(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2006-190281
(P2006-190281A)
(43)公開日　平成18年7月20日(2006.7.20)

(51)Int.Cl.			F I			テーマコード（参考）
G06F	9/46	(2006.01)	G06F	9/46	350	5B057
G06T	1/20	(2006.01)	G06T	1/20	Z	

審査請求 未請求　請求項の数 20　ＯＬ　（全 36 頁）

(21)出願番号　特願2005-375509 (P2005-375509)
(22)出願日　平成17年12月27日 (2005.12.27)
(31)優先権主張番号　11/026,380
(32)優先日　平成16年12月30日 (2004.12.30)
(33)優先権主張国　米国(US)

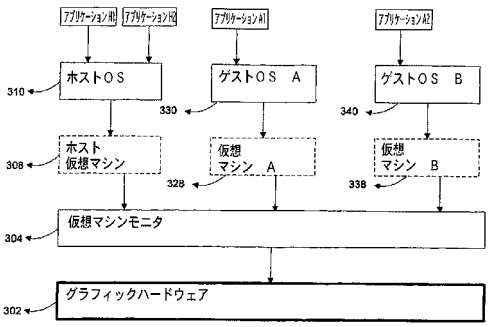
（特許庁注：以下のものは登録商標）
１．Ｍａｃｉｎｔｏｓｈ
２．Ｌｉｎｕｘ

(71)出願人　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　９８０５２－６３９９　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)代理人　100088915
弁理士　阿部　和夫
(72)発明者　デビッド　アール．ブライズ
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

Ｆターム(参考)　5B057 CH06　CH11　CH18　CH20

(54)【発明の名称】グラフィックサブシステムを仮想化するシステムおよび方法

(57)【要約】
【課題】グラフィックハードウェアに仮想マシンを適用するシステムおよび方法を提供すること。
【解決手段】ＣＰＵで監視コードが実行される間、実際のグラフィック作業項目は直接グラフィックハードウェアで実行され、監視コードは、グラフィック仮想マシンモニタとして構成される。仮想マシンモニタ（VMM）技術を使用して、別々の仮想マシン（VM）で、最初のＯＳバージョンなどの第１のオペレーティングシステム（ＯＳ）を新バージョンのＯＳなどの第２のＯＳと同時に実行することにより、アプリケーションの互換性が維持される。ホストプロセッサに適用されるVMM技術をグラフィック処理装置（ＧＰＵ）に拡張して、グラフィックアクセラレータへのハードウェアアクセスを可能にし、レガシーアプリケーションが最大パフォーマンスで動作することを保証する。
【選択図】図３Ａ



【特許請求の範囲】
【請求項１】
第１の仮想マシンでホストされる第１のオペレーティングシステムと、第２の仮想マシンでホストされる第２のオペレーティングシステムとを有するコンピューティングシステムでグラフィックデータを処理する方法であって、
前記コンピューティングシステムのグラフィックサブシステムのグラフィック処理装置（ＧＰＵ）によって実行されるべき１つまたは複数の作業項目を受け取るステップであって、前記１つまたは複数の作業項目が前記第１の仮想マシンまたは前記第２の仮想マシンに起因するステップと、
前記１つまたは複数の作業項目が、前記第１のオペレーティングシステムからのインタフェースに基づくか、前記第２のオペレーティングシステムからのインタフェースに基づくかに関係なく、前記ＧＰＵで前記１つまたは複数の作業項目を処理するステップと
を備えることを特徴とする方法。
【請求項２】
前記第１および第２の仮想マシンを可能にする仮想マシンモニタコンポーネントのグラフィック仮想マシンモニタコンポーネントにより、前記１つまたは複数の作業項目を処理するステップをさらに備えることを特徴とする請求項１に記載の方法。
【請求項３】
前記処理は、システムメモリまたはビデオメモリの少なくとも１つを含む少なくとも１つの物理リソースを前記１つまたは複数の作業項目に割り振るステップを含むことを特徴とする請求項２に記載の方法。
【請求項４】
前記少なくとも１つの物理リソースを割り振るステップは、前記第１の仮想マシンまたは前記第２の仮想マシンで実行される別個のアプリケーションに起因する作業項目に、ビデオメモリ空間またはシステムメモリ空間の少なくとも１つを別々に割り振ることを含むことを特徴とする請求項３に記載の方法。
【請求項５】
前記処理は、前記１つまたは複数の作業項目に特権があるか、または特権がないかを判定するステップと、特権のある作業項目を処理するためと特権のない作業項目を処理するためとで異なる規則のセットを適用するステップを含むことを特徴とする請求項２に記載の方法。
【請求項６】
前記第１および第２の仮想マシンを可能にする仮想マシンモニタコンポーネントによって作成される信頼されるコードベースから、前記１つまたは複数の作業項目の出力の表示を合成するステップをさらに備えることを特徴とする請求項１に記載の方法。
【請求項７】
前記第２のマシンで実行される少なくとも１つの信頼されるサービスを利用して前記出力を保護することを含めて、前記第１のマシンで実行される信頼されるアプリケーションの出力を処理するステップをさらに備えることを特徴とする請求項１に記載の方法。
【請求項８】
請求項１に記載の方法に従って通信を行うためのコンピュータ実行可能命令を備えることを特徴とするアプリケーションプログラミングインタフェース。
【請求項９】
請求項１に記載の方法を行うように調整された機能コンポーネントを備えることを特徴とするグラフィック処理装置。
【請求項１０】
請求項１に記載の方法を行う手段を備えることを特徴とするコンピューティングデバイス。
【請求項１１】
仮想マシンモニタコンポーネントにより第１の仮想マシンにおいてホストされる第１の

バージョンのオペレーティングシステムと、前記仮想マシンモニタコンポーネントにより第２の仮想マシンにおいてホストされる前記オペレーティングシステムの第２のバージョンとを有するコンピューティングシステムでグラフィックデータを処理する方法であって、

前記コンピューティングシステムのグラフィックサブシステムのグラフィック処理装置（ＧＰＵ）によって実行されるべき１つまたは複数の作業項目を前記第１の仮想マシンまたは第２の仮想マシンから受け取るステップであって、前記作業項目の少なくとも１つは、特権コマンドを含むステップと、

前記仮想マシンモニタコンポーネントによって前記特権コマンドを発見するステップと、

少なくとも１つのあらかじめ定義されたポリシーに基づいて、前記作業項目の前記少なくとも１つの前記特権コマンドをエミュレートするステップと

を備えることを特徴とする方法。

【請求項１２】

前記仮想マシンモニタコンポーネントのグラフィック仮想マシンモニタコンポーネントと通信するステップであって、前記グラフィック仮想マシンモニタコンポーネントは、前記第１の仮想マシンおよび前記第２の仮想マシンの両方に代わって、特権のあるＧＰＵコマンドを有する作業項目を処理するステップをさらに備えることを特徴とする請求項１１に記載の方法。

【請求項１３】

請求項１１に記載の方法に従って通信を行うためのコンピュータ実行可能命令を備えることを特徴とするアプリケーションプログラミングインタフェース。

【請求項１４】

請求項１１に記載の方法を行うように調整された機能コンポーネントを備えることを特徴とするグラフィック処理装置。

【請求項１５】

請求項１１に記載の方法を行う手段を備えることを特徴とするコンピューティングデバイス。

【請求項１６】

コンピューティングデバイスの１つまたは複数の仮想マシンで実行される第１のアプリケーションおよび第２のアプリケーションと、

権限を与えられない限り、前記第１のアプリケーションに対して許可されないあらゆる物理リソースと対話するための前記第１のアプリケーションからのグラフィックインタフェース呼び出しを許可しない仮想マシンモニタコンポーネントと

を備えることを特徴とするコンピューティングデバイス。

【請求項１７】

前記仮想マシンモニタコンポーネントは、前記第２のアプリケーションに関連付けられたシステムメモリまたはローカルメモリを変更する、またはそれらのメモリから読み出すための前記第１のアプリケーションからのグラフィックインタフェース呼び出しを許可しないことを特徴とする請求項１６に記載のコンピューティングデバイス。

【請求項１８】

前記仮想マシンモニタコンポーネントは、前記第１のアプリケーションおよび前記第２のアプリケーションそれぞれからの出力に基づいて、ディスプレイのウィンドウを提示するための信頼されるコードベースを含むことを特徴とする請求項１６に記載のコンピューティングデバイス。

【請求項１９】

前記第１のアプリケーションは、前記第１のアプリケーションに関連付けられた第１のデスクトップを有する第１のオペレーティングシステムプラットフォームで実行され、前記第２のアプリケーションは、前記第２のアプリケーションに関連付けられた第２のデスクトップを有する第２のオペレーティングシステムプラットフォームで実行され、前記信

頼されるコードベースによる提示するステップは、前記第１のデスクトップを第１のウィンドウに表示して提示するステップと、前記第２のデスクトップを第２のウィンドウに表示して提示するステップとを含むことを特徴とする請求項１８に記載のコンピューティングデバイス。

【請求項２０】

前記第１のアプリケーションは、前記第１のアプリケーションに関連付けられた第１のデスクトップを有する第１のオペレーティングシステムプラットフォームで実行され、前記第２のアプリケーションは、前記第２のアプリケーションに関連付けられた第１のデスクトップを有する第２のオペレーティングシステムプラットフォームで実行され、前記信頼されるコードベースによる前記提示するステップは、前記第１のデスクトップまたは前記第２のデスクトップを表示して提示するステップを含むことを特徴とする請求項１８に記載のコンピューティングデバイス。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、コンピューティングデバイスのグラフィックサブシステムを仮想化するシステム、装置、方法、ユーザインタフェース、プロトコル、およびアプリケーションプログラミングインタフェース（ＡＰＩ）を対象とする。より詳細には、本発明は、互換性、セキュリティ、およびデジタル著作権管理に関連する利益を提供する、仮想化されたグラフィックアーキテクチャを対象とする。

【背景技術】

【０００２】

現在のグラフィックパイプラインの現況に関する背景知識として、グラフィックサブシステムは、一般に、ホストコンピュータと協働して、例えば、モニタ、プリンタ、あるいは他のデバイスに表示するためのアプリケーションの出力を準備するなど、相対的に言って、生の計算能力を多大に必要とするタスクのような、特殊化された特定のタスクをホストコンピュータに代わって行う。例えば、３次元のコンピュータグラフィック表現を作成するには、描画しようとするオブジェクトが、グラフィックサブシステムのグラフィック処理装置（ＧＰＵ）により数学的に集約した処理を行うのに適した数学的モデルとしてコンピュータ内で表される。例えば、３Ｄモデルは、例えば、それぞれ幅、高さ、奥行きに対応するｘ軸、ｙ軸、ｚ軸からなる座標系内の幾何学的点で構成されることができる。オブジェクトは、頂点と呼ばれる一連の点によって定義される。点あるいは頂点の位置は、そのｘ、ｙ、ｚ座標（または他の座標系）によって定義される。グラフィックの術語では、１つの頂点は点であり、２つの頂点は線または線分を定義し、３つの頂点は三角形を定義し、これら３つはすべて「プリミティブ」である。そうした点の３つ以上がつながれるとポリゴンが形成され、三角形が最も単純なポリゴンであり、ポリゴンを使用してオブジェクトの３Ｄ形状を近似し、その３Ｄ形状にグラフィックデータを適用して、各種の芸術的および現実感のある効果と色の変化を作り出すことができる。グラフィックパイプラインは、各種の計算サブユニットを使用して、頂点および他のデータストリームを非常に高速かつ効率的に処理して、最終的に非常に複雑な３Ｄオブジェクトを２次元の表示空間に表現することができる。

【０００３】

そのため、画面への３次元（３Ｄ）グラフィックのレンダリングと表示などの一定のタスクは、通常、グラフィックサブシステムで行われるのに適した多くの計算および演算を伴う。単純なグラフィックシステムでは、そのような計算は、中央演算処理装置（ＣＰＵ）とグラフィック処理装置（ＧＰＵ）による何らかのレベルの協働的な処理あるいは共用処理（ｓｈａｒｅｄ　ｐｒｏｃｅｓｓｉｎｇ）に従って行われる。例示的なシナリオでは、命令が処理され、いくらかの最初の計算がＣＰＵで行われると、レンダリングされるオブジェクトを定義する座標点あるいは頂点のセットが、グラフィックパイプラインでＧＰＵによってさらに処理されるためにビデオメモリに記憶される。例えばデータが３Ｄグラ

フィックデータである場合は、表現されるオブジェクトの表面を効率的にカバーするように設計された所定のアルゴリズムに従って、テッセレータ（ｔｅｓｓｅｌｌａｔｏｒ）が、グラフィックデータを単純なポリゴンに分割することができ、このプロセスは、テッセレーションと呼ばれる。現在、大半のグラフィックパイプラインでは、ＧＰＵに伝えられる命令に応じて、データはその後、手続き型シェーダなどのＧＰＵの１つまたは複数の計算サブユニットによって操作されることができ、ＧＰＵは、ビデオメモリ、すなわちフレームバッファと協働して、そのデータの作業命令に従ってデータを処理し、例えばディスプレイやプリンタ装置など、適切に指示される場所にデータを出力する。

【０００４】

図１に示すように、グラフィックコプロセッシングシステムを有するコンピューティングシステムに関して、コンピューティングシステムは、ホストＣＰＵとグラフィックハードウェアに分割される。ＣＰＵは、使用を要求するアプリケーションとサービスによるグラフィックＡＰＩの呼び出しを容易にする。従来、アプリケーションとドライバはＣＰＵ側に位置し、それらのソースからの情報は、データをモニタに表示するなどの作業項目としてグラフィックパイプラインに送信される。初めに、情報は、ＡＰＩに従ってＣＰＵによってパッケージされた状態でＣＰＵからＧＰＵに送られる。そして、アプリケーションからの情報は、３Ｄグラフィックデータが処理される場合の頂点シェーダなどのＧＰＵの計算サブユニットによってスケジュールされ、処理されるまで、メモリで待機する。頂点シェーダが操作を完了すると、情報は通常、さらなる処理のためにピクセルシェーダからアクセスされるまで、頂点シェーダから特別なデータ経路を通ってピクセルシェーダに出力される。ピクセルシェーダが操作を行った後、情報はフレームバッファに置かれて、ディスプレイにスキャンアウトされるか、さらなる操作のためにホストに送り返される。

【０００５】

今日のグラフィックアーキテクチャにおける用語「フレームバッファ」は、一般に、ラスタ化および／またはデジタルアナログ変換器（ＤＡＣ）出力プロセスとの関連で使用されるあらゆるメモリ（一般にはＧＰＵとの相互動作のために含まれるビデオメモリ）を指す。この点で、用語「ラスタ化」はより一般的に用いられることもあるが、グラフィックパイプラインにおけるピクセル処理またはセットアップエンジン処理との関連で行われる処理を一般にはラスタ化と称する。それに対し、スキャンアウトまたはＤＡＣ出力は、フレームバッファの内容に基づいてモニタまたはＬＣＤに信号を送信するプロセスである。

【０００６】

タスクの特殊化および関連する計算リソースのために、ディスプレイ画面にグラフィックオブジェクトを表示するために、通例、コンピュータシステムのグラフィックサブシステムが使用される。３次元（３Ｄ）のコンピュータグラフィックの目的は、一般に、オブジェクトを３次元でリアルに表現する２次元（２Ｄ）画像をコンピュータ画面に作り出すことである。現実世界では、オブジェクトは、実際の高さ、実際の幅、および実際の奥行きを有する３次元を占める。写真は、３次元空間の２次元表現の例である。３Ｄのコンピュータグラフィックは、一般に、基礎となる画像が一般には３Ｄ形状と表面テクスチャとともにモデル化されることを除けば、コンピュータ画面の２Ｄ空間に３次元の世界を表現する点で、写真に似ている。

【０００７】

３Ｄのコンピュータグラフィックで作成された画像は、所与の時点におけるシーンの個々の光景をリアルな手法で描写するために、ビデオ娯楽ゲームから、航空機の飛行シミュレータにわたる幅広いアプリケーションで使用される。よく知られる３Ｄコンピュータグラフィックの例には、「ターミネータＩＩ」、「ジュラシックパーク」、「トイストーリー」などのハリウッド映画の特殊効果が含まれる。ここ数年に特に多大な量の成長を見ている業界の１つは、コンピュータゲーム業界であり、現在の世代のコンピュータゲームは、かつてないほど多く３Ｄグラフィック技術を適用するようになっている。同時に、プレイの速度は加速されている。この組み合わせにより、比較的安価なシステムで３Ｄグラフィックを高速かつ柔軟なレンダリングに対する純粋な必要性が増大している。

