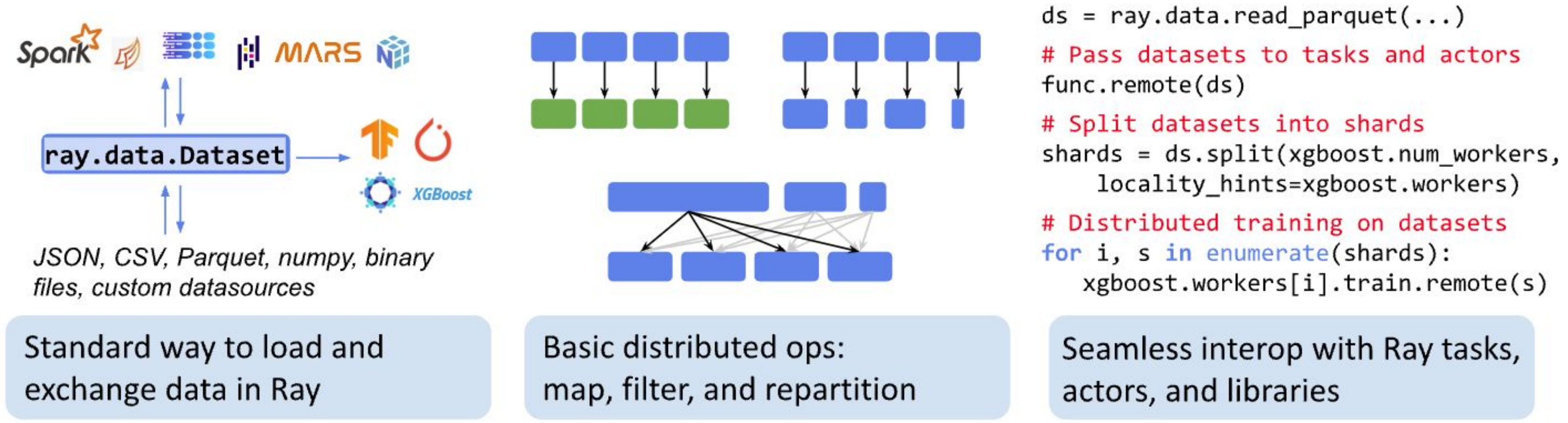
PA2 Discussion Session

DSC 204a, Winter 2024

Programming Assignment 2

- Introduction to Ray Data: simple ops on a multi-node setup (20, Easy)
- MapReduce with Ray Actors (40, Easy)
- Collective Communication with Ray (40, Medium/Hard)

Ray Data



```
ds = ray.data.read_parquet(...)
# Pass datasets to tasks and actors
func.remote(ds)
# Split datasets into shards
shards = ds.split(xgboost.num_workers,
    locality_hints=xgboost.workers)
# Distributed training on datasets
for i, s in enumerate(shards):
   xgboost.workers[i].train.remote(s)
```

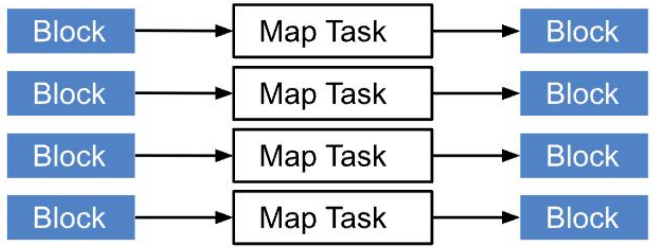
Standard way to load and exchange data in Ray

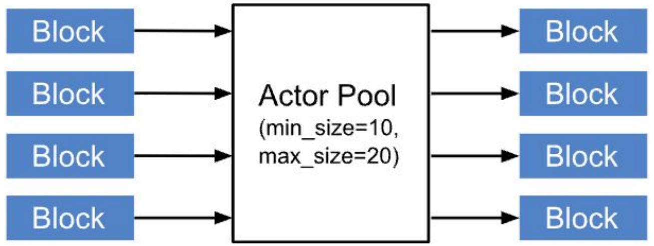
Basic distributed ops: map, filter, and repartition

