User-centric Systems for Data Science **Prof. John Liagouris**
Fall Semester 2022 **liagos@bu.edu**

Assignment 4

Submission deadline: December 10 2022, 11:59 pm

This is the last course assignment and focuses on performance analysis for distributed data processing applications. For this assignment, you will use:

- *Remote sensing project in the assignment-4 branch*
- *The Ray framework to distribute your Python code*
- *A tracing system (Jaeger) to instrument, collect, and visualize end-to-end application traces*
- *The New England Research Cloud (NERC) to deploy your application in a cluster of machines*

The first three tasks of the assignment can be done locally (on your computer) but the fourth task must be done on NERC. Setting up the cloud environment for the fourth task requires some time and we highly recommend that you start early (see detailed instructions below).

In contrast to previous assignments, this assignment requires less coding and more independent exploration of Jaeger, Ray, and NERC. Make sure you read the provided documentation carefully.

The assignment must be completed in teams of two students, however, discussion with other students is encouraged.

0. Create an account on NERC

Follow the steps in this Piazza post to ensure you have access to NERC. You will be added to a project which you can verify by logging into https://coldfront.mss.mghpcc.org/. Once done you will be able to log in to https://stack.nerc.mghpcc.org/ to begin creating your instances.

If you are unable to log in please reach out on Piazza so that you can complete Task 4 on time.

1. TASK I: OpenTelemetry tutorial (credits: 20/100)

The first task of this assignment is to familiarize yourself with Jaeger. OpenTelemetry has an excellent tutorial for beginners.

Hint #1: The tutorial suggests that you use Jaeger within a Docker container. To do so, you will first need to install Docker Desktop.

Hint #2: You may also want to have a look at the OpenTelemetry documentation.

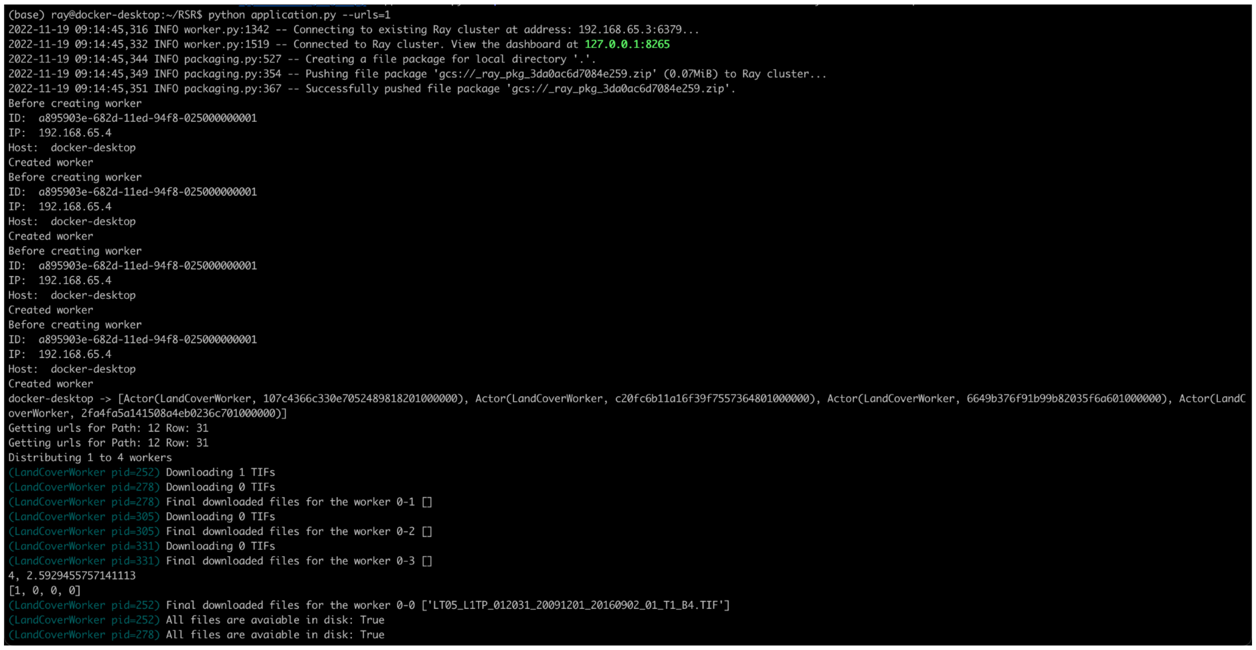
2. TASK II: Run the Remote Sensing project locally (credits: 20/100)

The second task is to run the remote sensing project locally.