【発明の開示】
【発明が解決しようとする課題】
【０００８】
しかし、グラフィックサブシステムに関連するこうした急速に進展する傾向線に従い、それに呼応して互換性、セキュリティ、および著作権管理が問題として生じている。新世代のＧＰＵとそれに関連するハードウェアが１つ前の世代に取って代わるたびに、ハードウェアに加えて、パイプラインとのインタフェース（例えばＤｉｒｅｃｔＸ８、ＤｉｒｅｃｔＸ９、ＧＤＩ、ＧＤＩ＋など）が発展しなければならず、それらが「古い」インタフェースの呼び出しに対処することが期待される場合は、古いハードウェアのサポートを保証する別個のコードが書かれる必要がある。２～３世代以上が経過すると、ある特定のオペレーティングシステムのグラフィックインタフェースコードがなお管理可能のままである可能性は下がり、互換性コードのファンアウトの影響が定着するのにつれて、互換性のない関数呼び出し（ｆｕｎｃｔｉｏｎ　ｃａｌｌ）に起因する処理中のエラーの可能性が増す。追加的な問題は、種々のオペレーティングシステム、あるいはプラットフォーム（Ｍａｃｉｎｔｏｓｈ、Ｌｉｎｕｘ、Ｗｉｎｄｏｗｓ（登録商標）など）、あるいは同じプラットフォームに含まれる異なる系統（ＯｐｅｎＧＬ、ＧＤＩ＋、ＤｉｒｅｃｔＸなど）の存在のために、グラフィックインタフェースコードの完全に別個の「系統」が展開することである。したがって、様々なバージョンのオペレーティングシステム、グラフィックインタフェース、およびそれに関連付けられたハードウェアがすべて共にサポートされない場合があるという問題が残り、この問題は、時間の経過とともにより複雑になりつつある。特定のオペレーティングシステムあるいはインタフェースのセットとともに提供されるコアコードに変更を加える必要なしに、この問題を多少とも解決することが望ましい。
【０００９】
同じ背景から、グラフィックパイプラインとの関連で、セキュリティとコンテンツ保護／デジタル著作権管理（ＤＲＭ）の問題が、解決するのがより複雑になってきている。種々のオペレーティングシステム、インタフェース、ハードウェアのせいで、知られるすべてのパイプラインにわたってデータのセキュリティを実施する、またはコンテンツの保護／デジタル著作権管理の条件を強制する共通の機構はない。そのため、グラフィックパイプラインに依存せずに機能する、データのセキュリティおよび／またはＤＲＭに関する問題に対処する共通の手段またはレイヤが望まれる。
【００１０】
したがって、ホストプロセッサとＧＰＵの間の対話の層として仮想マシンを提供することにより、コンピュータシステムのグラフィックパイプラインを仮想化することが望ましい。さらに、各種バージョンのオペレーティングシステムまたはインタフェース、または各種のＧＰＵのアーキテクチャが利用される場合に遭遇する、現在の互換性、セキュリティ、コンテンツ保護／ＤＲＭの問題の欠点を克服するシステムおよび方法を実施することが望ましい。
【課題を解決するための手段】
【００１１】
当技術分野の上記で明らかにした欠点および他の欠点に鑑みて、本発明は、グラフィックハードウェア空間に仮想マシンを適用するシステムおよび方法を提供する。本発明の各種実施形態で、ＣＰＵで監視コード（ｓｕｐｅｒｖｉｓｏｒｙ　ｃｏｄｅ）が実行される間、実際のグラフィック作業項目は、直接グラフィックハードウェアで実行される。非制限的な一実施形態では、監視コードは、グラフィック仮想マシンモニタとして構成される。
【００１２】
仮想マシンモニタ（ＶＭＭ）技術を使用して、別々の仮想マシン（ＶＭ）で、最初のＯＳバージョンなどの第１のオペレーティングシステム（ＯＳ）を、新バージョンのＯＳなどの第２のＯＳと同時に実行することにより、アプリケーションの互換性が維持される。一実施形態では、ホストプロセッサに適用されるＶＭＭ技術をグラフィック処理装置（Ｇ

ＰＵ）に拡張して、グラフィックアクセラレータへのハードウェアアクセスを可能にし、レガシーアプリケーションが最大パフォーマンスで動作することを保証する。本発明は、複数のアプリケーションを別々のＶＭで実行する間に、ユーザエクスペリエンスを表面上シームレスにする方法も提供する。本発明の他の態様では、ＶＭＭ技術を用いることにより、本発明の仮想化グラフィックアーキテクチャが拡張されて、信頼されるサービスとコンテンツ保護を提供する。

【００１３】

本発明のこの他の利点および特徴は以下で述べる。

【００１４】

本発明によるＧＰＵを仮想化するシステムおよび方法について、添付図面を参照してさらに説明する。

【発明を実施するための最良の形態】

【００１５】

概要

上記で述べたように、本発明は、グラフィックハードウェア空間に仮想マシンを適用するシステムおよび方法を提供する。本発明の各種実施形態で、グラフィック仮想マシンモニタなどの監視コードがＣＰＵで実行される間、実際のグラフィック作業項目は、グラフィックハードウェアで直接実行される。

【００１６】

本発明の非制限的な一態様では、グラフィックハードウェア機能を、特権操作と非特権操作に分け、特権操作を効率的に仮想化し、同時にシームレスなユーザエクスペリエンスを維持するシステムおよび方法が提供される。本発明の別の非制限的な態様では、ハードウェア割り込み処理の効率的な仮想化が達成される。さらなる態様で、本発明は、メモリ、処理サイクル、画面の面積（ｒｅａｌ　ｅｓｔａｔｅ）などのリソースのあらかじめ固定された割り当てをパーティションに割り振る、かつ／またはアプリケーション間共通のおよびプラットフォーム間共通のパラメータ／システムの需要に応じてそのようなリソースを動的に割り当てるリソース管理機能をエミュレーション層に提供する。

【００１７】

このアーキテクチャの別の有益な結果は、本発明では、ゲストオペレーティングシステムまたはアプリケーションに対する変更を必要とすることなく、仮想パーティションが、グラフィックハードウェアへの直接のアクセス権を持つことができることである。本発明の基本アーキテクチャからは、多くの応用も得られる。例えば、レガシーコードに依存したアプリケーションを実行するために、レガシーオペレーティングシステムのインスタンスを使用して、古いオペレーティングシステムにそのレガシーコードを残すことができる。別の例は、信頼できるサービスと信頼できないサービスを構築するという発想であり、言い換えるとそれらのサービスを使用して、信頼できるグラフィック出力またはコンテンツ保護技術をサポートすることができるという発想である。

【００１８】

本発明により提供される他のシステムおよび方法は、ユーザエクスペリエンスに関連する。複数のパーティションにより、本発明により提供されるユーザエクスペリエンスの１つは、各ゲストオペレーティングシステムについて表示されるデスクトップが、表示装置（または「ｍｕｌｔｉｍｏｎ」と呼ばれることもある複数のモニタがある場合は複数の表示装置）のウィンドウ（あるいは代替のデスクトップビュー）にリダイレクトされる「ウィンドウ内のデスクトップ（ｄｅｓｋｔｏｐ　ｉｎ　ａ　ｗｉｎｄｏｗ）」方式である。他の実施形態では、本発明は、異なるゲストオペレーティングシステムからのウィンドウを混合して１つの統一された表示にし、それにより、特定のウィンドウに関連付けられた「ピクセル」が識別されることができるユーザエクスペリエンスを可能にする。一実施形態で、本発明は、ゲストオペレーティングシステムに「ヘルパプログラム」を追加して各種ウィンドウの部分を識別することにより、ピクセルとそのウィンドウの関連付けの識別を実現する。他の実施形態で、本発明は、ゲスト「ウィンドウマネジャ」との対話を可能

にして、メインデスクトップで隠蔽されないウィンドウが、仮想のゲストオペレーティングシステムのデスクトップで隠蔽されないようにし、その結果、それらのウィンドウが抽出され、メインデスクトップで表示できるようにする。他の実施形態では、複数の異なるウィンドウからピクセルのメモリ位置を識別することを部分的に、または完全に助けるグラフィックハードウェアサポートが提供される。

【００１９】

本発明は、仮想デスクトップでウィンドウを管理することを助ける追加的なグラフィックハードウェアサポートを提供する実施形態も含む。将来のオペレーティングシステムがこのタスクをより容易にするサポートを追加することができる領域ではあるが、その発想は、異なるゲストオペレーティングシステムからの個々のウィンドウを識別するフレームワークを可能にし、それら複数のソースを構成する統一されたプレゼンテーションサービス（デスクトップコンポジションエンジンと称されることもある）がそのデータを利用できるようにするというものである。このエンジンは、複数のゲストのグラフィックメモリ状態を入手することができ、一方、ゲストは、通常、互いの関数呼び出しとアクティビティを見ることができない。

【００２０】

個々のゲストＯＳの内容に関連してディスプレイ出力に複雑な階層化された複合効果を可能にする手法の１つは、複合デスクトップを構築するために、プレゼンテーションエンジンに様々なゲストＯＳからのウィンドウ内容を「引き出させる」ものである。代替の手法は、ゲストＯＳが単一のメインデスクトップに直接レンダリングする、すなわちそのメモリにレンダリングできるようにするが、特別なハードウェアサポートを使用して、そのゲストＯＳが描画できるデスクトップの部分を制約するものである。例えば、矩形のリストあるいはビットマップが、どのピクセルが書き込み可能かを示すマスクを定義するハードウェアウィンドウクリッピングに使用される機構と同様の別個の特権機構を使用して、許容できる領域を制御することができる。これを「プッシュ」動作として実施することもでき、その場合、ゲストＯＳは直接デスクトップに描画し、プレゼンテーションエンジンが、各ゲストＯＳが書き込むことが可能な領域を制御する。

【００２１】

各種の他の実施形態で、本発明は、第１のＯＳにレンダリングされるウィンドウと第２のＯＳにレンダリングされるウィンドウのピクセルを識別および位置特定し、そのウィンドウを１度に１つずつ、または、何らかの他の方式でともに組み合わせて、表示のために提示するシステムおよび方法を提供する。本発明は、割り当てられたリソースには第２のＯＳから直接アクセスすることができ、割り当てられないリソースにはアクセスできないように、ビデオメモリやシステムメモリなどの物理リソースを第２のＯＳに割り振り、割り当てるシステムおよび方法も含む。本発明はさらに、特権操作をインターセプトし、効率的にエミュレートするシステムおよび方法を提供する。

【００２２】

下記で本発明の他のより詳細な態様を説明するが、初めに、以下の説明では、それらの用語がオペレーティングシステムとホストプロセッサ（「ＣＰＵ」）の仮想化技術との関連で知られるようになっているので、仮想マシンについての概要といくつかの一般的な語彙とそれに関連する術語を提供する。その際、当業者にとって、以下の本発明によるコンピューティングシステムのグラフィックサブシステム側にエミュレーション機能を提供する装置、システム、および方法の説明に役立つと思われる語彙のセットを説明する。

【００２３】

仮想マシンの概要

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汎用的な中央演算処理装置（ＣＰＵ）すなわち「プロセッサ」を含む。同様のアーキテクチャあるいは設計仕様を有するプロセッサのグループは、同じプロセッサファミリのメンバと見なしてよい。現在のプロセッサファミリの例には、Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｂｕｓｉｎｅｓｓ Ｍａｃｈｉｎｅｓ（ＩＢＭ）またはアリゾナ州フェニックスのＭｏｔｏｒｏｌａ社

製造のＭｏｔｏｒｏｌａ６８０Ｘ０プロセッサファミリ、カリフォルニア州サニーベールのＩｎｔｅｌ社製造のＩｎｔｅｌ８０Ｘ８６プロセッサファミリ、Ｍｏｔｏｒｏｌａ社によって製造され、カリフォルニア州クパティーノのＡｐｐｌｅ　Ｃｏｍｐｕｔｅｒ社製造のコンピュータで使用されるＰｏｗｅｒＰＣプロセッサファミリがある。１つのグループのプロセッサは、それらのアーキテクチャと設計上の考慮事項が似ているので同じファミリであり得るが、プロセッサは、クロック速度および他の性能パラメータに応じて１つのファミリ内で大きく異なる場合がある。

【００２４】

各マイクロプロセッサファミリは、そのプロセッサファミリに固有の命令を実行する。１つのプロセッサまたはあるプロセッサファミリが実行することができる命令の集合セットは、そのプロセッサの命令セットと称される。一例として、Ｉｎｔｅｌ８０Ｘ８６プロセッサファミリに使用される命令セットは、ＰｏｗｅｒＰＣプロセッサファミリで使用される命令セットとは互換性がない。Ｉｎｔｅｌ８０Ｘ８６命令セットは、ＣＩＳＣ（複合命令セットコンピュータ）フォーマットに基づく。Ｍｏｔｏｒｏｌａ　ＰｏｗｅｒＰＣ命令セットは、ＲＩＳＣ（縮小命令セットコンピュータ）フォーマットに基づく。ＣＩＳＣプロセッサは、多数の命令を使用し、そのうち一部はやや複雑な機能を実行することができるが、一般に実行するのに多くのクロックサイクルを必要とする。ＲＩＳＣプロセッサは、より少ない数の利用可能命令を使用して、はるかに高いレートで実行されるより単純な機能のセットを実行する。

【００２５】

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独自性は、通常は、それらの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他の要素間の非互換性をもたらす。Ｉｎｔｅｌ８０Ｘ８６プロセッサファミリのプロセッサで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のプロセッサで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ことになる。プロセッサ命令セットと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独自性のために、アプリケーションソフトウェアプログラムは、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で実行されるように書かれる。

【００２６】

一般に、コンピュータ製造者は、そのコンピュータ製造者の製品ラインに関連付けられたマイクロプロセッサファミリで少なからぬアプリケーションを実行させることにより、自社のマーケットシェアをできる限り最大化しようとする。あるコンピュータシステムで実行することができるオペレーティングシステムとアプリケーションプログラムの数を拡大するために、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性のないタイプのＣＰＵの命令をホストコンピュータがエミュレートできるようにするバーチャライザプログラムを備える技術分野が発展している。したがって、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を呼び出させるアプリケーションを実行し、そのようにして、ホストコンピュータは、そのホストコンピュータのハードウェアアーキテクチャのために設計されたソフトウェアと、関連性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のために書かれたソフトウェアの両方を実行することができる。

【００２７】

より具体的な例として、例えばＡｐｐｌｅ　Ｃｏｍｐｕｔｅｒ社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のために書かれたオペレーティングシステムとプログラムを実行することができる。バーチャライザプログラムを使用して、単一のＣＰＵで複数の互換性のないオペレーティングシステムを同時に実行することも可能である場合がある。この後者の構成では、各オペレーティングシステムは互いに互換性がないが、バーチャライザプログラムが、いくつかのオペレーティングシステムのそれぞれをホストすることができ、それにより、本来は互換性のないオペレーティングシステムを同じホストコンピュータシステムで同時に実行できるようにする。

【００２８】
ホストコンピュータシステムでゲストコンピュータシステムが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、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のみホストコンピュータシステムに存在するので、ゲストコンピュータシステムは、「仮想マシン」と言われる。バーチャライザ、エミュレータ、ダイレクトエクゼキュータ、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、当業者に知られ、理解される１つまたはいくつかの手法を使用して、１つの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あるいはエミュレートする能力を表すために時々同義で使用される。さらに、あらゆる形での用語「エミュレーション」の使用はすべて、この広い意味を伝えることが意図されており、エミュレーションの命令実行の概念と、仮想マシンでのオペレーティングシステム命令の直接の実行を区別することを意図するものではない。したがって、例えば、カリフォルニア州サンマテオのＣｏｎｎｅｃｔｉｘ社によって作成されたＶｉｒｔｕａｌ　ＰＣソフトウェアは、Ｉｎｔｅｌ８０Ｘ８６Ｐｅｎｔｉｕｍ（登録商標）プロセッサおよび各種のマザーボードコンポーネントとカードを含むコンピュータ全体を（命令実行のエミュレーションおよび／または直接の実行により）「エミュレート」し、それらのコンポーネントの動作は、ホストマシンで実行される仮想マシンで「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とハードウェアアーキテクチャで実行されるバーチャライザプログラムは、ゲストコンピュータシステム全体の動作を模倣する。

【００２９】
仮想化の一般的事例では、１つのプロセッサアーキテクチャが他のプロセッサアーキテクチャ由来のＯＳとプログラム（例えばｘ８６Ｗｉｎｄｏｗｓ（登録商標）でＰｏｗｅｒＰＣ　Ｍａｃのプログラムを実行する。またその逆など）を実行できるようにするが、重要な特殊事例の１つは、基礎となるプロセッサアーキテクチャが同じ場合である（ｘ８６上でｘ８６Ｌｉｎｕｘの各種バージョンあるいはｘ８６Ｗｉｎｄｏｗｓ（登録商標）の各種バージョンを実行する等）。この後者の場合は、基礎となる命令セットが同じなので、ゲストＯＳとそのアプリケーションをより効率的に実行できることが見込まれる。そのような場合、制御権を失う、あるいはシステムを攻撃にさらされる状態にすることなく、ゲスト命令がプロセッサで直接実行されることが許される（すなわちゲストＯＳがサンドボックスされる）。下記で詳細に説明するように、これが、特権ありと特権なしとの分離と、メモリへのアクセスを制御する技術が関与する箇所である。アーキテクチャ上の不一致（ＰｏｗｅｒＰＣ＜－＞ｘ８６）がある仮想化では、２つの手法を使用することが可能である。すなわち、命令単位のエミュレーション（比較的低速）か、またはゲスト命令セットからネイティブ命令セットへの変換（より効率的であるが変換ステップを用いる）である。命令のエミュレーションが用いられる場合は、環境を堅牢にすることは比較的容易であるが、変換が用いられる場合は、プロセッサアーキテクチャが同じである特殊事例に戻ることになる。

