Parallel Neighbourhood Modelling *†

D. Hutchinson, M. Lanthier, A. Maheshwari,
D. Nussbaum, D. Roytenberg, and J.-R. Sack
PARADIGM Group, School of Computer Science
Carleton University, Ottawa, Ont., Canada K1S 5B6
<http://www.scs.carleton.ca/~gis>

1 Introduction

It has been observed that in recent years Geographical Information Systems(GIS) and Spatial Information Systems have gone through substantial changes with respect to users, problems, problem domains, and data. The GIS community is rapidly expanding to include users from different sectors of the economy; along with this come different demands regarding the type, required speed, scope and scale of applications. Users in decision making positions require rapid, close to instantaneous responses even to complex queries. In addition, users today have access to an unprecedented amount of high resolution and high-quality data through scanners, satellites, range finders, medical equipment and other devices.

The effect of these changes is a rapid and huge increase in the computational demands placed on GIS. Processing a large raster often takes hours or even days; (processing a large raster, say of size 6000x6000 cells, at a speed of 1000 cells per second, would take 10 hours). Lengthy computation times in different applications have been reported as stated e.g., in [22]. To keep up with the computational demands without sacrifice (i.e., reduction in resolution or scope of model), parallel computing appears to be the only solution. Parallel hardware is readily available at a good price-to-performance ratio (in the small to medium range).

Several researchers and groups have used or advocated parallelism for GIS. This includes the work described in [13, 17, 10, 5, 25, 23, 14, 47, 6, 19, 42, 38, 36, 37, 22, 31, 43, 46, 15].

Faust et al. [18] state that "*the real issues in computing in the next decade involve innovations that will allow relatively unsophisticated users to access the power of the computer hardware, without having to become experts in programming and computer operating systems. The tools for GIS .... should become easier to use ... and at the same time be able to take advantage of the new advances in hardware and software technology.*"

While parallelism in GIS appears to be necessary, the task of providing parallelism is highly challenging, requiring novel ideas and specialized knowledge from areas of computer science outside GIS. Our primary research and development objective is to enable users, researchers and developers within GIS to use parallel computers without paying the high price of having to deal with the complex issues inherent to it (we provide *transparent parallelism*).

Here we report on our first major milestone towards achieving this objective: the design and prototype implementation of an environment for parallel raster-based NEighbourhood MOdelling (NEMO). NEMO is primarily intended for (but not necessarily restricted to) coarse-grained parallelism with a limited number of processors.

2 NEMO

Before we discuss the NEMO system, we define neighbourhood modelling. Let g_{ij} be a cell in an $m \times n$ raster corresponding to the location of a point in a geographic terrain (each cell stores one or more attributes, e.g., urban, rural). Cell g_{ij} may be time variant (denoted by $g_{ij}(t)$ for time t). We define the *neighbourhood* $N(g_{ij})$ of a cell g_{ij} to be a set of cells in the raster associated with g_{ij}. Typically, $N(g_{ij})$ includes all of the cells which are within the vicinity of g_{ij}. *Neighbourhood modelling* involves performing one or more operations on each raster cell and its neighbourhood. The operation performed is a function which calculates a new value for a cell or its

*This R&D project is supported by ALMERCO Inc. and the Natural Sciences and Engineering Research Council of Canada.

†Part of this work has been presented at SPAA'96 [26] as a research summary.

neighbourhood based on the attributes stored in the cell and the cells in its neighbourhood.

Neighbourhood modelling applications fall into three main categories: neighbourhood analysis, cellular automata and propagation. NEMO is not tailored to any specific application. Rather, it is designed to support applications falling under the umbrella of these three raster neighbourhood modelling categories. NEMO encompasses these categories through corresponding neighbourhood modelling drivers (Section 2.1.1): Neighbourhood Analysis Driver(ND), Cellular Automata Driver(CD) and Propagation Driver(PD). Two additional client-server components (Section 2.1.2): Display Manager(DM) and Raster Database Manager(RDM) handle all data I/O and visualization issues. A communication layer is used to connect the host to the processors of the parallel machine (see Figure 1).

Our model of computation is Multiple Instruction Multiple Data (MIMD) using a distributed memory architecture. Our present implementation is on an AVX II manufactured by ALEX Informatique, Canada.

Since the display and I/O components are designed as separate modules, applications are portable to different architectures. In addition, the communication layer can be configured in a number of different ways. Currently, NEMO uses a *mesh* configuration to interconnect the processors since it is more natural for the three categories of neighbourhood modelling considered.

2.1 System Components Overview

Although the three application drivers differ in functionality, the principles under which they operate are similar. The drivers accept as input: one or more raster images (where each raster image may have one or more attributes associated with each cell), a user-defined neighbourhood function, and a local neighbourhood definition. Using the client-server components, the application drivers load the data into the parallel machine (tiling the data as necessary), activate user-defined neighbourhood functions on each raster-cell, output the results to the database and/or display the resulting data.

