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

特許第6437579号
(P6437579)

(45)発行日　平成30年12月12日(2018.12.12)　(24)登録日　平成30年11月22日(2018.11.22)

(51)Int.Cl.			F I		
G06F	***9/50***	***(2006.01)***	G06F	9/50	120A
G06F	***9/455***	***(2006.01)***	G06F	9/455	150
G06F	***9/38***	***(2006.01)***	G06F	9/38	370C
G06T	***1/20***	***(2006.01)***	G06T	1/20	A

請求項の数 24　(全 23 頁)

(21)出願番号　特願2016-574043 (P2016-574043)
(86)(22)出願日　平成26年6月26日 (2014.6.26)
(65)公表番号　特表2017-522659 (P2017-522659A)
(43)公表日　平成29年8月10日 (2017.8.10)
(86)国際出願番号　PCT/CN2014/080814
(87)国際公開番号　WO2015/196409
(87)国際公開日　平成27年12月30日 (2015.12.30)
審査請求日　平成28年12月19日 (2016.12.19)

(73)特許権者　593096712
インテル　コーポレイション
アメリカ合衆国　９５０５４　カリフォルニア州　サンタ　クララ　ミッション　カレッジ　ブールバード　２２００
(74)代理人　100107766
弁理士　伊東　忠重
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100091214
弁理士　大貫　進介
(72)発明者　ティエン，クン
中華人民共和国　２００２４１　シャンハイ　ズージュウ　サイエンス　パーク　ズー　シン　ロード　ナンバー８８０
最終頁に続く

(54)【発明の名称】仮想化環境におけるインテリジェントＧＰＵスケジューリング

(57)【特許請求の範囲】
【請求項１】
仮想化されたグラフィックス処理ユニット（ＧＰＵ）に対するワークロードサブミッションをスケジューリングする計算デバイスであって、
複数の仮想マシンを有する仮想化環境を確立するための仮想化サービスであって、前記仮想マシンの各々は前記ＧＰＵと通信するためのグラフィックスドライバーと、ＧＰＵコマンドを記憶する複数のコマンドバッファとを有する、仮想化サービスと、
ＧＰＵスケジューラモジュールとを有し、
前記ＧＰＵスケジューラモジュールは、
すべての前記仮想マシンのすべての前記コマンドバッファのＧＰＵコマンドを評価し、
前記ＧＰＵコマンドの評価の出力に応じて、複数の異なるスケジューリングポリシーからスケジューリングポリシーを動的に選択し、
動的に選択されたスケジューリングポリシーにより、前記ＧＰＵにより処理する少なくとも１つのＧＰＵコマンドをスケジュールし、
同じ仮想マシンの異なるコマンドバッファ中の２つのＧＰＵコマンド間のクロスバッファ依存性の検出に応じて、前記スケジューリングポリシーを、リングごとのスケジューリングポリシーから集団スケジューリングポリシーに切り替えるためのものである、
計算デバイス。
【請求項２】
仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンし、同じ仮想マシ

ンの異なるコマンドバッファ中のＧＰＵコマンド間のクロスバッファ依存性を示すデータを発生するためのコマンドスキャナーモジュールを有し、スケジューリングポリシーはクロスバッファ依存性を示すデータに基づいて動的に選択される、
請求項１に記載の計算デバイス。
【請求項３】
仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンし、各ＧＰＵコマンドのコマンドタイプを決定するためのコマンドスキャナーモジュールを有し、前記ＧＰＵコマンドのスケジューリングポリシーは前記ＧＰＵコマンドのコマンドタイプに基づいて動的に選択される、請求項１または２に記載の計算デバイス。
【請求項４】
前記ＧＰＵスケジューラモジュールは、動的に選択されたスケジューリングポリシーにより前記仮想マシンのうちの１つのＧＰＵコマンドをスケジュールし、異なるスケジューリングポリシーにより前記仮想マシンのうちの他の１つのＧＰＵコマンドをスケジュールするためのものである、請求項１または２に記載の計算デバイス。
【請求項５】
前記ＧＰＵスケジューラモジュールは、異なる仮想マシンのＧＰＵコマンドを評価し、前記異なる仮想マシンのＧＰＵコマンドを集団スケジューリングポリシーでないスケジューリングポリシーによりスケジュールするためのものである、
請求項１に記載の計算デバイス。
【請求項６】
各コマンドバッファはリングバッファとして実施され、前記ＧＰＵスケジューラモジュールは、クロスリング同期プリミティブの仮想マシンのＧＰＵコマンドをスキャンし、前記仮想マシンのＧＰＵコマンドのためのスケジューリングポリシーを、クロスリング同期プリミティブの存在または不存在に基づき選択するためのものである、
請求項１、２、または５いずれか一項に記載の計算デバイス。
【請求項７】
前記ＧＰＵスケジューラモジュールは、前記ＧＰＵコマンドの評価の出力に基づき、異なる仮想マシンの異なるスケジューリングポリシーを動的に選択するためのものである、
請求項１、２、または５いずれか一項に記載の計算デバイス。
【請求項８】
各仮想マシンのコマンドスキャナーモジュールを有し、各コマンドスキャナーモジュールは前記仮想マシンのコマンドバッファ間の依存性を示すデータを含むコマンドデータベースを生成するためのものである、
請求項１、２、または５いずれか一項に記載の計算デバイス。
【請求項９】
すべての仮想マシンのコマンドデータベースを評価し、すべてのコマンドデータベースの評価に基づいて、少なくとも１つの仮想マシンのスケジューリングポリシーを選択するためのアービターモジュールを有する、
請求項８に記載の計算デバイス。
【請求項１０】
前記ＧＰＵスケジューラモジュールは、仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンし、同じ仮想マシンの異なるコマンドバッファ中のＧＰＵコマンド間のクロスバッファ依存性を検出し、ある時間にわたるクロスバッファ依存性の発生の頻度を決定し、クロスバッファ依存性の発生の頻度に基づいて前記スケジューリングポリシーを変更するためのものである、
請求項１または２に記載の計算デバイス。
【請求項１１】
前記ＧＰＵスケジューラモジュールは、所定数のコマンドバッファにクロスリング同期プリミティブが検出されないことが、すべての仮想マシンについて成り立てば、スケジューリングポリシーを、リングごとのスケジューリングポリシーとするためのものである、

請求項１、２、または５いずれか一項に記載の計算デバイス。
【請求項１２】
仮想化されたグラフィックス処理ユニット（ＧＰＵ）に対するワークロードサブミッションをスケジューリングする方法であって、
複数の仮想マシンを有する仮想化環境を確立するステップであって、前記仮想マシンの各々は前記ＧＰＵと通信するためのグラフィックスドライバーと、ＧＰＵコマンドを記憶する複数のコマンドバッファとを有する、ステップと、
すべての前記仮想マシンのすべての前記コマンドバッファのＧＰＵコマンドを評価するステップと、
前記ＧＰＵコマンドの評価の出力に応じて、複数の異なるスケジューリングポリシーからスケジューリングポリシーを動的に選択するステップと、
動的に選択されたスケジューリングポリシーにより、前記ＧＰＵにより処理する少なくとも１つのＧＰＵコマンドをスケジュールするステップと、
同じ仮想マシンの異なるコマンドバッファ中の２つのＧＰＵコマンド間のクロスバッファ依存性の検出に応じて、前記スケジューリングポリシーを、リングごとのスケジューリングポリシーから集団スケジューリングポリシーに切り替えるステップと、
を含む方法。
【請求項１３】
仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンするステップと、同じ仮想マシンの異なるコマンドバッファ中のＧＰＵコマンド間のクロスバッファ依存性を示すデータを発生するステップと、クロスバッファ依存性を示すデータに基づきスケジューリングポリシーを動的に選択するステップとを含む、
請求項１２に記載の方法。
【請求項１４】
仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンするステップと、各ＧＰＵコマンドのコマンドタイプを決定するステップと、ＧＰＵコマンドのコマンドタイプに基づいて前記ＧＰＵコマンドのスケジューリングポリシーを動的に選択するステップと、を含む、
請求項１２に記載の方法。
【請求項１５】
動的に選択されたスケジューリングポリシーにより前記仮想マシンのうちの１つのＧＰＵコマンドをスケジュールするステップと、異なるスケジューリングポリシーにより前記仮想マシンのうちの他の１つのＧＰＵコマンドをスケジュールするステップとを含む、
請求項１２に記載の方法。
【請求項１６】
異なる仮想マシンのＧＰＵコマンドを評価するステップと、前記異なる仮想マシンのＧＰＵコマンドを、集団スケジューリングポリシーではない異なるスケジューリングポリシーによりスケジュールするステップとを含む、
請求項１２に記載の方法。
【請求項１７】
各コマンドバッファはリングバッファとして実施され、前記方法は、クロスリング同期プリミティブの仮想マシンのＧＰＵコマンドをスキャンするステップと、前記仮想マシンのＧＰＵコマンドのためのスケジューリングポリシーを、クロスリング同期プリミティブの存在または不存在に基づき選択するステップとを含む、
請求項１２に記載の方法。
【請求項１８】
異なる仮想マシンの異なるスケジューリングポリシーを、前記ＧＰＵコマンドの評価の出力に基づいて、動的に選択するステップを含む、
請求項１２に記載の方法。
【請求項１９】

各仮想マシンに対して、前記仮想マシンのコマンドバッファ間の依存性を示すデータを含むコマンドデータベースを生成するステップを含む、
請求項１２に記載の方法。
【請求項２０】
すべての仮想マシンのコマンドデータベースを評価するステップと、すべてのコマンドデータベースの評価に基づいて、少なくとも１つの仮想マシンのスケジューリングポリシーを選択するステップとを含む、
請求項１９に記載の方法。
【請求項２１】
仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンするステップと、同じ仮想マシンの異なるコマンドバッファ中のＧＰＵコマンド間のクロスバッファ依存性を検出するステップと、ある時間にわたるクロスバッファ依存性の発生の頻度を決定するステップと、クロスバッファ依存性の発生の頻度に基づいて前記スケジューリングポリシーを変更するステップとを含む、
請求項１２に記載の方法。
【請求項２２】
所定数のコマンドバッファにクロスリング同期プリミティブが検出されないことが、すべての仮想マシンについて成り立てば、スケジューリングポリシーをリングごとのポリシーとするステップとを含む、
請求項１２に記載の方法。
【請求項２３】
計算デバイスに請求項１２－２２いずれか一項に記載の方法を実行させるコンピュータプログラム。
【請求項２４】
請求項２３に記載のコンピュータプログラムを格納した一以上の機械読み取り可能記憶媒体。
【発明の詳細な説明】
【背景技術】
【０００１】
計算デバイスにおいて、グラフィックス処理ユニット（ＧＰＵ）は、数学演算を高速で実行できる電子回路を提供することにより、セントラル処理ユニット（ＣＰＵ）を補足することができる。こうするために、ＧＰＵは広範囲の並列処理及び多数の並列スレッドを利用する。ＧＰＵはその能力により、ビジュアルメディアと並列計算タスクの処理を加速するのに有用である。例えば、ＧＰＵはビデオ符号化／復号、２次元及び３次元のグラフィックスレンダリング、その他の汎用計算アプリケーションに用いることができる。ＧＰＵを仮想化するのに係わる複雑性を解消できる場合、仮想化技術を多数の異なるタイプの計算プラットフォーム上のグラフィックス処理ユニットに適用することができる。
【図面の簡単な説明】
【０００２】
ここで説明するコンセプトを、添付した図面において、限定ではなく実施例により説明する。説明を単純かつ明確にするため、図に示した要素は必ずしもスケール通りには描いていない。適当であれば、対応または類似する要素を示すため、複数の図面で同じ参照レベルを用いた。
【０００３】
【図１】ここに開示するインテリジェントＧＰＵスケジューリングを有する少なくとも１つの計算デバイスを含む計算システムの少なくとも１つの実施形態を示す簡略化したブロック図である。
【０００４】
【図２】図１のサーバ計算デバイスの環境の少なくとも１つの実施形態を示す簡略化したブロック図である。

