Capstone Write-up

Xueyin Wang **Total Time: 45 hours**

Accessibility:

Capstone: ec2-52-201-160-246.compute-1.amazonaws.com
Utilities: ec2-52-91-18-43.compute-1.amazonaws.com
web: https://xueyin.ucmpcs.org/

Source code in S3:

https://s3.amazonaws.com/mpcs-students/xueyin/mpcs_final_project.zip

Main python files:

jobs_runner.py, run.py: mpcs-Utils/mpcs_anntools/
results_notify.py, glacier_notify.py: mpcs-Utils/
script to test annotator scale-out: mpcs-Utils/annotator_test.py

Design decision:

Archive data to Glacier:

A message queue is used to dealing with checking time interval in background. In run.py, after the job finishes, it not only publishes a notifincation to `xueyin_results_queue` to inform the job is done, but also publish a notification inlcuding attribute `complete_time`, `input_file_name` etc to `xueyin_glacier_queue`, with its message type `ARCHIVE`.

When the message queue get the `ARCHIVE` type message, it will always calculate the time interval from `complete time` to current time, and check with several criteria `(int(time.time()) - complete_time) > free_user_limit and isObjExist(obj) and checkUserRole(usrname) == free_user`. If it's TURE, move the file to `gas-archive` bucket and delete this message.

Restore data:

When the user upgrade to `premium_user`, first filter out all its objects in `gas-archive`, with each object sending a message to `xueyin_glacier_queue`. If the object is in `Glacier` Storage Class, call `obj.restore_object()` method to restore it from Glacier.

```
objs = bucket.objects.filter(Prefix = key_prefix)
for obj in objs:
    if obj.key.split("#")[0] == key_prefix + username:

        if obj.storage_class == 'GLACIER' and obj.restore is None:
            resp = obj.restore_object(
                RequestPayer='requester',
                RestoreRequest={'Days': 1}
            )
```

```
         content = {"job_id":"", "username":username,
"time":int(time.time()), "result_file":obj.key, "input_file":"",
"status": "COMPLETED", "type":"restore"}
        sns = boto3.resource('sns')
        topic = sns.Topic(glacier_arn)
        response = topic.publish(
            Message=json.dumps(content),
        )
```

The message sent here has type of `RESTORE`. When the message queue receives `RESTORE` type messages, check with its Storage Class and Glacier attribute: `obj.storage_class != 'GLACIER' or 'ongoing-request="false"' in restore`. If it's TRUE, move these objects back to `gas-input` and `gas-result` bucket for user to access to.

Error handling:

In mpcs_app.py, it mainly handle `404 Not Found` and `500 Internal Server Error`. When the `HTTPError` is caught, `my_handler` is triggered to deal with the error.

```
def my_handler(code, msg, error):
return template(request.app.config['mpcs.env.templates'] +
'error_page', code=code, msg=msg, error=error, auth=auth)

@error(404)
def error404(error):
    msg = "   Resource Not Found"
    return my_handler(404, msg, error)

@error(500)
def error500(error):
    msg = "   Internal Server Error"
    return my_handler(500, msg, error)
```

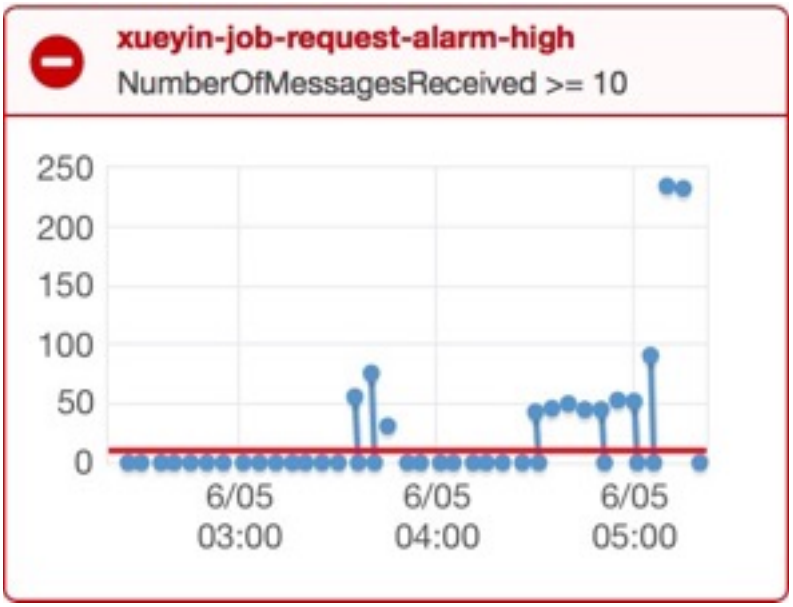
Scalability Tesing:

Capstone web server:

Bees are prespared to attack. During the attacking period, the number of healthy hosts is increasing.

Annotator:

Run the script `annotator_test.py` to test the scalability of annotator. The alarm will



alarm when the high bound is reached, and one server would be added each time alarming.