AGENT-BASED SCIENTIFIC SIMULATION

Natural organic matter (NOM) is a heterogeneous mixture of compounds affecting ecosystem function and drinking water treatment. An agent-based stochastic model simulates NOM transformations, allowing forward modeling of NOM's synthesis and degradation. Its use of Java and Web technologies makes the simulation model accessible to scientists worldwide.

Natural organic matter (NOM) is a complex mixture of compounds resulting from the breakdown of animal and plant material in the environment. NOM is ubiquitous in terrestrial, aquatic, and marine ecosystems, playing a crucial role in such processes as soil evolution and fertility, pollutant mobility and transport, and mineral growth and dissolution.[1] NOM impacts all aspects of drinking water treatment and is responsible for most coagulant demand. Therefore, water with high levels of dissolved organic carbon usually has a high coagulant requirement and consequent high treatment costs. NOM can cause major water treatment problems because it reacts with chlorine to form disinfection byproducts (DBPs). Many of these DBPs are reportedly toxic and carcinogenic to humans if ingested over an extended period of time. Removing NOM and hence reducing DBPs is a major goal in water treatment.

NOM attracts numerous researchers, including chemists, geologists, and even computer scientists. Despite decades of research determining NOM's average properties and behavior, we know relatively little about the chemical structure of individual molecules within the mixture.[2–4] Current models of NOM behavior typically deal with bulk or average properties and use empirically determined "average" reaction parameters that can fit experimental data, but provide limited predictive capability.

Synthesis of NOM from its biological precursor compounds is both an interesting biogeochemical problem and an important aspect of predictive environmental modeling. On one hand, carbon-cycling models based on average properties of various organic carbon pools are too simplistic to represent NOM's heterogeneous structure and its complex behavior in the environment. On the other hand, molecular models employing connectivity maps or electron density are too computationally expensive to be useful for large-scale environmental simulations.

We propose a middle path: a stochastic model that allows, for the first time, forward modeling of the evolution of NOM's structure and properties. We implemented a simulation of NOM behavior using Java and the Swarm library. To make the simulation accessible to scientists around the world, we use the Java 2 Enterprise Edition (J2EE) and rela-

1521-9615/05/$20.00 © 2005 IEEE
Copublished by the IEEE CS and the AIP

YINGPING HUANG, XIAORONG XIANG, AND GREGORY MADEY
University of Notre Dame
STEVE E. CABANISS
University of New Mexico

tional database management system (RDBMS) technologies. Historic concerns about the use of Java for compute-intensive applications are fading as Java reaches performance parity with other languages (such as C/C++) and is increasingly used in scientific simulations.[5–7]

Agent-Based Stochastic Model

Our stochastic model represents individual molecules as discrete objects of specified elemental and functional group composition, size, and reactivity. We simulate NOM's temporal evolution from biological precursor compounds such as lignin, polysaccharides, and proteins using Monte Carlo algorithms in which we assign specific probabilities to particular transformations. These algorithms use new pseudorandom number generators with long periods and robust statistical properties from the Swarm library. We incorporate both physicochemical and biochemical effects probabilistically and predict the reactivity of the resulting NOM assemblage over time based on molecular property distributions.

This stochastic approach has several advantages:

- It's much less computationally intensive than molecular modeling or explicit kinetic simulation of hundreds of compounds.
- It's readily adaptable to various time scales and processes.
- It intrinsically handles NOM structural and functional heterogeneity.

Furthermore, this approach's self-manageable infrastructure lets users run simulations anytime from anywhere with reliable results.

Simulation Process

Because representing each molecule as a detailed chemical structure is too computationally expensive, we designed a data structure that is much simpler to work with yet encapsulates much of what we know and what we wish to know about NOM. The data structure includes

- the elemental formula—that is, the number of C, H, O, N, S, and P atoms in the molecule;
- a record of the molecular *origin*—that is, the type and size of the precursor molecule and the time it entered the simulation—letting us calculate separate turnover time and apparent ages for individual molecules and fractions; and
- functional group counts, including carboxylic acid, alcohol, ester, ketone, aldehyde, thiol, sulfato, amine, and peptide groups.

We implement the NOM simulation in a discrete 2D space with discrete time. The simulated space is a rectangular lattice. Each molecule occupies at most one cell, and each cell hosts one or more molecules. Molecules in the simulation represent a sample from a large population in the system under study. When we execute the simulation, molecules move to other cells or stay in fixed cells according to predefined simple rules describing physical processes. In chemical reactions, one molecule might split; two or more molecules might combine and occupy just one cell.

