Presented at the Asian Technology Information Program (ATIP) 3rd China HPC Workshop
SC07 Conference, Reno, Nevada, November 11, 2007

Study of Content-Based Image Retrieval Using Parallel Computing Technique

Yongquan Lu, Pengdong Gao, Rui Lv, Zhiwu Su

High Performance Computing Center
Communication University of China

Wenhua Yu

Department of Electrical Engineering
Pennsylvania State University

yqlu@cuc.edu.cn

ATIP 3rd China HPC Workshop, November 11, 2007, Reno, NV

Study of Content-Based Image Retrieval Using Parallel Computing Technique

Yongquan Lu, Pengdong Gao, Rui Lv, Zhiwu Su
High Performance Computing Center
Communication University of China, Beijing, China 100024
yqlu@cuc.edu.cn

Wenhua Yu
Department of Electrical Engineering
Pennsylvania State University, University Park, USA 16802
wxy6@psu.edu

ABSTRACT
In this paper, we implement a parallel technique to a content-based image retrieval system and investigate the influence of different hardware devices on the retrieval system performance. Our system is developed on cluster architecture. The feature extraction and similarity comparison of visual features, which are widely used for the content-based image retrieval, are realized by using the parallel computing technique. Therein, the feature vector, which identifies one image uniquely, is composed of several different color features. Numerical experiments have demonstrated that the parallel computing techniques can be applied to significantly improve the performance of retrieval system.

Categories and Subject Descriptors
H.3 [**Information Storage and Retrieval**]: H.3.3 Information Search and Retrieval -- Query Formulation; H.3.4 Systems and Software -- Performance evaluation.

General Terms
Management, Performance, Experimentation.

Keywords
content-based image retrieval, parallel computing, cluster architecture, color feature.

1. INTRODUCTION
The content-based image retrieval (CBIR) is the application of computer vision to the image retrieval problem, namely, the problem of searching for images in a large database. With the rapid development and wide applications of digital technology, it is the fact that the multimedia databases of images and video information are becoming larger and larger. Besides the improvement of the image retrieval technique, the content-based image retrieval from a large database requires more computing resource and efficiency in the real world[1, 2]. The way to carry on a valid organization and management to such a large multimedia database and query the information effectively has become the most urgent problem. Since the image is one of the most popular forms in multimedia, therefore, in recent years the CBIR technology has drawn a general attention, and a lot of results have been published[3, 4].

The conventional CBIR approaches are evaluated based on whether the retrieval result is correct or not, however, in practical applications for millions even billions of image files, the feature extraction and similarity comparison will be both very time consuming process. Therefore, the response time is also a critical factor for the evaluation of an image retrieval system. Hence, it is more meaningful to improve the searching efficiency of a retrieval system[2]. One of the best choices to tackle this issue is to use the parallel computing technique that allows us to retrieval a large database at the same time by using a high performance computing cluster[5, 6, 7].

Some parallel or distributed computing methods have been introduced into the CBIR before[5, 6, 7]. However, most of these systems were designed only for some special applications. And they did not do more research on finding the factors that would affect the searching efficiency of the retrieval system. In this paper, we propose a content-and-cluster based parallel image retrieval system. Meanwhile, the influence of hardware devices on the efficiency of the retrieval system, such as the I/O equipment, the processor ability and the message exchange system, is also fully investigated. Numerical experiments have shown that the parallel computing technology is able to significantly improve the efficiency of CBIR system. And the result of this paper will be meaningful to further improve the efficiency of the content-and-cluster based parallel image retrieval system.

2. PARALLEL IMPLEMENTATION OF CONTENT-BASED IMAGE RETRIEVAL

2.1 Method Description
Due to the simplicity of the color histogram and its insensitivity to the image size and rotation variety, in this paper, the color is used as the main feature of the image retrieval system. There exist three algorithms for the feature extraction: namely, averaging algorithm of choosing the RGB pieces, partially accumulating histogram and improved reference color table[8]. The Gauss normalization of each feature will be carried out before calculating the similarity. And a power exponent is taken to compute the similarity of feature vectors, in which the Manhattan distance is adopted for the line distance calculation. Finally, we use the parallel sorting algorithm to sort the result of similarity computing.

