

STRATEGIES FOR ALTERNATE APPROACHES FOR VEGETATION INDICES COMPOSITING USING PARALLEL TEMPORAL MAP ALGEBRA

Bijay Shrestha

Charles G. O'Hara

Nicolas H. Younan

GeoResources Institute

Mississippi State University ERC

2, Research Blvd.

Starkville, MS 39759.

bijay@GRI.MsS-tate.edu

cgohara@GRI.MsState.edu

younan@ECE.MsState.edu

ABSTRACT

Remotely sensed images from satellite sensors such as MODIS and AVHRR provide high temporal resolution and wide area coverage. Unfortunately, these images frequently include undesired cloud and water cover. Areas of cloud or water cover preclude analysis and interpretation of terrestrial land cover, vegetation vigor, and/or analysis of change. Multi-temporal image compositing techniques may be employed to create a synthetic cloud free image that includes representative values derived from a set of possibly cloudy satellite images collected during a given time period of interest. However, spatio-temporal analytical processing methods that utilize moderate spatial resolution satellite imageries with high temporal resolution to create multi-temporal composites are data intensive and computationally intensive. Therefore, a study of the compositing strategies using high performance parallel solutions based on their computation and IO characteristics is required. This research focuses on analyzing alternate compositing strategies for vegetation indices using parallel temporal map algebra. The report provides objective findings on computational expense, IO complexity, and the relative benefits observed from various analysis methods and parallelization strategies.

INTRODUCTION

Remotely sensed images from satellite sensors such as MODIS and AVHRR provide high temporal resolution and wide area coverage, and therefore are highly useful in performing land use analysis. Unfortunately, these images frequently include undesired cloud and water cover. Areas of cloud or water cover preclude analysis and interpretation of terrestrial land cover, vegetation vigor, and/or analysis of change. Multi-temporal image compositing techniques may be employed to create a synthetic cloud free image that includes representative values derived from a set of possibly cloudy satellite images collected during a given time period of interest (Cihlar et al., 1994). This study is basically concerned with vegetation indices. Therefore, all the references to composites would implicitly refer to composites of vegetation indices.

Spatio-temporal satellite image analysis is a technique for monitoring spatial and temporal changes of land cover and oceanic locations on earth. Temporal Map Algebra (TMA) is a novel technique developed by Jeremy Mennis and Roland Viger for analyzing a time series of satellite imagery using simple algebraic operators that treats time series of imagery as a three-dimensional data set, where two dimensions encode planimetric position on earth surface and the third dimension encodes time (Mennis and Viger, 2004). The high dimensionality of raster data leads to high computational cost, which is why parallel computation is attractive. This paper describes the design, implementation, and performance evaluation of parallel compositing of vegetation indices derived from MODIS datasets using TMA.

TEMPORAL MAP ALGEBRA

The quantitative nature of digital images provides a foundation for the spatial mathematics. Map algebra is a useful spatial analysis and cartographic modeling framework using simple mathematics operators (Tomlin, 1990). In this approach, raster data handling is performed by treating spatial data as variables and applying mathematical local, focal and zonal operations on them.

Temporal map algebra (TMA) functions are temporal extensions to conventional map algebra. The temporal map algebra functions treat a time series of imagery as a three dimensional data set, where two dimensions encode planimetric positions on Earth's surface, and the third dimension encodes time. This allows the time series of satellite imagery to be manipulated according to well-established and standardized raster processing techniques. Mennis and Viger used temporal map algebra for determining the El Nino /Southern Oscillation (ENSO) effect on southern African vegetation intensity over different land cover types. The temporal map algebra function assisted in effectively obtaining the ENSO effects over various land cover from a time series NDVI datasets using a zonal function for a given multi-temporal dataset. The study concluded that the use of cube function approach assisted in implementation of temporal map algebra functions. The logical data model for temporal map algebra is a regular tessellation of R3, a three dimensional matrix.

