COP5612 – Fall 2020 Alin Dobra

Project 1 Submission

Akshay Kumar (UF: 4679-9946) | Rajat Rai (UF: 1417-2127)

Problem definition: An interesting problem in arithmetic with deep implications to the elliptic curve theory is the problem of finding perfect squares that are sums of consecutive squares.
A classic example is the Pythagorean identity: $3^2 + 4^2 = 5^2$ that reveals that the sum of squares of 3, 4 is itself a square.
A more interesting example is Lucas‘ Square Pyramid: $1^2 + 2^2 + ... + 24^2 = 70^2$.
In both examples, sums of squares of consecutive integers from the square of another integer. The goal of this first project is to use F# and the actor model to build a good solution to this problem that runs well on multi-core machines.

Requirements: The input provided (as command line to your program, e.g. my app) will be two numbers: N and k. The overall goal of your program is to find all k consecutive numbers starting at 1 and up to N, such that the sum of squares is itself a perfect square (square of an integer).

Our Solution: Our solution involves having a dynamic number of workers actors that will be initialized by a Supervisor Actor. The supervisor actor simply breaks the N tasks into [ceiling of N/k] tasks and keeps track of how many of them are completed by maintaining a counter.

Algorithm Optimization: Instead of using a traditional loop method to get the sum of squares for every start or sequence, the range 1 to N, we decided to a more computationally efficient method calculate the sum of squares using the formula below:

$$P_n = \sum_{k=1}^{n} k^2 = \frac{2n^3 + 3n^2 + n}{6}$$

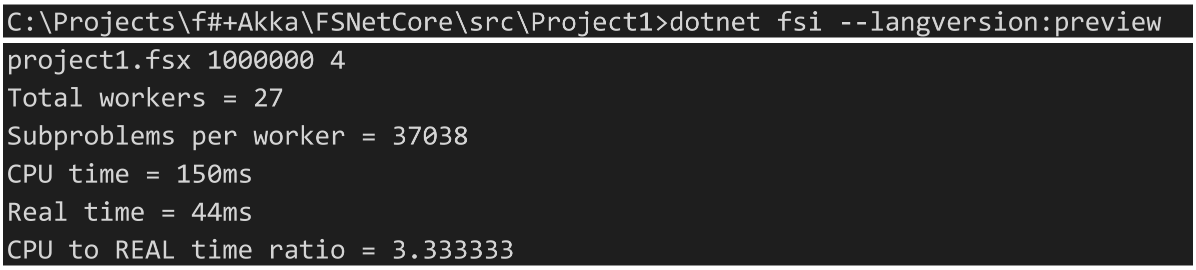
Here P_n is the sum squares of first n numbers. So to find the sum of squares, we just need to check if P_{i+k} -P_i is a perfect square or not for i in range of [1,n].

Size of the work unit: We observed that a lot of variables contribute to the optimum CPU/REAL Ratio, like:

- Length of the range
- Total no. of interconnected systems
- Total no. of Physical/Logical Cores of the systems
- CPU Loads

Command: dotnet fsi --langversion:preview project1.fsx 1000000 4

Result: The most optimum number we came across on our machine (Intel® Core™ i7-8550U Processor; # of Cores 4,# of Threads8) for the problem of N= 1000000 and k=4 are as follows:



```
C:\Projects\f#+Akka\FSNetCore\src\Project1>dotnet fsi --langversion:preview
project1.fsx 1000000 4
Total workers = 27
Subproblems per worker = 37038
CPU time = 150ms
Real time = 44ms
CPU to REAL time ratio = 3.333333
```

Additional Results:

Sr. No	N	K	Runtime Info
1	**10^6**	**24**	**Total workers = 25 Subproblems per worker = 40000 CPU time = 130ms Real time = 38ms CPU to REAL time ratio = 3.333333**
			Total workers = 27 Subproblems per worker = 37038 CPU time = 140ms Real time = 40ms CPU to REAL time ratio = 3.414634
			Total workers = 50 Subproblems per worker = 20000 CPU time = 160ms Real time = 58ms CPU to REAL time ratio = 2.666667
			Total workers = 75 Subproblems per worker = 13334 CPU time = 170ms Real time = 56ms CPU to REAL time ratio = 2.982456
			Total workers = 100 Subproblems per worker = 10000 CPU time = 170ms Real time = 55ms CPU to REAL time ratio = 2.982456
2	**10^8**	**24**	**Total workers = 50 Subproblems per worker = 2000000 CPU time = 7020ms Real time = 1176ms CPU to REAL time ratio = 5.959253**
			Total workers = 75 Subproblems per worker = 1333334 CPU time = 6570ms Real time = 1227ms CPU to REAL time ratio = 5.341463
			Total workers = 100 Subproblems per worker = 1000000 CPU time = 6550ms

			Real time = 1228ms CPU to REAL time ratio = 5.320877
3	**10^9**	**24**	**Total workers = 50 Subproblems per worker = 20000000 CPU time = 75870ms Real time = 12913ms CPU to REAL time ratio = 5.872291**
			Total workers = 75 Subproblems per worker = 13333334 CPU time = 58210ms Real time = 10391ms CPU to REAL time ratio = 5.600885
			Total workers = 100 Subproblems per worker = 10000000 CPU time = 65140ms Real time = 11807ms CPU to REAL time ratio = 5.515198
4	**10^{10}**	**24**	**Total workers = 50 Subproblems per worker = 200000000 CPU time = 594750ms Real time = 106756ms CPU to REAL time ratio = 5.570907**
			Total workers = 75 Subproblems per worker = 133333334 CPU time = 584060ms Real time = 105753ms CPU to REAL time ratio = 5.522504
			Total workers = 100 Subproblems per worker = 100000000 CPU time = 623180ms Real time = 110346ms CPU to REAL time ratio = 5.647253
5	**10^{11}**	**24**	**Total workers = 1000 Subproblems per worker = 100000000 CPU time = 6814760ms Absolute time = 1165205ms CPU to REAL time ratio =5.848535**