Agent-based Scientific Applications and Collaboration Using Java

Xiaorong Xiang

Advisor: Dr. Kevin Bowyer

Department of Computer Science and Engineering
University of Notre Dame
Sponsored by NSF/ITR-DEB
Objectives

- New approach for NOM modeling
  - Agent-based modeling
- E-Science on the Web
- Intelligent interface components
- Build the NOM Collaboratory
- Performance analysis for scientific applications
Outline

- Introduction
- Modeling
- Core simulation engine
- Intelligent Web-based interface
- The NOM collaboratory
- Java performance analysis
- Conclusion
- Future work
Introduction

- What is Natural Organic Matter (NOM)?
- Role of NOM in various science disciplines
  - Mobility and transport of pollutants
  - Availability of nutrients for microorganisms and plant communities
  - Affects quality of drinking water
- Need to understand the evolution and heterogeneous structure of NOM
Forest Service Bog [DOC] 7 MW 2200

Twomile Creek [DOC] 17 MW 1500

Nelson Creek [DOC] 79 MW 900
Previous models

- **Two examples:**
  - **Daisy** (S. Hansen, H. E. Jensen, and N. E. Nielsen 1990-present): a soil plant atmosphere system model
  - **StochSim** (C. J. Morton-Firth 1998-present): Stochastic simulation of cell signaling pathways
Elemental Cycling

Scale
(size, temporal)

Small
(One molecule)
(nanoseconds)

Large
(Large ecosystem)
(Years)

Detail
(structure)

low
(Atoms number
Percentage)

high
(Forces between atoms
Electron density)

NOM1.0

Connectivity Maps

Daisy
StochSim


Elemental Cycling
Our model

- Agent-based modeling (Individual-based modeling)
  - Agent-based model
  - Each molecule as an individual object with spatial properties
  - Bottom-up approach
  - Stochastic model
  - Trace changes of the system $\Rightarrow$ Database and data mining technologies
Our model (cont.)

- Web-based scientific application
  - Serve as an example for E-Science
    - R. M. Jakobovits, J. F. Brinkley, C. Rosse, and E. Weinberger (1998): Enabling clinicians, researchers, and educators to build custom Web-based biomedical information system
  - Support the collaborations, data and information sharing between scientists
  - No installation, expensive computation resources needed by scientists
Outline

- Introduction
- **Modeling**
- Core simulation engine
- Intelligent Web-based interface
- The NOM collaboratory
- Java performance analysis
- Conclusion
- Future work
Modeling

- A complex system
  - Consists of a large number of objects
    - Molecules, Microbes
      - Heterogeneous properties
      - Individual behaviors
      - Interaction between objects
      - Objects behavior and interaction are stochastically determined by:
        - Attributes
        - Reactions rates
        - Environment condition
  - No central control
  - Emergent properties
Modeling (cont)

- Agent Attributes
  - Elemental composition (C, H, O, N, S, P)
  - Functional groups (double bonds, ring structure, alcohols …)
  - The origin of objects (spatial position in the system, parents of the objects)
  - Probability table of physical and chemical reactions
  - Molecule weight
Modeling (cont.)

- **Agent Behaviors:**
  - Transport through soil pores by water (spatial mobility)
  - Adsorb onto or desorbed from mineral surfaces
  - Chemical reactions
    - Total 10 types in current model
    - First order
    - Second order
  - Stochastically determined

- **Space:**
  - 2-D grid
Outline

- Introduction
- Modeling
- **Core simulation engine**
- Intelligent Web-based interface
- The NOM collaboratory
- Java performance analysis
- Conclusion
- Future work
Core simulation engine

- Implementation
  - Swarm toolkit
  - Java programming language (JDK 1.4.1_01)

- GUI version
  - View the animation of molecules
  - Easy for debugging and modeling

- Web-based version, the NOM simulation model
Core simulation engine (cont.)

