COP5612 – Fall 2020 / Project 1

#Group members:
Hsiang-Yuan Liao, UFID:4353-5341, hs.liao@ufl.edu
Tung-Lin Chiang, UFID:9616-8929

#Project files:
proj1.fs (main file)
proj1.fsx (for support script file) (direct references #r)
directory file for sharp project (.fsproj)

#Implementation of our Actor Model:
We implemented the Boss-Worker actor model, which boss actor will deliver jobs to worker actors separately. When ever a worker actor get the computation done, it will send message to boss actor for extra job to do.
There is a stateful sink in our boss actor which helps boss actor to deliver jobs correctly.

#**Size of the work unit:**
We determined to let N=100,000,000 k=2 for tuning the size of work unit. It is better to choose a large N to show if it really compute in parallel.

First we try the work unit size = N, which has no parallelism at all, there will be only one actor that do the whole computation. The result is shown in Figure 1. And we have tried size = 1, which we know it will be really inefficient for parallelism. And this refers to figure 2.

```
AlexLiao-MBP:FirstActorAkka hsiang-yuanliao$ time dotnet run 100000000 2
3
20
119
696
4059
23660
137903
803760
4684659
27304196


real    0m4.306s
user    0m4.345s
sys     0m0.365s
```

Figure 1.
Size of Work Unit = N

Next we have tried work unit size = 1. Although we already know that it will be really inefficient, it achieves parallelism. The CPU time to REAL time ratio is almost equal to 6.59. This refers to figure 2.

```
AlexLiao-MBP:FirstActorAkka hsiang-yuanliao$ time dotnet run 100000000 2
27304196
4684659
803760
137903
23660
4059
696
119
20
3


real    2m6.960s
user    13m57.122s
sys     0m52.582s
```

Figure 2.
Size of Work Unit = 1
ratio ~= 6.6

Last, we think of than we have fixed 100 worker actors in this model, so it is better to let the work unit size to be MAX(N/100, 1). The ratio is almost equal to 3, which runs on a Quad-Core laptop. It show in Figure 3.

```
AlexLiao-MBP:FirstActorAkka hsiang-yuanliao$ time dotnet run 100000000 2
27304196
3
20
119
696
4059
23660
4684659
137903
803760


real    0m3.981s
user    0m12.010s
sys     0m0.467s
```

Figure 3.
Size of Work Unit
= MAX(N/100, 1)
ratio ~= 3

#dotnet fsi proj1.fsx 1000000 4

With F# Script (#time ”on”) setting, there won’t be any answer for N=1,000,000 k=4

```
AlexLiao-MBP:FirstActorAkka hsiang-yuanliao$ dotnet fsi proj1.fsx 1000000 4
 Real: 00:00:00.242, CPU: 00:00:00.217, GC gen0: 1, gen1: 0, gen2: 0
AlexLiao-MBP:FirstActorAkka hsiang-yuanliao$ dotnet fsi proj1.fsx 1000000 4
 Real: 00:00:00.257, CPU: 00:00:00.231, GC gen0: 1, gen1: 0, gen2: 0
```

#time dotnet fsi proj1.fsx 1000000 4

For the time command for F# script, we have some concerns.
The CPU time to REAL time ratio is not as we expected as showed below.
BUT, when we have the same code running in .fs file, everything is different, it shows that our code is available to parallel computation. And more interesting is that, in .fsx file, the same code gives wrong answer when N comes to 100000000.
Same logic acts differently under different files, maybe there are some constraint for F# scripts?

```
AlexLiao-MBP:FirstActorAkka hsiang-yuanliao$ time dotnet fsi proj1.fsx 1000000 4
Real: 00:00:00.276, CPU: 00:00:00.235, GC gen0: 1, gen1: 0, gen2: 0

real    0m3.834s
user    0m2.965s
sys     0m0.555s
```

#The largest Problem Set
We use uint64, as it won't overflow it could support unsigned 64bit integer which is 0 to 18,446,744,073,709,551,615.
Just tried N = 1,000,000,000 k = 11, figure is shown below.

```
AlexLiao-MBP:FirstActorAkka hsiang-yuanliao$ time dotnet run 100000000 11
73024176
18
38
456
854
9192
17132
183474
341876
3660378
6820478


real    0m3.876s
user    0m11.800s
sys     0m0.526s
AlexLiao-MBP:FirstActorAkka hsiang-yuanliao$
```