

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7681803号
(P7681803)

(45)発行日 令和7年5月22日(2025.5.22)

(24)登録日 令和7年5月14日(2025.5.14)

(51)国際特許分類

F I

G 0 6 F 9/50 (2006.01)

G 0 6 F 9/50 1 5 0 D

G 0 6 F 9/50 1 5 0 C

請求項の数 13 (全32頁)

(21)出願番号	特願2024-524725(P2024-524725)	(73)特許権者	517392436
(86)(22)出願日	令和4年11月17日(2022.11.17)		▲騰▼▲訊▼科技(深▲セン▼)有限公
(65)公表番号	特表2024-538875(P2024-538875 A)		司
(43)公表日	令和6年10月24日(2024.10.24)		TENCENT TECHNOLOGY
(86)国際出願番号	PCT/CN2022/132457		(SHENZHEN) COMPANY
(87)国際公開番号	WO2023/151340		LIMITED
(87)国際公開日	令和5年8月17日(2023.8.17)		中華人民共和國518057▲広▼▲東
審査請求日	令和6年4月24日(2024.4.24)		▼省深▲セン▼市南山区高新区科技中一
(31)優先権主張番号	202210135158.4		路 ▲騰▼▲訊▼大厦35▲層▼
(32)優先日	令和4年2月14日(2022.2.14)		35/F, Tencent Build
(33)優先権主張国・地域又は機関	中国(CN)		ing, Kejizhongyi Ro
			ad, Midwest District
			of Hi-tech Park, Nan
			shan District, Shen
			zhen, Guangdong 518
			最終頁に続く

(54)【発明の名称】 グラフィックスプロセッサのリソース管理方法、装置及び機器、並びにプログラム

(57)【特許請求の範囲】

【請求項1】

グラフィックスプロセッサのリソース管理機器が実行するグラフィックスプロセッサのリソース管理方法であって、

アプリケーションプロセスを処理するための複数のグラフィックスプロセッサを決定するステップと、

処理対象の複数のアプリケーションプロセスを取得し、前記複数のアプリケーションプロセスの各アプリケーションプロセスに、前記複数のグラフィックスプロセッサのうちの1つのグラフィックスプロセッサを割り当てるステップと、

1つのグラフィックスプロセッサに割り当てられる少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記グラフィックスプロセッサの現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定するステップであって、前記使用可能な残りのリソース量は、前記グラフィックスプロセッサの過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量に関連付けられる、ステップと、

前記現在の所定リソースにおける前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに使用可能な残りのリソース量に基づいて、前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスへのリソース割り当て命令を決定するステップであって、前記リソース割り当て命令は、前記アプリケーションプロセスに対して処理を行うか否かを指示するものである、ステップと、を含み、

前記リソース割り当て命令は、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにするために用いられる、グラフィックスプロセッサのリソース管理方法。

【請求項 2】

前記複数のアプリケーションプロセスの各アプリケーションプロセスは、所定のリソース需要重みを有し、

前記グラフィックスプロセッサのリソース管理方法は、

前記少なくとも 1 つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を取得し、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量を決定するステップをさらに含み、

10

前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定するステップは、

前記過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量、及び前記アプリケーションプロセスのリソース需要重みに基づいて、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定するステップを含み、

前記リソース割り当て命令は、前記アプリケーションプロセスの第 1 誤差を第 2 誤差よりも大きくするために用いられ、前記第 1 誤差は、前記アプリケーションプロセスのリソース需要重みと、前記過去の所定リソースにおける前記アプリケーションプロセスによって使用されるリソースの割合との誤差であり、前記第 2 誤差は、前記アプリケーションプロセスのリソース需要重みと、前記現在の所定リソースにおける前記アプリケーションプロセスによって使用されるリソースの割合との誤差である、請求項 1 に記載のグラフィックスプロセッサのリソース管理方法。

20

【請求項 3】

グラフィックスプロセッサの過去の所定リソースと現在の所定リソースにそれぞれ含まれるリソース量は、いずれも所定リソース量であり、

前記複数のアプリケーションプロセスの各アプリケーションプロセスのリソース需要重みは、前記所定リソース量における前記アプリケーションプロセスに必要なリソース量の割合を示すものであり、

30

前記複数のアプリケーションプロセスの各アプリケーションプロセスに、前記複数のグラフィックスプロセッサのうちの 1 つのグラフィックスプロセッサを割り当てるステップは、

前記複数のグラフィックスプロセッサの各グラフィックスプロセッサの使用可能リソース比を決定するステップであって、前記使用可能リソース比は、前記グラフィックスプロセッサにおけるアプリケーションプロセスの処理に使用可能なリソースの割合である、ステップと、

前記複数のアプリケーションプロセスの各アプリケーションプロセスのリソース需要重み、及び前記複数のグラフィックスプロセッサの各グラフィックスプロセッサの使用可能リソース比に基づいて、各アプリケーションプロセスに割り当てられるグラフィックスプロセッサを決定するステップと、を含み、

40

1 つのグラフィックスプロセッサに割り当てられる少なくとも 1 つのアプリケーションプロセスのリソース需要重みの合計は、前記グラフィックスプロセッサの使用可能リソース比以下である、請求項 2 に記載のグラフィックスプロセッサのリソース管理方法。

【請求項 4】

前記リソース割り当て命令は、対応するアプリケーションプロセスが対応するグラフィックスプロセッサに処理タスクを送信して前記グラフィックスプロセッサに処理させるか否かを指示するものであり、前記グラフィックスプロセッサによる前記処理タスクの処理は、前記アプリケーションプロセスによる前記グラフィックスプロセッサのリソースの使

50

用に対応し、

前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量を決定するステップは、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量の第1増分を決定するステップであって、前記第1増分は、前記アプリケーションプロセスに対応するグラフィックスプロセッサによる前記アプリケーションプロセスからの処理タスクの前の処理に対応する、ステップを含む、請求項2に記載のグラフィックスプロセッサのリソース管理方法。

【請求項5】

前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量の第1増分を決定するステップは、

前記グラフィックスプロセッサによる前記処理タスクの処理方式に基づいて、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量の第1増分を決定するステップであって、前記処理方式は、同期レンダリング又は非同期レンダリングのうちの少なくとも1つを含む、ステップを含む、請求項4に記載のグラフィックスプロセッサのリソース管理方法。

【請求項6】

前記グラフィックスプロセッサによる前記処理タスクの処理方式に基づいて、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量の第1増分を決定するステップは、

前記グラフィックスプロセッサによる前記処理タスクの処理方式が非同期レンダリングである場合、前記前の処理の開始と終了をマークすることにより、前記第1増分を決定するステップを含む、請求項5に記載のグラフィックスプロセッサのリソース管理方法。

【請求項7】

前記グラフィックスプロセッサによる前記処理タスクの処理方式に基づいて、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量の第1増分を決定するステップは、

前記グラフィックスプロセッサによる前記処理タスクの処理方式が同期レンダリングである場合、クエリ命令を使用することにより、前記グラフィックスプロセッサから前記第1増分を取得するステップを含む、請求項5に記載のグラフィックスプロセッサのリソース管理方法。

【請求項8】

前記現在の所定リソースにおける前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに使用可能な残りのリソース量に基づいて、前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスへのリソース割り当て命令を決定するステップは、

前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記アプリケーションプロセスに対応する現在の所定リソースにおける使用可能な残りのリソース量がゼロ以下である場合、前記アプリケーションプロセスへのリソース割り当て命令が前記アプリケーションプロセスに対して処理を行わないことを指示すると決定するステップと、

前記少なくとも1つのアプリケーションプロセスのうち、使用可能な残りのリソース量がゼロより大きい他のアプリケーションプロセスに対して、前記他のアプリケーションプロセスの各アプリケーションプロセスの優先度に基づいて、各アプリケーションプロセスへのリソース割り当て命令を決定するステップと、を含む、請求項4に記載のグラフィックスプロセッサのリソース管理方法。

【請求項9】

前記他のアプリケーションプロセスの各アプリケーションプロセスの優先度は、前記アプリケーションプロセスの処理待ち時間の長さ、及びその最新の第1増分を決定する時間的順序に関連付けられ、

前記他のアプリケーションプロセスの各アプリケーションプロセスの優先度に基づいて

10

20

30

40

50

、各アプリケーションプロセスへのリソース割り当て命令を決定するステップは、

前記他のアプリケーションプロセスの各アプリケーションプロセスに対して、処理待ち時間の長さが所定の条件を満たすアプリケーションプロセスが存在する場合、前記アプリケーションプロセスの最新の第1増分を決定する時間的順序に基づいて、前記アプリケーションプロセスへのリソース割り当て命令を決定するステップと、

処理待ち時間の長さが所定の条件を満たすアプリケーションプロセスが存在しない場合、前記他のアプリケーションプロセスの各アプリケーションプロセスの最新の第1増分を決定する時間的順序に基づいて、各アプリケーションプロセスへのリソース割り当て命令を決定するステップと、を含む、請求項8に記載のグラフィックスプロセッサのリソース管理方法。

10

【請求項10】

グラフィックスプロセッサのリソース管理機器が実行するグラフィックスプロセッサのリソース管理方法であって、

スケジューリングプロセスを起動するステップであって、前記スケジューリングプロセスは、割り当てスレッド及び複数の処理スレッドを含む、ステップと、

前記割り当てスレッドにより、アプリケーションプロセスを処理するための複数のグラフィックスプロセッサを決定し、前記複数のグラフィックスプロセッサの各グラフィックスプロセッサに1つの処理スレッドを割り当てるステップと、

複数のアプリケーションプロセスを起動するステップであって、前記複数のアプリケーションプロセスの各アプリケーションプロセスは、前記スケジューリングプロセスによって予め構築されたスケジューリングライブラリを含む、ステップと、

20

前記複数のアプリケーションプロセスの各アプリケーションプロセスに対して、前記アプリケーションプロセスのスケジューリングライブラリ及び前記割り当てスレッドにより、前記複数のアプリケーションプロセスの各アプリケーションプロセスに、前記複数のグラフィックスプロセッサのうちの1つのグラフィックスプロセッサ及び対応する処理スレッドを割り当てるステップと、

1つのグラフィックスプロセッサに割り当てられる少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記アプリケーションプロセスに対応する処理スレッドにより、前記グラフィックスプロセッサの現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定して、前記アプリケーションプロセスへのリソース割り当て命令を決定するステップであって、前記リソース割り当て命令は、前記アプリケーションプロセスに対して処理を行うか否かを指示するものである、ステップと、を含む、

30

前記使用可能な残りのリソース量は、前記グラフィックスプロセッサの過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量に関連付けられ、前記リソース割り当て命令は、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにするために用いられる、グラフィックスプロセッサのリソース管理方法。

【請求項11】

アプリケーションプロセスを処理するための複数のグラフィックスプロセッサを決定するように構成されるプロセッサ決定モジュールと、

40

処理対象の複数のアプリケーションプロセスを取得し、前記複数のアプリケーションプロセスの各アプリケーションプロセスに、前記複数のグラフィックスプロセッサのうちの1つのグラフィックスプロセッサを割り当てるように構成されるプロセッサ割り当てモジュールと、

1つのグラフィックスプロセッサに割り当てられる少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記グラフィックスプロセッサの現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定するように構成される残りリソース決定モジュールであって、前記使用可能な残りのリソース量は、前記グラフィックスプロセッサの過去の所定リソースにおける前記アプリ

50

ケーションプロセスに使用可能な残りのリソース量に関連付けられる、残りリソース決定モジュールと、

前記現在の所定リソースにおける前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに使用可能な残りのリソース量に基づいて、前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスへのリソース割り当て命令を決定するように構成されるリソース割り当てモジュールであって、前記リソース割り当て命令は、前記アプリケーションプロセスに対して処理を行うか否かを指示するものである、リソース割り当てモジュールと、を備え、

前記リソース割り当て命令は、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにするために用いられる、グラフィックスプロセッサのリソース管理装置。

10

【請求項12】

1つ又は複数のプロセッサと、

1つ又は複数のメモリと、を含むグラフィックスプロセッサのリソース管理機器であって、

前記メモリには、コンピュータ実行可能なプログラムが記憶され、前記コンピュータ実行可能なプログラムが前記プロセッサによって実行されると、請求項1～10のいずれか一項に記載のグラフィックスプロセッサのリソース管理方法が実行される、グラフィックスプロセッサのリソース管理機器。

