DISTRIBUTED OPERATING SYSTEMS

COP5612 – Fall 2020

PROJECT: 1

Members:

Name: Nikhilesh Reddy Tummala

UFID: 8350 - 1593

Email: tummalanikhilesh@ufl.edu

Name: Sayed Mohammed Afif

UFID: 4841 - 0855

Email: sayedafif@ufl.edu

Report:

Instructions to Execute the Program: In command line, type the following:
dotnet fsi "--langversion:preview" Project-1.fsx 1000000 4
Where Project-1.fsx is the file name and 1000000 4 are the input parameters N and k respectively.

The Program coded in F# works in the following manner:
For the two input parameters given to the program from the command line (Let's say N and k where N is the upper limit and k is the number of elements), the boss divides the task to its workers (actors) in such a fashion that each worker will perform it's operation on at least one range. That means, each worker is given a range (start and end), where it determines the sum of the squares of all the consecutive numbers, beginning from the 'start' element until 'start + k' element and checks if the obtained sum is a perfect square or not. If it is, then the 'start' element is printed and increments the 'start' element by 1 until it reaches the 'end'.

By executing this program, we found that the best performance of our implementation was with an optimal work unit size 7 for a large input N. For small input sizes we take the work unit size to be 5, again found using trial and error. The result of executing the program with input parameters N = 1000000 and k = 4, didn't reveal any output. Running the Program on MacBook Pro early 2016 model having 16 Gb RAM. 4 cores configuration gave the following output. Ratio of CPU time to Real time = 1.015 / 0.388 = 2.61. Below are the screenshots showing the command being executed and it's CPU time output respectively.

```
nikhileshreddy@Nikhileshs-MacBook-Pro F# % dotnet fsi "--langversion:preview" Project_Test3.fsx 1000000 4
Real: 00:00:00.000, CPU: 00:00:00.000, GC gen0: 0, gen1: 0, gen2: 0
```

```
Real: 00:00:00.388, CPU: 00:00:01.015, GC gen0: 110, gen1: 1, gen2: 0
nikhileshreddy@Nikhileshs-MacBook-Pro F# %
```

The largest input data that could be solved on the above specified machine was with N = 10^8 and k = 24

```
Real: 00:00:00.388, CPU: 00:00:01.015, GC gen0: 110, gen1: 1, gen2: 0
nikhileshreddy@Nikhileshs-MacBook-Pro F# % dotnet fsi "--langversion:preview" Project_Test3.fsx 100000000 24
Real: 00:00:00.000, CPU: 00:00:00.000, GC gen0: 0, gen1: 0, gen2: 0
```

```
19623573

82457176

8329856

34648837

Real: 00:00:29.528, CPU: 00:02:18.848, GC gen0: 26282, gen1: 4, gen2: 0
nikhileshreddy@Nikhileshs-MacBook-Pro F# %
```

It gave us 42 sequences in total, with a CPU to Real time ratio coming to be 2.18/0.29 = 4.66.