The main difference between the three drivers is the order in which the cells are processed; the implications of this for the implementation are substantial. The cellular automata and neighbourhood analysis drivers process the cells in an arbitrary order whereas the propagation driver processes the cells in a user-specified order (defined at run time). For instance, consider a neighbourhood function $N(g_{ij})$ that operates on a raster cell g_{ij}. The differences between the three families can be illustrated via pseudo code:

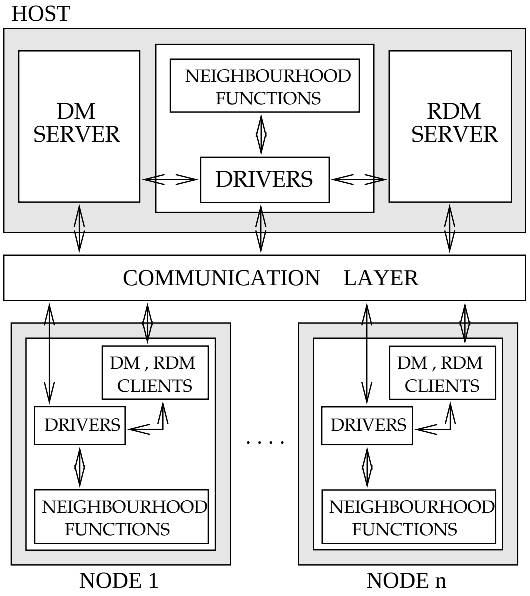

Figure 1: NEMO system overview.

Neighbourhood Analysis
FOR (each cell g_{ij} - arbitrary order) DO
    $N(g_{ij})$

Cellular Automata
FOR (each generation - sequentially) DO
    FOR (each cell g_{ij} - arbitrary order) DO
        $N(g_{ij})$

Propagation
FOR (next cell g_{ij} in priority queue) DO
    $N(g_{ij})$

The two client-server components communicate with the user's application via the drivers. The parallel machine processors act as clients from which raster data can be requested, displayed and/or stored. The DM allows graphical data to be displayed from the application, and also provides a dynamic zooming and scrolling interface to the displayed data. The RDM allows reading and writing of raster data that reside on the host machine. All interaction (I/O requests) between the host machine and the parallel machine takes place through the communication network (using message passing).

2.1.1 Neighbourhood Modelling Drivers

Neighbourhood Analysis Driver (ND) - Neighbourhood analysis involves executing one, or a series of distinct, single pass neighbourhood functions on a raster. It encompasses neighborhood modelling as a whole. Applications in this category which we have implemented include several image processing operations and cartographic modelling using Tomlin's "Map Algebra" [45, 9, 35]. Li [32] discusses issues regarding the implementation of map algebra in a data parallel programming language. Three general families of neighbourhood operations are identified in the ND design, differing primarily in the amount of intra-raster communication required:

- Point-wise operations - cell by cell combination of rasters (e.g., map overlay) which requires no communication between processors.
- Local neighbourhood operations - transformation of each cell according to some function of its neighboring cells' values. It requires limited communication, typically between neighbouring processors (e.g., image processing operations such as noise reduction and edge enhancement).
- Global operations - computations involving an arbitrary number of cells of a raster which require extensive data communication (e.g., calculating aggregate statistics for a raster such as a colour histogram). These operations provide the most interesting challenges to transparent parallelism.

Cellular Automata Driver (CD) - The CD is a general tool supporting the development of cellular automata (CA) applications. The CD processes all cells (with their respective neigbourhoods) in a raster one at a time in arbitrary order. Each cell in a cellular automaton can be in one of several predefined states. When processed, a cell may change its state depending on its current state and the state of its neighbouring cells. The CD processes the raster(s) one *generation* at a time; a generation is the collection of cells at some discrete time. The next generation is obtained by processing each cell of the current generation once. After a cell has been processed its new state is saved for the next generation. Upon completion of a generation the CD starts to process the next generation. The CD stops when the desired number of generations is reached.

CA were introduced by Codd [12] (made famous through Conway's "Game of Life" [21]) as an elegant mathematical model for a class of processes operating in discrete time and space. GIS modelling/simulation using CA has been described for forest growth [30] and dieback [29], forest fires [2, 11], forest infestation [24], earthquakes [1, 4] [33], and avalanches [39]. Bonfatti et al. show a CA that models the propagation of tides over a lagoon [7]. Itami [27] has studied CA for residential site selection (using Tomlin's map analysis package). More recently [28], he has studied the incorporation of CA into a GIS system, including Tomlin's map algebra. Batty and Xie [3] use CA to model urban growth and form. Brinch Hansen [8] describes a model program for parallel execution of cellular automata adapted to a simple forest fire model.

Tobler[44] introduces the notion of *cellular geography* in which he classifies different CA models covering a wide range of applications and generalizations. The CD is intended for general cellular automata GIS applications. It supports all of the models proposed by Tobler.

