Homework 2
COSC 594

Jiahui Guo

January 29, 2014

1 Program Implementation

1.1 Test Data

The matrices are initialized with random number ranging from 0 to 100 with double type.

1.2 Program Organization

The program is organized in the following way:

src /

- **omp_gemm.c**
- **omp_trsm.c**
- **c_timer.c**
- **Makefile**

include /

- **c_timer.h**

bin /

- **run.sh** Run the executable files and generates plots
- **avg.py**
- **plotTime.gnu**

Makefile

ReadMe Illustrating how to execute the program.

2 Experiment Platform

The experiment was conducted on a hydra machine. The configuration of the machine is list below:

Processor Intel Core i7-3770 @ 3.40GHz

Memory 16.0GB DDR3

Hard drive 500GB 7200 RPM SATA @ 6Gbps, 16MB cache

Thread(s) per Core 4

Core(s) per socket 2

Socket(s) 1

3 Approaches

3.1 GEMM

This program implements the GEMM, which is defined as:

$$C \leftarrow \alpha AB + \beta C$$

Then the program is executed with different parameter settings, which are listed below:

- Matrix Size: From 100 to 3000 with step size 100
- Chunk Size: 10 and 50
- Number of Threads: 4 and 8
- Schedule Type: Static and dynamic

3.2 TRSM

The TRSM short for TRiangular Solve with Matrix, it solve the following problem:

$$AX = B$$

where A is either upper or lower triangular, and the solution matrix X is returned in the space occupied by B.

4 Obtain OpenMP environment information

Information of OpenMP environment is obtained through the following built-in function:

Number of processors available	`omp_get_num_procs()`
Number of threads being used	`omp_get_num_threads()`
Set the number of threads	`omp_set_num_threads(NTHREADS)`
Maximum number of threads available	`omp_get_max_threads()`
If in parallel region	`omp_in_parallel()`
If dynamic threads are enabled	`omp_get_dynamic()`
If nested parallelism is supported	`omp_get_nested()`

All the information would be printed to shell if the DEBUG macro is uncommented in source code.

5 Experiment Results

5.1 GEMM

The experiments results using diffrent parameter settings for gemm are shown in Fig.1 to Fig.9.

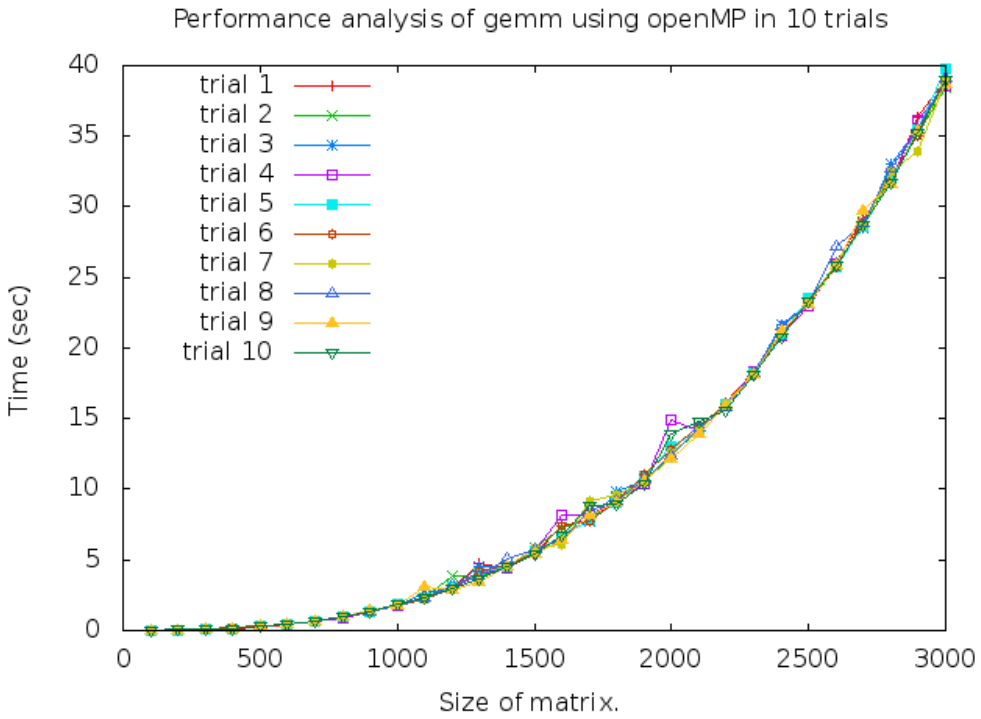

Figure 1: Execution time with static schedule, 8 threads, chunk size 10 in 10 trials

