GPU 云服务器

GPU 共享

产品文档

腾讯云

文档目录

GPU 共享
qGPU 概述

最近更新时间：2022-09-15 10:09:23

qGPU 是腾讯云推出的 GPU 共享技术，支持在多个容器间共享 GPU 卡并提供容器间显存与算力强隔离的能力，从而在更小粒度使用 GPU 卡的基础上，保证业务安全，达到提高 GPU 使用率、降低用户成本的目的。
qGPU 依托腾讯云容器服务 TKE 对外开源的 Elastic GPU 框架，可实现对 GPU 算力与显存的细粒度调度，并支持多容器共享 GPU 与多容器跨 GPU 资源分配。同时依赖底层强大的 qGPU 隔离技术，可做到 GPU 显存和算力的强隔离，在通过共享使用 GPU 的同时，尽量保证业务性能与资源不受干扰。

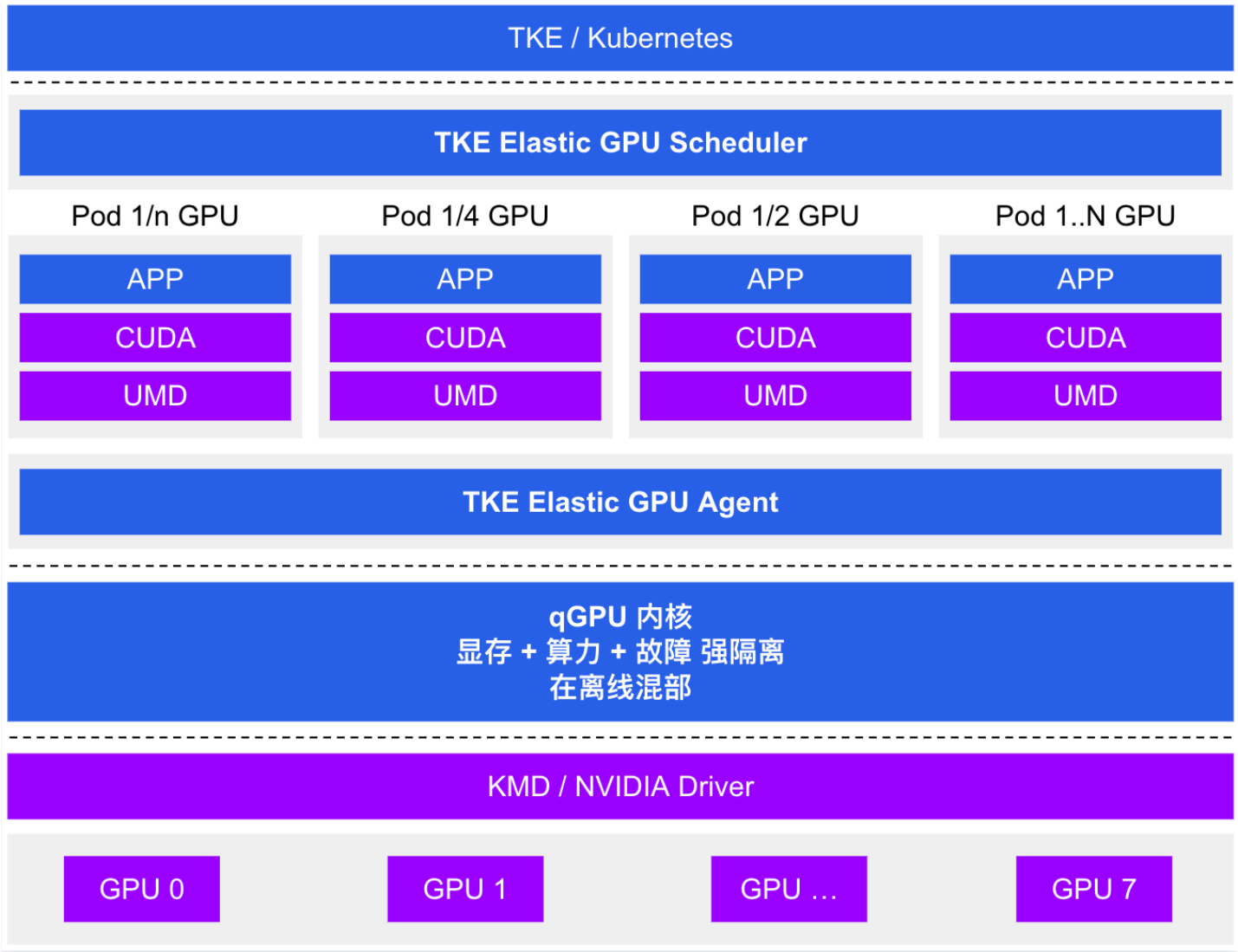
qGPU 功能及优势

qGPU 方案通过对 NVIDIA GPU 卡上任务的有效调度，达到给多个容器共享使用的目的。功能优势如下：

- 灵活性：用户可以精细配置 GPU 的显存大小和算力占比。
- 强隔离：支持显存和算力的严格隔离。
- 在离线：支持业界唯一在离线混部能力，GPU 利用率压榨到极致。
- 覆盖度：支持主流架构 Volta（如 V100）、Turing（如 T4 等）、Ampere（如 A100、A10）。
- 云原生：支持标准的 Kubernetes 和 NVIDIA Docker。
- 兼容性：业务不重编、CUDA 库不替换、业务无感。
- 高性能：在底层对 GPU 设备进行操作，高效收敛，吞吐接近0损耗。

方案框架图

qGPU 方案框架图如下：



使用 qGPU

请参考 qGPU 使用 了解并开始使用。

qGPU 使用

最近更新时间：2023-04-23 14:09:33

操作场景

本文介绍如何通过腾讯云容器服务 TKE 使用 qGPU。

使用须知

- **TKE 版本支持**：需 ⩾ v1.14.x。
- **操作系统支持**：请参见 TKE 支持的公共镜像列表。推荐使用 TencentOS Server 3.1 (TK4)，公共镜像为更稳定、高效、易维护的使用方式。不推荐您使用市场镜像。
- **GPU 卡架构**：支持 Volta（如 V100）、Turing（如 T4）、Ampere（如 A100、A10)。
- **驱动版本**：支持驱动版本由镜像和机型所共同决定，具体可参见 准备 GPU 资源。

> **说明**
> 为保证兼容性，推荐您在节点上安装 NVIDIA 驱动，无需在 POD 内部重复安装。

- **共享粒度**：每个 qGPU 最小分配1G显存，精度单位是1G。算力最小分配5（代表一张卡的5%），最大100（代表一张卡），精度单位是5（即5、10、15、20 ... 100）。