Propagation Driver (PD) - The propagation model is designed to process applications which operate on the active border principle in which only a subset of cells need be processed at a given time. Although propagation can be emulated with cellular automata and vice versa, it is better to separate the two families to increase performance. For example, if propagation is emulated with cellular automata, then it would take a full generation (i.e., complete processing of the raster) in order to advance the active border. In the PD, instead of modifying the attributes of the center cell g_{ij} in the local neighbourhood (as in the CD), the center cell g_{ij} may affect the attributes of its neighbours.

For example, in a forest fire only the areas near the fire front (i.e., the active border) are of interest and must be processed. All other areas do not require processing at this moment. Since only a portion of the raster is active at any given time, it is more efficient. Other applications include producing cost surfaces or calculating propagation functions such as noise propagation [41].

2.1.2 Client Server Components

The client-server components of NEMO take care of I/O communication aspects between the host computer and the internal processors. To retrieve/output data or to display a raster, each processor operates, as though it was a sequential machine and not a processor in a parallel environment. Each component has two subcomponents: a communication component allowing the parallel processors to transparently interact with the host and a set of library functions executing the requests issued by the processors.

Display Manager (DM) - The DM controls all aspects of displaying raster images and application results. It provides a flexible environment for data display while freeing the user from having to handle any of the issues that relate to data display (e.g., scaling, clipping and geo-referencing). It provides dynamic viewing of application data by allowing continuous updating of raster images on the screen. The DM also provides synchronization mechanisms that allow data to be collected before it is displayed.

Raster Database Manager (RDM) - The RDM manages the raster database by servicing all I/O requests. The RDM provides a client server environment where the processors are the clients and the RDM is the server. It controls and manipulates a central database which, at present, is stored on the host system on one or more disks. The nature of NEMO has allowed us to choose a "shared external memory" model over "distributed memory model" in order to avoid data and cache coherence problems and to avoid common deadlock situations.

2.2 Parallel Issues

While parallel computers are becoming widely available, the development and implementation of (complex) algorithmic techniques to exploit the features of parallel architectures is very challenging. Parallel algorithms present a host of issues that are independent of neighbourhood modelling, which must be addressed in order to efficiently implement a parallel neighbourhood modelling system. NEMO's primary objective of providing transparent parallelism relieves the burden of these parallel issues from the user. The main issues are as follows:

- Data I/O
- Data Visualization
- Architecture and Machine Independence
- Communication Bottlenecks
- Causality Errors
- Load Balancing
- Data Coherence
- Deadlock and Simulation Termination

Efficient handling of these issues requires knowledge of parallel machines and algorithms. It would only hinder the efforts of a GIS application designer if he/she had to deal with them. Thus, NEMO handles most of these issues in a manner which is independent of the application itself (provided that the application falls under one or more of the three modelling families mentioned earlier). As a result, the user works with a system that "acts" like a sequential GIS, with the exception that applications run faster. NEMO is able to hide all parallel details through the use of the application drivers and the client-server components. Due to space constraints, we will discuss two techniques that we use to help alleviate parallel issues of load balancing, causality errors and data I/O.

2.2.1 Causality Errors and Load Balancing

In this section we describe our solution to causality errors, load balancing and visualization that arise during parallelization. Causality errors arise when a processor in a parallel simulation does not wait for up-to-date input but rather processes the data and advances its local simulation time ahead of the global minimum simulation time. Due to changes in the data it must backtrack from its current simulation time and repeat the computation. Causality errors are bound to happen in the PD due to its operational nature - the shortest path principal. For example: two forest fires may occur simultaneously in two different locations - fire A spreads at a speed of 50km/h while fire B spreads at 10km/h. If two processors process different forest fires without knowledge of each other, then the processor which processes fire B will have to repeat its computation when it receives the data of fire A. Broadcasting the global simulation time in the network is not viable since it would reduce the performance of the parallel computer to that of a sequential machine, or worse.

Since each processor receives different portions of the raster, the amount of computation will vary depending on the data. For example, in fire propagation, a section of data representing forested regions would require more computation than regions containing water. Also, in many PD and CD applications the operations on the data effect only a relatively small number of cells which may be clustered (e.g., in a forest fire the activity will occur only on the boundary of the fire). Load balancing is required to ensure that the computation is evenly distributed among processors.

Another key issue is the animation of the simulation. In many cellular automata and propagation modelling applications the progress of the simulation is just as important as the final output (result). For example, in a forest fire, we are not only interested in determining what areas have burnt but also in the spreading behaviour of the fire.