Seamless interop with Ray tasks, actors, and libraries

Source

Ray Data: Transform Internals



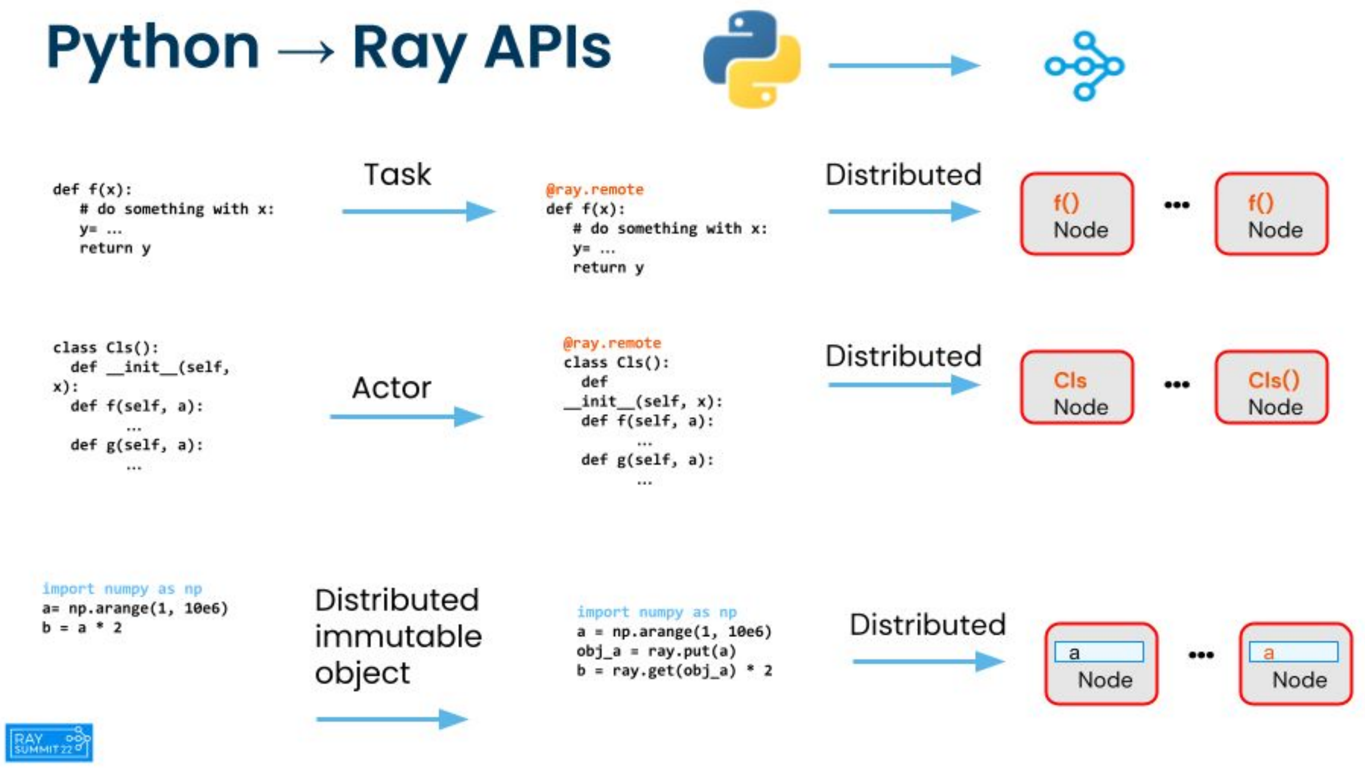


```
ds.map_batches(stateless_fn)
```

```
ds.map_batches(callable_cls,
    compute=ActorPoolStrategy(
        min_size=10, max_size=20,
    ))
```

Source

Revisiting Ray Core: Ray Actors



```
def f(x):
    # do something with x:
    y= ...
    return y
```

```
@ray.remote
def f(x):
    # do something with x:
    y= ...
    return y
```

```
class Cls():
  def __init__(self,
x):
  def f(self, a):
      ...
  def g(self, a):
      ...
```

```
@ray.remote
class Cls():
  def
__init__(self, x):
  def f(self, a):
      ...
  def g(self, a):
      ...
```

