实验报告

课程：高性能计算应用实践

姓名：王峻阳

学号：220110317

学院：计算机科学与技术学院

学期：2023 年秋季学期

实验日期：2023 年 9 月 18 日

一、实验内容

内容一：使用 GDB 调试 C 程序。

学习了 GDB 的基本用法，包括设置断点、打印程序状态、修改程序状态三类操作。断点可以设置为在一定条件下暂停程序；可以打印的程序状态包括 C 变量和语句、寄存器的值、栈的相关信息等；某一内存地址的值、寄存器的值都可以修改。

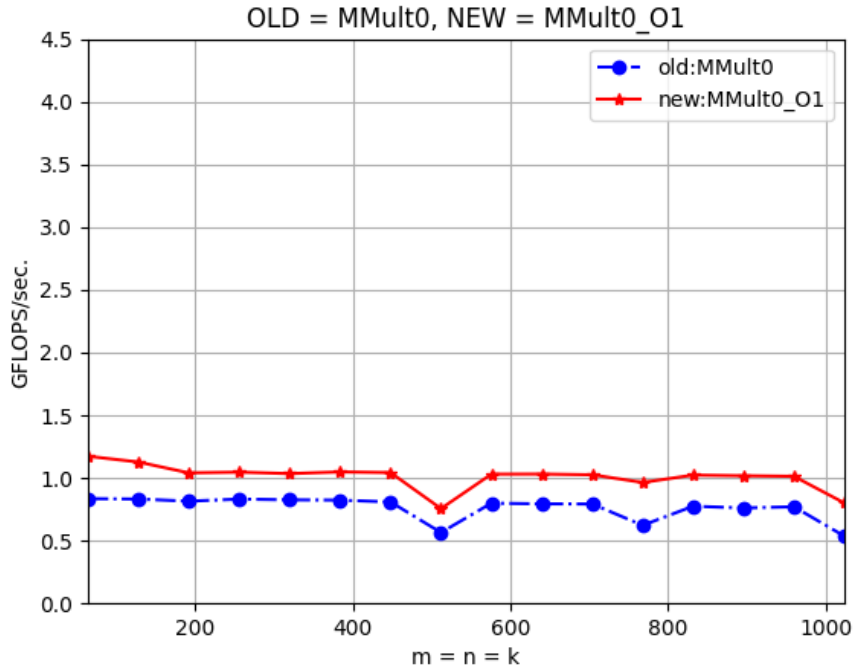
内容二：观察段错误（segmentation fault）。

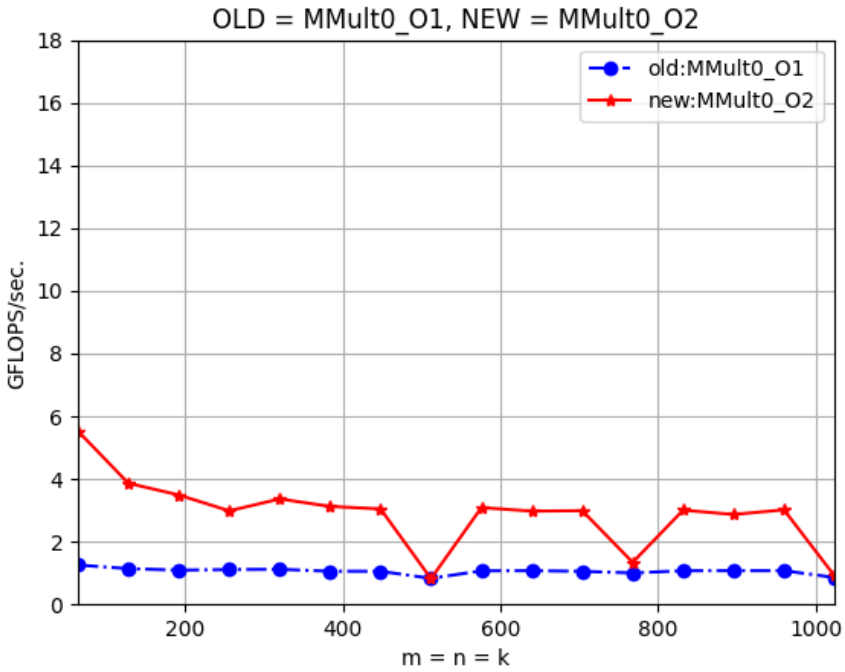
用 GDB 观察了数组越界、缓冲区溢出、栈溢出三种程序行为。观察到 `stackoverflow.c` 程序的栈会越堆越大，以及打开`-O2` 优化选项后将会优化掉 tail call 的调用函数的栈，栈区不会增大，不会发生 segfault。数组越界并不一定造成 segfault，一般是访问到不可访问的地址，或者触发了编译器提供的保护机制，才会产生 segfault。学习到用 GDB 的 `backtrace` 指令观察函数调用栈的方法。

