Distributed Operating Systems Project 1

Group Members:
Balaji Balasubramani - 98763981
Divyareddy Surapa Reddy - 13727408

Outline:
We have implemented 3 models. All of them perform the subtasks using the sliding window algorithmic approach. But it differs in the way they are implemented (How actors are spawned).

Model 1: SingleMachine.fsx
Implementation:
- This is a Single machine model where all the actors are spawned on the same machine based on it's cores available.
- It takes the input from the command line and spawns the boss actor which creates and maintains a pool of worker actors.
- The boss actor splits the input based on the CPU cores present for the machine and allocates the subproblems to the worker actors in the pool in a round-robin manner.
- Worker actors receive the input from the Boss and do the processing and return the completed message every time it finishes the subproblem.

Steps to compile and run:

1. Extract the zip file - unzip balasubramani_surapareddy.zip
2. cd balasubramani_surapareddy/SingleMachine_Model
3. run - dotnet fsi --langversion:preview SingleMachine.fsx 1000000000 24

Model 2: Remote_Model_1
Implementation: This is a remote actor model implemented using two machines.
System1: RemoteClient_Model1.fsx -
- It spawns the boss actor and takes input from the command line.
- Boss actor receives the problem input and creates and maintains a pool of local system worker actors.
- Boss splits the tasks based on cores count and allocates the subproblems to the remote machine boss and to the worker actors in the pool (using Round-Robin) consecutively.
- The subproblems sent to the remote boss allocates them to it's child actors in the remote machine.

System2: RemoteServer_Model1.fsx
- It spawns the remote boss and receives the subproblems from System1 and allocates them to it's pool of worker actors using round robin manner.
- The worker actor sends the results and the completed message to the System1's Boss after processing.

Steps to compile and run: (Always System 2 should start first as it should up and listening before System 1 send the message)

1. Extract the zip file - unzip balasubramani_surapareddy.zip
2. cd balasubramani_surapareddy/Remote_Model_1
3. First start system 2 using: (Remote Machine)
   dotnet fsi --langversion:preview System2.fsx
4. Then start system 1 using: (Current Machine)
   dotnet fsi --langversion:preview System1.fsx 1000000000 15

Model 3: Remote_Model_2
Implementation: This is also a remote actor model implemented using two machines. But it differs from the above model as below:

- In the above model, both System1 and System2 have a Boss actor and a pool of its own worker actors respectively. System1 sends the subproblems to System2 which are processed by the worker actors in System2.
- Now, in this model, there are no actors created in System2. But rather, the actors that are created by the System1 are remotely deployed at System2 using "**spawne**" command.

Steps to compile and run: (Always System 2 should start first as it should up and listening before System 1 send the message)

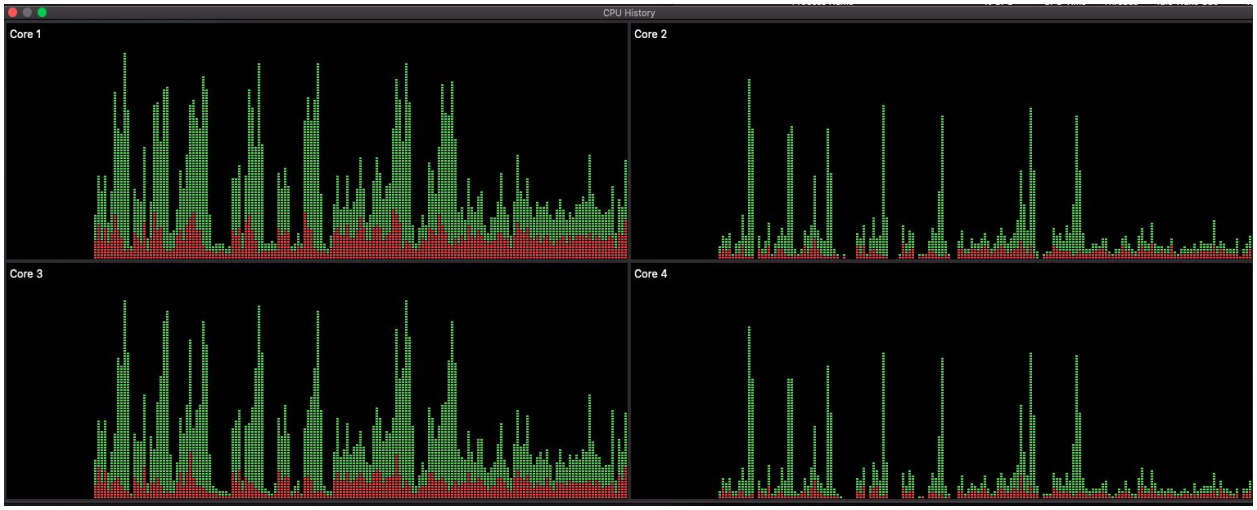
1. Extract the zip file - unzip balasubramani_surapareddy.zip
2. cd balasubramani_surapareddy/Remote_Model_2
3. Goto system 2 using: (Remote Machine)
   cd System2/
   run: time dotnet run System2.fsproj
4. Then start system 1 using: (Current Machine)
   cd System1/
   run: time dotnet run System1.fsproj 1000000000 24

