

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7464732号
(P7464732)

(45)発行日 令和6年4月9日(2024.4.9)

(24)登録日 令和6年4月1日(2024.4.1)

(51)国際特許分類		F I		
G 0 6 T	15/00 (2011.01)	G 0 6 T	15/00	5 0 1
A 6 3 F	13/35 (2014.01)	A 6 3 F	13/35	
A 6 3 F	13/52 (2014.01)	A 6 3 F	13/52	
請求項の数 15 (全29頁)				
(21)出願番号	特願2022-548779(P2022-548779)	(73)特許権者	517392436	
(86)(22)出願日	令和3年6月15日(2021.6.15)		▲騰▼▲訊▼科技(深▲セン▼)有限公	
(65)公表番号	特表2023-513717(P2023-513717 A)		司	
(43)公表日	令和5年4月3日(2023.4.3)		TENCENT TECHNOLOGY	
(86)国際出願番号	PCT/CN2021/100026		(SHENZHEN) COMPANY	
(87)国際公開番号	WO2022/017070		LIMITED	
(87)国際公開日	令和4年1月27日(2022.1.27)		中華人民共和國518057▲広▼▲東	
審査請求日	令和4年8月10日(2022.8.10)		▼省深▲セン▼市南山区高新区科技中一	
(31)優先権主張番号	202010704005.8		路 ▲騰▼▲訊▼大厦35▲層▼	
(32)優先日	令和2年7月21日(2020.7.21)		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】 画像処理方法及びその装置、サーバ並びにコンピュータプログラム

(57)【特許請求の範囲】

【請求項1】

画像処理方法であって、

目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、前記操作イベントに対応する目標シェーダコードを取得するステップと、

ゲームリソースから、前記目標シェーダコードの目標コンパイル結果を取得するステップであって、前記ゲームリソースは、前記目標ゲームのゲームミラーファイル、及び前記目標コンテナと前記目標コンテナの関連コンテナとの間の目標共有ディレクトリのうちの少なくとも1つを含み、前記ゲームミラーファイルは、基本的なオペレーティングシステムファイルとゲーム関連ファイルの両方を1つに合併した形式のファイルであり、前記目標共有ディレクトリは、前記関連コンテナがゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果を含む、ステップと、

前記目標コンパイル結果に基づいて図形処理を実行して、前記操作イベントに対応するフィードバック画像を取得するステップと、を含むことを特徴とする、画像処理方法。

【請求項2】

前記ゲームリソースは、前記ゲームミラーファイル及び前記目標共有ディレクトリを含み、前記ゲームミラーファイルは、前記目標ゲームを事前に実行するときにコンパイルされた少なくとも1つのシェーダコードの第2コンパイル結果を含み、各第2コンパイル結果は、前記ゲームミラーファイルを生成するときに前記ゲームミラーファイルに追加され

たものであり、

ゲームリソースから、前記目標シェーダコードの目標コンパイル結果を取得する前記ステップは、

前記目標ゲームのゲームミラーファイルにおいて、前記目標シェーダコードの第2コンパイル結果の有無を検出するステップと、

前記目標シェーダコードの第2コンパイル結果がある場合、前記ゲームミラーファイルから、前記目標シェーダコードの第2コンパイル結果を目標コンパイル結果として取得するステップと、

前記目標シェーダコードの第2コンパイル結果がない場合、前記目標コンテナと前記目標コンテナの関連コンテナとの間の目標共有ディレクトリから、前記第1コンパイル結果を前記目標シェーダコードの目標コンパイル結果として取得するステップと、を含むことを特徴とする、

10

請求項1に記載の画像処理方法。

【請求項3】

前記ゲームミラーファイルは、前記少なくとも1つのシェーダコードのインデックス値を更に含み、1つのインデックス値は1つの第2コンパイル結果に対応し、前記目標ゲームのゲームミラーファイルにおいて、前記目標シェーダコードの第2コンパイル結果の有無を検出する前記ステップは、

前記目標シェーダコードに対してハッシュ演算を実行して、前記目標シェーダコードの目標インデックス値を取得するステップと、

20

前記ゲームミラーファイルが前記目標インデックス値を含む場合、前記ゲームミラーファイルに前記目標シェーダコードの第2コンパイル結果があると決定するステップと、

前記ゲームミラーファイルが前記目標インデックス値を含まない場合、前記ゲームミラーファイルに前記目標シェーダコードの第2コンパイル結果がないと決定するステップと、を含むことを特徴とする、

請求項2に記載の画像処理方法。

【請求項4】

前記目標ゲームはクラウドゲームシステムにおけるクラウドゲームであり、前記クラウドゲームシステムは、少なくとも1つのエッジサーバ及び複数のゲームクライアントを含み、各エッジサーバには少なくとも1つのシステムコンテナが展開され、各システムコンテナは少なくとも1つのゲームクライアントに接続され、

30

前記各システムコンテナは1つ又は複数のゲームを実行するために使用され、前記各システムコンテナは、任意のゲームを実行する過程で、前記任意のゲームのゲーム画面を、前記各システムコンテナに接続されたゲームクライアントに伝送して表示し、

前記目標コンテナは、目標エッジサーバのサーバオペレーティングシステムに基づいて実行される任意のシステムコンテナであり、前記目標エッジサーバは、前記クラウドゲームシステムの任意のエッジサーバであり、前記関連コンテナは、前記目標ゲームに関連付けられた関連ゲームを実行するシステムコンテナであり、前記関連ゲームは、前記目標ゲーム、又は前記目標ゲームと同じゲームエンジンを使用するゲームを含むことを特徴とする、

40

請求項1に記載の画像処理方法。

【請求項5】

前記画像処理方法は、

前記目標エッジサーバによって送信されたディレクトリマウント命令を受信するステップであって、前記ディレクトリマウント命令は前記目標共有ディレクトリを搬送する、ステップと、

前記目標コンテナ内で動的キャッシュディレクトリを指定し、前記目標共有ディレクトリと前記動的キャッシュディレクトリとの間のデータ同期を維持するように、前記目標共有ディレクトリを前記動的キャッシュディレクトリにマウントするステップと、を更に含み、

50

前記目標共有ディレクトリは前記関連コンテナの所定のディレクトリに更にマウントされ、これによって、前記関連コンテナが前記関連ゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果をキャッシュするようにすることを特徴とする、

請求項4に記載の画像処理方法。

【請求項6】

前記動的キャッシュディレクトリは第1環境変数によって指定され、前記ゲームミラーファイルにおける第2コンパイル結果のストレージディレクトリは第2環境変数によって指定されることを特徴とする、

請求項5に記載の画像処理方法。

10

【請求項7】

前記目標共有ディレクトリは、前記サーバオペレーティングシステムが配置されているサーバホスト機器内にあることを特徴とする、

請求項4ないし6のいずれか一項に記載の画像処理方法。

【請求項8】

前記共有ディレクトリはネットワークファイルシステム内にあり、前記ネットワークファイルシステムと前記クラウドゲームシステムの各エッジサーバは、相互に独立し且つ前記各エッジサーバのシステムコンテナからのアクセスを許可することを特徴とする、

請求項4ないし6のいずれか一項に記載の画像処理方法。

【請求項9】

前記サーバホスト機器は、前記目標共有ディレクトリ以外の他の共有ディレクトリを更に含み、前記サーバホスト機器の他の共有ディレクトリは、前記目標コンテナに関連しない、前記目標エッジサーバのシステムコンテナをマウントするために使用されることを特徴とする、

請求項7に記載の画像処理方法。

20

【請求項10】

前記ネットワークファイルシステムは、前記目標共有ディレクトリ以外の他の共有ディレクトリを更に含み、前記ネットワークファイルシステムの他の共有ディレクトリは、前記目標コンテナに関連しない、前記クラウドゲームシステムのシステムコンテナをマウントするために使用されることを特徴とする、

請求項8に記載の画像処理方法。

30

【請求項11】

前記ゲームリソースは前記目標シェーダコードの目標コンパイル結果を含まず、前記画像処理方法は、

前記ゲームリソースからの前記目標コンパイル結果の取得に失敗した場合、前記目標シェーダコードをコンパイルして、前記目標コンパイル結果を取得するステップと、

前記目標コンパイル結果を前記目標共有ディレクトリに同期させるために、前記目標コンパイル結果を前記動的キャッシュディレクトリにキャッシュし、これにより、前記関連コンテナが、前記目標シェーダコードをコンパイルする必要がある場合、前記目標共有ディレクトリから前記目標コンパイル結果を取得するようにする、ステップと、を更に含むことを特徴とする、

請求項5に記載の画像処理方法。

40

【請求項12】

前記操作イベントは、前記目標コンテナに接続された目標ゲームクライアントに表示される前記目標ゲームのゲーム画面から検出されたものであり、前記操作イベントに対応するフィードバック画像を取得した後、前記画像処理方法は、

前記目標ゲームクライアントに前記フィードバック画像を表示するステップを更に含むことを特徴とする、

請求項1に記載の画像処理方法。

【請求項13】

50

画像処理装置であって、

取得ユニットと、処理ユニットと、を備え、

前記取得ユニットは、目標コンテナ内で実行される目標ゲームに対する操作イベントに
応答して、前記操作イベントに対応する目標シェーダコードを取得するように構成され、

前記取得ユニットは更に、ゲームリソースから、前記目標シェーダコードの目標コンパ
イル結果を取得するように構成され、前記ゲームリソースは、前記目標ゲームのゲームミ
ラーファイル、及び前記目標コンテナと前記目標コンテナの関連コンテナとの間の目標共
有ディレクトリのうちの少なくとも1つを含み、前記ゲームミラーファイルは、基本的な
オペレーティングシステムファイルとゲーム関連ファイルの両方を1つに合わせた形式の
ファイルであり、前記目標共有ディレクトリは、前記関連コンテナがゲームを実行する過
程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパ
イル結果を含み、

10

前記処理ユニットは、前記目標コンパイル結果に基づいて図形処理を実行して、前記操
作イベントに対応するフィードバック画像を取得するように構成されることを特徴とする
、画像処理装置。

【請求項14】

サーバであって、入力インターフェース及び出力インターフェースを備え、前記サーバ
は更に、

1つ又は複数の命令を記憶するように構成されるコンピュータ記憶媒体と、

前記コンピュータ記憶媒体に記憶された1つ又は複数の命令をロードして、請求項1な
いし12のいずれか一項に記載の画像処理方法を実行するように構成されるプロセッサと
、を備えることを特徴とする、サーバ。

20

【請求項15】

コンピュータに、請求項1ないし12のいずれか一項に記載の画像処理方法を実行させ
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】

【関連出願の相互参照】

本願は、2020年7月21日に中国特許局に提出された、出願番号が第202010
704005.8号であり、発明の名称が「画像処理方法、装置、サーバ並びに媒体」で
ある中国特許出願の優先権を主張し、その内容の全てが引用により本願に援用される。

30

【0002】

【技術分野】

本願は、インターネット技術分野、具体的には、画像処理技術分野に関するものであり
、特に、画像処理方法、画像処理装置、サーバ及びコンピュータ記憶媒体に関するもので
ある。

【背景技術】

【0003】

インターネット技術の発展に伴い、クラウドゲーム（Cloud gaming）は幅
広く注目されている。いわゆるクラウドゲームは、クラウドコンピューティングに基づく
ゲームであり、オンデマンドゲーム（gaming on demand）とも呼ばれ、こ
こでのクラウドコンピューティングは、インターネットに基づくコンピューティング方式
である。クラウドゲーム技術により、図形処理とデータ演算能力が比較的制限されたク
ライアント機器で高品質のゲームを実行できるようになる。クラウドゲームのシナリオで
は、ゲームは、プレーヤユーザのゲームクライアントで実行されるのではなく、クラウド
サーバで実行され、クラウドサーバによって、ゲームシーンをビデオとオーディオのスト
リームにレンダリングして、ネットワークを介してプレーヤユーザのゲームクライアント
に伝送して再生する。プレーヤユーザのゲームクライアントは、強力な図形演算及びデー
タ処理能力を必要とせずに、基本的なストリーミングメディア再生機能と、プレーヤユー

