A Parallel Structured Ecological Model for High End Shared Memory Computers

Dali Wang[1], Michael W. Berry[1], and Louis J. Gross[2]

[1] Department of Computer Science,
203 Claxton Complex,
University of Tennessee, Knoxville, TN 37996
{dwang,berry}@cs.utk.edu
[2] The Institute for Environmental Modeling,
569 Dabney Hall,
University of Tennessee, Knoxville, TN 37996
gross@tiem.utk.edu

Abstract. This paper presents a new approach to parallelize spatially-explicit structured ecological models. Previous investigations have mainly focused on the use of spatial decomposition for parallelization of these models. Here, we exploit the partitioning of species age structures (or layers) as part of an integrated ecosystem simulation on a high-end shared memory computer using OpenMP. As an example, we use a parallel spatially-explicit structured fish model (ALFISH) for regional ecosystem restoration to demonstrate the parallelization procedure and associated model performance evaluation. Identical simulation results, validated by a comparison with a sequential implementation, and impressive parallel model performance demonstrate that layer-wised partitioning offers advantages in parallelizing structured ecological models on high-end shared memory computers. The average execution time of the parallel ALFISH model, using 1 computational thread, is about 11 hours, while the execution of the parallel ALFISH model using 25 computational threads is about 39 minutes (the speedup factor being about 16).

1 Introduction

Recently, emphasis on integrated multi-component ecosystem modeling, involving complex interactions between some or all of the trophic layers, has increased, resulting in coupled ecosystem models with large numbers of state variables. Current efforts in integrated ecosystem modeling have led to the realization that model and software development utilizing only a single approach within a traditional computing framework can hinder innovation of complex highly-integrated model simulation involving diverse spatial, temporal and organismal scales. The Across Trophic Level System Simulation (ATLSS) [1] is an example of a new type of spatially-explicit ecosystem modeling package that utilizes different modeling approaches or ecological multimodeling [2]. In this paper, we focus on the parallelization of an age-structured population model, within the context of ATLSS, for freshwater fish functional groups (ALFISH) in South Florida.

ALFISH is an Intermediate Trophic Level Functional Groups model which includes two main subgroups (small planktivorous fish and large piscivirous fish), structured by size. In the complex integrated system of ATLSS, ALFISH is an important link to the higher-level landscape models, since it provides a food base for several wading bird models [3]. An objective of the ALFISH model is to compare, in a spatially explicit manner, the relative effects of alternative hydrologic scenarios on fresh-water fish densities across South Florida. Another objective is to provide a measure of dynamic, spatially-explicit food resources available to wading birds.

The ALFISH model has been developed in part to integrate with two wading bird models: a proxy individual-based wading bird (WB) model [4], and a Spatially-Explicit Species Index (SESI) wading bird model [5]. Major concerns associated with integrating ALFISH into the ATLSS architecture include its long runtime. The average runtime is 35 hours on a 400MHz (Ultra Sparc II-based) Sun Enterprise 4500 for a typical 31-year simulation. One objective of this parallelization is to speedup the ALFISH execution for its practical use within ATLSS by allowing real-time data generation and information sharing between multiple ATLSS models, including the individual-based wading bird model.

2 Parallelization Strategy

A successful approach to the parallelization of landscape based (spatially-explicit) fish models is spatial decomposition [6, 7]. For these cases, each processor only simulates the ecological behaviors of fish on a partial landscape. This approach is efficient in stand-alone fish simulations because the low movement capability of fish does not force large data movement between processors [8]. However, in an integrated simulation with an individual-based wading bird model, intensive data immigration across all processors is inevitable, since a birds flying distance may cover the whole landscape. In this paper, we present an age-structured decomposition (or layer-wised partition). As opposed to a spatial decomposition, this approach partitions the computational domain along the fishs age-structure, so that each processor computes the dynamics of fish at certain ages on the whole landscape. This parallel strategy is suitable for shared-memory computational platforms, where the behind-the-scenes data exchanges between processors have been highly optimized by the hardware and underlying operating system through related library routines and directives.