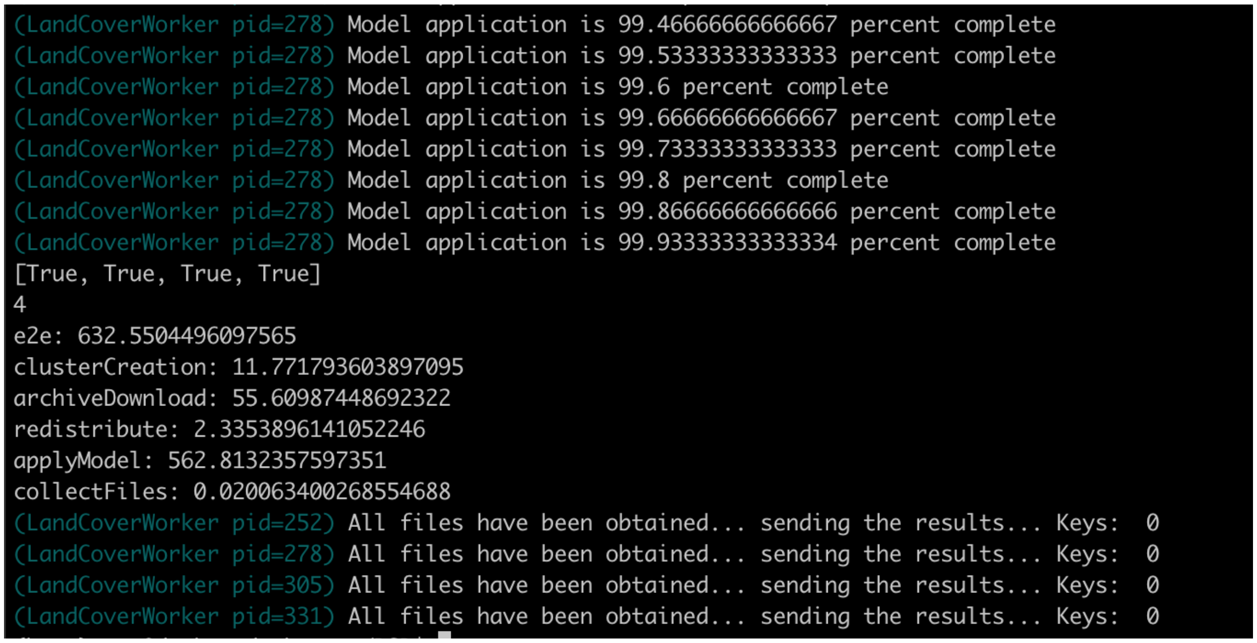
Prerequisites:

1. Please make sure that Docker Desktop is installed on your computer so as to run.
2. Pull the latest code from the main repository. You will find a new branch called **assignment-4**
3. Detailed instructions on how to run the code locally are given in the README.md file of the **RSR** folder.

The files `application.py` gives a sample program to run which downloads, redistributes and applies a mock model on the generated data. Go through the functions used in the `application.py` to understand the flow of the program as this is required to complete the next tasks. Push the related screenshot(s) of your terminal to your Gitlab repository.



```
(base) ray@docker-desktop:~/RSR$ python application.py --urls=1
2022-11-19 09:14:45,316 INFO worker.py:1342 -- Connecting to existing Ray cluster at address: 192.168.65.3:6379...
2022-11-19 09:14:45,332 INFO worker.py:1519 -- Connected to Ray cluster. View the dashboard at 127.0.0.1:8265
2022-11-19 09:14:45,344 INFO packaging.py:527 -- Creating a file package for local directory '.'.
2022-11-19 09:14:45,349 INFO packaging.py:354 -- Pushing file package 'gcs://_ray_pkg_3da0ac6d7084e259.zip' (0.07MiB) to Ray cluster...
2022-11-19 09:14:45,351 INFO packaging.py:367 -- Successfully pushed file package 'gcs://_ray_pkg_3da0ac6d7084e259.zip'.
Before creating worker
ID:  a895903e-682d-11ed-94f8-025000000001
IP:  192.168.65.4
Host:  docker-desktop
Created worker
Before creating worker
ID:  a895903e-682d-11ed-94f8-025000000001
IP:  192.168.65.4
Host:  docker-desktop
Created worker
Before creating worker
ID:  a895903e-682d-11ed-94f8-025000000001
IP:  192.168.65.4
Host:  docker-desktop
Created worker
Before creating worker
ID:  a895903e-682d-11ed-94f8-025000000001
IP:  192.168.65.4
Host:  docker-desktop
Created worker
docker-desktop -> [Actor(LandCoverWorker, 107c4366c330e7052489818201000000), Actor(LandCoverWorker, c20fc6b11a16f39f7557364801000000), Actor(LandCoverWorker, 6649b376f91b99b82035f6a601000000), Actor(LandC
overWorker, 2fa4fa5a141508a4eb0236c701000000)]
Getting urls for Path: 12 Row: 31
Getting urls for Path: 12 Row: 31
Distributing 1 to 4 workers
(LandCoverWorker pid=252) Downloading 1 TIFs
(LandCoverWorker pid=278) Downloading 0 TIFs
(LandCoverWorker pid=278) Final downloaded files for the worker 0-1 []
(LandCoverWorker pid=305) Downloading 0 TIFs
(LandCoverWorker pid=305) Final downloaded files for the worker 0-2 []
(LandCoverWorker pid=331) Downloading 0 TIFs
(LandCoverWorker pid=331) Final downloaded files for the worker 0-3 []
4, 2.5929455757141113
[1, 0, 0, 0]
(LandCoverWorker pid=252) Final downloaded files for the worker 0-0 ['LT05_L1TP_012031_20091201_20160902_01_T1_B4.TIF']
(LandCoverWorker pid=252) All files are avaiable in disk: True
(LandCoverWorker pid=278) All files are avaiable in disk: True
```



```
(LandCoverWorker pid=278) Model application is 99.46666666666667 percent complete
(LandCoverWorker pid=278) Model application is 99.53333333333333 percent complete
(LandCoverWorker pid=278) Model application is 99.6 percent complete
(LandCoverWorker pid=278) Model application is 99.66666666666667 percent complete
(LandCoverWorker pid=278) Model application is 99.73333333333333 percent complete
(LandCoverWorker pid=278) Model application is 99.8 percent complete
(LandCoverWorker pid=278) Model application is 99.86666666666666 percent complete
(LandCoverWorker pid=278) Model application is 99.93333333333334 percent complete
[True, True, True, True]
4
e2e: 632.5504496097565
clusterCreation: 11.771793603897095
archiveDownload: 55.60987448692322
redistribute: 2.3353896141052246
applyModel: 562.8132357597351
collectFiles: 0.020063400268554688
(LandCoverWorker pid=252) All files have been obtained... sending the results... Keys:  0
(LandCoverWorker pid=278) All files have been obtained... sending the results... Keys:  0
(LandCoverWorker pid=305) All files have been obtained... sending the results... Keys:  0
(LandCoverWorker pid=331) All files have been obtained... sending the results... Keys:  0
```

