Project 1

Name: Jinru Han Student ID: 521021910982

Abstract

In this program I learned designing and developing systems programs using C, learn how to use Linux system calls for process control and management, such as fork, exec, wait, pipe, and kill system call API, and in the problem of calculating matrix using multiple threads, I learned how to use Posix Pthread library for concurrency. In addition, I also learned how to use graphic tools (i.e. gnuplot) to analyze data, and this report mainly focuses on analyzing executing time in Lab 1 and Lab 3.

Contents

1 Lab01 : copy **1**

2 Lab03 : matrix **2**

2.1 Executing time with different matrix size using 4 threads 2

2.2 Executing time with different number of threads 3

3 Appendix **3**

1 Lab01 : copy

In the experiment, the relationship between program execution time and buffer size, as well as the number of characters in "src.txt" were tested, and the results are shown in Figure 1. Detailed execution time and data information can be found in Appendix Table 3.

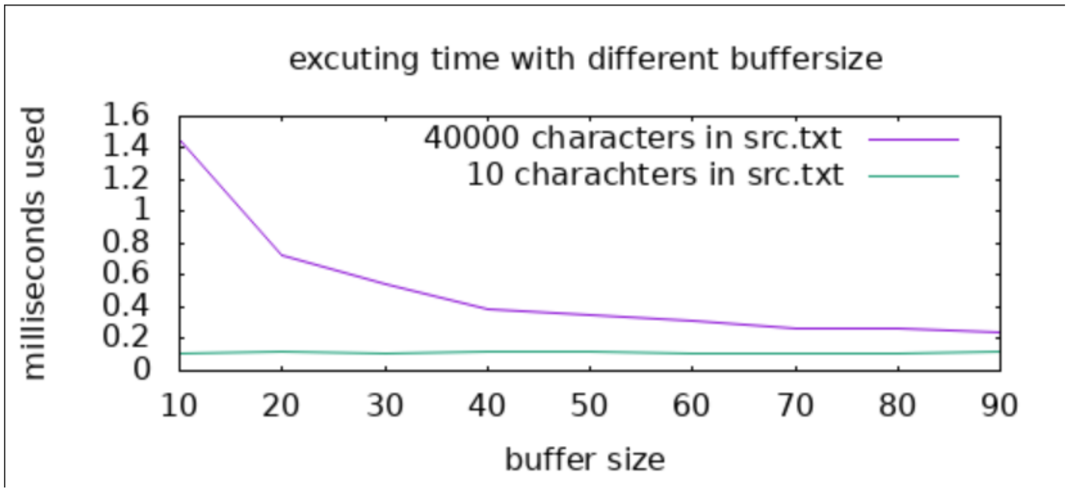

Figure 1: executing time with different buffer size

It can be seen that when the number of characters in "src.txt" is large (about 40,000 characters in this experiment), the program time decreases with the increase of buffer size. However, when the number of characters in "src.txt" is small (about 10 characters in this experiment), the relationship between program execution time and buffer size is relatively small.

This is because the program buffers the strings in ”src.txt” into buffer size when reading them. In the case where the number of characters is large and buffer size is small, the program needs to read data from the target file and cache it into the buffer multiple times, resulting in a large time overhead. In theory, the program execution time and buffer size are inversely proportional when the number of characters in the target file is constant. However, in the actual execution process, various other reasons may cause the relationship between time and buffer size to not strictly be inversely proportional.

2 Lab03 : matrix

2.1 Executing time with different matrix size using 4 threads

In the experiment, the relationship between matrix computation time and matrix size was tested when the number of threads was fixed at 4, and the results are shown in Figure 2. During the execution process, the uncertainty of the size of the random numbers generated caused large fluctuations in the computation time when the matrix size was small. Therefore, to obtain more accurate experimental data, the range of random numbers in the program was set to $[0 - 10]$, and the time calculation of matrices of the same size was averaged over 3 times. Detailed execution time and data information can be found in Table 1.