3 Computational Platform and Parallel Programming Model

The computational platform used in this research is a 256-processor SGI Altix system (Ram) at the Center for Computational Sciences (CCS) of Oak Ridge National Laboratory. Ram is unique in the CCS in that it has a very large, shared memory. Ram is comprised of 256 Intel Itanium2 processors running at

1.5 GHz, each with 6 MB of L3 cache, 256K of L2 cache, and 32K of L1 cache. Ram also has 8 GB of memory per processor for a total of 2 Terabytes of total system memory. This system has a theoretical total peak performance of 1.5 TeraFLOP/s. The operating system on Ram is a 64-bit version of Linux. The parallel programming model in this research is supported by the multithreaded application programming interface referred to as OpenMP [9].

4 Functionality and Parallel Implementation of ALFISH model

4.1 Model structure

The study area for ALFISH modeling contains 26 regions as determined by the South Florida Water Management Model [10]. A complete list of these regions is provided in Fig. 1.

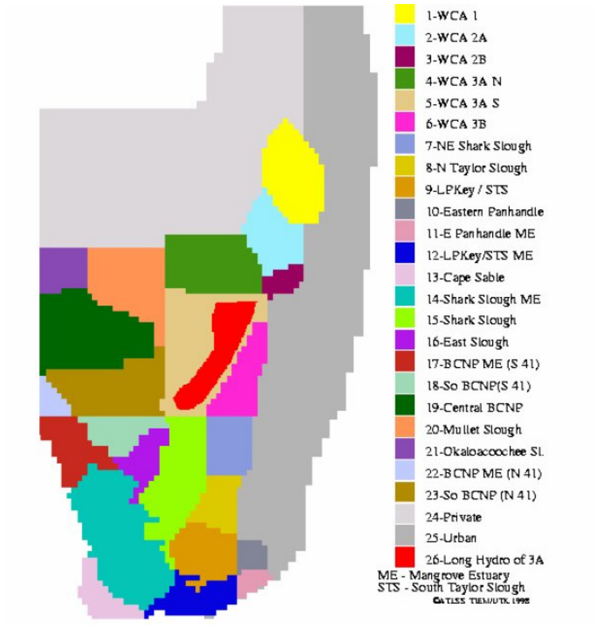

Fig. 1. Subregions used by the ALFISH model

The total area of Everglades modeled in ALFISH contains approximately 111,000 landscape cells, with each cell 500m on a side. Each landscape cell has two basic types of area: marsh and pond. The difference between marsh and pond areas is that the latter is always considered wet (contains standing water) regardless of any available water data. In the marsh area of each cell, there is a distribution of elevations based upon a hypsograph [3]. This hypsograph is used to determine the fraction of the marsh area that is still under water at a given water depth. A pond refers to permanently wet areas of small size, such as alligator holes, which are a maximum of 50 m^2 or 0.02% of the cell.

The fish population simulated by ALFISH is size-structured and is divided into two functional groups: small and large fishes. Both of these groups appear in each of the marsh and pond areas. Each functional group in each area is further divided into several fish categories according to age, referred to as ageClass, and each age class has 6 size classes, referred to as sizeClass. The fish population in each cell is summarized by the total fish density (biomass) within that cell. Each cell, as an element of the landscape matrices, contains an array of floating-point numbers representing individual fish density of various age classes. The length of the array corresponds to the number of age classes for that functional group. Normally, when a fish density is referenced, the value reflects the total fish densities of all the fish age classes combined. Fig. 2 shows the simple age and size categorization of the fish functional groups in four landscape matrices.

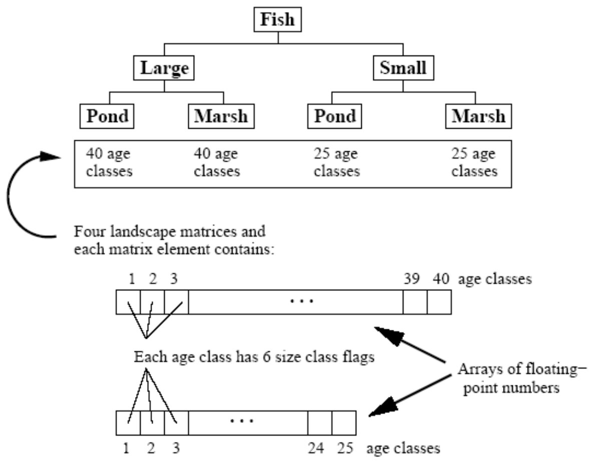

Fig. 2. Simple fish functional group categorization

In ALFISH, spatial and temporal fluctuations in fish populations are driven by a number of factors, especially the water level. Fluctuations in water depth, which affect all aspects of the trophic structure in the Everglades area, are provided through an input hydrology data file for each timestep throughout the execution of the model.

