National University
of computer and emerging sciences

PCN Lab Report

Day 1

Fatima Qurban

25/06/24

Ray:

Ray is an open-source unified framework designed for scaling Python applications across multiple nodes and handling distributed computing tasks.

Ray AI Runtime is a unified framework for **Data, Train, Tune, Serve**

Why do we need Ray:

In an end to end ML application. We have four steps; **Data preprocessing, Training, hyper parameter Tuning and Serving**. All of these steps run on distributed systems (Each node performs its own tasks independently but collaborates with others to accomplish larger goals). We can also scale each of these systems. We would need to stitch them all together to build our end to end pipeline. However, this solution is far from optimal; because;

1. Hard to develop.
2. Hard to deploy
3. Hard to manage
4. Slow(due to communication overhead, transfer of data)

Instead of using different system to scale the ML pipeline, with ray we can use different libraries running on top of Ray to scale each of these stage

Ray is able to implement an entire ML application using a single system.

Ray Functions:

1. **@ray.remote**

   Transforms a Python function into a Ray remote function, allowing it to be executed in parallel.

   E.g @ray.remote

   def add(x, y):

   return x + y

   # Calling the remote function

   result = add.remote(2, 3)

2. **ray.init()**

   Initializes the Ray runtime, connecting to an existing Ray cluster if specified.python

3. **ray.get()**

Retrieves the result of a remote function call.

E.g result = ray.get(add.remote(2, 3))

print(result) # Output: 5

4. **ray.put()**

   Puts an object in the Ray object store, returning an ObjectRef that can be shared among tasks.

   E.g ref = ray.put([1, 2, 3])

   data = ray.get(ref)

   print(data)

The FAST compute model (components of Rays):

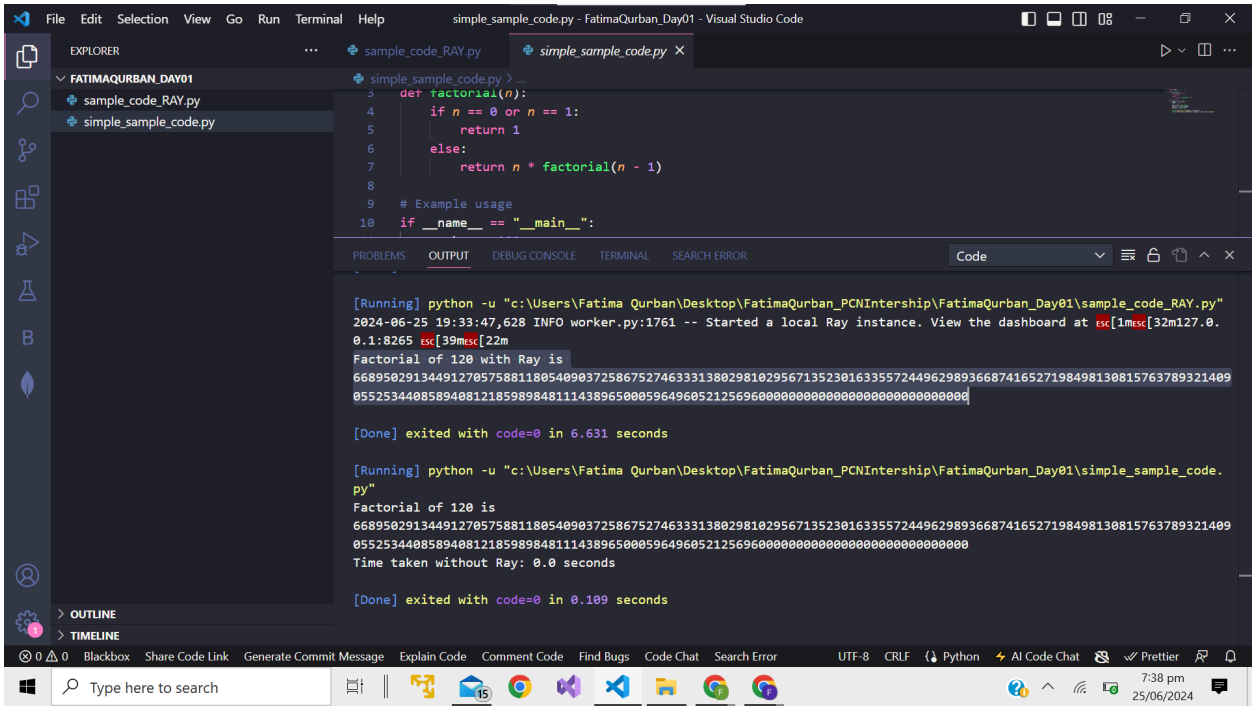
Actors: Remote **class** instance.

Tasks: In Ray, tasks are created when you call a remote **function.**

Futures: Reference to objects which either a function or a class return. When you invoke a remote function in Ray, it immediately returns an ObjectRef (a future) that can be used to access the result once the computation is complet

Shared In-Memory Distributed Object Stores in Ray allow different tasks and actors to share data efficiently by storing objects in a distributed, in-memory store.

Implementation of RAY:



Although we studied, using Ray for parallel processing should speed up computations by distributing the workload across multiple CPU cores or even machines. However we are seeing opposite effects because of the reason ray is used for computing intensive tasks like training the machine etc. However small-scale tasks like computing the factorial doesn't require much computing power leading to taking much more time because of the init **overhead head of the ray**, **managing remote tasks**, and **handling the parallel execution** outweighs the benefits of parallelism for such a small and quick computation.