分布式机器学习

实验指导书

课程号：85990072

授课教师：王智副教授

编者：袁新杰助教

清华大学
清华大学深圳国际研究生院

清華大學深圳国际研究生院
Tsinghua Shenzhen International Graduate School

[最后编辑于 2023 年 3 月 5 日]

前言

这里说一些前言，当然空着也行。

目录

第 1 章　环境配置

本章将主要介绍分别使用本地计算资源、深研院计算资源和华为云计算资源时构建环境的方法。

1.1　使用本地环境与 GPU

使用本地计算资源可以不收网络链接状况约束，随时随地调试程序，对于简单的项目，本地调试也可能更省时间。

本节以助教所使用的计算机为例，展示环境配置过程。助教使用的计算机系统与配置为：

- 系统：windows10 专业教育版；22H2
- 处理器：Intel(R) Core(TM) i7-8700 CPU
- 内存：16GB
- 显卡：NVIDIA GeForce RTX 2060
- 编辑器：Visual Studio Code

1.1.1　安装 CUDA 工具箱

对于包含英伟达显卡的计算机，我们推荐首先安装 CUDA 工具包以使用 GPU 加速计算。

注意，仅包含英伟达 GPU 的计算机需要安装 CUDA 工具箱以使用 GPU 加速计算。使用核显或 AMD 显卡的计算机再后续步骤中使用 CPU 计算即可

GPU 型号、CUDA 工具包、PyTorch 版本相互关联。因此需要一起规划好。

(https://developer.nvidia.com/zh-cn/cuda-gpus) 查看得到我的显卡的 2060 的算力为 7.5。

支持 CUDA 的 GeForce 和 TITAN 产品

GeForce 和 TITAN 产品

GPU	计算能力
GeForce RTX 3090	8.6
GeForce RTX 3080	8.6
GeForce RTX 3070	8.6
NVIDIA TITAN RTX	7.5
GeForce RTX 2080 Ti	7.5
GeForce RTX 2080	7.5
GeForce RTX 2070	7.5
GeForce RTX 2060	7.5
NVIDIA TITAN V	7.0

Figure 1.1: CAPTION holder

https://en.wikipedia.org/wiki/CUDA 查看得到支持我显卡的 CUDA 版本为 ≥10.0

https://docs.nvidia.com/cuda/cuda-toolkit-release-notes/index.html 同时 CUDA 对显卡驱动的最低版本也提出了要求，但显卡驱动对 CUDA 向下兼容，因此一般安装了最近发布的显卡驱动版本即可，无需与 CUDA 版本特别对应。

https://pytorch.org/get-started/locally/ 最后要注意，PyTorch 并不一定支持最新的 CUDA 版本，因此安装前再去 PyTorch 上看一眼 PyTorch 支持哪些 CUDA 版本。

我们发现 PyTorch 最高支持到 CUDA 11.7，满足显卡算力对 CUDA 版本 ≥10.0 的要求，因此我们可以选择安装 CUDA 11.7。

选择对应版本 CUDA 安装包并下载安装，安装过程略。https://developer.nvidia.com/cuda-toolkit-archive

GPUs supported [edit]

Supported CUDA Compute Capability versions for CUDA SDK version and Microarchitecture (by code name):

Compute Capability (CUDA SDK support vs. Microarchitecture)

CUDA SDK version(s)	Tesla	Fermi	Keppler (early)	Kepler (late)	Maxwell	Pascal	Volta	Turing	Ampere	Ada Lovelace	Hopper
1.0[29]	1.0 – 1.1										
1.1	1.0 – 1.1+x										
2.0	1.0 – 1.1+x										
2.1 - 2.3.1[30][31][32][33]	1.0 – 1.3										
3.0 - 3.1[34][35]	1.0 –	2.0									
3.2[36]	1.0 –	2.1									
4.0 - 4.2	1.0 –	2.1+x									
5.0 - 5.5	1.0 –			3.5							
6.0	1.0 –			3.5							
6.5	1.1 –				5.x						
7.0 - 7.5		2.0 –			5.x						
8.0		2.0 –				6.x					
9.0 - 9.2			3.0 –				7.0				
10.0 - 10.2			3.0 –					7.5			
11.0[37]				3.5 –					8.0		
11.1 - 11.4[38]				3.5 –					8.6		
11.5 - 11.7.1[39]				3.5 –					8.7		
11.8[40]				3.5 –							9.0
12.0					5.0 –						9.0