【請求項13】

20

コンピュータプログラムであって、

コンピュータに、請求項1～10のいずれか一項に記載のグラフィックスプロセッサのリソース管理方法を実行させる、ことを特徴とするコンピュータプログラム。

【発明の詳細な説明】

【技術分野】

【0001】

本願は、2022年2月14日に中国国家知識産権局に提出された、出願番号が2022101351584で、発明の名称が「グラフィックスプロセッサのリソース管理方法、装置及び機器、並びに記憶媒体」である中国特許出願の優先権を主張するものであり、その全ての内容は、参照により本願に組み込まれるものとする。

30

【0002】

本願は、クラウド技術の分野に関し、より具体的には、グラフィックスプロセッサのリソース管理方法、装置及び機器、コンピュータ読み取り可能な記憶媒体、並びにコンピュータプログラム製品に関する。

【背景技術】

【0003】

クラウド技術の継続的な発展に伴い、クラウドゲーミングは、ゲーム業界においてますます人気が高まっている。クラウドゲーミングは、ゲームの画面をクラウドサーバ側のグラフィックスプロセッサ(Graphic Processing Unit、GPU)でレンダリングし、ネットワークを介してレンダリング結果をユーザのクライアントに伝送する。しかしながら、クラウドサーバ側のGPUリソースが限られており、クラウドゲーミングを提供するクラウドサーバでは、該クラウドサーバにロードされた複数のゲームプロセスがクラウドサーバのハードウェアコンピューティングリソースを共有する可能性があるため、ハードウェアコンピューティングリソースに対する競合が発生する。1つのGPU上で複数のゲームプロセスが同時に実行されると、これらのゲームプロセスがGPUリソースを競合するため、レンダリング効果に影響を与え、各ゲームプロセスに個別のGPUを提供すると、各ゲームプロセスのレンダリング品質を保証することができるが、必然的にGPUリソースの深刻な浪費を招く。

40

【0004】

したがって、限られたGPUリソースで、より高品質なマルチゲームプロセスのレンダ

50

リングを実現する効率的なGPUリソース管理方法が求められている。

【発明の概要】

【発明が解決しようとする課題】

【0005】

上記課題を解決するために、本願は、各タスクプロセスのリソース需要及び既に発生したリソース消費に基づいて、これらのタスクプロセスに対する処理順序をリアルタイムに決定することにより、グラフィックスプロセッサのリソースに対する効率的な割り当てを実現する。

【課題を解決するための手段】

【0006】

本願の実施形態は、グラフィックスプロセッサのリソース管理方法、装置及び機器、コンピュータ読み取り可能な記憶媒体、並びにコンピュータプログラム製品を提供する。

【0007】

本願の実施形態は、アプリケーションプロセスを処理するための複数のグラフィックスプロセッサを決定するステップと、処理対象の複数のアプリケーションプロセスを取得し、前記複数のアプリケーションプロセスの各アプリケーションプロセスに、前記複数のグラフィックスプロセッサのうちの1つのグラフィックスプロセッサを割り当てるステップと、1つのグラフィックスプロセッサに割り当てられる少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記グラフィックスプロセッサの現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定するステップであって、前記使用可能な残りのリソース量は、前記グラフィックスプロセッサの過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量に関連付けられている、ステップと、前記現在の所定リソースにおける前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに使用可能な残りのリソース量に基づいて、前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスへのリソース割り当て命令を決定するステップであって、前記リソース割り当て命令は、前記アプリケーションプロセスに対して処理を行うか否かを指示するものである、ステップと、を含み、前記リソース割り当て命令は、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにするために用いられる、グラフィックスプロセッサのリソース管理方法を提供する。

【0008】

本願の実施形態は、スケジューリングプロセスを起動するステップであって、前記スケジューリングプロセスは、割り当てスレッド及び複数の処理スレッドを含む、ステップと、前記割り当てスレッドにより、アプリケーションプロセスを処理するための複数のグラフィックスプロセッサを決定し、前記複数のグラフィックスプロセッサの各グラフィックスプロセッサに1つの処理スレッドを割り当てるステップと、複数のアプリケーションプロセスを起動するステップであって、前記複数のアプリケーションプロセスの各アプリケーションプロセスは、前記スケジューリングプロセスによって予め構築されたスケジューリングライブラリを含む、ステップと、前記複数のアプリケーションプロセスの各アプリケーションプロセスに対して、前記アプリケーションプロセスのスケジューリングライブラリ及び前記割り当てスレッドにより、前記複数のアプリケーションプロセスの各アプリケーションプロセスに、前記複数のグラフィックスプロセッサのうちの1つグラフィックスプロセッサ及び対応する処理スレッドを割り当てるステップと、1つのグラフィックスプロセッサに割り当てられる少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記アプリケーションプロセスに対応する処理スレッドにより、前記グラフィックスプロセッサの現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定して、前記アプリケーションプロセスへのリソース割り当て命令を決定するステップであって、前記リソース割り当て命令は、前記アプリケーションプロセスに対して処理を行うか否かを指示するものである、ステップと、

10

20

30

40

50

を含み、前記使用可能な残りのリソース量は、前記グラフィックスプロセッサの過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量に関連付けられており、前記リソース割り当て命令は、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにするために用いられる、グラフィックスプロセッサのリソース管理方法をさらに提供する。

【0009】

本願の実施形態は、アプリケーションプロセスを処理可能な複数のグラフィックスプロセッサを決定するように構成されるプロセッサ決定モジュールと、処理対象の複数のアプリケーションプロセスを取得し、前記複数のアプリケーションプロセスの各アプリケーションプロセスに、前記複数のグラフィックスプロセッサのうちの1つのグラフィックスプロセッサを割り当てるように構成されるプロセッサ割り当てモジュールと、1つのグラフィックスプロセッサに割り当てられる少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記グラフィックスプロセッサの現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定するように構成される残りリソース決定モジュールであって、前記使用可能な残りのリソース量は、前記グラフィックスプロセッサの過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量に関連付けられている、残りリソース決定モジュールと、前記現在の所定リソースにおける前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに使用可能な残りのリソース量に基づいて、前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスへのリソース割り当て命令を決定するように構成されるリソース割り当てモジュールであって、前記リソース割り当て命令は、前記アプリケーションプロセスに対して処理を行うか否かを指示するものである、リソース割り当てモジュールと、を備え、前記リソース割り当て命令は、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにするために用いられる、グラフィックスプロセッサのリソース管理装置を提供する。

【0010】

本願の実施形態は、1つ又は複数のプロセッサと、1つ又は複数のメモリを含むグラフィックスプロセッサのリソース管理機器であって、前記1つ又は複数のメモリには、コンピュータ実行可能なプログラムが記憶され、前記コンピュータ実行可能なプログラムが前記プロセッサによって実行されると、上記のようなグラフィックスプロセッサのリソース管理方法が実行される、グラフィックスプロセッサのリソース管理機器を提供する。

【0011】

本願の実施形態は、コンピュータ実行可能な命令が記憶されるコンピュータ読み取り可能な記憶媒体であって、プロセッサによって前記命令が実行されると、上記のようなグラフィックスプロセッサのリソース管理方法が実現される、コンピュータ読み取り可能な記憶媒体を提供する。

【0012】

本願の実施形態は、コンピュータ命令を含むコンピュータプログラム製品又はコンピュータプログラムであって、該コンピュータ命令がコンピュータ読み取り可能な記憶媒体に記憶される、コンピュータプログラム製品又はコンピュータプログラムをさらに提供する。コンピュータ機器のプロセッサは、コンピュータ読み取り可能な記憶媒体から該コンピュータ命令を読み取り、プロセッサは、該コンピュータ命令を実行して、該コンピュータ機器に本願の実施形態に係るグラフィックスプロセッサのリソース管理方法を実行させる。

【発明の効果】

【0013】

本願の実施形態に係る方法は、従来のグラフィックスプロセッサのリソース管理方法に比べて、グラフィックスプロセッサ上で実行されるアプリケーションプロセスの過去のリソース消費量をリソース割り当ての参照として、各アプリケーションプロセスの実際に使用可能なリソース量に基づいて、リソース割り当てをリアルタイムに調整することにより

10

20

30

40

50

、複数のアプリケーションプロセス間のリソース競合を回避する。

【0014】

本願の実施形態に係る方法は、同一のグラフィックスプロセッサ上で同時に実行される複数のアプリケーションプロセスに対して、過去のリソース割り当てにおけるこれらのアプリケーションプロセスに使用可能な残りのリソースを考慮し、グラフィックスプロセッサのリソースにおけるこれらのアプリケーションプロセスの現在使用可能なリソース量に基づいて、リソース割り当て方式をリアルタイムに決定することにより、グラフィックスプロセッサのリソースに対する効率的な割り当てを実現する。本願の実施形態に係る方法によれば、各アプリケーションプロセスのリソース需要に応じて、グラフィックスプロセッサのリソースを合理的に割り当てることができ、複数のアプリケーションプロセス間の競合の影響を回避し、グラフィックスプロセッサのリソースの使用率を向上させる。

10

【図面の簡単な説明】

【0015】

本願の実施形態における技術的手段をより明確に説明するために、以下、実施形態の説明に使用される必要がある図面を簡単に説明する。明らかなように、以下の説明における図面は、本願のいくつかの例示的な実施形態に過ぎず、当業者であれば、創造的な労働をせずに、これらの図面に基づいて他の図面を得ることができる。

【図1】本願の実施形態に係る複数のアプリケーションプロセスがGPUリソースを使用するシーンを示す例示的な概略図である。

【図2A】本願の実施形態に係るグラフィックスプロセッサのリソース管理方法を示すフローチャートである。

20

【図2B】本願の実施形態に係るグラフィックスプロセッサのリソース管理方法を示す概略フローチャートである。

【図2C】本願の実施形態に係るグラフィックスプロセッサのリソース管理方法を示すシーケンス概略図である。

【図3】本願の実施形態に係る複数のアプリケーションプロセスにグラフィックスプロセッサを割り当ててことを示す概略図である。

【図4A】本願の実施形態に係る2種類のリソース使用状況を示す概略図である。

【図4B】本願の実施形態に係る複数のアプリケーションプロセスのリソース割り当て割合を示す概略図である。

30

【図5】本願の実施形態に係る収集キュー及び処理キューを示す概略図である。

【図6】本願の実施形態に係る現在の所定リソースにおけるアプリケーションプロセスの使用済みリソース量の第1増分を決定することを示す例示的な概略図である。

【図7】本願の実施形態に係る第1増分の取得と処理を示す概略図である。

【図8A】本願の実施形態に係る第1増分を決定する際のCPUとGPUのダブルバッファリング方法を示す概略図である。

【図8B】本願の実施形態に係る第1増分を決定する際のCPUとGPUの適応バッファリング方法を示す概略図である。

【図9A】本願の実施形態に係るグラフィックスプロセッサのリソース管理方法を示す概略図である。

40

【図9B】本願の実施形態に係るグラフィックスプロセッサのリソース管理方法のスケジューリングロジックを示す概略図である。

【図10】本願の実施形態に係るグラフィックスプロセッサのリソース管理装置を示す概略図である。

【図11】本願の実施形態に係るグラフィックスプロセッサのリソース管理機器を示す概略図である。

【図12】本願の実施形態に係る例示的なコンピューティング機器のアーキテクチャを示す概略図である。

【図13】本願の実施形態に係る記憶媒体を示す概略図である。

【発明を実施するための形態】

50

【0016】

本願の目的、技術的手段及び利点をより明確にするために、以下、図面を参照しながら本願に係る例示的な実施形態を詳細に説明する。明らかに、説明された実施形態は、本願の一部の実施形態に過ぎず、全てではなく、本願は、ここで説明された例示的な実施形態に限定されない。

【0017】

本明細書及び図面において、実質的に同一又は類似のステップ及び要素については、同一又は類似の符号を付することにより重複する説明を省略する。また、本願の説明において、「第1」、「第2」などの用語は、説明を区別するためだけに用いられ、相対的な重要性又は順序を指示又は暗示するものとして理解されるべきではない。また、用語「複数」は、少なくとも2つであると理解されてもよい。

