**The title of the university: HDU-ITMO**

**Group member: Wang Xinyue, Fan Xulin, Shi Chiye**

**Laboratory research #2. ”Study of the parallel libraries for C-programs effectiveness”**

目录

[1 Processor description 3](#_Toc37508055)

[2 Description of the parallel library configuration features 4](#_Toc37508056)

[3 Full text of the resulting parallel program and the scripts 5](#_Toc37508057)

[3.1 the resulting parallel program 5](#_Toc37508058)

[3.2 the scripts used to compile the program and conduct experiments. 7](#_Toc37508059)

[4 Graphs of program execution time functions 9](#_Toc37508060)

[5 Detailed conclusions with an analysis of the graphs and the results obtained. 10](#_Toc37508061)

# 1 Processor description

Processor: 2.9 GHz Inter Core i5

Operating system: ubuntu18.04

GCC compiler used for experiments: gcc version 7.5.0 (Ubuntu 7.5.0-3ubuntu1~18.04)

bit depth: 64

number of cores: 4

RAM capacity: 12G

# 2 Description of the parallel library configuration features

**a. First of all, in order to use the shared libraries, create the following symbolic links:**

For 64 bit installation:

cd lib/FW\_1.3.1\_Lin64/lib

Then create the following soft links using the following commands:

ln -sf ./libfwBase.so.1.3.1 libfwBase.so.1

ln -sf ./libfwImage.so.1.3.1 libfwImage.so.1

ln -sf ./libfwJPEG.so.1.3.1 libfwJPEG.so.1

ln -sf ./libfwSignal.so.1.3.1 libfwSignal.so.1

ln -sf ./libfwVideo.so.1.3.1 libfwVideo.so.1

ln -sf ./libfwBase.so.1.3.1 libfwBase.so

ln -sf ./libfwImage.so.1.3.1 libfwImage.so

ln -sf ./libfwJPEG.so.1.3.1 libfwJPEG.so

ln -sf ./libfwSignal.so.1.3.1 libfwSignal.so

ln -sf ./libfwVideo.so.1.3.1 libfwVideo.so

ln -sf ./libfwBase.so.1 libfwBase.so

ln -sf ./libfwImage.so.1 libfwImage.so

ln -sf ./libfwJPEG.so.1 libfwJPEG.so

ln -sf ./libfwSignal.so.1 libfwSignal.so

ln -sf ./libfwVideo.so.1 libfwVideo.so

**b. Second, compile the lab2.cpp file that uses FW, and all FW library have dependency on fwBase:**

g++ -I /mnt/c/Users/luanjiejie/lib/FW\_1.3.1\_Lin64 -L /mnt/c/Users/luanjiejie/lib/FW\_1.3.1\_Lin64/lib lab2.cpp -o lab2 -lm -lfwBase -lfwSignal

**c. Third, set the lib into the environment’s shared library:**

$exportLD\_LIBRARY\_PATH=/mnt/c/Users/luanjiejie/lib/FW\_1.3.1\_Lin64/lib:$LD\_LIBRARY\_PATH