Figure 1.2: CAPTION holder

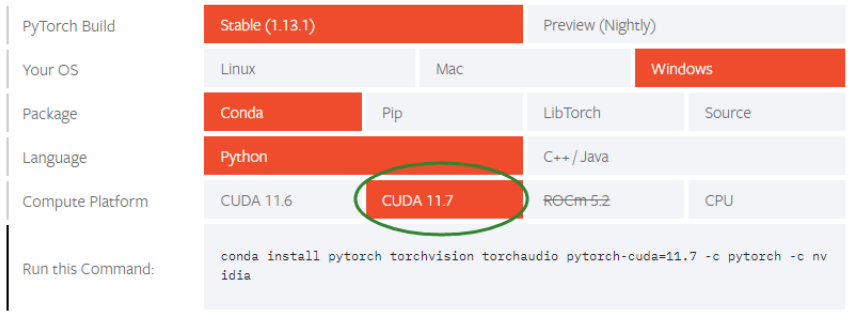

Figure 1.3: CAPTION holder

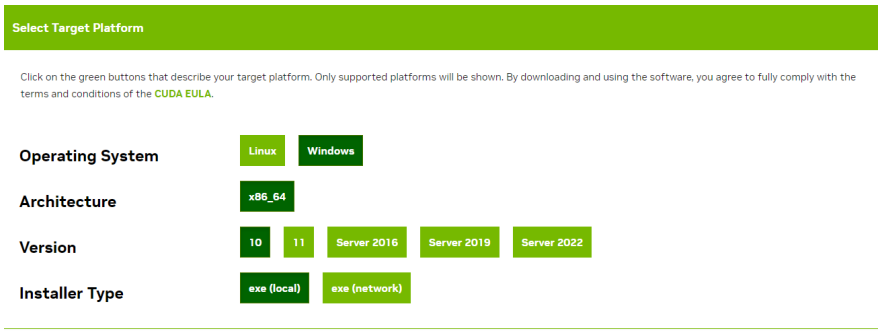

Figure 1.4: CAPTION holder

在这一步完成后，我们打开终端输入`nvcc -V`以及`nvidia-smi`应当分别能看到图1.5和图1.6类似的输出，这说明我们安装完成。

```
(base) PS C:\Users\MMLab_Cantjie> nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Tue_Mar__8_18:36:24_Pacific_Standard_Time_2022
Cuda compilation tools, release 11.6, V11.6.124
Build cuda_11.6.r11.6/compiler.31057947_0
```

Figure 1.5: caption:nvcc-v-install-success

```
(base) PS C:\Users\MMLab_Cantjie> nvidia-smi
Sun Mar  5 11:03:16 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 531.18                 Driver Version: 531.18       CUDA Version: 12.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name                     TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf           Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                                        |                      |               MIG M. |
|=======================================+======================+======================|
|   0  NVIDIA GeForce RTX 2060       WDDM | 00000000:01:00.0  On |                  N/A |
| 45%   36C    P8              13W / 160W|   1071MiB /  6144MiB |      5%      Default |
|                                        |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      6964    C+G   C:\Windows\explorer.exe                 N/A      |
```

Figure 1.6: caption:nvidia-smi-install-success

1.1.2 安装 Anaconda

我们可能同时有多个项目或作业在处理，而不同的项目或作业可能使用了不同 python 版本、不同的工具包等，为了避免冲突，我们通常会为每一个项目或作业指定一个虚拟环境，以使得各个环境之间互不干扰。为此，我们 Anaconda 以创建并管理虚拟环境。

