Geographical Information System data Handling with Open Multicore Processing

Natalija Stojanovic[1], Dragan Stojanovic[2]
[1]Faculty of Electronic Engineering
University of Nis, Aleksandra Medvedeva 14
18000 Niš, Serbia
natalija.stojanovic@elfak.ni.ac.rs
[2]Faculty of Electronic Engineering
University of Nis, Aleksandra Medvedeva 14
18000 Niš, Serbia
dragan.stojanovic@elfak.ni.ac.rs

ABSTRACT: *We have developed a multicore architecture to support the geospatial data management. For the multicore processing and using Geographic Information Systems, we have used parallel processing based on the application of Open Multi Processing. This will help to improve the use of map-matching computing system for the large spatial datasets with road segment networks. The algorithm designed is useful for performance of multicore system. We have conducted testing and found that the high performance computing in geographical information system is possible for validation.*

Keywords: High-Performance Processing, GIS, Multicore Processors, OpenMP, Map-Matching, Viewshed Analysis

Received: 10 January 2022, Revised 4 May 2022, Accepted 19 May 2022

DOI: 10.6025/jic/2022/13/3/60-66

1. Introduction

Many today's data and computing intensive applications require more processing power then even before. Computer games, database searching, Web search engines, financial and economic forecasting, climate modeling, medical imaging, etc. are application domains from a broad range of applications that demand accelerating and performance improvements [1]. One of the approaches for achieving performance improvements, which has been used in last decades, is to decrease the size of components used to build computers. The technology progress has made possibility to put billions of transistors on a single chip. As the size of transistors has decreased their speed also has increased and therefore the speed of overall integrated circuit has to be increased. But, increasing of the transistor speed, as a major approach of computer performance advance, has limited due to heat-dissipation. As a consequence of heat dissipation integrated circuit could become unreliable [2]. These limitations have led to alternative strategies for creating more powerful computers, and have made the use of parallelism to become one of the crucial solutions for accelerating applications. For example, chip manufactures have started to produce processors with several computing units, called cores, on one chip, that have independent control and have access to the same, shared memory. Therefore, using multicore processor has made each desktop or a laptop computer a commodity parallel system. It is important that

application software must be able to make effective use of parallelism that is present in available hardware resources. Software developers can't except that the increasing of computer power can be automatically used by their application programs. This is due to the fact that there is not appropriate automatic transformer from a sequential program to a parallel program that runs efficiently on the new architectures. Research in the area of parallelizing compilers has shown that in many situations it is not possible to extract enough parallelism from sequential programs. Instead, a software developer has to transform its software to run efficiently on new architectures.

Therefore, OpenMP (Open-Multiprocesing) was developed to enable creation of parallel programs for shared-memory multiprocessor platforms. OpenMP represents a set of compiler directives, library routines, and environment variables that provide programmer to tell the compiler which instructions to execute in parallel. Also, OpenMP provides programmer to define how to distribute them among the threads that will run the code.

Many modern software applications, which are developed to model real world, process a large amount of data and thus cause long execution times on today's computers. One such application domain that could significantly benefit from parallel processing is Geographic Information System (GIS) that includes processing and analytics of massive geospatial data (Big Geospatial Data). In this paper, we use OpenMP parallel implementation for a map matching, a process of integration of raw vehicle trajectory data to underlying road network providing richer semantics in registered movement. Also, the algorithm for viewshed analysis is parallelized and accelerated on a multicore system. The performances of our implementations have been evaluated. Experimental results are examined and discussed.

2. High-performance Computing in GIS

There are many ways to parallelize applications in order to improve their performance. Just as there are several different classes of parallel hardware, so there are different models of parallel programming. Therefore, OpenMP (Open- Multiprocesing) was developed to enable development of parallel programs for shared-memory multiprocessor platforms. Also, there are other high performance parallel/distributed techniques and methods. For example, MPI (Message Passing Interface) programming library use multiple computers connected by high-speed LAN for distributed computing, while OpenCL (Open Computing Language) and CUDA (Compute Unified Device Architecture) enable general-purpose applications to access massively parallel Graphics Processing Units (GPU) for nongraphical computing [2].