4.2 Layer-wised partition

To support the parallel computation of fish population dynamics in all age classes, two data structures (*groupInfo* and *layerInfo*) are introduced to support layer-wised partitioning.

```
Struct layerInfo       // information on fish function groups
  {  int Ngroup;       // number of function group
     int size[5];      // size of each function group
  };

Struct groupInfo       // information on layer-wised partition
  {  int Ngroup;       // number of fish function group on
                          this partition domain
     int group[2];     // size of each fish function group
                          on this domain
     int position[10]  // start and end layer position of
                          fish function group
  };
```

In our case, *layerInfo.Ngroup* is 2, and *layerInfo.size* is set as [25 40]. We partition all 65 age classes (including 25 classes for small fish and 40 classes for large fish) into all processors, and use the structure *groupInfo* to store appropriate information. Fig. 4 illustrates the initial layer-wised partition across 3 processors.

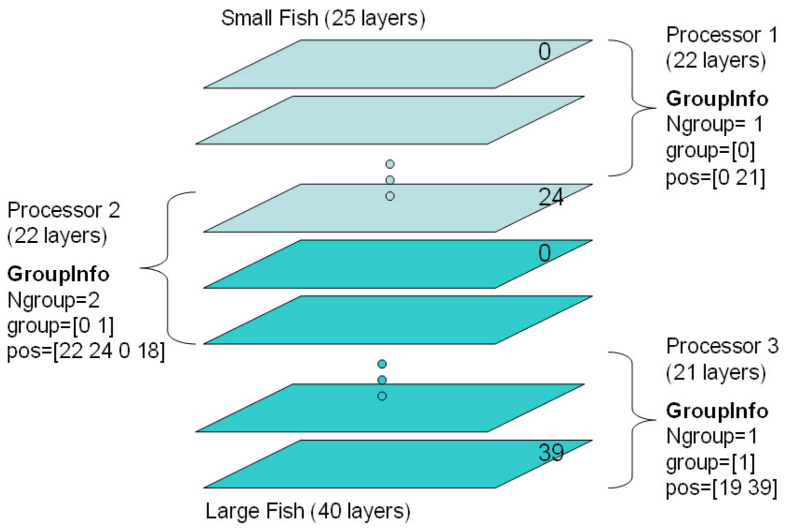

Fig. 3. Layer-wise partition across 3 processors

Initially, the value of position in *groupInfo* is equivalent to the value of age classes. However, every 30 days (6 timesteps of 5-days each), all of the fish are moved to the next age class. Thus, we introduce an *ageOffset* variable to eliminate the data movement involved in the sequential implementation (more details are provided in Sect. 4.3.

4.3 Fish dynamics and parallelization

The fish population model simulated by ALFISH is size-structured and is divided into two functional groups: small fish and large fish. Both of these groups are used in each of the marsh and pond areas. Each functional group in each area is further divided into several fish categories according to size. The fish population that occupies a cell area is represented as the fish density (*biomass*) within that cell. Basic behaviors of fish, including escape, diffusive movement, mortality, aging, reproduction and growth, are simulated in the model. Beside the fish dynamics, the model has to update the lower trophic level (food resources for the fish) data, hydrological data and execute a certain amount of I/O operations at each timestep. The parallel ALFISH deploys a master-slave communication model, which is illustrated in Fig. 4.

