Deep Bodra
5801 1841

README

GROUP MEMBER NAMES

Individual Project

RUNNING THE CODE

Install Dependencies

For square root of BigInteger (more precision than float)

```
dotnet add package Extreme.Numerics --version 7.0.15
```

For Akka Actors

```
dotnet add package Akka.FSharp --version 1.4.10
```

For Akka Remote

```
dotnet add package Akka.Remote --version 1.4.10
```

For serializing discriminated unions between 2 machines

```
dotnet add package Akka.Serialization.Hyperion --version 1.4.10
```

Run proj1.fsx

```
dotnet fsi --langversion:preview .\proj1.fsx 1000000 20
```

WORK UNIT

Definition

1. For this problem, the work unit is the number of sequences that each worker/actor has to check to determine if its squared sum is a perfect square or not
2. If the number of Actors is **`nActors`** then each actor get **`n/nActors`** sequences.
3. However if n is not divisible by nActors then the remainder of the work (**`n%nActors`**) will be equally distributed among the first **`n%nActors`** actors
4. For eg, if `nActors = 3` and `n=20` then the assigned work will be `7, 7 and 6` respectively

Determining

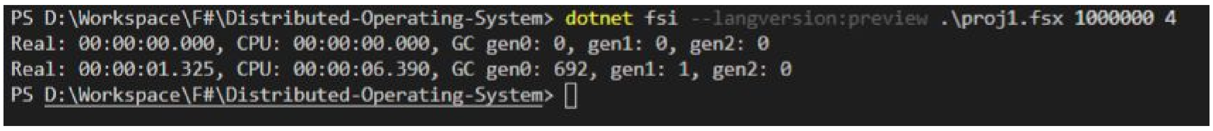
1. If number of actors is less than the number of cores then all of the cores are not utilized
2. If the number of actors is very large then all of the cores will be utilized but the ratio of CPU time to real time will drop because the assignment of an actor to the thread will take up extra time

3. If the number of actors is in the order of the number of cores then maximum ratio is obtained. I am able to get a ratio of 4.5 to 4.9 on large problems with 8 workers. (My Laptop has 4 physical cores or 8 logical processors)

