1. 深度学习环境安装部署

1.1. Docker 环境部署

操作系统	CentOS7.3_x64、RedHat7.3_x64(英文系统)
内核	≥3.10.0-327
需要/var/lib/docker 可用空间	>50GB
需要安装文件	Docker-inspur
其他需要	可用的本地 yum 源

- Docker-inspur 目录下载地址：
  svn://10.166.15.100/Docker-inspur
  用户名：inspur　密码：inspur123
- 将 Docker-inspur 复制到/home/inspur 目录下

1.1.1. Docker 安装（集群中所有节点）

1. 进入 docker-install 目录
   cd /home/inspur/Docker-inspur/docker-install
2. 卸载旧的 docker 环境
   yum remove docker \
   docker-common \
   docker-selinux \
   docker-engine
3. 安装 docker 安装时需要依赖包
   yum install -y yum-utils device-mapper-persistent-data lvm2
4. 安装 docker
   yum install -y docker-engine-17.05.0.ce-1.el7.centos.x86_64.rpm docker-engine-selinux-17.05.0.ce-1.el7.centos.noarch.rpm
5. 收集容器需要的 nvidia 驱动文件
   ./nvidia-volume.py

   (如果节点重新安装了其他版本驱动，则需要删除目录/usr/local/docker-inspur/nvidia-volumes/volume，然后重新运行此脚本)

如果集群没有 IB 网络，则步骤 6、步骤 7 不用操作

6. 安装 docker 需要的 ib 驱动依赖库
   ./install-ib-lib.sh

7. 安装 nvidia 与 ib 卡协调工作运行包
   ./install-nvidia-peer.sh

1.1.2. Docker 集群网络安装配置

管理节点安装步骤：

1. cd /home/inspur/Docker-inspur/docker-net
2. 编辑 server.config 文件，将内容改为管理节点内网 ip 地址(通过此 IP 地址能够和其他计算节点通信)
3. 在管理节点执行 install-flannel.sh 脚本

计算节点安装步骤(如果没有计算节点则不执行)：

1. cd /home/inspur/Docker-inspur/docker-net
2. 在所有计算节点执行 install-flannel-node.sh

1.2. 深度学习镜像部署

1.2.1. 导入深度学习镜像（每个节点都需要执行）

1. 进入 docker-images 目录
   cd /home/inspur/Docker-inspur/docker-images
2. 将镜像导入 docker 中
   a) docker load -i caffe.tar
   b) docker load -i tensorflow.tar
   c) docker load -i mxnet.tar

   如果导入后镜像名称显示为 none，则使用 docker tag [imageID] caffe:latest 命令格式配置镜像名称，例如：docker tag 1774a0d9a9bf caffe:latest
3. 导入成功后通过 docker images 查看导入结果

```
[root@node2 docker-install]# docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
dockercaffe                 latest              1774a0d9a9bf        39 hours ago        3.477 GB
dockermxnet                 latest              13bf956d0c06        39 hours ago        2.941 GB
dockertensorflow            latest              d4f85609a7a2        39 hours ago        2.65 GB
registry                    latest              c2a449c9f834        2 weeks ago         33.18 MB
```

1.2.2. 私有镜像库安装部署（只管理节点进行操作）

管理节点：

1. 导入 docker 私有仓库镜像
   cd /home/inspur/Docker-inspur/docker-images
   docker load -i registry.tar
2. 运行仓库镜像
   docker run -d --restart=always -p 5000:5000 -v /var/lib/registry:/var/lib/registry registry

并且将上述命令添加的开机启动中

3. 修改镜像名称并上传 docker 镜像到镜像仓库(其中 **10.166.15.189** 是管理节点内网 IP 地址)

   docker tag caffe 10.166.15.189:5000/caffe
   docker push 10.166.15.189:5000/caffe

   docker tag caffe 10.166.15.189:5000/tensorflow
   docker push 10.166.15.189:5000/tensorflow

   docker tag caffe 10.166.15.189:5000/mxnet
   docker push 10.166.15.189:5000/mxnet

1.2.3. 深度学习环境启动和运行工具使用

所有节点都需要安装

1. 安装

   进入 docker-tools 目录
   cd /home/inspur/Docker-inspur/docker-tools
   执行./install-tools.sh

2. 测试命令使用

   a) dockerrun 运行一个容器

      示例 1：dockerrun –v /home/inspur:/home/inspur
         将主机/home/inspur 映射到容器中的/home/inspur 目录下
         将主机的所有 nvidia gpu 设备添加到容器中使用
      示例 2：dockerrun –v /home/inspur:/root –d 0,1,2
         将主机/home/inspur 映射到容器中的/root 目录下
         将主机的 gpu0,gpu1,gpu2 设备添加到容器中使用

      进入容器后，默认作为守护容器运行，所以执行 exit 后容器不会销毁，依旧在后台运行。

   b) dockerenter 再次进入容器进行命令行操作

      进入使用 dockerrun 启动的容器
      如果当前只有一个容器将直接进入此容器，如果有多个容器将显示容器列表供用户选择。

   c) dockerdestroy 销毁正在运行的容器
      如果当前只有一个容器将直接销毁此容器，如果有多个容器将显示容器列表，输入容器列表 id 将销毁相应的容器，输入 all 将会销毁该用户所有真正运行的容器。

   d) dockermanage 管理正在运行的容器

      dockermanage 支持多个输入命令包括：
      ls：查看容器列表，普通用户只能看到自己的。

enter：进入到某容器，选择对应的容器号即可。
rm：删除某容器，选择对应的容器号即可。
commit：提交一个镜像
push：push 某镜像到本地镜像仓库。
pull：从本地镜像仓库 pull 一个镜像到本地。
exit：退出终端
help：获取帮助

1.2.4. 镜像测试

红色字体为需要依据具体环境进行更改

1. dockercaffe 镜像测试

dockerrun -i caffe -d 0
cd /root/caffe-caffe-0.15
./train_lenet.sh

2. tensorflow 镜像测试

dockerrun -i tensorflow -v /home/inspur/:/home/inspur -d 0,1
cd home/inspur/Docker-inspur/docker-framework/tensorflow_mnist
. /train.sh

3. mxnet 镜像测试

dockerrun -i mxnet -v /home/inspur/:/home/inspur -d 0
cd /home/inspur/Docker-inspur/docker-framework/mxnet _mnist
. /train.sh

2. AIStation 安装配置

2.1. 安装前准备

1. 所有节点的 /etc/hosts 文件中包含所有的节点配置
2. 关闭所有节点的防火墙并取消开机启动
3. 配置管理节点与计算机的双向无密码访问
4. 确认安装启动 mysql 或 mariadb 服务，并配置了用户密码和远程访问权限；需要开启数据库的定时事件。
5. 确认安装目录为 nfs 共享目录，确认/home 目录为共享目录
6. 确认所有节点安装配置了 nis 服务
7. 确认所有管理节点安装以下软件包: lm_sensors, gcc-c++, libtool 1.5.22+, boost-devel

1.36.0+, openssl-devel, libxml2-devel, patch, libcgroup
yum install -y lm_sensors gcc-c++ libtool boost-devel openssl-devel libxml2-devel patch libcgroup

8. 所有计算节点安装以下软件包: lm_sensors ipmitool
yum install -y lm_sernsors ipmitool libcgroup

