安装

Clara Train SDK 的 NoteBooks 的 docker(与 horovod 的安装 docker 步骤不同)

这是一组多个笔记本，它将引导您了解 Clara train SDK 的特性和功能。
有多个笔记本显示：

- 性能增益
- 基于 AutoML 的超参数优化
- 联合学习功能
- 特定领域示例
- 辅助注释

Pre-requisites

1. 1 个或更多 Nvidia GPU. (推荐用 2 个以上 GPUS 来使用高级功能例如 AutoML).
2. 如果已经安装了 docker，那么 user 应该在 docker group 里。否则，需要 sudo 来安装 pre requisite

这个 NoteBooks 包含:

1. 脚本用来:
    i. 安装必备组件 Docker 19, docker compose
    ii. 开始拉取 clara docker 和运行 jupyter lab
2. SampleData：开始使用 sdk。
3. 显示 sdk 所有功能的多个笔记本

Getting started

1. 安装预先安装的组件

如果您已经有 docker 19+、nvidia docker 和 docker compose，则可以跳过此步骤。否则，可以使用提供的脚本安装 docker 和 docker compose

1.首先把 github 上的 clara-train-examples 下载下来

git clone https://github.com/NVIDIA/clara-train-examples.git

进入目录

Cd clara-train-examples/NoteBooks/scripts

（注意，这里如果要运行 horovod 并有共享数据的话，需要更改.sh 的内容）

Vi installDocker.sh
内容如下（黄色高亮的是需要注意改动的地方）

```
#!/bin/bash

# SPDX-License-Identifier: Apache-2.0

DOCKER_IMAGE=nvcr.io/nvidia/clara-train-sdk:v3.1.01

DOCKER_Run_Name=claradevday-dataShare-hvd-unlim
      #这里名字如果新启动一个 docker 就需要改名
jnotebookPort=$1
GPU_IDs=$2
AIAA_PORT=$3
##################################### check if parameters are empty
if [[ -z  $jnotebookPort ]]; then
    jnotebookPort=8890
fi
if [[ -z  $GPU_IDs ]]; then  #if no gpu is passed
    # for all gpus use line below
    GPU_IDs=all
    # for 2 gpus use line below
    #GPU_IDs=2
```

```
    # for specific gpus as gpu#0 and gpu#2 use line below
#    GPU_IDs='"device=1,2,3"'
fi
if [[ -z  $AIAA_PORT ]]; then
    AIAA_PORT=5000
fi
##################################### check if name is used then exit
docker ps -a|grep ${DOCKER_Run_Name}
dockerNameExist=$?
if ((${dockerNameExist}==0)) ;then
  echo --- dockerName ${DOCKER_Run_Name} already exist
  echo ----------- attaching into the docker
  docker exec -it ${DOCKER_Run_Name} /bin/bash
  exit
fi

echo -----------------------------------
echo starting docker for ${DOCKER_IMAGE} using GPUS ${GPU_IDs}
jnotebookPort ${jnotebookPort} and AIAA port ${AIAA_PORT}
echo -----------------------------------

extraFlag="-it "
cmd2run="/bin/bash"

extraFlag=${extraFlag}" -p "${jnotebookPort}":8890 -p "${AIAA_PORT}":80"
#extraFlag=${extraFlag}" --net=host "
#extraFlag=${extraFlag}" -u $(id -u):$(id -g) -v /etc/passwd:/etc/passwd -v
/etc/group:/etc/group "

echo starting please run "./installDashBoardInDocker.sh" to install the lab
extensions then start the jupeter lab
echo once completed use web browser with token given
yourip:${jnotebookPort} to access it

docker run -itd --net=host  -v /data1:/data1 -v /home/devops1/clara-
training-examples:/home/devops1/clara-training-examples \
#这里是共享的文件夹，和之前 nfs 的文件夹名称一样。如/data1
#/home/devops1/clara-training-examples 是当前 clara-train-examples 的文件夹，
#如果需要使用这个文件夹中的内容，也需要加上这个文件
  --rm ${extraFlag} \
  --name=${DOCKER_Run_Name} \
  --gpus ${GPU_IDs} \
  -v ${PWD}/../:/claraDevDay/ \
  -w /claraDevDay/scripts \
```

```
  --runtime=nvidia \
  --shm-size=126g  \  #这里视具体情况来定分配 shm 的空间大小
 --ulimit memlock=-1:-1 --ulimit stack=67108864 \
  ${DOCKER_IMAGE} \
  ${cmd2run}

echo -- exited from docker image
```

