

图像处理方法、装置、服务器及介质

本申请要求于 2020 年 8 月 6 日提交中国专利局、申请号为 202010794044.1、发明名称为“图像处理方法、装置、服务器及介质”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

5 技术领域

本申请涉及互联网技术领域，具体涉及图像处理技术领域，尤其涉及一种图像处理方法、一种图像处理装置、一种服务器及一种计算机存储介质。

发明背景

随着互联网技术的发展，云游戏（Cloud gaming）受到了广泛关注；所谓的云游戏又可称为游戏点播（gaming on demand），是一种以云计算为基础的游戏方式，此处的云计算是一种基于互联网的计算方式。云游戏技术使图形处理与数据运算能力相对有限的客户端设备能运行高品质游戏。在云游戏场景下，游戏并不在玩家用户的游戏客户端，而是在云端服务器中运行；由云端服务器将游戏场景所涉及的游戏画面编码为视频流，通过网络传输给玩家用户的游戏客户端进行播放。玩家用户的游戏客户端无需拥有强大的图形运算与数据处理能力，仅需拥有基本的流媒体播放能力以及获取并发送玩家用户的输入指令至云端服务器的能力即可。

在云游戏场景下，云端服务器的处理资源（如 GPU（Graphics Processing Unit，图形处理器）资源）的有效利用率是一个重要的指标。因此，如何合理地利用云端服务器的处理资源成为了研究热点。

发明内容

20 本申请实施例提供了一种图像处理方法、装置、服务器及介质，可实现对服务器端的处理资源的合理化利用，提升处理资源的有效利用率。

一方面，本申请实施例提供了一种图像处理方法，所述方法由服务器执行，包括：

在运行目标云游戏的过程中，获取目标游戏客户端发送的反馈数据；所述反馈数据用于反映所述目标游戏客户端的帧率需求；

25 根据所述反馈数据对编码帧率进行动态调整，得到调整后的编码帧率；

按照所述调整后的编码帧率，对所述目标云游戏的游戏画面进行图像编码，得到编码数据；

将所述编码数据传输至所述目标游戏客户端，使所述目标游戏客户端根据所述编码数据显示所述游戏画面。

另一方面，本申请实施例提供了一种图像处理装置，所述装置包括：

30 获取单元，用于运行目标云游戏的过程中，获取目标游戏客户端发送的反馈数据；所述反馈数据用于反映所述目标游戏客户端的帧率需求；

处理单元，用于根据所述反馈数据对编码帧率进行动态调整，得到调整后的编码帧率；

所述处理单元，还用于按照所述调整后的编码帧率，对所述目标云游戏的游戏画面进行图像编码，得到编码数据；

35 传输单元，用于将所述编码数据传输至所述目标游戏客户端，使所述目标游戏客户端根据所述编码数据显示所述游戏画面。

再一方面，本申请实施例提供了一种服务器，所述服务器包括输入接口和输出接口，所述服务器还包括：

计算机存储介质，用于存储一条或多条指令；

处理器，用于加载所述计算机存储介质存储的一条或多条指令，以执行如下步骤：

在运行目标云游戏的过程中，获取目标游戏客户端发送的反馈数据；所述反馈数据用于反映所述目标游戏客户端的帧率需求；

根据所述反馈数据对编码帧率进行动态调整，得到调整后的编码帧率；

5 按照所述调整后的编码帧率，对所述目标云游戏的游戏画面进行图像编码，得到编码数据；

将所述编码数据传输至所述目标游戏客户端，使所述目标游戏客户端根据所述编码数据显示所述游戏画面。

再一方面，本申请实施例提供了一种计算机存储介质，所述计算机存储介质存储有一条或多条指令，所述一条或多条指令适于由处理器加载并执行如下步骤：

10 在运行目标云游戏的过程中，获取目标游戏客户端发送的反馈数据；所述反馈数据用于反映所述目标游戏客户端的帧率需求；

根据所述反馈数据对编码帧率进行动态调整，得到调整后的编码帧率；

按照所述调整后的编码帧率，对所述目标云游戏的游戏画面进行图像编码，得到编码数据；

15 将所述编码数据传输至所述目标游戏客户端，使所述目标游戏客户端根据所述编码数据显示所述游戏画面。

附图简要说明

为了更清楚地说明本申请实施例的技术方案，下面将对实施例描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图是本申请的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

20 图 1a 是本申请实施例提供的一种云游戏系统的系统架构图；

图 1b 是本申请实施例提供的一种边缘服务器与多个游戏客户端之间的连接示意图；

图 1c 是本申请实施例提供的一种边缘服务器和游戏客户端的结构示意图；

图 1d 是本申请实施例提供的一种边缘服务器和游戏客户端进行图像处理的示意图；

图 1e 是本申请实施例提供的另一种边缘服务器和游戏客户端的结构示意图；

25 图 1f 是本申请实施例提供的一种将目标帧率的帧率信息传输至编码模块和渲染模块的示意图；

图 1g 是本申请实施例提供的一种边缘服务器传输游戏画面至游戏客户端进行显示的示意图；

图 2 是本申请实施例提供的一种图像处理方法的流程示意图；

30 图 3 是本申请另一实施例提供的一种图像处理方法的流程示意图；

图 4a 是本申请实施例提供的一种基于两种编码方式进行帧率调整的示意图；

图 4b 是本申请实施例提供的一种在按需编码方式下进行帧率调整的原理示意图；

图 4c 是本申请实施例提供的一种在固定采样周期编码方式下执行编码操作的逻辑示意图；

图 5 是本申请实施例提供的一种图像处理装置的结构示意图；

35 图 6 是本申请实施例提供的一种服务器的结构示意图。

实施方式

下面将结合本申请实施例中的附图，对本申请实施例中的技术方案进行清楚、完整地描述。

为了更好地运行云游戏，本申请实施例提出了一种云游戏环境。在该云游戏环境下，可通过运行系统容器的方式，将多个操作系统运行在一个独立的服务器（如采用 ARM /x86 等架构的服务器）上，并通过视频流的方式将相关图像传递至远端接收程序进行处理。其中，ARM 架构是一种 32 位/或 64 位精简指令集的处理架构，x86 架构（The X86 architecture）是微处理器执行的计算机语言指令集。容器是指操作系统级虚拟化的一种类型，容器可用于承载操作系统；其通过隔离机制（例如命名空间（namespace））可实现：在内核态，多个操作系统（即服务器操作系统和设备操作系统）共用同一内核；在用户态，多个操作系统保持相互独立。此处的服

40

务器操作系统是指服务器内的通用操作系统，如 Linux 操作系统等等；设备操作系统是指容器内的操作系统，如安卓（Android）操作系统、IOS 操作系统等。

相应的，系统容器是指容器的一种实例，其可基于服务器操作系统（如 Linux 操作系统）运行；例如该系统容器可以是在已开源的 Linux 操作系统上运行的 Android 容器，一个 Linux 操作系统上可同时运行多个 Android 容器。Android 容器加载的是 Android 镜像，所谓的镜像是一种文件存储形式；通过镜像将多个文件合并成一个镜像文件，可便于文件的分发和使用。应理解的是，本申请实施例所提及的系统容器并不局限于 Android 容器；例如，若 IOS 操作系统支持开源研发，则该系统容器还可以是 IOS 容器，等等。可见，在本申请实施例所提出的云游戏环境下，可通过在一台独立的服务器上部署大量的系统容器，充分利用服务器端强大的中央处理器（Central Processing Unit, CPU）能力以及图形处理器（Graphics Processing Unit, GPU）能力，实现高并发地执行系统操作，提升云游戏的运行速度。

