ALIBABA CLOUD

阿里云

GPU云服务器

最佳实践

文档版本：20201105

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.深度学习

1.1. 在GPU实例上部署NGC环境

本章节以搭建TensorFlow深度学习框架为例介绍如何在GPU实例上部署NGC环境。

前提条件

在开始搭建TensorFlow深度学习框架之前，您必须先完成以下工作：

- 注册阿里云账号，并完成实名认证。具体步骤，请参见注册阿里云账号和实名认证 。
- 登录NGC网站，注册NGC账号。
- 登录NGC网站，获取NGC API key并保存到本地。登录NGC容器环境时需要验证您的NGC API Key。

背景信息

NGC（NVIDIA GPU CLOUD）是NVIDIA开发的一套深度学习生态系统，可以使开发者免费访问深度学习软件堆栈，建立适合深度学习的开发环境。

目前NGC在阿里云gn5实例作了全面部署，并且在镜像市场提供了针对NVIDIA Pascal GPU优化的NGC容器镜像。通过部署镜像市场的NGC容器镜像，开发者能简单快速地部署NGC容器环境，即时访问优化后的深度学习框架，大大缩减产品开发以及业务部署的时间，实现开发环境的预安装；同时支持调优后的算法框架，并且保持持续更新。

NGC网站提供了目前主流深度学习框架不同版本的镜像（例如Caffe、Caffe2、CNTK、MxNet、TensorFlow、Theano、Torch），您可以选择需要的镜像部署环境。

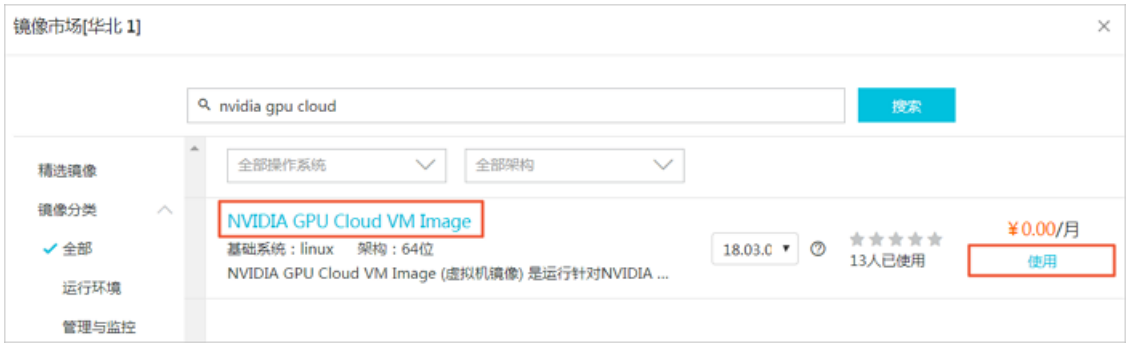
支持部署NGC环境的实例规格族包括：

- gn4、gn5、gn5i、gn6v、gn6i、gn6e
- ebmgn5i、ebmgn6i、ebmgn6v、ebmgn6e

下面以gn5实例为例，为您演示创建GPU实例和部署NGC环境的步骤。

操作步骤

1. 创建一台gn5实例。具体操作，请参见使用向导创建实例。 在配置参数时，您需要注意以下几点：
   - **地域**：只能选择华北1（青岛）、华北2（北京）、华北3（张家口）、华北5（呼和浩特）、华东1（杭州）、华东2（上海）、华南1（深圳）。
   - **实例**：选择gn5实例规格。
   - **镜像**：单击**镜像市场**，在弹出的对话框中，找到**NVIDIA GPU Cloud VM Image**，然后单击**使用**。



   - **公网带宽**：选择**分配公网IP地址**。

> **说明** 如果这里没有分配公网IP地址，则在实例创建成功后，需要绑定EIP地址。

- **安全组**：选择一个安全组。安全组里必须开放TCP 22端口。如果您的实例需要支持HTTPS或DIGIT 6服务，必须开放TCP 443（用于HTTPS）或TCP 5000（用于DIGITS 6）端口。

ECS实例创建成功后，请登录ECS管理控制台，记录实例的公网IP地址。

2. 连接ECS实例。 根据创建实例时选择的登录凭证选择以下任一方式连接ECS实例：
   - 使用密码验证连接ECS实例
   - 使用SSH密钥对验证连接ECS实例
3. 按界面提示输入NGC官网获取的NGC API Key后按回车键，即可登录NGC容器环境。
4. 运行 `nvidia-smi` 命令。 您能查看当前GPU的信息，包括GPU型号、驱动版本等，如下图所示。
5. 按以下步骤搭建TensorFlow深度学习框架。
   i. 登录NGC网站，在TensorFlow镜像页面，获取 `docker pull` 命令。
   ii. 下载TensorFlow镜像。

   ```
   docker pull nvcr.io/nvidia/tensorflow:18.03-py3
   ```

   iii. 查看下载的镜像。

   ```
   docker image ls
   ```

   iv. 运行容器，完成TensorFlow开发环境的部署。

   ```
   nvidia-docker run --rm -it nvcr.io/nvidia/tensorflow:18.03-py3
   ```

6. 选择以下任一种方式测试TensorFlow。
   - 简单测试TensorFlow。

   ```
   $python
   ```

   ```
   >>> import tensorflow as tf
   >>> hello = tf.constant('Hello, TensorFlow!')
   >>> sess = tf.Session()
   >>> sess.run(hello)
   ```

   如果TensorFlow正确加载了GPU设备，返回结果如下图所示。
   - 下载TensorFlow模型并测试TensorFlow。

   ```
   git clone https://github.com/tensorflow/models.git
   cd models/tutorials/image/alexnet
   python alexnet_benchmark.py --batch_size 128 --num_batches 100
   ```

运行状态如下图所示。

7. 保存TensorFlow镜像的修改。否则，下次登录时配置会丢失。

1.2. GPU AI模型训练最佳实践

本方案适用于AI图片训练场景，使用CPFS/NAS作为共享存储，利用容器服务Kubernetes版管理GPU云服务器集群进行图片AI训练。

实现的方式

- 搭建AI图片训练基础环境。
- 使用CPFS存储训练数据。
- 使用飞天AI加速训练服务加速训练。
- 使用Arena一键提交作业。

部署架构图

选用的产品

- GPU服务器

  GPU云服务器是基于GPU应用的计算服务，多适用于AI深度学习、视频处理、科学计算、图形可视化等应用场景。

  更多关于GPU服务器的介绍，请参见GPU服务器产品详情页。

- 文件存储NAS

  阿里云文件存储NAS是一个可共享访问、弹性扩展、高可靠、高性能的分布式文件系统。兼容POSIX文件接口，可支持数千台计算节点共享访问，可以挂载到弹性计算ECS、神龙裸金属、容器服务ACK、弹性容器ECI、批量计算BCS、高性能计算EHPC、AI训练PAI等计算业务上提供高性能的共享存储，用户无需修改应用程序，即可无缝迁移业务系统上云。

