International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
International Conference On Quality Up-gradation in Engineering, Science & Technology
(IC-QUEST- 11th April 2015)

RESEARCH ARTICLE OPEN ACCESS

Performance Analysis of Fast Fourier Transform For Complex Data Vectors on Different Systems Using Sequential and Parallel Approach

Mrs. Saroj A. Shambharkar, Mr. Manish Bhardwaj
Kavikulguru Institute of Technology and Science *Ramtek-441 106* Nagpur, India
saroj.shambharkar@gmail.com
Kavikulguru Institute of Technology and Science *Ramtek-441 106Nagpur, India*
manish.bhardwaj1sept@gmail.com

Abstract
The Fast Fourier Transform is very useful in many applications like weather forecasting, Computational Fluid Dynamics Problem, Automobile crash Analysis. In this paper ,the execution of Fast Fourier Transform is analyzed and the results are compared based on serial and parallel programming on different processors. When the execution of Fast Fourier Transform of complex data vectors using OpenMP for parallel execution is tested then it is giving better results as compared to serial execution. In the parallel approach the result is varying with the number of cores available in the system. The program using OpenMP is tested by varying the value of N. N is length of vector. To measure the performance using Fast Fourier Transform ,the parameters used for comparison are Time and MFLOPS. The programs of Fast Fourier Transform based on different approaches were tested on dual core and quad core machines. The first model used is Intel(R) Pentium(R) CPU G640 consider for testing with processor speed of 2.80GHz and CPU's speed is 1700 MHz and second model Intel(R) Core(TM) i3 with processor speed is 2.53 GHz and CPU's speed is 931 MHz Both the models having different processors and they are giving different performance. It has been observed that the performance of Fast Fourier Transform is found improved when the processors executed their code using parallel approach as compared to sequential approach and we can say its a better utilization of processor time . On Dual core processor for N value 33554432, the performance is 502.27 MFLOPS and time required is 16.7 seconds using sequential approach and the performance is 681.6 MFLOPS and time required is 12.30 seconds using parallel approach. On Quad core machine, we have achieved successfully the performance of about 597.27 MFLOPS and time required is 14.04 seconds using parallelism and for the same N value 33554432,the getting on quad core processor and for sequential execution getting 368.32 MFLOPS and time required is 22.77 seconds .
Index Terms: Scalable clusters, hyper threading, cache coherence, Instruction level Parallelism, super scalar architecture, Digital Signal Processor.

I. INTRODUCTION

The Discrete Fourier Series is technique invented by C. F. Gauss in 1805, helps in computing coefficients efficiently. After that, J. W. Cooley and J. W. Tukey in 1965 introduced another technique which is commonly called as Fast Fourier Transform. The Fast Fourier Technique was implemented on computer system for the same purpose that is to computing the coefficients of a discrete Fourier series but it is known to be faster than the earlier one. Many engineers are making use of the Fast Fourier Transform in many applications and it is one of the most important techniques found in many fields of engineering, science, and mathematics. A Fast Fourier Transform algorithm is also used to compute the inverse of discrete Fourier transform. And most of the Fast Fourier Transform algorithms sometimes, work efficiently only when the input size having the power of a small prime number [11].

Now coming to the point that why one has to go for parallelism. In the past few decades the components used to build both high-end and desktop machines have continually decreased in size. Billions of transistors on a single chip were found as technology progress. As data paths became shorter, the rate at which instructions were issued could be increased. Raising the clock speed became a major source of advances in processor performance. Advances limitations due power consumption and heat emission, which is increasingly hard to dissipate. Due to this reason computer architects have begun to emphasize other strategies from increasing hardware performance and making better use of the available space on the chip[6].

Multiple processors that share memory are configured in a single machine and increasingly on chip. The advantage of using new generation shared – memory parallel computers are they are not very costly and intended for general purpose usage. Some recent computer designs permit a single multiple instruction streams in an interleaved way. Simultaneous multi threading for example interleaves instructions attempt to use more of the hardware components at any given time. This is everyone tries to do. In hyper threading technology, a computer might multiply two values from one set of instructions, same time; fetch a value from memory that is needed to perform an operation in a different set of instructions. Supercomputing replaced by scalable clusters of commodity multi core processors, Earlier the people are using workstation, now the desktop or laptop's with multi core system in more demand. In parallel programming it is not very simple, it is quite difficult for a processor, to know about which processor is thread is going to run which thread. Architectural and program constraints must be considered in scaling up any parallel program [6].

