基于WSL2的GPU加速部署及测试实验

1. 实验目的

本实验旨在帮助用户掌握在WSL2（Windows Subsystem for Linux 2）环境下配置和使用GPU加速计算的方法，具体目标包括：

1. 熟悉WSL2环境的配置和切换方法；
2. 掌握在WSL2中安装和配置CUDA工具包的流程；
3. 学习如何在WSL2中部署GPU加速的Python环境；
4. 验证GPU相对于CPU在矩阵计算等任务中的性能优势；
5. 了解Docker容器中调用GPU进行加速计算的方法；
6. 为后续在RflySim平台上进行GPU加速仿真计算打下基础。

2. 实验要求

- 软件要求：Windows 11 22H2及以上版本；RflySim工具链 v4.10及以上版本[1]。
- 硬件要求：笔记本/台式电脑1台[2]。

3.实验地址

例程目录：
[安装目录]\RflySimAPIs\1.RflySimIntro\2.AdvExps\e11.WSL2_GPUAccConfig

- gpu_test.py：核心测试程序，用于验证GPU加速效果，包含系统信息打印模块、矩阵乘法测试模块等功能。
- WinWSL.bat：Windows批处理脚本，用于进入WSL2的Ubuntu环境。
- cuda12_4.sh：一键安装脚本，用于自动安装CUDA12.4.1并安装GPU版本的torch。
- OfflineInstall.sh：离线安装脚本，用于离线部署GPU加速环境。

4. 实验内容或步骤

4.1 步骤1：RflySim环境切换

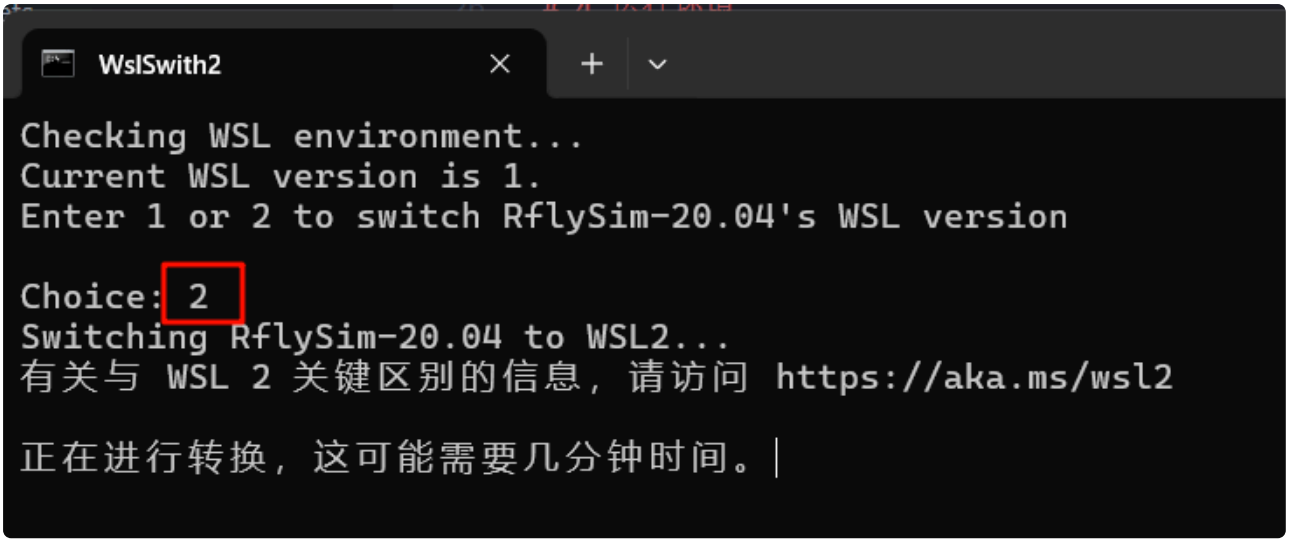
目前主要有两种方法能够将WinWSL环境切换到WSL2形式。

1）安装时切换：本实验中是基于RflySim v4.10及以上版本工具链进行，在第4项，直接选2即可

该版本中内置WSL 2的编译器功能，若首次安装时已经选择为WSL系列的编译器，可在Windows系统终端中输入 `wsl -l -v` 查看当前WSL的版本。若为WSL1，则需要切换为WSL2。

