Université Jean Monnet Saint-Etienne

Et

Ecole des Mines de Saint-Etienne

Cloud Computing

Encadré par :

Prof. Charlotte Laclau

Réalisé par :

1. Mourad benkaraache
3. Youssef lamzaouak
4. Mohamad Nour Badr
4. Rediet Tadesse
5. Jehad Melad

Année Académique:

2020-2021

Content:

- ## Introduction to Cloud Computing:

Technology potential:

To explain the potential of cloud computing, it is useful to start with a definition. Cloud computing is commonly defined as the deployment of IT resources as a service over the Web and other networks. Those resources—such as processing power, storage, computing platforms, and applications are remote from the user, or “in the clouds,” and are paid for only as they are used.

The nature of computing changes when IT resources become more abundant and can be instantly and affordably deployed, flexibly managed, and universally accessed from a broad range of devices, including PCs and mobile handsets. This “variabilization” of both the costs and scope of IT resources is central to the promise of cloud computing and helps explain the gathering interest in it. In the enterprise environment, cloud computing has the following advantages, not only for the CIO but also for all IT users:

◊ Accelerated deployment of new applications without consuming the computing resources of the enterprise

◊ Reduced capital requirements for up-front investments in IT since the enterprise is able to utilize the infrastructure, applications, and platforms in the cloud

◊ The flexibility to meet sudden changes in demand.

◊ The capability to provide applications or services that meet demand precisely and can match future demand.

◊ Significant cost savings in selected situations, notably when the scale of an enterprise’s computing resources is relatively small compared with that of cloud providers.

Cloud Providers:

Different are the cloud providers of this technology and each one of them has its pricing policy ,methods of orchestration and his own infrastructure but the major players are :Google cloud ,Microsoft Azure Amazon Web services.the last one is the one we will be using in this lab and from each wide services we are going to use just 2 which are EC2 and S3.

EC2:

Amazon Elastic Compute Cloud or EC2 is a service offered by Amazon that allows third parties to rent servers on its own applications.

S3: Amazon S3 (Amazon Simple Storage Service) is a file hosting site offered by Amazon Web Services.

- Statistical Computing Application:

Introduction:

The AWS cloud has several services enabling users to get work done as expected due to quality, performance, availability, and cost . In this part of the report we are launching an application consisting of Client file **`client.py`** and worker **`serverEC2.py`**. This application exchanges data among the client and the worker (server). In which the client sends a request asking for results and in return the worker processes the request and sends the result back to the client. This interaction has been done by some AWS services and at the end of this interaction the worker will save the transaction on the S3 cloud bucket.

Pre-requirements:

Considering this report there are some necessities (e.g. programming language and modules or packages) to run the application properly by installing:

- Python environment
- Aws cli
- Boto3
    - SQS
    - EC2
    - S3
- Other packages used:
    - Logging
    - Time
    - botocore

Mechanism:

Setting Up all the pre-requirements will allow us to start to create our application.

Client side

- First of all, the client will start to importing all packages and calls a required services (i.e. SQS). As a starting point, it will create a queue with name **`"InputQueue"`** then will call the queue by its name. In the following shows important lines:
    - `boto3.resource('sqs')`
    - `sqs.create_queue(QueueName=AWS_SQS_QUEUE_NAME_INPUT)`

    - `sqs.get_queue_by_name(QueueName=AWS_SQS_QUEUE_NAME_INPUT)`
- In which to send the request (List of numbers) to this particular queue. We used:
    - `self.inputqueue.send_message(MessageBody=message)`
- After the client sends the request it waits for a response from the worker which will be retrieved after the worker processes it and gets a result.

Worker side:

The idea is to run a worker file using an EC2 instance of AWS, therefore after finishing the coding on the worker we need to install its requirements on the instance. After that we upload a worker file on the instance and run it.

- `User-ec2 # python3 serverEC2.py`

On the worker side the worker will use SQS and S3 services. Therefore it will start by creating a SQS queue called **`"OutputQueue"`** to send the reply to the client by this queue. Then It will call the queue of the requests **`”InputQueue”`** to access all requests in this queue. Therefore the worker will get the client request from **`”InputQueue”`** and will process it (calculate the results).