OpenMP is not a new programming language. Rather it is a notation that can be added to a sequential program in C, C++ and Fortan to describe how the work can be shared among the threads that execute on different processors or cores and to order access to shared data as needed [3].

OpenMP supports the so-called fork-join programming model. This approach assumes that the program begins execution as a single thread, just like an ordinary sequential program. The thread that executes this piece of code is called initial (main) thread. Every time this thread encounters OpenMP parallel construct during program execution, it creates a set of threads. It then becomes a parent thread and cooperates with other threads of the program execution. (Fig. 1). At the end of parallel construct only the initial (parental, main) thread continues execution, interrupting the execution of others.

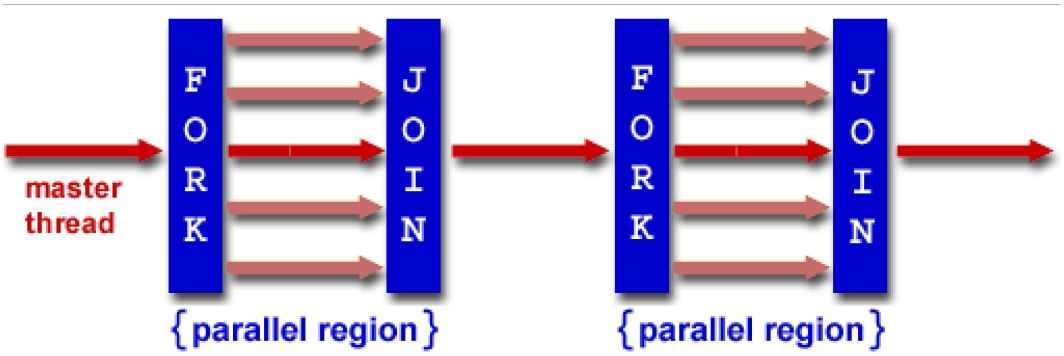

Figure 1. Fork-join programming model

OpenMP is expected that the developer specifies the parallel parts of the program and the method of parallelism applied. Thus, it provides a notation to indicate the area of OpenMP program that should be executed in parallel. Also, it is possible to obtain additional information on how this should be achieved. OpenMP job is to classify the parts of the program and create appropriate threads, as well as to allocate a piece of code to execute by each thread. The method of work division can have a significant impact on the program performance.

Until recently, the application of high performance computing techniques especially those available at PC workstations and their networked clusters are mostly neglected. Nowadays, with the increasing volume of geospatial data required for computational- and data-intensive problem solving in different GIS application domains, it has emerged as a prominent research area [4], [5].

Akhter et al. [6] develop a methodology and propose GRASS GIS module extension with parallel and distributed computing for remote sensing image processing. Different implementations for distributed GRASS modules are examined on three different programming platforms (MPI, Ninf-G and OpenMP) and their performance are presented.

Zhang in [7] considers a new HPC framework for processing geospatial data in a personal computing environment. He argued that modern personal computers equipped with multi-core CPU and many-core GPU provide excelent support for spatial data processing comparing with cluster computing using MPI and newly emerged cloud computing using MapReduce framework.

In our previous paper [8] we evaluate two parallel/distributed architectures and programming models: MPI (Message Passing Interface) over network of workstations (NoW) and CUDA (Compute Unified Device Architecture) on GPU in well-known problems in GIS: map matching and slope computations. Experimental evaluations indicate improvement in performance and shows feasibility of using network of workstations and GPU for high performance computing in GIS.

3. Geospatial Data Processing Using OpenMP

In this paper, we consider the spatial join between a large dataset related to trajectories of moving objects and the dataset on the road network on which they move in order to perform map-matching. The result of the map-matching process is a dataset containing points at the road segments that are the closest to the appropriate trajectory points. This way moving points that represent trajectories are matched to corresponding road segments at which their movements occur.