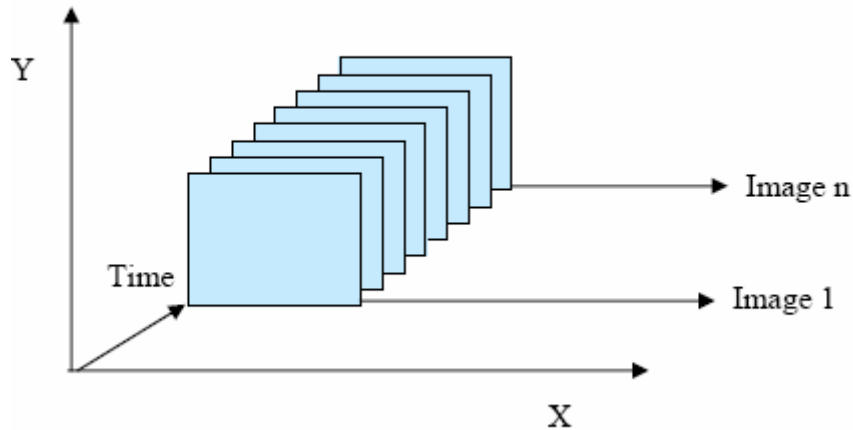


Figure 1. Data Model for Temporal Map Algebra

Mennis and Viger imply that temporal map algebra takes one or more grids as input and outputs a grid, but this study uses input of one or more grids and outputs a raster. Local and focal functions will be used for creating composites using TMA. The main objective of using temporal map algebra for vegetation index compositing is to obtain improved capabilities to perform compositing processes on a multi-temporal datasets such as a time series NDVI and extract the required information effectively. The functions that are of interest for compositing are listed in the following table.

Table 1. Operational Complexity for TMA

Functions	Function Complexity
Local Maximum, Local Minimum, Local Mode	$O(n)$
Local Median	$O(n \log n)$
Focal Maximum, Focal Minimum, Focal Mode	$O(n)$
Focal Median	$O(n \log n)$

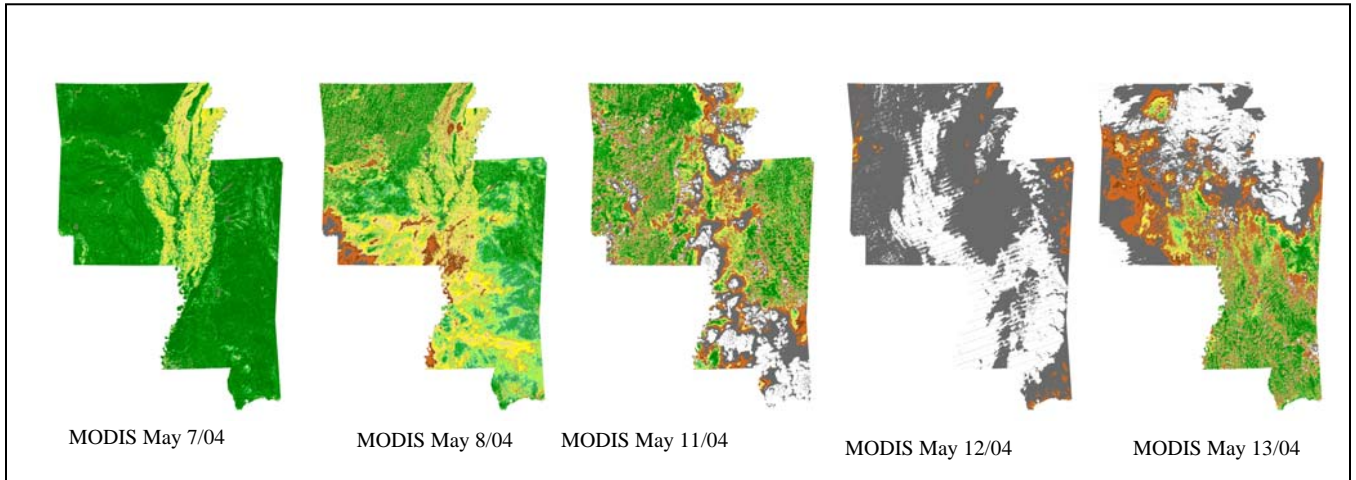


Figure 2. Some examples of cloud covered daily dataset for MODIS (250m) within the temporal range of May 05- May 18. Grey Color shows cloud cover, the low NDVI values are in orange and yellow colors, the increase in NDVI is shown by increase in darkness of green color. In MODIS datasets the white color represent the 'No Data' values.