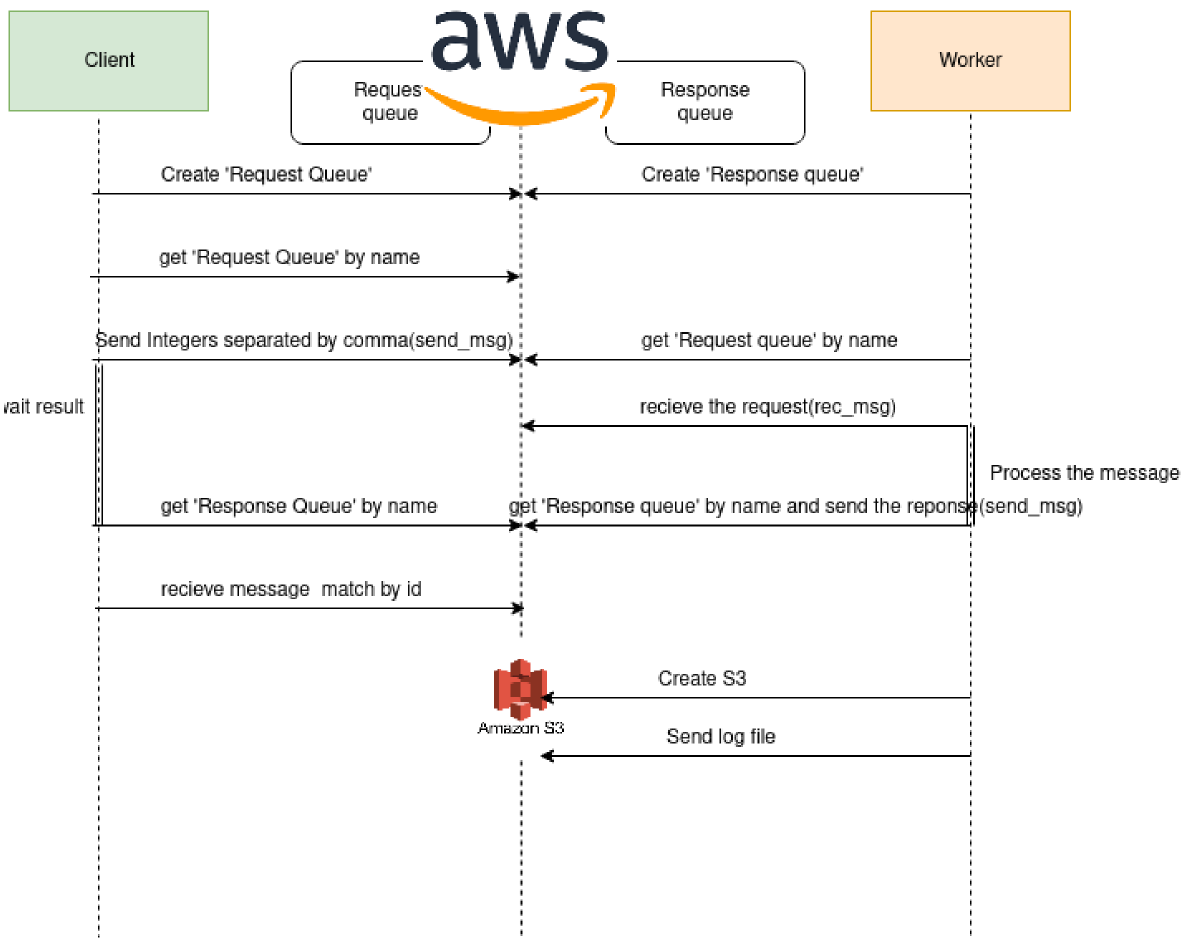
- `self.inputqueue = sqs.get_queue_by_name(QueueName=AWS_SQS_QUEUE_NAME_INPUT)`
- `self.queue = sqs.create_queue(QueueName=AWS_SQS_QUEUE_NAME_OUTPUT)`
- `queue = sqs.get_queue_by_name(QueueName=AWS_SQS_QUEUE_NAME_OUTPUT)`

Right after the results obtained, the worker will send the answers to the response queue **`"OutputQueue"`**. While The client was waiting on this queue **`"OutputQueue"`** for a result.

- `response = self.queue.send_message(MessageBody=message)`

The worker will send the results to the client and all these transactions need to be stored. Due to that, on the worker side (**`serverEC2.py`**), it creates a bucket to store a log file (log.txt) which contains all transactions thanks to S3 service.