In order to achieve map-matching, finding nearest road segment in a series of segments for each point in the series of points is needed. Points are given by their x and y coordinates and segments are represented as polylines defined by the coordinates of their vertices. Finding the nearest segment for the corresponding point is performed by determining the minimal distance between a point and corresponding segment. Determining this minimal distance is presented in Figure 2.

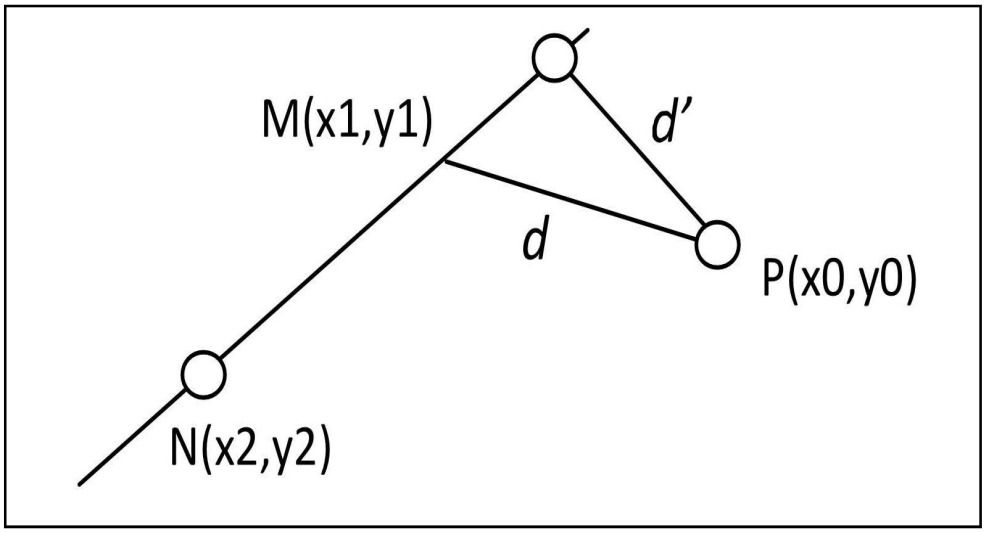

Figure 2. Determining minimal distance between point an a line segment

Determining the shortest distance beetween a point and a segment depends on a value of scalar dot between vectors NM and MP, i.e. scalar dot between vectors MN and NP and is calculated using following expression:

$$d = \begin{cases} \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}, \overrightarrow{NM} \cdot \overrightarrow{MP} > 0 \\ \sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2}, \overrightarrow{MN} \cdot \overrightarrow{NP} > 0 \\ \dfrac{Ax_0 + By_0 + C}{\sqrt{A^2 + B^2}}, otherwise \end{cases}$$

where A = y1-y2, B = x2-x1 and C = x1y2-x2y1.

In order to accelerate the sequential execution of map matching computation using parallel OpenMP techniques the distribution of iterations to the corresponding threads is performed. Within each iteration, the coordinates of the point lying on the corresponding segment which is part of matched trajectory, are found. Iterations are evenly distributed among threads.

Viewshed analysis is one of the common algorithms in terrain spatial analysis and use of terrain models. A viewshed is the area(s) of the land/terrain surface that is visible from one or more viewpoints. Viewshed analysis is used in a variety of applications such as locating radio and TV transmitters and cellular communication base stations for maximum coverage, site selection for the forest lookout stations and determining the areas of the resort that would be visible from the new restaurant to determine its scenic qualities.

The location and height of the viewpoint can change the size and shape of a viewshed. Viewing parameters can also include the viewing angle, the search distance and even the tree height. The process of deriving viewsheds is called viewshed or visibility analysis. A viewshed analysis requires two input datasets. The first dataset represents a point layer containing one or more defined viewpoints. The second input dataset represents the land surface and can be defined as a DEM (Digital Elevation Model) - a raster spatial data containing elevation at spatial grid points, or a TIN (Triangulated Irregular Network).