- **整卡分配**：开启了 qGPU 能力的节点可按照 `tke.cloud.tencent.com/qgpu-core: 100 | 200 | ...`（N * 100，N 是整卡个数）的方式分配整卡。建议通过 TKE 的节点池能力来区分 NVIDIA 分配方式或转换到 qGPU 使用方式。
- **个数限制**：一个 GPU 上最多可创建16个 qGPU 设备。建议按照容器申请的显存大小确定单个 GPU 卡可共享部署的 qGPU 个数。
- **升级须知**：如需升级 Kubernetes Master 版本，请注意：
  - 对于托管集群，无需重新设置本插件。
  - 对于独立集群，master 版本升级会重置 master 上所有组件的配置，会影响 qgpu-scheduler 插件作为 Scheduler Extender 的配置，因此 qGPU 插件需要卸载后再重新安装。

操作步骤

> **说明**
> 由于使用 qGPU 能力需要使用特定镜像以及设置相关 Label，强烈建议您使用 TKE 的节点池能力来对节点进行分组管理（节点池的节点具备统一的 Label 以及镜像属性），详情请参见 创建节点池。

安装qGPU 组件

1. 登录 容器服务控制台，在左侧导航栏中选择**集群**。
2. 在**集群管理**页面单击目标集群 ID，进入集群详情页。
3. 选择左侧菜单栏中的**组件管理**，进入“组件列表”页面。
4. 在**组件列表**页面中选择**新建**，并在**新建组件**页面中勾选 qGPU（GPU 隔离组件）。
5. 单击**参数配置**，可以设置 qgpu-scheduler 的调度策略。
   - **spread**：多个 Pod 会分散在不同节点、不同显卡上，优先选择资源剩余量较多的节点。适用于高可用场景，避免把同一个应用的副本放到同一个设备上。
   - **binpack**：多个 Pod 会优先使用同一个节点，适用于提高 GPU 利用率的场景。
6. 单击**完成**即可创建组件。
   安装成功后，需要为集群准备 GPU 资源。

开启集群 qGPU 共享

1. 单击目标集群 ID，进入集群详情页。
2. 单击 qGPU 共享右侧的  ，开启 qGPU 共享。如下图所示：

开启后，**集群中所有新增 GPU 节点**默认开启 GPU 共享能力。您可以在 准备 GPU 资源 中通过 Label 控制是否开启隔离能力。

准备 GPU 资源

1. 在集群管理页面中，选择左侧工具栏中的**节点管理** > **节点池**。
2. 在**节点池列表**页面中，单击**新建节点池**。
3. 在创建节点池页面中，选择支持的镜像，例如 **TencentOS Server 3.1 (TK4)** 并设置相关驱动。
   设置镜像并选择机型后，可以根据需求选择 GPU 驱动的版本、CUDA 版本、cuDNN 版本。其余参数配置可参见 创建节点池 。如下图所示：

