Real-World-Application-Distributively-Asteroids-Processor

Tal Yitzhak - 204260533

Assaf Magrisso - 201247020

Technical Points:

Running the project –

Explanation of how our program works

\*We used t2.micro instances (ami-b73b63a0)

How much time it took the whole program to finish working:

………………………..

Did you think for more than 2 minutes about security?

Of course. We decided to secure the credentials like it is mentioned at the assignment instructions: "One way of doing that is by compressing the jar files with a password.",   
we have a file named AWSCredentials protected by a password, that we upload to S3.

Then, each of the elements in the work (Manager, LocalApplication, Workers) needs to download the file from S3, unzip it and give the right password to it.

- Did you think about scalability? Will your program work properly when 1 million clients connected at the same time? How about 2 million? 1 billion? Scalability is very important aspect of the system, be sure it is scalable!

In order to answer this question, we`ll first give an explanation of the way the manager handle local applications: We have 2 kinds of threads running inside the manager:

Local Application Handler – this thread is accepting new messages from local applications, and when a new one is coming, it doesn`t handle its request – it just opens up a new thread inside it`s own thread pool (with a fixed size), and keep answering new local application requests – so it is basically passing the mission of taking care of a local application to another thread – which makes him available to handle other requests. When million users send message to the Manager – he will be able to accept their messages, and the actual treatment is done by the thread pool. Handling a request also means the thread is responsible on opening a unique queue for that specific local application, so he could get the summary file ASAP. The fixed size thread pool represents the amount of locals that the manager can handle.

Workers Handler – this thread is accepting new messages from workers when they are done with their work. It is very similar to the Local Application Handler – it`s responsibility is only to get messages from workers and to pass that task to another thread in its own thread pool (also, fixed sized one). If a lot of analyzed data is sent from the workers side – it still gives attention to each message, which is processed and calculated by an available thread from the thread pool.

Once all tasks of a specific local application is done – we [immediately](https://www.google.co.il/search?client=firefox-b-ab&q=immediately&spell=1&sa=X&ved=0ahUKEwjmppa4hYvRAhWVOVAKHWR-DhEQvwUIFygA) (by the same thread) handle it and send a response back to the local application – so it can get the response ASAP, and we can clear the data structures.

- What about persistence? What if a node dies? Have you taken care of all possible outcomes in the system? Think of more possible issues that might arise from failures. What did you do to solve it? What about broken communications? Be sure to handle all fail-cases!

- maximum amount of 20 instances check.

- new api key from NSAS?

- close workers that does`nt

- Threads in your application, when is it a good idea? When is it bad? Invest time to think about threads in your application.

- Do you understand how the system works? Do a full run using pen and paper, draw the different parts and the communication that happens between them.

- Did you manage the termination process? Be sure all is closed once requested!

- Did you take in mind the system limitations that we are using? Be sure to use it to its fullest

-NASA API limit

-20 instances limit

- Are all your workers working hard? Or some are slacking? Why?

- add a feature to support that

-Lastly, are you sure you understand what distributed means? Is there anything in your system awaiting another?