The model simulates 10 types of chemical reactions: ester condensation, ester hydrolysis, amine hydrolysis, microbe uptake, dehydration, strong C=C oxidation, mild C=C oxidation, alcohol (C-O-H) oxidation, aldehyde C=O oxidation, and decarboxylation. Each molecule has a probability for each type of chemical reaction. We base the probability calculation on the molecule's structure and the environment in which it resides. These environmental variables include time-step length; microbe, light, and fungal density; and temperature. After each chemical reaction, the simulation recalculates the probabilities of these reaction types, which it stores in an array associated with each molecule.

In each time step, for each molecule, the system generates a random number, which it uses to determine which chemical reaction will occur, if any. In the simulation, we control the sum of all the reaction probabilities to be less than 1 percent and partition the interval [0, 1] into 11 subintervals. The first interval's length equals the probability of the first reaction type; the second interval's length equals the probability of the second reaction type; and so on. The last interval's length is the probability that no reaction will occur. The generated random number from the interval [0, 1] resides in one of these intervals. Figure 1 shows these processes.

If the chosen chemical reaction type is a second-order reaction (the probability of higher-order (> 2) reactions is so small that we can safely ignore them)—that is, two molecules will be combined in the reaction—the system chooses the second molecule from one of its nearest neighbors who aren't yet involved in a chemical reaction. After the reaction occurs, the system updates the description and probability tables for one of the molecules at the end of the time step and removes the second molecule from the simulation.

Agent-Based Approach

Agent-based modeling (ABM), also known as indi-

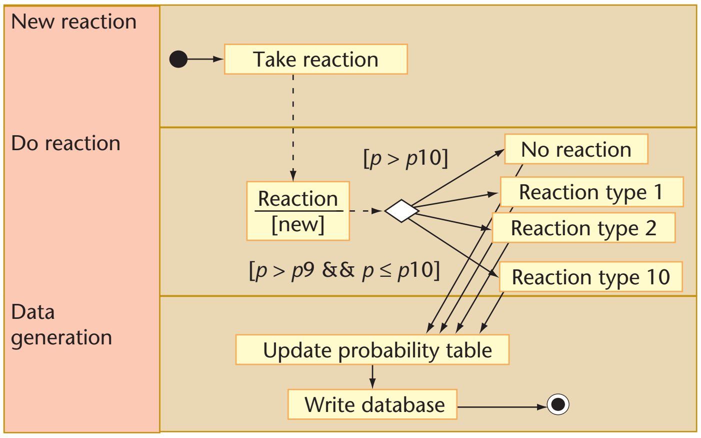


Figure 1. Simulation process. At each time step, the simulation system generates a random number for each molecule, which it uses to determine whether a chemical reaction will occur.

vidual-based modeling (IBM),[8] and equation-based modeling (EBM) are approaches for modeling complex systems. ABM and EBM differ significantly with respect to focus. ABM focuses on the individual's characteristics, tracking those characteristics through time, whereas EBM focuses on the population's average characteristics, tracking changes to those average population characteristics through time.[9] EBM simulates changes to the average population characteristics by iteratively solving a set of differential equations, whereas ABM simulates changes to each individual through time. For example, in a simulation of traffic flow through an intersection, an EBM would calculate an average traffic flux in suitable units—say, vehicles per minute—for each possible direction, whereas an ABM would simulate each vehicle's position or progress. The EBM is more compact and should require less processor time; however, it also leads to impossible situations (traffic doesn't flow in all directions simultaneously) and can't simulate gridlock due to cars stuck in the intersection. The ABM would simulate each vehicle as an agent, and could produce average traffic flux by averaging over the entire ensemble of vehicles; it would alternate traffic directions through the intersection and could simulate gridlock.

EBM can describe a system's known global properties, but rarely explains those properties' origins or tracks an individual component's behavior. ABM attempts to describe the heterogeneous aspects of individual agents and systems, providing a mechanism for interactions between agents and predicting phenomena at higher levels based on individual agents' actions in a bottom-up approach.

The NOM complex system consists of many heterogeneous molecules and microbes. Molecules move through soil via water flow, adsorb on the soil particle surfaces, and react with other molecules or microbes. Molecules' properties can change over time: new molecules can emerge (one molecule splits into two) or disappear (by reacting with microbes or combining with another molecule). Accurately capturing the NOM system's global properties with the EBM approach is difficult. Steve E. Cabaniss[10] shows that using average values to represent NOM's complexity can result in discrepancies when comparing the model results with laboratory study results. Therefore, ABM is more suitable for NOM modeling.