We have overcome these issues by introducing a folding technique. The folding is an efficient mapping of spatial data to processors. For 2D data, the folding divides the raster into many small tiles, but instead of mapping the tiles to the processors cyclically, it maps the tiles to the processors by folding the raster like an accordion in the X and Y directions (see Figure 2). The 2D folding has the following advantages:

- **Reduction of Causality Errors -** the 2D folding significantly reduces the number of causality errors that occur in the system while allowing the simulation to progress in parallel. As a result of the 2D folding, each processor in the parallel computer has different sections (pieces) of the spatial data. Thus, as the simulation progresses, each processor will have several active sections (sections containing cells that must be processed) from different locations. It can then compare the simulation time among all the sections and process the cells with the minimum simulation time. This allows the PD to implicitly broadcast the simulation time among the processors without sending messages (allowing each processor to have a current simulation time very close to the global simulation time).
- **Load Balancing -** In many PD and CD applications the active cells (cells to be processed) are not uniformly distributed throughout the raster. This results in areas which have clusters of active cells. The finer partitioning of the 2D folding allows the clustered data to be redistributed among all processors. Although raster partitioning into smaller sections is a common technique in parallel spatial processing, the 2D folding provides us with the additional advantage of reducing the distance that messages must travel in the system. The communication pattern of NEMO is mainly among neighbouring processors. The 2D folding ensures that processors containing adjacent sections of the raster are also adjacent in the parallel computer. This reduces the distance that a message must travel in the system to one link.

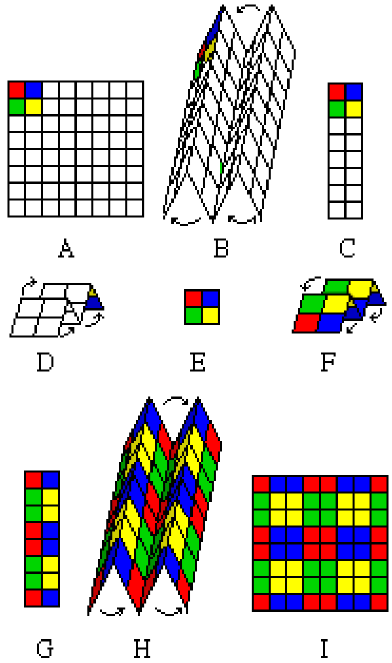

Figure 2: A folding sequence showing how a raster is mapped to 4 processors. The raster is shown as a grid of 8x8 blocks in A. B to E show how the raster is folded onto the 4 processors. F-H shows the unfolding back to the 8x8 grid. I shows the final mapping of the blocks.

2.2.2 Data I/O and Visualization

In many GIS applications, rasters have a very large amount of data which cannot be kept completely in memory. This may cause a problem when data is to be displayed in a dynamic fashion (such as allowing zooming or scrolling). The host machine of NEMO only stores and displays the raster data that fits in the window. Since there may not be enough memory to hold the entire raster on the host machine, we distribute the raster data and store it among the processors of the parallel machine. Whenever the raster data is to be displayed, each processor sends its portion to the host machine for display. Due to the large amount of raster data, each processor clips and samples the data before sending it to the host. This requires each processor to have the dimension, offset and zoom ratio of each window. In order to provide a transparent interface to the windowing system, NEMO creates an additional "helper" process on each processor which handles communication with the host machine for all display requests. When windows are opened by an application, the appropriate data (i.e., dimension etc.) is sent to the helper process on each processor for registration. Each helper process keeps a buffer containing the most recently displayed data. When a resize, zoom or scroll event occurs, the DM requests the last displayed data from each of the processors which send the appropriately clipped and sampled data back to the host for a display update. An interesting effect of this technique is that data is displayed in pieces, which is analogous to filling in pieces of a jigsaw puzzle.

3 Applications on Top of NEMO

As we have shown in previous sections the NEMO system itself is not an application but rather an environment for developing neighbourhood related applications. A large variety of applications can be developed and executed on top of NEMO (each of the drivers is designed as a development tool for a range of neighbourhood models). We have developed the following applications:

- Image processing applications (including edge detection and skeleton finding)
- Cartographic modelling using map algebra (see [16])
- Forest fire modelling (see Figure 6,7)
- Earthquake modelling
- Response time modelling (see Figure 3)
- Generation of weighted cost surfaces (see Figure 5)
- Ice tracking (see Figure 8)

Due to space constraints we restrict ourselves to discussing a simple example of an emergency response time modelling application. We show that the amount of work that the user must do in order to create a new application on top of NEMO is minimal and requires no special knowledge of parallelism.

3.1 Emergency Response Time Modelling

This application computes the area (city region) which can be reached by emergency vehicles (e.g., fire department) within a predetermined maximum allowable response time. For simplicity the problem can be described as follows: Given the locations of the emergency stations, find: for each station the area that it must service in case of an emergency; the expected arrival time for each region in the city; areas that cannot be covered by any of the selected emergency stations within the response time.