安装过程参考官网文档即可：https://docs.anaconda.com/anaconda/install/windows/

安装完成后启动终端，输入`conda -V`，如正确显示 conda 版本则说明安装成功。

```
(base) PS C:\Users\MMLab_Cantjie> conda -V
conda 22.9.0
```

Figure 1.7: CAPTION holder

1.1.3 创建虚拟环境并安装 PyTorch

安装完成 conda 后，我们新建一个预装了 Python 的、用来完成本门课程的虚拟环境。

需要注意的是，PyTorch 和 Python 版本也需要对应，在https://github.com/pytorch/vision#installation中，我们发现 torch 1.13 要求 python 介于 3.7.2 和 3.10 之间。

torch	torchvision	python
main / nightly	main / nightly	>=3.8, <=3.10
1.13.0	0.14.0	>=3.7.2, <=3.10
1.12.0	0.13.0	>=3.7, <=3.10
1.11.0	0.12.0	>=3.7, <=3.10
1.10.2	0.11.3	>=3.6, <=3.9
1.10.1	0.11.2	>=3.6, <=3.9

Figure 1.8: CAPTION holder

打开终端，输入下面命令以利用 conda 新建环境，

```
$ conda create --name <envname> python=3.9
$
```

将其中 <envname> 改成自定义的环境名称，如助教自己选择的 distributedml。

新建完成后，通过 `conda activate <envname>` 进入环境。在 pytorch 官网安装页面https://pytorch.org/get-started/locally/选择对应的 pytorch 版本、系统版本等，复制给出的命令并运行。

安装完成后，进入 Python 就可以 `import torch` 了，如图1.10.

1.2 使用虚拟环境与本地 GPU

上面的本地环境配置不可为不复杂，CUDA、显卡型号、显卡驱动、PyTorch、Python 等版本需要手动一一对应起来安装。那有没有什么更简单的利用本机 GPU 计算资源的方法呢?

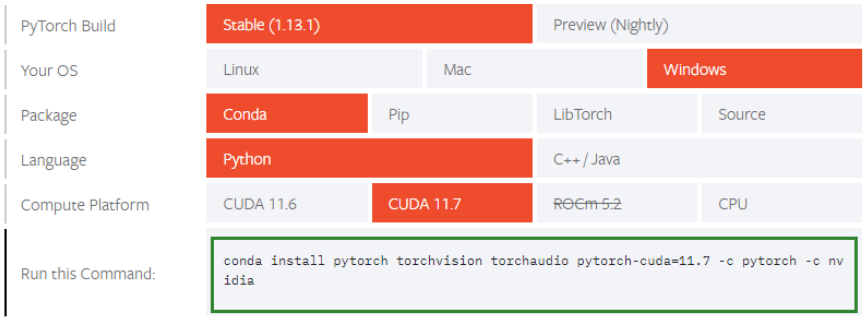

Figure 1.9: CAPTION holder

```
(distributedml) PS C:\Users\MMLab_Cantjie> python
Python 3.9.16 (main, Jan 11 2023, 16:16:36) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.__version__
'1.13.1'
>>>
```

Figure 1.10: caption:pytorch-install-success

在这一节，我们介绍直接利用 Docker 镜像搭配环境的方法。

1.2.1 安装并配置 Docker 引擎

首先在官网下载安装包https://docs.docker.com/desktop/install/windows-install/，安装过程略。

在安装完成后启动 Docker Desktop，在 windows 下，很可能会报错（具体内容是啥助教忘了截图了），一般错误的原因是缺少 wsl2 和 hyper-v。

为了启用 hyper-v，在控制面板中按照图1.11中的操作选中 Hyper-V 并确定。

为了启用 wsl2，参考https://learn.microsoft.com/en-us/windows/wsl/install，在终端下输入`wsl --install`等待安装完成即可。

