COMP90025 Project 1A: Diameter of Graph

Xinyan Shan (1047188) — Spartan username: sxy0322
Saier Ding (1011802) — Spartan username: Saier

Introduction

OpenMP is an application programming interface which is designed to process massively parallel data on multiprocessor shared memory machines.

In a graph, Floyd-Warshall (FW) algorithm, a kind of All Pairs Shortest Path (APSP) algorithm, has been widely employed to calculate the lowest weights. However, the implementation of this algorithm includes three loops, so the algorithm complexity achieves $O(N^3)$, where N is the number of the nodes in a given graph. Indeed, with the increase of the nodes, there is a rapid growth in the elapsed time, as shown in the figure 1. The long runtime restricts the adopting of this sequential version in practical applications involving plenty of nodes.

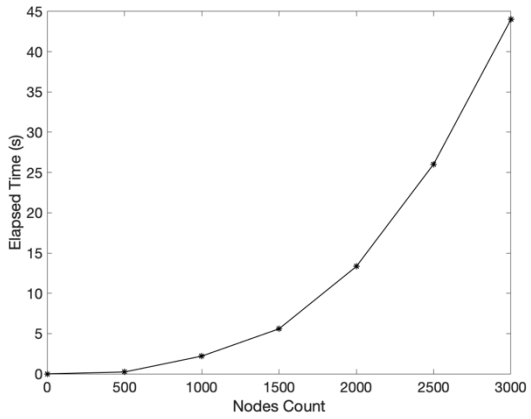

Figure 1. Performance of sequential diameter algorithm

In fact, there are other methods to improve the performance of shortest path algorithm. For example, Xiao and Lu (2010) propose the novel method by employing the alternative algorithm, Dijkstra algorithm. Albalwi, Thulasiraman and Thulasiram (2013) present the task level parallelization to speed up the shortest path algorithm in OpenMP. Both methods achieve a better performance than other traditional strategies.

Hypothesis and Method

Based on the analysis of time complexity, the hypothesis that rational planning of thread resources in OpenMP can improve the performance of traditional method for calculating the diameter of a given graph is proposed.

The first solution, which makes every thread process strips of column in parallel, is to distribute the i row of data into parallel threads, after identifying the mediate node k, but this method is cache friendly.

The second solution is the same as the first one, but the OpenMP directives are different. Instead of employing the statement ***#pragma omp parallel for*** before the second loop, ***#pragma omp parallel*** is exploited before the first loop, and typing ***#pragma omp for*** before the second loop. Meanwhile, in the third loop, the if statement which examines whether there is not a path between a pair of nodes can be deleted, because the value of this kind of path is always larger than others.

Results and Analysis

Figure 2 and 3 show the elapsed time of two solutions separately.

Nodes / Threads	1000	2000	3000	4000	5000
Serial	2.05	16.27	53.83	124.73	237.03
2	0.64	5.05	17.29	44.55	91.45
4	0.34	2.57	8.66	22.22	44.88
6	0.23	1.80	5.98	14.95	30.05
8	0.18	1.32	4.36	10.90	23.22
10	0.15	1.18	3.71	8.63	18.36
12	0.13	0.95	2.96	7.60	15.74

Figure 2. The elapsed time (s) of solution 1 based on the different number of nodes and threads

In fact, the first solution distributes all the computation into several threads, and each thread runs all n iterations of j, but only n/t iterations of i, and therefore every thread processes strips of columns in parallel. However, in this solution, the iteration k is the global shared variable.

Nodes / Threads	1000	2000	3000	4000	5000
Serial	2.05	16.27	53.83	124.73	237.03
2	0.50	3.95	14.09	37.20	78.24
4	0.26	1.99	6.94	18.24	37.68
6	0.18	1.33	4.95	11.64	25.25
8	0.14	1.01	3.46	8.65	19.21
10	0.12	0.81	2.81	7.18	16.10
12	0.10	0.72	2.34	5.67	13.02

Figure 3. The elapsed time (s) of solution 2 based on the different number of nodes and threads

Consequently, in the second solution, k has been privatized by declaring the parallel directive before the first loop, so each thread processes its own private i. According to the data in the Figure 3, the elapsed time has a significant drop.

Conclusion

Obviously, both the two solutions improve the performance of the sequential version effectively by employing OpenMP. Furthermore, the second solution is the more efficient method to decrease the runtime of the traditional graph diameter algorithm. In order to better measure the performance of the method, the speedup ratio is introduced. The speedup ratio can be calculated by the following formula:

$$R(t,n) = T_s(n)/T_p(t,n)$$

where t represents the number of threads; n is the number of the nodes in a given graph; T_p indicates the time spent by the parallel solution; T_s represents the time spent by the serial solution. According to this formula, the speedup ratio of the solution 2 is shown in the Figure 4.

Nodes / Threads	1000	2000	3000	4000	5000
2	4.10	4.12	3.82	3.35	3.03
4	7.88	8.17	7.76	6.84	6.29
6	11.39	12.23	10.87	10.72	9.39
8	17.86	16.11	15.56	14.42	12.34
10	17.08	20.09	19.16	17.37	14.72
12	20.50	22.60	23.00	22.00	18.21
Max	20.50	22.60	23.00	22.00	18.21

Figure 4. The speedup ratio of solution 2 based on the different number of nodes and threads

The elapsed time can be 13.02s when the diameter of 5000 nodes is computed by the second solution. The maximum speedup ratio achieves 23.00 when a graph involves 3000 nodes and is calculated by 12 threads. Generally, as the increase of the number of nodes, the speedup ratio declines gradually. Moreover, the more threads involved in the computation, the higher of the speedup ratio the parallel algorithm produces.

This experiment proves that the graph diameter program based on Floyd-Warshall algorithm can be improved by adopting the proposed solution, in terms of the computing efficiency.

Reference List

Albalwi, E., Thulasiraman, P., & Thulasiram, R. (2013). Task Level Parallelization of All Pair Shortest Path Algorithm in OpenMP 3.0. *Proceedings of the 2nd International Conference on Advances in Computer Science and Engineering*. doi: 10.2991/cse.2013.26

Xiao, J. X., & Lu, F. L. (2010). An improvement of the shortest path algorithm based on Dijkstra algorithm. *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE), 2*, 383-385. doi: 10.1109/iccae.2010.5451564

COMP90025 Project 1A: Diameter of Graph

Xinyan Shan (1047188) — Spartan username: sxy0322
Saier Ding (1011802) — Spartan username: Saier

ILLIAC is an array machine that uses 64 processing units to process under unified control. The central processing unit of this supercomputer is divided into four controllers that can execute separate instruction sets, each controller managing several processing units for a total of 256 processing units. The first computer to fully use large-scale integrated circuits as logic components and memory is the US ILLIAC-IV which marks the fourth generation of computer development.

We believe that our submitted program can be operated on ILLIAC IV. Massively parallel computer ILLIAC IV is characterized by its SIMD (Single Instruction Multiple Data) architecture. In fact, the branch statement hinders the performance of vector computation from improving, because the program counter value might be ambiguous when processing the branch statement simultaneously. However, in ILLIAC IV, each PE is equipped with a "model" register, which indicates whether the specific PE (processing unit) conducts the current operation. Indeed, there are several if statements in the program submitted, but based on the "model" register, ILLIAC can operate the program in parallel properly, because the add operation can be vectorized in the end of calculating the length of shortest path between nodes. In fact, an OpenMP SIMD version is applied in our test program, we construct a vector every four data in the third loop. However, the complier could not recognize the flag "-fopenmp-simd" under the c99 standard, so the declaration #pragma omp simd is omited.