  更多关于文件存储NAS的介绍，请参见文件存储NAS产品详情页。

- 文件存储CPFS

  文件存储CPFS（Cloud Parallel File Storage），是阿里云完全托管、可扩展的并行文件存储系统，针对高性能计算场景的性能要求进行了深度优化，提供对数据毫秒级的访问和高聚合IO、高IOPS的数据读写请求，可以用于AI深度训练、自动驾驶、基因计算、EDA仿真、石油勘探、气象分析、机器学习、大数据分析以及影视渲染等业务场景中。

  更多关于文件存储CPFS的介绍，请参见文件存储CPFS详情页。

- 容器服务 ACK

  容器服务Kubernetes版（ACK）提供高性能可伸缩的容器应用管理能力，支持企业级容器化应用的全生命周期管理。整合阿里云虚拟化、存储、网络和安全能力，打造云端最佳容器化应用运行环境。

  更多关于容器服务ACK的介绍，请参见容器服务 ACK产品详情页。

详细信息

点击查看最佳实践详情

更多最佳实践

点击查看更多阿里云最佳实践

2.机器学习

2.1. 在GPU实例上使用RAPIDS加速机器学习任务

本文介绍了如何在GPU实例上基于NGC环境使用RAPIDS加速库，加速数据科学和机器学习任务，提高计算资源的使用效率。

前提条件

- 使用本教程进行操作前，请确保您已经注册了阿里云账号并完成实名认证。详情请参见账号注册和个人实名认证 。
- 获取NGC API Key。
  i. 在NGC注册页面注册NGC账号。
  ii. 登录NGC网站。
  iii. 前往**CONFIGURATION**页面，单击**Get API Key**。
  iv. 单击**Generate API Key**。
  v. 在**Generate a New API Key**页面中，单击**Confirm**。

  > **说明** 新的NGC API Key会覆盖旧的NGC API Key。如果您已持有NGC API Key，请确保不再需要旧的NGC API Key。

  vi. 复制API Key并保存到本地。

背景信息

RAPIDS，全称Real-time Acceleration Platform for Integrated Data Science，是NVIDIA针对数据科学和机器学习推出的GPU加速库。更多RAPIDS信息请参见官方网站。

NGC，全称NVIDIA GPU CLOUD，是NVIDIA推出的一套深度学习生态系统，供开发者免费访问深度学习和机器学习软件堆栈，快速搭建相应的开发环境。NGC网站提供了RAPIDS的Docker镜像，预装了相关的开发环境。

JupyterLab是一套交互式的开发环境，帮助您高效地浏览、编辑和执行服务器上的代码文件。

Dask是一款轻量级大数据框架，可以提升并行计算效率。

本文提供了一套基于NVIDIA的RAPIDS Demo代码及数据集修改的示例代码，演示了在GPU实例上使用RAPIDS加速一个从ETL到ML Training端到端任务的过程。其中，ETL时使用RAPIDS的cuDF，ML Training时使用XGBoost。本文示例代码基于轻量级大数据框架Dask运行，为一套单机运行的代码。

> **说明** NVIDIA官方RAPIDS Demo代码请参见Mortgage Demo。

RAPIDS预装镜像已经发布到阿里云镜像市场，创建GPU实例时，您可以在镜像市场中搜索*NVIDIA RAPIDS*并使用RAPIDS预装镜像。

> **说明** 该RAPIDS预装镜像使用Ubuntu 16.04 64-bit操作系统。

操作步骤

如果您创建GPU实例时使用了RAPIDS预装镜像，只需运行RAPIDS Demo，从启动JupyterLab服务开始操作即可。详情请参见启动JupyterLab服务。

如果您创建GPU实例时没有使用RAPIDS预装镜像，按照以下步骤使用RAPIDS加速机器学习任务：

1. 步骤一：获取RAPIDS镜像下载命令
2. 步骤二：部署RAPIDS环境
3. 步骤三：运行RAPIDS Demo

步骤一：获取RAPIDS镜像下载命令

完成以下操作，获取RAPIDS镜像下载命令：

1. 登录NGC网站。
2. 打开**MACHINE LEARNING**页面，单击**RAPIDS**镜像。
3. 获取docker pull命令。本文示例代码基于RAPIDS 0.8版本镜像编写，因此在运行本示例代码时，使用Tag为0.8版本的镜像。实际操作时，请选择您匹配的版本。
   i. 单击**Tags**页签。
   ii. 找到并复制Tag信息。本示例中，选择 `0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6` 。
   iii. 返回页面顶部，复制**Pull Command**中的命令到文本编辑器，将镜像版本替换为对应的Tag信息，并保存。本示例中，将 `cuda9.2-runtime-ubuntu16.04` 替换为 `0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6` 。

   保存的docker pull命令用于在步骤三中下载RAPIDS镜像。

步骤二：部署RAPIDS环境

完成以下操作，部署RAPIDS环境：

1. 创建一台GPU实例。详细步骤请参见使用向导创建实例。参数配置说明如下：
   - **实例**：RAPIDS仅适用于特定的GPU型号（采用NVIDIA Pascal及以上架构），因此您需要选择GPU型号符合要求的实例规格，目前有gn6i、gn6v、gn5和gn5i，详细的GPU型号请参见实例规格族。建议您选择显存更大的gn6i、gn6v或gn5实例。本示例中，选用了显存为16 GB的GPU实例。
   - **镜像**：在镜像市场中搜索并使用 `NVIDIA GPU Cloud VM Image` 。
   - **公网带宽**：选择**分配公网IPv4地址**或者在实例创建成功后绑定EIP地址。具体操作，请参见弹性公网IP文档中的绑定ECS实例。
   - **安全组**：选择的安全组需要开放以下端口：
     - TCP 22 端口，用于SSH登录
     - TCP 8888端口，用于支持访问JupyterLab服务
     - TCP 8787端口、TCP 8786端口，用于支持访问Dask服务
2. 连接GPU实例。连接方式请参见连接方式介绍。
3. 输入NGC API Key后按回车键，登录NGC容器环境。

4. （可选）运行nvidia-smi查看GPU型号、GPU驱动版本等GPU信息。 建议您了解GPU信息，预判规避潜在问题。例如，如果NGC的驱动版本太低，新Docker镜像版本可能会不支持。
5. 运行在步骤二中获取的docker pull命令下载RAPIDS镜像。

```
docker pull nvcr.io/nvidia/rapidsai/rapidsai:0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6
```

6. （可选）查看下载的镜像。

   建议您查看Docker镜像信息，确保下载了正确的镜像。

```
docker images
```

7. 运行容器部署RAPIDS环境。

```
docker run --runtime=nvidia \
    --rm -it \
    -p 8888:8888 \
    -p 8787:8787 \
    -p 8786:8786 \
    nvcr.io/nvidia/rapidsai/rapidsai:0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6
```

