IJCSET |October 2012 | Vol 2, Issue 10, 1465-1467 www.ijcset.net | ISSN:2231-0711

An Automatic Parallization in Multicore Computer Using OMP

Bharti Sen, Sanjay Kumar

Pt. Ravishankar Shukla University, Raipur (Chhattisgarh) 492010 India
[1]bhartisen123@gmail.com
[2]sanraipur@rediffmail.com

***Abstract*— The past several years has witnessed an ever-increasing acceptance and adoption of parallel processing. We propose dynamically adaptive programs, programs that adapt themselves to their runtime environment. We discuss the key issues in successfully applying this approach and show examples of its application. The method is based on the data dependency analysis of loops and well-known loop parallelization techniques. The OpenMp standard was chosen for representing the parallelism of algorithms.**

***Keywords*—Super-Scalar processor,Performance of scalar processor, Performance of Uniprocessor and Multi-Processor, Comparisions between Multi-processors and Uniprocessor, Pipelining.**

I. INTRODUCTION

OpenMP enables the creation of shared-memory parallel programs. In this paper a brief overview of parallel computing and the main approaches taken to create parallel programs are discussed.Using OpenMP to parallelize an application is not hard. In general, the effort of parallelizing a program with OpenMP goes mainly into identifying the parallelism, and not in reprogramming the code to implement that parallelism. One of the toughest problems facing an OpenMP programmer is to learn how to avoid introducing bugs into a program as part of the parallelization process.

The OpenMP Application Programming Interface (API) was developed to enable portable shared memory parallel programming. It aims to support the parallelization of applications from many disciplines. Moreover, its creators intended to provide an approach that was relatively easy to learn as well as apply. The API is designed to permit an incremental approach to parallelizing an existing code, in which portions of a program are parallelized, possibly in successive steps. In this paper performance evaluation of Super-Scalar processor, Scalar-processor and uniprocessor are evaluated on the basis of two parameters(speedup and no. of clock cycle required to execute the instruction) by using OMP, Comparision between Multi-processor and Uniprocessor are also explained.

II. PIPELING

In computing, a pipeline is a set of data processing elements connected in series, so that the output of one element is the input of the next one. The elements of a pipeline are often executed in parallel or in time-sliced fashion; in that case, some amount of buffer storage is often inserted between elements [9]. A linear pipeline processor is a series of processing stages which are arranged linearly to perform a specific function over a data stream. The basic usages of linear pipeline are instruction execution, arithmetic computation and memory access.A non-linear pipeline (also called dynamic pipeline) can be configured to perform various functions at different times. In a dynamic pipeline there is also feed forward or feedback connection. [9]

III. UNIPROCESSOR AND MULTI-PROCESSOR STYLE

A. Uniprocessor

A uniprocessor system is a computer system with a single central processing unit.[7]

B. Multiprocessor

A multi-core CPU is a computer processor which has two or more sections. Each section of the chip executes instructions as if it was a separate computer. The actual processors are still on one chip. On this chip every core looks mostly like the other. They are several mostly independent cores which work together in parallel. A dual-core processor is a multi-core processor with two independent microprocessors. A quad-core processor is a multi-core processor with four independent microprocessors..Scalar processors represent the simplest class of computer processors. A scalar processor processes one datum at a time (typical data items being integers or floating point numbers. A scalar processor is classified as a SISD processor (Single Instructions, Single Data) [8].

A superscalar CPU architecture implements a form of parallelism called instruction level parallelism within a single processor. It therefore allows faster CPU throughput than would otherwise be possible at a given clock rate. A superscalar processor executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor. Each functional unit is not a separate CPU core but an execution resource within a single CPU such as an arithmetic logic unit, a bit shifter, or a multiplier.[7]

IV .EXPERIMENTAL SETUP

This works were performed using OpenMP in Fedora16 Operating System. OMP is chosen as the simulation tool because it supports an automatic Parallelization in both Multi-core Processors and Unicore Processors. OpenMP enables the creation of shared-memory parallel programs. For experiments Intel Core2 Duo computer systems

were taken With 1.5 GHz speed. Matrices of different orders were multiplied with and without OMP. Computational times were recorded as shown in different tables and graph was plotted .Performance analysis of both Multi-Core Processor and UniCore-processors are discussed, Computational values are recorded in different tables and graph was plotted according to computational values.

V. PERFORMANCE MATRICES

A. *Speed Up*

Speedup is defined by the following formula:[12]

$$S_p = \frac{T_1}{T_p}$$

where:

- p is the number of processors
- T_1 is the execution time of the sequential algorithm
- T_p is the execution time of the parallel algorithm with p processors

B *C.P.I*

Clock Cycle required to execute an instruction.

VI. RESULT AND DISSCUSION

It is clear from different tables and graph that execution speed of Multiprocessor is much high as compare to that of Uniprocessor.Total no. of clock cycles required for execution in Multiprocessor is much less than that of Uniprocessor.It is Oberseved that on comparision between Scalar and Super-scalar Processor, Computational time required for execution of an instruction in Superscalar is much less than that of Scalar Processor.Also it is obsereved from table OMP is more than computational time without OMP. Beyond that computational time with OMP becomes less than the computational time without OMP. This shows that size of problem also matters with parallelism. When size of problem is low, then because of under utilization of resources more overheads in parallel environments computation time with parallelism is more.

