COMP90025
Parallel and Multicore Computing
Project1A Report
Chuan Yang chuany1
Mengchen Wang mengchenw1

1. Introduction

In this part of project, we use parallel computing to accelerate the Floyd-Warshall algorithm to find the diameter of a graph, which is the largest value of distance of any pair of nodes. OpenMP is used in experiments, the task is run on a single node, which contains eight cores of cloud machine.

2. Parallelization and Optimization

The essence of Floyd Warshall algorithm is working from local optima to global optimum. In this problem, we identify the shortest path for any pair of nodes by using k nodes as an intermediate step stone; as k approaches the total number of nodes in the graph, the shortest path can be found. We parallel the two inner loops for better performance due to the dependence of nodes in the algorithm. Although doing fork and joint inside another loop can be a costly operation, the algorithm dictates that the k-th loop must not start before k-1-th loop finishes as it relies on the recursion

2.1 Optimization

In the original algorithm, there are three comparisons and two boolean operations in the third level loop's if statement, so for a dense matrix, there would exist about 3*n^3. However, those 3 comparisons are not necessarily needed if we do not use -1 to represent NOT_CONNECTED. So we adapted another better way in which we used a very large number(that means all the valid distance between nodes must be smaller than this value) to represent NOT_CONNECTED. The good result of this method is that only one comparison is needed in the innermost loop rather than 3, speeding up at most 3 times.

2.2 Parallelization

In the algorithm, the main part is 3-level loop, which can be paralleled with openMP explicitly. So we added the openMP's parallelization of 'for' out of the whole loop. Considering the loop count is far larger than the available threads(cores), we do not think fine granularity is necessary, so collapse is not adapted in our experiments.

We have another improvement on schedule to increase the performance. For different k value, the valid path may vary a lot. This means that each processor can potentially have large variance in its workload. Hence using dynamic scheduling can make sure that we take advantage of all the computing resources.

Another parallelization we have done is on finding the maxima of shortest distance from the Floyd-Warshall output matrix. As the number of nodes increase, the matrix size grows at the order of n^2. And from our test result, performance increased achieved by parallelization outweighs the cost of doing forks and joints..

Result Comparison

Test Data Size (node size)	Sequential Runtime (s)	Parallel Runtime(s)	Speedup ratio
1000	4.6007	2.9240	1.6
2000	37.2554	8.9295	4.2
4000	293.8157	41.9919	7.0
8000	2347.5136	256.8571	9.1

Table1. Runtime and Speedup Ratio of Sequential and Parallel Algorithm for different test case

In Table1, we run the sequential and parallel program on Spartan physical partition with 8 cores and get corresponding result. We can see that the parallelization scales for various problem size.

3. Discussion and Conclusion

Originally, we tried to incorporate the finding max distance part into the FW triple loop, but we found that if a local maxima is detected, the output will not

be correct due to the logic of algorithm (Only the maximum of distance is reported. In this case, it is not even the distance between two nodes). Thus, what we can do is another parallel threading at the find maximum algorithm. By replacing the NOT_CONNECTED value with 1000000, we make an optimistic assumption that no 2 nodes in a given graph may have an edge weights larger than this. This assumption is based on the sample test data and maximum size of nodes

From the experiments, we can see that with openMP, we can obviously accelerate the algorithm. The speedup-ratio for the algorithm is rising with the increase of data size, and it can be larger than 9 with 8000 data size. This may be because that if the workload is larger, the program can make better use of parallel threads, but for other parts which contain simple logic, the cost of threads fork makes a relatively larger percentage.