- `s3_client = boto3.client('s3')`
- `response = s3_client.upload_file(file_name, bucket, object_name)`


Client
aws
Reques queue
Response queue
Worker
Create 'Request Queue'
Create 'Response queue'
get 'Request Queue' by name
Send Integers separated by comma(send_msg)
get 'Request queue' by name
vait result
recieve the request(rec_msg)
Process the message
get 'Response Queue' by name
get 'Response queue' by name and send the reponse(send_msg)
recieve message match by id
Create S3
Amazon S3
Send log file

- Image processing application:

Prequirements:

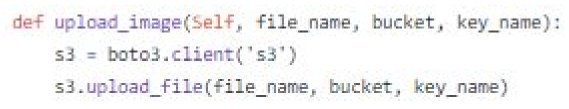
In addition to the previous requirements mentioned above ,the following packages are necessary for the code execution:

1. Skimage
2. Numpy
3. Matplotlib

Mechanism:

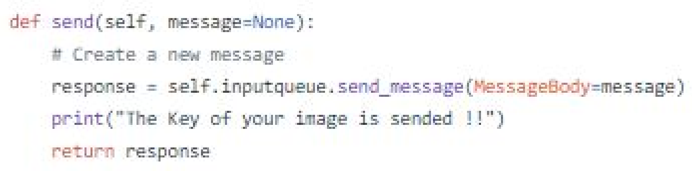
- **Client Side:**

1. First of all, the client will get to aws account ,then the required packages will be imported and calls an amazon queue service (i.e. SQS). As a starting point, it will create a queue with name **”InputQueue”** then will call the queue by its name as what we have done in the first part.
2. the second thing s that the client should have created an S3 bucket on his amazon account ,then he will upload the image to his bucket from his Desktop folder,then it will put this image on the provided S3 bucket name ,with the keyname that the user will choose.Here is the command :



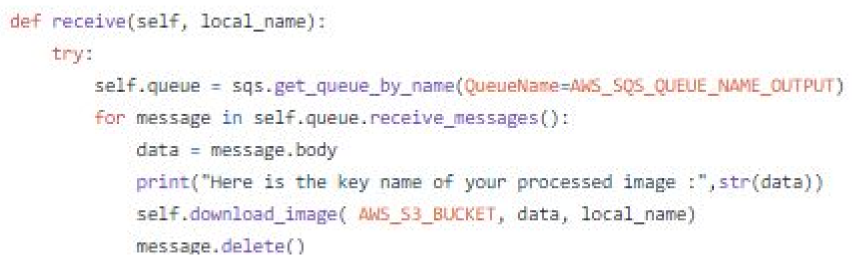
```
def upload_image(Self, file_name, bucket, key_name):
    s3 = boto3.client('s3')
    s3.upload_file(file_name, bucket, key_name)
```

3. In which to send the request (Keyname of the image to process) to this particular queue.the massage here will be replaced by the keyname of the image. Here is what we used:



```
def send(self, message=None):
    # Create a new message
    response = self.inputqueue.send_message(MessageBody=message)
    print("The Key of your image is sended !!")
    return response
```

4. After the client sends the request it waits for a response from the worker which will be retrieved after the worker processes it and gets a results by downloading the processed image from the S3 bucket .Here is the code:

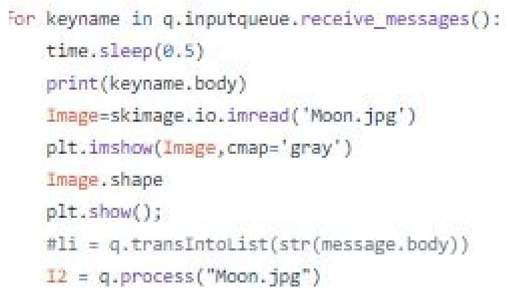


```python
def receive(self, local_name):
    try:
        self.queue = sqs.get_queue_by_name(QueueName=AWS_SQS_QUEUE_NAME_OUTPUT)
        for message in self.queue.receive_messages():
            data = message.body
            print("Here is the key name of your processed image :",str(data))
            self.download_image( AWS_S3_BUCKET, data, local_name)
            message.delete()
```

- **Worker Side:**

1. The idea is to run a worker file using an EC2 instance of AWS, therefore after finishing the coding on the worker we need to install its requirements on the instance. After that we upload a worker file on the instance and run it.
   - `User-ec2 # python3 serverEC2.py`

