく本明細書に存在しない。さらに、VM A108および／またはVM B110をそれぞれ介するアプリケーション116、118、および120の間の対話（おそらく、ハードウェアエミュレーションのシナリオにおける）についてのいずれの叙述も、実際には、アプリケーション116、118、および120とVMMの間の対話であると解釈されなければならない。同様に、アプリケーションVM A108および／またはVM B110とホストオペレーティングシステム104および／またはコンピュータハードウェア102の間の対話（おそらく、コンピュータハードウェア102上で直接に、または間接的にコンピュータ命令群を実行する）についてのいずれの叙述も、実際には、適宜、VMMとホストオペレーティングシステム104またはコンピュータハードウェア102の間の対話であると解釈されなければならない。

20

【0026】

図3Aを再び参照すると、ホストOS104（またはVMM104'）の上には、例えば、仮想化されたIntel 386プロセッサであることが可能なVM A108、および、例えば、Motorola 680X0ファミリのプロセッサの1つの仮想化されたバージョンであることが可能なVM B110という2つの仮想マシン（VM）実施形態が存在する。VM A108およびVM B110のそれぞれの上には、ゲストOS（ゲストオペレーティングシステム）A112およびB114がそれぞれ存在する。ゲストOS A112の上では、アプリケーションA1116およびアプリケーションA2118という2つのアプリケーションが実行されており、ゲストOS B114の上には、アプリケーションB1120が存在する。

30

【0027】

従来、仮想マシンは、実際には、ゲストOS群（例えば、ゲストOS A112）が、他のゲストOS群（例えば、ゲストOS B114）とハードウェアを共用していながら、ゲストOS群が専用ハードウェア上で実行されているという錯覚に基づいてきた。ホストOS104は、この錯覚を保つことを担う。本発明は、この錯覚を緩和し、自らがVMにおいて実行されているかどうかをゲストOS群が確かめ、その後、その認識に基づいて自らの挙動を変更することができるようにする。VM環境において動作するゲストOS群の効率を高める挙動の変更を説明する。

40

【0028】

VM環境において動作すること

図4は、図3Aに示すのと類似する仮想化されたコンピューティングシステムを示すが、図4では、VM認識ゲストOS A132およびVM認識ゲストOS B134が、それぞれ、ゲストOS A112およびゲストOS B114に取って代わっている。VM認識ゲストOS A132およびVM認識ゲストOS B134は、自らが仮想マシン環

50

境において動作しているかどうかを確認することができ、仮想マシン環境において動作している場合、より効率的に動作するように自らの挙動を変更することができるオペレーティングシステムである。

【0029】

図4のVM認識ゲストOS A132およびVM認識ゲストOS B134の動作を、図5を参照して説明する。図5は、仮想マシン環境における効率を向上させるために自らの挙動を変更する能力を有するVM認識オペレーティングシステムを実施する方法140を示す流れ図である。ステップ142で、方法はまず、VM認識オペレーティングシステム（例えば、VM認識ゲストOS A132またはVM認識ゲストOS B134）を開始することを含む。

10

【0030】

ステップ144で、VM認識OSは、自らがVM環境において動作しているかどうかを判定する。この判定は、「SYSTEMS AND METHODS FOR USING SYNTHETIC INSTRUCTIONS IN A VIRTUAL MACHINE」という名称の、2003年10月14日に出願した（米国特許出願公開第10/685051号明細書）（以降、「051特許出願」）において説明する、合成命令群の使用を含む、様々な方法のいずれかによって行われる。「051特許出願は、中央処理装置のIDを表す値を戻すための合成命令（例えば、VMCPUID）を実行することにより、オペレーティングシステムが、仮想化されたプロセッサ上で実行されているか、またはx86プロセッサ上で直接に実行されているかをオペレーティングシステムが判定する方法を説明する。値が戻された場合、ゲストOSは、オペレーティングシステムが仮想化されたプロセッサ上で実行されていると結論し、合成命令に応答して例外が発生した場合、ゲストOSは、オペレーティングシステムがx86プロセッサ上で直接に実行されていると結論する。ゲストOSがVM環境で実行されているかどうかを判定するための別の方法には、一連の試験スレッドを実行して、現在の環境のパフォーマンスを履歴上の結果と比較することが含まれる。いずれにしても、自らがVM環境において動作していないとVM認識OSが判定した場合、方法140は、ステップ146に進む。代替として、自らがVM環境において動作しているとVM認識OSが判定した場合、方法140は、ステップ150に進む。