CPU Utilzation and Performance

Model 1: Single Machine Model

Input: N = 10^8 K = 24

# Worker Actors	Real	CPU	Ratio
4	3.312	10.792	3.258
16	3.956	9.956	2.516
32	4.028	8.967	2.2261
64	3.967	8.230	2.074
100	3.934	8.039	2.0434
500	4.979	9.728	1.95
1000	5.121	9.07	1.771

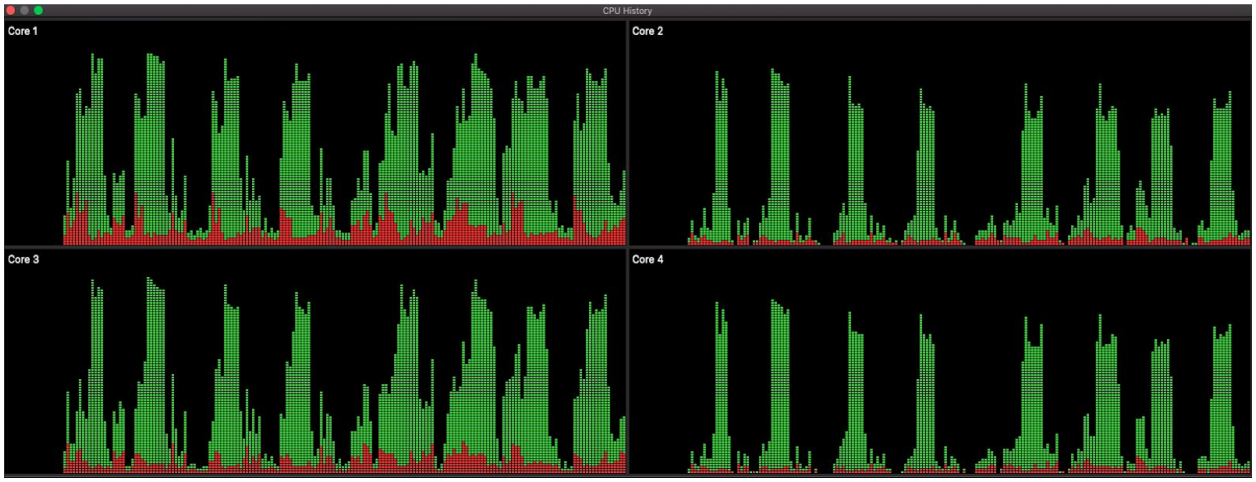


Observation:
We observed as the number of actors increases the ratio decreases. So, to obtain the maximum ratio we have kept the number of actors to 4, which is equal to the number of processors in the machine.

Model 2: Remote_Model_1
Input: N = 10^8 K = 24
Running same number of actors in both the systems

# Worker Actors	Real	CPU	Ratio
4	12.885	37.004	2.871
16	12.961	29.660	2.288
32	13.774	30.386	2.206
64	16.892	34.236	2.026
100	18.614	37.466	2.012
500	16.582	34.017	2.054
1000	17.115	35.450	2.071

System1



System2

Observation:
We observed that having remote connections leads to communication latency between the actors.