2.2 Parallel Implementation of Image Feature Extraction
The purpose of image feature extraction is to build a feature vector that can be used to distinguish an image, and then we can

Sponsored by CNGI Project(CNGI-04-12-2A)

use it to determine the similarity between two different images when we search for an image. A typical process is described as follows: reading the parameters for the image feature extraction from the configuration file, identifying all image files in the specified directory, reading the image, and then generating a feature vector to make a vision record of the image itself, finally saving all the vectors to database. During this procedure, reading the image and generating the feature vector will consume most of system computing time. Therefore, two tasks can be assigned to each image: one is used for reading the file, and another one is used for generating the feature vector. Its data correlogram is illustrated in Fig.1[9].

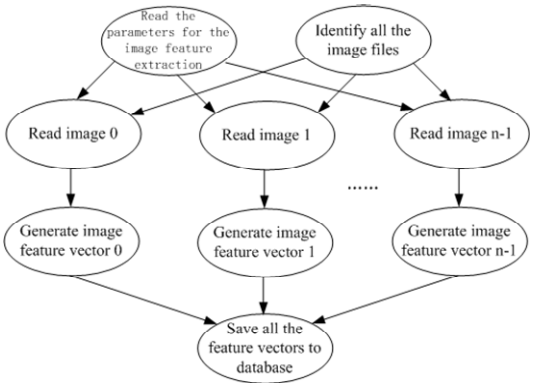


Fig.1 Correlogram of feature extraction process

Neither the task amount in Fig.1 can be determined during compiling, and nor is communication among the tasks required. Since each image has different size, the time consuming on each task may be dramatically different. According to the decision process described by the decision tree[10], the imaging of task should take place during the runtime; and the task scheduling algorithm will carry out the task imaging during the runtime.

In this paper, we adopt the concentrated task scheduling algorithm to develop a parallel code based on the manger-worker pattern[11]. In the manger process, first of all, the *n* images will be recognized in the specified directory, and the length of a feature vector *k* is received from the worker 0 process, then assigns a *n k* size of matrix **S** to store these feature vectors for each image that comes from each worker process. After finishing initialization, the manager process will get into the loop until all the worker processes have been finished. In this loop, the manger process will receive messages from each worker process. If this message includes a feature vector of image, the manger process will place it into the corresponding position in the **S** matrix; otherwise, it indicates that the worker process is ready to accept a new image file, which will only happen one time for each worker process. If any image has not been treated yet, the manager process would assign the name of image to the worker process. After all images have been already handled, the manager process would assign a terminate message to all the worker processes, and then exit the loop. After that, the manager process stores all image feature vectors in the **S** matrix into a linear image feature database in order.

In the worker process, the worker 0 process will open the configuration file first, and read the configuration parameters, and then broadcast the information to other worker processes and also send the length of feature vector *k* to the manager process. Once the preparation completed, the worker processes will inform the manager process, and then be involved in the loop. In this loop, it would receive a message from the manager process. A terminate message indicates that all the images have been handled, and the worker process will stop. A name message will inform the worker process to read the corresponding image, generate a feature vector, and send this vector back to the manager process, and then start next loop.