步骤三：运行RAPIDS Demo

完成以下操作，运行RAPIDS Demo：

1. 在GPU实例上下载数据集和Demo文件。

```
# Get apt source address and download demos.
source_address=$(curl http://100.100.100.200/latest/meta-data/source-address|head -n 1)
source_address="${source_address}/opsx/ecs/linux/binary/machine_learning/"
cd /rapids
wget $source_address/rapids_notebooks_v0.8.tar.gz
tar -xzvf rapids_notebooks_v0.8.tar.gz
cd /rapids/rapids_notebooks_v0.8/xgboost
wget $source_address/data/mortgage/mortgage_2000_1gb.tgz
```

2. 在GPU实例上启动JupyterLab服务。推荐直接使用命令启动。

```
# Run the following command to start JupyterLab and set the password.
cd /rapids/rapids_notebooks_v0.8/xgboost
jupyter-lab --allow-root --ip=0.0.0.0 --no-browser --NotebookApp.token='YOUR PASSWORD'
# Exit JupyterLab.
sh ../utils/stop-jupyter.sh
```

   - 除使用命令外，您也可以执行脚本 `sh ../utils/start-jupyter.sh` 启动jupyter-lab，此时无法设置登录密码。
   - 您也可以连续按两次 `Ctrl+C` 退出JupyterLab服务。

3. 打开浏览器，在地址栏输入 `http://您的GPU实例IP地址:8888` 远程访问JupyterLab。

   > **说明** 推荐使用Chrome浏览器。

   如果您在启动JupyterLab服务时设置了登录密码，会跳转到密码输入界面。

4. 运行NoteBook代码。该案例是一个抵押贷款回归的任务，详细信息请参见代码执行过程。登录成功后，可以看到NoteBook代码的代码包括以下内容：
   - *xgboost_E2E.ipynb*文件： XGBoost Demo文件。双击文件可以查看文件详情，单击下图中的执行按钮可以逐步执行代码，每次执行一个Cell。
   - *mortgage_2000_1gb.tgz*文件： 2000年的抵押贷款回归训练数据（1G分割的perf文件夹下的文件不会大于1G，使用1G分割的数据可以更有效的利用GPU显存）。

代码执行过程

该案例基于XGBoost演示了数据预处理到训练的端到端的过程，主要分为三个阶段：

- ETL（Extract-Transform-Load）：主要在GPU实例上进行。将业务系统的数据经过抽取、清洗转换之后加载到数据仓库。
- Data Conversion：在GPU实例上进行。将在ETL阶段处理过的数据转换为用于XGBoost训练的DMatrix格式。
- ML-Training：默认在GPU实例上进行。使用XGBoost训练梯度提升决策树 。

NoteBook代码的执行过程如下：

1. 准备数据集。

   本案例的Shell脚本会默认下载2000年的抵押贷款回归训练数据（*mortgage_2000_1gb.tgz*）。

   如果您想获取更多数据用于XGBoost模型训练，可以设定参数download_url指定下载路径，具体下载地址请参见Mortgage Data。

   示例效果如下：

2. 设定相关参数。

参数名称	说明
start_year	指定选择训练数据的起始时间，ETL时会处理start_year到end_year之间的数据。
end_year	指定选择训练数据的结束时间，ETL时会处理start_year到end_year之间的数据。
train_with_gpu	是否使用GPU进行XGBoost模型训练，默认为*True*。
gpu_count	指定启动worker的数量，默认为*1*。您可以按需要设定参数值，但不能超出GPU实例的GPU数量。
part_count	指定用于模型训练的performance文件的数量，默认为*2 * gpu_count*。如果参数值过大，在Data Conversion阶段会报错超出GPU内存限制，错误信息会在NoteBook后台输出。

   示例效果如下：

3. 启动Dask服务。

   代码会启动Dask Scheduler，并根据gpu_count参数启动worker用于ETL和模型训练。启动Dask服务后，您也可以通过Dask Dashboard直观地监控任务，打开方法请参见Dask Dashboard。

   示例效果如下：

4. 启动ETL。

   ETL阶段会进行到表关联、分组、聚合、切片等操作，数据格式采用cuDF库的DataFrame格式（类似于pandas的DataFrame格式）。

   示例效果如下：

5. 启动Data Conversion。

   将DataFrame格式的数据转换为用于XGBoost训练的DMatrix格式，每个worker处理一个DMatrix对象。

   示例效果如下：

6. 启动ML Training。

   使用dask-xgboost启动模型训练，dask-xgboost负责多个dask worker间的通信协同工作，底层仍然调用xgboost执行模型训练。

   示例效果如下：

Dask Dashboard

Dask Dashboard支持任务进度跟踪、任务性能问题识别和故障调试。

Dask服务启动后，在浏览器地址栏中访问 `http://您的GPU实例IP地址:8787/status` 即可进入Dashboard主界面。

相关函数

函数功能	函数名称
下载文件	def download_file_from_url(url, filename):
解压文件	def decompress_file(filename, path):
获取当前机器的GPU个数	def get_gpu_nums():
管理GPU内存	• def initialize_rmm_pool(): • def initialize_rmm_no_pool(): • def run_dask_task(func, **kwargs):
提交DASK任务	• def process_quarter_gpu(year=2000, quarter=1, perf_file=""): • def run_gpu_workflow(quarter=1, year=2000, perf_file="", **kwargs):

函数功能	函数名称
使用cuDF从CSV中加载数据	• def gpu_load_performance_csv(performance_path, **kwargs): • def gpu_load_acquisition_csv(acquisition_path, **kwargs): • def gpu_load_names(**kwargs):
处理和提取训练数据的特征	• def null_workaround(df, **kwargs): • def create_ever_features(gdf, **kwargs): • def join_ever_delinq_features(everdf_tmp, delinq_merge, **kwargs): • def create_joined_df(gdf, everdf, **kwargs): • def create_12_mon_features(joined_df, **kwargs): • def combine_joined_12_mon(joined_df, testdf, **kwargs): • def final_performance_delinquency(gdf, joined_df, **kwargs): • def join_perf_acq_gdfs(perf, acq, **kwargs): • def last_mile_cleaning(df, **kwargs):

相关文档

- 在GPU实例上使用RAPIDS加速图像搜索任务

2.2. RAPIDS加速机器学习最佳实践

本方案适用于使用RAPIDS加速库和GPU云服务器来对机器学习任务或者数据科学任务进行加速的场景。相比CPU，利用GPU和RAPIDS在某些场景下可以取得非常明显的加速效果。