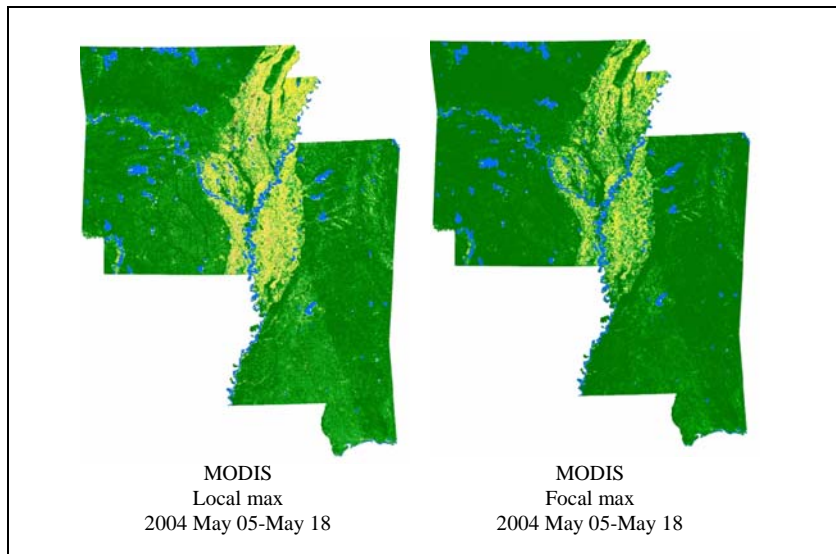


Figure 3. Biweekly MODIS Composites for Mississippi Arkansas Study Region for period of May 05-May 18, 2004. The color mapping is similar as in Fig.4.

Temporal map algebra can be used for spatial analysis that are performed by map algebra over a period of time using local, focal and zonal statistics functions like minimum, maximum, mean, median, mode, variance, etc. Local functions will give statistics for each point over time; focal functions will give temporal statistical values for a defined region over each pixel; and zonal functions will give statistical data with respect to each defined region.

Mali et al. have utilized temporal map algebra to create vegetation index composites of AVHRR and MODIS datasets (Mali et al., 2005a). The results from temporal map algebra compositing were then compared with traditional compositing results. They have concluded that compositing using Focal Maximum criteria produces cloud free composites with good results. The algorithm takes a model-based approach to manipulate input data by considering surface reflectance quality and scan angles to select data that meet model criteria for input to the TMA compositing process.

PARALLEL PROCESSING

Parallel processing is performed by decomposing one large process into small processes that can be solved simultaneously and hence providing faster execution time (Grama et al., 2003). Thus, parallelization provides a way to solve a large computationally intensive process like temporal image compositing, which would otherwise take a very long time on a sequential machine.

A parallel application needs to divide a large problem into a set of smaller problems so that the problems may be allocated to a set of processes running on the processors of a parallel machine for simultaneous executions. There are two aspects of parallel decomposition: data decomposition and functional decomposition. Functional parallelism is based upon the assumption that a computational task may be broken into relatively independent functions or processes. Parallelism is then achieved by assigning these functions or processes to multiple processors. This technique best suited for MIMD machines where different processes can run on different processors. Pipelining is an example of functional decomposition.

In this study, regular domain decomposition is relevant as the data sets concerned are three-dimensional raster grid data represented as three dimensional matrices. Any parallel application requires methods for reading input data from disk, distribution of data among processors, gathering resultant data from processors, and writing the combined resulting data to disk.

Performance Metrics for Parallel Systems

It is important to study the performances of parallel programs to determine the best algorithm, evaluate hardware platforms and to examine the benefits from parallelism. Below, a number of metrics that are widely accepted for the evaluation of performance of parallel system are presented.

Execution Time. *Serial runtime* of a program is the total time taken by the program to execute on a sequential computer. *Parallel runtime* is the time that is elapsed from the moment a parallel computation starts to the moment the last processing element finishes execution. In this document, serial runtime is denoted by T_s and parallel runtime by T_p , where p is the number of processors used in parallel processing.

Speedup. Speedup represents the performance gain achieved by parallelizing a given application over a sequential application. Speedup is defined as ratio of the time taken to solve a problem on a parallel computer with p identical processing elements and is denoted by the symbol S .

$$S = \frac{T_s}{T_p}$$

Where, T_s represents serial runtime and T_p represents parallel runtime using p number of processors.