40

50

ザの入力命令を取得してクラウドサーバに送信する能力を備えるだけでよい。

【0004】

クラウドゲームの場合、ゲームの実行にかかる時間は、クラウドゲームの実行速度を測定するための重要な指標である。ゲームの実行にかかる時間が長いほど、クラウドゲームの実行速度が低下し、ゲームの実行にかかる時間が短いほど、クラウドゲームの実行速度が向上する。これに基づき、クラウドゲームの実行速度を向上させるために、如何にクラウドゲームの実行にかかる時間を短縮するかが、注目される研究課題となっている。

【発明の概要】

【課題を解決するための手段】

【0005】

本願実施例は、目標シェーダコードのコンパイル時間を効果的に短縮することで、目標ゲームの実行速度を効果的に向上させることができる、画像処理方法及びその装置、サーバ並びに媒体を提供する。

【0006】

1つの態様によれば、本願実施例は画像処理方法を提供し、当該画像処理方法は、目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、前記操作イベントに対応する目標シェーダコードを取得するステップと、

ゲームリソースから、前記目標シェーダコードの目標コンパイル結果を取得するステップであって、前記ゲームリソースは、前記目標ゲームのゲーム鏡像、及び前記目標コンテナと前記目標コンテナの関連コンテナとの間の目標共有ディレクトリのうちの少なくとも1つを含み、前記目標共有ディレクトリは、前記関連コンテナがゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果を含む、ステップと、

前記目標コンパイル結果に基づいて図形処理を実行して、前記操作イベントに対応するフィードバック画像を取得するステップと、を含む。

【0007】

別の態様によれば、本願実施例は画像処理装置を提供し、当該画像処理装置は、取得ユニットと、処理ユニットと、を備え、

前記取得ユニットは、目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、前記操作イベントに対応する目標シェーダコードを取得するように構成され、

前記取得ユニットは更に、ゲームリソースから、前記目標シェーダコードの目標コンパイル結果を取得するように構成され、前記ゲームリソースは、前記目標ゲームのゲーム鏡像、及び前記目標コンテナと前記目標コンテナの関連コンテナとの間の目標共有ディレクトリのうちの少なくとも1つを含み、前記目標共有ディレクトリは、前記関連コンテナがゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果を含む、

前記処理ユニットは、前記目標コンパイル結果に基づいて図形処理を実行して、前記操作イベントに対応するフィードバック画像を取得するように構成される。

【0008】

更に別の態様によれば、本願実施例はサーバを提供し、前記サーバは、入力インターフェース及び出力インターフェースを備え、前記サーバは更に、

1つ又は複数の命令を記憶するように構成されるコンピュータ記憶媒体と、前記コンピュータ記憶媒体に記憶された1つ又は複数の命令をロードすることにより、以下のステップを実行するように構成されるプロセッサと、を備え、前記ステップは、

目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、前記操作イベントに対応する目標シェーダコードを取得するステップと、

ゲームリソースから、前記目標シェーダコードの目標コンパイル結果を取得するステップであって、前記ゲームリソースは、前記目標ゲームのゲーム鏡像、及び前記目標コンテナと前記目標コンテナの関連コンテナとの間の目標共有ディレクトリのうちの少なくとも1つを含み、前記目標共有ディレクトリは、前記関連コンテナがゲームを実行する過程で

10

20

30

40

50

少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果を含む、ステップと、

前記目標コンパイル結果に基づいて図形処理を実行して、前記操作イベントに対応するフィードバック画像を取得するステップと、を含む。

【0009】

更に別の態様によれば、本願実施例はコンピュータ記憶媒体を提供し、前記コンピュータ記憶媒体には1つ又は複数の命令が記憶されており、前記1つ又は複数の命令は、プロセッサに、以下のステップを実行させ、前記ステップは、

目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、前記操作イベントに対応する目標シェーダコードを取得するステップと、

ゲームリソースから、前記目標シェーダコードの目標コンパイル結果を取得するステップであって、前記ゲームリソースは、前記目標ゲームのゲーム鏡像、及び前記目標コンテナと前記目標コンテナの関連コンテナとの間の目標共有ディレクトリのうちの少なくとも1つを含み、前記目標共有ディレクトリは、前記関連コンテナがゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果を含む、ステップと、

前記目標コンパイル結果に基づいて図形処理を実行して、前記操作イベントに対応するフィードバック画像を取得するステップと、を含む。

【図面の簡単な説明】

【0010】

【図1a】本願実施例に係るクラウドゲームシステムのシステムアーキテクチャの概略図である。

【図1b】本願実施例に係る、エッジサーバと複数のゲームクライアントとの間の接続を示す概略図である。

【図1c】本願明実施例に係る、ゲーム画面をゲームクライアントに伝送して表示することを示す概略図である。

【図2】本願実施例に係る画像処理方法の例示的なフローチャートである。

【図3a】本願実施例に係る、ゲーム鏡像を作成することを示す概略図である。

【図3b】本願実施例に係る、目標共有ディレクトリをマウントすることを示す概略図である。

【図3c】本願実施例に係る、目標共有ディレクトリをマウントすることを示す別の概略図である。

【図3d】本願実施例に係る、目標共有ディレクトリをマウントすることを示す別の概略図である。

【図3e】本願実施例に係る、目標共有ディレクトリをマウントすることを示す別の概略図である。

【図4】本願の別の実施例に係る画像処理方法の例示的なフローチャートである。

【図5a】本願の別の実施例に係る、目標共有ディレクトリをマウントすることを示す概略図である。

【図5b】本願の別の実施例に係る、目標共有ディレクトリをマウントすることを示す別の概略図である。

【図5c】本願の別の実施例に係る、目標共有ディレクトリをマウントすることを示す別の概略図である。

【図5d】本願の別の実施例に係る、目標共有ディレクトリをマウントすることを示す別の概略図である。

【図5e】本願実施例に係る、目標コンパイル結果をキャッシュすることを示す概略図である。

【図6】本願実施例に係る画像処理装置の概略構造図である。

【図7】本願実施例に係るサーバの概略構造図である。

【発明を実施するための形態】

10

20

30

40

50

【0011】

以上では、本願実施例の技術的解決策をより明確に説明するための、本願実施例の説明で使用される図面について簡単に紹介したが、当然ながら、以上で説明された図面は、本願のいくつかの実施例に過ぎず、当業者にとっては、創造的な労力なしに、これらの図面から他の図面を取得することもできる。

【0012】

以下、本願実施例における図面を参照して、本願実施例における技術的解決策を明確かつ完全に説明する。

【0013】

クラウドゲームをより良く実行するために、本願実施例は、クラウドゲーム環境を提案する。当該クラウドゲーム環境では、システムコンテナを実行することにより、1つの独立したサーバ（例えば、ARMアーキテクチャ／x86アーキテクチャを使用するサーバ）上で複数の操作システムを実行し、関連画像をビデオストリームの形でリモート受信プログラムに転送して処理することができる。ここで、ARMアーキテクチャは、32ビット／64ビットの簡易命令セットのプロセッサアーキテクチャであり、x86アーキテクチャ（The X86 architecture）は、マイクロプロセッサによって実行されるコンピュータ言語の命令セットである。コンテナ（container）とは、操作システムレベルの仮想化の一種であり、コンテナは、操作システムを搭載するために使用され、コンテナは分離メカニズム（例えば、namespace（名前空間））によって実現でき、カーネルモードでは、複数の操作システム（つまり、サーバ操作システム及び機器操作システム）は同じカーネルを使用し、ユーザモードでは、複数の操作システムは互いに独立している。ここでのサーバ操作システムとは、サーバ内の汎用操作システム（例えば、Linux（登録商標）操作システムなど）を指し、サーバ操作システムが配置されているホスト機器はサーバホスト機器（すなわち、Host機器）と呼ばれる。機器操作システムとは、コンテナ内の操作システム（例えば、Android（登録商標）操作システム、iOS（登録商標）操作システムなど）を指す。

【0014】

これに対応して、システムコンテナはコンテナの1つの具現例であり、システムコンテナは、サーバ操作システム（例えば、Linux操作システム）に基づいて実行できる。例えば、当該システムコンテナは、オープンソースのLinux操作システムで実行されるAndroidコンテナであってもよく、1つのLinux操作システムは、複数のAndroidコンテナを同時に実行することができる。AndroidコンテナはAndroidゲーム鏡像をロードし、いわゆるAndroidゲーム鏡像とは、基本的なAndroid関連ファイルとゲーム関連ファイルの両方を含む鏡像ファイルを指し、いわゆる鏡像（ミラー）とは、ファイルストレージ形式の一種であり、鏡像により複数のファイルを1つの鏡像ファイルに合併することで、ファイルの配布と使用が容易になる。理解されたいこととして、本願実施例で言及されているシステムコンテナはAndroidコンテナに限定されず、例えば、iOS操作システムがオープンソースの研究開発をサポートする場合、当該システムコンテナはiOSコンテナであってもよい。ここから分かるように、本願実施例で提案されるクラウドゲーム環境では、1つの独立したサーバに大量のシステムコンテナを展開することにより、サーバの強力な中央処理装置（Central Processing Unit、CPUと略称）能力及び図形プロセッサ（Graphics Processing Unit、GPUと略称）能力を最大限に活用して、システム操作の高度な同時実行を実現し、クラウドゲームの実行速度を向上させることができる。

【0015】

上記のクラウドゲーム環境に基づき、本願実施例はクラウドゲームシステムを提案する。図1aを参照すると、当該クラウドゲームシステムは、少なくとも1つのエッジサーバ11及び複数のゲームクライアント12を備えることができる。ここで、エッジサーバ11とは、システムコンテナ実行用のサーバを指し、当該サーバは任意の独立した物理サー

10

20

30

40

50

バであってもよいし、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、コンテンツデリバリネットワーク（Content Delivery Network、CDNと略称）、及びビッグデータ及び人工知能プラットフォームなどの基本的な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。図1bに示されるように、各エッジサーバ11には少なくとも1つのシステムコンテナが展開でき、各システムコンテナは、少なくとも1つのゲームクライアント12に接続されることができる。ここで、各システムコンテナは、1つ又は複数のゲームを実行するために使用されることができ、各システムコンテナは、任意のゲームを実行する過程で、任意のゲームのゲーム画面を、前記各システムコンテナ12に接続されたゲームクライアントに伝送して表示することができる。ゲームクライアント12は、基本的なストリーミングメディア再生能力、ヒューマン・マシン・インタラクション（Human-Machine Interaction）能力及び通信能力などの基本的な能力を有する任意の端末機器（端末と略称する）であってもよく、例えば、スマートフォン、タブレットコンピュータ、ノートブックコンピュータ、デスクトップコンピュータ、スマートTVなどであってもよい。或いは、当該ゲームクライアント12は、端末機器で実行されるアプリケーション（application、APPと略称）であってもよい。理解されたいこととして、図1aは、クラウドゲームシステムのシステムアーキテクチャを例示的に示すためのものに過ぎず、クラウドゲームシステムの具体的なアーキテクチャを限定するものではない。例えば、他の実施例では、クラウドゲームシステムは、ゲーム鏡像を生成するための鏡像サーバを更に含み得る。