【０００５】
【図３】図２のＧＰＵスケジューリングの環境の少なくとも１つの実施形態を示す簡略化したブロック図である。
【０００６】
【図４】図１の計算デバイスの一以上により実行されてもよいインテリジェントＧＰＵスケジューリングの方法の少なくとも一実施形態を示す簡略化したフロー図である。
【０００７】
【図５】図１の計算システムの少なくとも１つの実施形態のユースケースを示す簡略化したタイミング図である。
【図面の詳細な説明】
【０００８】
本開示のコンセプトはいろいろな修正を施したり代替的形式を取ったりすることもできるが、その具体的な実施形態を図面において実施例で示し、ここに詳細に説明する。しかし、言うまでもなく、開示した具体的な形式に本開示を限定する意図ではなく、逆に、本発明は本開示と添付した特許請求の範囲に沿ったすべての修正、等価物及び代替物をカバーするものである。
【０００９】
本明細書において「ｏｎｅ　ｅｍｂｏｄｉｍｅｎｔ」、「ａｎ　ｅｍｂｏｄｉｍｅｎｔ」、「ａｎ　ｉｌｌｕｓｔｒａｔｉｖｅ　ｅｍｂｏｄｉｍｅｎｔ」などと言う場合、記載した実施形態が、ある機能、構造、または特徴を含むが、かならずしもすべての実施形態がその機能、構造、または特徴を含んでも含まなくてもよい。さらに、かかる文言は必ずしも同じ実施形態を参照しているとは限らない。さらに、ある機能、構造、または特徴をある実施形態について説明した場合、明示的に記載していようがいまいが、他の実施形態に関するそれらの機能、構造、または特徴に影響が及ぶことは、当業者には自明である。また、言うまでもなく、「少なくとも１つのＡ、Ｂ及びＣ」は、（Ａ）、（Ｂ）、（Ｃ）、（Ａ及びＢ）、（Ｂ及びＣ）、（Ａ及びＣ）又は（Ａ、Ｂ、及びＣ）を意味し得る。同様に、「Ａ、Ｂ又はＣのうちの少なくとも１つ」との形式のリストに含まれるアイテムは、（Ａ）、（Ｂ）、（Ｃ）、（Ａ及びＢ）、（Ｂ及びＣ）、（Ａ及びＣ）、又は（Ａ、Ｂ、及びＣ）を意味し得る。
【００１０】
開示した実施形態は、幾つかの場合には、ハードウェア、ファームウェア、ソフトウェア、またはそれらの任意の組み合わせで実装できる。開示の実施形態は、一時的又は非一時的な機械読み取り可能（例えば、コンピュータ読み取り可能）記憶媒体により担われ、又はそれに格納された（一以上のプロセッサにより読み取られ実行され得る）命令として実装することもできる。機械読み取り可能記憶媒体は、機械により読み取り可能な形式で情報を格納又は送信する任意のストレージデバイス、メカニズム、又はその他の物理的構造（例えば、揮発性又は不揮発性メモリ、メディアディスク、又はその他のメディアデバイス）として実施できる。
【００１１】
図中、幾つかの構造的又は方法フィーチャを具体的な構成及び／又は順序で示すかも知れない。しかし、言うまでもなく、かかる具体的な構成及び／又は順序は必須ではない場合がある。むしろ、幾つかの場合には、かかるフィーチャは、例示した図面に示したのとは異なる方法及び／又は順序で構成することもできる。また、ある図面に構造的又は方法フィーチャを含めることは、かかるフィーチャがすべての実施形態において必要であることを意味するのではなく、幾つかの実施形態では、含まれなくてもよいし、他のフィーチャと組み合わされてもよい。
【００１２】
図１を参照して、計算システム１００の一実施形態はクライアント計算デバイス１１０とサーバ計算デバイス１４２とを含む。クライアント計算デバイス１１０とサーバ計算デバイス１４２は一以上のネットワーク１４０に通信可能に結合している。クライアント計

算デバイス１１０とサーバ計算デバイス１４２の一方または両方は、ここに開示の技術を利用するように構成されてもよい。そのため、クライアント計算デバイス１１０とサーバ計算デバイス１４２のうち一方または両方は、グラフィックス処理ユニット１２６、１５２と、ＧＰＵスケジューラモジュール１３８、１７６を含むＧＰＵ仮想化を提供できる仮想化サービス１３２、１６２を備えていてもよい。説明を容易にするため、「グラフィックス処理ユニット」すなわち「ＧＰＵ」は、なかんずく、グラフィックス処理ユニット、グラフィックスアクセラレータ、またはその他のタイプの専門的電子回路またはデバイス、例えば汎用ＧＰＵ（ＧＰＧＰＵ）、ビジュアル処理ユニット、アクセラレーテッド処理ユニット（ＡＰＵ）、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）、またはその他のデバイスまたは回路であって、計算デバイス１１０、１４２により用いられてグラフィックスタスク及び／又はその他の計算動作であってアクセラレーテッド処理により利益があるものを指す。例示の仮想サービス１３２、１６２は、ネイティブグラフィックドライバが各ＶＭ内で実効されるように、複数の異なる仮想マシン（ＶＭ）を含むＧＰＵ１２６、１５２の仮想環境を確立するように構成されている。

【００１３】

例示のＧＰＵスケジューラモジュール１３８、１７６は、ＧＰＵハードウェア（ＧＰＵ１２６、ＧＰＵ１５２）上で実効するために異なるＶＭにより発行される、あるタイプのＧＰＵコマンドのスケジューリングを処理する。例えば、ＧＰＵスケジューラモジュール１３８、１７６は、特権コマンド（ｐｒｉｖｉｌｅｇｅｄ　ｃｏｍｍａｎｄｓ）のスケジューリングを処理してもよく、ＶＭのネイティブグラフィックスドライバはＧＰＵ１２６、１５２のフレームバッファやコマンドバッファなどの性能影響リソース（ｐｅｒｆｏｒｍａｎｃｅ　ｃｒｉｔｉｃａｌ　ｒｅｓｏｕｒｃｅｓ）に直接アクセスできる。後でより詳しく説明するように、ＧＰＵスケジューラモジュール１３８、１７６は、ＧＰＵコマンド、ＧＰＵコマンドバッファ依存性、及び／又はその他の決定基準のうち一以上の属性に基づいて、可能性のある複数のスケジューリングポリシーの中から１つのスケジューリングポリシーを動的に選択し、動的に選択されたスケジューリングポリシーに応じてＧＰＵコマンドをスケジューリングする。このように、効率を高くするなどの理由で、ＧＰＵスケジューラモジュール１３８、１７６は、なかんずく、ＶＭ及び／又はＧＰＵコマンドのうち一以上に適用可能なスケジューリングポリシーをインテリジェントに変更できる。一例として、ＶＭごとに実行するＧＰＵワードロードのタイプが異なるとき、性能を向上するため、又はその他の理由で、ＧＰＵスケジューラモジュール１３８、１７６は、集団スケジューリングポリシー（ｇａｎｇ　ｓｃｈｅｄｕｌｉｎｇ　ｐｏｌｉｃｙ）ではなく、バッファごとのスケジューリングポリシー（例えば、リングバッファの場合には「リングごとの」ポリシー）を実装できる。ここで、「ワークロード」とは、なかんずく、一組のＧＰＵコマンド（一以上のＧＰＵコマンドを含んでもよい）を指しても良い。さらに、デッドロック状態を回避するなどの理由で、ＧＰＵスケジューラモジュール１３８、１７６は、クロスバッファ同期（ｃｒｏｓｓ－ｂｕｆｆｅｒ　ｓｙｎｃｈｒｏｎｉｚａｔｉｏｎ）をするため、必要に応じて集団スケジューリングポリシー（ｇａｎｇ　ｓｃｈｅｄｕｌｉｎｇ　ｐｏｌｉｃｙ）に切り替えられる。

【００１４】

ここでクライアント計算デバイス１１０をより詳細に見ると、例示のクライアント計算デバイス１１０は、中央処理ユニット（ＣＰＵ）１１２とグラフィックス処理ユニット１２６とを含む。ＣＰＵはＧＰＵコマンドを含むワークロードをＧＰＵ１２６に送り、これは一般的にＣＰＵメモリ１１６とＧＰＵメモリ１２８との間のダイレクトメモリアクセスにより行われる。クライアント計算デバイス１１０は、ここに説明の機能を実行するあらゆるタイプのデバイスとして実施してもよい。例えば、クライアント計算デバイス１１０は、スマートフォン、タブレットコンピュータ、ウェアラブル計算デバイス、ラップトップコンピュータ、ノートブックコンピュータ、モバイル計算デバイス、セルラフォン、ハンドセット、メッセージングデバイス、車両テレマティックスデバイス、サーバコンピュータ、ワークステーション、分散計算システム、マルチプロセッサシステム、コンシュー

マエレクトロニクスデバイス、及び／又はここに説明の機能を実行するように構成されたその他の任意の計算デバイスとして実施してもよく、これらに限定されない。図１に示すように、クライアント計算デバイス１１０はさらに、入出力サブシステム１１４、データストレージデバイス１１８、ディスプレイ１２０、計算サブシステム１２２、ユーザインターフェースサブシステム１２４、オペレーティングシステム１３０、仮想サービス１３２、グラフィックスドライバー１３４、及びＧＰＵスケジューラモジュール１３８を含む。クライアント計算デバイス１１０はさらに、ＧＰＵ１２６とＧＰＵメモリ１２８とを含む。クライアント計算デバイス１１０は、他の一実施形態では、モバイル及び／又は据え置き型コンピュータに一般的に見つかるようなその他の又は追加的なコンポーネント（例えば、様々なセンサーや入出力デバイスなど）を含み得る。また、幾つかの実施形態では、例示したコンポーネントのうち一以上は、他のコンポーネントに組み込まれていても良く、又はその一部を構成していても良い。例えば、幾つかの実施形態では、ＣＰＵメモリ１１６又はその部分はＣＰＵ１１２に組み込まれてもよい、及び／又はＧＰＵメモリ１２８はＧＰＵ１２６に組み込まれてもよい。クライアント計算デバイス１１０のコンポーネントは、ソフトウェア、ファームウェア、ハードウェア、又はソフトウェアとハードウェアの組み合わせとして実施されてもよい。

