International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Optimistic Analysis of Processor Using FFT Equation Execution

Rajat V. Bodankar[1], Sandip S. Bramhankar[2], Sarika Wangulkar[3], Saroj Shambharkar[4]

[1,2,3] UG Students, Department of IT, KITS, Ramtek, India

[4]Project Guide, Department of IT, KITS, Ramtek, India

Abstract: ***This paper optimistic analysis of processor using Fast Fourier Transformation equation performed. Now a day's most desktop PCs have multiprocessing technology such as Hyper-Threading (HT), Dual-Core, and Quad-Core processors. Technological developments in microprocessor design have enabled hardware vendors to put multiple cores on a single socket. The main idea is to build software applications that can fully exploit the capabilities of a multi-core system using multithreading approach for faster performance. The aim to show the performance enhancement in terms of execution time using multiple threads on multi-processor.***

Keywords: FFT, multithreading, and multi-processor

1. Introduction

We report an in-depth study on such challenges based on our experience of optimizing the Fast Fourier Transform (FFT)on the IBM Cyclops-64 chip architecture a large-scale multi-core chip architecture consisting 160 thread units, associated memory banks and an interconnection network that connect them together in a shared memory organization. We demonstrate how multi-core architectures like theC64 could be used to achieve a high performance implementation of FFT both in 1D and 2D cases. We analyze the optimization challenges and opportunities including problem decomposition, load balancing, work distribution, and data-reuse, together with the exploiting of the C64 architecture features such as the multi-level of memory hierarchy and large register files[1].

This paper provides a basic understanding of Intel processor architectures and multithreading concepts. The aim of this paper is to illustrate the advantages of implementing multithreaded software applications on multi-core systems for achieving performance enhancement in terms of execution time. Both sequential and multithreaded software applications have been developed using an open source FFT library called FFTw and their comparison in terms of execution time, on different Intel processor systems has been presented

2. Fast Fourier Transform

The FFT is a fast algorithm for computing the Discrete Fourier Transform (DFT). In the literature, the FFT has been extensively studied and implemented as an important frequency analysis tool in many areas such as image processing, signal processing, and other domains. Consider the computation of the N=2tpoints DFT, x(n). The algorithm divides this N-point data sequence into two N/2-point data sequences f1(n) and f2(n), corresponding to the even-indexed and odd-indexed points of x(n), respectively. Consequently, the FFT gives an Θ(Nlog2N)algorithm for computing DFT. The Fourier transform is most commonly associated with its use in transforming time-domain data into frequency-domain data. However, it is important to understand that there is nothing inherent in the Fourier transform regarding either the time domain or the frequency domain. Rather, the Fourier transform is a general-purpose transform that is used to transform a set of complex data in one domain into a different set of complex data in another domain. It is purely happenstance that it happens to be so valuable in describing the relationship between the time domain and the frequency domain [4].

3. Intel Processor Architectures

A. Multiprocessor Technology

Multiprocessor systems offered by Intel comprise of multiple processor chips on a single board. Today, Intel Itanium and Dual Xeon servers are in use having two or more processors on one physical board and connected through a high speed communication interface [2]. The main disadvantage of multiprocessor systems is that they are expensive.

B. Hyper-Threading Technology

Intel Hyper-Threading (HT) technology allows a single CPU to present itself as a dual CPU to the operating system. An HT processor only has duplicate set of registers and pretends to be two processors. Cache and execution units are shared between these two virtual processors, with appropriate logic to switch between these processors so that only one processor uses each unit at a time. Intel HT processor consists of two APICs (Advanced Programmable Interrupt Controller) that provide multiprocessor interrupt management and distribution across the two processors [2]. Intel Pentium IV processors support HT technology.

C. Dual-Core Processors

Dual-core processors contain two identical processor cores on a single chip. Each processor core has its own resources except the on-die cache memory that can be provided to each processor core in three possible ways as designed by Intel; 1) separate on-die cache, 2) on-die cache to be shared between the two cores, 3) each processor core can have a portion of on-die cache exclusively for itself. Both processor cores must have a communication path to the system front-side bus [2].

D. Multi-Core Processors

Multi-core systems can have any number of CPUs on a single chip i.e. 2, 4, and 8. Their architecture is an extension of dual-core processors [2]. The main challenge is in the area of software development so that maximum performance can be taken out from multiple processor cores.

4. Multithreading

A. Multithreading Concept

In general, a computer program is a series of instructions that are executed by a CPU. Multithreaded programming takes this idea and replicates it. A multithreaded program has many sequences, each sequence within an independent thread. It is like having different small programs all running together in parallel. In case of a single processor system e.g. Intel HT processor, the operating system has a procedure called scheduler that provides the illusion that multiple threads are running in parallel. The Windows scheduler provides options to set thread priorities thereby allowing threads to get their time slice in an orderly fashion. For multiprocessor and multi-core systems, the Windows scheduler detects the presence of multiple CPUs and allows threads to run in parallel by assigning each thread to a CPU respectively. Hence for an operating system like Windows, a software application must have a multithreaded architecture in order to benefit from multiple CPUs. The main task is to divide the processing load among several threads in order to achieve maximum performance. In most applications, only the CPU intensive sections of code would be multithreaded [3].