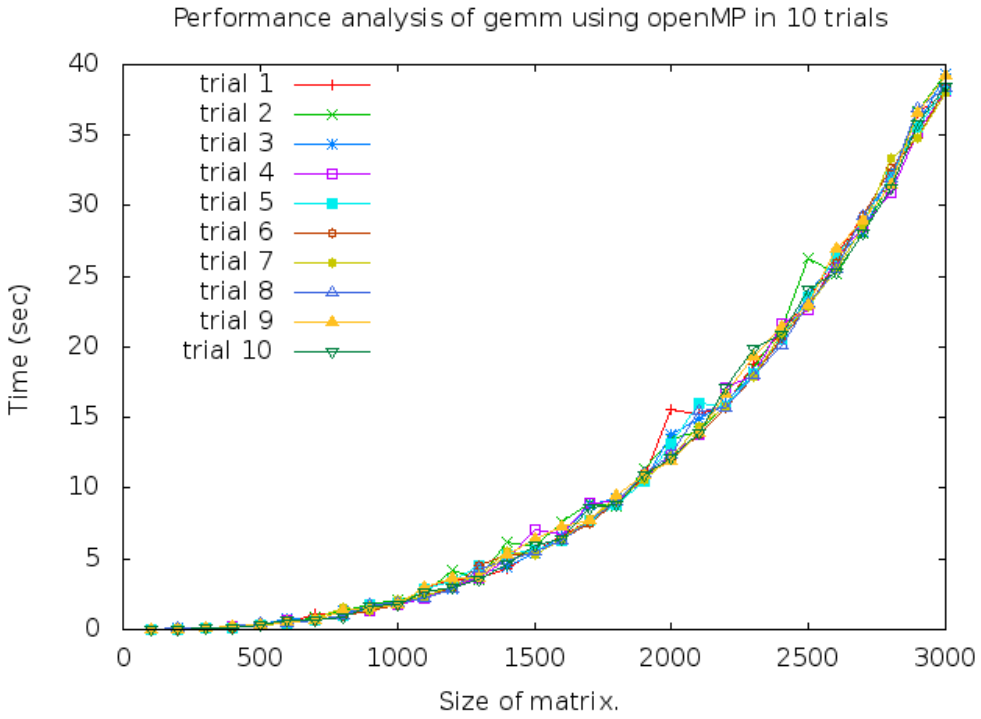

Figure 2: Execution time with static schedule, 4 threads, chunk size 10 in 10 trials