基于上述所提及的云游戏环境，本申请实施例提出了一种云游戏系统；参见图 1a 所示，该云游戏系统可包括至少一个边缘服务器 11 以及多个游戏客户端 12。其中，边缘服务器 11 是指用于运行系统容器的服务器；该服务器可以是独立的任一物理服务器，也可以是提供云服务、云数据库、云计算、云函数、云存储、网络服务、云通信、中间件服务、域名服务、安全服务、内容分发网络（Content Delivery Network, CDN）、以及大数据和人工智能平台等基础云计算服务的云服务器。每个边缘服务器 11 内可部署有至少一个系统容器，每个系统容器可与至少一个游戏客户端 12 相连接，如图 1b 所示。其中，每个系统容器内可安装一个或多个游戏应用程序（application, APP），并通过安装的一个或多个游戏 APP 来运行一个或多个云游戏；且每个系统容器在运行任一云游戏的过程中，可传输任一云游戏的游戏画面至相连接的游戏客户端 12 中进行显示。并且，每个边缘服务器 11 中还可包括用于进行图像渲染的渲染模块，以及用于进行图像编码的编码模块，如图 1c 所示；该渲染模块和编码模块可位于每个系统容器的内部，或者位于每个系统容器的外部。其中，编码模块可包括：可将模拟音视频信号压缩成编码数据（或编码文件）的硬件设备/软件代码。为便于阐述，后续均以渲染模块和编码模块位于系统容器的内部，且编码模块为软件设备为例进行说明。

游戏客户端 12 可以是具有基本的流媒体播放能力、人机交互能力以及通信能力等基本能力的任一终端设备（简称终端），例如，智能手机、平板电脑、笔记本电脑、台式计算机、智能电视等；或者该游戏客户端 12 可以是运行于终端设备中的应用程序。具体的，游戏客户端 12 中还可包括用于进行图像解码的解码模块；该解码模块可包括：可将编码数据（或编码文件）转换为模拟音视频信号的硬件设备/软件代码。为便于阐述，后续均以解码模块为软件设备为例进行说明；此情况下的解码模块可位于游戏客户端 12 中的应用程序软件开发工具包（Software Development Kit, SDK）中。应理解的是，图 1a 只是示例性的表征云游戏系统的系统架构，并不对云游戏系统的具体架构进行限定；例如在其他实施例中，云游戏系统中还可包括用于调度的后台服务器，等等。

下面将基于上述的云游戏系统，对任一云游戏的具体运行过程进行阐述：

在制作完成云游戏的镜像文件后，可将该云游戏的镜像文件部署至边缘服务器中，以使得边缘服务器可通过加载该镜像文件以启动对应的系统容器。当接收到游戏客户端发送的关于云游戏的连接请求时，边缘服务器可分配相应的系统容器与该游戏客户端建立连接。在被分配的系统容器和游戏客户端成功建立连接后，被分配的系统容器可运行云游戏。并且在云游戏的运行过程中，系统容器或系统容器中的游戏 APP 可在边缘服务器端调用渲染指令发起渲染操作，以触发边缘服务器中的渲染模块通过执行渲染操作来实时渲染云游戏的游戏画面；此处的渲染指令可包括但不限于：opengl 指令、opengles 指令、vulkan 指令等。其中，opengl 是用于渲染 3D 图形的跨平台应用程序接口（Application Programming Interface, API）；opengles 是 opengl API 的一个子集，其主要运用在嵌入式设备（例如：智能手机等）中；vulkan 是另一个跨平台的 2D 和 3D 图形应用程序编程接口 API。需要说明的是，opengl、opengles 以及 vulkan 的应用程序接口均可通过 Mesa 实现；此处的 Mesa 是一个开放源代码的计算机图形库。

其次，可调用编码模块按照视频编码标准（如 H.264、H.265 等）对渲染得到的游戏画面进行图像压缩编码操作，以节省后续图像传输时占用的网络带宽；然后，可采用视频流的方式将图像压缩编码得到的编码数据（即压缩图像）传输至游戏客户端中，如图 1d 所示。相应的，游

游戏客户端在接收到编码数据后,可通过解码模块对该编码数据执行解码操作以还原出原始的游戏画面,并输出显示解码所得到的游戏画面。并且,游戏客户端在显示游戏画面的过程中,还可通过数据流的方式将玩用户针对游戏画面的用户输入事件(例如:鼠标点击事件、键盘操作事件等),发送给相应的系统容器。相应的,该相应的系统容器还可负责抓取与该操作事件相对应的渲染缓冲区图像,并将抓取到的图像返回给游戏客户端进行显示。

在上述云游戏的运行过程中,本申请实施例提出了一种基于反馈的动态帧率调整机制;此处的基于反馈的动态帧率调整机制是指:根据游戏客户端实时或周期性反馈的帧率需求,动态调整服务器端的渲染帧率和编码帧率中的至少一个帧率的机制。其中,所谓的帧率是指在单位时间(通常为1秒)内的帧数,可采用fps(frame per second)表示;此处的帧数是指帧的数量。对于渲染过程来说,渲染模块(如GPU硬件)将渲染完成的图像数据(如游戏画面)存入显示缓冲区(通常称为framebuffer),每存入一次所涉及的图像数据可代表一帧;对于编码过程来说,编码模块从显示缓冲区读取相应的图像数据,每读取一次所涉及的图像数据则代表一帧。通过本申请实施例所提出的基于反馈的动态帧率调整机制,可通过动态调整渲染帧率和/或编码帧率,实现动态调整处理资源(如GPU资源、CPU资源)的有效利用率,从而达到处理资源的有效利用率的最大化。

为了更好地实现该基于反馈的动态帧率调整机制,本申请实施例还分别在上述云游戏系统中的边缘服务器和游戏客户端中,进一步增加了一个数据模块,以用于边缘服务器和游戏客户端之间的数据反馈,如图1e所示。其中,游戏客户端中的数据模块主要负责:获取可用于反映游戏客户端的帧率需求的反馈数据;并通过网络的方式与边缘服务器中的数据模块进行通信,以将获取到的反馈数据传输至边缘服务器。相应的,边缘服务器中的数据模块主要负责:根据从游戏客户端的数据模块处所接收到的反馈数据,计算生成渲染模块和编码模块可接受且能够识别的帧率信息(如合适的帧率所对应的帧率参数、帧率指令或者帧率配置文件等);并将该帧率信息传递给渲染模块和编码模块(如图1f所示),使得渲染模块和编码模块可根据接收到的帧率信息,控制相应的代码逻辑调整相应的渲染帧率和编码帧率,并按照调整后的渲染帧率和编码帧率分别进行图像渲染和图像编码。在得到编码数据后,边缘服务器可通过系统容器将编码数据传输至游戏客户端,使得游戏客户端可根据编码数据进行游戏画面的显示,如图1g所示。

需要说明的是,图1e只是示例性地表征各模块之间的位置关系和连接关系,并不对其进行限定。例如,在图1e中,边缘服务器中的渲染模块和编码模块,与数据模块是位于不同的系统容器中的;但在实际应用中,渲染模块、编码模块和数据模块这三个模块均可位于同一个系统容器中,或者这三个模块均可位于所有系统容器的外部。又如,在图1e中,游戏客户端中的数据模块和解码模块之间未连接;但在实际应用中,游戏客户端中的数据模块也可和解码模块进行数据交互,等等。并且,还需说明的是,边缘服务器中的数据模块和游戏客户端中的数据模块之间的数据交互,可以发生在游戏客户端刚和边缘服务器中的系统容器建立连接的时刻(即云游戏运行的开始时刻),也可以发生在云游戏已运行一段时间后的任一时刻。

根据上述基于反馈的动态帧率调整机制的相关描述,本申请实施例提出一种图像处理方法;该图像处理方法可以由上述云游戏系统中的目标边缘服务器执行,该目标边缘服务器是指云游戏系统中的任一边缘服务器。请参见图2,该方法可由图6所示的服务器执行,可包括以下步骤S201-S204:

S201,在运行目标云游戏的过程中,获取目标游戏客户端发送的反馈数据。

在本申请实施例中,目标云游戏在目标容器内运行。目标容器是指目标边缘服务器中所运行的任一系统容器,在目标容器内所运行的目标云游戏可以为任一云游戏。由前述可知,在目标云游戏的运行过程中,该目标云游戏所涉及的各个游戏画面均需下发至目标游戏客户端进行显示;此处的目标游戏客户端是指与目标容器建立连接的任一游戏客户端。经研究表明,针对目标云游戏的游戏画面而言,目标游戏客户端通常存在不同的帧率需求:

