

Distributed Simulation of Spatially Explicit Ecological Models

Kevin Glass^{*}, Marilyn Livingston, and John Conery

Computational Science Institute

University of Oregon

Eugene, OR 97403

email: kglass, mlivings, conery @cs.uoregon.edu

Abstract

Large-scale ecological simulations are natural candidates for distributed discrete event simulation. In optimistic simulation of spatially explicit models, a difficult problem arises when individuals migrate between physical regions simulated by different logical processes. We present a solution to this problem that uses shared object states. Shared states allow for efficient communication between LPs and for early detection of canceled events. We briefly describe an optimistic simulation environment called EcoKit, which operates on top of the WarpKit implementation of Time Warp. Our experiments with this system on a shared memory multiprocessor show that EcoKit promises to scale well both with the number of processors and the number of individuals simulated.

Keywords: Distributed simulation, ecological models, individual-based models, optimistic simulation.

1. Introduction

In an effort to construct more realistic and detailed descriptions of plant and animal populations, ecologists are increasingly turning to individual-based models [1][3]. Rather than summarizing average case behavior with systems of differential equations, an individual based model uses a representation of each individual plant or animal. A spatially explicit ecological model is one that is based on an accurate representation of the physical territory the individuals inhabit. The model includes interactions between the environment and the individuals as well as among the individuals themselves.

^{*} Kevin Glass is supported by a US Department of Energy Computational Science Graduate Fellowship.

Discrete event simulation (DES) is a natural method for implementing individual-based models. Since realistic ecological models are highly complex and may involve many thousands of individuals in territories of several hundred square kilometers, distributed simulations running on high performance parallel machines will be an important enabling technology in this field.

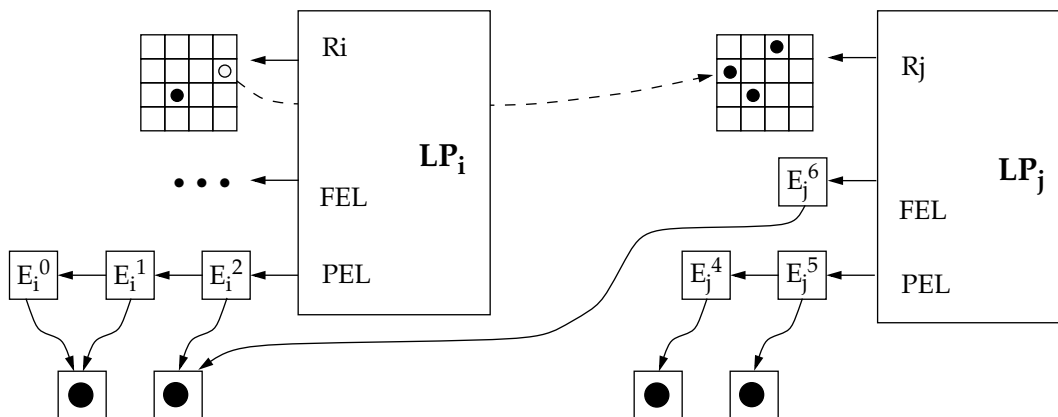
In this paper we describe the structure and performance of an optimistic distributed simulation system for spatially explicit ecological modeling. Our simulation engine, which we call EcoKit, operates on top of WarpKit, a shared memory implementation of the Time Warp Operating System [2]. The next section describes a central problem of implementing distributed simulations of spatially explicit models. Following that we describe our solution and some performance data.

2. Spatially Explicit Models

In a spatially explicit model the representation of the physical territory over which the individuals interact could be as simple as an array of undifferentiated cells that serve as little more than place holders for the individuals being modeled, or as complex as collections of diverse cells that contain information about temperature, elevation, vegetation, and type of terrain. If the environment can change over time, the simulation will contain two types of mutable objects: objects that represent individuals, and objects that represent cells.

A straightforward approach to distributed simulation is to partition the physical territory into discrete regions, each of which is assigned to a single logical process (LP). For example, if a simulation used $10\text{m} \times 10\text{m}$ cells and the simulated region is $1\text{km} \times 1\text{km}$, there will be a total of 10,000 cells, and an 8-processor simulation could allocate regions of 1250 cells to each LP.

Figure 1. An LP has a pointer (R) to the region it simulates in addition to future and past event lists. In this example event E_i^2 causes a mouse to move to LP_j , where its arrival is event E_j^6 . Both events share the object that represents the state of the mouse (its location, whether it is migrating or nesting, etc.).



When an individual moves from one region to another, there is a considerable amount of state information that must be passed or shared between the LPs. In an optimistic distributed simulation, this problem is exacerbated by the necessity to maintain state information to handle potential rollbacks. Clearly the old states of an individual have to be retained, but also previous states of cells have to be stored. For example, if an event that moves an animal from LP_i to LP_j is rolled back, cells modified by the animal while it is in LP_j have to be restored when the animal is returned.