20

【0031】

ステップ146で、VM認識OSは、「従来の」形で動作する。というのは、VM認識OSは、VM環境においてではなく、専用ハードウェア上で動作しているからである。ステップ148で、VM認識オペレーティングシステムは、「シャットダウン」コマンドが受け取られたかどうかを判定する。「シャットダウン」コマンドが受け取られている場合、VM認識OSは、シャットダウンし、方法140は、終了する。「シャットダウン」コマンドが受け取られていない場合、VM認識OSは、ステップ146で説明したとおり、「従来の」形で動作する。

30

【0032】

ステップ150で、VM認識OSは、VM環境においてより効率的に動作するために自らの挙動を変更する。挙動の変更の実施例には、1) スレッドスケジューリング、2) 「SYSTEMS AND METHODS FOR BIMODAL DEVICE VIRTUALIZATION OF ACTUAL AND IDEALIZED HARDWARE-BASED DEVICES」という名称の、2003年12月12日に出願した米国特許出願第1-734450号明細書（以降、「450特許出願」）において説明する2モードデバイス群を使用して、デバイス群の効率を向上させること、および3) ホストOS104に対して実行優先順位を含む合成命令（「051特許出願で見られるような）を利用することが含まれるが、以上には限定されない。

40

【0033】

ステップ152で、VM認識OSは、「シャットダウン」命令が受け取られているかどうかを判定する。「シャットダウン」命令が受け取られている場合、VM認識OSは、シャットダウンし、方法140は、終了する。「シャットダウン」コマンドが受け取られていない場合、VM認識OSは、ステップ150で説明したとおり、変更された高い効率の

50

モードで動作すること続ける。

【0034】

スケジューリング実施例

オペレーティングシステムにおいて、スケジューラが、プロセッサを実行コンテキスト、つまりスレッドに割り当てる。スケジューラは、すべてのレディ（ready）スレッドを点検し、処理のためにスレッドをスケジュール設定する。行われるべき作業が存在しない場合、スケジューラは、ある期間にわたって作業を探しながらループしてから、最終的に、ビジー（繁忙）待機（busy-wait）ゾーンに入る。オペレーティングシステムが仮想マシンにおいて動作していない場合、この挙動は、パフォーマンスに有害ではない。しかし、仮想マシン環境では、この挙動は、レディであるが、プロセッサが占有されている作業を有する他のゲストオペレーティングシステム群には有害である。このループおよびビジー待機の時間は、仮想マシン環境においてオペレーティングシステム群を実行することに関連する高いオーバーヘッドレベルに寄与する。

10

【0035】

図6に示す実施例は、挙動の変更のための典型的なシステムおよび方法である。ただし、本発明は、スレッドスケーリング挙動の変更に限定されない。

【0036】

図6は、ホストOS 104とVM認識ゲストOS A132の間に構成された共有通信領域A162を含む、仮想化されたコンピューティングシステムを示す。同様に、共有通信領域B164が、ホストOS 104とVM認識ゲストOS B134の間に構成される。この仮想化されたコンピューティングシステムは、ゲストOS群がより効率的に動作する仕方を提供する。スケジューリング実施例において、図6に示す仮想化されたコンピューティングシステムは、ゲストOSが追加のスレッド情報（スレッドの優先順位、およびスレッドが実行される予期される継続時間に関する情報などの）を提供する仕方を含む。その情報が、他のゲストOS群からの情報と結合されると、ホストOSは、システム内のリソース群を求める要求のすべての状況をはるかに明確に知り、したがって、システムの効率を大幅に向上させる決定を行うことができる。