2.2. 安装 AIStation

1. 解压安装包 tar –zxvf AIStation.tar.gz
2. 进人安装包目录
3. 编辑 install.config 文件，配置安装目录、计算节点名称(与 hosts 文件一致)、 管理节点 IP 地址、管理节点数据库 root 用户访问密码、集群中节点最大 GPU 数
4. 执行"./install"，并耐心等待安装完成
5. 安装完成后执行命令"source /etc/profile"
6. 如果集群节点 GPU 数不统一，例如有部分节点 4 个卡，部分节点 2 个卡，则需要修改以下文件对节点 gpu 数进行修正$torque6/server_priv/nodes 中的每行”gpus=gpu 卡数值”；$MAUI_HOME/maui.cfg 文件中的末尾，”NODECFG[localhost] GRES=gpu:6” 修改 gpu 卡数值
7. 然后执行命令"service aiserver start"来启动 AIStation0.1
8. 然后执行命令"service ainodes start"来启动计算节点的服务
9. 在浏览器中输入"http://管理节点 IP:8080"来管理系统

2.3. 卸载 AIStation

进入安装包目录，执行命令"./uninstall",并耐心等待卸载完成

3. AIStation 功能验证

完成本章的 3.1、3.2 节中所述步骤进行 AIStation 运行任务相关功能验证；并且在完成示例任务后，查看完成任务页面显示是否正常。

3.1. AIStation 任务提交功能

使用 inspur 用户登录 AIStation 管理界面

3.1.1. Base 模板

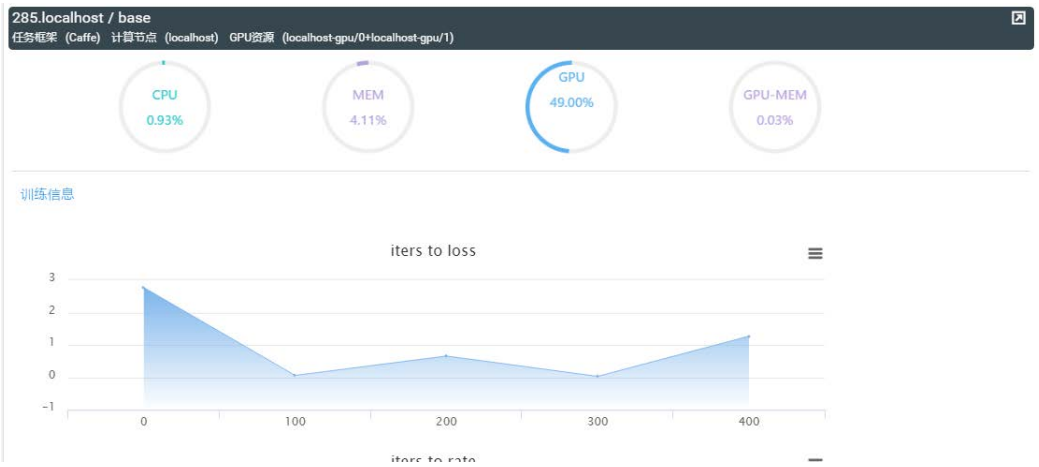
1. 打开任务提交界面，选择模板“base”
2. 依次输入参数
任务框架：caffe

训练步数：5000

启　动　任　务　：　/inspur/caffe-caffe-0.15/build/tools/caffe　train　--solver=/home/inspur/Docker-inspur/docker-framework/caffe-mnist/lenet_solver.prototxt --gpu=0