运行 shell 安装 docker 的预设要求

```
sudo installDocker.sh
```

2. 运行 docker

在 clara-train-examples/NoteBooks/scripts 目录下运行 docker 的 shell，它将拉取最新的 clara train sdk 并以交互模式启动它，格式为：

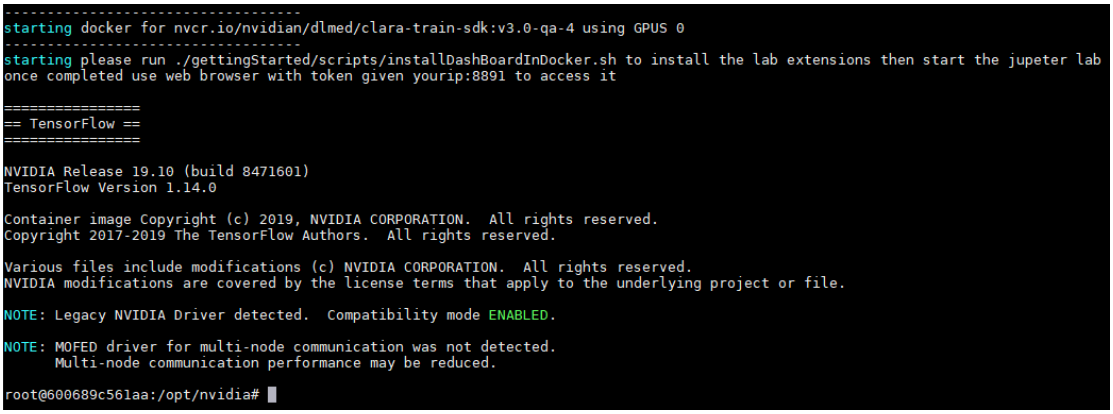
```
./startDocker.sh <portForNotebook> <gpulist>  <AIAA_PORT>
```

例如 4 个 gpu 在 8890 端口和 AIAA server 在 5000 端口

```
./startDocker.sh 8890 '"device=1,3"' 5000
```

默认./startDocker.sh 将使用所有可用的 gpu 和端口 8890 的 NoteBooks 和 5000 的 AIAA

现在可以看到类似这样的输出



3. 启动 jupyter lab

为了只启动 jupyter 实验室，您可以运行下面的简单命令。您也可以安装 GPU 扩展，然后启动 Jupyter 实验室，如步骤 3.1 所示.

```
/claraDevDay/scripts/startJupyterLabOnly.sh
```

3.1 （可选）安装 GPU 仪表板扩展并启动 jupyter lab

这个 docker 使用 NVIDIA 的 RAPIDS AI 团队的 GPU 仪表板扩展 (https://github.com/rapidsai/jupyterlab-nvdashboard). 请在 docker 中运行以下命令来安装插件并运行 jupyterlab

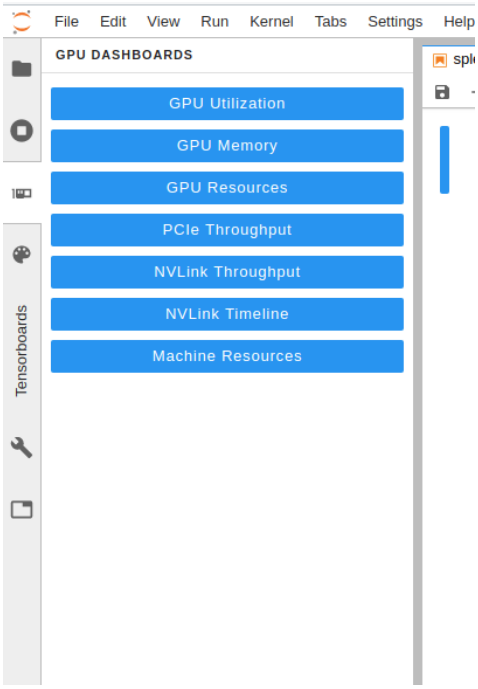
```
./claraDevDay/scripts/installDashBoardInDocker.sh
```

