International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 - 5161

Available at http://inpressco.com/category/ijcet

Research Article

Comparative Study of Multi-Threading Libraries to Fully Utilize Multi Processor/Multi Core Systems

Priya Mehta[Á*], Sarvesh Singh[Á], Deepika Roy[Ḃ] and M. Manju Sarma[Ḃ]

[Á]Computer Science and Engineering, Jayoti Vidyapeeth Womens University, Jaipur, India
[Á]National Remote Sensing Centre, ISRO, India

Accepted 10 July 2014, Available online 01 Aug 2014, **Vol.4, No.4 (Aug 2014)**

Abstract

The development of multi-core technology has led to a great shift from sequential programming to parallel programming. This has made substantial challenges to software industries and government organizations to take full advantage of the performance intensification offered by the multi core systems. This paper aspires to compare and analyze the parallel computing ability offered by OpenMP(Open Multiprocessing), Intel Cilk Plus and MPI(Message passing Interface). Some proposals are also provided in parallel programming. The parallel programming features provided by these libraries are also studied and compared. The study is done by parallelizing problems related to Remote Sensing data processing which is large in volume and whose sequential processing is very much time consuming. This makes it pertinent to speed up the processing times by introducing parallelism in processing and for efficient utilization of multi processor multi core systems. The paper aims at exploring these libraries and studying the speed up achieved by using parallel processing and Data Parallelism paradigm.

Keywords: *Multi-Core , Multi-CPU, OpenMP, Intel Cilkplus, MPI, Data parallelism, Multi-threading*

1. Introduction

In this multi-core era both for programmer productivity and performance there is lot of importance for serial program parallelization (Minjang Kim, June 2013). In this paper we have tried to bring about a comparative study of the speedup achieved by converting time consuming sequential processes into a multithreaded ones by using OpenMP, Intel Cilkplus and MPI. The problem domain comprises of problems related to processing of Remote Sensing data of Indian Remote Sensing Satelites..For parallelization in a shared memory system it is considered that OpenMP and Intel Cilkplus are more suitable and Hybrid parallelization by using the MPI is more suitable on distributed memory system.

2. Problem description

Our problem domain contains problem related to pre-processing of Indian Remote Sensing Satellite data. This pre-processing involves initial processing of the RS data before data product generation. The data volume of Remote sensing satellites runs into GBs and sequential processing of this data is very time consuming. Also the multi-core multi-cpu system are not efficiently utilized by the sequential processing .Thus it becomes pertinent to employ parallel processing techniques for faster processing and better turnaround time. This also leads to better utilization of system resources in terms of processing power and memory. We implemented multithreading using OpenMP, Cilkplus and MPI on the following three problems:

1) Histogram Equalization
2) Pattern search
3) Bayer Demosaicing

All these problems are data intensive and hence speedup was achieved up by applying data parallelism.

3. Levels of parallelism

Manufacturers implement parallelism at several different levels

1. Task level parallelism
2. Data level parallelism
3. Instruction level parallelism
4. Bit level parallelism.

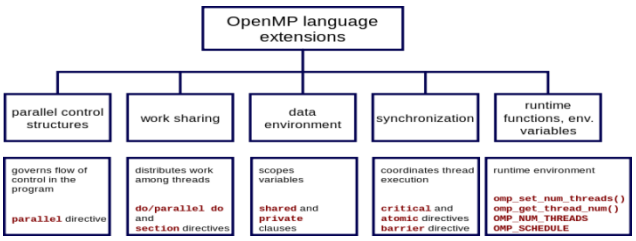
This paper focuses on **Data level parallelism**. Data parallelism is achieved when a huge data set is split into smaller chunks and each processor performs the same task on these chunks of data. In some situations, a single execution thread controls operations on all pieces of data. In others, same code but different threads control the operation (Sreepathi Pai).

*Corresponding author: **Priya Mehta; Sarvesh Singh** is working as Asst. Prof; **Deepika Roy** and **M. Manju Sarma** are working as Scientist/Engineer-'SD' Gr.Head,(SG) respectively.