【００１５】

ＣＰＵ１１２は、ここに説明する機能を実行できる任意のタイプのプロセッサとして実施してもよい。例えば、このＣＰＵ１１２は、シングル又はマルチコアプロセッサ、デジタルシグナルプロセッサ、マイクロコントローラ、又はその他のプロセッサ又は処理／制御回路として実施できる。　ＧＰＵ１２６は、ここに説明する機能を実行できる任意のタイプのグラフィックス処理ユニットとして実施してもよい。例えば、このＧＰＵ１２６は、シングルコア又はマルチコアプロセッサ、デジタルシグナルプロセッサ、マイクロコントローラ、浮動小数点演算アクセラレータ、コ・プロセッサ又はその他のプロセッサ又は処理／制御回路であってメモリ中のデータを高速で操作及び変更するように設計されたものとして実施してもよい。図と説明を単純化するため、クライアント計算デバイス１１０の幾つかの側面を、後で説明するサーバ計算デバイス１４２を参照して、より詳しく図示し説明する。例えば、ＧＰＵ１２６、ＧＰＵメモリ１２８、及びＧＰＵスケジューラモジュール１３８の態様を、対応するサーバ計算デバイス１４２のコンポーネントを参照して、より詳しく説明する。一般的に、計算デバイス１１０、１４２の一方のコンポーネントに関する説明は、他方の計算デバイス１１０、１４２の同様のコンポーネントに等しくあてはまる。

【００１６】

クライアント計算デバイス１１０のＣＰＵメモリ１１６とＧＰＵメモリ１２８とは、それぞれここに説明する機能を実行できる任意のタイプの揮発性又は不揮発性メモリ又はデータストレージとして実施してもよい。動作中、メモリ１１６は、オペレーティングシステム、アプリケーション、プログラム、ライブラリ、及びドライバなど、計算デバイス１１０の動作中に用いられる様々なデータとソフトウェアを記憶できる。例えば、ＣＰＵメモリ１１６の一部は、少なくとも一時的に、コマンドバッファと、ＣＰＵ１１２により生成されるＧＰＵコマンドとを記憶し、ＧＰＵメモリ１２８の一部は、少なくとも一時的に、ＣＰＵメモリから受け取ったＧＰＵコマンドを記憶してもよい。

【００１７】

ＣＰＵメモリ１１６は、例えばＩ／Ｏサブシステム１１４を介してＣＰＵ１１２と通信可能に結合され、ＧＰＵメモリ１２８は、同様に、ＧＰＵ１２６に通信可能に結合される。Ｉ／Ｏサブシステム１１４は、ＣＰＵ１１２、ＣＰＵメモリ１１６、ＧＰＵ１２６、ＧＰＵメモリ１２８、及びその他のクライアント計算デバイス１１０のコンポーネントとの入出力動作をさせる（ｆａｃｉｌｉｔａｔｅ）回路及び／又はコンポーネントとして実施されてもよい。例えば、Ｉ／Ｏサブシステム１１４は、メモリコントローラハブ、入出力制御ハブ、ファームウェアデバイス、通信リンク（すなわち、ポイント・ツー・ポイントリンク、バスリンク、ワイヤ、ケーブル、光ガイド、プリント回路板トレースなど）及び

／又は入出力動作を容易にするその他のコンポーネント及びサブシステムとして、又はこれらを含むものとして実施できる。幾つかの実施形態では、Ｉ／Ｏサブシステム１１４は、システム・オン・チップ（ＳｏＣ）の一部を形成してもよく、単一の集積回路チップ上に、ＣＰＵ１１２、ＣＰＵメモリ１１６、ＧＰＵ１２６、ＧＰＵメモリ１２８、及び／又は計算デバイス１１０のその他のコンポーネントとともに、組み込むことができる。

【００１８】

データストレージデバイス１１８は、例えば、メモリデバイスと回路、メモリカード、ハードディスクドライブ、固体ドライブ、その他の記憶デバイスなどの、データを短期又は長期にわたり格納するように構成された任意のタイプのデバイスとして実施できる。データストレージデバイス１１８は、計算デバイス１１０のデータとファームウェアコードを記憶するシステムパーティションを含んでいてもよい。データストレージデバイス１１８は、計算デバイス１１０のオペレーティングシステム１３０のデータファイル及び実行ファイルを記憶するオペレーティングシステムパーティションを含んでいてもよい。

【００１９】

ディスプレイ１２０は、液晶ディスプレイ（ＬＣＤ）、発光ダイオード（ＬＥＤ）、プラズマディスプレイ、陰極線管（ＣＲＴ）、又はその他のタイプのディスプレイデバイスなど、デジタル情報を表示できる任意のタイプのディスプレイとして実施されてもよい。幾つかの実施形態では、ディスプレイ１２０は、計算デバイス１１０とのユーザインターラクションを可能とするタッチスクリーン又はその他のユーザ入力デバイスに結合されても良い。ディスプレイ１２０は、ユーザインターラクションサブシステム１２４の一部であってもよい。ユーザインターラクションサブシステム１２４は、物理的又は仮想的制御ボタン又はキー、マイクロホン、スピーカ、一方向又は双方向のスチル及び／又はビデオカメラ、その他を含む計算デバイス１１０とユーザインターラクションをさせる複数の付加的デバイスを含んでいてもよい。ユーザインターフェースサブシステム１２４は、動きセンサー、近接センサー、視線追跡デバイスなどのデバイスを含んでいてもよく、これらのデバイスは計算デバイス１１０と関わるその他の形式のヒトとのインターラクションを検出、捕捉、及び処理するように構成されてもよい。

【００２０】

計算デバイス１１０はさらに通信サブシステム１２２を含む。これは計算デバイス１１０とその他の電子デバイスとの間の通信を可能にする任意の通信回路、デバイスまたはそれらの集まりとして実施されてもよい。通信サブシステム１２２は、一以上の任意の通信技術（例えば、無線または有線通信）と、かかる通信を行う関連プロトコル（例えば、イーサネット（登録商標）、Ｂｌｕｅｔｏｏｔｈ（登録商標）、Ｗｉ－Ｆｉ（登録商標）、ＷｉＭＡＸ、３Ｇ／ＬＴＥなど）とを用いるように構成されていてもよい。通信サブシステム１２２は、無線ネットワークアダプタを含むネットワークアダプタとして実施されてもよい。

【００２１】

例示の計算デバイス１１０は、仮想化サービス１３２、グラフィックスドライバー１３４、オペレーティングシステム１３０、及びＧＰＵスケジューラモジュール１３８などの複数のコンピュータプログラムコンポーネントも含む。特に、オペレーティングシステム１３０は、ＧＰＵスケジューラモジュール１３８と仮想化サービス１３２などのコンピュータプリケーションと、計算デバイス１１０のハードウェアコンポーネントとの間の通信を可能とする（ｆａｃｉｌｉｔａｔｅ）。オペレーティングシステム１３０は、Ｍｉｃｒｏｓｏｆｔ　Ｃｏｒｐｏｒａｔｉｏｎのウィンドウズ（登録商標）、Ｇｏｏｇｌｅ，　Ｉｎｃ．のアンドロイドなどの一バージョンなど、ここに説明の機能を実行できる任意のオペレーティングシステムとして実施してもよい。ここで、「コンピュータアプリケーション」とは、なかんずく、エンドユーザが計算デバイス１１０とインターラクトしてもよい「ユーザスペース」ソフトウェア及び／又はハードウェアアプリケーションを指しても、プログラミングコードが計算デバイス１１０のハードウェアコンポーネントと直接インターラクトできる「システムスペース」ハードウェア及び／又はソフトウェアアプリケーシ

ョンを指しても良い。計算デバイス１１０のシステムスペースコンポーネントは、計算デバイス１１０のユーザスペースコンポーネントより大きな特権を有していてもよい。

【００２２】

例示の実施形態では、グラフィックスドライバー１３４は、コンピュータプリケーションとハードウェアコンポーネント（ディスプレイ１２０など）との間の通信を処理する。幾つかの実施形態では、グラフィックスドライバー１３４は、例えば、デバイス独立グラフィックスレンダリングタスクを異なるハードウェアコンポーネント（例えば、異なるタイプのディスプレイ）に通信できる「汎用」ドライバと、デバイス独立タスクを特定ハードウェアコンポーネントが実行して要求タスクを実現できるコマンドに変換する「特定デバイス」ドライバとを含んでいてもよい。他の実施形態では、汎用及び特定デバイスドライバの部分を結合して、１つのドライバーコンポーネント（例えば、グラフィックスドライバー１３４）にしてもよい。幾つかの実施形態では、グラフィックスドライバー１３４の部分は、オペレーティングシステム１３０に含まれてもよい。グラフィックスドライバー１３４は例示的にディスプレイドライバである。しかし、開示のＧＰＵスケジューラモジュール１３８の態様は、例えば、（例えば、ＧＰＵ１２６がＧＰＧＰＵとして構成されている場合に）ＧＰＵ１２６にオフロードされてもよい任意の種類のタスクなどの他のアプリケーションで用いてもよい。

【００２３】

例示の仮想化サービス１３２は、ある種のハイパーバイザのとして実施される。これはファームウェアから直接ブートローダにより、又はオペレーティングシステム１３０により起動されてもよい。仮想化サービス１３２は、「シン（ｔｈｉｎ）」ハイパーバイザ又は従来のハイパーバイザ、仮想マシンマネージャ（ＶＭＭ）、又は同様の仮想化プラットフォームとして実施されてもよい。例えば、仮想化サービス１３２は、ＸＥＮベース（タイプＩ）ＶＭＭ、カーネルベース仮想マシン（ＫＶＭ）ベース（タイプＩＩ）ＶＭＭ、又はウィンドウズベースＶＭＭとして実施してもよい。幾つかの実施形態では、仮想化サービス１３２は、「ベアメタル」ハイパーバイザとして実施してもよい。これはシステムハードウェアから直接実行できる。仮想化サービス１３２は、計算デバイス１１０、特にＧＰＵ１２６の共有リソースの仮想化を可能とし管理する特権ソフトウェア又はファームウェアコンポーネントとして実施される。このように、仮想化サービス１３２は、計算デバイス１１０のより高い特権システムモードであって、仮想化サービス１３２がＧＰＵ１２６及び／又は計算デバイス１１０の他のハードウェアリソースをほぼ完全にコントロールする実行される。上記の通り、仮想化サービス１３２は、複数の仮想マシンを含むＧＰＵ１２６の仮想化環境を確立でき、各仮想マシンはグラフィックスドライバー１３４の自分自信のインスタンスを実行する。ＧＰＵ仮想化サービスの例は、Ｉｎｔｅｌ　Ｃｏｒｐ．のＸｅｎＧＴとｎＶｉｄｉａ　ＣｏｒｐのＧＲＩＤ　ＶＧＺを含む。ＧＰＵスケジューラモジュール１３８は、仮想化サービス１３２のコンポーネントとして実施してもよい。動作中、ＧＰＵスケジューラモジュール１３８は、ＶＭ中のグラフィックスドライバー１３４の仮想インスタンスと通信し、上記の通り、ＧＰＵコマンドのＧＰＵ１２６への送信を制御する。