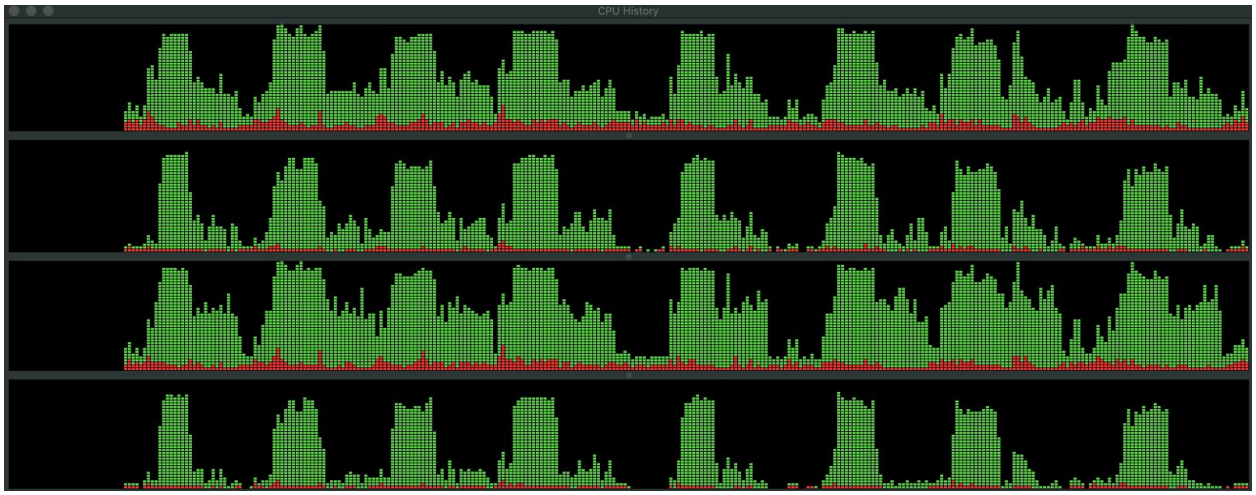
Model 3: Remote_Model_2

Input: N = 10^9 K = 24

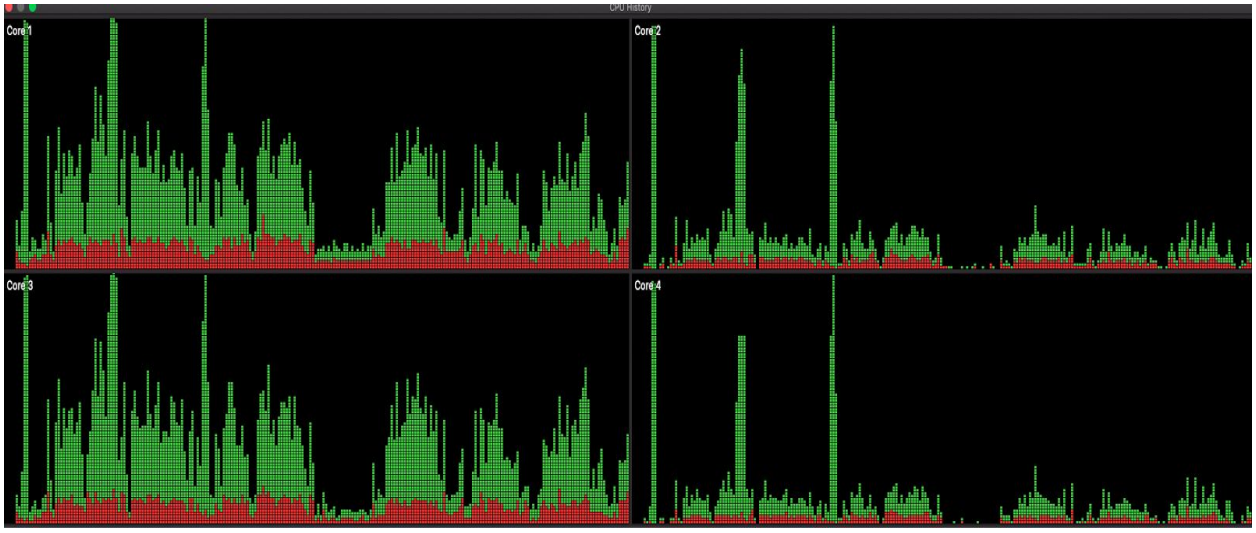
Running same number of actors in both the systems

# Worker Actors	Real	CPU	Ratio
4	13.785	39.123	2.838
16	14.146	40.254	2.845
32	14.235	33.126	2.327
64	17.145	35.236	2.055
100	19.012	38.465	2.024
500	16.888	35.139	2.080
1000	17.789	36.014	2.095

System1



System2



Result of the following command: dotnet fsi proj1.fsx 1000000 4
The answer is **NULL**

Runtime Output:
Model2:
Real: 00:00:01.847, CPU: 00:00:02.478, GC gen0: 29, gen1: 2, gen2: 0
Model1:
Real: 00:00:00.653, CPU: 00:00:00.894, GC gen0: 31, gen1: 2, gen2: 0

The largest problem you managed to solve: ***N = 10^9, K = 24***