The emergency vehicles in this example can travel at a speed of 80 km/h on main roads and at a speed of 15 km/h off the main roads. In addition to the speed, each station can be assigned a weight factor which reduces or increases the travel time. The maximum allowable response time is 10 minutes. The input rasters are a road

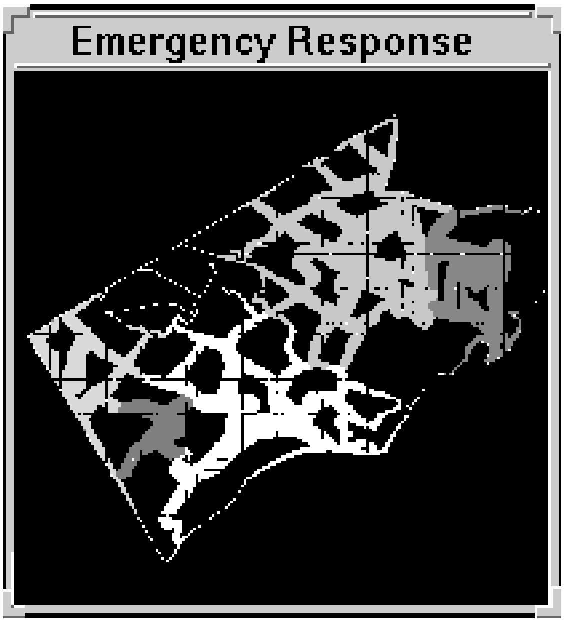

Figure 3: Snapshot of the emergency response time modelling simulation. The regions that can be reached from 5 stations within the allowed response time are shown in different shades of grey.

map of the city and a digital elevation model map. The output is a raster with two attributes per cell. The first attribute is the id of the station that should respond to any emergency call within the area and the second is the travel time to this location. Figure 3 shows a screen snapshot for this example at an intermediate stage in which the travel time to certain areas from five emergency stations has already been computed.

3.2 Performance Results

Figure 4 shows the speed-up (measured as (time for 1 processor) / (time for p processors)) results for the ERTM example for 5 stations using 3 rasters of size 1472 x 1472 for a total of 23MB of data. The graph shows that the ERTM application achieves very good speed-up.

The efficiency of the applications that we have developed ranges from 50% to 90% of the optimal for all ranges of processors measured. Since there is a direct link between the speed-up and the amount of time it takes to process a single cell (depending on its neighbourhood), the nature of the application is the most important factor affecting performance. Since NEMO is a general purpose tool and is not geared towards any application in particular, all fine tuning is dynamic with

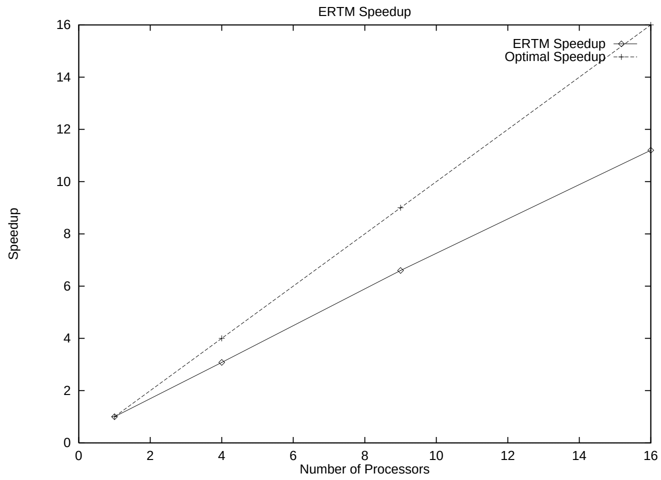

Figure 4: Speed-up results of the emergency response time modelling example.

respect to the application. We are further continuing to explore this dynamic relationship between computation and communication which will allow NEMO to efficiently execute a wide variety of applications.

3.3 Writing a New NEMO Application

NEMO is designed for general purpose use and to accommodate new neighbourhood applications easily. To meet our main objective, transparent parallelism, and to allow the user to concentrate on his/her application, the system requires the user to provide only a handful of functions that will be linked to NEMO. For ERTM, which is an application built on the propagation driver, the following set of five functions is required:

- **Modelling Function -** This is the core function of the neighbourhood model. Here the behavior and the nature of the model are defined. This function determines the direction of spread, its rate and the effect of the cell on its neighbouring cells. Most of the user's effort will be incorporated in this function. In the ERTM application when this function is activated, it determines for each of the neighbouring cells, the arrival time of the rescue team. The function calculates the additional travel time by first computing the additional distance (between the center cell and its neighbour) based on the DEM map and then computing the time needed to cover this distance (on or off road travel). If the rescue team can reach the neighbouring cell faster than the previously stored time, the function replaces the arrival time and the servicing station and notifies the PD about the changes.
- **Initialization Function -** This is the first function which is called by the propagation driver. The user, via this function, provides the PD with all the information it needs to start the modelling. Examples are the neighbourhood size, the rasters needed, the neighbourhood function, the cell and time comparison functions and when to stop the simulation. In addition to the information required by the driver, the user can initialize his/her own private data structures which are needed for the application and then inform the PD about their existence.
- **Simulation Time Comparison Function -** The PD processes the cells in a prioritized order, such that the cell with the minimum time is the next one to be processed. Each of the cells is assigned a "time" which is defined by the application. Thus, in order to determine the processing order of the cells the PD must be able to compare the "time" associated with the cells. Since the "time" is known only to the application, this function is required to compare the simulation time. In the case of the ERTM example, the "time" consists of two keys: the travel time it takes to reach the cell (primary key) and the id of the station (secondary key).
- **Cell Comparison Function -** This function is required in order to allow the PD to operate in a parallel environment. The guiding principle behind the PD is the priority queue concept. Namely, the cell with the minimum time is processed next. However, since the PD operates in an asynchronous mode it may happen that a cell A is a neighbour of two other cells which are processed by two different processors. Hence, the PD must determine which of the two copies of A contains the minimum time. To do so, the PD requires that the application, which has the notion of "time" and the content of the cell, will compare the two cells.
- **Termination Function -** This function is called by the PD when the simulation is complete. It allows the application to clean up (i.e., to close any open files free the memory etc.).