【００２４】

ここでサーバ計算デバイス１４２をより詳細に参照する。サーバ計算デバイス１４２は、ここに説明する機能を実行する任意のタイプのデバイスとして実施してもよい。例えば、サーバ計算デバイス１４２は、スマートフォン、タブレットコンピュータ、ウェアラブル計算デバイス、ラップトップコンピュータ、ノートブックコンピュータ、モバイル計算デバイス、セルラフォン、ハンドセット、メッセージングデバイス、車両テレマティックスデバイス、サーバコンピュータ、ワークステーション、分散計算システム、マルチプロセッサシステム、コンシューマエレクトロニクスデバイス、及び／又はここに説明の機能を実行するように構成されたその他の任意の計算デバイスとして実施してもよく、これらに限定されない。クライアント計算デバイス１１０の上記のコンポーネントと同じ又は類似の名称を有するサーバ計算デバイス１４２のコンポーネントは、同様に実施してもよく

、したがって、ここで説明は繰り返さない。さらに、言うまでもなく、クライアント計算デバイス１１０は、サーバ計算デバイス１４２のどのコンポーネントを含んでいても良く、かかるコンポーネントの説明はクライアント計算デバイス１１０の同様のコンポーネントに等しく当てはまる。

【００２５】

例示のサーバ計算デバイス１４２は、ＣＰＵ１４４、入出力サブシステム１４６、ダイレクトメモリアクセス（ＤＭＡ）サブシステム１４８、ＣＰＵメモリ１５０、オペレーティングシステム１６０、仮想化サービス１６２、グラフィックスドライバモジュール１６４、データストレージデバイス１６６、ディスプレイ１６８、通信サブシステム１７０、ユーザインターフェースサブシステム１７２、及びＧＰＵスケジューラモジュール１７６を含む。サーバ計算デバイス１４２はさらに、ＧＰＵ１５２、レンダーエンジン１５４、ＧＰＵメモリ１５６、及びコマンドバッファ１５８を含む。サーバ計算デバイス１４２は、他の一実施形態では、モバイル及び／又は据え置き型コンピュータに一般的に見つかるようなその他の又は追加的なコンポーネント（例えば、様々なセンサーや入出力デバイスなど）を含み得る。また、幾つかの実施形態では、例示したコンポーネントのうち一以上は、他のコンポーネントに組み込まれていても良く、又はその一部を構成していても良い。サーバ計算デバイス１４２のコンポーネントは、ソフトウェア、ファームウェア、ハードウェア、又はソフトウェアとハードウェアの組み合わせとして実施されてもよい。

【００２６】

ＧＰＵ１５２は、複数のレンダーエンジン１５４を含み、これはＧＰＵ１５２のハードウェア実行ユニット、例えばプロセッサコア又は並列プロセッサのアレイであって、それぞれ複数の並列スレッドを実行できるものとして実施してもよい。ＧＰＵ１５２は（例えば、個別のグラフィックスカード上の）周辺デバイスとして実施されてもよいし、ＣＰＵマザーボード上にあってもＣＰＵダイ上にあってもよい。レンダーエンジン１５４はそれぞれ、あるタイプのＧＰＵタスクを処理するように構成されていてもよい。例えば、幾つかの実施形態では、複数の異なるレンダーエンジン１５４が、３Ｄレンダリングタスク、ｂｌｉｔｔｅｒ（例えば、２Ｄグラフィックス）ビデオ、及びビデオ符号化／復号タスクを個別に処理するように構成されていてもよい。

【００２７】

ＣＰＵメモリ１５０の一部は、コマンドバッファと、ＣＰＵ１４４により生成されるＧＰＵコマンドを少なくとも一時的に記憶してもよく、ＧＰＵメモリ１５６の一部は、コマンドバッファ１５８にＧＰＵコマンドを少なくとも一時的に記憶する。ＧＰＵコマンドは、ＣＰＵ１４４により、ダイレクトメモリアクセスサブシステム１４８によりコマンドバッファ１５８に転送される。ダイレクトメモリアクセス（ＤＭＡ）サブシステム１４８は、ＣＰＵメモリ１５０とＧＰＵメモリ１５６との間のデータ転送を容易にする。幾つかの実施形態では、ＤＭＡサブシステム１４８により、ＧＰＵ１５２はＣＰＵメモリ１５０に直接アクセスでき、ＣＰＵ１４４はＧＰＵメモリ１５６に直接アクセスできる。ＤＭＡサブシステム１４８は、Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ（ＰＣＩ）デバイス、Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ－Ｅｘｐｒｅｓｓ（ＰＣＩ－Ｅｘｐｒｅｓｓ）デバイス、Ｉ／Ｏ Ａｃｃｅｌｅｒａｔｉｏｎ Ｔｅｃｈｎｏｌｏｇｙ（Ｉ／ＯＡＴ）デバイスその他の、ＤＭＡコントローラ又はＤＭＡ「エンジン」として実施されてもよい。

【００２８】

例示のコマンドバッファ１５８はリングバッファとして実施され、各リングバッファは多数のバッチバッファをつなげてもよい。幾つかの実施形態では、計算デバイス１４２は、各レンダーエンジン１５４に対して異なるコマンドバッファ（例えば、３Ｄ、ｂｌｉｔｔｅｒ、ビデオ、及びビデオ符号化／復号エンジン１５４のそれぞれに対する別々のリングバッファ）を実装する。リングバッファは、例えば、同期入出力動作に有用な、ある種のｆｉｒｓｔ－ｉｎ／ｆｉｒｓｔ－ｏｕｔ（ＦＩＦＯ）データ構造である。他の実施形態では、他のタイプのＦＩＦＯデータ構造、又はその他の好適なタイプのデータ構造を用い

ても良い。

【００２９】

仮想化サービス１６２を通じて、仮想化サービス１６２により確立された各ＶＭが一組のコマンドバッファ１５８（例えば、各ＶＭ中の３Ｄ、ｂｌｉｔｔｅｒ、ビデオ及びビデオ符号化／復号コマンドバッファ）を含み、各コマンドバッファが異なるコマンドパーサ（例えば、３Ｄ、ｂｌｉｔｔｅｒ、ビデオ、及びビデオ符号化／復号コマンドパーサ）により並列に解析されるように、コマンドバッファ１５８が仮想化される。ＧＰＵスケジューラモジュール１７６は、新しいＶＭがＧＰＵ１５２にアクセスするようスケジュールされている場合に、コンテキスト切り替え（ｃｏｎｔｅｘｔ　ｓｗｉｔｃｈ）を行っても良い。コンテキスト切り替えに応じて、ＧＰＵ１５２は一組の異なるコマンドバッファ（例えば、その新しいＶＭに関連するコマンドバッファ）を提供してもよい。

【００３０】

例示のＧＰＵスケジューラモジュール１７６は、コマンドスキャナーモジュール１７８、コマンドデータベース１８０、アービターモジュール１８２、及び複数のスケジューリングポリシー１８４を含む。ＧＰＵスケジューラモジュール１７６、コマンドスキャナーモジュール１７８、コマンドデータベース１８０、及びアービターモジュール１８２は、任意のタイプのプロセッサ実行可能コード、モジュール及び／又はデータ構造、又はこれらの組み合わせとして実装してもよい。図３を参照して、コマンドスキャナーモジュール１７８、コマンドデータベース１８０、及びアービターモジュール１８２の態様をより詳しく説明する。

【００３１】

例示的に、スケジューリングポリシー１８４は、リングごとのスケジューリングポリシー１８６と、集団スケジューリングポリシー１８８とを含む。リングごとのスケジューリングポリシー１８６により、リングバッファは、互いに独立にコンテキスト切り替えでき、それにより異なるＶＭからのＧＰＵコマンドが異なるレンダーエンジン１５４を利用する場合には、それらを同時にスケジュールできる。しかし、リングごとの（ｐｅｒ－ｒｉｎｇ）スケジューリングポリシーを用いると、次の場合にデッドロック問題が生じ得る。異なるバッファ上のＧＰＵコマンドがそれぞれ他方のコマンドによりシグナルされる状態に依存する場合、又は異なるバッファ上の２つのコマンドが、同じ条件が満たされることに依存する場合である。ＶＭのあるリングバッファ上のあるコマンドの、同じＶＭの他のリングバッファ上の他のコマンドによりシグナルされた結果への依存性は、ｗａｉｔコマンドが依存する状態とともに、コマンドバッファ中に同期または「ｗａｉｔ」コマンドがあることにより証拠だてられる。この種のコマンド依存性は、（同じＶＭの異なるバッファ上のＧＰＵコマンドの）「クロスリング同期」またはより一般的にはクロスバッファ依存性又はクロスバッファ同期と呼んでもよい。

【００３２】

一例として、３Ｄエンジン１５４で実行するため、リングごとのスケジューリングポリシー１８６が、第１の仮想マシンの３Ｄリングバッファから ＧＰＵワークロードを受け取りスケジューリングするものとする。リングごとのスケジューリングポリシー１８６は、他の仮想マシンＶＭ２のｂｌｉｔｔｅｒリングバッファからＧＰＵワークロードも受け取る。リングごとのスケジューリングポリシーにしたがって、ＶＭ２ブリッタタスクをブリッタエンジン１５４でスケジュールすることができる。ブリッタエンジン１５４はＶＭ１により使われないからである。ＶＭ１　３Ｄワークロードは、３Ｄレンダリングタスクに関する複数のＧＰＵコマンドを含むが、３Ｄレンダリングタスクは、ブリッタ（ｂｌｉｔｔｅｒ）エンジン１５４でスケジュールされる必要がある他のＶＭＩタスクと同期されねばならない。これはＶＭＩ３Ｄワークロード中のｗａｉｔコマンドにより証拠付けられ（ｅｖｉｄｅｎｃｅｄ）、ｗａｉｔコマンドは状態値ＣＯＮＤ１に依存する。値ＣＯＮＤ１はＶＭ１ブリッタタスクによりシグナルされる。しかし、ＶＭ１ブリッタワークロードは、ｗａｉｔコマンドも含む。これは状態値ＣＯＮＤ２に依存するが、ＣＯＮＤ２はＶＭ１　３Ｄタスクによりシグナルされる。ＶＭ１ブリッタタスクは、ＶＭ２ブリッタタスクが

終了するまで、スケジュールされるのを待たねばならない。ＶＭ１　３ＤタスクはＶＭ１ブリッタタスクがＣＯＮＤ１をシグナルするまで完了できず、ＶＭ１ブリッタタスクはＶＭ１　３ＤタスクがＣＯＮＤ２をシグナルするまで完了できない。リングごとのスケジューリングポリシー１８６にしたがって、２つのＶＭ１タスクが異なる時間にスケジュールされ（例えば、ブリッタエンジン１５４が利用できるので、リングごとのポリシーによりＶＭ２ブリッタコマンドがスケジュールされたことによる）、デッドロック状態が生じ、ＧＰＵ１５２による処理がハングする。ＧＰＵ１５２がコマンドプリエンプションサポートを有する構成になっており、ｗａｉｔコマンドがあるタイムアウト期間後にプリエンプト（ｐｒｅｅｍｐｔ）され得ても、２つの異なるコマンドが同じＣＯＮＤ値を使ってしまった場合、ＶＭ１中のｗａｉｔコマンドは、ＶＭ２からシグナルされるＣＯＮＤ値により不正に完了されるおそれがある。

