HW3: Matrix Multiplication

MGP 2022
Due - 4/4 Mon 11:59pm

0 Introduction

In this project, you will implement matrix multiplication with multi-threading.
The specification has the following contents.

!!!Caution: Plagiarism!!!

You need to be careful to comply with plagiarism regulations. There is an ambiguity between discussion, idea, and solution. Therefore, be careful not to take away the opportunity of your colleague.

- No web code-searching
- No code sharing
- **No idea sharing**
- Ok discussion about the project

1 Environment Setup

The same manner in HW1, HW2.

You don't need to set up the environment. We prepared everything you need for an experiment about multi-threading. Check GNU GCC/G++ 9.2.1, GNU make, and HTCondor documentation for more information.

2 Framework

2-1 Matrix Multiplication

2-1-0 Structure of template code

```
/HW3/matmul$ tree .
.
├── build                  // make file generate objective files to here
│   └── driver.o           // includes main(). Only the objective file provided!
├── Makefile               // Compile, Clean, Format, Submit
├── matmul4096.cmd         // Command for condor
├── matmul2048.cmd         // Command for condor
├── result                 // Result from condor
└── src                    // Source Code
    ├── matmul.cpp
    └── matmul.h
```

2-1-1 Implement your optimized solution

```
/HW3/matmul$ cat src/matmul.cpp

#include "matmul.h"

void matmul_ref(const int* const matrixA, const int* const matrixB,
                int* const matrixC, const int n, const int m) {
  // You can assume matrixC is initialized with zero
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
```

```
    for (int k = 0; k < m; k++)
      matrixC[i * n + j] += matrixA[i * m + k] * matrixB[k * n + j];
}

void matmul_optimized(const int* const matrixA, const int* const matrixB,
                      int* const matrixC, const int n, const int m) {
  // TODO: Implement your code
}
```

2-1-2 Compile and run matmul

The 'matmul' program get two arguments.

- inputPath: path to input file
- outputPath: path to output file

To test the correctness of your code, you can use the input/output files in '**/nfs/home/mgp2022_data/**' directory.

Each input/output file is named in the format of 'input/output_n_m.txt'.

- n: matrix size (of matrix A row, matrix B column)
- m: matrix size (of matrix A column, matrix B row)

In the input file, the first two lines contain n and m. The following lines contain matrix A[nxm] and B[mxn].
In the output file, the first line contains n. The following lines contain matrix C[nxn], which is the result of the multiplication of A and B.

There are a total of five sizes (n,m) given:
(64, 64), (64, 128), (2048,4096), (2048,6114), (4096,8192)

```
# Compile
/HW3/matmul$ make
g++ -std=c++11 -pthread -lpthread -fopenmp -Wl,--no-as-needed -c src/matmul.cpp -o
```

```
build/matmul.o
g++ -std=c++11 -pthread -lpthread -fopenmp -Wl,--no-as-needed -o matmul \
      build/driver.o \
      build/matmul.o

# Show help
/HW3/matmul$ ./matmul
./matmul inputPath outputPath

# Local run
/HW3/matmul$ ./matmul /nfs/home/mgp2022_data/input_64_64.txt
/nfs/home/mgp2022_data/output_64_64.txt

====================================
      Matrix Multiplication
====================================
The size of Matrix: 64, 64
====================================

Read input file(/nfs/home/mgp2022_data/input_64_64.txt)...
Read output file(/nfs/home/mgp2022_data/output_64_64.txt)...

Run your solution...

matmul_optimal took 9.1e-08 sec
!!!!!!!Incorrect!!!!!!!
```

If you have properly implemented matmul_optimized, it will print 'Correct' like in the following example.

```
# Local run
/HW3/matmul$ ./matmul /nfs/home/mgp2022_data/input_64_64.txt
/nfs/home/mgp2022_data/output_64_64.txt

====================================
      Matrix Multiplication
====================================
The size of Matrix: 64, 64
====================================
```

```
Read input file(/nfs/home/mgp2022_data/input_64_64.txt)...
Read output file(/nfs/home/mgp2022_data/output_64_64.txt)...

Run your solution...

matmul_optimal took 0.00151535 sec
Correct
```

3 Remote Execution

There are two submit function, **submit_2048** and **submit_4096**. Each number means the size n of matrix. The size m of the matrix in this submit function is nx2. This means that the input for submit_2048 is (2048, 4096)x(4096,2048) and the input for submit_4096 is (4096, 8192)x(8192, 4096).

