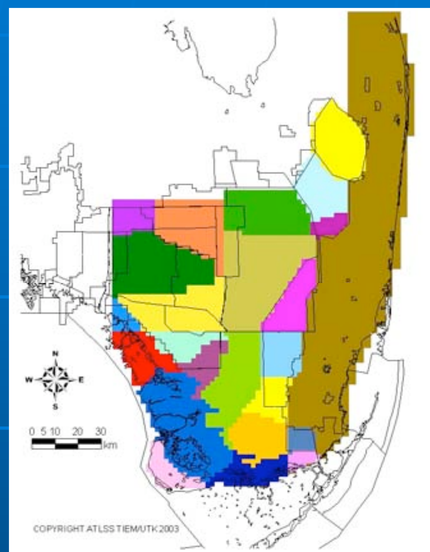
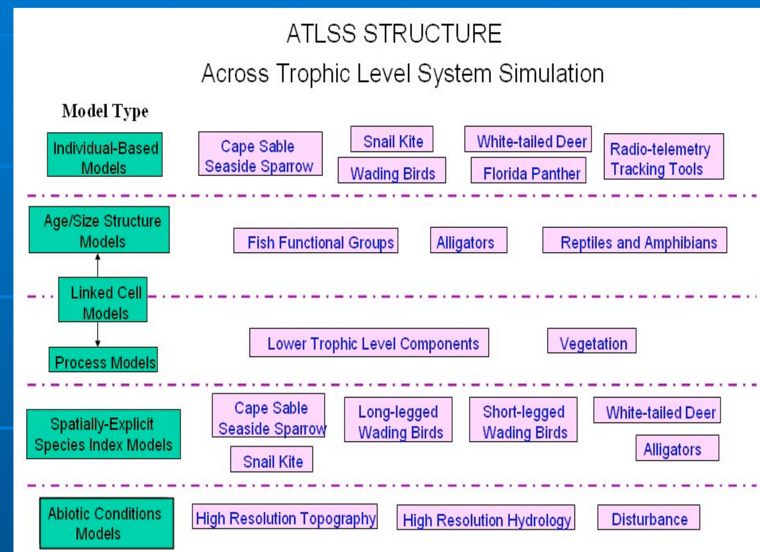
A Parallel Structured Ecological Model for High End Shared Memory Computers

Dali Wang

Department of Computer Science, University of Tennessee, Knoxville
dwang@cs.utk.edu

Background and Context

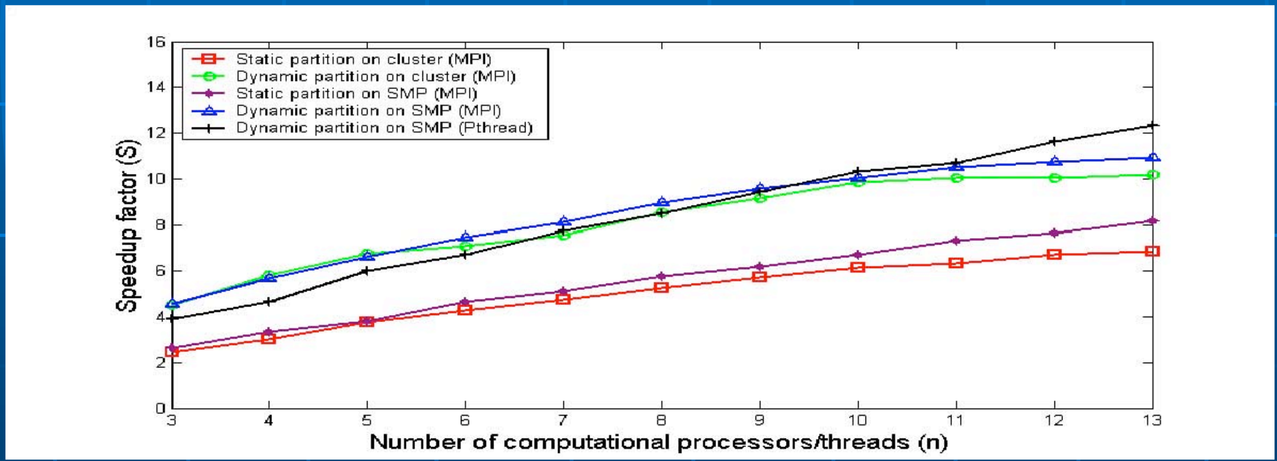




to provide a quantitative modeling package to assist stakeholders in the South and Central Florida restoration effort.
to aid in understanding how the biotic communities of South Florida are linked to the hydrologic regime and other abiotic factors, and
to provide a tool for both scientific research and ecosystem management.