【0016】

上記のクラウドゲームシステムに基づいて、1つ又は複数のクラウドゲームの実行を実現することができる。ここで、任意のクラウドゲームの実行原理は次のとおりである。クラウドゲームのゲーム鏡像を作成した後、当該クラウドゲームのゲーム鏡像をエッジサーバに展開し、これにより、エッジサーバは、ゲーム鏡像をロードすることにより対応するシステムコンテナを起動することができる。ゲームクライアントによって送信された、クラウドゲームに関する接続要求を受信した場合、エッジサーバは、対応するシステムコンテナを割り当てて、当該ゲームクライアントとの接続を確立することができる。割り当てられたシステムコンテナとゲームクライアントとの間の接続が正常に確立された後、割り当てられたシステムコンテナは、クラウドゲームを実行し、クラウドゲームのゲーム画面をリアルタイムでレンダリングし、その後、図1cに示されるように、レンダリングされたゲーム画面をビデオストリームの形でゲームクライアントに伝送して表示する。ゲーム画面を表示するプロセスにおいて、ゲームクライアントは、プレーヤユーザによるゲーム画面への操作イベントをデータストリームの形で、対応するシステムコンテナに送信することができる。これに対応して、システムコンテナは、当該操作イベントに対応するレンダリングバッファ画像（すなわち、操作イベントに対応するフィードバック画像）を取得してレンダリングし、当該操作イベントに対応するフィードバック画像をゲームクライアントに返して表示することができる。

【0017】

ここで、システム容器は、操作イベントに対応するフィードバック画像をレンダリングするときに、通常、画像シェーディング処理、テクスチャ処理、変形処理などの多段階の画像処理を実行する。これに基づき、本願実施例は、画像シェーディング処理段階のための、共有GPU shader cacheメカニズムを提案し、当該共有GPU shader cacheメカニズムにより、画像シェーディング処理時間を短縮することで、フィードバック画像をレンダリングするのに必要なレンダリング時間を短縮し、これにより、ゲームの実行にかかる合計時間を短縮して、クラウドゲームの実行速度を加速させることができる。ここで、GPUシェーダ（GPU shader）は、GPUハードウェア機能の一部であり、ソフトウェアの実装では、GPUのドライバは、ユーザが記述したGPU shaderコード（シェーダコード）を、GPUによって実行可能な目標コー

ドにコンパイルすることができる。GPU shader cacheメカニズムの原理は次のとおりである。単一のシステムコンテナの場合、当該単一のシステムコンテナによりGPU shaderコードをコンパイルすることによって取得されたコンパイル結果をキャッシュし、これにより、当該単一のシステムコンテナがクラウドゲームを実行し続けるか、又はクラウドゲームを再実行するときに、同じGPU shaderコードをコンパイルする必要がある場合、コンパイルプロセスをスキップしてコンパイル結果を直接使用することができる。これに対応して、共有GPU shader cacheメカニズムの原理は、単一のシステムコンテナによってキャッシュされたGPU shaderコードのコンパイル結果を他のシステムコンテナと共有することであり、これにより、他のシステムコンテナが同じGPU shaderコードをコンパイルする必要があるときに、コンパイルプロセスをスキップして、共有で得られたコンパイル結果を直接使用することができる。

10

【0018】

特定の実現では、本願実施例で提案される共有GPU shader cacheメカニズムは、主に、以下の2つのメカニズムを含み得る。

【0019】

(1) 静的共有GPU shader cacheメカニズム

ここでの静的共有GPU shader cacheメカニズムとは、少なくとも1つのGPU shaderコードの第2コンパイル結果をゲーム鏡像に追加することで、当該ゲーム鏡像をロードする各システムコンテナがいずれも当該第2コンパイル結果を共有できるようにする共有メカニズムを指す。具体的には、当該静的共有GPU shader cacheメカニズムの原理は、大体、次のとおりである。クラウドゲームのゲーム鏡像の作成プロセスにおいて、クラウドゲームを事前に実行することができ、クラウドゲームの実行中に、少なくとも1つのGPU shaderコードをコンパイルして第2コンパイル結果を取得し、その後、第2コンパイル結果をクラウドゲームのゲーム鏡像に追加する。このようにして、クラウドゲームのゲーム鏡像を、異なるエッジサーバのシステムコンテナ又は同じエッジサーバの異なるシステムコンテナに展開する場合、当該ゲーム鏡像をロードする各システムコンテナは、いずれもこれらの第2コンパイル結果を含み得、これにより、各システムコンテナが特定のGPU shaderコードをコンパイルする必要がある場合、ゲーム鏡像に当該GPU shaderコードの第2コンパイル結果が既に含まれていると、各システムコンテナは、ゲーム鏡像から対応する第2コンパイル結果を直接取得して後続の処理を実行することができ、これにより、ゲームの実行加速を向上させることができる。ここから分かるように、静的共有GPU shader cacheメカニズムでは、ゲーム鏡像にGPU shaderコードの第2コンパイル結果が既に含まれており、そのため、追加の共有ディレクトリをシステムコンテナにマウントすることなく、同じゲーム間のコンパイル結果の共有の問題を解決することができる。

20

30

【0020】

(2) 動的共有GPU shader cacheメカニズム

ここでの動的共有GPU shader cacheメカニズムとは、ゲーム実行中に、共有ディレクトリにより、複数のシステムコンテナ間での同じGPU shaderコードのコンパイル結果の共有を実現する共有メカニズムを指す。特定の実現では、当該動的共有GPU shader cacheメカニズムの原理は大体、次のとおりである。各システムコンテナの実行プロセスにおいて、1つの共有ディレクトリを各システムコンテナにマウントして、GPU shaderコンパイルキャッシュディレクトリとして使用することができ、これにより、ゲーム実行中に各システムコンテナによりコンパイルされた各GPU shaderコードの第1コンパイル結果をすべて当該共有ディレクトリに同期的に格納することができる。このようにして、特定のシステムコンテナaが特定のGPU shaderコードをコンパイルする必要がある場合、当該GPU shaderコードが他のシステムコンテナによって既にコンパイルされ、且つその対応する第1コンパイル結果が目標共有ディレクトリに既にキャッシュされていると、当該システムコンテナa

40

50

は、目標共有ディレクトリから当該第1コンパイル結果を直接取得して後続の処理を実行することができ、これにより、ゲームの実行加速を向上させることができる。

【0021】

動的共有GPU shader cacheメカニズムでは、共有ディレクトリは複数のシステムコンテナにマウントでき、そのため、現在のゲームの実行中に単一のシステムコンテナにより新しく生成されたGPU shaderコードのコンパイル結果を共有ディレクトリにキャッシュすることで、当該コンパイル結果が動的に複数のシステムコンテナによってリアルタイムで共有及び使用できるようにする。ゲーム実行中に単一のシステムコンテナによってコンパイルされたGPU shaderコードの第1コンパイル結果を効果的に利用することにより、他の複数のシステムコンテナにおける同じ又は異なるクラウドゲームの実行速度を向上させることができる。ここから分かるように、本願実施例で提案される動的共有GPU shader cacheメカニズムは、同じゲーム間のコンパイル結果の共有の問題を解決することができるだけでなく、異なるゲーム間のGPU shader cacheの共有の問題も解決することができる。

10

【0022】

上記の説明から分かるように、本願実施例で提案される静的共有GPU shader cacheメカニズムと動的共有GPU shader cacheメカニズムの2つの共有GPU shader cacheメカニズムは両方とも、各システムコンテナ間のコンパイル結果の共有の問題をある程度解決でき、キャッシュされたコンパイル結果を使用して後続処理を直接実行することで、GPU shaderコードのコンパイルプロセスをスキップし、これにより、コードコンパイル時間を効果的に節約することができ、クラウドゲームの実行速度を向上させることができる。理解されたいこととして、実際の応用では、本願実施例で提案される2つの共有GPU shader cacheメカニズムは、個別に使用してもよいし、組み合わせて使用してもよく、本願実施例はこれに対して特に限定しない。

20

【0023】

上記の2つの共有GPU shader cacheメカニズムの関連する説明に基づき、本願実施例は画像処理方法を提案する。当該画像処理方法は、上記のクラウドゲームシステムにおける目標エッジサーバ内で実行される目標コンテナによって実行でき、ここでの目標エッジサーバは、クラウドゲームシステム内の任意のエッジサーバであり、目標コンテナは、目標エッジサーバのサーバ操作システムに基づいて実行される任意のシステムコンテナである。図2を参照すると、当該画像処理方法は、次のステップS201～ステップS203を含み得る。

30

【0024】

ステップS201において、目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、操作イベントに対応する目標シェーダコードを取得する。

【0025】

ここで、操作イベントは、目標コンテナに接続された目標ゲームクライアントに表示される目標ゲームのゲーム画面から検出されたものである。特定の実現では、目標ゲームを実行するプロセスでは、目標コンテナは、目標ゲームのゲーム画面をビデオストリームの形で目標ゲームクライアントに送信して表示することができる。これに対応して、ゲーム画面を表示するプロセスでは、目標ゲームクライアントは、目標ゲームへの操作イベントがあるかどうかを検出することができる。ここでの操作イベントは、物理制御コンポーネント（例えば、マウス、キーボードなど）を操作することによって入力されたイベント、重力検知装置（例えば、ジャイロスコープなど）を制御することによって入力されたイベント、ゲーム画面内の任意の画面内容を指でタッチすることによって入力されたイベントなどを含み得るが、これらに限定されない。目標ゲームクライアントは、当該操作イベントを検出した後、当該操作イベントをデータストリームの形で目標コンテナに伝送することができ、これに対応して、目標コンテナは、当該操作イベントを受信した後、当該操作イベントに対応する目標シェーダコードを取得することができる。

40

50

【0026】

ステップS202において、ゲームリソースから目標シェーダコードの目標コンパイル結果を取得する。

【0027】

本願実施例において、ゲームリソースは、目標ゲームのゲーム鏡像、及び目標コンテナと目標コンテナの関連コンテナとの間の目標共有ディレクトリのうちの少なくとも1つを含み得る。ここで、目標ゲームのゲーム鏡像は、上記の静的共有GPU shader cacheメカニズムに基づいて取得でき、当該目標ゲームのゲーム鏡像は、目標ゲームを事前に実行するときにコンパイルされた少なくとも1つのシェーダコード（すなわち、GPU shaderコード）の第2コンパイル結果を含み得、各第2コンパイル結果はすべて、当該ゲーム鏡像を作成するときにゲーム鏡像に追加されたものである。目標コンテナと関連コンテナとの間の目標共有ディレクトリは、上記の動的共有GPU shader cacheメカニズムに基づいて取得でき、当該目標共有ディレクトリは、関連コンテナがゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果を含み得る。理解されたいこととして、目標共有ディレクトリは、関連コンテナの第1コンパイル結果だけでなく、目標コンテナがゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果も含み得る。ここで、関連コンテナとは、目標エッジサーバのサーバ操作システムに基づいて実行される任意のシステムコンテナを指し、ここでの関連ゲームは、目標ゲーム、又は目標ゲームと同じゲームエンジンを使用するゲームを含み得る。

【0028】

ここから分かるように、本願実施例は、上記の静的共有GPU shader cacheメカニズムと動的共有GPU shader cacheメカニズムを個別に使用してもよいし、その2つを組み合わせ使用してもよい。静的共有GPU shader cacheメカニズムを個別に使用する場合、ゲームリソースは、目標ゲームのゲーム鏡像のみを含み得る。この場合、ステップS202の具体的な実施形態は、目標ゲームのゲーム鏡像が目標シェーダコードの第2コンパイル結果を含む場合、目標ゲームのゲーム鏡像から当該目標シェーダコードの第2コンパイル結果を目標コンパイル結果として取得し、そうでない場合、目標シェーダコードをコンパイルして目標コンパイル結果を取得することであり得る。