4. 使用给定的 token 打开浏览器

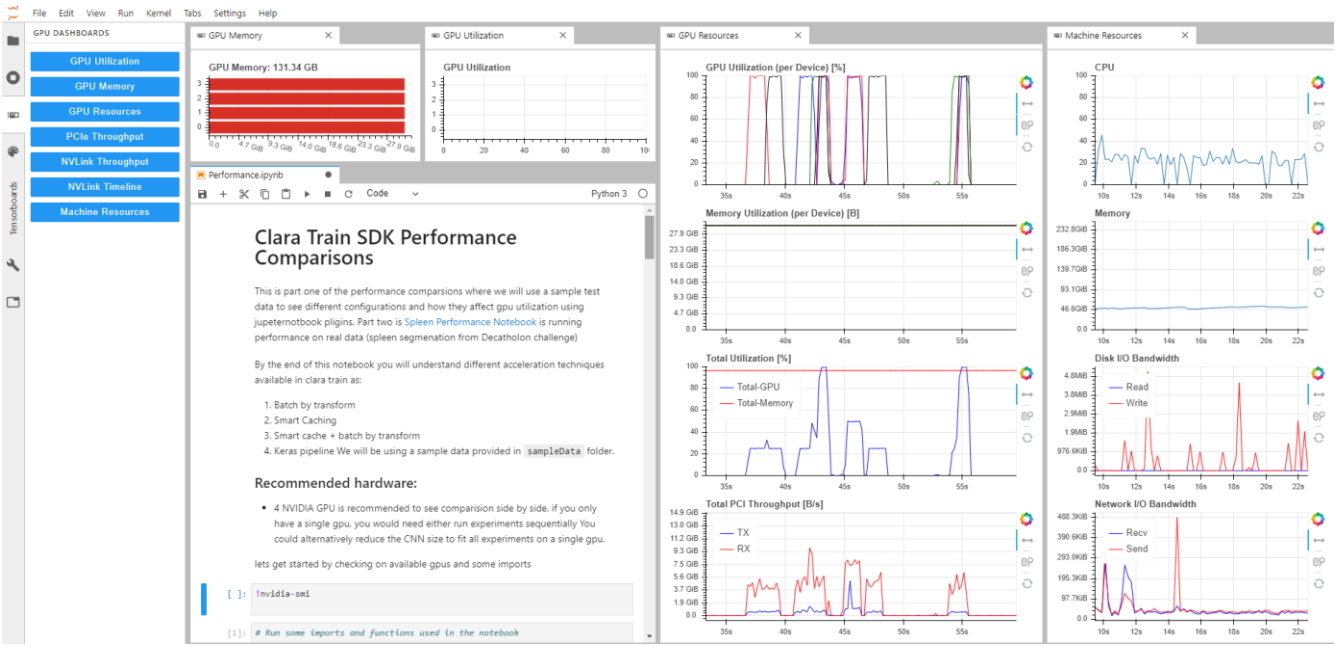
现在，您可以使用终端中提供的 token 在上面指定的端口（默认值为 8890）上转到浏览器。你应该看看 jupyter lab，在那里你应该开始运行欢迎笔记本 Welcome Notebook.。此页显示所有可用笔记本

5. 激活 GPU 仪表板（如果执行 3.1，则为可选）

在开始之前，我们需要激活 GPU 仪表板。查看左侧边栏并单击系统仪表板。



接下来，单击 `GPU Utilization` (GPU 利用率)、`GPU Memory` (GPU 内存)、`GPU Resources` (GPU 资源)和 `MachineResources`(机器资源)。所有这些都将在新选项卡中打开。单击并按住 `GPU Utilization`(GPU 利用率)选项卡，然后将其拖动到屏幕的最右侧区域。它将把标签固定在笔记本的顶部。使用 `GPU Memory` (GPU 内存)选项卡执行相同的过程，并将其停靠在屏幕右下角。结果应该与以下相似



现在我们可以看到 GPU 利用率和 GPU 内存，而我们通过 NoteBook 运行。

接下来是使得 clara docker 可以多机多卡训练的流程：

前三步与 horovod 相似：

使用多机多卡需要满足以下 3 个先决条件：

1. 不同机器可以访问相同的文件：nfs
2. 不同机器使用相同的训练环境：Docker（clara docker 已经安装）
3. 不同机器可以 ssh 交互：ssh 免密登录