Step 1. Submit a job to condor.
Step 2. Wait until finished.(Check **result/matmul.log** for Job terminated)
Step 3. Check the output(**result/matmul.out**). There will be **TIME** and **CORRECTNESS** of your solution.

```
/HW3/matmul$ cat matmul2048.cmd
#####################################################
##
## Matrix Multiplication Condor command file
##
#####################################################

executable      = matmul
output          = result/matmul.out
error           = result/matmul.err
log             = result/matmul.log
request_cpus = 16
should_transfer_files   = YES
when_to_transfer_output = ON_EXIT
```

```
arguments              = /nfs/home/mgp2022_data/input_2048_4096.txt
/nfs/home/mgp2022_data/output_2048_4096.txt
queue


/HW3/matmul$ make submit_2048
condor_submit matmul2048.cmd
Submitting job(s).
1 job(s) submitted to cluster 12738.

/HW3/matmul$ cat result/matmul.out
====================================
       Matrix Multiplication
====================================
The size of Matrix: 2048, 4096
====================================

Read input file(input_2048_4096.txt)...
Read output file(output_2048_4096.txt)...

Run your solution...

matmul_optimal took TIME sec
CORRECTNESS
```

We check only **the runtime of submit_4096**. submit_2048 will be a faster indicator of whether you are doing well in our server. So, do not hesitate to use submit_2048.

4 Criteria

4-1 Requirements

4-1-0 General

- It has to be parallel (no serial implementation)
- No open source, or parallel library - you can only use STL.
  You can use boost::barrier, but if you want to use anything else, contact us before doing so.

- You can use pthread, std::thread, openMP, or MPI as your parallelization framework. If you want to use something else (e.g., Apache Spark), contact us first.
- Use of SIMD is fine: AVX, SSE, etc
- Do not override driver.o

4-1-1 Matrix Multiplication

- Problem size: (4096, 8192)x(8192, 4096)
- Performance target: **1.5**sec for (4096, 8192)x(8192, 4096)
- Correctness: Should work correctly for any matrix input (n,m)x(m,n) between size of $64 \leq n, m \leq 8192$
- You can use techniques such as the Strassen or Winograd algorithm, provided that you parallelize them yourself.

4-2 Measurements

We are measuring performance from a real machine, and it involves a certain amount of luck. To reduce the luck effect as much as possible, we do the following:

- Turn off DVFS of the machine
- Warm up the server for a few minutes before measuring. This means that the **program will run slightly slower than your own tests** for all of you during our grading.
- We will run your program at least five times and **use the minimum exec.time**.

5 Grading

Correct parallel implementation (10) - finish within **1 minute**, produce **correct** result with a parallel implementation

Performance bar (60) - Based on last year's students, we have set the performance target to be 1.5 sec. If your execution time is x seconds where x > 1.5, here's your score:

*score = (60*1.5/x)*

Extra point (≤75) - If you get your execution time below 1.5sec, congratulations! you will get some extra points. Since it's called 'extra', the amount of points should be less than

the main performance scores. Last year's top-ranker achieved 0.9 sec, so we have designed the score equation so that you will get about 30 points by catching the top ranker of the last year.

$$extra = 50*(1.5-x), \quad x \leq 1.5$$

6 Server Info

tell us if you need anything else!

```
$ cat /proc/cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 23
model           : 113
model name      : AMD Ryzen 7 3700X 8-Core Processor
stepping        : 0
microcode       : 0x8701012
cpu MHz         : 2196.672
cache size      : 512 KB
physical id     : 0
siblings        : 16
core id         : 0
cpu cores       : 8
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 16
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc
rep_good nopl nonstop_tsc cpuid extd_apicid aperfmperf pni pclmulqdq monitor ssse3 fma
cx16 sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand lahf_lm cmp_legacy svm
extapic cr8_legacy abm sse4a misalignsse 3dnowprefetch osvw ibs skinit wdt tce topoext
perfctr_core perfctr_nb bpext perfctr_llc mwaitx cpb cat_l3 cdp_l3 hw_pstate sme ssbd
mba sev ibpb stibp vmmcall fsgsbase bmi1 avx2 smep bmi2 cqm rdt_a rdseed adx smap
clflushopt clwb sha_ni xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc
```

cqm_mbm_total cqm_mbm_local clzero irperf xsaveerptr wbnoinvd arat npt lbrv svm_lock nrip_save tsc_scale vmcb_clean flushbyasid decodeassists pausefilter pfthreshold avic v_vmsave_vmload vgif umip rdpid overflow_recov succor smca
bugs : sysret_ss_attrs spectre_v1 spectre_v2 spec_store_bypass
bogomips : 7186.75
TLB size : 3072 4K pages
clflush size : 64
cache_alignment : 64
address sizes : 43 bits physical, 48 bits virtual
power management: ts ttp tm hwpstate cpb eff_freq_ro [13] [14]

… (continues until processor 16)

7 Reference

- GNU GCC/G++ 9.2.1
- GNU make
- HTCondor
- Our Lecture Slides
- Kazushige Goto and Robert A van de Geijn. Anatomy of high-performance matrix multiplication. ACM Transactions on Mathematical Software (TOMS), 2008.
- Tyler M Smith, Robert Van De Geijn, Mikhail Smelyanskiy, Je R Hammond, and Field G Van Zee. Anatomy of high-performance many-threaded matrix multiplication. In International Parallel and Distributed Processing Symposium, 2014.
- Tyler M Smith and Robert A van de Geijn. The momms family of matrix multiplication algorithms. arXiv preprint arXiv:1904.05717, 2019.
- Ask Professor, TA(Of course not about the code)