【００３０】
本発明によれば、グラフィックプラットフォームが仮想化され、したがって、グラフィックアーキテクチャに適用される本発明による例示的シナリオは、ＮＶｉｄｉａハードウェアを用いたＡＴＩカードのエミュレーションである。グラフィックの仮想化の場合では、各種実施形態で、本発明は、同じグラフィックアーキテクチャからのコマンドの直接の実行に着目する。グラフィックのコンテクストでは、変換に相当するものは実際には存在せず、エミュレーションと変換が同じ事を意味する。コードが一旦変換されると、コードは複数回実行されることができ、それにより変換コストを償却するので、変換は、プロセッサに対して効果を発揮する。しかし、グラフィックアーキテクチャが構築される方式のために、変換を行う（シェーダを無視する）プログラムに相当するものは必ずしも存在しない。代わりに、コマンドストリーム（作業項目のリスト）が常に再生成されるので、変換コストを償却することができる再使用は必ずしもない。より複雑な実施形態では、作業

項目のリストを調べて、それらがあらかじめ変換されたリストと一致するかどうかを判定することができる。

【００３１】

下記の各種実施形態で述べるように、本発明の各種実施形態によれば、（変換から、または同じプロセッサアーキテクチャが関連するために）ゲスト命令を直接実行しながらの仮想化は、システムのセキュリティを維持できる場合には、性能上の利益がある。したがって、本発明は、各種実施形態で、基礎となる物理リソース（メモリ、デバイスなど）の一部またはすべてへのゲストアクセスを制御するシステムおよび方法を記載する。

【００３２】

バーチャライザプログラムは、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で実行されるソフトウェア（オペレーティングシステム、アプリケーションなど）により送信される命令との間のインターチェンジ（ｉｎｔｅｒｃｈａｎｇｅ）の役割を果たす。このバーチャライザプログラムは、物理的なコンピュータハードウェアで直接実行されるオペレーティングシステムであるホストオペレーティングシステム（ＨＯＳ）であってよい（本明細書では下記でより詳細に説明するハイパーバイザを備えることができる）。あるいは、エミュレートされる環境は、ハードウェアの上で直接実行されるソフトウェア層である仮想マシンモニタ（ＶＭＭ）であってもよく、これは、恐らくはホストオペレーティングシステムと並行して実行され、オペレーティングシステムと連係して機能し、そのＶＭＭが仮想化しているハードウェアと同じインタフェースを公開することにより、ホストマシンのすべてのリソース（および特定の仮想リソース）を仮想化することができる。この仮想化は、バーチャライザ（およびホストコンピュータシステム自体）が、その上で実行されるオペレーティングシステム層に認識されずに済むことを可能にする。

【００３３】

このように、プロセッサエミュレーションは、物理ハードウェアとホストオペレーティングシステムの両方を備えるホストコンピュータシステムで実行されるバーチャライザによって作り出される仮想マシンで、ゲストオペレーティングシステムを実行することを可能にする。

【００３４】

概念的な観点から見ると、コンピュータシステムは一般に、基礎となるハードウェア層で実行される１つまたは複数のソフトウェア層を備える。この階層化は、抽象化の理由から行われる。所与のソフトウェア層のためのインタフェースを定義することにより、その層は、それより上の他の層によって異なる形で実装されることができる。設計が優れたコンピュータシステムでは、各層は、そのすぐ下の層についてのみ認識する（そしてその層のみに依拠する）。これは、層あるいは「スタック」（複数の隣接する層）をより上の層にマイナスの影響を与えずに、その層あるいはスタックを置き換えることを可能にする。例えば、ソフトウェアアプリケーション（上層）は、通常、それより低いレベルのオペレーティングシステム（下層）に依拠して、何らかの形態の固定記憶装置にファイルを書き込み、そ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、あるいはネットワークフォルダへのデータの書き込みの違いを理解する必要がない。この下層が、ファイルを書き込むための新しいオペレーティングシステムコンポーネントに置き換えられる場合、上層のソフトウェアアプリケーションの動作は、影響を受けない。

【００３５】

階層化されたソフトウェアの柔軟性により、仮想マシン（ＶＭ）が、実際には別のソフトウェア層である仮想ハードウェア層を提示することが可能になる。このようにして、ＶＭは、それより上層のソフトウェア層に対して、それらのソフトウェア層が専用のプライベートなコンピュータシステムで実行されているという錯覚を作り出すことができ、したがって、ＶＭは、複数の「ゲストシステム」を１つの「ホストシステム」で同時に実行することを可能にする。このレベルの抽象化が、図２Ａの図に表される。

【００３６】

図２Ａは、コンピュータシステム内でエミュレートされる動作環境のためのハードウェアとソフトウェアアーキテクチャの論理階層化を表す図である。この図で、エミュレーションプログラム９４が、物理ハードウェアアーキテクチャ９２の上で直接または間接的に実行される。エミュレーションプログラム９４は、（ａ）ホストオペレーティングシステムと並行して実行される仮想マシンモニタ、（ｂ）ネイティブのエミュレーション機能を有する特殊なホストオペレーティングシステム、または（ｃ）エミュレーションを行うハイパーバイザコンポーネントを備えるホストオペレーティングシステム、である。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（このコンポーネントが「仮想マシン」、すなわち実際には存在せず、代わりにエミュレーションプログラム９４によってエミュレートされるハードウェアであることを表すために点線で示す）をエミュレートする。ゲストオペレーティングシステム９８がゲストハードウェアアーキテクチャ９６で実行され、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で実行される。図２Ａのエミュレートされる動作環境では、エミュレーションプログラム９４の動作により、ソフトウェアアプリケーション１００が、一般に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と互換性のないオペレーティングシステムで実行されるように設計されている場合でも、ソフトウェアアプリケーション１００が、コンピュータシステム９０で実行されることができる。

【００３７】

図２Ｂに、物理的なコンピュータハードウェア１０２の上で直接実行されるホストオペレーティングシステム層１０４を備える仮想化されたコンピューティングシステムを示し、このシステムでは、ホストオペレーティングシステム（ホストＯＳ）１０４が、そのホストＯＳがエミュレートする（あるいは「仮想化する」）ハードウェアと同じインタフェースを公開することにより、物理的コンピュータハードウェア１０２のリソースへのアクセス権を提供するが、言い換えると、これは、ホストＯＳが、その上で実行されるオペレーティングシステム層に認識されないことを可能にする。この場合も、エミュレーションを行うために、ホストオペレーティングシステム１０２は、ネイティブのエミュレーション機能を備える、特別に設計されたオペレーティングシステムであっても、あるいは、エミュレーションを行うために組み込まれたハイパーバイザコンポーネント（図示せず）を備える標準的なオペレーティングシステムであってもよい。

【００３８】

再び図２Ｂを参照すると、ホストＯＳ１０４の上には２つの仮想マシン（ＶＭ）の実装があり、ＶＭ　Ａ１０８は、例えば仮想化されたＩｎｔｅｌ３８６プロセッサであり、ＶＭ　Ｂ１１０は、例えばＭｏｔｏｒｏｌａ６８０Ｘ０プロセッサファミリのプロセッサの仮想化バージョンである。ＶＭ１０８と１１０それぞれの上には、それぞれゲストオペレーティングシステム（ゲストＯＳ）Ａ１１２とＢ１１４がある。ゲストＯＳ　Ａ１１２の上で、アプリケーションＡ１　１１６とアプリケーションＡ２　１１８の２つのアプリケーションが実行され、ゲストＯＳ　Ｂ１１４の上ではアプリケーションＢ１　１２０が実行される。

【００３９】

図２Ｂに関して、ＶＭ　Ａ１０８とＶＭ　Ｂ１１０（点線で表す）は、ソフトウェア構成としてのみ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、それらは、ＶＭ　Ａ１０８とＶＭ　Ｂ１１０をそれぞれゲストＯＳ　Ａ１１２とゲストＯＳ　Ｂ１１４に提示するだけでなく、ゲストＯＳ　Ａ１１２とゲストＯＳ　Ｂ１１４に必要なすべてのソフトウェアステップを実行して実際の物理的なコンピュータハードウェア１０２と間接的に対話する、特殊化されたエミュレーションソフトウェアが実行されるために可能になることに留意することが重要である。

【００４０】

図２Ｃに、ホストオペレーティングシステム１０４”と並行して実行される仮想マシンモニタ（ＶＭＭ）１０４’によってエミュレーションが行われる、代替の仮想化コンピューティングシステムを示す。ある実施形態では、ＶＭＭは、ホストオペレーティングシス

テム１０４の上で実行され、そのホストオペレーティングシステム１０４を通じてのみ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とすることができる。他の実施形態では、図２Ｃに示すように、ＶＭＭは、代わりに、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備えることができるが、他のレベルでは、ＶＭＭがコンピュータシステム１０２と直接対話する（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対話する方式と同じように）。さらに他の実施形態では、ＶＭＭは、ホストオペレーティングシステム１０４を利用せずに（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等の場合は、なおホストオペレーティングシステム１０４と対話するが）、すべてのレベルで直接コンピュータハードウェア１０２と対話する、完全に独立したソフトウェアシステムを備えることができる（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対話する方式と同じように）。

【００４１】

仮想マシンを実装するためのこれらの変形はすべて、本明細書に記載される本発明の代替実施形態を形成することが見込まれ、本明細書の記載内容はいずれも、本発明を特定のエミュレーション実施形態に限定するものと解釈すべきでない。また、それぞれＶＭ　Ａ１０８および／またはＶＭ　Ｂ１１０を介したアプリケーション１１６、１１８、１２０間の対話（ハードウェアエミュレーションのシナリオを想定する）の言及は、実際には、アプリケーション１１６、１１８、１２０と、仮想化を作り出したバーチャライザとの間の対話と解釈されたい。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との対話の言及（コンピュータハードウェア１０２の上で直接または間接的にコンピュータ命令を実行することを想定）は、実際には、仮想化を作り出したバーチャライザと、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との対話と適宜解釈されたい。

【００４２】

ＧＰＵおよびグラフィックパイプラインを仮想化するシステムおよび方法

すでに述べたように、本発明は、仮想マシンをグラフィックハードウェア空間に適用するシステムおよび方法を提供し、それにより、監視コードがＣＰＵで実行されるが、グラフィックの作業項目は、直接グラフィックハードウェアによって処理される。図３Ａに、いくつかの一般的な概念を具体的に示す非制限的な第１のアーキテクチャを示す。例えば、この図に示されるように、コンピューティングデバイスのグラフィックハードウェア３０２の上に階層化された仮想マシンモニタ３０４がある。この実施形態は、非制限的な形で、それぞれが仮想マシンを受け取る点でゲストＯＳに似るホストＯＳを扱う。アプリケーションＨ１およびアプリケーションＨ２を実行するホストＯＳ　３１０は、ホスト仮想マシン３０８によってホストされる。アプリケーションＡ１を実行するゲストＯＳ　Ａ３３０は、仮想マシンＡ３２８によってホストされる。アプリケーションＢ１を実行するゲストＯＳ　Ｂ３４０は、仮想マシンＢ３３８によってホストされる。非制限的な一実施形態では、監視コードは、グラフィック仮想マシンモニタとして構成される。

【００４３】

本発明の目標は、新しいオペレーティングシステムにおける古いグラフィックソフトウェアの継続的なサポートを提供することであり、本発明の各種の仮想化されたアーキテクチャがこの目標を達成する。古い機能を新しいオペレーティングシステムに持ち越す解決法は、新しいオペレーティングシステムを複雑にするだけであり、本発明のアーキテクチャに従って回避しようとする結果である。本発明により提供される解決法は、新しいオペレーティングシステムとともに古いオペレーティングシステムの仮想バージョンを実行し、古いオペレーティングシステムのインスタンスでのみ古いグラフィックソフトウェアをサポートすることを含む。これは、上記の項で説明した、Ｖｉｒｔｕａｌ　ＰＣやＶＭｗａｒｅなどの市販製品で使用される仮想マシンと仮想マシンモニタの概念を利用する。しかし、そうしたＰＣベースのアーキテクチャでは、仮想オペレーティングシステムの「パ

ーティション」にあるグラフィックソフトウェアがしばしばエミュレートされ、元の非仮想化実装に比べてはるかに低速に動作する。したがって、本発明によれば、グラフィックソフトウェアスタックが、古い（すなわちゲスト）オペレーティングシステム下で実行される一方で、基礎となるグラフィックハードウェアに直接のアクセスを持ち続けることが許される。これは、仮想マシンモニタによる高コストのエミュレーションを行わずにプロセッサで直接ゲストオペレーティングシステムを実行することができる、ＴｙｐｅＩＩ仮想マシンモニタと称されるものに似ている。

【００４４】

互いに認識されず、またグラフィックハードウェアが仮想化されていることを認識しないように、複数のオペレーティングシステムパーティションをグラフィックハードウェアで実行できるようにするために、グラフィックハードウェアでのサポートが必要とされる。そのため、本発明の実施形態によれば、グラフィック仮想マシンモニタがリソースを管理し、グラフィックハードウェアでどのパーティションが実行されるかを制御する。本発明を使用すると、複数のゲストオペレーティングシステムが並行して実行されることができ、それぞれがグラフィックハードウェアへの（ほぼ）最大速度の直接のアクセスをもつことができる。一部のゲストは、古いソフトウェアを実行するレガシーオペレーティングシステムであってよく、一方、他のゲストは、他の特定のタスクを行うことができる。

【００４５】

本発明の拡張は、セキュアな処理を提供する実施形態をもたらし、セキュアな処理は、「ホスト」または「ゲスト」のオペレーティングシステムで実行されることができる。本発明によりこれを実現することは、ゲストオペレーティングシステムが相互に干渉することができないように、グラフィックハードウェアを堅牢かつセキュアに仮想化するという有利な結果になる。ゲストオペレーティングシステムが相互に干渉することができないと、別のゲストオペレーティングシステムで悪意のあるソフトウェアがユーザに気づかれずに動作する可能性がなくなる。このシステムは、仮想マシンモニタを、信頼できる基盤、あるいは「信頼できるカーネル」と見なす。本発明の各種実施形態で、信頼できる基盤を別のゲスト（あるいはホスト）オペレーティングシステムの中に拡張して、追加的な信頼されるグラフィックサービスを提供することができる（すべてのサービスを仮想マシンモニタあるいはカーネルに入れるのではなく）。それらのサービスは、例えば、セキュアあるいは信頼されるウィンドウマネジャまたはデスクトップコンポジションエンジンとともに、ディスプレイで見えるものに対する制御を含むことができる。これは、複数のアプリケーションを混合したものを、信頼されるオペレーティングシステムパーティションと信頼されないオペレーティングシステムパーティションで実行し、統一された出力が１つの（または複数の）物理的なグラフィックデバイスを通じて提供されることを可能にする。

【００４６】

本発明によりセキュアな基盤をさらに拡張して、例えば文書、ビデオ、オーディオなどに付随するデジタル著作権を維持するためなどのコンテンツ保護を可能にする、より堅牢なシステムを提供することができる。そのようなシステムは、信頼される仮想マシンモニタと信頼されるパーティションで動作する信頼されるサービスを利用して、デジタル著作権を施行する。しかし、再生アプリケーション（メディアライブラリ管理、再生リストなど）のユーザインタフェースと他の機能は、分離し、信頼されるパーティションにあるセキュアなサービスと通信する信頼されないパーティションで実行することができる。

【００４７】

顧客が新しいオペレーティングシステムと新しいオペレーティングシステムバージョンに移る際、主として、古いソフトウェアのインタフェースと挙動を入念に保つことにより互換性が保証され、開発サイクル中の増大し続ける不利益が避けられない。設計と実装の複雑性が増大し、レガシーの挙動を不注意に壊す危険性が全開発サイクルを通じて続く。そのため、本発明は、顧客の投資を保護しながら、新しい開発から戦略的かつ安全にレガシーの挙動を摘み取ることができる代替の開発戦略を提案する。

【００４８】

アプリケーションの互換性は、仮想マシンモニタ（ＶＭＭ）技術を使用して別個の仮想マシン（ＶＭ）で新しいＯＳと同時に元のＯＳバージョンを実行することにより維持される。ホストマシンの従来のＶＭＭ技術がグラフィック処理装置（ＧＰＵ）に拡大適用されて、グラフィックアクセラレータへのハードウェアアクセスを可能にし、レガシーアプリケーションが最大の性能で動作することを保証する。本発明は、複数のアプリケーションを異なるＶＭで実行する際に、ユーザエクスペリエンスを表面上シームレスにする方法も提供する。本発明の他の態様では、ＶＭＭ技術を用いることにより、本発明の仮想化されたグラフィックアーキテクチャが拡張されて、信頼されるサービスとコンテンツ保護／デジタル著作権管理を提供する。

【００４９】

長期間にわたるソフトウェアの成功は、必然的にレガシーソフトウェアのインタフェースと挙動につながる。新しいソフトウェアの革新と下位互換性という相反する要件の結果、ソフトウェア設計サイクルがより複雑で時間を要するものになる。レガシーの選択ビットを破棄することにより、増大する複雑性を簡素化しようとする試みは、決定を評価するためにより多くのリソースを使用し、間違った決定が行われた場合はリスクを付加し、必ず十分には安心できないものとなる。最終的な結果として、コア技術の革新と提供のペースが緩慢になり、現状を乱さない革新が注目される。

