Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 5 (6): 1056-1061, 2005
ISSN 1812-5654

An Efficient Approach of Load Balancing for Parallel Visualization of Blood Head Vessel Angiography on Cluster of WS and PC

[1]Najib A. Kofahi and [2]Kalim Uddin Qureshi
[1]Department of Computer Sciences,
Faculty of Information Technology and Computer Sciences, Yarmouk University, Irbid, Jordan
[2]Department of Math and Computer Science, Kuwait University Khalidya Campus,
P.O. Box 5969, Safat 13060 Kuwait

Abstract: This study introduces a method to obtain an efficient load balancing for distributed medical image processing application. The proposed strategy has a dynamic task partition and adaptive load balancing features that coop the machine's performance variation and heterogeneity of the distributed system. Since in medical imaging applications massive amounts of data movement occurs, the effect of data compression is also investigated in this study. The measured results show that the proposed strategy improves the performance of distributed medical imaging system on heterogeneous clusters of Personal Computers (PC) and Workstations (WS) and remedy the defects of Runtime Task Scheduling (RTS) strategy. The proposed strategy is called Effective Load Balancing (ELB) for parallel visualization of blood head vessel angiography on cluster of PC and WS. The Parallel Virtual Machine (PVM) environment was used as a tool for parallelization.

Key words: Data compression, heterogeneous cluster, koad balancing, medical imaging, PVM, Runtime Task Scheduling

INTRODUCTION

A medical imaging system collects a massive amount of data in a short period of time and requires highly intensive computational requirements for both image reconstruction and image analysis. Parallel processors in medical imaging are useful for real-time or near-real-time processing. Real-time imaging provides the physician or technician the ability to see the results of an imaging scan while the patient is still present.

The recent development of powerful inexpensive desktop personal computers and high speed networks have paved the way for the emergence of a wide range of multimedia applications with diverse requirements especially those involving the transfer and manipulation of large amounts of data. In the past high performance applications such as visualization and scientific simulations used specialized expensive workstations and supercomputers. However, the decreasing costs and increasing performance of both computer hardware and networks now offer great potential for distributed network computing. Clusters of Workstations (WS) and clusters Personal Computers (PC) can be connected to a high speed Local Area Network (LAN) and be used to solve different types of scientific problems[1]. With distributed network computing, clusters of workstations collaborate with each other to solve some common task. One of the major benefits of distributed computing is its scalability in terms of the amount of computing power and resources available for large-scale applications. In this study, we explore the performance of a distributed medical visualization application using clusters of WS and low-cost PC.

A distributed medical visualization application: The visualization application were discussed in this study uses 8 bit/pixel gray scale volume rendered human head blood vessel angiography. The head can be rotated and viewed at different angles after the full application is being completely processed. The image set consists of 20 image frames each of size 512x64x512 files. The generated image is shown in Fig. 1.

Professionals using these visualization applications demand high resolution images to improve the quality of the images, which in turn determines the quality of the diagnosis. In addition, the timely delivery of the images is also essential to the efficiency of the professional, as long delays will impact the time spent with a patient. These

Corresponding Author: Najib A. Kofahi, Department of Computer Sciences,
Faculty of Information Technology and Computer Sciences Yarmouk University, Irbid, Jordan
Tel: +962 2 7211111/2677 Fax: +962 2 7211128 E-mail: nkofahi@yu.edu.jo

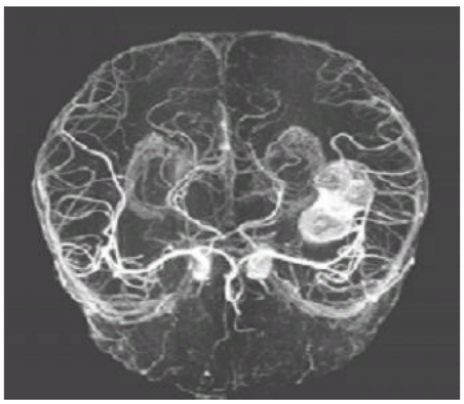

Fig. 1: Complete generated image from the application running on one of client machines

characteristics place considerable demands such as high bandwidth and low-latency support for such professional applications[2]. In order to achieve guaranteed end-to-end user level performance for multimedia applications, it is important to have Quality of Service (QoS) support at different levels including the application, operating system and the network. An example of such a QoS architecture is the QoS broker which is described by Sharma *et al*.[1].

