Client Module

The Client Module consists of a python application that simulates benchmarks being run on the client's device. The results are then saved in a text file that is sent to the Server Module for processing and storing.

The Client is first asked to enter his name. The benchmarks are then run sequentially with resting time between each one.

The python application contains 7 benchmark tests:

1. Bubble Sorting List Benchmark:
   This benchmark involves using the Bubble Sort Algorithm to sort in ascending order a list of 10 000 integers in descending order. This is the worst case scenario for a Bubble Sort Algorithm causing it to have a complexity of $O(n^2)$.

2. List Appending Benchmark:
   This Benchmark tests the ability of the device to make a decision, append and store a list. At First we are generating a random list of numerical and alphabetic characters then the function go through this list and test if the character is numerical and append it to a list and if not append it to another list. Moreover we are testing the capacity of referencing a new list by another already created.

3. Writing Benchmark:
   The Writing Benchmark tests a device's disk writing speed by creating a randomly filled dummy text file of 1GB. The process mentioned beforehand is repeated 5 times to fetch results when the client's device is under higher pressure. (Needless to say, the files created during this benchmark are then deleted)

4. Find Prime Numbers in List Benchmark:
   A random list of numbers is created and is then run through by this test to store all prime numbers.

5. Find Vowel in List Benchmark:
   A random list of letters is created and is then run through by this test to store all vowels.

6. Caesar Cipher Benchmark:
   A Caesar Cipher is an encryption technique that entails shifting each letter of the original text through the alphabet several positions. As an example, for a Shift Key = 3: A→D, B→E, C→F,… These Benchmarks generate a random text and then choose a random key between 0 and 26 and then encipher the created text. When the encipher process is done it decipher the same generated text by shifting the characters using the random generated key that was given.

7. ICMP Request Benchmark:
   The Client's device is tasked to ping google.com 50 times. A ping command is usually used to troubleshoot connectivity or to test a network however, the performance of a certain device running this benchmark can also affects its execution time.

When each test is done, a file with the Client's name saves the description, name and execution time in order for the Client Module to calculate the SPEC ratio and geo mean value of the Client's device and finally store the data in a Database.

Server module

In this part we will talk about the server module. Our server is written in java.

And it consists of 4 classes:

- OfficialBenchmark.java: contains the constructor of a general benchmark. We represented a benchmark by its Description, name and Execution Time.
- Server.java and Client.java both contains a list of benchmarks. In addition to it the server file contains a specific method to calculate the spec Ratio.
- TestBenchmark.java: Contain the main program to run and store the results in the database.

For this project we are using MySQL Database to store the result that returned from the server.

How it Works?

We send the Benchmarks.py file to the client. He will run it and send us back the feedback in a file.txt that will be stored in the server module.

Once we click on **run** in the server application *(TestBenchmark.java*), it will go and fetch the lists of Clients benchmarks from the local folder “Clients”, and then it will interpret them one by one and construct an Array of Clients according to the Client.java module.

The server is implemented locally according to a reference PC in this case it’s the PC of Jean_Yves.

Then we go through the array of Clients and get the spec Ratio of each Client after that we evaluate the Geometric Mean of the benchmarks and store it in the Database.

After the successful storing to the database is made, a CSV file is created in the current repository of the server module. This file contains the results of the clients ordered from the best the greatest to the lowest geometric Mean. The name of the file is “results.csv”. The path of the file is printed out on the screen if the user wants to go and find the file by its path.

The clients files can be found in the clients folder in the src folder and the result.csv file also can be found in the src.

Kindly find below the results of the clients that ran the benchmark files on their devices:

IdServer	IdClient	ServerName	ClientName	GeoMean
1	2	jean Yves	AmdEC	1.958
1	5	jean Yves	Jean Kesrewani	1.869
1	11	jean Yves	ramy	1.73
1	13	jean Yves	Rim	1.728
1	4	jean Yves	Chady Keryekos	1.548
1	10	jean Yves	Michel_Haddad	1.466
1	7	jean Yves	Karl Maamary	1.322
1	14	jean Yves	Rola Hadi	1.275
1	16	jean Yves	Victor	1.197
1	1	jean Yves	Ahmad El Cheikh	1.124
1	15	jean Yves	Serena Dahdah	1.006
1	6	jean Yves	Jean Yves	1
1	12	jean Yves	rhea	0.979
1	3	jean Yves	Anthony Haddad	0.913
1	8	jean Yves	marnie	0.7
1	9	jean Yves	Michel Khoury	0.311
NULL	NULL	NULL	NULL	NULL

Work presented by:

Ahmad el Cheikh

Jean Yves Youssef