Distributed Operating System Principles Project-1
Fall 2021

Kavya Gopal Rema Veeranna Gowda

09/24/2021

1 Problem Statement

Bitcoins are the most popular crypto-currency in common use. At their heart, bitcoins use the hardness of cryptographic hashing to ensure a limited "supply" of coins. In particular, the key component in a bit-coin is an input that, when "hashed" produces an output smaller than a target value. In practice, the comparison values have leading 0's, thus the bitcoin is required to have a given number of leading 0's.The hash you are required to use is SHA-256.
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.

2 Requirements

Input: The input provided will be, the required number of 0's of the bitcoin(K) and workload(N).
Output: Print the input string(randomly generated), and the corresponding SHA256 hash separated by a TAB, for each of the bitcoins you find.

3 Solution

- We implemented a remote actor model using two machines to mine bitcoins.
- In each machine, the number of actors are created based on the number of cores in that machine.
- The task, to mine bitcoin in this case, is divided among child actors by the parent actor.
- If only the server machine is running, all tasks are run by child actors on the server.
- When a client node is available, the tasks are divided between them.
- The parent actor in the server divides tasks among the child actors in server and sends tasks to client machines.

- The workload is equally distributed among server and client.
- The results from the client are sent back to the server and are printed at server end.

4 Implementation Details

4.0.1 Server

- Takes input from command line(workload and the number of zero's that should prefix the bitcoin).
- It spawns the parent actor and creates a list of child actors based on the number of processors in the system.
- The parent actor assigns tasks to child actors in server machine and sends tasks to client machines.

4.0.2 Client

- Takes the IP of server and port number as input in command line.
- It spawns remote parent actor and child actors based on the processor count on the machine.
- The remote boss actor divides tasks received from the server among child actors.
- Remote child actors send the results to server parent actors.

4.0.3 Running the Code

- Client machine:
  dotnet fsi –langversion:preview client.fsx serverIP serverPort
  Note: This should be done before starting server
- Server machine:
  dotnet fsi –langversion:preview server.fsx N K
  N is workload and K is number of zeros the bitcoin should start with

5 CPU Utilization and Performance

Input 1:
N = 1000000 (10^6)
k = 4
Bitcoins mined = 18

Real: 00:00:07.039, CPU: 00:00:22.171, GC gen0: 464, gen1: 4, gen2: 0
CPU/Real: 3.14

Observation: When the workload is relatively small, the CPU utilization across cores is quite less averaging around 18% as shown in Figure 1.

Input 2:
N = 100000000 (10^8)
k = 4
Bitcoins Found = 1002
Real: 00:06:46.647, CPU: 00:27:01.468, GC gen0: 45261, gen1: 50, gen2: 4
CPU/Real: 4.5

Observation: As the workload increases, we notice an increase in CPU utilization across cores as shown in Figure 2.

Input 3:
N = 10000 (10^4)
k = 4
Bitcoins mined = 0
Real: 00:00:02.703, CPU: 00:00:04.265, GC gen0: 17, gen1: 2, gen2: 0
CPU/Real: 2

Observation: The probability of mining bitcoins is less when the workload is less. It increases as we increase the workload and can be noticed from Input 1 and Input 2. Although, the tasks were divided among actors, we did not find a bitcoin (see Figure 3).

6 Findings

- The coin with the most 0s you managed to find: **8**.
- The largest number of working machines you were able to run your code with: **2**.