解决的问题

- 搭建RAPIDS加速机器学习环境。
- 使用容器服务Kubernetes版部署RAPIDS环境。
- 使用NAS存储计算数据。

部署架构图

选用的产品

- GPU服务器

  GPU云服务器是基于GPU应用的计算服务，多适用于AI深度学习、视频处理、科学计算、图形可视化等应用场景。

  更多关于GPU服务器的介绍，请参见GPU服务器产品详情页。

- 文件存储NAS

阿里云文件存储NAS是一个可共享访问、弹性扩展、高可靠、高性能的分布式文件系统。兼容POSIX文件接口，可支持数千台计算节点共享访问，可以挂载到弹性计算ECS、神龙裸金属、容器服务ACK、弹性容器ECI、批量计算BCS、高性能计算EHPC、AI训练PAI等计算业务上提供高性能的共享存储，用户无需修改应用程序，即可无缝迁移业务系统上云。

更多关于文件存储NAS的介绍，请参见文件存储NAS产品详情页。

- 容器服务 ACK

  容器服务Kubernetes版（ACK）提供高性能可伸缩的容器应用管理能力，支持企业级容器化应用的全生命周期管理。整合阿里云虚拟化、存储、网络和安全能力，打造云端最佳容器化应用运行环境。

  更多关于容器服务ACK的介绍，请参见容器服务 ACK产品详情页。

详细信息

点击查看最佳实践详情

更多最佳实践

点击查看更多阿里云最佳实践

2.3. 在GPU实例上使用RAPIDS加速图像搜索任务

本文以使用RAPIDS加速图像搜索任务为例，介绍如何在预装镜像的GPU实例上使用RAPIDS加速库。

前提条件

使用本教程进行操作前，请确保您已经注册了阿里云账号。如还未注册，请先完成账号注册。

背景信息

RAPIDS，全称Real-time Acceleration Platform for Integrated Data Science，是NVIDIA针对数据科学和机器学习推出的GPU加速库。更多RAPIDS信息请参见官方网站。

基于图像识别和搜索，图像搜索任务可以实现以图搜图，在不同行业应用和业务场景中帮助您搜索相同或相似的图片。

图像搜索任务背后的两项主要技术是特征提取及向量化、向量索引和检索。本文案例中，使用开源框架TensorFlow和Keras配置生产环境，然后使用ResNet50卷积神经网络完成图像的特征提取及向量化，最后使用RAPIDS cuML库的KNN算法实现BF方式的向量索引和检索。

> **说明** BF（Brute Force）检索方法是一种百分百准确的方法，对距离衡量算法不敏感，适用于所有的距离算法。

本文案例在阿里云gn6v（NVIDIA Tesla V100）实例上执行。执行案例后，对比了GPU加速的RAPIDS cuml KNN与CPU实现的scikit-learn KNN的性能，可以看到GPU加速的KNN向量检索速度为CPU的近600倍。

本文案例为单机单卡的版本，即一台GPU实例搭载一块GPU卡。

操作步骤

执行以下操作完成一次图像搜索任务：

1. 创建GPU实例

2. 启动和登录JupyterLab
3. 执行图像搜索案例

步骤一：创建GPU实例

具体步骤请参见使用向导创建实例。

- **实例**：RAPIDS仅适用于特定的GPU型号（采用NVIDIA Pascal及以上架构），因此您需要选择GPU型号符合要求的实例规格，目前有gn6i、gn6v、gn5和gn5i。本文案例中，选用了ecs.gn6v-c8g1.2xlarge实例规格。
- **镜像**：在镜像市场中使用关键字*RAPIDS*，搜索并使用预装了RAPIDS加速库的镜像。
- **安全组**：选择的安全组需要开放TCP 8888端口，用于支持访问JupyterLab服务。

步骤二：启动和登录JupyterLab

1. 连接GPU实例，运行以下命令启动JupyterLab服务。

   > **说明** 连接GPU实例的步骤请参见连接方式概述ECS远程连接操作指南。

   ```
   # Go to the notebooks directory.
   cd /rapids

   # Run the following command to start JupyterLab and set the logon password:
   jupyter-lab --allow-root --ip=0.0.0.0 --no-browser --NotebookApp.token='your logon password'

   # Exit jupyterlab: press Ctrl+C twice.
   ```

2. 在您的本地机器上打开浏览器。输入 `http://(IP address of your GPU instance):8888` 远程访问JupyterLab。

   > **说明** 推荐使用Chrome浏览器。

3. 输入启动命令中设置的密码，然后单击Log in。

步骤三：执行图像搜索案例

1. 进入案例所在目录*rapids_notebooks_v0.7/cuml*。
2. 双击*cuml_knn.ipynb*文件。
3. 单击。

   > **说明** 单击一次执行一个cell，请单击至案例执行结束，详细说明请参见案例执行过程。

案例执行过程

图像搜索案例的执行过程分为三个步骤：处理数据集、提取图片特征和搜索相似图片。本文案例结果中对比了GPU加速的RAPIDS cuml KNN与CPU实现的scikit-learn KNN的性能。

1. 处理数据集。
   i. 下载和解压数据集。 本文案例中使用了STL-10数据集，该数据集中包含10万张未打标的图片，图片的尺寸均为：96 x 96 x 3。您可以使用其他数据集，为便于提取图片特征，请确保数据集中图片的尺寸相同。

      本文案例提供了 `download_and_extract(data_dir)` 方法供您下载和解压STL-10数据集。RAPIDS镜像中已经将数据集下载到*./data*目录，您可以执行 `download_and_extract()` 方法直接解压数据集。

   ii. 读取图片。 从数据集解压出的数据为二进制格式，执行 `read_all_images(path_to_data)` 方法加载数据并转换为NHWC（batch, height, width, channels）格式，以便用Tensorflow提取图片特征。

   iii. 展示图片。 执行 `show_image(image)` 方法随机展示一张数据集中的图片。

   iv. 分割数据集。 按照9:1的比例把数据集分为两部分，分别用于创建图片索引库和搜索图片。

2. 提取图片特征。 使用开源框架Tensorflow和Keras提取图片特征，其中模型为基于ImageNet数据集的ResNet50（notop）预训练模型。
   i. 设定Tensorflow参数。 Tensorflow默认使用所有GPU显存，我们需要留出部分GPU显存供cuML使用。您可以选择一种方法设置GPU显存参数：
      - 方法1：依据运行需求进行显存分配。

        ```
        config.gpu_options.allow_growth = True
        ```

      - 方法2：设定可以使用的GPU显存比例。本案例中使用方法2，并且GPU显存比例默认设置为0.3，即Tensorflow可以使用整块GPU显存的30%，您可以依据应用场景修改比例。

