N87-26533

COMPARISON OF THE MPP WITH OTHER SUPERCOMPUTERS FOR LANDSAT DATA PROCESSING

by
Martin Ozga
United States Department of Agriculture
National Agricultural Statistics Service

ABSTRACT

The MPP is compared to the CRAY X-MP and the CYBER-205 for Landsat data processing. The maximum likelihood classification algorithm is the basis for comparison since this algorithm is simple to implement and vectorizes very well. The algorithm was implemented on all three machines and tested by classifying the same full scene of Landsat MSS data. Timings are compared as well as features of the machines and available software.

INTRODUCTION

The National Agricultural Statistics Service (NASS) of the United States Department of Agriculture has for several years been using Landsat MSS (multispectral scanner) data to aid in crop acreage estimation. The estimates are provided for several states, mostly in the Midwest. For each state, as much Landsat data as possible is obtained, the ideal being coverage for the entire state. Therefore, large amounts of data must be processed. This large-scale processing has long been seen as a useful application of supercomputers, first the ILLIAC-IV and more recently the CRAY X-MP (Ref. 1). One of the key programs in the analysis is the classification or categorization of the data into classes representing crop types. This classification is done on all scenes for all states.

LANDSAT MSS DATA

Each Landsat MSS scene consists of about 10.5 million pixels. Each pixel covers a square on the earth's surface of about 57 meters on a side. Each pixel consists of the reflectance intensity in 4 spectral bands or channels. The intensity is a value between 0 and 255 (more commonly between 0 and 127) and thus occupies 1 byte of data. Thus each pixel occupies 32 bits of data, 1 word on many machines or a half word on many supercomputers.

Often, to improve classification accuracy, multitemporal data are used. A multitemporal data set is made up of two overlaid Landsat data sets from the same area on the earth's surface. The improvement comes in that the two data sets are taken at different times and, if the dates are chosen properly, may emphasize differences in reflectance of various crops. However, since the multitemporal data sets contain eight channels of data, they do take longer to process.

For the USDA-NASS 1986 analysis, approximately 80 scenes of Landsat data will be used of which approximately 25 will be multitemporal. For the purposes of the test described in this paper, only unitemporal data sets were used.

THE CLASSIFICATION ALGORITHM

The maximum likelihood classification algorithm consists of applying a discriminant function for each class to each pixel and assigning the pixel to the class for which the discriminant function yields the highest value. Since each pixel is processed independently, the algorithm is ideal for vector machines and indeed performs well on all machines tested. The discriminant functions is:

$$G(X, I) = B(I) - .5 (X - M(I))^{T} V(I) (X - M(I))$$

where X is the pixel
I is the class
M(I) is the vector of mean values for class I
V(I) is the inverted variance-covariance matrix for class I
B(I) = .5 log (determinant (V (I))

There are three parts to the algorithm. First, the data must be unpacked and converted to floating point form. That is, each byte (channel intensity value) must be converted to the equivalent floating point value. Second, the discriminant function must be applied for each class to each pixel and the class for which the discriminant function yields the highest value saved for each pixel. Third, the data are repacked to one byte per pixel for economy in storage for further processing. This means that the number of classes must be less than or equal to 255; in practice at USDA-NASS the number of classes is always well under 255. Obviously the second step is by far the most time consuming.

MPP IMPLEMENTATION

The MPP classification algorithm was implemented on the MPP using Parallel Pascal (Ref. 2, 3, 4). The entire algorithm was implemented without requiring use of assembly language so Parallel Pascal was certainly adequate to the task, although, perhaps coding the key loop of discriminant functions evaluation in assembly language may have speeded up the algorithm somewhat.

The input-output was done using supplied fast video routines callable from Parallel Pascal. The data was mapped so that all four channels of each pixel were transferred to the same PE. Of the 1024 bits available in each PE, 512 were used for pixel storage allowing 16 pixels to be stored in each PE at any one time, or 262144 pixels in the entire PE memory.

Thus, 42 I/O operations were necessary to bring in all the data. When the program was written and debugged, facilities were not available for overlapping processing and I/O. The class associated with each pixel was stored over a portion of the original pixel values since once the pixel was converted to floating point, the original values were not needed any longer. An additional 128 bits were needed for the four 32-bit floating point channel values. Additional bits were used for various temporary values. Some space was left available for a possible future expansion to multitemporal, eight-channel data.