【００３３】

集団スケジューリングポリシー１８８は、ＶＭ中のすべてのリングバッファが、リング毎のスケジューリングポリシー１８６では独立であるのと異なり、一緒にコンテキスト切り替えされることを要する。集団スケジューリングポリシー１８８は、クロスリング同期問題の解消に使える。上記の例を続けて、リング毎のスケジューリングポリシー１８６の替わりに、集団スケジューリングポリシー１８８を用いるとする。この場合、集団スケジューリングポリシー１８８は、（３Ｄエンジン及びブリッタエンジン１５４にそれぞれ）ＶＭ１　３ＤコマンドとＶＭ１ブリッタコマンドとを一緒にスケジュールする。ＶＭ１ブリッタコマンドがＣＯＮＤ１をシグナルする時、ＶＭ１　３Ｄコマンドは継続できる。ＶＭ１　３ＤコマンドがＣＯＮＤ２をシグナルする時、ＶＭ１ブリッタコマンドは継続できる。ＶＭ２からのコマンドは、すべてのＶＭ１リングバッファからのすべてのＶＭ１コマンドが終わるのを待って、スケジュールされなければならない。

【００３４】

しかし、集団スケジューリングポリシー１８８はＧＰＵ仮想化においては非効率的になり得る。例えば、ＧＰＵスケジューラモジュール１７６が３Ｄエンジン１５４で実行するＶＭ１ワークロードをスケジュールし、他のどのＶＭ１バッファには他のワークロードが無いものとする。（ＶＭ１ビデオバッファにはＧＰＵコマンドが無いため）ビデオエンジン１５４は、（例えば）ＶＭ１で使われていなくても、集団スケジューリングポリシー１８４により、ＶＭ１　３Ｄタスクが完了するまで、ビデオエンジン１５４を用いてＶＭ２ビデオタスクを実行できない。

【００３５】

ここで図２を参照して、幾つかの実施形態では、サーバ計算デバイス１４２は、動作中に環境２００を確立する。環境２００は、特権仮想環境２１０、ＧＰＵスケジューラモジュール２１４、グラフィックスドライバモジュール２１６、コマンドバッファ２２０、コマンドパーサ２２２、グラフィックスドライバモジュール２１８、仮想化サーバ２２６、及びＧＰＵハードウェア１５２を含む。環境２００の様々なモジュールは、ハードウェア、ファームウェア、ソフトウェア、またはこれらの組み合わせとして実施されてもよい。例示の環境２００は、仮想化サービス１６２の実行インスタンス（仮想化サービス２２６）を含む。これは特権仮想環境２１０と「Ｎ個の」仮想マシン２１２（Ｎは正整数）とを確立する。特権仮想環境は、（幾つかの実装では「ドメイン０」と呼ばれ、）ＧＰＵスケジューラモジュール１７６の実行インスタンス（ＧＰＵスケジューラモジュール２１４）と、グラフィックスドライバモジュール１６４の実行インスタンス（グラフィックスドライバモジュール２１６）とを含む。各ＶＭ２１２は、グラフィックスドライバモジュール１６４の実行インスタンス（グラフィックスドライバモジュール２１８）、各コマンドバッファ１５８のインスタンス（コマンドバッファ２２０）、及びコマンドパーサ２２２を含む。各ＶＭのコマンドバッファ２２０は、例えば、３Ｄ、ブリッタ、ビデオ、及びビデオ符号化／復号リングバッファを含む。グラフィックスドライバモジュール２１６、２１８は、非特権コマンド２３０、２３２を、ＧＰＵハードウェア１５２に直接送っても良い。特権ＧＰＵコマンド２２４は、ＧＰＵスケジューラモジュール２１４により（例えば、

トラップアンドエミュレーション法により）処理され、ここに説明の動的ハイブリッドスケジューリングアプローチを用いてＧＰＵ１５２に送られる。

【００３６】

ＧＰＵスケジューラモジュール２１４は、全ＶＭの全コマンドバッファのＧＰＵコマンドを評価し、ＧＰＵコマンドの評価の出力に応じて、複数の異なるスケジューリングポリシーからスケジューリングポリシーを動的に選択する。ＧＰＵスケジューラモジュール２１４は、動的に選択されたスケジューリングポリシーにより、ＧＰＵにより処理するため、少なくとも１つのＧＰＵコマンドをスケジュールする。ＧＰＵスケジューラモジュール２１４により実行される「動的」及び／又は「ハイブリッド」のスケジューリングの幾つかの例を説明する。ＧＰＵスケジューラモジュール２１４は、２つの異なるスケジューリングポリシーにより、２つの異なるＶＭのＧＰＵコマンドをスケジューリングしてもよい。ＧＰＵスケジューラモジュール２１４は、例えば、同じ仮想マシンの異なるコマンドバッファ中の２つのＧＰＵコマンド間のクロスバッファ依存性の検出に応じて、一以上のＶＭに適用可能なスケジューリングポリシーを、リングごとのスケジューリングポリシーから集団スケジューリングポリシーに切り替えても良い。ＧＰＵスケジューラモジュール２１４は、リングごとのスケジューリングポリシーから集団スケジューリングポリシーに切り替えた後、異なるＶＭのＧＰＵコマンドを評価し、異なるスケジューリングポリシーにより異なるＶＭのＧＰＵコマンドをスケジュールしてもよい（例えば、ＧＰＵスケジューラモジュール２１４は、異なるＶＭに対してリングごとのスケジューリングポリシーに切り替え戻しても良く、１つのＶＭでは集団スケジューリングポリシーを、他のＶＭでリングごとのスケジューリングポリシーを、同時に実行してもよい）。幾つかの実施形態では、ＧＰＵスケジューラは、すべての仮想マシンのすべてのコマンドバッファにわたり、クロスバッファ依存性の出現をトラックし続け、一定数のコマンドバッファに又は一定時間にわたりクロスバッファ依存性が検出されなければ、又はより一般的に、クロスバッファ依存性の発生頻度に基づいて、（例えば、リングごとのポリシーから集団ポリシーへ、又はその逆に）スケジューリングポリシーを変更してもよい。環境２００（例えば、ＧＰＵスケジューラモジュール２１４）の様々なモジュールまたはコンポーネントは、ハードウェア、ファームウェア、ソフトウェア、またはこれらの組み合わせとして実施されてもよい。また、幾つかの実施形態では、環境２００のモジュールの一部または全部は、他のモジュール又はソフトウェア／ファームウェア構造と一体かされても、またはその一部を構成してもよい。

【００３７】

ＧＰＵスケジューラモジュール２１４の動作をトリガーするため、幾つかの実施形態では、グラフィックスドライバモジュール２１６、２１８は、ＧＰＵコマンド２２４をコマンドバッファ２２０にキュー（ｑｕｅｕｅ）し、次いでメモリマップト入出力（ＭＭＩＯ）レジスタ（テール）を書き、コマンドパーサ２２２によるキューしたコマンドの解析を開始する。仮想環境（例えば、環境２００）では、最後のＭＭＩＯ書き込みがトラップされ、ＧＰＵスケジューラモジュール２１４がＶＭのＧＰＵコマンドのスケジューリングを実行できる。

【００３８】

ここで図３を参照して、幾つかの実施形態では、サーバ計算デバイス１４２は、動作中に環境３００を確立する。環境３００は、ＧＰＵスケジューラモジュール２１４、コマンドスキャナー３１４、３１６、コマンドデータベース３１８、３２０、リングごとのスケジューリングポリシー３２２、集団スケジューリングポリシー３２４、及びアービターモジュール３２６を含む。環境３００の様々なモジュールは、ハードウェア、ファームウェア、ソフトウェア、またはこれらの組み合わせとして実施されてもよい。例示の環境３００は、コマンドスキャナーモジュール１７８の実行インスタンス（コマンドスキャナーモジュール３１４、３１６）（例えば、ＶＭごとに１つのインスタンス）、コマンドデータベース３１８，３２０のインスタンス（例えば、ＶＭごとに１つのデータベース）、アービターモジュール１８２の実行インスタンス（アービターモジュール３２６）、リングご

とのスケジューリングポリシー１８６のインスタンス（リングごとのスケジューリングポリシー３２２）、及び集団スケジューリングポリシー１８８のインスタンス（集団スケジュールポリシー３２２）を含む。各コマンドスキャナーモジュール３１４、３１６は、各ＶＭの全コマンドバッファの、キューされたＧＰＵコマンドをスキャンする。例えば、ＶＭ１コマンドスキャナーモジュール３１４は、（例えば、ＶＭ１の３Ｄ、ブリッタ、ビデオ、及びビデオ符号化／復号バッファなどの複数のリングバッファを含む）ＶＭ１コマンドバッファ２２０に含まれたＧＰＵコマンドをスキャンする。例えば、ＶＭ　Ｎコマンドスキャナーモジュール３１６は、（例えば、ＶＭ　Ｎの３Ｄ、ブリッタ、ビデオ、及びビデオ符号化／復号バッファなどの複数のリングバッファを含む）ＶＭ　Ｎコマンドバッファ３１２に含まれたＧＰＵコマンドをスキャンする。

【００３９】

各コマンドスキャナーモジュール３１４、３１６は、同じ仮想マシンの異なるコマンドバッファ中のＧＰＵコマンド間のクロスバッファ依存性を示すデータを発生し、コマンドデータベース３１８、３２０を生成し、クロスバッファ依存性を示すデータをコマンドデータベース３１８、３２０に記憶する。すなわち、各コマンドスキャナーモジュール３１４、３１６は、その各ＶＭのＧＰＵコマンドにおけるクロスバッファ依存性（ｃｒｏｓｓ－ｂｕｆｆｅｒ　ｄｅｐｅｎｄｅｎｃｉｅｓ）を特定し、依存性を示すデータを、その各ＶＭのコマンドデータベース３１８、３２０に記憶する。また、コマンドスキャナーモジュール３１４、３１６は、ＶＭの各ＧＰＵのコマンドタイプを決定し、コマンドタイプを示すデータをコマンドデータベース３１８、３２０に記憶する。コマンドタイプは、例えば、３Ｄ、ブリッタ（ｂｌｉｔｔｅｒ）、ビデオ、ビデオ符号化／復号など、ＧＰＵ１５２により実行されるタスクのタイプ、又はそのタスクを実行する具体的なレンダーエンジン１５４に対応してもよい。