【００５０】

代替の手法は、古いバージョンのオペレーティングシステムソフトウェアを持ち越し、同時に、それを新しい開発に関与させないようにすることにより、古いバージョンのオペレーティングシステムソフトウェアを採用するものである。このように、本発明は、仮想マシンモニタを使用して、別個の仮想マシンで最新のＯＳと同時に以前のＯＳバージョンを実行し、同時に、古いバージョンのインタフェースを使用するアプリケーションが、シームレスまたはほぼシームレスにそれらの設計対象の環境と同じ環境で実行され、同じように振舞う、魅力的なユーザエクスペリエンスを実現させる。

【００５１】

本発明の各種実施形態を概説する前に、一つの例が有益である。それ故に、あるＯＳの２番目のバージョンである例えばＯＳ＿Ｂには、そのＯＳの最初のバージョンである例えばＯＳ＿Ａでサポートされるグラフィックａｐｉの一部のセットがないことが望ましいものとする。しかし、目標は、ＯＳ＿ＢとＯＳ＿Ａを同時に実行することである。すなわち、ＯＳ＿Ａのグラフィックインタフェースを使用するアプリケーションがＯＳ＿Ａの仮想「ビュー」で実行され、一方、新しいアプリケーションがＯＳ＿Ｂのビューで実行され、どちらも等しく「優れた」ユーザエクスペリエンスを伴うことである。何が優れたユーザエクスペリエンスをなすかの定義は、少々漠然としているが、いくつかの基本的な原則には、（Ａ）どちらのＯＳビューを使用する時でも、ユーザ入力（キーボード、マウス等）とアプリケーション出力（表示、印刷、マルチメディア等）がシームレスであること、（Ｂ）知覚できる性能上の差がほとんど、あるいは全くないこと、が様々な形で含まれることができる。適用することができる尺度の１つは、ＯＳ＿Ａのビューで実行される任意のＯＳ＿Ａアプリケーションのパフォーマンスが、そのＯＳのネイティブの最初のバージョンで仮想モニタなしで実行される同じアプリケーションの５％以内であるべきということであり、（Ｃ）ＯＳ＿Ａの仮想ビューは、そのＯＳの最初のバージョンに使用される、変更されないソフトウェアを使用すべきということである。ゲストオペレーティングシステムへの変更が許されると、背景技術で説明したファンアウトを介して互換性を損なう危険性が再浮上するので、これは望ましい。この要件を緩和する方法の１つは、シームレスな動作を容易にするために、変更が加えられていないゲストＯＳの下で追加的なヘルパアプリケーションを実行できるようにすることによる。

【００５２】

ユーザエクスペリエンスにおけるパフォーマンスの低下を最小に抑える際の問題の１つは、グラフィックレンダリングサービスのハードウェアアクセラレーションを提供することである。グラフィックデバイスインタフェース（ＧＤＩ）のようなＡＰＩの場合は、ス

タック全体をソフトウェアでエミュレートすることが実際的である場合があるが、ハードウェアアクセラレーションに依存する新しいＡＰＩの場合は、パフォーマンスの低下が許容できない場合がある。可能性の１つは、例えば、よく知られるグラフィックハードウェアデバイスの低レベルハードウェアインタフェースのソフトウェアエミュレーション／トラッピングを使用して、ゲストＯＳからのグラフィック処理要求をホストＯＳにリダイレクトすることである。

【００５３】

この従来の手法は、高レベルの抽象化を有するデバイス（１つの要求当たりの作業が多い）、または単純なデバイス（例えばシリアルポート）にはうまくいくが、現在のグラフィックハードウェアには実際的でない。このレベルにおけるトラッピング要求は、一般にあまりにも非効率的であり（グラフィックハードウェアは１ＧＢ／秒以上の入力データを受け取るように設計される）、現在のグラフィックハードウェアは、一般に、このレベルのディテールで忠実にエミュレートするには複雑すぎる（現在のグラフィックデバイスは～２億個のトランジスタを備えることを思い出されたい）。過去には、この手法は、トランジスタが１０００万個未満のデバイスと、それほどグラフィック的に集約的でないアプリケーションには実際的であったかもしれないが、そうした状況においても、必要とされる程度のパフォーマンスはもたらされなかった。

【００５４】

別の選択肢は、ストリーム中の高い（デバイスに依存しない）ポイントでグラフィックプロトコルストリームをインターセプトし、そのストリームをホストＯＳにリダイレクトするものである。これは、複雑なデバイスをエミュレートすることにより問題を解消するが、高帯域幅のプロトコルストリームが関連する場合は、結果として重大なパフォーマンス上の問題が生じる可能性がある。結果的に生じる可能性のある別の問題は、ゲストＯＳに大幅な変更を加えなければプロトコルストリームがインターセプトに適さないシナリオに関連する。手法の１つは、グラフィックＡＰＩを含むシステムダイナミックリンクライブラリ（ＤＬＬ）を、ゲストＯＳにリダイレクトするプロトコルスタブに置き換えるものであるが、元のＡＰＩの状態を管理する方法がこの手法を意図して設計されていない場合には、この手法にはなお大きな問題がある。リモートで実行される（「ｒｅｍｏｔｅされる」）ように設計されたＡＰＩは、恐らくは、これを実現するために使用することができる適当な挙動を有するが、すべてのグラフィックＡＰＩおよびメディアＡＰＩがこの解決法を念頭に入れて設計されている訳ではない。この手法も、システムレベルのコンポーネント変更する（置き換える）ことを必要とし、したがって互換性のリスクを生じさせる。

【００５５】

本発明の各種実施形態の手法は、この問題をハードウェアに押し込め、ゲストＯＳが、ＣＰＵに使用されるＶＭＭ技術に似た直接のハードウェアアクセスを持つことができるようにし続けるものである。この技術は、効率的に実施された場合いくつかの利点がある。第１の利点は、ゲストＯＳに変更を加える必要性がなくなることである。第２の利点は、ゲストＯＳがハードウェアでネイティブに実行された場合と、同じ、あるいはほぼ同等のパフォーマンスレベルを提供することである。また、この方式では、グラフィックハードウェアベンダがレガシーＯＳでの新しいハードウェアのサポートを提供し続ける限り、レガシーＯＳが新しいグラフィックハードウェアをサポートできることにも留意されたい。したがって、本発明の各種の仮想化アーキテクチャでは、仮想化することができるグラフィックハードウェアが構築され、ホストＯＳとＶＭＭは、ハードウェアを効率的に利用するような構造とされ、プレゼンテーション（表示）は、複合デスクトップに組み込むことができるように仮想化される。

【００５６】

このように、本発明は、パイプラインを端から端まで仮想化するために、各ＶＭが、レンダリングリソースにアクセスし、ＧＰＵの上にＶＭＭ環境を作り出すことを可能にする。詳細には、これには、複数のグラフィッククライアントのためにＧＰＵを効率的に仮想化するという概念が含まれる。各種実施形態で、本発明の仮想化アーキテクチャは、次の

要件、すなわち、明確に定義された実行状態（コンテクスト）、短い待ち時間のコンテクスト切り替えサポート、保護された仮想アドレス空間、要求時ページングされるメモリの管理（ｄｅｍａｎｄ　ｐａｇｅｄ　ｍｅｍｏｒｙ　ｍａｎａｇｅｍｅｎｔ）、および特権のある実行モード、の１つまたは複数に準拠する。これらの要件が使用されて、複数のクライアント（アプリケーション）間のＧＰＵハードウェアリソースの効率的で、セキュアで、堅牢な共有を提供する。

【００５７】

仮想化の説明を簡単にするために、本発明により意図される仮想化サポートの程度を要約することが参考になる。最も単純なたとえは、複数のクライアント（アプリケーション）間でＣＰＵを共有するために現在のＣＰＵに見られるレベルとほぼ同じレベルのＧＰＵ共有があるべきであるというものである。これは、ＧＰＵの発展の当然の結果である。すなわち、ＧＰＵは、複数の独立したクライアント間で効率的に共有されるべき、強力な計算リソースである。ＣＰＵとの違いの１つは、ＧＰＵは、ＧＰＵではなくＣＰＵで実行される実行可能コードによって管理されることである。それに対し、一部のドライバモデルでは、カーネルグラフィックサブシステムが、スケジューリングとメモリ管理のサービスと、ビデオディスプレイの構成、電力管理などの大域リソース（ｇｌｏｂａｌ　ｒｅｓｏｕｒｃｅ）に対する制御を提供するミニエクゼクティブ（ｍｉｎｉ－ｅｘｅｃｕｔｉｖｅ）として機能する。このミニエクゼクティブは、カーネルエクゼクティブとは別個に動作し、ＧＰＵクライアントの要件に合わせられたアルゴリズムとポリシーを用いて、独自のスケジューリングとメモリ管理の決定を行う。

【００５８】

仮想化の必要条件が満たされると、クライアントドライバをＧＰＵハードウェアに直接アクセスさせることが可能になる。この際の規則は、他のカーネル機能を仮想化する際の規則と同様であり、すなわち、ホストされるドライバはそのドライバがＶＭＭ上で実行されていることを検出することができてはならず、また、ＶＭＭによって導入されている保護機構のいずれも回避することができてはならない。この説明では、ＧＰＵ　ＶＭＭはカーネルＶＭＭに埋め込まれており、すべての実用的な目的のためにカーネルＶＭＭにおいてドライバとして動作するものとする。グラフィックＶＭＭは、デバイスに依存しない部分とデバイスに依存する部分を含むことができる（後者はグラフィックハードウェアの製造者によって提供される）。

【００５９】

図３Ｂに、本発明によりグラフィックパイプラインを仮想化するために提供される別の非制限的な例示的アーキテクチャを示す。仮想マシンモニタ３０４は、本発明によるグラフィック機能と管理を扱うコンポーネントグラフィックＶＭＭを含む。ホストグラフィックドライバＨＧＦを有するホストＯＳ、グラフィックドライバＡＧＤを有するゲストＯＳ　Ａ、およびグラフィックドライバＢＧＤを有するゲストＯＳ　Ｂはいずれも、ＶＭＭ３０４を介してグラフィックＶＭＭと対話することができる。

【００６０】

ＶＭＭを使用する動機の１つは効率性なので、ハードウェアサポートとともにＴｙｐｅ　ＩＩのグラフィックＶＭＭを直接実装することは理に適う。高機能のディスプレイドライバモデル（ＤＤＭ）を提供することで、アクセスが制御された仮想アドレス空間、プリエンプション（ｐｒｅｅｍｐｔｉｏｎ）、および特権動作に対するハードウェアサポートを追加する。効率的なＶＭＭサポートについてのリトマス試験の１つは、ドライバに変更を加えずに、グラフィックＶＭＭの上で既存の高機能ＤＤＭドライバを実行するものである。これは、カードの初期化、ページテーブルの設定、ディスプレイコントローラのプログラム、ＰＣＩアパチャ（ａｐｅｒｔｕｒｅ）のセットアップ、物理アドレスとして解釈された入力の操作等を行うグラフィックサブシステムのカーネル動作が、効率的な解釈のためにＶＭＭへのトラップを生成すべきことを意味する。割り込みは、ＶＭＭによってインターセプトされることができ、該当する（ゲスト）ドライバに転送されるべきである。レンダリング動作などの頻度の高い特権動作は、ＶＭＭによる介入なしにグラフィックハー

ドウェアによって実行できるべきである。

【００６１】

高機能ドライバモデルハードウェアは、複数の実行コンテクストをサポートし、各コンテクストは、別個の仮想アドレス空間と、コマンドストリームの入力ＤＭＡバッファを管理するための入力キュー（リングバッファ）を備える。ローカルおよびシステムの物理的なメモリリソースは、その物理アドレスをポイントするようにページテーブルエントリを初期化することにより、あるコンテクストの仮想アドレス空間にマッピングされる。コマンドストリーム動作の完了は、メモリへの書き込み（フェンス）と割り込みを使用して伝えられる。

【００６２】

メモリ保護を管理する方法の１つは、ローカルのグラフィックメモリ物理アドレス空間を分割し、各仮想マシン（ゲストＯＳ）に、固定された連続するパーティションを割り当てるものである。割り当ての外側にあるメモリは、仮想マシンには見えない。この機構は、グラフィックＶＭＭによってプログラムされる規定レジスタとリミットレジスタを通じて、単純なレベルの間接性を使用して実施することができる。これには、ドライバに見えるＰＣＩバスコンフィギュレーションレジスタの内容（ＶＭＭによってトラップされる読み出し）を制御することが必要となる場合がある。このメモリ管理方式では、各ゲストは、そのゲストの存続期間にわたって固定されたメモリ割り当てを見る。ハードウェアに見える物理アドレス空間のウィンドウがＶＭＭによって入念に制御されるので、これは、ＶＭＭがページテーブル操作（頻繁に行われる場合がある）をトラップする必要をなくす。これは、恐らくはメモリリソースのあまり効率的でない利用を提供するが、ＶＭＭを詳細なメモリ管理から開放するので、仮想化メモリのサポートを実施するのに妥当な最初の手法である。

【００６３】

長期的にはＶＭＭを詳細なメモリ管理動作により関与させることが予想できる。ページテーブルエントリをプログラムするためにドライバによって使用される動作は、ＶＭＭにトラップされることになるので、この部分は、より複雑になる。ページングのパフォーマンス目標に応じて、ハードウェア設計は、それらの動作を検出し、エミュレートする効率を向上するようにより入念に設計される必要がある可能性がある。例えば、複数のマッピングが１つのグループとして指定されることができ、グラフィックＶＭＭでそれらのマッピングが容易に認識され、１つのグループとして復号されることができるように、ページマッピング動作（コマンド）を設計することが有益である場合がある。これは、ＶＭＭを通じたトラップの回数を最少にする。また、ＶＭＭは、ゲストパーティションにある（信頼される）サービスを使用して、補助記憶装置（ゲストＤＭＭドライバが実装するものを超えた、別のレベルの補助記憶装置）へのページング動作をサポートすることもできる。この方式はより複雑であるが、各ゲストがすべてのハードウェアメモリリソースを利用することができる。

【００６４】

本発明の上記の実施形態に関して記載される方式は、ソフトウェアで仮想化されたページテーブルの使用を示唆する。ソフトウェアで仮想化されたページテーブルに替わる手法は、ＶＭＭが直接使用するグラフィックハードウェアに、特権のあるページテーブルの別のセット、すなわち別のレベルの間接性を追加するものである。これは、固定されたパーティションについて述べた基底（ｂａｓｅ）レジスタとリミットレジスタの一般化であり、ＶＭＭが、ページレベルの細かさ（ｇｒａｎｕｌａｒｉｔｙ）で物理リソースのグラフィックハードウェアのビューを制御できるようにする。

【００６５】

ここまでのメモリ管理の論議は、グラフィックコンテクストへのローカルグラフィックメモリのマッピングの管理に注目した。この動作も物理メモリアドレスを使用するので、システムメモリページをハードウェアにマッピングする際には同様の要件がある。プロセッサＶＭＭが物理メモリを管理する方式が重要な要素となる。一般に、物理アドレスを使

用するデバイスは、それが行うマッピングに対する完全な制御を保持するように入念に管理される。この場合も、利用できるシステムメモリのウィンドウを制限するグラフィックハードウェア制御を追加することにより、ビデオメモリに提案される単純な固定パーティション方式を、システムメモリに機能するように拡張することができる。しかし、プロセッサＶＭＭが、ゲストに割り振り動作をサービスする際に連続しない物理メモリを割り当てる場合には、これは適用することができない。これは、システムメモリリソースについては、詳細なメモリ管理のためのより複雑な方式が実施されなければならないことを意味する（それは同時にローカルのグラフィックメモリ解決を行うことを示唆する）。物理メモリの管理と、入念に調べる必要があるグラフィックＶＭＭとの潜在的な対話に関して、プロセッサＶＭＭによってなされる何らかの追加的な仮定がありうる。

【００６６】

割り込みをより効率的に処理できるようにするために、割り込みのタイプを、ＶＭＭに処理される必要がある割り込みと、そうでない割り込みに区分することができる。後者のカテゴリは、コマンドの完了などを含むことができ、そうした割り込みは、ゲストドライバによって直接処理されることができる。

【００６７】

もう一つの主要な管理カテゴリは、処理リソースのスケジューリングである。高機能ドライバモデルは、コンテクストのプリエンプションをサポートし、そのため、最も単純な管理モデルでは、ＶＭＭに時間スライスをパーティションに割り当てさせ、現在実行されているパーティションからの現在実行中のグラフィックコンテクストを優先する。現在の高機能ドライバモデルは、実行する新しいコンテクストリストを指定する際にダブルバッファリングされた実行リストを使用する。保留中の実行リストは１つ以上はないという前提がある。仮想化されたスケジューリングモデルは、ＶＭＭが実行リストの境界でパーティションを切り替えることを容易にし、さらに、新しいパーティションへの遷移が行われる時には、コンテクスト状態が古いパーティションに保存され、新しいパーティションから復元されることを保証するためにいくつかの変更を必要とする（例えば、コンテクストの保存と復元の際の適切な時に、該当するパーティションのメモリリソースがマッピングされる）。この場合も、前提と、考慮されるべきプロセッサＶＭＭのスケジューラとの対話がある。