Efficiency. Efficiency is a measure of the fraction of time for which a processing element is usefully employed and is defined as the ration of speedup to the number of processing elements.

$$E = \frac{S}{p}$$

In an ideal parallel system, speedup is equal to the number of processors employed and efficiency is equal to one. However, because of the overheads incurred, efficiency is generally less than 1.

Cost. Cost of solving a problem on a parallel system is defined as the product of parallel runtime and the number of processing elements used. Cost denotes the sum of the time each processing element spends solving the problem.

$$Cost = pT_p$$

Total Parallel Overhead. The overheads incurred by a parallel program are encapsulated into a single expression referred to as the overhead function. The overhead function is the total time collectively spent by all the processing elements over and above that required by the fastest known sequential algorithm for solving the same problem on a single processing element. The overhead function is represented by the symbol T_o .

$$T_o = pT_p - T_s$$

The total time spent in solving a problem summed over all processing elements is pT_p . T_s units of time are spent on performing the useful work, which is also the serial unit. The difference between the two represents the overhead, as represented by the equation above.

METHODOLOGY

After computing NDVI/EVI for each image for a temporal period, maximum value compositing (MVC) technique chooses the greenest pixel value (i.e., the highest value in the range [-1, 1]) of a location over a period of time. The selection of greenest pixel implies that the observation was acquired under conditions when the atmosphere contained the least amount of water vapor (Holben, 1987). United States Geological Survey (USGS) constructs weekly and biweekly NDVI composites from multiple AVHRR daily observations. They also produce various other data products of the environment from AVHRR observations (USGS, 2005). CV-MVC (Constrained View Angle – Maximum Value Compositing) is another modification applied to MVC to obtain further enhanced composites by only taking pixel values that are near to nadir.

The compositing procedures are as follows:

1. Collection of spatial data items of interest: this step involves the collection of the satellite imagery of the spatial area of interest; the satellite imageries of interest are from MODIS sensors.
2. Preprocessing of spatial data: this step includes the preprocessing of the satellite imagery which includes atmospheric, geometric, and radiometric corrections; then NDVI and satellite angles need to be calculated.
3. Creation of data cubes: this step is the actual creation of the data cubes. MODIS data comes with satellite sensor angle and quality information about each pixel; three data cubes namely NDVI, View Angle, and Surface Reflectance Quality are created.
4. Masking: this step utilizes model-based approach for manipulating data streams and creating a subset of observations that meet view angle and quality criteria. This process is generally referred to hereafter as masking.
5. Creation of composites using TMA: this process is the actual creation of temporal composites using the TMA functions; the TMA function focal maximum with angular constraint is used, considering satellite view angle and the quality value constraints.

The MODIS data sets that were used were required to create the NDVI, Quality and Angle cubes are MYD09GQK, MYD09GST, and MYDMGGAD. MYD09GQK is 250m resolution surface reflectance data, MYDMGGAD is 1 Km resolution Geo-location angles data and MYD09GST is 1 Km resolution surface reflectance quality data (NASA, 05).

Computation cost of spatio-temporal analysis using TMA depends on the number of input data-cubes, size of a problem, and the complexity of the operations that are needed for analysis. We can construct multiple temporal

image cubes to encode attribute information such as image quality, scan angle, or other attribute per each pixel. Therefore, multiple cubes may be utilized to manipulate image data and generate model-specific results.

RESULTS AND ANALYSIS

The compositing process consists of the following steps namely acquisition, preprocessing, and composite computation using Temporal Map Algebra. All the different processes can be parallelized if automated processes can be developed for acquisition and pre-processing as all of them are lengthy and time-consuming process, and therefore an ideal candidate for parallel processing. However, this study focused on the actual composite computation using Temporal Map Algebra.

Mali et al. have compared the results of various compositing algorithms using temporal map algebra and have concluded that compositing using Focal-Maximum criteria produces cloud free composites with good results (Mali et al., 2005a). Therefore, in this study we compare the performance evaluation of the compositing algorithm using Focal Maximum criteria. The algorithm also takes into consideration the surface reflectance quality and geo-location angles metadata derived from the MODIS data products.

The evaluation is mainly based on the compositing process on fourteen day MODIS data cubes using focal maximum criteria of Temporal Map Algebra. The size of the input datasets, which are of dimensions $6^\circ \times 6^\circ$. The details about the files have been provided in the Table 2. The input files contained bi-weekly *NDVI* data, scan angle metadata, and surface reflectance quality mask data. The datasets used were derived from MODIS data products to construct bi-weekly cloud-free synthetic image composites.

