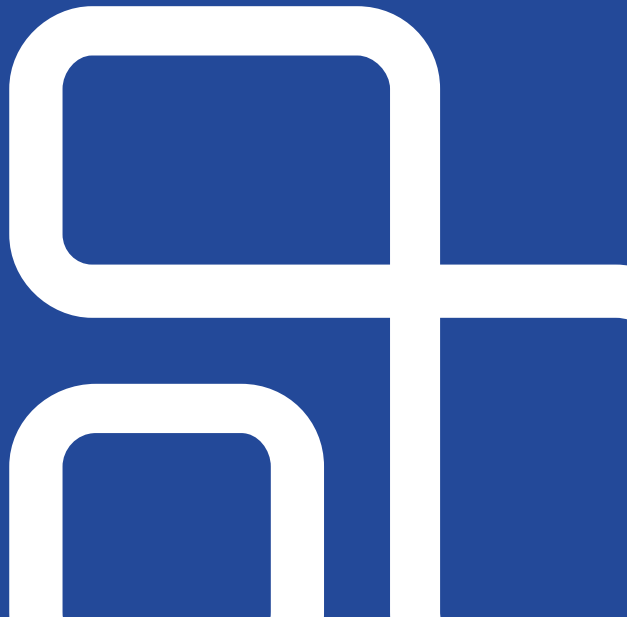
Anyscale Devops



anyscale

Agenda Today

- Anyscale Systems Overview
- CLI - Anyscale from your laptop
- SDK - Scripting Anyscale
- Application Lifecycle
- Ray Serve

Instructors & TAs

- Instructor: Charles Greer
  - Solutions Architect @ Anyscale
  - Technical Point of Contact for Koch (find me on teams!)
- TAs
  - Product Team @ Anyscale

What is Ray?

RAY Ecosystem

What is Ray?

- A place to run tasks in a massively parallel way.
- A framework for executing and managing distributed computation.
- A set of high-level ML-oriented libraries that leverage this united computing environment.
- A community of thousands of developers and ML engineers.

What is Anyscale?

Company founded by the Creators of Ray

Ion Stoica (Leader RISELab, AMPLab, Co-Founder Databricks, Conviva),
Michael Jordan (Leader RISELab, AMPLab),
Robert Nishihara (Ph.D. Berkeley, Ray co-creator),
Philipp Moritz (Ph.D. Berkeley, Ray co-creator)

Experts in Distributed Computing + ML

ex-Uber, Stripe, Databricks, Google Brain

Backed by Top Tier VCs:

A16z, NEA, Intel Capital

Why are you here?

