

A Trick of Using Tensorflow in Remote Desktop

文档时间: 2018 年 10 月 20 日 17:03:00

适用平台: Unix-like OS



v-slam 研究小组

1 软件平台推荐

Windows 平台推荐使用 WinSCP 和 PuTTY 登录到远程机上, WinSCP 可以用来下载和上传文件, PuTTY 可以在远程机上执行指令。

- WinSCP: <https://winscp.net/eng/download.php>
- PuTTY: <https://www.putty.org/>

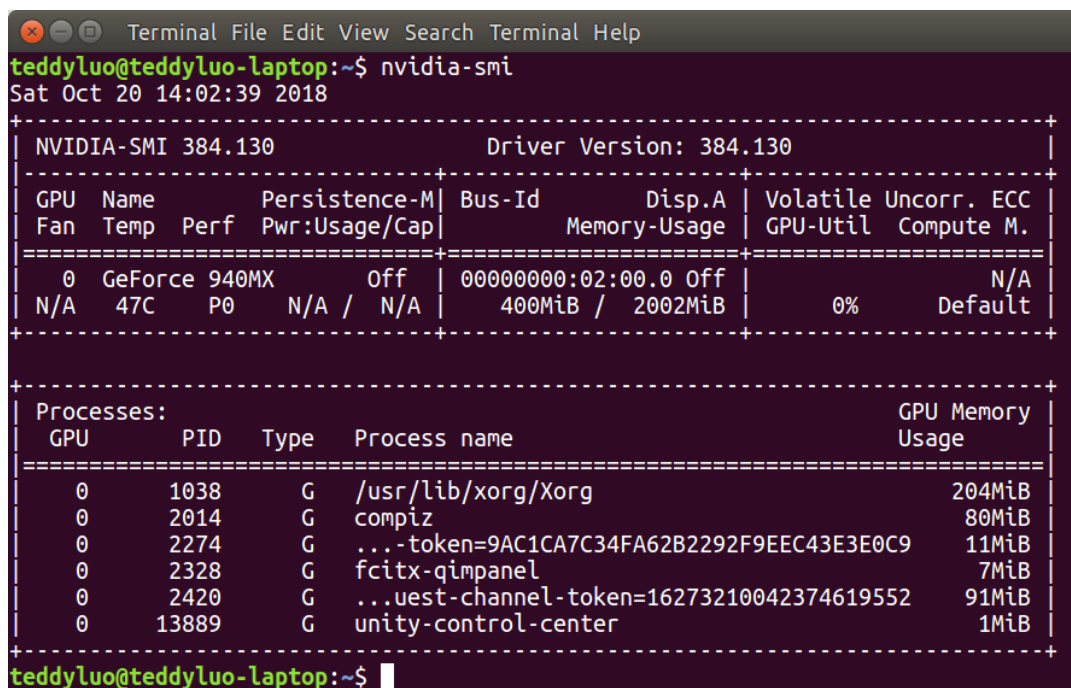
使用前确保远程机的 ssh 服务已打开。

2 在远程机上指定特定的 GPU 执行 python 脚本

两种方式实现, 一种是通过 bash 的命令行参数, 一种是在 python 的主函数里添加环境代码。
查看可用的 GPU 及其索引号 (index) 可用如下指令:

```
1 | nvidia-smi
```

例如图1仅有一个 GPU, 其 GPU 的 index 为 0。



```
teddyluo@teddyluo-laptop:~$ nvidia-smi
Sat Oct 20 14:02:39 2018

+-----+
| NVIDIA-SMI 384.130                  Driver Version: 384.130          |
+-----+-----+
| GPU   Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|  0  GeForce 940MX      Off          | 00000000:02:00.0 Off |          0%      N/A |
|N/A   47C    P0      N/A /  N/A  |  400MiB /  2002MiB |              Default  |
+-----+-----+

+-----+
| Processes:                       GPU Memory |
|  GPU       PID    Type    Process name      Usage  |
+-----+-----+
|    0      1038     G   /usr/lib/xorg/Xorg        204MiB |
|    0      2014     G   compiz                80MiB |
|    0      2274     G   ...-token=9AC1CA7C34FA62B2292F9EEC43E3E0C9  11MiB |
|    0      2328     G   fcitx-qimpanel         7MiB |
|    0      2420     G   ...uest-channel-token=16273210042374619552  91MiB |
|    0     13889     G   unity-control-center    1MiB |
+-----+

teddyluo@teddyluo-laptop:~$
```