容器镜像：caffe:latest

3. 点击提交
4. 在任务管理界面双击提交的任务，查看详情



3.1.2. Script 模板

1. 开任务提交界面，选择模板“script”
2. 依次输入参数

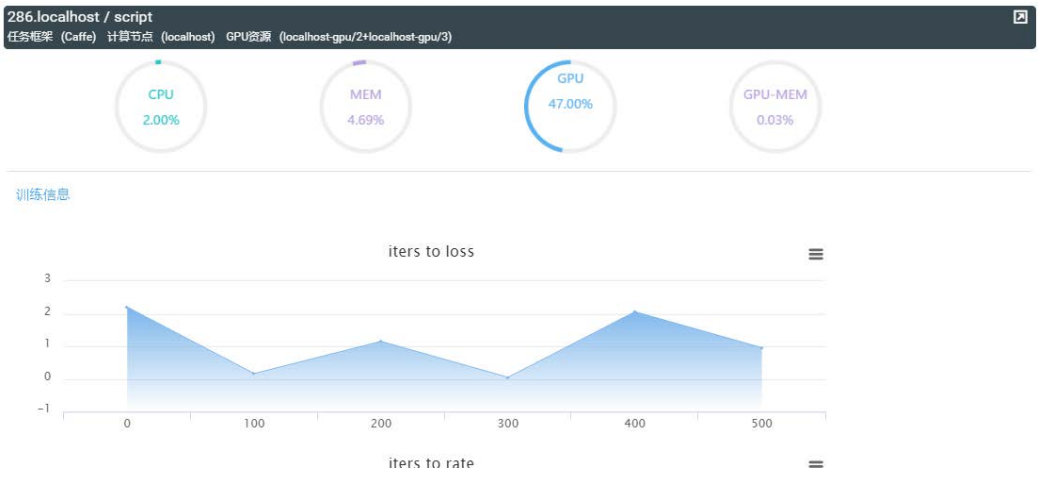
任务框架：caffe

训练步数：5000

脚本：/home/inspur/Docker-inspur/docker-framework/caffe-mnist/train_lenet.sh

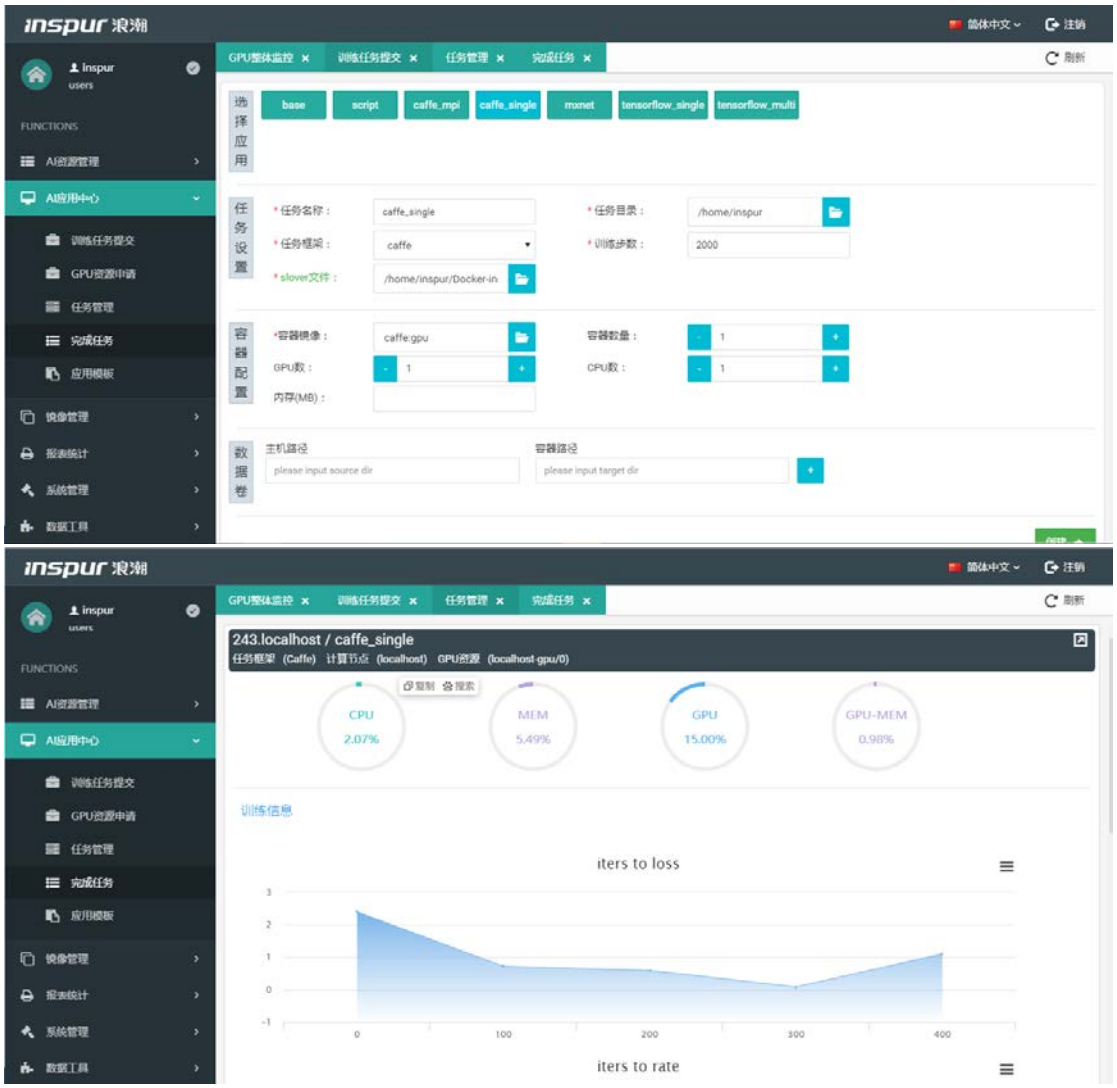
容器镜像：caffe:latest

3. 点击提交
4. 在任务管理界面双击提交的任务，查看详情



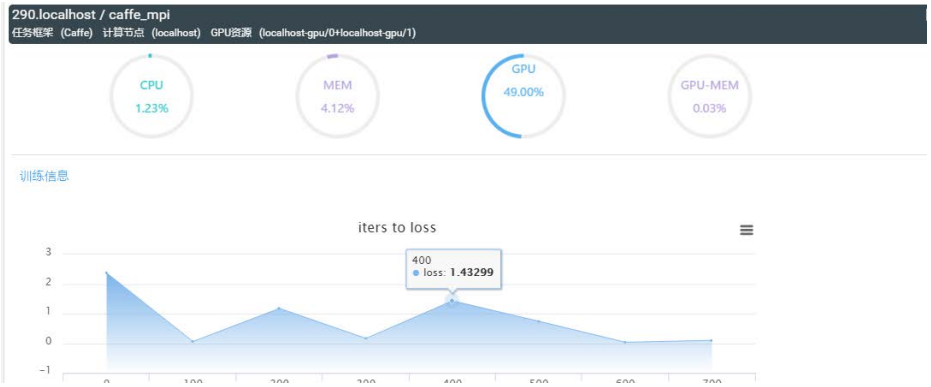
3.1.3. Caffe

1. 打开任务提交界面，选择模板“caffe_single”
2. 依次输入参数
   任务框架：caffe
   训练步数：5000
   Slover 文件：home/inspur/Docker-inspur/docker-framework/caffe-mnist/lenet_solver.prototxt
   容器镜像：caffe:latest
3. 点击提交
4. 在任务管理界面双击提交的任务，查看详情



3.1.4. Caffe-mpi

1. 打开任务提交界面，选择模板“caffe-mpi”
2. 依次输入参数
   任务框架：caffe
   训练步数：5000
   Slover 文件：/home/inspur/Docker-inspur/docker-framework/caffe-mnist/lenet_solver.prototxt
   容器镜像： caffe:latest
   选择容器数量：2
3. 点击提交
4. 在任务管理界面双击提交的任务，查看详情

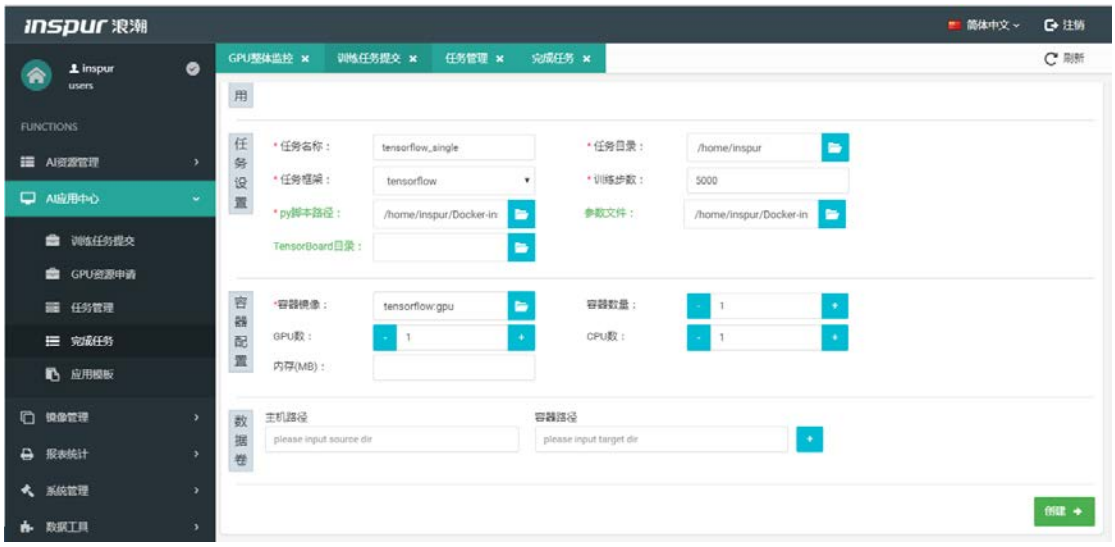


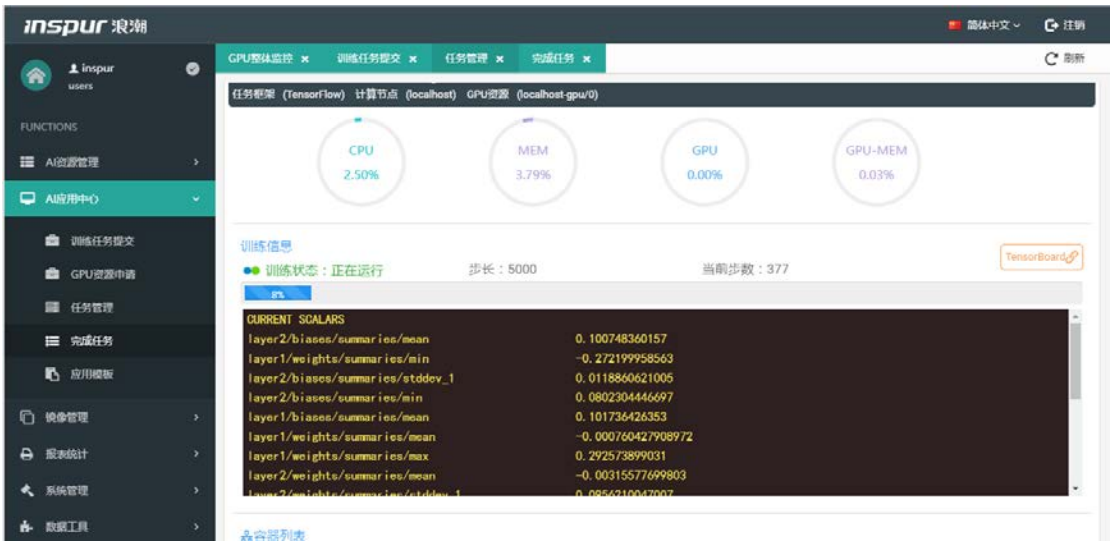

```
localhost
/caffelib/bin/mpirun --allow-run-as-root -machinefile /opt/tsce4/tools/dockerShare/290/machinefile -mca btl_openib_want_cuda_gdr 1 --mca io ompio -np 1 -npernode 1 /root/caffe-mpi/build/tools/caffe train --solver=/home/inspur/Docker-inspur/docker-framework/caffe-mnist/lenet_solver.prototxt --gpu=0,1 2>&1
Hello
--------------------------------------------------------------------------
You requested to run with CUDA GPU Direct RDMA support but this OFED
installation does not have that support.  Contact Mellanox to figure
out how to get an OFED stack with that support.

 Local host:          249b9a1d4e55
--------------------------------------------------------------------------
I0716 01:00:36.207762   46 caffe.cpp:204] Using GPUs 0, 1
I0716 01:00:36.208900   46 caffe.cpp:209] GPU 0: Tesla P100-SXM2-16GB
I0716 01:00:36.209588   46 caffe.cpp:209] GPU 1: Tesla P100-SXM2-16GB
I0716 01:00:37.868439   46 solver.cpp:50] Initializing solver from parameters:
```