RESULT

1. `dotnet fsi --langversion:preview .\proj1.fsx 1000000 4`

   `nActors = 8.0`

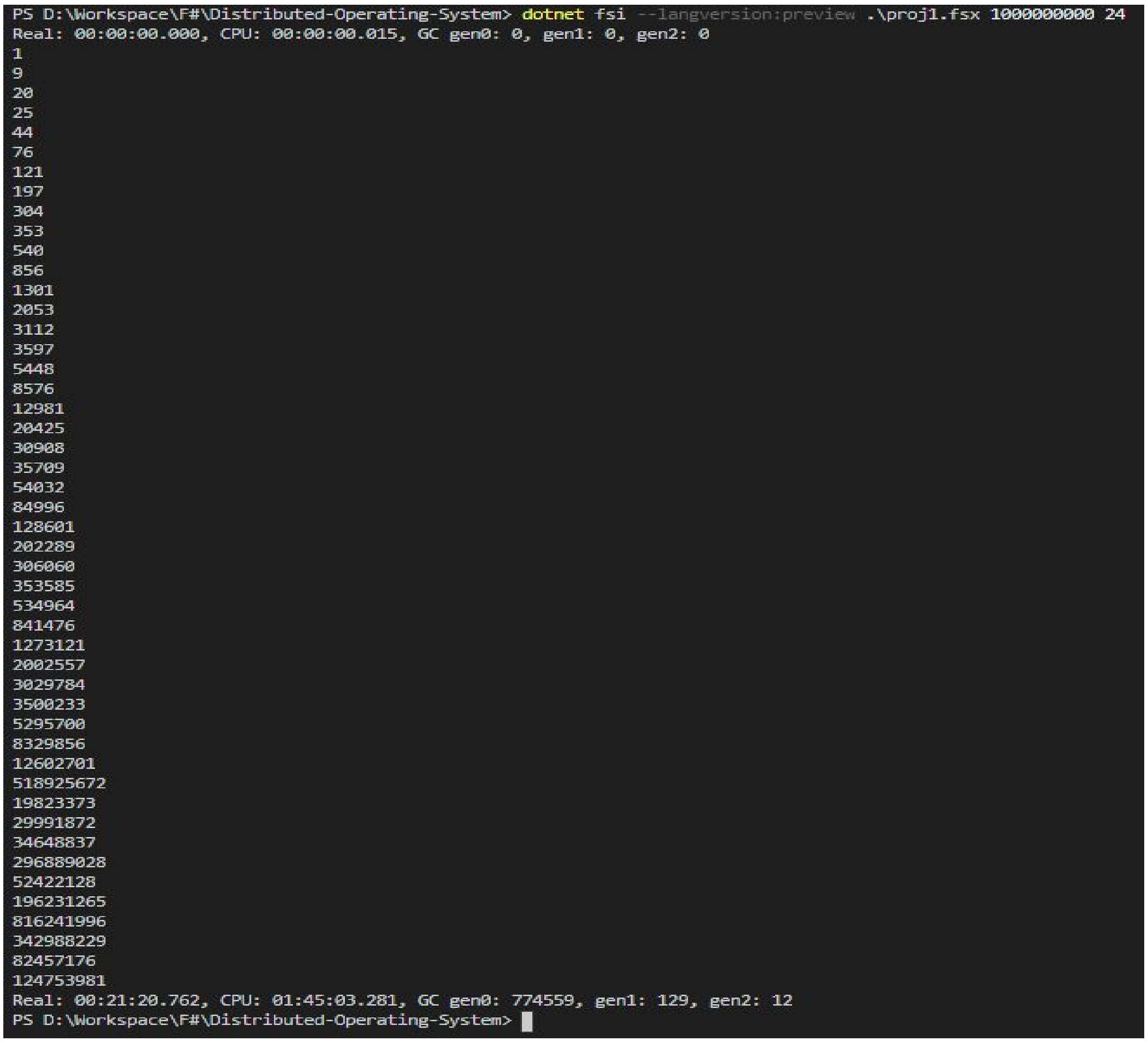


```
PS D:\Workspace\F#\Distributed-Operating-System> dotnet fsi --langversion:preview .\proj1.fsx 1000000 4
Real: 00:00:00.000, CPU: 00:00:00.000, GC gen0: 0, gen1: 0, gen2: 0
Real: 00:00:01.325, CPU: 00:00:06.390, GC gen0: 692, gen1: 1, gen2: 0
PS D:\Workspace\F#\Distributed-Operating-System> 
```

   `CPU TIME/REAL TIME = 4.82`

LARGEST PROBLEM

`n = 10`^{9} `and k = 24`



```
PS D:\Workspace\F#\Distributed-Operating-System> dotnet fsi --langversion:preview .\proj1.fsx 1000000000 24
Real: 00:00:00.000, CPU: 00:00:00.015, GC gen0: 0, gen1: 0, gen2: 0
1
9
20
25
44
76
121
197
304
353
540
856
1301
2053
3112
3597
5448
8576
12981
20425
30908
35709
54032
84996
128601
202289
306060
353585
534964
841476
1273121
2002557
3029784
3500233
5295700
8329856
12602701
518925672
19823373
29991872
34648837
296889028
52422128
196231265
816241996
342988229
82457176
124753981
Real: 00:21:20.762, CPU: 01:45:03.281, GC gen0: 774559, gen1: 129, gen2: 12
PS D:\Workspace\F#\Distributed-Operating-System> 
```

1. `CPU TIME / REAL TIME = 4.92`
2. The precision of F# float is only upto 13-14 places and when I run it for 10^9 I did not get many of the numbers in the screenshot.
3. So I switched to `BigInteger` and used a library called "Extreme.Numerics.FSharp" which provides square root for `BigInteger`. F# does not provide support for square root of `BigInteger`