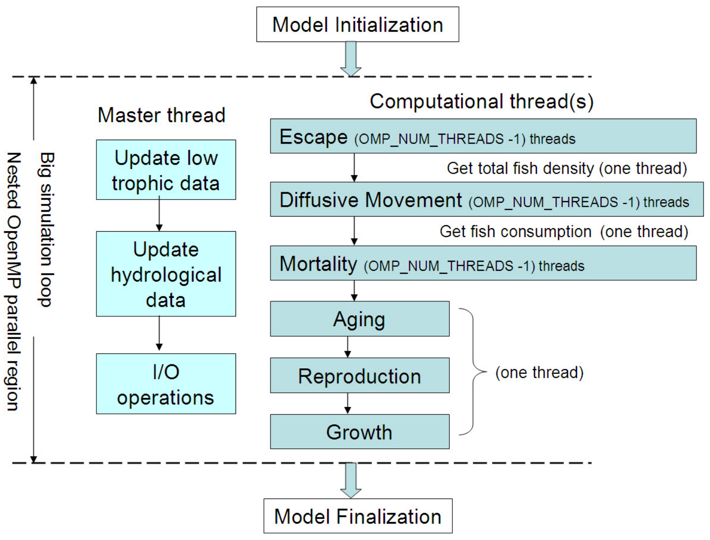
Model Initialization
Master thread
Computational thread(s)
Update low trophic data
Update hydrological data
I/O operations
Escape (OMP_NUM_THREADS -1) threads
Get total fish density (one thread)
Diffusive Movement (OMP_NUM_THREADS -1) threads
Get fish consumption (one thread)
Mortality (OMP_NUM_THREADS -1) threads
Aging
Reproduction
Growth
(one thread)
Big simulation loop
Nested OpenMP parallel region
Model Finalization

Fig. 4. Computational Model of Parallel ALFISH

There are some advantages associated with this implementation. From a performance aspect, the time-consuming I/O operations are separated from the much faster computational phases, which increase the efficiency of data access during the simulation. More importantly, the master thread provides a uniform, transparent interface for integration with other ATLSS ecological models. The main task of the master thread is to collect the data from computational threads, and write the results to the local disk. Therefore, the time-consuming output operations on the master processor are no longer a key component of the total execution time. All the computational threads are dedicated to simulate the fish dynamics. Since it is a multithreaded code, no explicit data exchange is needed during simulation. Data sharing in this implementation is supported by globally accessing the same memory location. The synchronizations required within each computational step are guaranteed by either explicit OMP_BARRIER or implicit barriers associated with the OpenMP parallel code sections. In this model, all the computational threads (OMP_NUM_THREADS-1) were used to simulate computational intensive fish behaviors (i.e. *Escape, Diffusive Movement* and *Mortality*), while only one thread was used to simulate other fish behaviors (i.e. *Aging, Reproduction* and *Growth*). Considering the efficiency of thread creation and join, those operations are deployed within the computational loop. More details on the nested OpenMP parallel region is shown in Appendix.

Escape. This function is designed to simulate the fish movement between marsh and pond areas within each cell caused by drying out or reflooding. At the current timestep, if the water depth of the cell has increased, and the cell has a pond, an appropriate fraction of fish (of the sizes previously too large) is moved from pond areas into marsh areas. If the cell water has decreased, an appropriate fraction of those fish are moved from the marsh area into pond area, another fraction of those fish are moved to adjacent cells, and the remaining portion of those fish are eliminated, referred as dying-out mortality [11]. Following this, one thread collects the total fish density in both marsh and pond areas, which will be used in the following diffusive movements.

Diffusive movement. This function is designed to simulate the movement of fish between adjacent cells mainly due to the relative differences in water depth and fish densities. The mathematical formula used to determine the number of fish to be moved is not presented here (see [3, 11]). Since this kind of fish movement is density dependent, the fish landscape matrix is only updated after all movement calculations are complete, to remove any order-based bias. After that, one computational thread summarizes the amount of biomass needed (*req_biomass*) for every fish to grow from its current size class to the next. At this stage, different parameters were classified to describe the bio-consumption related to fish function group (small and large fish) and the fish size. The *req_biomass* in each landscape will be used to calculate the fish mortality.

