Comparing Elixir and Java Performance for Project 1 Problem statement over different number of Threads.

1. Problem Definition:

An interesting problem in arithmetic with deep implications to elliptic curve theory is the problem of finding perfect squares that are sums of consecutive squares. A classic example is the Pythagorean identity:

$$3^2 + 4^2 = 5^2 \text{ ----------(1)}$$

that reveals that the sum of squares of 3; 4 is itself a square. A more interesting example is Lucas` Square Pyramid :

$$1^2 + 2^2 + ::: + 24^2 = 70^2 \text{ ----------(2)}$$

In both of these examples, sums of squares of consecutive integers form the square of another integer.

2. Inputs:

Last Number to search for perfect square sequence N = 100000000

Window size of Perfect Square sequence k = 24

Different Number of Threads considered: [1,2,4,8,16,32,64,128,256,512,1024]

3. Code:

i. Elixir :

I have modified to my Elixir code to take number of threads as additional Input. Code submitted for project 1 is modified to handle bigger problems for each actor.

Github URL of diff for code changed:
https://github.com/srgothi92/DOSPrj1/commit/664dd78d3800c18e547e08b44ca8154aa2675ee5

Project Github URL: https://github.com/srgothi92/DOSPrj1

Code is also zipped along with this document.

ii. Java

Wrote a new code in java with same logic as in elixir but using threads class. A variant of Producer Consumer pattern is used. Instead of producing and waiting to be consumed I producing all a once and consuming when all threads have finished.

Project github URL: https://github.com/srgothi92/DOSPrj1Java

Code is also zipped along with this document.

4. Performance measurements:

All the measurements were done on same machines with 2 cores Intel I3. Alternate measurements were taken from each code over each thread to avoid CPU state bias at that point of time.

No of Threads	N	K	Elixir (s)	Java (s)
1	100000000	24	45.224000	11.523887249
2	100000000	24	27.597000	8.223046662
4	100000000	24	24.040000	6.594456930
8	100000000	24	22.682000	5.019929070
16	100000000	24	24.196000	5.335440201
32	100000000	24	24.680000	5.226319210
64	100000000	24	22.870000	5.756379563
128	100000000	24	23.010000	6.334044222
256	100000000	24	22.589000	6.334044222
512	100000000	24	22.792000	5.578921648
1024	100000000	24	22.979000	6.148460857
2048	100000000	24	22.729000	8.648238797
4096	100000000	24	22.729000	11.412636022
8192	100000000	24	22.963000	7.784233905
10000	100000000	24	25.131000	6.524431458
100000	100000000	24	27.237000	12.556664595
1000000	100000000	24	60.123000	112.338334275

Plotting Line For both Java and Elixir.