10

【0018】

別段の定義がない限り、本明細書で用いられる全ての技術用語及び科学用語は、本願が属する技術分野の当業者によって一般的に理解されるのと同じ意味を有する。本明細書で使用する用語は、本発明の実施形態を説明するためのものにすぎず、本発明を限定することを意図するものではない。

【0019】

本願に係るグラフィックスプロセッサのリソース管理方法は、クラウド技術（Cloud technology）に基づくことができる。

【0020】

本願に係るグラフィックスプロセッサのリソース管理方法は、クラウドゲーミング（Cloud gaming）に基づくことができる。

20

【0021】

図1は、本願の実施形態に係る複数のアプリケーションプロセスがGPUリソースを使用するシーンを示す例示的な概略図である。

【0022】

現在、多くの携帯電話アプリケーション又はコンピュータソフトウェアは、ネットワークを介してその機能を実現する必要がある、ゲームアプリケーションについてはなおさらである。ネットワークは、インターネット及び／又は電気通信網に基づくモノのインターネット（Internet of Things）であってもよく、有線ネットワークであっても無線ネットワークであってもよく、例えば、ローカルエリアネットワーク（LAN）、メトロポリタンエリアネットワーク（MAN）、広域ネットワーク（WAN）、セルラーデータ通信ネットワークなどの情報交換機能を実現できる電子ネットワークであってもよい。図1に示すように、ユーザ端末における携帯電話アプリケーション又はコンピュータソフトウェアは、ユーザによって入力される制御命令をサーバに送信して、対応するアプリケーションプロセスを起動することができる。サーバは、様々なハードウェアコンピューティングリソース、例えば、中央演算装置ユニット、通信インタフェース、メモリなどを有してもよい。図1に示すGPUリソースを例として、サーバに複数のGPU（例えば、GPU-1、GPU-2など）があり、これらのGPUのそれぞれは、異なる複数のアプリケーションプロセスに対して関連計算を行うことができる。

30

40

【0023】

任意選択で、該サーバは、独立した物理サーバであってもよく、複数の物理サーバで構成されたサーバクラスター又は分散システムであってもよく、さらに、クラウドサービス、クラウドデータベース、クラウドコンピューティング、クラウド関数、クラウドストレージ、ネットワークサービス、クラウド通信、ミドルウェアサービス、ドメインネームサービス、セキュリティサービス、CDN、並びに、ビッグデータ及び人工知能プラットフォームなどの基礎クラウドコンピューティングサービスを提供するクラウドサーバであってもよい。ユーザ端末は、スマートフォン、タブレットコンピュータ、ノートパソコン、デスクトップコンピュータ、スマートスピーカー、スマートウォッチなどであってもよいが、これらに限定されない。ユーザ端末及びサーバは、有線又は無線通信方式により直接的

50

又は間接的に接続することができ、本願において限定されない。例えば、本願の実施形態において、ネットワークに接続されるゲームアプリケーションは、一般的に、クラウドサーバ上のGPUによって、ユーザ端末に表示されるゲーム画面を合成したり、ハードウェアエンコードを行ったりし、このようなゲームアプリケーションは、クラウドゲーミングとも呼ばれる（「ゲームオンデマンド」とも呼ばれる）。ユーザ端末は、制御ストリームを介して、ユーザからのゲーム操作データをクラウドサーバに伝送することができ、クラウドサーバは、データストリームを介して、1フレーム又は複数フレームのオーディオフレーム及びビデオフレームをユーザ端末に伝送する。

【0024】

クラウドゲーミングにおいて、ゲームは、リモートクラウドサーバにおいて記憶、同期及びプレゼントされ、ストリーム技術によりプレイヤーに伝送され、従来のタイプとは全く異なるオンラインゲームサービスである。具体的には、クラウドサーバは、ゲームを実行し、そのグラフィカル出力をプレゼントしてビデオにエンコードし、そして、ビデオをネットワーククライアントにストリーミングし、クライアントは、プレイヤーがゲームとインタラクションするように、ビデオストリームをデコードして表示するとともに、プレイヤーによって入力される制御命令をクラウドサーバに送信する。このように、クラウドゲーミングは、ゲームの計算負荷をクライアントからクラウドに移行させることにより、プレイヤー機器に対する制約を解除するとともに、プレイヤーが時間をかけてゲームクライアントをダウンロードしてインストールすることなく、直ちにゲームを開始することができる。これらの利点により、クラウドゲーミングは、学术界及び産業界に大きな注目を集めている。

【0025】

以上より、クラウドサーバは、プレイヤーの入力に対する解釈、ゲームコードの実行及びグラフィックレンダリングを担当し、ネットワークを介してクライアントにゲームシーンを伝送し、クライアントは、デコードしてプレイヤーにゲームシーンを表示し、ゲームプレイヤーのゲームに対する操作をリアルタイムにキャプチャして送信し、クラウドサーバの入力とする。クラウドサーバがGPUによりグラフィックレンダリングを実行する過程において、GPUハードウェアコンピューティングリソースが限られており、クラウドゲーミングを提供するクラウドサーバでは、該クラウドサーバにおける複数の仮想エンティティがクラウドサーバのハードウェアコンピューティングリソースを共有する可能性があるため、ハードウェアコンピューティングリソースに対する競合が発生する。一般的には、GPUハードウェアのレンダリングは、その上で実行する全てのゲームプロセスからのレンダリング要求を先着順でレンダリング処理し、単一のゲームプロセスのレンダリング負荷の増加は、他のゲームプロセスの正常なレンダリングに影響を与える。例えば、いずれかのゲームプロセスのレンダリングがタイムアウトする場合、他のゲームプロセスのレンダリング時間が強制的に短縮されるため、そのレンダリング品質が高くなり、また、このようなレンダリング品質の劣化がレンダリングプロセスの進行に伴って徐々に蓄積され、ユーザのゲーム体験に深刻な影響を与える可能性がある。

【0026】

現在のGPUリソース管理方法は、GPUリソースに対する管理において、リソース割り当てを完全に分離するか、又はリソースプリエンブションを無視するが、GPUメモリとPCI-E（peripheral component interconnect express）バス規格のバスの帯域幅に対して分離を行わないため、ある瞬間においてGPUの占有率が非常に高いが、PCI-Eバスがナール状態にあり、最終的なリソース全体の使用率が高くない可能性がある。例えば、仮想化GPUの技術によるGPUリソースの分割は、GPUリソースの2分の1又は4分の1の分割粒度しか実現できず、より細かい粒度のリソース分割を行うことができず、このようなリソース管理方法は、特定のアプリケーションプロセスに対する事前設定に依存し、アプリケーションプロセスのエントリ／エグジットに応じてGPUコンピューティングリソースをリアルタイムで柔軟に割り当てることができない。

【0027】

本願は、上記に鑑みて、グラフィックスプロセッサのリソース管理方法を提供し、各タスクプロセスのリソース需要及び既に発生したリソース消費に基づいて、これらのタスクプロセスに対する処理順序をリアルタイムに決定することにより、グラフィックスプロセッサのリソースに対する効率的な割り当てを実現する。

【0028】

本願の実施形態に係る方法は、従来のグラフィックスプロセッサのリソース管理方法に比べて、グラフィックスプロセッサ上で実行するアプリケーションプロセスの過去のリソース消費量をリソース割り当ての参照として、各アプリケーションプロセスの実際に使用可能なリソース量に基づいて、リソース割り当てをリアルタイムに調整することにより、複数のアプリケーションプロセス間のリソース競合を回避する。

10

【0029】

本願の実施形態に係る方法は、同一のグラフィックスプロセッサ上で同時に実行する複数のアプリケーションプロセスに対して、過去のリソース割り当てにおけるこれらのアプリケーションプロセスに使用可能な残りのリソースを考慮し、グラフィックスプロセッサのリソースにおけるこれらのアプリケーションプロセスの現在使用可能なリソース量に基づいて、リソース割り当て方式をリアルタイムに決定することにより、グラフィックスプロセッサのリソースに対する効率的な割り当てを実現する。本願の実施形態に係る方法によれば、各アプリケーションプロセスのリソース需要に応じて、グラフィックスプロセッサのリソースを合理的に割り当てることができ、複数のアプリケーションプロセス間の競合の影響を回避し、グラフィックスプロセッサのリソースの使用率を向上させる。

20

【0030】

図2Aは、本願の実施形態に係るグラフィックスプロセッサのリソース管理方法200を示すフローチャートである。図2Bは、本願の実施形態に係るグラフィックスプロセッサのリソース管理方法を示す概略フローチャートである。図2Cは、本願の実施形態に係るグラフィックスプロセッサのリソース管理方法を示すシーケンス概略図である。

【0031】

図2Aに示すように、ステップ201では、アプリケーションプロセスを処理するための複数のグラフィックスプロセッサを決定してもよい。

【0032】

任意選択で、GPUに対するリソース管理は、図2Cに示すようなアプリケーションプロセスとスケジューリングサービスとの2つの部分で協同して行うことができ、アプリケーションプロセス部分は、GPUリソース管理プロセスにおいてアプリケーションプロセス（例えば、ゲームインスタンス）が実行する操作に対応し、スケジューリングサービス部分は、対応するアプリケーションプロセスに対するGPUリソース管理スケジューリング操作に対応する。任意選択で、該アプリケーションプロセスは、ゲームプロセス、ビデオプロセス及び会議プロセスなどの様々なアプリケーションプロセスであってもよく、本願において、限定ではなくOpenGL（Open Graphics Library、オープングラフィックスライブラリ）などのグラフィックスエンジンによって開発されたゲームプロセスを例として説明し、GPUリソーススケジューリングを必要とする任意のプロセスは、いずれも本願のGPUリソース管理方法が適用可能である。

30

40

【0033】

図2Cに示すように、アプリケーションプロセスが起動される前に、スケジューリングサービスにおける割り当てスレッドは、まず、アプリケーションプロセスの処理に現在使用可能な複数のGPUを決定し、スケジューリングサービスにおいて、これらのGPUのために、対応する処理スレッドをそれぞれ作成することができるため、その後の各GPUに対するリソース割り当て管理は、その対応する処理スレッド上で行うことができる。例えば、ゲームインスタンスなどのアプリケーションプロセスのグラフィックレンダリングタスクについて、決定されたGPUは、一定のレンダリング計算を実行する能力を有するべきである。なお、以下、多くの場合、クラウドゲーミングシーンでゲームインスタンス

50

のグラフィックレンダリングについて説明するが、本願に係るGPUリソース管理方法は、同様に他のアプリケーションプロセスの処理に適用され、例えば、ビデオプロセス及び会議プロセスの処理に適用され、ビデオ画面の画像レンダリングを処理してもよく、会議画面の画像レンダリングを処理してもよい。

【0034】

ステップ202では、処理対象の複数のアプリケーションプロセスを取得し、前記複数のアプリケーションプロセスの各アプリケーションプロセスに、前記複数のグラフィックスプロセッサのうちの1つのグラフィックスプロセッサを割り当ててもよい。

【0035】

図2Cに示すように、各ユーザ端末がアプリケーションを起動した後、サーバ側は、対応するアプリケーションプロセスをそれぞれ登録することができ、各アプリケーションプロセスにそのグラフィックレンダリング処理用のGPUを割り当て、割り当てられたGPUのインデックス及びそれに対応する処理スレッドなどの情報である登録情報をアプリケーションプロセスに返すことを含むが、これらに限定されない。

【0036】

任意選択で、アプリケーションプロセスのGPUリソース需要が決定された各GPUの使用可能なリソースを超える場合、アプリケーションプロセスのGPUリソースに対する要求を異なるGPUに転送して実行することにより、複数のGPUを1つのGPUに仮想化して、この場合のアプリケーションプロセス処理を実現することができる。

【0037】

任意選択で、GPUの計算性能の制限を満たすことを保証する場合、複数のアプリケーションプロセスを同一のGPUに割り当てることができ、即ち、同一のGPU上で複数のアプリケーションプロセスの処理タスクを同時に実行する。