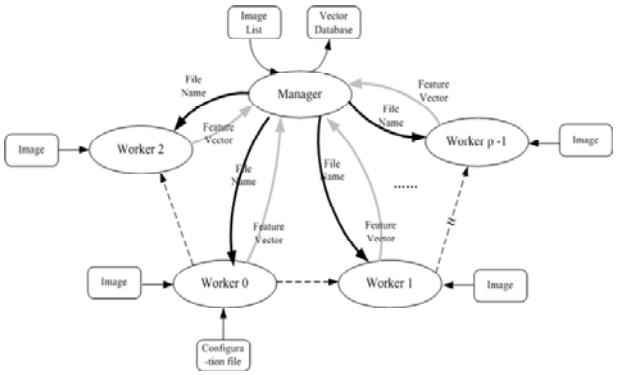


Fig.2 Task-Channel diagram of the parallel algorithm for the image feature extraction

We can use a Task-Channel diagram[10] to describe this parallel algorithm for the image feature extraction based on the manager-worker pattern, as shown in Fig.2. The dotted line with an arrowhead in Fig.2 indicates a channel that broadcasts the parameter configuration information, and the black with an arrowhead shows a channel that sends out an image name to the worker, the dark line with an arrowhead indicates a channel that sends out an image feature vector to the manager process.

2.3 Parallel Implementation of Image Feature Similarity Comparison

The comparison of image similarity is to compare the relativity between the feature vector of query image and each one in the image feature vector database so that we can obtain some images that are most similar to the query image. The typical process of the image feature similarity comparison can be described as follows: first, obtaining the image retrieval configuration parameters from a configuration file, and loading the image feature database from the hard disk, then recognizing the query image in the specified input file. For the query image, we first open the image file and read its data, and generate the feature vector, and then compute the similarity between the feature vector of query image and each one in feature vector database and sort the results according to value of the similarity in order from large to small. Finally, based on the sorted results, a corresponding match list of images is generated as the image retrieval result, which will be written into the result file. The two processes of reading the image retrieval configuration parameters and identifying the query image can be carried on at the same time. During the entire process, because of the massive feature vector database, the similarity computing will take up overwhelming majority of calculation time, and hence the comparison process should create a task for each feature vector in the database, which is used to compute its similarity. After completing the similarity calculation, we need to sort this huge sequence, and it is also a

time-consuming work. The data correlogram is shown in Fig.3[9]. We can use the function decomposition to divide the problem, in which each operation is mapped as an original task, and each original task is related to an image feature vector.

Fig.3 Correlogram of similarity comparison process

The amount of tasks in Fig.3 is related to the number of vectors in the image feature database, is usually relatively stable, and has a structure correspondence mode between masks as well. Since all the feature vectors of images have the same size, so the time cost on computing the similarity is the same. According to the decision process described by the decision tree[10], the tactic of mapping task is as follow: minimizing the communication by getting the tasks together and creating a task for each processor.

Using a so-called idea "divide and rule", we develop a parallel code based on the Master-Slaver design pattern[12]. In the master process, the query image in the specified input file is first identified, and then the image retrieval configuration parameters will received from the slaver process 0. After the initialization, the master process opens the query image file, reads the image data, and broadcasts the image length and data to all the slaver processes. The master process wait until all the salver processes finish the task, and then gather the results from all the slaver processes and place them in the corresponding positions. Afterwards, the master process checks the global result and makes sure all the items in the correct order. According to the image retrieval parameters, the master process takes out the first *N* items or all the items whose value of image similarity is greater than a criteria value *T*, and then writes the corresponding images into the result file.

In the slaver process, first of all, slaver process 0 will open an configuration file for the image retrieval, read the parameters, and broadcast them to all the processes including the master process. We can open the image feature vector database from the slave process 0, compute the range of the subset in the image feature vector database which is in the charge of each slaver process respectively, then read its vectors and then send them to the corresponding slaver process. After the initialization, the slaver process will receive the image length and data from the master process, generate the image feature vector and then compute the similarity between the vector of the query image and each one in the database. After the computation, we use the Parallel Sorting by Regular Sampling (PSRS) approach proposed by Li et al. to sort the results based on the value of the similarity [13,14]. According to the image configuration parameters, we take the first *N* images from the current local results or all the results whose value is larger than a specified criteria value *T* as the final results, and then send them back to the master process.