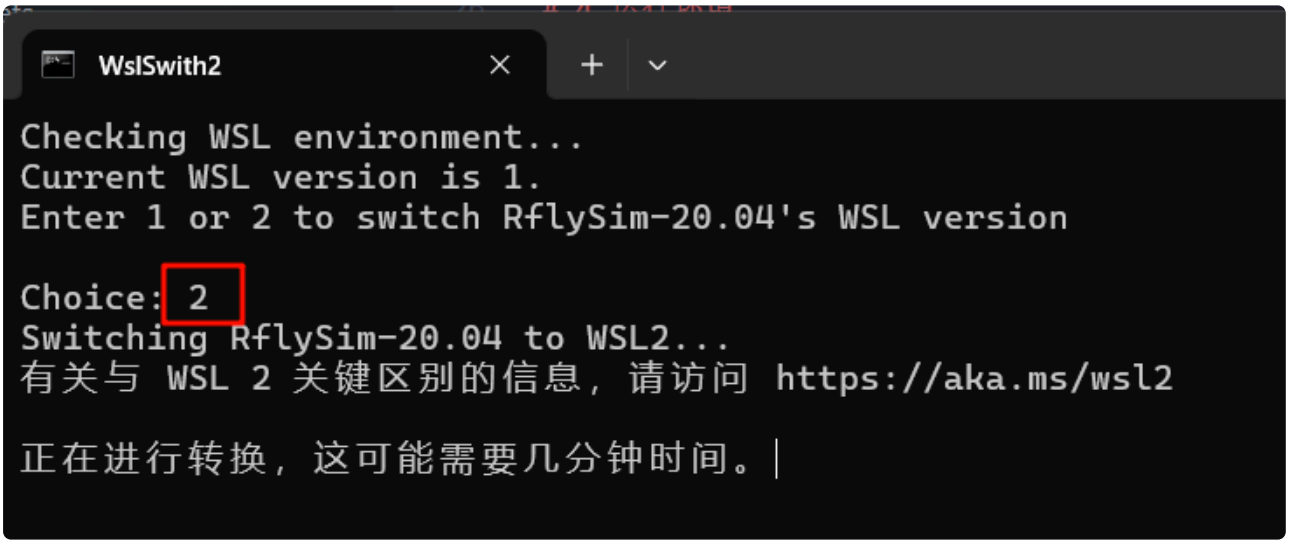
注：本方法速度较快，但是会删除镜像中的所有文件，并还原。若其中已经包含了开发的源码，请提前备份再操作，或者按后文执行手动切换，但是这种方法通常切换速度较慢，可能需要半个小时以上，请耐心等待。

2）安装后手动切换：切换方式为双击 `*\Desktop\RflyTools\WslSwith2.lnk` ,在弹出的窗口中输入"2"，即可自动切换为WSL2。



切换成后关闭窗口即可，可在Windows系统终端中输入`wsl -l -v`查看当前WSL的版本，验证是否切换成功。

若在安装的时候选择的是非WSL的编译器（如：Cygwin编译器），则需要在安装环境的时候，进行如下设置：



并点击"确定"即可部署成功。

4.2 步骤2：CUDA-GPU加速环境部署（在线部署）

本实验主要演示基于英伟达显卡的CUDA加速测试，因此需要安装CUDA环境。由于CUDA安装包较大（约5G），安装之前，我们可进入WSL2环境中进行初步测试，双击WinWSL.bat脚本进入WSL2的Ubuntu环境中，并运行以下命令进行测试：

```
python3 gpu_test.py
```

测试结果如下：

```
PUAccConfig# python3 gpu_test.py
系统信息：
操作系统： Linux
Python版本： 3.10.12 (main, Aug 15 2025, 14:32:43) [GCC 11.4.0]
PyTorch版本： 2.8.0+cpu
CUDA是否可用： False

开始矩阵乘法测试...

使用矩阵大小： 20000x20000
创建CPU矩阵...
执行CPU计算...
CPU计算耗时： 26.5878 秒
```