【0038】

図3は、本願の実施形態に係る複数のアプリケーションプロセスにグラフィックスプロセッサを割り当ててことを示す概略図である。図3に示すように、3つのアプリケーションプロセスA、B、C、及び3つの使用可能なGPU1、2、3が存在する場合、ステップ202におけるGPUの割り当て操作を実行することにより、GPU1には、処理のためにアプリケーションプロセスA及びCが割り当てられ、GPU2には、処理のためにアプリケーションプロセスBが割り当てられ、GPU3には、いずれのアプリケーションプロセスの処理も割り当てられない。したがって、GPU1上で共同して実行されるアプリケーションプロセスA及びCについて、これらのアプリケーションプロセスは、GPU1のコンピューティングリソースを共有することができるため、これらのアプリケーションプロセスのリソース使用にも競合関係がある可能性があり、本願に係るGPUリソース管理方法は、これらのアプリケーションプロセス間の競合影響を回避することができる。

【0039】

本願の実施形態によれば、前記複数のアプリケーションプロセスの各アプリケーションプロセスは、所定のリソース需要重みを有してもよい。任意選択で、各アプリケーションプロセスのリソース需要重みは、その計算に必要なGPUリソース量に基づいて決定されてもよく、該リソース需要重みは、予め決められ、登録時にスケジューリングサービスに通知されてもよい。例えば、複数のゲームインスタンスについて、その画面レンダリングの複雑さ及び計算量に基づいて、その対応するリソース需要重みを予め決めることができ、該重みは、単位GPUリソースにおけるゲームインスタンスの画面レンダリングに必要な割合であってもよく、例えば、ゲームインスタンスの画面レンダリングが1sのGPUハードウェアコンピューティング時間において200msを必要とする場合、該ゲームインスタンスのリソース需要重みは、0.2(200ms/1s)であってもよい。

【0040】

本願の実施形態によれば、ステップ202において、前記複数のアプリケーションプロセスの各アプリケーションプロセスに、前記複数のグラフィックスプロセッサのうちの1つのグラフィックスプロセッサを割り当てることは、前記複数のグラフィックスプロセッ

10

20

30

40

50

サの各グラフィックスプロセッサの使用可能リソース比を決定するステップであって、前記使用可能リソース比は、前記グラフィックスプロセッサにおけるアプリケーションプロセスの処理に使用可能なリソースの割合である、ステップと、前記複数のアプリケーションプロセスの各アプリケーションプロセスのリソース需要重み、及び前記複数のグラフィックスプロセッサの各グラフィックスプロセッサの使用可能リソース比に基づいて、各アプリケーションプロセスに割り当てられるグラフィックスプロセッサを決定するステップと、を含んでもよい。

【0041】

以上より、該複数のGPUのそれぞれは、アプリケーションプロセスの処理に現在使用可能なGPUであるが、これらのGPUにおける使用可能なコンピューティングリソース量は、必ずしも等しくなくてもよいし、全てのコンピューティングリソース量と等しくなくてもよい。ため、特定のリソース需要を有するアプリケーションプロセスにGPUを割り当てる前に、使用可能なGPUの使用可能リソース量を決定する必要がある。上記リソース需要重みに関する説明と同様に、GPUの使用可能リソース量は、その使用可能リソース比で表すことができ、該使用可能リソース比は、GPUにおけるアプリケーションプロセスの処理に使用可能なリソース量がその単位リソースに占める比率を表すことができ、例えば、1sのGPUハードウェアコンピューティング時間において、アプリケーションプロセスの処理に使用可能な時間が0.8sである場合、該GPUの使用可能リソース比は、0.8であってもよい。

【0042】

したがって、各アプリケーションプロセスのリソース需要重み及び各GPUの使用可能リソース比の双方に基づいて、アプリケーションプロセスへのGPU割り当てを決定することができる。本願の実施形態によれば、1つのグラフィックスプロセッサに割り当てられる少なくとも1つのアプリケーションプロセスのリソース需要重みの合計は、前記グラフィックスプロセッサの使用可能リソース比以下である。処理のために1つのGPUに割り当てられるアプリケーションプロセスのリソース需要重みの合計は、該GPUの使用可能リソース比以下であり、即ち、該GPUにおけるアプリケーションプロセスの処理に実際に使用されるリソース量は、予想されるリソース量以下である。

【0043】

ステップ203では、1つのグラフィックスプロセッサに割り当てられる少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記グラフィックスプロセッサの現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定してもよく、前記使用可能な残りのリソース量は、前記グラフィックスプロセッサの過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量に関連付けられてもよい。本願の実施形態によれば、グラフィックスプロセッサの過去の所定リソースと現在の所定リソースにそれぞれ含まれるリソース量は、いずれも所定リソース量であってもよい。

【0044】

任意選択で、過去の所定リソースと現在の所定リソースにそれぞれ含まれるリソース量（即ち、所定リソース量）は、上記単位GPUリソースであってもよく、該過去の所定リソースは、現在の所定リソースの前のアプリケーションプロセスの所定リソースである。単位GPUリソースの使用量により、該GPUの使用率（GPUの実際の動作時間と実行時間との比）を決定することができるため、本願におけるGPUリソースの割り当ては、単位GPUリソースの割り当てに基づいて行うことができる。例えば、該単位GPUリソースは、上述した1s又は1フレーム時間などの単位長さの計算時間であってもよく、本願は、これを限定しない。本願の実施形態によれば、前記複数のアプリケーションプロセスの各アプリケーションプロセスのリソース需要重みは、前記所定リソース量における前記アプリケーションプロセスに必要なリソース量の割合を示してもよい。以上より、アプリケーションプロセスのリソース需要重みは、単位GPUリソースにおける該アプリケーションプロセスの処理に必要なリソースの割合であってもよい。

【0045】

本願の実施形態において、現在のリソース割り当ては、アプリケーションプロセスに対するリソース割り当ての過去の状態に基づいて決定されてもよい。該過去の状態は、過去のリソース割り当てにおける該アプリケーションプロセスの残りのリソース量、即ち、使用可能であるが未使用のリソース量を含んでもよい。該使用可能な残りのリソース量の存在は、他のアプリケーションプロセスの処理のタイムアウトにより該アプリケーションプロセスのリソースが占有されることに起因する場合があります、このようなリソース占有により、該アプリケーションプロセスのレンダリング効果が低くなる可能性があるため、このようなリソース占有の蓄積を回避するために、後続のリソース割り当てにおいて対応する調整を行うことができ、例えば、現在のリソース割り当てにおけるアプリケーションプロセスに使用可能なリソース量と、過去のリソース割り当てにおけるその使用可能な残りのリソース量とを関連付ける。

10

【0046】

本願の実施形態によれば、ステップ203において、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定することは、前記過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量、及び前記アプリケーションプロセスのリソース需要重みに基づいて、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定するステップを含んでもよい。

20

【0047】

任意選択で、アプリケーションプロセスのリソース需要重み及び過去の所定リソースにおけるその使用可能な残りのリソース量に基づいて、現在の所定リソースにおける該アプリケーションプロセスに使用可能なリソース総量を決定することができるため、その上で、現在の所定リソースにおける該アプリケーションプロセスの使用済みリソース量に基づいて、現在の所定リソースにおけるその使用可能な残りのリソース量を決定することができ、即ち、現在の所定リソースにおける使用可能なリソース総量から現在の所定リソースにおける使用済みリソース量を減算して、現在の所定リソースにおける使用可能な残りのリソース量を得ることができる。

【0048】

過去の所定リソースにおけるアプリケーションプロセスに使用可能な残りのリソース量に基づいて、現在の所定リソースにおけるその使用可能な残りのリソース量を決定することにより、過去のリソース割り当てにおける誤差に基づいて、現在のリソース割り当てを調整して、該リソース割り当ての誤差を低減し、さらに解消することができる。例えば、過去の所定リソースにおけるアプリケーションプロセスに使用可能な残りのリソース量がゼロより小さい場合、現在の所定リソースにおける該アプリケーションプロセスに使用可能な残りのリソース量から過去の所定リソースにおける該アプリケーションプロセスに使用可能な残りのリソース量の絶対値を減算することにより、該リソース割り当ての誤差を効果的に低減することができる。

30

【0049】

したがって、本願の実施形態によれば、グラフィックスプロセッサのリソース管理方法200は、前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を取得し、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量を決定するステップをさらに含んでもよい。

40

【0050】

任意選択で、現在の所定リソースにおけるアプリケーションプロセスの使用済みリソース量は、アプリケーションプロセスが前回のリソース割り当てで使用したリソース量と、現在の所定リソースに対するより早いリソース割り当てで使用したリソース量とを含んでもよく、該アプリケーションプロセスが前回のリソース割り当てで使用したリソース量は

50

、その最新の処理に使用したコンピューティングリソースに対応する。

【0051】

したがって、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量を決定するステップは、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量の第1増分を決定するステップであって、前記第1増分は、前記アプリケーションプロセスに対応するグラフィックスプロセッサによる前記アプリケーションプロセスからの処理タスクの前の処理に対応する、ステップを含んでもよい。

【0052】

任意選択で、ゲームインスタンスのレンダリングタスクに対して、該第1増分は、GPUが該ゲームインスタンスのレンダリング処理を前回実行した時間に対応してもよく、図2Cに示すように、該レンダリング時間がアプリケーションプロセスによって取得されてもよく、そこに設定されたスケジューリングライブラリによってスケジューリングサービスに通知し、それにより、スケジューリングサービスは、該レンダリング時間に基づいてレンダリング時間を処理し、現在の所定リソースにおけるアプリケーションプロセスに使用可能な残りのリソース量などを決定することを含む。

【0053】

以上より、図2Cに示すように、現在の所定リソースにおける同一のGPU上の各アプリケーションプロセスに使用可能な残りのリソース量を決定することにより、スケジューリングサービスは、現在のリソース割り当て方式を決定するとともに、レンダリングするか否かをアプリケーションプロセスに通知することができ、アプリケーションプロセスは、この間に、スケジューリングサービスからのレンダリング通知を待つ。したがって、各アプリケーションプロセスに使用可能な残りのリソース量、特に各アプリケーションプロセスの第1増分を取得することは、本願のGPUリソース管理にとって非常に重要であり、割り当ての誤差を効果的に低減することができる。第1増分の取得方式について、以下の図6及び図7に関する関連説明を参照することができ、ここでは詳述しない。

【0054】

ステップ204では、前記現在の所定リソースにおける前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに使用可能な残りのリソース量に基づいて、前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスへのリソース割り当て命令を決定してもよく、前記リソース割り当て命令は、前記アプリケーションプロセスに対して処理を行うか否かを指示するものである。

【0055】

本願の実施形態によれば、ステップ204は、前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量がゼロ以下である場合、前記アプリケーションプロセスへのリソース割り当て命令が前記アプリケーションプロセスに対して処理を行わないことを示すと決定するステップと、前記少なくとも1つのアプリケーションプロセスのうち、使用可能な残りのリソース量がゼロより大きい他のアプリケーションプロセスに対して、前記他のアプリケーションプロセスの各アプリケーションプロセスの優先度に基づいて、各アプリケーションプロセスへのリソース割り当て命令を決定するステップと、を含んでもよい。

【0056】

任意選択で、以上より、現在の所定リソースにおける使用可能な残りのリソース量がゼロ以下であるアプリケーションプロセスの場合、他のアプリケーションプロセスの処理に影響を与えないように、現在の所定リソースから、それにリソースを割り当てなくてもよい。現在の所定リソースにおける使用可能な残りのリソース量がゼロより大きいアプリケーションプロセスの場合、これらのアプリケーションプロセスにGPUコンピューティングリソースを割り当て続けることが考えられる。任意選択で、これらのアプリケーションプロセスに対するリソース割り当ては、これらのアプリケーションプロセスの優先度に基づ

10

20

30

40

50

づいて行ってもよく、上述した先着順で処理する競合モードのみに基づいて行うわけではない。

【0057】

本願の実施形態によれば、前記他のアプリケーションプロセスの各アプリケーションプロセスの優先度は、前記アプリケーションプロセスの処理待ち時間の長さ、及びその最新の第1増分を決定する時間的順序に関連付けられてもよい。例えば、画面のフリーズなどが発生しそうな処理対象のアプリケーションプロセスの場合、優先的に処理するように、その優先度を高く設定してもよく、このような非常事態が発生しない処理タイミングの場合、これらのアプリケーションプロセスの第1増分を取得した時間的順序に基づいて優先度を決定してもよく、例えば、先に第1増分を取得した、現在の所定リソースにおける使用可能な残りのリソース量がゼロより大きい、アプリケーションプロセスを優先的に処理することができる。