例如,对于不同的硬件配置,其解码能力是不同的;那么随着解码能力的不同,目标游戏客户端的帧率需求通常是不同的。具体的,对于一定的分辨率来说(例如720p或是1080p),若目标客户端具有比较好的硬件配置(如GPU、CPU等),则目标客户端通常可具有60帧/秒(甚至更高)的解码能力;那么当目标边缘服务器端传输每秒60帧的游戏画面的解码数据到目标游

戏客户端进行解码时,其解码能力是足够的。因此对于这种硬件配置比较好的目标游戏客户端而言,其帧率需求可以是60帧/秒(甚至更高)。但是,若目标客户端具有比较差的硬件配置,则目标游戏客户端通常只有30帧/秒(甚至更低)的解码能力;那么当目标边缘服务器端传输每秒60帧的游戏画面的解码数据到目标游戏客户端进行解码时,其解码能力是不够的,容易造成部分图像的丢弃,造成不必要的浪费。因此对于这种硬件配置比较差的目标游戏客户端而言,其帧率需求可以是30帧/秒(甚至更低)。

又如,目标云游戏在目标游戏客户端中通常具有多个应用场景,如挂机场景、用户操作场景等;此处的挂机场景是指:由程序控制目标游戏客户端对应的游戏对象(目标云游戏中的虚拟人物)的场景。而对于不同的应用场景,目标游戏客户端的帧率需求也是不同的。具体的,对于挂机场景来说,目标游戏客户端侧的玩家用户使用挂机的目的通常不是为了游戏操作的流畅性;而是为了在挂机的过程中,可保证游戏自动运行即可。那么在这种情况下,目标游戏客户端使用低帧率模式即可;即在挂机场景下,目标游戏客户端的帧率需求可以是30帧/秒(甚至更低)。相反,对于非挂机场景来说,目标游戏客户端可使用高帧率模式来满足玩家用户的游戏流畅性的需求;即在非挂机场景下,目标游戏客户端的帧率需求可以是60帧/秒(甚至更高)。

再如,受限于目标游戏客户端和目标边缘服务器端之间的网络环境,目标游戏客户端的帧率需求也是不同的。对于目标游戏客户端来说,在网络拥塞的情况下,目标边缘服务器端提供30帧/秒(甚至更低)的帧率可能已是其处理极限,更高的帧率只会导致目标游戏客户端侧出现更严重的网络拥塞,进而影响目标游戏客户端的体验。那么在这种情况下,目标游戏客户端使用低帧率模式即可,此时的目标游戏客户端的帧率需求可以是30帧/秒(甚至更低)。然而在较好的网络环境下,则可以适当提高服务器端的帧率,这样可以提高目标客户端的体验。可见,较差的网络环境对应于较低的帧率需求,较好的网络环境可对应于较高的帧率需求。

再如,目标云游戏中可能存在不同用户类型的玩家用户;此处的用户类型可包括:第一类型或者第二类型。其中,第一类型所对应的游戏服务优于第二类型所对应的游戏服务;如第一类型的玩家用户可以是会员用户,第二类型的玩家用户可以是非会员用户。而对于不同用户类型的玩家用户而言,目标游戏客户端的帧率需求也可以是不同的。具体的,由于不同类型的玩家用户通常享受不同的游戏体验;如可为第一类型的玩家用户(如会员用户)提供60帧/秒的帧率,为第二类型的玩家用户(如非会员用户)提供30帧/秒的帧率。那么相应的,若目标游戏客户端是由第一类型的玩家用户使用的,则目标游戏客户端的帧率需求可以是60帧/秒;若目标游戏客户端是由第二类型的玩家用户使用的,则目标游戏客户端的帧率需求可以是30帧/秒,等等。

综上所述,目标边缘服务器可通过考虑目标游戏客户端的不同帧率需求,针对性地为目标游戏客户端提供不同帧率的游戏画面的编码数据,以满足目标游戏客户端的不同帧率需求,从而使得目标游戏客户端能够更好地进行游戏画面的显示。基于此,在运行目标云游戏的过程中,目标边缘服务器可实时或周期性地获取目标游戏客户端发送的、用于反映目标游戏客户端的帧率需求的反馈数据,以便于后续可基于该反馈数据对相关帧率(如编码帧率、渲染帧率等)进行动态调整,从而按照动态调整后的帧率为目标客户端提供相应的编码数据。具体的,反馈数据可指示以下至少一项内容:目标游戏客户端的硬件配置、目标云游戏在目标游戏客户端中的应用场景、目标游戏客户端和目标边缘服务器之间的网络情况、以及该目标游戏客户端对应的玩家用户的用户类型,等等;通过反馈数据所指示的内容,便可确定目标游戏客户端的帧率需求。

S202,根据反馈数据对编码帧率进行动态调整,得到调整后的编码帧率。

由前述可知:在目标边缘服务器端,由于渲染操作之后需要进行相应的编码操作,以节省目标边缘服务器和目标游戏客户端之间进行数据传输时所占用的网络带宽。因此,在基于反馈数据对相关帧率进行动态调整的过程中,可先通过编码模块根据反馈数据对编码帧率进行动态调整,得到调整后的编码帧率。需要说明的是,若步骤S202是在目标云游戏开始运行的时刻执行的,则此处的编码帧率可以是指启动目标容器前,提前为目标容器设置的默认编码帧率;若步骤S202是在目标云游戏已运行一段时间后执行的,则此处的编码帧率可以是指当前运行目标云游戏所使用的当前编码帧率。

具体的,若反馈数据所指示的目标游戏客户端的硬件配置较低(如CPU型号比较旧,即GPU型号早于某个年代或某个版本的型号)、或者反馈数据所指示的目标云游戏在目标游戏客户

端中的应用场景为挂机场景,或者反馈数据所指示的目标游戏客户端和目标边缘服务器之间的网络情况比较拥塞,或者反馈数据所指示的用户类型为第二类型(即目标游戏客户端的玩家用户是第二类型的用户(如非会员用户)),则可通过编码模块对编码帧率进行动态调整,使得调整后的编码帧率为一个较小的帧率(如30帧/秒,或者更小)。

- 5 若反馈数据所指示的目标游戏客户端的硬件配置较高(如CPU型号比较新,即GPU型号晚于某个年代或某个版本的型号)、或者反馈数据所指示的目标云游戏在目标游戏客户端中的应用场景为非挂机场景(即正常使用场景),或者反馈数据所指示的目标游戏客户端和目标边缘服务器之间的网络情况比较好,或者反馈数据所指示的用户类型为第一类型(即目标游戏客户端的玩家用户是第一类型的用户(如会员用户)),则可通过编码模块对编码帧率进行动态调整,
- 10 使得调整后的编码帧率为一个较大的帧率(如60帧/秒,或者更大)。

S203,按照调整后的编码帧率,对目标云游戏的游戏画面进行图像编码,得到编码数据。

S204,将编码数据传输至目标游戏客户端。

- 在步骤S203-S204中,在得到调整后的编码帧率后,目标边缘服务器便可控制编码模块按照调整后的编码帧率,对目标云游戏的游戏画面进行图像编码,得到编码数据。然后,可将编码数据传输至目标游戏客户端,使目标游戏客户端根据编码数据显示游戏画面。相应的,目标游戏客户端在接收到该编码数据后,可对该编码数据进行解码以得到目标云游戏的游戏画面;
- 15 然后,在用户界面中显示该目标云游戏的游戏画面。

- 本申请实施例在运行目标云游戏的过程中,可获取目标游戏客户端发送的反馈数据。由于该反馈数据可反映目标游戏客户端的帧率需求,因此服务器可考虑该反馈数据所反映的帧率需求,对编码帧率进行动态调整,并按照调整后的编码帧率对目标云游戏的游戏画面进行图像编码。这样可使得服务器端在图像编码过程中所占用的处理资源均是有价值的,减少非必要的处理资源的占用;从而实现对服务器端的处理资源的合理化利用,提升处理资源的有效利用率。
- 20 在通过图像编码得到编码数据后,可将图像编码所得到的编码数据传输至目标游戏客户端;由于编码数据是通过考虑目标游戏客户端的帧率需求进行编码得到的,因此编码数据所对应的帧率可较好地适配于目标游戏客户端,使得目标游戏客户端可较好地根据编码数据显示游戏画面。