The viewshed analysis is performed following a series of steps [9]. First, the location of the viewpoint is connected by a line of sight or a ray to all specified location in the terrain. Second, a set of intermediate points is derived along each sightline. These intermediate points are determined from the intersection between the sightline and the grid lines of the elevation raster (DEM). Finally the computing algorithm examines the elevations of the intermediate points, looking for locations that are higher and thus visible from the viewpoint or not (Figure 3.) Higher points along this sightline obscure the lower points that are behind them. The above procedure is repeated for each grid point in the elevation raster as a target, or only for specified locations of interest. In the parallel OpenMP implementation the viewshed computing for the whole set of grid points is evenly distributed

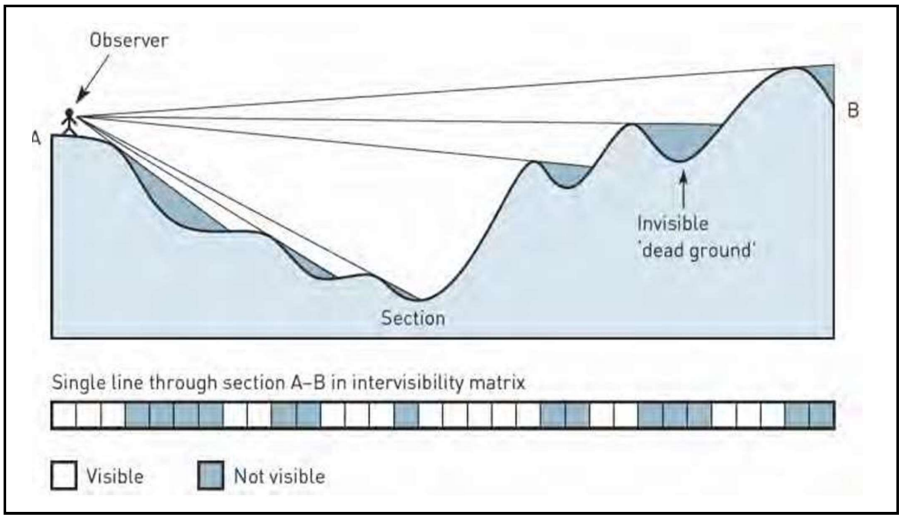

Figure 3. Ray tracing for visibility analysis [9]