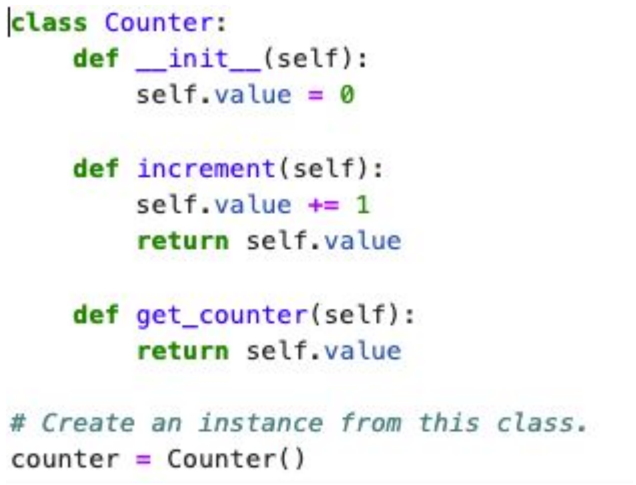
```
import numpy as np
a= np.arange(1, 10e6)
b = a * 2
```

```
import numpy as np
a = np.arange(1, 10e6)
obj_a = ray.put(a)
b = ray.get(obj_a) * 2
```

Source

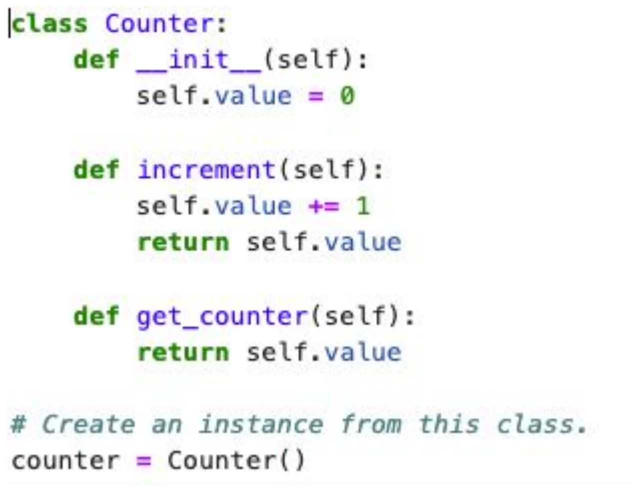
Motivation

- Come back to single-process land
    - How would a simple Counter look like?
- How can you use this in a multi-process setting?
- Multi-process: How do you handle concurrent updates?
- Multi-process: How can we do this in a multi-node setting?
- Multi-process: Can I keep code complexity to a minimum?

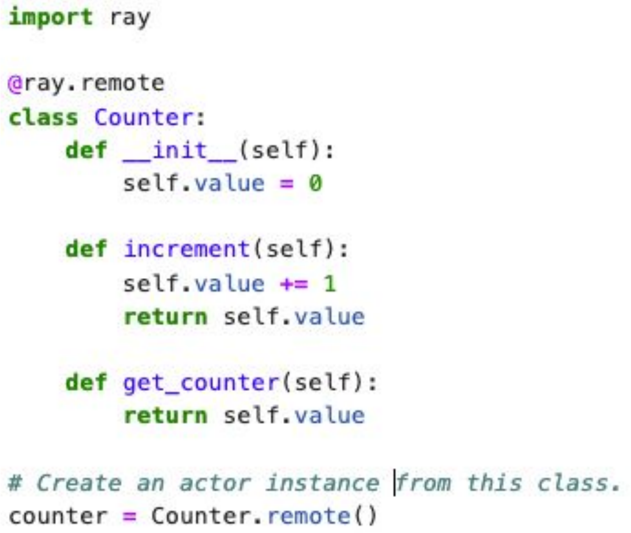


```python
class Counter:
    def __init__(self):
        self.value = 0

    def increment(self):
        self.value += 1
        return self.value

    def get_counter(self):
        return self.value

# Create an instance from this class.
counter = Counter()
```