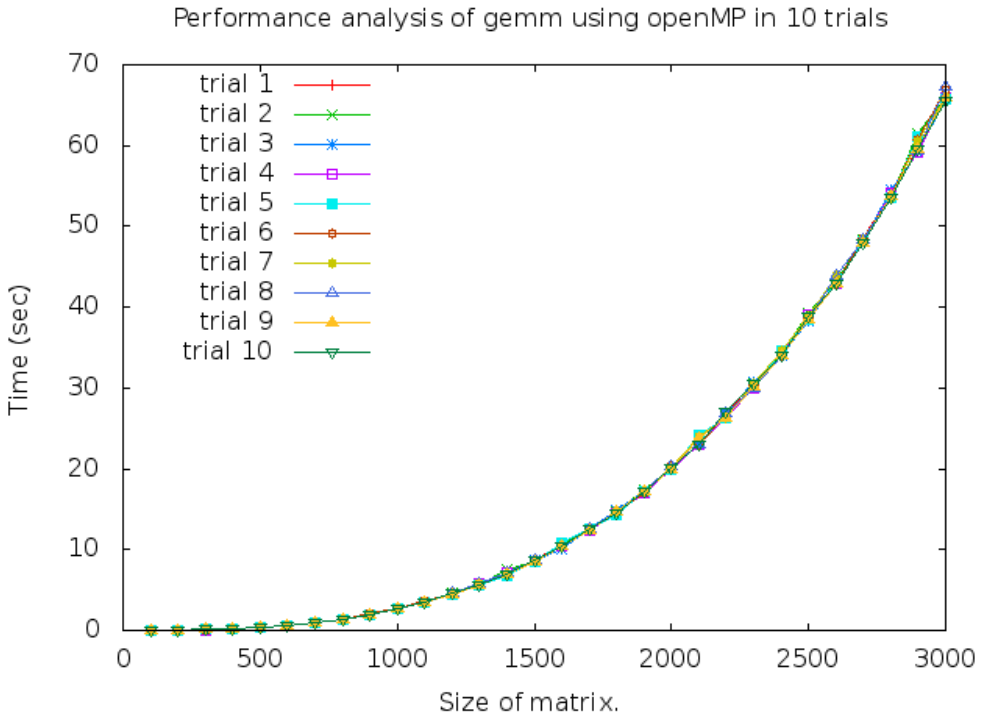

Figure 3: Execution time with static schedule, 2 threads, chunk size 10 in 10 trials

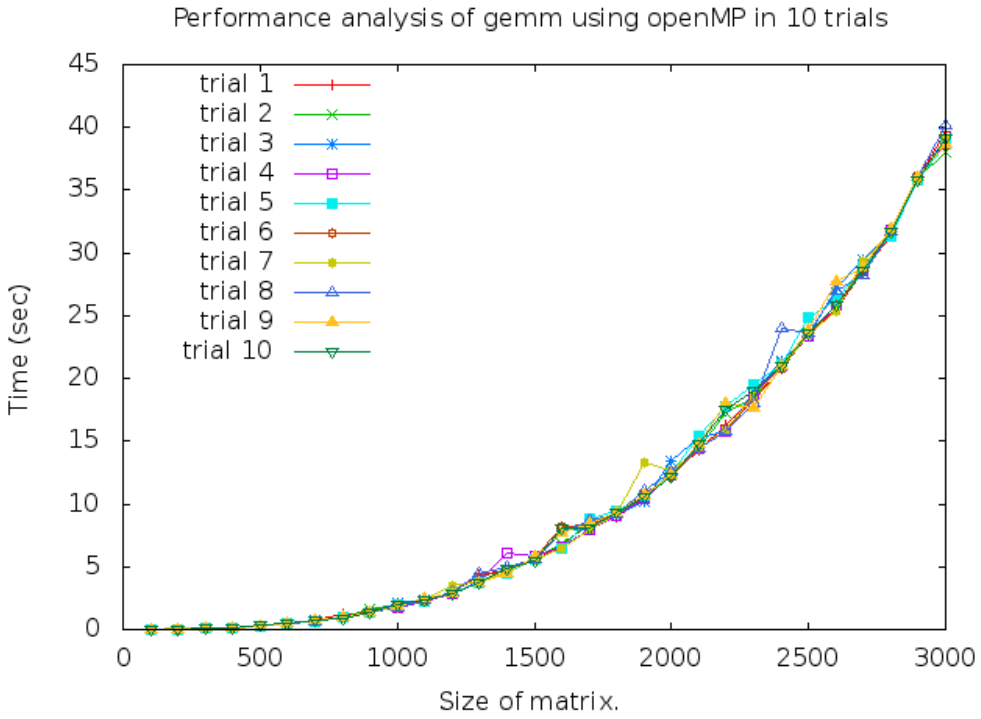

Figure 4: Execution time with static schedule, 8 threads, chunk size 50 in 10 trials