It is evident that when processing 10.5 million pixels at 16384 at a time, all PE's are generally doing productive work. The lack of hardware floating point did slow this algorithm. Some tests on the CRAY X-MP with integer processing revealed that integer processing led to severe deterioration of classification results unless certain intermediate results were stored is very large integers, being due to the significant digits which must be maintained using the inverted variance - covariance matrix.

The repacking of data, a rather tedious tasks on the other machines, was easily realized by proper storage of the data on the MPP and correct maps for sending the data through the staging buffer to the output categorized file.

CRAY X-MP IMPLEMENTATION

The CRAY X-MP/48 at NASA-Ames was used for the test. The program, which is in production use by USDA-NASS, is coded in Cray FORTRAN (CFT) (Ref. 5) with key routines coded in CRAY assemble Language (CAL) (Ref. 6). The use of assembly language is mainly for historical reasons since at the time this program was originally written, the CFT compiler was less well developed than it is today. If this program were re-written, much more of it would be coded in CFT.

The CRAY X-MP vectorization uses as operands vector registers each of 64 64-bit words. The vectorization is achieved largely through pipelining data. There is some advantage to having longer vectors, particularly when coding in CFT, since overhead for vector set-up is decreased. A vector size of 16384 was used for the CRAY implementation.

CYBER-205 IMPLEMENTATION

The Control Data CYBER-205 at NASA-

Ames was used for the test. The program was coded in FORTRAN (Ref. 7), but makes extensive use of special extensions and subroutines. This was necessary since at the time the program was written and debugged, the CYBER-205 FORTRAN compiler did a rather poor job of vectorization, a condition which is being gradually improved.

Vector operations on the CYBER-205 operate directly out of memory and may be of indeterminate length, limited by the size of memory. Vectorization on the CYBER-205 is achieved largely by pipelining data. The CYBER-205 implements virtual memory with paging so there is a potential loss of efficiency if a vector crosses a page boundary or if two operands to a vector operator are in different pages in that a page fault may cause a break in the vector operation. With these constraints in mind, the CYBER-205 operates best on long vectors. Also, internally, vector operands are pointed to by descriptors which may be used explicity by the FORTRAN programmer desiring maximum efficiency. Finally, CYBER-205 FORTRAN provides many subroutine calls to perform vector operations which are not directly expressed in FORTRAN, that is for most of the capabilities of the machine outside of ordinary arithmetic operations. In effect, when all these features are used, one has a parallel FORTRAN, somewhat analogous to Parallel PASCAL on the MPP.

The classification algorithm was coded on the CYBER-205 using descriptors, several of the special subroutines, and with consideration for paging. The vector length was 16384. Such a coding is in the parallel form of FORTRAN and it provides for an efficient use of the machine.

CPU TIMES

The test performed involved classifying a full scene of data (10.5 million four channel pixels) into 51 categories. The same results, as determined by a count of the number of pixels per category, were obtained on all machines except for a difference of two pixels on the CYBER-205, an insignificant difference.

The following CPU times were observed:
MPP: 90 seconds (approximate)
CRAY X-MP: 157 seconds
CYBER-205: 58 seconds.

It is interesting to note that the MPP lies between the CRAY X-MP and the CYBER-205 in timings. Improvements in the code generated by Parallel Pascal or the use of assembly language would no doubt bring the timing down closer to that of the CYBER-205; similar improvements on the CYBER-205 are unlikely since the code already takes advantage of special features of the CYBER-205. One would suppose that implementation of hardware floating point operations on the MPP would cause it to run faster than the CYBER-205 on this problem.

Both the MPP and the CYBER-205 are known to operate best on long vectors whereas the CRAY X-MP has a shorter vector but faster scalar speed. The classification algorithm certainly shows the advantages of the long vectors when applied to an algorithm for which they can be profitably used.

The CPU times for the CRAY X-MP and CYBER-205 were obtained directly from the listing received from batch execution of the program. Such a listing was not available for the MPP. Therefore, calls to timing routines were placed around the main classification loop. These timings totaled 88.76 seconds. The CPU times spent in the remainder of the program would be quite small.

