Parallelization of graceful labeling using OpenMP

K.Aravindh*,R.P.Santharuban**,S.Sanjay***
*(Computer Science and Engineering, Kamaraj College of Engineering and Technology
Email: aravindh.vel1999@gmail.com)
*(Computer Science and Engineering, Kamaraj College of Engineering and Technology
Email: sanjaymanian@gmail.com)
*(Computer Science and Engineering, Kamaraj College of Engineering and Technology
Email: santharuban97@gmail.com)

----------**********----------

Abstract:

Parallelization is the process used for reducing the time that provides an efficient result for the real time applications. Multi-core architecture is a general purpose processor that consists of multiple cores on the same die and can execute programs simultaneously and parallelization can be applied effectively. Though the multiple cores are available in multi-core architecture, only single core is utilized unless the programmer intervenes. It is very crucial to use the multi-cores effectively. There are many applications such as communication network addressing, X-Ray Crystallography, Radar Communication, Astronomy and Circuit design which use graceful graph labelling problem for finding solutions. In real time, solving graceful graph labelling problem is time consuming process when numbers of nodes are processed sequentially. In this work, parallelization is applied to the Graceful Graph labelling problem in multi-core using OpenMP. It is found that speedup and execution time are reduced. After parallelization, the speed up is constantly improved as the size of the graph becomes large.

Keywords — **Multi-core, OpenMP, Graceful, Graph Labelling, Speedup, Execution Time .**

----------**********----------

1. INTRODUCTION

The program which uses a large number of data as an input takes more time for the execution. This issue in the execution is because of the serial execution of the program. To avoid this issue the proposed system uses the OpenMP to execute the program in a parallel way.

Parallelization is the act of designing a computer program or system to process data in parallel. Normally, computer programs compute data serially. They solve one problem, and then the next, then the next. If a computer program or system is parallelized, it breaks a problem down into smaller pieces that can each independently be solved at the same time by discrete computing resources. When optimized for this type of computation, parallelized programs can arrive at a solution much faster than programs executing processes in serial.

Parallel processing is the method of evenly distributing computer processes between two or more computer processors. This requires a computer with multiple CPUs, or a CPU (or GPU) equipped with multiple cores. It also requires an operating system capable of supporting parallel processing, or software written specifically to process tasks in parallel. Parallel processing is the method of evenly distributing computer processes between two or more computer processors. Each new generation of processors approaches the physical limitations of microelectronics, which is a major engineering concern in CPU design.

Because individual chips are approaching their fastest possible speeds, parallel processing becomes an important area in which to improve computing performance. The majority of modern desktop computers and laptops have multiple cores on their CPU that help parallel processing in the operating system.

.

1.2 Multi-core Architecture

Multi-core refers to an architecture in which a single physical processor incorporates the core logic of more than one processor. Multi-core architecture places multiple cores and bundles them as a single physical processor. The objective is to create a system that can complete more tasks at the same time, thereby gaining better overall system architecture

A single integrated circuit is used to package or hold these processors. These single integrated circuits are known as a die. Multi-core architecture consists of multiple cores on the same die and can execute programs simultaneously, thereby gaining better overall system performance. The processing tasks are divided into sub tasks and assigned to different cores. At the time of task completion, the processed data from each core is collected and combined. This technique significantly enhances performance compared to a single-core processor of similar speed.

1.3 OpenMP

Open MP stands for open multi-processing. OpenMP uses a portable, scalable model that gives programmers a simple and flexible interface for developing parallel applications for platforms ranging from the standard desktop computer to the supercomputer. Open MP is an application programming interface (API) that supports multi-platform shared memory multi-processing programming in C, C++, and Fortran, on many platforms, instruction set architectures and operating systems, including Solaris, , Linux, macOS, and Windows. OpenMP consists of a set of compiler directives, library routines, and environment variables.

OpenMP is an implementation of multithreading, forks a specified number of slave threads and the system divides a task among them. The threads then run concurrently, with the runtime environment allocating threads to different processors.

3. System Design

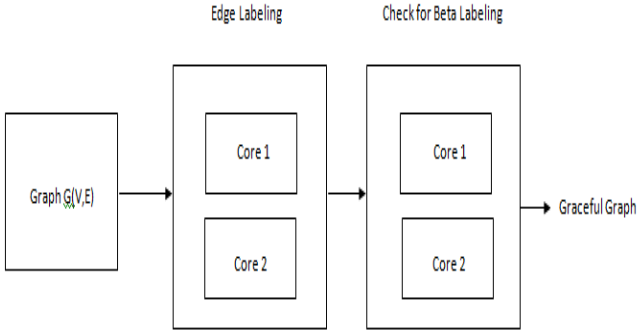

Fig 1:System Design

This system is designed in such a way that to parallelize the graph labelling for different inputs. Normally, a graph consists of number of vertices and edges. The system accepts the edges and vertices as an input. The labelling can be calculated by finding the difference between the adjacent vertices. The checking of alpha labelling is performed. These processes are parallelized. Speedup is calculated based on the time taken for the execution of both serial and parallel program. The results are compared and analysed.