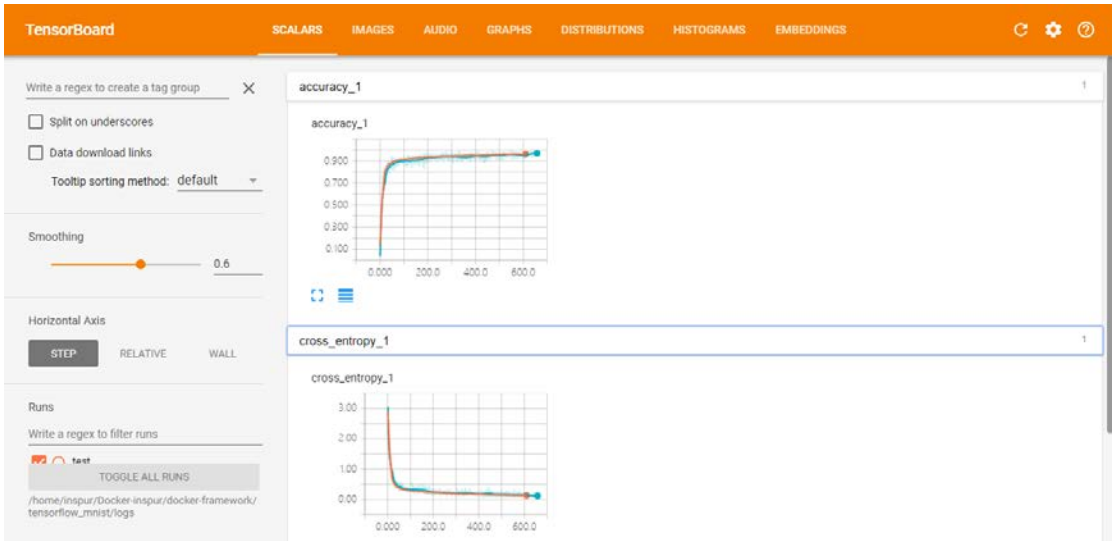
3.1.5. Tensorflow

1. 打开任务提交界面，选择模板“tensorflow_single”
2. 依次输入参数
   任务框架：tensorflow
   训练步数：5000
   Py 脚本路径：/home/inspur/Docker-inspur/docker-framework/tensorflow_mnist/mnist_with_summaries.py
   参数文件：/home/inspur/Docker-inspur/docker-framework/tensorflow_mnist/params
   Tensorboard 日志目录：/home/inspur/Docker-inspur/docker-framework/tensorflow_mnist/logs
   容器镜像：tensorflow:latest
3. 点击提交
4. 在任务管理界面双击提交的任务，查看详情





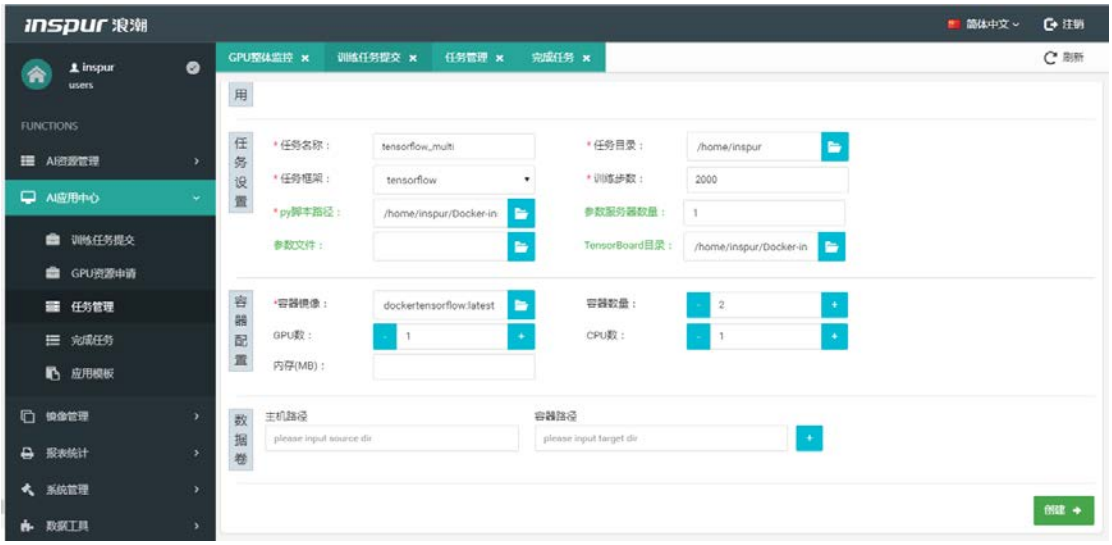
点击“TensorBoard”按钮可通过 TensorBoard 工具查看训练信息

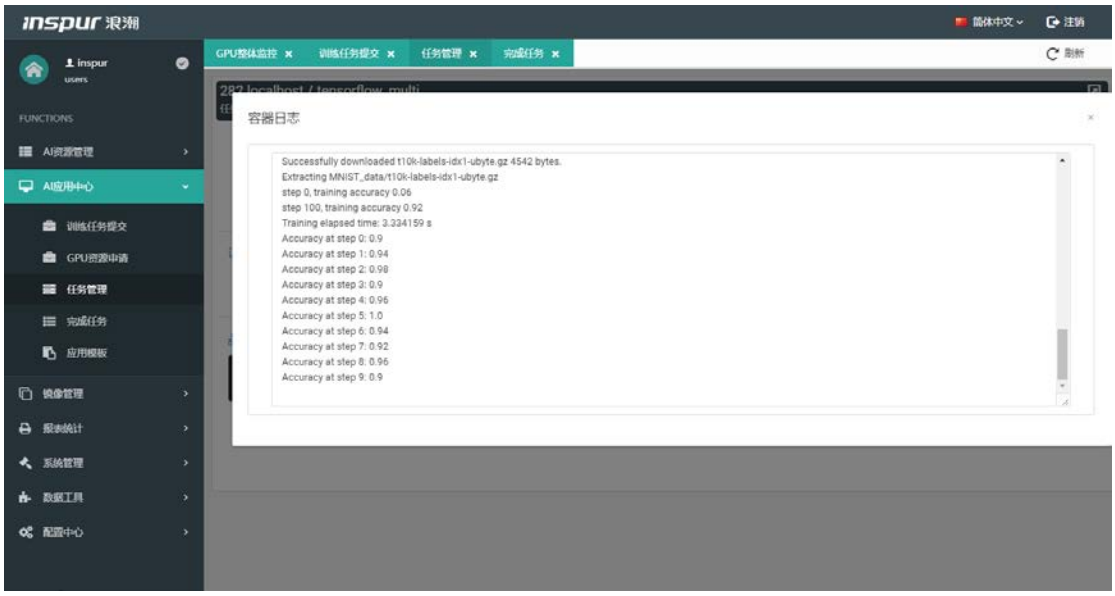


3.1.6. Tensorflow-distributed

1. 打开任务提交界面，选择模板“tensorflow_multi”
2. 依次输入参数
   任务框架：tensorflow
   训练步数：5000
   Py 脚本路径：/home/inspur/Docker-inspur/docker-framework/deepMNIST-distributed-example/distributed_trainer_deepMnist_v2.py
   参数服务器数量：1
   参数文件：/home/inspur/Docker-inspur/docker-framework/deepMNIST-distributed-example/params
   Tensorboard 日志目录：/home/inspur/Docker-inspur/docker-framework/deepMNIST-distributed-example/summaries
   容器镜像：tensorflow:latest
   容器数量：2
3. 点击提交