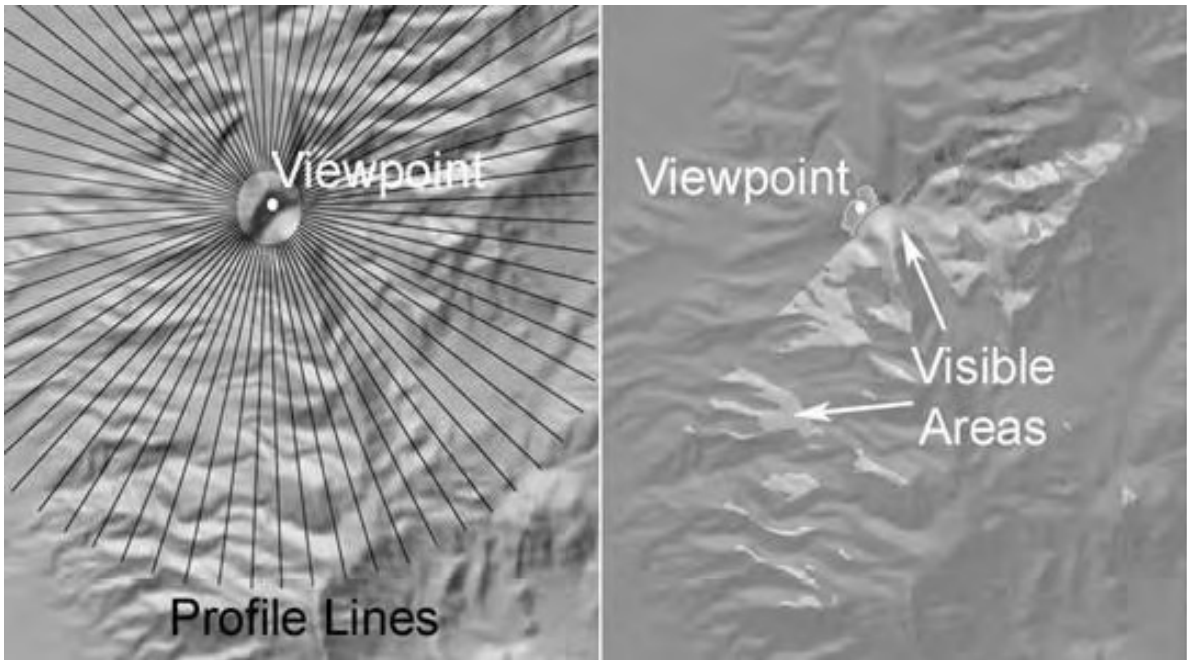

Figure 4. Results of viewshed analyses for single point

among several threads (e.g. 2, 4, 6, 8, 10, etc.) and thus can be performed on separate cores in a multi core architecture. Using OpenMP the effective use of a multicore architecture is achieved.

Through repeated 'ray tracing' a viewshed map is built. This is a Boolean raster map that classifies raster cells into the visible and invisible groups and indicates which areas of terrain are visible or not from the location of interest. This map is more effectively interpreted and understood by its visualization over the terrain surface (Figure 4.).

Various algorithms have been developed for computing viewsheds and appropriate tools are available in several contemporary commercial and open-source GIS software. Such tools provide basic capabilities for setting up the parameters for viewshed analysis but usually do not provide choices for algorithms and information of adopted algorithm.

Both map matching and viewshed analysis algorithms are time consuming and computationally intensive operations that are performed by applying relatively simple geometric operations over massive vector or raster geospatial datasets. As such these operations are well suited for speeding up by employing parallel processing techniques in the computation. In the next section we will examine application of OpenMP on common GIS algorithms, map matching and viewshed analysis to show improvement in performance and feasibility of effective use of multicore architecture for high performance computing in GIS.

4. Experimental Evaluation

In order to estimate proposed parallel solutions we used the speedup as measure. Speedup (S_P) is used to compare the execution time of observed computation on p processors (T_P) to the execution time of sequential version (T_1) i.e.

$$S_p = \frac{T_1}{T_P}$$

As the values of speedup values are indication of processor's performance, the maximum obtained speedup can be used to locate the best number of threads to be used for corresponding application on observed architecture. In our experiments for both applications we used Intel Core 2 Duo T5870 2GHz CPU, 4GB of RAM, and Intel i7-2670QM 2,2GHz, 4GB of RAM multicore architecture. Intel Core 2 Duo has two cores, while Intel Core i7 has 4 cores with implemented hyper-threading (HT) technology.

With hyperthreading technology, the instructions from two threads are interleaved in the processor pipeline. Circuits of the processor which store architectural state of processor are duplicated, but the main execution resources are shared and not duplicated. For the development of parallel application on multicore computer, Visual Studio 2008 with included support for OpenMP is used.

In the case of map-matching application two datasets are used and stored in corresponding files. The first dataset represents the set of segments (each polyline consists of straight line segments) where each record contains segment identifier and (*x, y*) coordinates of the segment endpoints. The segment dataset contains 7035 records. The second dataset contains the set of moving points where each record contains point identifier, sequence number, the timestamp, (*x, y*) coordinate of point and the current speed. The point dataset consists of 329273 records. In the case of viewshed application DEM with 1201x1201 points is used.

Performance of map-matching application as well as viewshed application, obtained for different number of threads on the Intel Core 2 Duo and Intel Core i7 are shown in Table 1 and Table 2. In both tables the execution time (in seconds) and speedup for corresponding thread count are shown.