【００６８】

グラフィックＶＭＭが様々なコンテクストの作業量と優先順位をより認識する、より精緻なスケジューリング方式を実施することができる。その情報をプロセッサＶＭＭが入手できるようにして、よりインテリジェントなスケジューリング決定を行うことができる。グラフィックＶＭＭが、プロセッサＶＭＭが実行されているパーティションと異なるパーティションからコンテクストを実行することができるかどうかに関して、いくつかの興味深い設計上の問題がある。ハードウェアの利用を最大にするために、パーティション間の遷移の際にいくらかの処理の重複を許すことが可能であり、必要な場合がある。しかし、それには、遷移の際にリソースがどのように管理され、マッピングされるかについてのより入念な検討が必要となる。

【００６９】

プレゼンテーションと対話の仮想化に関しては、第１の実施形態で、グラフィックハードウェアが仮想化されると、各ゲストは、そのゲストのデスクトップを作成し、そのデスクトップにゲストアプリケーションからのウィンドウをレンダリングする。このゲストデスクトップは、ローカルのグラフィックメモリに記憶され、通常の環境では、表示装置（ＣＲＴやフラットパネル）にスキャンアウトされる。仮想化された環境では、複数のゲストデスクトップの内容がともに集められ、１つのディスプレイに表示される。この表示を作成するには、複数のデスクトップの提供（１度に単一のデスクトップを表示する）、専用ウィンドウへのデスクトップの提供、または統一された単一のデスクトップ等のいくつかの手法がある。

【００７０】

第１の実施形態では、１つのゲストデスクトップが「現在の」デスクトップとして選択され、表示される。ユーザは、何らかのＵＩコントロールを使用して、異なるゲストの中から選択することができるが、一度に１つのみのゲストが表示される。この方式は、個々のゲストデスクトップ画像を単に表示することができるので、最も単純である。実際は、ＵＩコントロールが、どのゲストがディスプレイコントローラのスキャンアウトレジスタを設定することを許されるのかを決定する。

【００７１】

図４Ａに、「一度に単一のデスクトップ」を表示する実施形態を示す。各仮想マシンは、ローカルのグラフィックメモリにその仮想マシンのデスクトップ空間を受け取り、各空間は他の空間から独立しており、それにより、ユーザによって選択された「アクティブな」デスクトップだけが表示のためにスキャンアウトされることが理解できよう。そのため、ゲストは、各自の個々の作業項目をラスタ化し、それをメモリのいずれかの場所に残し、プレゼンテーションエンジンが、どのメモリ（ラスタ化されたビットを含む）が表示されるかを制御することができ、この制御には、内容のサイズを変更する、または内容を移動するための追加的なラスタ化処理が関連する場合もある。

【００７２】

当然のことながら、この第１の実施形態では、複数のデスクトップを同時に見えるようにはしない（ただしｍｕｌｔｉｍｏｎ構成では、複数の異なるディスプレイコントローラが、すべて同じゲストからのデスクトップではなく、異なるゲストからのデスクトップを表示することができる）。代替の手法は、ホストデスクトップ上の別々のコンテナウィンドウに個々のデスクトップが表示される第２の実施形態である。この手法は、各ゲストデスクトップの内容へのアクセス権を有するホストで実行されるサービスを使用する。このサービスは、ゲストデスクトップを入力として使用してホストデスクトップを構成する複合動作を行う。各ゲストデスクトップにプロキシウィンドウが使用され、デスクトップのサイズと位置を制御する。ゲストデスクトップの内容は、そのデスクトップのディスプレイコントローラ設定への変更をインターセプトすることにより、ビデオメモリの内容から直接抽出することができる（その変更を使用してメモリアドレス、大きさ、および他のパラメータを判定する）。複合ホストデスクトップは、デスクトップコンポジションエンジン（ＤＣＥ）を使用して一般のデスクトップが構築されるのとほぼ同じように構築されるが、ハードウェアグラフィックアクセラレーションも使用する。この第２の実施形態を第１の実施形態と組み合わせて、ゲストデスクトップの単一の直接的なビューを伴うモード、または別個のウィンドウでの複数のデスクトップの複合ビューを伴うモードをサポートすることができる。

【００７３】

図４Ｂに、代替の「ウィンドウ内のデスクトップ」の実施形態を示し、この実施形態では、隠蔽を考慮するプレゼンテーションエンジンにより、すべてのデスクトップが同時に表示されることができる。例えば、ホストデスクトップは、デフォルトで、ディスプレイ上の背景デスクトップウィンドウとすることができ、それぞれが専用のコンテナウィンドウに表示されるゲストデスクトップは、背景デスクトップウィンドウの上に表示される独立したウィンドウを受け取る。

【００７４】

この第２の実施形態は、ゲストデスクトップのウィンドウの位置とサイズを同期する際に、入力の調整に関していくらかの複雑化させ始める。ゲストデスクトップについてホストコンテナウィンドウでウィンドウが操作されると、それらの対話がトラップされ、処理のためにゲストに送られる。ゲスト、例えばゲストのウィンドウマネジャがゲストデスクトップを更新し、新しいウィンドウ位置が再構築と同時にホストデスクトップで反映される。このタイプの入力調整は、「ウィンドウ中のデスクトップ」を実装する他の仮想化製品ですでに扱われており、グラフィックハードウェアの仮想化に影響されるべきでない。

【００７５】

第２の実施形態は、大きな改良であるが、複数のゲストデスクトップからのウィンドウ

を混合する、すなわちゲストデスクトップがある事実を隠すことがなお望ましい。この問題は、各ゲストデスクトップの個々のウィンドウ内容を見つけることに相当するが、個々のウィンドウ情報は、グラフィックハードウェアに送られるデータのストリームから容易にはインターセプトされないので、デスクトップ内容を見つけることよりも難しい。しかし、ゲストオペレーティングシステムの変更に関する規則に意図的に違反する場合には、いくらかのフックを加えて、個々のウィンドウのピクセル（メモリ位置）を識別することを助けることができる。ここで取ることが可能な異なる手法がいくつかある。

【００７６】

１つの手法は、ゲストデスクトップのウィンドウをタイル表示状態された状態（重なり合わない）に保って、常に全内容を利用できるようにし、同時に、ホストウィンドウマネジャ／コンポジションエンジンが使用するために、ゲストデスクトップの各ウィンドウのサイズと位置についての情報を保持するものである。この手法は、２つの別個のウィンドウ情報セット、すなわち仮想の情報（ホストデスクトップ）とゲストデスクトップ上の物理的な位置、を維持することを伴う。仮想ウィンドウを操作する動作は、物理ウィンドウに影響する場合もしない場合もあるので（例えば仮想ウィンドウが隠蔽される場合は、ゲストデスクトップの領域（ｒｅａｌ　ｅｓｔａｔｅ）を再要求することが有益である場合がある）、この方式も入力の調整を複雑にする。仮想ウィンドウ内での対話は、物理ウィンドウに適した対話に変換される（これは入力座標へのオフセットと同程度に簡単である）。

【００７７】

これに替わる手法は、ゲストデスクトップウィンドウを、管理するのがより容易なメモリにリダイレクトするものである。これは、別個の「バックバッファ」が各ウィンドウに維持される、グラフィックデバイスインタフェース（ＧＤＩ）について行われるリダイレクトに似る。それらバックバッファの内容は、ホストウィンドウマネジャ／コンポジションエンジンが利用できるようにされる。そして、ユーザが仮想ウィンドウの内容と対話すると、仮想の入力座標と物理的な入力座標との間で同様のマッピングが行われる。

【００７８】

いくらかの洞察と計画により、そのオペレーティングシステムがゲストとして実行される時にそのようなインターセプトを容易にするように将来のオペレーティングシステムを形成することが可能になる。各ゲストデスクトップのシェルの配置（ユーザから見てシェルが存在しなくなる場合）、シェル拡張の処理、スタートメニューのオプション、アプリケーションのショートカット、クリップボードなどが、統一されたデスクトップビューに併合される。これらは解決することが可能な問題であるが、各ゲストにおける対応する実装についてのより多くの詳細を理解し、恐らくは（静的に）その情報を抽出し、ともに併合して統一されたビューを生成することを必要とする。統一されたアクセシビリティのサポートについても同様の問題が生じる。

【００７９】

同様の問題は、例えば記憶、インストールされたアプリケーション、ネットワーキングなど、複数ゲストシステムの他の部分のシームレスなビューを提供する際にも生じる可能性がある。独立したフィンガープリントを有するウィンドウ式のデスクトップとして複数のゲストを公開する暫定的な戦略は、堅牢な仮想化インフラストラクチャを構築する作業に集中することを可能にする。それと同時に、信頼される処理と、デジタル著作権管理のためのコンテンツ保護など、いくつかの追加的な機能に対応することができる。

【００８０】

セキュアな（信頼される）グラフィック処理の基盤として、上記の仮想化されたグラフィックシステムは、仮想マシンを使用するセキュアな処理のための他のモデルと符合する。仮想化されたグラフィック環境の特性は、信頼される処理と信頼されない処理を混合したものを、信頼される方式で同じシステムで最大限のパフォーマンスで行うことを容易に可能にする。そのようなモデルでは、グラフィックＶＭＭがグラフィックデバイスの信頼性を堅牢に確認し、ブートストラップすることが必要とされる。無論、そのようなモデル

は、他の信頼されるコンポーネントに依拠してグラフィックＶＭＭをセキュアにブートストラップする。仮想化環境は、複数のゲストとホストパーティションを互いから堅牢に保護するので、信頼されるグラフィック処理を行うことができる、信頼されるパーティションを構築することができる。信頼されるパーティションでウィンドウマネジャとコンポジションエンジンがサービスとして実行される場合、それらは、上述した表示方法を使用して、信頼されるアプリケーションデータと信頼されないアプリケーションデータを信頼できる方式で混合することができる。

【００８１】

図４Ｃに、本発明による、信頼されるアプリケーションデータと信頼されないアプリケーションデータを混合するための例示的アーキテクチャを示す。先に述べたように、グラフィックドライバＨＧＤを有するホストＯＳ　Ｈと同様に、グラフィックドライバＴＧＤを有する信頼されるパーティションとして信頼されるＯＳ　Ｔが提供される。信頼されるＯＳ　Ｔは、ウィンドウマネジャあるいはウィンドウコンポジタを含み、これは、グラフィックパイプラインによるレンダリングの前に、信頼されるデータと信頼されないデータの記憶とレンダリングについての制約に関するグラフィックＶＭＭコードから入力を受け取るコンポーネントである。

【００８２】

信頼されるサービスは、独立したゲストデスクトップ表示データへのアクセスを必要とするが、この情報は、コンポジションサービスによって制御データと解釈されない。この情報は、レンダリングの入力として使用され、そのため共有されるデータの内容からの攻撃の危険性はない。グラフィックハードウェアが適正に仮想化された場合、１つのパーティションのコードが別のパーティションのコードを攻撃する危険性はない。ゲストドライバは、ハードウェアの制御権を得ることも、他のゲストに干渉することもできない。グラフィックハードウェアの内側で実行されるアプリケーションコードは、同じゲストパーティションは言うまでもなく、他のゲストパーティションの他のアプリケーションに干渉することはできない。

【００８３】

本発明の仮想化のサポートでは、ハードウェア認証、ディスプレイ出力に対する堅牢な制御、または１つの信頼されるパーティションが表示装置に表示データを確実に引き渡すことを保証するための他の要件の必要性はなくならないが、複数のパーティションが、パフォーマンスにほとんど影響を与えずにグラフィックハードウェアを堅牢に共有することが可能になる。

【００８４】

本発明による処理および表示の際のコンテンツ保護の基盤に関しては、コンテンツを保護する「保護されたビデオパス」を使用して、フルモーションビデオデータのデジタル著作権の施行が現在実現されている。保護されたビデオパスは、次の方法のいくつかを統合して、ソースデバイスから表示デバイスへのビデオデータのセキュアな移送と処理を提供する。

【００８５】

（１）カーネルデバッガ等を許可しないようにカーネル環境が強化され、保護されるコンテンツの再生中は、信頼されるコンポーネントだけがカーネルモード（「リング０」）で動作することを許される。

【００８６】

（２）保護されるコンテンツを扱うユーザモードプロセスが外部からの攻撃からさらに堅固にされる（保護された環境）。システムまたはグラフィックメモリにステージされた（またはそれらから表示される）ビデオデータには、他のプロセスからアクセスすることができない。

【００８７】

（３）ユーザからアクセス可能なバスを通じて送られるビデオデータは、暗号化される。

【００８８】

（４）グラフィックドライバは、保護されるコンテンツを再生することを認定されなければならず、グラフィックハードウェアが保護されたコンテンツの有効なシンクであることを検証する。

【００８９】

（５）ディスプレイ出力の保護がグラフィックハードウェア（Ｍａｃｒｏｖｉｓｉｏｎ、ＣＧＳＭ－Ａ、ＨＤＣＰ）によってサポートされ、メディア再生サブシステムによって堅牢に操作されることができる。

【００９０】

機構（１）と（２）は、本発明により、より一般的なセキュアなコンピューティングベースに置き換えることができる。このセキュアなコンピューティングベースは、セキュアなブートストラッププロセスを破壊できないようにすることにより、仮想化技術を通じた迂回をさらに防止すべきである。機構（３）～（５）は、ビデオデータがシステムメモリからグラフィックハードウェアを通じてディスプレイに移動する際に、ビデオデータのセキュアな処理と移送を引き続き提供することをなお必要とされる。本発明のグラフィックハードウェア仮想化方法は、グラフィックパスに関連する特定のコンテンツ保護方法を置き換えることはせず、代わりに、それらの方法は、信頼されるグラフィックサービスを構築する基礎の役割を果たす。それらの信頼されるサービスは、保護されるメディア処理コンポーネント（メディア相互動作ゲートウェイ（Ｍｅｄｉａ　Ｉｎｔｅｒｏｐｅｒａｂｉｌｉｔｙ　Ｇａｔｅｗａｙ）、略してＭＩＧ）を含むように拡張することができ、それらを悪意あるゲストパーティションではなく「安全な」信頼されるパーティションで実行する。既存の保護されるビデオのインフラストラクチャは、すでに、保護されない処理（ＧＵＩの再生アプリケーションなど）から、保護される処理（権利の交渉、ポリシー、復号、解読）を分離している。この分離は、仮想化環境でも変わらず、保護されるコンポーネントは信頼されるパーティション（異なる［より良好な］保護環境）で実行され、保護されないコンポーネントは、信頼されないゲストパーティションに残る。

【００９１】

したがって、図４Ｄに、ゲストＯＳ　Ａで実行されるメディアプレーヤアプリケーションによってレンダリングされるコンテンツのための別個の保護層としてメディア相互動作ゲートウェイＭＩＧを含むように、図４Ｃとの関連で開発された信頼されるパーティションＴの概念を拡張した場合を示す。

【００９２】

Ｖｉｒｔｕａｌ　ＰＣ製品が多数のＯＳを実行できるのと同様に、本発明の仮想化されたグラフィック環境は、所定数のオペレーティングシステムまたはバージョンのためのドライバ／ドライバモデルをサポートすることができる。

【００９３】

各オペレーティングシステムの成功は、そのオペレーティングシステムが実行することができるアプリケーションの幅と、そのアプリケーションの効率で測ることができる。ゲストオペレーティングシステムを変更しようとする欲求を容赦なく回避することにより、ネイティブに実行されるオペレーティングシステムの下で実行される仮想化されたオペレーティングシステムの下で、どのようなアプリケーションでも実行することができるはずである。違いがある可能性のある箇所は、リソースの区分方法である。例えば、あるパーティションにはリソースのサブセット（ローカルグラフィックメモリの一部）のみが利用できるようにされる場合、アプリケーションは、同程度に良好に動作しない可能性がある。将来のオペレーティングシステムでは、「一次（ｆｉｒｓｔ　ｏｒｄｅｒ）」の仮想化、すなわち複数のアプリケーションで共有するためのリソースの仮想化が標準になるので、この事は、それほど問題とならない可能性がある。

【００９４】

完全に仮想化されたグラフィックハードウェアへと向かう発想は、ソフトウェア会社にとっては、レガシーコードの問題から開放され、一方で、継続的なアプリケーションの互換性のための実際的な道筋を提供することである。副次的な結果として、本発明は、信頼

されるグラフィック処理とコンテンツ保護のための優れた環境も提供する。

【００９５】

この２番目の成功の尺度は、前進するアプリケーションは、レガシーコードを使用するのをやめ、同時に前進しなければならないことを意味する。大きなアプリケーションコードベースの場合、これは、過度に難しい場合がある。例えば、ＭｉｃｒｏｓｏｆｔのＯｆｆｉｃｅのコードベースのようなＯＳバージョンとともに発展して行くレガシーアプリケーションのコードベースが、密接にそのＧＤＩソフトウェアに依存する場合には、ＧＤＩソフトウェアをレガシーコードとして捨てていくことができない場合がある。そのような場合は、検討することが可能な混合モデル（ｈｙｂｒｉｄ　ｍｏｄｅｌ）がもしかすると存在する可能性がある。例えば、ゲストオペレーティングシステムを一般的なサービスプロバイダと見なすことができる。再びＧＤＩを例に使用すると、レガシーオペレーティングシステムは、ＧＤＩレンダリングサービスを提供するものと見なすことができる。アプリケーションは、そのサービスを使用するリモーティングに似た機構を使用する。そのような場合の目標は、必ずしも、コードを変更せずにアプリケーションを実行することではなく、アプリケーションが、完全にＡＰＩから離れて移行する時間を稼ぐ間に、パーティション間サービスとしてのレガシーＧＤＩコードを使用し続けられるようにするための最小限の変更を行うことである。