20

【0037】

共有通信領域A162およびB164は、制御をホストOS 104に渡すことなしに、情報をホストOS 104に転送する効率的なやり方をVM認識ゲストOS A132およびVM認識ゲストOS B134に提供するメカニズムである。ホストOS 104に制御を渡すことには、時間がかかり、したがって、全体的なシステムパフォーマンスに有害であり、したがって、可能な場合、回避されるべきである。一実施例では、共有通信領域A162および共有通信領域B164は、共有メモリ空間を使用して実施される。別の実施例では、共有通信領域A162および共有通信領域B164は、それぞれ、VM認識ゲストOS A132とVM認識ゲストOS B134の間の直接通信リンクによって実施される。

30

【0038】

VM認識ゲストOS A132およびVM認識ゲストOS B134は、それぞれ、VM認識スケジューラA166およびVM認識スケジューラB168をさらに含む。VM認識スケジューラA166およびB168は、実行プロパティ（優先順位、期限、およびリザーブ（スレッドに割り当てられたプロセッサの一部分））を有するレディスレッドのセットを操作する。実行プロパティは、共有通信領域A162およびB164に入れられる。

40

【0039】

図6に示した仮想化されたコンピューティングシステムは、ホストOS 104内部のホストスケジューラ172をさらに含む。ホストスケジューラ172は、システムの次の2つの新たな特徴に基づき、より効率的なスケジューリング決定を行う。すなわち、第1に、VM認識ゲストOS A132およびVM認識ゲストOS B134が、自らがVM環境において動作していることを認識し、実行優先順位をホストスケジューラ172に送っ

50

て、複数のVM認識ゲストOSからのすべてのスレッドのより効率的な実行を可能にすることができ、第2に、共有通信領域A162およびB164が、制御をホストOS104に渡すことなしに、情報をホストOS104に送る効率的なやり方を提供する。

【0040】

ホストスケジューラ172は、共有通信領域A162およびB164に入れられた実行優先順位に従って、スレッドを処理するように複数の仮想プロセッサ170A~170Nを割り当てる。ホストスケジューラ172は、すべてのVM認識ゲストOS（例えば、VM認識ゲストOS A132およびVM認識ゲストOS B134）に関して実行優先順位を点検し、すべてのVM認識ゲストOSからの優先順位に基づいて合成実行リストを作成し、それに応じてスレッドを処理するように仮想プロセッサ群170を割り当てる。

10

【0041】

図6のVM認識ゲストOS群の動作を、図7を参照して説明する。図7は、本発明に従ってオペレーティングシステムにおいてスレッドをスケジュール設定する典型的な方法180を示す流れ図である。ステップ182で、方法は、VM認識ゲストOS A132がVM環境において動作しているかどうかをVM認識ゲストOS A132が判定することで始まる。この判定は、「051出願（前述した）において説明される合成命令群の使用を含むが、それには限定されない様々な方法のいずれかによって行われる。「はい」である場合、方法180は、ステップ192に進み、「いいえ」である場合、方法180は、ステップ184に進む。

【0042】

20

専用ハードウェア上のオペレーティングシステムの既定の動作モードであるステップ184で、VM認識ゲストOS A132は、スレッドを処理する。ステップ186で、VM認識ゲストOS A132は、さらなるスレッドが処理される準備ができているかどうかを判定する。「はい」である場合、方法180は、ステップ184に戻り、「いいえ」である場合、方法180は、ステップ188に進む。ステップ188で、VM認識ゲストOS A132は、「シャットダウン」コマンドが受け取られているかどうかを判定し、受け取られている場合、VM認識ゲストOS A132は、シャットダウンし、方法180は、終了する。「シャットダウン」コマンドが受け取られていない場合、方法180は、ステップ190に進む。ステップ190で、VM認識ゲストOS A132は、処理されるべきさらなるスレッドを待つ間、ビジー待機ステータスに入り、所定の時間後、方法180は、ゲストOS A132が処理すべきさらなるスレッドがないか調べるようにステップ186に戻る。