        ```
        config.gpu_options.per_process_gpu_memory_fraction = 0.3
        ```

ii. 下载ResNet50（notop）预训练模型。 连接公网下载模型（大小约91M），下载完成后默认保存到*/root/.keras/models/*目录。

参数名称	说明
weights	取值范围： ■ *None*：随机初始化权重值。 ■ *imagenet*：权重值的初始值设置为通过ImageNet预训练过的模型的权重值。 本案例中设置为*imagenet*。
include_top	取值范围： ■ True：包含整个ResNet50网络结构的最后一个全链接层。 ■ False：不包含整个ResNet50网络结构的最后一个全链接层。 本案例中，使用神经网络模型ResNet50的主要目的是提取图片特征而非分类图片，因此设置为*False*。
input_shape	可选参数，用于设置图片的输入shape，仅在include_top设置为*False*时生效。您必须为图片设置3个inputs channels，且宽和高不应低于32。此处设为*(96, 96, 3)*。
pooling	在include_top设置为*False*时，您需要设置池化层模式，取值范围： ■ *None*：输出为4D tensor。 ■ *avg*：平均池化，输出为2D tensor。 ■ *max*：最大池化，输出为2D tensor。 本案例中设置为*max*。

您可以执行 `model.summary()` 方法查看模型的网络结构。

iii. 提取图片特征。 对分割得到的两个图片数据集执行 `model.predict()` 方法提取图片特征。

3. 搜索相似图片。

i. 使用cuml KNN搜索相似图片。 通过 `k=3` 设置K值为3，即查找最相似的3张图片，您可以依据使用场景自定义K值。

其中， `knn_cuml.fit()` 方法为创建索引阶段， `knn_cuml.kneighbors()` 为搜索近邻阶段。

KNN向量检索耗时791 ms。

ii. 使用scikit-learn KNN搜索相似图片。 通过 `n_neighbors=3` 设置K值为3，通过 `n_jobs=-1` 设置使用所有CPU进行近邻搜索。

> **说明** ecs.gn6v-c8g1.2xlarge的配置为8 vCPU。

KNN向量检索耗时7分34秒。

iii. 对比cuml KNN和scikit-learn KNN的搜索结果。 对比两种方式的KNN向量检索速度，使用GPU加速的cuml KNN耗时791 ms，使用CPU的scikit-learn KNN耗时7min 34s。前者为后者的近600倍。

验证两种方式的输出结果是否相同，输出结果为两个数组：

- distance：最小的K个距离值。本案例中搜索了10000张图片，K值为3，因此 `distance.shape=(10000,3)` 。
- indices：对应的图片索引。 `indices.shape=(10000, 3)` 。

由于本案例所用数据集中存在重复图片，容易出现图片相同但索引不同的情况，因此使用distances，不使用indices对比结果。考虑到计算误差，如果两种方法得出的10000张图片中的3个最小距离值误差都小于1，则认为结果相同。

图片搜索结果

本案例从1万张搜索图片中随机选择5张图片并搜索相似图片，最终展示出5行4列图片。

第一列为搜索图片，第二列至第四列为图片索引库中的相似图片，且相似性依次递减。每张相似图片的标题为计算的距离，数值越大相似性越低。

2.4. RAPIDS加速图像搜索最佳实践

本方案适用于使用RAPIDS加速平台和GPU云服务器来对图像搜索任务进行加速的场景。相比CPU，利用GPU+RAPIDS在图像搜索场景下可以取得非常明显的加速效果。

解决的问题

- 搭建RAPIDS加速图像搜索环境。
- 使用容器服务Kubernetes版部署图像搜索环境。
- 使用NAS存储计算数据。

部署架构图

选用的产品

- GPU服务器

  GPU云服务器是基于GPU应用的计算服务，多适用于AI深度学习、视频处理、科学计算、图形可视化等应用场景。

  更多关于GPU服务器的介绍，请参见GPU服务器产品详情页。

- 文件存储NAS

  阿里云文件存储NAS是一个可共享访问、弹性扩展、高可靠、高性能的分布式文件系统。兼容POSIX文件接口，可支持数千台计算节点共享访问，可以挂载到弹性计算ECS、神龙裸金属、容器服务ACK、弹性容器ECI、批量计算BCS、高性能计算EHPC、AI训练PAI等计算业务上提供高性能的共享存储，用户无需修改应用程序，即可无缝迁移业务系统上云。

  更多关于文件存储NAS的介绍，请参见文件存储NAS产品详情页。

- 容器服务 ACK

容器服务Kubernetes版（ACK）提供高性能可伸缩的容器应用管理能力，支持企业级容器化应用的全生命周期管理。整合阿里云虚拟化、存储、网络和安全能力，打造云端最佳容器化应用运行环境。

更多关于容器服务ACK的介绍，请参见容器服务 ACK产品详情页。

详细信息

点击查看最佳实践详情

更多最佳实践

点击查看更多阿里云最佳实践

3.使用cGPU服务隔离GPU资源

cGPU服务可以隔离GPU资源，实现多个容器共用一张显卡。本章节介绍如何在GPU实例上安装和使用cGPU服务。

前提条件

安装cGPU服务前，请完成以下准备工作：

- 提交工单获取cGPU安装包下载链接。
- 确保GPU实例满足以下要求：
  - GPU实例规格为gn6i、gn6v、gn6e、gn5i、gn5、ebmgn6i或ebmgn6e。
  - GPU实例操作系统为CentOS 7.6、CentOS 7.7、Ubuntu 16.04、Ubuntu 18.04或Aliyun Linux。
  - GPU实例已安装418.87.01或更高版本的NVIDIA驱动。
  - GPU实例已安装19.03.5或更高版本的Docker。

背景信息

为了提高GPU硬件资源的利用率，需要在单张显卡上运行多个容器，并在多个容器间隔离GPU应用。

阿里云cGPU服务通过自研的内核驱动为容器提供虚拟的GPU设备，在保证性能的前提下隔离显存和算力，为充分利用GPU硬件资源进行训练和推理提供有效保障。您可以通过命令方便地配置容器内的虚拟GPU设备。

使用cGPU服务具有以下优势：

- 适配开源标准的Kubernetes和NVIDIA Docker方案。
- 无需重编译AI应用，无需替换CUDA库，升级CUDA、cuDNN的版本后无需重新配置。
- 支持同时隔离显存和算力。

安装cGPU服务

1. 下载并解压cGPU安装包。
2. 查看安装包文件。

   > **说明** 安装包版本不同时，安装包中包含的文件可能不同。

   ```
   # cd cgpu
   # ls
   cgpu-container-wrapper cgpu-km.c cgpu.o cgpu-procfs.c install.sh Makefile os-interface.c README u
   ninstall.sh upgrade.sh version.h
   ```