anyscale

Time to get Real

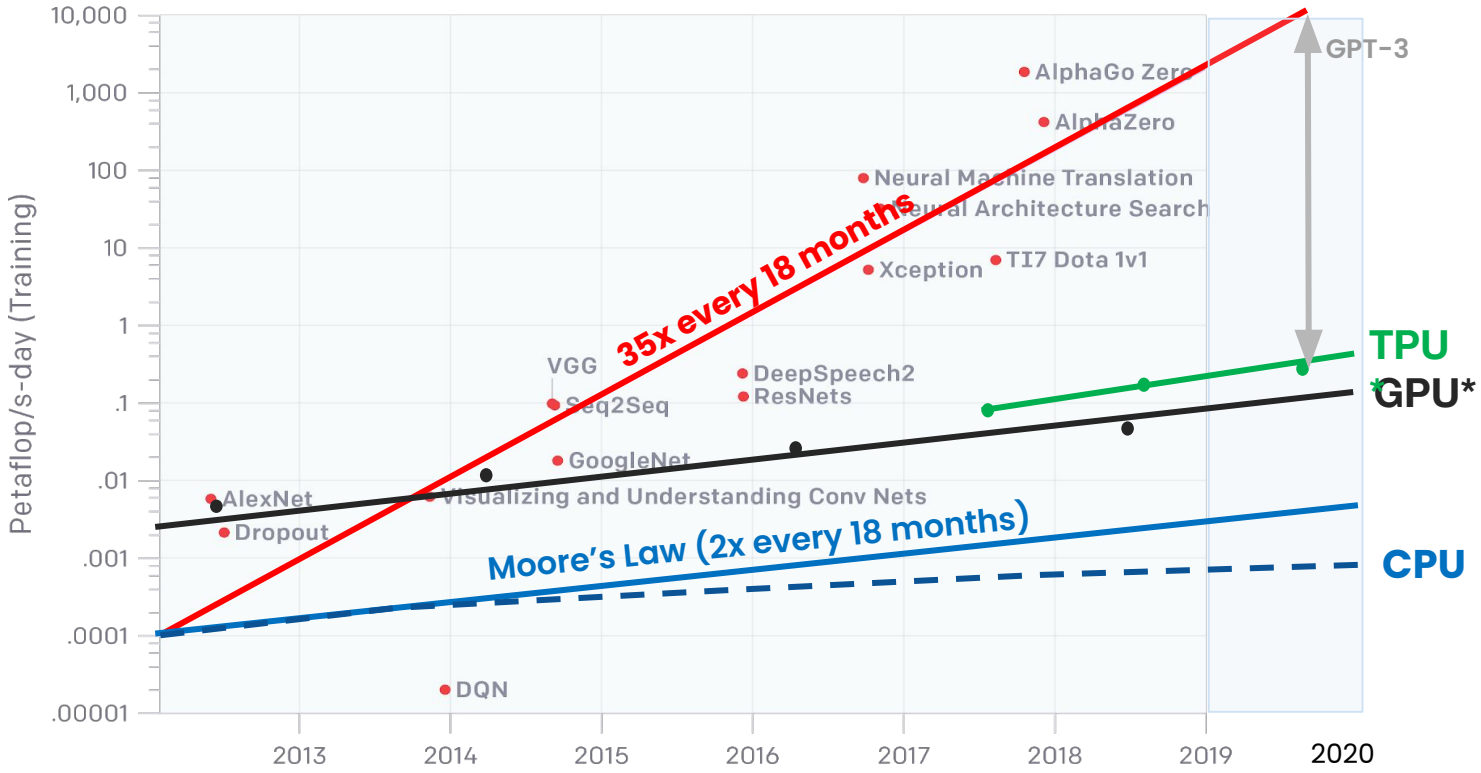
- Over the past 3 months we've been working with the Koch team on a couple of different use cases.
- Now we see how Anyscale ML workflows are incorporated into the Enterprise

Anyscale's goal: Koch focuses on building ML applications, not managing infrastructure

What is Ray?