> **注**: 如需要安装指定版本的CUDA可参考一键安装脚本： cuda12_4.sh, 该脚本将自动安装CUDA12.4.1，并安装GPU版本的torch。

显示CUDA不可用，需要单独部署安装，安装方式如下，在WSL 2的Ubuntu终端输入：

```
apt-get update
apt-get upgrade -y
apt install nvidia-cuda-toolkit -y
```

安装时间较长，请耐心等待...。安装完成后，在运行：

```
# 卸载原版本中内置的CPU版本torch
pip uninstall torch torchvision torchaudio -y

# 安装GPU版本torch
pip install torch torchvision torchaudio
```

等待安装完成后，直接在来运行测试程序，在当前终端中输入`python3 gpu_test.py`,将显示如下测试结果。

```
root[redacted]:/mnt/d/RflySim/RflySimAPIs/1.RflySimIntro/2.AdvExps/e11_WSL2_GPUAccConfig# python3 gpu_test.py
系统信息：
操作系统：Linux
Python版本：3.10.12 (main, Aug 15 2025, 14:32:43) [GCC 11.4.0]
PyTorch版本：2.8.0+cu128
CUDA是否可用：True
CUDA版本：12.8
GPU设备数量：1
GPU 0: NVIDIA GeForce RTX 3070
GPU 0 显存总量：8.00 GB
GPU 0 当前可用显存：0.00 MB

开始矩阵乘法测试...

使用矩阵大小：20000x20000
创建CPU矩阵...
执行CPU计算...
CPU计算耗时：29.8947 秒

转移数据到GPU...
预热GPU...
执行GPU计算...
GPU计算耗时：2.1156 秒
GPU加速比：14.13x
```

可以看到CUDA已经可用，当前本机中的CUDA版本为12.8，GPU为NVIDIA GeForce RTX 3070。测试程序中CPU计算耗时为29.8947秒，GPU计算耗时为2.1156秒，GPU相对于CPU的加速比为14.13。

4.3 步骤3：Docker环境下的GPU加速验证

WSL 2中已经内置了Docker，可运行：

```
systemctl start docker #启动Docker服务

systemctl status docker #查看Docker服务状态
```