假设现在要在两台服务器 A 和 B 上多机多卡跑 horovod，A 为主 worker，下面介绍怎么准备 horovod 的启动条件。

1.NFS

在 A 上的操作

```
# 在 A 上的操作
#1. 安装 nfs 服务器
sudo apt install nfs-kernel-server

#2. 编写配置文件
sudo vi /etc/exports
#/etc/exports 文件的内容如下
/data1/share *(rw,sync,no_subtree_check,no_root_squash)

#3. 创建共享目录
sudo mkdir -p /data1/share

#4. 重启 nfs 服务
sudo service nfs-kernel-server restart

#5. 常用命令工具：
#在安装 NFS 服务器时，已包含常用的命令行工具，无需额外安装。

#显示已经 mount 到本机 nfs 目录的客户端机器。
sudo showmount -e localhost

#将配置文件中的目录全部重新 export 一次！无需重启服务。
sudo exportfs -rv

#查看 NFS 的运行状态
```

```
sudo nfsstat

#查看 rpc 执行信息，可以用于检测 rpc 运行情况
sudo rpcinfo

#查看网络端口，NFS 默认是使用 111 端口。
sudo netstat -tu -4
```

在 B 上的操作

```
# 在 B 上的操作
#1. 安装 nfs 客户端
sudo apt install nfs-common

#2. 查看 NFS 服务器上的共享目录
sudo showmount -e A 的 ip

#3. 创建本地挂载目录
sudo mkdir -p /data1/share

#4. 挂载共享目录
sudo mount -t nfs A 的 ip:/data1/share /data1/share
```

2.Docker 的安装（已安装）

我们需要分别在 A,B 服务器上按照 clara 的 docker

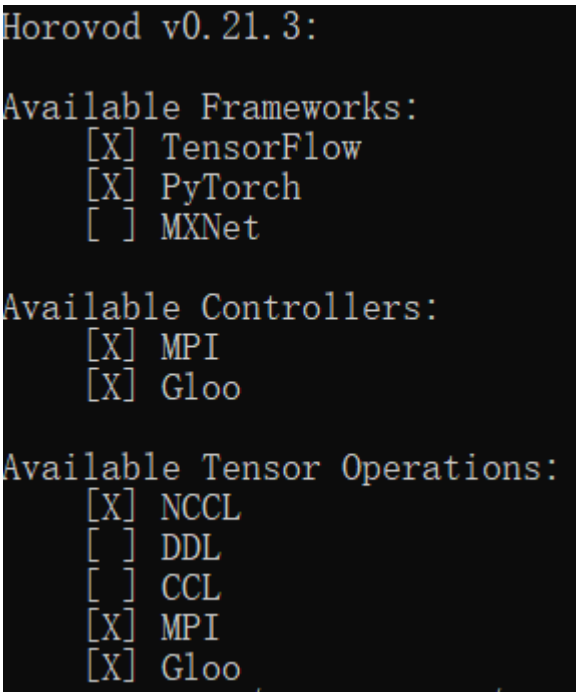
但有一步需要增加，安装完 docker 后

我们在 clara docker 上的终端运行

Horovodrun –check

来检查 horovod 的安装情况

有可能会发现 NCCL 没有安装的情况



NCCL 前面如果有[X]才表明开启，没有[X]表明没安装,于是我们需重新安装 nccl 和 horovod

现在应该是在 clara docker 内进行如下操作：
查看 cuda 版本
nvcc –V
cat /etc/issue
安装对应的 nccl 库的版本
可上网查询 nccl 和 cuda 对应的版本，
如这个网 https://developer.download.nvidia.cn/compute/cuda/repos/ubuntu1804/x86_64/
找到对应版本后安装
apt install libnccl2=2.8.4-1+cuda11.0 libnccl-dev=2.8.4-1+cuda11.0

如果没安装完成先做如下步骤
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin

mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-pin-600

add-apt-repository "deb
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/ /"

apt-get install software-properties-common

如果没安装完成，需要改一下 dns 服务器

vi /etc/resolv.conf
进入之后
nameserver 223.5.5.5
nameserver 223.6.6.6
更新一下 apt-get
```
apt-get update
```
现在安装
apt-get install software-properties-common

add-apt-repository "deb
http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/ /"

apt-get update
再来安装要安装的 nccl 库
apt install libnccl2=2.8.4-1+cuda11.0 libnccl-dev=2.8.4-1+cuda11.0
（之前如果有提示提到 apt-get dist-upgrade，可以跑一下这个命令，应该不是必须的）
apt-get dist-upgrade

