1997

Using ATM Networks for Processing Global Earth Data

Barbara L. Kess
University of Nebraska-Lincoln

Phillip R. Romig III
Colorado School of Mines, promig3@mines.edu

Stephen E. Reichenbach
University of Nebraska-Lincoln, reich@cse.unl.edu

Ashok K. Samal
University of Nebraska-Lincoln, asamal1@unl.edu

USING ATM NETWORKS FOR PROCESSING GLOBAL EARTH DATA*

Barbara L. Kess bkess@cse.unl.edu
Phillip R. Romig III romig@cse.unl.edu
Stephen E. Reichenbach reich@cse.unl.edu
Ashok Samal samal@cse.unl.edu

Department of Computer Science & Engineering
University of Nebraska, Lincoln
Lincoln, NE 68588-0115

ABSTRACT

Current workstation technology provides an unprecedented amount of computational power to researchers at an affordable cost, making it feasible to use workstations rather than expensive supercomputers to perform scientific analysis of large data sets, such as the Global Land 1-Km AVHRR data. In addition to this, inexpensive high speed ATM networks have the potential to improve the overall computational efficiency of workstations by using several workstations in a distributed environment.

This research studies the practicality of using distributed workstations interconnected with a 155 Mb ATM network for analysis and compression of the Global Land 1-Km AVHRR data versus sequential computing on one of the workstations. Performance comparisons are given for three algorithms associated with the compression of these data. Based on the test results, a discussion is given regarding the practicality of using a distributed system to enhance performance when processing global earth data.

1. INTRODUCTION

The proliferation of powerful inexpensive workstations, or desktop computers, is making it feasible for users in the remote sensing community to use workstations for data analysis of large data sets, whereas previously they had to arrange for time on supercomputers which are often expensive and inconvenient to use. Even though computing power is increasingly affordable, processing large data sets, such as the Global Land 1-Km AVHRR data [1], still poses significant problems in terms of processing time and disk resources. One possible solution is to process large data sets in a distributed environment, in which several workstations are interconnected via a Local Area Network (LAN) and simultaneously working on the same problem. Previous to current technological developments, this was not a viable alternative for processing global earth data because the speed of LAN technology was too slow to efficiently distribute large data sets such as the global AVHRR data. However, the introduction of new high speed network technology may change this and allow for efficient computing on global earth data over a network of distributed workstations. In this paper the performance of several algorithms for compression and analysis of Global Land 1-Km AVHRR data is reported using one workstation versus a distributed systems with one server and eight clients. A discussion compares the characteristics of problems that are well suited for the upcoming generation of distributed systems to problems that are still solved more efficiently in a monolithic environment.

2. MODELING DISTRIBUTED PERFORMANCE

One goal of this research is to model the performance of distributed algorithms for the analysis and compression of the Global Land 1-Km AVHRR data set, to determine in advance whether an algorithm merits the time and expense involved in converting it from a sequential to a distributed program. To do this a simplified model for runtime performance is used. Given an algorithm that reads in N data elements, performs operations on them, and then stores M data elements back onto the disk, the algorithm is divided into two parts: a sequential part which contains start up and shutdown overhead and any other operations that are not done in parallel, and a parallel part which contains all operations that may be executed in parallel. The time spent executing code which is not parallelizable is labeled T_s and the time spent executing code which is parallelizable is labeled T_p. The times required to read and write one data element to and from the disk are labeled as D_r and D_w, respectively. An equation for modeling the runtime of the sequential algorithm is given as

$$R_s = (D_r * N) + T_s + T_p + (D_w * M). \quad (1)$$

When modeling the distributed case, it is necessary to add the time required to transfer the data to and from the clients. Given a network which can transfer one data element in ω seconds, the time required to transfer the data to the clients is $\omega * N$, and the time required to transfer the results back to

*Support was provided in part by NASA Graduate Student Researchers Program grants NGT-51293 and NGT-595-204 and the Center for Communication and Information Sciences (CCIS) at the University of Nebraska.