Core 2 Duo			i7		
threads	T_P	S_P	threads	T_P	S_P
2	181,006	1,824	2	71,585	1,954
4	186,489	1,771	4	42,661	3,278
6	191,441	1,725	6	31,016	4,509
8	222,351	1,485	8	25,823	5,416

Table 1. Experimental Results For Map-matching Algorithm

Core 2 Duo			I7		
threads	T_P	S_P	threads	T_P	S_P
2	221,185	1,991	2	149,575	1,826
4	250,743	1,756	4	84,276	3,240
6	282,675	1,558	6	73,443	3,719
8	245,047	1,797	8	65,632	4,162

Table 2. Experimental Results For Viewshed Algorithm

For Core 2 Duo the best performance results are when using two threads, while performance decreases for both algorithms when engaging more than two threads. For the Core i7 processor, hyper-threading technology enhances the speedup for applications with two to eight threads. After that, it stagnates. The maximum speedup in the case of mapmatching is obtained for eight threads and its value is 5,416.

For the viewshed algorithm the maximum obtained speedup is for eight threads and its value is 4,162. The speedup values more than 4 are obtained due to HT technology.

From Tables 1 and 2 it can be concluded that the obtained speedup makes this OpenMP implementations a candidate for sequential acceleration solutions. As can be concluded from the experimental evaluation, the use of OpenMP implementation for map-matching algorithm and viewshed analysis is a major challenge, and as a result gives satisfactory performance improvements.

5. Conclusion

With advances in remote sensing, geosensor networks and pervasive positioning, the amount of geospatial data that needs to be processed and analyzed has exploded in recent years. It leads to a rising interest in using high performance parallel and distributed techniques for large scale geospatial data processing and analysis. This paper shows that the application of OpenMP parallel processing technique to effectively employ multicore computer architecture could improve the performance in executing common computation and data intensive GIS algorithms, such as map matching and viewshed analysis, over large scale raster and vector geospatial data.

The future research will consider the usage of other HPC techniques and platforms in GIS, such as cloud computing (MapReduce/Hadoop) and the adaptation of various GIS algorithms to cloud infrastructure.

Acknowledgments

Research presented in this paper is funded by Ministry of education, science and technological development, Republic of Serbia as part of the project 'Environmental Protection and Climate Change Monitoring and Adaptation', Nr. III-43007.

References

[1] Patel, S. & Hwu, W.W. Accelerator architectures. IEEE Micro, 28,4(2008), 4–12 [DOI: 10.1109/MM.2008.50].

[2] Pacheco, P. (2011). An Introduction to Parallel Programming, Morgan Kaufman.

[3] Chapman, B., Jost, G. & Van Der Pas, R. (2008). *Using OpenMP: Portable Shared Memory Parallel Programming.* MIT Press: Cambridge, USA.

[4] Clematis, A., Mineter, M.J. & Marciano, R. (2003). *High performance computing with geographical data*, *Special issue Parallel Computing*, 1275–1279.

[5] Shekhar, S. (2010). High Performance Computing with Spatial Datasets, Invited Talk, *International Workshop on High Performance and Distributed GIS*. San Jose, CA, USA.

[6] Akhter, S., Aida, K. & Chemin, Y. (2010). GRASS GIS on high performance computing with MPI, *OpenMP and Ninf-G programming framework*. Proceeding of ISPRS (2010). Japan.

[7] Zhang, J. (2010) Towards personal high-performance geospatial computing (HPC-G): Perspectives and a case study, *International Workshop on High Performance and Distributed GIS*. San Jose, CA, USA, pp. 3–10.

[8] Stojanovic, N. & Stojanovic, D. (2013). www.inderscience.com/info/ingeneral/forthcoming.php?jcode=ijris High-performance computing in GIS: Techniques and applications. International Journal of Reasoning-Based Intelligent Systems, 5 [DOI: 10.1504/IJRIS.2013.055126].

[9] Heywood, I., Cornelius, S. & Carver, S. (2012). *An Introduction to Geographical Information Systems*, 4th Edition. Prentice Hall: Englewood Cliffs, USA.