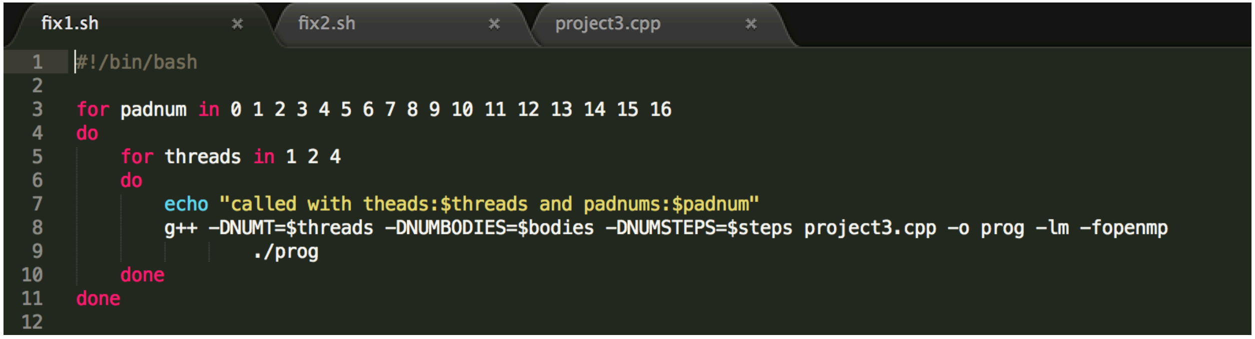
Parallel Programming

CS575

Chao Zhang

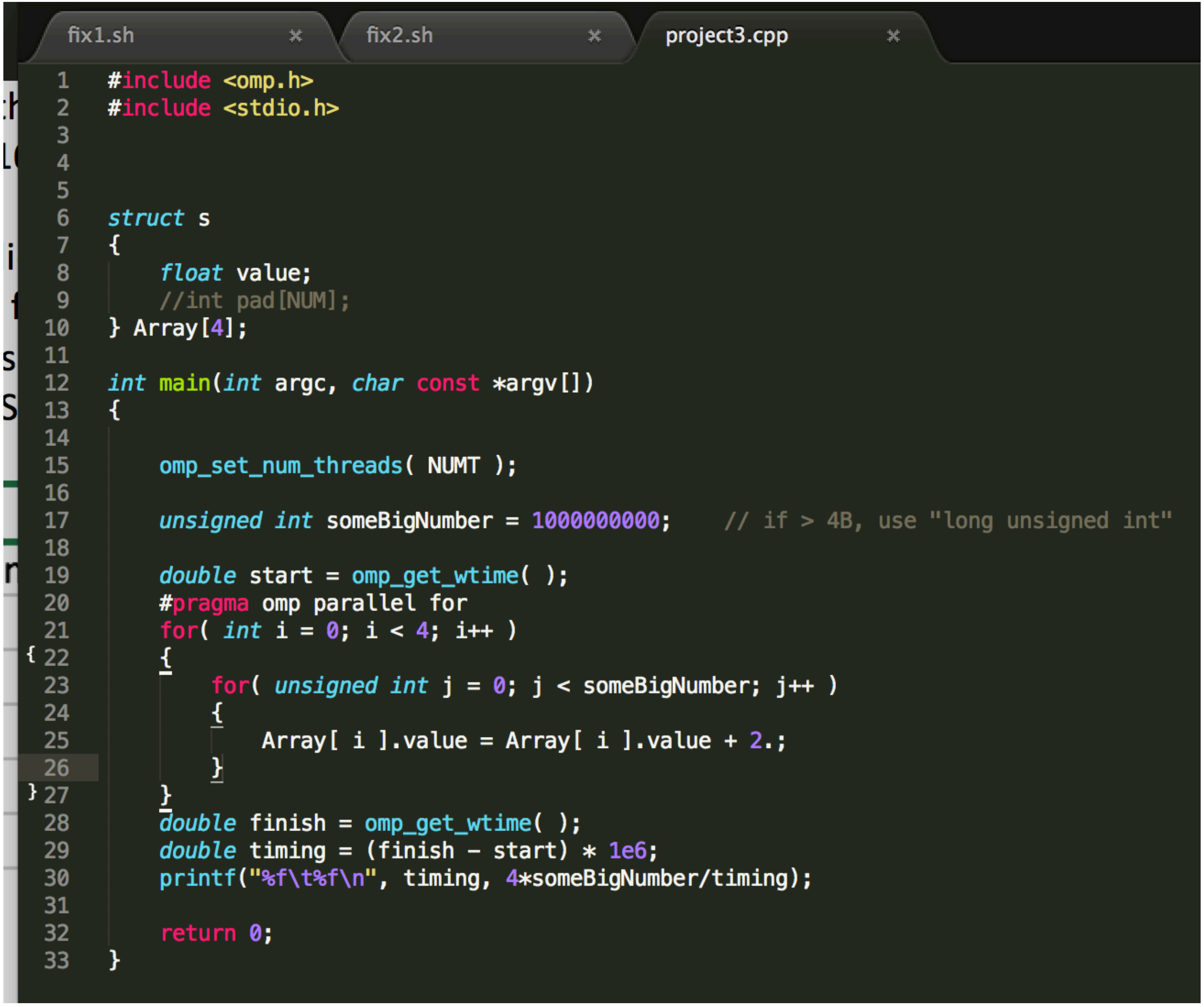
Project #3

1. Source listing



```bash
#!/bin/bash

for padnum in 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
do
    for threads in 1 2 4
    do
        echo "called with theads:$threads and padnums:$padnum"
        g++ -DNUMT=$threads -DNUMBODIES=$bodies -DNUMSTEPS=$steps project3.cpp -o prog -lm -fopenmp
                ./prog
    done
done

```

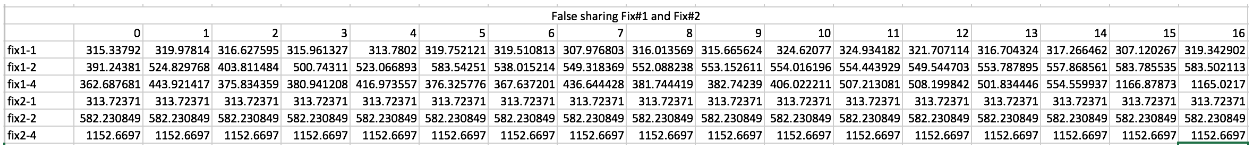
This is the fix1.sh file, it is used to run the program file which is the project3.cpp. The threads are 1, 2, 4. And I give the number of padding from 0 to 16.



```cpp
#include <omp.h>
#include <stdio.h>



struct s
{
    float value;
    //int pad[NUM];
} Array[4];

int main(int argc, char const *argv[])
{

    omp_set_num_threads( NUMT );

    unsigned int someBigNumber = 1000000000;    // if > 4B, use "long unsigned int"

    double start = omp_get_wtime( );
    #pragma omp parallel for
    for( int i = 0; i < 4; i++ )
    {
        for( unsigned int j = 0; j < someBigNumber; j++ )
        {
            Array[ i ].value = Array[ i ].value + 2.;
        }
    }
    double finish = omp_get_wtime( );
    double timing = (finish - start) * 1e6;
    printf("%f\t%f\n", timing, 4*someBigNumber/timing);

    return 0;
}
```

This code used for the fix. It will get the padnum the the NUMT from the .sh file. The out put will be the running time and the performance.