The applications were developed using MATLAB, MatlabMPI (Kepner, 2001) and LibTIFF libraries. The experiments were conducted on the EMPIRE cluster at ERC at Mississippi State University. The EMPIRE cluster is an IBM Linux cluster with 1038 1 GHz and 1.266 GHz Pentium III processors on 519 nodes and 607.5 Gigabytes of RAM. The system has a total of 9.7 terabytes of disk space acting as secondary memory, with each processor having a two Gigabytes of personal disk space.

The timings have been measured over a number of iterations, and the average time for all the iterations are represented in the graphs. All the timings shown in this chapter are in seconds. Instead of presenting one graph for each set of readings, an effort is made to provide a subset of graphs to illustrate the performance.

Serial Run Time

This test measures the serial run-times taken for the computation of fourteen day MODIS composites. The statistics related to the input datasets for $6^\circ \times 6^\circ$ cubes, and the average serial run-time are tabulated in Table 6.2. As can be seen from the table, the total I/O required for the operation is nearly 650 MB. The serial run-time for the process of compositing is approximately 13 minutes. The serial run-time is composed of the time taken to read-in the input data sets, computation of each point in output data set, and the actual writing-out of output TIFF image.

Table 2. $6^\circ \times 6^\circ$ Data and Serial Run-Time using Focal Max Algorithm

$6^\circ \times 6^\circ$ Cube	Properties
Cube Size	$2411 \times 2856 \times 14$
Resolution	231.64×231.65 meters
NDVI File Size	368.05 MB
Angle File Size	184.18 MB
Mask File Size	92.25 MB
Total I/O	644.48 MB
Average Serial Run-Time	765.81 seconds

Parallel Run-times

To perform this experiment, parallel software was developed using MatlabMPI that could simultaneously execute MATLAB in more than one processor. The software works such that the different chunks of centralized input data, located in a Network File Server, are read simultaneously by different processors and then processed. This is a classic case of block decomposition, where the input data is divided and assigned to different processors. There are 3 three- dimensional cubes of same dimensions as input sets: *NDVI* cube, Quality-Mask cube and the Angle cube. Therefore, each processor calculates the dimensions of the multi-dimensional input data that it needs to read. The precise chunk of data is then read by each processor from the three input data cubes. The processors then compute the composite from the cubes using the focal maximum value compositing, with angular constraint and surface reflectance quality mask. The processed results are then written as a MATLAB mat file in a central file server, which is then read by central processor and then combines the results to give the final output composite image in a TIFF format.

Table 3. Run Time Statistics

Number of Processors	Average Time (Seconds)	Maximum Time (Seconds)	Minimum Time (Seconds)
1	765.81	834.15	739.02
4	251.95	333.85	199.15
8	151.54	171.05	139.35
10	153.31	185.50	117.22
12	152.64	196.57	124.17
14	152.61	174.02	138.75
16	152.98	185.19	114.31
20	149.48	182.94	108.78
25	159.85	209.21	127.16
30	168.09	186.53	146.77

Table 3 illustrates the maximum, minimum, and average run times for centralized data. Figure 4 represents the values in a graph.