4. Multithreading Libraries used

4.1 OpenMP

OpenMP is a portable API for parallel programming on shared-memory computer systems.The OpenMP API is defined and updated by the OpenMP Architecture Review Board (ARB), which consists of major hardware and software vendors.OpenMP is targeted at shared-memory multiprocessors such as the multicore CPUs found in modern computer systems. code is easier to understand and maybe more easily maintained



1) Hardware Platforms that supports OpenMP are

Vendor/source	Compiler	Information operating system use
GNU INTEL	GCC C/C++ /Fortran	Free and open Source – Linux,solaris,AIX,MacOSX,WINDOWS,OpenMP 3.1 is supported GCC 4.7 Window,linux and MacOSX Compile with -Qopenmp or just -fopenmp

Hardware platform required - INTEL, AMD, IBM, Cray, Red hat, NEC, Oracle Corporation.

4.2 Cilkplus

Cilkplus is an extension to the C/C++ programming language designed for multithreading parallel computing. It is simple for programming development with well structured analysis and verification of programs offered by cilkplus (Sreepathi Pai). It has powerful tools like cilkview and cilkscreen which add scalability and code analyzer in cilkplus. It is also very convenient to detect race conditions with these tools (Yuxiong He, 2010).

1) Hardware Platform that support Cilkplus are

Vendor/source	Compiler	Information operating system used
gcc	G++	Free and open Source – Linux,MacOSX,WINDOWS
INTEL	C/C++/ Fortran ICC ICPC	Window,linux and MacOSX

4.3 MPI

MPI is a message passing library typically used for parallel and distributed computing. MPI addresses primarily the message-passing parallel programming model, in which data is moved from the address space of one process to that of another process through cooperative operations on each process. In this paper we have used OpenMPI which is an open source, freely available implementation of MPI

Most MPI implementations consist of a specific set of routines (i.e., an API)directly callable from C, C++, Fortran and any language able to interface with such libraries, including c#, Java or Python.
The advantages of MPI over older message passing libraries are portability because MPI has been implemented for almost every distributed memory architecture and speed as each implementation is in principle optimized for the hardware on which it runs. MPI is more suitable for a distributed memory communication environment.

5. Features of multithreading libraries

In this paper we implemented multithreading using three types of libraries in C/C++ language in Linux environment. Data Parallelism was implemented in each of the problem. Our problem consists of applying the same set of operation on each chunk of data and thus it yielded very well to the Data Parallelism paradigm.

5.1 OpenMP

1) #pragma omp parallel

The pragma omp parallel is used to fork additional threads to carry out the work enclosed in the construct in parallel (Shen Hua1, 2013). The original thread will be denoted as master thread with thread ID 0. *#pragma omp parallel* spawns a group of threads, while *#pragma omp for* divides loop iterations between the spawned threads. Both things can be at once with the fused *#pragma omp parallel* for directive . The number of threads created by these compiler directives is equal to the number of processing nodes. This number can be modified by using certain environment variables or function calls given by the library.

2) Reduction

Reduction (op : list) can be used inside a parallel or a work-sharing construct: A local copy of each list variable is made and initialized depending on the operator "op" (e.g. 0 for "+"). Compiler finds standard reduction expressions containing "op" and uses them to update the local copy. Local copies are reduced into a single value and combined with the original global value. The variables in "list" must be shared in the enclosing parallel region.

5.2 Intel CilkPlus

cilk_for – It specifies that the iterations of a loop can be executed in parallel.
cilk_spawn – It specifies that a function call can execute asynchronously, without requiring the caller to wait for it to return. This is an expression of an opportunity for

parallelism, not a command that mandates parallelism. The Intel Cilk Plus runtime will choose whether to run the function in parallel with its caller.
cilk_sync – This acts like a barrier, waiting for all spawned children to complete before processing. There is an implied cilk_sync at the end of every function that contains a cilk_spawn.

1) Reduction

Converting a sequential program to a parallel program and changing the for loop to a cilk_for causes the loop to run in parallel, but creates a data race on the shared variable which is updated by each thread. To resolve the race condition, we make this shared variable a reducer which is a variable that can be safely used by multiple threads running in parallel. The library internally takes care of avoiding race conditions on the shared variable.

