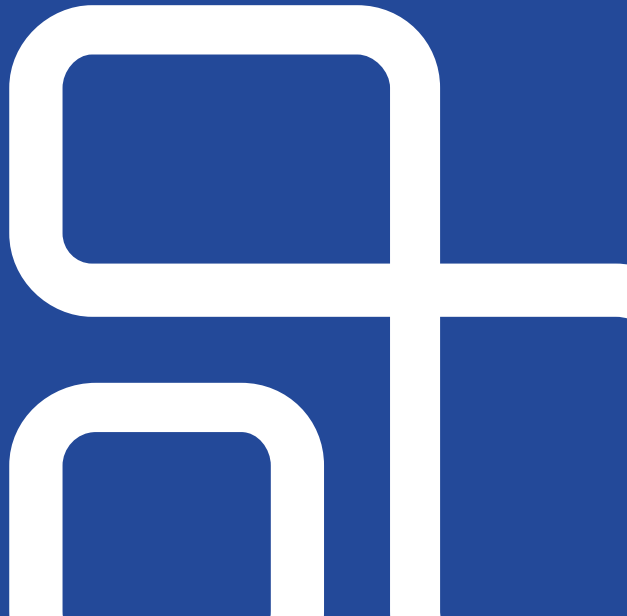
Introduction to Ray Core & Ray Ecosystem

Jules S. Damji, @2twitme

Lead Developer Advocate, Anyscale

TBD

Software Engineer Ray Core , Anyscale



anyscale

Overview

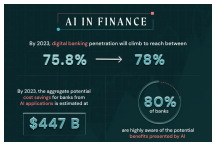
- What, Why Ray & Ray Ecosystem
- Ray Architecture & Components
- Ray Core APIs
- Demo
- Q & A

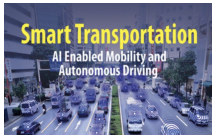
Why Ray?

- **Machine learning is pervasive in every domain**
- Distributed machine learning is becoming a necessity
- Distributed systems is notoriously hard