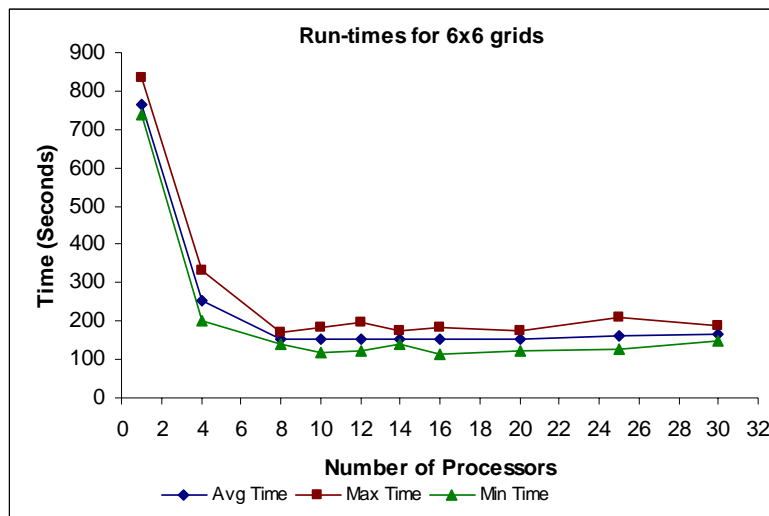


Figure 4. Run-Times with Centralized Data

Parallel Run-times With Distributed Data

The bandwidth available to each node, the network load during the execution of the application, and the cluster bandwidth are also important factors to be considered as they may adversely affect the run-times by affecting the I/O performance. To meet the high performance I/O needs of our application, parallel I/O is a requirement. Parallel I/O would make our application independent of the network load. The computation facility at ERC currently does not support parallel I/O. Therefore, an experiment was conducted to simulate parallel I/O on different numbers of processors. Each individual node of the EMPIRE cluster has a local disk drive, which were used for the distributed I/O.

Table 4. Run time statistics

Number of Processors	Maximum Time (Seconds)	Minimum Time (Seconds)	Average Time (Seconds)
1	834.15	739.02	765.81
4	262.96	246.97	253.17
8	192.47	175.48	182.58
10	188.16	165.30	170.66
12	166.71	157.69	160.75
14	177.29	159.57	163.83
16	175.80	156.45	165.04
20	179.57	159.40	164.37
25	180.28	164.25	171.05
30	191.43	170.43	181.65

Table 4 illustrates the maximum, minimum, and average run times for distributed data. Figure 5 represents the values in a graph.

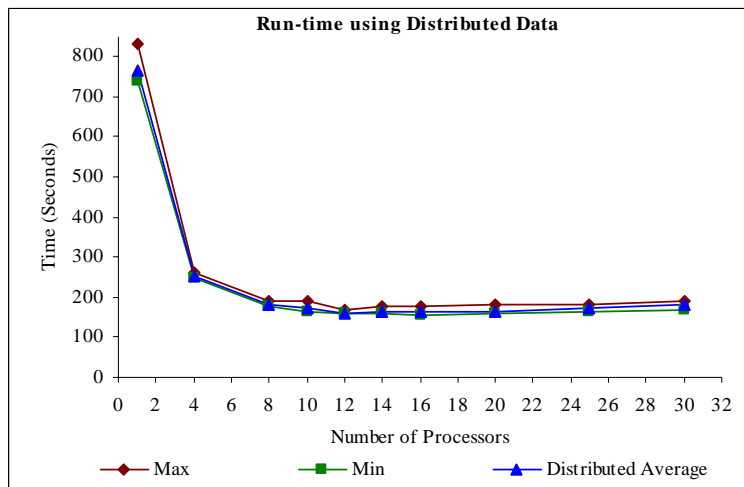


Figure 5. Run-Times with Distributed Data

As the number of processors is increased, the application becomes more fine-grained, thereby reducing the problem size for each processor and leading to faster execution time. However, we can observe that this is not the

case and timing results do not decrease significantly as we increase the number of processors. The curves on the graphs reach the lowest value when number of processors is eight and twelve respectively.

Performance Metrics

The previous sections provided the serial execution time, centralized data parallel execution time, and distributed data parallel execution time. Now let us use the results from the previous section to compute the quality metrics: speedup, efficiency, and total parallel overhead.

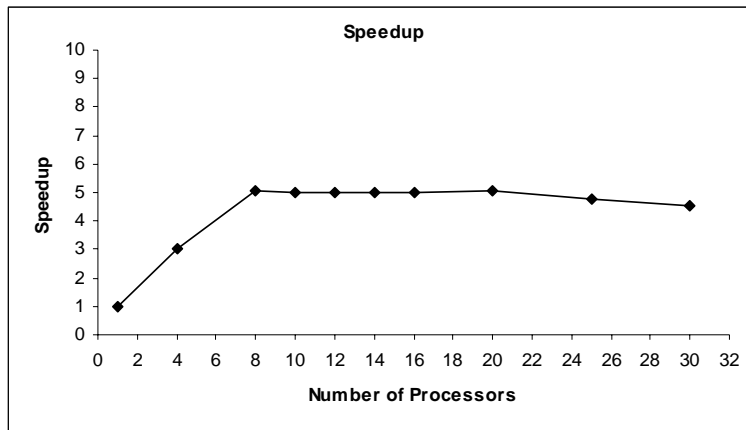


Figure 6. Speedup

Figure 6 charts the speedup obtained using various numbers of processors and Figure 7 represents the efficiency obtained. As can be seen from the figures, maximum speedup of around five was achieved with 60% efficiency. It can be noted is that we get moderately better speedup as we increase the processors from one to eight. From that point onwards, the number of processors does not make much of a difference and the run-times remain more or less the same.

