Java	Single Thread (ms)	Multi Thread No Locks (ms)	Multi Thread w/ Locks (ms)
PC	6	120	444
Genuse30	6	113	488

PC:

Time Multithread no locks runs 1900% slower than single thread
Time Multi-Threads with locks (10) runs 270% slower than multi-thread no locks

Genuse30:

Time Multithread no locks runs 1783% slower than single thread
Time Multi-Threads with locks (10) runs 331% faster than multi-thread no locks

Python	Single Thread (ms)	Multi Thread w/ Locks (avg of 10)(ms)
PC	7467	32752
Genuse	7231	44010

PC:

Time Multi-Threads with locks (10) runs 338% slower than single thread

Genuse30:

Time Multi-Threads with locks (10) runs 508% slower than single thread

For timing we recorded the system time at the start of each process and the end and subtracted them. Note for my python program to prevent the results of the multi and single thread from affecting each other, I ran two separate programs, one with multithreaded code and one with single threaded code. In this way I could use the built in time command to accurately time each code. The single thread operation constantly beats both multithread operations due to the raw system overhead in both Java and Python to create multiple threads. The single thread code is able to execute in full before either language can finish creating and running the other threads. The multithreaded code with locks is slower because the operating system is not able to interrupt the threads will they work on global data due to the locks. This avoids race conditions but ultimately slows down the code.