Parallel Implementation of 2D Gabor Filter with Cluster OpenMP

Zhang Ji[1],Zhao Xiao-jing[2]
1.School of Information and Electronic Engineering, Beijing institute of technology,Beijing,China
2.School of Information and Electronic Engineering, Beijing institute of technology,Beijing,,China
1. zhangji@bit.edu.cn, 2. zxjing1980@yahoo.com.cn

Abstract: This electronic document, a parallel implementation method of 2D Gabor filter based on Cluster OpenMP is presented. Cluster OpenMP and the image processing of Gabor filter are researched, and a new parallel method is designed. Because multi-core processor is popular around industry market, OpenMP in cluster entertainment may reduce the cost of parallel programming, and reach high performance meanwhile. With experiments, methods of sequence implementation, parallel implementation based on OpenMP and parallel implementation based on Cluster OpenMP, are compared. The result demonstrates that performance of Gabor filter can be accelerated highly with Cluster OpenMP.

Keywords: Cluster OpenMP; Gabor; parallel programming; cluster

1 Introduction

OpenMP, a thread-level parallelism, has been widely used in multi-core/multi-processor machines, which shares memory address. OpenMP provides programmers an easy way to work with parallel program[1]. But, it's limited in SMP machines. For parallelism crossing multi-machines or in a cluster, we have to add some other mechanism like MPI, and then we face more complicated design and implementation[2].

In 2006, Intel released a new parallel mechanism, Cluster OpenMP(CLOMP). Besides of the features to parallel in SMP computers, CLOMP can work in a cluster directly. The key advantage is that programmers need to change little of their old OpenMP sources. CLOMP inherits most of directives in OpenMP.

Gabor filter is an important method for time-frequency analysis. It has been proved to be useful in biomedical field, signal detection, image compression, and image enhancement. However, the computational complexity must be concerned especially when processing big sized images such as remote sensing images[3].

Theories for parallel implementation of Gabor algorithm[3], cannot work for SMP cluster, and hard to implement in most cases. The method presented here, is quite easy, and can work in clustered SMP environment.

2 Cluster OpenMP

Cluster OpenMP (CLOMP), is a run-time library that supports running an OpenMP program on a cluster, and it's released with Intel compilers in 2006 firstly. With it, the serial program can be ported to parallel, and then it can run in multi-nodes in a computer cluster. In addition, because of supporting all features of OpenMP, you can use the capability of multi-core resources in each node.

Software Distributed Shared-Memory (SDSM) Model, which is used in CLOMP, is a relaxed consistency model. It means that CLOMP will only synchronize when barrier, flush or dependence is reached. All shared variables are grouped together on certain pages in memory, and the basic mechanism relies on protecting memory pages with an *mprotect* system call[4].

When programs are running on multi-nodes of a cluster, the node starting the executive is called Home node, and others are called remote nodes. When reaching code regions for parallel, multi-threads will be generated. In each node, one is called main thread, and other threads are called working thread (Chart 1).

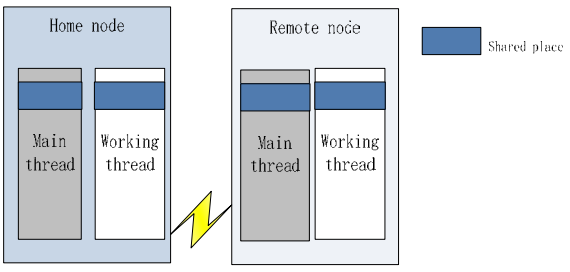

Chart 1 Topology of CLOMP Architecture

Liking OpenMP, CLOMP view barrier as a point to synchronize, and the difference is that there are two levels

of barrier in CLOMP. One is barrier among threads in one node, and the other is barrier crossing nodes. Threads will be blocked and waiting in barriers firstly in one node and then among nodes. As to program, developer doesn't need to know the mechanism at all.

When porting programs from OpenMP to CLOMP, the mere concern is about sharing data in heap, which will not be shared automatically. Talking to C language, programmers should use *kmp_sharable_malloc* other than *malloc* to allocate memory in heap, and add the pointer of the memory block to sharable directive.

3 2D Gabor Filter

Gabor filter is a useful tech for image processing. The filter has both frequency-selective and orientation selective properties as well as optimal joint resolution in both spatial and frequency domains[5].

The steps to process a 2D image with Gabor filter includes:

1) Calculate direction for each point.
2) Calculate frequency for each point.
3) Estimate masks.
4) Filter each point with Gabor.
5) Mask filtered data.

The most time consuming part is step 4. The Gabor filter with even symmetric can be defined as follows:

$$G(x, y, \theta, f_0) = \exp\left\{-\frac{1}{2}\left(\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}\right)\right\}\cos(2\pi f_0 x_\theta) \quad (1)$$

Where $\begin{bmatrix} x_\theta \\ y_\theta \end{bmatrix} = \begin{bmatrix} \sin\theta & \cos\theta \\ -\cos\theta & \sin\theta \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$, and θ is direction of a point, and f_0 is the frequency which will be calculated in Step 2nd above, and σ_x and σ_y are constant values for Gaussian envelop.

The filter for 2D images can be defined as:

$$E(i,j) = \begin{cases} 255 & M(i,j) = 0 \\ \sum_{u=-w/2}^{w/2}\sum_{v=-w/2}^{w/2} G(u,v,O(i,j),F(i,j)) \bullet N(i-u, j-v) & M(i,j) \neq 0 \end{cases} \quad (2)$$

Where $E(i, j)$ is image data after filtering, $N(i, j)$ is image data before filtering, w is template size, and $M(i, j)$ is mask value determining whether the block is a valid region.