REMOTE ACTORS

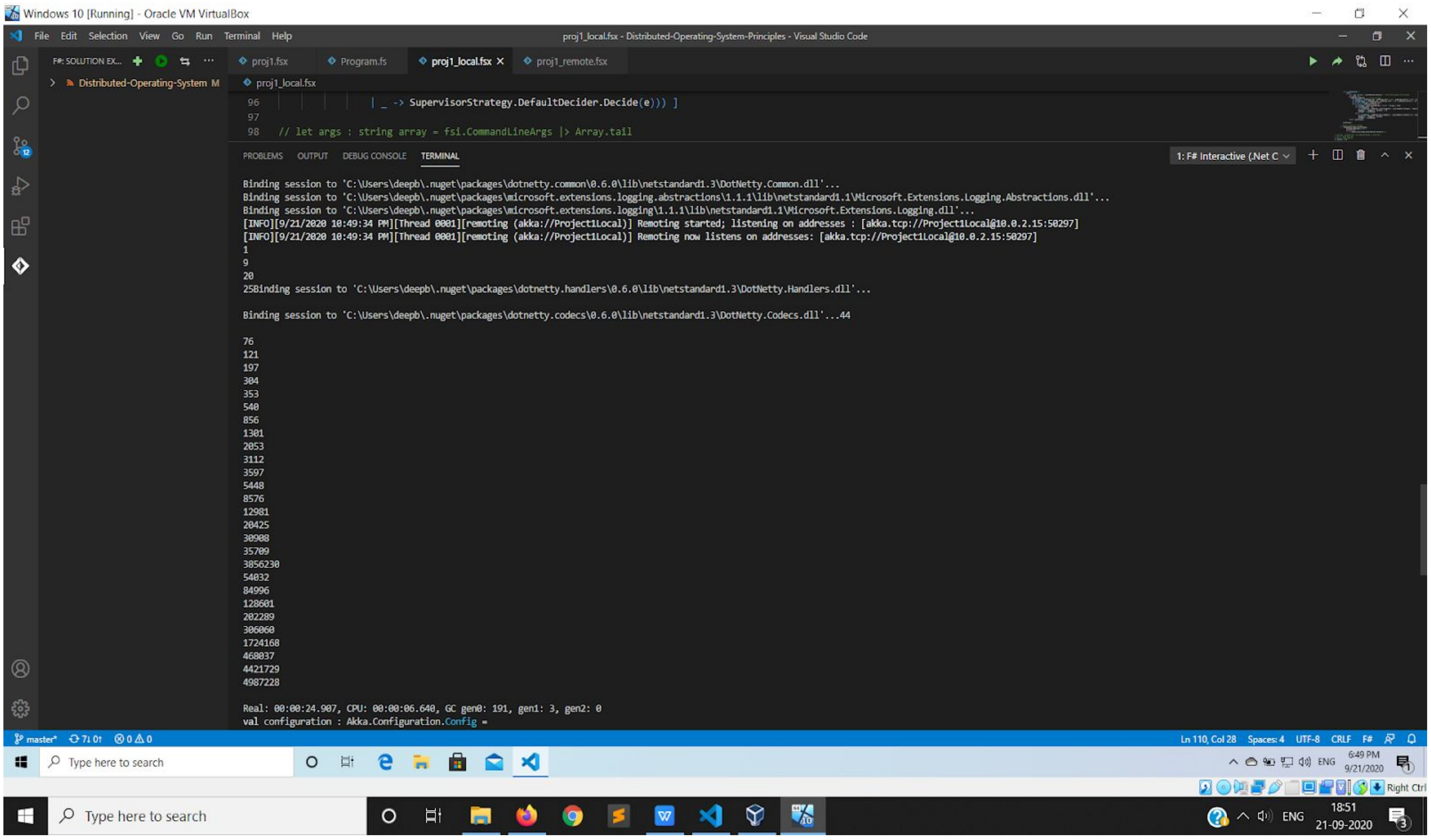
SETUP

1. Since I am doing the project individually, to simulate two machines I am using a virtual machine (virtual box) as a second machine
2. The host machine acts as a remote actor and the virtual machine acts as the local actor
3. For given, n and k, the virtual machine solves the problem for x=1 to x=n/2 and the host machine solves the problem for x=n/2+1 to x=n
   For eg, if n=40 then virtual machine solves [1, 20] and host machine solves [21, 40]

