**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:
   1. 安装必备组件 Docker 19, docker compose
   2. 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：

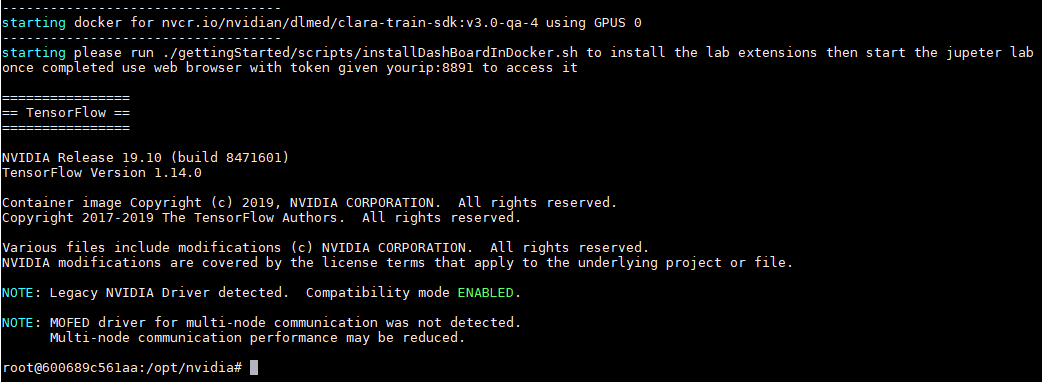
./startDocker.sh <portForNotebook> <gpulist> <AIAA\_PORT>

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

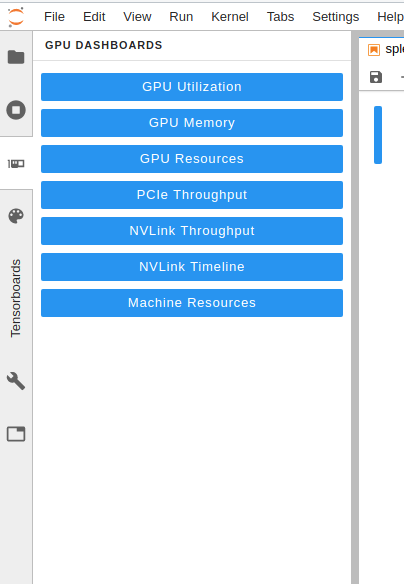
这个docker使用NVIDIA的RAPIDS AI团队的GPU仪表板扩展(https://github.com/rapidsai/jupyterlab-nvdashboard). 请在docker中运行以下命令来安装插件并运行jupyterlab

