COL733: Fundamentals of Cloud Computing Semester II, 2021-2022

Lab-1: Batch processing 10 January 2022

Submission Instructions

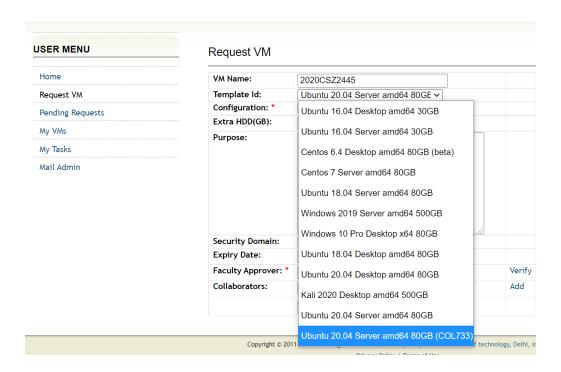
- 1. You can **only** use Python for this Lab. You are restricted to using only the following components: Celery, RabbitMQ, and Redis, already installed in the virtual machine. **Use of any other libraries** will lead to zero marks in the Lab.
- 2. You will submit the source code in **zip** format to <u>Moodle</u> (Lab 1). The naming convention of the zip file should be <Entry_Number>_<First_Name>.zip.

 Additionally, you need to submit a **pdf** for analysis questions on <u>Gradescope</u> (Lab1: Analysis).
- 3. The Lab would be **auto-graded**. Therefore, **follow** the same naming conventions described in the Deliverables section. Failing to adhere to these conventions will lead to zero marks in the Lab.
- 4. You should write the code without taking help from your peers or referring to online resources except for documentation. The results reported in the report should be generated from Baadal-VM. Not doing any of these will be considered a breach of the honor code, and the consequences would range from zero marks in the Lab to a disciplinary committee action.
- 5. You can use Piazza for any queries related to the Lab.

Setup Instructions

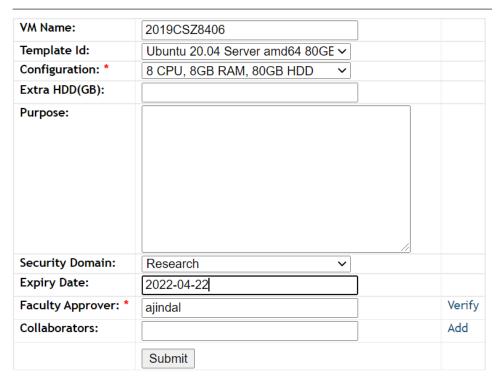
How to get your Virtual Machine?

- Go to BaadalVM website to request a VM https://baadal.iitd.ac.in/user/request_vm
- Use `Ubuntu 20.04 Server amd64 80GB (COL733)` in Template ID.



• Use your entry number as the VM name. Choose 8 CPU, 8GB RAM, 80GB HDD in configuration and 22 April 2022 as the VM expiry date. Add `ajindal` as faculty approver and submit.

Request VM



• Once the VM is created, you will be able to check the VM's Private IP by clicking `My VMs`. Please submit this VM IP to this Google Form.



Using your Virtual Machine

- If you're outside the IITD campus, you will first need to get VPN access. See here: <u>VPN instructions</u>.
- After verifying that you're able to ssh into a CSC machine and after receiving your VM IP, ssh into your VM as a `student` user with password `pass123`. You may receive an email with a default username (baadalvm) and password during VM creation, but please ignore that and use the username `student`.

```
$ ssh student@<YOUR_PRIVATE_IP>
```

• You can change your password after your first login by running:

```
$ passwd student
```

Note: Remember to note this password. If you forget your password, there may not be any way to recover it.