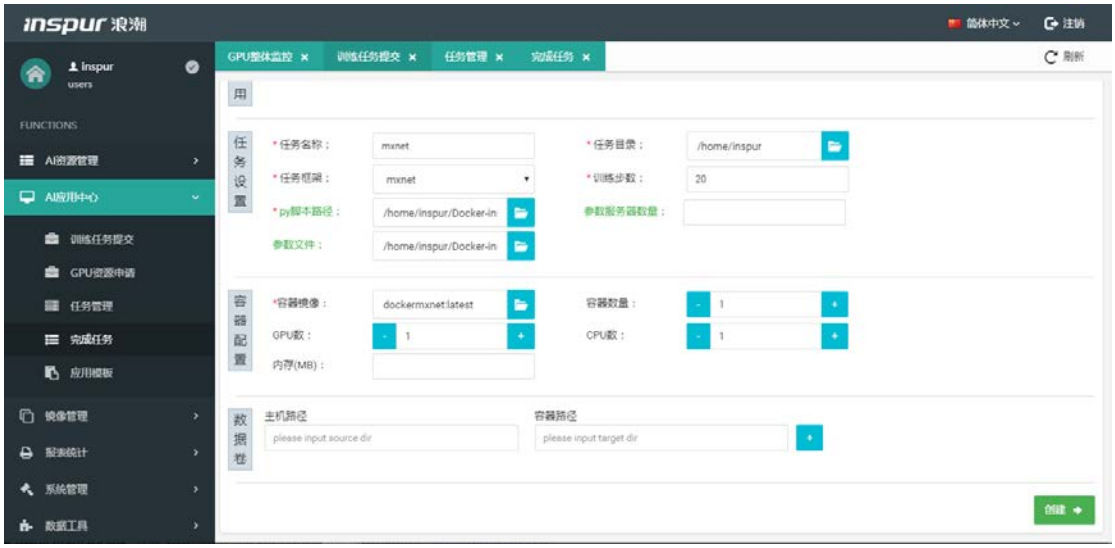
在任务管理界面双击提交的任务，查看详情





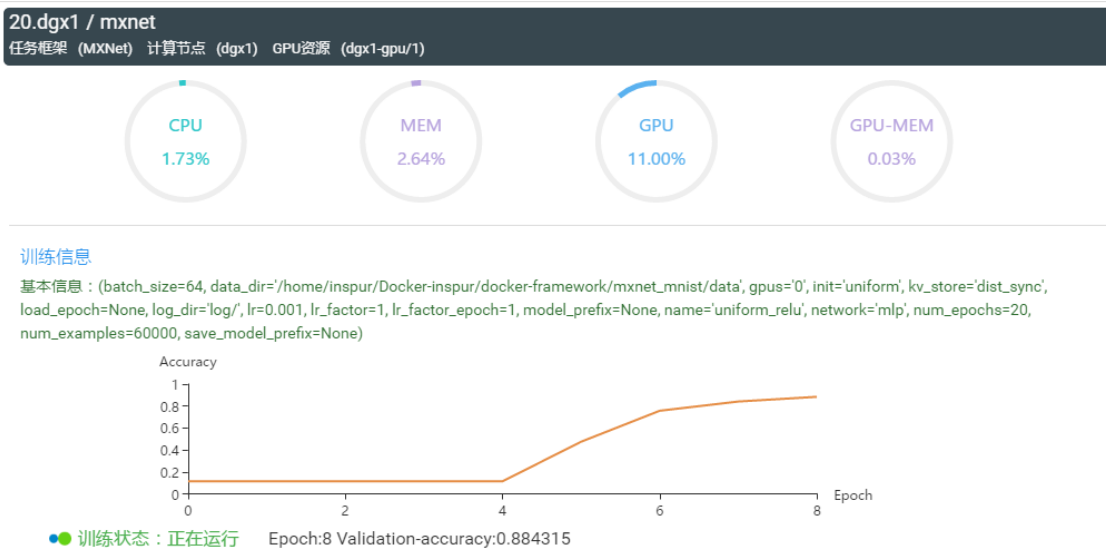
3.1.7. Mxnet

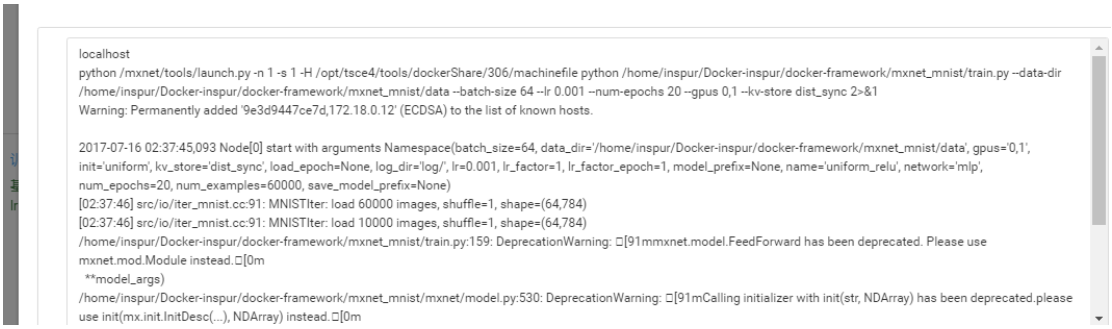
1. 打开任务提交界面，选择模板“mxnet”
2. 依次输入参数
   任务框架：mxnet
   训练步数：20
   Py 脚本路径：/home/inspur/Docker-inspur/docker-framework/mxnet_mnist/train.py
   参数文件：/home/inspur/Docker-inspur/docker-framework/mxnet_mnist/MXParams
   容器镜像： mxnet:latest
3. 点击提交
4. 在任务管理界面双击提交的任务，查看详情



3.1.8. Mxnet-distributed

1. 打开任务提交界面，选择模板“mxnet”
2. 依次输入参数
   任务框架：mxnet
   训练步数：20
   Py 脚本路径：/home/inspur/Docker-inspur/docker-framework/mxnet_mnist/train.py
   参数服务器数量：1
   参数文件：/home/inspur/Docker-inspur/docker-framework/mxnet_mnist/MXParams
   容器镜像： mxnet:latest
   容器数量：2
3. 点击提交
4. 在任务管理界面双击提交的任务，查看详情