30

【0043】

ステップ192で、現時点では、拡張「VM認識」モードで動作しているVM認識ゲストOS A132は、スレッドを処理することで開始し、次に、ステップ194で、VM認識ゲストOS A132は、さらなるスレッドが処理される準備ができているかどうかを判定する。「はい」である場合、方法180は、さらなる処理のためにステップ192に戻る。準備ができていない場合、方法180は、ステップ195に進む。ステップ195で、ゲストOS A132は、「シャットダウン」コマンドが受け取られているかどうかを判定し、受け取られている場合、ゲストOS Aは、シャットダウンし、方法180は、終了する。受け取られていない場合は、ステップ196で、VM認識ゲストOS A132は、ゲストOS A132が、現在、行うべき作業を有さない（つまり、処理されるべきスレッドを有さない）ことをホストOS 104に（または、一部の代替の実施形態の場合、VMM104'に）示す。一実施例では、この指示は、前述したとおり、共有通信領域A162を介してVM認識ゲストOS A132によって送られる。別の実施例では、この指示は、VM認識ゲストOS A132が、合成命令をホストOS104（もちろん、前記合成命令を理解するようにプログラミングされた）に送り、次に、ステップ198で、他のVM群からの作業が、処理される準備ができているかどうかをホストOS104が判定することによって実行される。

40

【0044】

50

VM認識ゲストOS A132が、ステップ196で、自らがまったく作業を有さないことをホストOS104に示した（例えば、共有通信領域B132を介して、または合成命令を介して）場合、ホストOS104は、別の場所でプロセッサリソース群の必要が存在するかどうかを判定し、存在しない場合、ホストOS104は、ゲストOS A132が、プロセッサリソース群を受け取りつづけることができるようにし、これにより、基本的にゲストOS A132は、ゲストOS A132が、現在、行うべき作業をまったく有さなくても、処理を続けることができるようになり、このため、ゲストOS A132は、ステップ199でビジー待機ループに入ってから、ステップ194に戻り、実行されるべきさらなるスレッドが存在するかどうかを確かめる。他方、プロセッサリソース群の必要性が別の場所で存在するとホストOS104が判定した場合、ステップ200で、ホストOS104は、ゲストOS A132向けのプロセッサリソース群を取り上げ、そのリソース群を、ホストOS104上で実行されている別のプロセス（別のVMおよびゲストオペレーティングシステム、例えば、ゲストOS B134などの）に一時的に与えてから、そのリソース群がホストOS104に返され、ホストOS104がゲストOS A132にプロセッサリソース群を再び提供するまで、ゲストOS A132を実質的に一時停止させ、返された時点で、ゲストOS A132は、オプションとして、ビジー待機状態（この状態は、一部の代替の実施形態では飛ばされる）を続けてから、ステップ194に戻る。

【0045】

結論

本明細書で説明する様々なシステム、方法、および技術は、ハードウェアで、またはソフトウェアで、あるいは、適切な場合、ハードウェアとソフトウェアの組合せで実施することができる。このため、本発明の方法および装置、または方法および装置のいくつかの態様または部分は、フロッピー（登録商標）ディスク、CD-ROM、ハードドライブ、または他の任意のマシン可読記憶媒体などの実体のある媒体で実体化されたプログラムコード（すなわち、命令群）の形態をとることができ、プログラムコードは、コンピュータなどのマシンに読み込まれ、マシンによって実行されると、そのマシンが、本発明を実施するための装置になる。プログラマブルコンピュータ上のプログラムコード実行のケースでは、コンピュータは、一般に、プロセッサ、プロセッサが読み取り可能な記憶媒体（揮発性メモリおよび不揮発性メモリ、および／または記憶要素を含む）、少なくとも1つの入力デバイス、および少なくとも1つの出力デバイスを含む。1つまたは複数のプログラムが、好ましくは、高級手続き言語、またはオブジェクト指向プログラミング言語で実装されて、コンピュータシステムと通信する。ただし、プログラムは、所望される場合、アセンブリ言語または機械語で実装することもできる。いずれにしても、言語は、コンパイルされる言語、または解釈される言語であることが可能であり、ハードウェア実施形態と組み合わせられることが可能である。

【0046】

本発明の方法および装置は、電気配線またはケーブル配線を介して、光ファイバを介して、または他の任意の形態の伝送を介してなど、何らかの伝送媒体を介して伝送されるプログラムコードの形態で実施することもでき、プログラムコードが、EPROM、ゲートアレイ、PLD（プログラマブル論理デバイス）、クライアントコンピュータ、ビデオレコーダなどのマシンによって受信され、読み込まれて、実行されると、そのマシンが、本発明を実施するための装置になる。汎用プロセッサ上に実装される場合、プログラムコードは、プロセッサと一緒に、本発明のインデックス作成（indexing）機能を実行するように動作する独自の装置を提供する。