Mortality. Besides the mortality caused by the drying out of a landscape cell (see Sect. 4.3), the ALFISH model also simulates background mortality (*AgeMortality*) and density-dependent mortality (*FoodMortality*) in both marsh and pond areas in each landscape cell. *AgeMortality* is dependent on the individual fish age class, but it is independent of population size. *FoodMortality* is due to starvation. As this density of available prey decreases, the mortality rate of that specific age class and functional group increases. It is assumed that the starvation affects all age classes equally. In ALFISH, these two types of mortality are compared and the greater one is applied to the population. The mathematical formula to determine those fish mortalities are presented in [3, 11].

Aging. The age classes for the fish functional groups are defined by 30-day intervals. Every 30 days (6 timesteps of 5-days each), all of the fish are moved to the next age class. We do not use the mathematical formula defined in the sequential implementation, which would require extra date movement in parallelization. Rather, an ageOffset variable affiliated with each partition was introduced. The value *ageOffset* was initialized as 0, and every 30 days, *ageOffset* is increased by 1. Therefore, the value of the fish age can be derived using: *age = (position + ageOffset) % nAge*, where *age* is the value of fish age classes, position refers to the value of *groupInfo.position* and *nAges* is the total number of age classes of a particular fish functional group. For small and large fish, the values of *nAge* are 25 and 40, respectively.

Reproduction. For each functional group, if it is the appropriate time of year, the number of offspring is calculated. To prevent the population from producing too many new fish in a reproductive event, a constant maximum reproduction density is used. The new fish population from this stage is collected and used to update the fish population at position (layer) p, where the value of [*(p+ageOffset) % nAge*] equals 0.

Growth, Finally, fish that have survived are moved into the next size class since the size classes are equivalent to the 5-day timestep. All fish with an age class advance synchronously in size class, with the size class s incremented using *s=(s % 6)+1.*

5 Selected Model Results and Performance

5.1 Scenarios

The ALFISH models are mainly used to determine the pattern of fish density on the landscape for a variety of hydrology scenarios. The motivation for the particular scenarios chosen was the Restudy process for the selection of a plan for Everglades restoration [12]. In this paper, one scenario referred to as F2050 was applied. F2050 is a standard base scenario, which uses water data based on a 31-year time series of historical rainfall from 1965 through 1995, as well as sea level, population level and socioeconomic conditions projected for the year 2050. It also includes all of the previously legislated structural changes for the water control measures. Therefore, the simulation time of both the sequential and parallel ALFISH models is 31 years, from 1965 to 1995 using a timestep of 5 days.

5.2 Comparison of selected outputs

To verify parallel model correctness, that is, its ability to produce results similar to those of the sequential model, we compared outputs of both the sequential and parallel models. We analyzed several outputs and selected one set of data for comparison (the 31-year mean fish density and distribution on October 1). Fig. 5 shows the mean fish density map comparison on October 1. The left graph represents the output from the parallel ALFISH model, and the right graph is the output from the sequential ALFISH model. There are no observable differences between the outputs of these two models.

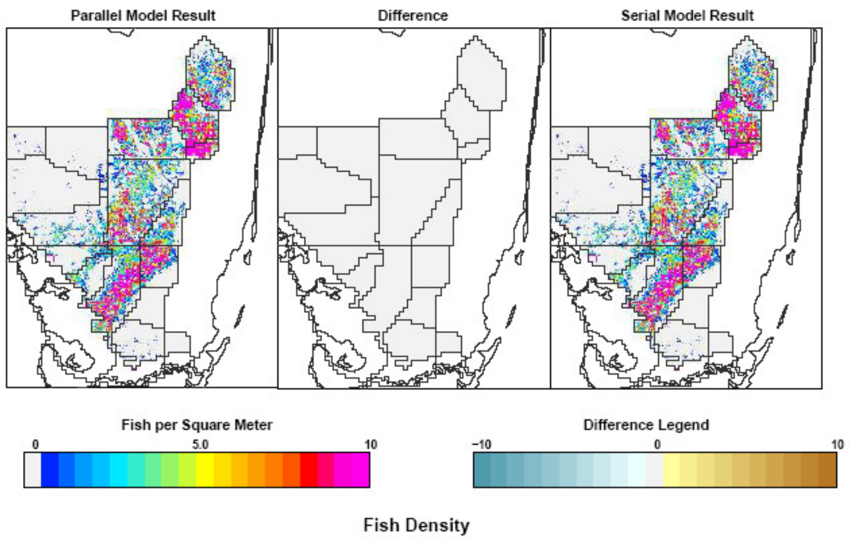

Fig. 5. Spatial 31-year average fish density map comparison in Everglades on Oct 1.