the server is $\omega * M$. If P is the number of clients, then the distributed runtime is modeled as

$$R_d = (D_r*N)+(\omega*N)+T_s+\frac{T_p}{P}+(\omega*M)+(D_w*M). \quad (2)$$

The speedup gained by distribution is the ratio, $S = R_s/R_d$. This means that two conditions must hold in order for distribution to provide a speedup. First, the network transfer time ($\omega * N$) must be small. This paper assumes that the transfer time is indeed small when using a high-speed ATM network. The second condition is

$$(D_r * N) + (D_w * M) + T_s << T_p \quad (3)$$

If the left side of this inequality is not significantly less than T_p, then the speedup ratio, S, will tend towards one. If T_p is larger than the other terms then the speedup will approach P. Assuming that the network transfer time is small, performance prediction is based on the above inequality.

3. DESCRIPTION OF THE ALGORITHMS

Performance comparisons are given for three algorithms used by Kess, Steinwand and Reichenbach [2] to analyze and compress Global Land 1-Km AVHRR data. The algorithms vary in their computational complexity and although none of the algorithms has a high computational complexity, they are representative of typical algorithms used to analyze and compress large data sets. One of the merits of a good data compression algorithm is low computational complexity, making it unrealistic to report results from highly complex algorithms as representative of whether distributed systems can improve the speed of data compression algorithms for large data sets. The data used in the tests are the NDVI band of the April 1-10, 1992 data set, which contains 694,417,757 samples of 8-bit data. The first algorithm computes the histogram, the second computes the entropy of the residual image that results from each of the eight lossless JPEG [3] linear predictors, and the third algorithm compresses the data.

Computing the histogram is easily parallelized. In the distributed implementation each client receives a portion of the image, tallies the counts for the histogram, and sends the final results back to the server. The server sums the results from all of the clients and writes the final histogram to a file. This is a very simple program and the computation required for both T_s and T_p is very small. This means that the disk read time, $D_r * N$, will dominate the speedup ratio, S, and as a result S is expected to approach one. Thus, the distributed histogram is expected to perform only slightly better than the sequential algorithm.

The second algorithm finds the entropy of the residuals created from using each of the JPEG linear predictors for lossless compression. This algorithm was used by Kess *et al.* to determine which of the eight linear predictors in the lossless JPEG compression standard performs the best on the Global Land 1-Km AVHRR data. In the distributed version each client computes the residuals for a portion of the image. Each client sends eight histograms back to the server. As the server receives results from the client, it adds the histograms together. After all of the results are received, the server computes the entropy of each histogram and writes the results to an output file. The sequential time, T_s, is expected to be larger than T_s is for the histogram algorithm, but still small. The total processing time for tasks performed by the clients, T_p, is expected to be significantly larger than T_p for the histogram, and in fact it should dominate the sequential processing time, T_s, the disk read time, D_r*N, and the disk write time, D_w*M given in Inequality refpred. Hence, the entropy program is expected to give a speedup that is significantly greater than one.

The third program compresses the land data with the method used by Kess *et al.* This compression program divides the image into subimages and compresses each subimage independently of the others. Approximately 80% of the image is composed of solid regions for the water, interrupted areas in the Goode's Homolosine map projection, and land where there is no data. These areas are compressed by Kess *et al.* with a quadtree algorithm and the results from this compression are not reported. The land compression algorithm reads in all of the subimages and only compresses the data values that represent land. There are approximately 135 million land samples in the image. The compression algorithm for the land computes a residual image using the JPEG lossless linear predictor with the lowest entropy from the JPEG entropy tests, and then uses adaptive Huffman coding to compress the residual image.

This algorithm, like the others, is also easy to distribute because the compression is performed independently on subwindows of the image and the data file is preprocessed so that the subwindows are stored contiguously in the data file. The number of operations for each data element is the highest of all three programs, but these operations are only performed on the 20% of the data that represents land values. Thus, the expected total parallel processing time, T_p, is less than the JPEG entropy process. The sequential time, T_s, and the disk read time, $D_r * N$, are similar to the other two algorithms. However, the disk write time, $D_w * M$ is significantly larger because the server writes the compressed data to a file. Thus, the speedup is expected to be greater than one, but not as large as the JPEG entropy speedup.