- 25 基于上述图2所示的图像处理方法的实施例的相关描述,本申请实施例还提出一种更为具体的图像处理方法的流程示意图;该图像处理方法可以由上述云游戏系统中的目标边缘服务器执行,该目标边缘服务器是指云游戏系统中的任一边缘服务器。在本申请实施例中,目标云游戏的游戏画面均是由渲染模块渲染得到的,图像编码是由编码模块执行的;请参见图3,该图像处理方法可包括以下步骤S301-S306:

S301,在运行目标云游戏的过程中,获取目标游戏客户端发送的反馈数据。

- 在本申请实施例中,目标云游戏可运行在目标容器中,目标游戏客户端与目标容器相连接。反馈数据可包括至少一个反馈指标下的反馈信息;此处的反馈指标可包括但不限于:硬件配置指标、应用场景指标、网络状况指标以及用户类型指标。具体的:

- 35 硬件配置指标下的反馈信息是指目标游戏客户端的硬件配置信息,其可包括以下至少一项:目标游戏客户端对应的图形处理器(即GPU)的硬件信息,以及目标游戏客户端对应的中央处理器(即CPU)的硬件信息。具体的,目标游戏客户端对应的图形处理器的硬件信息可包括但不限于:目标游戏客户端对应的图形处理器的厂商标识和型号标识;目标游戏客户端对应的中央处理器的硬件信息可包括但不限于:目标游戏客户端对应的中央处理器的厂商标识和型号标识。
- 40

应用场景指标下的反馈信息是指目标云游戏在目标游戏客户端中的应用场景信息,其可包括但不限于:目标云游戏在目标游戏客户端中的应用场景的场景标识。其中,应用场景可包括挂机场景或者非挂机场景。

- 网络状况指标下的反馈信息是指目标游戏客户端的网络状况信息,其可包括以下至少一项:网络带宽、网络抖动信息以及网络丢包率。其中,网络带宽是指在单位时间(一般指的是1秒)内传输的数据量;网络带宽越大,则表明网络环境越好。网络抖动是指数据传输所涉及的最大延迟和最小延迟之间的时间差,所谓的延迟是指数据的发送时刻与接收时刻之间的时长;网络抖动越小,则表明网络环境越稳定。网络丢包率是指数据包丢失部分与所传数据包总数的比值,网络丢包率越小,则表明网络环境越稳定。
- 45

用户类型指标下的反馈信息是指目标游戏客户端对应的用户类型信息，其可包括：登陆目标云游戏的用户账号的用户类型的类型标识。其中，用户类型包括：第一类型（如会员）或者第二类型（如非会员）。

在具体实施过程中，目标边缘服务器和目标游戏客户端之间可通过自定义协议进行数据交互。其中，自定义协议中可包括各反馈指标对应的协议变量；例如，硬件配置指标对应的 cloudgame\_cpu\_vendor（用于记录目标游戏客户端对应的 CPU 厂家标识的协议变量），cloudgame\_cpu\_device\_id（用于记录目标游戏客户端对应的 CPU 型号标识的协议变量），cloudgame\_gpu\_vendor（用于记录目标游戏客户端对应的 GPU 厂家标识的协议变量），cloudgame\_gpu\_device\_id（用于记录目标游戏客户端对应的 GPU 型号标识的协议变量）；网络状况指标对应的 cloudgame\_network\_dither（用于记录网络抖动信息的协议变量），cloudgame\_network\_loss\_rate（用于记录网络丢包率的协议变量），用户类型指标对应的 cloudgame\_user\_type（用于记录类型标识的协议变量），应用场景指标对应的 cloudgame\_scene\_id（用于记录场景标识的协议变量），等等。那么在目标容器内运行目标云游戏的过程中，目标游戏客户端可获取各反馈指标下的反馈信息，并将获取到的反馈信息赋值给相应的协议变量以得到反馈数据。然后，将该反馈数据传输至目标边缘服务器。相应的，在步骤 S301 的具体实现中，目标边缘服务器可接收目标游戏客户端传输的反馈数据。

S302，根据反馈数据对编码帧率进行动态调整，得到调整后的编码帧率。

在一种具体实现中，步骤 S302 的具体实施方式可以是：先获取反馈数据中的各反馈指标下的反馈信息对应的参考帧率。具体的，可先根据目标边缘服务器和目标游戏客户端之间约定的自定义协议，解析该反馈数据以得到各反馈指标下的反馈信息（即各协议变量的取值）。然后，可获取各反馈指标下的反馈信息对应的参考帧率。其中，一个反馈指标对应一个帧率查询表，该帧率查询表中包括多个信息以及各信息对应的帧率；例如，硬件配置指标对应的帧率查询表中可包括多个硬件配置信息（如 CPU 型号、GPU 型号等）以及各硬件配置信息对应的帧率；网络状况指标对应的帧率查询表中可包括多个网络抖动信息对应的帧率，以及多个网络丢包率对应的帧率，等等。那么，相应的，反馈数据中的任一反馈指标下的反馈信息对应的参考帧率，可以从任一反馈指标对应的帧率查询表中查询到的。

其次，可根据各反馈指标下的反馈信息对应的参考帧率计算得到目标帧率。具体的，可从各反馈指标下的反馈信息对应的参考帧率中，选取最小参考帧率作为目标帧率。或者，对各反馈指标下的反馈信息对应的参考帧率进行均值运算，得到目标帧率。或者，获取各反馈指标的权重值，并采用各反馈指标的权重值，对各反馈指标下的反馈信息对应的参考帧率进行加权求和，得到目标帧率。或者，还可确定各反馈指标的优先级，并选取优先级最高的反馈指标下的反馈信息对应的参考帧率作为目标帧率，等等。需要说明的是，本申请实施例只是示例性地列举了几种目标帧率的计算方式，并非穷举。例如，在其他实施例中，还可根据各反馈指标下的反馈信息对应的参考帧率计算得到中间帧率；此处的中间帧率可包括但不限于：各反馈指标下的反馈信息对应的参考帧率中的最小参考帧率，各反馈指标下的反馈信息对应的参考帧率的均值，采用反馈指标的权重值，对各反馈指标下的反馈信息对应的参考帧率进行加权求和所得到的帧率，优先级最高的反馈指标下的反馈信息对应的参考帧率，等等。然后，求取中间帧率和当前的编码帧率之间的均值，得到目标帧率。采用这种计算方式，可减小目标帧率和当前的编码帧率之间的差异，使得后续可较为平滑地采用目标帧率进行帧率调整。进一步的，在求取中间帧率和当前的编码帧率之间的均值之前，可先判断中间帧率和当前的编码帧率之间的差值是否大于差值阈值；若大于，则可执行求取中间帧率和当前的编码帧率之间的均值的步骤；若不大于，则表明中间帧率和当前的编码帧率之间的差异不大，因此可直接将中间帧率作为目标帧率。

然后，可按照编码方式采用目标帧率对编码帧率进行动态调整，得到调整后的编码帧率，参见图 4a 所示；此处的编码方式可以为按需编码方式，或者固定采样周期编码方式。其中，按需编码方式是指：编码模块按照渲染模块的渲染进度进行编码的方式；固定采样周期编码方式是指：编码模块按照固定的采样周期对渲染模块所渲染的游戏画面进行编码的方式。随着编码方式的不同，采用目标帧率对编码帧率进行动态调整的具体实施方式也随之不同，具体参见下述描述：

### (一) 编码方式为按需编码方式

由于在按需编码方式下, 当一帧游戏画面渲染结束后, 渲染模块会通知编码模块进行一次图像编码操作, 再由目标边缘服务器将编码模块执行图像编码操作后的编码数据传送到目标游戏客户端。因此, 在这种情况下, 编码模块的编码帧率是由渲染模块的渲染帧率决定的; 那么