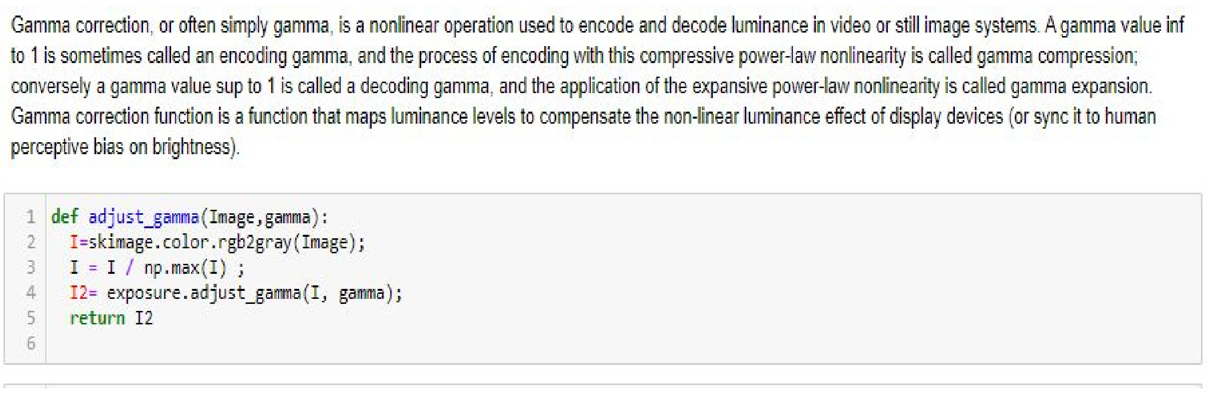
2. On the worker side the worker will use SQS and S3 services. Therefore it will start by creating a SQS queue called **`"OutputQueue"`** which contains yhe keyname of the image+ processed' then send the reply to the client by this queue. Then It will call the queue of the requests **`”InputQueue”`** which contains the keyname of the image to process .here is the commands:



```python
For keyname in q.inputqueue.receive_messages():
    time.sleep(0.5)
    print(keyname.body)
    Image=skimage.io.imread('Moon.jpg')
    plt.imshow(Image,cmap='gray')
    Image.shape
    plt.show();
    #li = q.transIntoList(str(message.body))
    I2 = q.process("Moon.jpg")
```

For the processing part ,many functions were proposed here are the functions and their impacts on the images.

1.Adjusting gamma:

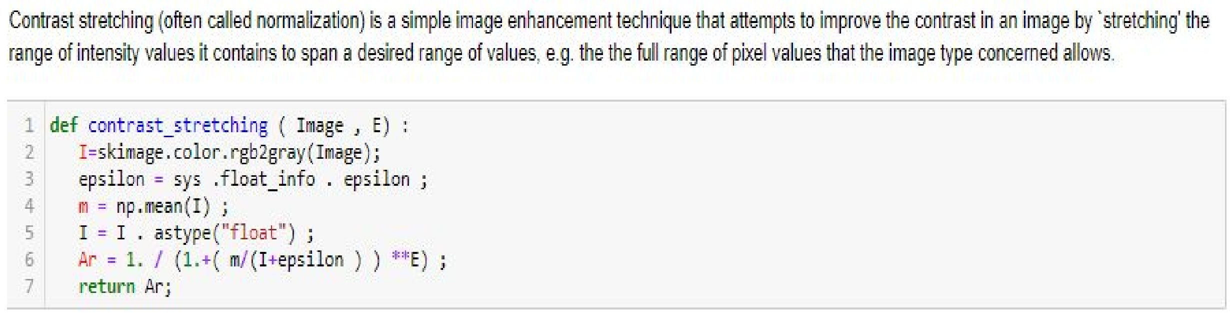


Gamma correction, or often simply gamma, is a nonlinear operation used to encode and decode luminance in video or still image systems. A gamma value inf to 1 is sometimes called an encoding gamma, and the process of encoding with this compressive power-law nonlinearity is called gamma compression; conversely a gamma value sup to 1 is called a decoding gamma, and the application of the expansive power-law nonlinearity is called gamma expansion. Gamma correction function is a function that maps luminance levels to compensate the non-linear luminance effect of display devices (or sync it to human perceptive bias on brightness).

```python
def adjust_gamma(Image,gamma):
  I=skimage.color.rgb2gray(Image);
  I = I / np.max(I) ;
  I2= exposure.adjust_gamma(I, gamma);
  return I2

