百度智能云

GPU 文档

目录

功能发布记录

发布时间	功能概述
2018-12	• 华东、华南region已正式开放V100实例的售卖 • 后付费4卡、8卡GPU实例不再默认对外开放购买，需要通过白名单开通使用权限
2018-09	新增支持Tensor Core功能的NVIDIA Tesla V100英伟达最新GPU卡

产品描述

产品介绍

GPU云服务器是百度智能云中配备GPU卡的云计算服务，可以帮助您快速、便捷地获得高质量的GPU计算资源，能够大幅提高机器学习及科学计算等大规模计算框架的运行速度，为搭建人工智能及高性能计算平台提供基础架构支持。

GPU云服务器提供与使用标准BCC云服务器一致的管理方式，您可以快速灵活的申请并管理GPU云服务器，结合私有网络VPC、负载均衡BLB构建云环境中高可用的GPU算力服务。

百度智能云提供丰富的实例规格的GPU云服务器，分类如下：

- 虚拟机规格：规格灵活，支持不同GPU数量。
- 裸金属规格：资源独占，性能更稳定。
- 弹性高性能计算集群规格：资源独占，支持高性能网络，优化分布式计算任务。

详情可参考GPU云服务器实例规格。

GPU云服务器相关组件

除了基础的计算资源，百度智能云也提供以下组件，结合业务场景，可有效提升GPU的计算效率，提升资源利用率。

- AIAK加速组件：AIAK是面向人工智能任务提供的加速引擎，用于优化基于AI主流计算框架搭建的模型，能显著提升AI任务开发、部署的运行效率。百度智能云可提供从AI训练到AI推理的加速能力，详情可见AIAK加速组件。

基于GPU云服务器的典型实践

百度智能云为您提供使用业界主流GPU工具的典型实践，包含如何开启新一代GPU的新特性，有效利用GPU相关工具优化业务部署效率，提高GPU资源利用率等。详情可见典型实践。

产品优势

百度智能云GPU云服务器聚焦弹性、易运维、高性能等核心特性，具备以下优势：

高性能： 提供业界高端的硬件配置，包含超高的RDMA通信带宽和全闪存的磁盘规格，充分发挥GPU的计算能力。

易用性： 支持自定义GPU运行环境，灵活变更实例规格，支持多种GPU监控指标。

弹性按需： 支持多种GPU虚拟化技术，提升业务对GPU资源的利用率。可弹性伸缩，秒级实现多台服务器的创建和释放。支持多种实例规格满足不同场景的需求。

与自建GPU服务器对比

应用场景

GPU的主流应用场景包含计算型应用和渲染型应用。

计算型应用可按照业务负载类型按以下分类。

人工智能训练

- 针对深度学习的训练负载，有大批量的数据，例如图片、语音、文本等，需要不断更新、迭代神经网络中的参数以满足业务对预测精度的要求。
- 可选择高性能的GPU型号来缩短网络模型的收敛时间，深度学习中存在大量矩阵计算，建议选择支持Tensor Core功能的GPU做计算加速。
- 进一步提高计算效率可选择分布式训练并选择支持高速GPU互联能力的型号.

人工智能推理

- 针对深度学习的在线推理场景，相比训练负载，推理负载对GPU性能的要求降低，但对运行稳定性要求更高，对服务器响应延时也有了更高要求。
- 可选择NVIDIA Tesla A10、NVIDIA Tesla T4等GPU类型，在满足性能要求的同时，提供更具性价比的选择，同时支持GPU硬件级的解码功能并加速端到端的图片类推理性能。

高性能计算

- 常见的高性能计算应用包括计算流体力学、分子动力学、有限元分析等，通常需要高精度算力来满足应用对精度的要求。
- 可选择NVIDIA Tesla A100、NVIDIA Tesla V100等支持双精度浮点计算的GPU型号。

渲染型应用可按照业务负载类型按以下分类。

图像渲染

- 渲染是用软件从模型生成图像的过程，需要 GPU 卡实现图形加速及实时渲染并常存在CPU、GPU频繁交互的场景。
- 推荐使用单精度FP32性能高并支持光线追踪的GPU型号，例如Tesla A10、Tesla T4等。

远程图形工作站

- 终端或者客户端通过专用网络连接到主机来进行日常的工作，主机服务器常集中部署在数据中心机房，并通过GPU卡处理图形工作负载。
- 推荐使用Tesla T4型号GPU。

GPU卡详情

GPU服务器使用的NVIDIA GPU卡基本参数信息如下表所示：

	GPU云服务器	自建GPU服务器
性能	• 保障CPU、GPU等算力稳定性 • 提供业界领先算力的GPU计算卡	• 服务器折旧损耗，无法保证性能的稳定性 • 业务升级，受制于成本，无法及时更换更高性能服务器 • 高性能服务器的成本指数增加，往往无法承担
弹性	• 灵活配置GPU、CPU、内存等，符合业务需求 • 随时拓展或缩减GPU云服务器数量，快速响应业务变化 • 调整配置，数据不丢失	• 服务器规模固定，无法应对突发的业务规模增长，或造成长期闲置，带来成本的大幅提高
可靠性	• 自动排除故障，自动备份数据，数据持久性高于99.9999999% • 采用领先的智能调度技术，节点发生故障，可几秒内将数据快速迁移恢复，应对高可靠性要求的场景，如金融、游戏	• 往往缺少对机器的实时全面监控，一旦发生问题，很难快速锁定排除故障，可能导致数据的永久丢失 • 需花费较多运维成本保障机器可靠性
可管理性	• 控制台功能强大，支持对上千数量级实例的多重实时管理 • 多维度监控预警，帮助您提前发现问题，减少损失	• 需运维持续投入人力，不断人工重复部署和维护工作
安全	• 免费提供多层次DDoS攻击防护，百Gbps级流量防攻击 • 分钟级响应速度，及时发现并有效抵御黑客攻击 • 主、子账号权限细分管理，保障数据安全	• 需额外购买安全防护服务 • 对突发的恶意攻击事件，应对能力不足
成本	• 支持包年包月或按量计费，按需购买，大大节约成本 • 无需维护硬件和网络设施，提高运维效率 • GPU云服务器推出了1年8.3折、2年7折、3年5折的优惠，后付费按分钟计费	• 租用费用高，只能包年包月 • 需持续投入人力维护，运维成本高

GPU卡型号	CUDA Cores	Tensor Cores	显存容量	FP64浮点性能	FP32浮点性能	FP16浮点性能	INT8性能
NVIDIA Tesla H800	16896	528	80GB	--	60 Tflops	989 Tflops	1978 Tops
NVIDIA Tesla A800	6912	432	80GB	--	19.5 Tflops	312 Tflops	624 Tops
NVIDIA Tesla A10	9216	288	24GB	--	31 Tflops	125 Tflops	250 Tops
NVIDIA Tesla A100-40G	6912	432	40GB	9.7 Tflops	19.5 Tflops	312 Tflops	624 Tops
NVIDIA Tesla T4	2560	320	16GB	--	8.1 Tflops	65 Tflops	130 Tops
NVIDIA Tesla V100-32G	5120	640	32GB	7.8 Tflops	15.7 Tflops	125 Tflops	60 Tops
NVIDIA Tesla P4	--	2560	8GB	--	5.5 Tflops	--	22 Tops

实例规格

GPU实例命名规则

GPU云服务器在实例命名上可以直观看到硬件配置差异情况，让用户能够精准定位产品的属性，便于与应用选型、配置推荐对应。

GPU云服务实例命名由4部分组成，其格式为： 产品代号+实例规格族+基本规格信息+扩展规格信息。

- 产品代号：如bcc，代表该实例规格所属产品系列。
- 实例规格族：如gn5，代表该实例规格包含的资源类型。
- 基本规格信息：如c128m1024g，代表该实例规格中包含的vCPU数量（单位：个）及内存大小（单位：GiB）。
- 扩展规格信息：如4a10，代表该实例规格所包含的扩展硬件资源配置，比如GPU卡数量，RDMA网口数量，实例本地盘数量等。

以下为实例规格可能出现的代号以及相关含义解释：

命名组成	说明	含义
产品代号	字母组成	bcc：云服务器 ebc：弹性裸金属服务器 ehc：弹性高性能计算集群
实例规格族	字母加数字组成	l：包含本地盘 gn：Nvidia GPU nkl：昆仑芯 NPU 1/2/3/5：Intel CPU平台不同系列 a2：AMD CPU平台
基本规格信息	字母加数字组成	c：vCPU m：内存 数字：数量
扩展规格信息组成	字母加数字组成	a10/a100/a800等：GPU卡名称 r200：NPU卡名称 re：RoCE类型组网的高性能计算集群 d：本地盘数量 数字：数量

GPU计算型

GPU计算型面向复杂的高密度计算类业务场景，例如人工智能计算，高性能计算等。

GPU计算型按照GPU虚拟化形态分为两种：

- 透传GPU实例：将整张GPU卡透传给实例使用，性能等同于物理GPU的性能。例如计算型GN5、GN3等都属于透传GPU实例。
- vGPU实例：将分片后的GPU卡分配给实例使用，具有更好的性价比，例如vGN3-C属于vGPU类实例。

以下为该系列各规格族的具体规格

售卖情况	实例规格族	实例类型	可用区
主售	计算型GN5	Ampere A100-80G	华北-北京、华东-苏州
		Ampere A10	华北-北京、华东-苏州、华南-广州
	计算型GN3	Volta V100	华北-北京、华东-苏州
		Turing T4	华北-北京、华南-广州、华东-苏州
在售	计算型vGN3-C	Volta V100	华北-北京
		Turing T4	华北-北京
	计算型GN1	Pascal P4	华北-北京、华南-广州、华东-苏州

计算型GN5

搭载Intel Xeon Icelake以及英伟达Ampere架构系列GPU

Ampere A100-80G系列

- 适用场景：
  - 人工智能大规模训练：例如图像识别、目标检测、无人驾驶、语义分析、广告推荐等场景
  - 高性能计算：生命科学、工业制造仿真、气象预测、计算物理等场景
  - 高性能数据分析：数据集可视化、大数据分析等场景
- 规格特点：
  - 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA Tesla A100，FP16算力达到312TFLOPS，单GPU显存80GB HBM2，支持双向带宽600GB的Nvlink互联
  - 内存：最高可提供960GiB内存
  - 存储：支持CDS云磁盘
  - 网络：最高可支持200Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求
- 实例规格

实例	显卡数量	vCPU	内存大小（GiB）	内网带宽（Gbps）	网络收发包（万pps）	队列数
bcc.gn5.c14m120.1A100-80g	1张	14核	120	25	625	8
bcc.gn5.c28m240.2A100-80g	2张	28核	240	50	1250	16
bcc.gn5.c56m480.4A100-80g	4张	56核	480	100	2500	32
bcc.gn5.c112m960.8A100-80g	8张	112核	960	200	5000	32

Ampere A10系列

- 适用场景：
  - 人工智能推理：例如图像识别、目标检测、无人驾驶、语义分析、广告推荐等场景
  - 高性能数据分析：数据集可视化、大数据分析等场景
  - 图形渲染：3D图形渲染、图形设计、云游戏等场景
- 规格特点：
  - 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA Tesla A10，支持RTX光线追踪算法加速， FP16算力达到125TFLOPS，单GPU显存24GB
  - 内存：最高可提供476GiB内存，处理器与内存配比为1:4
  - 存储：支持CDS云磁盘
  - 网络：最高可支持25Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求
- 实例规格

实例	显卡数量	vCPU	内存大小（GiB）	内网带宽（Gbps）	网络收发包（万pps）	队列数
bcc.gn5.c16m64.1a10	1张	16核	64	6	120	8
bcc.gn5.c32m128.2a10	2张	32核	128	12	250	16
bcc.gn5.c64m256.4a10	4张	64核	256	18	450	32
bcc.gn5.c28m112.1a10	1张	28核	112	8	180	8
bcc.gn5.c56m224.2a10	2张	56核	224	14	300	16
bcc.gn5.c112m476.4a10	4张	112核	476	25	800	32

计算型GN3

搭载Intel Xeon Cascade系列以及英伟达主流数据中心级GPU

Volta V100系列

- 适用场景：
  - 人工智能训练及大模型推理：例如图像识别、目标检测、无人驾驶、语义分析、广告推荐等场景
  - 高性能HPC计算：生命科学、CAE、气象预测、碰撞仿真等场景
  - 高性能数据分析：数据集可视化、大数据分析等场景
  - 编解码：视频流、图片流编解码等场景
- 规格特点：
  - 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA Tesla V100 SXM2，FP16算力达到125TFLOPS，单GPU显存32GB HBM2，GPU互联支持带宽300GB/s NVLink
  - 内存：最高可提供640GiB内存
  - 存储：支持CDS云磁盘
  - 网络：最高可支持25Gbps内网带宽，超高网络收发包能力，满足极高的内网传输需求

- 实例规格

实例	显卡数量	vCPU	内存大小（GiB）	内网带宽（Gbps）	网络收发包（万pps）	队列数
bcc.gn3.c10m80.1v100-32g	1张	10核	80	4	150	8
bcc.gn3.c20m160.2v100-32g	2张	20核	160	7	280	8
bcc.gn3.c40m320.4v100-32g	4张	40核	320	13	500	16
bcc.gn3.c80m640.8v100-32g	8张	80核	640	25	700	16

Turing T4系列

- 适用场景：
  - 人工智能推理：例如图像识别、目标检测、无人驾驶、语义分析、广告推荐等场景
  - 高性能数据分析：数据集可视化、大数据分析等场景
  - 编解码：视频流、图片流编解码等场景
- 规格特点：
  - 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA Tesla T4，INT8算力达到130TFLOPS，单GPU显存16GB
  - 内存：最高可提供320GiB内存
  - 存储：支持CDS云磁盘
  - 网络：超高网络收发包能力，满足极高的内网传输需求
- 实例规格

实例	显卡数量	vCPU	内存大小（GiB）	内网带宽（Gbps）	网络收发包（万pps）	队列数
bcc.gn3.c20m80.1t4	1张	20核	80	6	280	8
bcc.gn3.c40m160.2t4	2张	40核	160	11	500	16
bcc.gn3.c80m320.4t4	4张	80核	320	22	700	16

计算型vGN3-C

搭配NVIDIA主流数据中心分片虚拟化后的vGPU，提高GPU使用性价比以及GPU的资源利用率

Volta V100系列

- 适用场景：
  - 人工智能推理：例如图像识别、目标检测、无人驾驶、语义分析、广告推荐等场景
  - 人工智能小规模训练：例如图像识别、目标检测等场景
  - 人工智能的模型开发、教学环境
- 实例规格：
  - 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA Tesla V100 vGPU，GPU显存支持8G或16G
  - 内存：最高可提供24GiB内存

- 存储：支持CDS云磁盘
- 网络：超高网络收发包能力，满足极高的内网传输需求

- 实例规格

实例	显卡数量	显存大小（GiB）	vCPU	内存大小（GiB）	内网带宽(Gb)	网络收发包（万pps）
bcc.vgn3.c2m12.1v100-32g-8C	1/4张	8	2核	12	1.5	40
bcc.vgn3.c4m24.1v100-32g-16C	1/2张	16	4核	24	1.5	80

Turing T4系列

该实例适用于计算场景，如人工智能推理。如果您需要实例支持例如OpenGL或者Windows DirectX等完整图形图像功能，请使用百度智能云渲染型GPU实例vGN3-Q

- 适用场景：
  - 人工智能推理：例如图像识别、目标检测、无人驾驶、语义分析、广告推荐等场景
  - 人工智能的模型开发、教学环境
- 实例规格：
  - 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA T4 vGPU，单实例GPU显存支持4 GB或8 GB
  - 内存：最高可提供40GiB内存
  - 存储：支持CDS云磁盘
  - 网络：超高网络收发包能力，满足极高的内网传输需求
- 实例规格

实例	显卡数量	显存大小（GB）	vCPU	内存大小（GiB）	内网带宽(Gb)	网络收发包（万pps）
bcc.vgn3.c4m16.1t4-4C	1/4张	4	4核	16	1.5	80
bcc.vgn3.c10m40.1t4-8C	1/2张	8	10核	40	4	150

计算型GN1

搭载英伟达主流数据中心级GPU P4

Pascal P4系列

- 适用场景：
  - 人工智能推理：例如图像识别、目标检测、语义分割等场景
  - 数据分析：数据集可视化、大数据分析等场景
  - 编解码：视频流、图片流编解码等场景
- 规格特点：
  - 处理器：Intel Xeon Skylake 6148，主频 2.4GHz

- GPU：NVIDIA P4，整数型算力达到22TFLOPS，单GPU显存8GB
- 内存：最高可提供160GiB内存
- 存储：支持CDS云磁盘

- 实例规格

实例	显卡数量	vCPU	内存大小（GiB）
bcc.lgn1.c12m40.1p4	1张	12核	40
bcc.lgn1.c24m80.2p4	2张	24核	80
bcc.lgn1.c48m160.4p4	4张	48核	160

弹性高性能计算集群

弹性高性能计算集群

弹性高性能计算集群EHC是基于高性能RDMA网络，将多个裸金属服务器进行互联，每个裸金属服务器配置单独的RDMA网卡，可提供高带宽、低时延的通信能力，极大提升计算任务的加速比的计算集群，适用于超大模型训练、科学计算等大规模计算场景。百度智能云可提供业界主流RDMA技术路线，满足多场景计算需求。

名词解释：

RDMA：RDMA全称为Remote Direct Memory Access，即远程直接内存访问，允许本地应用程序直接读写远程应用程序的用户态虚拟内存。

RoCE v2：RoCE（全称RDMA over Converged Ethernet）是一种兼容性技术，允许RDMA的传输层运行在以太网络之上，避免了独立的网络基础设施投资，也解决了与外部以太网互联互通的问题。

RDMA IP：由百度智能云为弹性高性能计算实例中的RDMA网卡分配的Ip，降低RDMA网络的配置复杂度。用户可在EHC实例详情中查看RDMA IP地址，如下图。

弹性高性能计算集群LGN5

适用场景：

- 人工智能超大规模训练：例如无人驾驶、NLP大模型、广告推荐等场景
- 高性能计算：生命科学、计算物理等场景

规格特点：

- 具备百度太行·弹性裸金属服务器的特性
- GPU镜像包含GPU驱动、CUDA、OFED网卡驱动等必备环境。
- 处理器：Intel Xeon Platinum 8350C，主频 2.6GHz，睿频 3.1GHz
- GPU：NVIDIA Tesla A800 SXM，FP16算力达到312TFLOPS，单GPU显存80GB HBM2，支持双向带宽400GB的Nvlink互联
- 存储：可支持16TB容量全闪存本地SSD存储，支持GPU Direct Storage

- 网络：支持180G VPC网络带宽及800G的RDMA专有网络，并支持GPU Direct RDMA特性

> 注意：VPC网络和RDMA专有网络相互隔离，即实例VPC IP与RDMA IP之间不互通

实例规格	GPU显卡数量	vCPU	内存大小（GiB）	硬盘	内网带宽（Gbps）	RDMA网络带宽（Gbps）	网络收发包（pps）	队列数
ehc.lgn5.c128m1024.8a800.8re.4d	8张	128核	1024	4* 4T NVMe SSD	180	RoCE v2 800	3000万	32

GPU渲染型

渲染型vGN3-Q

> 该规格族目前以白名单形式开放，如有需要可以提交工单

搭配NVIDIA主流数据中心GPU T4以及license vDWS授权，支持3D内容创作应用例如OpenGL图形显示、DirectX等图形功能，方便用户使用GPU的全部功能

- 适用场景：
  - 图形图像处理，例如3D渲染，光线追踪，云游戏，图形数据库，视频编解码等场景。
- 规格特点：
  - 处理器：Intel Xeon Gold 6271C，主频 2.6GHz，睿频 3.1GHz
  - GPU：NVIDIA Tesla T4 vGPU，GPU显存支持2GB、4GB、8GB显存
  - 内存：最高可支持40GiB内存
  - 存储：支持CDS云磁盘

实例规格

实例	T4显卡数量	GPU显存大小（GiB）	vcpu数量	内存大小（GB）	内网带宽（Gbps）	网络收发包（万pps）
bcc.vgn3.c2m8.1t4-2C	1/8张	2	2	8	1.5	40
bcc.vgn3.c4m16.1t4-4C	1/4张	4	4	16	1.5	80
bcc.vgn3.c10m40.1t4-8C	1/2张	8	10	40	4	150

产品定价

计费模式

GPU云服务器的计费模式和BCC云服务器一致，可参考BCC云服务器计费概述。

计费价格

GPU云服务器包含多种实例规格，具体请参见GPU价格详情。

快速入门

GPU云服务器提供GPU算力服务，百度智能云采用成熟的PassThrough技术将GPU卡透传给云服务器，使每位用户都能独享一块或多块GPU卡，保证GPU性能的稳定性。

在购买GPU云服务器前，应先完成注册百度智能云账号以及实名认证

实名认证

请根据自身情况，进行”企业认证”或者“个人认证”，具体请参考认证流程。

如果您在创建GPU实例的时候仍未完成实名认证，你可以点击页面上的『认证』提示按钮到实名认证页面完成相关操作。

操作流程介绍

完成前置条件后，您可通过创建GPU云服务器来按需创建、管理GPU云服务器。

操作指南

创建GPU实例

操作步骤

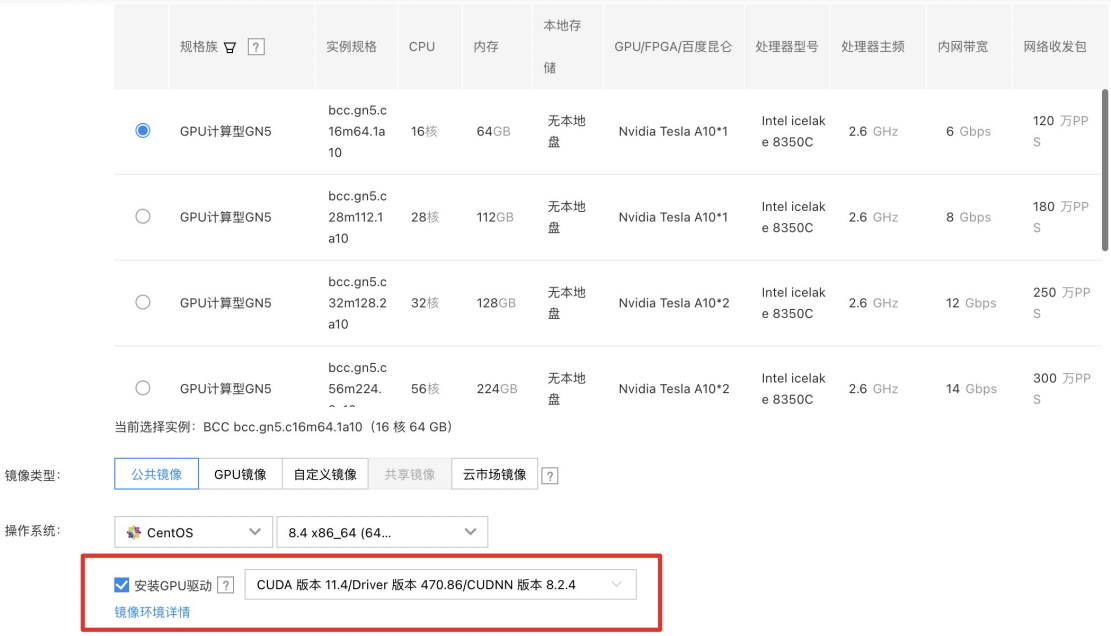
基本配置

1. 登录BCC管理控制台主界面。
2. 登录成功后，选择“产品服务>云服务器 BCC”，进入“实例列表”页面，点击“创建实例”。
3. 根据需要选择当前地域，可用区，付费方式。GPU云服务器支持预付费、后付费和抢占实例付费模式，具体配置信息如下。

配置信息	说明
类型	GPU 实例
付费方式	预付费（包年包月）、后付费（按需购买）、抢占实例
可选地域	华北-北京、华南-广州、华北-保定、华东-苏州
可用区	可用区是指在同一区域下，电力和网络互相独立的区域，故障会被隔离在一个可用区内
网络类型	当前服务器所属的虚拟私有网络，缺省情况下系统默认私有网络

4. 在配置栏中架构选择**异构计算GPU/FPGA/NPU**并选择需要的实例分类。
5. 选择镜像，创建GPU实例支持安装多种镜像类型。

- 公共镜像：推荐使用。由百度智能云官方提供，包含基础操作系统环境。创建GPU计算型规格族时，使用Linux公共镜像同时可勾选**安装GPU驱动**来指定使用GPU所需要的CUDA、驱动、CUDNN版本，如下图。实例创建后会自动执行安装驱动的脚本命令，等待10-15分钟后即可获取GPU运行环境，之后可使用该实例，或者打包自定义镜像，以自定义镜像方式快速创建新的GPU实例。建议使用最新的GPU驱动环境以获取最新最全的GPU功能。