3. TASK III: Trace your operator library on Ray (credits: 40/100)

The third task is to instrument the Remote Sensing project using Jaeger and generate traces for `application.py`. The result should be a typical span tree, like those you generated in Task I. Each span in the tree should represent a call of the functions mentioned in `application-jaeger.py`.

Explore the span tree using Jaeger's web-based UI. Push the related screenshot(s) to your Gitlab repository.

4. TASK IV: Distributed tracing on MOC (credits: 20/100)

The fourth and final task is to deploy your instrumented Remote Sensing Project on NERC and perform Task III in a distributed setting.

A. As a first step, you need to create a small cluster of 2 VMs on MOC of `cpu.a4` flavour. We recommend that you use `Ubuntu 20.04` machines. Make sure you activate a Floating IP ("Networks→Floating IPs") for at least one VM so that you can access the cluster from the outside world. To do so, you will also need to allow SSH connections ("Networks→Security Groups") and upload your public SSH key ("Key Pairs"). In the end, you should be able to access your cluster via SSH using the Floating IP.
B. The next step is to install docker on the server. Please check here for more details.
C. Copy your local files to each of the servers. The steps to run the head and worker nodes in each of the servers are given in the README.md file in the RSR folder.
D. Setup Jaeger on the head node as done in Task I. You will need to change the hostname on `application-jaeger.py` to the hostname of your head node.
E. Push the related Jaeger screenshot(s) to your Gitlab repository.

6. Testing

You must have a simple test for each operator you implement and we strongly recommend using Pytest for this purpose. In case you are not familiar with Pytest, you might want to first spend some time reading the code snippets provided in the documentation.

All test functions must be added to the separate `tests.py` file provided with the code skeleton. Before submitting your solution, make sure the command `pytest tests.py` runs all your test functions successfully.

7. Logging

Logging can save you hours when debugging your program, and you will find it extremely helpful as your codebase grows in size and complexity. Always use Python's logger to generate messages and avoid using `print()` for that purpose. Keep in mind that logging has some performance overhead even if set to INFO level.

8. Git

You will use git to maintain your codebase and submit your solutions. If you are not familiar with git, you might want to have a look here. Note that well-documented code is always easier to understand and grade, so please make sure your code is clean, readable, and has comments.

Each time you finish a task (including the related tests), we strongly recommend that you use a commit message “Complete *Task X*”. You will find these commits very helpful in case you want to rollback and create new branches in your repository.

9. Deliverables

Each submission must be marked with a commit message “*Submit Assignment X*” in git. If there are multiple submissions with the same message, we will only consider the last one before the deadline. **We cannot accept late submissions for Assignment #4**. **All solutions must be submitted by December 10th at 11.59pm.**

Your submission must contain:

1. The code you wrote to solve the assignment tasks
2. The screenshots you generated as images (in a new folder `traces`). Use the following naming convention for screenshots: `task_x_screenshot_y.png`

Before submitting your solution, always make sure your code passes all tests successfully.

10. Resources

- Jaeger: https://www.jaegertracing.io
- NERC documentation: https://nerc-project.github.io/nerc-docs/openstack/
- OpenTelemetry: https://opentelemetry.io/
- Python tutorial: https://docs.python.org/3/tutorial/
- Python logger: https://docs.python.org/3/library/logging.html
- Pytest: https://docs.pytest.org/en/stable/
- Ray documentation: https://docs.ray.io/en/latest/
- Git Handbook: https://guides.github.com/introduction/git-handbook/