【0047】

本発明を様々な図の好ましい諸実施形態に関連して説明してきたが、本発明を逸脱することなく、本発明と同一の機能を実行するために、他の類似した諸実施形態を使用すること、または説明した実施形態に変更または追加を行うことも可能であることを理解されたい。例えば、本発明の典型的な諸実施形態を、パーソナルコンピュータの機能をエミュレ

ートするデジタルデバイス群の文脈で説明したが、本発明は、本出願で説明したようなデジタルデバイス群に限定されず、有線であるか、無線であるかにかかわらず、ゲームコンソール、ハンドヘルドコンピュータ、ポータブルコンピュータなどのいくつかの既存の、または出現しつつあるコンピューティングデバイス、またはコンピューティング環境に適用することができ、通信網を介して接続され、そのネットワークを介して対話するいくつかのそのようなコンピューティングデバイスに適用することができることが、当業者には認識されよう。さらに、特に無線ネットワーク化されたデバイスの数が増え続けているなかで、ハンドヘルドデバイスオペレーティングシステム群、およびその他の特定用途向けのハードウェア／ソフトウェアインターフェースシステム群を含む、様々なコンピュータプラットフォームが、本明細書で企図されていることを強調しておかなければならない。したがって、本発明は、いずれの単一の実施形態にも限定されず、むしろ、添付の特許請求の範囲に従う広がり範囲で解釈されなければならない。

10

【0048】

最終的に、本明細書で説明する開示した諸実施形態は、他のプロセッサアーキテクチャ、コンピュータベースのシステム、またはシステム仮想化において使用するように適合させることができ、そのような諸実施形態は、本明細書で行う開示によって明確に予期されており、したがって、本発明は、本明細書で説明する特定の諸実施形態に限定されず、もっとも広く解釈されなければならない。同様に、プロセッサ仮想化以外の諸目的で合成命令群を使用することも、本明細書で行う開示によって予期されており、プロセッサ仮想化以外の文脈における合成命令群のいずれのそのような利用も、本明細書で行う開示に最も

20

【図面の簡単な説明】**【0049】**

【図1】本発明の諸態様を組み込むことができるコンピュータシステムを示すブロック図である。

【図2】コンピュータシステムにおけるエミュレートされた動作環境に関するハードウェアおよびソフトウェアのアーキテクチャの論理的な階層化を示す図である。

【図3A】仮想化されたコンピューティングシステムを示す図である。

【図3B】ホストオペレーティングシステムと並行に実行される仮想マシンモニタを含む仮想化されたコンピューティングシステムの代替の実施形態を示す図である。

30

【図4】VM認識ゲストオペレーティングシステム群を有するホストオペレーティングシステムをさらに含む図3Aからの仮想化されたコンピューティングシステムを示す図である。

【図5】仮想マシン環境における効率を向上させるために自らの挙動を変更する能力を有するVM認識ゲストオペレーティングシステムを実施する方法を示す流れ図である。

【図6】ホストオペレーティングシステムとVM認識ゲストオペレーティングシステム群の間で共有通信領域を有するホストオペレーティングシステムを含む典型的な仮想化されたコンピューティングシステムを示す図である。