当前系统支持的GPU环境版本您可参考GPU驱动选装发布记录查看。

自动安装驱动的脚本也可以通过自行复制到用户数据注入（详情见本文档第8条中的高级配置）来手动执行安装，或者在实例创建完后以命令行方式执行，脚本的内容如下，其中DRIVER_VERSION、CUDA_VERSION、CUDNN_VERSION可更换为兼容的其他版本号。

```
#### !/bin/bash

set -ue

DRIVER_VERSION="470.86"
CUDA_VERSION="11.4"
CUDNN_VERSION="8.2.4"
WORK_DIR="/root/auto_install"
SCRIPT_URL="http://mirrors.baidubce.com/nvidia-binary-driver/api/auto_install.sh"

mkdir ${WORK_DIR}
cd ${WORK_DIR}
wget --timeout=10 -t 10 ${SCRIPT_URL}
bash ${WORK_DIR}/$(basename ${SCRIPT_URL}) ${DRIVER_VERSION} ${CUDA_VERSION} ${CUDNN_VERSION}
rm -rf ${WORK_DIR}
```

注：如勾选安装GPU驱动选项，实例后台会自动执行安装脚本，预计耗时10到15分钟。部分实例在执行完安装脚本后会自动重启一次。在安装过程中对实例进行关机、重装、重启等操作都会造成安装失败，可登录实例查看最新的安装进展并在安装结束后正常使用实例，安装进展的日志可通过如下命令查看。

```
cat /root/install_info.log
```

- GPU镜像：由百度智能云官方提供，包含主流的基础操作系统环境和固定的GPU驱动及CUDA版本，使用此类型的镜像可以快速获取运行GPU的必备环境，以下为镜像版本详情

支持的GPU专用镜像	CUDA版本	深度学习框架版本	支持的GPU规格族
Ubuntu 16.04 LTS amd64 (64bit)-CUDA8.0	CUDA 8.0	无	LGN1、GN3
Ubuntu 16.04 LTS amd64 (64bit)-CUDA9.0	CUDA 9.0	无	LGN1、GN3
Ubuntu 16.04 LTS amd64 (64bit)-CUDA9.1	CUDA 8.1	无	LGN1、GN3
Ubuntu 16.04 LTS amd64 (64bit)-CUDA9.2	CUDA 9.2	无	LGN1、GN3
Ubuntu 16.04 LTS amd64 (64bit)-CUDA10.0	CUDA 10.0	无	LGN1、GN3
Ubuntu 16.04 LTS amd64 (64bit)-CUDA10.1	CUDA 10.1	无	LGN1、GN3
16.04 LTS amd64 (64bit)-CUDA9.0-framework-integration	CUDA 9.0	深度学习开发镜像集成 TensorFlow_gpu 1.10.1、PaddlePaddle_gpu 0.14.0、Caffe2	LGN1、GN3
Ubuntu 16.04 LTS amd64 (64bit)-CUDA8.0-paddlepaddle_0.11.0	CUDA 8.0	PaddlePaddle_gpu 0.11.0	LGN1、GN3
Ubuntu 16.04 LTS amd64 (64bit)-CUDA8.0-tensorflow_1.3.0	CUDA 8.0	TensorFlow_gpu 1.3.0	LGN1、GN3
CentOS 7.5 x86_64 (64bit)-CUDA9.0	CUDA 9.0	无	LGN1、GN3
CentOS 7.5 x86_64 (64bit)-CUDA9.1	CUDA 9.1	无	LGN1、GN3
CentOS 7.5 x86_64 (64bit)-CUDA9.2	CUDA 9.2	无	LGN1、GN3
CentOS 7.5 x86_64 (64bit)-CUDA10.0	CUDA 10.0	无	LGN1、GN3
CentOS 7.5 x86_64 (64bit)-CUDA10.1	CUDA 10.1	无	LGN1、GN3
CentOS 6.8 x86_64 (64bit)-CUDA9.2	CUDA 9.2	无	LGN1、GN3
CentOS 7.5 x86_64 (64bit)-CUDA11.2	CUDA 11.2	无	GN5
CentOS 7.5 x86_64 (64bit)-CUDA11.4	CUDA 11.4	无	LGN1、GN3、GN5

- 其他类型镜像：例如自定义镜像，市场镜像的使用方式和云服务器BCC的使用方式一致，可参考创建实例。

6. 选择 **存储** 相关配置

配置信息	必选/可选	说明
系统盘	必选	用于安装操作系统。 非异构实例且镜像OS是Linux，默认大小20GB。若镜像OS是Windows，默认大小40GB。异构实例不区分操作系统默认均为40GB。 根据地域以及实例规格的不同可供选择的云盘类型也不同，以页面实际提供云盘类型为准。
CDS云盘	可选	即挂载的数据盘，用于提高云服务器的存储容量。默认不选择，您可根据需求选择云磁盘的容量和挂载数量，最多可同时挂载5块云磁盘。
快照策略	可选	默认关闭绑定快照策略。通过快照，您可以实现磁盘数据备份，磁盘数据恢复以及磁盘镜像的制作。

网络和存储配置

7. 配置以下信息，并单击 下一步。

配置项	必选/可选	说明
网络类型	必选	如果您没有创建私有网络，可以选择默认私有网络。目前，百度智能云私有网络VPC、子网、安全组、ACL、路由表免费，关于私有网络的更多介绍可参考 私有网络VPC。
安全组	必选	如果您没有创建安全组，可以选择默认安全组。
弹性资源	可选	如需公网访问请购买弹性公网IP，或购买成功后绑定已有弹性公网IP。 公网带宽支持以下购买方式。 包年包月计费：需要提前一次性支付所选时间段内的带宽费用，购买预付费BCC时，费用将合并在实例中收取。 按使用流量计费：根据用户实际传输的数据量计费，流量使用没有上限，但可限定最大峰值带宽。 按使用带宽计费：根据用户选择的固定带宽值进行计费，最大可购买200Mbps的带宽。

信息配置

8.配置以下信息，并单击 **确认购买**。

配置项	必选/可选	配置说明
标签	可选	当您拥有多台实例时，可以通过设置标签云服务器实现资源的分类管理。具体可参考 标签 进行设置。
部署集	可选	部署集在指定部署集中创建BCC实例时，会和处于同一部署集中的其他BCC实例严格按照物理服务器打散，保障在硬件故障等异常情况下的服务高可用性。具体可参考 部署集 设置。当前已支持选择最多2个部署集。
系统信息	必选	根据您的实际对系统信息进行设置。 • **实例名称：**您可自定义设置实例名称或由系统随机生成实例名称。 • **主机名：**您可自定义设置主机名（hostname）或由系统随机生成主机名。 • **有序后缀：**用户选择打开此选项，则有序后缀从 0001 开始递增，最大不能超过 9999。例如：InstanceName0001，InstanceName0002 和 HostName0001，HostName0002。 • **Domain开关：**用户打开后，主机名将带有domain后缀，可以支持DNS解析。 • **管理用户名：**Windows系统的管理员账号为Administrator，Linux系统的管理员账号为root。 • **管理员密码：**根据实例操作系统的不同，密码可供选择的设置方式也不同。 ◦ 用户自定义：自定义设置登录实例的密码。

		◦ 随机生成：购买成功后需登录控制台重置密码，参考 重置密码。 ◦ 密钥对：Linux操作系统可以选择使用密钥对的方式连接云服务器，SSH密钥对是一种比常规密码更安全的登录云服务器的方式，具体可参考 密钥对 设置。
高级配置	可选	• **用户数据注入：**作为实例自定义脚本在启动实例时执行，实现自动化配置实例，例如获取并安装软件资源包、开启服务、打印日志、初始化服务环境等操作。可使用User-Data脚本（仅在实例首次启动时运行一次）。通过以下示例，实例创建成功后可在/root/test文件中看到Welcome to Baidu AI Cloud。 示例： #!/bin/sh echo "Welcome to Baidu AI Cloud." | tee /root/test
购买信息	必选	• **购买时长(预付费)：**1-9个月或1-3年选择配置。时长周期为实例创建日起的单位周期。如2015年5月20日购买BCC服务一个月，则使用周期为2015年5月20日-2015年6月20日（默认单位月份时间为30天）。 • **自动续费（预付费）：** ◦ 默认不勾选，不进行自动续费。 ◦ 勾选后可选择续费周期。 • **自动释放（后付费）：** ◦ 默认不勾选，不进行自动释放。 ◦ 勾选后实例将在指定时间自动释放，具体可参考 定时释放。

9. 在 **确认订单** 页核对各项配置信息及费用明细，点击提交订单进入支付环节。

> **说明：** 如果您有百度智能云代金券，可以使用代金券结算抵扣费用，如果代金券金额不足，则可以使用银行卡完成支付操作。

10. 单击 **确认付款**，完成支付。支付成功后，系统在后台进行云服务器的创建。
11. 点击链接 **管理控制台**，进入 **实例列表** 界面，查看创建实例的状态。
    - 创建时，实例状态为 **创建中**。
    - 创建成功后，实例状态显示为 **运行中**。
12. 实例创建成功后，系统将以短信形式发送云服务器的信息给用户，包括IP和用户名。为了安全因素考虑，密码将不再通过短信形式发送，如果您忘记创建实例时配置的密码可以重置密码。

管理GPU实例

您可以对当前账户中的GPU执行登录、查看、启动、停止、重启、释放、配置变更、续费、计费变更、重置密码、名称修改等操作，具体可参考BCC。

- 登录实例
- 查看实例
- 启动实例
- 停止实例
- 重启实例
- 释放实例
- 实例配置变更
- 实例续费
- 计费变更
- 重置密码
- 实例名称修改

安装GPU驱动

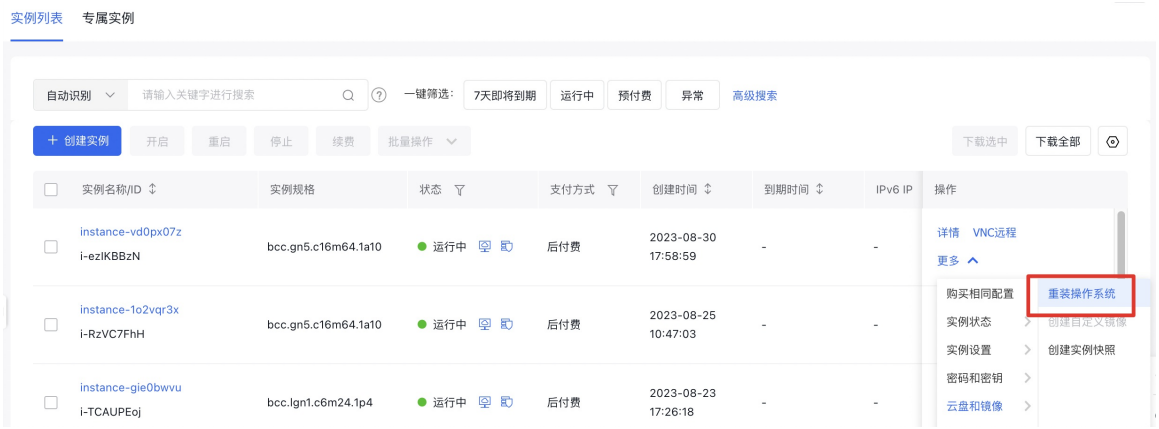
自动安装GPU驱动及CUDA（推荐）

GPU的驱动和CUDA是使用GPU计算的必备组件。您在使用GPU云服务器的过程中，可实现自动为GPU实例安装驱动及CUDA。

通过控制台勾选自定义GPU驱动安装驱动

创建GPU实例 请参考创建实例，并按照向导选择安装GPU驱动。该操作会自动帮助您安装GPU驱动、CUDA、Cudnn、DCGM以及Fabric manager（如果GPU包含NVSwitch）。 **重装GPU实例** 您可通过重装实例为已经运行中的实例安装或更新所需要的GPU驱动，操作如下：

登录云服务器控制台，并为需要安装或者更新GPU驱动的实例点击重装。



在弹窗中选择需要的公共镜像的操作系统，并勾选安装GPU驱动，选择需要的版本。

通过控制台或API的数据注入安装驱动

在创建、重装实例时可以通过注入如下的安装脚本实现安装GPU驱动：

API数据注入参数：

userData	String	否	RequestBody参数	若实例满足使用实例自定义数据的限制，您可传入UserData信息。因为传输API请求时，不会加密您设置的UserData，建议不要以明文方式传入机密的信息，例如密码和私钥等。如果必须传入，建议加密后，然后以Base64的方式编码后再传入，在实例内部以同样的方式反解密。

控制台

数据注入入口：

脚本内容，替换如下脚本中的DRIVER_VERSION，CUDA_VERSION，CUDNN_VERSION参数为所需要的版本号，建议您参考建议安装的GPU驱动列表选择版本：

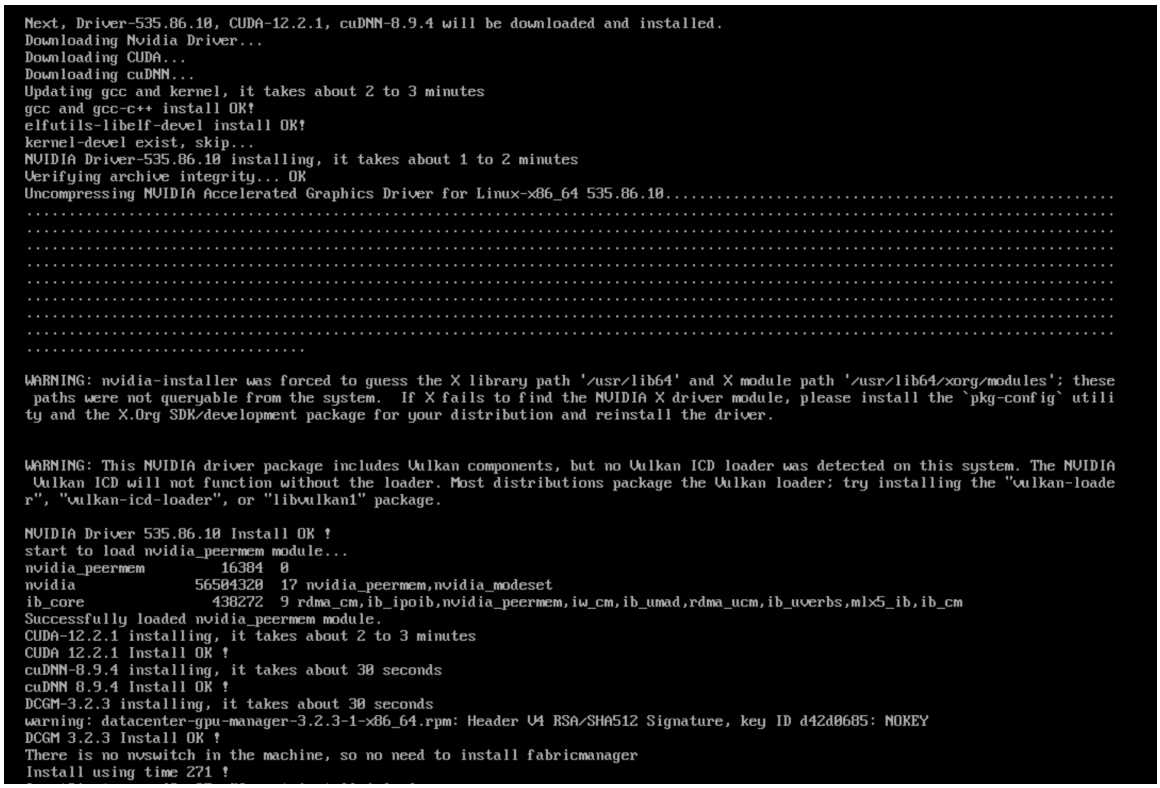
```
 ##### !/bin/bash
DRIVER_VERSION="535.216.03"
CUDA_VERSION="12.5.1"
CUDNN_VERSION="9.6.0"
WORK_DIR="/root/auto_install"
SCRIPT_URL="http://mirrors.baidubce.com/nvidia-binary-driver/api/auto_install.sh"

mkdir ${WORK_DIR}
pushd ${WORK_DIR}
for ((i=0; i<120; i++))
do
    wget --timeout=10 -t 10 ${SCRIPT_URL}
    if [ $? -eq 0 ]; then
        break
    else
        sleep 1
    fi
done
bash ${WORK_DIR}/$(basename ${SCRIPT_URL}) ${DRIVER_VERSION} ${CUDA_VERSION} ${CUDNN_VERSION}
popd
rm -rf ${WORK_DIR}

cmdline=$(cat /proc/cmdline)
if [[ "${cmdline}" =~ "pci=realloc" ]]; then
    echo "remove 'pci=realloc' cmdline arg and update grub"
    default_grub_arg="/etc/default/grub"
    sed -i 's/pci=realloc//g' ${default_grub_arg}
    if command -v grub2-mkconfig; then
        efi_grub_cfg=/boot/efi/EFI/centos/grub.cfg
        if [ -f /boot/efi/EFI/rocky/grub.cfg ]; then
            efi_grub_cfg=/boot/efi/EFI/rocky/grub.cfg
        fi
        grub2-mkconfig -o $efi_grub_cfg
    fi
    if command -v update-grub; then
        update-grub
    fi
    reboot
else
    echo "there is no 'pci=realloc' arg in current cmdline, do nothing"
fi
```

查看自定义GPU驱动安装进展

在实例状态变为运行中后，登录实例可通过以下命令查看当前安装GPU驱动的进展：

```
 cat install_info.log
```

看到如下的安装提示后，可通过nvidia-smi检查驱动是否正常工作。



注意：安装过程中请避免执行重启实例、重装实例等涉及实例关机的操作，否则安装无法完整执行

配置BCM事件通知 如您的业务系统需要自动化获取GPU驱动安装状态，可订阅BCM事件中的驱动安装成功并配置报警策略实现，系统将在实例中的GPU驱动安装完成后推送此事件。

手动安装GPU驱动以及CUDA（Linux）

背景

GPU的驱动和CUDA是使用GPU计算的必备组件。如果您在创建实例始时，没有选择自动安装驱动以及CUDA版本，则需要在创建实例后，手动安装。本页面介绍如何手动安装驱动，以及如何使用工具测试GPU的重要性能。

前提条件

已创建GPU计算型实例，例如GN3或者GN2规格族。本流程不适合需要安装GRID驱动的vGPU规格族。

安装前准备工作

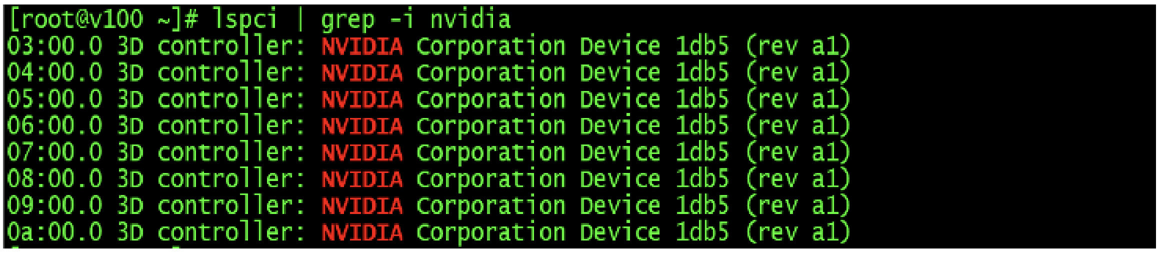
1. 确认GPU型号和操作系统版本，本示例中以V100以及操作系统为Centos 7.5.1804进行操作。
2. 准备GPU驱动和CUDA 11.1软件包，在nvidia官网进行驱动包和CUDA包下载
   - linux系统均选择 Linux 64-bit
   - CUDA Toolkit选择最新版本
   - 如您需要老版本CUDA，请前往老版本CUDA下载

   本示例中使用CUDA 11.1.1。
3. 检查服务器GPU识别情况

   安装GPU驱动之前需要在操作系统下查看GPU卡是否能够完全识别,如不能识别需要进行重新插拔、对调测试等步骤进行硬件排查。

   查看到所有的GPU

   ```
   lspci | grep -i nvidia
   ```



4. 老版本软件包卸载（可选）
   - GPU驱动卸载

   ```
   /usr/bin/nvidia-uninstall
   ```

   - CUDA卸载方法：

   ```
   /usr/local/cuda/bin/cuda-uninstaller
   ```


```
- 如您的CUDA是老版本卸载

  /usr/local/cuda/bin/uninstall_cuda_X.Y.pl
```

5. 安装gcc、g++编译器

   CUDA安装samples测试程序进行make时需要g++，但安装CUDA软件包时不需要。

```
检查版本
gcc -v
g++ -v
```

软件包安装

```
yum install gcc
yum install gcc-c++
```

6. 禁用系统自带的nouveau模块

   检查nouveau模块是否加载，已加载则先禁用

```
lsmod | grep nouveau
```

```
[root@v100 ~]# lsmod | grep nouveau
nouveau              1869738  0
mxm_wmi                13021  1 nouveau
wmi                    21636  2 mxm_wmi,nouveau
video                  24538  1 nouveau
i2c_algo_bit           13413  1 nouveau
drm_kms_helper        179394  2 cirrus,nouveau
ttm                   114635  2 cirrus,nouveau
drm                   429744  5 ttm,drm_kms_helper,cirrus,nouveau
```

   没有blacklist-nouveau.conf文件则创建

```
vim /usr/lib/modprobe.d/blacklist-nouveau.conf
blacklist nouveau
options nouveau modeset=0
```

执行如下命令使内核生效（需要重启服务器后才可真正禁用nouveau）

```
dracut -force
```

重启操作系统

7. 修改系统运行级别为文本模式 GPU驱动安装必须在文本模式下进行

```
systemctl set-default multi-user.target
```

8. 重启系统，然后检查禁用nouveau模块配置与文本模式是否生效。

```
lsmod | grep nouveau
```

```
[root@v100 ~]# lsmod | grep nouveau
[root@v100 ~]#
```

GPU驱动安装

1. root用户下进行GPU驱动

```
chmod +x NVIDIA-Linux-x86_64-450.80.02.run
./NVIDIA-Linux-x86_64-450.80.02.run --no-opengl-files --ui=none --no-questions --accept-license
```

2. 配置GPU驱动内存常驻模式

```
nvidia-persistenced
```

设置开机自启动

```
vim /etc/rc.d/rc.local
```

在文件中添加一行

```
nvidia-persistenced
赋予/etc/rc.d/rc.local文件可执行权限

chmod +x /etc/rc.d/rc.local
```

若无/etc/rc.d/rc.local，也可修改

```
vim /etc/rc.local
chmod +x /etc/rc.local
```

3. 安装完GPU驱动后查看GPU状态查看及相关配置。

```
nvidia-smi
```

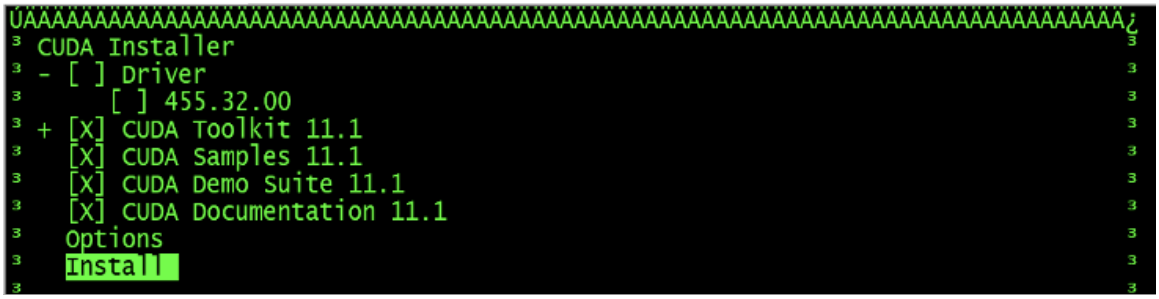
CUDA安装

1. 安装CUDA

安装CUDA时需注意，如果已经安装过GPU驱动，安装CUDA时就不要再选择GPU驱动安装了。

```
chmod +x cuda_11.1.1_455.32.00_linux.run
./cuda_11.1.1_455.32.00_linux.run --no-opengl-libs
```

新版本CUDA安装界面： 注意Driver选项，表示是否安装GPU驱动，如果已经安装了GPU驱动，这里不要再勾选。



2. 配置环境变量

添加到/etc/profile文件中，对所有用户生效

```
vim /etc/profile
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH
source /etc/profile
```

测试CUDA安装是否正确，环境变量是否识别成功

```
nvcc -V
```

CUDA samples程序测试

1. BandwidthTest测试

BandwidthTest测试GPU卡与主机server、GPU与GPU卡之间的显存带宽，测试结果中Host to Device Bandwidth和Device to Host Bandwidth分别为主机server至GPU卡和GPU卡至主机server的显存带宽，Device to Device Bandwidth测试的是GPU卡之间的显存带宽。

编译、测试

```
 cd /usr/local/cuda/samples/1_Utilities/bandwidthTest/
make
./bandwidthTest
```

测试结果参考 Host to Device Bandwidth和Device to Host Bandwidth两项数值在6.6GB/s-7.1GB/s之间

```
 Device to Device Bandwidth测试显存带宽，该值根据GPU显存硬件配置而异，
下表列出个GPU理论显存带宽，实测值在理论值70%以上即认定状态正常。
```

GPU型号	显存带宽理论值
P100-PCIE-12GB	548GB/s
P100-PCIE-16GB/P100-SXM2-16GB	732GB/s
P40	346GB/s
P4	192GB/s
V100-PCIE/SXM2全系列	900GB/s