5.3 MPI(Message passing interface)

We have used MPI_Send and MPI_Recv functions for message passing between processors as there is no shared memory between processes like OpenMP and Intel CilkPlus. MPI_Comm_rank and MPI_Comm_size are first used to determine the world size along with the rank of the process. The master sends blocks of data to the workers using MPI_Send function. A message is sent to a specific process and is marked by a tag (integer value) specified by the user. Tags are used to distinguish between different message types a process might send/receive. It is crucial to note that MPI_Send is a blocking call and will block till a corresponding MPI_Recv is not found. So it is important that the communication is optimized so as to prevent any deadlocks. The receiving process calls MPI_Recv and blocks until the data transfer is complete. Once the data is distributed between the processes, they operate on it and then send the result back to the master process for consolidation (Chao-Chin Wu, 2012,vol. 60, no.1).

6. Hardware platform and system specification

We implemented our problem on a 4 CPU, 8 core machine. So in total we had 32 cores available for parallel processing. The system had 32 GB RAM. The operating system was Red Hat Enterprises Linux version 5.6.

7. Result

We implemented parallelism in the below mentioned problem types using three different parallelism libraries. We have also done a comparative study of the speedup achieved. The following are the problem description and the speedup gains obtained.

7.1 Histogram Equalization

Histogram equalization is a technique for adjusting image intensities to enhance contrast by using the image histogram. An image histogram is a graphical representation of the intensity distribution of an image. It quantifies the number of pixels for each intensity value considered. Histogram equalization is a method that improves the contrast in image by stretching out the intensity range. The following figures show the low contrast image of Resourcesat-2 LISS-4 sensor with its concentrated histogram.

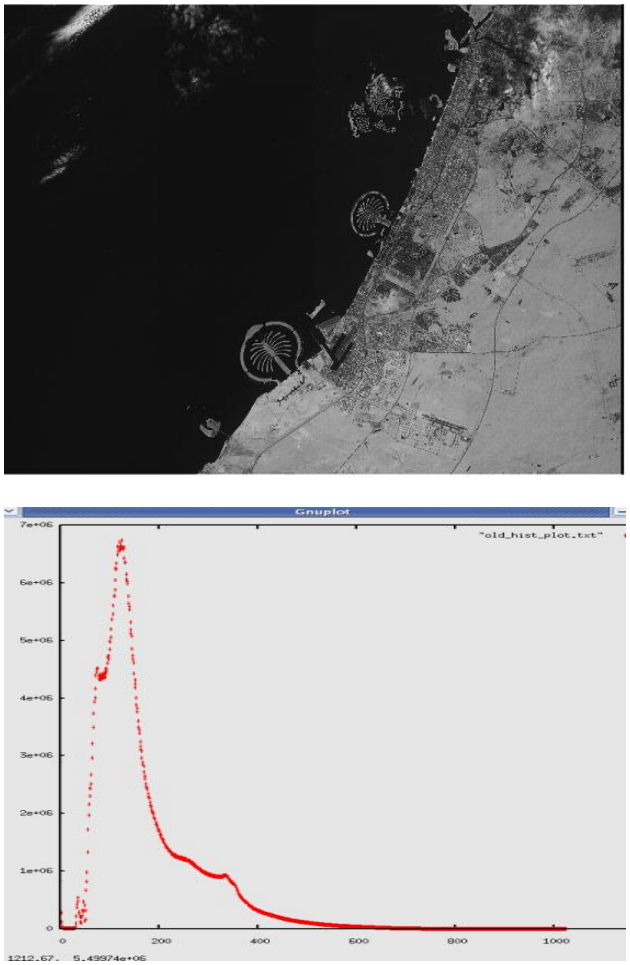


Fig. 1 Original image and its histogram

The next figure shows the image after histogram equalization. Its stretched out histogram after contrast stretching is also shown.

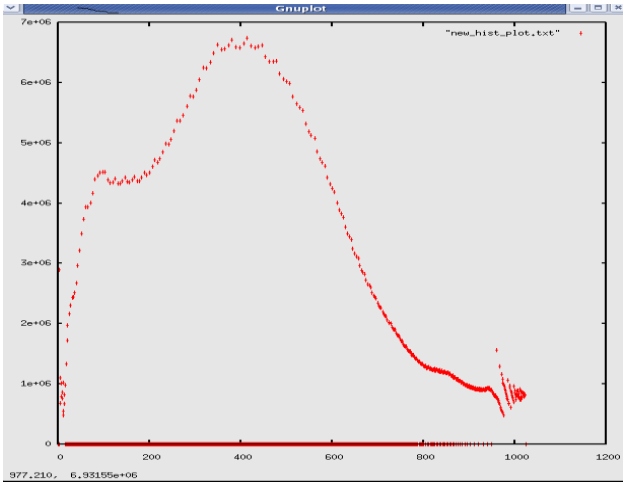


Fig. 2 Image and its histogram after Histogram equalization

The analysis was done on Resourcesat-2 LISS-4 sensor image. As the data was very large, the processing time of the sequential program was large as well. The use of multithreaded libraries improved the speedup. Multithreading was implemented using C/C++ using OpenMP and CilkPlus.