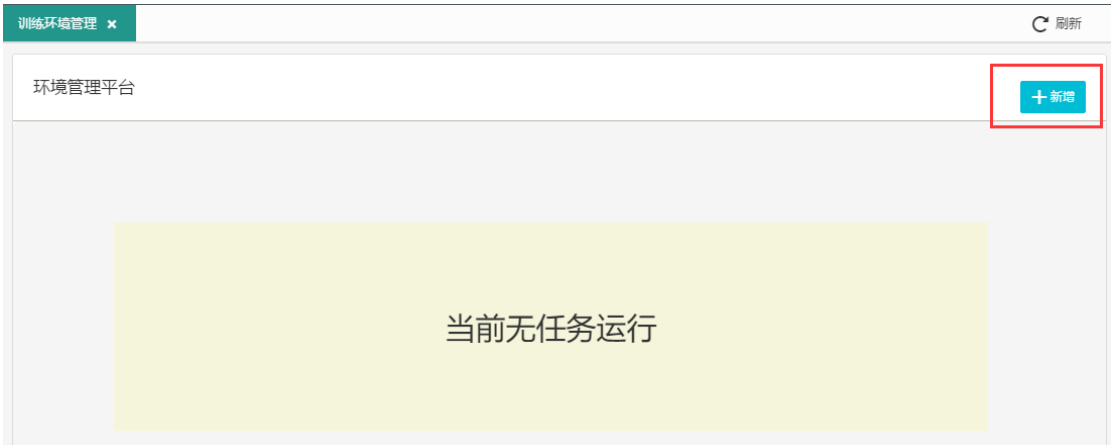




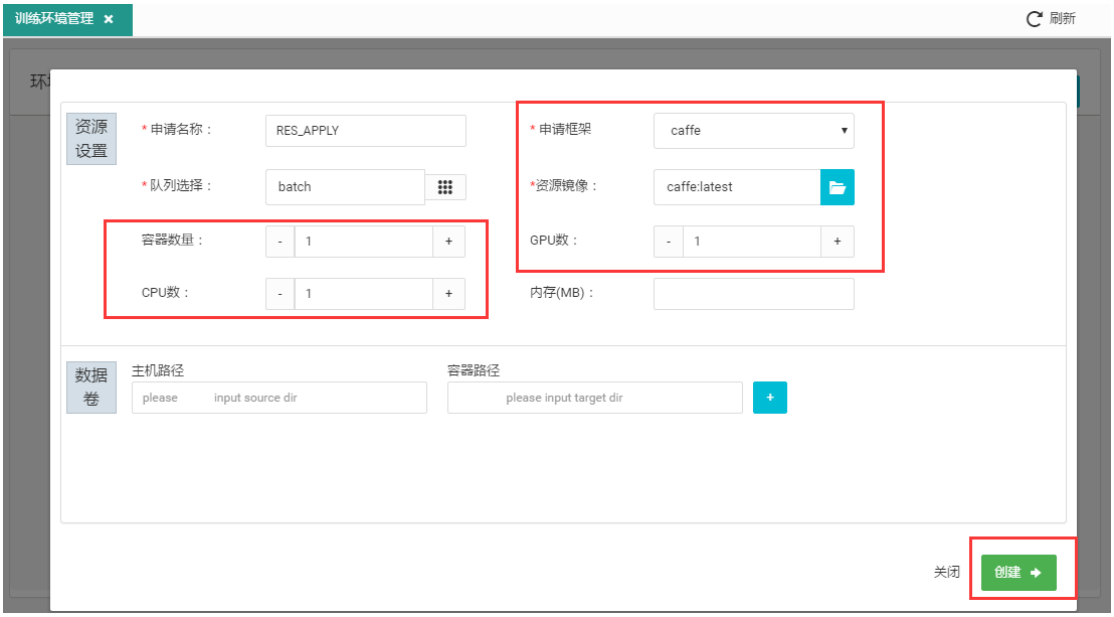
3.2. AIStation 运行环境管理测试

1. 环境创建

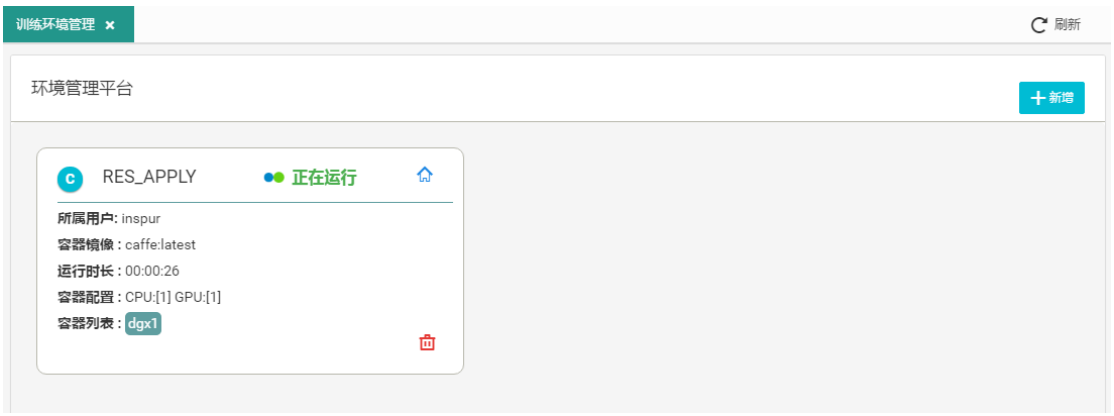
“训练环境管理”菜单点击“新建”按钮



选择框架、镜像及资源数量，点击“创建”