3. 安装cGPU服务。

   ```
   sh install.sh
   ```

4. 验证安装结果。

   ```
   # lsmod | grep cgpu
   cgpu_km         71355 0
   ```

   显示cgpu服务的状态信息，表示已成功安装cGPU服务。

运行cGPU服务

影响cGPU服务的环境变量如下表所示，您可以在创建容器时指定环境变量的值，控制容器可以通过cGPU服务获得的算力。

环境变量名称	取值类型	说明	示例
CGPU_DISABLE	Boolean	是否启用cGPU服务，取值范围： • *false*：启用cGPU服务。 • *true*：禁用cGPU服务，使用默认的NVIDIA容器服务。	无
ALIYUN_COM_GPU_MEM_DEV	Integer	设置GPU实例上每张显卡的总显存大小，和实例规格有关。 **说明** 显存大小按GiB取整数。	一台GPU实例规格为ecs.gn6i-c4g1.xlarge，配备1张NVIDIA® Tesla® T4显卡。在GPU实例上执行**nvidia-smi**查看总显存大小为15109 MiB，取整数为15 GiB。
ALIYUN_COM_GPU_MEM_CONTAINER	Integer	设置容器内可见的显存大小，和ALIYUN_COM_GPU_MEM_DEV结合使用。如果不指定本参数或指定为0，则不使用cGPU服务，使用默认的NVIDIA容器服务。	在一张总显存大小为15 GiB的显卡上，设置环境变量 `ALIYUN_COM_GPU_MEM_DEV=15` 和 `ALIYUN_COM_GPU_MEM_CONTAINER=1`，效果是为容器分配1 GiB的显存。

环境变量名称	取值类型	说明	示例
ALIYUN_COM_GPU_VISIBLE_DEVICES	Integer或uuid	指定容器内可见的GPU显卡。	在一台有4张显卡的GPU实例上，执行nvidia-smi -L查看GPU显卡设备号和UUID。返回示例如下所示： GPU 0: Tesla T4 (UUID: GPU-b084ae33-e244-0959-cd97-83******) GPU 1: Tesla T4 (UUID: GPU-3eb465ad-407c-4a23-0c5f-bb******) GPU 2: Tesla T4 (UUID: GPU-2fce61ea-2424-27ec-a2f1-8b******) GPU 3: Tesla T4 (UUID: GPU-22401369-db12-c6ce-fc48-d7******) 然后，设置以下环境变量： • ALIYUN_COM_GPU_VISIBLE_DEVICES=0,1，效果是为容器分配第1和第2张显卡。 • ALIYUN_COM_GPU_VISIBLE_DEVICES=GPU-b084ae33-e244-0959-cd97-83****,GPU-3eb465ad-407c-4a23-0c5f-bb****,GPU-2fce61ea-2424-27ec-a2f1-8b******，效果是为容器分配3张指定UUID的显卡。
ALIYUN_COM_GPU_SCHD_WEIGHT	Integer	设置容器的算力权重，取值范围：1~min(max_inst, 16)。	无

以ecs.gn6i-c4g1.xlarge为例演示2个容器共用1张显卡。

1. 执行以下命令创建容器并设置容器内可见的显存。

```
docker run -d -t --gpus all --privileged --shm-size=1g --ulimit memlock=-1 --ulimit stack=67108864 --nam
e gpu_test1 -v /mnt:/mnt -e ALIYUN_COM_GPU_MEM_CONTAINER=6 -e ALIYUN_COM_GPU_MEM_DEV=15
nvcr.io/nvidia/tensorflow:19.10-py3
docker run -d -t --gpus all --privileged --shm-size=1g --ulimit memlock=-1 --ulimit stack=67108864 --nam
e gpu_test2 -v /mnt:/mnt -e ALIYUN_COM_GPU_MEM_CONTAINER=8 -e ALIYUN_COM_GPU_MEM_DEV=15
nvcr.io/nvidia/tensorflow:19.10-py3
```

> **说明** 该命令以使用TensorFlow镜像nvcr.io/nvidia/tensorflow:19.10-py3为例，请根据实际情况更换为您自己的容器镜像。使用TensorFlow镜像搭建TensorFlow深度学习框架的操作，请参见在GPU实例上部署NGC环境。

本示例中，通过设置环境变量ALIYUN_COM_GPU_MEM_CONTAINER和ALIYUN_COM_GPU_MEM_DEV指定显卡的总显存和容器内可见的显存。命令执行结果是创建了2个容器：

- gpu_test1：分配6 GiB显存。
- gpu_test2：分配8 GiB显存。

2. 进入容器。本示例中，以gpu_test1为例。

```
docker exec -it gpu_test1 bash
```

3. 执行以下命令查看显存等GPU信息。

```
nvidia-smi
```

容器gpu_test1中可见的显存为6043 MiB，如下图所示。

查看procfs节点

cGPU服务运行时会在*/proc/cgpu_km*下生成并自动管理多个procfs节点，您可以通过procfs节点查看和配置cGPU服务相关的信息。下面介绍各procfs节点的用途。

1. 执行以下命令查看节点信息。

```
# ls /proc/cgpu_km/
0 default_memsize inst_ctl major version
```

节点信息说明如下表所示。

节点	读写类型	说明
0	读写	cGPU服务会针对GPU实例中的每张显卡生成一个的目录，并使用数字作为目录名称，例如0、1、2。本示例中只有一张显卡，对应的目录ID为0。
default_memsize	读写	如果没有设置ALIYUN_COM_GPU_MEM_CONTAINER参数，默认为新创建的容器分配的显存大小。
inst_ctl	读写	控制节点。

节点	读写类型	说明
major	只读	cGPU内核驱动的主设备号。
version	只读	cGPU的版本。

2. 执行以下命令查看显卡对应的目录内容。本示例中，以显卡0为例。

```
# ls /proc/cgpu_km/0
012b2edccd7a  0852a381c0cf  free_weight  max_inst  policy
```

目录内容说明如下表所示。

节点	读写类型	说明
容器对应的目录	读写	cGPU服务会针对运行在GPU实例中的每个容器生成一个的目录，并使用容器ID作为目录名称。您可以执行`docker ps`查看已创建的容器。
free_weigt	只读	用于查询和修改可用的权重。如果`free_weight=0`，新创建容器的权重值为0，该容器不能获取GPU算力，不能用于运行需要GPU算力的应用。
max_inst	读写	用于设置容器的最大数量，取值范围为1~16。
policy	读写	cGPU服务支持以下算力调度策略： ○ 0：平均调度。每个容器占用固定的时间片，时间片占比为`1/max_inst`。 ○ 1：抢占调度。每个容器占用尽量多的时间片，时间片占比为`1/当前容器数`。 ○ 2：权重抢占调度。当ALIYUN_COM_GPU_SCHD_WEIGHT的取值大于1时，自动使用权重抢占调度。 您可以通过修改policy的值实时调整调度策略。更多调度策略说明，请参见cGPU服务算力调度示例。

3. 执行以下命令查看容器对应的目录内容。本示例中，以012b2edccd7a容器为例。

```
# ls /proc/cgpu_km/0/012b2edccd7a
id  meminfo  memsize  weight
```

目录内容说明如下表所示。

节点	读写类型	说明
id	只读	容器的ID。

节点	读写类型	说明
memsize	读写	用于设置容器内的显存大小。cGPU服务会根据ALIYUN_COM_GPU_MEM_DEV参数自动设定此值。
meminfo	只读	包括容器内剩余显存容量、正在使用GPU的进程ID及其显存用量。输出如下所示： Free: 6730809344 PID: 19772 Mem: 200278016
weight	读写	用于设置容器获取显卡最大算力的权重，默认值为1。所有运行中的容器的权重之和必须小于等于max_inst。