References

https://docs.microsoft.com/en-us/dotnet/fsharp/tour

https://bartoszsypytkowski.com/how-create-an-akka-net-cluster-in-f/

https://www.fsharpreactivepatterns.com/index.html

https://github.com/akkadotnet/getakka.net/blob/master/src/docs/FSharp%20API.md

https://en.wikibooks.org/wiki/F_Sharp_Programming/Values_and_Functions

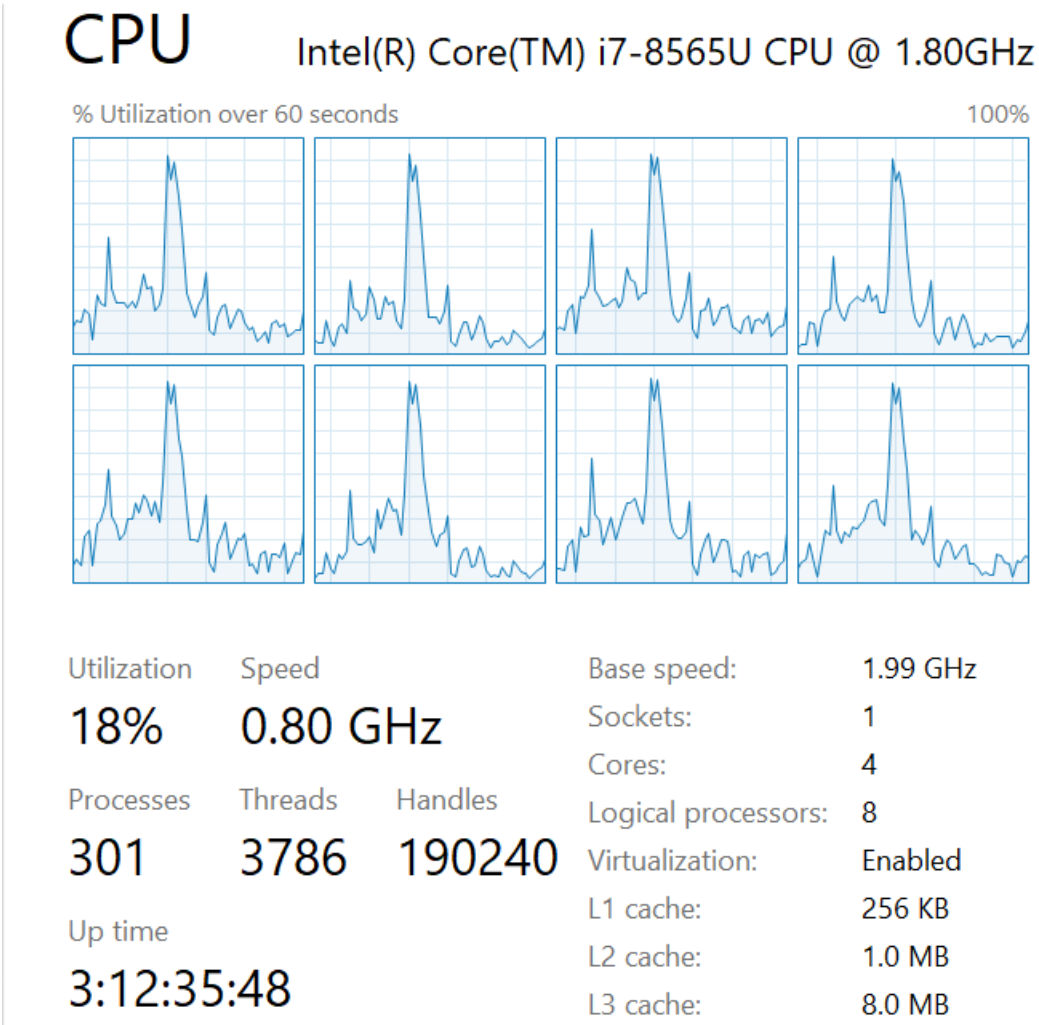

Figure 1: CPU Utilization for input 1.