Data file compression and decompression: Network bandwidth is a kind of limited and precious resources in modern distributed computing environments. Insufficient bandwidth will severely degrade the performance of a distributed computing task in exchanging massive data among the networked hosts. A feasible solution to save bandwidth is to incorporate data compression during transmission. It also assesses the resultant impact on the overall performance of the system in terms of latency, response time and available bandwidth.

Data compression techniques can be divided into two main categories: lossless and lossy. Yang and Kieffer[4], develop a new practical lossless data compression algorithms. So far, the most widely used universal lossless compression algorithms are arithmetic coding algorithms, Lempel-Ziv algorithms and their variants[4].

Arithmetic coding algorithms and their variants are statistical model-based algorithms. The compression ratios of the statistical lossless coding schemes, such as Huffman coding and arithmetic coding, are affected by the degree of skewness in the message sequences. Most data however only exhibit small skewness, thus most statistical coding schemes do not perform well.

A statistical model is either static (i.e., assumed to exist in advance) or can be built dynamically during the encoding process[4]. Several approaches have been proposed in the literature to build dynamically a statistical model. These include the prediction by partial match algorithm, dynamic Markov modeling, context gathering algorithm and context-tree weighting method[4]. Typically, in all these methods, the next symbol in the data sequence is predicted by a proper context and coded by the corresponding estimated conditional probability[4].

The other important subcategory of lossless algorithms is the dictionary model-based algorithms. The major part of the computation steps in dictionary-based coding schemes is string matching, whereas the major parts in statistical coding schemes are probabilistic. Hence, the dictionary-based coding schemes run faster and achieve better compression ratio than statistical coding ones in general.

The compression ratio in lossy is generally higher than lossless ones since some information are lost during compression. lossy compression is often used in the compression of graphics (e.g. JPEG) speeches and videos (e.g. MPEG) lossy compression schemes are inappropriate for distributed computation tasks[5].

Compression techniques generally tradeoff compression ratio for decompression time due to the algorithmic complexity of the former. Total delay is defined to include the transfer time and the compression time of a compressed file or data[6]. Data should be compressed such that it minimizes the total delay. The transfer and compression times depend on the current underlying resource performance of the system and hence, no one compression format minimizes the total delay for all resource performance characteristics.

To tackle this situation, a Dynamic Compression Format Selection (DCFS) is proposed by Krintz and Calder[6], which performs a dynamic and automatic selection of competitive compression formats based on predicted values of future resource performance.

For this application we used zlib (http://www.zlib.org/) compression library for compression in our SUN SPARC workstations. The zlib library used huffmann coding and its compression ratio is 45% at compression level 9.

Load balancing: In Distributed Computing System (DCS), static and dynamic (Runtime Task Scheduling (RTS)) task distribution strategies are used to balance the loads among the WS and the PC of the cluster. Since WS/PC has a performance variation characteristic[7], therefore, the static task distribution is not effective for DC systems[8]. RTS strategy can achieve nearly perfect load balancing[8], because of the machines' performance variations and non-homogeneous nature of the application (image) are adjusted at runtime[9]. The RTS strategy performance

depends upon the size of the sub-task. If the sub-task size is too small then it generates a serious inter-process communication overhead. On the other hand, if the sub-task size is too large then it may create a longer master (client) machine waiting time due to the inappropriate sub-tasks size of slow performance machine[9]. Many researchers suggested enhancement to the dynamic task scheduling strategies for homogeneous DC systems[10]. However, these strategies do not work well for heterogeneous DC systems without further modifications. The crucial point is that they are based on fixed parameters that are tuned for the specified hardware. In heterogeneous systems, this tuning is often not possible because both the computational power and the network bandwidth are not known in advance, which may change unpredictably during runtime.