# 3 Full text of the resulting parallel program and the scripts

## 3.1 the resulting parallel program

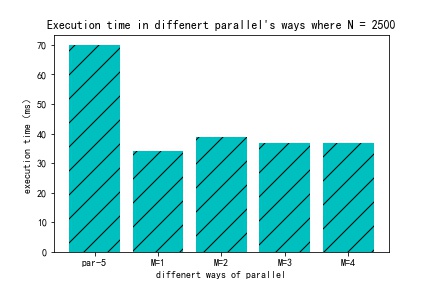
1. #include <stdio.h>
2. #include "/Users/wxy/desktop/并行计算/ExampleDir/FW\_1.3.1\_Mac64/fwBase.h"
3. #include "/Users/wxy/desktop/并行计算/ExampleDir/FW\_1.3.1\_Mac64/fwSignal.h"
4. #include <stdlib.h>
5. #include <math.h>
6. #include <sys/time.h>
8. **int** main(**int** argc, **char** \*argv[])
9. {
11. **int** i,j;
13. **struct** timeval T1, T2;
15. **long** delta\_ms;
17. **size\_t** N;
18. **if**(argc > 1)
19. N = (**size\_t**) atoi(argv[1]);
20. **else**
21. N = 20;
22. **if**(argc > 2) {
23. fwSetNumThreads(atoi(argv[2]));
24. } **else** {
25. fwSetNumThreads(4);
26. }
28. gettimeofday(&T1, NULL); /\* 记住当前时间T1 remember the current time T1 \*/
30. unsigned **int** seed = 0;
32. **for** (i = 0; i < 10; ++i) /\* 50 experiments50 experiments \*/
33. {
34. **double** M1[N] ,M2[(**int**)(N/2)];
35. srand(i); /\* 初始化RNG的初始值 initialize the initial value of the RNG\*/
36. **int** A = 24;
37. **for**(j = 0; j < N; j++){/\* 用N个数据填充初始数组 Fill the initial data array with size N \*/
38. M1[j] = 1 + rand\_r(&seed) % A;
39. }
40. **for**(j = 0; j < N/2; j++){
41. M2[j] = rand\_r(&seed)%(A\*9) + A;
42. }
43. /\* Complete the task, fill in the array with the results \*/
44. /\* Map Stage. Apply Hyperbolic cotangent of number’s root to each element in the M1 array \*/
45. fwsTanh\_64f\_A50 ( M1, M1, N);
46. fwsPowx\_64f\_A50 ( M1, -1, M1, N);
48. /\* in the M2 array, add each element in turn with the previous one ,and apply Sine modulus to the result of the addition\*/
49. **double** M2\_temp[(**int**)(N/2)+1];
50. fwsCopy\_64f(M2,M2\_temp,N/2);
51. fwsAdd\_64f(&M2[1], M2\_temp, &M2[1], N/2-1 );
52. fwsSin\_64f\_A50(M2,M2,N/2);
53. fwsAbs\_64f(M2,M2,N/2);
55. /\* Merge Stage. In arrays M1 and M2 apply Raising to a power to all elements in pairs with the same indexes in \*/
56. fwsPow\_64f\_A50(M1, M2, M2, N/2);
58. /\* Sort the array with the results by the specified method \*/
59. /\* Stupid sort \*/
60. **int** x = 0;
61. **while** (x < N/2)
62. {
63. **if** (x == 0 || M2[x - 1] <= M2[x])
64. x++;
65. **else**
66. {
67. **double** tmp = M2[x];
68. M2[x] = M2[x - 1];
69. M2[--x] = tmp;
70. }
71. }
72. /\* Reduce Stage. determining parity \*/
73. **double** min = A\*10;
74. **for**(j = 0; j < N/2; j++){/\* Find the minimum \*/
75. **if**(min > M2[j] && M2[j]!=0)
76. min = M2[j];
77. }
78. **double** sum = 0;
79. **for**(j = 0; j < N/2; j++){
80. **int** tmp = (**int**)(M2[j]/min);
81. **if**(tmp%2==0)
82. sum = sum + sin(M2[j]);
83. }
84. printf("i->%d,X->%f\n",i,sum);
85. }

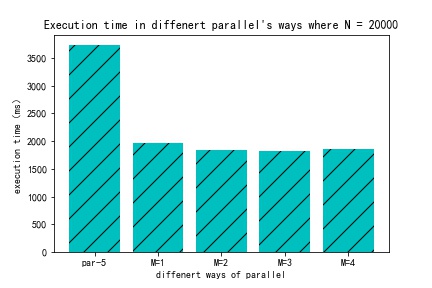
88. gettimeofday(&T2, NULL);
90. /\* remember the current time T2 \*/
92. delta\_ms = 1000 \* (T2.tv\_sec - T1.tv\_sec) + (T2.tv\_usec - T1.tv\_usec) / 1000;
94. printf("\nN=%d. Milliseconds passed: %ld\n", N, delta\_ms); /\* T2 -T1 \*/
95. **return** 0;
96. }

## 3.2 the scripts used to compile the program and conduct experiments.

1. #!/bin/bash
3. **if** [ ! -f lab2.o ];then
4. g++ -I /mnt/c/Users/luanjiejie/lib/FW\_1.3.1\_Lin64 -L /mnt/c/Users/luanjiejie/lib/FW\_1.3.1\_Lin64/lib lab2.cpp -o lab2 -lm -lfwBase -lfwSignal
5. fi
7. export LD\_LIBRARY\_PATH=/mnt/c/Users/luanjiejie/lib/FW\_1.3.1\_Lin64/lib:$LD\_LIBRARY\_PATH
9. N1=2500 # 100ms
10. N2=20000 #2000ms depends on your cp ,change by yourself
11. DELTA=$(expr $(expr ${N2} - ${N1}) / 10)
13. corenum=`sudo cat /proc/cpuinfo |grep processor|sed '$!d'|awk '{print $3}'`
15. #exce=a.out
16. para=1
18. **while** [ ${para} -le ${corenum} ]
19. **do**
20. N=$N1
22. **if** [ ! -d res ];then
23. mkdir res
24. fi
26. echo "N,TIME,X">./res/para-${para}.csv
28. **while** [ ${N} -le ${N2} ]
29. **do**
31. RESULT=$(./a.out ${N} ${para})
32. echo "${RESULT} "para="${para}"
33. TIME=`echo ${RESULT} | awk '{print $3}'`
34. X=`echo ${RESULT} | awk '{print $10}'`
35. echo "$N,$TIME,$X">>./res/para-${para}.csv
36. N=$(expr ${N} + $DELTA)
38. done
39. para=$(expr $para + 1)
40. done

