

MonkeySim

CS 1632 – DELIVERABLE 4: Performance Testing Using VisualVM

Author:

Anthony Poerio (adp59@pitt.edu)

Brandon Hedges (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 **Monkey.generateId()** was taking up 98% of the time on CPU. (And sometimes even more!) With this in mind, we traced 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 **MonkeySim.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 **MonkeySim.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 `stringifyResults()`

VisualVM 1.3.9

Applications: Local, Remote, VM CoreDumps, Snapshots

Start Page: MonkeySim (pid 1807), MonkeySim (pid 2002)

Overview, Monitor, Threads, **Sampler**, Profiler

MonkeySim (pid 2002)

Sampler ☐ Settings

Sample: ☒ CPU ☐ Memory

Status: application terminated

CPU samples Thread CPU Time

Hot Spots - Method	Self Time	Self Time (C...	Total Time	Total Time (...)
Monkey. generateId ()	56... (97.4%)	56,507 ms	56,507 ms	56,507 ms
MonkeySim. stringifyResults .	1,... (2.6%)	1,493 ms	1,493 ms	1,493 ms
MonkeySim. nextMonkeyAnd	0.... (0%)	0.000 ms	56,507 ms	56,507 ms
MonkeySim. addMoreMonkey	0.... (0%)	0.000 ms	56,507 ms	56,507 ms
Monkey.<init> ()	0.... (0%)	0.000 ms	56,507 ms	56,507 ms
MonkeySim. getFirstMonkey..	0.... (0%)	0.000 ms	0.000 ms	0.000 ms
MonkeySim. main ()	0.... (0%)	0.000 ms	58,001 ms	58,001 ms
MonkeySim. runSimulation ()	0.... (0%)	0.000 ms	58,001 ms	58,001 ms

Method Name Filter (Contains)

VisualVM Before → Next, we refactored `getFirstMonkey()`

MonkeySim (pid 2578)