TABLE-I

Scalar Proccessor (One processor with pipelining)

Proccesors	No. of instruction	C.P.I	Speed up	No.of Stages	No.of Processor
Scalar	3	7	2.1	1	1
Scalar	6	10	3	1	1
Scalar	9	13	3.4	1	1

TABLE-II

Uniprocessor(without pipeling)

Proccesors	No. of instruction	C.P.I	Speed up	No.of Stages
Uniprocessor	3	15	1	1
Uniprocessor	6	30	1	1
Uniprocessor	9	45	1	1

TABLE-III

Super-Scalar Proccessor or Multi-processor(with pipeling)

Proccesor	No. of instruc -tion	C.P.I	Speed up	No.of Stage	No.of Processor
Scalar	3	7	2.1	1	1
Scalar	6	10	3	1	1
Scalar	9	13	3.4	1	1
Super-scalar	3	5	3	1	3
Super-scalar	6	6	5	1	3
Super-scalar	9	7	6.4	1	3

TABLE-IV

Scalar Proccessor Vs Super-scalar Processor

Proccesor	No. of instru -ction	C.P.I	Speed up	No.of Stages	No. of proceesor
Super-Scalar	3	5	3	1	3
Super-Scalar	6	6	5	1	3
Super-Scalar	9	7	6.4	1	3