【００９６】

図４Ｅに、例示的な対話を示し、ここでは、ホストＯＳ　Ｈで実行されるゲストＯＳ　Ａのレガシーアプリケーションが、ゲストＯＳ　Ａのレガシーサービスコンポーネントと対話して、ホストＯＳ　Ｈの観点から見てなおレガシーの挙動をエミュレートする。

【００９７】

図５に、ゲストＯＳで実行されるアプリケーションが、本明細書に図示あるいは記載される仮想化アーキテクチャにより、グラフィックサブシステムのリソースに関連するコマンドを発行する例示的なシーケンスを示す。５００で、ゲストＯＳで実行されるアプリケーションが、グラフィック作業項目（画面の座標「ｘ，ｙ」に青い円を描画するなど）を出力する。５１０で、ゲストＯＳがその要求を処理し、その要求をグラフィック作業キューとそれに関連付けられたスケジューラに送る。５２０で、グラフィック作業項目が実行可能な状態になる（スケジュールされる）と、そのグラフィック作業項目データへのポインタが、グラフィックドライバによりハードウェアに渡される。５３０で、そのグラフィック作業項目が特権動作である場合は、グラフィックＶＭＭが作業要求を受け取り、ポリシー（ゲストＯＳ対ホストＯＳ、メモリポリシー、コマンドのタイプなど）に基づいて処理する。グラフィック作業項目が特権動作でない場合は、代わりに５４０で、グラフィックハードウェアが、直接要求を処理する。

【００９８】

このように、本発明は、仮想化環境で実行される複数のゲストオペレーティングシステムに、高パフォーマンスのレンダリングの利益を提供することを目的として、グラフィックアクセラレータに合わせてプロセッサハードウェアを仮想化する。そのような環境を使用していくつかの問題を効率的に解決することができる。（Ａ）仮想マシン環境のレガシーオペレーティングシステムでレガシーアプリケーションを実行することにより、アプリケーションの互換性を維持しながら、新しいオペレーティングシステムの開発の際にレガシーコードを排除する。（Ｂ）信頼される処理と信頼されない処理を同じ仮想化環境で安全に混合するインフラストラクチャを提供する。（Ｃ）信頼される環境を利用して、デジタル著作権管理のためのコンテンツ保護を提供する。

【００９９】

例示的なネットワーク環境および分散環境

当業者は、本発明は、コンピュータネットワークの一部として、または分散コンピューティング環境に配置することができるコンピュータあるいは他のクライアントまたはサーバデバイスとの関連で実施できることを理解することができよう。この点で、本発明は、本発明によるグラフィックパイプラインの仮想化との関連で使用することができる任意数

のメモリまたは記憶装置と、任意数の記憶単位またはボリュームにまたがって行われる任意数のアプリケーションおよびプロセスを有するコンピュータシステムまたは環境に関する。本発明は、リモートまたはローカルのストレージを有する、ネットワーク環境あるいは分散コンピューティング環境に配置されたサーバコンピュータとクライアントコンピュータを有する環境に適用することができる。本発明は、リモートまたはローカルのサービスとの関連で情報を生成、受信、送信するためのプログラミング言語機能、解釈（ｉｎｔｅｒｐｒｅｔａｔｉｏｎ）、および実行機能を備える独立型のコンピューティングデバイスにも適用されることができる。ゲーム環境では、グラフィックパイプラインは、特に、ネットワーク環境または分散コンピューティング環境で動作するコンピューティングデバイスに関連し、そのため、本発明によるグラフィックパイプラインの仮想化は、そうした環境で多大な有効性をもって適用することができる。

【０１００】

分散コンピューティングは、コンピューティングデバイスおよびシステム間での交換によるコンピュータリソースとサービスの共有を提供する。そうしたリソースとサービスには、情報の交換、ファイルのキャッシュ記憶およびディスク記憶が含まれる。分散コンピューティングは、ネットワーク接続を利用して、複数のクライアントがその集合的な力を活用して企業全体を利することを可能にする。この点で、各種のデバイスが、本発明のグラフィックプロセスに関連する可能性のあるアプリケーション、オブジェクト、またはリソースを有することができる。

【０１０１】

図６Ａに、例示的なネットワーク環境または分散コンピューティング環境の略図を提供する。この分散コンピューティング環境は、コンピューティングオブジェクト１０ａ、１０ｂ等と、コンピューティングオブジェクトまたはデバイス６１０ａ、６１０ｂ、６１０ｃ等を備える。これらのオブジェクトは、プログラム、メソッド、データストア、プログラマブルロジック等からなる。オブジェクトは、ＰＤＡ、オーディオ／ビデオデバイス、ＭＰ３プレーヤ、パーソナルコンピュータなど、同じデバイスまたは異なるデバイスの一部からなってもよい。各オブジェクトは、通信ネットワーク１４を利用して別のオブジェクトと通信することができる。このネットワークはそれ自体、図６Ａのシステムにサービスを提供する他のコンピューティングオブジェクトおよびコンピューティングデバイスからなることができ、また、それ自体が、複数の相互接続されたネットワークに相当することができる。本発明の一態様によれば、各オブジェクト１０ａ、１０ｂ等、または６１０ａ、６１０ｂ、６１０ｃ等は、ＡＰＩ、または他のオブジェクト、ソフトウェア、ファームウェアおよび／またはハードウェアを利用して本発明のグラフィック仮想化プロセスの使用を要求する可能性のあるアプリケーションを含むことができる。

【０１０２】

６１０ｃなどのオブジェクトは、別のコンピューティングデバイス１０ａ、１０ｂ等、または６１０ａ、６１０ｂ等でホストされることができることも理解できよう。したがって、図の物理的環境では、接続されたデバイスをコンピュータとして示すが、このような図示は例示的なものに過ぎず、物理的環境は、これに代えて、ＰＤＡ、テレビ、ＭＰ３プレーヤなどの各種のデジタルデバイス、インタフェース、ＣＯＭオブジェクトなどのソフトウェアオブジェクトを備えるものと図示または説明されることができる。

【０１０３】

分散コンピューティング環境をサポートする各種のシステム、コンポーネント、およびネットワーク構成がある。例えば、コンピューティングシステムは、有線システムまたは無線システム、ローカルネットワーク、または広域分散ネットワークによりともに接続することができる。現在、ネットワークの多くはインターネットに結合されているが、インターネットは、広域分散コンピューティングのためのインフラストラクチャを提供し、多くの種々のネットワークを包含する。そのインフラストラクチャはいずれも、本発明のグラフィックハードウェア仮想化プロセスに関連して行われる例示的通信に使用されることができる。

【０１０４】
家庭のネットワーク環境では、電線、データ（有線および無線）、音声（電話など）、および娯楽媒体など、それぞれが固有のプロトコルをサポートすることができる少なくとも４つの異種のネットワーク移送媒体がある。電灯のスイッチや電気器具などの大半の家庭用の制御デバイスは、接続に電線を使用することができる。データサービスは、ブロードバンド（ＤＳＬまたはケーブルモデム）として家屋に入り、家屋内では無線（ＨｏｍｅＲＦや８０２．１１Ｂ）または有線（ＨｏｍｅＰＮＡ、Ｃａｔ５、Ｅｔｈｅｒｎｅｔ（登録商標）、あるいは電線）の接続を使用してアクセスされることができる。音声トラフィックは、有線（Ｃａｔ３など）または無線（携帯電話など）として家屋に入り、家屋内ではＣａｔ３配線を使用して分配されることができる。娯楽媒体あるいは他のグラフィックデータは、衛星またはケーブルを通じて家屋に入ることができ、家屋内では通常同軸ケーブルを使用して分配される。ＩＥＥＥ１３９４とＤＶＩもメディアデバイスのクラスタのためのデジタル相互接続である。これらのネットワーク環境およびプロトコル標準として台頭する可能性のある他のネットワーク環境をすべて相互接続してイントラネットなどのネットワークを形成することができ、そのネットワークは、インターネットを通じて外部世界に接続することができる。要約すると、データの記憶と送信には各種の異種のソースが存在し、その結果、いずれ、コンピューティングデバイスは、本発明による仮想化グラフィックサービスを利用するプログラムオブジェクトに関連してアクセスあるいは利用されるデータなどのデータを共有する手段を必要とするようになると考えられる。

【０１０５】
インターネットとは、通例、コンピュータネットワーキングの分野ではよく知られるＴＣＰ／ＩＰプロトコルスイートを利用するネットワークとゲートウェイの集合を指す。ＴＣＰ／ＩＰは、「伝送制御プロトコル／インターネットプロトコル」の頭字語である。インターネットは、ユーザがネットワークを通じて対話し、情報を共有することを可能にするネットワーキングプロトコルを実行するコンピュータによって相互に接続された、地理的に分散した遠隔のコンピュータネットワークからなるシステムと表現することができる。そのように広域に分散した情報共有のため、インターネットなどのリモートネットワークは、これまでのところ、一般に、開発者が、基本的に制約を受けずに、特殊な動作またはサービスを行うソフトウェアアプリケーションを設計することができるオープンシステムに発展している。

【０１０６】
したがって、このネットワークインフラストラクチャは、クライアント／サーバ、ピアツーピア、あるいは混合アーキテクチャなどの多数のネットワークトポロジを可能にしている。「クライアント」は、そのクライアントが関連しない別のクラスまたはグループのサービスを使用するクラスまたはグループのメンバである。したがって、コンピューティングの際のクライアントは、プロセス、すなわち大まかに言うと別のプログラムから提供されるサービスを要求する命令またはタスクのセットである。クライアントプロセスは、他のプログラムあるいはサービス自体についての作業の詳細を一切「知る」必要なく、要求したサービスを利用する。クライアント／サーバアーキテクチャ、特にネットワーク化されたシステムでは、クライアントは、通例、別のコンピュータ、例えばサーバによって提供される共有ネットワークリソースにアクセスするコンピュータである。図６Ａの例では、コンピュータ６１０ａ、６１０ｂ等をクライアントと考えることができ、コンピュータ１０ａ、１０ｂ等をサーバと考えることができ、サーバ１０ａ、１０ｂ等は、データを保持し、そのデータがクライアントコンピュータ６１０ａ、６１０ｂ等で複製されるが、状況に応じてどのコンピュータもクライアント、サーバ、またはその両方と見なされることができる。これらのコンピューティングデバイスはいずれも、本発明の仮想化グラフィックアーキテクチャの実施に関連する可能性があるデータを処理する、またはサービスあるいはタスクを要求する可能性がある。

【０１０７】
サーバは通例、インターネットなど、リモートまたはローカルのネットワークを通じて

アクセスすることが可能なリモートコンピュータシステムである。第１のコンピュータシステムでクライアントプロセスがアクティブであり、第２のコンピュータシステムでサーバプロセスがアクティブであることが可能であり、通信媒体を通じて相互と通信し、それにより、分散した機能を提供し、複数のクライアントがサーバの情報収集能力を利用できるようにすることができる。本発明の仮想化アーキテクチャを利用することに準拠して利用されるソフトウェアオブジェクトは、複数のコンピューティングデバイスあるいはオブジェクトに分散することができる。

【０１０８】

クライアントとサーバは、プロトコル層によって提供される機能を利用して互いに通信する。例えば、ＨＴＴＰ（ＨｙｐｅｒＴｅｘｔ Ｔｒａｎｓｆｅｒ Ｐｒｏｔｏｃｏｌ）は、ＷＷＷ（Ｗｏｒｌｄ Ｗｉｄｅ Ｗｅｂ）あるいは「Ｗｅｂ」の関連で使用される一般的なプロトコルである。通常、インターネットプロトコル（ＩＰ）アドレスなどのコンピュータネットワークアドレスまたはＵＲＬ（Ｕｎｉｖｅｒｓａｌ Ｒｅｓｏｕｒｃｅ Ｌｏｃａｔｏｒ）などの他の参照を使用して、サーバコンピュータまたはクライアントコンピュータを互いに対して識別することができる。ネットワークアドレスは、ＵＲＬアドレスと呼ぶことができる。通信は、通信媒体を通じて提供されることができ、例えば、クライアントとサーバは、大容量通信のためにＴＣＰ／ＩＰ接続を介して相互と結合することができる。

【０１０９】

したがって、図６Ａに、本発明が用いられることが可能な、サーバがネットワーク／バスを介してクライアントコンピュータと通信する、例示的なネットワーク環境あるいは分散環境を示す。より詳細には、いくつかのサーバ１０ａ、１０ｂ等が、本発明により、ＬＡＮ、ＷＡＮ、イントラネット、インターネットなどの通信ネットワーク／バス１４を介して、携帯型コンピュータ、ハンドヘルドコンピュータ、シンクライアント、ネットワーク対応の電気器具、またはＶＣＲ、ＴＶ、オーブン、電灯、ヒータなどの他のデバイスなどのいくつかのクライアントあるいはリモートコンピューティングデバイス６１０ａ、６１０ｂ、６１０ｃ、６１０ｄ、６１０ｅ等と相互に接続される。したがって、本発明は、それとの関連で本発明によるゲストグラフィックインタフェースおよびオペレーティングシステムを実施することが望ましいコンピューティングデバイスに適用できることが意図されている。

【０１１０】

例えば通信ネットワーク／バス１４がインターネットであるネットワーク環境では、サーバ１０ａ、１０ｂ等は、クライアント６１０ａ、６１０ｂ、６１０ｃ、６１０ｄ、６１０ｅ等がＨＴＴＰなどのいくつかの知られるプロトコルのいずれかを介して通信するウェブサーバとすることができる。分散コンピューティング環境の特徴であるように、サーバ１０ａ、１０ｂ等は、クライアント６１０ａ、６１０ｂ、６１０ｃ、６１０ｄ、６１０ｅ等として機能する場合もある。通信は、適宜有線でも無線でもよい。クライアントデバイス６１０ａ、６１０ｂ、６１０ｃ、６１０ｄ、６１０ｅ等は、通信ネットワーク／バス１４を介して通信してもしなくともよく、そのデバイスに関連付けられた独立した通信を有することが可能である。例えば、ＴＶやＶＣＲの場合は、その制御にネットワーク化された面がある場合もない場合もある。各クライアントコンピュータ６１０ａ、６１０ｂ、６１０ｃ、６１０ｄ、６１０ｅ等とサーバコンピュータ１０ａ、１０ｂ等は、各種のアプリケーションプログラムモジュールあるいはオブジェクト６３５と、各種タイプの記憶要素またはオブジェクトへの接続またはアクセスを備えることができ、その記憶要素またはオブジェクトに、ファイルまたはデータストリームを記憶することができ、または、その記憶要素またはオブジェクトにデータまたはデータストリームの一部をダウンロード、送信、または移動することができる。コンピュータ１０ａ、１０ｂ、６１０ａ、６１０ｂ等の１つまたは複数が、本発明により処理されるデータを記憶するデータベースあるいはメモリ２０などのデータベース２０または他の記憶要素の維持と更新を担うことができる。したがって、本発明は、コンピュータネットワーク／バス１４にアクセスし、対話すること

ができるクライアントコンピュータ６１０ａ、６１０ｂ等と、クライアントコンピュータ６１０ａ、６１０ｂ等および他の同様のデバイスおよびデータベース２０と対話することが可能なサーバコンピュータ１０ａ、１０ｂ等とを有するコンピュータネットワーク環境で利用することができる。

【０１１１】

例示的コンピューティングデバイス

図６Ｂと以下の説明は、本発明がそれとの関連で実施されることが可能な適切なコンピューティング環境の簡単で一般的な説明を提供するものである。ただし、ハンドヘルド型、携帯型、およびすべての種の他のコンピューティングデバイスおよびコンピューティングオブジェクトが本発明との関連で、すなわちＧＰＵがコンピューティング環境に存在する場所はどこでも使用されることが意図されていることを理解されたい。下記では汎用コンピュータについて述べるが、これは一例に過ぎず、本発明は、ネットワーク／バスの相互運用性と対話を備えるシンクライアントで実施されてよい。したがって、本発明は、非常に少ない、あるいは最小限のクライアントリソースが関連する、ネットワーク化されたホストされるサービスからなる環境、例えば、クライアントデバイスが、電気器具の中に配置されたオブジェクトなど、単にネットワーク／バスとのインタフェースの役割を果たすネットワーク環境で実施されることができる。基本的に、データを記憶することができる、あるいはそこからデータを取り出す、または別のコンピュータに送信することができる環境であれば、本発明によるグラフィック仮想化技術の運用に望ましい、あるいは適した環境となる。

【０１１２】

これは必須ではないが、本発明は、デバイスまたはオブジェクトのためのサービスの開発者によって使用されるために、オペレーティングシステムを介して全体または一部が実施されることができ、かつ／または、本発明の仮想化グラフィックパイプラインとの関連で動作するアプリケーションソフトウェアに含めることができる。ソフトウェアは、クライアントワークステーション、サーバ、または他のデバイスなどの１つまたは複数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コンテクストで説明することができる。一般に、プログラムモジュールには、特定のタスクを行うか、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。通常、プログラムモジュールの機能は、各種実施形態で必要に応じて組み合わせても、分散してもよい。さらに、当業者は、本発明が、他のコンピュータシステム構成およびプロトコルとともに実施されてよいことを理解されよう。本発明に使用するのに適する可能性がある他のよく知られるコンピューティングシステム、環境、および／または構成には、これらに限定しないが、パーソナルコンピュータ（ＰＣ）、現金自動預け払い機、サーバコンピュータ、ハンドヘルドまたはラップトップデバイス、マルチプロセッサシステム、マイクロプロセッサを利用したシステム、プログラム可能な家庭用電化製品、ネットワークＰＣ、電気器具、電灯、環境制御要素、ミニコンピュータ、メインフレームコンピュータなどがある。本発明は、通信ネットワーク／バスまたは他のデータ伝送媒体を通じてつながれた遠隔の処理デバイスによってタスクが行われる分散コンピューティング環境で実施してもよい。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は、メモリ記憶装置を含むローカルおよびリモート両方のコンピュータ記憶媒体に配置することができ、クライアントノードがサーバノードとして振舞うことができる。