2. P2pBandwidthLatencyTest测试

P2pBandwidthLatencyTest测试GPU卡之间的带宽。可查看是否使用GPU中NVLINK的性能结果

编译、测试

```
 cd /usr/local/cuda/samples/1_Utilities/p2pBandwidthLatencyTest/
make
./p2pBandwidthLatencyTest
```

```
P2P Connectivity Matrix
     D\D     0     1     2     3     4     5     6     7
     0       1     1     1     1     0     0     0     1
     1       1     1     1     1     0     0     1     0
     2       1     1     1     1     0     1     0     0
     3       1     1     1     1     1     0     0     0
     4       0     0     0     1     1     1     1     1
     5       0     0     1     0     1     1     1     1
     6       0     1     0     0     1     1     1     1
     7       1     0     0     0     1     1     1     1
Unidirectional P2P=Disabled Bandwidth Matrix (GB/s)
   D\D     0      1      2      3      4      5      6      7
     0 767.44   6.02   5.88   6.53   6.42   6.76   6.70   5.88
     1   6.05 770.84   5.90   6.42   6.42   6.74   6.70   5.86
     2   6.53   6.41 770.84   6.35   6.49   6.83   6.72   5.95
     3   6.49   6.41   5.71 771.60   6.44   6.77   6.72   5.91
     4   6.42   6.42   5.83   6.52 771.60   6.36   6.76   5.86
     5   6.44   6.41   5.83   6.42   5.98 770.84   6.76   5.83
     6   6.44   6.41   5.84   6.40   6.45   6.80 770.84   5.46
     7   6.44   6.44   5.85   6.49   6.43   6.81   6.31 771.60
Unidirectional P2P=Enabled Bandwidth (P2P Writes) Matrix (GB/s)
   D\D     0      1      2      3      4      5      6      7
     0 768.95  24.25  48.48  24.26   6.45   6.78   6.78  48.49
     1  24.26 772.75  24.25  48.48   6.45   6.78  48.48   5.89
     2  48.48  24.25 771.60  48.48   6.50  24.25   6.80   5.98
     3  24.25  48.48  48.48 771.60  24.25   6.86   6.77   5.92
     4   6.42   6.42   5.82  24.25 771.22  48.49  48.48  24.25
     5   6.43   6.40  24.25   6.53  48.48 771.60  24.25  48.48
     6   6.44  48.48   5.85   6.54  48.49  24.25 771.60  24.26
     7  48.48   6.43   5.85   6.43  24.25  48.48  24.25 771.60
Bidirectional P2P=Disabled Bandwidth Matrix (GB/s)
   D\D     0      1      2      3      4      5      6      7
     0 771.41   7.22   9.04   9.53   9.45   9.62   9.56   9.11
```

解释：

- 互联矩阵中，有1标记即表示GPU间有P2P访问支持功能，0表示没有支持。
- Unidirection测试单向P2P带宽，Bidirectiona测试双向P2P带宽。同时提供是否使用NVLINK的性能测试结果。

根据GPU硬件版本不同，带宽和延迟存在差异，按照GPU型号，提供参考数据如下

参考带宽：

GPU型号（带宽实测均值）	支持P2P	不支持P2P
PCIE GPU	22-26GB/s	17-20GB/s
P100-SXM2(NVLINK1.0)	35-38GB/s	17-20GB/s
V100-SXM2(NVLINK2.0)	90-96GB/s（双link）和45-48GB/s （单link）	15-20GB/s

参考时延：

GPU型号（延迟实测均值）	矩阵值1	矩阵值0
PCIE GPU	≤10us	≤20us
P100-SXM2(NVLINK1.0)	≤10us	≤20us
V100-SXM2(NVLINK2.0)	≤10us	≤20us

3. BatchCuBlas

BatchCuBlas为GPU浮点运算能力测试、加压。测试包括sgemm（测试单精度浮点计算能力）和dgemm（双精度）两部分，关注GFLOPS测试值。测试值达到理论值90%以上认定状态正常。

编译、测试

```
 在测试命令中m、n、k值可调整，一般GPU缓存大小16G时可选8192。
在测试命令中，使用--device=1参数指定测试ID号为1的GPU卡，默认测试ID号为0的GPU卡。
 cd /usr/local/cuda/samples/7_CUDALibraries/batchCUBLAS/
 make
 ./batchCUBLAS -m8192 -n8192 -k8192    ##默认测试ID号为0的GPU卡
 ./batchCUBLAS -m8192 -n8192 -k8192 --device=1 ##指定测试ID号为1的GPU卡
```

测试结果中，可重点关注Running N=10 with streams部分，包含单精度和双精度测试，

```
 ==== Running N=10 with streams ====
Testing sgemm
#### args: ta=0 tb=0 m=8192 n=8192 k=8192  alpha = (0x40000000, 2) beta= (0x40000000, 2)
#### args: lda=8192 ldb=8192 ldc=8192
^^^^ elapsed = 0.70538688 sec  GFLOPS=15587.4
@@@@ sgemm test OK
Testing dgemm
#### args: ta=0 tb=0 m=8192 n=8192 k=8192  alpha = (0xbff0000000000000, -1) beta= (0x0000000000000000, 0)
#### args: lda=8192 ldb=8192 ldc=8192
^^^^ elapsed = 1.40515780 sec  GFLOPS=7824.83
@@@@ dgemm test OK
```

手动安装GPU驱动以及CUDA（Windows）

背景

GPU的驱动和CUDA是使用GPU的必备环境。如果您在创建实例时，没有自动安装驱动以及CUDA版本，则需要在创建实例后手动安装。本页面介绍如何在Windows环境中手动安装GPU驱动和CUDA。

前提条件

已创建GPU计算型实例且实例可访问公网，例如GN5规格族。本流程不适合需要安装GRID驱动的vGPU规格族。

GPU驱动安装

1. 远程连接实例，可通过SSH或VNC访问实例。
2. 通过浏览器访问NVIDIA官方驱动下载地址。
3. 根据实例规格及需求的环境版本，选择相应的版本下载驱动包

- 产品类型：选择Data Center/Tesla。
- 产品系列：根据实例规格选择，对应关系如下

实例规格	产品系列
GN5	A-Series
GN3 V100	V-Series
GN3 T4	T-Series
GN1 P4	P-Series

- 产品家族：根据实例类型中的GPU类型选择相应的GPU型号
- 操作系统：根据公共镜像中的操作系统版本选择对应版本。
- CUDA Toolkit：根据需求选择对应版本

推荐您按照以下的兼容性列表选择相应的驱动和CUDA版本，以下版本可兼容百度智能云主售的GN5和GN3实例规格。

CUDA版本	驱动版本	支持的公共镜像
12.0	527.41	Windows Server 2016、2019、2022
11.7	517.71、516.94、516.94	Windows Server 2016、2019、2022
11.6	513.91、513.46、512.78、511.65	Windows Server 2016、2019、2022

4.点击搜索后，选择需要的驱动版本即可完成下载。

5.下载完成后，双击安装文件，按提示完成安装。

6.检查驱动已成功安装，前往设备管理器界面中的显示适配器，查看是否显示GPU型号名称及驱动信息，如果显示则代表安装

成功

为GPU实例安装GRID驱动(Windows)

当您的GPU计算型实例需要支持OpenGL等图形显示时，需要安装NVIDIA官方发布的GRID驱动来使用OpenGL等更丰富的图形功能。另外，仅仅安装GRID驱动后实例配备的NVIDIA Tesla系列计算卡尚未激活NVIDIA GRID License，此时GPU图形功能受到限制。为获得完整的图形功能，需要进行激活操作。本文以操作系统为Windows Server 2019 64-bit的GPU计算型实例为例，介绍如何安装GRID驱动。

前提条件

1. 准备好一台windows GPU实例，并绑定EIP，保证可以访问公网。
2. 使用Windows 远程桌面或者在本地机器上安装远程连接工具，例如VNC Viewer。
3. 可以选择购买官方授权（入口）或者申请 90 天试用版授权（入口） 获取GRID License。

- 该方式需要自建License服务器，您可以购买BCC实例并参考NVIDIA官网教程自行搭建。
- 临时测试授权仅供评估用途，不能用于生产任务。

4. 提交工单，获取兼容的GPU GRID驱动。

操作步骤

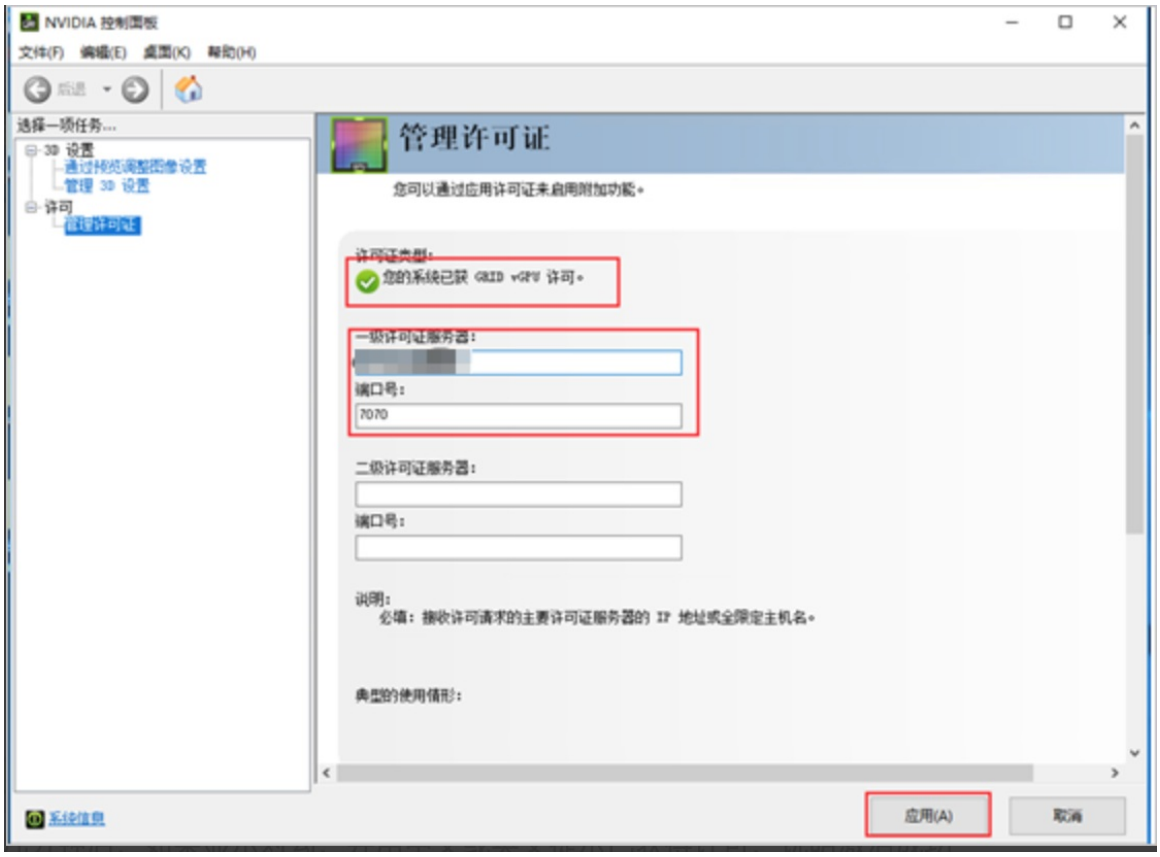
1. 远程连接BCC实例。
2. 打开GRID驱动安装包，按提示完成安装。

> 说明：对于Windows系统，GPU驱动安装生效后，Windows自带的远程连接（RDP）协议有可能不支持DirectX、OpenGL等相关应用，您需要自行安装VNC服务和客户端。

3. 重启实例。
4. 激活驱动产品许可。

安装完成GRID驱动后，在控制面板中会有NVIDIA的程序选项，点击打开。

在管理 License 选项中填入 License Manager 地址,端口默认 7070(注意检查防火墙设置，确保防火墙对该端口网络通信放行)，点击应用。



几分钟后，如果显示对勾，并用中文或英文提示已获得许可，说明激活成功。

数据上传

数据上传

百度智能云提供对象存储BOS服务，这是一个类似于百度网盘的存储空间，提供简单的客户端和CLI工具来进行使用。

对于GPU使用过程中需要用到的数据集，您可以先将他们统一上传到您的对象存储中，在需要用GPU作训练时，再将数据拉取到GPU云服务器的本地进行处理。

BOS设置

1. 访问BOS产品页，点击立即购买登录控制台；
2. 在Bucket管理页面点击新建Bucket，命名后选择所属地域（建议选择与GPU服务器相同的地域），并设置读写权限为私有；
3. 创建完成后，在Bucket管理页面点击创建完成的Bucket名字进入Bucket，即可在web上查看和上传相关数据文件；
4. 文件上传完成后，点文件名即可获取文件地址，在GPU服务器平台上直接通过wget方式获取。

BOS在GPU服务器上的使用功能方法

BOS CLI是BOS调用的命令行工具，以下介绍如何在GPU云服务器中使用BOS CLI工具。

1. 从BOS CLI工具文档页获取linux下BOS CLI工具的压缩包。
2. 解压CLI工具包:

```
$ unzip bce-cli-0.8.3.zip
```

3. 将bcecli的库安装到系统的python目录下:

```
$ python setup.py install
```

4. 使用BOS CLI工具之前，推荐先设置Access Key、Secure Key、Region；可以通过 -c/–configure 来设置AK、SK、Region，CLI会在配置后自动写入当前用户主目录；AK、SK可以从登录BOS控制台后，点击右上角按钮悬窗中Access Key链接获得。Region则应与BOS设置中，所属地域的选择一致，选择北京填bj，选择广州则填gz，后续选项均可以直接回车填默认值。

```
 $ bce -c
$ BOS Access Key ID [None]: Enter Your AK
$ BOS Secret Access Key [None]: Enter Your SK
$ Default region name [bj]: Enter Your region
$ Default domain [bj.bcebos.com]: Enter Your host
```

5. 以将GPU云服务器上的 text.txt文件上传至mybucket为例，上传命令如下：

```
$ bce bos cp text.txt bos:/mybucket/text.txt
```

BOS的其他功能及计费

参见BOS文档

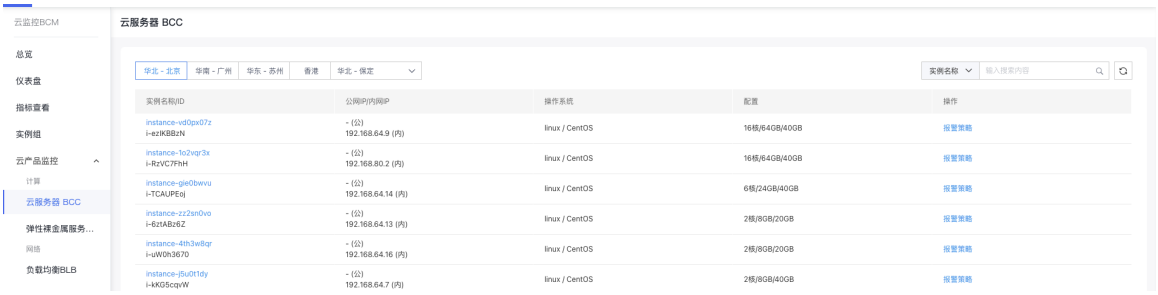
查看GPU云服务器监控

对比BCC云服务器，GPU云服务器中包含额外的硬件，例如GPU卡以及RDMA网卡。GPU云服务器支持对这些硬件资源进行监控，相关的监控指标项可在云产品指标列表中查询。以下为如何查看GPU实例的监控数据。

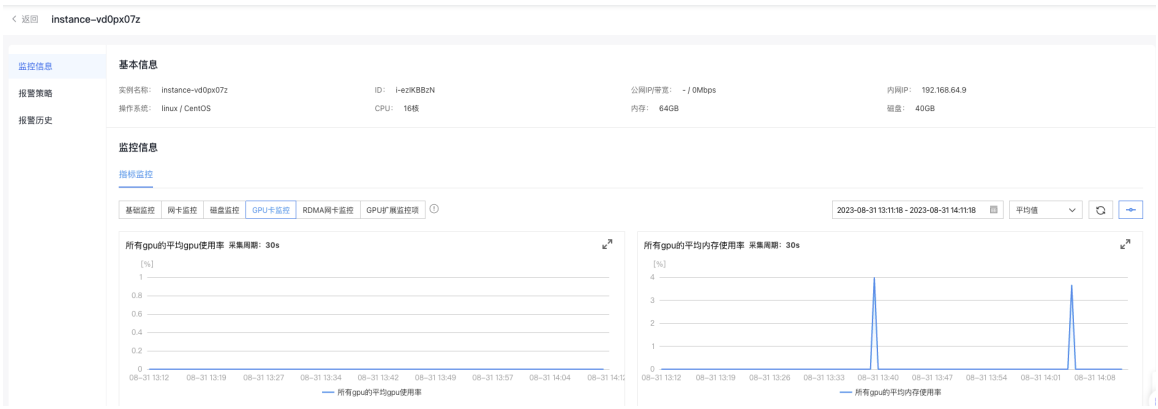
GPU监控和GPU扩展监控依赖实例已安装了GPU驱动，如何安装GPU驱动可参考 GPU创建。

GPU监控

1. 选择“产品服务>云监控BCM”，左侧侧边栏选择云产品监控并选择云服务器BCC。

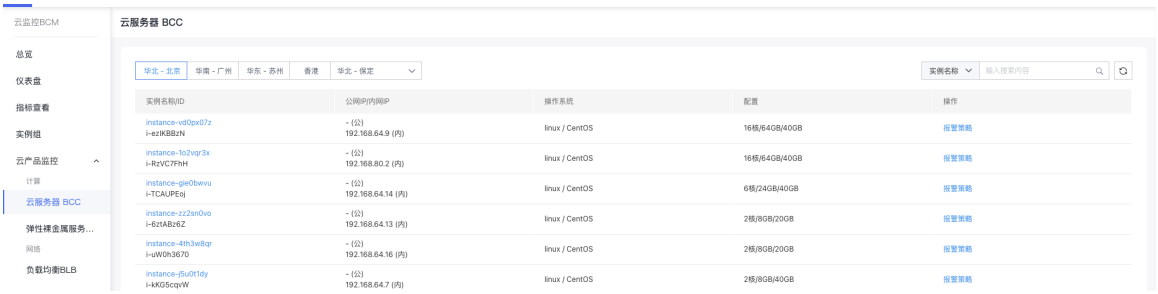


2. 选择您需要查看监控数据的实例，点击GPU卡监控页面，在此页面可以查看GPU卡监控。



RDMA监控

1. 选择“产品服务>云监控BCM”，左侧侧边栏选择云产品监控并选择云服务器BCC。

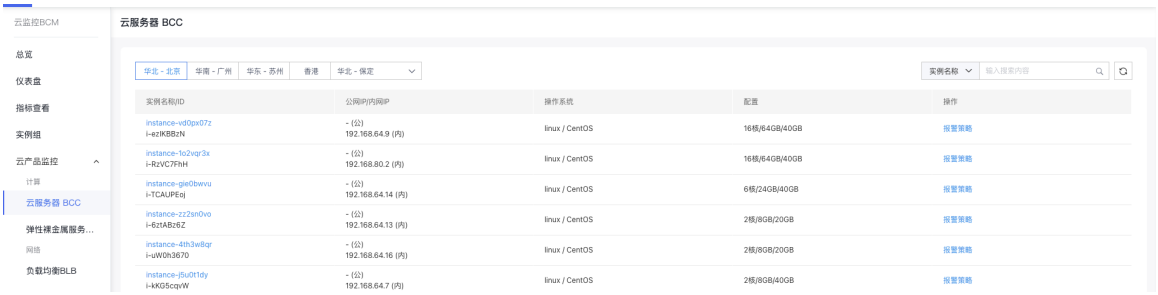


2. 选择您需要查看监控数据的实例，点击RDMA网卡监控页面，在此页面可以查看RDMA网卡监控。

GPU扩展监控

如标准的GPU监控项无法满足您的数据采集需求，GPU云服务器可提供额外的GPU监控项。由于GPU扩展监控项会带来较高的工作负载，请您结合业务情况按需开启。以下为如何查看GPU实例的GPU扩展监控数据。

1. 选择“产品服务>云监控BCM”，左侧侧边栏选择云产品监控并选择云服务器BCC。



2. 选择您需要查看监控数据的实例，点击GPU卡监控页面，在此页面可以查看GPU扩展监控。

GPU扩展监控依赖**3.0以上的DCGM组件**，可登录实例并通过以下命令查看dcgmi版本

```
dcgmi --version
```

通过以下命令启动相关服务：

```
nv-hostengine
systemctl restart bcm-agent
```

回显如下：

```
[root@instance-vd0px07z ~]# nv-hostengine
Host engine already running with pid 67498
[root@instance-vd0px07z ~]# systemctl restart bcm-agent
[root@instance-vd0px07z ~]#
```

如果当前实例未安装dcgm组件，您可选择以下的方式之一安装：

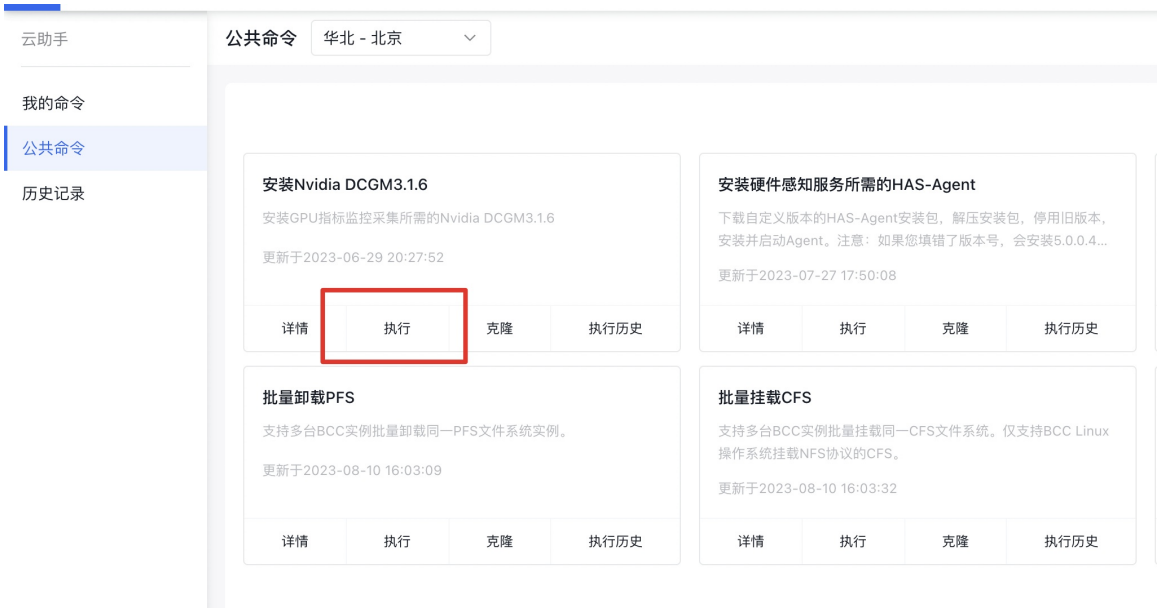
方法一：通过BCC控制台自选GPU驱动安装

在创建实例或者重装实例时，如您选择了公共镜像和自选GPU驱动，系统会自动帮您安装DCGM组件。

```
请选择安装450版本以上的驱动以获取符合要求的dcgm版本
```

方法二：通过云助手安装

登录云助手控制台并 选择执行安装Nvidia DCGM



典型实践

搭建PaddlePaddle环境完成文本情感分类

背景

飞桨（PaddlePaddle）以百度多年的深度学习技术研究和业务应用为基础，集深度学习核心训练和推理框架、基础模型库、端到端开发套件、丰富的工具组件于一体。本章将以bcc.vgn3（Ubuntu18.04 LTS）为例，介绍如何快速搭建飞桨的GPU环境并使用预训练的模型完成文本情感分类预测。

前提条件

- 拥有一台GPU实例如gn3、gn2、vgn3等
- 在GPU实例中安装Cuda、GPU driver，Cuda版本建议为11.2、10.2或10.1。
- 已购买EIP可访问公网

操作步骤

登录GPU实例，并查看Cuda、GPU驱动版本

```
nvidia-smi
```

查看当前的python环境，确认python版本为a3.6/3.7/3.8/3.9

```
which python
python --version
```

安装pip3并升级

```
apt-get install python3-pip
pip3 install -U pip
```

安装paddlepaddle-gpu框架 本章以Cuda 10.2为例，如您需使用其他cuda版本搭建paddlepaddle环境，请查看其他Cuda版本安装。

```
python -m pip install paddlepaddle-gpu -i https://mirror.baidu.com/pypi/simple
```

安装预训练模型管理工具paddlehub

```
pip install paddlehub
```

使用paddlehub下载预训练模型senta_bilstm

```
hub install senta_bilstm==1.2.0
```

运行senta_bilstm

```
hub run senta_bilstm --input_text "我爱人工智能"
```

显示预测结果如下，模型预测文本为正向情感。

```
[{'text': '我爱人工智能', 'sentiment_label': 1, 'sentiment_key': 'positive', 'positive_probs': 0.9409, 'negative_probs': 0.0591}]
```

基于GPU实例部署NGC环境

背景介绍

NGC，Nvidia GPU Cloud是由NVIDIA和第三方ISV提供的GPU优化过的软件仓库，主要用于AI，HPC及虚拟化等领域。其中提供了众多容器(containers)、预训练的模型(pre-trained models)、用于Kubernetes部署的Helm charts、以及带有软件开发工具包(SDK)的行业特定AI工具包等。使用NGC可以为开发人员简化建立、定制化和GPU优化的软件的集成等过程，加速整个开发的实现过程。

前提条件

用户需要注册NGC的账号：https://ngc.nvidia.com/signin

操作方法

1. 创建一台GPU实例，操作方法请参考创建GPU实例。

   注：选择实例镜像时候，需要选择NGC-Ready的系统镜像，目前NGC支持的系统镜像包括： Ubuntu 16.04, 18.04, and 20.04 RHEL 7.5 and 7.6

百度智能云目前已经提供了支持NGC-Ready的系统镜像，如需查看百度云支持的公共系统镜像请详见这里

2. 对GPU实例安装GPU驱动，建议安装针对具体操作系统的最新版本的驱动，安装工驱动的方法，可以参考公共镜像使用步骤中提到的方式。

3. 安装Docker 和 针对NVIDIA GPU的Docker Utility Engine，即nvidia-docker

Docker的安装方法可以参考这里: Ubuntu, CentOS.