We also faced some race conditions while summing up variables shared by threads and synchronization problems between threads. Race condition was overcome by using reducers. Synchronization issues were solved by restructuring the code and introducing synchronization constructs between threads. The time taken to run the sequential and parallel codes is as shown in the following table

Table 3 Result of Time taken and speedup achieved using OpenMP and CilkPlus for Histogram Equalisation

Sequential Code	**Parallel Code using OpenMP**		**Parallel Code Using CilkPlus**	
	Time Taken	**Speedup Achieved**	**Time Taken**	**Speedup Achieved**
36.201 sec	26.594 sec	1.36	16.251 sec	2.22

7.2 Patten Search

The aim of this problem was to search for a specific 4 byte pattern in a file containing binary data. The pattern can occur repeatedly in the data and the number of occurrences of this pattern had to be counted and the positions where it is found had to be marked. This problem was very much time consuming as the input file was of 2GB size. However the search logic had to be implemented repeatedly for every line. Hence data parallelism approach was followed here also.

Race conditions were overcome by applying reducers. The code was optimized and further improvement was done by making each thread to read simultaneously from the file using pread function call.

In MPI, problem of deadlock faced due to blocking send/receive was also overcome by restructuring the code.

Sequential Code	Parallel Code using OpenMP		Parallel Code Using CilkPlus		Parallel code using MPI	
	Time Taken	Speedup Achieved	Time Taken	Speedup Achievd	Time Taken	Speedup Achieved
0m3.673 sec	0.987sec	3.721	0.677sec	5.425	4.247 sec	8.52

7.3 Bayer Demosaicing

Bayer demosaicing algorithm is a digital image process used to reconstruct a full color image from the incomplete color samples output from an image sensor overlaid with a color filter array. A **Bayer filter** mosaic is a color filter array (CFA) for arranging RGB color filters on a square grid of photosensors. Its particular arrangement of color filters is used in most single-chip digital image sensors used in digital cameras, camcorders, and scanners to create a color image. The filter pattern is 50% green, 25% red and 25% blue, hence is also called **GRGB** or other permutation such as **RGGB**. By using this filter, each pixel will be assigned only a single color (i.e. either R or G or B).

In order to generate a useable image from this raw data i.e. to assign all three colors(RGB) to a pixel, some reconstruction is required . The process used to recover this information is called demosaicing. This process interpolates the missing color information from adjacent pixels in order to generate a full color image. The algorithm used here for demosaicing is Bilinear Interpolation. The following image shows the RAW image obtained from a sensor using Bayer filter.

Fig. 3 Raw Bayer Image

The following image shows the demosaiced image after applying Bilinear Interpolation of pixel color values on the input RAW image.

Fig. 4 Demosaiced Image using Bilinear Interpolation

Table 1 Comparing the features of openmp, cilkplus, and mpi

	Openmp	**Cilkplus**	**Mpi**
Compatibilty (comipler ,os)	• COMPILER- GCC/ICC IN C G++/ICC IN C++ • OS- LINUX,SOLARISIS, MACOSX,WINDOWS	• COMPILER ICPC ,ICC,GCC IA-32 and Intel® 64 architecture programs (32-bit and 64-bit) that run on Windows, Linux, and OS X	MPICC IN C MPICXX IN C++
Open source or not	YES	YES	YES
Hardware platform ,processor	PROCESSOR- INTEL,AMD,CRAY,REDHAT	WINDOW ,LINUX ,OS X	WINDOW,LINUX OSX
Type of parallism	TASK AND DATA PARALLISM	TASK AND DATA PARALLISM	TASK PARALLISM
Language library or compiler directive	COMPILER DIRECTIVE – Eg-#pragma omp parallel LIBRARY ROUTINE **#include <omp.h>** **int omp_get_num_threads(void)** ENVIROMENT VARIABLE EG-**export** **OMP_NUM_THREADS=8**	#pragma simd	-----
header file	#include<omp.h>	#include<cilk.h>	#include<mpi.h>