【０１１３】

したがって、図６Ｂには、本発明が実施されることが可能な適切なコンピューティングシステム環境６００の一例を示すが、上記で明確にしたように、コンピューティングシステム環境６００は、適切なコンピューティング環境の一例に過ぎず、本発明の使用または機能の範囲についての制限を示唆するものではない。また、コンピューティング環境６００は、例示的動作環境６００に示される構成要素の１つまたは組み合わせに関連する依存性または要件を有するものとも解釈すべきでない。

【０１１４】
図６Ｂを参照すると、本発明を実施する例示的システムは、コンピュータ６１０の形態の汎用コンピューティングデバイスを含む。コンピュータ６１０の構成要素は、これらに限定しないが、処理装置６２０、システムメモリ６３０、システムメモリを含む各種のシステム構成要素を処理装置６２０に結合するシステムバス６２１を含むことができる。システムバス６２１は、各種のバスアーキテクチャを使用した、メモリバスあるいはメモリコントローラ、ペリフェラルバス、ローカルバスを含む数種のバス構造のいずれでもよい。限定ではなく例として、そのようなアーキテクチャには、ＩＳＡ（Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＭＣＡ（Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、ＥＩＳＡ（Ｅｎｈａｎｃｅｄ ＩＳＡ）バス、ＶＥＳＡ（Ｖｉｄｅｏ Ｅｌｅｃｔｒｏｎｉｃｓ Ｓｔａｎｄａｒｄｓ Ａｓｓｏｃｉａｔｉｏｎ）ローカルバス、ＰＣＩ（Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ）バス（メザニンバスとも称される）、およびＰＣＩｅ（ＰＣＩ Ｅｘｐｒｅｓｓ）がある。

【０１１５】
コンピュータ６１０は、通常、各種のコンピュータ可読媒体を含む。コンピュータ可読媒体は、コンピュータ６１０によるアクセスが可能な利用可能媒体でよく、揮発性および不揮発性の媒体、取り外し可能および取り外し不能の媒体を含む。限定ではなく例として、コンピュータ可読媒体は、コンピュータ記憶媒体と通信媒体を含むことができる。コンピュータ記憶媒体には、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、または他のデータなどの情報を記憶するための方法または技術で実施された、揮発性および不揮発性、取り外し可能および取り外し不能の媒体が含まれる。コンピュータ記憶媒体には、これらに限らないが、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュメモリ、または他のメモリ技術、ＣＤ－ＲＯＭ、デジタル多用途ディスク（ＤＶＤ）、または他の光ディスク記憶、磁気カセット、磁気テープ、磁気ディスク記憶、または他の磁気記憶装置、あるいは、所望の情報を記憶するために使用することができ、コンピュータ６１０によるアクセスが可能な他の媒体が含まれる。通信媒体は通常、コンピュータ可読命令、データ構造体、プログラムモジュール、または他のデータを、搬送波や他の移送機構などの変調されたデータ信号中に具現化するものであり、情報伝達媒体を含む。用語「変調されたデータ信号」とは、信号中に情報を符号化するような方式で特性の１つまたは複数を設定または変化させた信号を意味する。限定ではなく例として、通信媒体には、有線ネットワークや直接配線接続などの有線媒体と、音響、ＲＦ、赤外線、他の無線媒体などの無線媒体がある。上記の媒体の組み合わせもコンピュータ可読媒体の範囲に含める。

【０１１６】
システムメモリ６３０は、読出し専用メモリ（ＲＯＭ）６３１およびランダムアクセスメモリ（ＲＡＭ）６３２などの揮発性および／または不揮発性メモリの形態のコンピュータ記憶媒体を含む。起動時などにコンピュータ６１０内の要素間の情報転送を助ける基本ルーチンを含んだ基本入出力システム６３３（ＢＩＯＳ）は、通常ＲＯＭ６３１に記憶される。ＲＡＭ６３２は通常、処理装置６２０から即座にアクセスすることができる、かつ／または処理装置６２０によって現在操作されているデータおよび／またはプログラムモジュールを保持する。限定ではなく例として、図６Ｂには、オペレーティングシステム６３４、アプリケーションプログラム６３５、他のプログラムモジュール６３６、およびプログラムデータ６３７を示す。

【０１１７】
コンピュータ６１０は、他の取り外し可能／取り外し不能、揮発性／不揮発性のコンピュータ記憶媒体も含むことができる。単なる例として、図６Ｂには、取り外し不能、不揮発性の磁気媒体の読み書きを行うハードディスクドライブ６４１、取り外し可能、不揮発性の磁気ディスク６５２の読み書きを行う磁気ディスクドライブ６５１、およびＣＤ－ＲＯＭや他の光学媒体などの取り外し可能、不揮発性の光ディスク６５６の読み書きを行う光ディスクドライブ６５５を示す。例示的動作環境で使用することができる他の取り外し

可能／取り外し不能、揮発性／不揮発性の媒体には、これらに限定しないが、磁気テープカセット、フラッシュメモリカード、デジタル多用途ディスク、デジタルビデオテープ、固体素子ＲＡＭ、固体素子ＲＯＭなどがある。ハードディスクドライブ６４１は通常、インタフェース６４０などの取り外し不能メモリインタフェースを通じてシステムバス６２１に接続され、磁気ディスクドライブ６５１と光ディスクドライブ６５５は通例、インタフェース６５０などの取り外し可能メモリインタフェースでシステムバス６２１に接続される。

【０１１８】

上記で説明し、図６Ｂに示すこれらのドライブとそれに関連付けられたコンピュータ記憶媒体は、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、および他のデータの記憶をコンピュータ６１０に提供する。図６Ｂでは、例えば、ハードディスクドライブ６４１に、オペレーティングシステム６４４、アプリケーションプログラム６４５、他のプログラムモジュール６４６、およびプログラムデータ６４７が記憶されている。これらのコンポーネントは、オペレーティングシステム６３４、アプリケーションプログラム６３５、他のプログラムモジュール６３６、およびプログラムデータ６３７と同じであっても異なってもよいことに留意されたい。ここではそれらが少なくとも異なるコピーであることを表すために、オペレーティングシステム６４４、アプリケーションプログラム６４５、他のプログラムモジュール６４６、およびプログラムデータ６４７には異なる参照符号を付している。ユーザは、キーボード６２２と、一般にはマウス、トラックボール、タッチパッドと称されるポインティングデバイス６６１などの入力装置を通じてコンピュータ６１０にコマンドと情報を入力することができる。他の入力装置（図示せず）としては、マイクロフォン、ジョイスティック、ゲームパッド、衛星受信アンテナ、スキャナ等が可能である。上記および他の入力装置は、多くの場合、システムバス６２１に結合されたユーザ入力インタフェース６６０を通じて処理装置６２０に接続されるが、パラレルポート、ゲームポート、ユニバーサルシリアルバス（ＵＳＢ）などの他のインタフェースおよびバス構造で接続してもよい。これらは、本発明のアーキテクチャによって仮想化される構造の種類である。ノースブリッジで実装されるインタフェースの１つなどのグラフィックインタフェース６８２もシステムバス６２１に接続されることができる。ノースブリッジは、ＣＰＵあるいはホスト処理装置６２０と通信し、ＰＣＩ、ＰＣＩｅ、およびアクセラレーテッドグラフィックポート（ＡＧＰ）通信などの通信を担うチップセットである。本発明は、統合された（ノースブリッジ内部の）グラフィック実装と、個別（ノースブリッジ外部）のグラフィック実装の両方を意図する。１つまたは複数のグラフィック処理装置（ＧＰＵ）６８４が、グラフィックインタフェース６８２と通信することができる。これに関して、ＧＰＵ６８４は一般に、レジスタ記憶などのオンチップメモリ記憶を含み、ＧＰＵ６８４は、ビデオメモリ６８６と通信する。しかし、ＧＰＵ６８４は、コプロセッサの一例に過ぎず、そのため、コンピュータ６１０には各種のコプロセッシング装置を含めることができ、そのコプロセッシング装置は、ピクセルや頂点シェーダなどの各種の手続き型シェーダを含むことができる。モニタ６９１または他のタイプの表示装置もビデオインタフェース６９０などのインタフェースを介してシステムバス６２６に接続され、ビデオインタフェース６９０は、ビデオメモリ６８６と通信する。モニタ６９１に加えて、コンピュータは、スピーカ６９７やプリンタ６９６などの他の周辺出力装置も含むことができ、それらの出力装置は、出力周辺装置インタフェース６９５を通じて接続されることができる。

【０１１９】

コンピュータ６１０は、リモートコンピュータ６８０などの１つまたは複数のコンピュータとの論理接続を使用するネットワーク環境あるいは分散環境で動作することができる。リモートコンピュータ６８０は、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、あるいは他の一般的なノードであり、図６Ｂにはメモリ記憶装置６８１のみを示すが、通例はコンピュータ６１０との関連で上述した要素の多くまたはすべてを含む。図６Ｂに示す論理接続は、ローカルエリアネットワーク（ＬＡＮ）６７１

とワイドエリアネットワーク（ＷＡＮ）６７３を含むが、他のネットワーク／バスを含んでもよい。このようなネットワーク環境は、家庭、オフィス、企業内のコンピュータネットワーク、イントラネット、インターネットで一般的である。

【０１２０】

ＬＡＮネットワーク環境で使用される場合、コンピュータ６１０は、ネットワークインタフェースあるいはアダプタ６７０を通じてＬＡＮ６７１に接続される。ＷＡＮネットワーク環境で使用される場合、コンピュータ６１０は通常、インターネットなどのＷＡＮ６７３を通じて通信を確立するためのモデム６７２あるいは他の手段を含む。モデム６７２は、内蔵される場合も、外付けされる場合もあり、ユーザ入力インタフェース６６０あるいは他の適切な機構を介してシステムバス６２１に接続することができる。ネットワーク環境では、コンピュータ６１０との関連で図示されるプログラムモジュールまたはその一部は、遠隔のメモリ記憶装置に記憶することができる。限定ではなく例として、図６Ｂでは、リモートアプリケーションプログラム６８５がメモリ装置６８１にある。図のネットワーク接続は例示的なものであり、コンピュータ間に通信リンクを確立する他の手段を使用してよいことは理解されよう。

【０１２１】

アプリケーションおよびサービスが本発明の仮想化アーキテクチャ、システム、および方法を使用することを可能にする、例えば適切なＡＰＩ、ツールキット、ドライバコード、オペレーティングシステム、コントロール、スタンドアロンまたはダウンロード可能なソフトウェアオブジェクト等、本発明を実施する手段は複数ある。本発明は、ＡＰＩ（または他のソフトウェアオブジェクト）の観点から、また、本発明による上述の技術を受け入れるソフトウェアまたはハードウェアオブジェクトの観点から、本発明の使用を意図する。したがって、ここに記載される本発明の各種実施は、完全にハードウェアである態様、一部がハードウェアで一部がソフトウェアである態様、ならびにソフトウェアの態様を有することができる。

【０１２２】

上記で述べたように、本発明の例示的実施形態について各種のコンピューティングデバイスおよびネットワークアーキテクチャとの関連で説明したが、基礎となる概念は、ＧＰＵを用いることが望ましいコンピューティングデバイスまたはシステムに適用することができる。例えば、デバイス上の独立したオブジェクト、別のオブジェクトの一部、再使用可能なコントロール、サーバからダウンロードすることが可能なオブジェクト、デバイスまたはオブジェクトとネットワークとの「仲介役」、分散されたオブジェクト、ハードウェアとして、メモリ中に、前述の組み合わせ等として提供される、本発明の各種アルゴリズムとハードウェア実施は、コンピューティングデバイスのオペレーティングシステムに適用することができる。当業者は、本発明の各種実施形態によって達成される機能と同じ機能、ほぼ同等、あるいは同等の機能を達成するオブジェクトコードと術語（ｎｏｍｅｎｃｌａｔｕｒｅ）を提供する手段は多数あることを理解されよう。

【０１２３】

先に述べたように、ここに記載される各種技術は、ハードウェアまたはソフトウェアとの関連で、または適切な場合には両方の組み合わせで実施されることができる。したがって、本発明の方法および装置、またはその特定の態様あるいはその一部は、フロッピー（登録商標）ディスケット、ＣＤ－ＲＯＭ、ハードドライブ、または他の機械可読の記憶媒体などの有形媒体として実施されたプログラムコード（すなわち命令）の形を取ることができ、そのプログラムコードがコンピュータなどのマシンにロードされ、実行されると、マシンは、本発明を実施する装置となる。プログラム可能なコンピュータでのプログラムコード実行の場合、コンピューティングデバイスは、一般に、プロセッサ、プロセッサによって読み取ることが可能な記憶媒体（揮発性および不揮発性メモリおよび／または記憶素子を含む）、少なくとも１つの入力装置、および少なくとも１つの出力装置を含む。例えばデータ処理ＡＰＩ、再使用可能なコントロールなどの使用を通じて本発明のＧＰＵ仮想化技術を実施または利用することが可能な１つまたは複数のプログラムは、高水準の手

続きプログラミング言語またはオブジェクト指向のプログラミング言語で実施されて、コンピュータシステムと通信することが好ましい。ただし、プログラムは、必要な場合はアセンブリ言語あるいは機械言語で実施することができる。いずれの場合も、言語は、コンパイルあるいは解釈される言語であり、ハードウェア実装と組み合わせられる。

【０１２４】

本発明の方法および装置は、電気配線あるいはケーブル配線、光ファイバ、あるいは他の形態の伝送など、何らかの伝送媒体を通じて送信されるプログラムコードの形態で実施された通信を介して実施してもよく、そのプログラムコードが受信され、ＥＰＲＯＭ、ゲートアレイ、ＰＬＤ（プログラマブルロジックデバイス）、クライアントコンピュータなどのマシンにロードされ、実行されると、そのマシンは、本発明を実施する装置となる。汎用プロセッサで実施される場合、プログラムコードは、プロセッサと連携して、本発明の機能を呼び出すように動作する独自の装置を提供する。また、本発明との関連で使用される記憶技術は、いずれもハードウェアとソフトウェアの組み合わせとすることができる。

【０１２５】

本発明について各種図の好ましい実施形態との関連で説明したが、他の同様の実施形態を使用してよく、または本発明から逸脱することなく、本発明と同じ機能を行うために、ここに記載される実施形態に変更と追加を加えてよいことは理解されたい。例えば、本発明の例示的なネットワーク環境は、ピアツーピアのネットワーク環境などのネットワーク環境のコンテクストで説明したが、当業者は、本発明はこれに制限されず、本願に記載される方法は、有線であっても無線であっても、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コンピューティングデバイスまたは環境に適用することができ、通信ネットワークを介して接続され、そのネットワークを通じて対話する任意数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を理解されよう。さらに、特に無線ネットワークデバイスの数が増え続けているのに伴い、ハンドヘルドデバイスのオペレーティングシステムおよび他のアプリケーション固有のオペレーティングシステムを含む各種のコンピュータプラットフォームが意図されることを強調したい。

【０１２６】

例示的実施形態では、グラフィックパイプラインのコンテクストで本発明を利用すると言及するが、本発明はそれに限定されず、他の理由でメインプロセッサと協働する第２の特殊化処理装置を仮想化するためにも実施されることができる。さらに、本発明は、同じバージョンまたはリリースのＯＳの複数のインスタンスが、本発明により別々の仮想マシンで動作するシナリオを意図する。本発明の仮想化は、ＧＰＵが使用される動作に依存しないことは理解されることができる。また、本発明は、これらに限定しないが複数のＧＰＵ実装、ならびに単一のＧＰＵインタフェースの錯覚（ｉｌｌｕｓｉｏｎ）を提供する複数ＧＰＵの実施形態を含む、すべてのＧＰＵアーキテクチャに適用されることも意図される。さらに、本発明は、複数の処理チップまたはデバイスで、あるいはそれらにまたがって実施することができ、記憶も同様に複数のデバイスにわたって実施されることができる。したがって、本発明は、どの１つ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よる広さと範囲で解釈されたい。

【図面の簡単な説明】

【０１２７】

【図１】従来技術によるグラフィックパイプラインの実装を表すブロック図である。

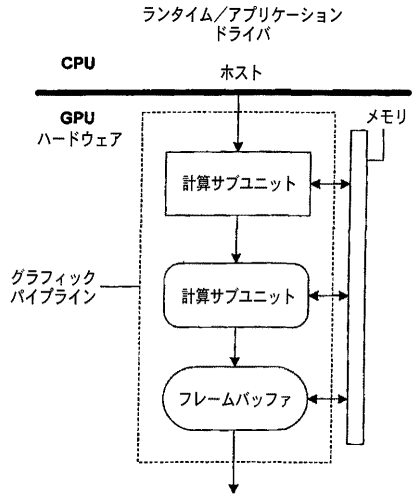
【図２Ａ】コンピュータシステムにおけるエミュレートされた動作環境のためのハードウェアとソフトウェアアーキテクチャの論理階層化を表すブロック図である。

