High Performance Computing

2023 Fall

Lab 3. Debugging in Linux

Optimization of Matrix Multiplication

Yongzhuo Ma

September 22, 2023

Chapter 1

Introduction to the Environment

1.1 Host Machine

Item	Value
OS version	macOS Ventura 13.5.2
Apple clang version	14.0.3
CPU	Apple M2 Max
CPU Frequency	3.54 - 3.70 GHz
CPU Cores	12
Memory	64 GB

1.2 Virtual Machine

Item	Value
Virtualization	Parallels
OS version	Ubuntu 22.04.3 LTS
gcc version	11.4.0
CPU	Apple M2 Max
CPU Frequency	3.66 GHz
CPU Cores	8
Memory	48 GB

Chapter 2

Optimization of Matrix Multiplication on GCC Level

In this chapter, I will modify the matrix multiplication code templates, and compare the efficiency of different optimization choices.

Optimization: O0,O1,O2,O3

There are 4 common choices of code optimization for GCC to optimize the performance of the code given. They are O0,O1,O2,O3. O0 means no optimization. O1,O2,O3 will optimize the code. Generally, O3 will optimize the code further than O2, and the same for O2 than O1.

Here the math is,

$$\boldsymbol{A} = \boldsymbol{L} \times \boldsymbol{R} + \boldsymbol{C}$$

First, the header **parameters.h** should be modified.

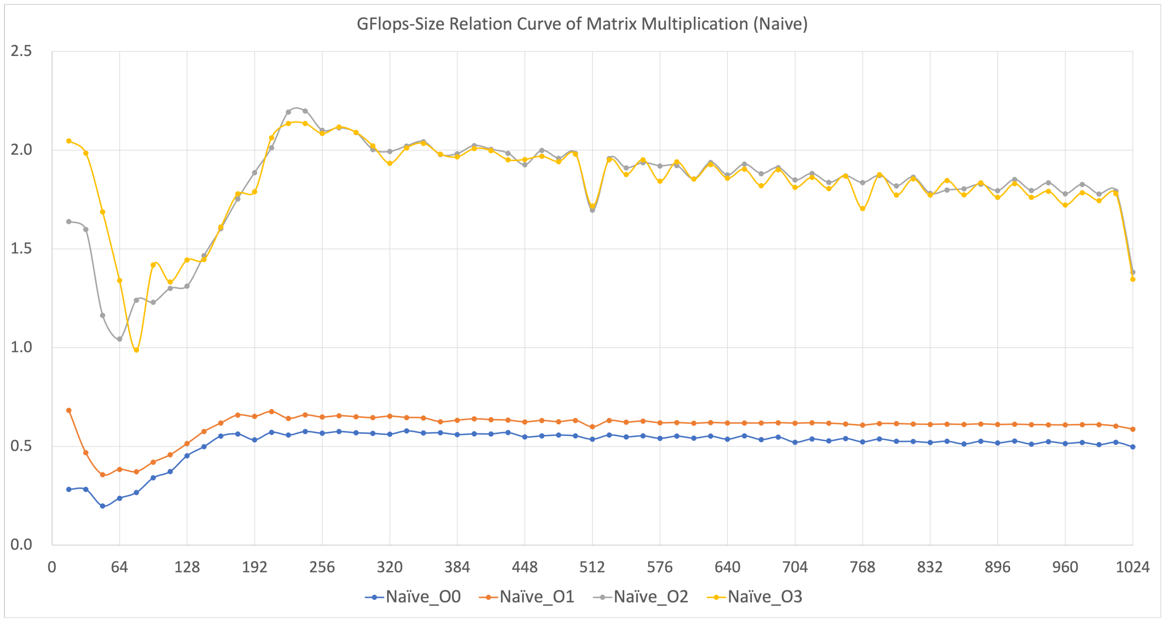
```
#define PFIRST 16
#define PLAST 1024
#define PINC 16
```

This means the test will start at the matrix size of 16×16 to 1024×1024, increased by 16 each time. Then **makefile** should be modified,

```
CFLAGS := -Wall -g -O2
```

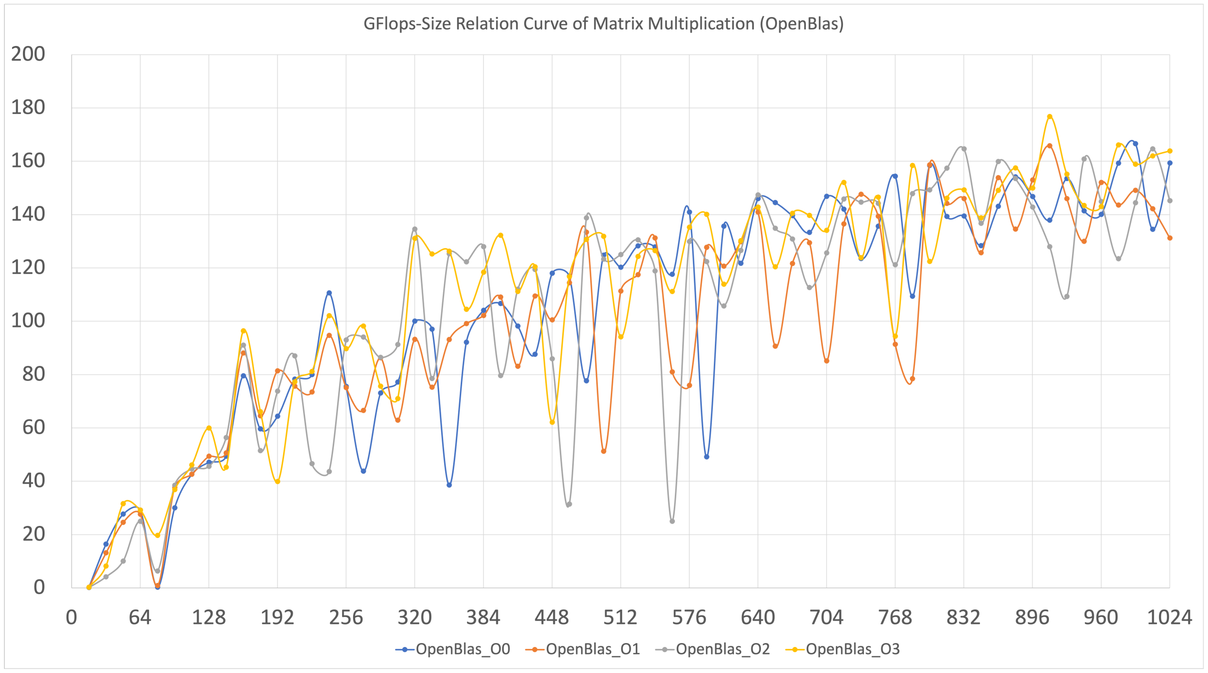
the parameter -O2 means the O2 optimization. Different ones like O1,O3 can also be the alternative parameters. Leaving out this parameter means O0, that is no optimization.

Collecting the data, we can generate a graph,



This graph shows that O1 did not do a great job optimizing this naive matrix multiplication code, but O2 and O3 did do a 3-time performance enhancement approximately.

Then, we try to optimize matrix multiplication with OpenBlas.
The results are given below,



Easy to mind that OpenBlas has an absolute advantage to the naive multiplication.
We can see O1,O2,O3 of this code did not give us an obvious acceleration.
We can guess that this code has been optimized already, and further optimization will not do much to the efficiency of this code.

Problems

When calling the function cblas_dgemm(), an error raised: undefined reference to cblas_dgemm.
This problem is tackled by using the absolute path of **cblas.h** but not the relative path.