In most simulation models that have been successfully parallelized by the Time Warp approach, the objects passed between logical processes carry little or no state information. Thus they contribute little extra information to the state of the logical processes, and techniques like incremental state saving can effectively manage the amount of state saved and later restored by a rollback. In spatially explicit ecological models, however, we expect there will be significant interactions between the LPs and the objects that occupy the space managed by the LPs.

Two methods introduced previously to deal with these types of complex interactions are state-copying and space-time memory. Space-time memory, presented by Ghosh and Fujimoto [6], was designed as an abstract representation of spatial cells that change state over time. The sharks world simulation used as an example in that paper is similar to the one we describe below, except the individuals in that model — the sharks — did not change state as the simulation progressed. It should be possible to implement a spatially explicit model with space-time memory, for example by using two object memories, one for spatial cells and one for individuals, but this would lead to more overhead to process each cell or individual update. By sep-

arating the representation of individuals from the representation of the spatial cell they occupy, and by associating events with a single spatial cell, we are able to implement object updates with fewer locks.

The state copying approach was used by Deelman et al in their distributed memory models of the spread of Lyme disease [4][5]. In their system, when an individual moves between the region controlled by LP_i to the region of LP_j , the state of the individual is copied. The old state of the object is kept in a “ghost list” on LP_i . This allows each LP to access the object without locks, and it leads to an efficient rollback mechanism since an object does not have to be copied back. A drawback of this implementation is the amount of time it takes to send messages between LPs. If LP_i needs to roll back past the event that moved the individual, the delay in notifying LP_j means LP_j may process a large number of events that will now have to roll back.

3. Shared Object States

The event representation we developed takes advantage of a shared memory to allow for a tighter coordination of LPs and a more efficient transfer of individuals between LPs.

Instead of copying the states of individuals as they move between LPs, we maintain one global copy of each individual (Figure 1). Initially this individual is associated with the LP where it was created. When an event causes an individual to move to a new LP, the event inserted into the future event list (FEL) of the receiving LP simply contains a pointer to the individual’s state.

Object sharing introduces a major new complication: individuals can now be accessed by two or more LPs at a

#LPs	Net Events	Execution Time		#cells per LP	Rollbacks	
		Total (sec)	per event (μ sec)		Total	per LP
1	151351	34.74	229.5	20000		
2	152598	17.04	111.7	10000	809	405
3	145635	14.01	96.2	6667	1406	469
4	152397	12.90	84.6	5000	1766	441
5	154626	12.94	83.7	4000	2416	483
6	154843	12.84	82.9	3333	2887	481
7	164332	13.40	81.5	2857	3727	532

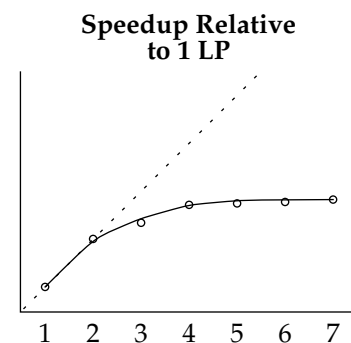


Figure 2. Data from simulations of 10,000 individuals in 20,000 cells.

time. For example, after individual x moves from LP_i to LP_j , LP_i might process a straggler that causes it to roll back to a time before the move. Now both LPs will think they “own” x , and they may simultaneously try to update x ’s state.

We store an individual’s previous states in a linked list associated with the individual itself. The entire list is protected by a lock. When an LP wants to update the state of an individual, either during normal execution or during a rollback, it must use the lock in order to gain exclusive access. When LP_i rolls back an individual to a time t , it will mark all states in the previous-state list that have a time stamp greater than t as invalid. Then when LP_j tries to update the object and sees these invalid-state flags, it aborts the processing the state transition and moves on to its next event. Since we are using a shared memory system, there is a minimal time delay between the time LP_i begins its rollback and the time when LP_j learns that it should no longer process events for the individual.

To test our shared object representation we used it in the implementation of a mouse migration model similar to that used by Deelman et al in their state-copying simulator [4]. Mice in this model alternate between nesting and migrating, and there is a constraint that no more than one mouse may nest in any given grid cell. The grid cells were divided evenly among the LPs, so that the number of cells per LP decreased as more LPs were used. The simulations were run on an SGI PowerChallenge with eight R8000 processors and 512MB RAM. In each case we ran one LP per processor and the data was collected during a period when the simulation was the only program running on the system.

A direct comparison of the execution times of shared objects with object-copying is difficult because the systems were implemented on different hardware (SGI vs. SP-2) and in different software environments (WarpKit vs. custom software). However, we can compare the scalability of the two different methods and their sensitivity to the number of rollbacks in the system.