【図7】本発明によるVM認識オペレーティングシステムにおいてスレッドをスケジューリング設定する典型的な方法を示す流れ図である。

40

【符号の説明】**【0050】**

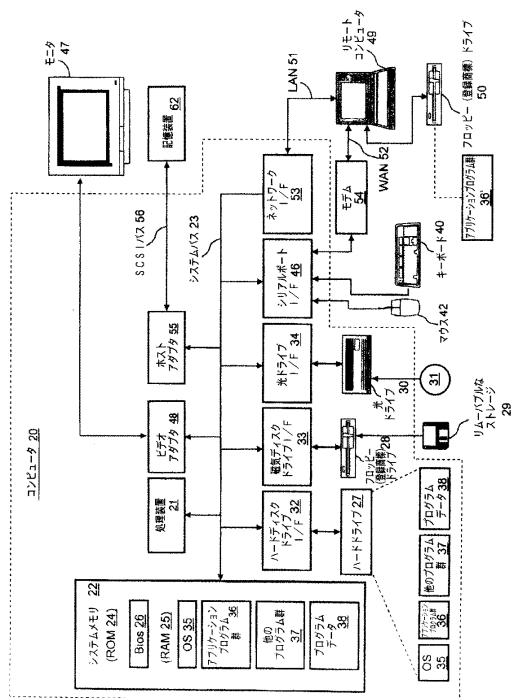
- 20 コンピュータ
- 21 処理装置
- 22 システムメモリ
- 23 システムバス
- 27 ハードドライブ
- 28 フロッピー（登録商標）ドライブ
- 50 フロッピー（登録商標）ドライブ
- 29 リムーバブルなストレージ

50

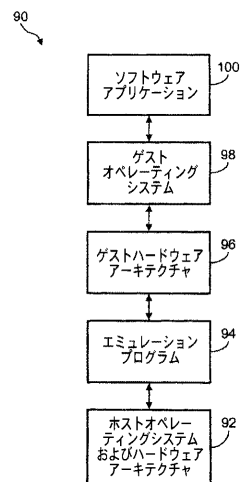
- | | |
|-------|-------------------|
| 3 0 | 光ドライブ |
| 3 2 | ハードディスクドライブ I / F |
| 3 3 | 磁気ディスクドライブ I / F |
| 3 4 | 光ドライブ I / F |
| 3 6 | アプリケーションプログラム群 |
| 3 6 , | アプリケーションプログラム群 |
| 3 7 | 他のプログラム群 |
| 3 8 | プログラムデータ |
| 4 0 | キーボード |
| 4 2 | マウス |
| 4 6 | シリアルポート I / F |
| 4 7 | モニタ |
| 4 8 | ビデオアダプタ |
| 4 9 | リモートコンピュータ |
| 5 3 | ネットワーク I / F |
| 5 4 | モデム |
| 5 5 | ホストアダプタ |
| 5 6 | SCSIバス |
| 6 2 | 記憶装置 |