B. Parallel Processing

Parallel processing means executing multiple threads on multiple processing units. The purpose is to decrease the program execution time and is one of the key benefits of multi-core systems. The idea is to identify how many serial applications can run on a multi-core system at once without interference. These applications can be additional copies of the application under test, or they can be entirely different processes. Hence for a multi-core system, as the number of processors increases, a multithreaded application always provides enhanced performance [5].

5. Related Work

Multithreading helps to create scalable applications that allow the programmer to add threads as and when needed. Managing individual threads is important as they share the same memory and data variables. A thread can be considered as a semi-process with a starting point, an execution sequence and a termination point. One thread shares memory and data with other threads running on a system [3]. On a single processor system, threads can be run either in a cooperative mode or preemptive mode. In cooperative mode, each thread can control the CPU as long as it needs. It has no concept of priorities and multithreading. Windows 3.x operating system used this kind of implementation. In preemptive mode, operating system itself distributes the processor time among threads and decides the running sequence of the threads. This approach exists in majority of popular operating systems today not only Windows. This mode allows fast switching among threads and it appears as if all threads are running in parallel. On a multiprocessor system, operating system can allocate the individual threads to the separate processor cores and hence improves the program execution. The efficiency of multithreading increases due to the distribution of threads on several processors or cores is faster than sharing processor time on a single processor. It is particularly useful for high speed processing applications like signal and image processing.

A. Detected the Processor

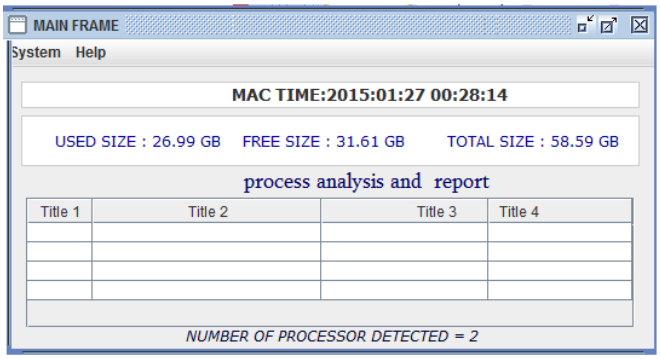

Figure 1: In the fig to detected number of processor in our PCs.

B. Load the process and FFT program

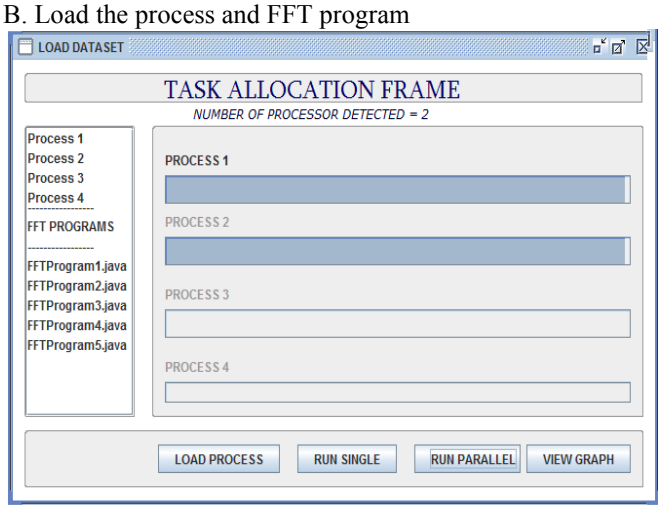

Figure 2: In this fig. load the process and FFT program

The above figure shown the run single and run parallel. When the run single than only one processor is start work and execute the process and FFT program. When the run parallel than dectected all processor work parallel execution process and FFT program.Example of FFT program shown below.

```
package fftsample;

public class FFT {

    // compute the FFT of x[], assuming its length is a power of 2
    public static Complex[] fft(Complex[] x) {
        int N = x.length;

        // base case
        if (N == 1) return new Complex[] { x[0] };

        // radix 2 Cooley-Tukey FFT
        if (N % 2 != 0) { throw new RuntimeException("N is not a power of 2");

        // fft of even terms
        Complex[] even = new Complex[N/2];
        for (int k = 0; k < N/2; k++) {
            even[k] = x[2*k];
        }
        Complex[] q = fft(even);

        // fft of odd terms
        Complex[] odd  = even;  // reuse the array
        for (int k = 0; k < N/2; k++) {
            odd[k] = x[2*k + 1];
        }
        Complex[] r = fft(odd);

        // combine
        Complex[] y = new Complex[N];
        for (int k = 0; k < N/2; k++) {
            double kth = -2 * k * Math.PI / N;
```

Figure 3: FFT example to use in execution time.