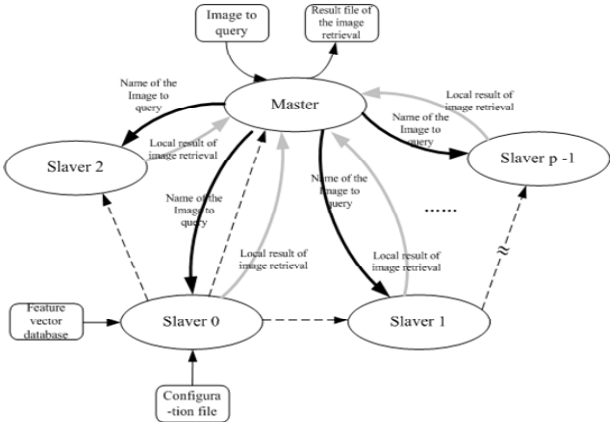


Fig.4 Task-Channel diagram of the parallel algorithm for the similarity comparison

We can use a Task-Channel[10] diagram to describe this parallel algorithm for the image similarity comparison based on the Manager-Worker pattern, as shown in Fig.4, The dotted line with an arrowhead in Fig.4 indicates a channel that broadcasts the parameter configuration information, the black line with an arrowhead shows a channel that sends out the query image to the slaver process, and the dark line with an arrowhead indicates a channel that sends out the partial result of image retrieval to the master process. It should be mentioned that for the sake of simplicity and clarity the message passing between the slave processes through the PSRS sorting approach is not shown in Fig.4.

3. DESCRIPTION OF NUMERICAL EXPERIMENTAL RESULTS

To validate the influence of different hardware devices on the performance of the retrieval system, in this section, we have implemented a content-and-cluster based parallel system and designed four kinds of cases to test above-mentioned image feature extraction and similarity comparison algorithm.

Fig.5 is the architecture of our cluster. It is composed of computing subsystem, management subsystem, storage subsystem and the interconnection subsystem. Therein, the computing subsystem includes 48 compute nodes, and each node is equipped with two Intel Xeon 3.8GHz processors and 4GB memory. The management subsystem only includes the frontend node. The storage subsystem includes four I/O nodes and the EMC CX300 array, which are connected by the fiber. The interconnection subsystem includes a special Myrinet 2000 switch and two Foundry Edge Iron 48 GS Gigabit Ethernet switches. All nodes

are connected with the Ethernet except for the computing nodes connected with the Myrinet.

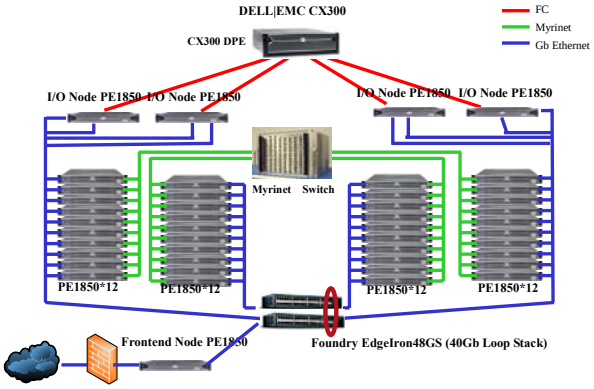
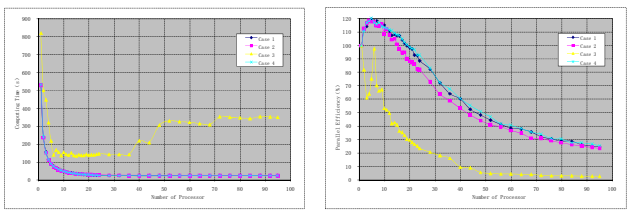