【0029】

動的共有GPU shader cacheメカニズムを個別に使用する場合、ゲームリソースは、目標コンテナと関連コンテナとの間の目標共有ディレクトリのみを含み得る。この場合、ステップS202の具体的な実施形態は、目標共有ディレクトリが目標シェーダコードの第1コンパイル結果を含む場合、目標共有ディレクトリから当該目標シェーダコードの第1コンパイル結果を目標コンパイル結果として直接取得し、そうでない場合、目標シェーダコードをコンパイルして目標コンパイル結果を取得することであり得る。ここで、当該目標シェーダコードの第1コンパイル結果は、関連コンテナが関連ゲームを実行する過程で目標シェーダコードをコンパイルすることによって取得したものであってもよいし、目標コンテナが目標ゲームを実行する履歴過程で目標シェーダコードをコンパイルすることによって取得したものであってもよく、本願実施例はこれに対して特に限定しない。

【0030】

静的共有GPU shader cacheメカニズムと動的共有GPU shader cacheメカニズムを組み合わせ使用する場合、ゲームリソースは、目標ゲームのゲーム鏡像及び目標共有ディレクトリを含み得る。この場合、ステップS202の具体的な実施形態は、目標ゲームのゲーム鏡像において、先ず、目標シェーダコードの第2コンパイル結果の有無を検出し、前記目標シェーダコードの第2コンパイル結果がある場合、ゲーム鏡像から目標シェーダコードの第2コンパイル結果を目標コンパイル結果として取得し、前記目標シェーダコードの第2コンパイル結果がない場合、目標コンテナと目標コン

テナの関連コンテナとの間の目標共有ディレクトリから、第1コンパイル結果を目標シェーダコードの目標コンパイル結果として取得することであり得る。留意されたいこととして、目標共有ディレクトリからの目標コンパイル結果の取得に失敗したことは、目標共有ディレクトリに目標コンパイル結果が存在しないことを意味し、つまり、当該目標シェーダコードが目標コンテナ及び関連コンテナによってまだコンパイルされていないことを意味し、この場合、目標コンテナは、目標シェーダコードをコンパイルすることにより目標コンパイル結果を取得することができる。

【0031】

ステップS203において、目標コンパイル結果に基づいて図形処理を実行して、操作イベントに対応するフィードバック画像を取得する。

【0032】

ステップS203を実行した後、当該画像処理方法は、前記フィードバック画像を目標ゲームクライアントに表示することを更に含み得る。

【0033】

目標コンパイル結果を取得した後、目標コンテナは、GPUを呼び出して、当該目標コンパイル結果に基づいて一連の図形処理を実行することにより、操作イベントに対応するフィードバック画像を取得することができる。ここで、図形処理は、アルファチャネルマージ処理、画素値座標の調整処理などを含み得るが、これらに限定されない。フィードバック画像を取得した後、目標コンテナは、当該フィードバック画像をビデオストリームの形で目標ゲームクライアントに送信することによって、目標ゲームクライアントに当該フィードバック画像を表示することができる。

【0034】

本願実施例では、目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、操作イベントに対応する目標シェーダコードを取得した後、目標シェーダコードのコンパイルプロセスをスキップして、ゲーム鏡像又は目標コンテナと目標コンテナの関連コンテナとの間の目標共有ディレクトリから目標シェーダコードの目標コンパイル結果を直接に取得することができる。その後、目標コンパイル結果に基づいて図形処理を実行して、操作イベントに対応するフィードバック画像を取得し、目標ゲームクライアントにフィードバック画像を表示することができる。ここから分かるように、本願実施例は、ゲーム鏡像又は目標共有ディレクトリ内の、関連コンテナによってコンパイルされた目標コンパイル結果を直接使用して図形処理を実行することで、目標シェーダコードのコンパイル時間を効果的に短縮することができ、これにより、目標ゲームの実行速度を向上させることができる。

【0035】

上記の図2に示される画像処理方法の実施例の関連説明に基づき、以下、図2に示される方法の実施例に関する目標ゲームのゲーム鏡像の具体的な作成プロセス、及び目標共有ディレクトリの具体マウントプロセスをそれぞれ説明する。

【0036】

(1) 静的共有GPU shader cacheメカニズムに基づく目標ゲームのゲーム鏡像の作成

図3aを参照すると、目標ゲームのゲーム鏡像を作成するときに、まず、機器操作システムの基礎鏡像ファイル(例えば、Android基礎鏡像ファイルなど)をロードすることができる。次に、目標ゲームをインストールして、目標ゲームに関する第1部分のGPU shaderコードのコンパイルをトリガして、第1部分のコンパイル結果を取得し、当該第1部分のコンパイル結果を目標ゲームのゲーム鏡像に追加することができる。1つの実施形態において、当該第1部分のコンパイル結果及び第1部分のGPU shaderコードを対応する目標ゲームのゲーム鏡像に直接追加することができる。別の実施形態において、第1部分のコンパイル結果及び第1部分のGPU shaderコードのインデックス値を、対応する目標ゲームのゲーム鏡像に追加することができ、ここでのインデックス値は、GPU shaderコードに対してハッシュ演算を実行することによ

10

20

30

40

50

り取得されたものであり得る。その後、目標ゲームを実行し、当該目標ゲームのシミュレーション操作イベントを入力して、当該シミュレーション操作イベントに対応する第2部分のGPU shaderコードのコンパイルをトリガすることで、第2部分のコンパイル結果を取得し、当該第2部分のコンパイル結果を目標ゲームのゲーム鏡像に追加することができる。1つの実施形態において、当該第2部分のコンパイル結果及び第2部分のGPU shaderコードに対応する目標ゲームのゲーム鏡像に直接追加することができる。別の実施形態において、第2部分のコンパイル結果及び第2部分のGPU shaderコードのインデックス値に対応する目標ゲームのゲーム鏡像に追加することができる。

【0037】

(2) 動的共有GPU shader cacheメカニズムに基づく目標共有ディレクトリのマウント

まず、目標エッジサーバは目標共有ディレクトリを選択することができる。次に、目標共有ディレクトリを目標コンテナ及び関連コンテナにマウントすることにより、目標コンテナ及び関連コンテナでのそれぞれのゲームの実行プロセスにおいて、コンパイルされた各GPU shaderコードの第1コンパイル結果をすべて当該目標共有ディレクトリに同期的にキャッシュすることができる。具体的には、目標エッジサーバは、目標共有ディレクトリを含むディレクトリマウント命令を生成することができ、当該ディレクトリマウント命令を目標コンテナ及び関連コンテナにそれぞれ送信することにより、目標コンテナは、当該目標共有ディレクトリを目標コンテナ内の動的キャッシュディレクトリにマウントすることができ、関連コンテナは、当該目標共有ディレクトリを関連コンテナ内の所定のディレクトリにマウントすることができる。目標共有ディレクトリを目標コンテナ及び関連コンテナに正常にマウントした後、関連コンテナの実行中に、関連コンテナが少なくとも1つのGPU shaderコードをコンパイルして対応する第1コンパイル結果を取得した場合、当該関連コンテナによってコンパイルされた第1コンパイル結果を目標共有ディレクトリに追加することができる。同様に、目標コンテナの実行中に、目標コンテナが少なくとも1つのGPU shaderコードをコンパイルして第1コンパイル結果を取得した場合、当該目標コンテナによってコンパイルされた第1コンパイル結果を目標共有ディレクトリに追加することができる。

【0038】

ここで、目標共有ディレクトリは、2つの選択方式をサポートすることができる。

【0039】

選択方式1：目標エッジサーバのサーバ操作システムが配置されているサーバホスト機器（すなわち、Host機器）で目標共有ディレクトリ（例えば、/home/shadercache）を選択して、目標エッジサーバで実行されるすべてのシステムコンテナの使用のために提供し、つまり、この選択方式では、目標共有ディレクトリは、サーバ操作システムが配置されているサーバホスト機器内にある。実際の応用では、システムコンテナが様々な種類のクラウドゲームを実行する過程で、対応するGPU shaderコードの第1コンパイル結果を生成するため、複数のシステムコンテナによって蓄積されたGPU shaderコードの第1コンパイル結果のデータ量は、通常、比較的に多い。そのため、これらのシステムコンテナのすべてを同じ共有ディレクトリにマウントすると、当該共有ディレクトリにキャッシュされているコンパイル結果のデータ量が比較的に多くなり、その結果、後続で当該共有ディレクトリから関連するGPU shaderコードのコンパイル結果を照会するのに長い時間がかかる。これに基づき、サーバホスト機器は、目標エッジサーバの各システムコンテナによって実行されるクラウドゲームの種類の数に基づいて、提供される共有ディレクトリの数を決定することができる。

【0040】

目標エッジサーバの各システムコンテナによって実行されるクラウドゲームの種類が少ない場合（すなわち、クラウドゲームの種類の数が数量閾値より小さい場合）、サーバホスト機器は、1つの共有ディレクトリのみを提供することにより、目標エッジサーバの各システムコンテナが当該共有ディレクトリにマウントするようにすることができる。この

場合、目標エッジサーバは、サーバホスト機器によって提供された当該共有ディレクトリを、目標コンテナと関連コンテナとの間の目標共有ディレクトリとして使用することができる。この実施形態において、目標コンテナが目標共有ディレクトリにマウントする概略図は、図3bに示されるとおりである。図3bから分かるように、サーバホスト機器は目標共有ディレクトリのみを含み、そのため、当該目標共有ディレクトリは、目標コンテナ及び目標コンテナの関連コンテナをマウントするために使用できるだけでなく、目標エッジサーバにおける目標コンテナに関連しないシステムコンテナをマウントするためにも使用できる。

【0041】

目標エッジサーバの各システムコンテナによって実行されるクラウドゲームの種類が多い場合（すなわち、クラウドゲームの種類の数(quantity)が数量閾値以上である場合）、サーバホスト機器は、複数の共有ディレクトリを提供することができる。同じクラウドゲーム又は同じゲームエンジンを共有するクラウドゲームを実行する関連システムコンテナの場合、これらの関連システムコンテナは、同じGPU shaderコードを使用する可能性が高いため、これらの関連システムコンテナは、同じ共有ディレクトリにマウントすることができる。非関連システムコンテナの場合、これらの非関連システムコンテナがサーバホスト機器の他の共有ディレクトリにマウントするようにすることができる。この場合、目標エッジサーバは、サーバホスト機器によって提供される複数の共有ディレクトリから、1つの共有ディレクトリを、目標コンテナと関連コンテナとの間の目標共有ディレクトリとしてランダムに選択することができる。この実施形態において、目標コンテナが目標共有ディレクトリにマウントする概略図は、図3cに示されるとおりである。図3cから分かるように、サーバホスト機器は目標共有ディレクトリを含むだけでなく、目標共有ディレクトリ以外の他の共有ディレクトリも含み、そのため、当該目標共有ディレクトリは、目標コンテナ及び目標コンテナの関連コンテナをマウントするためにのみ使用でき、サーバホスト機器の他の共有ディレクトリは、目標エッジサーバにおける目標コンテナに関連しないシステムコンテナをマウントするために使用できる。理解されたいこととして、クラウドゲームの数が比較的に多い場合、目標共有ディレクトリから関連するGPU shaderコードのコンパイル結果を照会することにかかる時間という指標を考慮しない場合、サーバホスト機器は、1つの共有ディレクトリのみを提供してもよい。

【0042】

選択方式2：インターネットネットワーク上のネットワークファイルシステムから目標共有ディレクトリ（例えば、192.168.10.10:/mnt/share）を選択して、ネットワーク上のすべてのエッジサーバの各システムコンテナの使用のために提供する。当該ネットワークファイルシステムとクラウドゲームシステムの各エッジサーバは、相互に独立し且つ各エッジサーバのシステムコンテナからのアクセスを許可する。つまり、この選択方式では、目標共有ディレクトリはネットワークファイルシステム内にある。特定の実現では、サーバホスト機器が、目標エッジサーバの各システムコンテナによって実行されるクラウドゲームの種類の数に基づいて、提供される共有ディレクトリ数を決定する理由と同様に、ネットワークファイルシステムも、目標エッジサーバの各システムコンテナによって実行されるクラウドゲームの種類の数に基づいて、提供される共有ディレクトリ数を決定することができる。