Researchers in cell biology are also investigating agent-based modeling of molecules as objects. At Cambridge University, for example, Thomas Shimizu and colleagues[11–13] created a stochastic simulator for chemical reactions among molecules. In their model, an independent software object (an agent in our model) represents each molecule.

The agent-based approach models NOM's evolution process more flexibly and accurately, but it is much slower than EBM. Performance is an important issue in agent-based simulation. To achieve high performance, developers must determine and understand the factors affecting the simulation's performance from a software engineering perspective, and identify and eliminate the bottlenecks limiting scalability during development. Approaches for exploring the simulation model's performance and scalability improvement include runtime optimization, database access, object use, and parallel and distributed computing. Using these technologies and implementing the distributed simulation model with the message-passing interface in Java (MPJ)[14] can improve performance by 25 times. We expect performance to increase more as the number of agents in the system increases. The advantages of using ABM in the NOM system might overcome performance concerns.

Verification and Validation

Verification and validation (V&V) processes increase confidence in simulations. Verification is about getting the simulation right whereas validation is about getting the right simulation. Although neither process guarantees absolute confidence, we used numerous V&V techniques on the NOM simulation.

We adapted Robert Sargent's V&V process to describe our agent-based model, as Figure 2 shows.[15] The simulation process starts by identifying a research question of interest. Through analysis and modeling, we developed a conceptual NOM model that includes the features relevant to the question. We based the conceptual model on theory and domain knowledge from environmental chemistry. This theoretical foundation includes

- NOM molecule heterogeneity;
- important NOM interactions with mineral surfaces such as adsorption, hemi-micelle formations, acid or complexing dissolution, and reductive dissolution;
- NOM interaction with pollutants;
- relationships between NOM adsorption and mineral surfaces and molecular weight; and
- probabilistic reaction kinetics based on elemental composition and the nature of functional groups in the molecules.

Incorporating such theory and domain knowledge provides initial *face validity*—that is, to domain experts, the conceptual model's logic includes appropriate mechanisms and properties for the research problem. Six scientists—two biologists, a chemist, a geomicrobiologist, and two soil scientists—evaluated the conceptual model for face validity. After its initial validation, we coded the agent-based simulation. At this step, verification methods such as code walk-throughs, trace analysis, input/output testing, and boundary testing confirmed the simulation's correctness.

To date, we've validated the simulation by comparing simulation behavior with mathematical models and experimental results. One version of the simulation that tracks the molecular weight of molecules adsorbing to mineral surfaces has yielded a lognormal distribution similar to that experimentally observed and reported by Cabaniss[3] and colleagues. Additional details on these validation results are reported by Leilani Arthurs and colleagues.[16] We've completed additional comparisons between theoretical, empirical, and simulation predictions.

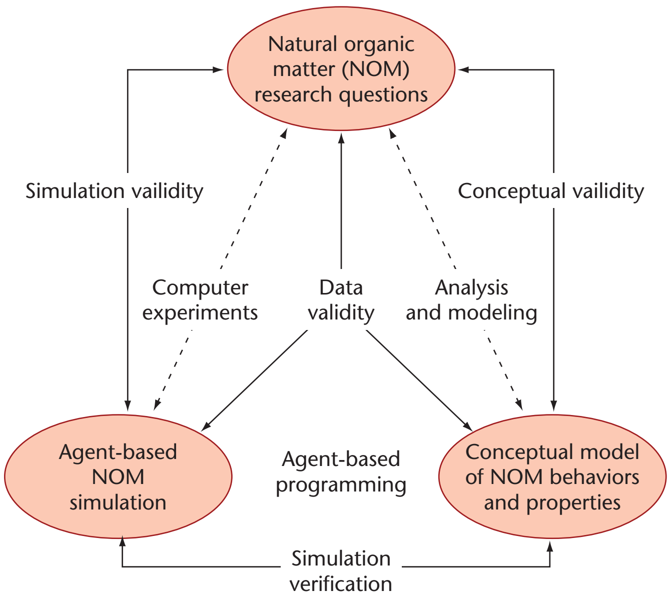
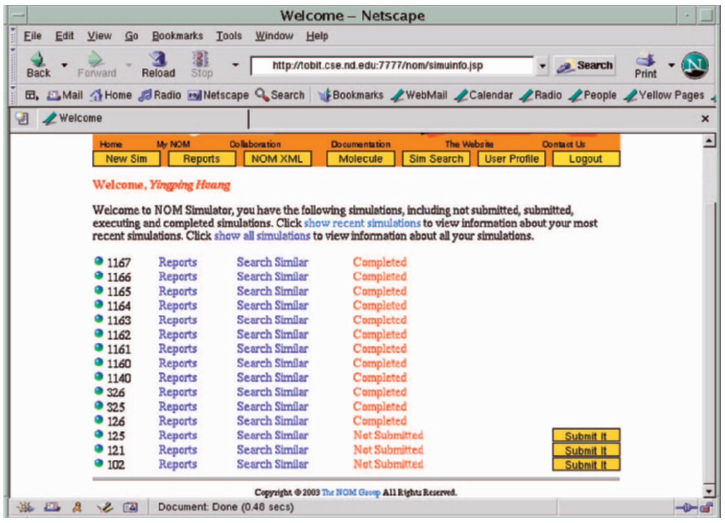


