MASS Java Annotations

Work performed, Next Steps

Matthew Sell

Spring 2020

Introduction

My work performed on MASS during Spring 2020 focused on creating an implementation of a Global Logical Clock:

1. Added “Clocked” annotation for Agent
2. Implemented Global Logical Clock that triggers “Clocked” agent methods

There were also project management related tasks performed since the end of Winter 2020:

1. Refactored MASS Java Applications repository to remove child project disconnects and inconsistencies
2. (Re) Fixed most MASS-Java Core unit tests
3. Tried out Google Analytics for obtaining project download statistics (did not result in any usable information, though)
4. Assisted students with building/running MASS-Java, branching and merging, and repository creation

Global Logical Clock

For Spring Quarter 2020 my main accomplishment was the creation of a Global Logical Clock for MASS-Java. This clock, configured and driven through annotated methods, should simplify the development of applications where time is simulated or measured.

There is only one annotation used with the clock, “@Clocked”. This annotation is used to specify an agent method to be invoked every clock cycle (default), or the behavior can be modified though the use of two parameters:

- “onMultiplesOf”: Method will be executed when the clock value reaches multiples of these values
- “onValuesOf”: Method will be executed when the clock value reaches these specific values

Parameter values are provided by means of a list:

@Clocked(onValuesOf = { 3, 7 })

@Clocked(onMultiplesOf = { 5, 8 })

The clock value is represented by a “long”, which will “roll over” to zero if allowed to continue past a value of 9,223,372,036,854,775,807 (2^63 – 1). The “long” datatype was chosen so that it would be extremely unlikely for this value to be reached during normal execution.

In addition to the annotation, two methods are provided (in “MASS”) to working with the clock:

- “resetClock”: Resets the clock back to a value of zero
- “getClock”: For obtaining the current clock value

Three methods were added to “Agent” as well:

- “sleep”: Pause executions of Clocked methods
- “resume”: Resume execution of Clocked methods
- “sleepUntil”: Pause execution of Clocked methods until the clock reaches the specified value

It should be noted that while an Agent is “sleeping”, no clock-driven methods will be invoked. If the clock reaches a value where the method would normally be invoked, but the Agent is “sleeping”, the method will not be called. Once the Agent “resumes”, no missed executions of clocked methods will be rescheduled – those events were missed.

The clock works as follows:

1. One or more Agent methods are annotated with “Clocked”.
2. Application starts and MASS is initialized. At this point the clock starts with a value of zero.
3. For every call to “AgentsBase.manageAll” the clock value increments
4. When the clock is set to a new value (either by increment or “fast forward”), several actions occur:
    a. Each Agent on the node is evaluated to see if it has any “Clocked” methods that need to be executed. If there are, the method is queued for execution
    b. For each Agent, the next clock cycle value where an event will occur is determined, and the lowest value (soonest) is remembered
5. Clocked methods scheduled for execution are invoked, in parallel
6. The node returns the next clock cycle where an event will occur (upon completion of “AGENTS_MIGRATION_REMOTE_REQUEST”)
7. The soonest value of the clock where an event will occur from all nodes is compared against the current clock value. If a “fast forward” can be performed, then a “CLOCK_SET_VALUE” message is sent to all nodes setting all clocks to the next value.

Several classes were added to support implementation of the Global Logical Clock:

- GlobalLogicalClock [Interface]
- SimpleGlobalClock [Implementation]
- Clocked [Annotation]

Source Code

My work is located in the MASS-Java Core and Applications repositories, under the branch name of “msell-initial-annotations”. I have been keeping this branch up-to-date with the “develop” branch, but my work is isolated until we are comfortable with the changes. I will be happy to merge this work into the “develop” branch once we feel it is ready for release (see “Next Steps” below).

Next Steps

It has become clear that students experience difficulties in figuring out where to start when it comes to development of MASS-Java applications. Mostly there is confusion in regards to adding a new application to the current Maven structure of projects and with the initial setup of multiple nodes in a

cluster. This is occurring because of a lack of documentation and not a failing of the students; because of this I believe that we need new or revised documentation:

- Frequently Asked Questions – Troubleshooting Guide
- Networked Cluster Howto
- Application Development Quick Start

In addition to these guides, I need to create a proper developer's guide for using annotations with MASS.

With the event-driven annotations (Fall 2019 and Winter 2020) and the Global Logical Clock (Spring 2020), a series of benchmarks and sample applications should be created to further test the implementation of the clock and exercise Hazelcast more in a multi-node environment.

Volunteering

I will be studying at Bothell for at least another whole school year, and there are some loose ends that I work on to prevent leaving things open:

- Documentation (previously mentioned)
- Sample Applications for Annotations
- More extensive multi-node testing

I will also make myself available for assistance with project management tasks (such as branching/merging and helping students with clustering MASS).