# 作业一 编译测试

## 1. 代码

```
// main.cpp
#include "mul.hpp"
#include <time.h>

int main(){

    float* A = (float*)malloc(N * N * sizeof(float));
    float* B = (float*)malloc(N * N * sizeof(float));
    float* C = (float*)malloc(N * N * sizeof(float));

    clock_t s, e;

    Init(A);
    Init(B);
    Init(C);

    Mul(A, B, C);

    return 0;
}
```

```
// mul.hpp
#include <cstdio>
#include <cstdlib>
#include <random>
#include <iostream>

#define N 2048
#define FLOAT_MIN 1
#define FLOAT_MAX 100

void Init(float* M);

void Mul(float* A, float* B, float* C);
```

```cpp
#include "mul.hpp"

void Init(float* M){
    std::random_device rd;
    std::default_random_engine eng(rd());
    std::uniform_real_distribution<float> distr(FLOAT_MIN, FLOAT_MAX);
    for(int i=0; i<N*N; ++i){
        M[i] = distr(eng);
    }
}

void Mul(float* A, float* B, float* C){
    for(int i=0; i<N; ++i){
        for(int j=0; j<N; ++j){
            for(int k=0; k<N; ++k){
                C[i*N+j] += A[i*N+k] * B[k*N+j];
            }
        }
    }
}
```

# 2. 使用g++测试

1. 无编译选项 158.7s

```
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ g++ -o exp main.cpp mul.cpp
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ time ./exp

real    2m38.705s
user    2m38.560s
sys     0m0.140s
```

2. -O1 133.4s

```
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ g++ -O1 -o exp main.cpp mul.cpp
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ time ./exp

real    2m13.386s
user    2m13.365s
sys     0m0.020s
```

3. -O2 114.5s

```
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ g++ -O2 -o exp main.cpp mul.cpp
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ time ./exp

real    1m54.457s
user    1m54.416s
sys     0m0.041s
```

# 3. 使用icpc测试

1. -O0 128.6s

```
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ icpc -O0 -o exp main.cpp mul.cpp
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ time ./exp

real    2m8.576s
user    2m8.563s
sys     0m0.010s
```

2. -O1 124.3s

```
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ icpc -O1 -o exp main.cpp mul.cpp
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ time ./exp

real    2m4.295s
user    2m4.295s
sys     0m0.000s
```

3. -O2 117.34

```
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ icpc -o exp main.cpp mul.cpp
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ time ./exp

real    1m57.360s
user    1m57.350s
sys     0m0.011s
```

4. -O2 -ipo 2.6s

```
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ icpc -O2 -ipo -o exp main.cpp mul.cpp
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ time ./exp

real    0m2.622s
user    0m2.622s
sys     0m0.000s
```

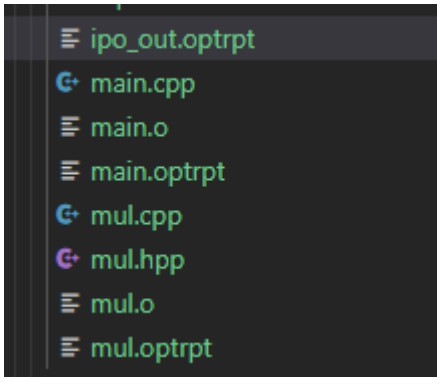
5. -O2 -PGO 102.1s

```
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ icpc -prof-gen -prof-dir./prof -c main.cpp mul.cpp
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ icpc -o exp main.o mul.o
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ ./exp
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ icpc -prof-use -prof-dir./prof -o exp main.cpp mul.cpp
warning #30005: Existing ./prof/pgopti.dpi will be overwritten.
warning #30005: Existing ./prof/pgopti.dpi will be overwritten.
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ time ./exp

real    1m42.087s
user    1m42.067s
sys     0m0.021s
```

6. -O2 -vec -xAVX 101.1s

```
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ icpc -O2 -vec -xAVX -o exp main.cpp mul.cpp
icpc: warning #10193: -vec is default; use -x and -ax to configure vectorization
(pytorch) mumu@Xiamu-ssr:~/Git/ParallelComputing/exper1$ time ./exp

real    1m41.095s
user    1m41.075s
sys     0m0.020s
```

7. -O2 -ipo -qopt-report 为源文件和编译选项都生成的优化报告文件 包含inline和loop




等等优化。

# 4. 总结

在不添加其他编译选择时，icpc比g++在-O优化上表现好一些。 对于icpc来说，无论过程间优化IPO，还是文件指导优化PGO，以及自动向量化，都有加速效果。其中ipo最为明显。 -qopt-report可以生成优化报告文件，里面是编译器所做的优化的细节描述。