COP5615 – Fall 2018

Project 1

Group Members

- Siddhant Mohan Mittal UFID: 6061-8545
- Prafful Mehrotra UFID: 1099-5311

Project Directory Structure

- Main directory contains Project Problem statement pdf and directory named code which contains the elixir mix project
- In lib we have two files [proj1.exs] and [multi_machine.exs] which are used in this project to solve the problem
- [proj1.exs] contains the main code
- [multi_machine.exs] which contain the code for running the system on two machines
- [worker.ex] in code(directory name) is helper module for multi machine processes.

Instructions for running the code

After unziping the file

```
$ cd code
$ mix run lib/proj1.exs 3 2
```

Input Format

Command line arguments: mix run lib/proj1.exs arg1 arg2

Where arg1 is the number N which is range in which the first number of solution set should lie. And arg2 is the value of K that is maximum size of the solution set.

Output Format

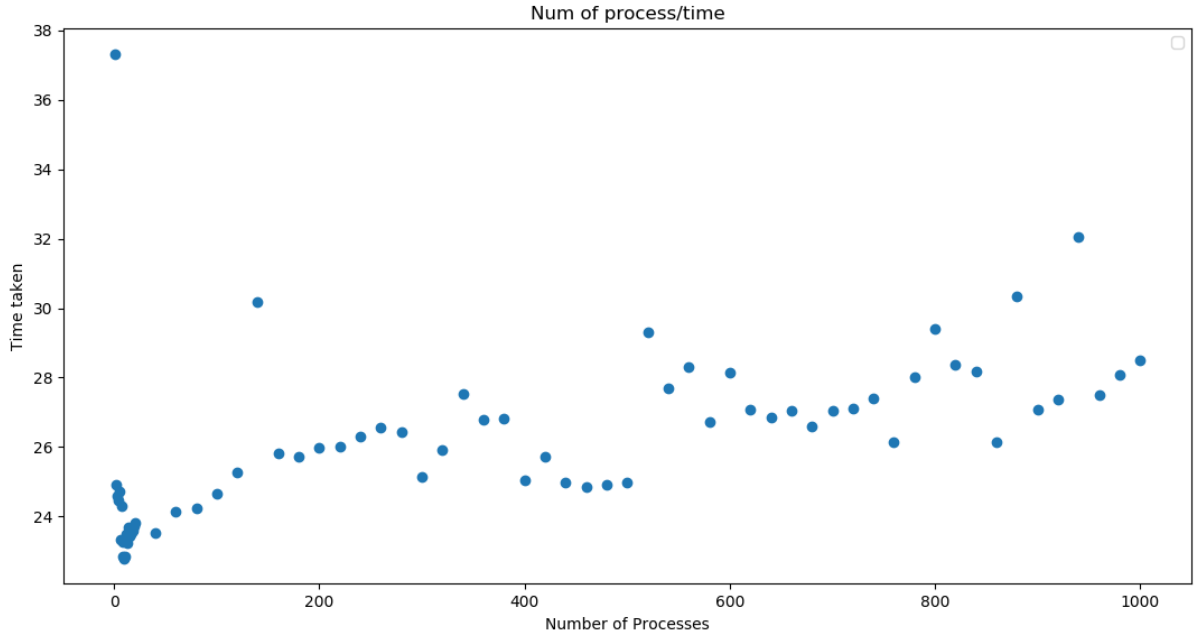
We are running for num_process increasing from 1 to 20 in each iteration. Hence each iteration prints the number of workers and time taken in that iteration

The result of the project will be solution set of the problem in first line followed by time taken in each iteration

Questions to be answered

1. Size of the work unit that you determined results in best performance for your implementation and an explanation on how you determined it. Size of the work unit refers to the number of sub-problems that a worker gets in a single request from the boss.

- Following graph is for N=1_000_000 and k=1_000 to calculate the optimized value of workUnit for the best performance.



- We can observe from the graph that optimized value of workunit that gives the best results is in the range 6-10.

2. The result of running your program for mix run proj1.exs 1_000_000 4

- No solution was found for this case and with time 1.63034 sec and CPU/time=3.01

3. The running time for the above as reported by time for the above. The ratio of CPU time to REAL TIME tells you how many cores were effec- tively used in the computation.

Depending upon the workunit

Workunit	Time
2	2.149726
4	1.63813
6	1.650756
8	1.63034
10	1.624344
12	1.86529
14	1.940466
16	1.793477
18	2.464876
20	1.80413