Previous Experiments

A successful approach to the parallelization of landscape based (spatially-explicit) fish models is spatial decomposition. For these cases, each processor only simulates the ecological behaviors of fish on a partial landscape. This approach is efficient in standalone fish simulations because the low movement capability of fish does not force large data movement between processors.



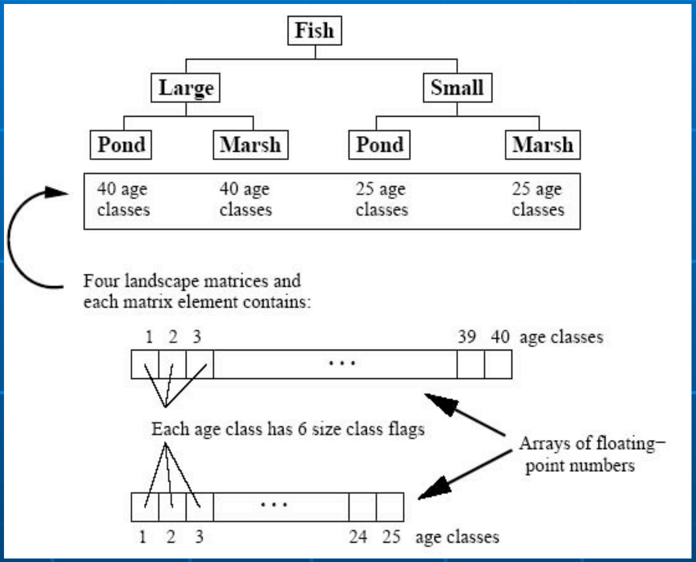
However, in an integrated simulation with an individual-based wading bird model, intensive data immigration across all processors is inevitable, since a bird's flying distance may cover the whole landscape.

Typical memory-intensive applications

Design a new partition approach
- to minimize the data transfer;
- to efficiently utilize the advanced features of shared-memory computational platforms;

Model Structure and Fish Dynamics

Computational domain: approximately 111,000 landscape cells, each has two basic types of area: marsh and pond.