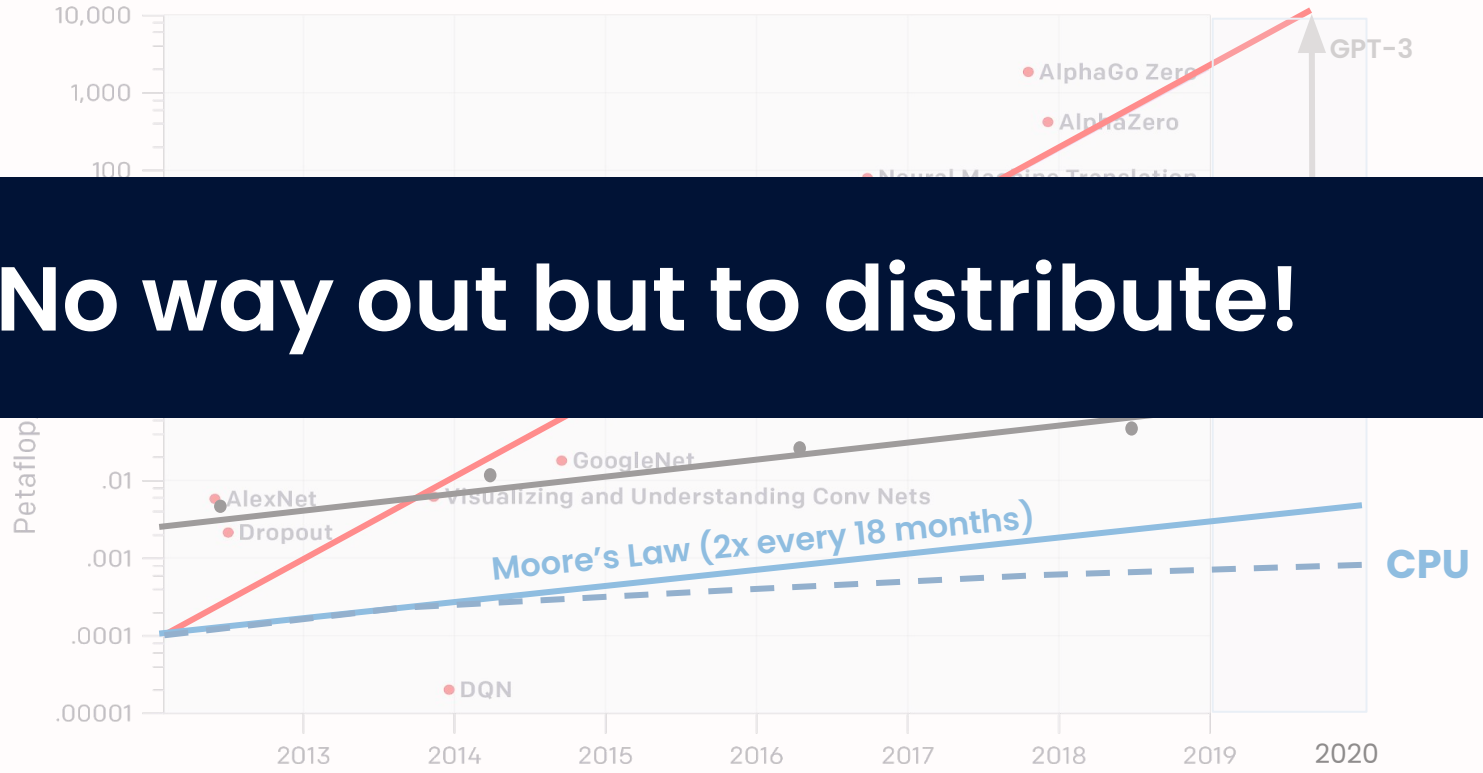
RAY Ecosystem

Specialized hardware is also not enough



https://openai.com/blog/ai-and-compute/

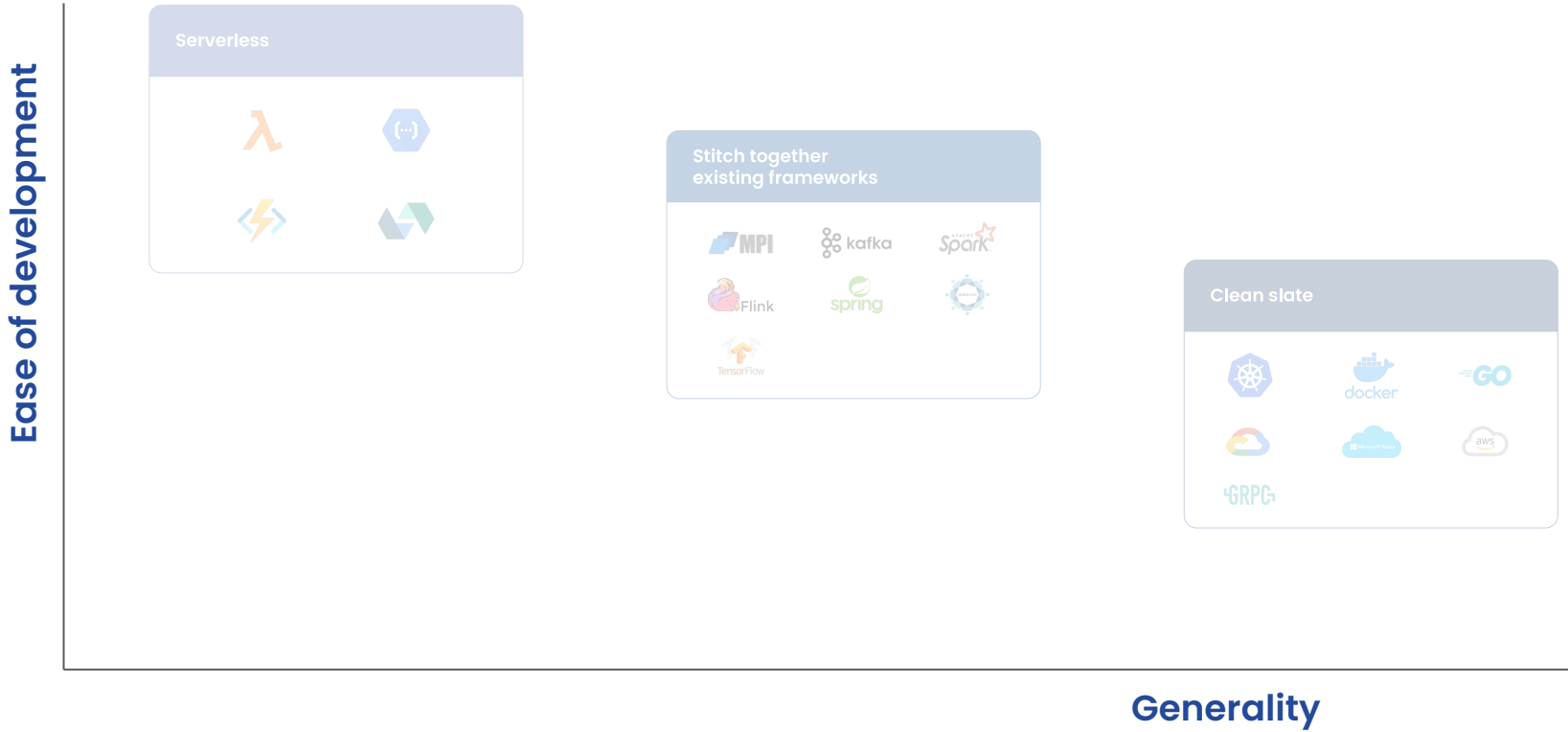
Specialized hardware is also not enough



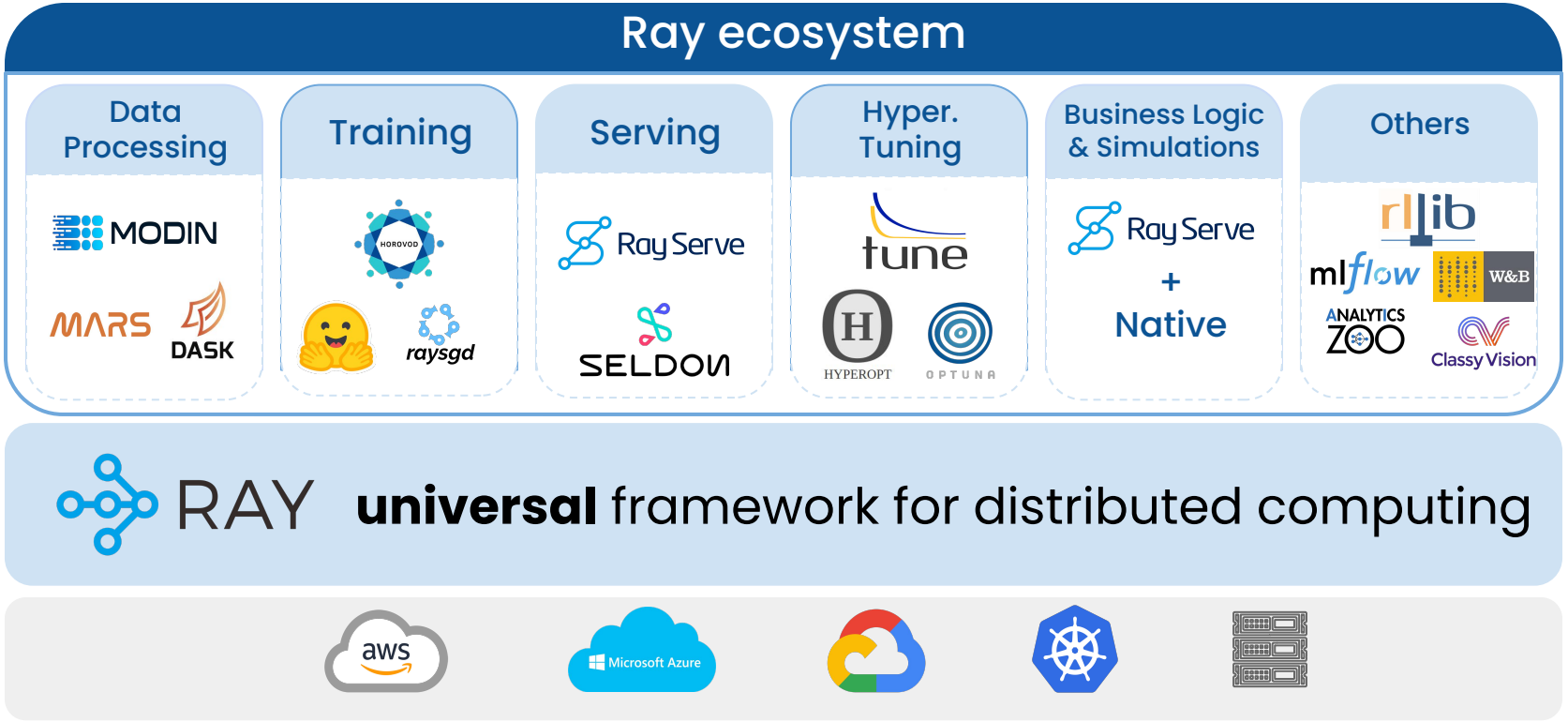
No way out but to distribute!

https://openai.com/blog/ai-and-compute/

Stitching together different frameworks to go end-to-end?



Ray's Ecosystem of ML and Data Libraries



Anyscale's Mission

01 Enable users to focus on apps instead of infrastructure.

02 Provide rapid iteration + reduce time to market for new products

03 Support variable infrastructure and clouds
- make compute magical

What is Ray - 3 Key Ideas

Execute functions remotely as tasks, and instantiate classes remotely as actors

- **Support both stateful and stateless computations**