10

【0058】

本願の実施形態によれば、前記他のアプリケーションプロセスの各アプリケーションプロセスの優先度に基づいて、各アプリケーションプロセスへのリソース割り当て命令を決定するステップは、前記他のアプリケーションプロセスの各アプリケーションプロセスに対して、処理待ち時間の長さが所定の条件を満たすアプリケーションプロセスが存在する場合、前記アプリケーションプロセスの最新の第1増分を決定する時間的順序に基づいて、前記アプリケーションプロセスへのリソース割り当て命令を決定するステップと、処理待ち時間の長さが所定の条件を満たすアプリケーションプロセスが存在しない場合、前記他のアプリケーションプロセスの各アプリケーションプロセスの最新の第1増分を決定する時間的順序に基づいて、各アプリケーションプロセスへのリソース割り当て命令を決定するステップと、を含んでもよい。

20

【0059】

以上より、アプリケーションプロセスの優先度の設定については、次の3つの要因（これらの要因を含むが、これらに限られない）が考えられる。(1)使用可能な残りのリソース量であり、現在の所定リソースにおける使用可能な残りのリソース量がゼロ以下であるアプリケーションプロセスに対して低優先度を設定してもよい（実行しないことを指示する）。(2)緊急性であり、現在の早急に処理する必要があるアプリケーションプロセスに対して高優先度を設定してもよく、例えば、画面のフリーズ又は他の非常事態が発生する。(3)第1増分の取得順序であり、第1増分を先に取得したアプリケーションプロセスに対して高優先度を設定してもよく、GPUリソースの割り当てをより連続的にすることにより、GPUリソースの使用率を向上させる。アプリケーションプロセスの処理の優先度を設定する際に、本願の方法は、他の様々な要因を考慮してもよく、上記で挙げた要因は、ここでは、あくまで例示であり、これらに限定されるものではないことを理解されたい。

30

【0060】

本願の実施形態によれば、前記リソース割り当て命令は、対応するアプリケーションプロセスが対応するグラフィックスプロセッサに処理タスクを送信して前記グラフィックスプロセッサに処理させるか否かを指示するものであり、前記グラフィックスプロセッサによる前記処理タスクの処理は、前記アプリケーションプロセスによる前記グラフィックスプロセッサのリソースの使用に対応する。

40

【0061】

以上より、上記各種の操作ステップの実行は、従来のように、全てのアプリケーションプロセスのレンダリング命令をスケジューリングプロセスのレンダリングに必要な命令フローシステムにリダイレクトすることを必要とせず、イベントトリガー方式で行われるため、開発効率を向上させ、コストを低減することができる。

【0062】

図2Bに示すように、本願のGPUリソース管理は、各アプリケーションプロセスによるGPUハードウェアリソースのリソース使用状況を評価することにより、これらのアプリケーションプロセスによる処理タスクの送信を制御する。図2Bには、グラフィックレ

50

ンダリングタスクを例とし、GPUリソース管理は、リソース割り当て命令により各アプリケーションプロセスによるレンダリング命令の配信を制御することができる。

【0063】

任意選択で、各アプリケーションプロセスは、受信されるリソース割り当て命令に基づいて、対応するGPUに処理タスク（例えば、図2B中のGPUレンダリング命令キューへのレンダリング命令の配信）を送信するか否かを決定することができ、GPUは、一定数のGPUハードウェアリソースを消費して該処理タスクを実行し、該一定数のコンピューティングリソースは、次回のリソース割り当てで参照となる該アプリケーションプロセスの第1増分に対応することができる。

【0064】

本願の実施形態によれば、前記リソース割り当て命令は、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにすることができる。任意選択で、リソース割り当て命令により、各アプリケーションプロセスに対するリソース割り当ての誤差は、減少する傾向がある。該所定の目標値は、予め設定された使用可能な残りのリソース量が到達すべき値であり、必要に応じて設定することができ、例えば、使用可能な残りのリソース量をゼロに近づけて、リソースを節約するために、ゼロに設定してもよい。

【0065】

図4Aは、本願の実施形態に係る2種類のリソース使用状況を示す概略図である。例えば、アプリケーションプロセスに使用可能な残りのリソース量がゼロより大きい場合（例えば、図4Aに示す正常な場合）、該アプリケーションプロセスへのリソース割り当て命令により、該アプリケーションプロセスは、処理タスクを送信し、さらに、一定量のコンピューティングリソースを使用し、ゼロより大きい使用可能な残りのリソース量を減少させることができ、アプリケーションプロセスに使用可能な残りのリソース量がゼロより小さい場合（例えば、図4Aに示すタイムアウトの場合）、該アプリケーションプロセスは、現在の所定リソースにおけるいずれのリソースも使用できなくなるため、現在の所定リソースから該アプリケーションプロセスにいずれのリソースも割り当てなくなり、その使用可能な残りのリソース量が減少し続けることがなくなる。

【0066】

本願の実施形態によれば、前記リソース割り当て命令は、前記過去の所定リソースにおける前記アプリケーションプロセスによって使用されるリソースの割合よりも、前記現在の所定リソースにおける前記アプリケーションプロセスによって使用されるリソースの割合を、前記アプリケーションプロセスのリソース需要重みに近づける。即ち、該リソース割り当て命令は、前記アプリケーションプロセスの第1誤差を第2誤差よりも大きくするものであり、前記第1誤差は、前記アプリケーションプロセスのリソース需要重みと、前記過去の所定リソースにおける前記アプリケーションプロセスによって使用されるリソースの割合との誤差であり、前記第2誤差は、前記アプリケーションプロセスのリソース需要重みと、前記現在の所定リソースにおける前記アプリケーションプロセスによって使用されるリソースの割合との誤差である。

【0067】

図4Bは、本願の実施形態に係る複数のアプリケーションプロセスのリソース割り当て割合を示す概略図である。図4Bに示すように、3つのアプリケーションプロセスA、B及びCのリソース需要重みは、それぞれ0.5、0.2及び0.3である。過去の所定リソースにおいて、アプリケーションプロセスAは、その所定のリソース需要を超えるリソース量（58%）を使用するため、他の2つのアプリケーションプロセスBとCのリソースが不足する（それぞれ17%と25%であり、いずれもその所定のリソース重みに対応するリソース割合より少ない）。

【0068】

したがって、本願に係るGPUリソース管理方法によれば、過去の所定リソースにおけるアプリケーションプロセスAに使用可能な残りのリソース量がゼロより小さいため（例

10

20

30

40

50

例えば、図4Bの例では、 -0.08 である）、その過剰に使用されたリソース量（即ち、 0.08 ）に応じて、現在の所定リソースからのリソースの割り当てを減少させる。したがって、現在の所定リソースのリソース割り当て結果において、リソース割り当ての調整により、アプリケーションプロセスAに 42% の使用可能なリソース量を割り当て、リソースが占有される他のアプリケーションプロセスB及びCに対して、それぞれ占有されるリソース量に応じてリソース補償を行う（即ち、アプリケーションプロセスBに使用可能なリソース量が $20\% + (20\% - 17\%) = 23\%$ であり、アプリケーションプロセスCに使用可能なリソース量が $30\% + (30\% - 25\%) = 35\%$ である）。

【0069】

上記のように、現在の所定リソースにおけるアプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにすることにより、現在の所定リソースにおけるアプリケーションプロセスによって使用されるリソース量を所望のリソース量、即ち、そのリソース需要重みに対応するリソース需要量により近づけることができる。したがって、アプリケーションプロセスへのオンデマンドリソース割り当てを実現し、GPUリソースの効率的な使用を実現することができる。

【0070】

上記GPUハードウェアリソースのタイミング割り当てにおいては、タイミング割り当てをトリガーするために、処理スレッドに正確なタイマーをバインディングする必要があることを考慮すると、方法の複雑さを低減するために、レンダリング時間を収集するための1つの収集キュー、及びレンダリング時間を処理するための1つの処理キューを作成することにより、図2Cにおけるレンダリング時間通知及びレンダリング時間処理の操作を完了することができる。

【0071】

図5は、本願の実施形態に係る収集キュー及び処理キューを示す概略図である。図5に示すように、収集キューは、入力イベントにより複数の新しいレンダリング時間データを順次挿入し、処理キューは、この間にレンダリング時間処理を実行し、その処理が完了した後に収集キューと交替して、収集キューを新しい処理キューにし、処理キューを新しい収集キューにし、キューの交替によって時間に対するタイミング割り当てを実現する。

【0072】

レンダリング時間の収集中に、アプリケーションプロセスの異なるレンダリング方式によって、第1増分を取得する方式もそれに応じて変化する可能性がある。本願の実施形態によれば、前記第1増分を決定する方式は、前記グラフィックスプロセッサによる前記処理タスクの処理方式に基づいて決定されてもよく、前記処理方式は、同期レンダリング又は非同期レンダリングのうちの少なくとも1つを含んでもよい。

【0073】

本願の実施形態によれば、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量の第1増分を決定することは、前記前回の処理の開始と終了をマークすることにより、前記第1増分を推定すること、又は、クエリ命令を使用することにより、前記グラフィックスプロセッサから前記第1増分を取得すること、のうちの1つによって行われてもよい。

【0074】

任意選択で、非同期レンダリング方式について、第1増分は、信号でGPUハードウェアキューをインターセプトすることによって決定されてもよい。この場合、本願の実施形態によれば、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量の第1増分を決定することは、前記前回の処理の開始と終了をマークすることにより、前記第1増分を推定することによって行われてもよい。

【0075】

図6は、本願の実施形態に係る現在の所定リソースにおけるアプリケーションプロセスの使用済みリソース量の第1増分を決定することを示す例示的な概略図である。

【0076】

任意選択で、描画関数の実行時間を計算して、レンダリングの実際の所要時間を決定し、該レンダリングの実際の所要時間を現在の所定リソースにおける使用済みリソース量の第1増分としてもよい。図6に示すように、描画関数の実行時間は、準備時間と実際のレンダリング時間を含んでもよく、準備時間において、描画関数のP C I-Eチャネルによる伝送とレンダリングの準備が実行され、レンダリング命令の実行が含まれ、実際のレンダリング時間は、実際に描画命令実行部分である。したがって、描画命令の前後に命令信号Fを挿入して、該信号Fがトリガーされる時にアプリケーションプロセス（又はスケジューリングサービス）に計時開始又は計時終了を通知することができる。

【0077】

任意選択で、同期レンダリング方式について、アプリケーションスレッドローカルで第1増分をクエリにより決定することができる。この場合、本願の実施形態によれば、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量の第1増分を決定することは、クエリ命令を使用することにより、前記グラフィックスプロセスから前記第1増分を取得することによって行われてもよい。

【0078】

図7は、本願の実施形態に係る第1増分の取得と処理を示す概略図である。

【0079】

同期レンダリングの場合、クエリによりGPUにクエリ操作を送信することができ、該クエリ操作は、GPUにより2つの指定されたクエリ点間の時間を決定することができる。図7に示すように、「クエリ開始」及び「クエリ終了」は、それぞれGPUに挿入された2つの指定されたクエリ点であり、その間のレンダリング命令の描画関数の実行にかかるGPU時間、即ち、レンダリング時間をクエリし、挿入された最初の「クエリ終了」点は、それに対応する「クエリ開始」点の挿入位置を知ることができないため、当該クエリを破棄する。図7に灰色の枠で示す部分は、取得されたレンダリング時間に基づいて、GPUリソース管理処理を実行することを示し、即ち、取得されたレンダリング時間をスケジューリングサービスに通知してから、スケジューリングサービスによるレンダリング通知を待ち、スケジューリングサービスによりレンダリング時間処理を実行し、イベント通知によりアプリケーションプロセスにレンダリング通知（即ち、リソース割り当て命令）を送信する。