The strategies based on task distribution and then task migration from heavy loaded to light-loaded workers are explained[11]. Task migration generally has two serious drawbacks:

i) All workers should continuously monitor the status of other workers.
ii) During the computations, a worker has to search its load and float the information on the network, hence, produces a large amount of communication overhead.

In contrast to existing strategies, we proposed an ELB strategy with several important characteristics, i.e. it has adaptive nature; it reduces interprocess communication time overhead; and effectively balances the load among the workers.

TASK PARTITIONING AND SCHEDULING

Efficient execution of a parallel program on a distributed system depends on the effective partitioning of the program into modules (or sub-partitions) and on scheduling of these modules. The goal of the program partitioning is to minimize the overhead caused by inter-task communication time, while preserving the highest possible degree of parallelism. Here we describe the investigated strategies.

The static tasks distribution strategies' chief advantage is their simplicity; there is no need to collect current system state information. Their disadvantage is that they cannot respond to changes in system states, so the performance improvement they provide is limited[12].

Runtime Task Scheduling (RTS) strategy: In our experiment, a unit of task (one file size of 13.59 MB) is distributed at runtime. As the worker completes the previous assigned sub-task, a new task is assigned to the worker by runtime task scheduler for processing. The RTS strategy was used as a reference to compare the performance of the proposed strategy.

Effective Load Balancing (ELB) strategy for medical imaging system: In ELB strategy, the master first measures the performance of each node/worker by sending one image file to each worker. The time taken by the worker to completely process the task will be used to measure the worker's performance (Wpi). The Distributed Computing System Performance (DCSp) will be estimated by using Eq. 1.

$$DCSp = \sum_{i=0}^{Wt} Wpi \quad (1)$$

To compute the optimum task size of each worker, each worker normalized performance (Wnpi) need to be calculated using Eq. 2.

$$Wnpi = \frac{Wpi}{DCSp} \quad (2)$$

From our previous experience about the task size computation, we know that the size of the task for each worker is proportional to (Wnpi) and inversely proportional to the number of machines/workers in DS system. Therefore, each worker's task size (Ti) is estimated by using the following formula:

$$Ti = \frac{Tb^* Wnpi}{Wt} \quad (3)$$

where, Wt is the total number of workers in the distributed system configuration and Tb is the total number of unprocessed task.

Each time a worker reports a result to the manager, the manager compute the worker's performance (Wpi), worker's normalize performance (Wnpi) and worker's task size (Ti) for new task assignment. In the proposed strategy, 70% of the total task is scheduled according to the above describe procedure and for the remaining, the manager reduces the sub-task size (Ti) of each worker by 30% on each worker's request. This reduces the work-load-imbalance[9].

To achieve a better load balancing and avoid master waiting time at the end of the application, the following components are included in this strategy:

- **Slow worker omission:** The distributed Computing System (DCS) is composed of co-operative computing resources. Its performance also depends upon the other users application load. In some cases, the WS's performance degrades or the WS becomes

unavailable. This creates a long waiting time for the master/client machine at the end of the parallel application, which degrades the DC system performance. To avoid such type of waiting, the following checkpoint is included in the proposed strategy:

After 70% of the total task completion, the manager checks those workers, which still have not completed the first assigned task. The manager marks that worker as not available in the current DC configuration and adds its assigned task (Ti) to the unprocessed task balance (Tb). It eliminates excessive manager waiting time that may occur at the end of application.

- **Sub-task duplication:** When all the tasks of the application are distributed to the workers, no unprocessed task is left to assign further to only remaining idle worker. The faster worker finishes the last assigned task earlier as compared to the slow workers; the slow worker very often creates a manager waiting time. To avoid such type of waiting, the following slow worker's task duplication mechanism in this strategy is used:

  The manager searches the least performance worker in the current DC configuration and duplicates its assigned task to the next idle worker. It then creates a competition between two workers, the result of the worker who completes the task first is considered and that of the other is ignored. Each least worker's task is duplicated only once. Therefore, all computing resources are utilized at the end of the application. This also eliminates the excessive manager waiting time that would have occurred at the end of application.