【0043】

クラウドゲームシステムの各システムコンテナによって実行されるクラウドゲームの種類が少ない場合（すなわち、クラウドゲームの種類の数(quantity)が数量閾値より小さい場合）、ネットワークファイルシステムは、1つの共有ディレクトリのみを提供することで、クラウドゲームシステムの各システムコンテナが当該共有ディレクトリにマウントすることができる。この場合、目標エッジサーバは、ネットワークファイルシステムによって提供される当該共有ディレクトリを、目標コンテナと関連コンテナとの間の目標共有ディレクトリとして使用することができる。この実施形態において、目標コンテナが目標共有ディレクトリにマウントする概略図は、図3dに示されるとおりである。図3dから分

10

20

30

40

50

かるように、ネットワークファイルシステムは目標共有ディレクトリのみを含み、そのため、当該目標共有ディレクトリは、目標コンテナ及び目標コンテナの関連コンテナをマウントするために使用できるだけでなく、クラウドゲームシステムにおける目標コンテナに関連しないシステムコンテナをマウントするためにも使用できる。

【0044】

クラウドゲームシステムの各システムコンテナによって実行されるクラウドゲームの種類が多い場合（すなわち、クラウドゲームの種類の数数量閾値以上である場合）、ネットワークファイルシステムは、複数の共有ディレクトリを提供することができる。同じクラウドゲーム又は同じゲームエンジンを共有するクラウドゲームを実行する関連システムコンテナの場合、これらの関連システムコンテナは、同じGPU shaderコードを使用する可能性が高いため、これらの関連システムコンテナは、同じ共有ディレクトリにマウントすることができる。非関連システムコンテナの場合、これらの非関連システムコンテナがネットワークファイルシステムの他の共有ディレクトリにマウントすることができる。この場合、目標エッジサーバは、ネットワークファイルシステムによって提供される複数の共有ディレクトリから、1つの共有ディレクトリを、目標コンテナと関連コンテナとの間の目標共有ディレクトリとしてランダムに選択することができる。この実施形態において、目標コンテナが目標共有ディレクトリにマウントする概略図は、図3eに示されるとおりである。図3eから分かるように、ネットワークファイルシステムは目標共有ディレクトリを含むだけでなく、目標共有ディレクトリ以外の他の共有ディレクトリも含み、そのため、当該目標共有ディレクトリは、目標コンテナ及び目標コンテナの関連コンテナをマウントするためにのみ使用でき、ネットワークファイルシステムの他の共有ディレクトリは、クラウドゲームシステム内の、目標コンテナに関連しないシステムコンテナをマウントするために使用できる。理解されたいこととして、クラウドゲームの数比較的に多い場合、目標共有ディレクトリから関連するGPU shaderコードのコンパイル結果を照会することにかかる時間という指標を考慮しない場合、ネットワークファイルシステムは、1つの共有ディレクトリのみを提供してもよい。

【0045】

実際の応用では、目標エッジサーバは、上記の2つの選択方式のいずれかを選択して、目標共有ディレクトリを決定することができる。更に、選択方式1は、サーバホスト機器から目標共有ディレクトリを選択することであり、且つ目標共有ディレクトリと目標コンテナは両方とも目標エッジサーバにあるため、目標共有ディレクトリの各第1コンパイル結果への目標コンテナのアクセス速度を確保することができる。選択方式2は、ネットワークファイルシステムから目標共有ディレクトリを選択することであり、且つネットワークファイルシステムと目標エッジサーバは相互に独立したものであるため、ネットワークの輻輳などの要因により、目標共有ディレクトリの各第1コンパイル結果への目標コンテナのアクセス速度が遅くなる可能性がある。これに基づき、目標エッジサーバは、ネットワークファイルシステムの実際のアクセス速度に基づいて、上記の2つの選択方式から1つを選択して、目標共有ディレクトリを決定することができる。具体的には、ネットワークファイルシステムのアクセス速度が保証されていない場合、実施形態1を選択して目標共有ディレクトリを決定することができ、ネットワークファイルシステムのアクセス速度が保証されている場合、実施形態2を選択して目標共有ディレクトリを決定することができる。

【0046】

理解されたいこととして、本願実施例は、ネットワークファイルシステムのアクセス速度というパラメータに基づいて選択方式を選択することができることを例として説明したものに過ぎず、網羅的なものではない。つまり、他の実施例では、他のパラメータに基づいて選択方式を選択してもよい。例えば、選択方式1は、サーバホスト機器から目標共有ディレクトリを選択することであるため、目標共有ディレクトリの各第1コンパイル結果は、他のエッジサーバのシステムコンテナによってアクセスされることができない。選択方式2は、ネットワークファイルシステムから目標共有ディレクトリを選択することであ

るため、目標共有ディレクトリの各第1コンパイル結果は、他のエッジサーバのシステムコンテナによってアクセスされることができる。これに基づき、目標エッジサーバは、実際のゲーム展開要件に基づいて、上記の2つの選択方式から1つを選択して、目標共有ディレクトリを決定することができる。具体的には、ゲーム展開要件が、異なるエッジサーバ間でのコンパイル結果の共有を実現する必要がないことである場合、選択方式1を選択して目標共有ディレクトリを決定することができる。ゲーム展開要件が、異なるエッジサーバ間でのコンパイル結果の共有を実現する必要があることである場合、選択方式2を選択して目標共有ディレクトリを決定することができる。

【0047】

更に、上記の図2に示される画像処理方法の実施例の関連する説明、及び上記の静的共有GPU shader cacheメカニズムに基づく目標ゲームのゲーム鏡像の作成に関する具体的なプロセスの関連する説明、及び動的共有GPU shader cacheメカニズムに基づく目標共有ディレクトリのマウントに関する具体的なプロセスの関連する説明に基づき、本願実施例は、より具体的な画像処理方法の例示的なフローチャートを更に提案する。本願実施例は、主に、静的共有GPU shader cacheメカニズム及び動的共有GPU shader cacheメカニズムを組み合わせ使用することを例として説明し、当該画像処理方法は、上記のクラウドゲームシステムの目標エッジサーバ内で実行される目標コンテナによって実行でき、当該目標コンテナは、目標ゲームのゲーム鏡像をロードすることによって起動され、目標ゲームのゲーム鏡像は少なくとも、目標ゲームを事前に実行するときにコンパイルされた少なくとも1つのシェダコードの第2コンパイル結果を含み得る。図4を参照すると、当該画像処理方法は、次のステップS401～ステップS407を含み得る。

【0048】

ステップS401において、目標エッジサーバによって送信されたディレクトリマウント命令を受信し、当該ディレクトリマウント命令は前記目標共有ディレクトリを搬送する。

【0049】

ステップS402において、目標コンテナ内で動的キャッシュディレクトリを指定し、目標共有ディレクトリを動的キャッシュディレクトリにマウントする。

【0050】

1つの実施形態において、目標コンテナ内で任意のディレクトリを動的キャッシュディレクトリとして指定することができる。つまり、この実施形態では、動的キャッシュディレクトリと、ゲーム鏡像における第2コンパイル結果のストレージディレクトリは互いに同じでも異なってもよい。別の実施形態では、コンテナとディレクトリのマウントの実施形態の制限により、目標共有ディレクトリを目標コンテナ内部にマウントする場合、目標コンテナ内部の、当該目標共有ディレクトリがマウントされる対応するディレクトリのファイルがクリアされる。したがって、ゲーム鏡像の第2コンパイル結果がクリアされることを回避するために、目標コンテナ内部で2つの異なる環境変数を使用して、目標共有ディレクトリにマウントされる動的キャッシュディレクトリと、ゲーム鏡像における第2コンパイル結果のストレージディレクトリをそれぞれ指定することで、動的キャッシュディレクトリと第2コンパイル結果のストレージディレクトリを分離することができる。その結果、目標共有ディレクトリを動的キャッシュディレクトリにマウントするときに、動的キャッシュディレクトリのファイルのみがクリアされ、ストレージディレクトリの第2コンパイル結果に影響を与えない。

【0051】

この実施形態では、動的キャッシュディレクトリは、第1環境変数を使用して指定することができるが、当該第1環境変数は、Mesaコードの「MESA_GLSL_CACHE_DIR」環境変数を含み得るが、これに限定されない。ゲーム鏡像における第2コンパイル結果のストレージディレクトリは、第2環境変数を使用して指定することができるが、具体的には、当該第2環境変数は、Mesaコードの「MESA_GLSL_CACHE_DIR_STATIC」環境変数を含み得るが、これに限定されない。ここでのMesa

は、オープンソースのコンピュータ図形ライブラリであり、Mesaは、OpenGL／OpenGLsのアプリケーションインターフェースを実現するために使用できる。ここでのOpenGLは、主に、3D図形をレンダリングするためのクロスプラットフォーム・アプリケーション・プログラミング・インターフェース（API）であり、OpenGLsはOpenGL APIのサブセットであり、主に、組み込み型機器（例えば、スマートフォンなど）で使用される。

【0052】

目標コンテナ内で動的キャッシュディレクトリを指定した後、目標共有ディレクトリを動的キャッシュディレクトリにマウントすることで、目標共有ディレクトリと動的キャッシュディレクトリとの間のデータ同期を維持することができる。ここで、当該目標共有ディレクトリは、関連コンテナの所定のディレクトリに更にマウントされ、これによって、関連コンテナが関連ゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果をキャッシュするようにする。例えば、目標共有ディレクトリがサーバホスト機器にあり、当該サーバホスト機器が目標共有ディレクトリのみを含む場合、当該目標共有ディレクトリをマウントすることを示す概略図は、図5aに示されるとおりであり、当該サーバホスト機器が目標共有ディレクトリ及び他の共有ディレクトリを含む場合、当該目標共有ディレクトリをマウントすることを示す概略図は、図5bに示されるとおりである。別の例として、目標共有ディレクトリがネットワークファイルシステム内にあり、当該ネットワークファイルシステムが目標共有ディレクトリのみを含む場合、当該目標共有ディレクトリをマウントすることを示す概略図は、図5cに示されるとおりであり、当該ネットワークファイルシステムが目標共有ディレクトリ及び他の共有ディレクトリを含む場合、当該目標共有ディレクトリをマウントすることを示す概略図は、図5dに示されるとおりである。

【0053】

ステップS403において、目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、操作イベントに対応する目標シェーダコードを取得する。

【0054】

ステップS404において、目標ゲームのゲーム鏡像において、目標シェーダコードの第2コンパイル結果の有無を検出する。

【0055】

上記の説明から分かるように、ゲーム鏡像は少なくとも、目標ゲームを事前に実行するときにコンパイルされた少なくとも1つのシェーダコードの第2コンパイル結果を含み得る。1つの実施形態において、当該ゲーム鏡像は、各第2コンパイル結果に対応するシェーダコードを更に含み得、この実施形態では、目標ゲームのゲーム鏡像が目標シェーダコードを含むかどうかを直接に検出することができる。目標ゲームのゲーム鏡像が当該目標シェーダコードを含む場合、ゲーム鏡像に目標シェーダコードの第2コンパイル結果があると決定することができ、目標ゲームのゲーム鏡像が当該目標シェーダコードを含まない場合、ゲーム鏡像に前記目標シェーダコードの第2コンパイル結果がないと決定することができる。