SCREENSHOTS

N = 10^8 and k = 24

Virtual Machine (Local Actor) (proj1_local.fsx)
Run using: Alt+Enter



```
| _ -> SupervisorStrategy.DefaultDecider.Decide(e))) ]

// let args : string array = fsi.CommandLineArgs |> Array.tail
```

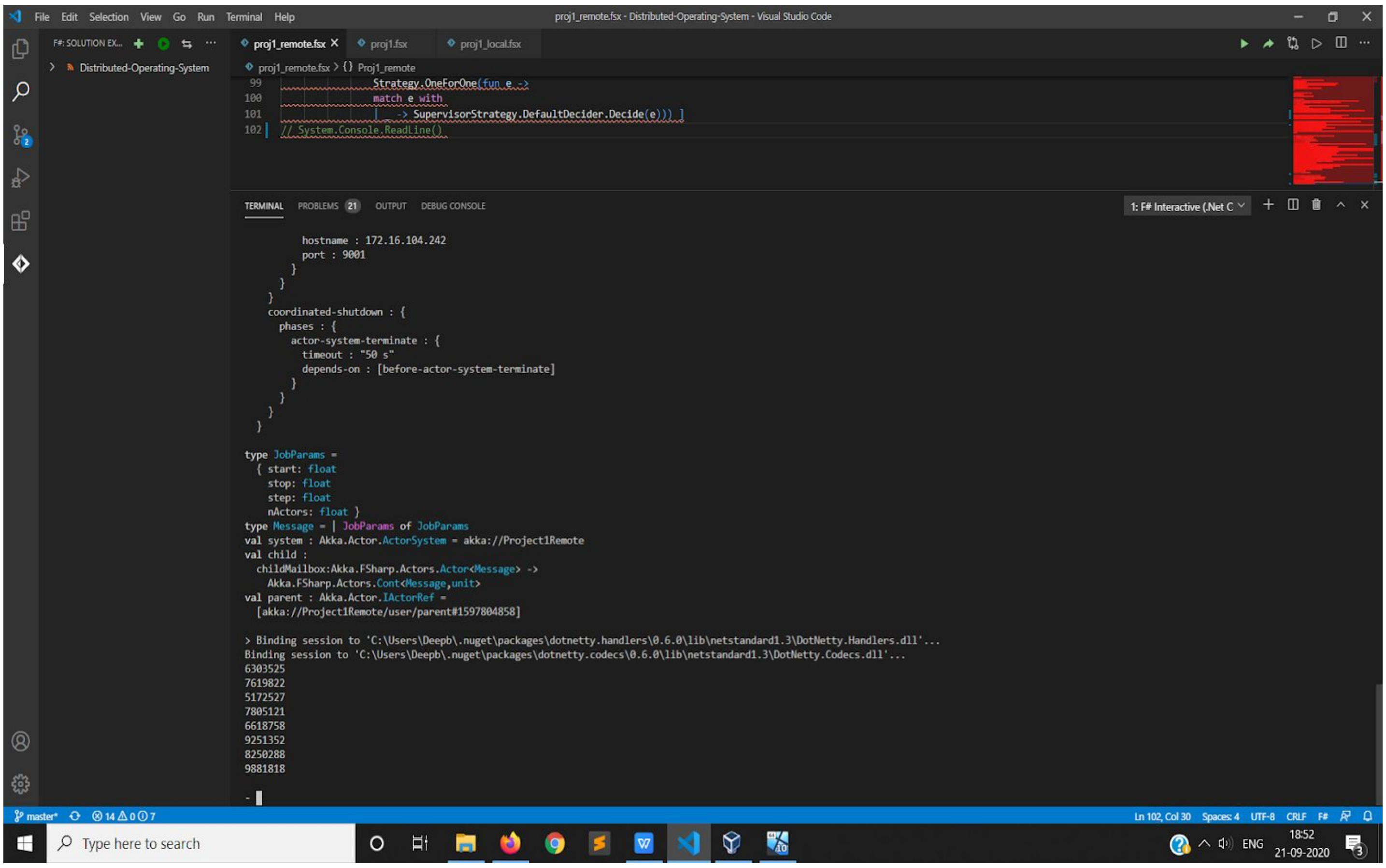
```
Binding session to 'C:\Users\deepb\.nuget\packages\dotnetty.common\0.6.0\lib\netstandard1.3\DotNetty.Common.dll'...
Binding session to 'C:\Users\deepb\.nuget\packages\microsoft.extensions.logging.abstractions\1.1.1\lib\netstandard1.1\Microsoft.Extensions.Logging.Abstractions.dll'...
Binding session to 'C:\Users\deepb\.nuget\packages\microsoft.extensions.logging\1.1.1\lib\netstandard1.1\Microsoft.Extensions.Logging.dll'...
[INFO][9/21/2020 10:49:34 PM][Thread 0001][remoting (akka://Project1Local)] Remoting started; listening on addresses : [akka.tcp://Project1Local@10.0.2.15:50297]
[INFO][9/21/2020 10:49:34 PM][Thread 0001][remoting (akka://Project1Local)] Remoting now listens on addresses: [akka.tcp://Project1Local@10.0.2.15:50297]
1
9
20
25Binding session to 'C:\Users\deepb\.nuget\packages\dotnetty.handlers\0.6.0\lib\netstandard1.3\DotNetty.Handlers.dll'...

Binding session to 'C:\Users\deepb\.nuget\packages\dotnetty.codecs\0.6.0\lib\netstandard1.3\DotNetty.Codecs.dll'...44

76
121
197
304
353
540
856
1301
2053
3112
3597
5448
8576
12981
20425
30908
35709
3856230
54032
84996
128601
202289
306060
1724168
468037
4421729
4987228

Real: 00:00:24.907, CPU: 00:00:06.640, GC gen0: 191, gen1: 3, gen2: 0
val configuration : Akka.Configuration.Config =
```