- Read simulation parameter from the database (JDBC)
  - Environmental parameters (pH, temperature, light intensity, and so on)
  - Molecule types and distributions
- User defined time has been separated to a large number of equal size time steps
- Write relevant data into the database every time step (JDBC)
  - Trace the dynamic properties of individuals and the system over time
Outline

- Introduction
- Modeling
- Core simulation engine
- **Intelligent Web-based interface**
- The NOM collaboratory
- Java performance analysis
- Conclusion
- Future work
Web-based model

- Distributed, Web-based scientific application model
- Based on Java 2 Enterprise Edition (J2EE)
  - Standard HTML Forms / JSP
  - Java Servlets, Java Beans
  - JDBC - Oracle
  - Oracle Forms and Reports
- Three parts:
  - Intelligent Web-based interface
  - Core simulation engine
  - Data analysis, Data mining
Access NOM simulation from Web
Web-based interface
Intelligent components

- Components:
  - Email notification
  - Running time prediction
    - Static: number of molecules, number of time steps
    - Dynamic: current time step, current wall clock time
Intelligent components (cont.)

- Find similar simulations
  - Environment parameters
  - Molecule types and distributions
  - Retrieve the data sets from database
  - Points on a high dimension space
  - Euclidean distance
  - Ordered list
  - Review the simulation results or restart
Intelligent components (cont.)

- Automatic reverter
  - Save the state of each object in the system to the database every check point
  - Load the state to the core simulation engine
Intelligent interface design

- Model-View-Controller (MVC) design pattern
  - Model $\rightarrow$ Application logic
  - View $\rightarrow$ Presentation logic
  - Controller $\rightarrow$ Session management
- Separate the design task, centralized control
- Code reuse
- Make the application more easily maintainable
- Well-suited for round-trips of requesting and displaying data
Web interface implementation
Outline

- Introduction
- Modeling
- Core simulation engine
- Intelligent Web-based interface
- **The NOM collaboratory**
- Java performance analysis
- Conclusion
- Future work
Previous work


- BioCoRE: [http://ks.uiuc.edu/Research/biocore](http://ks.uiuc.edu/Research/biocore)

The NOM Collaboratory

- Interdisciplinary project
  - Chemist
  - Biologist
  - Ecologist
  - Computer Scientist

- Build and integrate software using J2EE
  - NOM modeling & simulation software
  - Standard data format definitions
  - Data querying and manipulation tools
  - Electronic communication tools
NOM Collaboratory

- NOM Simulator: Provide an intelligent interface to facilitate using the NOM simulator
- Search Engine: Provide an interface to search simulation information
- Discussion Board: Provide a threaded discussion board
- Chat Room: Provide a real-time chat box
- Molecule Editor: Provide an interface to define new molecule type
- Molecule Validator: Provide an administration role to validate the newly added molecule for public usage
- NOML: Provide a XML-based Markup Language definition
- File Upload: Provide an interface to upload publications
XML-based NOM Markup Language (NOML)

- NOML:
  - Standard data format
    - Environment.dtd, Molecules.dtd, Setup.dtd
  - Facilitate communication
    - User ==== User
    - Application ==== Application
  - Extension
NOML uploader

Client Side

<table>
<thead>
<tr>
<th>JSPs</th>
<th>upload</th>
<th>NOML file format (.xml)</th>
</tr>
</thead>
</table>

Server Side

Servlet

<table>
<thead>
<tr>
<th>Multipart Parser</th>
<th>XML File Input Stream</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Call JavaBean</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Parse Molecule</th>
<th>Parse Environment</th>
<th>Parse Setup</th>
</tr>
</thead>
</table>

Database
Data input options

- SCM format
- XML format
- Web interface
Other tools

- Molecule editor
  - Define new molecule type
- Molecule validator
  - Authorized persons (Chemists) to validate data
  - Share the molecule type
- Search engine
  - Ad-hoc query
  - View results of the completed simulations
  - Restart some simulations
Outline

- Introduction
- Modeling
- Core simulation engine
- Intelligent Web-based interface
- The NOM collaboratory
- **Java performance analysis**
- Conclusion
- Future work
Java for Scientific Applications