Asynchronous execution using futures

- **Enable parallelism**

Distributed (immutable) object store

- **Efficient communication** (send arguments by reference)

What is Ray - API

```
def read_array(file):
    # read array a from file
    return a


def add(a, b):
    return np.add(a, b)

a = read_array(file1)
b = read_array(file2)
sum = add(a, b)
```

```
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value

c = Counter()
c.inc()
c.inc()
```

What is Ray - API

```
@ray.remote
def read_array(file):
    # read array a from file
    return a

@ray.remote
def add(a, b):
    return np.add(a, b)

a = read_array(file1)
b = read_array(file2)
sum = add(a, b)
```

```
@ray.remote
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value

c = Counter()
c.inc()
c.inc()
```

What is Ray - API

```
@ray.remote
def read_array(file):
    # read array a from file
    return a

@ray.remote
def add(a, b):
    return np.add(a, b)

ref1 = read_array.remote(file1)
ref2 = read_array.remote(file2)
ref = add.remote(ref1, ref2)
sum = ray.get(ref)
```

```
@ray.remote(num_gpus=1)
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value

c = Counter.remote()
ref4 = c.inc.remote()
ref5 = c.inc.remote()
```

What is Ray - API (Actor Handles)

Invoke actor methods from other tasks/actors/applications.

```python
@ray.remote(num_gpus=1)
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value

c = Counter.remote()
id = c.inc.remote()
```

```python
# Use the actor from a
# different task

@ray.remote
def use_actor(c):
    id = c.inc.remote()
    ray.get(id)

use_actor.remote(c)
```