10

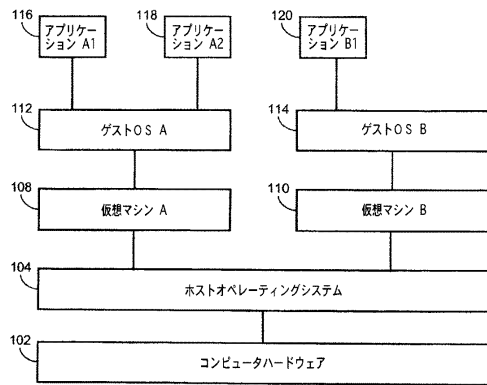
【図 1】



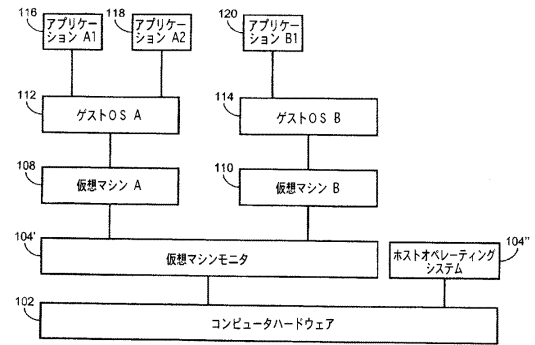
【図 2】



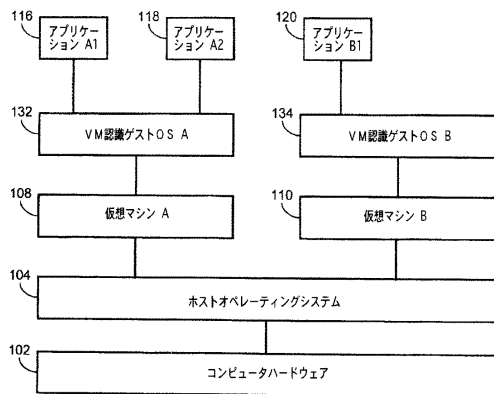
【図 3 A】



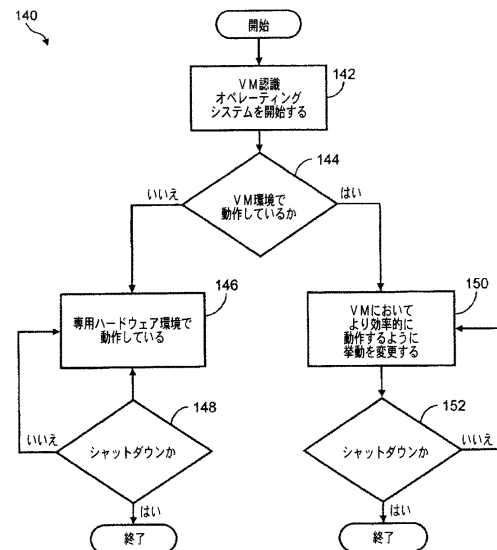
【図 3 B】



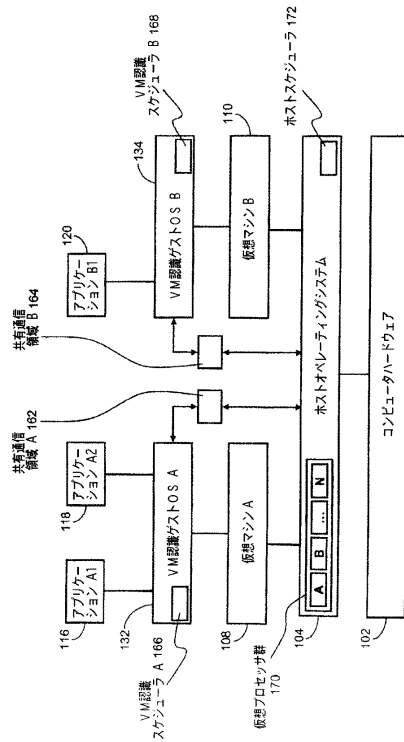
【図 4】



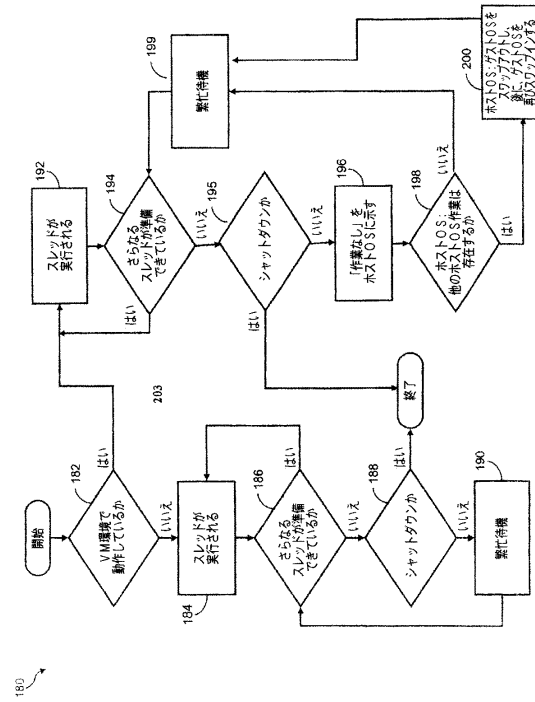
【図 5】



【図 6】



【図 7】



フロントページの続き

- (72)発明者 レーン アントニオ ベガ
アメリカ合衆国 98052 ワシントン州 レッドモンド ワン マイクロソフト ウェイ マ
イクロソフト コーポレーション内
- (72)発明者 エリック ピー. トラウト
アメリカ合衆国 98052 ワシントン州 レッドモンド ワン マイクロソフト ウェイ マ
イクロソフト コーポレーション内
- (72)発明者 マイク ニール
アメリカ合衆国 98052 ワシントン州 レッドモンド ワン マイクロソフト ウェイ マ
イクロソフト コーポレーション内

審査官 鈴木 修治

- (56)参考文献 特開平09-114683 (JP, A)
特開昭62-066336 (JP, A)
特開昭62-125438 (JP, A)
特開平04-213733 (JP, A)
特開昭57-212680 (JP, A)

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