安装完成后启动 Docker Desktop，为了加速下载，可以按照图1.12所示方法为 Docker 指定国内镜像服务器，即在原本的配置中加入如下内容。

```
    "registry-mirrors": [
        "http://hub-mirror.c.163.com",
        "https://docker.mirrors.ustc.edu.cn",
        "https://registry.docker-cn.com"
    ]
```

启动终端，输入`docker --version`，如图1.13，正常返回 Docker 版本就说明安装成功

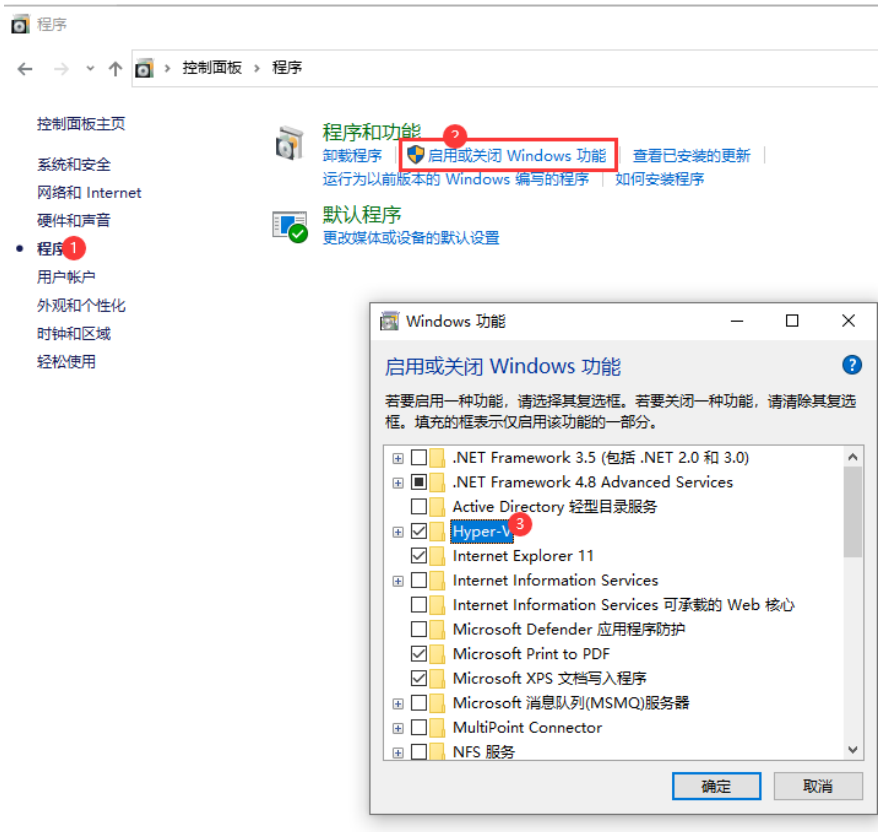

Figure 1.11: caption:turn-on-hyper-v

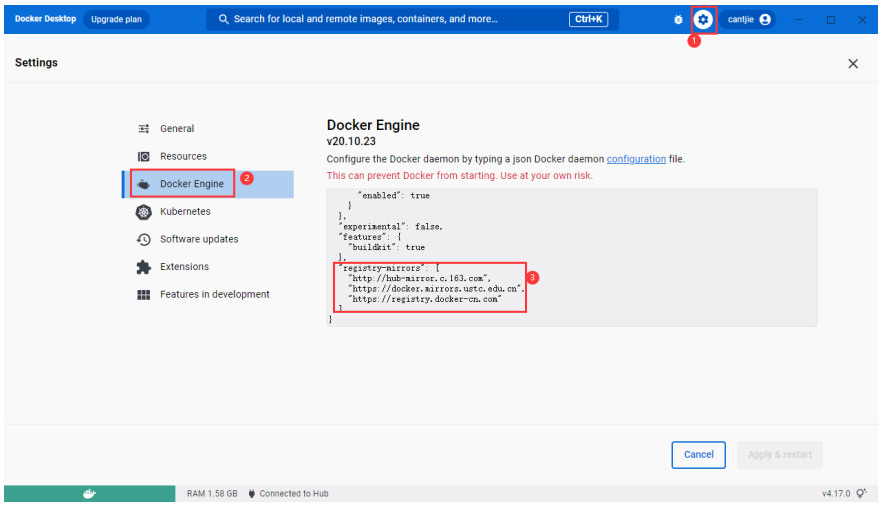

Figure 1.12: caption:docker-mirrors-setting

了。

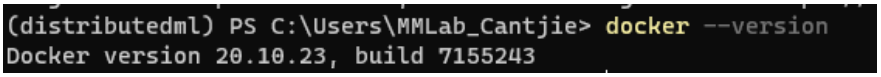

Figure 1.13: caption:docker-install-success

1.2.2 搜索并下载 PyTorch 镜像

Dockerhub 是一个共享镜像的平台https://hub.docker.com/。所谓镜像，类似于一个操作系统的 iso 文件：我们拿到 iso 文件后可以创建使用该操作系统的虚拟机；而当我们拿到镜像后，也可以利用该镜像创造一个使用该镜像的容器，即容器是一个镜像的实例。

因此，如果有人在某个容器中把 CUDA、PyTorch、Python 等环境都配置好，并打包成镜像共享给我们，我们就可以免去复杂的安装过程，从而直接使用镜像生成容器，在容器中直接运行我们所写的脚本。

在 DockerHub 中，我们搜索 `pytorch/pytorch`，可以找到对应的这个镜像https://hub.docker.com/r/pytorch/pytorch。点击网页中的 Tags 标签页，我们可以从图1.14看到这个镜像就是已经把 PyTorch 和 CUDA 安装好了的，我们直接使用这个镜像就好啦！

下载这个镜像前，还需要登录的。首先去注册个账号，然后打开终端，输入 `docker login` 登录。

然后就可以通过这条命令下载这个镜像了：

```
1    $ docker pull pytorch/pytorch:1.13.1-cuda11.6-cudnn8-runtime
2    $
```

