HW2 report

Kangdong Yuan

Introduction

This homework is intended to improve the performance of the program by different technical methods that include serial and data access optimization. Moreover, there are some faults in the program that slow the running time, so I also need to improve the performance by correcting some slow parts.

Part1

Problem statement

The program in part1 has two loops, which the command in the inner loop will be executed N^2 times. And the code in the inner loop is slow and redundant, because it includes many slow functions such as fmod, round, sin, and cos. Moreover, the sin and cos have been called multiple times to compute the same value. So, the performance of this code has been impaired by those low efficient code and the running time of this program is 20.8 second.

Solutions

The solutions to improve the efficiency are various. The first thing I do is use variable “a”, “b” to store the value of “sin(val)” and “cos(val)”. Then I use a variable “t” to store the value of “(a*a-b*b)”. After this change, the time improved from 20.8 seconds to 17 seconds. Then, the most inefficient part of this program is the arrangement of the loop, because the computations of variables “val”, “a”, “b”, and “t” are in the inner loop. The code in the inner loop will be executed N^2 times, which will need much time to compute. The solution is changing the order of the loop, which puts the “i” loop in outer and puts “j” loop in the inner part. By doing this loop change, the computations of variables “val”, “a”, “b”, and “t” will only be executed N times. After this step, the running time improved from 17 seconds to 0.86 second.

After this step, the running time is very fast. But the function fmod and round are still low efficient. In the program, the fmod takes 2 integers and returns 1 double. The function round is useless, because the fmod returns a double without number after decimal points. After this step, the running time improved from 0.86 second to 0.74 second.

Conclusions

The redundant computation in the inner loop is the key factor that slows the running time. And some expensive operations such as cos, sin, and fmod should be avoided. And, checking the data type is necessary to avoid some useless type conversion.

Part2

Problem statement

The program in part2 is the multiplication of the matrixes. But the dimensions of the matrices are very large, and there are 3 nested loops in this program, so the time complexity of this program is N^3. Moreover, because the huge dimensions of matrices, the data access and write speed are very slow. The reason for this slow data flow speed is the data cannot be stored in cache because of the data size. By meeting many cache misses, the access and write of the speed become very slow. Finally, the running time of this program is 76 seconds, which is unaccepted for daily use.

Solutions

The most important part of improvement is finding a method to reduce the occurrence of cache miss and increase the data access and write speed. If I use the original method, the data access method is computing the inner product of the matrix directly. If we assume the dimensions of two matrices are N*N, in every iteration, the CPU accesses the data of one row and one column of two matrices. However, if the dimension N is very large, the cache can't hold the data because of its limited size. The CPU has to get the data from the outer cache, even memory, so the data flow speed and the running time of this program have been impaired. So, I make improvements on data access optimization. In the main function, two original matrices have been divided into many small matrices, and the main function calls the helper function to do the matrix multiplications for those small matrices. The helper function does the same computation as the original matrix multiplications function, but the size of matrices is in fitted size so that all data can be accessed directly from cache. Then the final output is the sum of the results of each helper function. After this improvement, the running time is 8.5 second, which is much better than the original program.

But there is one place that can be improved further. The code of computation of the output array is

C[i * N + j] += A[i * N + k] * B[k * N + j]. This computation is based on the access data of the array. I use a variable "c" to store the value of the array in each iteration. This improvement avoids the data access in each iteration. After this improvement, the running is 7.7 second.

Conclusions

The data access speed is an important factor in performance. If the cache miss rate increases, the performance will be seriously weakened. Thus, the method to improve this program is avoiding the cache miss in execution. After the optimization of data access, the running time decreases tremendously.