这里我们以Ubuntu为例，具体信息请参考链接。

安装的操作顺序包括：

1. 安装Docker的预备条件。

2. 添加Docker官方的GPG key。

3. 添加官方稳定的Docker仓库。

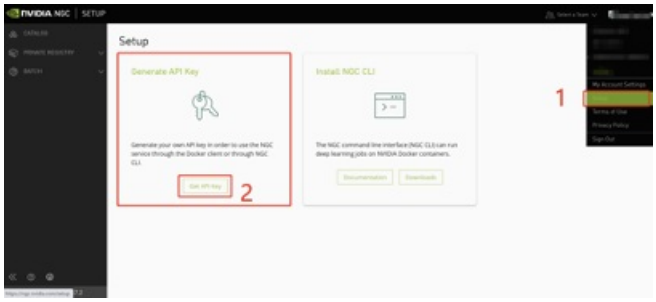
```
 $sudo apt-get install -y ca-certificates curl software-properties-common
$curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-ey add –
$sudo add-apt-repository “deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable”
```

安装nvidia-docker：

```
 curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add –
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee
/etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update
sudo apt-get install -y nvidia-docker2
sudo usermod -aG docker $USER
sudo systemctl daemon-reload
sudo systemctl reload docker
```

4. NGC的API key生成

成功注册完NGC账号之后，需要生成账户的API key，生成的过程需要登录NGC页面，点击账户名，选择Setup，会进入Setup页面，然后点击Get API Key，进入生成API Key的页面。



进入页面之后，点击Generate API Key

系统会让你确认是否生成API Key，点击Confirm确认之后，页面会变为类似于如下图所示：

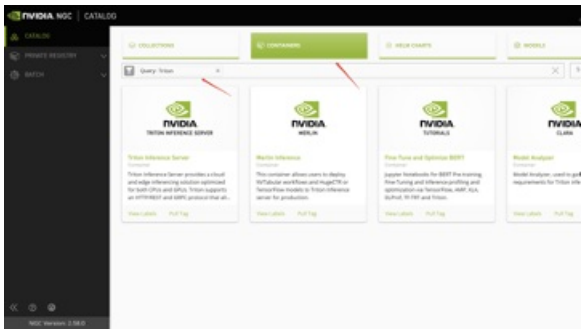
在Password处会显示一连串密码，用户返回GPU实例的shell界面按照图中的操作即可

```
 $ docker login nvcr.io
Username: $oauthtoken
Password:[输入您生成的密码]
```

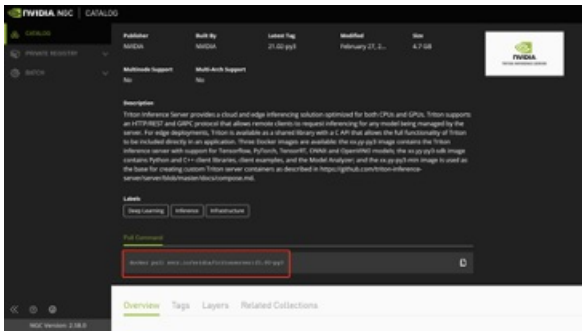
5. 使用NGC中的镜像

这里，我们以Triton为例：

• 进入NGC的CATALOG的目录部分，选择CONTAINERS分支，在Query查询中输入框架名称Triton



• 点击下面的Triton Inference Server的页面框，关于框架的介绍及拉取镜像的方法则会展示出来



• 我们按照上图中红色方框展示的命令，可以获得最新版本的容器镜像，继续在GPU实例的命令行中输入命令，即可

```
$ docker pull nvcr.io/nvidia/tritonserver:21.02-py3
```

这样，我们就可以以docker 容器的方式去使用框架或软件产品了。

使用TensorRT加速深度学习推理

背景介绍

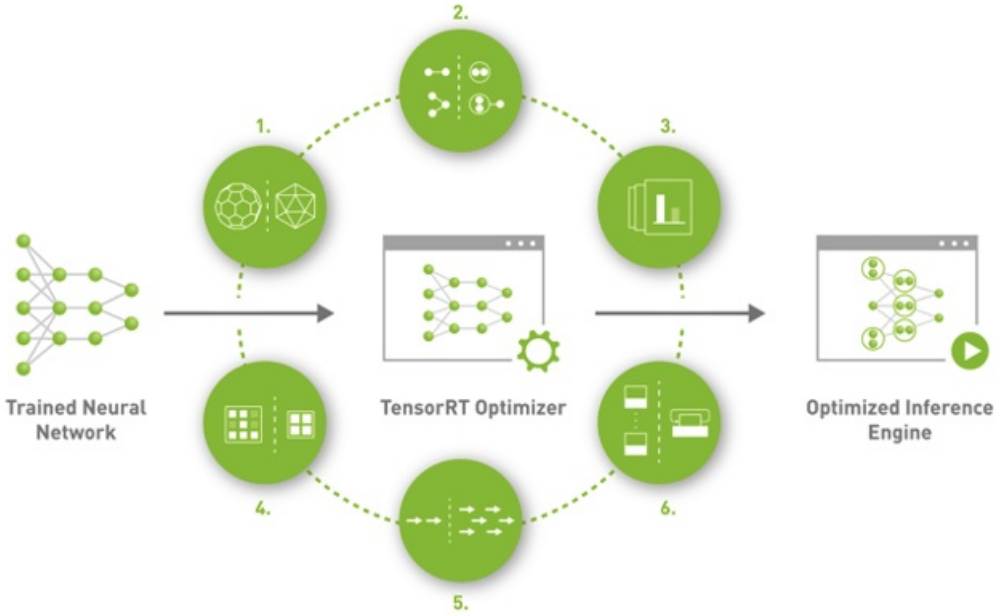
本文介绍如何在GPU云服务环境中下载、安装并使用TensorRT工具。

TensorRT，是Nvdia推出的一套专为深度学习推理打造的SDK。在推理阶段，基于TensorRT的应用可以提供同比单一CPU平台高达40倍的加速效果。TensorRT可以从所有主流的深度学习框架中导入训练好的模型并进行优化，同时提供相应的API和解析器(parser)，此外它还可以利用高精度对低精度进行校验。生成经过优化的运行时引擎可以部署到数据中心，车辆端和嵌入式环境中。

TensorRT主要包括一个深度学习推理的优化器(optimizer)和运行时(runtime)，可以为深度学习推理应用提供低延迟和高吞吐的特性。它支持C++和Pytorch两种代码方式。

TensorRT 是构建在NVIDIA的并行编程模型CUDA基础之上的，结合最新一代Ampere架构的GPU，TensorRT还可以利用Tensor Core实现稀疏性(Sparsity)加速的特点。对于深度学习推理应用的生产环境部署方面，TensorRT提供了INT8和FP16的优化，可针对视频流，语音识别，推荐领域，欺诈检测，自然语言处理等。低精度推理能够极大的降低应用的延迟，有益于实时服务，以及自动驾驶和嵌入式类型的应用。

TensorRT的详细开发者指南可参考：https://docs.nvidia.com/deeplearning/tensorrt/developer-guide



应用特点

- 支持混合精度

通过将模型量化为INT8最大化吞吐（支持PTQ和QAT两种量化方式）

- Layer和Tensor的融合

可以通过将多个节点融合为一个kernel来优化GPU显存和带宽的使用

- Kernel 的自动调节

基于目标GPU平台，选择最佳的数据层和算法

- Tensor 显存使用的动态化

最小化显存的使用，高效的反复利用显存存储tensor

- 多流执行

采用可扩展的设计可以并行的处理多个输入流

- 时间层面的融合

在时间步上利用动态生成的kernel优化了RNN（Recurrent Neural Network）

使用TensorRT的加速效果可参考：https://developer.nvidia.com/deep-learning-performance-training-inference#dl-inference

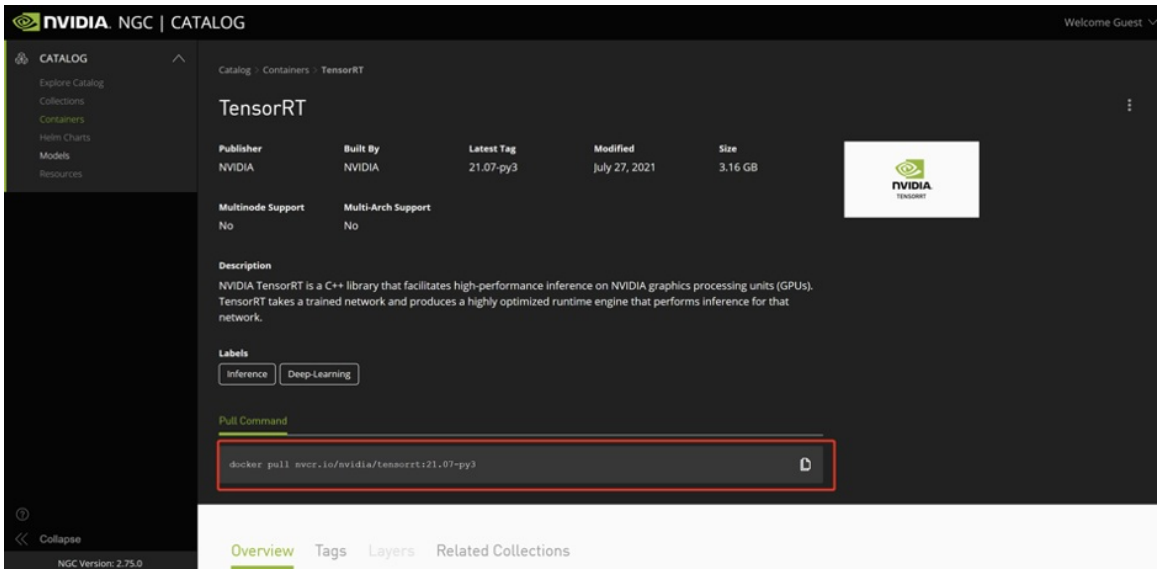
快速入门：https://developer.nvidia.com/blog/speeding-up-deep-learning-inference-using-tensorrt-updated/

操作方法

获取TensorRT

- 通过NGC获取镜像

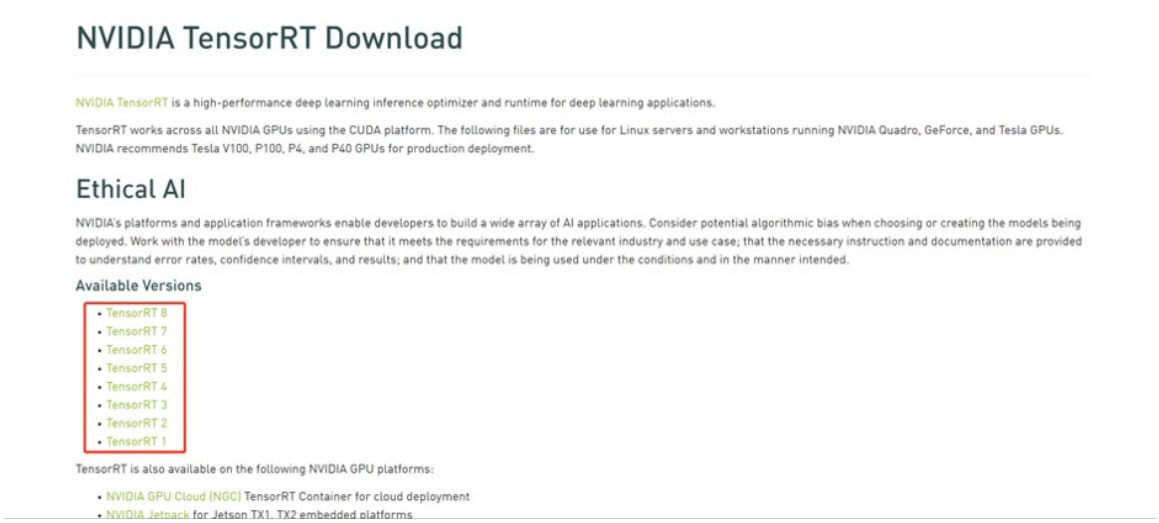
从NGC中可以获取已经容器化好的TensorRT的镜像，用户可以直接访问https://ngc.nvidia.com/catalog/containers/nvidia:tensorrt 通过docker拉取最新版本的TensorRT镜像进行使用。



如何获取NGC镜像可参考https://cloud.baidu.com/doc/GPU/s/fkppdq6rd

- 手动安装TensorRT

登录https://developer.nvidia.com/tensorrt-getting-started，选择Download Now。注册信息成功之后会进入TensorRT的下载页面，可以根据需要选择不同版本的TensorRT：



TensorRT相关的概念解释

- onnx parser：onnx解析器，它可以将一个onnx模型解析成为TensorRT可以识别的深度神经网络。此外，TensorRT还可以通过手动搭建API的方式构建整个网络(Python API, C++ API)
- Builder: 该模块会接收TensorRT中的一个哇昂罗，并生成一个针对目标平台优化好的TensorRT engine.
- Engine: Engine 会用来接收如数的数据，执行推理，并输出推理的结果。
- Logger：和Builder，Engine协同配合，在构建和推理过程中捕获errors，warning以及其它信息。

使用TensorRT的通用流程

1. 将通过原生框架(PyTorch, Paddle, TensorFlow等)预训练好的模型转成onnx格式。
2. 将onnx模型转入TensorRT
3. 应用优化器并生成一个TensorRT engine
4. 在GPU上执行推理过程

这里以MINIST数字识别模型为例，演示如何使用TensorRT：

注：如下演示的完整代码可通过NGC的镜像获取

```
docker pull nvcr.io/nvidia/tensorrt:21.09-py3
```

安装onnx python包

```
pip install onnx
```

代码关键部分如下：

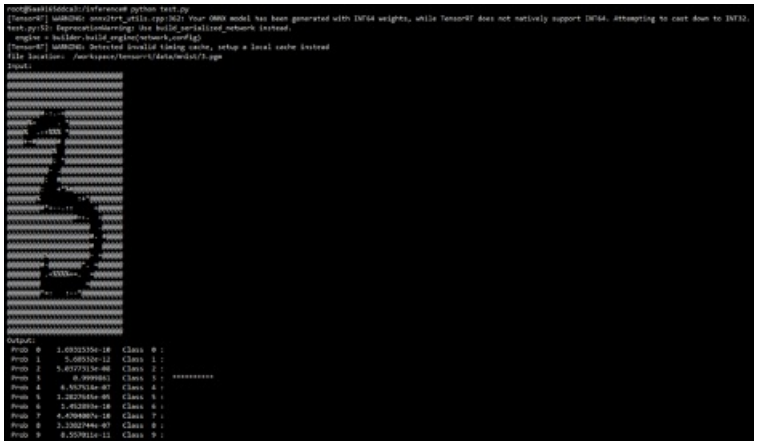
```
import tensorrt as trt
import pycuda.driver as cuda
import pycuda.autoinit
import onnx

# 构建TensorRT Logger
TRT_LOGGER = trt.Logger(trt.Logger.WARNING)

# 创建一个builder
builder = trt.Builder(TRT_LOGGER)

# 构建network
EXPLICIT_BATCH = 1 << (int)(trt.NetworkDefinitionCreationFlag.EXPLICIT_BATCH)
network = builder.create_network(EXPLICIT_BATCH)

# 构建network的配置
config = builder.create_builder_config()
#创建ONNX模型的解析器parser
parser = trt.OnnxParser(network, TRT_LOGGER)

# 将ONNX模型读入，并设定模型的执行配置
model = open("/workspace/tensorrt/data/mnist/mnist.onnx",'rb')
parser.parse(model.read())
builder.max_batch_size = 1
config.max_workspace_size = MiB(16)
config.set_flag(trt.BuilderFlag.GPU_FALLBACK)

# 构建engine
engine = builder.build_engine(network,config)
```

如上，我们就完成engine的构建，构建好的engine可以直接用来进行推理，也可以序列化后，用于线上部署等。下面我们就用构建好的engine直接进行推理服务

创建上下文进行推理 利用上面创建好的engine，我们构建上下文，并进行执行操作。这其中涉及数据从CPU读取到GPU显存，过程略，具体可见完整代码部分。

```
…
context = engine.create_execution_context()
…
# 数据传入GPU等…
…
context.execute(batch_size = 1,bindings=buffers)
# 数据传出GPU以及后续的操作等。
```

程序会随机生成一个数字，选定对应的手写数字图片，图片读入之后，TensorRT生成的engine会进行推理任务的执行，产生最后的预测结果，我们和最后的预测结果进行比对，输出结果类似如下：



其他工具

trtexec

trtexec 是一个命令行封装好的工具，主要用来在不部署应用的前提下快速使用TensorRT，主要目的是

1）用来对随机数据进行基准的神经网络性能进行测试

2）从模型生成序列化好的engine。对于前期快速定位模型问题，性能测试，生成engine，神经网络性能，选择的优化路径等具有重要的指导意义。

更多请参考：https://github.com/NVIDIA/TensorRT/tree/master/samples/trtexec

ONNX GraphSurgeon

有些情况下，我们导出的ONNX模型需要做特殊的修改，那么ONNX GraphSurgeon就是一个可以用来对当前ONNX Graph进行修改的小工具，同时还可以轻松地生成新的ONNX Graph。

更多请参考：https://github.com/NVIDIA/TensorRT/tree/master/tools/onnx-graphsurgeon

Polygraphy

Polygraphy是一个用来在多种框架下，协助运行和调试深度学习模型的工具包。它包含了Python API和命令行调用两种使用方式。

此外，Polygraphy还可以实现如下的事情：

可基于TensorRT或ONNX-Runtime后端下运行推理，并比较它们的结果 转换模型为多种格式，比如具有PTQ量化的TensorRT engine 查看多种模型类型下的结果信息 基于命令行的方式修改ONNX模型 提取子图 模型简化与剪裁 在TensorRT中隔离有问题的tactics(注：TensorRT中通过遍历不同配置的kernel进而选择最佳的kernel的过程叫做tactic选择) 更多请参考：https://github.com/NVIDIA/TensorRT/tree/master/tools/Polygraphy

使用RAPIDS加速数据科学任务

背景介绍

本文介绍如何在GPU云服务环境中下载、安装并使用RAPIDS软件库。

RAPIDS是一套开源的软件库，旨在提供给用户一整套能够完全在GPU上执行的端到端的数据科学及其分析的API调用。它面向解析和数据科学中的常规数据处理任务，囊括了多种端到端的机器学习算法。且不用耗费更多的开销就可以轻松实现加速。对多节点，多GPU的部署使其可以轻松的实现在更大尺度的数据集上完成训练和加速任务。

RAPIDS的详细介绍可参考链接

应用特点

数据预处理/ETL(Extract-Transform-Load)

- CuDF: 类似pandas的dataframe的操作库，包含GPU加速的ETL函数，可集成Dask与可扩展UCX

机器学习与图计算

- CuML：GPU原生的机器学习库，包含XGBoost，FIL，HPO等常见机器学习库
- cuGraph：GPU图解析，包括TSP，PageRank等常见图计算相关的库

可视化

- cuxfilter：GPU加速的交叉过滤
- pyViz集成：Plotly Dash，Bokenh，Datashader，HoloViews，hvPlot等

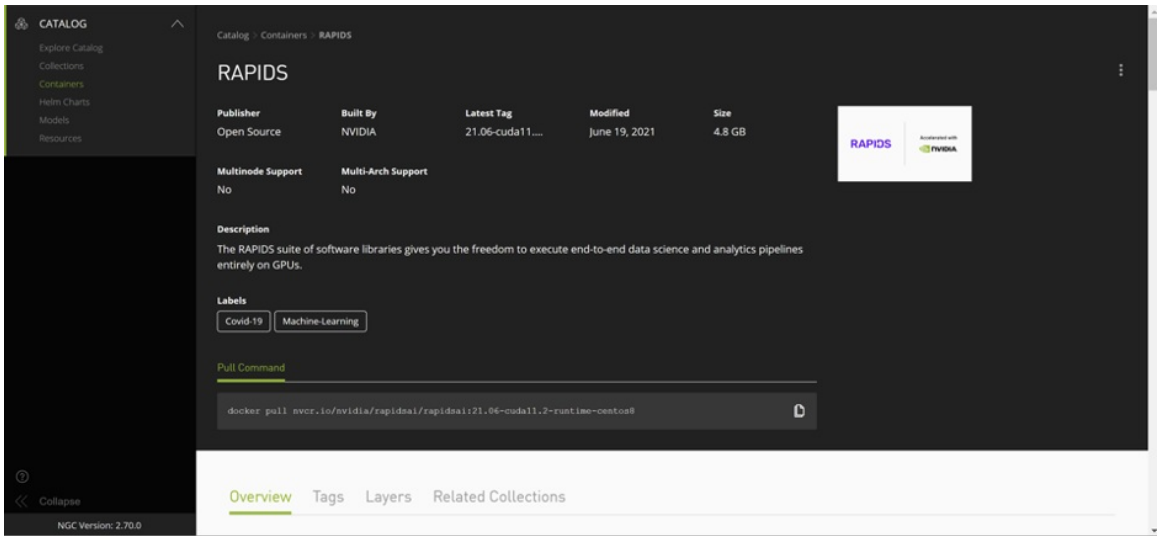
应用相关计算

与具体业务场景或特定领域强相关的一些软件库，包括：

- cuSignal：信号处理
- cuSpatial：空间解析
- CLX+Morpheus：网络日志处理+异常检测
- cuStreamz：流式化解析
- cuCIM：计算机视觉和图像处理的源语
- node-RAPIDS：用于node.js的绑定

获取RAPIDS

推荐您使用RAPIDS的container，可以通过注册NGC之后进行拉取，NGC的操作方法详见 基于GPU实例部署NGC环境 。 获取NGC环境之后，在页面中搜索RAPIDS即可获取链接



NCCL环境搭建

概览

NCCL是NVIDIA面向GPU提供的集合通信库，能实现Collective通信和点对点通信，是在大规模AI训练中必备的组件。结合云上的弹性高性能计算集群EHC GPU实例，可极大加速训练速度。本文将介绍如何准备在实例中测试NCCL所需要的必备环境。

需求场景

- 大规模人工智能训练
- 高性能计算HPC

配置步骤

环境准备

1. 安装GPU运行必备环境，包括GPU驱动等，可参考支持自动安装GPU环境的创建GPU实例或者手动安装GPU驱动。
2. 安装OFED驱动。百度智能云提供的GPU镜像已包含OFED驱动，无需手动安装。如您使用其他镜像，则可以访问OFED下载链接下载系统环境对应的OFED版本，建议您下载LTS 5.4-3.1.0版本。 如您使用CentOS 7u8或者CentOS 8u4，则需要额外下载依赖包

CentOS 7u8：

```
 yum install createrepo
yum install elfutils-libelf-devel python-devel redhat-rpm-config rpm-build libtool
```

CentOS 8u4：

```
 yum install createrepo
yum install kernel-rpm-macros python36 elfutils-libelf-devel python36-devel gdb-headless rpm-build libtool gcc
```

解压驱动安装包后，在安装目录执行以下命令。

```
 ./mlnxofedinstall --without-fw-update  --add-kernel-support --skip-distro-check
```

3. 使能GPU Direct RDMA需要加载nv_peer_mem服务，NVIDIA在驱动版本470已预装了该组件，可直接按以下步骤加载相应模块。

```
 modprobe nvidia_peermem
#### 可通过 lsmod|grep nvidia 检查
```

加载完后，可直接跳转至环境变量设置步骤。

4. 如果您使用470以下的版本，则需要手动下载并安装相应模块，下载及编译安装方法如下。

```
 git clone https://github.com/Mellanox/nv_peer_memory.git
#### 编译并安装 nv_peer_mem.ko
cd nv_peer_memory && make
cp nv_peer_mem.ko /lib/modules/$(uname -r)/kernel/drivers/video
depmod -a
modprobe nv_peer_mem
#### 可通过 lsmod|grep nv_peer_mem 检查
service nv_peer_mem start
```

5. ACSCtl 检查方法。

```
 lspci -vvv|grep ACSCtl:|grep SrvValid+
```

如果没有输出，则表示配置正常。如果有输出，则执行以下指令。

```
 for pdev in `lspci -vvv|grep -E "^[a-f]|^[0-9]|ACSCtl"|grep ACSCtl -B1|grep -E "^[a-
f]|^[0-9]"|awk '{print $1}'`
do
   setpci -s $pdev ECAP_ACS+06.w=0000
done
```