图 1: nvidia-smi 的示例输出结果

2.1 通过 bash 命令指定计算 GPU

执行指令时加入 `CUDA_VISIBLE_DEVICES=GPU_Index_Number` 参数。

假设有 4 个 GPU, index 分别为 $[0, \dots, 3]$, 执行的脚本主程序为 `my_script1.py`、`my_script2.py` 和 `my_script3.py`, 则在特定的 GPU 上执行的特定的脚本的命令为:

```

1  # Uses GPU 0
2  $ CUDA_VISIBLE_DEVICES=0 python my_script1.py
3  # Uses GPU 1:
4  $ CUDA_VISIBLE_DEVICES=1 python my_script2.py
5  # Uses GPUs 2 and 3:
6  $ CUDA_VISIBLE_DEVICES=2,3 python my_script3.py

```

2.2 直接在 python 脚本中指定计算的 GPU

直接在 `my_script1.py`、`my_script2.py` 和 `my_script3.py` 的引入库部分声明使用的 GPU 索引号。例如在 `my_script1.py` 中添加如下代码:

```

1  .....
2  import os
3  if True:
4      os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
5      os.environ["CUDA_VISIBLE_DEVICES"]="0"
6      #for debugging
7      from tensorflow.python.client import device_lib
8      print(device_lib.list_local_devices())
9  .....

```

my_script1.py

而 `my_script3.py` 添加代码为:

```

1  .....
2  import os
3  if True:
4      os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
5      os.environ["CUDA_VISIBLE_DEVICES"]="2,3"
6      #print device information
7      from tensorflow.python.client import device_lib
8      print(device_lib.list_local_devices())
9  .....

```

my_script3.py

3 在远程机上执行 Python 脚本

远程机上执行程序时, 结果可回传到 client 机上交互; 这种回传有时并不需要。

3.1 实时回传结果到客户机上

这种情况很简单, 直接在 PuTTY 执行命令本身即可。例如,

```

1  python my_script1.py

```

3.2 挂载到远程机上不影响 client 的终端，标准输出保存到标准文件

意思是执行命令后不影响 client 的终端，client 的终端可以关掉，也可以干别的事。

使用 `nohub... &` 命令，例如：

```
1 | nohub python my_script1.py &
```

其终端的输出会在同目录下的一个叫 `nohub.out` 的文件里。

当 client 注销终端重新登录后，可用如下命令查找运行的后端进程：

```
1 | ps aux | grep 文件名或用户名
```

3.3 挂载到远程机上不影响 client 的终端，标准输出保存到自定义文件

```
1 | nohub python my_script1.py > custom-out.log &
```

其终端的输出会在同目录下的一个叫 `custom-out.log` 的文件里；若文件名带完整路径，则输出会改变到相应目录。

3.4 挂载到远程机上不影响 client 的终端，忽略标准输出

```
1 | nohub python my_script1.py >/dev/null 2>&1 &
```

其中参数 2 表示错误输出也重定位到 1 的输出结果里（标准输出）。

注意文件 `my_script1.py` 里不应该有与终端交互的代码。如果有交互代码，例如终端需要用户输入一个字母或别的才能继续往下跑，此时进程会被输入挂起而不能完整地往下跑。所以比较安全的做法是在运行脚本时要忽略交互输入，用下面的指令会将终端交互忽略，完整地跑完整个程序。

推荐做法：

```
1 | nohub python my_script1.py </dev/null >/dev/null 2>&1 &
```

上述命令的参数可混合使用。例如如果需要将标准输出和错误输出分开，而忽略标准交互，则命令为：

```
1 | nohub python my_script1.py >/dev/null 2> errInfo.err < /dev/null &
```