【0080】

上記のようなレンダリング時間のクエリ操作において、クエリ時間を待つ操作が中央演算装置（CPU）側で同期操作であるため、レンダリング時間を取得する待ち過程は、アプリケーションの性能に大きな影響を与える可能性がある。したがって、本願に係るGPUリソース管理方法は、CPUがレンダリング命令を実行する時間で、GPUがクエリ結果を準備する時間を相殺することができる。

【0081】

図8Aは、本願の実施形態に係る第1増分を決定する際のCPUとGPUのダブルバッファリング方法を示す概略図である。

【0082】

GPUにおける非交差クエリメカニズム、即ち、クエリの開始点と終了点が非交差クエリに含まなければならないこと、つまり、複数の開始点を連続的に挿入することができないことを保証するために、本実施形態において、GPUが1回遅延して計算する方式でCPUとGPUとの間のインタラクション操作を実行する。図8Aに示すように、CPUとGPUのインデックスは、それぞれの処理順序を示し、CPU2の処理は、GPU1のクエリ時間に対応し、CPU3の処理は、GPU2のクエリ時間に対応する。このようなダブルバッファリング方式により、CPUがGPUのクエリ時間だけ待つ期間のCPUの処理性能を効果的に向上させる。しかし、CPUの処理がGPUより速い可能性があり、ダブルバッファリングの方式により、依然としてCPUの待ちを完全に避けることができない可能性があるため、上記ダブルバッファリング方法に基づいて、本願に係るGPU管理方法は、適応バッファリングメカニズムに基づいて、上記問題を解決することもできる。

10

20

30

40

50

【0083】

図8Bは、本願の実施形態に係る第1増分を決定する際のCPUとGPUの適応バッファリング方法を示す概略図である。

【0084】

適応バッファリングメカニズムにおいて、依然としてダブルバッファリングメカニズムを用いることができるが、前回の描画が完了することをクエリすることができない場合、前回の描画が必ず完了すると決定するまで、今回のクエリを終了しなくてもよく、この間に複数回の描画呼び出しを行ってもよい。

【0085】

適応バッファリング方法は、ダブルバッファリング方法に対して、クエリにおける描画呼び出しの回数を拡張し、クエリの1回の描画呼び出しをランダムな複数回の呼び出しに拡張するため、複数のアプリケーションプロセスを同時に処理する場合、システム精度が低下する可能性がある。これは、異なるアプリケーションプロセス間にクロスレンダリングが存在する可能性があり、GPUが現在のクエリ点の時間を読み取るにすぎず、異なるアプリケーションプロセスを区別することができないため、クエリが不正確になるからである。

【0086】

したがって、レンダリング命令キューのコミッションをよりよく制御するために、GPUのクエリ終了時に強制的にレンダリング命令バッファエリアに対して非同期リフレッシュを行ってもよく、これにより、大量の高速描画を生成し、このような描画は、適応バッファリング方法を使用する必要がなく、GPUのレンダリング時間結果を速く取得することができるが、これも通信伝送負荷を招く。通信効率を保証しつつ、複数のアプリケーションプロセス間の相互影響を低減するように、システム性能と精度のバランスを取るために、ダブルバッファリング方法の基本アルゴリズムと適応アルゴリズムとを組み合わせるモンテカルロアルゴリズムを用いることができ、確率論的統計学的方法により（最適解に対して）より良い解を得る。描画呼び出し回数又はシステム実行時間により、適応アルゴリズムからダブルバッファリング基礎アルゴリズムへの劣化を制限することができ、これらのパラメータの低い値は、通信性能の低下を招く可能性があり、高い値は、複数のアプリケーションプロセス間の影響を増加させる可能性があるため、モンテカルロアルゴリズムを用いることにより、システムの合理性をより一層保証することができる。

【0087】

図9Aは、本願の実施形態に係るグラフィックスプロセッサのリソース管理方法300を示す概略図である。図9Aに示すように、グラフィックスプロセッサのリソース管理方法300は、スケジューリングプロセス及びアプリケーションプロセスによりそれぞれ実行される2つの操作を含んでもよい。グラフィックスプロセッサのリソース管理方法300は、主に以下のステップ(1)～(6)を含んでもよく、各ステップの番号は、図9A中の符号に対応する。

【0088】

ステップ(1)では、スケジューリングプロセスを起動し、前記スケジューリングプロセスは、割り当てスレッド及び複数の処理スレッドを含んでもよい。

【0089】

図9Aに示すように、スケジューリングプロセスは、1つの割り当てスレッド及び複数の処理スレッドを含んでもよく、割り当てスレッドは、GPUとアプリケーションプロセスの割り当て及びメッセージの配信に用いられてもよく、処理スレッドは、GPUのレンダリング時間処理に用いられてもよい。

【0090】

ステップ(2)では、前記割り当てスレッドにより、アプリケーションプロセスを処理可能な複数のグラフィックスプロセッサを決定する。

【0091】

以上より、アプリケーションプロセスの処理に現在使用可能な複数のGPUを決定して

10

20

30

40

50

、その後、各GPUに対するリソース割り当て管理を行うことができる。

【0092】

ステップ(3)では、前記複数のグラフィックスプロセッサの各グラフィックスプロセッサに1つの処理スレッドを割り当てる。

【0093】

図9Aに示すように、現在決定される3つの使用可能なGPUに対して、スケジューリングプロセスは、3つの処理スレッド1、2及び3をそれぞれ、この3つのGPUのレンダリング時間処理に割り当てることができるため、後続の各GPUのリソース割り当ては、対応する処理スレッドによって実行することができる。

【0094】

ステップ(4)では、複数のアプリケーションプロセスを起動し、前記複数のアプリケーションプロセスの各アプリケーションプロセスは、前記スケジューリングプロセスによって予め構築されたスケジューリングライブラリを含んでもよい。

【0095】

任意選択で、該スケジューリングライブラリは、該複数のアプリケーションプロセスで共有されてよく、アプリケーションプロセスとスケジューリングプロセスとの情報のやり取りは、該スケジューリングライブラリによって実現されてもよい。アプリケーションプロセスが起動した後、スケジューリングライブラリは、該アプリケーションプロセスの情報をスケジューリングプロセスに送信して登録させることができ、アプリケーションプロセスの登録操作は、同期操作である。データの逆流を低減するために、登録後にアプリケーションプロセスは、メッセージを送信し続け、スケジューリングプロセスは、イベントを共有することによりアプリケーションプロセスに対する処理を通知することができる。

【0096】

ステップ(5)では、前記複数のアプリケーションプロセスの各アプリケーションプロセスに対して、前記アプリケーションプロセスのスケジューリングライブラリ及び前記割り当てスレッドにより、前記複数のアプリケーションプロセスの各アプリケーションプロセスに、前記複数のグラフィックスプロセッサのうちの1つのグラフィックスプロセッサ及び前記グラフィックスプロセッサに対応する処理スレッドを割り当てる。

【0097】

図9Aに示すように、3つのアプリケーションプロセスにそのグラフィックレンダリング処理のためのGPUを割り当てることができるが、例えば、GPU1には、グラフィックレンダリングのためにアプリケーションプロセス1及び2が割り当てられ、GPU3には、グラフィックレンダリングのためにアプリケーションプロセス3が割り当てられ、GPU2には、この3つのアプリケーションプロセスの処理が割り当てられない。スケジューリングプロセス上でアプリケーションプロセスの登録操作を完了した後、対応する登録情報、例えば、割り当てられたGPUのインデックス及びそれに対応する処理スレッドなどの情報を各アプリケーションプロセスに返すことができる。

【0098】

ステップ(6)では、1つのグラフィックスプロセッサに割り当てられる少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記アプリケーションプロセスに対応する処理スレッドにより、前記グラフィックスプロセッサの現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定して、前記アプリケーションプロセスへのリソース割り当て命令を決定し、前記リソース割り当て命令は、前記アプリケーションプロセスに対して処理を行うか否かを指示するものである。

【0099】

上記のようなスケジューリングサービスのレンダリング処理操作は、スケジューリングプロセスにおける対応する処理スレッドによって行うことができ、例えば、GPU1に対応する処理スレッド1は、アプリケーションプロセス1及び2のレンダリング時間を処理することができるが、現在の所定リソースにおけるこの2つのアプリケーションプロセスに使

10

20

30

40

50

用可能な残りのリソース量を決定し、現在の所定リソースにおけるこの2つのアプリケーションプロセスに使用可能な残りのリソース量に基づいて、対応するリソース割り当て命令を決定することを含む。各アプリケーションプロセスは、対応するリソース割り当て命令を受信した後、前記リソース割り当て命令に基づいて、対応するGPUに処理タスクを送信してGPUに処理させるか否かを決定する。

【0100】

以上より、第1増分を取得する方式は、GPUによる処理タスクの処理方式によって異なる。例えば、非同期レンダリングの場合、第1増分は、スケジューリングライブラリがGPUから信号インターセプションにより直接推定してもよく、同期レンダリングの場合、第1増分は、スケジューリングライブラリがGPUからクエリ命令により取得してもよく、即ち、GPUからスケジューリングライブラリへの実線に沿って引き返す。

10

【0101】

本願の実施形態によれば、前記使用可能な残りのリソース量は、前記グラフィックスプロセッサの過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量に関連付けられ、前記リソース割り当て命令は、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにする。上記のように、現在の所定リソースにおけるアプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにすることにより、現在の所定リソースにおけるアプリケーションプロセスによって使用されるリソース量を所望のリソース量、即ち、そのリソース需要重みに対応するリソース需要量により近づけることができる。したがって、アプリケーションプロセスに対するオンデマンドリソース割り当てを実現し、GPUリソースの効率的な使用を実現することができる。

20

【0102】

図9Bは、本願の実施形態に係るグラフィックスプロセッサのリソース管理方法のスケジューリングロジックを示す概略図である。任意選択で、本願のグラフィックスプロセッサのリソース管理方法は、一般的なグラフィックスシステムの現在の設計に基づいて、それぞれスケジューリングクライアント(Scheduling Client)、スケジューリングサービス(Scheduling Service)及びスケジューリング(Scheduling)である3つのランタイムライブラリに関わる。そのうち、スケジューリングクライアントとスケジューリングサービスは、関数インジェクションと外部のイベントとの通信方式のみを処理し、コアロジックは、スケジューリングにある。図9Bに示すように、図9B中の様々な関数(例えば、ResourceScheduling、SchedulingProtocolなど)を呼び出して、レンダリング時間を収集して計算し、レンダリング時間に基づいてレンダリングするか否かを計算することにより、各アプリケーションプロセスへのリソース割り当て命令を決定してもよい。

30

【0103】

図10は、本願の実施形態に係るグラフィックスプロセッサのリソース管理装置1000を示す概略図である。

【0104】

前記グラフィックスプロセッサのリソース管理装置1000は、プロセッサ決定モジュール1001と、プロセッサ割り当てモジュール1002と、残りリソース決定モジュール1003と、リソース割り当てモジュール1004とを含んでもよい。

40

【0105】

本願の実施形態によれば、プロセッサ決定モジュール1001は、アプリケーションプロセスを処理可能な複数のグラフィックスプロセッサを決定するように構成されてもよい。

【0106】

任意選択で、プロセッサ決定モジュール1001は、上述したステップ201で説明した動作を実行してもよい。

【0107】

任意選択で、アプリケーションプロセスは、ゲームプロセス、ビデオプロセス、及び会

50

議プロセスなどの様々なアプリケーションプロセスであってもよく、それに対応して、ゲームインスタンスなどのアプリケーションプロセスのグラフィックレンダリングタスクについて、決定されたGPUは、一定のレンダリング計算を実行する能力を有するGPUであるべきである。

【0108】

プロセッサ割り当てモジュール1002は、処理対象の複数のアプリケーションプロセスを取得し、前記複数のアプリケーションプロセスの各アプリケーションプロセスに、前記複数のグラフィックスプロセッサのうちの1つのグラフィックスプロセッサを割り当てるように構成されてもよい。