- Advantages
  - Portability, automatic memory management
  - Java built-in threads, Java RMI

- Disadvantage
  - Performance


Java Grande benchmark suite
http://www.epcc.ed.ac.uk/javagrande
Previous work

- Runtime environment optimizations
  - Just-In-Time compilers
  - Bytecode Optimizer
  - Adaptive compilers
  - Native code compilers

- J. M. Bull et al.: Benchmarking Java against C and FORTRAN for scientific applications. *In proceedings of the 2001 joint ACM-ISCOPE conference on Java Grande*

Software engineering perspective

- Profiling the program
  - Identify the bottlenecks
  - Determine the factors that affect performance

- Proper design
  - Eliminate bottlenecks
  - Improve scalability
Motivation

- The NOM simulation model is a typical large scale scientific application model
  - Long running time
  - Large amount of data output
  - Computation and I/O intensive
- Expect that our experiences can help other scientific applications developers using Java
Choice of data structure

Execution time (seconds)

<table>
<thead>
<tr>
<th></th>
<th>ArrayList</th>
<th>LinkedList</th>
<th>LinkList (with iterator)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get() &amp; shuffle()</td>
<td>5</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>Add() &amp; remove()</td>
<td>0</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Overall</td>
<td>5</td>
<td>20</td>
<td>15</td>
</tr>
</tbody>
</table>
Object reuse

- Object allocation 50% longer than in C++. (Sosnoski, 1999)
- Excessive object creation:
  - Increases the memory footprint
  - Forces more CPU cycles to be used for garbage collection
  - Increases the possibility of a memory leak
- Object reuse
  - Isolate the object
  - Reinitialize the object
  - Object pool management
**JDBC**

**Benchmark:**
- Case 1: Statement
- Case 2: PreparedStatement
- Case 3: Statement with transaction management
- Case 4: PreparedStatement with transaction management
- Case 5: Batch updates
Parallel data output with Java threads

Graphs showing:
1. Records insertion (x100) vs. Number of time steps
2. Speed up vs. Number of time steps
Choice of JVM

The graph shows the execution time (in seconds) as a function of the grid size. The execution time increases with the grid size for all JVMs. The Sun Client VMs have a lower execution time than the Sun Server VMs for all grid sizes. Additionally, the execution time for a 500 time step is lower than for a 1500 time step for both client and server VMs.
Choice of JVM (cont.)

![Graph showing speedup vs. grid size for different time steps.](image)
Scalability

- Two approaches:
  - Java built-in threads
    - Single JVM, shared memory
  - Java MPI (MPJ)
    - Multiple JVMs, distributed memory

- Equally separate the grid to 2 or 4 subset grids
- Synchronize all the threads or processes at each time step
Java thread model
Java thread model (cont.)

![Graph showing execution time vs grid size for Sequential model and Java threads model.](image)
Java thread model (cont.)
Message passing in Java (MPJ)

- **MPJ specification**

- **MPJ implementation**
  - **MPI wrapper**
  - **Pure Java implementation**
    - Jmpi (K. Dincer: Ubiquitous message passing interface implementation in Java: jmpi. 1999)
    - MPIJ (G. Judd: Dogma: Distributed object group management architecture. 1998)
Distributed memory model (MPJ)
Distributed memory model

![Graph showing execution time vs grid size for different scenarios: Time step 1500, MPJ 4 nodes (1500), MPJ 4 nodes (500), Time step 500.]

- LAM MPI, mpiJava. 4 machine in a cluster
Distributed memory model (cont.)

![Graph showing speedup as a function of grid size. The x-axis represents grid size with values from 200 to 1600. The y-axis represents speedup with values from 0 to 1.8. Two lines are plotted: one for Time step 500 and another for Time step 1500. The speedup increases with grid size.]

- Time step 500
- Time step 1500
Distributed memory model (cont.)