【0056】

別の実施形態において、当該ゲーム鏡像は、少なくとも1つのシェーダコードのインデックス値を更に含み得、1つのインデックス値は1つの第2コンパイル結果に対応する。この場合、この実施形態では、目標シェーダコードに対してハッシュ演算を実行して、目標シェーダコードの目標インデックス値を取得した後、ゲーム鏡像に当該目標インデックス値が含まれているかどうかを検出することができる。ゲーム鏡像が目標インデックス値を含む場合、ゲーム鏡像に目標シェーダコードの第2コンパイル結果があると決定することができ、ゲーム鏡像が目標インデックス値を含まない場合、ゲーム鏡像に前記目標シェーダコードの第2コンパイル結果がないと決定することができる。

【0057】

ゲーム鏡像に目標シェーダコードの第2コンパイル結果があると決定した場合、ステッ

プ S 4 0 5 を実行することができ、ゲーム鏡像に目標シェーダコードの第 2 コンパイル結果がないと決定した場合、ステップ S 4 0 6 を実行することができる。

【 0 0 5 8 】

ステップ S 4 0 5 において、ゲーム鏡像に前記目標シェーダコードの第 2 コンパイル結果がある場合、ゲーム鏡像から、前記目標シェーダコードの第 2 コンパイル結果を目標コンパイル結果として取得する。

【 0 0 5 9 】

ステップ S 4 0 6 において、ゲーム鏡像に前記目標シェーダコードの第 2 コンパイル結果がない場合、目標コンテナと目標コンテナの関連コンテナとの間の目標共有ディレクトリから、第 1 コンパイル結果を目標シェーダコードの目標コンパイル結果として取得する。

【 0 0 6 0 】

具体的な実施プロセスでは、目標共有ディレクトリは、少なくとも 1 つのシェーダコードの第 1 コンパイル結果を含み得、各第 1 コンパイル結果は 1 つのインデックス値に対応することができる。この場合、ステップ S 4 0 6 の具体的な実施形態は、図 5 e に示されるとおりであり、具体的には、目標コンテナは、目標共有ディレクトリに目標インデックス値があるかどうかを検出することができる。目標共有ディレクトリに目標インデックス値がある場合、コードのコンパイルプロセスをスキップして、目標インデックス値に対応する第 1 コンパイル結果を目標コンパイル結果として直接に取得する。目標共有ディレクトリに目標インデックス値がない場合、これは、目標シェーダコードの目標コンパイル結果がゲームリソースに含まれていないことを意味し、この場合、目標シェーダコードをコンパイルして、目標コンパイル結果を取得することができ、当該目標コンパイル結果を目標共有ディレクトリに同期させるために、目標コンパイル結果を動的キャッシュディレクトリにキャッシュし、これにより、関連コンテナが目標シェーダコードをコンパイルする必要がある場合、目標共有ディレクトリから前記目標コンパイル結果を直接取得することができる。更に、当該目標インデックス値を目標共有ディレクトリに同期させるために、目標シェーダコードの目標インデックス値を動的キャッシュディレクトリと一緒にキャッシュすることができる。

【 0 0 6 1 】

ステップ S 4 0 7 において、目標コンパイル結果に基づいて図形処理を実行して、操作イベントに対応するフィードバック画像を取得する。ステップ S 4 0 7 を実行した後、当該画像処理方法は、フィードバック画像を目標ゲームクライアントに表示することを更に含み得る。

【 0 0 6 2 】

本願実施例では、目標コンテナ内で実行される目標ゲームに対する操作イベントにตอบสนองして、操作イベントに対応する目標シェーダコードを取得した後、目標シェーダコードのコンパイルプロセスをスキップして、ゲーム鏡像又は目標コンテナと目標コンテナの関連コンテナとの間の目標共有ディレクトリから目標シェーダコードの目標コンパイル結果を直接に取得することができる。その後、目標コンパイル結果に基づいて図形処理を実行して、操作イベントに対応するフィードバック画像を取得し、目標ゲームクライアントにフィードバック画像を表示することができる。ここから分かるように、本願実施例は、ゲーム鏡像又は目標共有ディレクトリ内の、関連コンテナによってコンパイルされた目標コンパイル結果を直接使用して図形処理を実行することで、目標シェーダコードのコンパイル時間を効果的に短縮することができ、これにより、目標ゲームの実行速度を向上させることができる。

【 0 0 6 3 】

上記の画像処理方法の実施例の説明に基づき、本願実施例は、画像処理装置を更に開示し、前記画像処理装置は、上記の目標エッジサーバで実行されるコンピュータプログラム（プログラムコードを含む）であってもよい。当該画像処理装置は、図 2 又は図 4 に示される方法を実行することができる。図 6 を参照すると、前記画像処理装置は、取得ユニット 6 0 1 と、処理ユニット 6 0 2 とを実行することができる。

【0064】

取得ユニット601は、目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、前記操作イベントに対応する目標シェーダコードを取得するように構成され、前記操作イベントは、前記目標コンテナに接続された目標ゲームクライアントに表示される前記目標ゲームのゲーム画面から検出されたものであり、

前記取得ユニット601は更に、ゲームリソースから、前記目標シェーダコードの目標コンパイル結果を取得するように構成され、前記ゲームリソースは、前記目標ゲームのゲーム鏡像、及び前記目標コンテナと前記目標コンテナの関連コンテナとの間の目標共有ディレクトリのうちの少なくとも1つを含み、前記目標共有ディレクトリは、前記関連コンテナがゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果を含み、

10

処理ユニット602は、前記目標コンパイル結果に基づいて図形処理を実行して、前記操作イベントに対応するフィードバック画像を取得し、前記目標ゲームクライアントに前記フィードバック画像を表示するように構成される。

【0065】

1つの実施形態において、前記ゲームリソースは、前記ゲーム鏡像及び前記目標共有ディレクトリを含み、前記ゲーム鏡像は、前記目標ゲームを事前に実行するときにコンパイルされた少なくとも1つのシェーダコードの第2コンパイル結果を含み、各第2コンパイル結果は、前記ゲーム鏡像を作成するときに前記ゲーム鏡像に追加されたものであり、これに対応して、取得ユニット601は、ゲームリソースから前記目標シェーダコードの目標コンパイル結果を取得する場合、具体的に、

20

前記目標ゲームのゲーム鏡像において、前記目標シェーダコードの第2コンパイル結果の有無を検出し、

前記目標シェーダコードの第2コンパイル結果がある場合、前記ゲーム鏡像から、前記目標シェーダコードの第2コンパイル結果を目標コンパイル結果として取得し、

前記目標シェーダコードの第2コンパイル結果がない場合、前記目標コンテナと前記目標コンテナの関連コンテナとの間の目標共有ディレクトリから、前記第1コンパイル結果を前記目標シェーダコードの目標コンパイル結果として取得するように構成され得る。

【0066】

別の実施形態において、前記ゲーム鏡像は、前記少なくとも1つのシェーダコードのインデックス値を更に含み、1つのインデックス値は1つの第2コンパイル結果に対応し、これに対応して、取得ユニット601は、前記目標ゲームのゲーム鏡像において、前記目標シェーダコードの第2コンパイル結果の有無を検出する場合、具体的に、

30

前記目標シェーダコードに対してハッシュ演算を実行して、前記目標シェーダコードの目標インデックス値を取得し、

前記ゲーム鏡像が前記目標インデックス値を含む場合、前記ゲーム鏡像に前記目標シェーダコードの第2コンパイル結果があると決定し、

前記ゲーム鏡像が前記目標インデックス値を含まない場合、前記ゲーム鏡像に前記目標シェーダコードの第2コンパイル結果がないと決定するように構成され得る。

【0067】

40

更に別の実施形態において、前記目標ゲームはクラウドゲームシステムにおけるクラウドゲームである、前記クラウドゲームシステムは、少なくとも1つのエッジサーバ及び複数のゲームクライアントを含み、各エッジサーバには少なくとも1つのシステムコンテナが展開され、各システムコンテナは少なくとも1つのゲームクライアントに接続され、

前記各システムコンテナは1つ又は複数のゲームを実行するために使用され、前記各システムコンテナは、任意のゲームを実行する過程で、前記任意のゲームのゲーム画面を、前記各システムコンテナに接続されたゲームクライアントに伝送して表示し、

ここで、前記目標コンテナは、目標エッジサーバのサーバ操作システムに基づいて実行される任意のシステムコンテナであり、前記目標エッジサーバは、前記クラウドゲームシステムの任意のエッジサーバであり、前記関連コンテナは、前記目標ゲームに関連付けら

50

れた関連ゲームを実行するシステムコンテナであり、前記関連ゲームは、前記目標ゲーム、又は前記目標ゲームと同じゲームエンジンを使用するゲームを含む。

【0068】

更に別の実施形態において、処理ユニット602は更に、

前記目標エッジサーバによって送信されたディレクトリマウント命令を受信し、ここで、前記ディレクトリマウント命令は前記目標共有ディレクトリを搬送し、

前記目標コンテナ内で動的キャッシュディレクトリを指定し、前記目標共有ディレクトリと前記動的キャッシュディレクトリとの間のデータ同期を維持するように、前記目標共有ディレクトリを前記動的キャッシュディレクトリにマウントし、

ここで、前記目標共有ディレクトリは前記関連コンテナの所定のディレクトリに更にマウントされ、これによって、前記関連コンテナが前記関連ゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果をキャッシュするようにする。

【0069】

更に別の実施形態において、前記動的キャッシュディレクトリは、第1環境変数によって指定され、前記ゲーム鏡像における前記第2コンパイル結果のストレージディレクトリは、第2環境変数によって指定される。

【0070】

更に別の実施形態において、前記目標共有ディレクトリは、前記サーバ操作システムが配置されているサーバホスト機器にあり、又は、

前記共有ディレクトリは、ネットワークファイルシステム内にあり、前記ネットワークファイルシステムと前記クラウドゲームシステムの各エッジサーバは相互に独立しかつ前記各エッジサーバのシステムコンテナからのアクセスを許可する。

【0071】

更に別の実施形態において、前記目標共有ディレクトリが前記サーバホスト機器にある場合、前記サーバホスト機器は、前記目標共有ディレクトリ以外の他の共有ディレクトリを更に含み、前記サーバホスト機器の他の共有ディレクトリは、前記目標コンテナに関連しない、前記目標エッジサーバのシステムコンテナをマウントするために使用され、

前記目標共有ディレクトリが前記ネットワークファイルシステム内にある場合、前記ネットワークファイルシステムは、前記目標共有ディレクトリ以外の他の共有ディレクトリを更に含み、前記ネットワークファイルシステムの共有ディレクトリは、前記目標コンテナに関連しない、前記クラウドゲームシステムのシステムコンテナをマウントするために使用される。

【0072】

更に別の実施形態において、前記ゲームリソースは前記目標シェーダコードの目標コンパイル結果を含まず、これに対応して、処理ユニット602は更に、

前記ゲームリソースからの前記目標コンパイル結果の取得に失敗した場合、前記目標シェーダコードをコンパイルして前記目標コンパイル結果を取得し、

前記目標コンパイル結果を前記目標共有ディレクトリに同期させるために、前記目標コンパイル結果を前記動的キャッシュディレクトリにキャッシュし、これにより、前記関連コンテナが前記目標シェーダコードをコンパイルする必要がある場合、前記目標共有ディレクトリから前記目標コンパイル結果を取得するようにするように構成される。

【0073】

本願の1つの実施例によれば、図2又は図4に示される方法に関する各ステップはすべて、図6に示される画像処理装置の各ユニットによって実行されてもよい。例えば、図2に示されるステップS201～ステップS202は、図6に示される取得ユニット601によって実行されてもよく、ステップS203は、図6に示される処理ユニット602によって実行されてもよい。別の例として、図4に示されるステップS401～ステップS402及びステップS407は、図6に示される処理ユニット602によって実行されてもよく、ステップS403～ステップS406は、図6に示される取得ユニット601に