4. THE DISTRIBUTED SYSTEM

Results are reported using a homogeneous workstation cluster consisting of eight Hewlett Packard 9000/715s and a server HP 9000/735. All the machines are connected by an ATM network running *classical* TCP/IP over ATM[4]. PVM[5] was used for message passing and synchronization. Results are given for two different methods of distributing data to the pro-

Computing Technique	Program Execution Times		
	Histo-gram	JPEG Entropy	Land Compress
Sequential	8:17	25:25	20:10
Distributed			
Centralized Data	12:50	19:00	16:43
Distributed Data	6:34	12:06	15:51

Table 1: Runtimes in Minutes:Seconds

cessors. The first method uses a Network File System (NFS) in which the file system is centralized and the entire image is stored on a single disk that is connected to the system via a 20MB/s Fast/Wide SCSI III interface. The second method makes use of local disk drives to perform a simplified type of data striping. In this method the image is divided into equal size subimages and one subimage is placed on the local disk of each client in the distributed system.

5. RESULTS

Table 1 presents the sequential and distributed results for each of the three algorithms discussed in this paper. The distributed results are presented for centralized data and data distributed to the local disks of each client. The sequential tests were performed on the 735 which is a faster machine than the 715's used as clients for the distributed tests. Because of this the reported runtimes for the sequential algorithms are slightly faster than if they were executed on the clients. However, none of the 715s had enough local disk space to run the sequential algorithm without using the network and thereby incurring a network transfer penalty for the execution time. Although executing the sequential algorithms on the faster processor decreases the speedup ratios, the processor speed inaccuracy is not as large as the extra network transfer time incurred by running the sequential algorithm on a client processor.

The speedup ratios for the distributed algorithms were 0.64 for the histogram, 1.33 for JPEG Entropy and 1.2 for the compression algorithm. By distributing the data these values improved to 1.2, 2.1, and 1.2 respectively. The model did not accurately predict a slow down for the distributed histogram with centralized data. This is because the model does not incorporate the overhead required for distribution. The results for the distributed histogram with distributed data improved because the disk read time was less.

It is interesting to note the speedup improvement when the data is stored locally on the client disks rather than on the centralized server. There are at least two advantages for using a decentralized file system. One advantage is that a decentralized file system processes the data without using large, fast and expensive disks, and the other advantage is it achieves the best speed of all the methods tested. This speed of the decentralized file system is also impressive because the local disks are 5MB/s SCSI disks, which is much slower than the 20MB/s FW SCSI III central disk.

6. CONCLUSION

Clearly, distributed processing of global AVHRR data is beneficial in some cases. It takes time to modify existing programs to run in a distributed environment, which is an important consideration before deciding to distribute an algorithm. If programs are originally written for distribution, then the programming time is not an issue. The simple model given in section 2 provides a starting point for analysis of distributed versus sequential processing. A modification is needed to incorporate the overhead required for distribution. While the model is mathematically straightforward, quantifying prediction values for the terms in the model is difficult.

The results suggest that distributed computing with distributed data has more benefits for large data sets than distributed computing with centralized data. Decentralized data storage can use slower and smaller disks than centralized data storage and still achieve better runtime performance. One implication of this is that large data sets could be stored in a distributed format for faster retrieval from permanent storage devices. For example, compressed browse images could be stored on a local disk for easy browse retrieval and compressed full resolution data could be permanently stored in distributed archival locations for distributed retrieval.

7. REFERENCES

[1] J. Eidenshink and J. Faundeen. The 1-Km AVHRR Global Land Data Set: First stages in implementation. *International Journal of Remote Sensing*, 15(17):3443–3462, 1994.

[2] B. Kess, D. Steinwand, and S. Reichenbach. Compression of the Global Land1-Km AVHRR Data Set. *International Journal of Remote Sensing*, To appear.

[3] W. Pennebaker and J. Mitchell. *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, 1993.

[4] M. Laubach. Classical IP and arp over ATM (update). `ftp:\\ftp.com21.com`, August 1995.

[5] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Mancheck, and V. Sunderam. *PVM, Parallel Virtual Machine.* MIT Press, 1994.