【0109】

任意選択で、プロセッサ割り当てモジュール1002は、上述したステップ202で説明した動作を実行してもよい。

【0110】

任意選択で、各ユーザ端末がアプリケーションを起動した後、サーバ側は、対応するアプリケーションプロセスをそれぞれ登録することができ、各アプリケーションプロセスにそのグラフィックレンダリング処理用のGPUを割り当てることを含む。GPUの計算性能の制限を満たすことを保証する場合、複数のアプリケーションプロセスを同一のGPUに割り当てることができ、即ち、同一のGPU上で複数のアプリケーションプロセスの処理タスクを同時に実行するが、GPUに割り当てられる負荷は、該GPUの計算性能の制限を超えてはいけない。

【0111】

残りリソース決定モジュール1003は、1つのグラフィックスプロセッサに割り当てられる少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記グラフィックスプロセッサの現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定するように構成されてもよく、前記使用可能な残りのリソース量は、前記グラフィックスプロセッサの過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量に関連付けられる。

【0112】

任意選択で、残りリソース決定モジュール1003は、上述したステップ203で説明した動作を実行してもよい。過去の所定リソースにおけるアプリケーションプロセスに使用可能な残りのリソース量に基づいて、現在の所定リソースにおけるその使用可能な残りのリソース量を決定することにより、過去のリソース割り当てにおける誤差に基づいて、現在のリソース割り当てを調整して、該リソース割り当ての誤差を低減し、さらに解消することができる。

【0113】

リソース割り当てモジュール1004は、前記現在の所定リソースにおける前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに使用可能な残りのリソース量に基づいて、前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスへのリソース割り当て命令を決定するように構成されてもよく、前記リソース割り当て命令は、前記アプリケーションプロセスに対して処理を行うか否かを指示するものである。前記リソース割り当て命令は、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにするために用いられる。

【0114】

任意選択で、リソース割り当てモジュール1004は、上述したステップ204で説明した動作を実行してもよい。上記のように、現在の所定リソースにおけるアプリケーションプロセスに使用可能な残りのリソース量が所定の目標値に達するようにすることにより、現在の所定リソースにおけるアプリケーションプロセスによって使用されるリソース量を所望のリソース量、即ち、そのリソース需要重みに対応するリソース需要量により近づけることができる。したがって、アプリケーションプロセスに対するオンデマンドリソー

10

20

30

40

50

ス割り当てを実現し、GPUリソースの効率的な使用を実現することができる。

【0115】

一実施形態において、前記複数のアプリケーションプロセスの各アプリケーションプロセスは、所定のリソース需要重みを有し、前記装置は、

前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を取得し、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量を決定するように構成される使用済みリソース決定モジュールをさらに備え、

前記残りリソース決定モジュールは、さらに、前記過去の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量、及び前記アプリケーションプロセスのリソース需要重みに基づいて、前記現在の所定リソースにおける前記アプリケーションプロセスに使用可能な残りのリソース量を決定するように構成され、

前記リソース割り当て命令は、前記アプリケーションプロセスの第1誤差を第2誤差よりも大きくするために用いられ、前記第1誤差は、前記アプリケーションプロセスのリソース需要重みと、前記過去の所定リソースにおける前記アプリケーションプロセスによって使用されるリソースの割合との誤差であり、前記第2誤差は、前記アプリケーションプロセスのリソース需要重みと、前記現在の所定リソースにおける前記アプリケーションプロセスによって使用されるリソースの割合との誤差である。

【0116】

一実施形態において、前記リソース割り当て命令は、対応するアプリケーションプロセスが対応するグラフィックスプロセッサに処理タスクを送信して前記グラフィックスプロセッサに処理させるか否かを指示するものであり、前記グラフィックスプロセッサによる前記処理タスクの処理は、前記アプリケーションプロセスによる前記グラフィックスプロセッサのリソースの使用に対応し、

前記使用済みリソース決定モジュールは、さらに、前記現在の所定リソースにおける前記アプリケーションプロセスの使用済みリソース量の第1増分を決定するように構成され、前記第1増分は、前記アプリケーションプロセスに対応するグラフィックスプロセッサによる前記アプリケーションプロセスからの処理タスクの前回の処理に対応する。

【0117】

一実施形態において、前記リソース割り当てモジュールは、さらに、前記少なくとも1つのアプリケーションプロセスの各アプリケーションプロセスに対して、前記アプリケーションプロセスに対応する現在の所定リソースにおける使用可能な残りのリソース量がゼロ以下である場合、前記アプリケーションプロセスへのリソース割り当て命令が前記アプリケーションプロセスに対して処理を行わないことを指示すると決定し、前記少なくとも1つのアプリケーションプロセスのうち、使用可能な残りのリソース量がゼロより大きい他のアプリケーションプロセスに対して、前記他のアプリケーションプロセスの各アプリケーションプロセスの優先度に基づいて、各アプリケーションプロセスへのリソース割り当て命令を決定するように構成される。

【0118】

一実施形態において、前記他のアプリケーションプロセスの各アプリケーションプロセスの優先度は、前記アプリケーションプロセスの処理待ち時間の長さ、及びその最新の第1増分を決定する時間的順序に関連付けられ、

前記リソース割り当てモジュールは、さらに、前記他のアプリケーションプロセスの各アプリケーションプロセスに対して、処理待ち時間の長さが所定の条件を満たすアプリケーションプロセスが存在する場合、前記アプリケーションプロセスの最新の第1増分を決定する時間的順序に基づいて、前記アプリケーションプロセスへのリソース割り当て命令を決定し、処理待ち時間の長さが所定の条件を満たすアプリケーションプロセスが存在しない場合、前記他のアプリケーションプロセスの各アプリケーションプロセスの最新の第

10

20

30

40

50

1増分を決定する時間的順序に基づいて、各アプリケーションプロセスへのリソース割り当て命令を決定するように構成される。

【0119】

本願の別の態様によれば、グラフィックスプロセッサのリソース管理機器をさらに提供する。図11は、本願の実施形態に係るグラフィックスプロセッサのリソース管理機器2000の概略図を示す。

【0120】

図11に示すように、前記グラフィックスプロセッサのリソース管理機器2000は、1つ又は複数のプロセッサ2010と、1つ又は複数のメモリ2020とを含んでもよい。前記メモリ2020には、コンピュータ読み取り可能なコードが記憶され、前記コンピュータ読み取り可能なコードが前記1つ又は複数のプロセッサ2010によって実行されると、上記のようなグラフィックスプロセッサのリソース管理方法が実行されることができ

10

【0121】

本願の実施形態に係るプロセッサは、信号処理能力を有する集積回路チップであってもよい。上記プロセッサは、汎用プロセッサ、デジタル信号プロセッサ(DSP)、特定用途向け集積回路(ASIC)、フィールドプログラマブルゲートアレイ(FPGA)又は他のプログラマブルロジックデバイス、ディスクリートゲート又はトランジスタロジックデバイス、ディスクリートハードウェアコンポーネントであってもよい。本願の実施形態に開示される各方法、ステップ及びロジックブロック図を実現又は実行することができる。汎用プロセッサは、マイクロプロセッサ又は任意の一般的なプロセッサなどであってもよく、X86アーキテクチャ又はARMアーキテクチャであってもよい。

20

【0122】

一般的に、本願の様々な例示的な実施形態は、ハードウェア又は専用回路、ソフトウェア、ファームウェア、ロジック、又はそれらの任意の組み合わせで実施されてもよい。いくつかの態様は、ハードウェアで実施され、他の態様は、コントローラ、マイクロプロセッサ、又は他のコンピューティング機器によって実行され得るファームウェア又はソフトウェアで実施されてもよい。本願の実施形態の様々な態様は、ブロック図、フローチャート、又は他の何らかの画像表現を使用して例示及び説明されたが、本明細書に記載されるこれらのブロック、装置、システム、技術又は方法は、非限定的な例として、ハードウェア、ソフトウェア、ファームウェア、専用回路又はロジック、汎用ハードウェア又はコントローラ又は他のコンピューティング機器、又はそれらのいくつかの組み合わせで実施されてもよいことを理解されたい。

30

【0123】

例えば、本願の実施形態に係る方法又は装置は、図12に示すコンピューティング機器3000のアーキテクチャによって実現されてもよい。図12に示すように、コンピューティング機器3000は、バス3010と、1つ又は複数のCPU3020と、リードオンリーメモリ(ROM)3030と、ランダムアクセスメモリ(RAM)3040と、ネットワークに接続される通信ポート3050と、入力/出力コンポーネント3060と、ハードディスク3070とを含んでもよい。コンピューティング機器3000における記憶機器、例えば、ROM3030又はハードディスク3070は、本願に係るグラフィックスプロセッサのリソース管理方法の処理及び/又は通信に用いられる様々なデータ又はファイル、及びCPUにより実行されるプログラム命令を記憶することができる。コンピューティング機器3000は、ユーザインタフェース3080をさらに含んでもよい。もちろん、図11に示すアーキテクチャは、例示的なものに過ぎず、異なる機器を実現する場合、実際の必要に応じて、図12に示すコンピューティング機器における1つ又は複数のコンポーネントを省略することができる。

40

【0124】

本開示の別の態様によれば、コンピュータ読み取り可能な記憶媒体をさらに提供する。図13は、本願に係る記憶媒体の概略図4000を示す。

50

【0125】

図13に示すように、前記コンピュータ記憶媒体4020には、コンピュータ読み取り可能な命令4010が記憶される。前記コンピュータ読み取り可能な命令4010がプロセッサによって実行されると、以上の図面を参照して説明した本願の実施形態に係るグラフィックスプロセッサのリソース管理方法を実行することができる。本願の実施形態に係るコンピュータ読み取り可能な記憶媒体は、揮発性メモリ又は不揮発性メモリであってもよく、揮発性メモリ及び不揮発性メモリの両方を含んでもよい。不揮発性メモリは、リードオンリーメモリ（ROM）、プログラマブルリードオンリーメモリ（PROM）、消去可能プログラマブルリードオンリーメモリ（EPROM）、電氣的消去可能プログラマブルリードオンリーメモリ（EEPROM）又はフラッシュメモリであってもよい。揮発性メモリは、外部キャッシュとして用いられるランダムアクセスメモリ（RAM）であってもよい。制限的ではなく、例示的な説明によって、多くの形式のRAMが使用でき、例えば、スタティックランダムアクセスメモリ（SRAM）、ダイナミックランダムアクセスメモリ（DRAM）、同期ダイナミックランダムアクセスメモリ（SDRAM）、ダブルデータレート同期式ダイナミックランダムアクセスメモリ（DDRSDRAM）、強化型同期ダイナミックランダムアクセスメモリ（ESDRAM）、同期リンクダイナミックランダムアクセスメモリ（SLDRAM）及びダイレクトランバスランダムアクセスメモリ（DRRAM）が挙げられる。なお、本明細書で説明された方法のメモリは、これら及び任意の他の適切なタイプのメモリを含むが、それらに限定されないことを意図する。なお、本明細書で説明された方法のメモリは、これら及び任意の他の適切なタイプのメモリを含むが、それらに限定されないことを意図する。

10

20

【0126】

本願の実施形態によれば、コンピュータ命令を含み、該コンピュータ命令は、コンピュータ読み取り可能な記憶媒体に記憶される、コンピュータプログラム製品又はコンピュータプログラムがさらに提供される。コンピュータ機器のプロセッサは、コンピュータ読み取り可能な記憶媒体から該コンピュータ命令を読み取り、プロセッサは、該コンピュータ命令を実行して、該コンピュータ機器に本願の実施形態に係るグラフィックスプロセッサのリソース管理方法を実行させる。

【0127】

本願の実施形態は、グラフィックスプロセッサのリソース管理方法、装置、機器、コンピュータ読み取り可能な記憶媒体及びコンピュータプログラム製品を提供する。

30

【0128】

本願の実施形態に係る方法は、従来のグラフィックスプロセッサのリソース管理方法に比べて、グラフィックスプロセッサ上で実行されるアプリケーションプロセスの過去のリソース消費量をリソース割り当ての参照として、各アプリケーションプロセスの実際に使用可能なリソース量に基づいて、リソース割り当てをリアルタイムに調整することにより、複数のアプリケーションプロセス間のリソース競合を回避する。