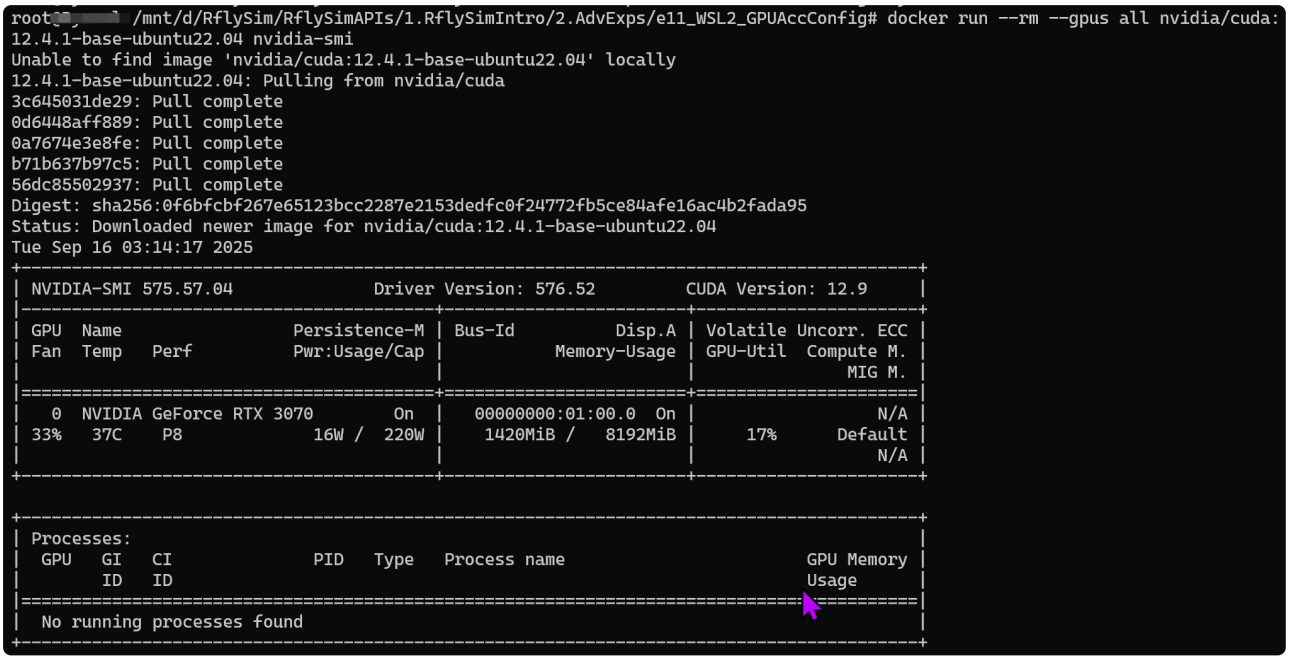
```
root[redacted]:/mnt/d/RflySim/RflySimAPIs/1.RflySimIntro/2.AdvExps/e11_WSL2_GPUAccConfig# systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; disabled; vendor preset: enabled)
     Active: active (running) since Tue 2025-09-16 11:10:55 CST; 5min ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 545 (dockerd)
      Tasks: 18
     Memory: 369.6M
        CPU: 5.422s
     CGroup: /system.slice/docker.service
             └─545 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Sep 16 11:10:52 Byzeal dockerd[545]: time="2025-09-16T11:10:52.536728507+08:00" level=info msg="Loading containers: start."
Sep 16 11:10:55 Byzeal dockerd[545]: time="2025-09-16T11:10:55.079245882+08:00" level=warning msg="Error (Unable to complete >
Sep 16 11:10:55 Byzeal dockerd[545]: time="2025-09-16T11:10:55.526106757+08:00" level=info msg="Loading containers: done."
Sep 16 11:10:55 Byzeal dockerd[545]: time="2025-09-16T11:10:55.589773422+08:00" level=info msg="Docker daemon" commit=bea959c>
Sep 16 11:10:55 Byzeal dockerd[545]: time="2025-09-16T11:10:55.590402214+08:00" level=info msg="Initializing buildkit"
Sep 16 11:10:55 Byzeal dockerd[545]: time="2025-09-16T11:10:55.611842185+08:00" level=info msg="Completed buildkit initializa>
Sep 16 11:10:55 Byzeal dockerd[545]: time="2025-09-16T11:10:55.614481223+08:00" level=info msg="Daemon has completed initiali>
Sep 16 11:10:55 Byzeal dockerd[545]: time="2025-09-16T11:10:55.614547940+08:00" level=info msg="API listen on /run/docker.soc>
Sep 16 11:10:55 Byzeal systemd[1]: Started Docker Application Container Engine.
Sep 16 11:14:17 Byzeal dockerd[545]: time="2025-09-16T11:14:17.367166720+08:00" level=info msg="ignoring event" container=24c>
lines 1-22/22 (END)
```

> **注：若需要经常使用Docker服务，可在终端中运行 `systemctl enable docker` 设置Docker服务为开机自启。**

确认Docker启动成功后，运行：

```
# 验证容器能否调用 GPU进行计算
docker run --rm --gpus all nvidia/cuda:12.4.1-base-ubuntu22.04 nvidia-smi
```

若最终显示。则表示调用GPU成功。



> **注：本实验中Docker中的CUDA版本为12.4.1，不同电脑中的GPU版本不同，可直接运行 `nvidia-smi` 查看支持的CUDA版本(右上角CUDA Version: X.X.X)。根据实际情况修改。**

4.4 步骤4：CUDA-GPU加速环境部署（离线部署）

RflySim自带的WSL编译器，为了避免安装包过大，精简了GPU加速、大模型训练等模块，只能进行基于CPU的图像识别等。本增量包是为了解决此问题，提供了更完整的开发环境。

请参考例程：`[安装目录]\RflySimAPIs\1.RflySimIntro\2.AdvExps\e13.WinWSL2-GPU\Readme.pdf` 来进行安装。