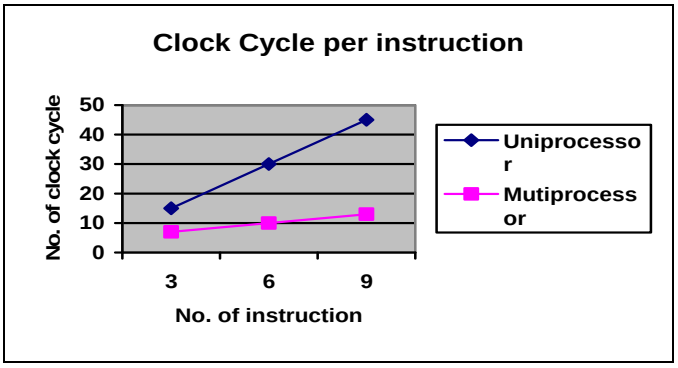

Fig. 1 Scalar Vs Uniprocessor

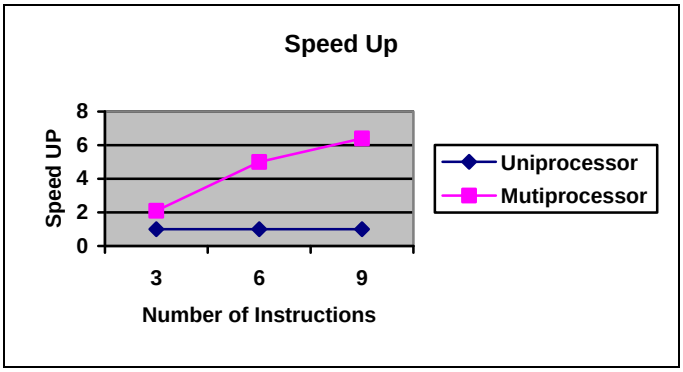

Fig. 2 Scalar Vs Uniprocessor

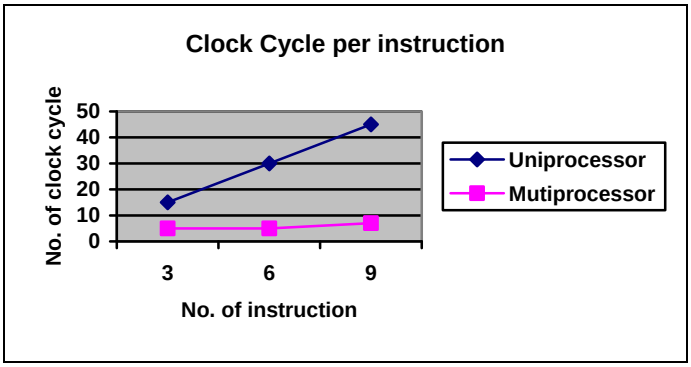

Fig. 3 Superscalar Vs Uniprocessor

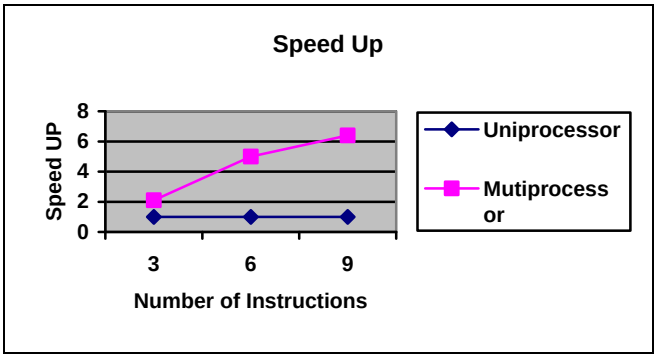

Fig. 4 Superscalar Vs Uniprocessor

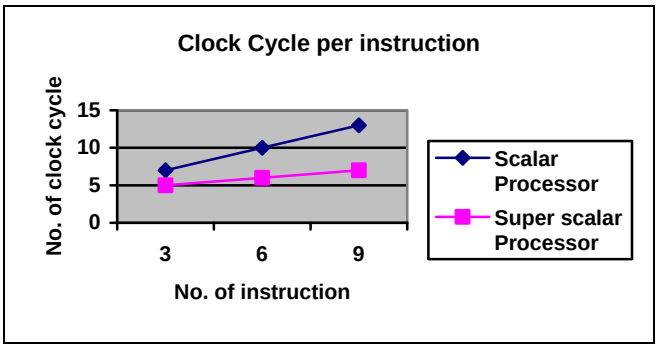

Fig. 5 Superscalar Vs Scalar

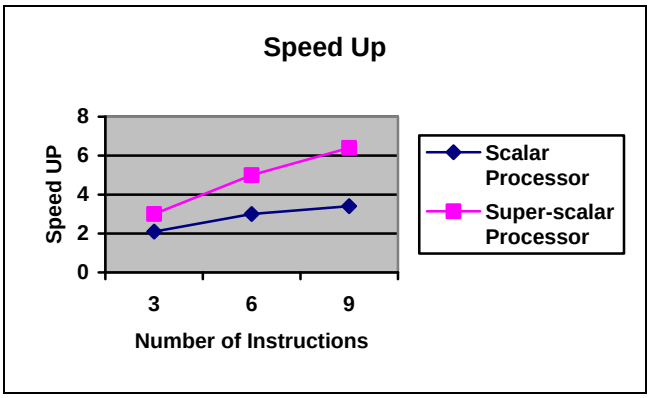

Fig. 6 Superscalar Vs Scalar

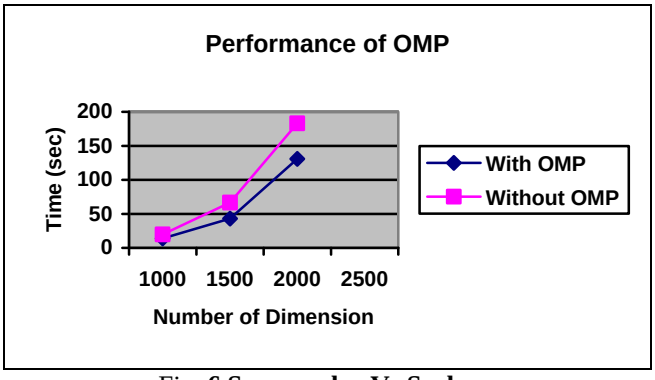

Fig. **6 Superscalar Vs Scalar**

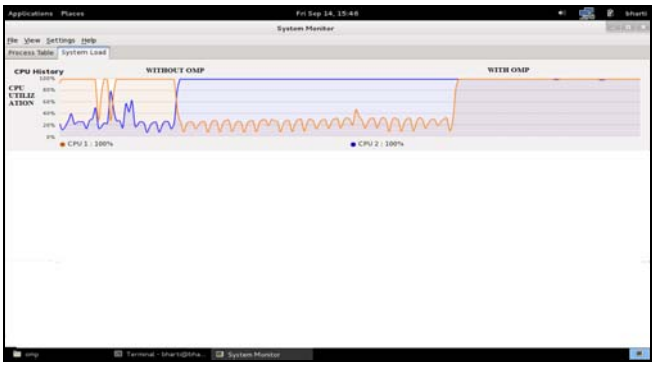

Fig. 7 Performance anaylsis with OMP and without OMP

VI. CONCLUSION

On the basis of Performance evaluations, it is observered that No.of clock cycle and speed up in Multi-Processor is much less than as compare to that of Uniproccesor. OMP is more than computational time without OMP. Beyond that computational time with OMP becomes less than the computational time without OMP.

REFERENCES

[1] GeorgiosToumavitis,ZhengWang,BjomFranke.MichaelF.P O'Bolye,,*"Towards a Holistic Approach to Auto-Parallelization",2011*.

[2] D.Bailey,T.Harris,W.C.Saphir,R.F.VanderWijngaart,A.Woo,M.Yarrow. *"The NAS Parallel Benchmarks 2.0,NAS Technical Report NAS-95-020"*,1995.

[3] E. Barton, J. Cownie, and M. McLaren., "Message Passing on the Meiko CS-2,Parallel Computing".,20(4):497-507, 1994.

[4] *Parallel Computing ForumPCF Parallel Fortan Extansions*,V5.0.ACM Sigplan Fortan Forum,10(3):1-57, 1991.

[5] *http://Open64.sourceforge.net*,2006

[6] *http://www.beowulf.org*, 2006.

[7] *http*://en.Wikipedia.org/wiki/Superscalar.

[8] *http*://en.Wikipedia.org/wiki/Scalar.

[9] *http*://en.Wikipedia.org/wiki/Pipeling.