Streamlined Workflows

```
ray.init()
```

```
import tensorflow
import pandas

@ray.remote
def f():
    # app logic

if __name__ == '__main__':
    main()
```

Ray cluster

Streamlined Workflows

Environments

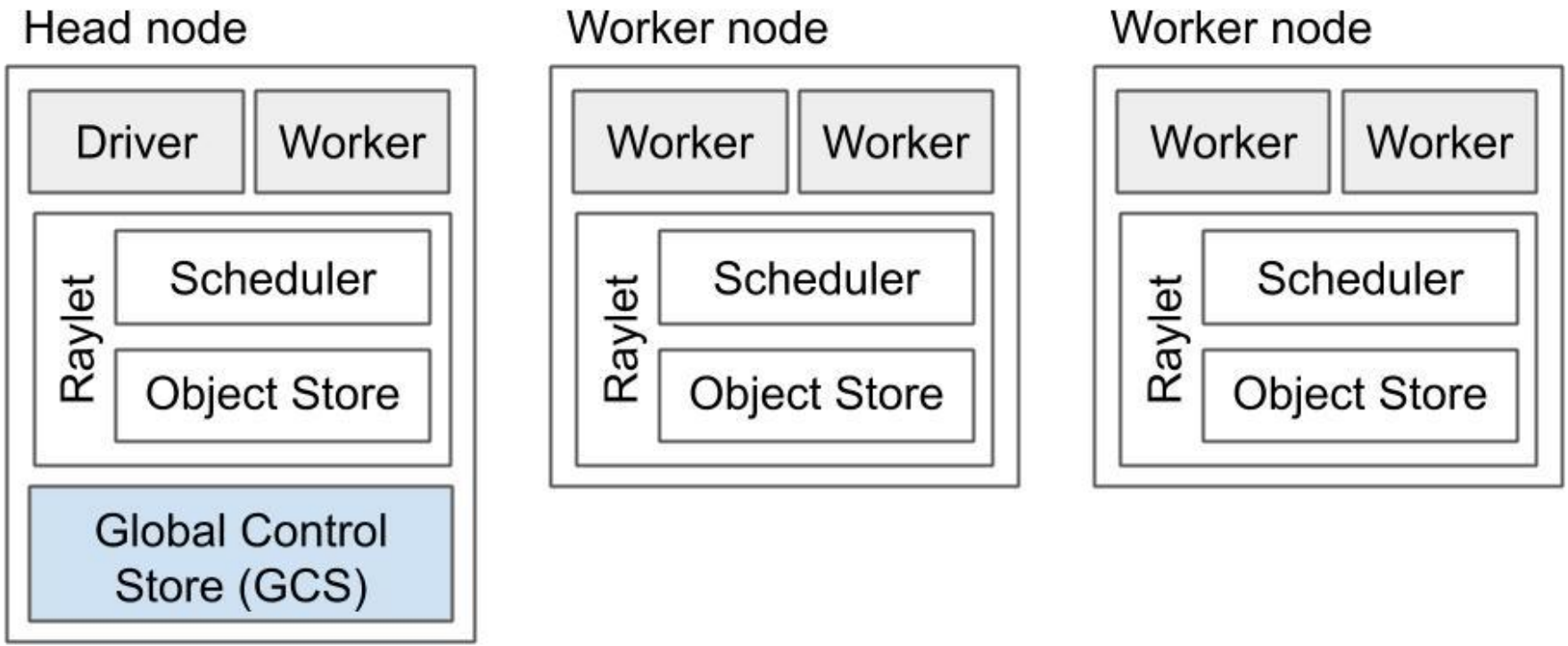
```
ray.init(runtime_env=
    {
        'pip': ['tensorflow', 'pandas'],
    }
)
```