No communication between processes
Distributed memory model (cont.)
Other issue

- High performance compiler
  - GCJ
    - Depends on the applications
  - IBM High-Performance compiler for RS6000 architecture
- Code clean up
## Summary - potential 25x

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Speedup</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choice of data structure</td>
<td>2.8</td>
<td>Evaluate overall performance</td>
</tr>
<tr>
<td>Object reuse</td>
<td>-</td>
<td>Performance gain is small</td>
</tr>
<tr>
<td>JDBC</td>
<td>3</td>
<td>Use different JDBC technologies</td>
</tr>
<tr>
<td>Parallel data output</td>
<td>1.3</td>
<td>Overlap the computation and I/O</td>
</tr>
<tr>
<td>Choice of JVM</td>
<td>1.4</td>
<td>IBM JVM is valuable to evaluate</td>
</tr>
<tr>
<td>Java threads model</td>
<td>1.1</td>
<td>Evaluate different OS</td>
</tr>
<tr>
<td>MPJ model with communication</td>
<td>1.5</td>
<td>Reduce the communication overhead</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Modeling
- Core simulation engine
- Intelligent Web-based interface
- The NOM collaboratory
- Java performance analysis
- Conclusion
- Future work
Conclusion

- Agent-based stochastic model for simulating the NOM evolution with discrete temporal and spatial properties
- A Web-based interface
- The NOM collaboratory
- Java performance analysis for large scale scientific applications
Outline

- Introduction
- Modeling
- Core simulation engine
- Intelligent Web-based interface
- The NOM collaboratory
- Java performance analysis
- Conclusion
- Future work
Future work

- Model testing
  - Testing on the sorption
  - More features need to add into the core simulation engine
  - Discrete event vs. Time step
- Advanced algorithm for search similar simulations
- Delicate way to save the JVM state and restart the simulation
- Collaboratory:
  - More communication tools
  - More simulation models for NOM study
  - NOML extension
Contributions

- New approach for NOM modeling
  - Agent-based modeling
- E-Science on the Web
- Intelligent interface components
- Built the NOM Collaboratory
- Performance analysis for scientific applications
Publications to date

■ Proceedings
  ■ Huang, Y., Madey, G., Xiang, X., and Chanowich, E., "Web-based Molecular Modeling Using Java/Swarm, J2EE and RDBMS Technologies", 7th Swarm Researchers Conference (Swarm2003), Notre Dame, IN, April 2003.

■ Abstracts

■ Posters
Publication planning

- **NOM simulation**
  - World Conference on Natural Resource modeling
  - Simulation Practice and Theory, International Journal of the Federation of European Simulation Societies – EUROSIM
  - SIAM Journal on Scientific Computing

- **Performance analysis of Java for Scientific Applications**
  - Winter Simulation Conference
  - Joint ACM Java Grande – ISCOPE Conference
  - High performance computing and networking (HPCN)
  - IBM Systems Journal – Java performance

- **Scientific Collaboratory**
  - ACM Conference on Computer Supported Cooperative work (CSCW)
  - Information Resources Management Association, IRMA international conference
  - ACM Collaborative Virtual Environments
  - International Conference on Human Computer Interaction
  - Communications of the ACM
Acknowledgement

Thank you

Questions?

ESG (Environment Scenario Generator)
Algorithm

1. Init
2. Add molecule
   - Randomly pick one
3. Sorption
   - move
   - Leave
   - First order
   - Random number
   - Random reaction probabilities
   - Get reaction probabilities
4. Randomly pick one
5. Without split
   - Find nearest empty cell
6. With split
   - Find nearest empty cell
   - Update the probability table & Molecule structure
7. All checked
   - yes
   - Terminated
8. Completed
   - yes
Web-based interface

- Input the simulation parameters
- Invoke the simulation
- Stop the simulation
- View the real-time simulation results
Scalability

Two aspects related to the scalability:

- Large grid size (large number of objects)
- Large number of time steps

---

Graph showing execution time (sec) vs. grid size with different time steps: 500, 1000, 1500, and 2000.