4 Parallel Implementation of Gabor Filter

According to the above Formula (2), Gabor filter for each point requires four kinds of data: image data, direction, frequency, and mask. And all required data can be prepared before step 4th. Because there is no dependence on Gabor filter result of any point, the algorithm can be parallelized based on unit of image region.

Assuming P parallel units are working, the work for p unit can be defined as the follows:

$$E_p(i,j) = \begin{cases} E(i,j) & i \bmod(P) == p-1 \\ no-action & others \end{cases} \quad (3)$$

Where $E_p(i, j)$ is image data after being filtered by p unit, $E(i, j)$ is to call Formula 2 to compute Gabor.

5 Experiment Results

The experiment is carried out on two personal computers (PC) with Intel Core 2 dual-core processor inside. The two computers are configured with the same hardware (1GB Memory), and connected with 100MB Ethernet. Red Hat Enterprise Linux 4 (bit 64) is installed for each. A cluster with the two computers is constructed. One computer works as both Head node and Compute node, and the other one works as Compute node.

5 pictures with different size are tested: 504*480, 1008*480, 1008*960, 1008*1920, 2016*1920, and 4032*1920. All pictures are gray image of 256 colors, stored in BMP format.

We implement and run the program in 5 different parallel levels:

1) Serial. The program is run with one thread in one node.
2) OpenMP parallel. With OpenMP library, the program is running with two threads in one node. The performance is shown in OpenMP 2T in Chart 2.
3) CLOMP parallel with two threads in one node. With CLOMP library, the program is run with two threads in one node. The performance is shown in CLOMP 1N2T in Chart 2.
4) CLOMP parallel with one thread for each of two nodes. With CLOMP library, the program is run with two threads (One thread for each node). The performance is shown in CLOMP 2N1T in Chart 2.
5) CLOMP parallel with two threads for each of two nodes. With CLOMP library, the program is run with

978-1-935068-01-3 © 2009 SciRes.

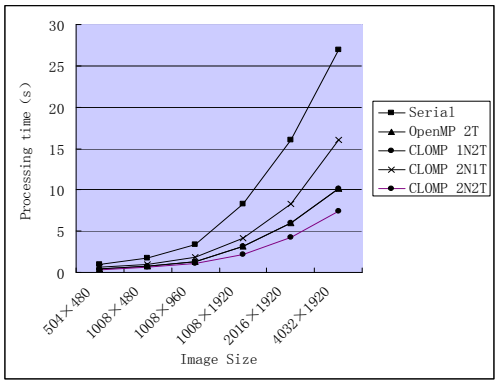

Chart 2 Time Consuming for Gabor Filter among 5 parallel levels

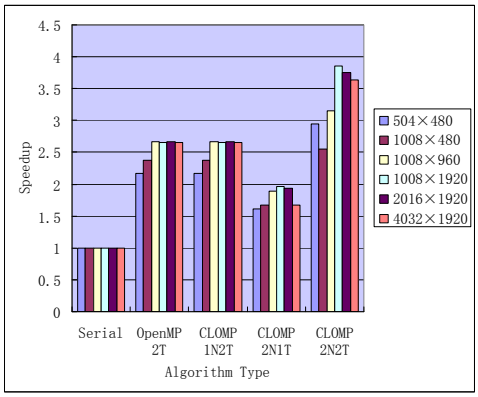

Chart 3 Speedup of Gabor Filter among 5 parallel levels

four threads (two threads for each node). The performance is shown in CLOMP 2N2T in Chart 2.

From the experiment result, we can conclude:

1) CLOMP improves the efficiency for Gabor Filter. Compared to the serial program, speedup can arrive at the most 3.8 times in 2 nodes.
2) If only one node is used, performance of OpenMP is better than CLOMP's, but the difference is little.
3) Consuming for synchronization should be concerned. The efficiency of CLOMP 2N1T is lower than the efficiency of CLOMP 1N2T.

6 Conclusion

This electronic document, we present parallelized Gabor algorithm with Cluster OpenMP, and experiments prove that Gabor filter algorithm parallelized in cluster of SMP machine get a high performance. Also, experiments show that Cluster OpenMP is efficient for processing images. Meanwhile, because of the consuming of communication and synchronization, images with small size can only get a low speedup.

Acknowledgements

Many people have made invaluable contributions, both directly and indirectly to my research. I would like to express my warmest gratitude to Zhang Ji, my supervisor, for his instructive suggestions and valuable comments on the writing of this thesis. Without his invaluable help and generous encouragement, the present thesis would not have been accomplished. I greatly appreciate my parents and husband's support and endless love. My heart swells with gratitude to all the people who helped me.

References

[1] Cai Jia-jia, Li Ming-shi,Deng Feng. OpenMP- Based Parallel Computation on Multi- Core PC [J]. Computer technology and development. 2007,17(10):87-91(Ch).

[2] Jie Cai, Alistair P. Rendell, Peter E. Strazdins, H'sien Jin Wong. Performance Models for Cluster-Enabled OpenMP Implementations[A]. Computer Systems Architecture Conference. ACSAC 2008. 13th Asia-Pacific 4-6 Aug. 2008 Page(s):1 - 8 .

[3] Tao Liang, Zhuang Zhen-quan. Parallel Lattice Structures of Block Time-recursive Algorithms for Real-valued Discrete Gabor Transforms [J]. Journal of Chizhou Teachers Colleg ,2006,20(3):25-26(Ch).

[4] Jay P.Hoeflinger. Extending OpenMP to Clusters[EB/OL].http://cache-www.intel.com/cd/00/00/28/58/285865_285865.pdf.Intel White Paper. 2006

[5] Xia Zhen-hua, Shi Yu, Yu Sheng-lin. Fingerprint Image Enhancement Based on Gabor Filters [J]. Journal of engineering graphics ,2006,5:80-85(Ch).