Instruction level Parallelism is a very low level of parallel processing. A processor that supports ILP said to have a super scalar architecture. Today's computer systems are made up of: (1) Multiple components, (2) Functional units. These two may be able to operate simultaneously and have specific tasks. Consider an example of where the processor of the computer is given a task to perform multiplication of two floating point numbers and some other task as determining whether the given value is greater than zero or not. To perform both tasks at the same time computer might fetch the data from the memory and at same time evaluate a branch condition, which is a instruction level parallelism. These operations are carefully recorded than we may keep the machine's components busy. The suitable ordering of operations is performed by the compiler. This is a common feature Instruction Level Parallelism in general purpose microprocessors. This is a common feature used in our Laptop's and PCs [6].

How compiler helps to achieve this or one can accomplish this, compiler writers developed techniques to determine dependencies between operations. And keeps many functional units and paths to memory busy with useful work. The Fast Fourier transform is the fastest known algorithm. It is used to bridge the gap between polynomial arithmetic and integer arithmetic and matrix computation. It is used for performing the basic arithmetic operations on integers, polynomials and dense structured matrices [2].

II. RELATED WORK

Da Lu and Hanging Fan, implement the modified FFT based on complex sequence on shared memory parallel computers. They run their code for one-dimensional FFT's. The MLGFIM-FFT algorithm consists of the near- field and far-field interaction, and for near-field and far-field interactions used OpenMP, to- assign approximately the same number of unknowns to each thread [3].

SHJ and Chan developed MLGFIM-FFT algorithm based on parallel programming using OpenMP standard on a shared – memory computer system [4].

According to Hang Zhou, Xirmin Wang and Ying Hua Guo, they describe in their paper in order to avoid the migrating of threads scheduling of Linux component can be bind to core, a thread binding interface of the FFT implementation in java .In our approach performance of cores analyzed by single core and multiple cores on different systems [5].

Digital signal processors have in great degree since it capable to execute millions instruction per second (MIPS) and able to perform difficult digital signal processing algorithms for different embedded electronic equipment. Multi core DSP capable to perform computation at higher speed with less power requirement so it is best for industry and academia purpose .TMS320C66XSoc, from Texas instrument is a multi core digital signal processor where the Fast Fourier transform algorithm was implemented. A FFT parallelization strategy was chosen and implemented in software with inter processor communication and open multiprocessing multi core programming technique. The authors used TMS320C66X device which is having 4 core architecture and which was supported by multi core software Development kit, which includes a high optimized digital signal processing library with typical DSP functions including FFT[9].

Signal processing and other scientific applications required high speed computations that can achieve by new approaches of architecture with better algorithm and high speed computing hardware. By this most of the computation work required data flow processor, Fast Fourier transforms. Processor, digital filters, symbolic arrays and signal flow graphs or data flow graphs can convert into synchronous symbolic arrays or data-driven wave front arrays. The wave front-oriented programming language used to describes the parallel data flow in symbolic or wave front-type arrays[10].

III. MULTICORE SYSTEMS

Multi core revolution demands simple parallel application development. Replicating substantial parts of a processor's logic on a single chip and behaving much like shared- memory parallel machines. This

approach is known as multi core. Simultaneous multi threading platforms, multi core machines and shared-memory parallel computer all provide system support for the execution of multiple independent instruction Streams or threads. With multi core technology ,one can able to make effective use of the parallelism that is present in our machine's hardware[6].The multi core implementation work come with challenges like parallel code development, code optimization and debugging. To develop new multi core implementation of signal processing algorithm is difficult task [7][8].

IV. OPENMP AND SHARED MEMORY PARALLEL COMPUTERS

The shared memory computers have symmetric multiprocessor system. A shared- memory parallel computer is the computer whose individual processors. Share memory in such a way that each of them can access any memory location with the same speed that is Uniform Memory Access time. Small shared memory machines are symmetric. But larger shared memory machines are not symmetric. Both small and larger shared memory machines having less difference between them. Such machine's have cache-coherent non- uniform memory access [6].

Today , the major hardware vendors all offer some form of shared-memory parallel computer, with sizes ranging from two hundreds and in a few cases, thousands of processors. Today NVidia provided a kit where we can use many number of cores efficiently. Today , the major h/w vendors all offers some form of shared-memory parallel computer, with sizes ranging from two hundreds and in a few cases, thousands of processors. Cache memory is not shared.