5.3 Model performance

In order to measure the scalability of the parallel ALFISH model, we deployed a series of parallel simulations using different numbers of threads (processors) (OMP_NUM_THREADS ranges from 2 to 66). Herein we define speedup (S) as

$$Speedup(S) = \frac{Model\ Execution\ Time(using\ 1\ computational\ thread)}{Model\ Execution\ Time(using\ N\ computational\ threads)} \quad (1)$$

Fig. 6 shows the speedup factor of the parallel ALFISH. Although static partitioning is applied in the model, the parallel ALFISH demonstrates satisfactory scalability when the number of computational threads is less than 25. The average execution time of the parallel ALFISH, using 1 computational thread, is about 11 hours, while the execution of the parallel ALFISH using 25 computational threads is about 39 minutes (the speedup factor being about 16). The speedup factor decreases when more than 25 computational threads were used, since (*i*) single thread execution portion within each computational loop is constant, and (*ii*) the parallelism overhead associated with thread creations and synchronizations increases when more threads are involved in the computation.

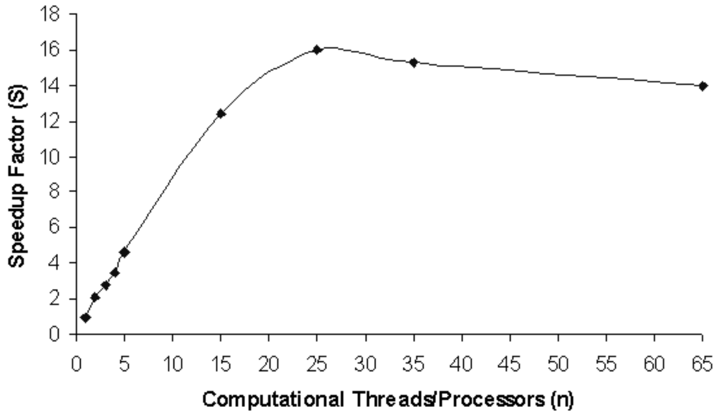

Fig. 6. Speedup factor of parallel ALFISH

6 Discussion and future work

The nearly identical outputs and excellent speed improvement (especially when the number of computational threads is less than 25) obtained from the parallel ALFISH model provide strong evidence that layer-wised partitioning can be highly effective for age- and size-structured spatially explicit landscape fish models on high-end shared memory computers using OpenMP. Our results indicate that even with simple static partitioning, the parallel ALFISH model demonstrates satisfactory scalability. In this paper, we adapted a one-dimensional layer-wised partitioning, which minimized the potential data sharing between computational threads, and allowed simple parallel implementation using OpenMP nested parallel sections. From user's perspective, more OpenMP features related to thread management will be appreciated. We are now developing a hybrid, reconfigurable two-dimensional partitioning (using both landscape (spatial) decomposition and age structure decomposition (layer-wised partition)) using a hybrid MPI/OpenMP model. Further plans for ALFISH model development include integration with other ATLSS models, (including an individual-based wading bird model) for parallel multi-scale ecosystem simulation on high performance computational platforms via grid computing [13, 14].

7 Acknowledgements

This research has been supported by the National Science Foundation under grant No. DEB-0219269. The sequential implementation was developed with support from the U.S. Geological Survey, through Cooperative Agreement No. 1445-CA09-95-0094 with the University of Tennessee, and the Department of Interior's Critical Ecosystem Studies Initiative. Authors also appreciate the support via the Computational Science Initiatives through the Science Alliance at the University of Tennessee (UT) and Oak Ridge National Library (ORNL). This research used resources of the Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725.