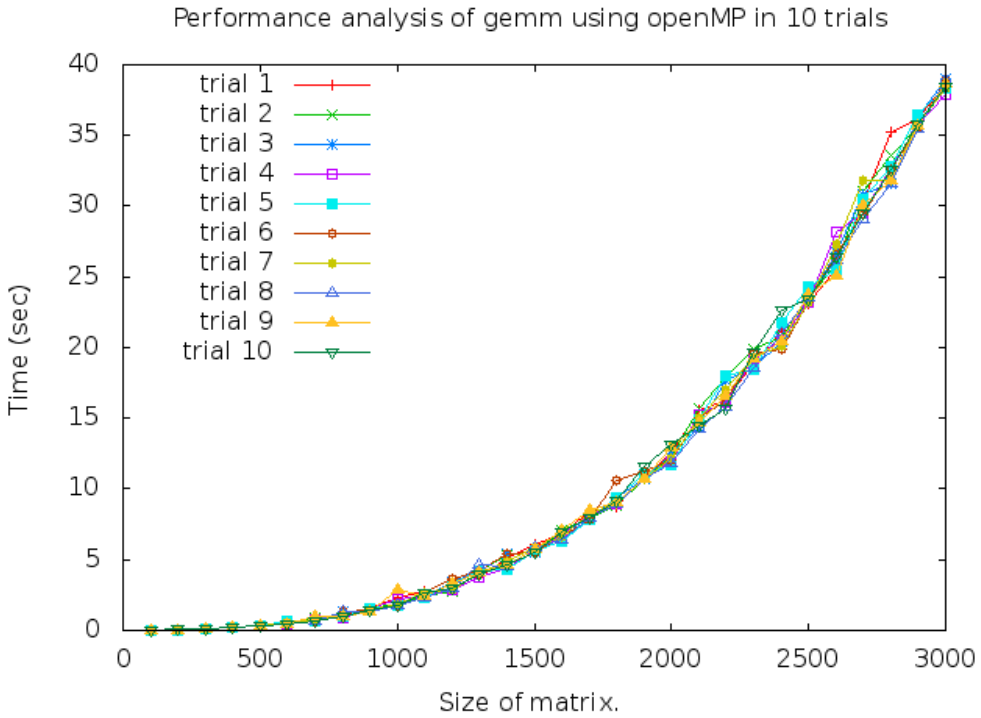

Figure 5: Execution time with static schedule, 4 threads, chunk size 50 in 10 trials

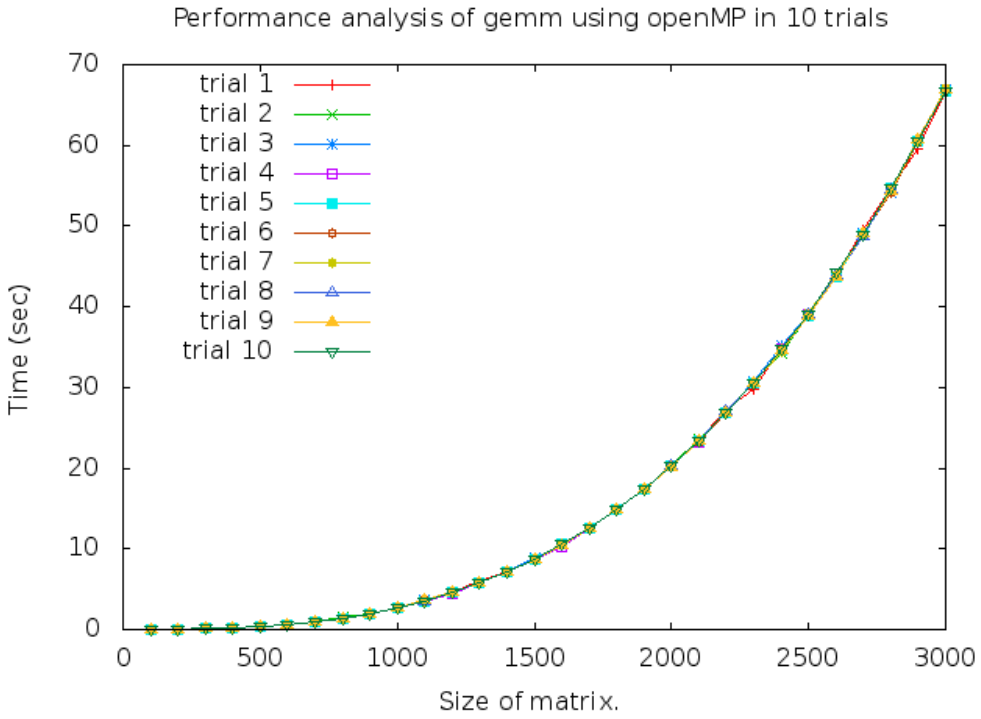

Figure 6: Execution time with static schedule, 2 threads, chunk size 50 in 10 trials