# 4 Graphs of program execution time functions

****

****

# 5 Detailed conclusions with an analysis of the graphs and the results obtained.

From the graphs which shows the execution time changes in different ways to parallel and different cores where N’s value is 2500 and 20000 respectively, we can conclude that after using parallel library, **the execution time is about reduced by half.**

After using the parallel library, even when M = 1, that is, when a thread is used to run the program, the speed is much faster. I think it should be the reason for replacing all the loops with mathematical functions in the map and Merge phases. These mathematical functions use SSE2 (Streaming SIMD Extensions 2), which makes the execution speed faster.

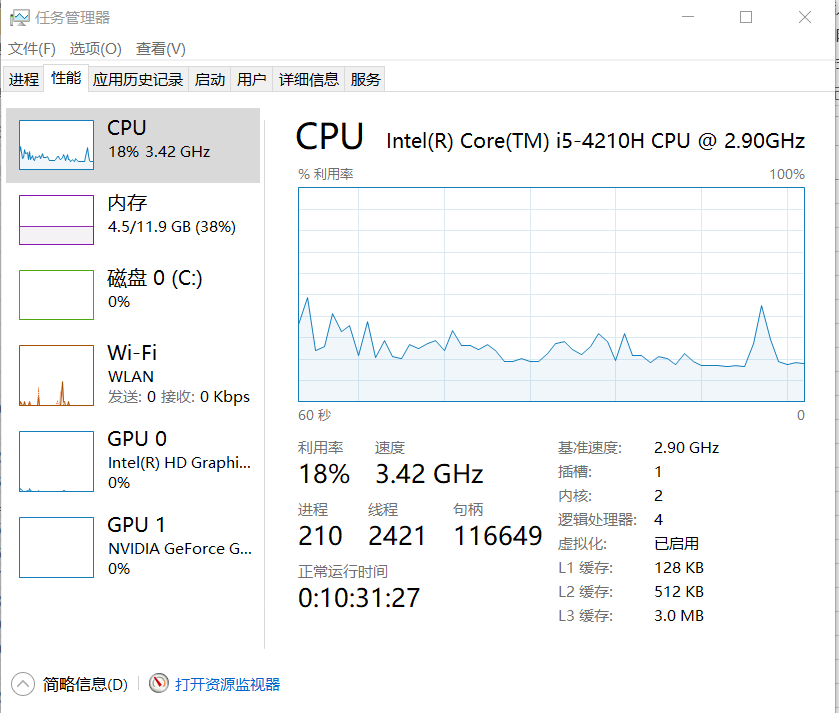
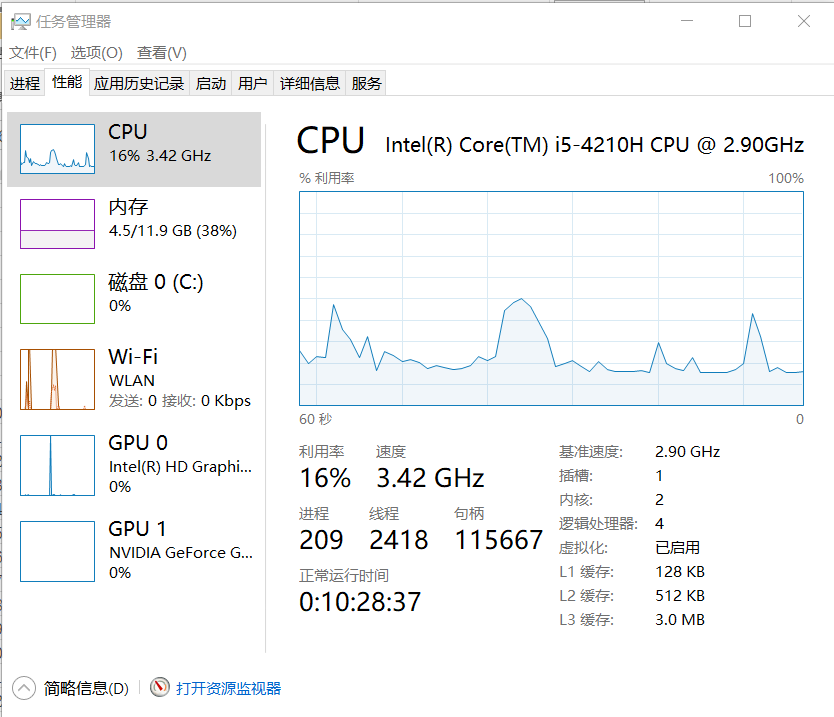
And for the program which uses parallel library, with the increase of the core number, the execution time has no obvious changes. Maybe the amount of data is not big enough.

