CSS499 - Mass Term Report

John Emau

Spring 2011

How to Use Mass

Decide to use either the Object or Primitive implementation of MASS. The primitive is faster but offers less of an Object-Oriented Paradigm.

Create a class to inherit either the Place or PlaceHandler class, follow the requirements.

You will need a copy of MASS and JSCH jar files. Place all files into jar files and in the same directory. Either put this directory on every machine in the network or on a network shared folder.

From process 0 you may now run your application, the first 4 parameters passed to MASS are required and used for MASS.init, they should either be passed into MASS.init or you can hardcode them into your application, but we recommend against this for security reasons.

Assuming you don't hard code your application here is a breakdown of

```
Java <ClassPath to Jar Files> <Application> <User Name> <Password> <MachineFile>
<Location of Mass Jar on Remote Nodes> <Optional Application Parameters>
```

Here is an example of the command being run from the DSLAB account using an application for performance testing CallALL called CALLALLPE . Not the option to set the max memory used by the application -Xmx

```
java -Xmx1g -cp './*' CallAllPE dslab ds1ab-302 mfile1.txt './MP/Mass.jar'
```

Developer Nodes

General Notes

Mass is in its early development stage. There are still a lot of bugs and untested areas. For example we have not tested clusters of size greater than 16 nodes. We know that 8 node clusters work and 16 node clusters don't work (reasons still unknown).

Please keep this in mind while working with MASS and Developing for MASS. The following notes are to help anyone continuing development of MASS.

Helpers

MASS has a number of helper class that are only used internally. These classes are all java package-private so a user should not be able to create an instance of them (a side from mNodeProcess).

Utilities class is used by the UWAgent project and was shared with the SensorGird team. It encapsulates work done for remote process launching using the JSCH Library. In the future as UWAgent and MASS merge this file should be shared across libraries.

Enums class is used to organize all the enums used throught the MASS Library. Before they were scattered across classes and now this consolidation makes for better readability.

The mNode class is a communication wrapper for all interprocess communication. Logic for this class is basically to setup and close all connections. Other helpers may use object streams that are managed by a node.

The PlacesNode is a helper for the Places classes. This class is just a wrapper for the mNodes. Because each places may have a slightly different view of the total network (based on the number of place elements).

The Mail class is a helper for the Places class and is the wrapper for messages sent between processes in the cluster.

mNodeProcess Command Parsing

It's important to know how the mNodeProcess parses its commands if you want to add any new commands across the places class.

1. Command Type
2. Places Handle
3. Function Id
4. Command
5. Any Specific Command Parameters

Note that this is slightly different for other command types such as Agent commands and 'None' (maintenance) commands.

You may notice that in the Places class these commands are written out in the node.writeObject method, this is because the node.writeObject method can take in any number of objects and send them to the remote node in that order. So you can create a message simply by providing your objects in the proper order. For example:

```
for ( PlacesNode node : placesNodes ) {
    node.writeObject(
            Enums.Type.Place,
            handle, fid,
            Enums.PlacesCommand.CallAllVoidObject,
            argument );           // Send Cmd, FunctId, Argument
}
```

Places vs. mNodePlaces

Theis a slight difference between the Places object that exists on Process 0 vs all the other Processes. This difference is essentially the remote process management overhead done by Process 0. To account for this only slight difference the mNodePlaces class was created and inherits from the original Places class. The mNodePlaces class interacts with the mNodeProcess as it's driver and does no communication, you can think of it like the "pure" form of the Places class without any splitting among multiple processes. A similar class will need to be created for the Agents class.

ToDo's/Known Bugs

Here are a list of open bugs/things that need to be added to the current version of MASS

Add Tolerance

Currently if any communication process, or most setup processes fail the system exits. This is useful because it does not leave process running on a remote node if Process 0 crashes. However it is not ideal if there is temporary lag in a connection or if a node can be reestablished after a crash.

Connection tolerance should be added to the mNode class, as that class encapsulates communication between processes.

Add More Detailed Exception and Error Message

Ideally a separate MASS Exception class would be created that provide detail based on the kind of MASS exception. This would allow for MASS to handle crashes differently based on certain failures and is cleaner than using simple text output.

CallAll and CallSome Ambiguity

These methods have examples used in Wave2DMASS and Wave2DPlaceHandler that do not provide an argument. This is a problem because an argument is required by the Place and PlaceHandler argument. In the current implementation a default value is provided internally and the Array Type is determined. However after more consideration these should be removed (not the array type).

We should enforce the user provides an argument, even if the argument is not used in the users implementation.

ExchangeHelper Thread Reset Connection

The writeObject method in the object output stream writes duplicate objects of arrays, even if the data in the arrays have been changed. This causes no new data to be sent. To fix this we should be able to use "writeUnshared/ReadUnshared" however in testing we found that not to be the case. The only fix we found was to reset the connection before every write, this is a costly operation that should be fixed.