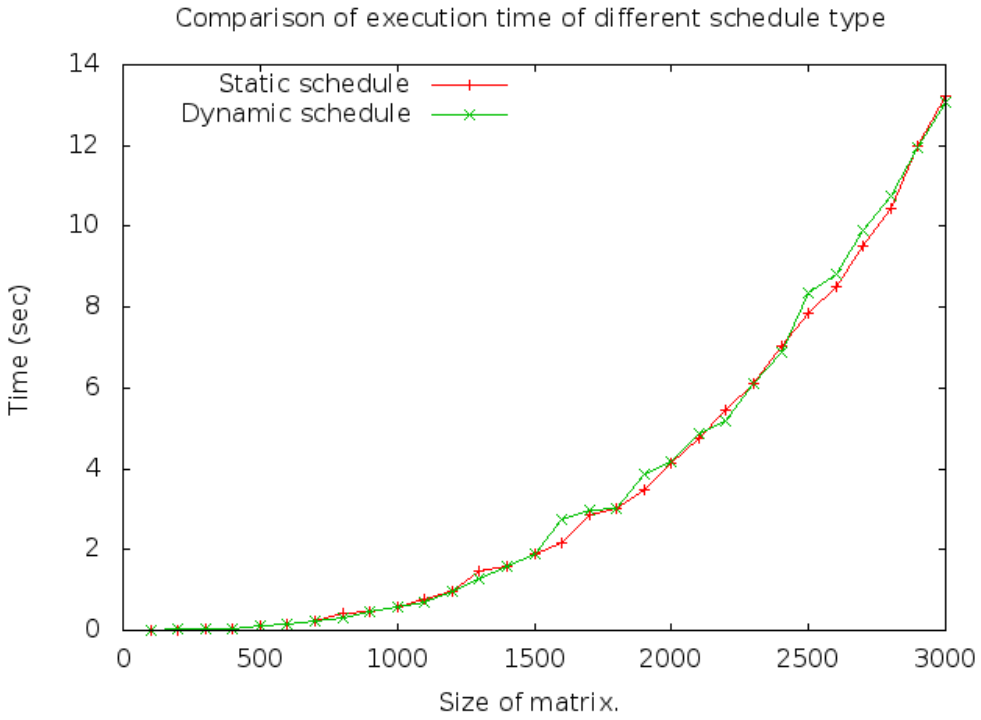

Figure 7: Comparison of execution time with diffrent schedule, 8 threads, chunk size 50

