Andrew VanKooten
August 23, 2016
Munehiro Fukuda
CSS 497

Final Report

Introduction

This document serves as my final report to document what I worked on for the Distributed Systems Laboratory on MASS Java and where to find the changes I made.

Project Scope

Ultimately, this project was to bring forward MASS Java into a state that is ready for release. To do this I had to accomplish four main tasks.

1. **MASS Java Verification with Test Programs**
   I was to test MASS Java by running the following test programs and update those test programs to the current version of MASS Java.
   a. Heat2D
   b. Wave2D
   c. RandomWalk
   d. Sugarscape
   e. Biological Network
   f. UWCA
2. **Debugger Verification**
   I was to integrate the MASS debugger into MASS Java and verify it to see if it still runs.
3. **AWS**
   I was to create multiple AWS EC2 instances and get MASS Java to run on those.
4. **Documentation**
   I was to create or update the following documents and upload them the dslab/MASS website.
   a. MASS Java Manual
   b. MASS Java Developers Guide
   c. MASS Java Debugger
   d. MASS Java AWS Setup Guide

Results

MASS Java Verification

Getting MASS Java running was difficult at first. This was partially due to my inexperience using maven and the fact I was working from older documentation. I spent a good amount of this time going through the source code and trying to understand MASS. From this process I was able to figure out how to update the test programs to get them to work with the current version. Also during this time, I helped identify which functions could be set to protected in MASS, Agents, and Places. Changes made to MASS and its applications were pushed to their repositories.

Heat2D

This test application could not be found in either the metis file server or the BitBucket repository, so this application was dropped from the requirements.

Wave2D

This test program is located in the **mass_java_appl** BitBucket repository under the **mass-wave2d** directory in the develop branch. To update this program, I had to modify the MASS function calls it made because the MASS functions had been changed. I also had to remove references to `callSome()` because that function was removed from MASS. Since those functions were used only for the GUI output, the output functions were moved from the MASS side to the application side.

I did most of the verification of MASS Java using this application. I was able to find some bugs by doing this and I was able to fix them. When first trying to run this application, I was trying to run it using the *machinefile.txt* file rather that the *nodes.xml* file. This was causing some issues due to the environment not being set up on the remote nodes correctly. This was not an issue when the *nodes.xml* was being used, so references to using *machinefile.txt* were removed.

There was also an issue encountered in the various `callAll()` functions for **Places** discovered when testing with this application. When null was being passed into the `argument` parameter, a NullPointerException was being thrown. To fix this, I added some null checks to those functions, but those functions may need a redesign in the future.

One issue came up with the use of the Messages class. `Places.exchangeAll()` was modified at one point to exclude the neighbors vector and that parameter was removed from the Messages constructor assuming that there was a corresponding constructor. There was no constructor for these arguments and Java was truncating an int into an Object which corresponding to a different type of data member to send. To fix this I added a new constructor in Messages that corresponded to the correct arguments.

RandomWalk

This test program is located in the **mass_java_appl** BitBucket repository under the **mass-randomwalk** directory in the develop branch. Modifications made to the Wave2D application also had to be made to this application.

This application was also an opportunity to test the **Agents** functionality in MASS. One issue was found in the `manageAll()` function when **Agents** were migrating between nodes. A ClassNotFoundException was being thrown in the ProcessAgentMigrationRequest in AgentBase. This issue was fixed when the application was packaged as a maven jar, so I will figure out how to package an application as a maven project and include instructions on how to do that in the documentation.

Sugarscape

This test program was located in the dslab account on the metis file server in the **Cherie** directory. I was able to mavenize this application easily and once it was mavenized, it ran without any issues.

Biological Network

The test program is maintained by Drew Anderson in his own BitBucket repository. These applications were updated and mavenized easily, and were able to be ran without large issue.

UWCA

This test program is maintained by Jason Woodring in a directory on the metis file server. This application is currently being updated and tested by Jason and will hopefully have it completed by August 31.