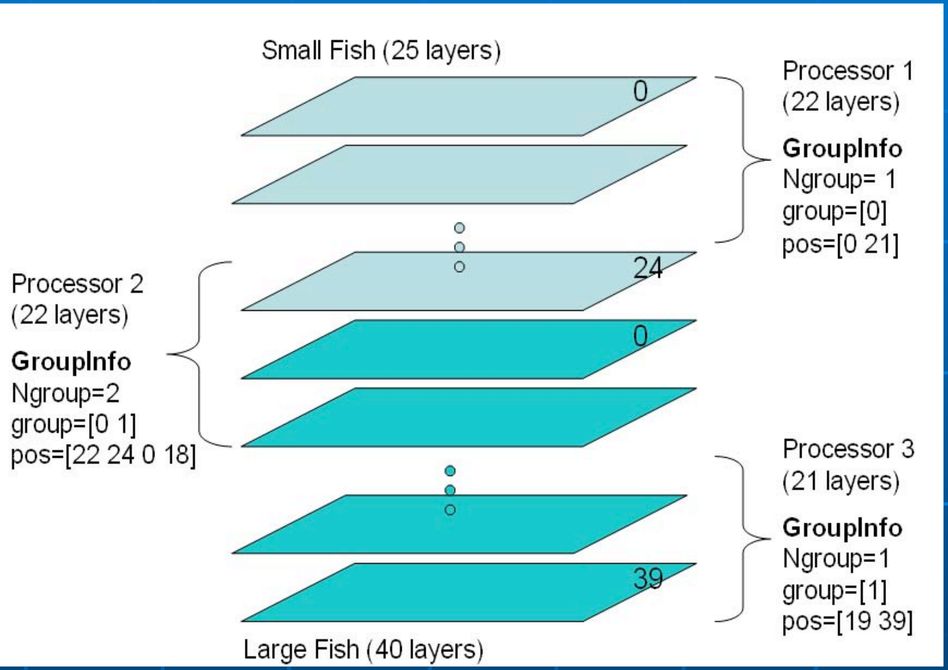
Fish Dynamics

Escape, Diffusive Movement, Mortality, Aging, Reproduction, Growth

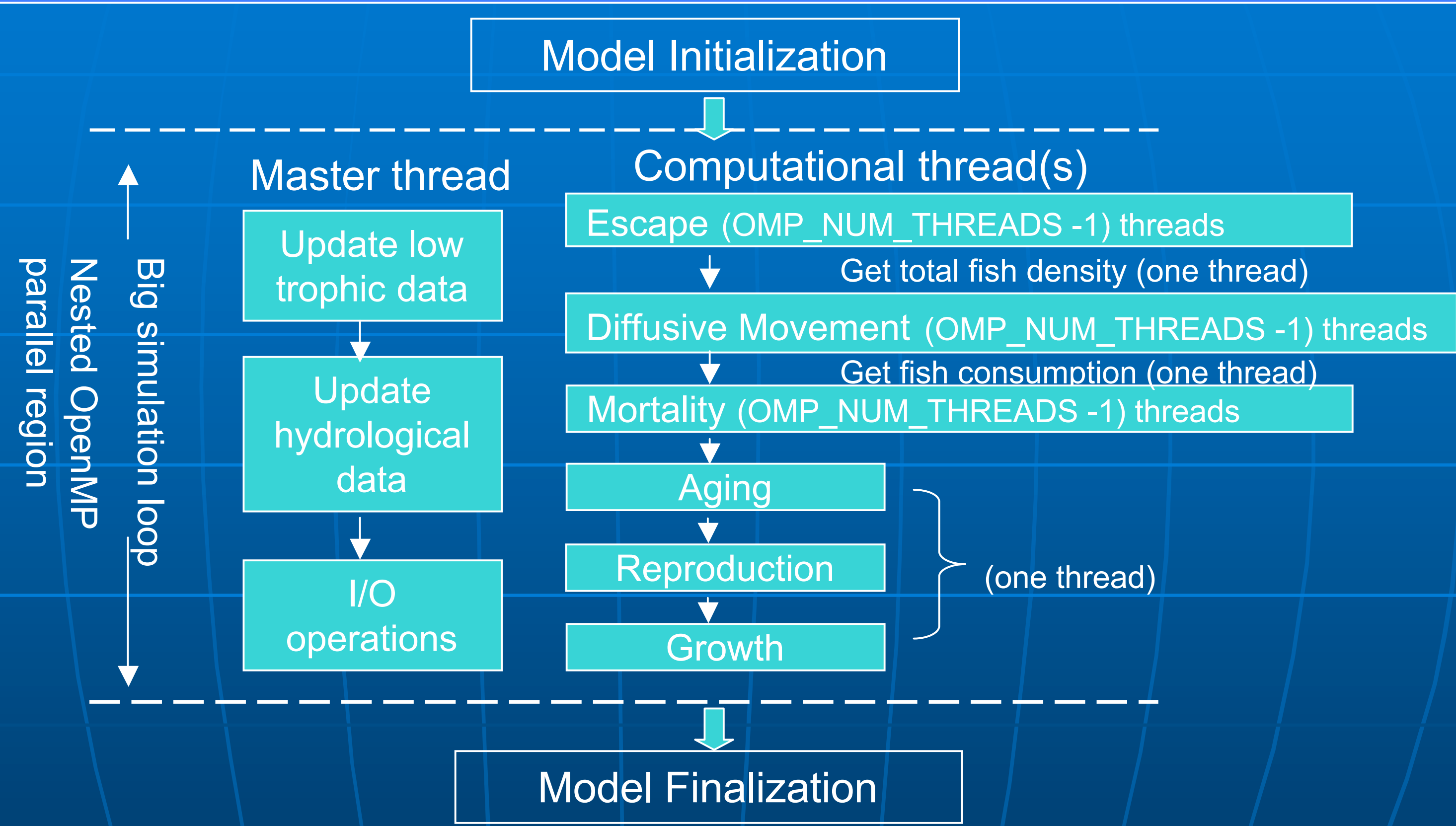
Parallelization Strategy

Layer-wised Partition:

1) Data transfer between processes can be minimized;
2) Dynamic load balancing can be easily implemented .