Starting RabbitMQ

• Verify if the **rabbitmq-server** is running using the `ps -ef | grep rabbit` command:

```
| grep rabbit
                                                                                                                                                          00:00:00 /bin/sh /usr/sbin/rabbitmq-server 00:00:07 /usr/lib/erlang/erts-10.6.4/bin/beam.smp -W w
                                                                                           0 23:03 ?
  rabbitmq
                                                                         2329 23 23:03 ?
                                            2350
   -A 128 -MBas ageffcbf -MHas ageffcbf -MBlmbcs 512 -MHlmbcs 512 -MMmcs 30 -P 1048576 -t 5000000 -st
 bt db -zdbbl 128000 -K true -- -root /usr/lib/erlang -progname erl -- -home /var/lib/rabbitmq --
bt db -zdbbl 128000 -K true -- -root /usr/lib/erlang -progname erl -- -home /var/lib/rabbitmq -- -pa /usr/lib/rabbitmq/lib/rabbitmq_server-3.8.2/ebin -noshell -noinput -s rabbit boot -sname rabbit t@baadalvm -boot start sasl -kernel inet_default_connect_options [{nodelay,true}] -rabbit tcp_list eners [{"127.0.0.1",5672}] -sasl errlog_type error -sasl sasl_error_logger false -rabbit lager_log_root "/var/log/rabbitmq" -rabbit lager_default_file "/var/log/rabbitmq/rabbit@baadalvm.log" -rabbit lager_upgrade_file "/var/log/rabbitmq/rabbitmdadalvm.log" -rabbit lager_upgrade_file "/var/log/rabbitmq/rabbit@baadalvm.plags" -rabbit enabled_plugins file "/etc/rabbitmq/enabled_plugins" -rabbit plugins_dir "/usr/lib/rabbitmq/plugins:/usr/lib/rabbitmq/lib/rabbitmq/server-3.8.2/plugins" -rabbit plugins_expand_dir "/var/lib/rabbitmq/mnesia/rabbit@baadalvm-plugins_expand" -os_mon_start_cpu_sup_false -os_mon_start_disksup_false -os_mon_start_memsup_false -mnesia_dir_"/var/lib/rabbitmg/mnesia/rabbit@baadalvm" -rad data_dir_"/var/lib/rabbitmg/mnesia/rabbit@baadalvm" -rad data_dir_"/var/lib/rabbitmg/mnesia/rabbit@baadalvm
 dir "/var/lib/rabbitmq/mnesia/rabbit@baadalvm" -ra data_dir "/var/lib/rabbitmq/mnesia/rabbit@baadalvm/quorum" -kernel inet_dist_listen_min 25672 -kernel inet_dist_listen_max 25672 --
rabbitmq 2723 2350 0 23:03 ? 00:00:00 erl_child_setup 65536
                                                                        2350
2723
                                                                                                                                                          00:00:00 inet gethost 4
                                            2760
                                                                                             0 23:03 ?
   rabbitmq
   rabbitmq
                                             2761
                                                                                             0 23:03 ?
                                                                                                                                                          00:00:00 inet gethost 4
                                                                                            0 23:04 pts/1
                                                                                                                                                          00:00:00 grep_rabbit
```

The first two processes are indicating that rabbitmq-server is running.

• If the rabbitmq-server is not running, then in a screen tab, login as the `rabbitmq` user using the `su - rabbitmq` command. The password is also `rabbitmq`.

\$ su - rabbitmq

• Run `rabbitmq-server` command

\$ rabbitmq-server

Starting Redis

• In another screen tab, run the redis server. The `redis.conf` file is present in the home directory of the `student` user in the VM.

redis-server <PATH_TO_REDIS_CONFIG>

Dataset Description

The dataset is available at ~/data directory. Each CSV file contains 7 attributes, following are a brief description of each attribute:

- tweet_id: A unique, anonymized ID for the Tweet. Referenced by response_tweet_id and in_response_to_tweet_id.
- author_id: A unique, anonymized user ID. <u>@s</u> in the dataset have been replaced with their associated anonymized user ID.
- *inbound:* Whether the tweet is "inbound" to a company doing customer support on Twitter. This feature is useful when re-organizing data for training conversational models.
- created at: Date and time when the tweet was sent.
- *text*: Tweet content. Sensitive information like phone numbers and email addresses are replaced with mask values like __email__.
- response_tweet_id: IDs of tweets that are responses to this tweet, comma-separated.
- in_response_to_tweet_id: ID of the tweet this tweet is in response to, if any.

Problem Statement

Garima is a famous customer support researcher who is interested in analysing the customer-support-on-twitter dataset[1]. She wishes to see the word counts inside GBs or TBs of tweets data (*text* attribute of the dataset). She wrote a serial program, *serial.py*, but the code is neither scalable nor fault-tolerant.

Garima hired you to design a scalable and fault-tolerant word count application with a better execution latency. You are provided with a machine that has only the following packages: Celery, RabbitMQ and Redis.

Deliverables

• Source code: You need to provide the source code for the word counting application implemented using Celery. The source code should be in a .zip format and should be uploaded to moodle. A sample source code folder structure is shown below:

When we unzip the submission then we should see the above files in the aforementioned structure.

The python file containing celery tasks should be named tasks.py.

• Your celery tasks should be runnable by the following command:

```
celery -A tasks worker --loglevel=INFO --concurrency=4 -n
task@%h
```

 Your word-count application should be named *client.py* and runnable by the following command, where DATA_DIR (*e.g.:* ~/data/) points to the directory containing the tweets.

```
python3 client.py <DATA_DIR>
```

- Analysis: Answer the following questions on Gradescope (Lab 1: Analysis):
 - What is the best speedup achieved over serial.py?
 - Given a fixed input size, measure how the efficiency of the word-count application varies with an increase in worker threads allocated to the application. Justify.
 - Given a fixed worker thread (= 8) allocated to the application, measure how the efficiency of the word-count application varies with input size. Justify.
 - The designed solution is scalable. Justify.
 - The designed solution is fault-tolerant. Justify.

Rubrics (30 marks)

- 1. 2 marks: Correctness of word-count application with single worker node.
- 2. 3 marks: Correctness of word-count application with multiple worker nodes.
- 3. 5 marks: The word-count application is fault-tolerant.
- 4. 10 marks: This has relative grading. The faster programs on multiple worker threads will receive higher marks.
- 5. 10 marks: Justifications and analysis as requested in the deliverables.

References

[1]: https://www.kaggle.com/thoughtvector/customer-support-on-twitter