Four out of the five above functions are required in any general purpose propagation tool. The only additional function is the cell comparison function which is needed as a result of parallelism constraints. Although the user must write this additional function, all notion of parallelism and the parallelism issues are hidden from the application. This allows the user to concentrate on his/her model and improve the quality of the model instead of "wasting" his/her valuable time on the parallelization of the model (since all parallelization tasks and other inherent issues are being dealt with by NEMO).

4 Conclusion and Current Research Activities

The NEMO system described in this paper is a parallel GIS for raster-based neighbourhood modelling. NEMO enables the scientific community to harness the power of parallel computers while avoiding the issues inherent to parallelism. NEMO has been delivered to our industrial partner, ALMERCO Inc., for subsequent commercial development.

In the process of implementing NEMO, we developed several tools including a graphical performance analyzer [34] and a simulator for the parallel AVX machine [40] Currently we are developing two general systems for modelling and processing of spatial data:

- **ATLANTIS -** a general purpose parallel vector based GIS.
- **Parallel Spatial Modelling Environment -** an integrated open parallel environment for processing, manipulating, storing and visualizing of spatial data. The target domains for the system include areas where spatial data arise e.g., GIS, remote sensing, image processing, medical computing, and robotics.

Acknowledgments
The authors of this paper are indebted to K. Clarke, now U. C. at Santa Barbara, and Mike Wotton, Forestry Canada for providing insights into and understanding of the two fire models developed by them. We thank W. Bitterlich now with E.S.R.I. for interesting discussions and ALEX Informatique, Canada for technical support. Finally, we thank the other members of the PARADIGM group at Carleton University for their support and for many useful discussions.

References

[1] P. Bak, C. Tang, *Earthquakes as a self-organized critical phenomenon*, J. Geophys. Res. 94, 1989, pp. 15635-15637.

[2] P. Bak, K. Chen, *A forest-fire model and some thoughts on turbulence*, Phys. Lett. A 147(5-6), 1990, pp. 297-299.

[3] M. Batty, Y. Xie, *From Cells to Cities*, Environment and Planning B, Planning and Design, Vol. 21, 1993, pp. S31-S48.

[4] Y. Ben-Zion, J.R. Rice, *Earthquake Failure Sequences Along a Cellular Fault Zone in a Three-Dimensional Elastic Solid Containing Asperity and Nonasperity Regions*, Journal of Geophysical Research, Vol. 98, No. B8, 1993, pp. 14109-14131.

[5] M. Bern, D. Eppstein and S.-H. Teng, *Parallel construction of quadtrees and quality triangulations*, 1993.

[6] G.E. Blelloch, J.J. Little, *Parallel solutions to geometric problems on the scan model of computation*, Proc. 1988 int. Conference on Parallel Processing, pp. 218-222, 1988.

[7] F. Bonfatti, G. Gadda, P.D. Monari, *Simulation of Dynamic Phenomena by Cellular Automata*, Computers and Graphics, Vol. 18, No. 6, 1994, pp. 831-836.

[8] P. Brinch Hansen, *Parallel Cellular automata: A model for computational science*, Concurrency: Practice and Experience 5(5), 1993, pp. 425-448.

[9] K.K.L. Chan, C.C. Tomlin, *Map algebra as a spatial language*, In: D.M. Mark and A.U. Frank (Editors), Cognitive and Linguistic Aspects of Geographic Space, Kluwer Academic Publishers, Dordrecht, Netherlands, 1991, pp. 351-360.

[10] G. Cheng, C. Faigle, G.C. Fox, W. Furmanski, B. Li, and K. Mills, *Exploring AVS for HPDC Software Intergration: Case Studies Towards Parallel Support for GIS*, AVS Conference AVS'93, Lake Buena Vista, FL, May 24-26, 1993.

[11] K. Clarke, J.A. Brass and P.J. Riggan *A Cellular Automaton Model of Wildfire Propagation and Extinction*, Photogrammetric Engineering and Remote Sensing, Vol. 60, No 11, 1994, pp. 1355-1367.

[12] E.F. Codd, *Cellular Automata*, Academic Press, New York, 1968.

[13] P.J. Densham, M.P. Armstrong, *A Heterogeneous Processing Approach to Spatial Decision Support Systems*, Proc. Sixth Int. Symp. on Spatial Data Handling: Advances in GIS research, 1994, pp. 29-45.