2. Execution Time

In this module, the execution time to find out the weight for the graph is calculated by using equation.

$$\text{Execution time} = \frac{(\text{float})(\text{end} - \text{begin})}{\text{Clocks per second}}$$

Where,

Begin-time at which program execution starts

End-time at which program execution ends

Execution time – Time taken to execute the program

The factor by which multi-core shows improvement in finding solution is called speed up the overall speedup *S* is given by the below equation .

$$\text{Speedup, } S = \frac{\text{Tseries}}{Tparallel}$$

Where,

Tserial – serial program execution time
Tparallel – parallel program execution time

No of Vertices	Serial Execution Time(Micro sec)	Parallel Execution Time(Micro sec)
10*10	0412	0371
50*50	2313	2062
100*100	72501	65210
150*150	18751	16782
200*200	27983	26123
250*250	42516	41710
300*300	66543	61287
350*350	78142	60286
400*400	90421	84371
450*450	111242	107342

Table 1:Execution Time

Table 1 shows that for 10*10, there is no much difference in execution time between serial and parallel execution times. However, for 450*450, there is a considerable amount of difference between serial execution time and parallel execution time. When the number of vertices increases, the corresponding difference between serial and parallel execution time also increases.

3. Speedup

The speedup is calculated by using the fraction of code parallelized with speed obtained from serial and parallel execution time which is shown in Table 2.

No of Vertices	Speedup
10*10	1.112
50*50	1.121
100*100	1.123
150*150	1.125
200*200	1.125
250*250	1.123
300*300	1.127
350*350	1.130
400*400	1.134
450*450	1.148

Table 2 Speedup

4. CONCLUSIONS

In this work, we learned how OpenMp programming techniques are beneficial to multicore systems. We also found out the execution time of both serial and parallel programs. From the vertices 10 to 450, it shows consistent improvement. It is concluded that by parallelization the speedup, and execution time are improved.

.

5. REFERENCES

[1] Brankovic,&Wanless (2011). Graceful Labelling: State of the Art, Applications and Future Directions. Mathematics in Computer Science, 5(1), 11–20. doi:10.1007/s11786-011-0073-6

[2] Yamazaki ,Kurzak , Wu , Zounon , &Dongarra (2018). Symmetric Indefinite Linear Solver Using OpenMP Task on Multicore Architectures. IEEE Transactions on Parallel and Distributed Systems, 29(8),

1879–
1892. doi:10.1109/tpds.2018.2808964

[3] Peipei , Tai , &Yuanyuan(2014). The Generation of k-Graceful Figure and Graceful Label.2014 Fourth International Conference on Instrumentation and Measurement, Computer, Communication and Control. doi:10.1109/imccc.2014.94

[4] Zeng K, Tang Y, & Liu F (2011). Parallization of Adaboost Algorithm through Hybrid MPI/OpenMP and Transactional Memory.2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing. doi:10.1109/pdp.2011.97

[5] SheelaKathavate and N.K. Srinath, Efficiency of Parallel Algorithms on Multi Core Systems Using OpenMP, International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 10, October 2014.

[6] ChetanArora, Subhashis Banerjee, PremKalra, and S.N. Maheshwari, An Efficient GraphCut Algorithm for Computer Vision Problems, Department of Computer Science and Engineering, Indian Institute of Technology, New Delhi, India@Springer-Verlag Berlin Heidelberg 2010.

[7] P.K.D Sridevi,A.Sindhuja,R.Muthuselvi "Parallelization of maze generation and solving in multicore Using OpenMP" International Conference on Innovations in Engineering and Technology, ICIET 16, 6th and 7th April 2016 conducted by KLN College of Engineering,Madurai.

[8] R.Muthuselvi, M.Muneeswari, K.Sudha, V.Vasantha "Parallelization of Graph Labelling Problem in Multicore using OpenMP**"** International Conference on Trends in Electronics and Informatics ICEI 2017 May 11th and 12th 2017 conducted by SCAD Engineering College, Thirunelveli

[9] Rohit Chandra, Leonardo Dagum, Dave Kohr, DrorMaydan, Jeff McDonald, Ramesh Menon ' Parallel Programming in OpenMP' ch.1-6, pp.1-249.

[10] J.A. Mac Dougall, M. Miller, Slamin and W. D. Wallis,Vertexmagic total labelling og graphs,Util,Math.,61(2002)3-4.

[11] K.A. Hawick, A. Leist and D.P. Playne, Parallel Graph Component Labelling with GPUs and CUDA,International Journal of Engineering and Technology Vol.3,2014.

.