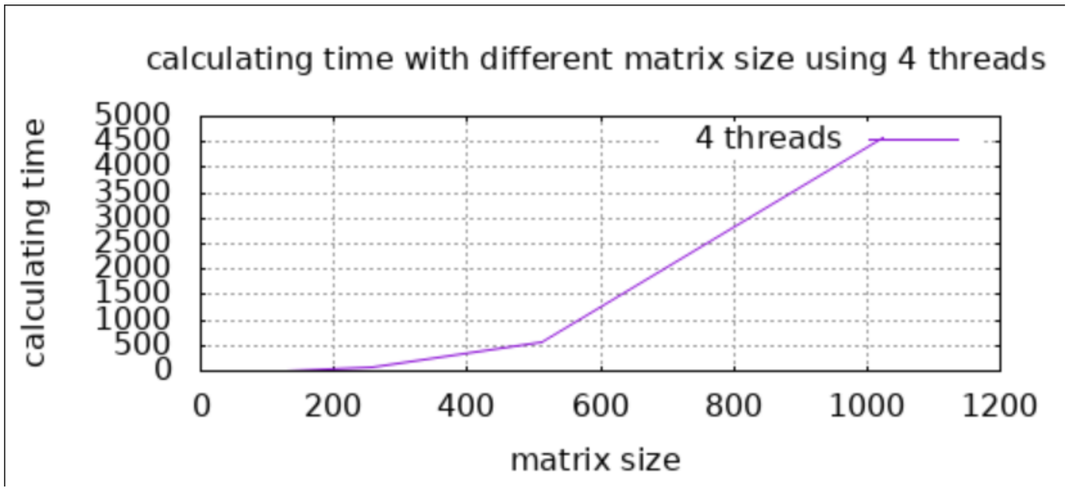

Figure 2: Executing time with different matrix size

matrix size	milliseconds
4	0.375
8	0.395
16	0.846
32	0.544
64	1.276
128	7.059
256	71.673
512	573.882
1024	4593.326

Table 1: concrete executing time with different matrix size

It can be observed that when the matrix size is large (after 64), the computation time tends to follow a cubic relationship with the matrix size. This is because the time complexity of computing a matrix is $O(n^3)$. However, when the matrix size is small, the relationship between computation time and matrix size may not strictly follow a cubic relationship due to other factors such as caching mechanisms, thread overhead, and memory access patterns. When the matrix size is small, the effect of caching mechanism is more significant, thread overhead is relatively small, and memory access pattern may also be simpler. Therefore, the computation time may not strictly follow a cubic relationship.

2.2 Executing time with different number of threads

In the experiment, the relationship between matrix calculation time and the number of threads was tested when the matrix size was fixed (128 * 128 matrices). The results are shown in Figure 3. To improve the accuracy of the calculation time, each case was executed three times and the average was taken. The detailed execution time and data information are shown in Table 2.

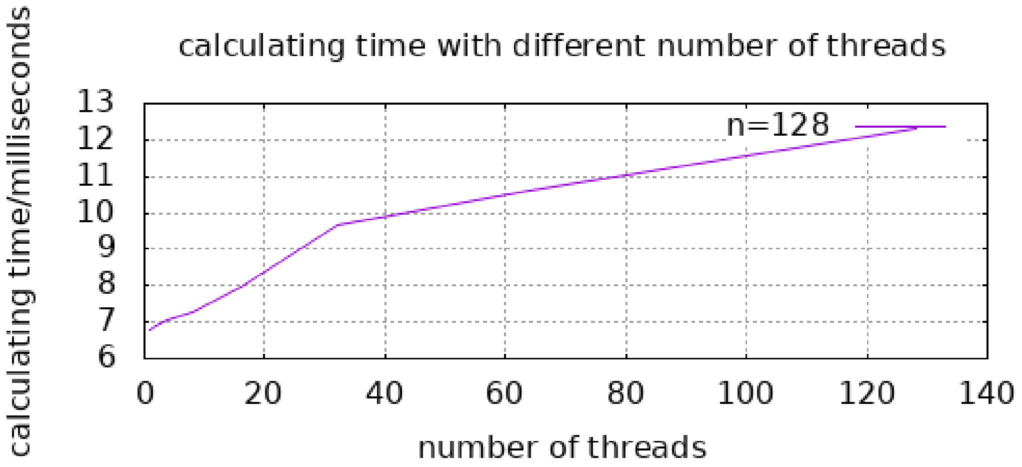

Figure 3: Executing time with different number of threads

num of threads	milliseconds
1	6.804
2	6.903
4	7.046
8	7.270
16	7.951
32	9.656
64	10.602
128	12.296

Table 2: concrete executing time with different number of threads

From the table, it can be seen that the calculation time slightly increases when the number of threads is small (less than 16), as the number of threads increases, the calculation time increases significantly. This may be due to additional time overhead caused by thread creation and destruction, thread synchronization, and thread switching.

3 Appendix

buffer size	40000 characters in src.txt	10 characters in src.txt
10	1.458000	0.108000
20	0.732000	0.116000
30	0.542000	0.105000
40	0.393000	0.127000
50	0.348000	0.120000
60	0.317000	0.112000
70	0.269000	0.112000
80	0.263000	0.108000
90	0.247000	0.117000

Table 3: concrete executing time with different buffer size