现在开始安装带有 nccl 的 horovod
如果我们原来有安装过 horovod，需要先卸载
Pip uninstall horovod
然后现在安装带有 nccl 的 horovod

HOROVOD_GPU_OPERATIONS=NCCL　HOROVOD_WITH_TENSORFLOW=1
HOROVOD_WITH_PYTORCH=1　pip install --no-cache-dir horovod
这个命令可以随机应变视需求而定，HOROVOD_GPU_OPERATIONS=NCCL 代表支持 nccl 运算
HOROVOD_WITH_TENSORFLOW=1 代表支持 tensorflow 框架，HOROVOD_WITH_PYTORCH=1 代表支持 pytorch 框架，具体命令可以参考
https://github.com/horovod/horovod/blob/master/docs/install.rst

安装完成后 horovod 就支持了 NCCL 库

3.ssh 免密登录

此时 A 和 B 应分别在 clara train sdk 容器内。

1. 先在 B 服务器上开启 ssh
   #1. 修改 sshd 配置
   vim /etc/ssh/sshd_config

#2. 改动如下
Port 12345
PermitRootLogin yes
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2

#3. 保存配置，启动 sshd
/usr/sbin/sshd

#4. 查看 ssh 是否启动
ps -ef | grep ssh

#5. 修改 root 的密码
passwd

2.
3. 在 A 服务器上创建秘钥并且免密登录到 B
#1. 生成秘钥，一直回车即可，注意生成秘钥位置
ssh-keygen -t rsa

#2. 在 B 上创建.ssh 文件夹
ssh -p 12345 B 的 ip mkdir -p .ssh

#3. 将公钥添加到 B 的 authorized_keys 里，注意 A 的秘钥路径是否正确
cat .ssh/id_rsa.pub | ssh -p 12345 B 'cat >> .ssh/authorized_keys'

#测试是否可以免密登录
ssh -p 12345 B 的 ip

启动测试

至此 horovod 的启动环境就搭好了，剩下的配套地修改训练代码可以参考 horovod 的 docs 去改。

这里以 horovod 的 github 为例测试一下是否可以正常启动多机多卡训练。以下操作在服务器 A 上进行。

1. 将 horovod 的代码下载到共享文件，注意下 tag 跟 docker 对应的版本
git clone -b v0.18.2 github.com/horovod/horo
2. 修改 examples 下的 pytorch_imagenet_resnet50.py，将 imagenet 路径修改为自己的路径(应在/data1/share 里)。
如果使用 pytorch_mnist.py 可以不修改代码。

3. 安装 TensorboardX 和 tqdm（服务器 B 也要安装）
   pip install tensorboardX
   pip install tqdm
4. 运行启动两台机多卡命令，每个服务器各用 4 张卡（可根据实际情况更改），8 表示一共的卡数
   horovodrun -np 8 -H localhost:4,B_ip:4 -p 12345 python pytorch_imagenet_resnet50.py

   分别查看 A 和 B 的显卡占用，是否多机多卡启动正常。

Unet 3d 测试

接下来是关于测试 unet 3d 的流程，使用的代码来自 github/monai

Git clone https://github.com/Project-MONAI/tutorials.git

安装需要的安装包

Pip install monai

```
python -m pip install -U pip
python -m pip install -U matplotlib
python -m pip install -U notebook
pip install -r https://raw.githubusercontent.com/Project-MONAI/MONAI/master/requirements-dev.txt
```

直接在相关目录下跑 horovod 多机多卡训练

Horovodrun –np X –H localhost:A,worker2_ip:B, Worker3_ip:C -p 12345 python /data1/share/tutorials/acceleration/distributed_training/unet_training_horovod.py

-H 指的是 host 模式

X 是总卡数

Localhost 是本地 ip

Worker2_ip，Worker3_ip 是其他的机器的 ip

A,B,C..是每个机器对应使用的卡数

X=A+B+C+...

-p 12345 是指端口数，可能需要根据具体情况来更改

/data1/share/tutorials/acceleration/distributed_training/unet_training_horovod.py

这个是程序的目录，/data1/share 是共享文件夹名，与之前的设置有关，可能会根据具体情况修改

如果跑完上面命令，nvidia-smi 查看各个机器的 GPU 占用，如果都在运行那表明正常运行。