**MonkeySim**

CS 1632 – DELIVERABLE 4:

Performance Testing Using VisualVM

Author:

Anthony Poerio ([adp59@pitt.edu](mailto:adp59@pitt.edu))

Brandon Hedges ([bjh6@pitt.edu](mailto:bjh6@pitt.edu))

Github URL:

<https://github.com/adpoe/MonkeySim>

For our fourth deliverable in CS1632, we were tasked with performance testing the legacy code for a (purposely) inefficient Monkey Simulator, named MonkeySim. Specifically, our goal was to use the profiler called VisualVM to identify the most problematic and inefficient functions in our code base, refactor them, and write tests to ensure that we have not affected the core functionality that legacy users may depend on.

After profiling the MonkeySim program using VisualVM, the first thing noticed was that generateId() was taking up 98% of the time on CPU. (And sometimes even more!) With this in mind, we traced it generateId() with a debugger, and we were able to determine a simpler and equivalent expression that did not necessitate the long loop found in that method. Essentially, we just need to add 223492 to whatever the Monkey’s “\_monkeyNum” is.

Next, we noticed that stringifyResults() was taking a lot of time. With some refactoring, I was able to avoid iterating 50,000 times each time this function was called, by bypassing the Header loop, without altering any core functionality, at least as far as our tests (or the user-facing output) were able to detect.

With these two methods refactored, the program was still running just a little bit slow. Because of this, we ran the profiler again to see what method is taking up the most time now that the most obvious inefficient methods had been refactored.

What we found was that getFirstMonkey() was taking up the most time at this point. Again, with the help of a debugger, we were able to simplify the expression and avoid the long loop that was being invoked each time this method was called.

Results

With these three methods refactored, we were able to achieve the following results.

**Before Refactor (3 runs):**

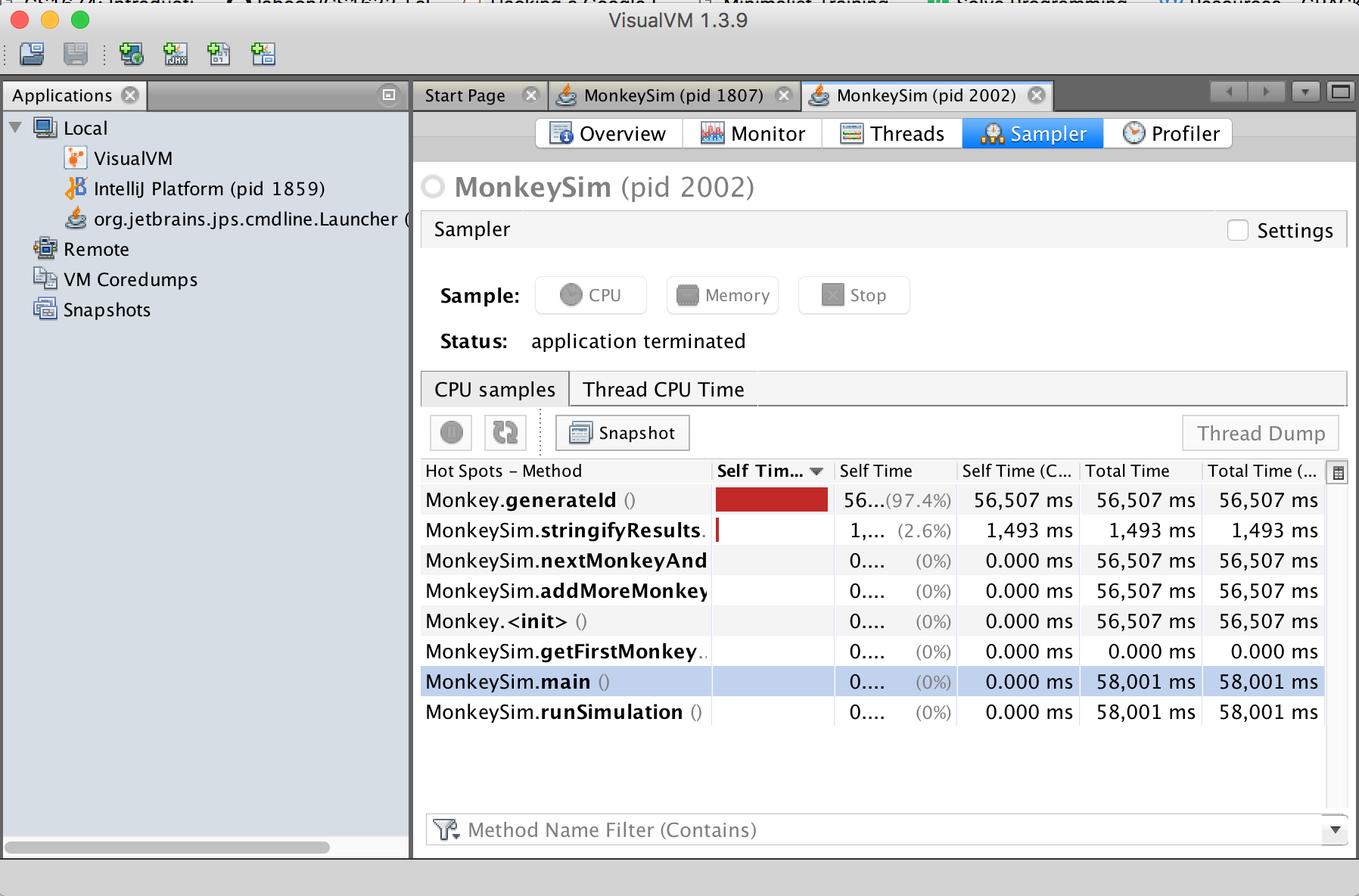
* **Run 1 Time:** 12.43s user 0.36s system 100% cpu 12.710 total
* **Run 2 Time:** 13.72s user 0.43s system 100% cpu 14.038 total
* **Run 3 Time:** 13.04s user 0.38s system 100% cpu 13.298 total
* **Mean time (3 runs):** 13.349s total
* **Median time (3 runs):** 13.298s total