[14] G. Vezina, G. Ratzer, V.V. Dongen and D. Poussart, *Parallel Spatial Analysis and Interactive Visualization Software*, Canadian Conference on GIS, Ottawa, 1994.

[15] L. De Floriani, C. Montani and R. Scopigno, *Parallelizing visibility computations on triangulated terrains*, Int. J. Geographical Information Systems, 1994, Vol. 8, No. 6, 515–531.

[16] D. Dubrule, P. Morin, J.-R. Sack, *A Parallel Cartographic Modelling System: Design Implementation and Performance*, to appear in Proc. GIS'97, Vancouver, 1997.

[17] *Parallel Architecture Laboratory for Geographical Information Systems*, GIS Group at the University of Edinburgh, Report, 1990.

[18] N.L. Faust, W.H. Anderson, and J.L. Star, *Geographic Information Systems and Remote Sensing Future Computing Environment*, Photogrammetric Engineering & Remote Sensing 57(6), 1991, pp. 655-668.

[19] W.R. Franklin, M. Kankanhalli, D. Sun, and P.Y. Wu, *Uniform grids: a technique for detection of line intersection on serial and parallel machines*, Proc. 9th Int. Symp. on Computer-Assisted Cartography AUTOCARTO 9, Baltimore, pp. 100-109, 1989

[20] R. M. Fujimoto, *Parallel Discrete Event Simulation*, Communication of the ACM, Vol. 33, No. 10, 1990, pp. 30-53.

[21] M. Gardner, *The fantastic combinations of John Conway's solitaire game "Life"*, Sci. Am. 223(10), 1970, pp. 120-123.

[22] B.L. Hickmanm M.P. Bishop, and M.V. Rescigno, *Advanced Computational Methods for Spatial Information Extraction*, Computers & Geosciences 21(1), 1995, pp. 153-173.

[23] S.Hopkins, R.G. Healey, and T.C. Waugh, *Algorithm scalability for line intersection detection*, Proc. 5th International Symp. on Spatial Data Handling, Charleston, pp. 210-218, 1992.

[24] F.C. Hoppensteadt, *Mathematical aspects of population biology*, in L.A. Steen, ed., Mathematics Today: Twelve Informal Essays, Springer Verlag, New York, 1978, pp. 297-320.

[25] Y. Hund and A. Rosenfeld, *Parallel processing of linear quadtrees on a mesh-connected computer*, Journal of Parallel and Distributed Computing 7, pp. 1-27, 1089.

[26] D. Hutchinson, L. Küttner, M. Lanthier, A. Maheshwari, D. Nussbaum, D. Roytenberg, J.-R. Sack, *Parallel Neighbourhood Modeling*, Proc. SPAA '96, Padua, Italy, June 1996, pp. 204-207.

[27] R.M. Itami, *Cellular Automatons as a framework for dynamic simulations in Geographic Information Systems*, Proc. GIS/LIS'88, Vol. 2, 1988, pp. 590-597.

[28] R.M. Itami, *Simulating Spatial Dynamics: Cellular Automata Theory*, Landscape and Urban Planning, Vol. 30, 1994, pp. 27-47.

[29] F. Jeltsch, C. Wissel, *Modelling Dieback Phenomena in Natural Forests*, Ecological Modeling, 75/76, 1994, pp. 111-121.

[30] Kazunori Sato, Yoh Iwasa, *Modeling of Wave Regeneration in Subalpine Abies Forests: Population Dynamics with Spatial Structure*, Ecology, Vol. 74(5), 1993, pp. 1538-1550.

[31] T. Kreitzberg, *A parallel toolset for intervisibility*, Transputer Research and Applications 6, ed. A.S. Wagner, IOS Press, pp. 12-20, 1993.

[32] B. Li, *Opportunities and challenges of parallel spatial data analysis: Initial experiments with data parallel map analysis*, GIS/LIS '92 Annual Conf. and Exposition, San Jose, pp. 445-458, 1992.

[33] J. Lomnitz-Adler, *Automaton Models of Seismic Fracture: Constraints Imposed by the Magnitude-Frequency Relation*, Journal of Geophysical Research, Vol. 98, No. B10, 1993, pp. 17745-17756.

[34] C. MacDonald, *NodeView: A Profiler/debugger for Parallel Computers Using Message Passing*, MCS Thesis, Carleton University, Ottawa, Canada, May 1996.

[35] B. Mills, *Map Algebra: an emerging standard for analysis in GIS*, GIS Europe, November 1994, pp. 18-20.

[36] M. Micklefield, *The applications of massively parallel processing to geographic information systems*, Parallel Computing and Transputer Applications: Part II, eds. M.Valero, E. Oñate, M. Jane, J.L. Larriba, B. Suárez, IOS Press, Amsterdam, pp. 1206-1211, 1993.

[37] J.E. Mower, *Building a GIS for parallel computing environments*, Proc. 5th International Symp. on Spatial Data Handling, Charleston, pp. 219-229, 1992.

[38] P.W. Newton, P.R. Zwart and M.E. Cavill (eds.) *Networking Spatial Information Systems*, Belhaven Press, London and New York, 1992.

