COP5615 – Project 1

Harkirat Singh Lamba, Srikanth Bharadwaj

September 2018

1 Logistics

1.1 Group Members

1. Harkirat Singh Lamba, 1968-9119
2. Srikanth Bharadwaj, 6347-8395

1.2 How to run

1. Unzip the file harkirat_srikanth.zip in a directory and navigate to the directory.
2. Run ***cd harkirat_srikanth*** on your terminal to navigate into project folder.
3. Run the mix command as follows to execute the project.
   mix runner N k

   Ex: ***mix runner 100 24***

2 Implementation Details

2.1 Algorithm

Given a starting number ‘i’, the naive brute force approach is to calculate square of all number starting from ‘i’ till ‘k’, summing them up and then checking if the result is a perfect square. This logic is slow as for ‘i+1’ it involves computing sub-problems which have already been computed for number ‘i’.

We have optimized the solution.

Sum of squares of k natural numbers starting from 1:

$$1^2 + 2^2 + ... + k^2 = \frac{k(k+1)(2k+1)}{6} - (0^2)$$

Sum of square of k natural numbers starting from 2:

$$2^2 + 3^2 + ... + (k+1)^2 = \frac{(k+1)(k+2)(2k+3)}{6} - (1^2)$$

Sum of square of k natural numbers starting from 3:

$$3^2 + 4^2 + ... + (k+2)^2 = \frac{(k+2)(k+3)(2k+5)}{6} - (1^2 + 2^2)$$

The term which is getting subtracted can be derived from the sum of squares formula.
Generalizing the above to get:

Sum of square of k natural numbers starting from i:

$$i^2 + (i+1)^2 + ... + (k+i-1)^2 = \frac{(k+i-1)(k+i)(2k+2i-1)}{6} - \frac{(i-1)(i)(2i-1)}{6}$$

$$\forall i \in \{1, .., n\}$$

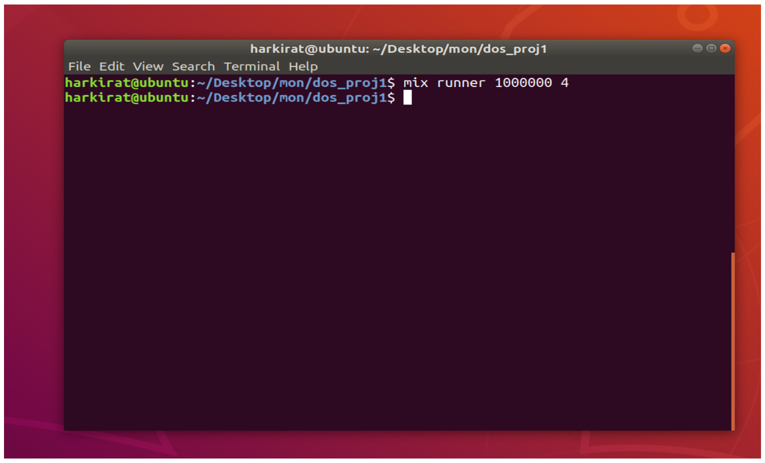
The above formula gives as an optimized solution to the problem. Instead of calculating squares of k numbers for each iteration, we are using the multiplication operation fixed number of times, which is a constant irrespective of the size of k.

2.2 Size of the work unit

In a single request from the boss, each worker gets assigned a number between one and n for which it will compute the above formula.