References

1. ATLSS: Across Trophic Level System Simulation. http://www.atlss.org.
2. Gross,L., DeAngelis,D., Multimodeling: New Approaches for Linking Ecological Models, 2002. in Predicting Species Occurrences: Issues of Accuracy and Scale, edited by J. M. Scott, P.J. Heglund, M.L. Morrison. pp.467-474.
3. Gaff,H., DeAngelis,D., Gross,L., Salinas,R., Shorrosh,M., 2000. A Dynamic Landscape Model for Fish in the Everglades and its Application to Restoration, Ecological Modeling, 127, pp.33-53.
4. Wolff, W.F., An Individual-Oriented Model of a Wading Bird Nesting Colony. 1994. Ecological Modelling, 114:72-75.
5. Curnutt, J. L., E.J. Comiskey, M. P. Nott and L. J. Gross. 2000. Landscape-based spatially explicit species index models for Everglade restoration. Ecological Applications 10:1849-1860.
6. 6. Wang,D., Gross,L., Carr,E., Berry, M., Design and Implementation of a Parallel Fish Model for South Florida, 2004. Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04). also at http://csdl.computer.org/comp/proceedings/hicss/2004/2056/09/205690282c.pdf
7. Immanuel,A., Berry,M., Gross,L., Palmer,M., Wang,D., 2005. A parallel Implementation of ALFISH: Compartmentalization Effects on Fish Dynamics in the Florida Everglades, Simulation Practice and Theory, 13:1, pp.55-76.
8. Wang,D., Berry,M., Carr,E., Gross,L., A Parallel Landscape Fish Model for Ecosystem Modeling, Simulation:The Transactions of The Society of Modeling and Simulation International. (in review)
9. OpenMP: Simple, Portable, Scalable SMP Programming. http://www.openmp.org.
10. Fennema,R., Neidrauer,C., Johnson,R., MacVicar,T., Perkins,W., 1994. A Computer Model to Simulate Everglades Hydrology, In Everglades: The Ecosystem and Its Restoration, edited by S.M.Davis and J.C.Ogden. St. Lucie Press, pp.249-289
11. Gaff,H., Chick,J., Trexler,J., DeAngelis,D., Gross,L., Salinas,R., 2004. Evaluation of and insights from ALFISH: a spatial-explicit, landscape-level simulation of fish populations in the Everglades, Hydrobiologia, 520, pp73-87.

12. U.S. Army Corps of Engineers, C & SF Restudy Draft Feasibility Report. http://www.restudy.org/overview.pdf. (Accessed 17 November 1998.)
13. Wang,D., Carr,E., Palmer,M., Berry,M., Gross,L., 2005. A Grid Service Module for Natural Resource Managers, IEEE Internet Computing, 9:1. pp.35-41.
14. Wang, D., Carr E., Ecomiske, J., Plamer M., Berry M., Gross, L., Grid Computing for Regional Ecosystem Restoration. http://www.sc-conference.org/sc2004-/schedule/index.php?module=Defaultaction=ShowDetaileventid=131

Appendix: Nested OpenMP parallel code section

```
for (date=current_date; date<=end_date; date+=5) {
#pragma omp parallel private(rank, myInfo, group, start, end)
  { rank = omp_get_thread_num();
    if (rank != 0) {
       remapping(current_partition, layers, rank, myInfo);
       #pragma omp parallel {
          compute fish.escape functionality
          #pragma omp barrier // synchronize computational threads
          if (rank == 1)  getfishTotaldensity
          #pragma omp barrier // make sure to update shared data
          compute fish.move functionality
          #pragma omp barrier // synchronize computational threads
          if (rank == 1)  getfishConsumption
          #pragma omp barrier // make sure to update shared data
          compute fish.mortality functionality
          #pragma omp barrier // synchronize computational threads
          if (rank == 1)  compute fish.aging functionality
          #pragma omp barrier // make sure to update shared data
          if (it is the right time))
               compute fish.reproduction functionality
          #pragma omp barrier // synchronize computational threads
          if (rank == 1) {
              if (it is the right time)
                  compute fish.reproduction functionality
              compute fish.growth functionality
          }
        }  // end of inner p-section (implicit synchronization)
    }
    #pragma omp barrier // block the master thread once
    if (rank ==0 ) {
       collect and save date at previous timestep
    }
} // end of external parallel section (implicit synchronization)
```