执行完后可再执行检查方法验证。

环境变量设置

环境变量	解释	设置
NCCL_SOCKET_IFNAME	指定用于通信的IP接口	设置成主机的host TCP/IP网卡，可通过ip a查找，默认是bond0
NCCL_IB_GID_INDEX	设置RDMA通信优先级	通过show_gids确认对应的IB网卡gid index
NCCL_IB_DISABLE	是否关闭IB通信	设置成1来启用TCP通信，一般需要设置成0或者默认不动
NCCL_IB_HCA	环境中的IB网卡	例如export NCCL_IB_HCA=mlx5_2,mlx5_3,mlx5_4,mlx5_5，可以通过ibstat查看IB网卡名
NCCL_DEBUG	从NCCL显示的调试信息	可设置为export NCCL_DEBUG=INFO

NCCL测试

```
 cd /home
mkdir nccl-tool && cd nccl-tool
mkdir dependency
```

1. 拉取并编译openmpi。

```
 wget https://download.open-mpi.org/release/open-mpi/v4.1/openmpi-4.1.3.tar.gz
tar -xzf openmpi-4.1.3.tar.gz && rm -rf openmpi-4.1.3.tar.gz && cd openmpi-4.1.3 ./configure --prefix=/home/nccl-
tool/dependency/openmpi
make -j && make install
```

如果上述报错/usr/bin/ld -lnuma，则采取以下指令。

```
 apt-get install numa
ln -s /usr/lib/x86_64-linux-gnu/libnuma.so.1 /usr/lib/x86_64-linux-gnu/libnuma.so
```

2. 验证openmpi。

```
 export PATH=/home/nccl-tool/dependency/openmpi/bin:$PATH
export LD_LIBRARY_PATH=/home/nccl- tool/dependency/openmpi/lib:$LD_LIBRARY_PATH
ompi_info
##### 输出一大串版本号则正常
```

3. 拉取2.12版本的nccl库。

```
 cd /home/nccl-tool
git clone -b v2.12.12-1 https://github.com/NVIDIA/nccl.git
cd nccl
make -j src.lib PREFIX=/home/nccl-tool/dependency/nccl
make install PREFIX=/home/nccl-tool/dependency/nccl
#### 通过 ls /home/nccl-tool/dependency/nccl/lib 查看libnccl.so库
```

4. 拉取nccl-tests测试脚本。

```
 cd /home/nccl-tool
git clone https://github.com/NVIDIA/nccl-tests.git
cd nccl-tests
make -j MPI=1 MPI_HOME=/home/nccl-tool/dependency/openmpi CUDA_HOME=/usr/local/cuda
NCCL_HOME=/home/nccl-tool/dependency/nccl BUILDDIR=/home/nccl-tool/bin
#### 分发工具包到其它实例上，下列的xxxx使用其他实例内网ip替换
scp -r /home/nccl-tool/bin root@${xxxx}:/home/nccl-tool
scp -r /home/nccl-tool/dependency root@${xxxx}:/home/nccl-tool
```

5. 建立实例之间的的ssh互信。

```
 ssh-keygen
ssh-copy-id -i ~/.ssh/id_rsa.pub ${ip2}
#### 新建文件hostfile用于 mpirun 识别多机，将需要互联的实例VPC IP分行写入 hostfile 文件中
vim hostfile
#### 测试mpi，输出2台实例的hostname
mpirun --prefix /home/nccl-tool/dependency/openmpi -np 2 --allow-run-as-root -- hostfile hostfile hostname
```

6. NCCL all_reduce测试。

```
 NCCL_HOME=/home/nccl-tool/dependency/nccl
CUDA_HOME=/usr/local/cuda
MPI_HOME=/home/nccl-tool/dependency/openmpi
export LD_LIBRARY_PATH=${NCCL_HOME}/lib:${CUDA_HOME}/lib64:${MPI_HOME}/lib:${LD_LIBRA
RY_PATH}
mpirun --allow-run-as-root \
--prefix /home/nccl-tool/dependency/openmpi \
-np 2 \
--hostfile ./hostfile \
-x NCCL_DEBUG=INFO \
-x NCCL_SOCKET_IFNAME=bond0 \
-x LD_LIBRARY_PATH \
-x PATH \
-x NCCL_IB_GID_INDEX=0 \
/home/nccl-tool/bin/all_reduce_perf -b 1024 -e 1G -f 2 -g 1 -t 8 -c 0 -n 20
```

得到回显结果如下。

```
 #### out-of-place                  in-place
#### size       count     type   redop    time   algbw   busbw  error     time   algbw   busbw  error
#### (B)    (elements)                  (us)  (GB/s)  (GB/s)           (us)  (GB/s)  (GB/s)
      1024         256    float     sum   44.40    0.02    0.04    N/A    43.90    0.02    0.04    N/A
      2048         512    float     sum   45.19    0.05    0.08    N/A    45.04    0.05    0.09    N/A
      4096        1024    float     sum   44.93    0.09    0.17    N/A    45.27    0.09    0.17    N/A
      8192        2048    float     sum   47.20    0.17    0.33    N/A    46.98    0.17    0.33    N/A
     16384        4096    float     sum   65.53    0.25    0.47    N/A    54.88    0.30    0.56    N/A
     32768        8192    float     sum   364.6    0.09    0.17    N/A    117.5    0.28    0.52    N/A
     65536       16384    float     sum   146.0    0.45    0.84    N/A    165.2    0.40    0.74    N/A
    131072       32768    float     sum   155.2    0.84    1.58    N/A    183.7    0.71    1.34    N/A
    262144       65536    float     sum   150.3    1.74    3.27    N/A    148.7    1.76    3.31    N/A
    524288      131072    float     sum   172.0    3.05    5.72    N/A    183.8    2.85    5.35    N/A
   1048576      262144    float     sum   204.6    5.13    9.61    N/A    203.5    5.15    9.66    N/A
   2097152      524288    float     sum   271.8    7.72   14.47    N/A    272.0    7.71   14.46    N/A
   4194304     1048576    float     sum   283.8   14.78   27.72    N/A    284.2   14.76   27.67    N/A
   8388608     2097152    float     sum   317.0   26.47   49.62    N/A    306.3   27.39   51.35    N/A
  16777216     4194304    float     sum   456.7   36.73   68.87    N/A    598.6   28.03   52.55    N/A
  33554432     8388608    float     sum   788.1   42.58   79.84    N/A    798.7   42.01   78.77    N/A
  67108864    16777216    float     sum  1232.4   54.45  102.10    N/A   1228.3   54.64  102.44    N/A
 134217728    33554432    float     sum  2270.0   59.13  110.86    N/A   2547.1   52.69   98.80    N/A
 268435456    67108864    float     sum  4609.7   58.23  109.19    N/A   4645.1   57.79  108.36    N/A
 536870912   134217728    float     sum  8093.6   66.33  124.37    N/A   7971.2   67.35  126.28    N/A
1073741824   268435456    float     sum   15227   70.51  132.21    N/A   14959   71.78  134.59    N/A
#### Out of bounds values : 0 OK
#### Avg bus bandwidth    : 39.4978
```

相关产品

GPU云服务器

基于Nvidia Clara Parabricks的基因测序加速

概览

Clara Parabricks是由NVIDIA开发，基于GPU的基因组分析软件，包含比对、预处理、突变检测、 UMI、BAM2FQ等多种功能。端到端处理流程包含DNA Germline、DNA Somatic、RNA数据处理。NVIDIA将行业标准版的CPU bwa、gatk等软件进行并行改写，即为Clara Parabricks。对比传统计算模式，在GPU使用Clara Parabricks进行基因分析能够保证一致性的结果，以及更高效的计算。

需求场景

- 基因测序。
- 药物研发。

前提条件

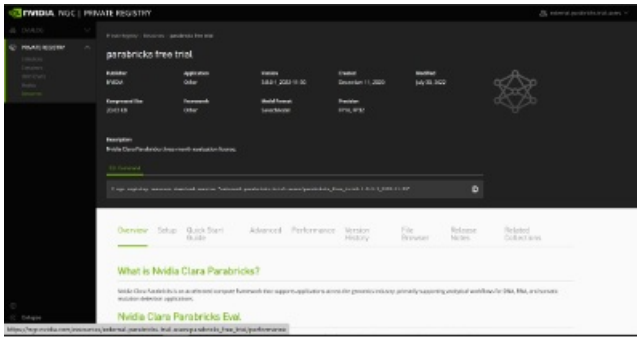
- 已在百度智能云注册账号并创建好GPU实例，推荐您使用最新的驱动版本以获取更好的效果，具体步骤可参考创建GPU实例。
- 在创建好的GPU实例中已安装好了NGC需要的环境，具体步骤可参考部署NGC环境。

配置步骤

申请Clara Parabricks试用license

- 登录申请页面，填写Clara Parabricks试用申请表，1-3个工作日后会收到一封来自Nvidia邮件，点击邮件中的accept invitation and sign in，登录NGC。

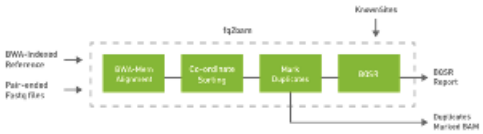
- 登录时，选择external-parabricks-trial-users作为您的组织，登录后可以看到下图的页面。在左侧目录选择Resources并下载Clara Parabricks license及安装包。



使用Clara Parabricks实现实现DNA及RNA分析流程

下面列举常用功能进行使用说明，详细使用指南请参阅Clara Parabricks User Guide。

fq2bam可实现序列比对，将测序仪产生的fq数据与标准基因组进行比对。此案例中人类标准基因组数据可从NCBI下载，测试数据使用标准样品NA12878，30X WGS，可以从这里下载。

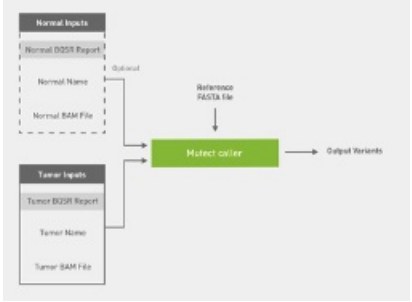



```
 pbrun fq2bam --ref Ref/Homo_sapiens_assembly38.fasta \
--in-fq Data/sample_1.fq.gz Data/sample_2.fq.gz  \
--out-bam mark_dups_gpu.bam \
--out-recal-file recal_gpu.txt \
--tmp-dir /raid/myrun
#### 可按需替换输入fq文件，指定output bam文件名及经过Base Quality校准过后的report文件名
```

Variant caller–Haplotypecaller基于比对产生的bam文件及校正报告，使用gtakHaplotypecaller模块检测生殖系突变。


```
 pbrun haplotypecaller --ref Ref/Homo_sapiens_assembly38.fasta \
--in-bam mark_dups_gpu.bam \
--in-recal-file recal_gpu.txt \
--out-variants result.vcf
#### 可按需替换mark_dups_gpu.bam 及recal_gpu.txt，并指定输出vcf文件名及路径
```

Variant caller-Mutectcaller基于比对产生的bam文件和校正报告，使用gatk Mutechcaller模块，检测体细胞突变。与生殖系突变检测不同，体细胞突变通常是由于后天因素发生。在检测时，一般需要癌组织与正常组织进行比较。



```
 pbrun mutectcaller \
--ref Ref/Homo_sapiens_assembly38.fasta \
--tumor-name tumor \
--in-tumor-bam tumor.bam \
--in-normal-bam normal.bam \
--normal-name normal \
--out-vcf output.vcf
#### 可按需替换肿瘤及正常组织的bam并输出需要指定vcf的文件名及路径。
```

RNA Pipeline比对及检测RNA fusion。

```
 pbrun rna_fq2bam \
--in-fq sample_X_1.fq.gz sample_X_2.fq.gz \
--genome-lib-dir HG38 \
--output-dir sample_X/ \
--ref ref.fasta
可按需替换测序仪产生的fq文件，并指定输出文件目录，此案例中指定名字为“sample_x.out.junction”的文件作为下一步的输入。

pbrun starfusion \
--chimeric-junction sample_x.out.junction \
 --genome-lib-dir HG38 \
 --output-dir sample_X/
```

相关产品

GPU云服务器

使用Nsight工具分析优化应用程序

概览

Nsight是NVIDIA面相开发者提供的开发工具套件，能提供深入的跟踪、调试、评测和分析，以优化跨 NVIDIA GPU和CPU的复杂计算应用程序。Nsight主要包含Nsight System、Nsight Compute、Nsight Graphics三部分。

- Nsight System

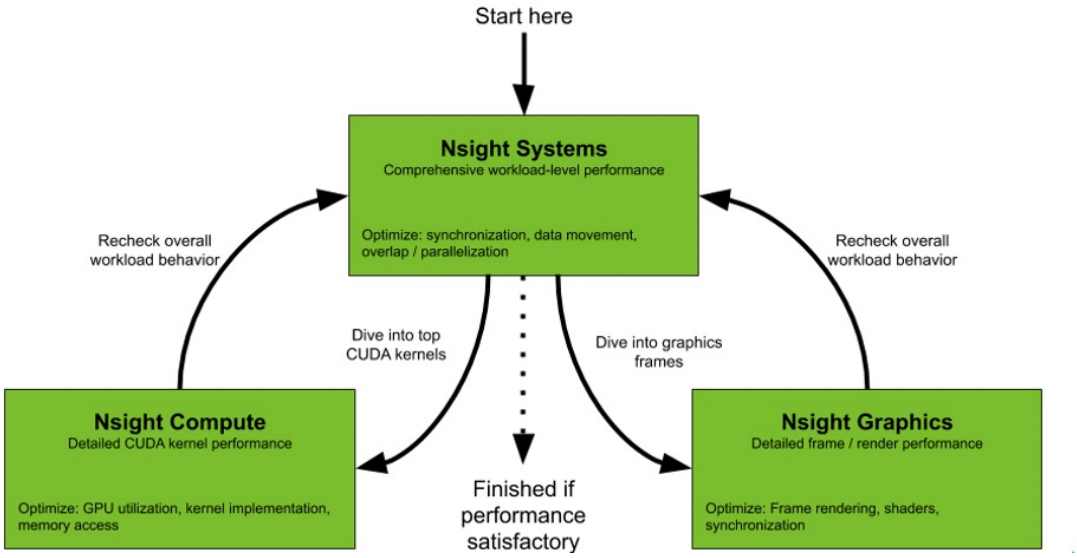
所有与NVIDIA GPU相关的程序开发都可以从Nsight System开始以确定最大的优化机会。Nsight System给开发者一个系统级别的应用程序性能的可视化分析。开发人员可以优化瓶颈，以便在任意数量或大小的CPU和GPU之间实现高效扩展。详情可访问NVIDIA官网。

- Nsight Compute

Nsight Compute是一个CUDA应用程序的交互式kernel分析器。它通过用户接口和命令行工具的形式提供了详细的性能分析度量和API调试。Nsight Compute还提供了定制化的和数据驱动的用户接口和度量集合，可以使用分析脚本对这些界面和度量集合进行扩展，以获得后处理的结果。详情可访问NVIDIA官网。

- Nsight Graphics

Nsight Graphics是一个用于调试、评测和分析Microsoft Windows和Linux上的图形应用程序。它允许您优化基于Direct3D 11，Direct3D 12， DirectX，Raytracing 1.1， OpenGL，Vulkan和KHR Vulkan Ray Tracing Extension的应程序的性能。详情可访问NVIDIA官网。



需求场景

- 人工智能训练性能分析
- 图形渲染的性能分析

配置步骤

环境准备

此次典型实践通过在云端进行人工智能训练，并将运算结果导出并发送给本地进行离线Nsight System分析，并快速优化训练性能。

- 本地：可以连接到Internet的PC或者笔记本电脑
- 云端：创建并挂载EIP的GPU云服务器

创建GPU云服务器时通过自定义驱动或者使用GPU镜像安装后即可自动安装Nisght System，如需手动安装可访问Nsight System的下载地址，根据本地和云端的运行环境分别下载对应的版本。

Nsight System使用示例

以手写数字数据库MNIST作为训练数据集，使用PyTorch框架进行神经网络训练。通过Nsight System对训练过程进行性能分析，进而找到性能瓶颈，指导优化训练过程。

1、下载训练所需的数据集和脚本 数据集采用MNIST，训练脚本我们采用该位置的PyTorch代码，基于单块NVIDIA Volta GPU我们将完成多batches和epochs的训练。一次完整的迭代过程包括如下几个方面：

- 从磁盘载入数据
- 数据加载到GPU设备端
- 前向传播
- 反向传播

运行以下命令，可完成一次训练，本次实践以bcc.gn3.c10m80.1v100-32g实例为例，约花费90s左右。

```
python main.py
```

2、利用Nsight system进行性能分析

（可选）开发者可通过添加NVIDIA Tools Extension(NVTX)；来对应用程序逻辑上的时间线进行注释，帮助分析人员理解应用算法的上下文逻辑，在main.py中的训练代码部分增加nvtx函数如下：

```
def train(args, model, device, train_loader, optimizer, epoch):
    model.train()
    nvtx.range_push("Data loading");
    for batch_idx, (data, target) in enumerate(train_loader):
        nvtx.range_pop();
        nvtx.range_push("Batch " + str(batch_idx))

        nvtx.range_push("Copy to device")
        data, target = data.to(device), target.to(device)
        nvtx.range_pop()

        nvtx.range_push("Forward pass")
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        nvtx.range_pop()

        nvtx.range_push("Backward pass")
        loss.backward()
        optimizer.step()
        nvtx.range_pop()

        nvtx.range_pop()

        if batch_idx % args.log_interval == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))
        nvtx.range_push("Data loading");
    nvtx.range_pop();
```

Data Loading

Copy to Device

Forward Pass

Backward Pass

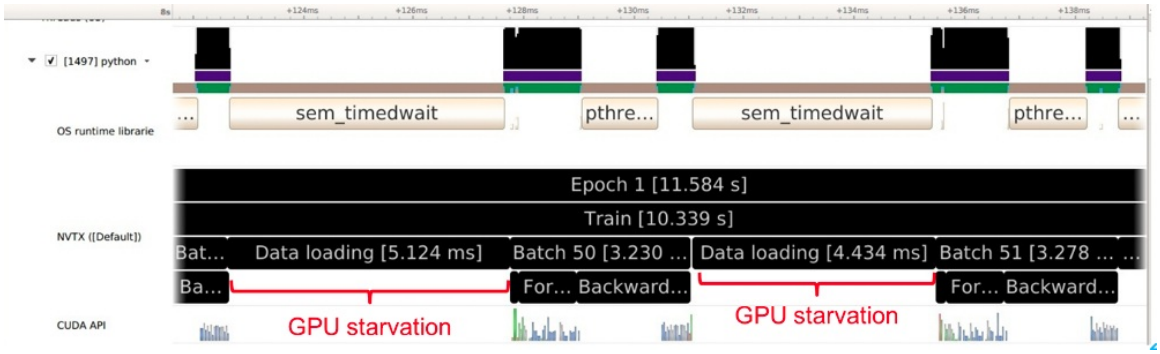
通过Nsight System的命令行方式，生成分析结果文件：

```
nsys profile –t cuda,osrt,nvtx –o baseline –w true python main.py
```

在这个例子中，采用的参数解释如下：

- -t 后面跟定的参数是我们要追踪的API，即需要CUDA API，OS runtime API以及NVTX API
- -o 给定的是输出的文件名称
- -w 后面表明是或否要在命令行中同时输出结果
- python main.py为程序的执行命令

将导出的baseline输出文件下载到本地，并拖拽到本地的Nsight System窗口即可获取性能结果展示，示例如下：



可以看到，通过产生的分析文件，我们发现训练的过程中，有很大一段时间GPU都处于空闲状态。

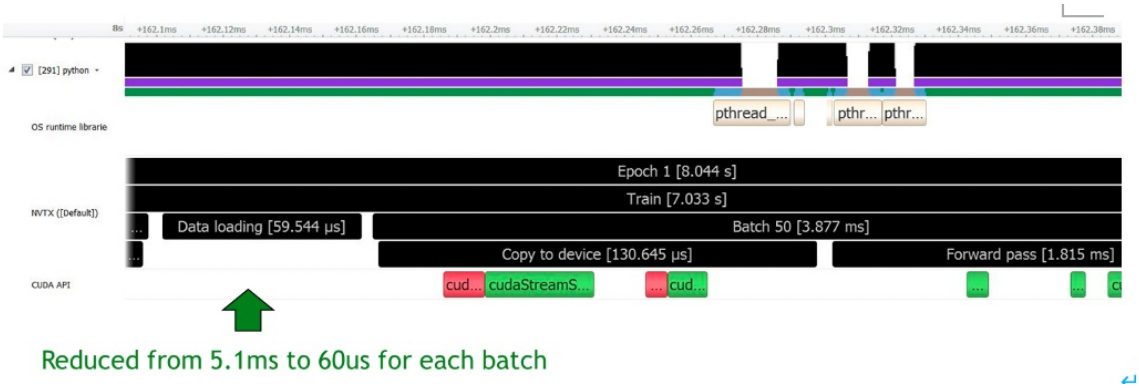
3、性能优化 可以发现，GPU空闲状态主要是因为CPU在处理数据加载耗时过多。我们可以对收据加载进行优化，优化的方式如下： 原有的数据加载采用的是一个CPU worker 线程：

```
kwargs = {'num_workers': 1, 'pin_memory': True} if use_cuda else {}
```

我们可以将num_workers的数量调大，这里我们调整为8

```
kwargs = {'num_workers': 8, 'pin_memory': True} if use_cuda else {}
```

重新按照步骤2执行，可观察到如下结果：



可以发现，经过优化之后，数据加载的时间明显缩短了。整体的训练时间也从原来的90s变成了21s，获得了显著的加速效果。

相关产品（必选）

GPU云服务器、弹性公网IP

基于GPU云服务器部署NIM

概览

NVIDIA NIM 是NVIDIA AI Enterprise的一部分，为用户提供了基于GPU加速的推理微服务容器。容器中含有预训练过的、定制化的AI模型，通过简单的命令即可完成云服务器部署。NIM 微服务对外开放了工业级标准的API，可与AI应用、开发框架和一些工作流程进行集成。容器内部的模型构建基于NVIDIA和社区的预优化的推理引擎，如NVIDIA TensorRT和TensorRT-LLM产生的推理引擎等。同时，NIM微服务针对每个基础模型的组合和运行时检测到的GPU系统，自动优化响应延迟和吞吐。容器还提供了标准的可观测数据反馈，并内置了对Kubernetes在GPU上的自动缩放的支持。

使用步骤

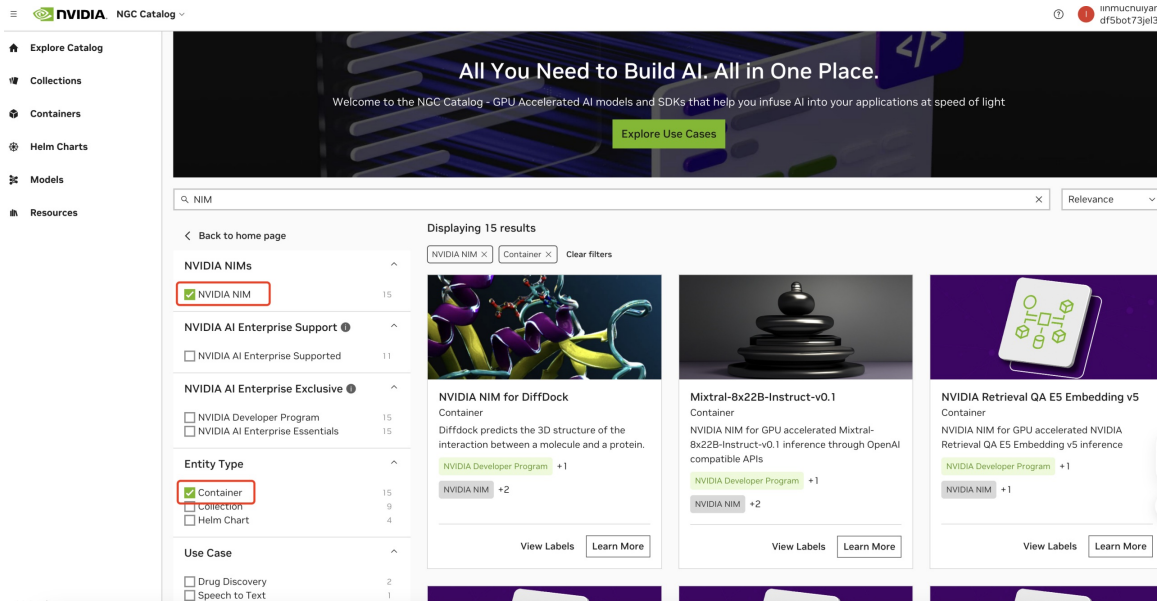
前提条件

1.用户需要提前注册NGC账号，同时需要生成并保存NGC的API key，具体可以参考基于GPU实例部署NGC环境。

2.已创建配置Ampere及以上架构GPU，例如A10、A800的云服务器实例。

基于GPU实例部署NIM容器

在NGC搜索界面中搜索NIM并勾选NVIDIA NIM与Container，可以看到目前NGC上提供的NIM 容器镜像。



此实践以Meta/Llama3-8b-instruct的1.0版本容器镜像为例。

进入容器镜像页面并复制容器镜像地址（下图中红色框选处的位置）。

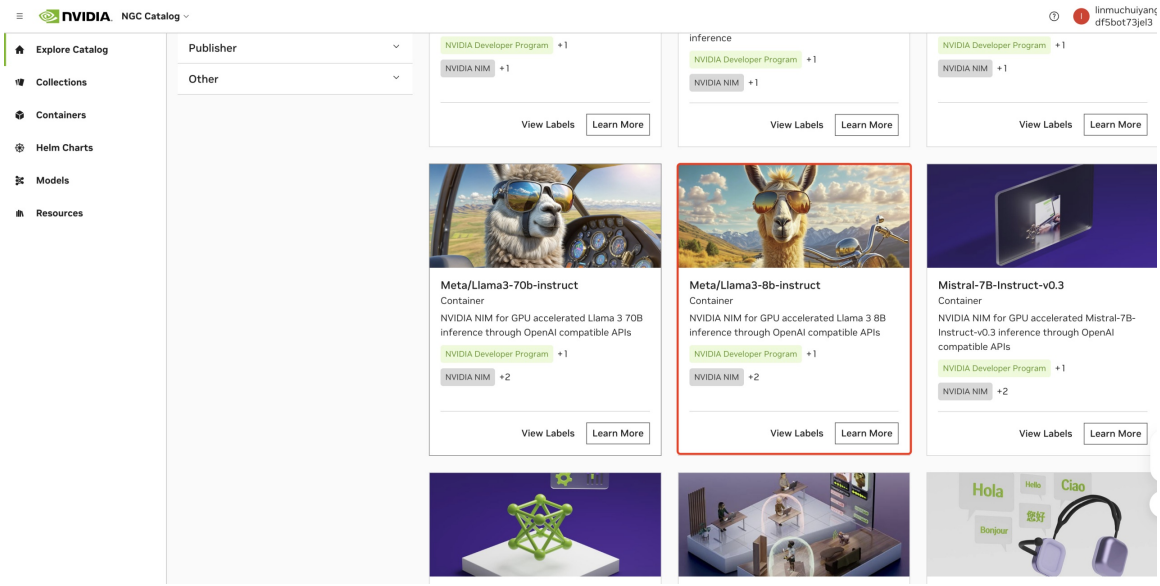


图2：Meta/Llama3-8b-instruct 在NGC上提供的容器镜像及其版本

登陆GPU云服务器并参考基于GPU实例部署NGC环境拉取nvcr.io/nim/meta/llama3-8b-instruct:1.0.0的容器镜像。

通过如下命令启动容器镜像服务

```
 $export NGC_API_KEY=您的NGC API Key
$nvidia-docker run -it --rm --name=LLaMA3-70B --shm-size=16GB -e NGC_API_KEY -
v${PWD}/.cache/nim:/opt/nim/.cache -u 0 -p 8000:8000 nvcr.io/nim/meta/llama3-70b-instruct:1.0.0
```