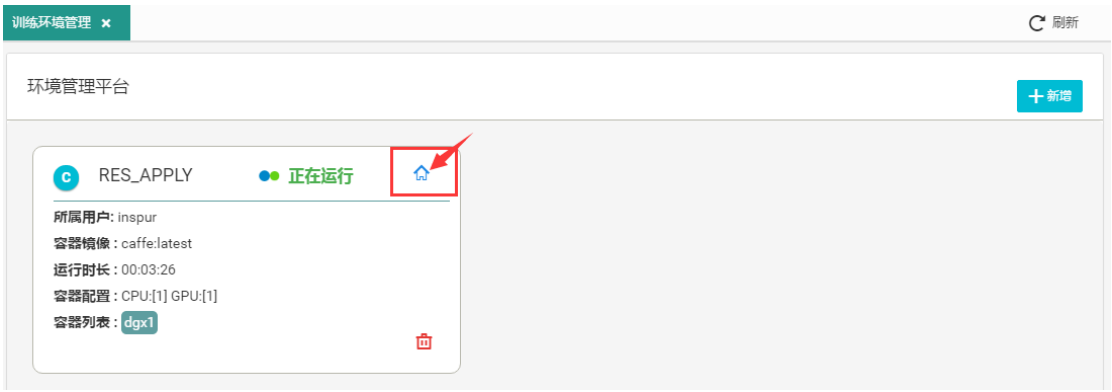


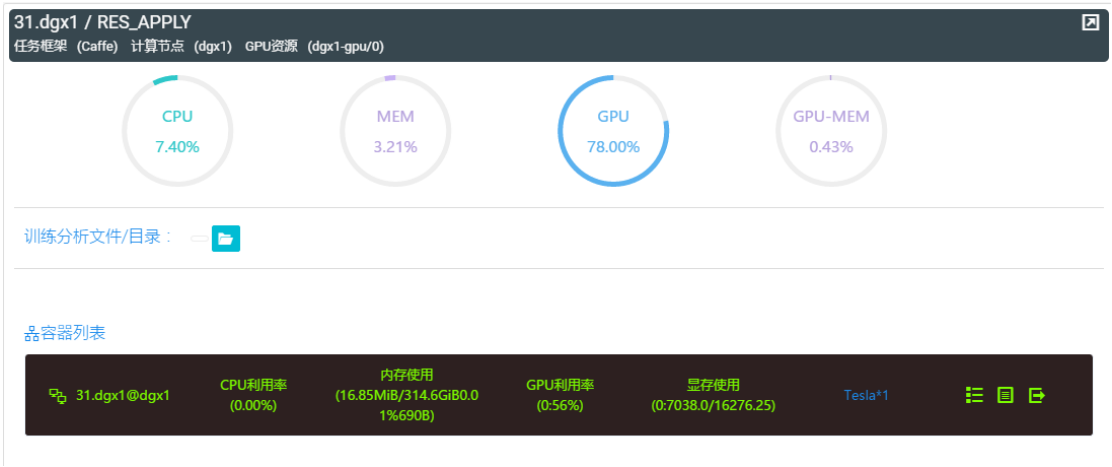
创建成功后在环境管理页面中查看



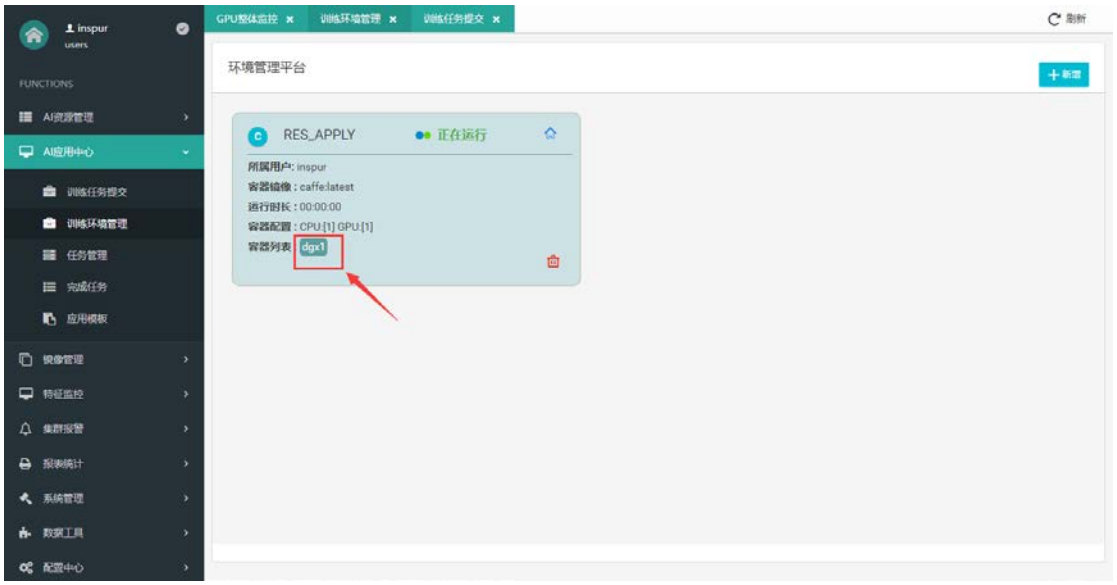
2. 环境详情查看

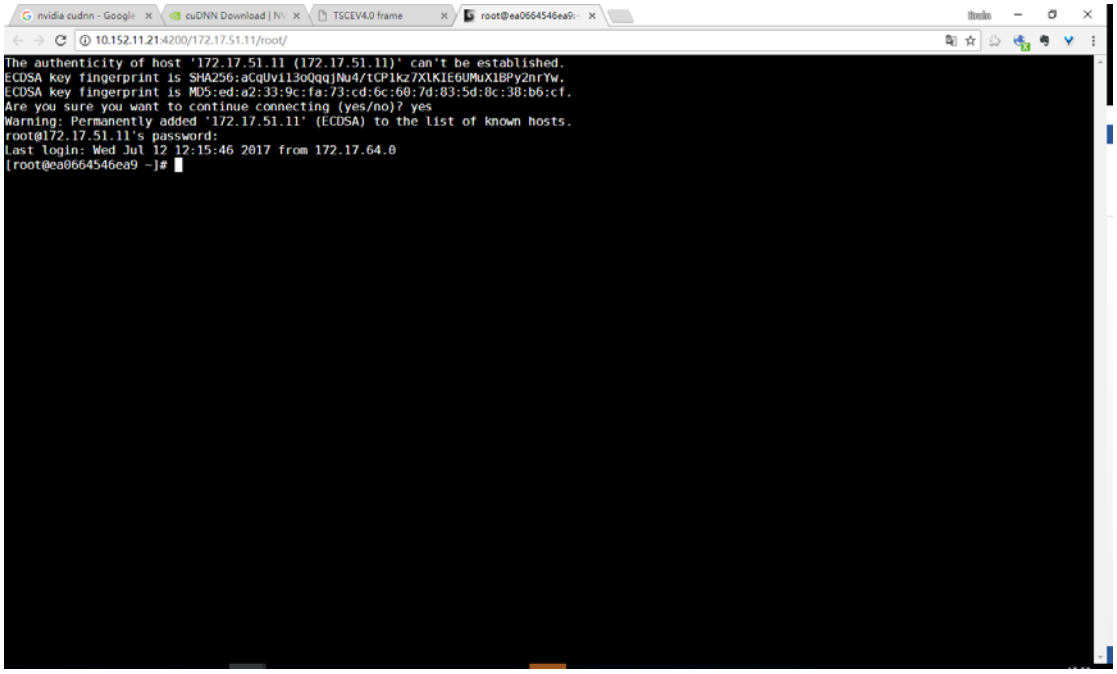
点击 home 按钮可查看训练环境的资源使用情况及容器列表





3. 点击容器名称，查看是否能够打开容器的 web shell 界面





3.3. Shell 任务运行

3.3.1. Caffe

dockerrun -i caffe -d 0
cd /home/inspur/Docker-inspur/docker-framework/caffe-mnist/
./train_lenet.sh

3.3.2. Caffe-mpi

注意：caffe-mpi 只能在 IB 网下使用

节点 1：

1. dockerrun -i caffe -d 0
2. /usr/sbin/sshd –D &
3. 使用 ifconfig 查看容器 ip 地址，以 192.168.10.2 为例

节点 2：

1. dockerrun -i caffe -d 0
2. /usr/sbin/sshd –D &
3. 使用 ifconfig 查看容器 ip 地址，以 192.168.50.2 为例
4. cd /root/caffe-mpi
5. /caffelib/bin/mpirun --allow-run-as-root -host 192.168.10.2, 192.168.50.2 -mca btl_openib_want_cuda_gdr 1 --mca io ompio -np 2 -npernode 1 ./build/tools/caffe train --solver=examples/mnist/lenet_solver.prototxt --gpu=0,

3.3.3. Tensorflow

1. dockerrun -i tensorflow -v /home/inspur/Docker-inspur/docker-framework:/mnt/data -d 0,1
2. cd /mnt/data/tensorflow_mnist
3. . /train.sh

3.3.4. Tensorflow-distributed

节点 1:

1. dockerrun -i tensorflow -v /home/inspur/Docker-inspur/docker-framework:/mnt/data -d 0,1
2. ifconfig 查看容器 IP 地址,以 192.168.10.2 为例
3. cd /mnt/deepMNIST-distributed-example
4. python distributed_trainer_deepMnist_v2.py --ps_hosts=192.168.10.2:2223 --worker_hosts=192.168.10.2:2222,192.168.50.2:2222 --folder_no=2 --job_name=ps --weights_name=30Aug_1kepcohs_2machines --epochs=1000 --task_index=0 &
5. python distributed_trainer_deepMnist_v2.py --ps_hosts=192.168.10.2:2223 --worker_hosts=192.168.10.2:2222,192.168.50.2:2222 --folder_no=2 --job_name=worker --weights_name=30Aug_1kepcohs_2machines --epochs=1000 --task_index=0

节点 2：

1. dockerrun -i tensorflow -v /home/inspur/Docker-inspur/docker-framework:/mnt/data -d 0,1
2. ifconfig 查看容器 IP 地址,以 192.168.50.2 为例

3. cd /mnt/deepMNIST-distributed-example
4. python distributed_trainer_deepMnist_v2.py --ps_hosts=192.168.10.2:2223 --worker_hosts=192.168.10.2:2222,192.168.50.2:2222 --folder_no=2 --job_name=worker --weights_name=30Aug_1kepcohs_2machines --epochs=1000 --task_index=1