ELASPED TIMES

Elapsed times on the CRAY X-MP and CYBER-205 are not important since both machines are operated in a multi-programming environment so the elapsed time would be very dependent on the mix of jobs in the system.

However, only one job is present at any one time on the MPP so that the MPP user would expect to be charged for the entire

elapsed time. The total job time on the MPP was 360 seconds. This time fluctuated somewhat, presumably depending on usage of the front-end VAX. The test was run at a time of relatively low VAX use, on a Friday evening. Since I/O-CPU overlap was not available, the time spent in reading the input file was measured to try to get an approximation of savings to be obtained if overlap were present. The input file was used since it is largest and only the input file was used since only one I/O operation may proceed at any one time. Approximately 150 seconds was spent reading the input file indicating that the program was I/O bound. Thus, if CPU and input I/O could be overlapped by reading smaller pieces of the input file into buffers in PE memory, one would expect the total job in the decrease to about 270 seconds. The time not accounted for in the above calculations, 120 seconds, was spent in writing the output file and also in reading the statistics file and preassembly in some overhead in communicating with the VAX. Thus, one would, in a production made, expect to pay for 270 seconds of MPP time to do a 90 second job. Of course, on the CRAY X-MP and CYBER-205 one is typically changed for I/O and often memory usage. The wide variance in applying these changes from site to site makes comparison difficult.

EASE OF USE

Ease of use is necessarily subjective term, based on one's opinions and experiences with various system. Nevertheless, some comments can be made. I found the MPP generally easy to use. Parallel Pascal provides a good interface to the machine. The biggest problem in Pascal and other languages is that one must often use special routines for I/O. Once I learned about the special routines, I found them generally convenient to use. The problem of using up the limited memory of the MPP with the stack space required for complicated expression can be severe; perhaps an option should be added to Parallel Pascal to tell the user how much stack space is used by any particular statement. Otherwise, at least for this particular problem, the size of the PE memory was not really a constraint.

Landsat processing is characterized by moving large amounts of data. While the facilities provided on the MPP were adequate for the test, they would not be adequate for production use. For the test, the Landsat data tape was copied on the VAX to a removable disk pack and this disk pack was then made available for the test. What is needed for production are facilities typically available at supercomputer sites wherein data are staged between tapes and dedicated, high-capacity, high-speed disks.

In general, I found both the software and hardware to be reliable. It was not necessary or even advantageous to use the MPP simulator since the MPP itself was generally available. Debugging facilities were perhaps a bit crude but adequate.

CONCLUSIONS

The MPP has promise for Landsat data processing. The most urgent need is for improved data handling. However, since the speed in this test was slower than on the CYBER-205, and since both are characterized by requiring long vectors, perhaps a more complex PE implementing hardware floating point is in order. It would be interesting to do a comparison on some algorithm requiring only or mostly integer computations. Unfortunately, none of the major programs currently used by USDA-NASS for Landsat processing fits this requirement so no such test was performed.

ACKNOWLEDGEMENT

The author would like to thank James Tilton of the NASA Goddard Space Flight Center for his generous help in answering my many questions and in locating appropriate subroutines, particularly for I/O.

REFERENCES

1. Ozga, Martin, "Experience with the Use of Supercomputers to Process Landsat Data," Symposium Proceedings,

Machine Processing of Remotely Sensed Data, Laboratory for Application of Remote Sensing, Purdue University, West Lafayette, Indiana, 1984.

2. Parallel Pascal Language Reference Manual, NASA Goddard Space Flight Center, 1986.

3. Parallel Pascal User's Guide, NASA Goddard Space Flight Center, 1986.

4. MPP Pascal Callable Procedure Library, NASA Goddard Space Flight Center, 1986.

5. FORTRAN (CFT) Reference Manual, SR-0009, Cray Research Inc., Mendota Heights, Minnesota, 1984.

6. CAL Assembler Version 1 Reference Manual, SR-0000, Cray Research , Inc., Mendota Heights, Minnesota, 1983.

7. FORTRAN 200 Version 1, 60480200, Control Data Corporation, Sunnydale, California, 1984.