10

20

30

40

50

よって実行されてもよい。

【0074】

本願の別の実施例によれば、図6に示される画像処理装置の各ユニットは、それぞれ別個のユニットであってもよいし、1つ又は複数の他のユニットに組み合わされてもよく、或いは、その1つ又は複数の特定のユニットは、より小さな機能を備えた複数のユニットに更に分割されてもよく、これは、本願の実施例の技術的效果の実現に影響を与えることなく同じ動作を実現することができる。上記のユニットは、ロジック機能に基づいて分割されたものであってもよく、実際の応用では、1つのユニットの機能を複数のユニットにより実現してもよく、或いは、複数のユニットの機能を1つのユニットにより実現してもよい。本願の他の実施例では、画像処理装置は他のユニットを含んでもよく、実際の応用では、これらの機能は他のユニットの協力によって実現することができ、複数のユニットの協力によって実現することができる。

10

【0075】

本願の別の実施例によれば、中央処理装置（CPU）、ランダムアクセスメモリ（RAM）、読み取り専用メモリ（ROM）などの処理要素及び記憶要素を含む、汎用コンピューティング機器（例えば、コンピュータなど）上で、図2又は図4に示される対応する方法に関する各ステップを実行可能なコンピュータプログラム（プログラムコードを含む）を実行することにより、図6に示される画像処理装置を構築することができ、これにより、本願実施例に係る画像処理方法を実現することができる。前記コンピュータプログラムは、例えば、コンピュータ可読記憶媒体に記憶され、コンピュータ可読記憶媒体を介して上記のコンピューティング機器に搭載され、当該コンピューティング機器で実行されることができる。

20

【0076】

本願実施例では、目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、操作イベントに対応する目標シェーダコードを取得した後、目標シェーダコードのコンパイルプロセスをスキップして、ゲーム鏡像又は目標コンテナと目標コンテナの関連コンテナとの間の目標共有ディレクトリから目標シェーダコードの目標コンパイル結果を直接に取得することができる。その後、目標コンパイル結果に基づいて図形処理を実行して、操作イベントに対応するフィードバック画像を取得し、目標ゲームクライアントにフィードバック画像を表示することができる。ここから分かるように、本願実施例は、ゲーム鏡像又は目標共有ディレクトリ内の、関連コンテナによってコンパイルされた目標コンパイル結果を直接使用して図形処理を実行することで、目標シェーダコードのコンパイル時間を効果的に短縮することができ、これにより、目標ゲームの実行速度を向上させることができる。

30

【0077】

上記の方法の実施例及び装置の実施例の説明に基づき、本願実施例は、サーバを更に提供し、当該サーバは、上記の目標エッジサーバであってもよい。図7を参照すると、当該サーバは少なくとも、プロセッサ701と、入力インターフェース702と、出力インターフェース703と、コンピュータ記憶媒体704と、を備える。ここで、サーバにおけるプロセッサ701、入力インターフェース702、出力インターフェース703及びコンピュータ記憶媒体704は、バス又は他の方式で接続することができる。

40

【0078】

前記コンピュータ記憶媒体704は、プログラム及びデータを記憶するように構成される、サーバ内のストレージ機器である。理解できることとして、ここでのコンピュータ記憶媒体704は、サーバ内の内部記憶媒体だけでなく、サーバによってサポートされる拡張記憶媒体も含むことができる。コンピュータ記憶媒体704は、記憶空間を提供し、当該記憶空間にはサーバの操作システムが記憶されている。更に、当該記憶空間には、プロセッサ701によってロードされて実行されることができる1つ又は複数の命令が更に記憶され、これらの命令は、1つ又は複数のコンピュータプログラム（プログラムコードを含む）であってもよい。留意されたいこととして、ここでのコンピュータ記憶媒体は、高

50

速RAMメモリであってもよいし、不揮発性メモリ（non-volatile memory）（例えば、少なくとも1つの磁気ディスクメモリなど）であってもよいし、上記のプロセッサから離れて配置された少なくとも1つのコンピュータ記憶媒体であってもよい。前記プロセッサ701（中央処理装置（Central Processing Unit、CPUと略称））は、サーバのコンピューティングコア及び制御コアであり、前記プロセッサ701は、1つ又は複数の命令を実現することができ、具体的には、1つ又は複数の命令をロードして実行することにより、対応する方法のプロセス又は対応する機能を実現することができる。

【0079】

1つの実施例において、プロセッサ701によりコンピュータ記憶媒体704に記憶された1つ又は複数の命令をロードして実行することにより、上記の図2又は図4に示される画像処理方法の実施例における対応する方法のステップを実現することができ、特定の実現では、プロセッサ701は、コンピュータ記憶媒体704内の1つ又は複数の命令をロードして、以下のステップを実行し、前記ステップは、

目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、前記操作イベントに対応する目標シェーダコードを取得するステップであって、前記操作イベントは、前記目標コンテナに接続された目標ゲームクライアントに表示される前記目標ゲームのゲーム画面から検出されたものである、ステップと、

ゲームリソースから、前記目標シェーダコードの目標コンパイル結果を取得するステップであって、前記ゲームリソースは、前記目標ゲームのゲーム鏡像、及び前記目標コンテナと前記目標コンテナの関連コンテナとの間の目標共有ディレクトリのうちの少なくとも1つを含み、前記目標共有ディレクトリは、前記関連コンテナがゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果を含む、ステップと、

前記目標コンパイル結果に基づいて図形処理を実行して、前記操作イベントに対応するフィードバック画像を取得し、前記目標ゲームクライアントに前記フィードバック画像を表示するステップと、を含む。

【0080】

1つの実施形態において、前記ゲームリソースは、前記ゲーム鏡像及び前記目標共有ディレクトリを含み、前記ゲーム鏡像は、前記目標ゲームを事前に実行するときにコンパイルされた少なくとも1つのシェーダコードの第2コンパイル結果を含み、各第2コンパイル結果は、前記ゲーム鏡像を作成するときに前記ゲーム鏡像に追加されたものであり、これに対応して、从ゲームリソースから前記目標シェーダコードの目標コンパイル結果を取得するときに、プロセッサ701は、前記1つ又は複数の命令をロードして、以下のステップを実行し、前記ステップは、

前記目標ゲームのゲーム鏡像において、前記目標シェーダコードの第2コンパイル結果の有無を検出するステップと、

前記目標シェーダコードの第2コンパイル結果がある場合、前記ゲーム鏡像から、前記目標シェーダコードの第2コンパイル結果を目標コンパイル結果として取得するステップと、

前記目標シェーダコードの第2コンパイル結果がない場合、前記目標コンテナと前記目標コンテナの関連コンテナとの間の目標共有ディレクトリから、前記第1コンパイル結果を前記目標シェーダコードの目標コンパイル結果として取得するステップと、を含む。

【0081】

更に別の実施形態において、前記ゲーム鏡像は、前記少なくとも1つのシェーダコードのインデックス値を更に含み、1つのインデックス値は1つの第2コンパイル結果に対応し、これに対応して、前記目標ゲームのゲーム鏡像において、前記目標シェーダコードの第2コンパイル結果の有無を検出するときに、プロセッサ701は、前記1つ又は複数の命令をロードして、以下のステップを実行し、前記ステップは、

前記目標シェーダコードに対してハッシュ演算を実行して、前記目標シェーダコードの

10

20

30

40

50

目標インデックス値を取得するステップと、

前記ゲーム鏡像が前記目標インデックス値を含む場合、前記ゲーム鏡像に前記目標シェーダコードの第2コンパイル結果があると決定するステップと、

前記ゲーム鏡像が前記目標インデックス値を含まない場合、前記ゲーム鏡像に前記目標シェーダコードの第2コンパイル結果がないと決定するステップと、を含む。

【0082】

更に別の実施形態において、前記目標ゲームは、クラウドゲームシステムにおけるクラウドゲームである、前記クラウドゲームシステムは、少なくとも1つのエッジサーバ及び複数のゲームクライアントを含み、各エッジサーバには少なくとも1つのシステムコンテナが展開され、各システムコンテナは少なくとも1つのゲームクライアントに接続され、

前記各システムコンテナは1つ又は複数のゲームを実行するために使用され、前記各システムコンテナは、任意のゲームを実行する過程で、前記任意のゲームのゲーム画面を、前記各システムコンテナに接続されたゲームクライアントに伝送して表示し、

ここで、前記目標コンテナは、目標エッジサーバのサーバ操作システムに基づいて実行される任意のシステムコンテナであり、前記目標エッジサーバは、前記クラウドゲームシステムの任意のエッジサーバであり、前記関連コンテナは、前記目標ゲームに関連付けられた関連ゲームを実行するシステムコンテナであり、前記関連ゲームは、前記目標ゲーム、又は前記目標ゲームと同じゲームエンジンを使用するゲームを含む。

【0083】

更に別の実施形態において、プロセッサ701は、前記1つ又は複数の命令をロードして、以下のステップを実行し、前記ステップは、

前記目標エッジサーバによって送信されたディレクトリマウント命令を受信するステップであって、前記ディレクトリマウント命令は前記目標共有ディレクトリを搬送する、ステップと、

前記目標コンテナ内で動的キャッシュディレクトリを指定し、前記目標共有ディレクトリと前記動的キャッシュディレクトリとの間のデータ同期を維持するように、前記目標共有ディレクトリを前記動的キャッシュディレクトリにマウントするステップと、を含み、

ここで、前記目標共有ディレクトリは、前記関連コンテナの所定のディレクトリに更にマウントされ、これによって、前記関連コンテナが前記関連ゲームを実行する過程で少なくとも1つのシェーダコードをコンパイルすることによって取得した第1コンパイル結果をキャッシュするようにする。

【0084】

更に別の実施形態において、前記動的キャッシュディレクトリは、第1環境変数によって指定され、前記ゲーム鏡像における前記第2コンパイル結果のストレージディレクトリは、第2環境変数によって指定される。

【0085】

更に別の実施形態において、前記目標共有ディレクトリは、前記サーバ操作システムが配置されたサーバホスト機器にあり、又は、

前記共有ディレクトリはネットワークファイルシステム内にあり、前記ネットワークファイルシステムと前記クラウドゲームシステムの各エッジサーバは、相互に独立し且つ前記各エッジサーバのシステムコンテナからのアクセスを許可する。

【0086】

更に別の実施形態において、前記目標共有ディレクトリが前記サーバホスト機器にある場合、前記サーバホスト機器は、前記目標共有ディレクトリ以外の他の共有ディレクトリを更に含み、前記サーバホスト機器の他の共有ディレクトリは、前記目標コンテナに関連しない、前記目標エッジサーバのシステムコンテナをマウントするために使用され、

前記目標共有ディレクトリが前記ネットワークファイルシステム内にある場合、前記ネットワークファイルシステムは、前記目標共有ディレクトリ以外の他の共有ディレクトリを更に含み、前記ネットワークファイルシステムの共有ディレクトリは、前記目標コンテナに関連しない、前記クラウドゲームシステムのシステムコンテナをマウントするために

10

20

30

40

50

使用される。

【0087】

更に別の実施形態において、前記ゲームリソースは、前記目標シェーダコードの目標コンパイル結果を含まず、これに対応して、プロセッサ701は、前記1つ又は複数の命令をロードして、以下のステップを実行し、前記ステップは、

前記ゲームリソースからの前記目標コンパイル結果の取得に失敗した場合、前記目標シェーダコードをコンパイルして前記目標コンパイル結果を取得するステップと、

前記目標コンパイル結果を前記目標共有ディレクトリに同期させるために、前記目標コンパイル結果を前記動的キャッシュディレクトリにキャッシュし、これにより、前記関連コンテナが前記目標シェーダコードをコンパイルする必要がある場合、前記目標共有ディレクトリから前記目標コンパイル結果を取得するようにするステップと、を含む。