Debugger Verification

Initially, I was trying to get the old debugger to work but after I was put in contact with Nick, I was able to get his updated debugger. I added the required code to MASS and got rid of the old debugger code. I was able to get Nick's debugger to run with few modifications listed below.

- Replaced the OutputStream and InputStream to ObjectOutputStream and ObjectInputStream respectively. (Don't remember which file)
- Changed a couple declarations in "GUI/CellPanel.java" by adding final to them to make it compatible with Java 1.7.

These changes hopefully will not break the C++ functionality, but that part of the debugger is not tested. I put the current debugger code in a BitBucket repository in the mass_library_developers group repository called **mass_debugger**. I also put the C++ code that is needed to make the C++ version of MASS compatible with the debugger in a directory called C++ Code. There is also a directory for the word documents called Documentation.

AWS

I first started out this process by creating my own AWS account and trying to figure out how to create EC2 instances. I found the process not too difficult and the documentation was quite helpful. After creating a Ubuntu instance, I was able to put the MASS files on and run Wave2D and RandomWalk without problem. After receiving Chris' AWS setup guide I was able to figure how to create images of EC2 instances and create remote nodes to test MASS on.

The process to make MASS C++ connect to remote nodes did not work with MASS Java, so I had to figure out how to use SSH private keys with MASS Java. Fortunately, Matthew already had code to do this so I just integrated his code into the version I was working with and it worked flawlessly. I also tested this private key functionality on the uw1 320 lab machines and it worked there as well, so I am going to push this change to the main version of MASS and include instructions on how to create a private key in the documentation. I believe this solution is much better than having a plain text file with passwords in it for everybody to view.

I only tested MASS on the micro instance because they are free. This showed a slowdown in MASS but if a more powerful instance type is used performance is expected to increase. I created an instance with java, maven, and MASS installed and saved it as an image on the dslab account.

Documentation

MASS Java Manual

I could not find an editable version of this document so I transcribed the pdf version to a word document. As I went through the documentation, I added clearer language to descriptions to aid in the understanding of MASS. I also updated the functions listed to what they are in the current version of MASS.

In the future, it may be a better idea to remove the method details and instead list them in a public online location, similar to how the Javadoc comments are shown in the Jenkins repository. I have also uploaded this to the dslab/MASS website.

MASS Java Developers Guide

I wrote this guide similar to what was written by Kasey in his MASS C++ Developers Guide, but with several MASS Java specific editions. Much of the information in this guide is useful for the MASS Java Manual, so that will have to be added to there at one point or the Manual that this guide should be merged. I would like Matt or someone also as knowledgeable with MASS Java to look over it at some point and see if there are any errors. I have also uploaded this to the dslab/MASS website.

MASS Java Debugger Guide

I found the current MASS Java Debugger Manual more than sufficient, but I added information about the debugger to the MASS Java Manual.

MASS Java AWS Setup Guide

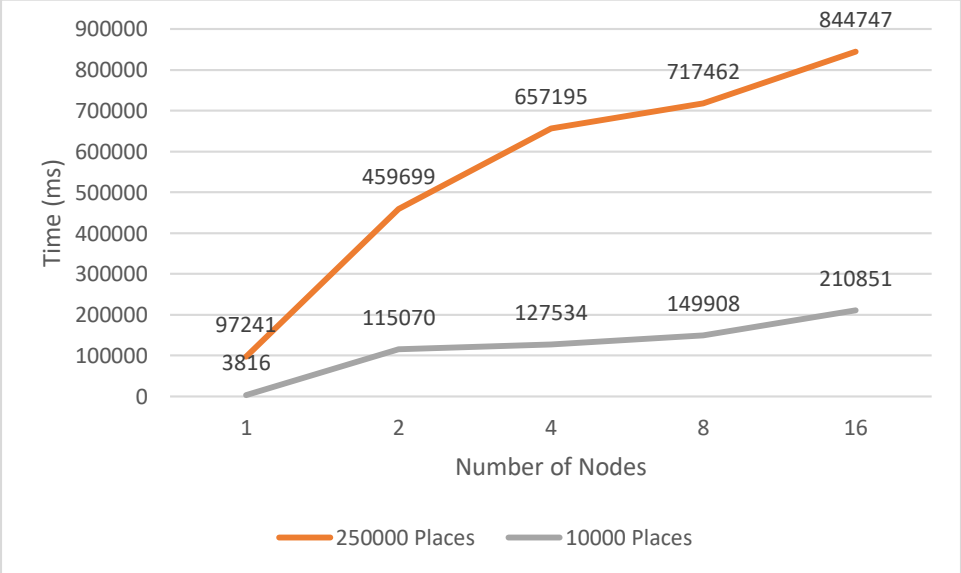
I followed Chris' AWS setup guide for this and tried to include step by step instructions on how to create AWS EC2 instances. I have also uploaded this to the dslab/MASS website.