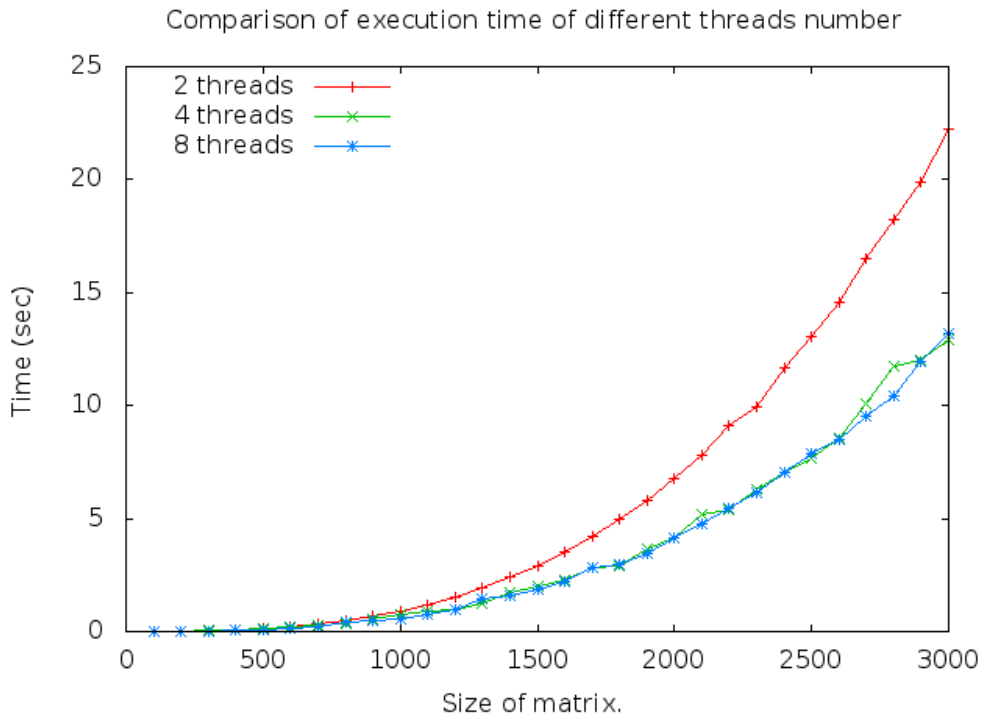

Figure 8: Comparison of execution time with static schedule, different threads, chunk size 50

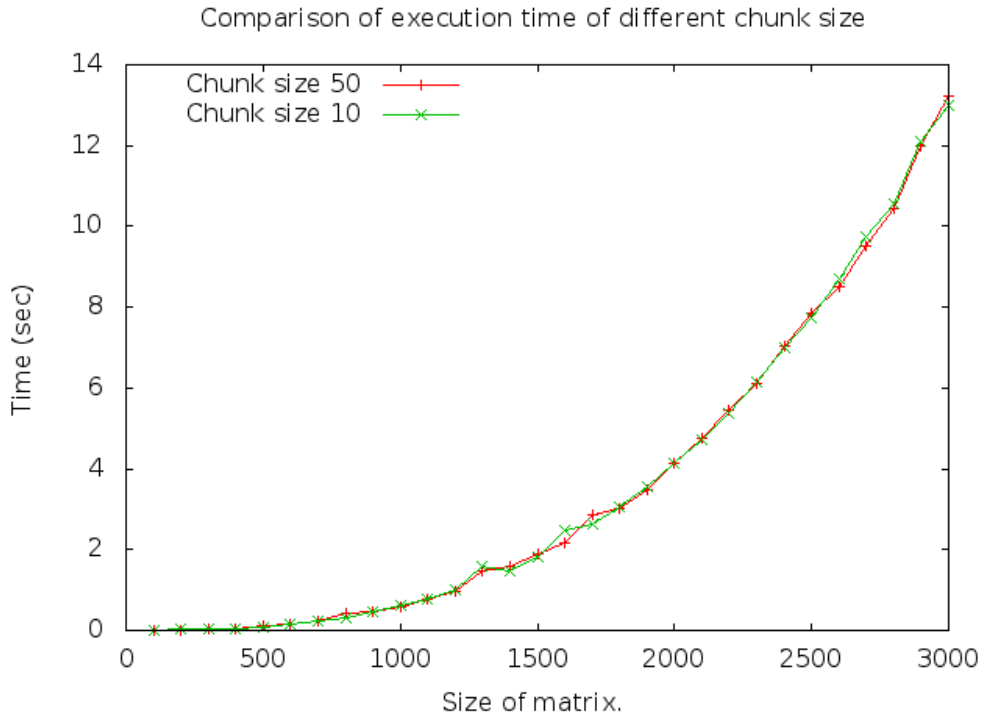

Figure 9: Comparison of execution time with static schedule, 8 threads, different chunk size

5.2 TRSM

The experiments results using diffrent parameter settings for trsm are shown in Fig.10 to Fig.11.