Figure 2. Verification and validation processes of the agent-based stochastic model. After identifying research questions of interest, we can develop a conceptual NOM model that includes relevant features.


Welcome – Netscape
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://tobit.cse.nd.edu:7777/nom/simuinfo.jsp
Search Print
Mail Home Radio Netscape Search Bookmarks WebMail Calendar Radio People Yellow Pages
Welcome
Home My NOM Collaboration Documentation The Website Contact Us
New Sim Reports NOM XML Molecule Sim Search User Profile Logout

Welcome, *Yingping Huang*

Welcome to NOM Simulator, you have the following simulations, including not submitted, submitted, executing and completed simulations. Click show recent simulations to view information about your most recent simulations. Click show all simulations to view information about all your simulations.

1167	Reports	Search Similar	Completed	
1166	Reports	Search Similar	Completed	
1165	Reports	Search Similar	Completed	
1164	Reports	Search Similar	Completed	
1163	Reports	Search Similar	Completed	
1162	Reports	Search Similar	Completed	
1161	Reports	Search Similar	Completed	
1160	Reports	Search Similar	Completed	
1140	Reports	Search Similar	Completed	
326	Reports	Search Similar	Completed	
325	Reports	Search Similar	Completed	
126	Reports	Search Similar	Completed	
125	Reports	Search Similar	Not Submitted	Submit It
121	Reports	Search Similar	Not Submitted	Submit It
102	Reports	Search Similar	Not Submitted	Submit It

Copyright © 2003 The NOM Group All Rights Reserved.

Document: Done (0.48 secs)

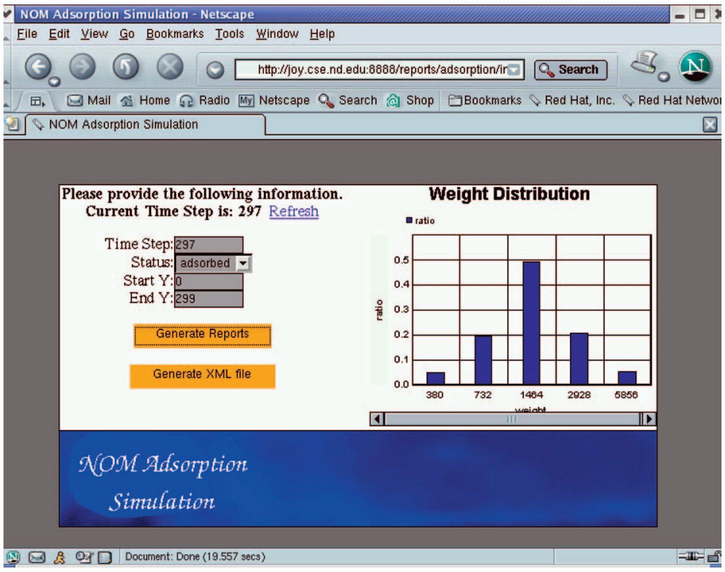
Figure 3. A snapshot of the NOM Web interface. Users interact with the interface to obtain information about their simulations.

New Paradigm of Scientific Inquiry

We used to speak of three approaches to science: experiment, theory, and the computational approach. Geoffrey Fox introduced a fourth paradigm,[17] which uses technologies such as the Web, databases, and data mining. We support the molecular simulation model using these technologies.

The NOM simulation system includes a Web interface, which lets users provide inputs for the simulations and obtain data analyses for them through a Web browser, as Figure 3 shows.