[39] V. Romer-Rochin, J. Lomnitz-Adler, E. Morales-Gamboa, R. Peralta-Fabi, *Avalanches in a Cellular Automaton*, Physical Review E., Vol. 51, No. 5, May 1995, pp. 3968-3976.

[40] D. Roytenberg, J.-R. Sack, *A Simulator for the Alex AVX Series II Parallel Computer*, Proc. of the 10th Annual International Symposium on High Performance Computers, June 1996.

[41] J. Strobel, *Conceptual Modeling of Spatial Diffusion Problems - Lessons from Noise Propagation Analysis*, GISDATA Specialist Meeting in GIS and Spatial Models, Stockholm, June 14-18, 1995.

[42] T.K. Peucker and D.H. Douglas, *Detection of surface specific points by local parallel processing of discrete elevation data*, Computer Graphics and Image Processing 4, pp. 357-387, 1975.

[43] F.Q. Stout, *Supporting divide-and-conquer algorithms for image processing*, Journal of Parallel and Distributed Computing 4, pp. 95-115.

[44] W.R. Tobler, *Philosophy in Geography*, edited by S. Gale and G. Olssen Issues for Design and Implementation, Cellular Geography, 1979.

[45] C.D. Tomlin, *Geographic Information Systems and Cartographic Modeling*, Prentice Hall, 1990.

[46] J. Vaughan, D. Whyatt and G. Brookes, *A parallel implementation of the Douglas-Peucker line simplification algorithm*, Software - Practice and Experience 21:3, pp. 331-336, 1991.

[47] T.C. Waugh and S. Hopkins, *An algorithm for polygon overlay using cooperative parallel processing*, Int. J. Geographical Information System 6:6, pp. 457-467, 1992.

Figure 5: Screen snapshot of the shopping center attractiveness application.

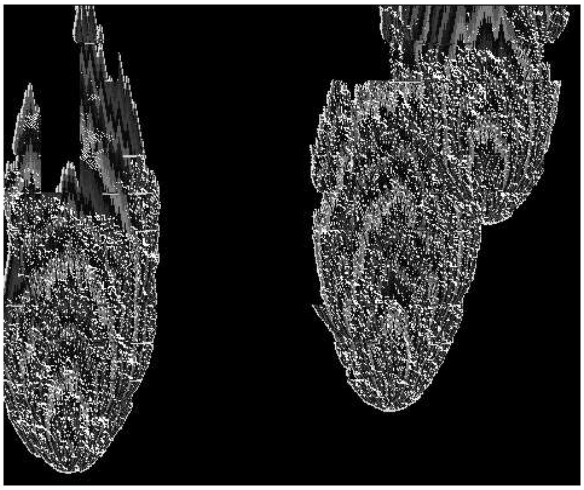

Figure 6: Screen snapshot of the parallel fire behavior prediction application which was developed in colaboration with Forestry Canada.

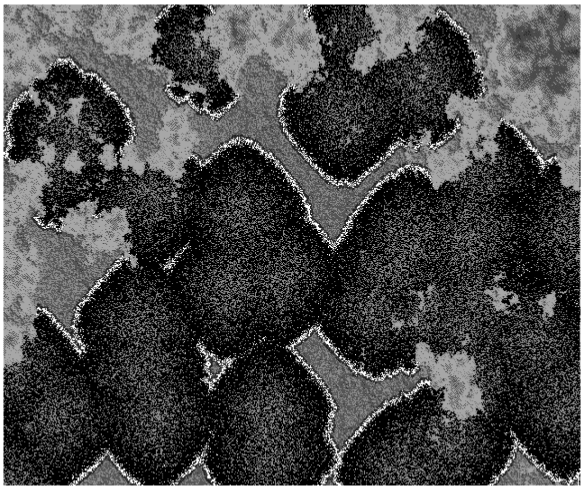

Figure 7: Screen snapshot of the Brinch Hansen fire modelling application.

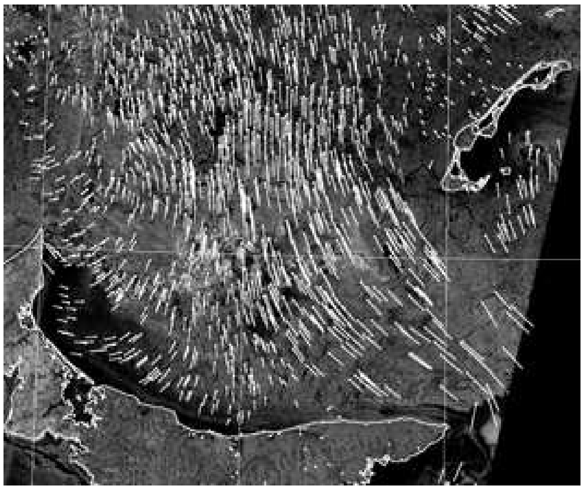

Figure 8: Screen snapshot of the parallel ice tracking application. (The image originated from the Canadian Space Agency.)