容器启动成功之后，预计输出如下：

```
 ===========================================
== NVIDIA Inference Microservice LLM NIM == ============================================
NVIDIA Inference Microservice LLM NIM Version 1.0.0
Model: nim/meta/llama3-8b-instruct
Container image Copyright (c) 2016-2024, NVIDIA CORPORATION & AFFILIATES. All rights reserved.
This NIM container is governed by the NVIDIA AI Product Agreement here: https://www.nvidia.com/en-us/data-center/products/nvidia-ai-enterprise/eula/.
A copy of this license can be found under /opt/nim/LICENSE.
The use of this model is governed by the AI Foundation Models Community License here: https://docs.nvidia.com/ai-foundation-models-community-license.pdf.
ADDITIONAL INFORMATION: Meta Llama 3 Community License, Built with Meta Llama 3.
A copy of the Llama 3 license can be found under /opt/nim/MODEL_LICENSE.
2024-06-04 09:27:50,266 [INFO] PyTorch version 2.2.2 available.
2024-06-04 09:27:50,814 [WARNING] [TRT-LLM] [W] Logger level already set from environment. Discard new verbosity: error
2024-06-04 09:27:50,814 [INFO] [TRT-LLM] [I] Starting TensorRT-LLM init.
2024-06-04 09:27:51,031 [INFO] [TRT-LLM] [I] TensorRT-LLM inited. [TensorRT-LLM] TensorRT-LLM version: 0.10.1.dev2024053000
INFO 06-04 09:27:51.718 api_server.py:489] NIM LLM API version 1.0.0 INFO 06-04 09:27:51.719 ngc_profile.py:217] Running NIM without LoRA. Only looking for compatible profiles that do not support LoRA.
INFO 06-04 09:27:51.719 ngc_profile.py:219] Detected 3 compatible profile(s).
...
INFO 06-04 09:28:07.393 api_server.py:456] Serving endpoints: 0.0.0.0:8000/openapi.json
0.0.0.0:8000/docs
0.0.0.0:8000/docs/oauth2-redirect
0.0.0.0:8000/metrics
0.0.0.0:8000/v1/health/ready
0.0.0.0:8000/v1/health/live
0.0.0.0:8000/v1/models
0.0.0.0:8000/v1/version
0.0.0.0:8000/v1/chat/completions
0.0.0.0:8000/v1/completions
```

```
INFO 06-04 09:28:07.393 api_server.py:460] An example cURL request: curl -X 'POST' \
'http://0.0.0.0:8000/v1/chat/completions' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"model": "meta/llama3-8b-instruct",
"messages": [
{
"role":"user",
"content":"Hello! How are you?"
},
{
"role":"assistant",
"content":"Hi! I am quite well, how can I help you today?"
},
{
"role":"user",
"content":"Can you write me a song?"
}
],
"top_p": 1,
"n": 1,
"max_tokens": 15,
"stream": true,
"frequency_penalty": 1.0,
"stop": ["hello"]
}'
INFO 06-04 09:28:07.433 server.py:82] Started server process [32]
INFO 06-04 09:28:07.433 on.py:48] Waiting for application startup.
INFO 06-04 09:28:07.438 on.py:62] Application startup complete.
INFO 06-04 09:28:07.438 server.py:214] Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

访问服务

使用curl命令验证推理服务，这里举了一个文本续写的例子。

```
$curl -s http://0.0.0.0:8000/v1/completions -H 'accept: application/json' -H 'Content-Type: application/json' -d '{"model":"meta/llama3-8b-instruct", "prompt": "Once upon a time", "max_tokens":64}' | jq
```

预期输出结果如下：


```
{
  "id": "cmpl-01e9b74b4eb746b6923b6ae1c4f2c561",
  "object": "text_completion",
  "created": [redacted],
  "model": "meta/llama3-8b-instruct",
  "choices": [
    {
      "index": 0,
      "text": ", there was a young man named Jack who lived in a small village at the foot of a towering mountain. Jack was a curious and adventurous soul, always eager to explore and discover new things. One day, he heard the tale of a magical spring hidden deep within the mountain, said to possess the power to grant wishes",
      "logprobs": null,
      "finish_reason": "length",
      "stop_reason": null
    }
  ],
  "usage": {
    "prompt_tokens": 5,
    "total_tokens": 69,
    "completion_tokens": 64
  }
}
```

部署满血版DeepSeek-R1模型SGlangServer（单机&多机部署&参数建议）

概览

本篇介绍如何在GPU云服务器部署671B参数的DeepSeek R1模型推理服务，部署方式为在GPU云服务器下载SGlang容器环境并在容器中构建SGlang Server，可通过单机GPU实例部署以及两机GPU实例部署。同时分享SGlang Server的部署参数建议，您可根据业务需求按需选择。

单机部署SGlang AIAK版服务

通过百度智能云优化版SGlang AIAK镜像可获取更高的性能，使用方式如下：

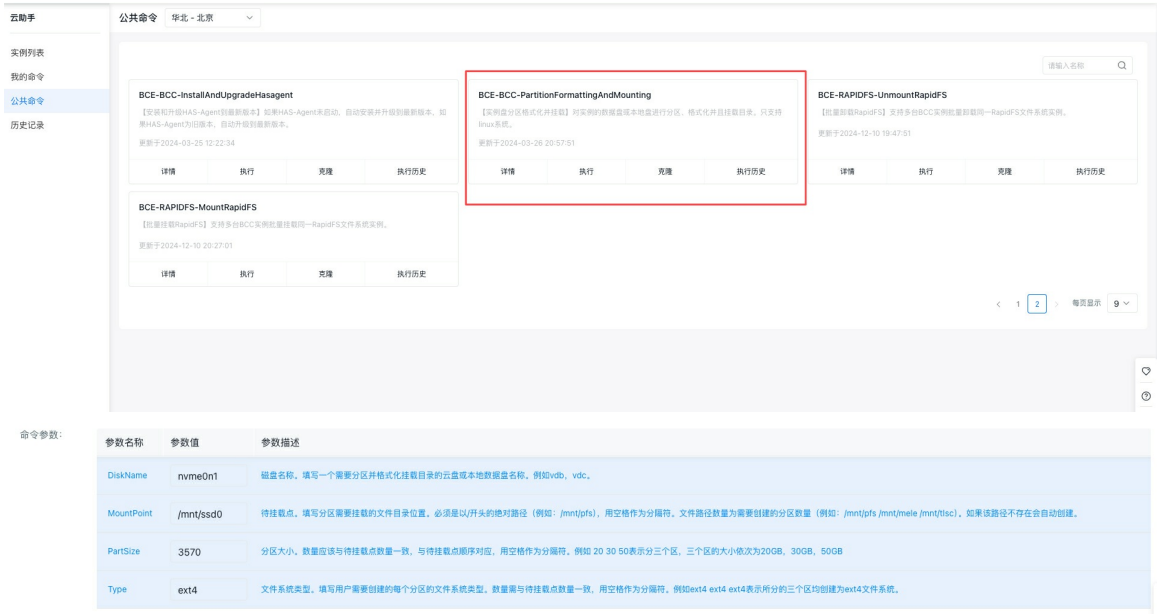
创建实例并配置环境

1、开通GPU云服务器，GPU建议选择显存容量141G及以上的型号，建议选择535及以上驱动版本：

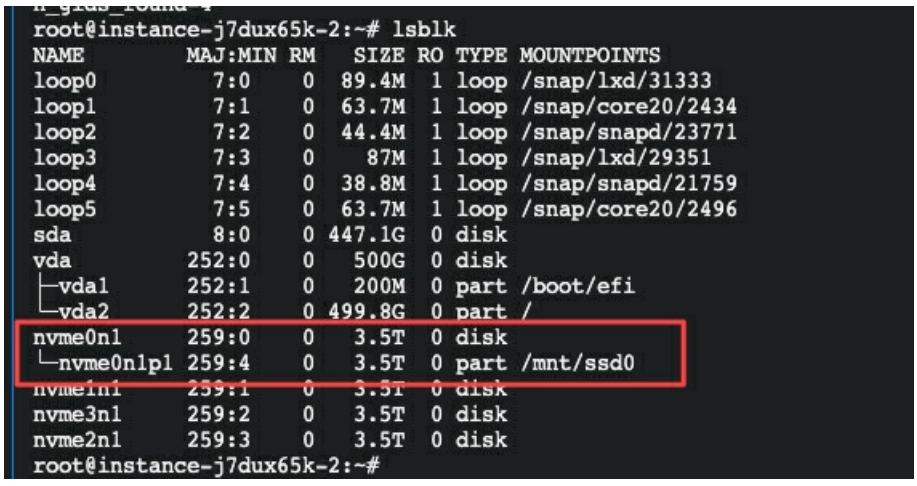
2、安装docker环境，您可通过云助手的公共命令安装docker或者登陆实例后执行以下脚本安装：

```
curl -sL http://mirrors.baidubce.com/nvidia-binary-driver/scripts/gpu_docker.sh | bash -
```

3、（可选）初始化数据盘，因模型容量较大，建议将模型保存在数据盘中，如您购买的实例包含本地盘，可通过云助手公共命令实现本地盘初始化及挂载，示例参数如下：



执行后可查看挂载结果:



下载模型及相关数据

建议您下载BOS对象存储CLI工具并从云内网下载模型。

1、安装BOS CLI工具：

下载BOS CLI：https://cloud.baidu.com/doc/BOS/s/Ejwvyqobd 到实例，并执行以下命令

```
#### 解压并添加链接
unzip linux-bcecmd-0.5.1.zip #BOS CLI工具定期更新 替换为最新版的名称
ln -s `pwd`/linux-bcecmd-0.5.1/bcecmd /usr/bin/ #BOS CLI工具定期更新 替换为最新版的名称
```

2、下载模型、镜像及测试数据集，文件保存在/mnt/ssd0中，可按需替换为其他目录：

目前aiak-sglang白名单开放，有需求请提交工单

```
* 配置AK/SK，配置的AKSK需要有bosfullcontrolaccess IAM权限，其余参数可通过回车跳过
bcecmd -c
* 下载aiak-sglang镜像
bcecmd bos cp bos://baidu-ai-cloud-iaas-bcc-bj/ai/private_docker/aiak-sglang-v0.4.4-v1.1.tar /mnt/ssd0/images/
* DeepSeek - R1 模型
bcecmd bos sync bos://baidu-ai-cloud-iaas-bcc-bj/ai/model/DeepSeek-R1 /mnt/ssd0/model/DeepSeek-R1
* MTP使能依赖的小模型
bcecmd bos sync bos://baidu-ai-cloud-iaas-bcc-bj/ai/model/DeepSeek-R1-NextN /mnt/ssd0/model/DeepSeek-R1-NextN
* 测试数据集：
bcecmd bos cp bos://baidu-ai-cloud-iaas-bcc-bj/ai/dataset/ShareGPT_V3_unfiltered_cleaned_split.json /mnt/ssd0/dataset/ShareGPT_V3_unfiltered_cleaned_split.json
```

启动Server服务

1、加载容器镜像：

```
docker load -i /mnt/ssd0/images/aiak-sglang-v0.4.4-v1.1.tar
```

2、启动docker： 如您创建的GPU云服务器是裸金属形态，可执行以下命令：

```
docker run --gpus all -itd --name aiak-sglang --shm-size=32g --privileged --user=root -v /mnt/ssd0/:/mnt/ssd0/ aiak-sglang-v0.4.4:v1.1 /bin/bash
```

如您创建的GPU云服务器是虚拟机形态，则需要将实例中的NCCL拓扑文件挂载到容器内，启动命令更新如下“

```
docker run --gpus all -itd --name aiak-sglang --shm-size=32g --privileged --user=root -v /mnt/ssd0/:/mnt/ssd0/ -v /var/run/nvidia-topologyd/virtualTopology.xml:/var/run/nvidia-topologyd/virtualTopology.xml aiak-sglang-v0.4.4:v1.1 /bin/bash
```

3、登陆docker：

```
docker exec -it aiak-sglang bash
```

4、通过如下脚本启动SGlang server，因开启pytorch compile, 启动耗时较长，需要等待半小时左右，如您的环境需要更快的启动速度，可关闭此参数：

```
  export USE_CUDA_ROPE=0
export USE_FUSED_INPUT_TO_FP8=1
export ENABLE_SELECT_EXPERTS=1
export SGL_ENABLE_JIT_DEEPGEMM=1
R1_MODEL_PATH=/mnt/ssd0/model/DeepSeek-R1 #更改为实际路径
NextN_MODEL_PATH=/mnt/ssd0/model/DeepSeek-R1-NextN #更改为实际路径
TP=8
PORT=8000
max_running_requests=32 #s控制server支持的最大并发数，数字越大编译所需时间越久

python3 -m sglang.launch_server --model-path $R1_MODEL_PATH --tp $TP --trust-remote-code --port $PORT --host 0.0.0.0 --mem-fraction-static 0.90 --max-running-requests $max_running_requests --enable-flashinfer-mla --enable-torch-compile --enable-flashinfer-mla --speculative-algorithm NEXTN --speculative-draft $NextN_MODEL_PATH --speculative-num-steps 2 --speculative-eagle-topk 1 --speculative-num-draft-tokens 2
```

5、Server显示 The server is fired up and ready to roll!，表示 server 已达到运行中状态，可开启一个新的终端并重复上述步骤3登陆容器内做性能测试，建议通过以下命令和脚本执行性能测试，

```
  bash test_benchmark.sh > test_benchmark.log 2>&1 &
```

如下脚本会依次测试1、4、8、16并发下的性能，如需测试更高并发，请更改max_concurrency。 test_benchmark.sh：

```
#### !/bin/bash

set -e

IC_IP=$(hostname -i)
PORT=8000

DATASET_PATH='/mnt/ssd0/dataset/ShareGPT_V3_unfiltered_cleaned_split.json'
MODEL_PATH='/mnt/ssd0/model/'
MODEL='DeepSeek-R1/'
LOG_PATH='logs/'

function run_bench() {
    for i in "${max_concurrency[@]}"; do
        num_prompts+=($((i * $MULTIPLE)))
    done

    # warm up
    aiakperf -if $INPUT_LEN -of $OUTPUT_LEN -m ${MODEL_PATH}${MODEL} -r openai -d raw_sharegpt -D
${DATASET_PATH} -a $IC_IP:8000 -n 16 -w 16 -M ds-test-model -igr False -o ${LOG_PATH}
    sleep 5

    for i in "${!max_concurrency[@]}"; do
        CONCURRENCY=${max_concurrency[$i]}
        PROMPTS=${num_prompts[$i]}

        echo "Running benchmark with concurrency $CONCURRENCY, prompts $PROMPTS"

        aiakperf -if $INPUT_LEN -of $OUTPUT_LEN -m ${MODEL_PATH}${MODEL} -r openai -d raw_sharegpt -D
${DATASET_PATH} -a $IC_IP:8000 -n $PROMPTS -w $CONCURRENCY -M ds-test-model  -igr False -o ${LOG_PATH} >
${LOG_PATH}${INPUT_LEN}_${OUTPUT_LEN}_${CONCURRENCY}_`date +%s`
        sleep 5

    done
}

#### 3.5kinput / 1.5koutput
INPUT_LEN=3500
OUTPUT_LEN=1500
MULTIPLE=10
declare -a max_concurrency=(1 4 8 16)
declare -a num_prompts=()
run_bench
```

单机部署SGlang 社区版服务

创建实例并配置环境

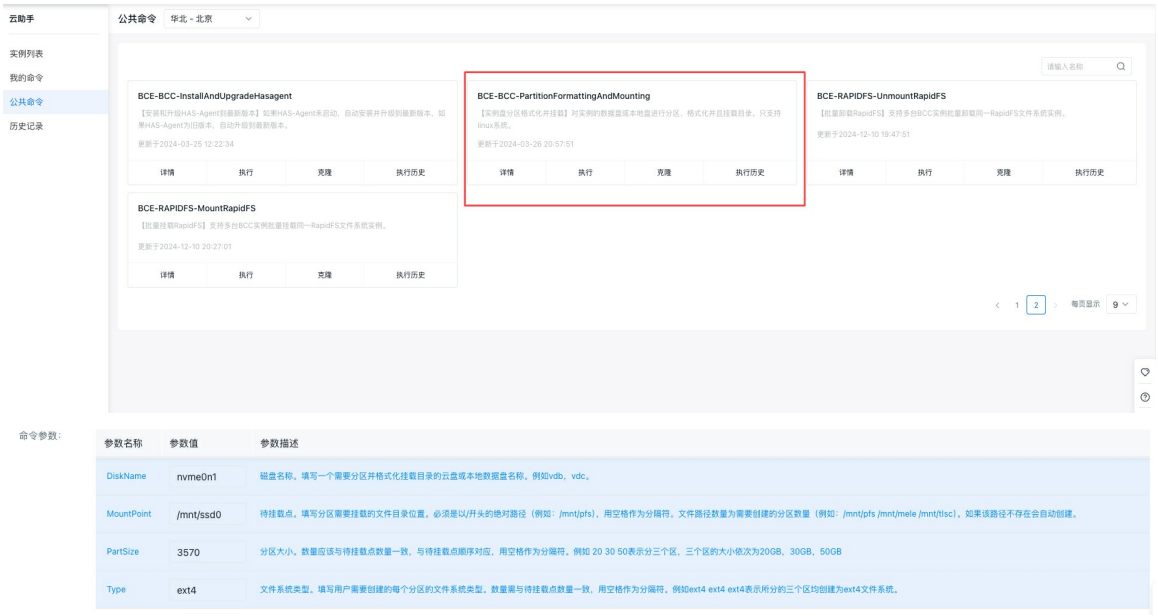
1、开通GPU云服务器，GPU建议选择显存容量96G及以上的型号，建议选择如下的GPU环境：

2、安装docker环境，您可通过云助手的公共命令安装docker或者登陆实例后执行以下脚本安装：

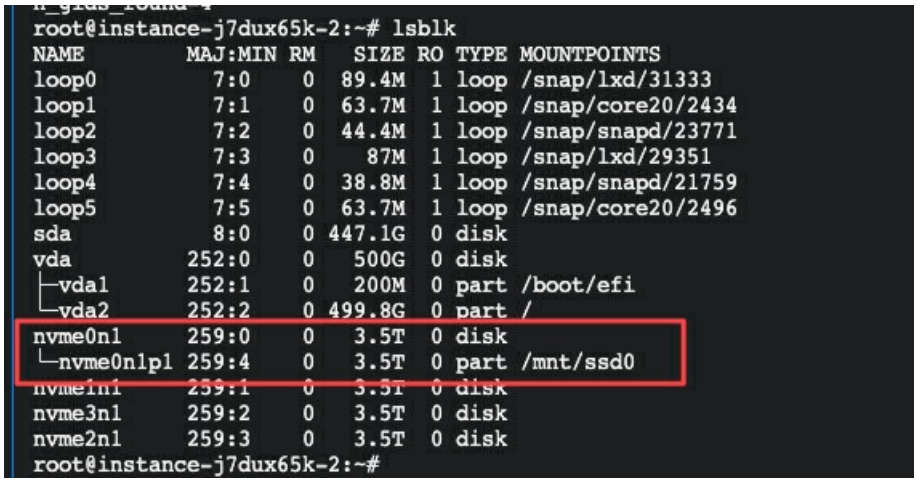
```
curl -sL http://mirrors.baidubce.com/nvidia-binary-driver/scripts/gpu_docker.sh | bash -
```

3、（可选）初始化数据盘，因模型容量较大，建议将模型保存在数据盘中，如您购买的实例包含本地盘，可通过云助手公共

命令实现本地盘初始化及挂载，示例参数如下：



执行后可查看挂载结果:



下载模型及相关数据

建议您下载BOS对象存储CLI工具并从云内网下载模型。

1、安装BOS CLI工具：

```
#### 下载bcecmd程序
wget http://mirrors.baidubce.com/nvidia-binary-driver/scripts/linux-bcecmd-0.3.9.zip
#### 解压
unzip linux-bcecmd-0.3.9.zip
ln -s `pwd`/linux-bcecmd-0.3.9/bcecmd /usr/bin/
```

2、下载模型、镜像及测试数据集，文件保存在/mnt/ssd0中，可按需替换为其他目录：

```
* 下载sglang镜像
bcecmd bos cp bos://baidu-ai-cloud-iaas-bcc-bj/ai/docker/sglang_v0.4.3.post2-cu124.tar
/mnt/ssd0/images/sglang_v0.4.3.post2-cu124.tar
* DeepSeek - R1 模型
bcecmd bos sync bos://baidu-ai-cloud-iaas-bcc-bj/ai/model/DeepSeek-R1 /mnt/ssd0/model/DeepSeek-R1
* 测试数据集：
bcecmd bos cp bos://baidu-ai-cloud-iaas-bcc-bj/ai/dataset/ShareGPT_V3_unfiltered_cleaned_split.json
/mnt/ssd0/dataset/ShareGPT_V3_unfiltered_cleaned_split.json
```

启动Server服务

1、加载容器镜像：

```
docker load -i /mnt/ssd0/images/sglang_v0.4.3.post2-cu124.tar
```

2、启动docker： 如您创建的GPU云服务器是裸金属形态，可执行以下命令：

```
docker run --gpus all -itd --name sglang_deepseek --shm-size=32g --privileged --user=root --network=host -v /mnt/ssd0/:/workspace/ lmsysorg/sglang:v0.4.3.post2-cu124 /bin/bash
```

如您创建的GPU云服务器是虚拟机形态，则需要将实例中的NCCL拓扑文件挂载到容器内，启动命令更新如下“

```
docker run --gpus all -itd --name sglang_deepseek --shm-size=32g --privileged --user=root --network=host -v /mnt/ssd0/:/workspace/ -v /var/run/nvidia-topologyd/virtualTopology.xml:/var/run/nvidia-topologyd/virtualTopology.xml lmsysorg/sglang:v0.4.3.post2-cu124 /bin/bash
```

3、登陆docker：