说明

- 勾选“后台自动安装 GPU 驱动”后，将在系统启动时进行自动安装，预计耗时15 – 25分钟。支持的驱动版本由 OS 以及 GPU 机型共同决定，详情可参见 各实例支持的 GPU 驱动版本及安装方式 。
- 若您未勾选**后台自动安装 GPU 驱动**，为了保证 GPU 机型的正常使用，针对某些低版本 OS，将会为您默认安装 GPU 驱动，完整的默认驱动版本信息可参见下表：

OS 名称	默认安装驱动版本
CentOS 7.6、Ubuntu 18、Tencent Linux2.4	450
Centos 7.2（不推荐）	384.111
Ubuntu 16（不推荐）	410.79

4. 在创建节点池页面中展开**更多设置**，通过高级配置设置 Label，指定 qGPU 隔离策略 。如下图所示：

- Label 键：`tke.cloud.tencent.com/qgpu-schedule-policy` 。
- Label 值：`fixed-share` 。设置 Label value 时，可填写全称或者缩写，更多取值请参考下表。
  当前 qGPU 支持以下三种隔离策略：

Label 值	缩写	英文名	中文名	含义
best-effort（默认值）	be	Best Effort	争抢模式	默认值。各个 Pods 不限制算力，只要卡上有剩余算力就可使用。如果一共启动 N 个 Pods，每个 Pod 负载都很重，则最终结果就是 1/N 的算力。
fixed-share	fs	Fixed Share	固定配额	每个 Pod 有固定的算力配额，无法超过固定配额，即使 GPU 还有空闲算力。
burst-share	bs	Guaranteed Share with Burst	保证配额加弹性能力	调度器保证每个 Pod 有保底的算力配额，但只要 GPU 还有空闲算力，就可被 Pod 使用。例如，当 GPU 有空闲算力时（没有分配给其他 Pod），Pod 可以使用超过它的配额的算力。注意，当它所占用的这部分空闲算力再次被分配出去时，Pod 会回退到它的算力配额。

5. 单击**创建节点池**即可。

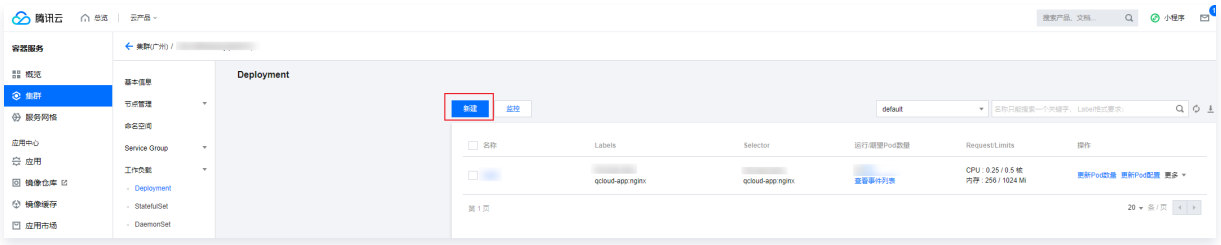
给应用分配共享 GPU 资源

通过给容器设置 qGPU 对应资源允许 Pod 使用 qGPU，您可以通过控制台或者 YAML 方式来设置：

> **说明**
> - 若应用需使用整数卡资源，只需填写卡数，无需填写显存（自动使用分配的 GPU 卡上全部显存）。
> - 若应用需要使用小数卡资源（即和其他应用共享同一张卡），需同时填写卡数和显存。

通过控制台设置 | 通过 YAML 设置

1.在**集群管理**页面单击目标集群 ID，进入集群详情页，单击**工作负载**展开选项。



2.在新建 Deployment 页面，填写 GPU 资源。如下图所示：

附录

部署在集群内的 Kubernetes 对象

Kubernetes 对象名称	类型	请求资源	Namespace
qgpu-manager	DaemonSet	每 GPU 节点一个 Memory: 300M, CPU:0.2	kube-system
qgpu-manager	ClusterRole	-	-
qgpu-manager	ServiceAccount	-	kube-system
qgpu-manager	ClusterRoleBinding	-	kube-system
qgpu-scheduler	Deployment	单一副本 Memory: 800M, CPU:1	kube-system
qgpu-scheduler	ClusterRole	-	-
qgpu-scheduler	ClusterRoleBinding	-	kube-system
qgpu-scheduler	ServiceAccount	-	kube-system
qgpu-scheduler	Service	-	kube-system