【図２Ｂ】ホストオペレーティングシステムによって（直接またはハイパーバイザを介して）エミュレーションが行われる仮想化コンピューティングシステムを表すブロック図である。

【図２Ｃ】ホストオペレーティングシステムと並行して実行される仮想マシンモニタによ

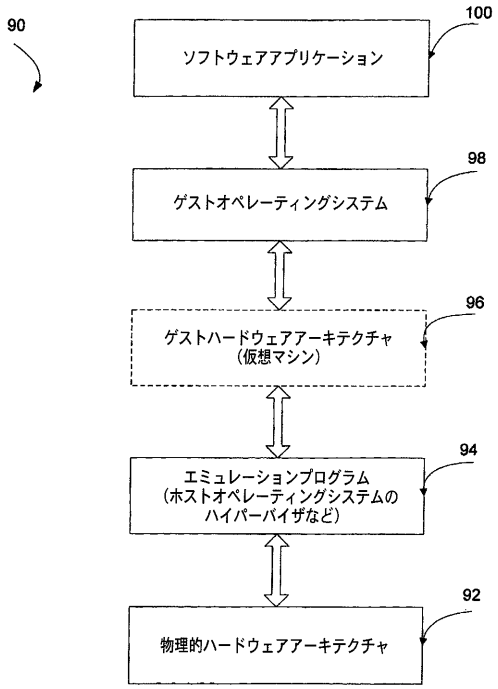
ってエミュレーションが行われる代替の仮想化コンピューティングシステムを表すブロック図である。
【図３Ａ】グラフィックハードウェア空間に仮想マシンを適用し、それにより監視コードがＣＰＵで実行されるが、グラフィック作業項目が直接グラフィックハードウェアによって処理される、第１の非制限的アーキテクチャの図である。
【図３Ｂ】本発明によりグラフィックパイプラインを仮想化するために提供される別の例示的な非制限アーキテクチャの図である。
【図４Ａ】本発明により「一度に単一のデスクトップ」を表示する実施形態の図である。
【図４Ｂ】本発明によりすべてのデスクトップが同時に表示されることができる、代替の「ウィンドウ内のデスクトップ」を表す図である。
【図４Ｃ】本発明により信頼されるアプリケーションデータと信頼されないアプリケーションデータを混合するための例示的アーキテクチャの図である。
【図４Ｄ】コンテンツ保護のための別個の層としてメディア相互動作ゲートウェイを含むように信頼されるパーティションを拡張することを示す図である。
【図４Ｅ】ホストＯＳで実行されるゲストＯＳレガシーアプリケーションが、ゲストＯＳのレガシーサービスコンポーネントと対話してレガシーの挙動をエミュレートする例示的対話を表す図である。
【図５】本発明により提供される仮想化グラフィックアーキテクチャの１つまたは複数の実施形態による、グラフィックデータの処理の例示的な非制限的シーケンスを表す流れ図である。
【図６Ａ】本発明が実施されることが可能な、各種のコンピューティングデバイスを有する例示的ネットワーク環境を表すブロック図である。
【図６Ｂ】本発明が実施されることが可能な例示的な非制限的コンピューティングデバイスを表すブロック図である。
【符号の説明】
【０１２８】
３１０　ホストＯＳ
３３０　ゲストＯＳ　Ａ
３４０　ゲストＯＳ　Ｂ
３０８　ホスト仮想マシン
３２８　仮想マシン　Ａ
３３８　仮想マシン　Ｂ
３０４　仮想マシンモニタ
３０２　グラフィックハードウェア

【図１】

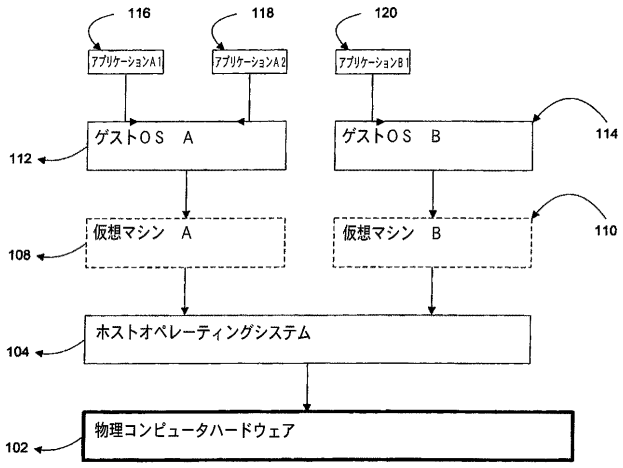


従来技術

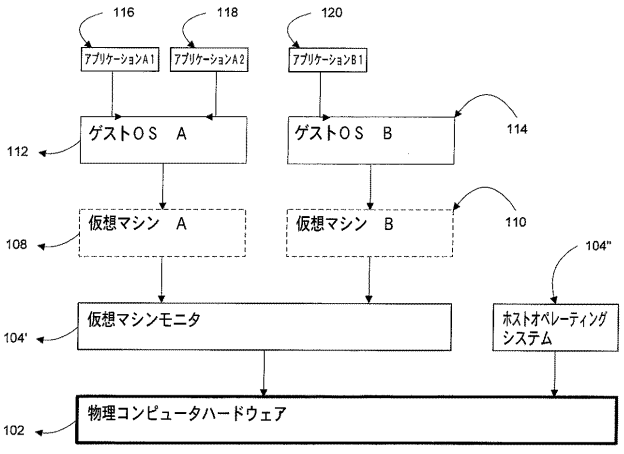
【図２Ａ】



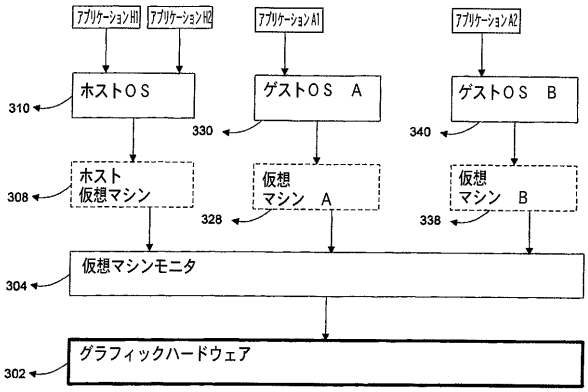
【図２Ｂ】



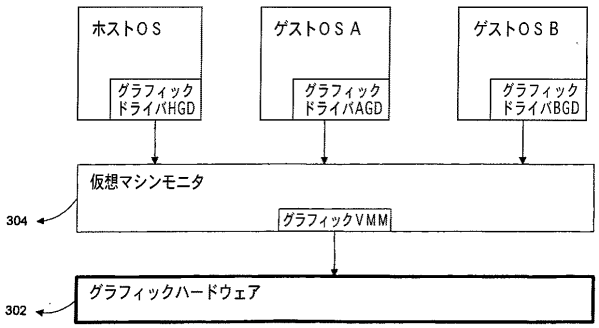
【図２Ｃ】



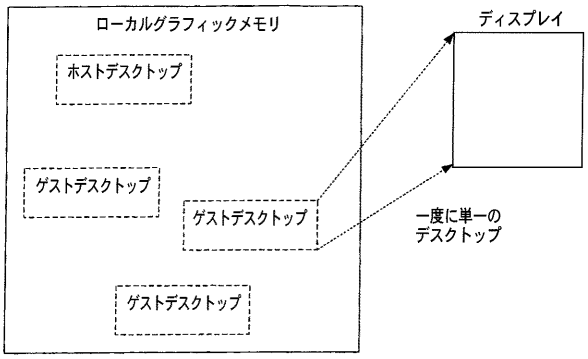
【図３Ａ】



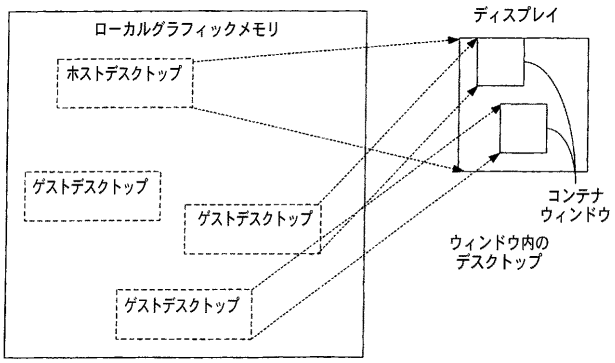
【図３Ｂ】



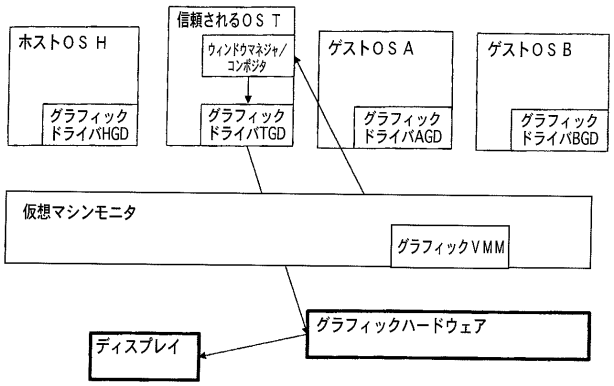
【図４Ａ】



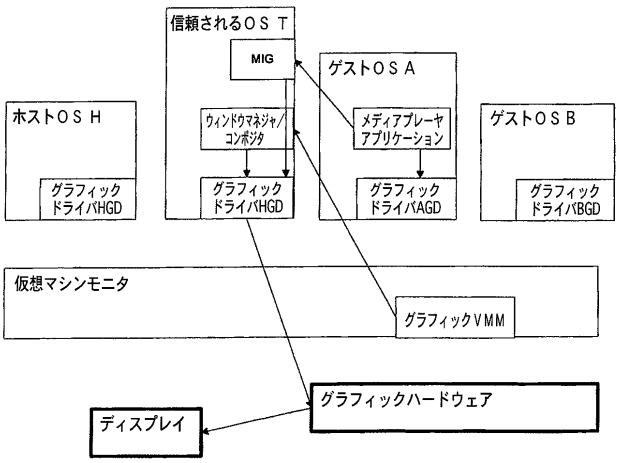
【図４Ｂ】



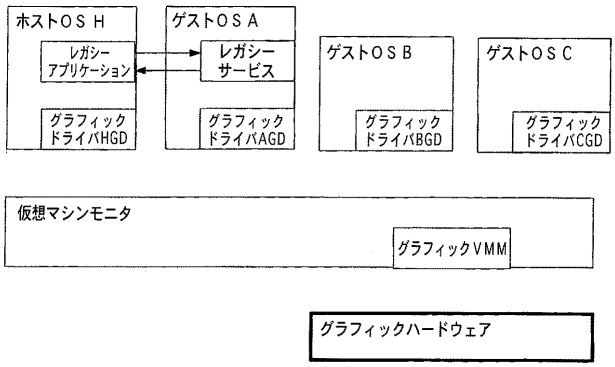
【図４Ｃ】



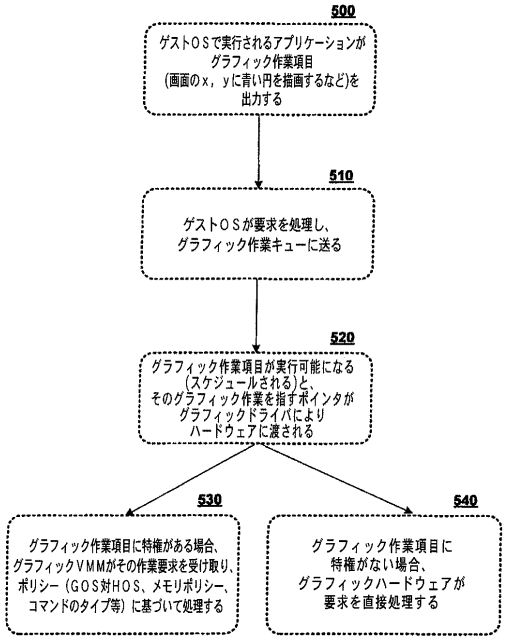
【図４Ｄ】



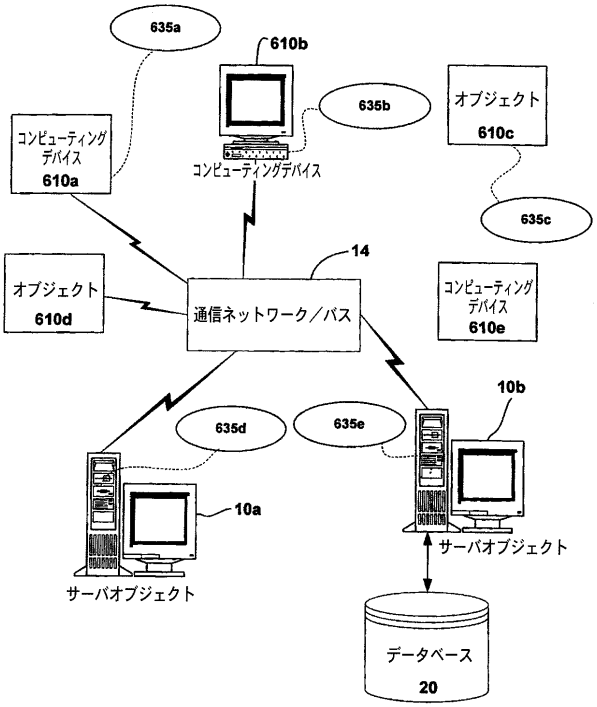
【図４Ｅ】



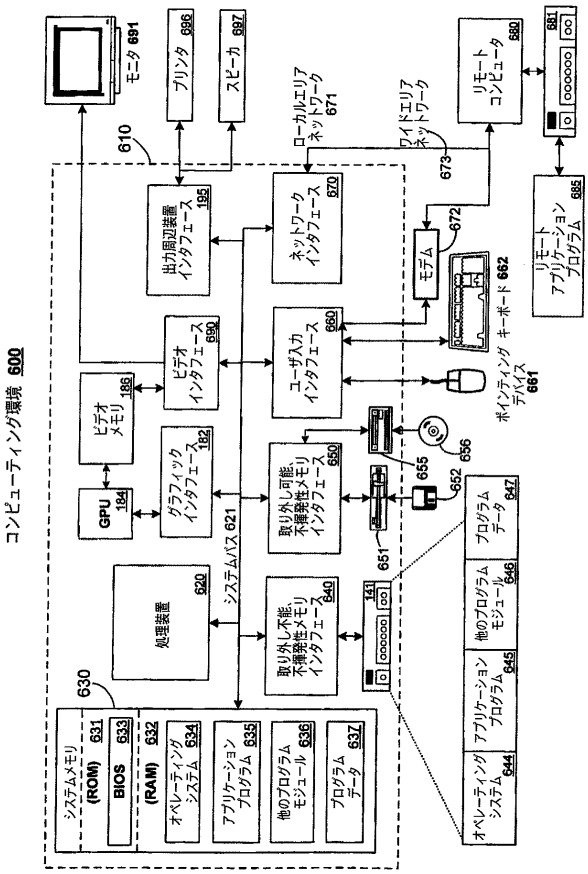
【図５】


500
ゲストOSで実行されるアプリケーションが
グラフィック作業項目
(画面のx，yに青い円を描画するなど)を
出力する
510
ゲストOSが要求を処理し、
グラフィック作業キューに送る
520
グラフィック作業項目が実行可能になる
(スケジュールされる)と、
そのグラフィック作業を指すポインタが
グラフィックドライバにより
ハードウェアに渡される
530
グラフィック作業項目に特権がある場合、
グラフィックVMMがその作業要求を受け取り、
ポリシー（GOS対HOS、メモリポリシー、
コマンドのタイプ等）に基づいて処理する
540
グラフィック作業項目に
特権がない場合、
グラフィックハードウェアが
要求を直接処理する

【図６Ａ】


635a
610b
635b
オブジェクト
610c
コンピューティング
デバイス
610a
コンピューティングデバイス
635c
14
オブジェクト
610d
通信ネットワーク／パス
コンピューティング
デバイス
610e
10b
635d
635e
10a
サーバオブジェクト
サーバオブジェクト
データベース
20

【図６Ｂ】


コンピューティング環境 600
610
モニタ 691
プリンタ 696
スピーカ 697
ローカルエリア
ネットワーク
671
ワイドエリア
ネットワーク
673
リモート
コンピュータ
680
681
出力周辺装置
インタフェース
695
ネットワーク
インタフェース
670
モデム
672
リモート
アプリケーション
プログラム
685
ビデオ
インタフェース
690
ユーザ入力
インタフェース
660
キーボード 662
ポインティング
デバイス
661
ビデオ
メモリ
186
GPU
184
グラフィック
インタフェース
182
取り外し可能、
不揮発性メモリ
インタフェース
650
656
655
652
651
システムバス 621
処理装置
620
取り外し不能、
不揮発性メモリ
インタフェース
640
141
プログラム
データ
647
他のプログラム
モジュール
646
アプリケーション
プログラム
645
オペレーティング
システム
644
630
システムメモリ
(ROM) 631
BIOS 633
(RAM) 632
オペレーティング
システム 634
アプリケーション
プログラム 635
他のプログラム
モジュール 636
プログラム
データ 637