Table 2 Comparison between programming Technologies

Multithreading libraries	Openmp	Cilkplus	MPI
1.What is it ?	API	Language (Actully c++ Extension)	API
2.Language supported	C,C++,Fortran	C	C ,C++,Fortran,Ada, python
3.Memory	Shared Memory	Shared Memory miser memory manager	Distributed Memory
4.granulity	Fine	Course /Fine	Fine
5.syncronization	Implicit	Implicit/explicit	explicit
6.Difficulty for programmer	Easy	Medium	Medium

Since the bilinear interpolation has to be applied on each pixel and for all the three colors, this takes time to generate a full color image. To speed up the generation of final full color image, multithreading and data parallelism concept was applied here also.

Sequential Code	Parallel Code Using OpenMP		Parallel Code Using CilkPlus	
	Time Taken	Speedup Achieved	Time Taken	Speedup Achieved
.670 sec	.344sec	1.94	.298 sec	2.24

Conclusion

This paper explains comparative study of multithreading libraries which is implemented on 4 CPU and 8 core system with Linux platform in C/C++ language. Since the problem deal with large volumes of data, hence the processing time was also very long. To achieve faster processing and optimize the use of multi cores system, we implemented Data Parallelism using three different libraries: OpenMP, Intel CilkPlus and MPI. We analyzed the speedup achieved using these libraries. OpenMp is only suitable to be used for multithreading in case of shared memory system. It enables multithreaded programming with quick and small processing with very small overhead of coding. Spawning threads involves including compiler directives provided by OpenMP. The number of threads to be spawned will be decided by the library depending on system specifications. Inter-thread communication is also easy as threads have access to a common shared memory. According to our experience OpenMP is best geared for loop level parallelization. We did not need to install any new compiler/library for running OpenMP.

CilkPlus gave the best results in all the three problems that we analyzed. But for implementing parallelism using CilkPlus we need Cilk Plus enabled compilers. This results in overhead in terms of installing either latest version of Intel Compilers or Cilk Support for GCC. CilkPlus also provides efficient tools for analyzing the code for scalability and race conditions which assist the programmer to track down bugs faster. It is suitable for both task and data parallelism.

MPI model is a message passing model best suited for programs which do not have complex communication process. In our problems it took more time as compared to other two libraries as it spent more time in passing the data between processes. MPI is suited for large system with long term processing. Each of the parallel programming frameworks has their own strengths and

weaknesses. The framework that offers best performance depends on both on the kind of application and how much the code has been tuned for a particular platform.

Future scope

There are many ongoing development on openMP from openMP(2011a) and there are many scope for development on openMP.According to openMP(2011b) it is consider to support for accelerators such as GPUs,also if we increase the file size more and more we can get the more speedup. we can explore more and more multithreading libraries and explore their features such as TBB,boost thread, Posix thread .In cilk in future we can explore on cilkview and cilscreen tools.From open MPI(2011) it is quite clear that there is a lot of on-going development on MPI.

Acknowledgement

We would like to express our deep thanks towards the Deputy Director, SDAPSA and Director, National Remote Sensing Center for giving us constant encouragement and support.

References

Chao-Chin Wu, L.-F. L.-T.-H. (2012),Using Hybrid MPI and Openmp programing to optimize communication in parallel loop self sheduling sceme for multicore pc clusters. *The Journal of Supercomputing* , pp 31-61.

Minjang Kim, P. K. (June 2013), Predicting Potential Speedup of Serial Code via Lightweight Profiling and Emulations with Memory Performance Model. *ComSIS Vol. 10, No. 3*.

Sabahat Saleem1, M. I. (2014),Multi-Core Program Optimization: Parallel Sorting Algorithms in Intel Cilk Plus. *International Journal of Hybrid Information Technology Vol.7, No.2 (2014)*, pp 151-164.

Shen Hua1, 2. Z. (2013), Comparison and Analysis of Parallel Computing Performance Using OpenMP and MPI. *The Open Automation and Control Systems Journal*, pp 5, 38-44.

Sreepathi Pai, R. G. (n.d.), Limits of Data-Level Parallelism.

Yuxiong He, C. E. (2010),The Cilkview Scalability Analyzer. *SPAA'10*.