了解procfs节点的用途后，您可以在GPU实例中执行命令进行切换调度策略、修改权重等操作，示例命令如下表所示。

命令	效果
echo 2 > /proc/cgpu_km/0/policy	将调度策略切换为权重抢占调度。
cat /proc/cgpu_km/0/free_weight	查看显卡上可用的权重。 如果 `free_weight=0` ，新创建容器的权重值为0，该容器不能获取GPU算力，不能用于运行需要GPU算力的应用。
cat /proc/cgpu_km/0/$dockerid/weight	查看指定容器的权重。
echo 4 > /proc/cgpu_km/0/$dockerid/weight	修改容器获取GPU算力的权重。

升级cGPU服务

1. 关闭所有运行中的容器。

```
docker stop (docker ps -a | awk '{ print $1}' | tail -n +2)
```

2. 执行upgrade.sh脚本升级cGPU服务至最新版本。

```
sh upgrade.sh
```

卸载cGPU服务

1. 关闭所有运行中的容器。

```
docker stop (docker ps -a | awk '{ print $1}' | tail -n +2)
```

2. 执行sh uninstall.sh脚本卸载cGPU服务。

```
sh uninstall.sh
```

cGPU服务算力调度示例

cGPU服务加载cgpu_km的模块时，会按照容器最大数量（max_inst）为每张显卡设置时间片（X ms），用于为容器分配GPU算力，本示例中以Slice 1、Slice 2...Slice N表示。使用不同调度策略时的调度示例如下所示。

- 平均调度

  在创建容器时，为容器分配时间片。cGPU服务会从Slice 1时间片开始调度，提交任务到物理GPU，并执行一个时间片（X ms）的时间，然后切换到下一个时间片。每个容器获得的算力相同，都为 `1/max_inst` 。如下图所示。

- 抢占调度

  在创建容器时，为容器分配时间片。cGPU服务会从Slice 1开始调度，但如果没有使用某个容器，或者容器内没有进程打开GPU设备，则跳过调度，切换到下一个时间片。

  示例如下：

  i. 只创建一个容器Docker 1，获得Slice 1时间片，在Docker 1中运行2个TensorFlow进程，此时Docker 1最大获得整个物理GPU的算力。

  ii. 再创建一个容器Docker 2，获得Slice 2时间片。如果Docker 2内没有进程打开GPU设备，调度时会跳过Docker 2的时间片Slice 2。

  iii. 当Docker 2内有进程打开GPU设备时，Slice 1和Slice 2都加入调度，Docker 1和Docker 2最大分别获得1/2物理GPU的算力。如下图所示。

- 权重抢占调度

  如果在创建容器时设置ALIYUN_COM_GPU_SCHD_WEIGHT大于1，则自动使用权重抢占调度。cGPU服务按照容器数量（max_inst）将物理GPU算力划分成max_inst份，但如果ALIYUN_COM_GPU_SCHD_WEIGHT大于1，cGPU服务会将数个时间片组合成一个更大的时间片分配给容器。

  设置示例如下：

  - Docker 1：ALIYUN_COM_GPU_SCHD_WEIGHT=m
  - Docker 2：ALIYUN_COM_GPU_SCHD_WEIGHT=n

  调度效果如下：

  - 如果只有Docker 1运行， Docker 1抢占整个物理GPU的算力。
  - 如果Docker 1和Docker 2同时运行，Docker 1和Docker 2获得的理论算力比例是m:n。和抢占调度不同的是，即使Docker 2中没有GPU进程也会占用n个时间片的时间。

    > **说明** m:n设置为2:1和8:4时的运行表现存在差别。在1秒内切换时间片的次数，前者是后者的4倍。

  权重抢占调度限制了容器使用GPU算力的理论最大值。但对算力很强的显卡（例如NVIDIA® V100显卡），如果显存使用的较少，在一个时间片内即可完成计算任务。此时如果m:n值设置为8:4，则剩余时间片内GPU算力会闲置，限制基本失效。因此建议根据显卡算力设置适当的权重值，例如：

- 如果使用NVIDIA® V100显卡，将m:n设置为2:1，尽量降低权重值，避免GPU算力闲置。
- 如果使用NVIDIA® Telsa® T4显卡，将m:n设置为8:4， 尽量增大权重值，保证分配足够的GPU算力。

4.FastGPU构建一键训练任务

FastGPU是一套阿里云推出的人工智能计算任务构建工具，提供便捷的接口和命令行，供您在阿里云IaaS资源上构建人工智能计算任务。本文以Ubuntu 18.04 64位为例介绍如何安装和使用FastGPU，并列出了FastGPU支持的运行时接口和命令行。

前提条件

客户端已安装Python 3.6或以上版本。

> **说明** 您的ECS实例和本地机器、阿里云Cloud Shell工具等均可以作为客户端安装FastGPU来构建人工智能计算任务。

背景信息

FastGPU作为衔接您的线下人工智能算法和线上阿里云海量GPU计算资源的关键一环，方便您将人工智能计算任务构建在阿里云的IaaS资源上。使用FastGPU构建人工智能计算任务时，您无需关心IaaS层的计算、存储、网络等资源部署操作，达到简单适配、一键部署、随处运行的效果。

FastGPU提供以下两套组件：

- 运行时组件ncluster：提供便捷的接口将线下的人工智能训练和推理脚本快速部署在阿里云的IaaS资源上，更多运行时组件使用说明请参见运行时说明。
- 命令行组件ecluster：提供便捷的命令行工具，用于管理阿里云上人工智能计算任务的运行状态和集群的生命周期，更多命令行组件使用说明请参见命令行说明。

FastGPU的组成模块如下图所示。

安装FastGPU

1. 在客户端下载FastGPU软件包。

   ```
   wget https://ali-perseus-release.oss-cn-huhehaote.aliyuncs.com/fastgpu/ncluster-1.0.8-py3-none-any.whl
   ```

2. 安装FastGPU。

   ```
   pip install ncluster-1.0.8-py3-none-any.whl
   ```

运行FastGPU demo

本步骤以在Cloud Shell中运行BERT finetune任务为例，展示如何使用FastGPU。demo中自动创建的实例规格为ecs.gn6v-c10g1.20xlarge（8卡V100机型），任务部署时间约2.5分钟，训练时长约11.5分钟，总共耗时约14分钟，训练精度达到0.88以上。

1. 打开Cloud Shell。本次测试时，Cloud Shell中使用Ubuntu 18.04 64，且默认已安装FastGPU，您可以直接开始准备项目文件并运行任务。
2. 准备项目文件。