Streamlined Workflows

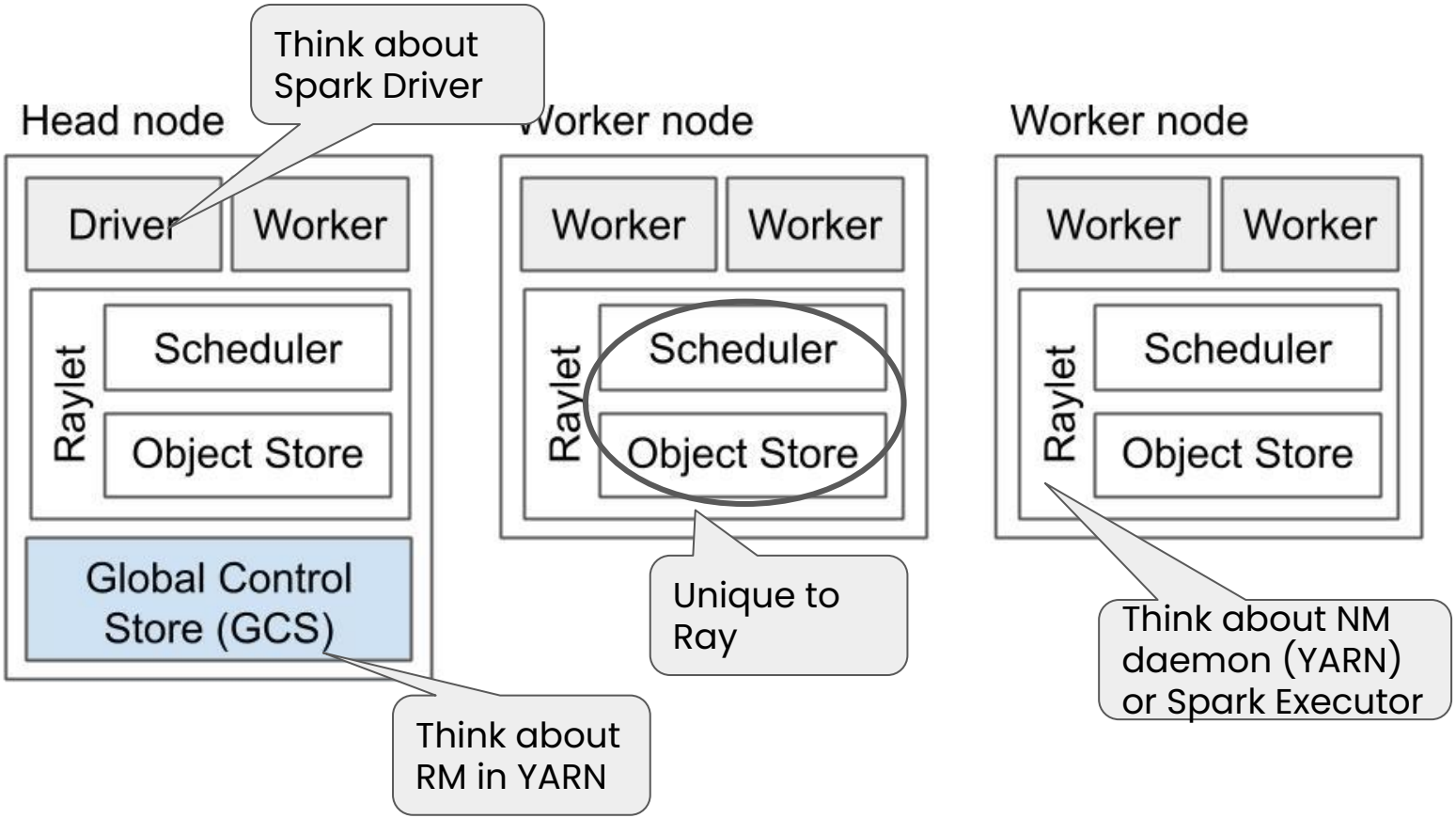
Environments

```
ray.init(runtime_env=
    {
        'pip': ['tensorflow', 'pandas'],
        'working_dir': '/home/my-project'
    }
)
```