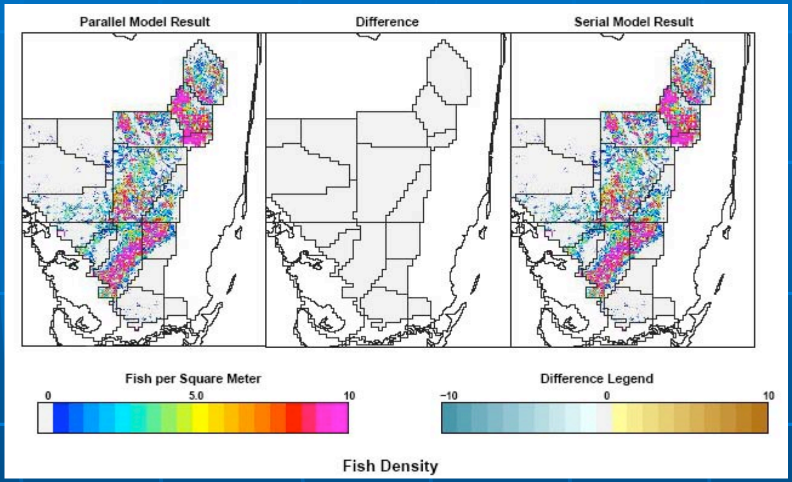
Computational Model

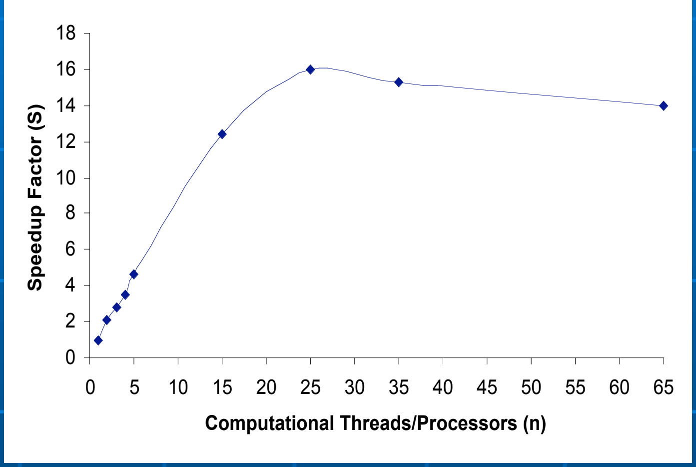


Computational Platform

- A SGI Altix system at the Center for Computational Sciences (CCS) of ORNL.
- 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.
- 8 GB of memory per processor for a total of 2 Terabytes of total system memory
- The operating system is a 64-bit version of Linux.
- The parallel programming model is supported by OpenMP.

Model Result and Performance





Future Work (Ecological aspect)

Field Data Calibration and Verification

Ecological Model Integration With

- individual-based wading bird model
- spatially-explicit spices index model

Ecological Impact Assessment (scenario analysis, …)

Simulation-based ecosystem management (spatial optimal control, real-time ecological system analysis)

Future Work (Computational aspect)

Large-scale simulation:

- Fine resolution- A hybrid, reconfigurable two-dimensional (spatial/temporal) partitioning using a hybrid MPI/OpenMP model.
- Fault tolerant computing/simulation

Model integration:

- A component based parallel simulation framework

Related References

Parallel Implementation

- D. Wang, et al. Design and Implementation of a Parallel Fish Model for South Florida. Proceedings of the 37th Hawaii International Conference on System Sciences.
- D. Wang, et al. A Parallel Landscape Fish Model for Ecosystem Modeling, Simulation: The Transactions of The Society of Modeling and International.

Grid Computing Module

- D. Wang, et al. A Grid Service Module for Natural Resource Managers, Internet Computing.

Performance Evaluations

- D. Wang, et al. On Parallelization of a Structured Ecological Model, International Journal on High Performance Computing Applications.

Simulation Framework

- D. Wang, et al. Toward Ecosystem Modeling on Computing Grids, Computing in Science and Engineering.
- D. Wang, et al. A Parallel Simulation Framework for Regional Ecosystem Modeling, IEEE Transactions on Distributed and Parallel Systems

Websites

- www.atlss.org
- www.tiem.utk.edu/gem

Acknowledgement

- National Science Foundation
- U.S. Geological Survey, through Cooperative Agreement with UT
- Department of Interior's Critical Ecosystem Studies Initiative
- Computational Science Initiatives through the Science Alliance at UT/ORNL

- This research used resources of the Center for Computational Sciences at ORNL, which is supported by the Office of Science of the Department of Energy

Parallel code section

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)
```

MPI/OpenMP

- Easy Process/Thread Management (dynamic vs. static)
- Minimum Code Modification vs. Flexible Performance Tuning (user involvement needed)
- Parallel Profiling Tools (TAU, PAPI, …)
- High Portability (SMP and Cluster, even New systems (multi-core, embedded)