**After Refactor (3 runs):**

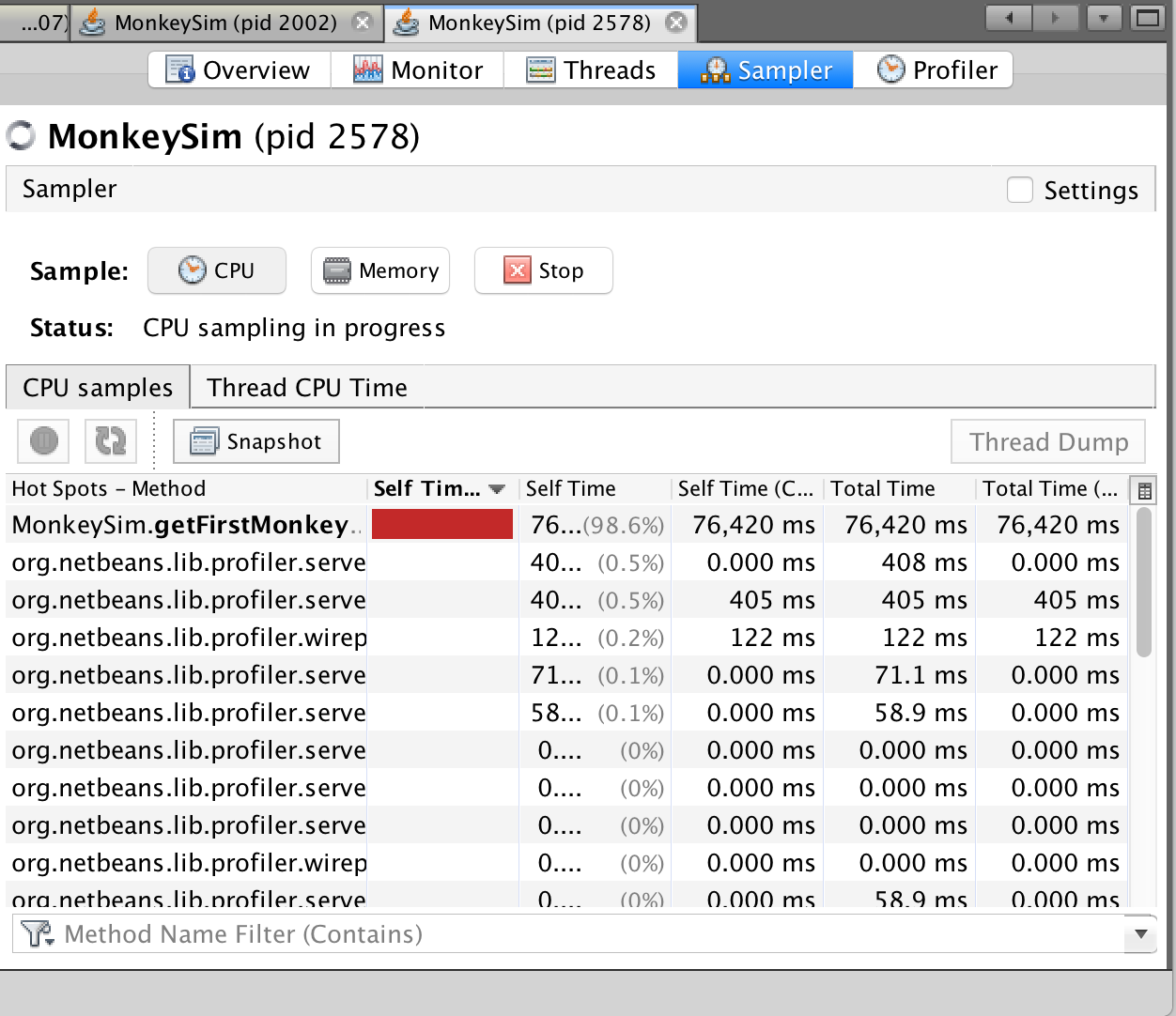
* **Run 1 Time:** 0.10s user 0.03s system 95% cpu 0.129 total
* **Run 2 Time:** 0.10s user 0.03s system 98% cpu 0.125 total
* **Run 3 Time:** 0.10s user 0.03s system 98% cpu 0.123 total
* **Mean time (3 runs):** 0.126s total
* **Median time (3 runs):** 0.125s total