【００４０】

コマンドスキャナーモジュール３１４、３１６は、ワークロード発出（ｓｕｂｍｉｓｓｉｏｎ）がクラスバッファ依存性を含むと決定した場合、フラグをその発出（ｓｕｂｍｉｓｓｉｏｎ）と関連付けても良い。こうするために、コマンドスキャナーは、一組のＧＰＵコマンドのインプレース計測（ｉｎ－ｐｌａｃｅ　ｉｎｓｔｒｕｍｅｎｔａｔｉｏｎ）を実行してもよい。コマンドバッファ２２０、３１２がリングバッファとして実施される場合、コマンドスキャナーモジュール３１４、３１６は、クロスリング同期プリミティブ（ｃｒｏｓｓ－ｒｉｎｇ　ｓｙｎｃｈｒｏｎｉｚａｔｉｏｎ　ｐｒｉｍｉｔｉｖｅｓ）を特定してもよく、クロスリング同期プリミティブの存否に関するデータをコマンドデータベース３１８、３２０に記憶してもよい。例えば、コマンドスキャナーモジュール３１４、３１６は、クロスリング同期プリミティブがあるか決定してもよく、クロスリング同期プリミティブがあるとき、関連するコマンドバッファ２２０、３１２に対してフラグ（例えば、ＲＥＱ　ＧＡＮＧ）を設定してもよい。アービターモジュール３２６は、フラグをチェックし、フラグが有効に（例えば、ＲＥＱ＿ＧＡＮＧ＝ＹＥＳに）設定されているとき、アービターモジュール３２６は、リングごとのスケジューリングポリシー３２２から集団スケジューリングポリシー３２４への切り替えを開始する。集団スケジューリングポリシー３２４からリングごとのスケジューリングポリシー３２２に切り替え戻すため、アービターモジュール３２６は、経験的ポリシー（例えば、「すべてのＶＭにおいて、次のＮ個のコマンドバッファにクロスリング同期プリミティブが検出されなければ、リングごとのスケジューリングポリシー３２２に切り替え戻す」）を利用してもよい。ここで用いるように、「プリミティブ（ｐｒｉｍｉｔｉｖｅ）」は、なかんずく、計算プラットフォーム（例えば、オペレーティングシステム）により提供される単純なソフトウェアメカニズムを指しても良く、より低いレベルのメカニズム（例えば、原子的動作、メモリバリア、スピンロック、コンテキスト切り替えなど）を含んでいてもよい。一例として、複数のリングバッファにより共有されるオブジェクトのクロスリング実行をシリアル化するために、セマフォコマンド（例えば、セマフォ「ｗａｉｔ」コマンド）が、計算プラットフォームにより提供されてもよい。

【００４１】
幾つかの実施形態では、コマンドスキャナーモジュール３１４、３１６は、各ＶＭの異なるコマンドバッファ間に直接的依存性を構築してもよい。こうするため、コマンドスキャナーモジュール３１４、３１６は、ＶＭのコマンドデータベース３１８、３２０のグラフデータ構造に、依存性及びコマンドタイプ情報を追加（ｐｏｐｕｌａｔｅ）してもよい。下記のコード例１、２は、ＶＭ１とＶＭ２のためにコマンドスキャナーモジュール３１４、３１６により生成され、ＶＭ１とＶＭ２のコマンドデータベース３１８、３２０に記憶されてもよいグラフデータ構造の疑似コードの例を示す。
【数１】

```
[Buf11, 3D ring, wait for [Buf12, blitter ring, COND1]
[Buf12, blitter ring], signal [Buf11, 3D ring, COND1]
[Buf13, 3D ring], no dependency
[Buf14, blitter ring], no dependency
```

コード例１　ＶＭ１のコマンドデータベース
【００４２】
コード例１では、Ｂｕｆ１１、Ｂｕｆ１２、Ｂｕｆ１３、Ｂｕｆ１４はＶＭ１の４つのリングバッファである。コード例１は、続く依存性情報により統制されるバッファタイプ又はコマンドタイプ（例えば、３Ｄ、ブリッタ）を指定している。例えば、Ｂｕｆ１１中の３ＤコマンドはＢｕｆ１２中のブリッタコマンドに対して、ＣＯＮＤ１をシグナルするにはＢｕｆ１２中のブリッタコマンドを待つ必要があるという点で、依存性を有する。コード例１では、Ｂｕｆ１３とＢｕｆ１４は依存性を有しないと示されている。
【００４３】
【数２】

```
[Buf21, video ring], no dependency
[Buf22, video ring], no dependency
```

コード例２　ＶＭ１のコマンドデータベース
【００４４】
コード例２では、Ｂｕｆ２１とＢｕｆ２２はＶＭ２の２つのバッファである。コード例２は、依存性情報は高い粒度レベルでコマンドスキャナーモジュール３１４、３１６により構成できることを示している。例えば、幾つかのバッファは共にスケジュール（ｃｏ－ｓｃｈｅｄｕｌｅｄ）でき、一方他のバッファはリングごとにスケジュール（ｐｅｒ－ｒｉｎｇ　ｓｃｈｅｄｕｌｅｄ）される。このように、コマンドスキャナーモジュール３１４、３１６により、スケジューリングポリシーがモードに対して柔軟（ｍｏｄｅ－ｆｌｅｘｉｂｌｅ）となる。一例として、ＶＭ１が３Ｄ／ブリッタ依存性のみを有するコマンドバッファをサブミット（ｓｕｂｍｉｔ）するが、ＶＭ２はビデオワークロードを含むコマンドバッファをサブミットする場合、ＧＰＵスケジューラモジュール２１４は、集団スケジューリングポリシーを用いてＶＭ１に３Ｄ／ブリッタバッファを共にスケジュールでき、一方リングごとのスケジューリングポリシーを用いてＶＭ２のビデオワークロードをスケジュールする。
【００４５】
幾つかの実施形態では、コマンドスキャナーモジュール３１４、３１６は、コマンドデータベース３１８、３２０に閾値テスト又は閾値を導入して、現在のバッファ動作に基づきポリシー切り替えをさらに制御してもよい。例えば、クロスリング同期プリミティブを頻繁に見ると（例えば、多くのコマンドバッファで、又はある時間にわたって多くの観察で）、これは、ネイティブグラフィックスドライバ１３４が通常それらのプリミティブを

用いるようにデザインされていることを示す。結果として、コマンドスキャナーモジュール３１４、３１６は、閾値を利用して、その閾値（例えば、検出される同期プリミティブの数）に到達するかそれを越えられるまで、集団スケジューリングポリシー１８８が実装されるのを防止することができる。

【００４６】

アービターモジュール３２６は、すべての仮想マシンのコマンドデータベースを評価し、すべてのコマンドデータベース３１８、３２０の評価に基づいて、少なくとも１つの仮想マシンのスケジューリングポリシーを選択する。アービターモジュール３２６は、コマンドデータベース３１８、３２０とスケジューリングポリシー３２２、３２４を利用して、一以上のＧＰＵコマンド及び／又はＶＭのスケジューリングポリシーを動的に選択する。これを行うため、アービターモジュール３２６は、コマンドデータベース３１８、３２０中の情報を、スケジューリングポリシー３２２、３２４を考慮して、分析する。この分析の結果として、アービターモジュール３２６は、リングごとのスケジューリングポリシー３２２から集団スケジューリングポリシー３２４への切り替え、又は集団スケジューリングポリシー３２４からリングごとのスケジューリングポリシー３２２への切り替えを行っても（ｉｎｉｔｉａｔｅ）良い。リングごとのスケジューリングポリシー３２２から集団スケジューリングポリシー３２４への切り替えにおいて、アービターモジュール３２６は、必要に応じて、すべてのリングバッファがアイドルになるのを待つ待ち合わせプロセス（ｒｅｎｄｅｚｖｏｕｓ　ｐｒｏｃｅｓｓ）を行っても良い。

【００４７】

ここで図４を参照して、ＧＰＵサブミッションのスケジューリングポリシーを動的に選択する方法４００の一例を示す。方法４００の部分々々は、計算デバイス１１０又は計算デバイス１４２のハードウェア、ファームウェア、及び／又はソフトウェアにより、例えばＣＰＵ１１２、１４４及びＧＰＵ１２６、１５２により実行されてもよい。ブロック４１０において、計算デバイス１１０、１４２は、ＶＭからＧＰＵサブミッション（ｓｕｂｍｉｓｓｉｏｎ）を受け取る。ここで、ＶＭは計算デバイス１１０、１４２のＧＰＵ仮想化サービスにより確立された複数のＶＭのうちの１つであり、ＶＭは複数の異なるコマンドバッファとネイティブグラフィックスドライバとを有する。サブミッションはＶＭの異なるコマンドバッファからのＧＰＵコマンドを含む。ブロック４１２において、計算デバイス１１０、１４２はサブミッションのＧＰＵコマンドをスキャンする。そうする際、計算デバイス１１０、１４２は、ＧＰＵコマンドのセマンティックスをチェックし、コマンドタイプ（例えば、どのエンジン１５４がコマンド（３Ｄ、ブリッタ、ビデオ、又はビデオ符号化／復号）を処理すべきか）を決定し、及び／又はクロスバッファ依存性（例えば、クロスリング同期プリミティブの存在）があるか決定する。

【００４８】

ブロック４１４において、計算デバイス１１０、１４２は、クロスバッファ依存性を（もしあれば）特定する。クロスバッファ依存性が特定されると、ブロック４１６において、計算デバイス１１０、１４２はＶＭサブミッションにフラグ（例えば、ＲＥＱ＿ＧＡＮＧ＝ＹＥＳ）を関連付ける。クロスバッファ依存性が特定されなければ、計算デバイス１１０、１４２は、ブロック４１８に進み、ＶＭからのサブミッション中の他のＧＰＵコマンドを（そのサブミッションが、スキャンすべき他のコマンドを含んでいれば）スキャンする。スキャンすべき他のＧＰＵコマンドがあれば、計算デバイス１１０、１４２はブロック４１２に戻る。サブミッション中にスキャンされる必要がある他のＧＰＵコマンドが無ければ、計算デバイス１１０、１４２は、ブロック４２０に進み、ＶＭのコマンドデータベースを構築する。計算デバイス１１０、１４２は、ブロック４２４において、クロスバッファ依存性情報及び／又はコマンドタイプ情報（例えば、ブロック４１２－４１６において実行されたスキャニングプロセスの出力）を備えたＧＰＵコマンドをコマンドデータベースに記憶する。

【００４９】

ブロック４２６において、計算デバイス１１０、１４２は、コマンドデータベースに含

まれる情報と、利用可能なスケジューリングポリシー（例えば、リングごとのスケジューリングポリシーと集団スケジューリングポリシーとを含むもの）とに基づいて、スケジューリングポリシーを選択する。これを行うために、計算デバイス１１０、１４２は、ある条件（例えば、クロスバッファ依存性の存在または不存在、又はある数または頻度のクロスバッファ依存性の存在または不存在など）が生じたか決定し、指定の条件の発生又は不発生に基づきポリシーを選択するプログラミングロジック（例えば、ブーリアンロジック）を実行してもよい。ブロック４２８において、計算デバイス１１０、１４２は、現在のスケジューリングポリシーの変更が必要か決定する。これを行うため、計算デバイス１１０、１４２は、ブロック４３０において、ブロック４２６で選択されたポリシーを現在有効なポリシーと比較し、ポリシー変更が必要であれば、新しいポリシーを実装するのに必要な動作（例えば、必要に応じて待ち合わせプロセス）を行う（ｉｎｉｔｉａｔｅ）。ブロック４３２において、計算デバイス１１０、１４２は、現在のスケジューリングポリシーを更新してもよい。例えば、上記の通り、計算デバイス１１０、１４２は、ある時間にわたり、多数のクロスバッファ依存性が発生したことに気づいた場合、集団スケジューリングポリシーを更新し、あす閾値数のクロスバッファ依存性が検出されてからのみ、集団スケジューリングポリシーが実装されるようにしてもよい。計算デバイス１１０、１４２は、更新されたポリシーを、例えばデータストレージデバイス１１８、１６６に記憶してもよい。