Ray Architecture



What is Ray? A Cluster Looks Like…

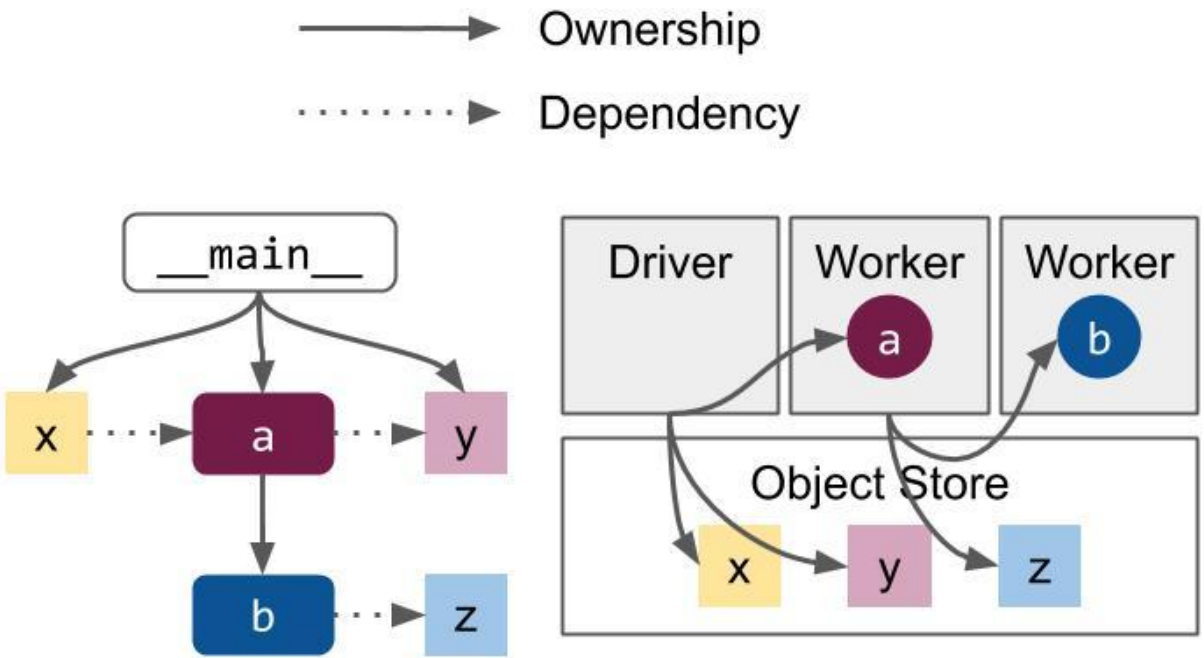


What is Ray? API -> Cluster

```
@ray.remote
def b():
  return

@ray.remote
def a(dep):
  z = b.remote()

x = ray.put(...)
y = a.remote(x)
```



Program **Task graph** **Physical execution**

Ray Application Lifecycle

- Experiment/Design on a Single Node
  - Jupyter Notebook on Anyscale
  - Laptop and local ray
- Optimize and Scale
- **Automate and Deploy**

```
github: anyscale/training-one
```

CLI and Anyscale Resources

- `pip install anyscale`
- `anyscale init`
- The project - an ambiguous term
  - A Project in Anyscale
  - Working Directory
- Python environment

Anyscale CLI - running commands

`anyscale run`

- Still experimental, let's return to this later.

Anyscale SDK

- Building Cluster Environments
- Launching Clusters
- Parallelizing SDK tasks (with ray!)

Application Lifecycle

- Project Structure
- Entry Points
  - Unit testing
  - Integration Testing
  - Application Interface

Ray Serve

- FastAPI
- Replication
- Rate Testing with Locust

Additional Resources

Documentation (docs.ray.io)

Quick start example, reference guides, etc

Forums (discuss.ray.io)

Learn / share with broader Ray community, including core team

Ray Slack

Connect with the Ray team and community