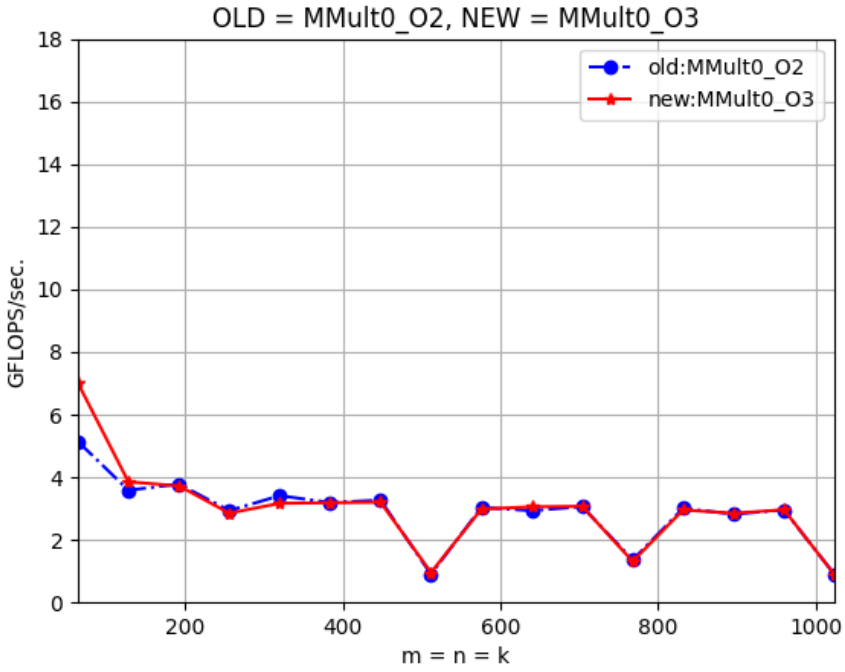
内容三：修改矩阵乘法代码框架。

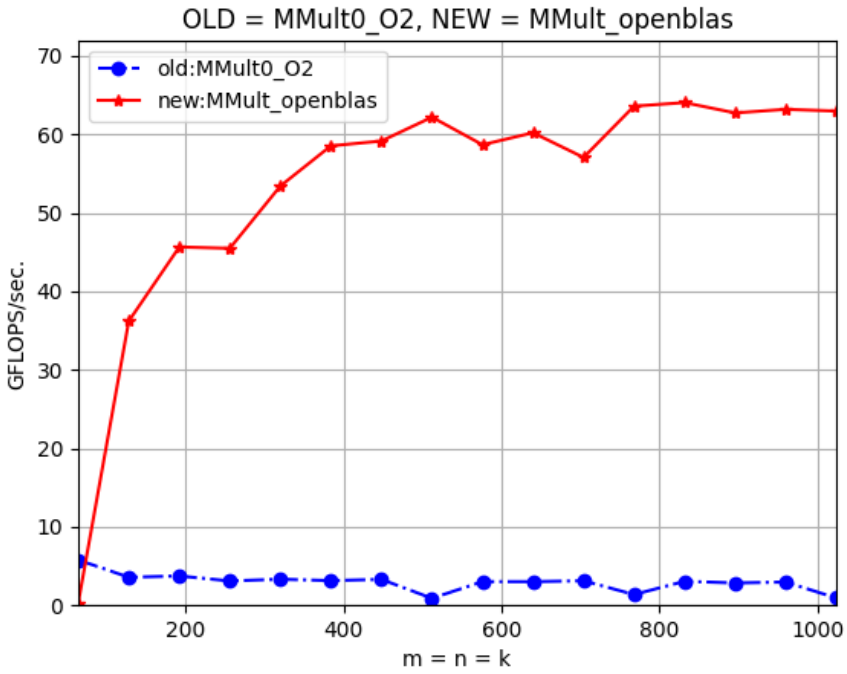
了解了 make 工具的思想和能力，学习了 `makefile` 的基本编写方法，修改了 `makefile` 的内容，以比较各种优化之间的性能差异。熟悉了用 `makefile` 构建程序的方法。

二、不同优化选项性能和 Openblas 性能对比









可见，O1 优化有一定效果，但效果不明显；O2 优化能再将运算速度提升两倍多，但在矩阵规模为 512、768、1024 时几乎没能进一步优化；O3 优化几乎没有更好的性能提升。Openblas 库提供的程序，运算速度能达到 O3 优化的 20 倍左右。

三、遇到的问题和解决办法

问题一：程序显示 `core dumped`，但 `core` 文件没有生成。

Ubuntu 似乎会将 `core` 文件存储在其他位置（`/mnt/wslg/dumps/`）。可以通过修改`/proc/sys/kernel/core_pattern` 文件的内容来在同一文件夹下获得 `core` 文件。

问题二：为什么在几种优化选项下都有几个突兀的性能低谷？

查询 CPU 的 cache 信息发现，512、768、1024 个双精度浮点数所相当的空间，刚好与 cache 的组数×块大小成比例，导致产生大量 cache miss，限制了优化效果。

问题三：在修改了矩阵乘法的循环次序后，观察到运算性能有明显变化。然而，GCC 的手册里写到，`-O3` 优化下会默认打开`-floop-interchange`，应该也会自动对循环次序进行优化，但实际上并没有。

暂未解决。

问题四：运行 `PlotAll.py`，在 `plt.show()`语句报错显示 `UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown`。

原因不明，可能是因为这条指令需要图形化界面，但 WSL 不支持。