Host Machine (Remote Actor) ((proj1_remote.fsx))
Run using: Alt+Enter


File Edit Selection View Go Run Terminal Help
proj1_remote.fsx - Distributed-Operating-System - Visual Studio Code
proj1_remote.fsx
proj1.fsx
proj1_local.fsx
Distributed-Operating-System
TERMINAL
PROBLEMS 21
OUTPUT
DEBUG CONSOLE
1: F# Interactive (.Net C
master*
Ln 102, Col 30 Spaces: 4 UTF-8 CRLF F#
Type here to search
18:52
21-09-2020

```
                        Strategy.OneForOne(fun e ->
                        match e with
                        | _ -> SupervisorStrategy.DefaultDecider.Decide(e))) ]
// System.Console.ReadLine()
```

```
          hostname : 172.16.104.242
          port : 9001
        }
      }
    }
    coordinated-shutdown : {
      phases : {
        actor-system-terminate : {
          timeout : "50 s"
          depends-on : [before-actor-system-terminate]
        }
      }
    }
  }

type JobParams =
  { start: float
    stop: float
    step: float
    nActors: float }
type Message = | JobParams of JobParams
val system : Akka.Actor.ActorSystem = akka://Project1Remote
val child :
  childMailbox:Akka.FSharp.Actors.Actor<Message> ->
    Akka.FSharp.Actors.Cont<Message,unit>
val parent : Akka.Actor.IActorRef =
  [akka://Project1Remote/user/parent#1597804858]

> Binding session to 'C:\Users\Deepb\.nuget\packages\dotnetty.handlers\0.6.0\lib\netstandard1.3\DotNetty.Handlers.dll'...
Binding session to 'C:\Users\Deepb\.nuget\packages\dotnetty.codecs\0.6.0\lib\netstandard1.3\DotNetty.Codecs.dll'...
6303525
7619822
5172527
7805121
6618758
9251352
8250288
9881818

- 
```

CONCLUSION

We used Actor model to achieve concurrency and parallelism without having to worry about threads and mutexes