Shared-memory parallel computer also have some memory that in not shared. Processors have been consistently getting faster. Operands from memory. Unfortunately, the speed with which data can be read from and written to memory has not increased at the same rate. Vendors have built computer with hierarchical memory systems. In hierarchical memory systems .memory used in small expensive and very fast. This memory is called cache memory sometimes Cache in short. This cache memory supplies the processors with data and instructions at high rates. Each processor of an SMP needs its own private cache if it is to be fed quickly. Hence, we can say not all memory is shared. This demand is fulfill by OpenMP. Open MP provide tools for this. Open MP have some limitation rooted in its technical origins. Open MP is a new portable programming interface for shared memory parallel computers. Any sequential program can be made into a high performance parallel program it is not always true. OpenMP scalability provides nested parallelism. OpenMP enables the creation of shared memory parallel programs. OpenMP allows very strong correctness checking. Open MP provides sequential programs and multiple scheduling algorithms. Intel's cluster Open MP provides new directive to differentiate shares variables and distributed memory variables. Open MP application developer doesn't need any understanding about cache coherence. How it work? Not required to him or her. Indeed Open MP can be implemented on a computer that does not provide cache coherency since it has its own set of rules on how data shared among the threads running on different processors. The programmer who is doing or working with the Open MP ,must be aware of the Open MP memory model which provides for shared and private data and specifies when updated shared values are guaranteed to be available to all of the code in an OpenMP program[6].

V. GENERIC CACHE BASED DUAL CORE PROCESSOR

There are two levels of cache .Level term is used to denote how far away(in terms of access time) a cache is from the CPU, or core .The higher the level ,the longer it takes to access the cache at that level. At level 1 we distinguish a cache from data called Data cache. One for instructions Instruction cache and the Translation-Look aside Buffer". The last of these is an address cache. These three caches Data cache, Instruction cache and Address cache are all private to a core that means other core or cores.

VI. RESULT AND PERFORMANCE ANALYSIS OF FAST FOURIER TRANSFORM

There are various ways to implement the Fast Fourier Transform, by writing the program in open Multiprocessing programming, executing on Multi core Digital Signal Processor and other approaches used by different authors mentioned in the previous section. The results shows that the parallel approach has reduces the complexity which is introduced that why the programmer has to go for parallelism while writing their application code. We are having future plan to test the same and compare the results by using CUDA parallel programming where it can be tested by increasing the number of cores.

TABLE I
PERFORMANCE ANALYSIS BY FIRST SYSTEM

N	PTime2	MFLOPS	STime2	MFLOPS
2	0.041	4.86	0.027	72.11
4	0.056	14.09	0.062	127.84
8	0.084	28.52	0.089	266.87
16	0.12	52.96	0.016	383.34
32	0.17	89.03	0.0032	487.82
64	0.023	164.29	0.0079	484.41
128	0.032	279.35	0.017	508.81
256	0.048	419.10	0.042	486.20
512	0.007	578.88	0.0086	534.01
1024	0.018	567.42	0.02	506.63
2048	0.028	798.72	0.041	543.34
4096	0.06	808.68	0.094	522.37
8192	0.014	753.24	0.019	546.17
16384	0.029	784.88	0.043	527.00
32768	0.052	928.39	0.089	549.45
65536	0.12	864.97	0.2	522.95
131072	0.028	779.34	0.043	514.38
262144	0.072	653.91	0.097	483.39
524288	0.14	685.52	0.19	503.30
1048576	0.32	641.50	0.43	482.90
2097152	0.65	671.37	0.87	500.78
4194304	1.4	646.52	1.9	485.21
8388608	2.84	678.82	3.84	502.17
16777216	6.14	655.78	8.26	487.15
33554432	**12.3**	681.60	**16.7**	502.28

Table I shows the result obtained by Dual Core system and Table II shows the result obtained from Quad core system. It has been observed from the Table I, the time required using for small value of N as 2 using sequential approach is 0.027 and using parallel approach is 0.0041.But,for large value of N parallel approach is giving better result that is time required is 12.3 seconds which is less as compared to sequential approach which requires 16.7 seconds. It is clearly shown from the Fig. 1 and table I.