5 为了满足目标游戏客户端的帧率需求, 可通过调整渲染模块的渲染帧率来达到调整编码帧率的目的。而经研究表明, 渲染模块的渲染过程通常是基于垂直同步信号量 (vsync) 机制进行同步的。以 Android 系统为例, 渲染模块的渲染过程的同步原理大致如下: 可根据 Android 系统中的硬件垂直信号量 (硬件 vsync), 衍生出两个软件垂直信号量 (软件 vsync); 其分别是: 可用于

10 触发图层数据准备的第一软件 vsync (如软件 app vsync), 以及用于触发图层合并工作的第二软件 vsync (软件 surfaceflinger vsync)。当数据准备模块 (如 Choreographer 模块) 收到第一软件 vsync (如软件 app vsync) 时, 可开始准备图层数据; 当图层合并模块 (如 surfaceflinger 模块) 收到第二软件 vsync (如软件 surfaceflinger vsync) 时, 可开始图层的合并工作, 以使得渲染模块可执行后续的渲染操作。由此可见, 软件 vsync 的产生周期可决定渲染帧率的大小; 并且, 软件 vsync 是基于硬件垂直信号量生成的, 且软件 vsync 可用于触发画面渲染。因此, 可通过调

15 整软件 vsync (如软件 app vsync 和软件 surfaceflinger vsync) 的产生周期, 来实现渲染帧率的调整。

由于图像的刷新渲染通常可包含几个步骤: preComposition (预合成)、doComposition (合成)、postComposition (已合成)。在 preComposition 阶段, 主要对图层的合成进行准备工作; 在 doComposition 阶段, 主要处理图层的合成工作; 在 postComposition 阶段, 主要做一些收尾工作, 例如判断是不是需要和硬件 vsync 同步以修正软件 vsync 的偏离。因此可在 postComposition

20 阶段, 利用这个同步的机会, 实施对 vsync 的干预, 以实现渲染帧率的调整。具体的干预原理可参见图 4b 所示: 可通过更改操作系统中的硬件合成器 (HWComposer) 模块中的硬件 vsync 的产生周期并触发软件 vsync 和硬件 vsync 进行同步操作, 使得相应的软件信号源模块 (如 DispSync 模块) 在计算 vsync 偏移量后, 并根据计算得到的 vsync 偏移量产生软件 vsync (如软件 app vsync 和软件 surfaceflinger vsync) 时, 可使得软件 vsync 的产生周期也做出相应调整, 从而使得渲染帧率得到改变, 进而使得编码帧率的动态调整。

基于此, 按照编码方式采用目标帧率对编码帧率进行动态调整, 得到调整后的编码帧率的具体实施方式可以是: 先根据目标帧率调整硬件垂直信号量的产生周期, 得到硬件垂直信号量 (即硬件 vsync) 的调整后的产生周期。具体的, 可求取目标帧率的倒数, 并按照求取的倒数调整

30 HWComposer 模块的硬件垂直信号量的产生周期, 使得硬件垂直信号量的调整后的产生周期为该目标帧率的倒数; 例如, 目标帧率为 60 帧/秒, 则硬件垂直信号量的调整后的产生周期可为 1/60。其次, 可通过硬件垂直信号量的调整后的产生周期, 同步调整软件垂直信号量 (即软件 vsync) 的产生周期; 并采用软件垂直信号量的调整后的产生周期, 调整渲染模块的渲染帧率。然后, 可基于调整后的渲染帧率对编码帧率进行动态调整, 得到调整后的编码帧率; 其中, 调整后的编码帧率等于调整后的渲染帧率。

35

### (二) 编码方式为固定采样周期编码方式

由于在固定采样周期编码方式下, 编码模块是按照固定的编码采样周期对获取到的游戏画面进行编码操作的。即参见图 4c 所示, 编码模块会实时检测编码采样周期是否到达; 每检测到编码采样周期到达, 则可执行一次编码操作。因此, 在这种情况下, 编码模块的编码帧率与

40 渲染模块的渲染帧率是相互独立的。基于此, 按照编码方式采用目标帧率对编码帧率进行动态调整, 得到调整后的编码帧率的具体实施方式可以是: 采用目标帧率更新编码帧率, 得到调整后的编码帧率; 即调整后的编码帧率等于所述目标帧率。需要说明的是, 此处所提及的采用目标帧率更新编码帧率是指: 根据目标帧率生成编码模块能够识别且接受的帧率信息, 并将该帧率信息传递至编码模块中, 由编码模块在根据该帧率信息识别得到目标帧率后, 采用目标帧率

45 更新编码帧率, 得到调整后的编码帧率。其中, 帧率信息可包括但不限于: 帧率参数 (如 Android 系统中的 property 值)、帧率指令或者帧率配置文件, 等等。

还需要说明的是, 在编码模块采用目标帧率更新编码帧率时, 可直接采用目标帧率替换该编码帧率, 从而实现帧率更新; 例如, 目标帧率为 60 帧/秒, 当前的编码帧率为 30 帧/秒, 那么可直接将当前的编码帧率从 30 帧/秒更改成 60 帧/秒, 从而实现帧率更新。或者, 也可根据目标

帧率在目标边缘服务器端设定相应的编码采样周期,从而实现帧率更新。例如,当目标帧率为 60 帧/秒时,可将编码采样周期重新设置为 1/60 秒;由于每采样一次便会对采样得到的游戏画面进行一次图像编码操作,因此编码模块便可在 1 秒内编码 60 帧游戏画面,从而使得编码模块的目标帧率为 60 帧/秒。同理,当目标帧率为 30 帧/秒时,可将编码采样周期重新设置为 1/30 秒即可。

5 对于固定采样周期编码方式来说,目标边缘服务器还可根据实际的使用需求,决定是否通过调整渲染帧率来进一步提高处理资源(如 GPU 资源)的有效使用率。例如,在目标云游戏在目标游戏客户端中的应用场景为挂机场景的情况下,由于目标游戏客户端的玩家用户的目的不是为了操作的流畅性,而是在挂机的过程中,保证目标云游戏自动运行即可。因此,在这种情况下,使用低帧率模式即可,例如将编码帧率从 60 帧/秒降低到 15 帧/秒(甚至更低)。那么,在这种场景下,目标边缘服务器便可利用上述所提及的渲染帧率的调整方法,通过调整 vsync 的产生周期来进一步降低渲染模块的渲染帧率,从而使得处理资源的使用更有价值,进而提升处理资源的有效利用率。当然,应理解的是:由于鼠标点击、触摸屏触摸等用户输入事件也和 vsync 机制相关;如果延长软件 vsync 的产生周期,那么可能会影响到目标容器对用户输入事件的响应速度。因此在考虑用户的操作流畅性的用户需求下,也可只调整编码模块的编码帧率,对于渲染模块的渲染帧率则不进行调整,从而使得游戏画面的流畅性和用户操作的流畅性都可以得到满足。

再一种具体实现中,步骤 S302 的具体实施方式还可以是:根据反馈数据中的各反馈指标下的反馈信息,确定帧率调整方向;其中,帧率调整方向可包括:帧率增大的方向,或者帧率减小的方向。具体的,可先根据目标边缘服务器和目标游戏客户端之间约定的自定义协议,解析该反馈数据以得到各反馈指标下的反馈信息(即各协议变量的取值)。然后,可获取各反馈指标下的反馈信息,确定帧率调整方向。例如,若根据硬件配置指标下的反馈信息中的 GPU 的厂家标识(即协议变量 cloudgame\_gpu\_vendor 的取值)和反馈信息中的 GPU 的型号标识(即协议变量 cloudgame\_gpu\_device\_id 的取值)确定目标游戏客户端的型号晚于某个年代或是某个版本,或是应用场景指标下的反馈信息中的场景标识(即协议变量 cloudgame\_scene\_id 的取值)为非挂机场景的场景标识,或是网络状况指标下的反馈信息中的网络抖动信息(即协议变量 cloudgame\_network\_dither 的取值)或网络丢包率(即协议变量 cloudgame\_network\_loss\_rate 的取值)位于预设数值范围内,或是用户类型指标下的反馈信息中的类型标识(即协议变量 cloudgame\_user\_type 的取值)为第一类型的类型标识,则可确定帧率调整方向为帧率增大的方向。