Future Work

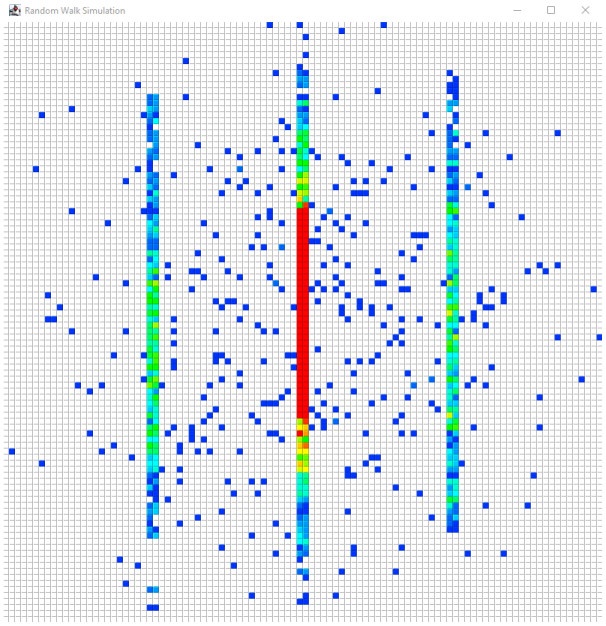
Found Bugs

Recently, I have found two bugs that I am not sure if I have time to fix.

- MASS seems to slow down on Wave2D when more nodes are added. This slowdown also appears in RandomWalk and Sugarscape. It look as exchangeAll() and init() are the functions casuing the speed up. This is shown in the graph below.



- Tests when running RandomWalk show the Agents grouping up near the borders of the Places. This error also shows up in Sugarscape as well. is visualized in the figure below.



- The AgentsBase constructor seems to do be generating the incorrect number of Agents. More tests will have to be done.
- The debugger will hang when using more than one computing node or more than one thread. I suspect this has something to do with the MASSBase.getPlaces() function.

Projects

In this section I will detail some potential projects or upgrades that could be made to MASS.

Message Class

This class contains multiple constructors for different types of messages to be sent to and from remote nodes. This caused an error when some of the functions were updated. I recommend that the Message class be broken up into a Message abstract class with the several derived classes for each type of message sent. The base class would have an enum for the type of message it is, similar to what it currently has and an accessor for that member. Each derived class would have its each own members and accessors for the data it needs to carry. The MProccess class would be able to convert the base Message to the message class it needed. The separate Message classes would then be organized into its own package within MASS Java.

Update callAll

I suggest that the Places and Agents classes have separate functions for returning output instead of just placing a null object array. Placing a null object array into the callAll would cause a NullPointerException, so I added a couple null checks in the code. This would also be simpler for writing applications.

Verify C++ Debugger

The C++ version of the debugger needs to be verified and the code for the C++ code needs to be updated to what is found in the C++ code directory in the **mass_debugger** repository.

Finish Debugger Functionality

Nick's debugger manual mentions several features that have yet to be implemented in his debugger.