3.3.5. Mxnet

dockerrun -i mxnet -v /home/inspur:/home/inspur -d 0,1
cd /home/inspur/Docker-inspur/docker-framework/mxnet_mnist
. /train.sh

3.3.6. Mxnet-distributed

节点 1：

1. dockerrun -i mxnet -v /home/inspur:/home/inspur -d 0
2. 使用 ifconfig 查看容器 ip 地址，以 192.168.10.2 为例

节点 2：

1. dockerrun -i mxnet -v /home/inspur:/home/inspur -d 0
2. 使用 ifconfig 查看容器 ip 地址，以 192.168.50.2 为例
3. cd /home/inspur/Docker-inspur/docker-framework/mxnet_mnist/
4. echo “192.168.10.2” > hostfile
5. echo “192.168.50.2” >> hostfile
6. python /mxnet/tools/launch.py -n 2 -s 1 -H hostfile python train.py --data-dir /home/inspur/Docker-inspur/docker-framework/mxnet_mnist/data --batch-size 128 --lr 0.001 --num-epochs 20 --kv-store dist_sync --gpus=0

4. 其他

4.1. Docker 环境测试,测试容器中使用 GPU 驱动是否正常

执行如下命令可以测试容器内 GPU 驱动是否能够正常使用

docker run --rm --volume=/usr/local/docker-inspur/nvidia-volumes/volume:/usr/local/nvidia:ro --device=/dev/nvidiactl --device=/dev/nvidia-uvm --device=/dev/nvidia-uvm-tools --device=/dev/nvidia0 caffe bash -c "source /etc/profile;nvidia-smi"

4.2. 提交作业测试

下列命令可以在命令行测试 CPU 任务、GPU 任务是否能够正常提交和运行

su inspur

echo “sleep 100” | qsub

echo “sleep 100” | qsub -l nodes=2:ppn=1 –w x=”GRES:gpu@1”

qstat

4.3. AIStation 安装完成后新增节点流程

1. 配置新节点基本服务

   配置与管理节点 ssh 无密码访问

   配置 nfs 挂载：挂载/home 文件夹和 AIstation 安装文件夹

   # mount -t nfs -o vers=3 node2:/home/ /home/

   # mount -t nfs -o vers=3 node2:/opt/ /opt/（opt 目录为软件的安装目录）

   配置 nis 服务

   新增的节点安装 lm_sensors

   yum install -y lm_sernsors ipmitool libcgroup
2. 添加节点信息到数据库中
   a) 在 AIstation 的安装包中 install.config 中 NODELIST 中新增需要增加的节点
   b) 进入安装包 tools 目录下，执行./createNodelist.py 重新生成 nodelist 文件，并且将 nodelist 文件复制到安装包 tsced/tools/目录下， cp nodelist ../tsced/tools/;将 nodelist 文件复制到 AIStation 安装目录的 tools/config/目录下，cp nodelist $TSCE_HOME/tools/config/;
   c) 生成新节点信息 sql 文件，进入安装包 tsced/tools/;执行./nodeinfosql.sh nodelist;将生成的 nodeinfo.sql 文件导入数据库，mysql -p111111 < nodeinfo.sql;
3. tsce 底层监控配置
   a) 修改$TSCE_HOME/tsced/config/tsce_index.conf 的配置文件,配置新增的节点名称,并且同步到每个节点，pleaseDo "cp $TSCE_HOME/tsced/config/tsce_index.conf /var/tsced/config"
   b) 为每个计算节点配置底层采集服务，pleaseDo "cp /opt/tsce/tools/share/init.d/tscecd /etc/init.d/"
   c) 启动底层监控服务

      service tscesd restart

      pleaseDo "service tscecd start"
4. Pbs 添加新增节点
   a) 修改 torque 配置文件,$TORQUE_HOME/server_priv/nodes 配置文件，添加新增的节点信息及节点 GPU 信息，例如:node10 np=0 gpus=4
   b) 进入目录$TORUQE_HOME/share，为新增节点添加 pbs 目录，cp -r master node10(其中 master 是管理节点或者已存在的节点，node10 是要新增加的节点)
   c) 依据上述步骤，进入新增节点 pbs 目录，修改 spool、undelivered、checkpoint 目录权限，执行 chmod 1777 spool;chmod 1777 undelivered;chmod 1777 checkpoint;
   d) 启动新增节点的 pbs 服务

pleaseDo "cp /opt/tsce4/tools/share/init.d/pbs_mom /etc/init.d/"
pleaseDo "service pbs_mom start"

e) 修改 maui 配置文件，在最后添加
NODECFG[hostname] GRES=gpu:[gpu 卡个数]

5. 安装驱动及 docker
为新增节点安装驱动、cuda
依据文档章节 1，安装配置 docker 服务

4.4. 更换管理节点 IP

建议：备份配置文件，重新安装

配置/etc/hosts，配置无密码访问
修改如下文件中管理节点 IP 地址：
/opt/tsce4/tools/config/nodelist master
/opt/tsce4/tools/config/install.config SERVER_IP VNCSERVER
/opt/tsce4/tsced/config/base.conf app_svr_ip
/var/tsced/config/base.conf app_svr_ip(需要修改每个节点)
/opt/tsce4/tomcat8/webapps/tsce4/module/vnc/vnc.html host.value
/opt/tsce4/tomcat8/webapps/tsce4/WEB-INF/classes/config/registryServer.conf
serverIp

如果数据库 IP 地址改变，修改如下文件 IP 地址：
/opt/tsce4/tools/config/install.config MYSQL_SERVER_IP
/opt/tsce4/tsced/config/ex_alarm.conf db server_ip
/opt/tsce4/tsced/config/base.conf db_server_ip
/opt/tsce4/tomcat8/webapps/tsce4/WEB-INF/web.xml jdbc:mysql

4.5. 自定义镜像使用方法

首先用户需要提供一个自定义镜像和示例应用

1. 需要用户提供应用运行方式，例如：python train.sh --data=./data --log=/tmp/log
2. 测试应用是否能够通过命令行运行
   a) 在命令行中，切换到普通用户，使用 dockerrun 命令创建自定义镜像的容器，并且将示例应用目录映射到容器中，例如:dockerrun -v /home/inspur/caffe-ssd:/home/inspur/caffe-ssd
   b) 执行示例应用测试是否能够运行
   cd /home/inspur/caffe-ssd;python train.sh --data=./data --log=/tmp/log
   如果成功运行，没有报错，将运行命令写入到一个脚本文件中进行保存。
3. 测试应用是否能够任务提交模块运行
   a) 在 AIStation 页面中，使用任务提交模块

b) 选择使用 script 模板
c) 模板参数：
   i. 任务框架选择测试应用相对应的框架，如果没有则选择 mxnet 即可
   ii. 训练步数输入测试应用配置的步数
   iii. 脚本选择之前命令行创建的测试命令脚本
   iv. Tensorboard 不用填写
   v. 容器镜像选择自定义镜像
   vi. GPU 和 CPU 酌情选择，一般为 1GPU、4CPU 或者 2GPU、8CPU 进行测试
   vii. 如果测试应用存储目录不在用户目录下，即不在页面中的任务目录下，则需要在数据卷中添加应用存储目录；例如：应用存储目录为/data/caffe-ssd，则添加数据卷：主机目录/data/caffe-ssd/，容器路径/data/caffe-ssd
d) 在任务管理模块查看任务是否正常运行，任务输出是否正常；如果自定义镜像不是 tensorflow、mxnet、caffe 则无法显示训练的可视化和训练进度；如果需要容器能够在 web 界面 ssh 正常使用，则需要在自定义容器中安装 sshd 服务。