30 若根据硬件配置指标下的反馈信息中的 GPU 的厂家标识(即协议变量 cloudgame\_gpu\_vendor 的取值)和反馈信息中的 GPU 的型号标识(即 cloudgame\_gpu\_device\_id 的取值)确定目标游戏客户端的型号早于某个年代或是某个版本,或是应用场景指标下的反馈信息中的场景标识(即协议变量 cloudgame\_scene\_id 的取值)为挂机场景的场景标识,或是网络状况指标下的反馈信息中的网络抖动信息(即协议变量 cloudgame\_network\_dither 的取值)或网络丢包率(即协议变量 cloudgame\_network\_loss\_rate 的取值)位于预设数值范围外,或是用户类型指标下的反馈信息中的类型标识(即协议变量 cloudgame\_user\_type 的取值)为第二类型的类型标识,则可确定帧率调整方向为帧率减小的方向。

40 然后,可获取帧率调整方向所对应的帧率调整因子,该帧率调整因子可以是根据业务需求或者经验值设置的;并且,帧率增大的方向所对应的帧率调整因子为正数,帧率减小的方向所对应的帧率调整因子为负数。然后,可采用帧率调整因子对编码帧率进行动态调整,得到调整后的编码帧率。由前述可知,编码模块的编码方式可能是按需编码方式,也可能是固定采样周期编码方式。那么相应的,若编码方式为按需编码方式,则可采用帧率调整因子对编码帧率进行缩放处理(即放大处理或缩小处理),得到一个缩放后的帧率。然后,根据该缩放后的帧率调整渲染帧率,并根据调整后的渲染帧率调整编码帧率,从而得到调整后的编码帧率。其中,根据该缩放后的帧率调整渲染帧率的具体实施方式可以参见上述所提及的根据目标帧率调整渲染帧率的相关实施方式,在此不再赘述。若编码方式为固定采样周期编码方式,则可直接采用帧率调整因子对编码帧率进行缩放处理,得到调整后的编码帧率。具体的,可直接根据帧率调整因子生成编码模块能够识别且接受的调整信息,并将该调整信息传递至编码模块中,由编码模

块在根据该调整信息识别得到帧率调整因子后,采用帧率调整因子对编码帧率进行缩放处理,得到调整后的编码帧率。其中,调整信息可包括但不限于:帧率调整因子对应的参数、指令或配置文件,等等。

S303,按照调整后的编码帧率,对目标云游戏的游戏画面进行图像编码,得到编码数据。

5 S304,将编码数据传输至目标游戏客户端。

S305,根据目标边缘服务器的图形处理器的资源总量和已占用资源量,确定目标边缘服务器的图形处理器的剩余资源量。

其中,已占用资源量至少包括:对游戏画面进行图像编码所占用的资源量。需要说明的是,图像编码所占用的资源量的多少通常与编码帧率的大小相关联;编码帧率越大,则图像编码所占用的资源量越多;编码帧率越小,则图像编码所占用的资源量就越少。由此可见,若调整后的编码帧率小于编码帧率,相对于按照所述编码帧率对所述游戏画面进行图像编码所占用的资源量,按照所述调整后的编码帧率对所述游戏画面进行图像编码所占用的资源量更少。也就是说,若调整后的编码帧率小于编码帧率,则按照调整后的编码帧率对游戏画面进行图像编码所对应的剩余资源量,大于按照编码帧率对游戏画面进行图像编码所对应的剩余资源量。

15 S306,根据剩余资源量,动态调整目标边缘服务器内运行的系统容器的数量。

具体的,若剩余资源量大于资源量阈值,则可在目标边缘服务器内并行地运行至少一个新的系统容器;若剩余资源量小于或等于资源量阈值,则可在目标边缘服务器中停止运行至少一个已运行的系统容器。由此可见,目标边缘服务器在CPU核数、GPU内存(GPU Memory)等配置均允许的情况下,可以实时地再运行一个或多个Android容器,达到提高服务器利用率的目的。

20 本申请实施例可考虑目标游戏客户端的帧率需求,对编码帧率进行动态调整,并按照调整后的编码帧率对目标云游戏的游戏画面进行图像编码。这样可使得服务器端在图像编码过程中所占用的处理资源均是有价值的,减少非必要的处理资源的占用;从而实现对服务器端的处理资源的合理化利用,提升处理资源的有效利用率。在通过图像编码得到编码数据后,可将图像编码所得到的编码数据传输至目标游戏客户端;由于编码数据是通过考虑目标游戏客户端的帧率需求进行编码得到的,因此编码数据所对应的帧率可较好地适配于目标游戏客户端,使得目标游戏客户端可较好地根据编码数据显示游戏画面。除此之外,服务器还可通过调整服务器端的渲染和编码能力,实时根据服务器端的GPU负载情况,动态增加或减少同时运行的系统容器的数量。在不需运行高帧率模式下的场景下,通过降低编码帧率和/或渲染帧率,可在一定程度上降低服务器端的单个系统容器的使用成本,有效节省处理资源;还可使得服务器端基于节省的处理资源来并发地运行一些新的系统容器,从而使得服务器端的处理资源(如GPU资源)的使用率可以得到极大的提高。

30 基于上述图像处理方法的实施例的描述,本申请实施例还公开了一种图像处理装置,所述图像处理装置可以是运行于上述所提及的目标边缘服务器中的一个计算机程序(包括程序代码)。该图像处理装置可以执行图2-图3所示的方法。请参见图5,所述图像处理装置可以运行如下单元:

获取单元501,用于在运行目标云游戏的过程中,获取目标游戏客户端发送的反馈数据;所述反馈数据用于反映所述目标游戏客户端的帧率需求;

40 处理单元502,用于根据所述反馈数据对编码帧率进行动态调整,得到调整后的编码帧率;

所述处理单元502,还用于按照所述调整后的编码帧率,对所述目标云游戏的游戏画面进行图像编码,得到编码数据;

传输单元503,用于将所述编码数据传输至所述目标游戏客户端,使所述目标游戏客户端根据所述编码数据显示所述游戏画面。

45 在一种实施方式中,目标云游戏运行在目标容器中,目标游戏客户端与目标容器相连接。所述反馈数据包括至少一个反馈指标下的反馈信息,所述反馈指标包括:硬件配置指标、应用场景指标、网络状况指标以及用户类型指标;

所述硬件配置指标下的反馈信息包括以下至少一项:所述目标游戏客户端对应的图形处理器的硬件信息,以及所述目标游戏客户端对应的中央处理器的硬件信息;

所述应用场景指标下的反馈信息包括:所述目标云游戏在所述目标游戏客户端中的应用场

景的场景标识；所述应用场景包括：挂机场景或者非挂机场景，所述挂机场景是指：由程序控制所述目标游戏客户端对应的游戏对象的场景；

所述网络状况指标下的反馈信息包括以下至少一项：网络带宽、网络抖动信息以及网络丢包率；

- 5 所述用户类型指标下的反馈信息包括：登陆所述目标云游戏的用户账号的用户类型的类型标识；所述用户类型包括：第一类型或者第二类型，所述第一类型所对应的游戏服务优于所述第二类型所对应的游戏服务。