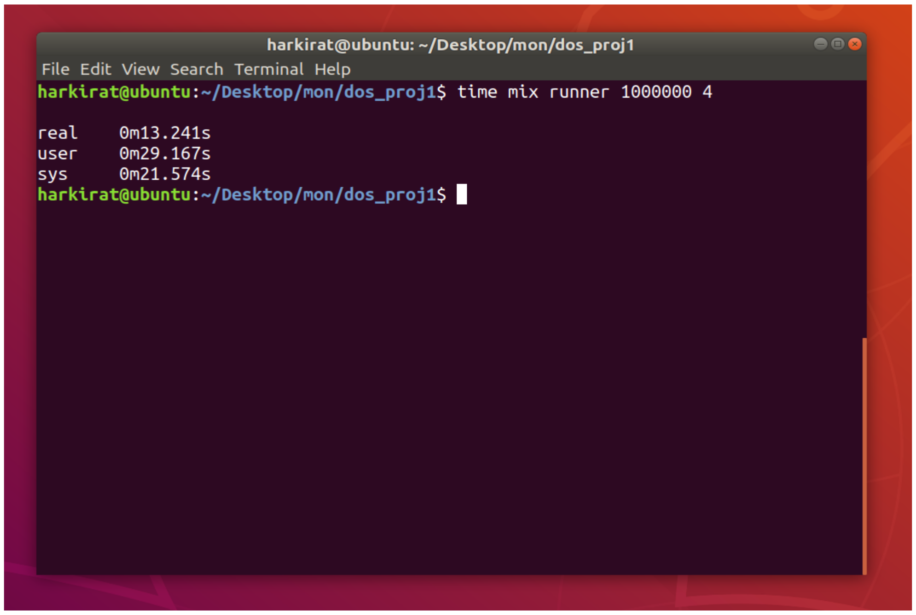
We approached this work size by doing an experiment. We incrementally spawned workers and calculated the time. For sample size of n = 1000, k = 24, we observed as the number of workers increased the overall time taken to compute results decreased. However, there was little difference in the total time after 100 workers. Since, in Elixir actors have little overhead and in our optimized solution there is no state preservation required, we are spawning 'n' instances of the worker, each solving one sub-problem.

2.3 Result of running n=1000000 k=4


harkirat@ubuntu: ~/Desktop/mon/dos_proj1
File Edit View Search Terminal Help

```
harkirat@ubuntu:~/Desktop/mon/dos_proj1$ mix runner 1000000 4
harkirat@ubuntu:~/Desktop/mon/dos_proj1$
```

2.4 CPU Utilization: Ratio of CPU time to REAL TIME


harkirat@ubuntu: ~/Desktop/mon/dos_proj1
File Edit View Search Terminal Help

```
harkirat@ubuntu:~/Desktop/mon/dos_proj1$ time mix runner 1000000 4

real    0m13.241s
user    0m29.167s
sys     0m21.574s
harkirat@ubuntu:~/Desktop/mon/dos_proj1$
```

Result of running the program with n=1000000 and k=4 on **4-core Intel i7 processor**.

Total CPU Time = 29.167s + 21.574s = 50.741s
Total Real Time = 13.241s
The ratio of CPU to Real Time = **3.832**

2.5 Largest problem we solved

Since we optimized our solution, the running time of the algorithm only depends on the input 'N'. The input 'k' doesn't affect our running time and is treated as a constant.

Hence, the largest problem we solved is N=100000000 (100 million) with k = 24.
Note: Since we optimized our solution, it is independent of the value of k.

The output is saved in the file named **output_max_on_one_sys.txt** in harkirat_srikanth directory.

3 Bonus

3.1 Machine Setup and How to run

- On remote machine

  1. Unzip the file harkirat_srikanth.zip in a directory and navigate to that directory.
  2. Enter the following commands to navigate to tasks directory.
     cd harkirat_srikanth/lib/mix/tasks

3. Enter the following command to set alias name and cookie for authentication between multiple systems:
   iex –name alias-name@ip_address –cookie niche –erl "+P 100000000"
   Ex: iex –name spiderman@10.136.49.119 –cookie niche –erl "+P 100000000"
4. Run iex shell by running following:
   iex
5. Run the following command to load the core logic module in the remote system
   c("remote_boss.ex")
   c("perfect_square_computer.ex")

- On local machine

1. Unzip the file harkirat_srikanth.zip in a directory and navigate to that directory.
2. Go to harkirat_srikanth - lib - mix - tasks.
3. Open distributed_boss.ex in Notepad++ (or any other editor) to specify the remote_machines.
4. Enter the remote machine to be connected in the remote_machines list (on line number 7) in the format shown in the file.
5. Run ***cd harkirat_srikanth*** on your terminal to navigate into project folder.
6. Run the following command to execute the program.
   time elixir –erl "+P 100000000" -S mix distributed.runner N k –no-halt
   Ex: time elixir –erl "+P 100000000" -S mix distributed.runner 100000000 20 –no-halt
7. In case the above command is not executed, use the alternate command:
   iex -S mix distributed.runner N k
   Ex: iex -S mix distributed.runner 100000000 20

3.2 Output of 100000000 20

The output of N = 100000000 k = 20 was calculated using 4 machines and saved in the file named **output_multiple_machines.txt** in harkirat_srikanth directory.