We implement the Web interface using J2EE (servlets, Java server pages, and Enterprise Java Beans) with the popular model, view, and controller (MVC) architecture, running on an Oracle9i application server. The simulation stores data output in Oracle databases with a Java database


NOM Adsorption Simulation - Netscape
File Edit View Go Bookmarks Tools Window Help
http://joy.cse.nd.edu:8888/reports/adsorption/in
Search
Mail Home Radio Netscape Search Shop Bookmarks Red Hat, Inc. Red Hat Network
NOM Adsorption Simulation
Weight Distribution
ratio
0.0
0.1
0.2
0.3
0.4
0.5
380
732
1464
2928
5856
weight
NOM Adsorption Simulation

Please provide the following information.
Current Time Step is: 297 Refresh

Time Step: 297
Status: adsorbed
Start Y: 0
End Y: 299

Generate Reports

Generate XML file

Document: Done (19.557 secs)

Figure 4. A sample report produced by the reports server.

connection (JDBC) and SQL*Loader. We generate reports using SQL, PL/SQL, XML SQL (XSQL), and an Oracle reports server. Figure 4 shows a sample report page for the NOM project. The "Simulation Technologies" sidebar briefly describes these components.

Self-Manageable Infrastructure

The NOM simulation uses an infrastructure built on a multitier architecture to support high scalability through load balancing and cross-simulation server migration, and high availability through redundancy and simulation resuming, as Figure 5 shows. The client tier communicates with the Web-server tier behind a firewall by specifying simulation input data, which is stored in the database-server tier. The simulation manager dispatches simulation jobs to the simulation-server tier, where an intelligent agent then invokes the simulations.

An intelligent agent is an autonomous process (implemented using SQL scripts and Bourne shell scripts) running on a simulation server. Intelligent agents perform the following operations:

- check for simulations submitted by users and dispatched by the simulation manager;
- run simulations and queue simulation results if the simulation manager is down;
- handle data transportation from local disk drives to the database-server tier;
- create database schema objects (such as tables and indexes) for reporting; and
- cancel simulations jobs as directed by the simulation manager.

The simulation manager uses a greedy algorithm that dispatches simulations to simulation servers with the least load average. The infrastructure is highly scalable at all tiers.

At the Web-server tier, application servers are platforms for J2EE applications. An Oracle9i application server cluster is a collection of application servers that can be configured as a single group. The application server cluster uses a metadata repository to configure each Oracle9i application server container for J2EE (OC4J) instance. We configure the metadata repository to be stored in the real application cluster in the database tier and can add new application servers during operation without downtime. Clients interact with the cluster as though interacting with a single application server. A load balancer routes client requests to instances of the cluster.

We scale the simulation-server tier linearly by installing new simulation servers running identical simulations. When an intelligent agent starts running on the new simulation server, the simulation manager automatically recognizes the server and can dispatch simulations to it. We include the simulation servers in our load-balancing and simulation-resuming algorithms (discussed later).

We use Oracle's real application cluster (RAC) in the database-server tier. In a RAC environment, all active nodes concurrently execute transactions against a shared database.

We achieve availability of the Web- and database-server tiers through some level of redundancy:

- At the Web-server tier, an Oracle9i application-server cluster eliminates the single point of failure by redundancy and failover in the system. We replicate client session state throughout the cluster, thereby protecting against the loss of session state in case of process failure.
- A simulation checkpoint (the saving of the simulation's internal state to the database so it can be restarted to recover from a hardware failure or continue a completed simulation from the precise state it ended in) and resuming feature handles the simulation-server tier's availability.
- At the database-server tier, nodes are isolated from each other so a failure on one node doesn't affect the whole system.

The simulation infrastructure is self-manag-

Simulation Technologies

Our simulation uses the following technologies, making it accessible to scientists around the world:

- *Java/Swarm* (http://wiki.swarm.org) is a Java library that implements agent-based technology. It can be used to model multiagent simulation of complex systems.
- *Oracle RDBMS* databases store and retrieve related information, including database schema objects to manage simulations and simulation input and output.
- *Java database connectivity* (JDBC) is a standard Java interface for connecting from Java to relational databases such as Oracle.
- *Java 2 Enterprise Edition* (J2EE) has historically been an architecture for building server-side deployments in Java. It can be used to build traditional Web sites, software components, or packaged applications.
- *XML-SQL* (XSQL) delivers XML documents of simulation results to users by querying the database.