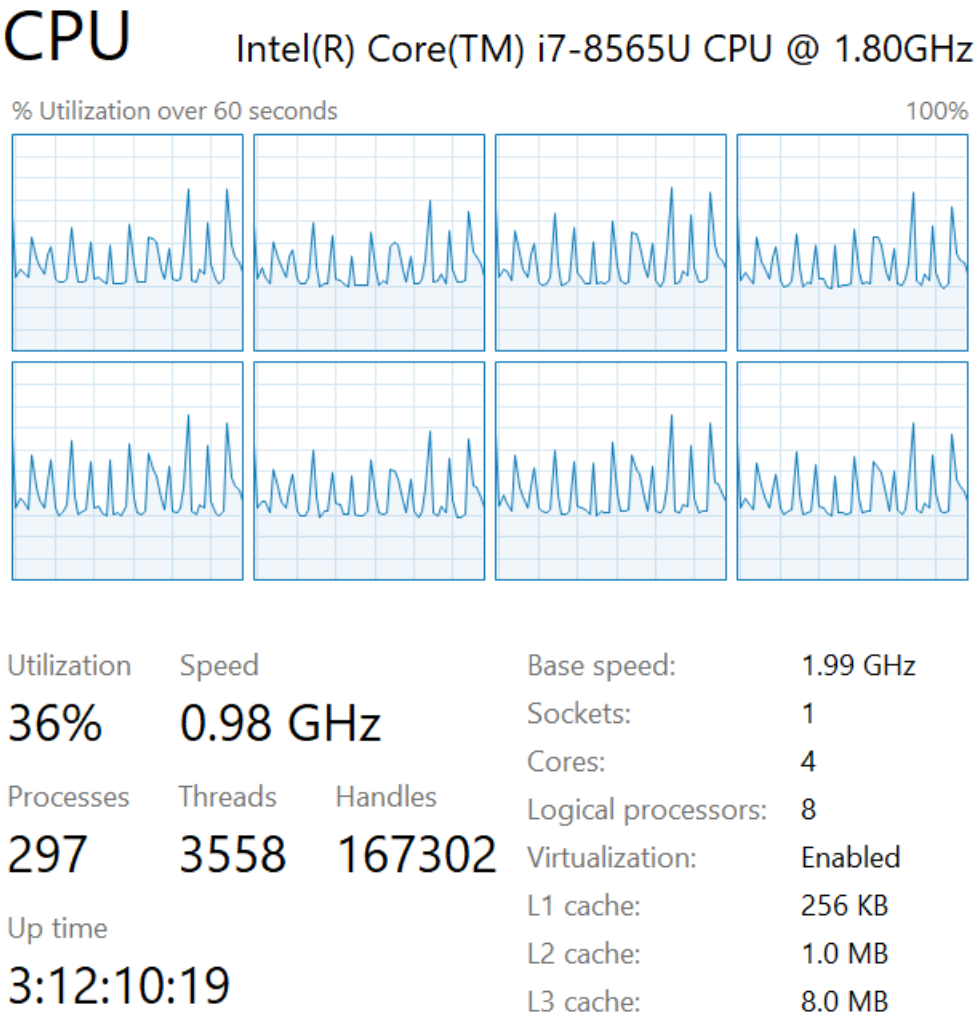

Figure 2: CPU Utilization for input 2.

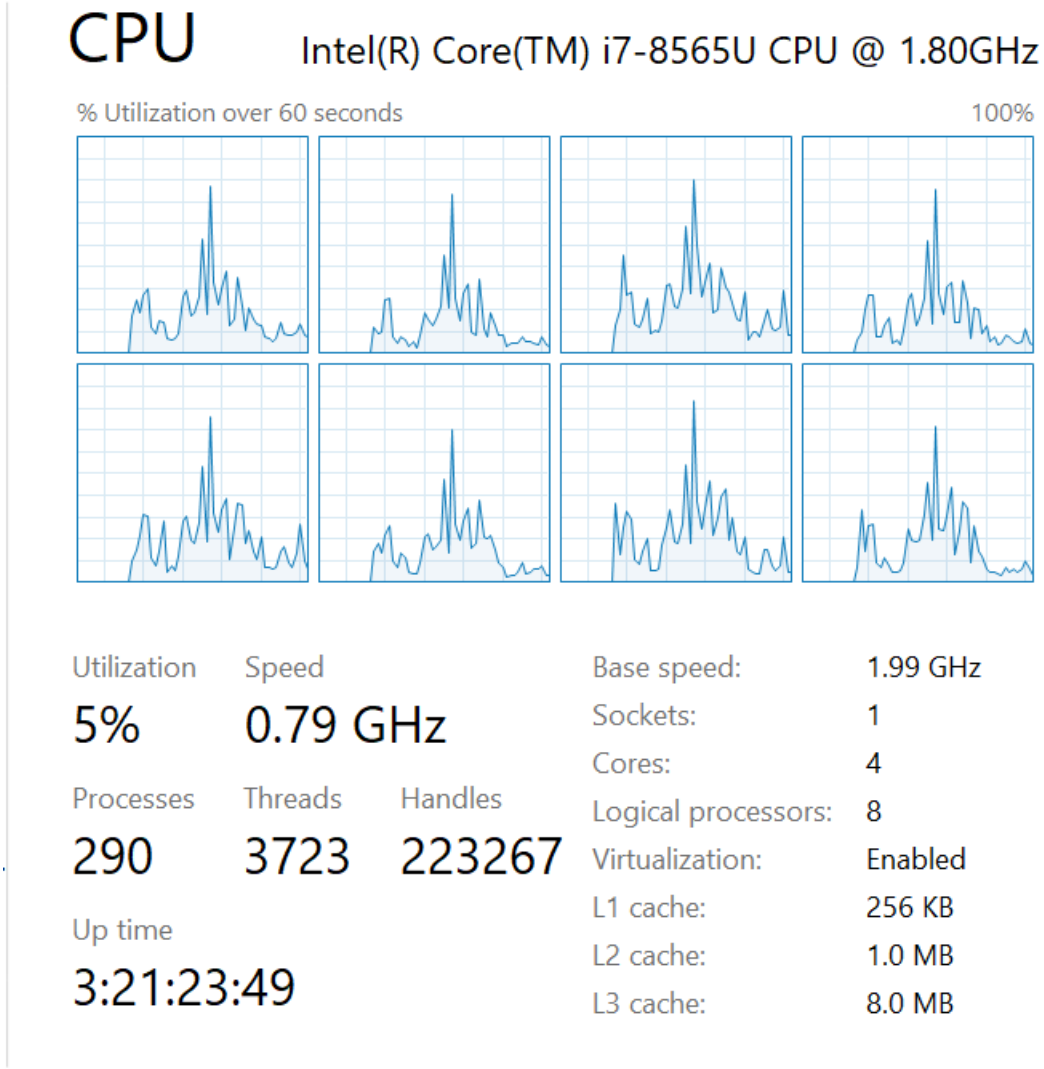

Figure 3: CPU Utilization for input 3.