Fig. 5 Architecture of the high performance computing cluster system

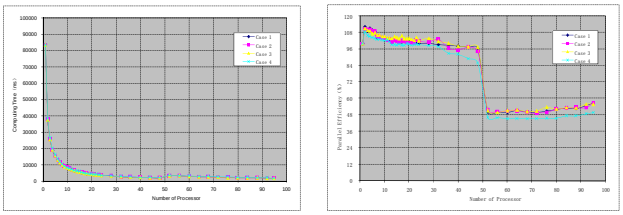
In the first test case, Myrinet is used as a message exchange network, the data files are stored on the EMC array, and based on the bonding technology provided by Linux we aggregate the two Gigabit-Ports of each I/O node in balance-alb mode. Additionally the two ports connect with the Gigabit Ethernet at the same time, and the compute nodes can access the data file by PVFS2. In the second case, only a port can connect with the Gigabit Ethernet, and the other configurations are the same as that in the first one. In contrast with the first case, the third case adopts the same message exchange network, but the storage system and the I/O configuration are different from those in the first case. Its data file is placed in the native disk of the frontend node, and there is only one Gigabit-Port in the frontend node linked with the Gigabit Ethernet. The compute nodes can access the data file by NFS. The last case adopts the same storage system and the I/O configuration as the first case, but the message exchange network has a slightly difference, which uses the Gigabit Ethernet as the MPI message exchange network. For the image feature extraction algorithm, our test data is composed of 12,844 BMP files, which is more than 9.3 GB all together, including all kinds of images such as people, scene, flowers, and so on. They were selected randomly from the database of CCTV. For the image similarity comparison algorithm, the test data is composed of 1,656,876 image feature vectors, which are more than 2.6GB. In our cluster, the parallel environment is MPICH 1.2.7p1; the compiler is Intel Compiler 9.0 for Linux with the o3 option and the development language is ANSI C.



(a) **(b)**

Fig. 6 (a) Computing time of feature extraction; (b) Parallel efficiency of feature extraction

The computing time and the parallel efficiency of image feature extraction in the four cases are plotted in Fig. 6. Figure 6 (a) shows that the cases 1, 2 and 4 have a similar computing time and gradually decrease with the number of processors increasing. When the number of processors exceeds 32, these three curves do not decrease anymore and keep a constant value. The computing time for the case 3 varies dramatically and increases when the number of processors is more than 6. It is evident that the computing time of case 3 is more than the other three cases because the I/O performance in the cases 1, 2 and 4 is better than that in case 3 and it will keep a constant when the I/O performance reaches a limit [15]. In contrast, the I/O in the case 3 will be a bottleneck for the data access among the processors. The parallel efficiency in the cases 1, 2 and 4 is over 100 percent for a small number of processors, as show in Fig.6 (b). It is because the data volume of the system I/O goes down due to the PVFS2 parallel file system, which can combine the multiple requests into to a single operation to reduce the number of operations on the hard disk[16].



(a) **(b)**

Fig. 7 (a) Computing time of similarity comparion; (b) Parallel efficiency of similarity comparison

The computing time and parallel efficiency of the image similarity comparison algorithm in four kinds of cases are plotted in Fig.7. The computing time is the same in the cases 1, 2 and 4 and gradually decreases when the number of processors increases. From Fig.7 (a), it's easy to see that the retrieval system using parallel technique is more effective than the non-parallel system (the number of processor is one). There is only less time needed to respond to the query of the retrieval system. The parallel efficiency in the four cases is basically the same except for that the efficiency of the case 4 is slightly less than the other three cases. It is observed that the parallel efficiency in Fig.7 (b) drops almost 50 points with the number of 48 processors. When the number of processes is less than 48, there is only one process on each node. Therefore, it can use all resource of the node. When the number of processes exceeds 48, there are two processes on some nodes, two processors on each node will participate the computing. Although the operating system could assign both processes to different processors on each node, they would still share the memory through the same data bus. In the similarity computing phase, considering the performance we adopt the DB-In-Memory policy. It would cause the resource competition under this situation, which would depress the efficiency of those nodes. In addition, since the PSRS algorithm is involved in all processes, we have to synchronize all the processes before we can sort the results. So the ultimate efficiency of the image similarity comparison will drop evidently.