- **Data compression:** For this medical imaging application, the total number of tasks/image files of 512x64x512 (= 16777216) is huge and each task requires a memory of 13.59 MB. To reduce these interprocess usages, data is compressed while sending tasks/results to worker/manager and worker/manager must decompress it before processing.

IMPLEMENTATION

The performance of both RTS and ELB strategies are evaluated in terms of speedup and workers idle time cost[13]. Here is a brief definition of these terms.

Speedup (SP): The speedup is used to quantify the performance gain from a parallel computation (Tpf) of an application over its computation (T_{DCS}) on a single fastest machine in the network of WSs/PCs. It is measured by the following formula:

$$\text{Speedup of DC system (SP)} = \frac{T_{pf}}{T_{DCS}} \quad (4)$$

Workers idle time cost: The activities of worker's process in a DC image processing system are mainly composed of three factors:

i) worker setup time to load pattern data file and initialize all programming parameters,
ii) worker raytracing computation time,
iii) worker time taken to report result (data) and getting new task from the master.

The value of the third factor is directly affected by the number of requests made by the workers. This overhead in time Qi is computed as:

$$Qi = (\text{Worker's new task starting time}) - (\text{worker's previous task completion time}) \quad (5)$$

Qi includes all communications and data access delays necessary to start the processing of new task on an idle worker.

The workers idle time cost can then be expressed as:

$$\text{Workers idle time cost} = \sum_{i=1}^{W_t} O_{(i)} \quad (6)$$

The distributed system used in our investigation is composed of seven Sun workstations (WSs) running SunOS/Solaris and thirteen PCs (Intel based machines) running Linux/FreeBSD operating system; all of these machines are connected via Ethernet. The network communication is handled by Parallel Virtual Machine (PVM) (http://www.csm.ornl.gov/pvm/pvm_home.html). The investigation is carried out on medical images, which can be parallelized without complex inter-processors communication.

RESULTS AND DISCUSSION

The proposed ELB and RTS strategies are compared in terms of speedup, number of request made by the master and workers idle time cost. The investigated distributed system configurations have high heterogeneity in nodes' performance. Some machines have a speed of 2 GHz and other have 100 MHz. The date compression for one (file) task is shown in Table 1.

The processing time taken to process the image by the single fastest machine in the network is 320 sec.

Table 1: Data compression

Uncompressed Magetic Resonance Imaging (MRI) data size in MByte	Compressed Magmetic Resonance Imaging (MRI) data size in MByte
13.59	9.20

Table 2: Comparison of RTS and ELB strategies in terms of speedup and number of requests made by the manager for task distribution in five DC system configurations

Distributed system configuration	Measured speedup using RTS strategy	Measured speedup using ELB strategy	No. of requests made by master in RTS strategy	No. of requests made by master in ELB strategy
4	2.7	3.2	1677216	403164
8	4.0	5.2	1677216	505125
12	4.6	5.9	1677216	703450
16	4.9	6.2	1677216	903450
20	5.4	6.7	1677216	1103450

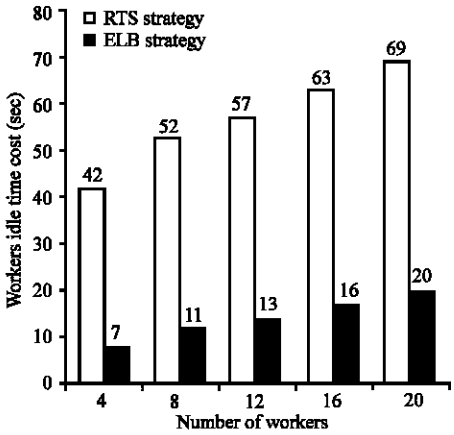

Fig. 2: Measured average workers' idle time cost (sec) in five DCS configurations obtained in RTS and ELB strategies

Runtime Task Scheduling (RTS) strategy: The RTS strategy has a potential to absorb the machine's performance variation characteristics and non-homogeneous nature of the application at runtime tasks assignment. The number of tasks processed by the worker is proportional to its runtime performance. The performance of RTS strategy is evaluated using the fixed unit of task (one frame of the image file).