再一种实施方式中，处理单元 502 在用于根据所述反馈数据对编码帧率进行动态调整，得到调整后的编码帧率时，可具体用于：

- 10 获取所述反馈数据中的各反馈指标下的反馈信息对应的参考帧率，并根据所述各反馈指标下的反馈信息对应的参考帧率计算得到目标帧率；

按照编码方式采用所述目标帧率对编码帧率进行动态调整，得到调整后的编码帧率。

再一种实施方式中，一个反馈指标对应一个帧率查询表，所述帧率查询表中包括多个信息以及各信息对应的帧率；

- 15 所述反馈数据中的任一反馈指标下的反馈信息对应的参考帧率，是从所述任一反馈指标对应的帧率查询表中查询到的。

再一种实施方式中，处理单元 502 在用于根据所述各反馈指标下的反馈信息对应的参考帧率计算得到目标帧率时，可具体用于：

- 20 从所述各反馈指标下的反馈信息对应的参考帧率中，选取最小参考帧率作为目标帧率；或者，

对所述各反馈指标下的反馈信息对应的参考帧率进行均值运算，得到目标帧率；或者，

获取所述各反馈指标的权重值，并采用所述各反馈指标的权重值对所述各反馈指标下的反馈信息对应的参考帧率进行加权求和，得到目标帧率。

- 25 再一种实施方式中，所述目标云游戏的游戏画面是由渲染模块渲染得到的，所述图像编码是由编码模块执行的；所述编码方式为按需编码方式，所述按需编码方式是指：所述编码模块按照所述渲染模块的渲染进度进行编码的方式；相应的，处理单元 502 在用于按照编码方式采用所述目标帧率对编码帧率进行动态调整，得到调整后的编码帧率时，可具体用于：

根据所述目标帧率调整硬件垂直信号量的产生周期，得到所述硬件垂直信号量的调整后的产生周期；

- 30 通过所述硬件垂直信号量的调整后的产生周期，同步调整软件垂直信号量的产生周期；所述软件垂直信号量是基于所述硬件垂直信号量生成的，且所述软件垂直信号量用于触发画面渲染；

采用所述软件垂直信号量的调整后的产生周期，调整所述渲染模块的渲染帧率；

- 35 基于调整后的渲染帧率对编码帧率进行动态调整，得到调整后的编码帧率；所述调整后的编码帧率等于所述调整后的渲染帧率。

再一种实施方式中，所述目标云游戏的游戏画面是由渲染模块渲染得到的，所述图像编码是由编码模块执行的；所述编码方式为固定采样周期编码方式，所述固定采样周期编码方式是指：所述编码模块按照固定的采样周期对所述渲染模块所渲染的游戏画面进行编码的方式；相应的，处理单元 502 在用于按照编码方式采用所述目标帧率对编码帧率进行动态调整，得到调整后的编码帧率时，可具体用于：

- 40 采用所述目标帧率更新所述编码帧率，得到调整后的编码帧率；所述调整后的编码帧率等于所述目标帧率。

再一种实施方式中，处理单元 502 在用于根据所述反馈数据对编码帧率进行动态调整，得到调整后的编码帧率时，可具体用于：

- 45 根据所述反馈数据中的各反馈指标下的反馈信息，确定帧率调整方向；所述帧率调整方向包括：帧率增大的方向，或者帧率减小的方向；

获取所述帧率调整方向所对应的帧率调整因子，并采用所述帧率调整因子对编码帧率进行动态调整，得到调整后的编码帧率。

再一种实施方式中，所述目标容器是指运行在云游戏系统中的目标边缘服务器中的任一系

统容器；处理单元 502 还可用于：

根据所述目标边缘服务器的图形处理器的资源总量和已占用资源量，确定所述目标边缘服务器的图形处理器的剩余资源量；

根据所述剩余资源量，动态调整所述目标边缘服务器内运行的系统容器的数量；

- 5 其中，所述已占用资源量至少包括：对游戏画面进行图像编码所占用的资源量；且若所述调整后的编码帧率小于所述编码帧率，相对于按照所述编码帧率对所述游戏画面进行图像编码所占用的资源量，按照所述调整后的编码帧率对所述游戏画面进行图像编码所占用的资源量更少。

- 10 再一种实施方式中，处理单元 502 在用于根据所述剩余资源量，动态调整所述目标边缘服务器内运行的系统容器的数量时，可具体用于：

若所述剩余资源量大于资源量阈值，则在所述目标边缘服务器内并行地运行至少一个新的系统容器；

若所述剩余资源量小于或等于所述资源量阈值，则在所述目标边缘服务器中停止运行至少一个已运行的系统容器。

- 15 根据本申请的一个实施例，图 2-图 3 所示的方法所涉及的各个步骤均可以是由图 5 所示的图像处理装置中的各个单元来执行的。例如，图 2 中所示的步骤 S201 可由图 5 中所示的获取单元 501 来执行，步骤 S202-S203 可由图 5 中所示的处理单元 502 来执行，步骤 S204 可由图 5 中所示的传输单元 503 来执行。又如，图 3 中所示的步骤 S301 可由图 5 中所示的获取单元 501 来执行，步骤 S302-S303 可由图 5 中所示的处理单元 502 来执行，步骤 S304 可由图 5 中所示的传输单元 503 来执行，步骤 S305-S306 可由图 5 中所示的处理单元 502 来执行，等等。

- 20 根据本申请的另一个实施例，图 5 所示的图像处理装置中的各个单元可以分别或全部合并为一个或若干个另外的单元来构成，或者其中的某个（些）单元还可以再拆分为功能上更小的多个单元来构成，这可以实现同样的操作，而不影响本申请的实施例的技术效果的实现。上述单元是基于逻辑功能划分的，在实际应用中，一个单元的功能也可以由多个单元来实现，或者多个单元的功能由一个单元实现。在本申请的其它实施例中，基于图像处理装置也可以包括其它单元，在实际应用中，这些功能也可以由其它单元协助实现，并且可以由多个单元协作实现。

- 25 根据本申请的另一个实施例，可以通过在包括中央处理单元（CPU）、随机存取存储介质（RAM）、只读存储介质（ROM）等处理元件和存储元件的，例如计算机的通用计算设备上运行能够执行如图 2-图 3 中所示的相应方法所涉及的各步骤的计算机程序（包括程序代码），来构造如图 5 中所示的图像处理装置，以及来实现本申请实施例的图像处理方法。所述计算机程序可以记载于例如计算机可读存储介质上，并通过计算机可读存储介质装载于上述计算设备中，并在其中运行。

- 30 本申请实施例在运行目标云游戏的过程中，可获取目标游戏客户端发送的反馈数据。由于该反馈数据可反映目标游戏客户端的帧率需求，因此服务器可考虑该反馈数据所反映的帧率需求，对编码帧率进行动态调整，并按照调整后的编码帧率对目标云游戏的游戏画面进行图像编码。这样可使得服务器端在图像编码过程中所占用的处理资源均是有价值的，减少非必要的处理资源的占用；从而实现对服务器端的处理资源的合理化利用，提升处理资源的有效利用率。在通过图像编码得到编码数据后，可将图像编码所得到的编码数据传输至目标游戏客户端；由于编码数据是通过考虑目标游戏客户端的帧率需求进行编码得到的，因此编码数据所对应的帧率可较好地适配于目标游戏客户端，使得目标游戏客户端可较好地根据编码数据显示游戏画面。

- 40 基于上述方法实施例以及装置实施例的描述，本申请实施例还提供一种服务器；该服务器可以是上述所提及的目标边缘服务器。请参见图 6，该服务器至少包括处理器 601、输入接口 602、输出接口 603 以及计算机存储介质 604。其中，服务器内的处理器 601、输入接口 602、输出接口 603 以及计算机存储介质 604 可通过总线或其他方式连接。

- 45 所述计算机存储介质 604 是服务器中的记忆设备，用于存放程序和数据。可以理解的是，此处的计算机存储介质 604 既可以包括服务器中的内置存储介质，当然也可以包括服务器所支持的扩展存储介质。计算机存储介质 604 提供存储空间，该存储空间存储了服务器的操作系统。并且，在该存储空间中还存放了适于被处理器 601 加载并执行的一条或多条的指令，这些指令可以是一个或一个以上的计算机程序（包括程序代码）。需要说明的是，此处的计算机存储介质

可以是高速 RAM 存储器,也可以是非易失性的存储器 (non-transitory memory),例如至少一个磁盘存储器;还可以是至少一个位于远离前述处理器的计算机存储介质。所述处理器 601 (或称中央处理器 (Central Processing Unit, CPU)) 是服务器的计算核心以及控制核心,其适于实现一条或多条指令,具体适于加载并执行一条或多条指令从而实现相应方法流程或相应功能。