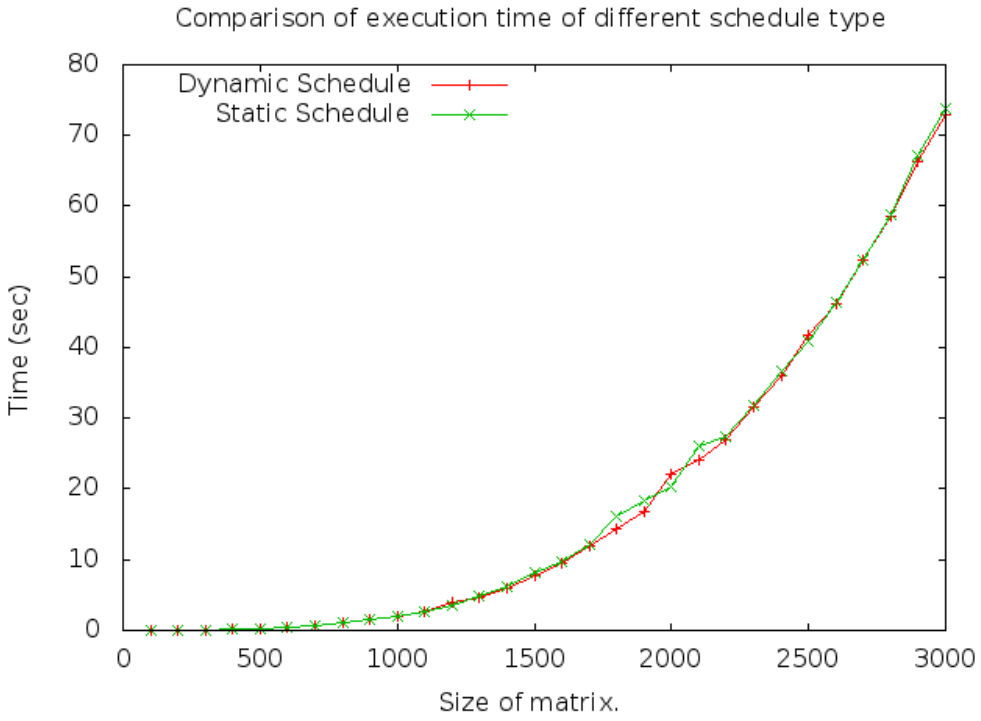

Figure 10: Comparison of execution time with diffrent schedule, 8 threads, chunk size 10