10

【0088】

本願実施例では、目標コンテナ内で実行される目標ゲームに対する操作イベントに応答して、操作イベントに対応する目標シェーダコードを取得した後、目標シェーダコードのコンパイルプロセスをスキップして、ゲーム鏡像又は目標コンテナと目標コンテナの関連コンテナとの間の目標共有ディレクトリから目標シェーダコードの目標コンパイル結果を直接に取得することができる。その後、目標コンパイル結果に基づいて図形処理を実行して、操作イベントに対応するフィールドバック画像を取得し、目標ゲームクライアントにフィールドバック画像を表示することができる。ここから分かるように、本願実施例は、ゲーム鏡像又は目標共有ディレクトリ内の、関連コンテナによってコンパイルされた目標コンパイル結果を直接使用して図形処理を実行することで、目標シェーダコードのコンパイル時間を効果的に短縮することができ、これにより、目標ゲームの実行速度を向上させることができる。

20

【0089】

留意されたいこととして、本願の1つの態様によれば、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を更に提供し、当該コンピュータ命令はコンピュータ可読記憶媒体に記憶される。コンピュータ機器のプロセッサは、コンピュータ可読記憶媒体から当該コンピュータ命令を読み取り、プロセッサは当該コンピュータ命令を実行することにより、当該コンピュータ機器に、上記の図2又は図4に示される画像処理方法の実施例における任意の方法を実行させる。

30

【0090】

更に、理解されたいこととして、以上に開示されているのは、本願の好ましい実施例に過ぎず、本願の特許の範囲を限定するものではない。したがって、本願の特許請求の範囲で行われる同等変形も、本願の保護範囲に含まれるものとする。

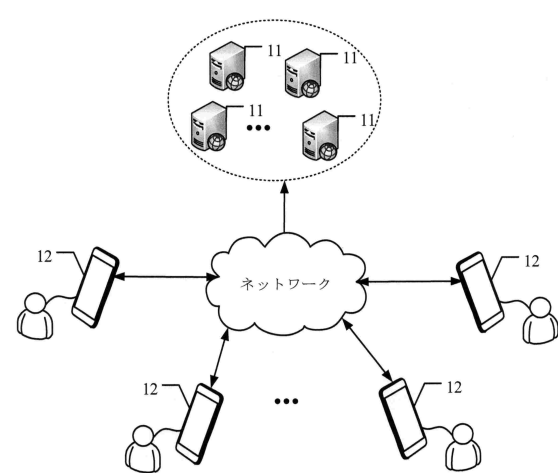
【符号の説明】

【0091】

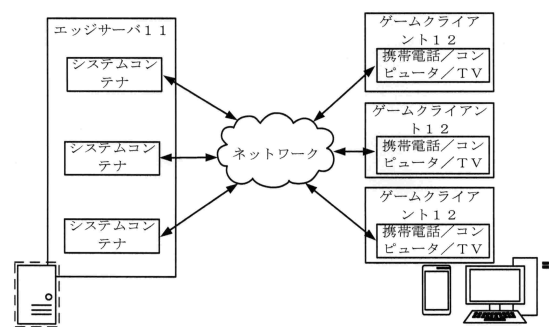
- 11 エッジサーバ
- 12 ゲームクライアント
- 601 取得ユニット
- 602 処理ユニット
- 701 プロセッサ
- 702 入力インターフェース
- 703 出力インターフェース
- 704 コンピュータ記憶媒体

40

【図面】
【図 1 a】

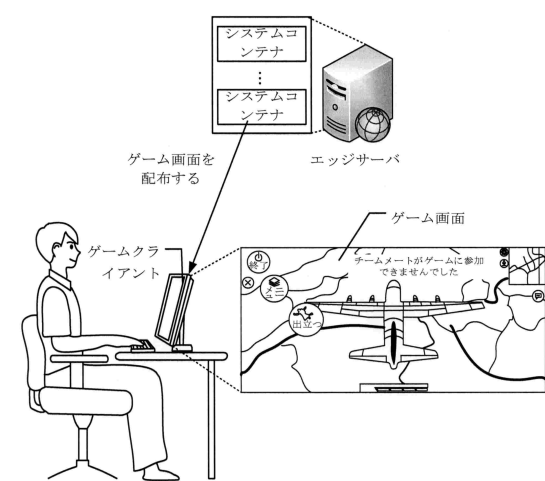


【図 1 b】

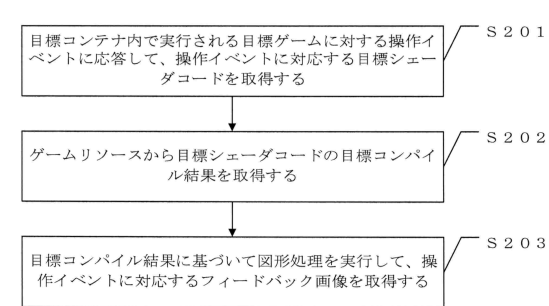


10

【図 1 c】



【図 2】



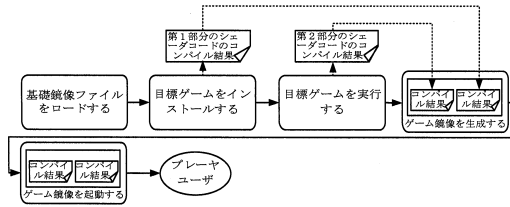
20

30

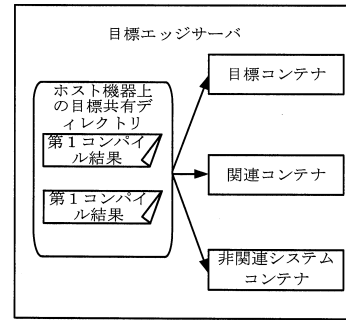
40

50

【図 3 a】

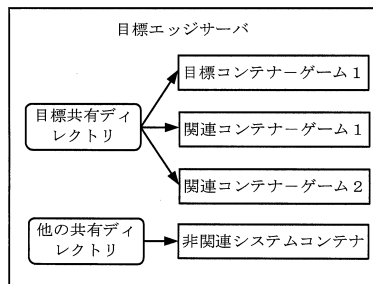


【図 3 b】

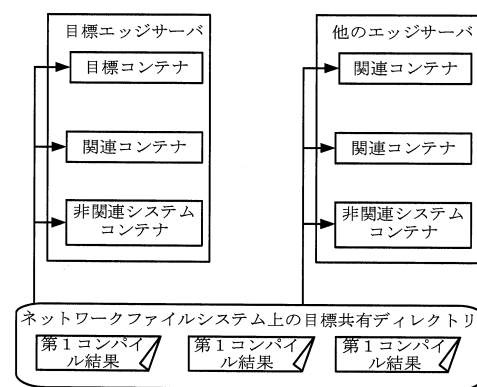


10

【図 3 c】

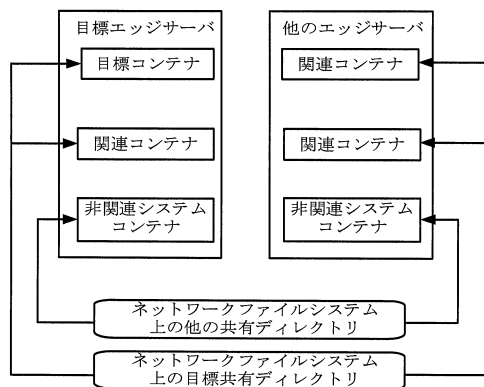


【図 3 d】

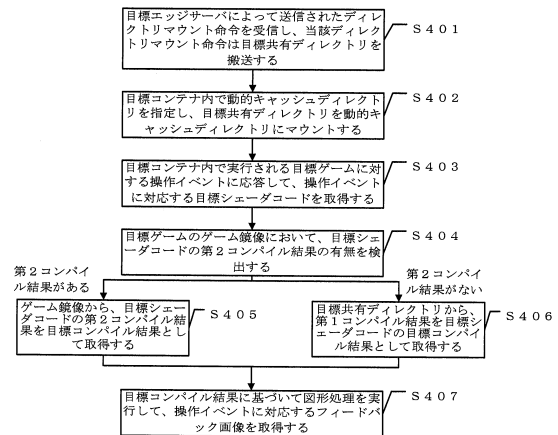


20

【図 3 e】



【図 4】

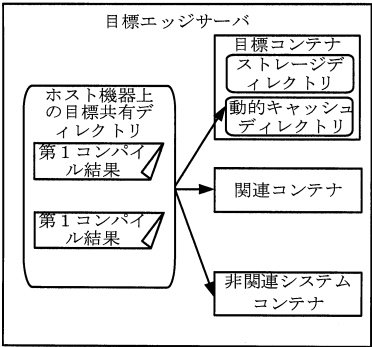


30

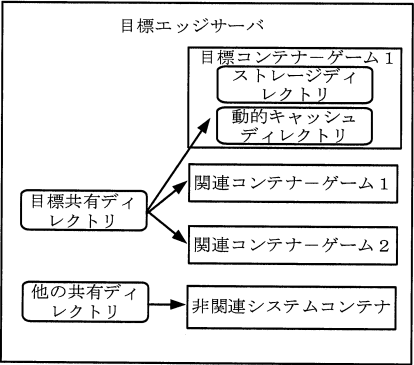
40

50

【図 5 a】

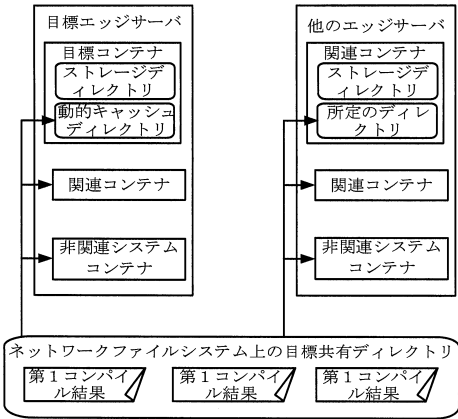


【図 5 b】

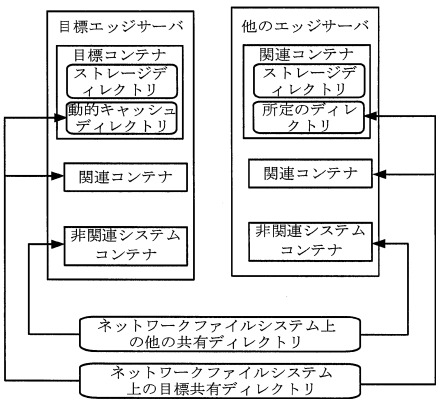


10

【図 5 c】

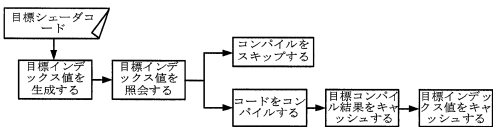


【図 5 d】

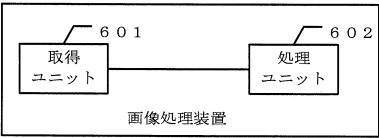


20

【図 5 e】



【図 6】

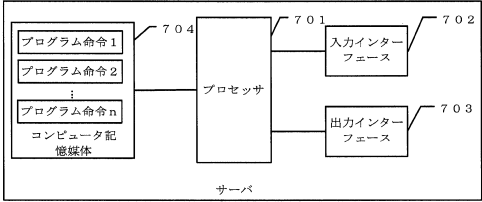


30

40

50

【図 7】



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 ▲趙▼ 新▲達▼
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 ▲ゴン▼ 志▲鵬▼
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 村松 貴士
(56)参考文献 国際公開第2019/199848 (WO, A1)
特開2008-310835 (JP, A)
米国特許出願公開第2015/0045118 (US, A1)
米国特許出願公開第2019/0308099 (US, A1)
国際公開第2019/074952 (WO, A2)
(58)調査した分野 (Int.Cl., DB名)
G06T 15/00 - 19/20
A63F 13/00 - 13/98