These state-of-the-art technologies, combined with autonomic computing, make developing a robust infrastructure to support Web-based simulations feasible.

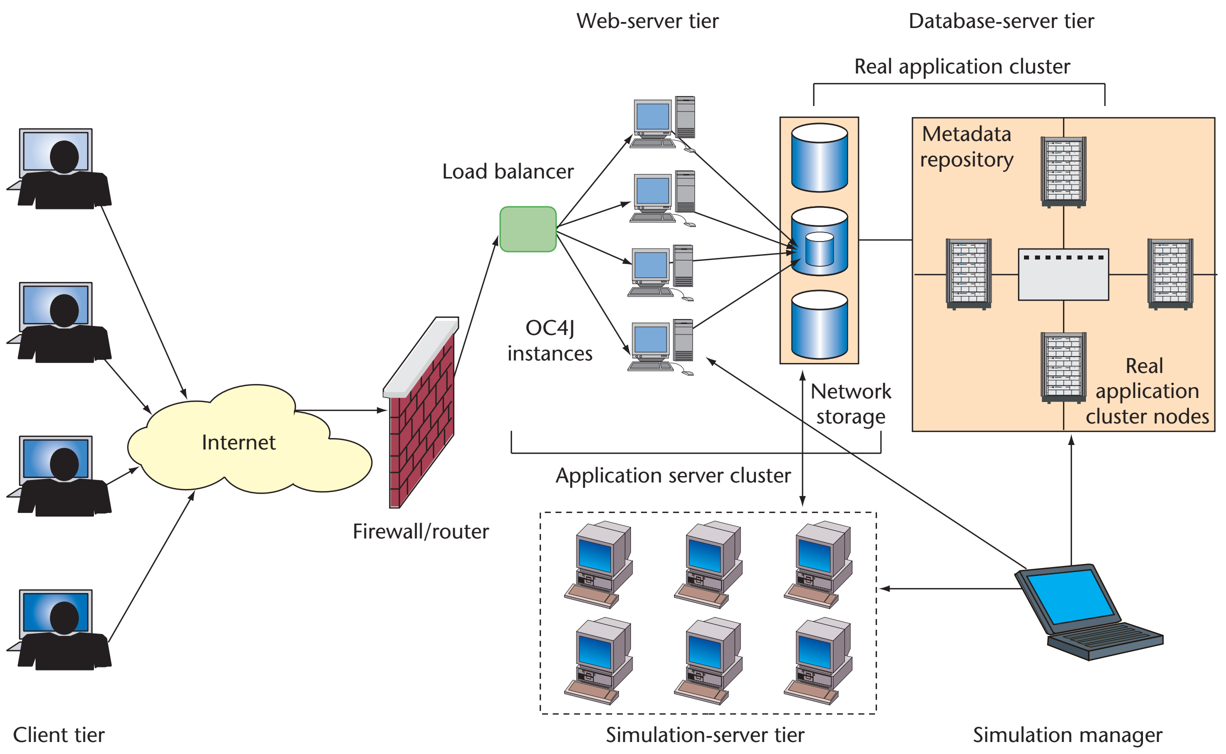


Figure 5. The NOM simulation multitier infrastructure. The client tier interacts with the Web-server tier. Data is stored in a database-server tier.

ing—that is, self-configuring, self-healing, self-optimizing, and self-protecting (see www.ibm.com/autonomic/ for case studies, white papers, and downloads on autonomic computing). For a detailed implementation of these features, please visit the project Web site at www.nd.edu/~nom/.

Self-Configuring

Self-configuring means the simulation infrastructure can configure itself to meet goals specified by the system administrators. The infrastructure automatically deploys new Web, simulation, and database servers, and removes existing ones. For example, an intelligent agent automatically recognizes a new simulation server and checkpoints running simulations on existing simulation servers, anticipating the possibility of one of those servers being removed for maintenance and enabling the

