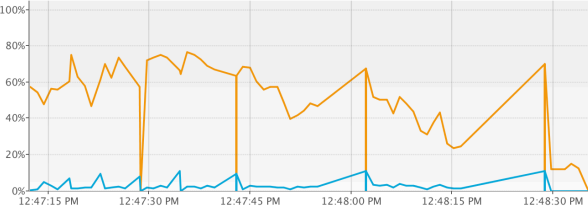
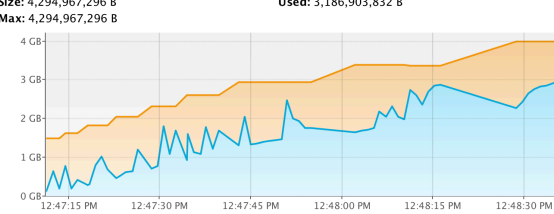
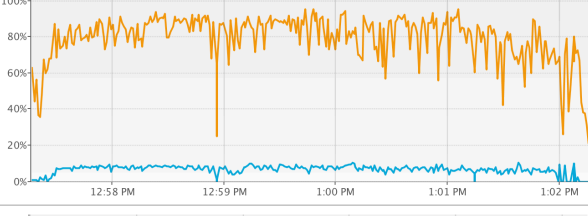

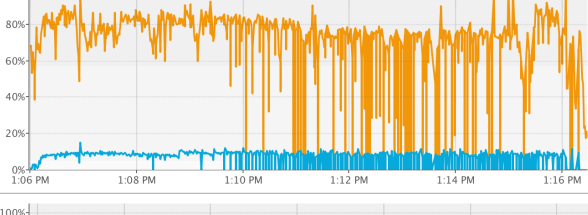
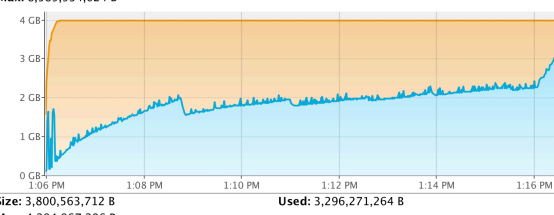
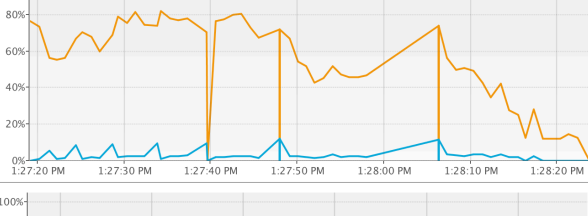
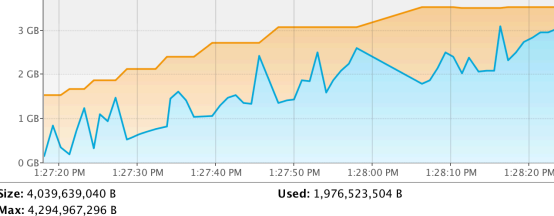
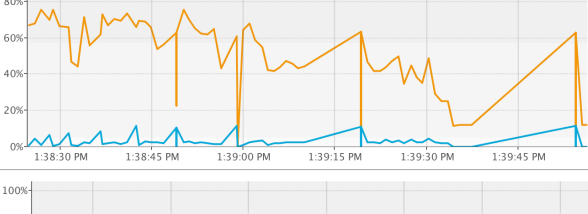
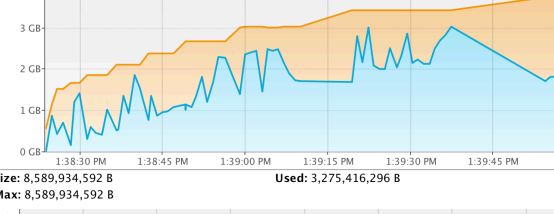
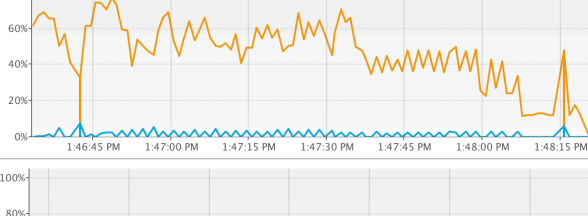
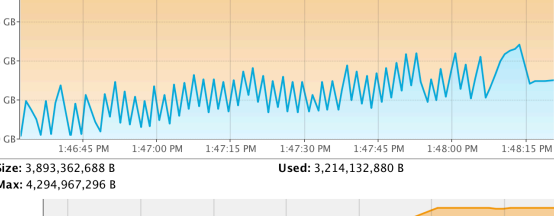
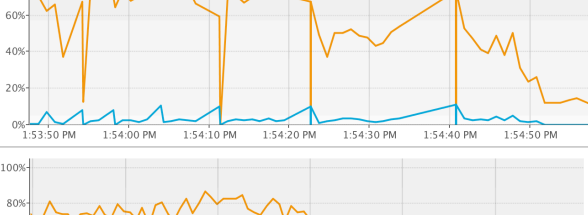
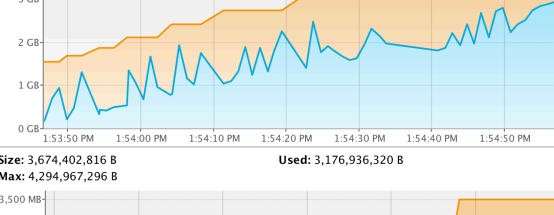
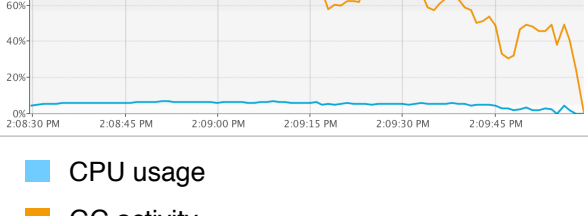
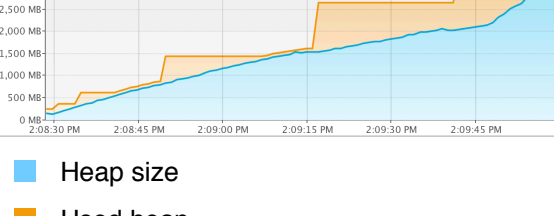