   ```
   git clone https://github.com/aliyun/alibabacloud-aiacc-demo
   ```

3. 进入任务脚本目录。

```
cd alibabacloud-aiacc-demo/tensorflow/bert
```

4. 运行任务脚本。

```
python train_news_classifier.py
```

运行任务时需要自动创建实例等资源，会提示涉及计费，请按提示确认继续。

> 注意 您可以在任务完成后手动释放实例，避免任务完成后实例继续计费。

脚本运行成功后显示如下图所示。

5. 查看运行任务时自动创建的实例。

```
ecluster ls
```

6. 登录实例查看训练过程日志。

```
ecluster tmux task0.perseus-bert
```

显示以下结果时，表明已完成训练。

运行时说明

您可以通过ncluster的接口将人工智能训练和推理脚本快速部署到云上进行计算。ncluster的接口主要提供以下功能：

- 获取阿里云账号AccessKey、默认地域、默认可用区等信息。

```
export ALIYUN_ACCESS_KEY_ID=L****    # Your actual aliyun access key id
export ALIYUN_ACCESS_KEY_SECRET=v****  # Your actual aliyun access key secret
export ALIYUN_DEFAULT_REGION=cn-hangzhou  # The actual region the resource you want to use
export ALIYUN_DEFAULT_ZONE=cn-hangzhou-i  # The actual zone of the region you want to use
```

- ncluster是一套Python库，使用时需要在Python脚本中导入ncluster。

```
import ncluster
```

- 创建任务所需的资源或者复用已经存在的资源。

```
job = ncluster.make_job(name=args.name,
                        run_name=f"{args.name}-{args.machines}",
                        num_tasks=args.machines,
                        image_name=IMAGE_NAME,
                        instance_type=INSTANCE_TYPE)
```

ncluster.make_job的参数如下表所示。

参数名称	参数说明	参数示例
name	job的名称。	'perseus-bert'
run_name	运行时的环境名，一般设置为job名+实例数量。	f"perseus-bert-1"
num_tasks	需要创建实例的个数。	1 表示创建1台实例，名称为task0.perseus-bert，对应perseus-bert.tasks[0]。
image_name	实例使用的镜像，支持公共镜像和自定义镜像。	'ubuntu_18_04_64_20G_alibase_20190624.vhd'
instance_type	需要创建实例的实例规格。	'ecs.gn6v-c10g1.20xlarge'

- 运行任务。支持以job或task的形式运行任务，job为一组task。

  > **说明** job和task支持相同的API，调用job的API作用于所有的task，调用task的API只作用于指定的task。

  示例如下：

  - 调用job的API

    ```
    # 为job中所有实例打开perseus-bert文件夹
    job.run('cd perseus-bert')

    # 将当前目录中的perseus-bert文件夹上传到job中所有实例的/root目录下
    job.upload('perseus-bert')
    ```

  - 调用task的API

    ```
    # 为task0对应实例打开perseus-bert文件夹
    job.tasks[0].run('cd perseus-bert')

    # 将当前目录中的perseus-bert文件夹上传到task0对应实例的/root目录下
    job.tasks[0].upload('perseus-bert')
    ```

命令行说明

您可以使用ecluster命令方便地管理资源的生命周期、查看运行过程的日志等操作。ecluster支持的命令如下表所示。

命令	命令说明	命令示例

命令	命令说明	命令示例
export	获取阿里云账号的信息，在本地机器使用FastGPU时需要获取AccessKey、默认地域、默认可用区等信息。	• export ALIYUN_ACCESS_KEY_ID=L**** • export ALIYUN_ACCESS_KEY_SECRET=v**** • export ALIYUN_DEFAULT_REGION=cn-hangzhou • export ALIYUN_DEFAULT_ZONE=cn-hangzhou-i
ecluster [help,-h,--help]	查看所有ecluster命令。	ecluster --help
ecluster {command} --help	查看指定的ecluser命令。	ecluster ls --help
ecluster create --config create.cfg	基于配置文件创建实例。create.cfg文件定义实例的配置环境，运行命令前您需要先创建create.cfg文件，具体内容请参见表格下方的示例。	ecluster create --config create.cfg
ecluster create --name {instance_name} --machines {instance_num} ...	基于参数创建实例。	ecluster create --name task0.ncluster-v100 --machines 1
ecluster ls	列出已自动创建的实例。包括以下信息： • name：实例名称。 • hours_live：实例创建至今的时间，以小时为单位。 • instance_type：实例规格。 • public_ip：实例的公网IP。 • key/owner：密钥对和账号名。 • private_ip：实例的内网。 • instance_id：实例的id。	ecluster ls
ecluster ssh {instance_name}	登录指定的实例。	ecluster ssh task0.ncluster-v100
ecluster tmux {instance_name}	连接到运行中的任务，如果没有tmux会话，则使用ssh连接。	ecluster tmux task0.ncluster-v100
ecluster stop {instance_name}	停止指定的实例。	• 停止task0对应的实例：ecluster stop task0.ncluster-v100 • 停止一组实例：ecluster stop {ncluster-v100}

命令	命令说明	命令示例
ecluster start {instance_name}	启动指定的实例。	• 启动task0对应的实例：ecluster start task0.ncluster-v100 • 启动一组实例：ecluster start {ncluster-v100}
ecluster kill {instance_name}	释放指定的实例。	• 释放task0对应的实例：ecluster kill task0.ncluster-v100 • 释放一组实例：ecluster kill {ncluster-v100}
ecluster mount {instance_name}	为指定的实例挂载NAS文件系统到*/ncluster*目录。	ecluster mount task0.ncluster-v100
ecluster scp {source} {destination}	安全拷贝文件或目录。	ecluster scp /local/path/to/upload task0.ncluster-v100:/remote/path/to/save
ecluster addip {instance_name}	将指定任务中实例的固定公网IP添加到安全组。	ecluster addip task0.ncluster-v100
ecluster rename {old_name} {new_name}	重命名指定实例。	ecluster rename task0.ncluster-v100 task1.ncluster-v100

基于配置文件创建实例时，您可以参考以下内容新建配置文件。

```
; config.ini

[ncluster]
; The job name for current creation job.
name=ncluster-v100
; The number of machine you want to create
machines=1
; The system disk size for instances in GB
system_disk_size=300
; The data disk size for instances in GB
data_disk_size=0
; The system image name you want to installed in the instances.
image_name=ubuntu_18_04_64_20G_alibase_20190624.vhd
; The instance type you want to create at Alibaba Cloud.
instance_type=ecs.gn6v-c10g1.20xlarge
; The spot instance option; If you want to buy spot instance, please set it to True.
spot=False
; If only used to create instances, it can set to True.
confirm_cost=False
; Confirm the next operation will cost money, if set to True will default confirmed.
skip_setup=True
; Nas create/mount options; Set True will disable nas mount for current job.
disable_nas=True
; The zone id info. The option provided to use resource in the zone.
zone_id=cn-hangzhou-i
; Specify the vpc name
vpc_name=ncluster-vpc

[cmd]
install_script=pwd
```