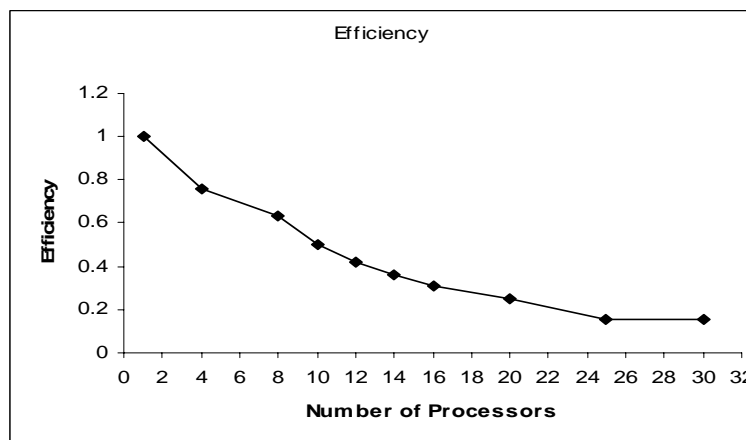


Figure 7. Efficiency

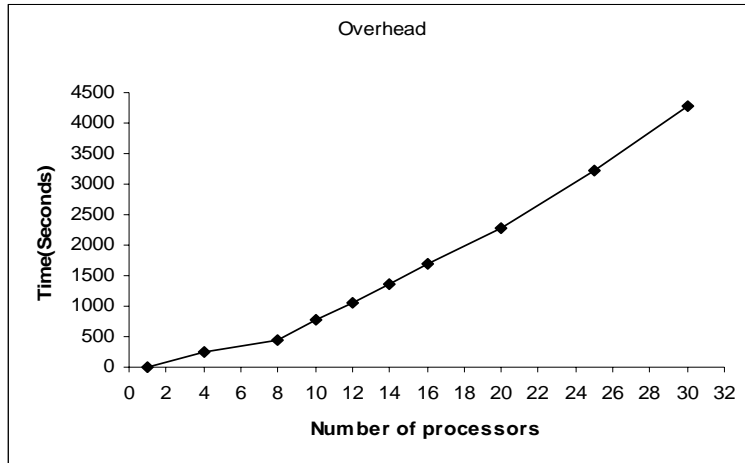


Figure 8. Overhead

From the earlier sections, we have observed that the run time stabilizes after a certain number of processors. Figure 6 illustrates that point and shows the speedup for each sets of processors used. The speedup curve moves almost linearly from 1 to the peak of 5.05. The maximum point is reached when number of processors used is eight. From that point onwards, there is not a significant change when more processors are used. A parallel result can be seen from Figure 7, which shows that the efficiency of the application is decreasing as more processors are used. As expected, since the speedup is not being gained as more processors are being added, there is significant loss of efficiency. Similarly, Figure 8 shows the overhead of the application is increasing almost linearly as the number of processors increase. The use of MatlabMPI as the programming environment has added some overhead to the total serial and parallel times for the computation. MATLAB is a large programming environment and the overhead to startup Matlab is higher than the startup of executables with smaller overhead.

