Autumn 2011 Term Report
Timothy Chuang

Thesis Subject
Design and Qualitative/Quantitative Analysis of Multi-Agent Spatial Simulation Library

Introduction
MASS (Multi-Agent Spatial Simulation) is a new parallelization library for multi-agent and spatial simulation that facilitates individual cell centered programming of multi-agents and simulation spaces. MASS composes of a user application of distributed array elements and multi-agents that represent each individual simulation cell or an entity. It contains a set of methods that allow users to easily manipulate the behavior of each individual cell in their simulation space, and are designed to accommodate a wide array of needs for general scientific computing applications such as molecular dynamics, artificial society, and home automation.

The current version of the MASS library consists of two parts – multi-threaded and multi-process versions. Both versions are identical in functionality. The multi-threaded version is designed to run on a computing node with multiple cores, while multi-process version is designed to run on multiple computing nodes.

As part of this research project, I will combine the two versions to utilize a cluster of multi-core computing nodes. The final version of MASS will have mobile agent implementation and will be evaluated against several target systems for performance and programmability comparison.

Goals of My Research

• Complete and optimize the MASS library with mobile agents by combining the two versions.
• Demonstrate the competitive performance to be gained from parallel execution of the MASS library when compared against other parallel execution platforms such as Java MPI and Java programs with multithreading and sockets.
• Demonstrate the programmability advantage of the MASS library using Java Grande Benchmark Tests as well as several applications such as temperature prediction, Wave2D, MatSim, computational fluid dynamics, and other similar benchmark tests.
• Participate in a major conference such as CCGrid, HPC, HPDC and SuperComputing to present this research work.

Objectives Completed in Autumn 2011

• Completed the first version of MASS and verified its correctness
• Conducted preliminary performance evaluation and tune-ups using Wave2D and CFD

Primary Objectives for Winter 2012

• Complete the integration of agent functionality into the MASS library
• Conduct performance evaluation and tune-up
• Draft conference paper for submission by January 26th, 2012
• Conduct programmability evaluation
**MASS Design and Implementation**

The combined version of multi-threaded and multi-process version of the MASS library is built upon John Spiger’s multi-threaded version of MASS. To facilitate inter-node communication, I have created a communication layer using JSCH and socket.

The design and implementation of major functionalities of the library is summarized below.

**Initialization**

In the initialization phase, the master node launches remote process $MProcess$ on each computing node and establishes communication channel using JSCH. JSCH is the main communication channel that allows the master node to send various commands to remote nodes. The remote process $MProcess$ sits in a blocking read awaiting commands from the master node. Each remote process and the master node create a pool of worker threads that are ready to perform parallel processing.

Log files are created to monitor progress of each remote computing node and to record errors should the application encounter unexpected behavior. The master node simply outputs errors to the console.

**Call-All**

The Call-All method allows the user to perform a user-specified operation on all cells in the simulation space in parallel.

The master node sends the “Call-All” command to remote computing node and all available computing nodes execute the command in parallel. If a return value is required, the master node uses the JSCH channel created per remote computing node during the initialization phase to receive the return values.

**Exchange-All**

The Exchange-All method allows the user to exchange data stored in each cell with all of its user-defined neighboring cells in the simulation space.

An exchange helper $ExchangeHelper$ object is instantiated during the initialization phase to act as a server thread that constantly listens to a specified port for client connection requests from other computing nodes. This ensures timely response whenever a connection needs to be established. During the exchange all call, if the exchange needs to be made with a remote node, each thread writes the host name of the remote node and the remote exchange request that needs to be sent to an exchange request map with the host name being the key. The following diagram demonstrates how connections are established between remote processes during the execution of exchange all call.
After the local exchange is complete, all worker threads begin processing the remote exchange requests by selecting a hostname on a first-come-first-serve basis and process the remote exchange with the selected host. This is done to ensure that the connection object, which is cached on a map per remote host using the remote host name as key, is not shared by multiple threads.

When a host is selected by a thread, it checks the connection map to see if the connection has already been established between the local host and the remote host. If not, the thread establishes connection and process the remote exchange request. The following diagram demonstrates how local and remote exchange requests are processed.

Ideally, there will be more threads than the computing nodes exchangeAll call needs to communicate with. However, there are cases where there are more computing nodes stored on the exchange request map than the available threads. In this case, one of the worker threads picks up the remaining exchange requests and processes them.
Preliminary Test Results for Wave2D

Wave2D is a two dimensional wave simulation application based on Schroedinger Equation that performs floating point calculation of wave height on each cell and each cell communicates with its surrounding cell to update wave height.

Wave2D is selected as a benchmark application due to its floating point operations and the need of each cell to communicate with all surrounding cells.

**Test setting**
Simulation size: 1500 by 1500 grid  
Simulation time: 100  
Update interval: 10

The performance gain is in an expected pattern as more computing nodes are added to the simulation. There is one major advantage of having access to multiple computing nodes’ memory as a single node is incapable of handling 1500 by 1500 grid without at least 6 gb of memory.
Exchange All calls are where the application spends the majority of the time processing. The performance also increases in an expected pattern as more computing nodes are added to the simulation. It is a good indication that network overhead is not contributing to the performance hit as more communication overhead is introduced by adding more computing nodes in the simulation provided that the simulation size is large enough.

**Detailed Progress Report**

I completed the first fully functional version of the combined multi-process and multi-threaded MASS library and conducted preliminary performance tests and various performance tune-ups to optimize the multi-threaded nature of the library.

I am continuing to fine-tune the performance of MASS and am currently working on integrating agent functionality into the MASS library. I plan on continuing the integration work in the coming weeks, and I believe that I am still on track for all of the deliverables.

**Detailed Research Plan by Quarter**

<table>
<thead>
<tr>
<th>Autumn 11</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1 to 2</td>
<td>Familiarization with existing MASS library code base</td>
</tr>
<tr>
<td>Week 3 to 4</td>
<td>Complete the MASS multi-process version</td>
</tr>
<tr>
<td>Week 5</td>
<td>Complete the MASS library (multi process and multi-threaded into one)</td>
</tr>
</tbody>
</table>
Autumn 2011 Term Report  
Timothy Chuang  

| Week 6     | Complete the MASS library          |
| Week 7     | Conduct preliminary performance evaluation |
| Week 8 - 9 | Conduct performance tune-up         |
| Week 10    | Draft design documents and manual of MASS |

<table>
<thead>
<tr>
<th><strong>Winter 12</strong></th>
<th><strong>Milestone</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1</td>
<td>Complete the MASS library with Agent implementation and deliver design documents and manual of MASS</td>
</tr>
<tr>
<td>Week 2 to 4</td>
<td>Conclude performance evaluation and complete its deliverables</td>
</tr>
<tr>
<td>Week 5</td>
<td>Co-author conference paper</td>
</tr>
<tr>
<td>Week 6 to 8</td>
<td>Conduct programmability evaluation</td>
</tr>
<tr>
<td>Week 9</td>
<td>Conclude programmability evaluation and complete its deliverables</td>
</tr>
<tr>
<td>Week 10</td>
<td>Begin writing Thesis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Spring 12</strong></th>
<th><strong>Milestone</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1 to 8</td>
<td>Writing thesis / fine-tuning the MASS library/authoring more conference paper if necessary</td>
</tr>
<tr>
<td>Week 9 to 10</td>
<td>Complete thesis and prepare for thesis defense</td>
</tr>
</tbody>
</table>