【0129】

本願の実施形態に係る方法は、同一のグラフィックスプロセッサ上で同時に実行される複数のアプリケーションプロセスに対して、過去のリソース割り当てにおけるこれらのアプリケーションプロセスに使用可能な残りのリソースを考慮し、グラフィックスプロセッサのリソースにおけるこれらのアプリケーションプロセスの現在使用可能なリソース量に基づいて、リソース割り当て方式をリアルタイムに決定することにより、グラフィックスプロセッサのリソースに対する効率的な割り当てを実現する。本願の実施形態に係る方法によれば、各アプリケーションプロセスのリソース需要に応じて、グラフィックスプロセッサのリソースを合理的に割り当てることができ、複数のアプリケーションプロセス間の競合の影響を回避し、グラフィックスプロセッサのリソースの使用率を向上させる。

40

【0130】

なお、図面におけるフローチャート及びブロック図は、本願の様々な実施形態に係るシステム、方法及びコンピュータプログラム製品の実現可能なアーキテクチャ、機能及び操

50

作を示す。これに関して、フローチャート又はブロック図における各ブロックは、1つのモジュール、プログラムセグメント又はコードの一部を表すことができ、前記モジュール、プログラムセグメント又はコードの一部は、規定されたロジック機能を実現するための少なくとも1つの実行可能な指令を含む。なお、いくつかの代替的な実現において、ブロックで表示された機能は、図面で表示された順序と異なる順序で発生することができる。例えば、連続して示された2つのブロックは、実際には、基本的に並行して実行される場合や、逆の順序で実行される場合があり、これは、関連する機能によって決定される。ブロック図及び／又はフローチャートにおける各ブロック、及びブロック図及び／又はフローチャートにおけるブロックの組み合わせは、規定された機能又は操作を実行する専用のハードウェアに基づくシステムによって実現されてもよく、専用のハードウェアとコンピュータ命令の組み合わせによって実現されてもよいことにも留意されたい。

10

【0131】

一般的に、本願の様々な例示的な実施形態は、ハードウェア又は専用回路、ソフトウェア、ファームウェア、ロジック、又はそれらの任意の組み合わせで実施されてもよい。いくつかの態様は、ハードウェアで実施され、他の態様は、コントローラ、マイクロプロセッサ、又は他のコンピューティング機器によって実行され得るファームウェア又はソフトウェアで実施されてもよい。本願の実施形態の様々な態様は、ブロック図、フローチャート、又は他の何らかの画像表現を使用して例示及び説明されたが、本明細書に記載されるこれらのブロック、装置、システム、技術又は方法は、非限定的な例として、ハードウェア、ソフトウェア、ファームウェア、専用回路又はロジック、汎用ハードウェア又はコントローラ又は他のコンピューティング機器、又はそれらのいくつかの組み合わせで実施されてもよいことを理解されたい。

20

【0132】

以上で詳細に説明したような本願の例示的な実施形態は、例示的なものに過ぎず、限定的なものではない。当業者であれば、本願の原理及び精神から逸脱することなく、これらの実施形態又はその特徴に対して様々な修正及び組み合わせを行うことができ、このような修正が本願の範囲内にあることを理解すべきである。

30

40

50

【図面】
【図 1】

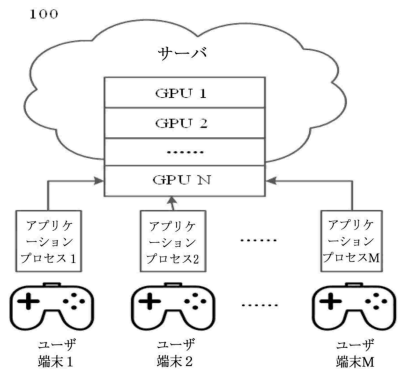


図 1

【図 2 A】

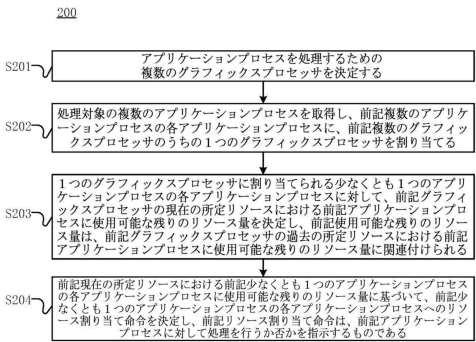


図2A

【図 2 B】

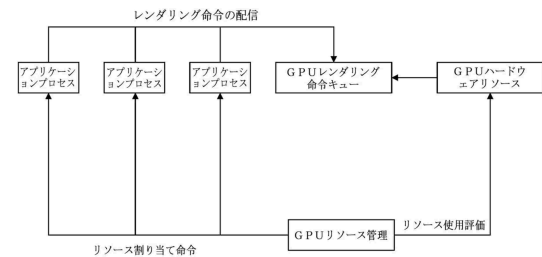


図 2B

【図 2 C】

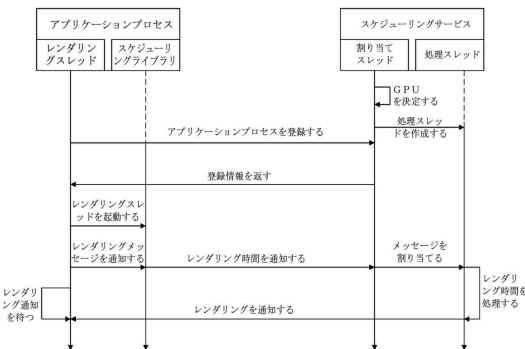


図2C

【図 3】

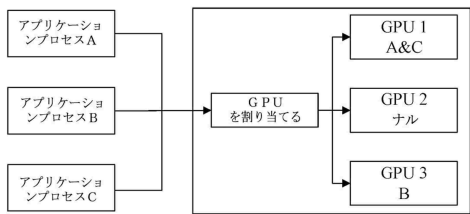


図 3

【図 4 A】

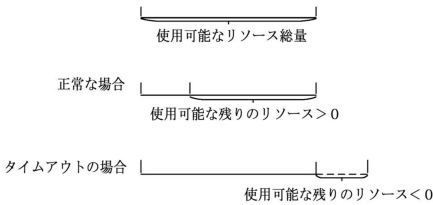


図4A

10

20

30

40

50

【図 4 B】

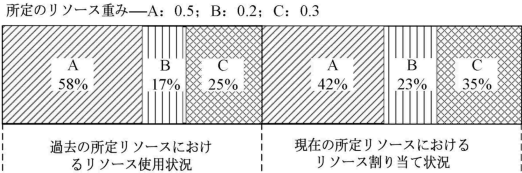
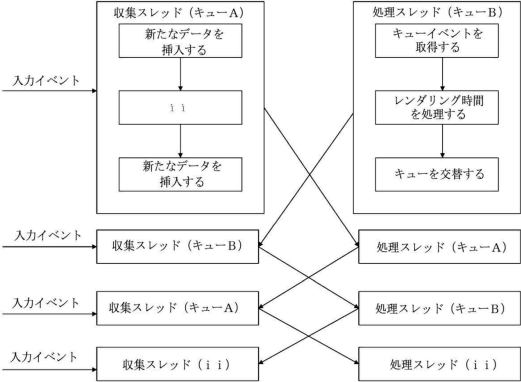
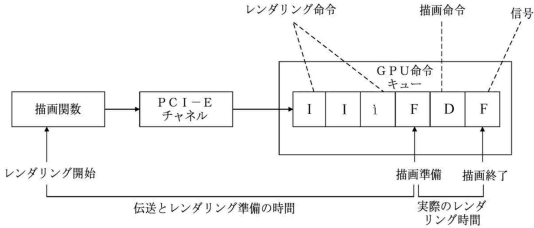


図 4B

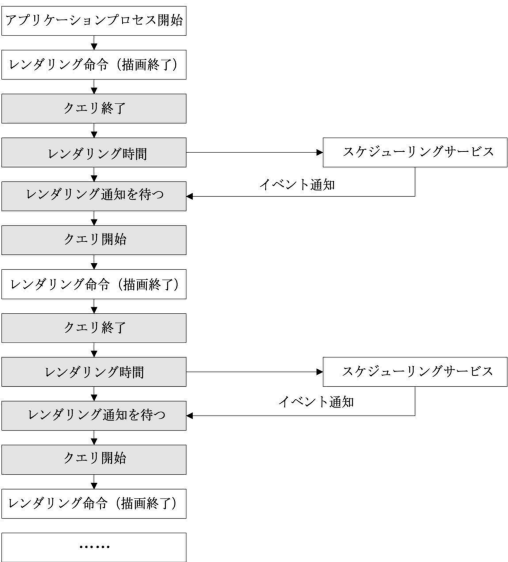
【図 5】



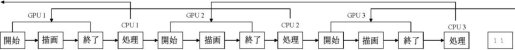
【図 6】



【図 7】



【図 8 A】



【図 8 B】



10

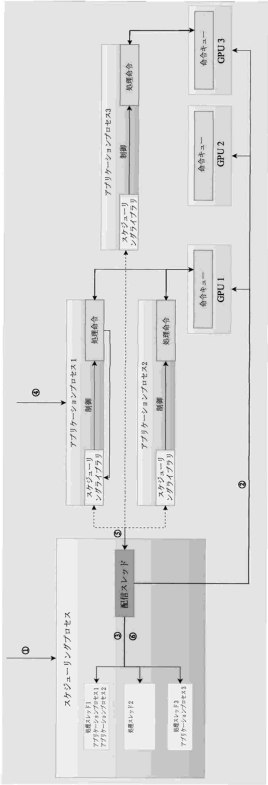
20

30

40

50

【図 9 A】



【図 9 B】

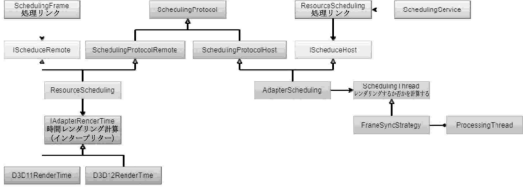


図 9B

10

20

【図 1 0】

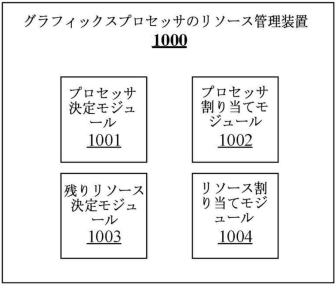


図 10

【図 1 1】

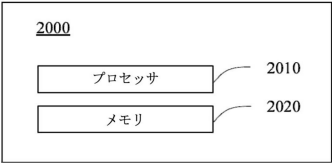


図 11

30

40

50

【図 1 2】

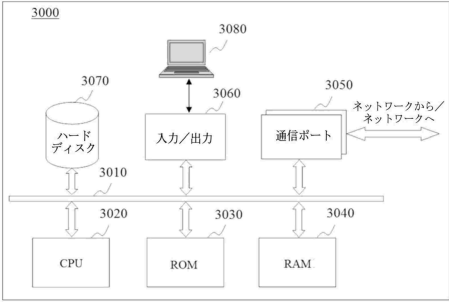


図12

【図 1 3】



図13

10

20

30

40

50

フロントページの続き

- 057, CHINA
- (74)代理人 100110364
弁理士 実広 信哉
- (74)代理人 100150197
弁理士 松尾 直樹
- (72)発明者 ▲楊▼ 坤
中華人民共和国518057▲広▼▲東▼省深▲セン▼市南山区高新区科技中一路▲騰▼▲訊▼大
厦35▲層▼
- (72)発明者 王 ▲賜▼▲ラン▼
中華人民共和国518057▲広▼▲東▼省深▲セン▼市南山区高新区科技中一路▲騰▼▲訊▼大
厦35▲層▼
- (72)発明者 唐 ▲璽▼
中華人民共和国518057▲広▼▲東▼省深▲セン▼市南山区高新区科技中一路▲騰▼▲訊▼大
厦35▲層▼
- (72)発明者 ▲楊▼ ▲広▼▲東▼
中華人民共和国518057▲広▼▲東▼省深▲セン▼市南山区高新区科技中一路▲騰▼▲訊▼大
厦35▲層▼
- 審査官 坂東 博司
- (56)参考文献 特開2019-121185 (JP, A)
特開2005-018560 (JP, A)
特開2007-011491 (JP, A)
米国特許出願公開第2008/0301673 (US, A1)
特開2010-033207 (JP, A)
- (58)調査した分野 (Int.Cl., DB名)
G06F 9/50