```

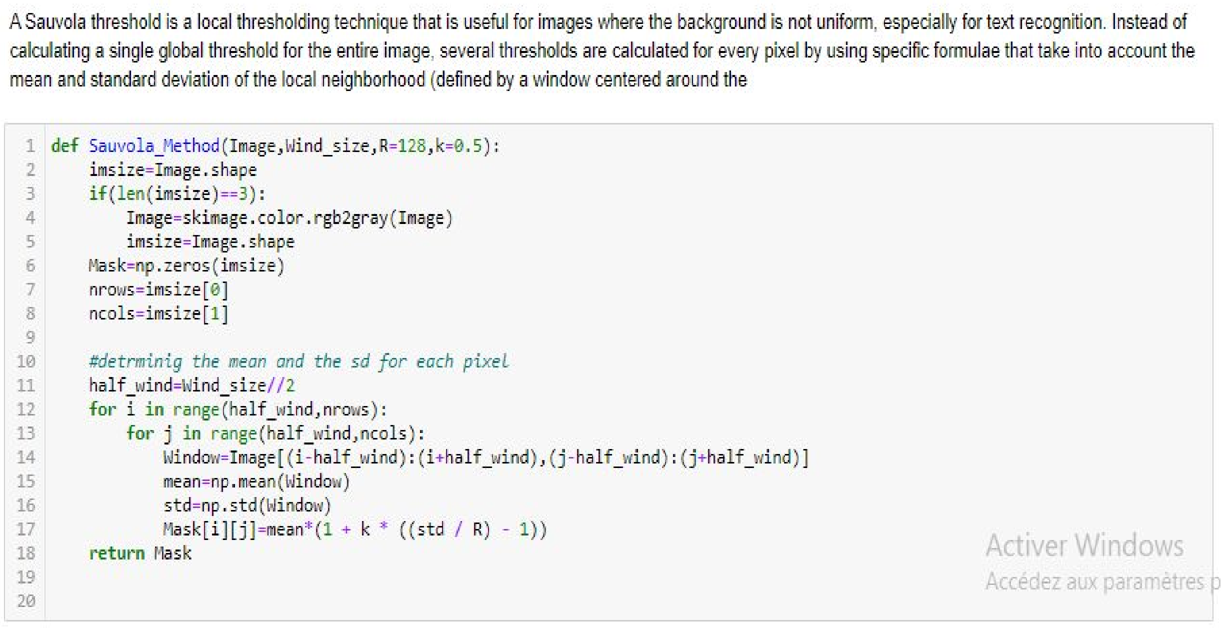
2.Contrast Stretching:



Contrast stretching (often called normalization) is a simple image enhancement technique that attempts to improve the contrast in an image by `stretching' the range of intensity values it contains to span a desired range of values, e.g. the the full range of pixel values that the image type concerned allows.

```python
def contrast_stretching ( Image , E) :
   I=skimage.color.rgb2gray(Image);
   epsilon = sys .float_info . epsilon ;
   m = np.mean(I) ;
   I = I . astype("float") ;
   Ar = 1. / (1.+( m/(I+epsilon ) ) **E) ;
   return Ar;
```

3-Sauvola Thresholding:



A Sauvola threshold is a local thresholding technique that is useful for images where the background is not uniform, especially for text recognition. Instead of calculating a single global threshold for the entire image, several thresholds are calculated for every pixel by using specific formulae that take into account the mean and standard deviation of the local neighborhood (defined by a window centered around the

```python
def Sauvola_Method(Image,Wind_size,R=128,k=0.5):
    imsize=Image.shape
    if(len(imsize)==3):
        Image=skimage.color.rgb2gray(Image)
        imsize=Image.shape
    Mask=np.zeros(imsize)
    nrows=imsize[0]
    ncols=imsize[1]

    #detrminig the mean and the sd for each pixel
    half_wind=Wind_size//2
    for i in range(half_wind,nrows):
        for j in range(half_wind,ncols):
            Window=Image[(i-half_wind):(i+half_wind),(j-half_wind):(j+half_wind)]
            mean=np.mean(Window)
            std=np.std(Window)
            Mask[i][j]=mean*(1 + k * ((std / R) - 1))
    return Mask


```

4. Right after the image is processed, the worker will send the answers to the response queue `"OutputQueue"` and will delete the keyname of the image allowing the coming of a new one.

- Conclusion:

As we have seen in Introduction ,Cloud Computing is a disruptive Technology that allows the entreprises no matter size they have or Technology they dispose ,they could also access the same capabilities as the giants .From these two applications we have been able to see how is the cloud is deployed for doing differents Computing Tasks-Calculating the min ,the max and the average and processing an image too.But the following question that raised in our minds is:Is there a possibility that the cloud provider could give the computing power that we need without accessing our data?We think this feature will be more and more asked from cloud provider if they want always to stay on the market.