【００５０】

ここで図５を参照して、ここに開示の動的に選択されたハイブリッドＧＰＵスケジューリングアプローチの実装の例５００を示した。例５００において、計算デバイス１１０、１４２は、ＶＭ１Ｂｕｆ１１（３Ｄ）バッファとＶＭ１Ｂｕｆ１２（ブリッタ）バッファとの間のクロスバッファ依存性を特定している。クロスバッファ依存性は、「ｗａｉｔ　ＣＯＮＤ１」と「ｓｉｇｎａｌ　ＣＯＮＤ１」同期プリミティブにより示されている。一例として、計算デバイス１１０、１４２は、ＶＭ１に対して集団スケジューリングポリシーを実装する。しかし、相互依存性はＶＭ１の３Ｄバッファとブリッタバッファとの間だけなので、ビデオエンジン１５４にはエンジン間依存性はなく、計算デバイス１１０、１４２はビデオエンジン１５４に対してリングごとのスケジューリングを実装する。結果として、ＶＭ２Ｂｕｆ２１とＶＭ２Ｂｕｆ２２ビデオワークロードは、ＣＰＵ１５２のビデオエンジン１５４により処理され、一方、３Ｄ及びブリッタエンジン１５４がそれぞれＶＭ１Ｂｕｆ１１とＶＭ１Ｂｕｆ１２を処理している。さらに、依存性条件が一旦削除されると、計算デバイス１１０、１４２は、３Ｄ及びブリッタエンジン１５４のリングごとのスケジューリングに戻る。結果として、ＶＭ１Ｂｕｆ１３とＶＭ１Ｂｕｆ１４は、それぞれ同時に３Ｄ及びブリッタエンジン１５４を処理できる。

＜実施例＞

【００５１】

本明細書に開示の技術の例を以下に記載する。本技術の実施形態は、以下に記載の一以上の例、及びそれらの組み合わせを含み得る。

【００５２】

実施例１は、仮想化されたグラフィックス処理ユニット（ＧＰＵ）に対するワークロードサブミッションをスケジューリングする計算デバイスを含む。本計算デバイスは、複数の仮想マシンを有する仮想化環境を確立する仮想化サービスであって、前記仮想マシンの各々は前記ＧＰＵと通信するグラフィックスドライバーと、ＧＰＵコマンドを記憶する複数のコマンドバッファとを有する、仮想化サービスを含む。また、本計算デバイスは、ＧＰＵスケジューラモジュールを含む。前記ＧＰＵスケジューラモジュールは、すべての前記仮想マシンのすべての前記コマンドバッファのＧＰＵコマンドを評価し、前記ＧＰＵコマンドの評価に応じて、複数の異なるスケジューリングポリシーからスケジューリングポリシーを動的に選択し、動的に選択されたスケジューリングポリシーにより、前記ＧＰＵにより処理する少なくとも１つのＧＰＵコマンドをスケジュールする。

【００５３】

実施例２は実施例１の主題を含み、仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンし、同じ仮想マシンの異なるコマンドバッファ中のＧＰＵコマンド間のクロスバッファ依存性を示すデータを発生し、クロスバッファ依存性を示すデータに基づきスケジューリングポリシーを動的に選択する、コマンドスキャナーモジュールを含む。

【００５４】

実施例３は実施例１または実施例２の主題を含み、仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンし、各ＧＰＵコマンドのコマンドタイプを決定し、ＧＰＵコマンドのコマンドタイプに基づいて前記ＧＰＵコマンドのスケジューリングポリシーを動的に選択する、コマンドスキャナーモジュールを含む。

【００５５】

実施例４は実施例１－３のいずれかの主題を含み、前記ＧＰＵスケジューラモジュールは、動的に選択されたスケジューリングポリシーにより前記仮想マシンのうちの１つのＧＰＵコマンドをスケジュールし、異なるスケジューリングポリシーにより前記仮想マシンのうちの他の１つのＧＰＵコマンドをスケジュールする。

【００５６】

実施例５は実施例１－４のいずれかの主題を含み、前記ＧＰＵスケジューラモジュールは、同じ仮想マシンの異なるコマンドバッファ中の２つのＧＰＵコマンド間のクロスバッファ依存性の検出に応じて、前記スケジューリングポリシーを、リングごとのスケジューリングポリシーから集団スケジューリングポリシーに切り替える。

【００５７】

実施例６は実施例５の主題を含み、前記ＧＰＵスケジューラモジュールは、異なる仮想マシンのＧＰＵコマンドを評価し、前記異なる仮想マシンのＧＰＵコマンドを集団スケジューリングポリシーでないスケジューリングポリシーによりスケジュールする。

【００５８】

実施例７は実施例６の主題を含み、前記ＧＰＵスケジューラモジュールは、リングごとのスケジューリングポリシーにより異なる仮想マシンのＧＰＵコマンドをスケジュールする。

【００５９】

実施例８は実施例１、５、又は６のいずれかの主題を含み、各コマンドバッファはリングバッファとして実施され、前記ＧＰＵスケジューラモジュールは、クロスリング同期プリミティブの仮想マシンのＧＰＵコマンドをスキャンし、前記仮想マシンのＧＰＵコマンドのためのスケジューリングポリシーを、クロスリング同期プリミティブの存在または不存在に基づき選択する。

【００６０】

実施例９は実施例１、５、又は６のいずれかの主題を含み、前記ＧＰＵスケジューラモジュールは、前記ＧＰＵコマンドの評価の出力に基づき、異なる仮想マシンの異なるスケジューリングポリシーを動的に選択する。

【００６１】

実施例１０は実施例１、５、又は６のいずれかの主題を含み、各仮想マシンのコマンドスキャナーモジュールを含み、各コマンドスキャナーモジュールは前記仮想マシンのコマンドバッファ間の依存性を示すデータを含むコマンドデータベースを生成する。

【００６２】

実施例１１は実施例１、５、又は６のいずれかの主題を含み、すべての仮想マシンのコマンドデータベースを評価し、すべてのコマンドデータベースの評価に基づいて、少なくとも１つの仮想マシンのスケジューリングポリシーを選択するアービターモジュールを有する。

【００６３】

実施例１２は実施例１、５、又は６のいずれかの主題を含み、前記ＧＰＵスケジューラモジュールは、仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンし、同じ仮想マシンの異なるコマンドバッファ中のＧＰＵコマンド間のクロスバッファ依存性

を検出し、ある時間にわたるクロスバッファ依存性の発生の頻度を決定し、クロスバッファ依存性の発生の頻度に基づいて前記スケジューリングポリシーを変更する。

【００６４】

実施例１３は実施例１、５、又は６のいずれかの主題を含み、前記ＧＰＵスケジューラモジュールは、すべての仮想マシンのすべてのコマンドバッファにわたるクロスバッファ依存性の発生をモニターし、所定数のコマンドバッファにおいてクロスバッファ依存性が検出されないとき、スケジューリングポリシーをリングごとのポリシーから集団ポリシーに変更する。

【００６５】

実施例１４は、仮想化されたグラフィックス処理ユニット（ＧＰＵ）に対するワークロードサブミッションをスケジューリングする方法である。該方法は、複数の仮想マシンを有する仮想化環境を確立するステップであって、前記仮想マシンの各々は前記ＧＰＵと通信するグラフィックスドライバーと、ＧＰＵコマンドを記憶する複数のコマンドバッファとを有する、ステップと、すべての前記仮想マシンのすべての前記コマンドバッファのＧＰＵコマンドを評価するステップと、前記ＧＰＵコマンドの評価の出力に応じて、複数の異なるスケジューリングポリシーからスケジューリングポリシーを動的に選択するステップと、動的に選択されたスケジューリングポリシーにより、前記ＧＰＵにより処理する少なくとも１つのＧＰＵコマンドをスケジュールするステップと、を含む。

【００６６】

実施例１５は実施例１４の主題を含み、仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンするステップと、同じ仮想マシンの異なるコマンドバッファ中のＧＰＵコマンド間のクロスバッファ依存性を示すデータを発生するステップと、クロスバッファ依存性を示すデータに基づきスケジューリングポリシーを動的に選択するステップとを含む。

【００６７】

実施例１６は実施例１４の主題を含み、仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンするステップと、各ＧＰＵコマンドのコマンドタイプを決定するステップと、ＧＰＵコマンドのコマンドタイプに基づいて前記ＧＰＵコマンドのスケジューリングポリシーを動的に選択するステップと、を含む。

【００６８】

実施例１７は実施例１４の主題を含み、動的に選択されたスケジューリングポリシーにより前記仮想マシンのうちの１つのＧＰＵコマンドをスケジュールするステップと、異なるスケジューリングポリシーにより前記仮想マシンのうちの他の１つのＧＰＵコマンドをスケジュールするステップとを含む。

【００６９】

実施例１８は実施例１４の主題を含み、同じ仮想マシンの異なるコマンドバッファ中の２つのＧＰＵコマンド間のクロスバッファ依存性の検出に応じて、前記スケジューリングポリシーを、リングごとのスケジューリングポリシーから集団スケジューリングポリシーに切り替えるステップを含む。

【００７０】

実施例１９は実施例１８の主題を含み、異なる仮想マシンのＧＰＵコマンドを評価するステップと、前記異なる仮想マシンのＧＰＵコマンドを、集団スケジューリングポリシーではない異なるスケジューリングポリシーによりスケジュールするステップとを含む。

【００７１】

実施例２０は実施例１９の主題を含み、リングごとのスケジューリングポリシーにより異なる仮想マシンのＧＰＵコマンドをスケジュールするステップを含む。

【００７２】

実施例２１は実施例１４の主題を含み、各コマンドバッファはリングバッファとして実施され、前記方法は、クロスリング同期プリミティブの仮想マシンのＧＰＵコマンドをスキャンするステップと、前記仮想マシンのＧＰＵコマンドのためのスケジューリングポリ

シーを、クロスリング同期プリミティブの存在または不存在に基づき選択するステップとを含む。

【００７３】

実施例２２は実施例１４の主題を含み、

異なる仮想マシンの異なるスケジューリングポリシーを、前記ＧＰＵコマンドの評価の出力に基づいて、動的に選択するステップを含む。

【００７４】

実施例２３は実施例１４の主題を含み、各仮想マシンに対して、前記仮想マシンのコマンドバッファ間の依存性を示すデータを含むコマンドデータベースを生成するステップを含む。

【００７５】

実施例２４は実施例２３の主題を含み、すべての仮想マシンのコマンドデータベースを評価するステップと、すべてのコマンドデータベースの評価に基づいて、少なくとも１つの仮想マシンのスケジューリングポリシーを選択するステップとを含む。

【００７６】

実施例２５は実施例１４の主題を含み、仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンするステップと、同じ仮想マシンの異なるコマンドバッファ中のＧＰＵコマンド間のクロスバッファ依存性を検出するステップと、ある時間にわたるクロスバッファ依存性の発生の頻度を決定するステップと、クロスバッファ依存性の発生の頻度に基づいて前記スケジューリングポリシーを変更するステップとを含む。

【００７７】

実施例２６は実施例１４の主題を含み、すべての仮想マシンのすべてのコマンドバッファにわたるクロスバッファ依存性の発生をモニターするステップと、所定数のコマンドバッファにおいてクロスバッファ依存性が検出されないとき、スケジューリングポリシーをリングごとのポリシーから集団ポリシーに変更するステップとを含む。

【００７８】

実施例２７は、計算デバイスであって、プロセッサと、前記プロセッサにより実行されたとき、前記計算デバイスに、実施例１４－２６いずれかに記載の方法を実行させる複数の命令を格納したメモリとを有する。