6. Simulation

In this paper we are show the processor simulation and execution by using complex FFT program and to generate graph of processor execution performance in based on the multithreading program.

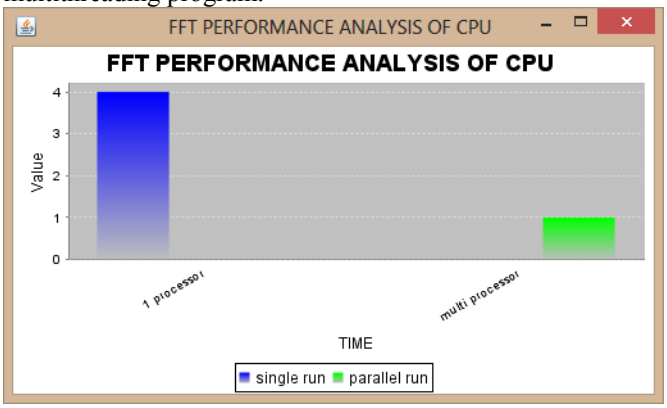

Figure 4: Both single and parallel execution than generate the graph.

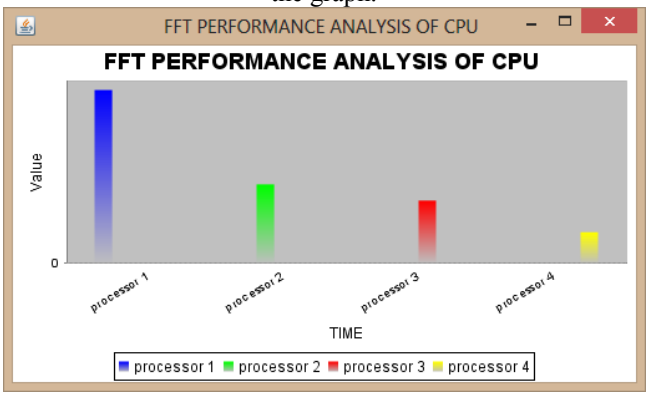

Figure 5: In this fig. parallel run and multiple processor execution

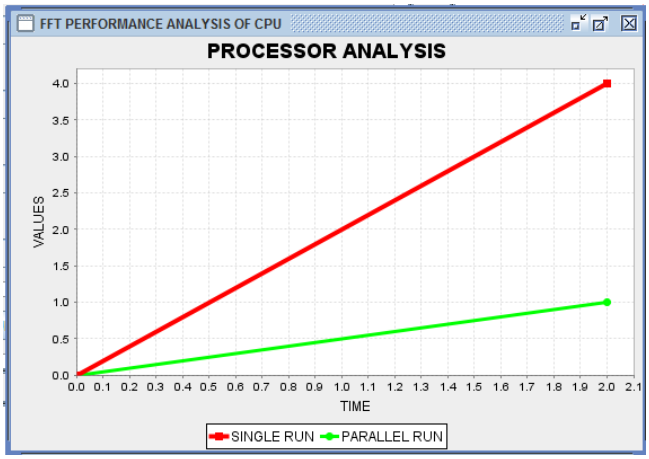

Figure 6: Analysis of the processor

The above shown figure analysis of the processor. When execution single run the process more time required to execute in single processor and when the parallel run process less time are required the execution and increase the performance of the CPU.

7. Conclusion

This paper concludes that the optimistic analysis of the processor using the multithreading FFT equation. The result of our analysis shows parallel execution has better than single execution and parallel execution takes less time to execution.

8. Acknowledgement

Our thanks to first and foremost I offer my sincerest gratitude to my college KITS RAMTEK, and my department of information technology which has provided the support and equipment. Which I have needed to complete my work. I extend my heartfelt gratitude to my guide, Mrs. Saroj Shambhakar. Who has supported me throughout our research with their patience and knowledge.

References

[1] Umar Hamid, Haroon Shahzad, and Muhammad Irfan," Parallel Implementation of 1-D Complex FFT Using Multithreading and Multi-Core Systems", International Journal of Computer and Communication Engineering, Vol. 2, No. 2, March 2013.
[2] T. Martinez and S. Parikh, "Understanding dual-processor," Hyper Threading Technology and Multi-Core Systems, Technical White Paper, 2005.
[3] Intel Hyper-Threading Technology, Technical User's Guide.
[4] Brenner, N.; Rader, C. (1976). "A New Principle for Fast Fourier Transformation". IEEE Acoustics, Speech & Signal Processing 24 (3): 264–266.
[5] Parallel Processing via MPI & OpenMP, M. Firuziaan, O. Nommensen. Linux Enterprise, 10/2002

Author Profile

Rajat Bodankar, B.E. in IT Final Year

Sandip Bramhankar, B.E.in IT Final Year

Sarika Wangulkar, B.E.in IT Final Year