- 5 在一个实施例中,可由处理器 601 加载并执行计算机存储介质 604 中存放的一条或多条指令,以实现上述有关图 2-图 3 所示的图像处理方法实施例中的相应方法步骤;具体实现中,计算机存储介质 604 中的一条或多条指令由处理器 601 加载并执行如下步骤:

在运行目标云游戏的过程中,获取目标游戏客户端发送的反馈数据;所述反馈数据用于反映所述目标游戏客户端的帧率需求;

- 10 根据所述反馈数据对编码帧率进行动态调整,得到调整后的编码帧率;  
按照所述调整后的编码帧率,对所述目标云游戏的游戏画面进行图像编码,得到编码数据;  
将所述编码数据传输至所述目标游戏客户端,使所述目标游戏客户端根据所述编码数据显示所述游戏画面。

- 15 在一种实施方式中,所述目标云游戏运行在目标容器内,所述目标游戏客户端与目标容器相连接。所述反馈数据包括至少一个反馈指标下的反馈信息,所述反馈指标包括:硬件配置指标、应用场景指标、网络状况指标以及用户类型指标;

所述硬件配置指标下的反馈信息包括以下至少一项:所述目标游戏客户端对应的图形处理器的硬件信息,以及所述目标游戏客户端对应的中央处理器的硬件信息;

- 20 所述应用场景指标下的反馈信息包括:所述目标云游戏在所述目标游戏客户端中的应用场景的场景标识;所述应用场景包括:挂机场景或者非挂机场景,所述挂机场景是指:由程序控制所述目标游戏客户端对应的游戏对象的场景;

所述网络状况指标下的反馈信息包括以下至少一项:网络带宽、网络抖动信息以及网络丢包率;

- 25 所述用户类型指标下的反馈信息包括:登陆所述目标云游戏的用户账号的用户类型的类型标识;所述用户类型包括:第一类型或者第二类型,所述第一类型所对应的游戏服务优于所述第二类型所对应的游戏服务。

再一种实施方式中,在根据所述反馈数据对编码帧率进行动态调整,得到调整后的编码帧率时,所述一条或多条指令可由处理器 601 加载并具体执行:

- 30 获取所述反馈数据中的各反馈指标下的反馈信息对应的参考帧率,并根据所述各反馈指标下的反馈信息对应的参考帧率计算得到目标帧率;

按照编码方式采用所述目标帧率对编码帧率进行动态调整,得到调整后的编码帧率。

再一种实施方式中,一个反馈指标对应一个帧率查询表,所述帧率查询表中包括多个信息以及各信息对应的帧率;

- 35 所述反馈数据中的任一反馈指标下的反馈信息对应的参考帧率,是从所述任一反馈指标对应的帧率查询表中查询到的。

再一种实施方式中,在根据所述各反馈指标下的反馈信息对应的参考帧率计算得到目标帧率时,所述一条或多条指令可由处理器 601 加载并具体执行:

从所述各反馈指标下的反馈信息对应的参考帧率中,选取最小参考帧率作为目标帧率;或者,

- 40 对所述各反馈指标下的反馈信息对应的参考帧率进行均值运算,得到目标帧率;或者,  
获取所述各反馈指标的权重值,并采用所述各反馈指标的权重值对所述各反馈指标下的反馈信息对应的参考帧率进行加权求和,得到目标帧率。

- 再一种实施方式中,所述目标云游戏的游戏画面是由渲染模块渲染得到的,所述图像编码是由编码模块执行的;所述编码方式为按需编码方式,所述按需编码方式是指:所述编码模块按照所述渲染模块的渲染进度进行编码的方式;相应的,在按照编码方式采用所述目标帧率对编码帧率进行动态调整,得到调整后的编码帧率时,所述一条或多条指令可由处理器 601 加载并具体执行:

根据所述目标帧率调整硬件垂直信号量的产生周期,得到所述硬件垂直信号量的调整后的产生周期;

通过所述硬件垂直信号量的调整后的产生周期，同步调整软件垂直信号量的产生周期；所述软件垂直信号量是基于所述硬件垂直信号量生成的，且所述软件垂直信号量用于触发画面渲染；

采用所述软件垂直信号量的调整后的产生周期，调整所述渲染模块的渲染帧率；

- 5 基于调整后的渲染帧率对编码帧率进行动态调整，得到调整后的编码帧率；所述调整后的编码帧率等于所述调整后的渲染帧率。

再一种实施方式中，所述目标云游戏的游戏画面是由渲染模块渲染得到的，所述图像编码是由编码模块执行的；所述编码方式为固定采样周期编码方式，所述固定采样周期编码方式是指：所述编码模块按照固定的采样周期对所述渲染模块所渲染的游戏画面进行编码的方式；相应的，在按照编码方式采用所述目标帧率对编码帧率进行动态调整，得到调整后的编码帧率时，所述一条或多条指令可由处理器 601 加载并具体执行；

采用所述目标帧率更新所述编码帧率，得到调整后的编码帧率；所述调整后的编码帧率等于所述目标帧率。

- 15 再一种实施方式中，在根据所述反馈数据对编码帧率进行动态调整，得到调整后的编码帧率时，所述一条或多条指令可由处理器 601 加载并具体执行；

根据所述反馈数据中的各反馈指标下的反馈信息，确定帧率调整方向；所述帧率调整方向包括：帧率增大的方向，或者帧率减小的方向；

获取所述帧率调整方向所对应的帧率调整因子，并采用所述帧率调整因子对编码帧率进行动态调整，得到调整后的编码帧率。

- 20 再一种实施方式中，所述目标容器是指运行在云游戏系统中的目标边缘服务器中的任一系统容器；所述一条或多条指令可由处理器 601 加载并具体执行；

根据所述目标边缘服务器的图形处理器的资源总量和已占用资源量，确定所述目标边缘服务器的图形处理器的剩余资源量；

根据剩余资源量，动态调整所述目标边缘服务器内运行的系统容器的数量；

- 25 其中，所述已占用资源量至少包括：对游戏画面进行图像编码所占用的资源量；且若所述调整后的编码帧率小于所述编码帧率，相对于按照所述编码帧率对所述游戏画面进行图像编码所占用的资源量，按照所述调整后的编码帧率对所述游戏画面进行图像编码所占用的资源量更少。

- 30 再一种实施方式中，在根据所述剩余资源量，动态调整所述目标边缘服务器内运行的系统容器的数量时，所述一条或多条指令可由处理器 601 加载并具体执行；

若所述剩余资源量大于资源量阈值，则在所述目标边缘服务器内并行地运行至少一个新的系统容器；

若所述剩余资源量小于或等于所述资源量阈值，则在所述目标边缘服务器中停止运行至少一个已运行的系统容器。

- 35 本申请实施例在运行目标云游戏的过程中，可获取目标游戏客户端发送的反馈数据。由于该反馈数据可反映目标游戏客户端的帧率需求，因此服务器可考虑该反馈数据所反映的帧率需求，对编码帧率进行动态调整，并按照调整后的编码帧率对目标云游戏的游戏画面进行图像编码。这样可使得服务器端在图像编码过程中所占用的处理资源均是有价值的，减少非必要的处理资源的占用；从而实现对服务器端的处理资源的合理化利用，提升处理资源的有效利用率。

- 40 在通过图像编码得到编码数据后，可将图像编码所得到的编码数据传输至目标游戏客户端；由于编码数据是通过考虑目标游戏客户端的帧率需求进行编码得到的，因此编码数据所对应的帧率可较好地适配于目标游戏客户端，使得目标游戏客户端可较好地根据编码数据显示游戏画面。

需要说明的是，根据本申请的一个方面，还提供了一种计算机程序产品或计算机程序，该计算机程序产品或计算机程序包括计算机指令，该计算机指令存储在计算机可读存储介质中。

- 45 计算设备的处理器从计算机可读存储介质读取该计算机指令，处理器执行该计算机指令，使得该计算设备执行上述图 2-图 3 所示的图像处理方法的实施例方面的各种方式中提供的方法。

并且，应理解的是，以上所揭露的仅为本申请较佳实施例而已，当然不能以此来限定本申请之权利范围，因此依本申请权利要求所作的等同变化，仍属本申请所涵盖的范围。