```
docker exec -it sglang_deepseek bash
```

4、启动Sglang server：

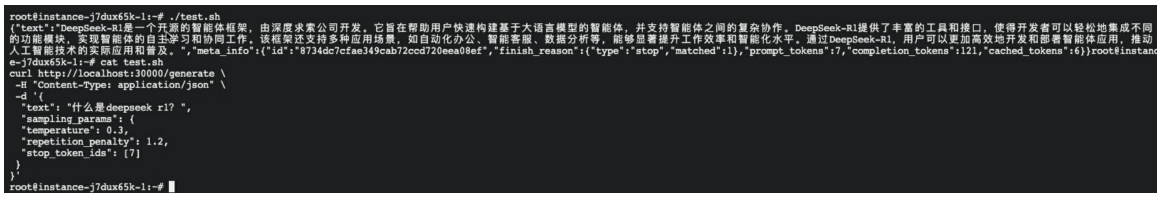
```
python3 -m sglang.launch_server --model-path /workspace/model/DeepSeek-R1/ --port 30000 --mem-fraction-static 0.9 --tp 8 --trust-remote-code
```

SGlang的启动参数建议：

参数名称	参数功能	使用建议
--mem-fraction-static	控制 sglang 初始化静态显存占用量系数，包含模型及kvcache占用，默认值0.8	建议调高--mem-fraction-static 保证有足够的显存支持kvcache或者更高性能，如遇到运行时cuda graph 初始化和 pytorch 运行时OOM报错，则调低该参数确保能正常运行
--enable-dp-attention	使能attention数据并行	dp-attention 能够节省kvcache 1. 注重并发场景建议开启 dp attention 来提升吞吐 token throughput 。2. 注重延时场景建议关闭 dp atttention 来降低首 token 延时 TTFT。 当前仅建议单机部署开启该参数。

5、测试服务，server启动后，可在该GPU实例打开新的终端，通过如下命令测试：

```
curl http://localhost:30000/generate \
-H "Content-Type: application/json" \
-d '{
 "text": "什么是deepseek r1？",
 "sampling_params": {
 "temperature": 0.3,
 "repetition_penalty": 1.2,
 "stop_token_ids": [7]
 }
}'
```



双机部署社区版SGlang服务

如您的业务需要Server支持更高的并发量，可通过双机部署Server提高性能，本实践介绍TP=16的双机部署方案。

1、创建2台同规格的GPU云服务器，并按照如上单机部署服务直男，完成启动server前的所有操作。

2、登陆2台实例并加载docker镜像：

```
docker load -i /mnt/ssd0/images/sglang_v0.4.3.post2-cu124.tar
```

3、在2台实例启动docker：

```
docker run --gpus all -itd --name sglang_deepseek --shm-size=32g --privileged --user=root --network=host -v
/mnt/ssd0/:/workspace/ lmsysorg/sglang:v0.4.3.post2-cu124 /bin/bash
```

3、登陆2台实例的docker：

```
docker exec -it sglang_deepseek bash
```

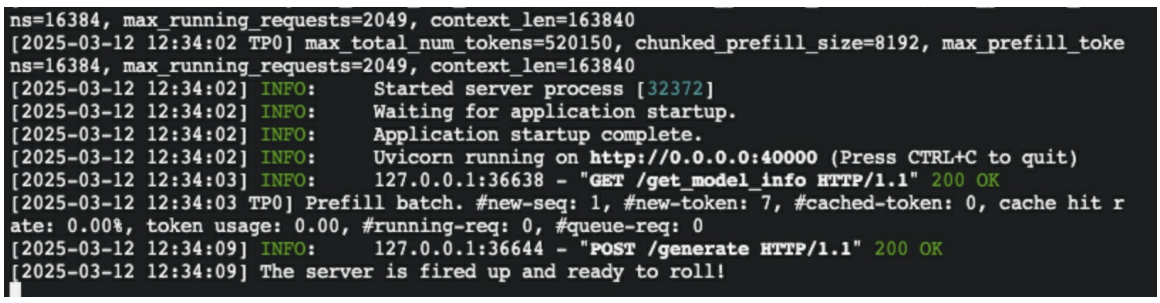
4、在2台实例docker中设置NCCL环境变量：

```
export NCCL_SOCKET_IFNAME=eth0
export NCCL_IB_GID_INDEX=3
export NCCL_IB_TIMEOUT=22
export NCCL_IB_RETRY_CNT=7
export NCCL_IB_QPS_PER_CONNECTION=8
export NCCL_DEBUG=INFO
```

4、在2台实例docker中分别执行以下命令，注意确保2台实例所在的安全组已放开20000端口，dist-init-addr替换为主节点私有网络IP地址：

```
#### 主节点：
python3 -m sglang.launch_server --model-path /workspace/model/DeepSeek-R1/ --tp 16 --dist-init-addr
192.168.0.6:20000 --nnodes 2 --node-rank 0 --trust-remote-code --host 0.0.0.0 --port 30000

#### 从节点：
python3 -m sglang.launch_server --model-path /workspace/model/DeepSeek-R1/ --tp 16 --dist-init-addr
192.168.0.6:20000 --nnodes 2 --node-rank 1 --trust-remote-code --host 0.0.0.0 --port 30000
```

如看到以下打印，则代表server已正常工作。



```
ns=16384, max_running_requests=2049, context_len=163840
[2025-03-12 12:34:02 TP11] max_total_num_tokens=520150, chunked_prefill_size=8192, max_prefill_tok
ens=16384, max_running_requests=2049, context_len=163840
[2025-03-12 12:34:02 TP15] max_total_num_tokens=520150, chunked_prefill_size=8192, max_prefill_tok
ens=16384, max_running_requests=2049, context_len=163840
[2025-03-12 12:34:02 TP14] max_total_num_tokens=520150, chunked_prefill_size=8192, max_prefill_tok
ens=16384, max_running_requests=2049, context_len=163840
[2025-03-12 12:34:02 TP12] max_total_num_tokens=520150, chunked_prefill_size=8192, max_prefill_tok
ens=16384, max_running_requests=2049, context_len=163840
[2025-03-12 12:34:02 TP13] max_total_num_tokens=520150, chunked_prefill_size=8192, max_prefill_tok
ens=16384, max_running_requests=2049, context_len=163840
[2025-03-12 12:34:02 TP10] max_total_num_tokens=520150, chunked_prefill_size=8192, max_prefill_tok
ens=16384, max_running_requests=2049, context_len=163840
[2025-03-12 12:34:02 TP8] max_total_num_tokens=520150, chunked_prefill_size=8192, max_prefill_toke
ns=16384, max_running_requests=2049, context_len=163840
INFO:     Started server process [3425]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:40000 (Press CTRL+C to quit)
```

5、测试server方法同单机部署。

常见Q&A

1、如在启动sglang server过程中遇到Fatal Python error: Segmentation fault错误，可尝试升级容器中NCCL版本解决，升级命令如下：

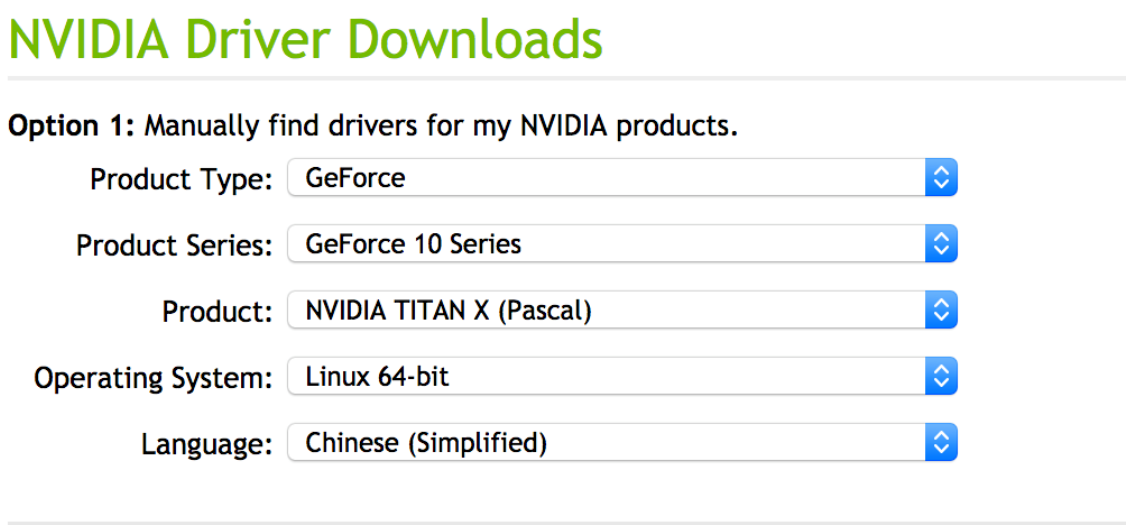
```
pip install nvidia-nccl-cu12==2.25.1 -i http://mirrors.baidubce.com/pypi/simple --trusted-host=mirrors.baidubce.com
```

常见问题

一般类问题

1、NVIDIA 深度学习开发卡应该使用哪种驱动？

请在Nvidia官网按照下图的选项进行驱动下载：



2、GPU云服务器支持的显卡型号有哪些？

关于GPU的显卡型号，GPU云服务支持多种GPU型号和实例规格，您可以参考GPU云服务器实例规格族。

3、GPU服务器是否有GPU卡状态监控和报警机制，包括GPU使用率、内存使用率、温度、状态等？

1. 登录百度智能云控制台选择云服务器BCC，点击实例进入实例列表页面。

2. 选择需要监控的实例名称，进入实例详情页面，然后点击监控按钮进入监控详情页面。

3. 在监控详情页面的最下侧，找到扩展监控项，然后选择GPU后，即可看到GPU卡的监控信息。

4、系统中使用 nvidia-smi 查看的CUDA版本与 nvcc -V 看到的CUDA版本不一样，应该以哪个为准？

您执行命令nvidia-smi查询到的CUDA版本代表CUDA Driver版本，此版本是驱动指定，您指定的CUDA 版本通常为CUDA Runtime版本，可通过在操作系统中通过nvcc --version 查看，绝大部分的应用依赖Runtime版本。

5、GPU实际无负载，但使用 nvidia-smi 查看GPU利用率时显示100%

Tesla系列GPU卡提供ECC功能，当GPU云服务器加载GPU驱动时，因 ECC Memory Scrubbing 机制存在，概率出现GPU利用率显示100%情况。遇到此情况时，需要用户在root权限下，执行 nvidia-smi -pm 1 命令，让GPU Driver 进入 Persistence 模式，从而解决此问题。

```
root@instance-xxxxxxxx:~# nvidia-smi -pm 1
```

6、怎么变更GPU实例的实例规格

GPU云服务器的变更策略如下： 支持同规格族同GPU型号的实例规格进行规格升配，暂不支持带本地盘的实例以及A100实例变配。

具体操作为：

1、登录BCC云服务器控制台 2、在需要编配的实例操作栏中，选择更多->实例设置->配置变更 3、选择目标变配的配置

7、购买GPU实例后，通过nvidia-smi获取的显存规格少于标称规格

GPU默认开启了ECC（error correcting code，错误检查和纠正）功能，用来检查和纠正可能会在数据传输和存储过程中引发的比特错误，开启时会使可用显存减少，并伴随部分性能损失。为提高数据的正确性，建议您保持开启状态。

8、按量付费/后付费的GPU实例是否支持关机不计费

对于后付费的GPU云服务器实例，关机不计费的限制如下:

- 支持不带本地盘的实例开启关机不计费。
- GN5系列A100 GPU实例暂不支持关机不计费。
- **特别提示: 选择关机不计费后，再次开机可能遇到因资源售罄导致的启动失败，请谨慎选择该选项。**

具体操作为： 在实例操作列表中的实例状态选择停止，并在弹出的页面中选择关机不计费，如果选择关机选项，则实例仍正常计费。

9、普通云服务是否支持变配GPU云服务器，或者GPU云服务器是否支持跨规格族，跨不同卡变配

不支持。

10、为什么购买GPU实例后，执行命令nvidia-smi找不到GPU显卡？

当您执行命令nvidia-smi无法找到GPU显卡时，通常是由于您的GPU实例未安装或者未成功安装NVIDIA驱动。请根据您所购买的GPU实例规格选择对应的操作指引来安装驱动，具体说明如下：https://cloud.baidu.com/doc/GPU/s/Sllz3tvfi

如何检测GPU常见故障

在GPU云服务器使用过程中可能会出现GPU硬件故障或者亚健康状态，如果您发现应用程序出现报错或者GPU硬件性能下降，可通过以下检测方法检测是否存在故障，发现故障后，可通过重启实例或者重置GPU卡等方式修复，如果问题持续发生，请您提交工单。

掉卡故障检测

您可依次通过以下几种检测方法，判断当前实例是否存在GPU掉卡故障。

方法一：检测GPU掉卡数量

检测步骤

1. 登录实例。
2. 执行命令：

```
lspci -d 10de:|grep "rev ff"
```

3. 若结果不为空，则表示有掉卡问题；结果的行数表示掉卡的个数。

解决方法

1. 登录控制台重启实例。
2. 如果重启无法恢复，请提交工单。

方法二：检测GPU数量

检测步骤

1. 登录实例。
2. 执行命令：

```
lspci -d 10de:|grep -v 1af1
```

3. 查看GPU卡数量是否和预期相符。

解决方法

请提交工单。

方法三：检测驱动是否能识别到卡

检测步骤和解决方法见如何采用Xid方法检测故障问题

方法四：驱动冲突检测

检测步骤

1. 登录实例。
2. 执行命令：

```
lsmod |grep nouveau
```

3. 结果中包含nouveau，如下图。

```
[root@v100 ~]# lsmod | grep nouveau
nouveau              1869738  0
mxm_wmi                13021  1 nouveau
wmi                    21636  2 mxm_wmi,nouveau
video                  24538  1 nouveau
i2c_algo_bit           13413  1 nouveau
drm_kms_helper        179394  2 cirrus,nouveau
ttm                   114635  2 cirrus,nouveau
drm                   429744  5 ttm,drm_kms_helper,cirrus,nouveau
```

解决方法

参考文档，禁用nouveau模块https://cloud.baidu.com/doc/GPU/s/0kqm6s30y

链路故障检测

您可依次通过以下几种检测方法，判断当前实例是否存在链路故障。

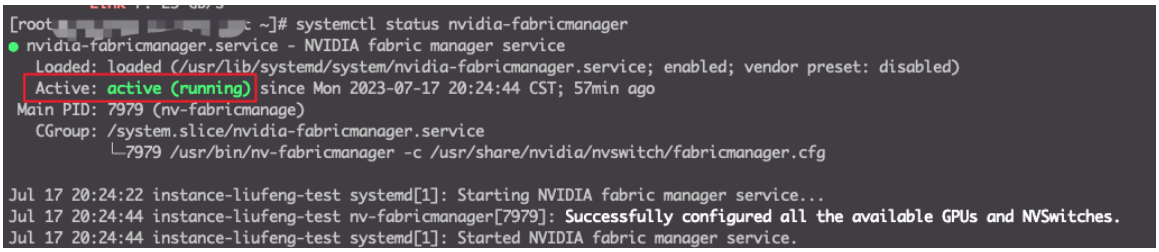
方法一：检测fabricmanager是否启动

检测步骤

1. 登录实例。（适用实例：EHC GN5 A100/A800，BCC GN5 A100-40G）
2. 执行命令：

```
systemctl status nvidia-fabricmanager
```

3. 检查结果，fabricmanager未启动为异常状态，下图为正常状态。



解决方法

请提交工单。

方法二：Nvlink驱动故障检测

检测步骤和解决方法见如何采用Xid方法检测故障问题

内存故障检测

您可根据GPU实例的架构类型（通用架构、ampere架构、其他架构），采用不同的检测方法，判断当前实例是否存在内存故障。

通用架构

适用于所有GPU实例规格族，您可通过以下检测方法，判断当前实例是否存在链路故障。

检测ECC MODE是否开启

检测步骤

1. 登录实例。
2. 执行命令：

```
nvidia-smi -q | grep EC
```

3. ECC模式已开启则显示"Current ECC Mode：Enabled"，未开启则显示"Current ECC Mode：Disabled"。

解决方法

1. 登录实例。
2. 执行命令：

```
nvidia-smi -i [GPU的ID] -e 1
```

其中[GPU的ID]是您未开启ECC模式的GPU的编号，比如0，1，2等。

ampere架构

适用于GN5型A100、A800和A10 GPU实例规格族，您可依次通过以下几种检测方法，判断当前实例是否存在内存故障。

方法一：检测是否出现显存重映射失败

检测步骤

1. 登录实例。
2. 执行命令：

```
nvidia-smi -q|grep "Remapping Failure Occurred    .+?Yes"
```

3. 命令有输出结果，如下图。

解决方法

请提交工单。

方法二：检测是否出现显存重映射阻塞

检测步骤

1. 登录实例。
2. 执行命令：

```
nvidia-smi -q|grep -P "Pending    .+?Yes"
```

3. 命令有输出，如下图。

解决方法

登录控制台，对异常节点进行重启。

方法三：检测显存重映射是否超过阈值

检测步骤

1. 登录实例。
2. 执行命令：

```
nvidia-smi -q|grep -P "(Low|None)    .+?[0-9]+ bank"|grep -v "0 bank"
```

3. 命令有输出，如下图。

解决方法

请提交工单。

方法四：检测不可恢复SRAM的值

检测步骤

1. 登录实例。
2. 执行命令：

```
nvidia-smi --query-gpu=ecc.errors.uncorrected.volatile.sram --format=csv -i ${bus_id}
```

3. 命令输出结果非0。

解决方法

请提交工单。

方法五：检测是否出现ECC错误 检测步骤和解决方法见如何采用Xid方法检测故障问题

其他架构

适用于GN1型P4，GN3型V100和T4 GPU实例规格族，您可依次通过以下几种检测方法，判断当前实例是否存在内存故障。

方法一：检测Single Bit ECC

检测步骤

1. 登录实例。
2. 执行命令：

```
nvidia-smi --query-gpu=retired_pages.single_bit_ecc.count --format=csv
```

3. 结果之和>60。

解决方法

请提交工单。

方法二：检测Double Bit ECC

检测步骤

1. 登录实例。
2. 执行命令：

```
nvidia-smi --query-gpu=retired_pages.double_bit.count --format=csv
```

3. 结果之和>5。

解决方法

请提交工单。

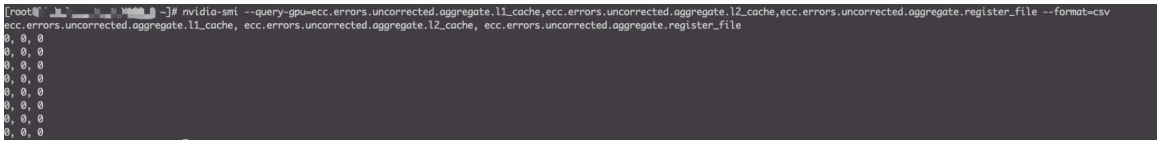
方法三：检测不可恢复的Cache的值

检测步骤

1. 登录实例。
2. 执行命令：

```
nvidia-smi --query-gpu=ecc.errors.uncorrected.aggregate.l1_cache,ecc.errors.uncorrected.aggregate.l2_cache,ecc.errors.uncorrected.aggregate.e --format=csv
```

3. 返回的结果任意一行，和值大于5（每一行表示一张GPU卡的结果值）。



解决方法

请提交工单。

方法四：检测是否出现ECC错误 检测步骤和解决方法见如何采用Xid方法检测故障问题

其他硬件故障检测

您可依次检测以下几项，判断当前实例是否存在其他硬件故障。

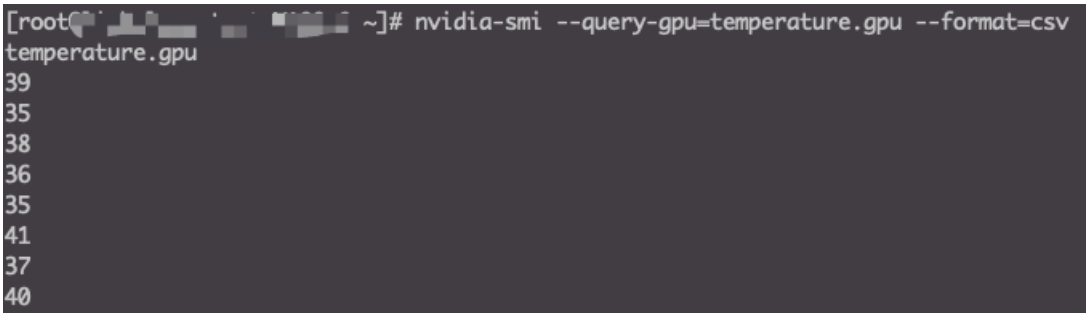
GPU温度检测

检测步骤

1. 登录实例。
2. 执行命令：

```
nvidia-smi --query-gpu=temperature.gpu --format=csv
```

3. 结果中任意一行的值大于85（每一行表示一个GPU卡的温度）。



解决方法

请提交工单。

GPU功耗检测

检测步骤

1. 登录实例。
2. 执行命令：

```
nvidia-smi
```

3. 存在某张卡的功率一栏是Unknown或者err。

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 450.248.02   Driver Version: 450.248.02   CUDA Version: 11.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:41:00.0 Off |                    0 |
| N/A   39C    P0    43W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000000:42:00.0 Off |                    0 |
| N/A   35C    P0    41W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  On   | 00000000:43:00.0 Off |                    0 |
| N/A   38C    P0    42W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  On   | 00000000:44:00.0 Off |                    0 |
| N/A   36C    P0    42W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   4  Tesla V100-SXM2...  On   | 00000000:61:00.0 Off |                    0 |
| N/A   35C    P0    41W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   5  Tesla V100-SXM2...  On   | 00000000:62:00.0 Off |                    0 |
| N/A   41C    P0    41W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   6  Tesla V100-SXM2...  On   | 00000000:63:00.0 Off |                    0 |
| N/A   37C    P0    43W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   7  Tesla V100-SXM2...  On   | 00000000:64:00.0 Off |                    0 |
| N/A   40C    P0    43W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

解决方法

1. 登录控制台重启实例。
2. 如果重启无法恢复，请提交工单。

如何采用Xid方法检测故障问题

检测步骤

1. 登录实例。
2. 执行命令。
   - centos：

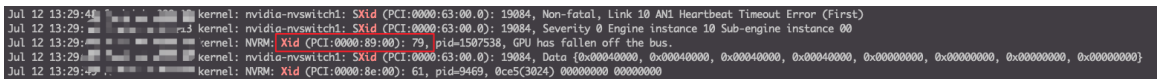
```
dmesg -T |grep Xid
```

   - ubuntu：

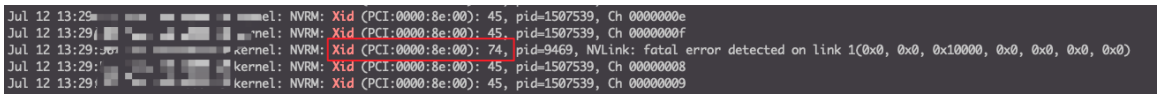
```
journalctl --since `date -d "10 days ago" "+%Y-%m-%d"`|grep Xid
```

3. 查看结果。

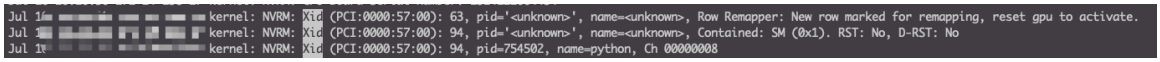
结果中包含Xid 79，如下图，则存在GPU掉卡故障。

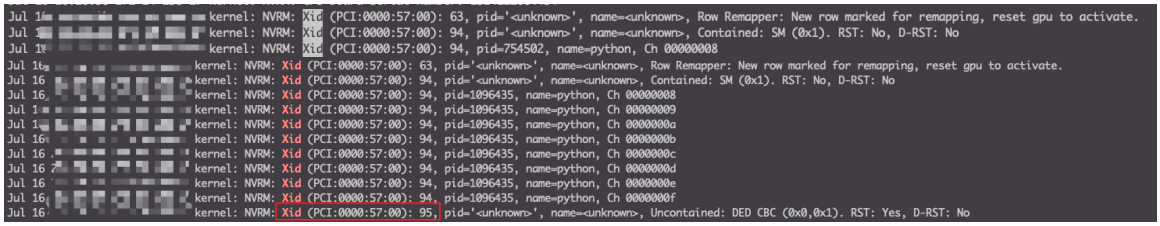


结果中包含Xid 74，如下图，则存在GPU链路故障。



结果中包含Xid 48/63/94/95，如下图，则存在内存故障。





解决方法

1. 登录控制台重启实例。
2. 如果重启无法恢复，请提交工单。

如何检测RDMA常见故障

在GPU云服务器使用过程中可能会出现RDMA硬件故障或者亚健康状态，如果您发现应用程序出现报错或者RDMA硬件性能下降，可通过以下检测方法检测是否存在故障，发现故障后，可通过重启实例等方式修复，如果问题持续发生，请您提交工单。

网卡状态检测

您可通过以下检测方法，判断当前实例是否存在网卡状态故障。

检测步骤

1. 登录实例。
2. 查询所有RDMA网卡的接口状态，执行命令：

```
ibdev2netdev |grep -v eth0
```

```
[root@testgpu tmp]# ibdev2netdev |grep -v eth0
mlx5_1 port 1 ==> eth1 (Up)
mlx5_2 port 1 ==> eth2 (Up)
mlx5_3 port 1 ==> eth3 (Up)
mlx5_4 port 1 ==> eth4 (Up)
mlx5_5 port 1 ==> eth5 (Up)
mlx5_6 port 1 ==> eth6 (Up)
mlx5_7 port 1 ==> eth7 (Up)
mlx5_8 port 1 ==> eth8 (Up)
[root@testgpu tmp]#
```

3. 查看是否有接口处于Down状态，如下eth2接口处于Down状态：