On the average, RTS has 25% degradation with respect to ELB strategy. Due to the large number of master's requests occurred in RTS strategy (Table 2), more workers idle time cost is generated (Fig. 2).

ELB strategy: The strategy is based on adaptive nature. Therefore, the number of request made by the master to distribute the whole task is less as compared to RTS strategy (Table 2). Due to incorporation of compression/decompression of data (task/file), the overhead of interprocess communication is reduced 50% as compared to RTS (Fig. 2).

This strategy is evaluated on maximum of 20 PC/WS. It has average improvement in speedup of 25% over RTS strategy.

CONCLUSIONS AND FUTURE WORK

In this work we have proposed and implemented an efficient load balancing strategy for parallel visualization of blood head vessel angiography on cluster of workstations/personal computers. The performance of the proposed strategy was compared with RTS strategy for speed up, number of requests made by the master and workstation idle time cost. The ELB strategy has 25% approximate speedup improvement over RTS strategy. The workers idle time cost also reduces approximately to 50% with respect to RTS strategy. In the future, we plan to investigate the high lossless data compression methods, which may be useful for such applications that demand high resources of space and network usages. We intend also to investigate a more effective load balancing strategies that should be capable to further reduce the number of request/reply and idle time cost of the machines/workers.

ACKNOWLEDGMENT

The publication of this study was supported by Yarmouk University Research Council, Irbid, Jordan.

REFERENCES

1. Sharma, P., E. Perry and R. Malpani, 2003. IP multicast operational network management: design, challenges and experiences. IEEE Network, 17: 49-55.
2. Prasad, R., C. Dovrolis, M. Murray and K. Claffy, 2003. Bandwidth estimation: Metrics, measurement techniques and tools. IEEE Network, 17: 27-35.
3. Chang, Shi Kuo, G. Polese and M. Cibelli, 2003. Visual authorization modeling in e-commerce applications. IEEE Multimedia, 10: 44-54.
4. En-Hui Yang and J.C. Kieffer, 2000. Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform. I. Without context models. IEEE Transactions on Information Theory, 46: 755-777.
5. Hann-Huei Chiou, Alexander I-Chi Lai and Chin-Laung Lei, 2000. Prediction-capable data compression algorithms for improving transmission efficiency on distributed systems. 20th International Conference on Distributed Computing Systems, 2000. ICDCS 2000, pp: 654-661.

6. Krintz, C. and B. Calder, 2001. Reducing delay with dynamic selection of compression formats. 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10'01), pp: 266-280.
7. Kalim Qureshi and Masahiko Hatanaka, 2000. A practical hybrid approach of task partitioning and scheduling on heterogeneous parallel distributed image computing system. Transaction of IEE Japan, 120-C: 151-157.
8. Najib, A., Kofahi and Q. Quazi Abidur Rahman, 2004. Empirical study of variable granularity and global centralized load balancing algorithms. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'04), Las Vegas, Nevada, USA, June 21-24, 2004, CSREA Press, 1: 283-288.
9. Khan, R.Z., Kalim Qureshi and A.Q. Ansari, 2002. Adaptive decentralized task scheduling strategy for a heterogeneous distributed ray tracing system. CSI J. Computer Sci. Informatics, 32: 26-31.
10. Yingwu Zhu and Yiming Hu, 2004. Towards efficient load balancing in structured P2P systems. Proceedings of the 18th International Parallel and Distributed Processing Symposium, pp: 20-29.
11. Kalim Qureshi and Masahiko Hatanaka, 2000. Parallel raytracing on heterogeneous distributed computing system. J. Current Sc. Special Issue on Computational Sci., 78: 818-820.
12. Kalim Qureshi and Masahiko Hatanaka, 2001. A practical approach of task scheduling and load balancing on heterogeneous distributed raytracing system. Information Processing Letter (IPL), Elsevier Press, 79: 65-71.
13. Boklund, A., Christian Jiresjö and Stefan Mankefors, 2003. The story behind midnight, a part time high performance cluster. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA '03, June 23-26, 2003, Las Vegas, Nevada, USA, 1: 173-178.