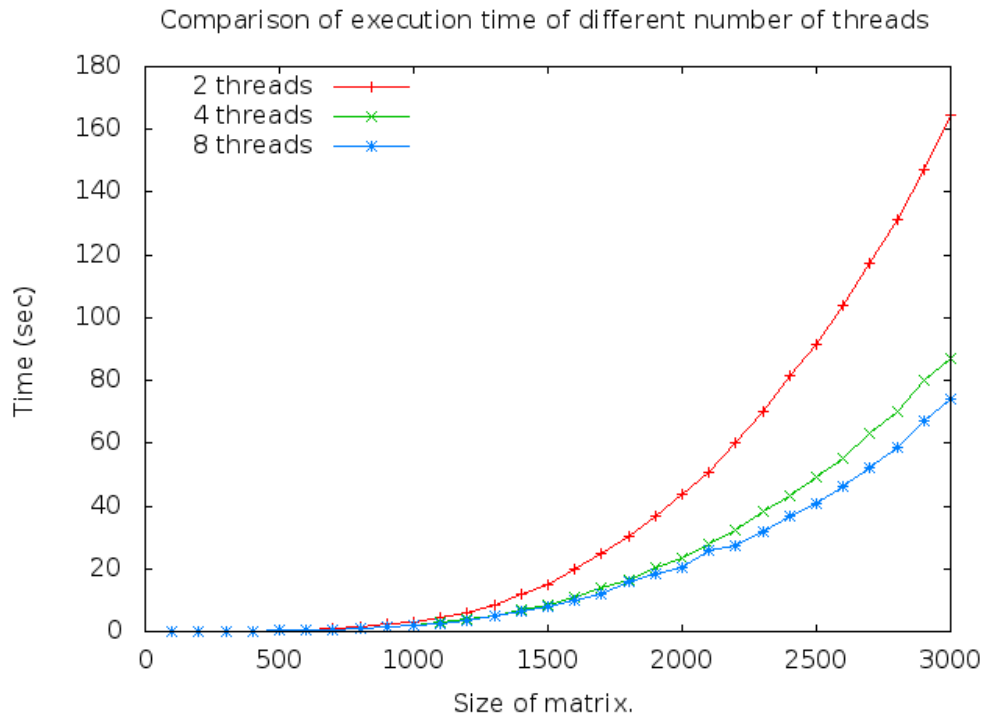

Figure 11: Comparison of execution time with static schedule, differentf number of threads, chunk size 10

6 Performance discussions

6.1 Schedule Type

From the results shown Fig.7 and Fig.10, it seems that the schedule type does not have influence on the execution time for GEMM or TRSM.

6.2 Chuck Size

From the results shown Fig.9, it seems that that the chunk size does not have influence on the execution time for GEMM.

6.3 Threads Number

Threads number has huge impacts on the execution time of a certain program. In GEMM, from Fig.8 we can see that the execution time for 4 and 8 threads are almost the same, but for 2 threads, the execution time nearly doubled when the matrix size increased to 3000. And in TRSM, from Fig.11, we can also the similar phenomenon.

6.4 Problem related

Fig.12 shows the execution time for GEMM and TRSM, using 8 threads, static schedule and chunk size 10, the result shows that with the increase of the input data size, the runtime for TRSM increases rapidly comparing to GEMM, the reason is in GEMM, all the multiplication can be parallel perfectly, but in TRSM, there are dependency for some elements in the matrix, thus TRSM cannot reach the same throughout as GEMM.

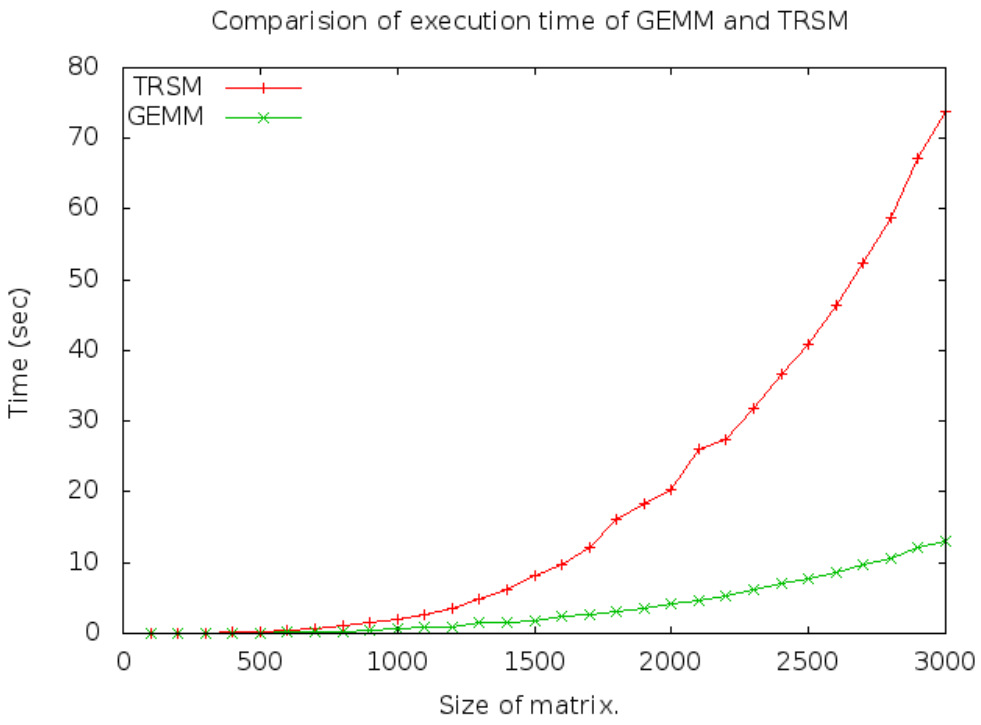

Figure 12: Comparison of execution time of gemm and trsm