Project 1 (Bitcoin Mining)

Due Sep 24, 2022 by 11:59pm **Points** 100 **Submitting** a file upload
Available Aug 24, 2022 at 12am - Sep 26, 2022 at 11:59pm

This assignment was locked Sep 26, 2022 at 11:59pm.

Project Guideline

- Due Date: September 24 (midnight)
- One submission per group
- Submit using CANVAS
- What to include:
  - README file including group members, other requirements specified below
  - project1.zip the code for the project (with all the parts)

Problem Definition

Bitcoins (seehttp://en.wikipedia.org/wiki/Bitcoin) are the most popular crypto-currency in common use. At their heart, bitcoins use the hardness of cryptographic hashing (for a reference seehttp://en.wikipedia.org/wiki/Cryptographichashfunction)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 (to ensure 3 leading 0's, you look for hashes smaller than*0x001000...* or smaller or equal to *0x000ff...*.The hash you are required to use is SHA-256. You can check your version against this online hasher:http://www.xorbin.com/tools/sha256-hash-calculator. For example, when the text "COP5615 is a boring class" is hashed, the value fb4431b6a2df71b6cbad961e08fa06ee6fff47e3bc14e977f4b2ea57caee48a4 is obtained. For the coins, you find, check your answer with this calculator to ensure correctness. The goal of this first project is to use Erlang and the Actor Model to build a good solution to this problem that runs well on multi-core machines.

Requirements

Input: The input provided (as command line to yourproject1.fsx) will be, the required number of 0's of the bitcoin.1

Output: Print, on independent

entry lines, the input string, and the correspondingSHA256 hash separated by a TAB, for each of the bitcoins you find. Obviously, your SHA256 hash must have the required number of leading 0s (k= 3 means3 0's in the hash notation). An extra requirement, to ensure every group finds different coins, is to have the input string prefixed by the gator link ID of one of the team members.

Example 1:

1

adobra;kjsdfk11 0d402337f95d018438aad6c7dd75ad6e9239d6060444a7a6b26299b261aa9a8b

indicates that the coin with 1 leading 0 is adobra;kjsdfk11and it is prefixed by the gatorlink ID adobra.

Distributed Implementation

The more cores you have to more coins you can mine. To this end, enlisting other machines adds to your coin mining capabilities. Extendproject1.scalaso that the argument is a computer address or IP address of the server. This program then becomes a "worker" and contacts the server to get work. This second program will not display anything. All the coins found have to be displayed by the server.

Example 2:

myprogram 10.22.13.155

will start a worker that contacts the Erlang server hosted at 10.22.13.155 and participates in mining. Hint. when testing this, have your project partner start a server, find the IP address of the server and then start the worker. Notice, that your server should be able to mine coins without any workers but has to accommodate workers as they become available.

Actor Modeling

In this project, you have to use exclusively the actor model in Erlang (projects that do not use multiple actors or use any other form of parallelism will receive no credit). Define worker actors that are given a range of problems to solve and a boss that keeps track of all the problems and perform the job assignment.

README file

In the README file, you have to include the following material:

- Size of the work unit that you determined results in the best performance for your implementation and an explanation of how you determined it. The size of the work unit refers to the number of sub-problems that a worker gets in a single request from the boss.
- The result of running your program for input 4
- The running time for the above is reported by time for the above and report the time. The ratio of CPU time to REAL TIME tells you how many cores were effectively used in the computation. If you are close to 1 you have almost no parallelism (points will be subtracted).
- The coin with the most 0s you managed to find.
- The largest number of working machines you were able to run your code with.

Project 1 Rubric

Criteria	Ratings		Pts
Code Correctness & Readability	**80 to >0.0 pts** **Full Marks**	**0 pts** **No Marks**	80 pts
Project ReadMe	**20 to >0.0 pts** **Full Marks**	**0 pts** **No Marks**	20 pts
			Total Points: 100