Table 1

Run	JVM arguments	Run time	CPU	Memory	Remarks			
1		1m 27s		 Size: 4,294,967,296 B Max: 4,294,967,296 B Used: 3,186,903,832 B	Initial run to establish a base case			
2	-XX:+UseG1GC	5m 3s		 Size: 4,294,967,328 B Max: 8,589,934,624 B Used: 3,167,237,736 B	Much worse in terms of, well, everything, but has interesting sidekick argument that worths a shot			
3	-XX:+UseG1GC -XX:+UseStringDeduplication	10m 33s		 Size: 4,294,967,328 B Max: 8,589,934,624 B Used: 3,330,815,824 B	Even worse, G1GC option just doesn't work for this task			
4	-XX:+AggressiveOpts	1m 8s		 Size: 3,800,563,712 B Max: 4,294,967,296 B Used: 3,296,271,264 B	Significantly better than the base case, definitely worth keeping in a final setup			
5	-XX:-BackgroundCompilation	1m 35s		 Size: 4,039,639,040 B Max: 4,294,967,296 B Used: 1,976,523,504 B	Slightly slower than base case, but consumes less memory			
6	-XX:+AggressiveHeap	1m 51s		 Size: 8,589,934,592 B Max: 8,589,934,592 B Used: 3,275,416,296 B	Slow memory hog			
7	-XX:+UseParallelGC	1m 14s		 Size: 3,893,362,688 B Max: 4,294,967,296 B Used: 3,214,132,880 B	Faster than the base case, worth keeping			
8	-XX:+UseConcMarkSweepGC	1m 35s		 Size: 3,674,402,816 B Max: 4,294,967,296 B Used: 3,176,936,320 B	A bit slow, but very nice memory footprint			
Legend			 CPU usage  GC activity	 Heap size  Used heap				
	All of memory and threading options combinations are impossible to test for human being, so I've delegated this task to a fellow Python script, you can see its output in the Appendix No more nice pictures, though							