JavaLife

Mark Shanoudy

CS 1699 - DELIVERABLE 4: Performance Testing Conway's Game of Life

Profiling

In order to have as accurate results as possible, the application was configured as follows across all runs: JavaLife 5 1 10 500000. The application was profiled using Java VisualVM. Once JavaLife began running, it selected from the list of running processes in VisualVM. The CPU profiler settings were changed from default to include profiling the classes Cell, World, and State — in addition to JavaLife. The CPU profiler measured that the method World.toString() consumed the highest percent of processing time with an arithmetic mean of 76.5% of the total time profiled across three runs. World.toString() was therefore determined to be the method which required refactoring.

Refactoring

World.toString() was refactored in the following manner. The String object, *toReturn*, was replaced by the StringBuilder object *sb*. The reason for this was because the String representing the current World was originally being built using a series of concatenations (i.e. *toReturn* += “some string”). This manner of building a String is both slow and resource intensive because a new String object must be created every time a concatenation occurs. In light of this, a StringBuilder was used due to the performance advantage of StringBuilder.append() over concatenation when dealing with dynamic string building.

Results

After refactoring World.toString(), the CPU profiler measured that World.toString() was no longer consuming the highest percent of processing time with an arithmetic mean of 30.6% of the total time profiled across three runs. This represents an average approximate decrease of 60% processing time of World.toString() due to the refactoring.