Screenshots

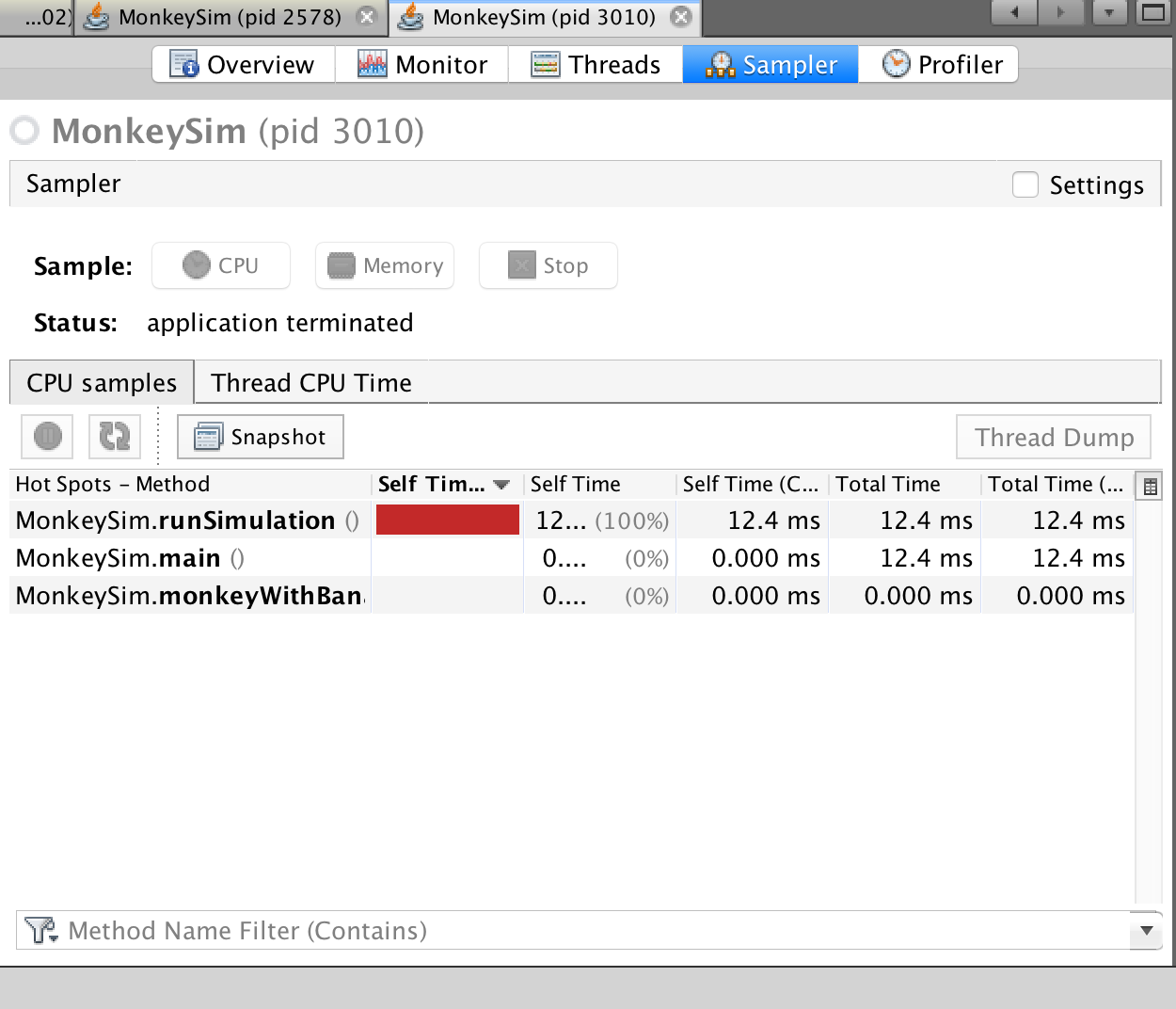
**VisualVM Before 🡪** we decided to refactor generateId() and stringifResults()



**VisualVM Before 🡪** Next, we refactored getFirstMonkey()



**VisualVM After 🡪** After refactor, we found that runSimulation() is where most time is spent during each run of the program. This is expected, and the majority of inefficiencies have been removed.



**TODO**

Pinning tests 🡪 need them for all three methods updated.

1. generateId()
2. getFirstMonkey()
3. stringifyResults()

Pinning tests are when we write a test to confirm the BEFORE behavior of a function (which is not necessarily the CORRECT behavior) matches the AFTER behavior. We must do this because it ensures that we don’t accidentally change something which is intended, but don’t understand the \_\_WHY\_\_ in legacy code.