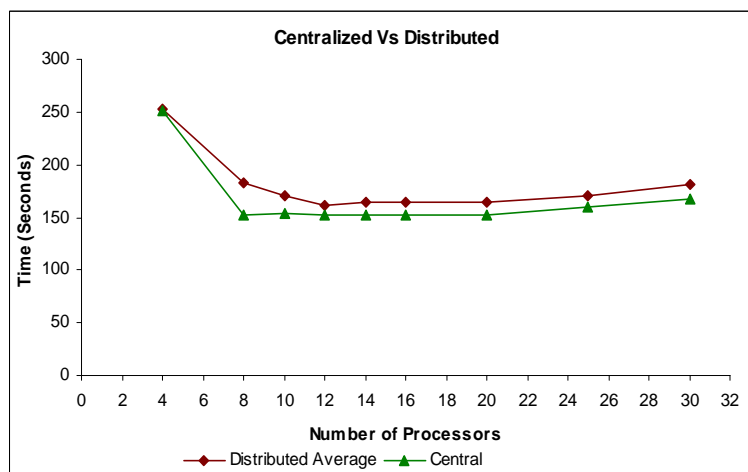


Figure 9. Distributed Versus Centralized Run-Times

The total time taken for compositing for the centralized and the simulated distributed data is plotted in Figure 9. The extra overhead incurred in the distributed simulation due to the copy operation is evident in the graph as we can see that centralized input data-sets consistently provide better timing than the distributed process.

CONCLUSION

The main motivation of the research was to check the feasibility of parallel Temporal Map Algebra using MatlabMPI. The performances were analyzed using various metrics. From the analysis in this research, we can say that our results are good evidence that support our hypothesis. The accuracy of the results, however, is affected by various factors such as network bandwidth, network traffic, processor speed, and network/cluster architecture. The use of MatlabMPI as the programming environment has added some overhead to the total serial and parallel times for the computation. MATLAB is a large programming environment and the overhead to startup Matlab is higher than the startup of executables with smaller overhead. We know that MATLAB has high overhead cost and initializing MATLAB using MatlabMPI in a number processor takes a considerable amount of time. The application was also programmed using MATLAB's interpreted system and that also added some overhead. These overheads can be reduced by programming in C or C++ that have lower overhead. An important point to be noted is that Temporal Map Algebra can be used to solve more complex spatio-temporal analysis problems on large datasets with many dimensions. Further work can be done to use achieve better performance in newer problems with higher complexity.

REFERENCES

- Cihlar, J., D. Manak, and M. D'Iorio (1994), "Evaluation of Compositing Algorithms for AVHRR Data Over Land," *IEEE Transactions on Geoscience and Remote Sensing*, 32(2):427 – 437.
- Grama, A., A. Gupta, G. Karypis, and V. Kumar (2003), *Introduction to Parallel Computing*, second edition, Addison Wesley.
- Holben, B. N (1987), "Characteristics of Maximum-Value Composite Images from Temporal AVHRR Data," *International Journal of Remote Sensing*, 7(11):1417–1434.
- Healey, R., S. Dowers, B. Gittings, and M. Mineter (1997), *Parallel Processing Algorithms for GIS*, CRC Press.
- Kepner, J. (2001), "Parallel Programming with MatlabMPI," *Proceedings of the High Performance Embedded Computing (HPEC 2001) Workshop*.
- Leffler, S., "Libtiff," www.libtiff.org (current 22 Jan. 2005).
- Mennis, J., and R. Viger (2004), Analyzing Time Series Satellite Imagery Using Temporal Map Algebra. *Proceedings of the American Society for Photogrammetry and Remote Sensing Conference.. Matlab 7 User's Guide(2004)*, The MathWorks, Inc.
- Mali, P., C. O'Hara, B. Shrestha, and V. Vijayraj (2005a), "Use and Analysis of Temporal Map Algebra for Vegetation Index Compositing," *International Workshop on the Analysis of Multi-Temporal Remote Sensing Images*.
- Mali, P., C. O'Hara, R. Viger, V. Vijayraj, J. Mennis, and B. Shrestha (2005b), "Vegetation Index Compositing and Analysis in Spatial and Temporal Dimensions," *Proceedings of the American Society for Photogrammetry and Remote Sensing Conference*.
- NASA, "MODIS Home Page," <http://modis.gsfc.nasa.gov/> (current 2 Feb. 2005).
- Tomlin, C. D. (1990), *Geographic Information Systems and Cartographic Modeling*, Prentice Hall, Englewood Cliffs, New Jersey.
- United States Geological Survey, "AVHRR Normalized Difference Vegetation Index (NDVI) Composites," <http://edc.usgs.gov/products/landcover/ndvi.html> (current 27 Jan. 2005).
- Xiong, D., and D. F. Marble (1996), "Strategies for Real-Time Spatial Analysis Using Massively Parallel SIMD Computers: an Application to Urban Traffic Flow Analysis," *International Journal Geographical Information Systems*, 10(6):769–789.