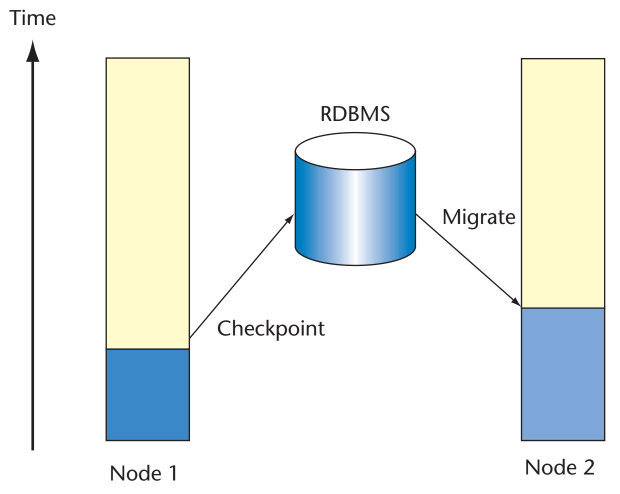


Figure 6. Checkpoint-initiated load balancing. After a simulation completes a checkpoint, the simulation manager decides whether to migrate a simulation server based on input from the intelligent agent.

migration of simulation jobs to take advantage of the new simulation server.

In addition, the infrastructure automatically deploys new simulation jobs based on simulation metadata (simulation description data). For example, it automatically generates Web interface and database objects from simulation metadata.

The infrastructure automatically reconfigures itself for newly submitted simulations. For example, it executes a high-priority new simulation immediately while checkpointing and pausing lower-priority simulations.

Self-Healing

Self-healing means that the simulation infrastructure can detect, identify, and recover failed components. The infrastructure has the following self-healing properties:

- automatic restart of failed Web and database servers, and
- automatic resumption of crashed simulations based on simulation checkpointing.

In long-running simulations, an intelligent agent checkpoints the simulation status to the database-server tier. In case of a crash, the agent restarts the simulation from a previous checkpoint. If a simulation server fails, the simulation manager temporarily removes the simulation server from the simulation-server tier and dispatches simulations to other simulation servers by restarting them from the most recent checkpoints. After the failed simulation server recovers, its intelligent agent can register the recovered simulation server to the simulation manager, which can then dispatch new simulations to the recovered simulation server.

The interval (that is, the number of time steps) between checkpoints varies by simulation. The mean time to crash, number of requested time steps, required time to restart, and required time to checkpoint determine the interval.

Self-Optimizing

Self-optimizing means the simulation infrastructure can optimize itself to meet user requirements. This infrastructure has the following self-optimizing properties:

- load balancing among Web servers;
- load balancing among database servers;
- a greedy load-balancing algorithm among simulation servers (the simulation manager picks the simulation server with lowest load average to run a new simulation); and
- a checkpoint-initiated load-balancing algorithm across simulation servers.

When a running simulation completes a checkpoint, an intelligent agent contacts the simulation manager about whether migrating the simulation to another simulation server is beneficial. The simulation manager decides which simulation server to migrate, accounting for the simulation migration overhead. Figure 6 shows the checkpoint-initiated load-balancing algorithm.

Self-Protecting

Self-protecting means the simulation infrastructure can prevent unauthorized access. Role-based access control, in which users are assigned different roles (such as owner or public), lets users share simulation results with restrictions.

The infrastructure also provides firewall protection via a packet-filtering tool called *iptables*. This tool scans all network packets and blocks or forwards them to appropriate servers based on a chain of rules maintained by an infrastructure agent. The simulation infrastructure is thus an intranet behind a firewall.

With the support of this autonomic infrastructure, we've deployed several NOM simulation models online. Scientists can use these simulation models to conduct experiments to better understand the behavior of

NOM. Currently, we're employing machine-learning methods to build intelligent agents based on collected simulation inputs.

We believe that this autonomic infrastructure can also be applied to support scientific simulation in other areas because it provides a reliable platform for deploying Web-based simulations to users. It also shows how information technologies and autonomic computing can help researchers in the field of scientific simulations.

Acknowledgments

The US National Science Foundation (ITR Grant 0112820) and the Center for Environmental Science and Technology at the University of Notre Dame partially funded this research. We thank Patricia Maurice and Leilana Arthurs, Civil Engineering and Geological Sciences, and Ryan Kennedy, Computer Science, University of Notre Dame, for support with model validation.