Deelman et al reported near linear speedups for small simulations (800 mice in 2,400 cells), but when the model size was increased to 8,000 mice in 32,000 cells the overhead from rollbacks overwhelmed the system. In the larger simulations execution time on eight processors was the same as the execution time on one processor [5]. They reported that the cause of this slowdown is due to the number of rollbacks executed: the larger the portion of the region simulated by an LP, the more rollbacks we can expect because a straggler message arriving at an LP will cause all individuals in that part of the region to be rolled back. Deelman et al reported on several techniques that improved their basic simulation, including running more LPs per physical processor, so that each LP was responsible for a smaller region, and restricting the amount of virtual time they allowed each LP to get out of synch with the others.

Results from a set of executions of our shared object method on the SGI system are shown in Figure 2. We used a maximum of seven LPs on our eight CPU system since one CPU was used by the WarpKit kernel. The table lists the net events processed (the total number of events minus canceled and rolled back events) and the overall execution time. Since the simulation is stochastic, a different number of events are executed on each run. To make more reliable

comparisons, our speedup curve and other analyses are based on the average time per event, computed by dividing the execution time by the number of events. The numbers are means from several executions of a model with 10,000 mice in 20,000 grid cells.

The gains from parallelism appear to drop off at about four LPs, but this is simply due to the small amount of work being done in each event. If we compare the time per event measured in the parallel simulator running on one LP (229.5 μ sec) with the time per event on a sequential simulator using the same simulation model (177.4 μ sec) we can estimate the total overhead due to state saving, etc., to be about 50 μ sec. Since this overhead is likely to be constant for any number of LPs, the flattening out of the speedup curve when the time per event gets down to 80 μ sec is simply the expected diminishing returns predicted by Amdahl's law.

A second observation from this data concerns the processing of rollbacks. The table in Figure 2 shows that the total number of rollbacks increases with more LPs. This is to be expected, since the global region size and the number of individuals remain constant. We used the same method for partitioning regions as Deelman et al and divided the global region with vertical cuts. Thus the number of rollbacks in the system should be proportional to the number of interfaces, i.e. the number of LPs. The last column in the table confirms that the number of rollbacks per LP is fairly constant.

Data from this set of simulations and others we have done show that the number of rollbacks is not affected by region size. If it was, we would see a higher percentage of rollbacks in simulations with more cells per LP. We attribute this efficiency to the early notification of other LPs when an individual's current state is no longer valid. When an LP that used to own an individual rolls back to a state where that individual again resides in that LP's region, the LP marks all later states of the individual as invalid. Any LP that attempts to use that state immediately sees that it is invalid, and this prevents those other LPs from moving far ahead and doing work that will later have to be rolled back.

4. Summary and Future Work

Spatially explicit ecological simulations introduce a new dimension to distributed simulation because simulated objects may move from one logical process to another. In this paper we presented a method for reducing the impact of rollbacks caused by these movements. Our method separates the states of individual objects from the states of the physical region. As a consequence of our implementation in a shared memory environment, when an object is being rolled back, locks on the states involved prevent the LPs

from processing too many events only to have to roll them back later.

Our long-range goal is to build a problem solving environment for ecological modeling. An example of the type of model we wish to support is a study performed by the Biodiversity Research Consortium, which analyzed possible futures of a large region of undeveloped land near Camp Pendleton in Southern California [7]. This study combined analytical models of species behavior, vegetation growth, climate, hydrology, and other factors to predict the impacts of different urban growth scenarios on the diversity of wildlife in the region. More precise simulations of such complicated systems are likely going to be a combination of time-stepped "cellular automaton" simulations for those parts of the system that need periodic and regular updates, plus a higher level of distributed discrete event simulation for individual based modeling of large animals and other objects that are described by discrete functions.

5. References

- [1] Bart, J. Acceptance criteria for using individual-based models to make management decisions. *Ecological Applications* 5:2, 1996, pp. 411-420.
- [2] Cleary, J., Covington, A., Franks, S., Gomes, F., Tsai, J., Unger, B. and Zhonge, X. *WarpKit Programmers Manual - Draft 1.9.1*. Dept. of Computer Science, University of Calgary.
- [3] DeAngelis, D.L. and Gross, L.J. *Individual Based Models and Approaches in Ecology*. Chapman and Hall, 1992.
- [4] Deelman, E., Caraco, T., and Szymanski, B. K. Parallel discrete event simulation of Lyme disease. Proc. Pacific Symposium on Biocomputing (Hawaii, Jan. 3-6), 1996, pp. 191-202.
- [5] Deelman, E., Caraco, T., and Szymanski, B. K. Simulating Lyme disease using parallel discrete event simulation. Winter Simulation Conference (San Diego, Dec.), 1996.
- [6] Ghosh, K. and Fujimoto, R. Parallel discrete event simulation using space-time memory. *Int. Conf. Par. Proc.* 1991, Vol. III, pp. 201-208.
- [7] Steinetz, C. (ed.). *Biodiversity and Landscape Planning: Alternative Futures for the Region of Camp Pendleton, California*. Harvard Graduate School of Design, 1996.