【００７９】

実施例２８は、実行されると、計算デバイスに実施例１４乃至２６いずれかに記載の方法を実行させる複数の命令を記憶した一以上の機械読み取り可能記憶媒体を含む。

【００８０】

実施例２９は、実施例１４－２６のいずれかに記載の方法を実行する手段を有する計算デバイスを含む。

【００８１】

実施例３０は、仮想化されたグラフィックス処理ユニット（ＧＰＵ）に対するワークロードサブミッションをスケジューリングするシステムを含む。該システムは、複数の仮想マシンを有する仮想化環境を確立する手段であって、前記仮想マシンの各々は前記ＧＰＵと通信するグラフィックスドライバーと、ＧＰＵコマンドを記憶する複数のコマンドバッファとを有する、手段と、すべての前記仮想マシンのすべての前記コマンドバッファのＧＰＵコマンドを評価する手段と、前記ＧＰＵコマンドの評価の出力に応じて、複数の異なるスケジューリングポリシーからスケジューリングポリシーを動的に選択する手段と、動的に選択されたスケジューリングポリシーにより、前記ＧＰＵにより処理する少なくとも１つのＧＰＵコマンドをスケジュールする手段と、を含む。

【００８２】

実施例３１は実施例３０の主題を含み、各仮想マシンに対して、前記仮想マシンのコマンドバッファ間の依存性を示すデータを含むコマンドデータベースを生成する手段を含む。

【００８３】

実施例３２は実施例３１の主題を含み、すべての仮想マシンのコマンドデータベースを評価する手段と、すべてのコマンドデータベースの評価に基づいて、少なくとも１つの仮想マシンのスケジューリングポリシーを選択する手段とを含む。

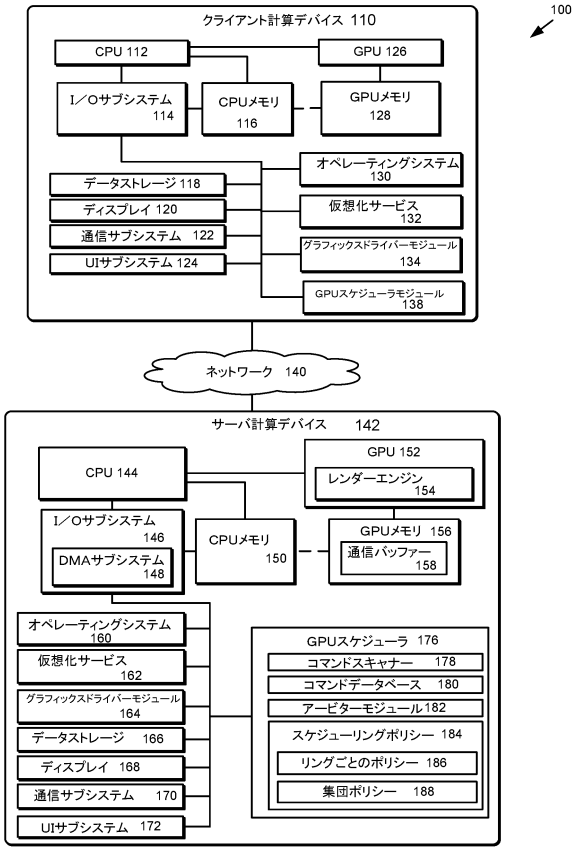
【００８４】

実施例３３は実施例３０の主題を含み、仮想マシンのすべてのコマンドバッファのＧＰＵコマンドをスキャンする手段と、同じ仮想マシンの異なるコマンドバッファ中のＧＰＵコマンド間のクロスバッファ依存性を検出する手段と、ある時間にわたるクロスバッファ依存性の発生の頻度を決定する手段と、クロスバッファ依存性の発生の頻度に基づいて前記スケジューリングポリシーを変更する手段とを含む。

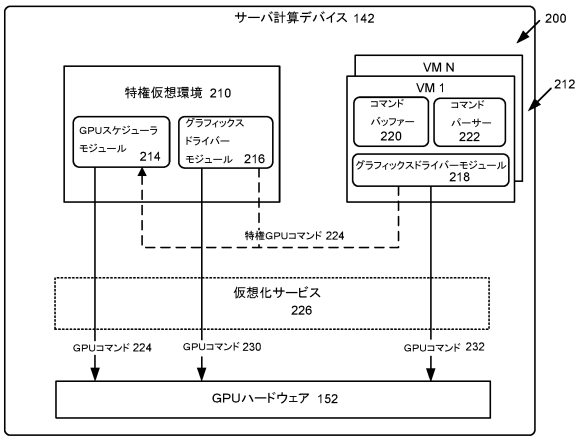
【００８５】

実施例３４は実施例３０の主題を含み、すべての仮想マシンのすべてのコマンドバッファにわたるクロスバッファ依存性の発生をモニターする手段と、所定数のコマンドバッファにおいてクロスバッファ依存性が検出されないとき、スケジューリングポリシーをリングごとのポリシーから集団ポリシーに変更する手段とを含む。

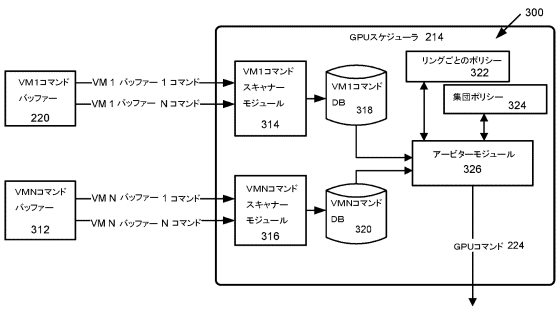
【図１】



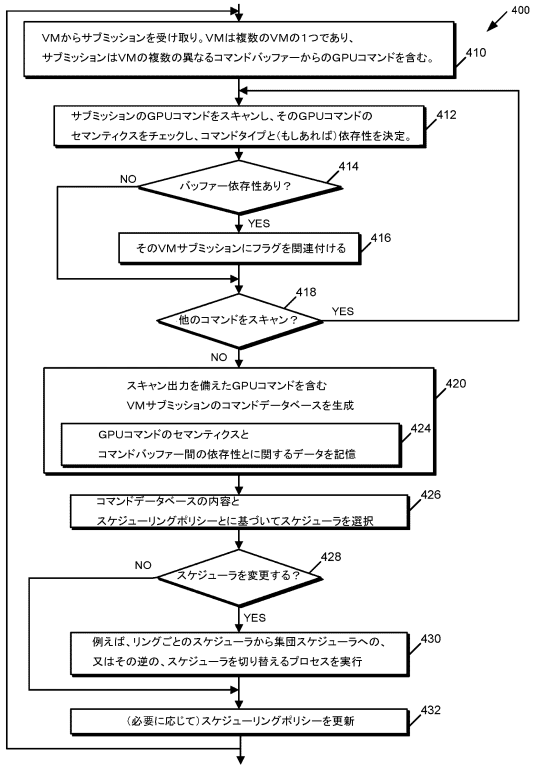
【図２】



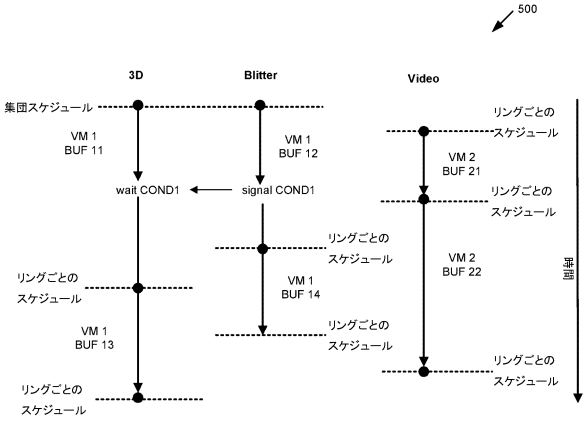
【図３】



【図４】


400
VMからサブミッションを受け取り。VMは複数のVMの１つであり、
サブミッションはVMの複数の異なるコマンドバッファーからのGPUコマンドを含む。
410
サブミッションのGPUコマンドをスキャンし、そのGPUコマンドの
セマンティクスをチェックし、コマンドタイプと（もしあれば）依存性を決定。
412
NO
バッファー依存性あり？
414
YES
そのVMサブミッションにフラグを関連付ける
416
他のコマンドをスキャン？
418
YES
NO
スキャン出力を備えたGPUコマンドを含む
VMサブミッションのコマンドデータベースを生成
420
GPUコマンドのセマンティクスと
コマンドバッファー間の依存性とに関するデータを記憶
424
コマンドデータベースの内容と
スケジューリングポリシーとに基づいてスケジューラを選択
426
NO
スケジューラを変更する？
428
YES
例えば、リングごとのスケジューラから集団スケジューラへの、
又はその逆の、スケジューラを切り替えるプロセスを実行
430
（必要に応じて）スケジューリングポリシーを更新
432

【図５】


500
3D
Blitter
Video
集団スケジュール
VM 1
BUF 11
VM 1
BUF 12
wait COND1
signal COND1
リングごとの
スケジュール
VM 2
BUF 21
VM 2
BUF 22
VM 1
BUF 14
VM 1
BUF 13
時間

フロントページの続き

(72)発明者 エルヴイ，ジーユエン
中華人民共和国 １００１９０ ペキン ハイディエン ディストリクト ジョーン グワン クン サウス ケシュエユエン ロード レイコム パート エー ８エフ ナンバー２
(72)発明者 ドーン，ヤオ ズゥ
中華人民共和国 ２０００４２ シャンハイ イエンピーン ロード １２３ ビルディング ナンバー５ アパートメント １０Ｉ（１００９）

審査官 清木 泰

(56)参考文献 特表２０１３－５４６１１１（ＪＰ，Ａ）
特表２０１３－５４６０９８（ＪＰ，Ａ）
特表２０１３－５０４１３１（ＪＰ，Ａ）
特開２００５－１０８２１４（ＪＰ，Ａ）
特開２００４－２７２８９４（ＪＰ，Ａ）
特開２００４－２７２４６６（ＪＰ，Ａ）
米国特許出願公開第２０１５／０３７１３５４（ＵＳ，Ａ１）
米国特許第０８３４１６２４（ＵＳ，Ｂ１）
米国特許出願公開第２０１４／０２５９０１６（ＵＳ，Ａ１）
米国特許第０８０４０９０１（ＵＳ，Ｂ１）
Kun Tian, Yaozu Dong, David Cowperthwaite, A Full GPU Virtualization Solution with Mediated Pass-Through, Proceedings of 2014 USENIX Annual Technical Conference (USENIX ATC '14), 米国, USENIX, ２０１４年 ６月１９日, Pages:121-132
Intel Corporation, Intel Open Source Graphics Programmer's Reference Manual (PRM) for the 2013 Intel Core Processor Family, including Intel HD Graphics, Intel Iris Graphics and Intel Iris Pro Graphics Volume 6: Command Stream Programming (Haswell), [online], ２０１４年 １月２１日, Pages:22,48,58, [平成30年10月3日検索], インターネット, ＵＲＬ, https://01.0rg/sites/default/files/documentation/intel-gfx-prm-osrc-hsw-command-stream-programming_1_0.pdf

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ ９／４５５－９／５４
Ｇ０６Ｆ ９／３８
Ｇ０６Ｔ １／２０