5. 关键知识点

关键知识点1：WSL2架构与GPU直通原理

WSL2采用轻量级虚拟机技术实现Linux内核兼容，通过GPU直通技术（GPU-PV）使Linux应用程序能够直接访问Windows系统中的GPU硬件。这种架构使得在WSL2中运行的深度学习框架能够充分利用NVIDIA GPU的计算能力。

关键知识点2：CUDA环境配置

CUDA是NVIDIA开发的并行计算平台和编程模型，通过在WSL2中安装CUDA工具包，可以为深度学习框架（如PyTorch、TensorFlow）提供GPU加速支持。配置过程包括安装CUDA工具链和相应的GPU驱动支持。

关键知识点3：PyTorch GPU加速机制

PyTorch通过CUDA接口将张量计算任务分发到GPU执行。在代码中通过简单的API调用（如 `.cuda()` 或 `.to('cuda')` ）即可将计算任务从CPU转移到GPU，大幅提升矩阵运算等并行计算密集型任务的执行效率。

关键知识点4：gpu_test.py程序详解

gpu_test.py是本实验的核心测试程序，用于验证GPU加速效果。程序主要包含以下功能模块：

1. **系统信息打印模块**（`print_system_info`函数）：
   该函数通过调用`torch.cuda.is_available()`检测CUDA是否可用，并在可用时显示GPU设备信息。
2. **矩阵乘法测试模块**（`matrix_multiply_test`函数）：
   该函数执行CPU和GPU上的矩阵乘法操作，并计算加速比。关键代码点包括：
   - `torch.randn(size, size)`：创建指定大小的随机矩阵
   - `torch.mm(matrix1, matrix2)`：执行矩阵乘法操作
   - `matrix.cuda()`：将张量从CPU转移到GPU
   - `torch.cuda.synchronize()`：确保GPU计算完成后再计时
   - `cpu_time/gpu_time`：计算GPU相对CPU的加速比
3. **程序入口**：
   程序入口首先打印系统信息，然后执行矩阵乘法测试。

程序通过对比大规模矩阵运算在CPU和GPU上的执行时间，直观展示GPU加速带来的性能提升。

关键知识点5：Docker GPU支持

NVIDIA Container Toolkit为Docker容器提供GPU支持，通过`--gpus`参数使容器内的应用程序能够访问宿主机的GPU资源，实现容器化环境中的GPU加速计算。

6.参考资料

1. NVIDIA CUDA Documentation
2. Windows Subsystem for Linux Documentation
3. PyTorch Documentation
4. Docker Documentation
5. NVIDIA Container Toolkit
6. RflySim官方文档

7.常见问题

Q1：WSL2中CUDA不可用怎么办?

A1：请确认以下几点：

1. 确保Windows系统中已安装NVIDIA显卡驱动；
2. 在WSL2中安装CUDA工具包：`apt install nvidia-cuda-toolkit -y`；
3. 重新安装支持GPU的PyTorch版本：

```
pip uninstall torch torchvision torchaudio
pip install torch torchvision torchaudio
```

4. 如果仍然不可用，尝试重启WSL2：`wsl --shutdown`然后重新启动。

Q2：GPU计算时报错"显存不足"如何解决?

A2：当执行大规模矩阵运算时，可能会遇到显存不足的问题。可以通过以下方式解决：

1. 减小测试矩阵的大小，修改gpu_test.py中的参数：

```
# 原来的设置
matrix_multiply_test(20000)
# 改为更小的值
matrix_multiply_test(10000)
```

2. 确保及时清理GPU内存，在计算完成后调用：

```
torch.cuda.empty_cache()
```

Q3：Docker中无法调用GPU怎么办？

A3：请确认以下几点：

1. 确保已安装nvidia-docker2：

```
apt install nvidia-docker2 -y
systemctl restart docker
```

2. 确保Docker命令中正确使用了GPU参数：

```
docker run --rm --gpus all nvidia/cuda:12.4.1-base-ubuntu22.04 nvidia-smi
```

3. 检查CUDA版本兼容性，根据实际环境调整镜像版本。

---

1. https://rflysim.com/ ↩
2. 推荐配置请见： https://rflysim.com/doc/zh/HowToInstall.pdf ↩