References

1. G. Aiken et al., *Humic Substances in Soil, Sediment, and Water—Geochemistry, Isolation, and Characterization,* John Wiley & Sons, 1985.
2. D. McKnight and G. Aiken, *Sources and Age of Humus*, Springer-Verlag, 1985.
3. S. E. Cabaniss et al., "A Log-Normal Distribution Model for the Molecular Weight of Aquatic Fulvic Acids," *Environmental Science & Technology*, vol. 34, no. 6, 2000, pp. 1103–1109.
4. K. Kalbitz et al., Controls on the Dynamics of Dissolved Organic Matter in Soils: A Review," *Soil Science*, vol. 165, no. 4, 2000, pp. 277–304.
5. V. Getov et al., "Multiparadigm Communications in Java for Grid Computing," *Comm. ACM*, vol. 14, no. 10, 2001, pp. 118–125.
6. G. Thiruvathukal, "Java at Middle Age: Enabling Java for Computational Science," *Computing in Science & Eng.*, vol. 4, no. 1, 2002, pp. 74–84.
7. O. Vormoor, "Quick and Easy Interactive Molecular Dynamics Using Java3D," *Computing in Science & Eng.*, vol. 3, no. 5, 2001, pp. 98–104.
8. J. Gross, "Agent-Based Modeling in Ethnobiology: A Brief Introduction to Outside," presentation, NSF Workshop on Priorities in Ethnobiology, 2002; www.tiem.utk.edu/~gross/missouri.402.talk.fmt.
9. H.V.D. Parunak, R. Savit, and R.L. Riolo, "Agent-Based Modeling vs. Equation-Based Modeling: A Case Study and User's Guide," J.S. Sichman, R. Conte, and N. Gilbert, eds., *Proc. Multiagent Systems and Agent-Based Simulation* (MABS98), Springer Verlag, 1998, pp. 10–25.
10. S.E. Cabaniss, "Modeling and Stochastic Simulation of NOM Reactions," Univ. of New Mexico, working paper, 2002; www.nd.edu/~nom.
11. L.N. Norere and T.S. Shimizu, "StochSim: Modeling of Stochastic Biomolecular Processes," *Bioinformatics*, vol 17, no. 6, 2001, pp. 575–576.
12. T.S. Shimizu and D. Bray, "Computational Cell Biology—The Stochastic Approach," *Foundations of Systems Biology*, H. Kitano, ed., MIT Press, 2001.
13. T.S. Shimizu, S.V. Aksenov, and D. Bray, "A Spatially Extended Stochastic Model of the Bacterial Chemotaxis Signalling Pathway," *J. Molecular Biology*, vol. 329, no. 2, 2003, pp. 291–309.
14. B. Carpenter et al., "MPJ: MPI-Like Message Passing for Java," *Concurrency: Practice and Experience*, vol. 12, no. 11, 2000, pp. 1019–1038.
15. R.G. Sargent, "Validation and Verification of Simulation Models," *Proc. 31st Winter Simulation Conf.*, ACM Press, 1999, pp. 39–48.
16. L. Arthurs et al., "Agent-Based Stochastic Simulation of Natural Organic Matter Adsorption and Mobility in Soils," *Proc. 11th Int'l Symp. Water-Rock Interaction*, 2004, pp. 921–925.
17. G. Fox, "E-Science Meets Computational Science and Information Technology" *Computing in Science & Eng.*, vol. 4, no. 4, 2002, pp. 84–85.

Yingping Huang is a graduate student in the Department of Computer Science and Engineering at the University of Notre Dame. His research interests include autonomic computing, simulation, data cleansing, and data mining. Huang is a member of the ACM, the IEEE Computer Society, and the Society for Modeling and Simulation International. Contact him at Dept. of Computer Science and Eng., Univ. of Notre Dame, Notre Dame, IN 46556; yhuang3@nd.edu.

Xiaorong Xiang is a graduate student in the Department of Computer Science and Engineering at the University of Notre Dame. Her research interests include Web services, the Semantic Web, Web portals for scientific collaboration, and agent-based modeling and simulation. Xiang is a member of the ACM, the IEEE Computer Society, and the Society for Modeling and Simulation International. Contact her at xxiang1@nd.edu.

Gregory R. Madey is director of graduate studies and an associate professor in the Department of Computer Science and Engineering at the University of Notre Dame. He has a PhD and an MS in operations research from Case Western Reserve University. Madey is a member of the ACM, the Association for Information Systems (AIS), the IEEE Computer Society, Informs, and the Society for Modeling and Simulation International. Contact him at gmadey@nd.edu.

Steve E. Cabaniss is professor of chemistry at the University of New Mexico. His research interests include problems in environmental chemistry, including developing experimental techniques and models to examine the behavior of metals and organic compounds in aquatic and terrestrial systems. Cabaniss has a BS in chemistry from the Massachusetts Institute of Technology and an MS and PhD in environmental science and engineering from the University of North Carolina at Chapel Hill. He is a member of the American Chemical Society and the American Society for Limnology and Oceanography. Contact him at cabaniss@unm.edu.