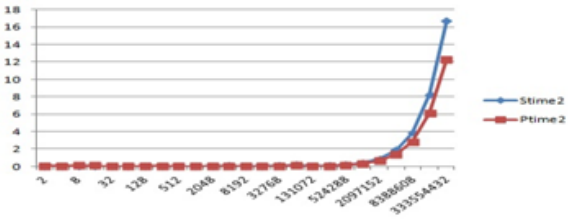

Fig. 1. Showing the performance on second system with four processor

TABLE II
PERFORMANCE OBTAINED BY SECOND SYSTEM

N	Ptime4	MFLOPS	Stime4	MFLOPS
2	0.031	6.32	0.005	39.26
4	0.047	16.85	0.0068	116.16
8	0.071	33.76	0.0095	250.56
16	0.1	61.06	0.023	267.62
32	0.016	99.27	0.0049	325.17
64	0.027	139.12	0.011	321.27
128	0.036	247.38	0.025	355.14
256	0.066	309.10	0.059	345.66
512	0.012	365.54	0.012	374.61
1024	0.026	386.98	0.028	357.37
2048	0.047	473.59	0.06	374.28
4096	0.1	457.99	0.13	363.69
8192	0.022	464.77	0.028	370.81
16384	0.047	480.09	0.063	363.93
32768	0.088	556.81	0.12	378.14
65536	0.2	523.93	0.28	367.41
131072	0.04	554.99	0.06	367.87
262144	0.086	542.91	0.13	354.90
524288	0.17	576.20	0.27	364.76
1048576	0.37	553.31	0.58	356.77
2097152	0.75	583.93	1.2	366.79
4194304	1.61	570.76	2.5	358.51
8388608	3.17	607.52	5.2	367.82
16777216	6.97	577.09	11.17	360.40
33554432	**14.04**	597.27	**22.77**	368.32

The Table I and Table II shows the values of N, PTime denotes the time required using parallel approach and STime denotes the execution time required using sequential approach.

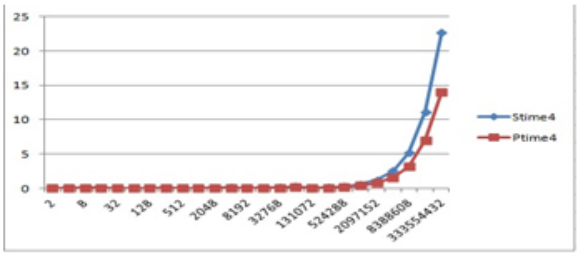

Fig. 2. Showing the performance on second system with four processor

These tables shows the results obtained by sequential and parallel approach and executed on two different systems with variable number of processors and speed.

It has been observed from the Table II for the system with 4 processors, the time required using for small value of N as 2 using sequential approach is 0.005 and using parallel approach is 0.031.But,for large value of N parallel approach is giving better result that is time required is 14.04 seconds which is less as compared to sequential approach requires 22.77 seconds. It is clearly shown from the Fig. 2 and Table II .

REFERENCES

[1] Tien – Hsiung weng , Sheng- Wei Huang ,Won Woo Ro ,Kuan – Ching Li, "*Implementing FFT using SPMD style of OpenMP* ,".

[2] Loannis Z. Emiris , Victor Y.Pan, "*Application of FFT*.

[3] Da Lu, Hanging Fan, "*Research on Parallel calculation of fast Fourier transform based on complex sequence*.

[4] Yan Shi and Chi Hou Chan, "*An OpenMp Paralyzed Multilevel Green's Function Interpolation Method Accelerated by Fast Fourier Technique*," IEEE transactions on antennas and propagation, Vol. 60, No. 7, July 2012.

[5] Hang Zhou ,Xi-Min Wang, Ying-Huo Guo, "*A Parallel Computing Component for Linux Cluster with Threads Binding Supports*," 2010 IEEE.

[6] Barbara Chapman, Gabriele Jost, Ruud van der Pas ," *Using OpenMP Portable Shared Memory Parallel Programming*, "The MIT Press,Cambridge, Massachusetts London, England, pp. 28-34.

[7] G. Blake, R.G. Dreslinski, T. Mudge ," *A survey of multicore processors*," S*ignal Processing Magazine*, vol.26, no. 6, pp. 26-37, Nov. 2009.

[8] L.J. Karam, I. AlKamal, A. Gatherer, G.A. Frantz ," *Trends in multicore DSP platforms* ," *Signal Processing Magazine*, vol. 26, no. 6, pp. 38-49, 2009.

[9] Aleksei Kharin, Sergey Vityazev, Vladimir Vityazev and Naim Dahnoun," *Parallel FFT Implementation on TMS320C66X multicore DSP*," Proceedings of Sixth European Embedded Design in Education and Reasearch,2014.

[10] Sun-Yuan Kung," *On supercomputing with Systolic/ Wavefront Array Processors*," Proceedings of IEEE Vol 72, No. 7, July 1964.

[11] Cooley, J. and J. Tukey," *An algorithm for the machine calculation of complex Fourier series, Mathematics of Computation*," 19:297-301,1965.