

实验3：Linux环境下调试与矩阵乘法优化

1. 实验目的

- (1) 进一步熟悉linux环境使用
- (2) 掌握linux下C语言开发常用的调试方法
- (3) 了解矩阵乘法优化基本原理

2. 实验内容

- (1) 阅读框架lab3/how-to-optimize-gemm代码，修改矩阵规模使得最大能跑到1024，最少跑4个不同size的数据，间距inc如有需要可以自行调整，默认从PFIRST开始每次翻倍。
- (2) 阅读代码，将openblas的实现集成到框架代码，与MMult0.c分析对比运行结果，注意需要修改makefile。

3. 框架代码介绍

test_MMult.c是程序入口，分配存储空间、初始化矩阵、矩阵乘，并与参考实现（REF_MMult）进行结果对比，MMult0.c的实现完全同REF_MMul.c，最后计算gflops值。

3.1 新增实现

makefile中的最开始的OLD 和NEW两个变量指定了对比的版本，版本的名称需与文件名相同，MMult0对应MMult0.c文件。

首次运行时，OLD 和 NEW 都赋值MMult0，生成的output_new.m和output_old.m文件内容相同。

```
OLD    := MMult0
NEW    := MMult0
```

`make run` 最终只会执行NEW变量对应文件中的MY_MMult函数进行的矩阵乘。

新增优化实现时，新建任意c文件xx.c，实现MY_MMult函数，再将makefile中NEW变量替换为c文件名xx。命名尽量能提现优化方法。

比如MMult1.c是另外一个实现，实现逻辑完全同MMult0.c，函数名也是MY_MMult函数，只是加了一行fprintf，将文件名输出到标准错误输出（屏幕）。

再将makefile中的NEW变量修改为MMult1，再执行make run就会执行MMult1.c中的MY_MMult函数。

```
OLD := MMult0
NEW := MMult1
```

3.2 m文件的数据说明

前面两行version是代码版本，date是运行的时间。MY_MMult性能数据，每行3个值，第一个值是矩阵大小，第二个值是gflops值，第三个值是误差大小，需要留意误差的值，误差过大并不会直接终止，仍会正常执行，需要确保优化的版本功能是正确的。

对于结果校验，可以使用[矩阵乘法在线计算](#)进行验证（注意只计算A*B，没有加C），或用python验证。

3.3 性能曲线图

先执行`pip install matplotlib`安装matplotlib包。

再执行命令会`python plotFlops.py`画出gflops性能曲线图，plotFlops.py通过读取_data/output_old.m和output_new.m文件的数据进行绘制。对m文件中的数据格式有严格的要求，如果增加其他格式的数据，会导致读取错误。

横坐标的最大值即峰值性能需要手动配置，跟所用处理器的核数、频率、每个时钟周期执行的浮点操作数有关，计算公式如下： $\text{max_gflops} = \text{nflops_per_cycle} * \text{nprocessors} * \text{GHz_of_processor}$

可以根据需要调整参数，尤其核数的差异比较大，现在PC好一点的处理器有8个核甚至更多。

plotAll.py是用于最后的数据处理，进行了多次优化之后，收集每次的gflops值，最后绘制到一张图，需要手动添加数据。也可以通过Excel进行实验结果的处理分析。

3.4 WSL运行GUI

在wsl启动的ubuntu中运行`python plotFlops.py`理应会弹出一个图形窗口并显示一张图片，即图形用户界面。如果没有显示图片且命令行有错误提示，则表明当前的WSL不支持运行Linux GUI应用程序，需要做相关配置。

以管理员身份启动PowerShell，分别执行下面两条命令，先更新wsl，再关闭WSL。

```
wsl --update
wsl --shutdown
```

启动WSL的Ubuntu，切到框架代码的目录，再执行`python plotFlops.py`。如果还显示失败，则点击[wsl_graphics_support](#)下载wsl_graphics_support_x64.msi并安装，即手动安装离线版。

Linux GUI support 1.0.28

- weston : [46756d](#)
- FreeRDP : [5f083f](#)
- pulseaudio: [2f0f0b8c](#)








Changes:

Weston:

- enable allow_zap to be configured from .wslgconfig
- fix menu is invisible at fullscreen mode
- fix hangul/hanja keys are reversed on Korean 103/106 keyboard
- clipboard to handle same data source is requested repeatedly when RDP

▼ Assets 8

第一步点击展开

 system-distro-debuginfo-arm64.tar.gz	6.95 MB	Sep 16, 2021
 system-distro-debuginfo-x64.tar.gz	7.07 MB	Sep 16, 2021
 WSLDVCPlugin.ARM64.pdb.zip	1.11 MB	Sep 16, 2021
 WSLDVCPlugin.x64.pdb.zip	1.16 MB	Sep 16, 2021
 wsl_graphics_support_arm64.msi	116 MB	Sep 16, 2021
 wsl_graphics_support_x64.msi	127 MB	Sep 16, 2021
 Source code (zip)		Sep 15, 2021
 Source code (tar.gz)		Sep 15, 2021

第二步点击下载

👍 6 🗨 3 8 people reacted

如最终仍能显示图片，可以把代码中的# `fig.savefig("test.png")`注释去掉将图片保存到本地进行查看。

主要参考：[WSLg](#)

4. 实验报告要求

该实验暂无需写实验报告，在后续相关优化完成后做统一的汇总分析，但请记录本实验当中遇到的自己认为值得记录的问题，以及在后面要提交的报告中回答下面问题：

- 问题（1）：多个c代码中有相同的MY_MMult函数，怎么判断可执行文件调用的是哪个版本的MY_MMult函数？是makefile中的哪行代码决定的？
- 问题（2）：性能数据_data/output_MMult0.m是怎么生成的？c代码中只是将数据输出到终端并没有写入文件。