Sampler ☐ Settings

Sample: CPU Memory Stop

Status: CPU sampling in progress

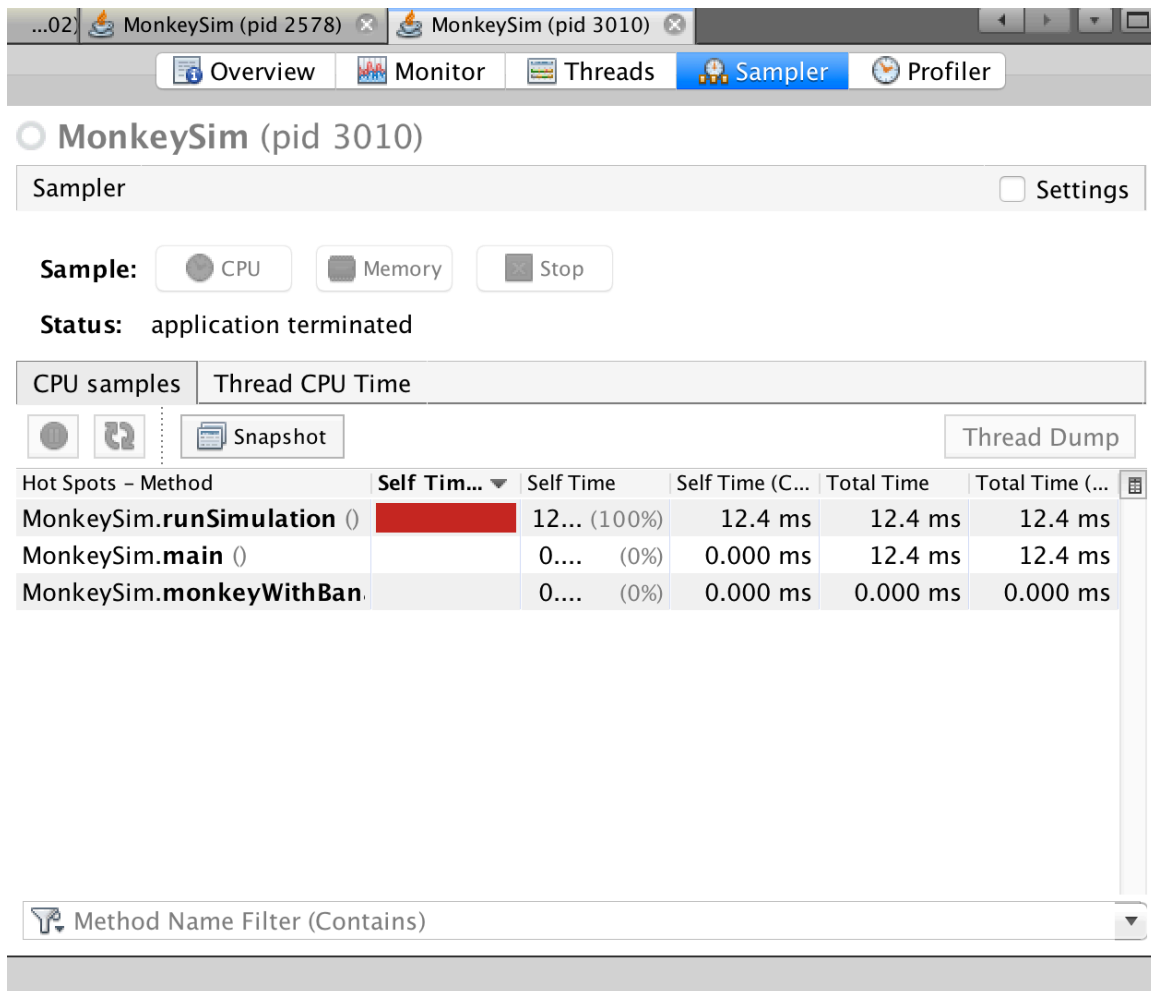
CPU samples Thread CPU Time

Snapshot Thread Dump

Hot Spots - Method	Self Tim...	Self Time	Self Time (C...	Total Time	Total Time (...)
MonkeySim.getFirstMonkey..		76...(98.6%)	76,420 ms	76,420 ms	76,420 ms
org.netbeans.lib.profiler.serve		40... (0.5%)	0.000 ms	408 ms	0.000 ms
org.netbeans.lib.profiler.serve		40... (0.5%)	405 ms	405 ms	405 ms
org.netbeans.lib.profiler.wirep		12... (0.2%)	122 ms	122 ms	122 ms
org.netbeans.lib.profiler.serve		71... (0.1%)	0.000 ms	71.1 ms	0.000 ms
org.netbeans.lib.profiler.serve		58... (0.1%)	0.000 ms	58.9 ms	0.000 ms
org.netbeans.lib.profiler.serve		0.... (0%)	0.000 ms	0.000 ms	0.000 ms
org.netbeans.lib.profiler.serve		0.... (0%)	0.000 ms	0.000 ms	0.000 ms
org.netbeans.lib.profiler.serve		0.... (0%)	0.000 ms	0.000 ms	0.000 ms
org.netbeans.lib.profiler.wirep		0.... (0%)	0.000 ms	0.000 ms	0.000 ms
org.netbeans.lib.profiler.serve		0.... (0%)	0.000 ms	58.9 ms	0.000 ms

Method Name Filter (Contains)

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.



MonkeySim (pid 3010)

Sampler ☐ Settings

Sample: ☒ CPU ☐ Memory ☐ Stop

Status: application terminated

CPU samples Thread CPU Time

Hot Spots - Method	Self Tim...	Self Time	Self Time (C...	Total Time	Total Time (...)
MonkeySim.runSimulation ()	<div></div>	12... (100%)	12.4 ms	12.4 ms	12.4 ms
MonkeySim.main ()		0.... (0%)	0.000 ms	12.4 ms	12.4 ms
MonkeySim.monkeyWithBan		0.... (0%)	0.000 ms	0.000 ms	0.000 ms

Method Name Filter (Contains)