./claraDevDay/scripts/installDashBoardInDocker.sh

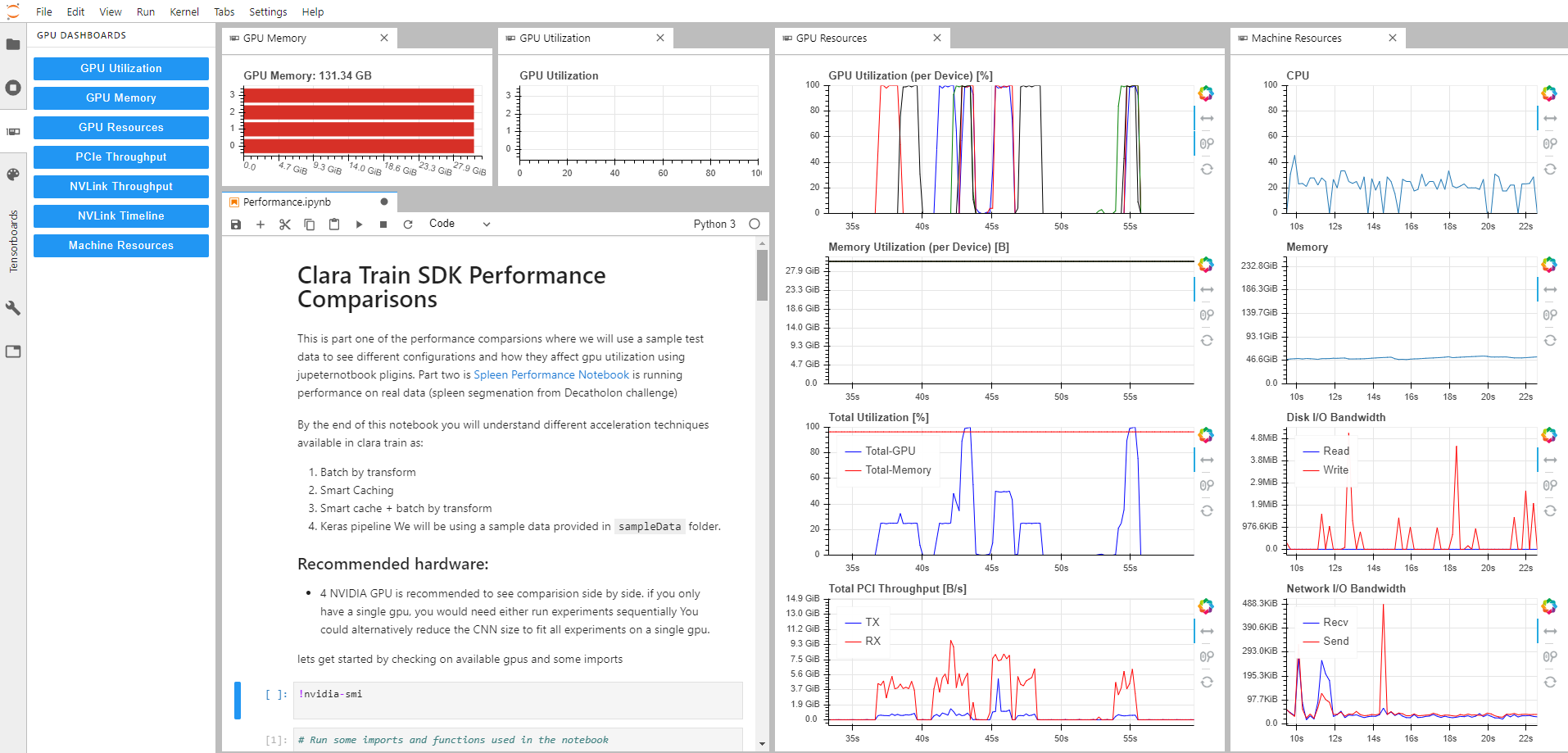
## 4. 使用给定的token打开浏览器

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](https://github.com/NVIDIA/clara-train-examples/blob/master/NoteBooks/Welcome.ipynb).。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。[](https://github.com/NVIDIA/clara-train-examples/blob/master/NoteBooks/screenShots/left_side_bar.png)

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

[](https://github.com/NVIDIA/clara-train-examples/blob/master/NoteBooks/screenShots/result.png)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**

1. # 在A上的操作
2. #1. 安装nfs服务器
3. sudo apt install nfs-kernel-server
4. #2. 编写配置文件
5. sudo vi /etc/exports
6. #/etc/exports文件的内容如下
7. /data1/share \*(rw,sync,no\_subtree\_check,no\_root\_squash)
8. #3. 创建共享目录
9. sudo mkdir -p /data1/share
10. #4. 重启nfs服务
11. sudo service nfs-kernel-server restart
12. #5. 常用命令工具：
13. #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。
14. #显示已经mount到本机nfs目录的客户端机器。
15. sudo showmount -e localhost
16. #将配置文件中的目录全部重新export一次！无需重启服务。
17. sudo exportfs -rv
18. #查看NFS的运行状态
19. sudo nfsstat
20. #查看rpc执行信息，可以用于检测rpc运行情况
21. sudo rpcinfo
22. #查看网络端口，NFS默认是使用111端口。
23. sudo netstat -tu -4

**在B上的操作**

1. # 在B上的操作
2. #1. 安装nfs客户端
3. sudo apt install nfs-common
4. #2. 查看NFS服务器上的共享目录
5. sudo showmount -e A的ip
6. #3. 创建本地挂载目录
7. sudo mkdir -p /data1/share
8. #4. 挂载共享目录
9. 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

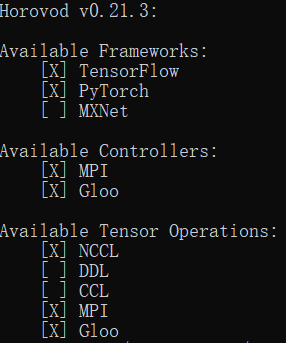
但有一步需要增加，安装完docker后

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. 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 [https://github.com/horovod/horovod.git](https://link.zhihu.com/?target=https%3A//github.com/horovod/horovod.git)
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。