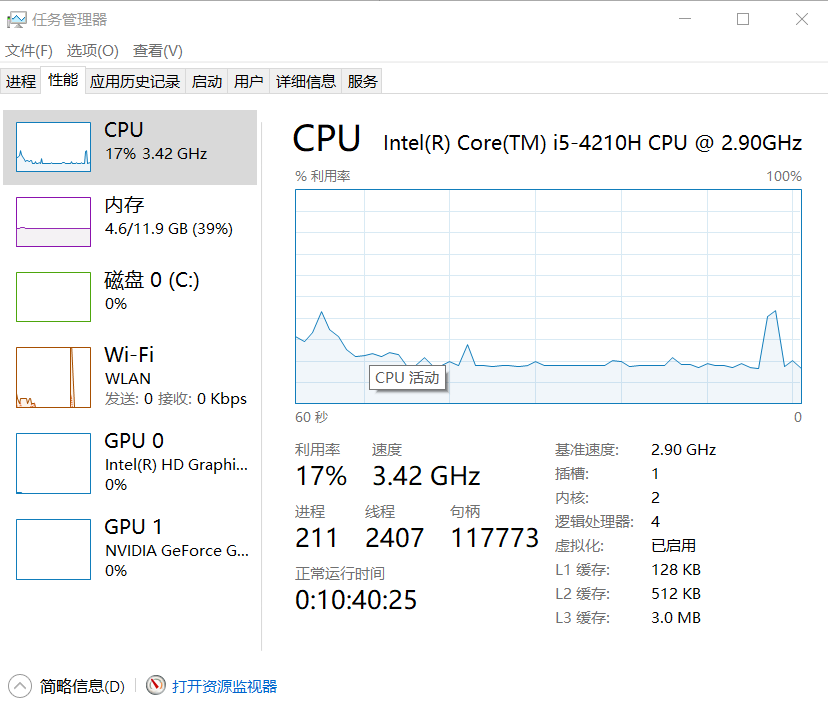
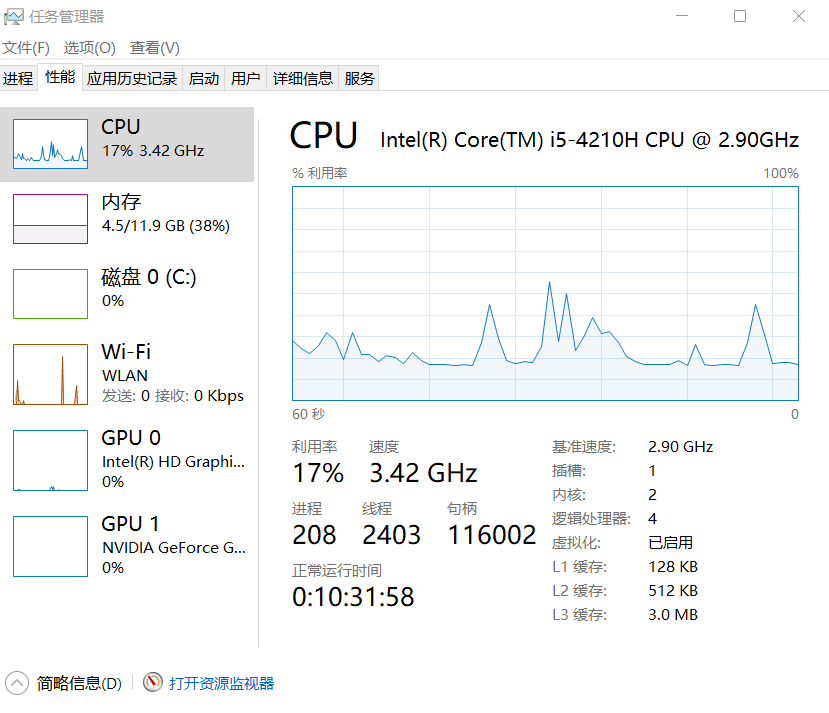
# Optional task #1

name of the used Manager: Windows Task Manager

screenshot of the Task Manager:

The position indicated by the red arrow is the start and end of the program





Analysis: Creating too many threads can cause unnecessary virtualization overhead. When the number of threads reaches a suitable value, and then create a new thread, the execution time of the program will become longer.