```
[root@testgpu tmp]# ibdev2netdev |grep -v eth0
mlx5_1 port 1 ==> eth1 (Up)
mlx5_2 port 1 ==> eth2 (Down)
mlx5_3 port 1 ==> eth3 (Up)
mlx5_4 port 1 ==> eth4 (Up)
mlx5_5 port 1 ==> eth5 (Up)
mlx5_6 port 1 ==> eth6 (Up)
mlx5_7 port 1 ==> eth7 (Up)
mlx5_8 port 1 ==> eth8 (Up)
[root@testgpu tmp]#
```

解决方法

- centos、rocky、baidulinux系统，请尝试使用如下命令恢复：

```
bash /var/lib/cloud/scripts/per-boot/bcc_elastic_net_centos_cloudinit.sh
```

- ubuntu系统，请尝试使用如下命令恢复：

```
bash /var/lib/cloud/scripts/per-boot/bcc_elastic_net_ubuntu_cloudinit.sh
```

如果运行命令后没有恢复，请提交工单。

网卡抖动检测

您可通过以下检测方法，判断当前实例是否存在网卡抖动。

检测步骤

1. 登录实例。
2. 查询所有RDMA网卡的接口名称，执行命令：

```
show_gids |grep v2|awk '{print $7}'|sed '/^$/d'|grep -v eth0
```

3. 依次查询每个RDMA网卡Link down的次数，执行命令：

```
dmesg -T | grep -i eth|grep -i link|grep -i down|wc -l
journalctl --since `date -d "10 days ago" "+%Y-%m-%d"`| grep -i eth|grep -i link|grep -i down|wc -l
```

4. 次数大于阀值8，说明网卡抖动。

解决方法

请提交工单。

网卡配置检测

您可依次检测以下几项，判断当前实例是否存在网卡配置故障。

mtu检测

检测步骤

1. 登录实例。
2. 查询所有RDMA网卡的接口名称，执行命令：

```
show_gids |grep v2|awk '{print $7}'|sed '/^$/d'|grep -v eth0
```

```
[root@testgpu tmp]# show_gids |grep v2|awk '{print $7}'|sed '/^$/d'|grep -v eth0
eth1
eth2
eth3
eth4
eth5
eth6
eth7
eth8
```

3. 依次查询每个RDMA网卡的mtu，执行命令：

```
ip -4 -j -p addr show dev ${ifname}|grep mtu
```

```
[root@testgpu tmp]# ip -4 -j -p addr show dev eth1|grep mtu
        "mtu": 4200,
```

4. mtu值不等于预期值。

解决方法

请提交工单。

ip地址检测

检测步骤

1. 登录实例。
2. 查询所有RDMA网卡的接口名称，执行命令：

```
show_gids |grep v2|awk '{print $7}'|sed '/^$/d'|grep -v eth0
```

3. 依次查询每个RDMA网卡的IP地址，执行命令：

```
ip -4 -j -p addr show dev ${ifname}|grep local
```

```
[root@testgpu tmp]# ip -4 -j -p addr show dev eth1|grep local
                "local": "25.5.0.2",
```

4. 如果输出没有结果。

解决方法

请提交工单。

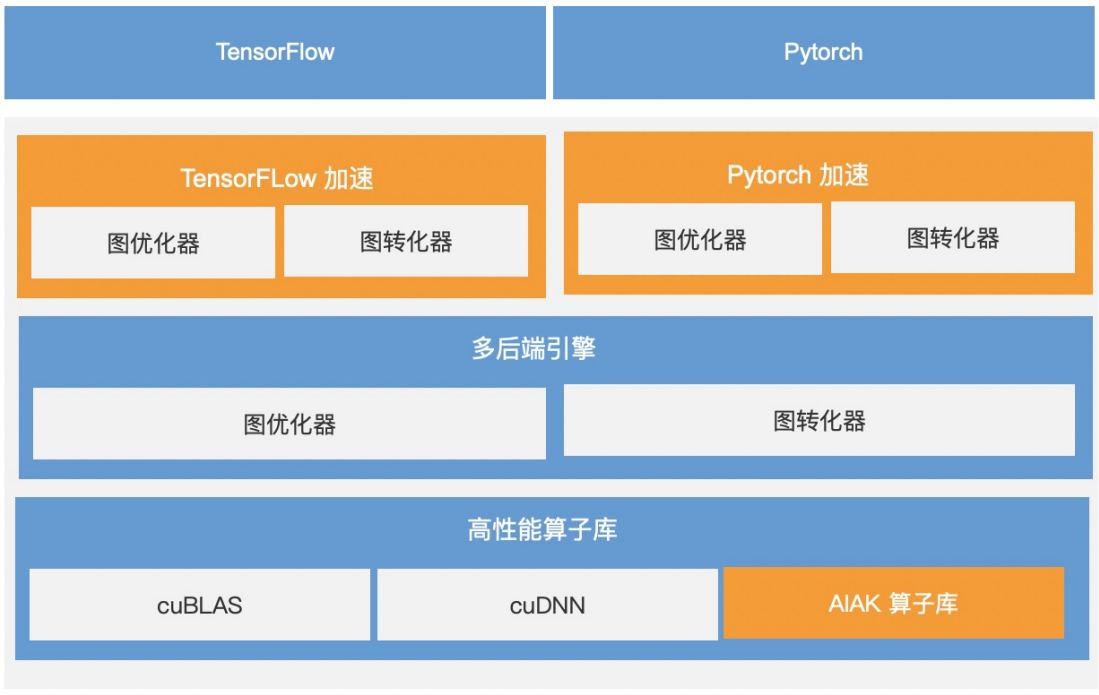
AI加速套件AIAK

AIAK推理加速组件

概览

AIAK是面向人工智能任务提供的加速引擎，用于优化基于AI主流计算框架搭建的模型，能显著提升AI任务开发、部署的运行效率。

其中，AIAK推理加速套件是通过优化主流的AI框架，例如：Tensorflow、PyTorch产出的模型，降低在线推理延迟、提升服务吞吐，大幅增加异构资源使用效率的推理优化引擎，结合百度智能云的IaaS资源，可进一步提升用户AI场景下的计算效率。



应用场景

AIAK推理加速可支持但是不限于以下场景模型：

- 自然语言处理，例如Bert、Transformer等。
- 图像识别，例如ResNet50、MobileNetSSD等。

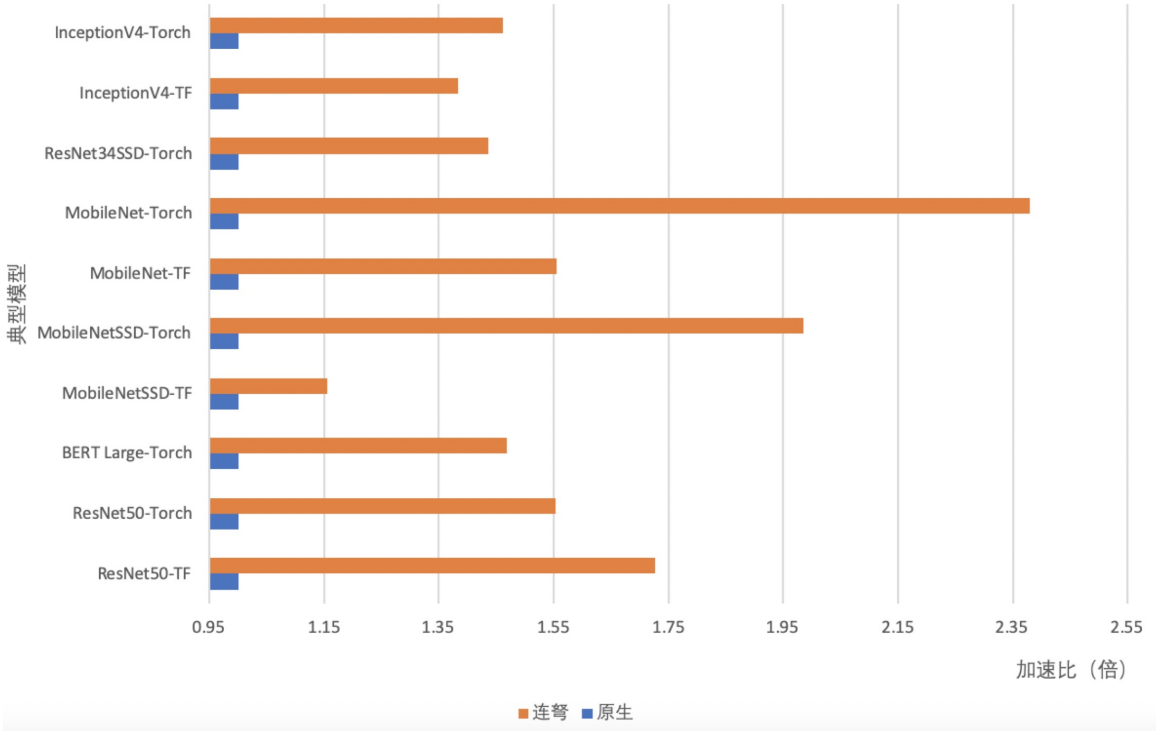
方案优势

AIAK推理加速组件具有以下优势。

- 多框架兼容：提供对TensorFlow和PyTorch等框架兼容。
- 多模型支持：支持对业界主流模型的加速。

- 轻量便捷：只需少量代码适配即可开启加速能力。

以下列举了一些典型模型基于AIAK和NVIDIA Tesla T4 GPU的推理时延收益，数值越高代表时延越低。



配置步骤

环境准备

- GPU云服务器资源。
- AIAK推理加速的部署需满足以下运行环境。
  - AI开发框架版本：Pytorch 1.8及以上版本，Tensorflow 1.15及以上版本。
  - GPU运行环境：Cuda 10.2及以上版本，TensorRT 7及以上版本。
  - Python版本：3.6版本。

使用方法

AIAK推理加速支持多产品使用，本文档以加速ResNet50为例子介绍如何在GPU云服务器中使用AIAK推理加速组件，如您需要结合百度智能云容器服务引擎，可参考云原生AI使用文档。

TensorFlow框架

1. 登录百度智能云GPU实例。
2. 提交工单获取最新的加速包下载链接。
3. 准备业务需要的模型，此处以ResNet50为示例。

```
 import os
import numpy as np
import tensorflow.compat.v1 as tf
tf.compat.v1.disable_eager_execution()

def _wget_demo_tgz():
    # 此处以下载一个公开的resnet50模型为例。
    url = 'https://cce-ai-native-package-bj.bj.bcebos.com/aiak-inference/examples/models/resnet50.pb'
    local_tgz = os.path.basename(url)
    local_dir = local_tgz.split('.')[0]
    if not os.path.exists(local_dir):
        luno.util.wget(url, local_tgz)
        luno.util.unpack(local_tgz)
    model_path = os.path.abspath(os.path.join(local_dir, "frozen_inference_graph.pb"))
    graph_def = tf.GraphDef()
    with open(model_path, 'rb') as f:
        graph_def.ParseFromString(f.read())
    # 以随机数作为测试数据，可替换为自己的数据集
    test_data = np.random.rand(1, 800, 1000, 3)
    return graph_def, {'image_tensor:0': test_data}

graph_def, test_data = _wget_demo_tgz()

input_nodes=['image_tensor']
output_nodes = ['detection_boxes', 'detection_scores', 'detection_classes', 'num_detections', 'detection_masks']
```

4. 运行模型并获取benchmark推理时延。

```
 import time

def benchmark(model):
    tf.reset_default_graph()
    with tf.Session() as sess:
        sess.graph.as_default()
        tf.import_graph_def(model, name="")
        # Warmup!
        for i in range(0, 1000):
            sess.run(['image_tensor:0'], test_data)
        # Benchmark!
        num_runs = 1000
        start = time.time()
        for i in range(0, num_runs):
            sess.run(['image_tensor:0'], test_data)
        elapsed = time.time() - start
        rt_ms = elapsed / num_runs * 1000.0
        # Show the result!
        print("Latency of model: {:.2f} ms.".format(rt_ms))

#### original graph
print("=====Original Performance=====")
benchmark(graph_def)
```

5. 引入AIAK推理优化组件。

```
 import luno
optimized_model = luno.optimize(
    graph_def,              # 待优化的模型，此处是tf.GraphDef, 也可以配置为SavedModel的路径。
    'o1',                   # 优化级别，o1或o2。
    device_type='gpu',      # 目标设备，gpu/cpu
    outputs=['detection_boxes', 'detection_scores', 'detection_classes', 'num_detections', 'detection_masks']
)
```

6. 运行优化后的模型并获取benchmark推理时延。

```
 #### optimized graph
print("=====Optimized Performance=====")
benchmark(optimized_model)
```

可看到同模型在使用AIAK组件后在推理时延上的收益。

```
2022-04-27 11:03:19 =====Original Performance=====
2022-04-27 11:03:19 Latency of model: 7.14 ms.
2022-04-27 11:03:19 [array([112])]
2022-04-27 11:03:19 =====Optimized Performance=====
2022-04-27 11:03:19 Latency of model: 4.23 ms.
```

Pytorch框架

1. 登录百度智能云GPU实例。

2. 提交工单获取最新的加速包下载链接。

3. 准备业务需要的模型，此处以ResNet50为示例。

```
 import os
import time
import torch
import torchvision.models as models

model = models.resnet50().float().cuda()
model = torch.jit.script(model).eval()     # 使用jit转为静态图
dummy = torch.rand(1, 3, 224, 224).cuda()
```

4. 运行模型并获取benchmark推理时延。

```
 @torch.no_grad()
def benchmark(model, inp):
    for i in range(100):
        model(inp)
    start = time.time()
    for i in range(200):
        model(inp)
    elapsed_ms = (time.time() - start) * 1000
    print("Latency: {:.2f}".format(elapsed_ms / 200))

#### benchmark before optimization
print("before optimization:")
benchmark(model, dummy)
```

5. 引入AIAK推理优化组件。

```
 import luno

optimized_model = luno.optimize(
    model,
    'gpu',
    input_shapes=[[1, 3, 224, 224]],
    test_data=[dummy],
)
```

6. 运行优化后的模型并获取benchmark推理时延。

```
 #### benchmark after optimization
print("after optimization:")
benchmark(optimized_model, dummy)
```

可看到同模型在使用AIAK组件后在推理时延上的收益。

```
2022-04-27 15:08:04 before optimization:
2022-04-27 15:08:04 Latency: 5.07
2022-04-27 15:08:04 after optimization:
2022-04-27 15:08:04 Latency: 3.26
```

GPU驱动版本选择推荐

本文主要记录线上GPU实例创建时，可以选装及通过自动化安装脚本安装的GPU驱动相关组件的发布记录信息。

NVIDIA驱动生命周期

根据NVIDIA驱动官网文档可知，NVIDIA有三种不同生命周期的驱动分支：New Feature Branch(NFB)、Production Branch(PB)、Long Term Support Branch(LTSB)。

Table 1. NVIDIA Driver Branches

	New Feature Branch (NFB)	Production Branch (PB)	Long Term Support Branch (LTSB)
Target Customers	Early adopters who want to evaluate new features	Use in production for enterprise/datacenter GPUs	Use in production for enterprise/datacenter GPUs and for customers looking for a longer cycle of support.
Major Release Cadence	At least once every 3 months	Twice a year. See also note below	At least once per hardware architecture. See also note below
Length of support	N/A	1 year	3 years
Minor release (bug updates and critical security updates)	N/A	Yes. Quarterly bug and security releases for 1 year.	Yes. Quarterly bug and security releases for 1 year.

1. NFB分支作为主要的功能发布分支，由新的分支X编号表示，主要针对那些希望评估新功能的早期采纳者发布，每三个月至少会发布一个NFB分支。
2. PB分支是可以在生产中使用GPU驱动分支，提供长达1年的bug修复和安全更新。每年会发布两个PB分支。
3. LTSB分支是PB分支的一种，但是它有更长的支持和维护周期（3年）。LTSB版本将在支持它们的3年内，通过小版本，在合理的努力基础上接收错误更新和关键安全更新。

当前CUDA工具包和NVIDA驱动的支持矩阵如下表：

	R470	R535	R550	R570
Branch Designation	Long Term Support Branch	Long Term Support Branch	Production Branch	Production Branch
End of Life	July 2024	June 2026	June 2025	February 2026
Maximum CUDA Version Supported	CUDA 11.0+ This driver branch supports CUDA 11.x (through CUDA minor version compatibility)	CUDA 12.0+ This driver branch supports CUDA 12.x (through CUDA minor version compatibility)	CUDA 12.0+ This driver branch supports CUDA 12.x (through CUDA minor version compatibility)	CUDA 12.0+ This driver branch supports CUDA 12.x (through CUDA minor version compatibility)

控制台驱动选装支持

控制台选装支持的驱动、CUDA、cuDNN版本版本将陆续进行收敛，预期只保留LTSB/PB分支最新的驱动版本，及对应的CUDA版本，最新的cuDNN版本。

NVIDIA Tesla驱动版本	CUDA版本	cuDNN版本	Supported GPU
570.124.06	12.8.1	9.8.0	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB
550.144.03	12.4.1	9.6.0	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB
535.230.02	12.2.2	9.6.0	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB

自动化安装脚本支持

对于不在控制台驱动选装列表里的驱动组件，可以选装通过自动安装脚本方式，在使用公共镜像创建GPU实例时注入自动安装脚本，并将驱动、CUDA、cuDNN版本修改为期望的版本，来实现在GPU实例创建时自动化安装GPU驱动组件(注意，实例创建后自动化安装驱动组件需要近十分钟时间，请勿执行重启、关机等操作)。

以下将列出自动化安装脚本方式支持的NVIDIA Tesla驱动、CUDA、cuDNN版本，及与GPU卡、OS等依赖关系。

驱动列表

驱动版本	Arch	Platform	Supported CUDA	Supported OS	Supported GPU	Release Notes
570.124.06	x86_64	linux	12.x	BaiduLinux 3.0 CentOS 8.0~8.4 Ubuntu 20.04/22.04/24.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 12.0.0	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB	tesla-release-notes-570-124-06
570.86.15	x86_64	linux	12.x	BaiduLinux 3.0 CentOS 8.0~8.4 Ubuntu 20.04/22.04/24.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB	tesla-release-notes-570-86-15

560.35.03	x86_64	linux	12.x	BaiduLinux 3.0 CentOS 8.0~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800,H20, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB	tesla-release-notes-560-35-03
550.144.03	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-550-144-03
550.127.08	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-550-127-08
550.127.05	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-	tesla-release-notes-550-

				Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	V100-SXM2-32GB, TITAN-X	127-05
550.90.12	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-550-90-12
550.90.07	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-550-90-07
550.54.15	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-550-54-15
				BaiduLinux		

535.230.02	x86_64	linux	12.x	2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-535-230-02
535.216.03	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-535-216-03
535.216.01	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-535-216-01
535.183.06	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-535-

				Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0		183-06
535.161.08	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-535-161-08
535.154.05	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04 Rocky Linux 8.5~8.8/9.0~9.3 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-535-154-05
535.129.03	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04 Rocky Linux 8.5~8.8/9.0~9.2 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-535-129-03
				BaiduLinux 2.0/3.0 CentOS 7.1~8.4		

535.104.12	x86_64	linux	12.x	7.1~8.4 Ubuntu 20.04 Rocky Linux 8.5~8.8/9.0~9.2 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-535-104-12
535.104.05	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04 Rocky Linux 8.5~8.8/9.0~9.2 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-535-104-05
535.86.10	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04 Rocky Linux 8.5~8.8/9.0~9.2 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-535-86-10
535.54.03	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04 Rocky Linux 8.5~8.8/9.0~9.2 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-535-54-03
				BaiduLinux		

525.147.05	x86_64	linux	12.x	2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04 Rocky Linux 8.5~8.8/9.0~9.2 Debian 11.7.0/12.0.0	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-525-147-05
525.125.06	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04 Rocky Linux 8.5~8.8/9.0~9.2 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-525-125-06
525.105.17	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04 Rocky Linux 8.5~8.8/9.0~9.2 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-525-105-17
525.85.12	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04 Rocky Linux 8.5~8.8/9.0~9.2 Debian	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-525-85-12

				11.7.0/12.0.0		
525.60.13	x86_64	linux	12.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04 Rocky Linux 8.5~8.8/9.0~9.2 Debian 11.7.0/12.0.0	A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-525-60-13
520.61.05	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04	A100-SXM4-40GB, A100-SXM4-80GB	无
515.105.01	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04 Rocky Linux 8.5~8.7/9.0~9.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-515-105-01
515.86.01	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04 Rocky Linux 8.5~8.6	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-515-86-01
515.65.07	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-515-65-07

515.48.07	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-515-48-07
510.108.03	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-510-108-03
510.73.08	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-510-73-08
470.256.02	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-470-256-02
470.239.06	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-470-239-06
470.223.02	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-470-223-02
470.199.02	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-470-199-02

470.182.03	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-470-182-03
470.161.03	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-470-161-03
470.141.10	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-470-141-10
470.129.06	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A30, A100-SXM4-40GB, A100-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-470-129-06
460.106.00	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A30, A100-SXM4-40GB, A100-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-460-106-00
460.91.03	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A30, A100-SXM4-40GB, A100-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-460-91-03
450.248.02	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.	A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-450-248-02

				04		
450.236.01	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-450-236-01
450.216.04	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-450-216-04
450.203.08	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-450-203-08
450.191.01	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A100-SXM4-40GB, A100-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-450-191-01
450.119.03	x86_64	linux	11.x	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04	A100-SXM4-40GB, A100-SXM4-80GB, Tesla-K40m, Tesla-P4, Tesla-P40, Tesla-T4, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	tesla-release-notes-450-119-03

CUDA列表

CUDA 版本	Arch	Platform	最低支持驱动版本	Supported OS	可选 cuDNN	Supported GPU	Release Notes
12.8.1	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-	12.8.1

				Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	
12.8.0	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.8.0
12.6.3	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.6.3
12.6.2	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.6.2
				BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu	9.8.0 9.6.0 9.5.0 9.4.0		

				Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1		
12.6.1	x86_64	linux	≥525.60.13	Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.6.1
12.6.0	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.6.0
12.5.1	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.5.1
12.4.1	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.4.1
				BaiduLinux 2.0/3.0	9.8.0		

12.4.0	x86_64	linux	≥525.60.13	CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, H20, H20-3e, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.4.0
12.3.2	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.3.2
12.3.1	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12.0.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.3.1
12.3.0	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0/12	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.3.0

				.0.0			
12.2.2	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.2.2
12.2.1	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.2.1
12.2.0	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, L20, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.2.0
12.1.1	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.1.1

12.0.1	x86_64	linux	≥525.60.13	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04/22.04 Rocky Linux 8.*/9.* Debian 11.7.0	9.8.0 9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, H800, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	12.0.1
11.8.0	x86_64	linux	≥450.80.02	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04/22.04 Rocky Linux 8.*/9.*	9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1 8.7.0 8.6.0 8.5.0 8.4.0~8.4.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	11.8.0
11.7.1	x86_64	linux	≥450.80.02	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20.04/22.04 Rocky Linux 8.*	9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8.9.7 8.8.1 8.7.0 8.6.0 8.5.0 8.4.0~8.4.1	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	11.7.1
			≥450.8	BaiduLinux 2.0/3.0 CentOS	9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-	

11.6.2	x86_64	linux	0.02	7.1~8.4 Ubuntu 18.04/20. 04	.9.7 8.8.1 8.7.0 8.6.0 8.5.0 8.4.0~8 .4.1 8.2.1	SXM4-80GB, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	11.6.2
11.5.2	x86_64	linux	≥450.8 0.02	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20. 04	9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8 .9.7 8.8.1 8.7.0 8.6.0 8.5.0 8.4.0~8 .4.1 8.3.3	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	11.5.2
11.4.4	x86_64	linux	≥450.8 0.02	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 18.04/20. 04	9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8 .9.7 8.8.1 8.7.0 8.6.0 8.5.0 8.4.0~8 .4.1 8.2.4 8.2.2	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	11.4.4
			≥450.8	BaiduLinux 2.0/3.0 CentOS	9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-	

11.3.1	x86_64	linux	0.02	7.1~8.4 Ubuntu 16.04/18. 04/20.04	.9.7 8.8.1 8.7.0 8.6.0 8.5.0 8.4.0~8 .4.1 8.2.1 8.2.0	SXM4-80GB, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	11.3.1
11.2.2	x86_64	linux	≥450.8 0.02	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 16.04/18. 04/20.04	9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8 .9.7 8.8.1 8.7.0 8.6.0 8.5.0 8.4.0~8 .4.1 8.1.1 8.1.0	A10, A30, A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	11.2.2
11.1.1	x86_64	linux	≥450.8 0.02	BaiduLinux 2.0/3.0 CentOS 7.1~8.4 Ubuntu 16.04/18. 04/20.04	9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0 8.9.0~8 .9.7 8.8.1 8.7.0 8.6.0 8.5.0 8.4.0~8 .4.1 8.0.5 8.0.4	A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	11.1.1
				BaiduLinux 2.0/3.0	9.6.0 9.5.0 9.4.0 9.3.0 9.2.1 9.1.1 9.0.0		

11.0.3	x86_64	linux	≥450.80.02	CentOS 7.1~8.4 Ubuntu 16.04/18.04/20.04	8.9.0~8.9.7 8.8.1 8.7.0 8.6.0 8.5.0 8.4.0~8.4.1 8.0.1~8.0.5	A100-SXM4-40GB, A100-SXM4-80GB, A800-SXM4-80GB, Tesla-P40, Tesla-V100-SXM2-16GB, Tesla-V100-SXM2-32GB, TITAN-X	11.0.3