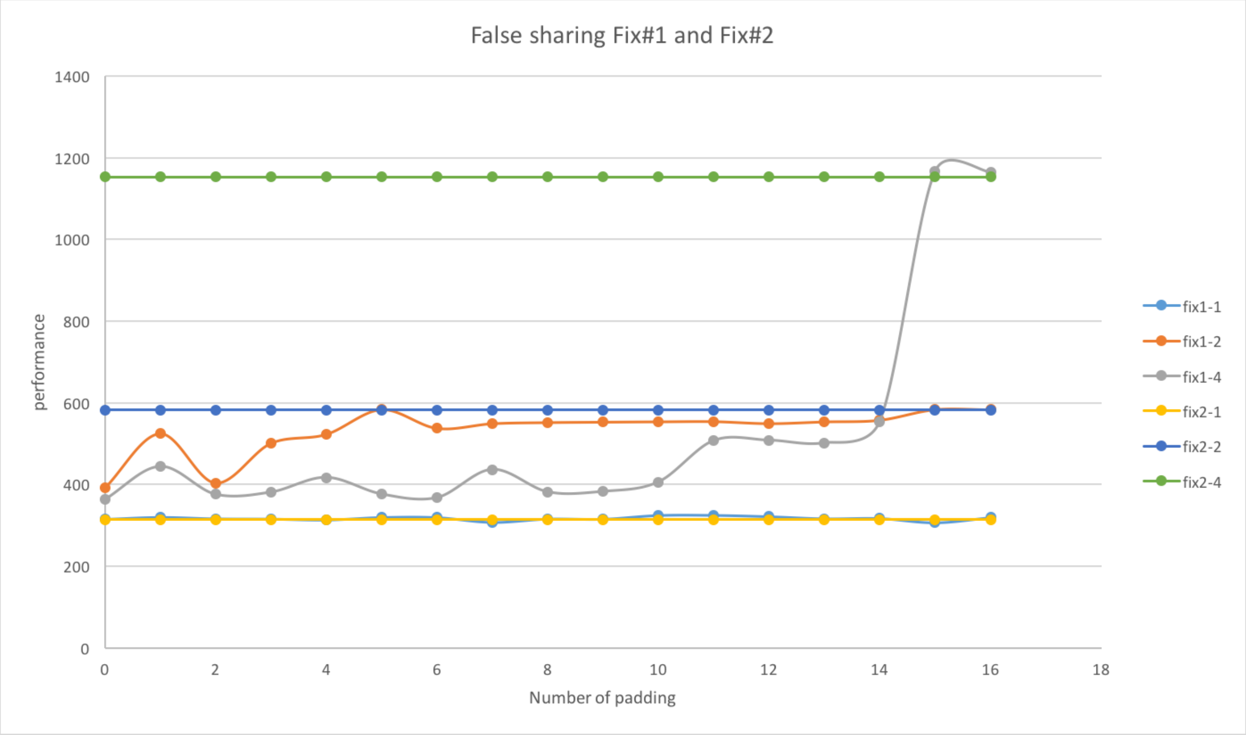
2. Result and analysis



False sharing Fix#1 and Fix#2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
fix1-1	315.33792	319.97814	316.627595	315.961327	313.7802	319.752121	319.510813	307.976803	316.013569	315.665624	324.62077	324.934182	321.707114	316.704324	317.266462	307.120267	319.342902
fix1-2	391.24381	524.829768	403.811484	500.74311	523.066893	583.54251	538.015214	549.318369	552.088238	553.152611	554.016196	554.443929	549.544703	553.787895	557.868561	583.785535	583.502113
fix1-4	362.687681	443.921417	375.834359	380.941208	416.973557	376.325776	367.637201	436.644428	381.744419	382.74239	406.022211	507.213081	508.199842	501.834446	554.559937	1166.87873	1165.0217
fix2-1	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371	313.72371
fix2-2	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849	582.230849
fix2-4	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697	1152.6697

False sharing Fix #1 and Fix #2 Table



False sharing Fix #1 and Fix #2 Graphs

For the Fix #1, the 1 thread is almost a straight line. The two threads line increased at 5 and 7, for the 5, there is one data assigned to two cache lines, so both threads can read those lines at same time. For the 7, the reason may be the the cache lines have the same numbers of data. For the threads 4, the line also increased at 11 and 15. For the 11, it should have the same reason like the 5 with threads 2. And the 15, I think the reason is all the data fit in one cache line.

For the Fix #2, the pad number will not influence the performance of each Fix. For all fix #2, each stack has its own stack to take care of the local variables. That why it not changes the performance.