Motivation



```python
class Counter:
    def __init__(self):
        self.value = 0

    def increment(self):
        self.value += 1
        return self.value

    def get_counter(self):
        return self.value

# Create an instance from this class.
counter = Counter()
```

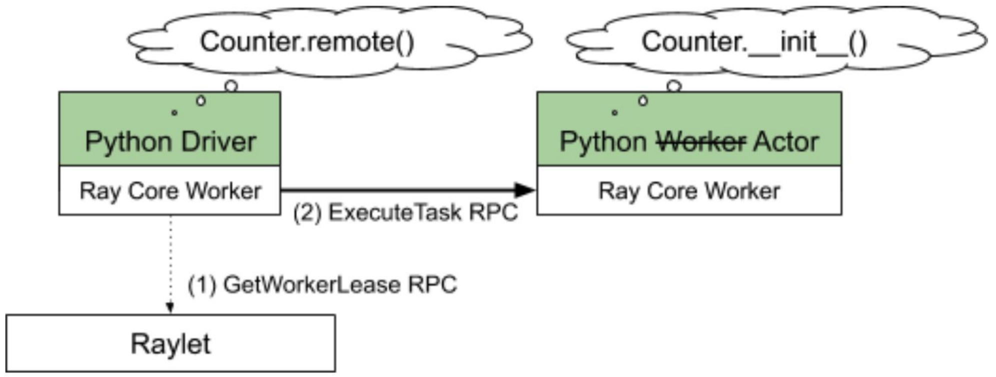


```python
import ray

@ray.remote
class Counter:
    def __init__(self):
        self.value = 0

    def increment(self):
        self.value += 1
        return self.value

    def get_counter(self):
        return self.value

# Create an actor instance from this class.
counter = Counter.remote()
```

Ray Actors: Checks all the boxes!

Ray Actors

- Recall: Ray tasks are remote functions. Executed asynchronously.
- Internals: A worker process handles the execution of a ray task.
- Ray Actor: A worker process with state.

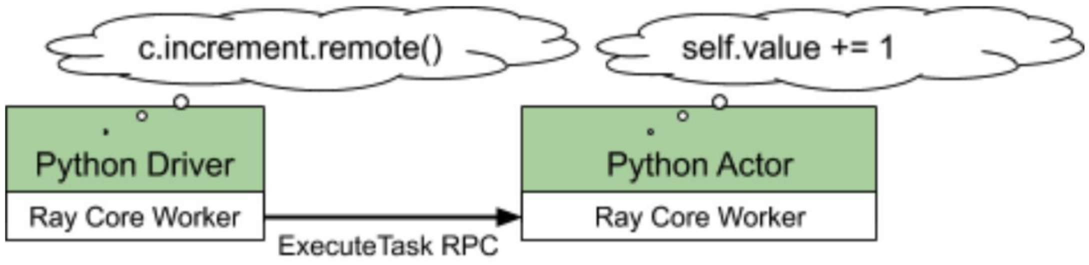


Creating an actor is done by scheduling the __init__ task for the actor. That actor creation task leases the worker forever until the actor is destroyed.

Source

Ray Actors

- The actor process handles all executions of methods and updates its state.



Once created, actor tasks translate into direct gRPC calls to the actor process. An actor can handle many concurrent calls, though here we only show one.

Source

Task 2

- Implement MapReduce with Ray Actors
- Prereq - MapReduce basics:
  - References: Definition, Whitepaper , Ray task implementation
  - Covered in class on Friday (Feb 23)
- Prereq - Ray Actors:
  - More in the discussion notebook

Task 3

- Implement AllReduce with Ray’s Collective Communication Lib
- Fun Fact: This lib was written by Prof. Hao Zhang!
- CPU-only communication with GLOO backend
- More in the discussion notebook

References

- https://docs.ray.io/en/latest/data/data-internals.html
- https://docs.ray.io/en/latest/data/data.html
- https://docs.ray.io/en/latest/ray-core/actors.html
- https://github.com/ray-project/ray-educational-materials