这个镜像比较大，下载需要一点时间。完成后，我们再输入 `docker image list` 就可以看到这个镜像了，见图1.15。

1.2.3 启动容器

下载完镜像，我们该通过这个镜像启动一个容器了，我们需要到容器里看看这个容器里面是不是有我们需要的环境。

打开终端，输入

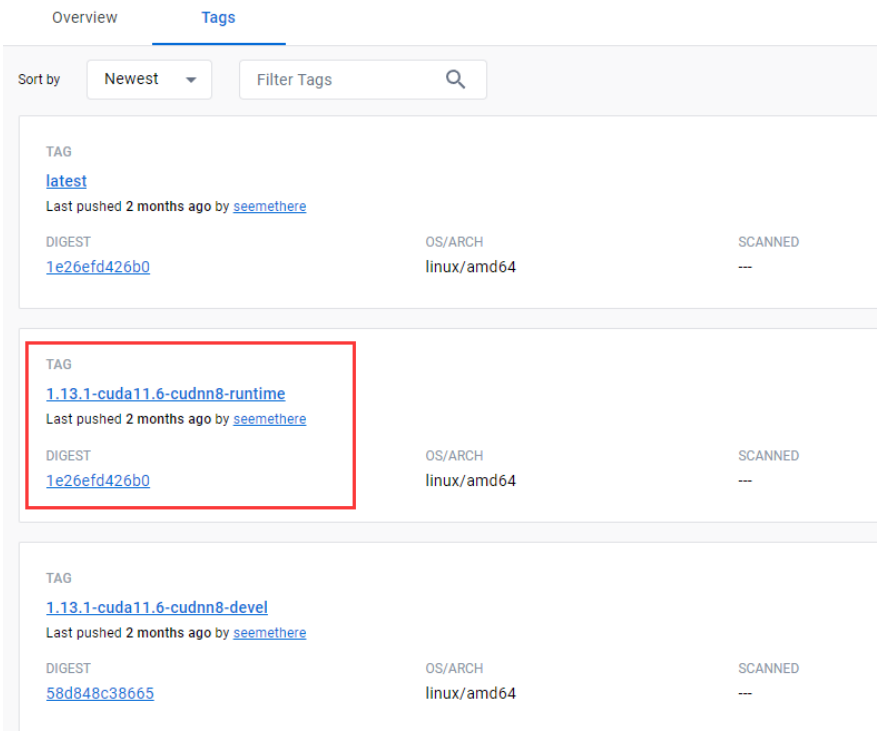

Figure 1.14: caption:pytorch-image-tags-web

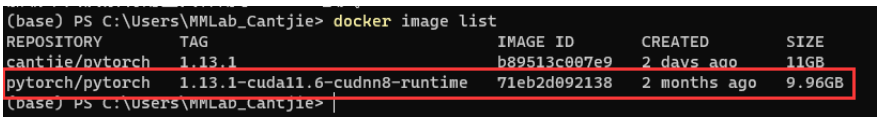

Figure 1.15: caption:docker-image-list-pytorch

```
$ docker run -it pytorch/pytorch:1.13.1-cuda11.6-cudnn8-runtime
```

我们发现我们进入了一个 linux 系统，进去运行一下`nvidia-smi`试试，诶，怎么 command not found，看不到显卡。这是因为容器启动时没有给他指定 GPU。我们输入`exit`，然后加上 GPU 参数再试一下

```
$ docker run --gpus all -it pytorch/pytorch:1.13.1-cuda11.6-cudnn8-runtime
```

进入容器后，我们输入`nvidia-smi`等命令，查看运行结果，如图1.16所示，发现正是我们所需要的环境。

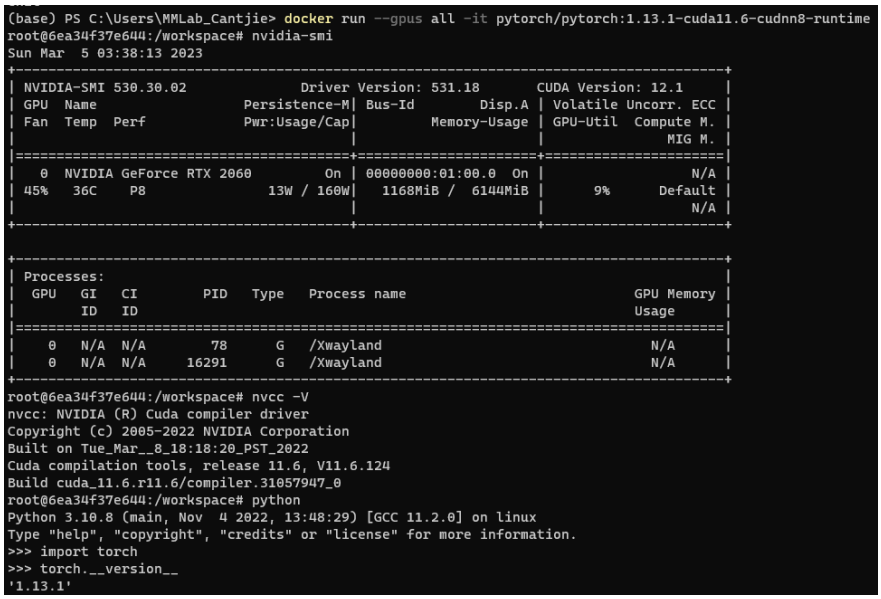

Figure 1.16: caption:docker-pytorch-container-env-check

可是如何使用这个环境呢，我们留到完成具体实验内容的时候再来讲。

1.2.4 限制 Docker 内存占用 (TODO，可选)

TODO

1.3 华为云计算资源

```
    some python code here?
            return model
```

Listing 1.1: Python example