We calculated the ratio of CPU/real time = 3.9 for N=1000_000 k=1000

real 22m14.858s user 80m42.700s sys 0m17.781s

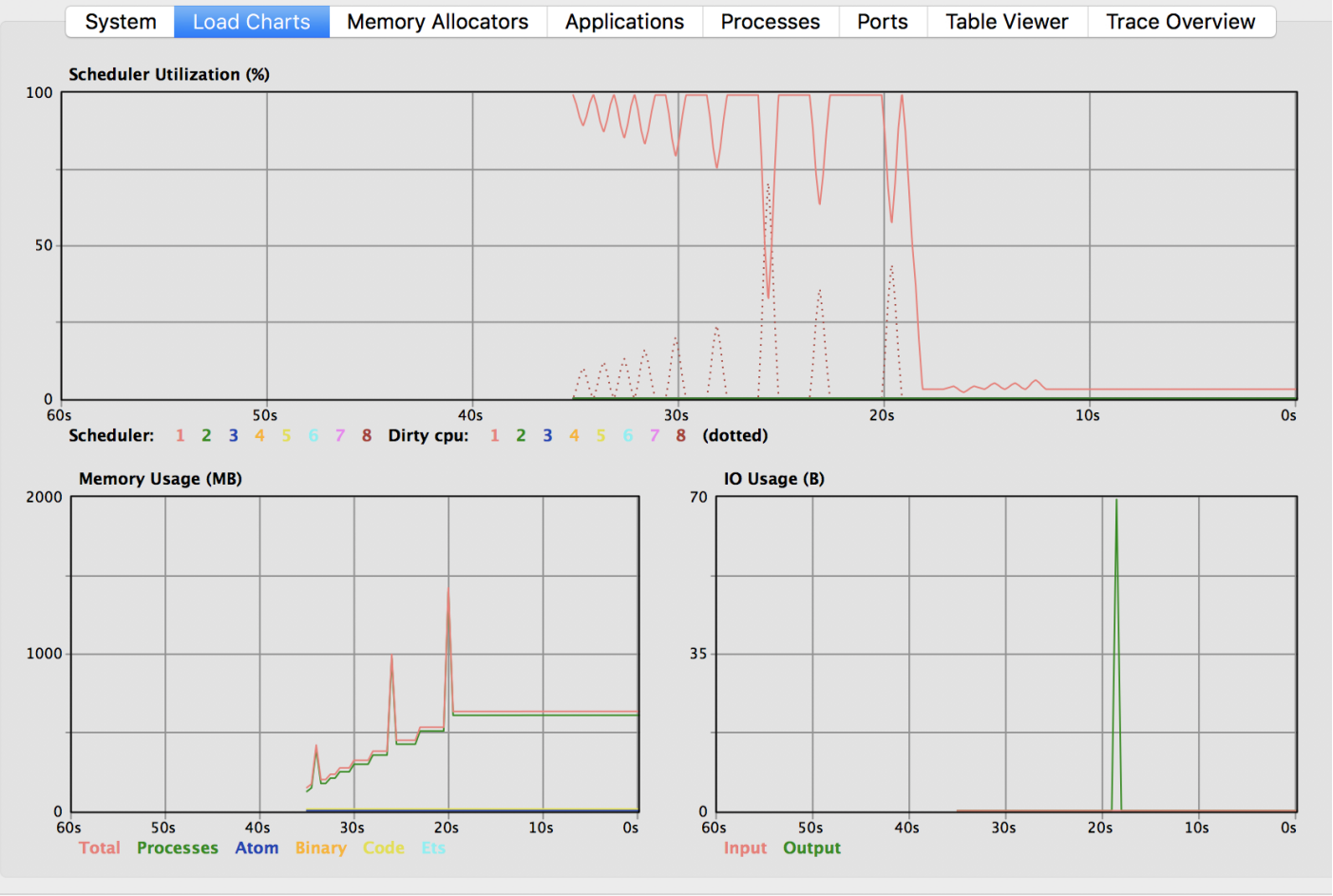
4. The largest problem you managed to solve

N = 1000_000 k = 1000 Workers: 8 time: 15.484459 sec

N = 10_000_000 k = 1000 Workers: 8 time: 238.7258 sec

Proof of concurrency

- For proof of concurrency one way of doing this is using the CPU/real time ratio which our system gives a value ranging from 3 to 4. As our system is 4 core system
- Another way of doing this by exploting various tools like fprof for profiling service and observer service of erlang VM. fprof gets clumsy hence we won't be explaining it here. We are going to run two different codes for showing concurrency and recording system usage using :observer.start()
- Observation for code lacking concurrency: this code calls various actors but sequential



- Observation for our current code having cocurrency: this code spwans various actors together and uses a scheduler to communicate through various actors. Here the actors run on different process in different cores

Instructions for bonus part

- We tried to connect two machines using the Node functionality of erlang VM and extened our previous code to run on two different machines.
- For this we have to use two files in our lib folder that is mutli-machine.exs and worker.exs.
- We have to set up a Node connection between two remote devices. The Importent thing here to noticed that cookie value should be same so that system can communicate with each other, there it is "elixir"
- Now on our remote system compile the worker module by runnig the iteractive elixir (iex) use the following commands. NOTE to be in the directory of the worker.ex

```
$ iex --sname prafful --cookie elixir
$ c('worker.ex')
```

Now run the following command on your host machine

```
$ elixir --sname siddhant1 --cookie elixir  lib/multi_machine.exs 1000000 4
```

We will be running 100 workUnits(50 on one and 50 on other)