4. CONCLUSIONS

In this paper we have implemented a content-and-cluster based parallel image retrieval system. In our system the image feature extraction and similarity comparison were realized by using the

parallel technique. Numerical experiments have demonstrated that the significant improvement on efficiency was obtained by using the parallel technique. However, there was no obvious modification to retrieval precision because this paper focused on the efficiency of the retrieval system and we didn't do any improvement to the retrieval methods. On the other hand, the results of four different test cases showed that the I/O performance and the processor ability of a cluster had more influence on the system performance than the message exchange network system. In the future more sophisticated image features and their extraction methods as well as the optimization of system configuration will be our main work.

5. REFERENCES

[1] Huang XL, Shen LB. Research on Content-Based Image Retrieval Techniques. ACTA ELECTRONIC SINICA. 2002, 30(7), 1065~1071 (in Chinese).

[2] Li XY, Zhuang YT, Pan YH. The Technique and System of Content-Based Image Retrieval. Journal of Computer Research & Development. 2001, 38(3), 344~354 (in Chinese).

[3] Sagarmay D, Zhang YCH. An Overview of Content-Based Image Retrieval Techniques. Proceedings of the 18th International Conference on Advanced Information Networking and Application. Fukuoka, 2004, 59~64.

[4] Carson C, Belongie S, Greenspan H, Malik J. Blobworld: Image Segmentation Using Expectation-Maximization and Its Application to Image Querying. IEEE Transactions on Pattern Recognition and Machine Intelligence. 2002, 24(8), 1026~1038.

[5] Odej Kao. Parallel and Distributed Methods for Image Retrieval with Dynamic Feature Extraction on Cluster Architectures. Proceedings of the 12th International Workshop on Database and Expert Systems Applications. Munich, 2001, 110~114.

[6] Kao O, Steinert G, Drews F. Scheduling Aspects for Image Retrieval in Cluster-Based Image Databases. Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid. Brisbane, 2001, 329~336

[7] Punpiti Pn, Sanan S. A Parallel Model for Multimedia Database on Cluster System Environment. Proceedings of the 7th IEEE International Symposium on Industrial Electronics, Pretoria. 1998, 648~652

[8] Huang Xianglin, Song Lei, Shen Lansun. A Method of Shape Encoding and Retrieval. Journal of Electronics，2002, 19(3), 302~306 (in Chinese)

[9] Banerjee, Utpal. Dependence Analysis for Supercomputing. The Kluwer International Series in Engineering and Computer Science. Boston: Kluwer Academic, 1988

[10] Quinn, Michael J. Parallel Programming in C with MPI and OpenMP. Dubuque, Iowa: McGraw-Hill, 2003

[11] Freisleben, B. and T. Kielmann. Coordination Patterns for Parallel Computing. Lecture Notes in Computer Science, 1282 (1997), 414

[12] Buschmann, Frank. Pattern-Oriented Software Architecture: A System of Patterns. Chichester: Wiley, 1996, 243~260

[13] Li X, Lu P, Schaeffer J, et al. Parallel Sorting by Regular Sampling. Tech. Rep. Department of Computing Science, University of Alberta, 1991

[14] Xiaobo Li, Paul Lu, Jonathan Schaeffer, et al. On the Versatility of Parallel Sorting by Regular Sampling. Parallel Computing, October 1993, 19(10), 1079~1103

[15] W.B. Ligon, III, R.B. Ross. Implementation and performance of a parallel file system for high performance distributed applications. The 5th IEEE International Symposium on High Performance Distributed Computing, 1996, 471

[16] R Thakur, W Gropp, E Lusk. Data sieving and collective I/O in ROMIO. The Seventh Symposium on the Frontiers of Massively Parallel Computation, 1999, 182~189