Apps increasingly incorporate AI/ML

AI/ML apps





Proliferated & Incorporated across Industries

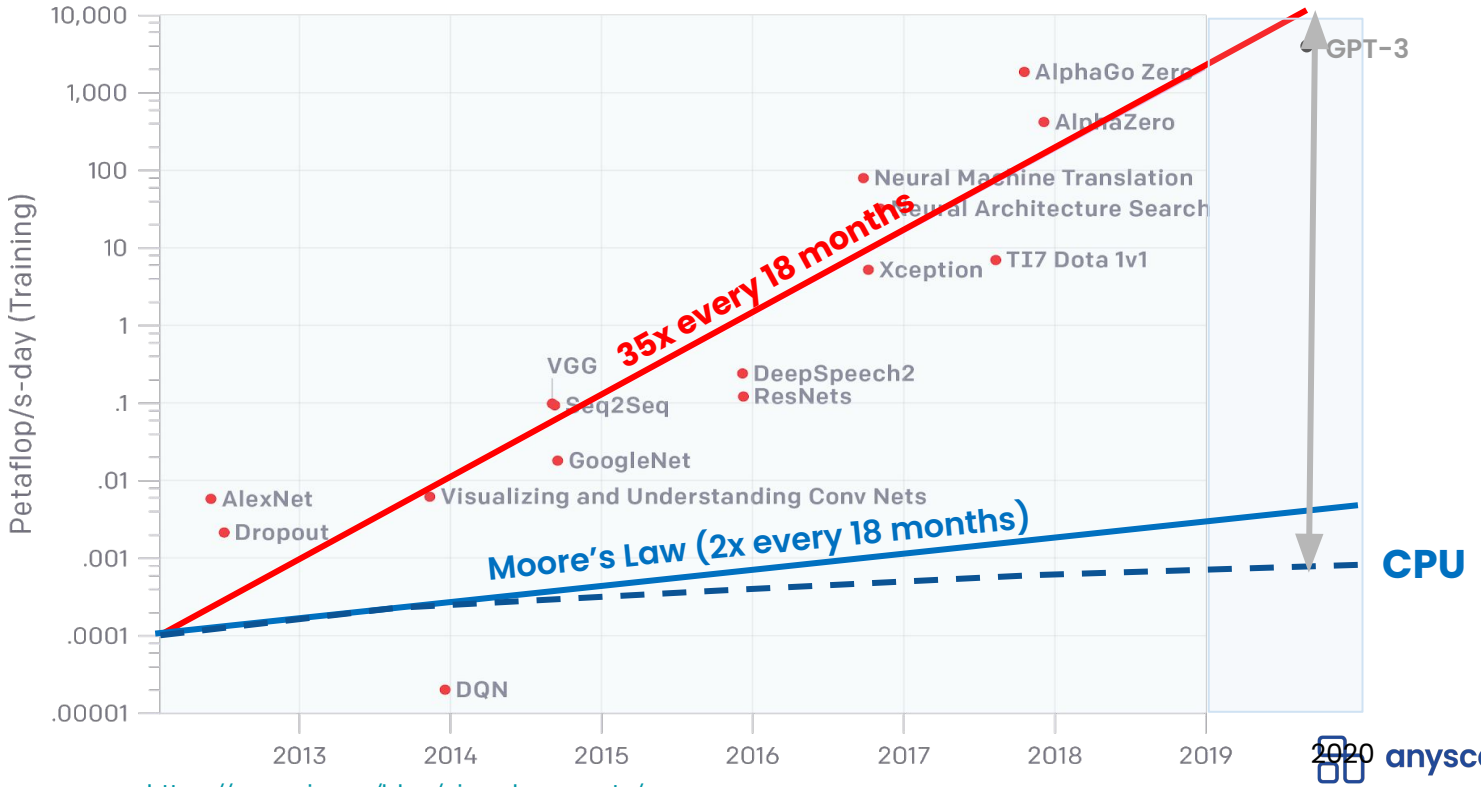
RAY

anyscale

Why Ray?

- Machine learning is pervasive in every domain
- **Distributed machine learning is becoming a necessity**
- Distributed systems is notoriously hard

Compute demand growing faster than supply

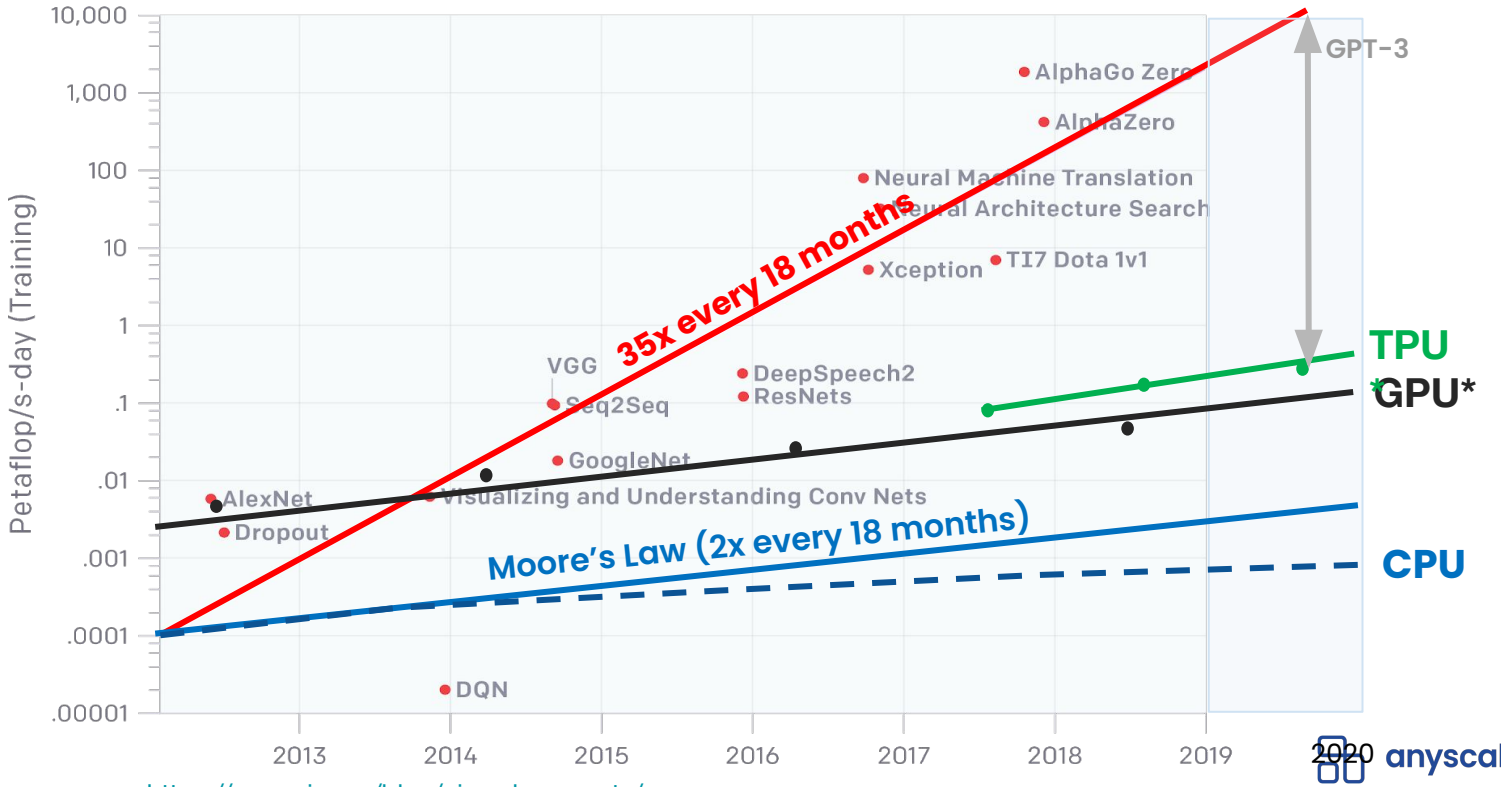


https://openai.com/blog/ai-and-compute/

RAY

anyscale

Specialized hardware is also not enough

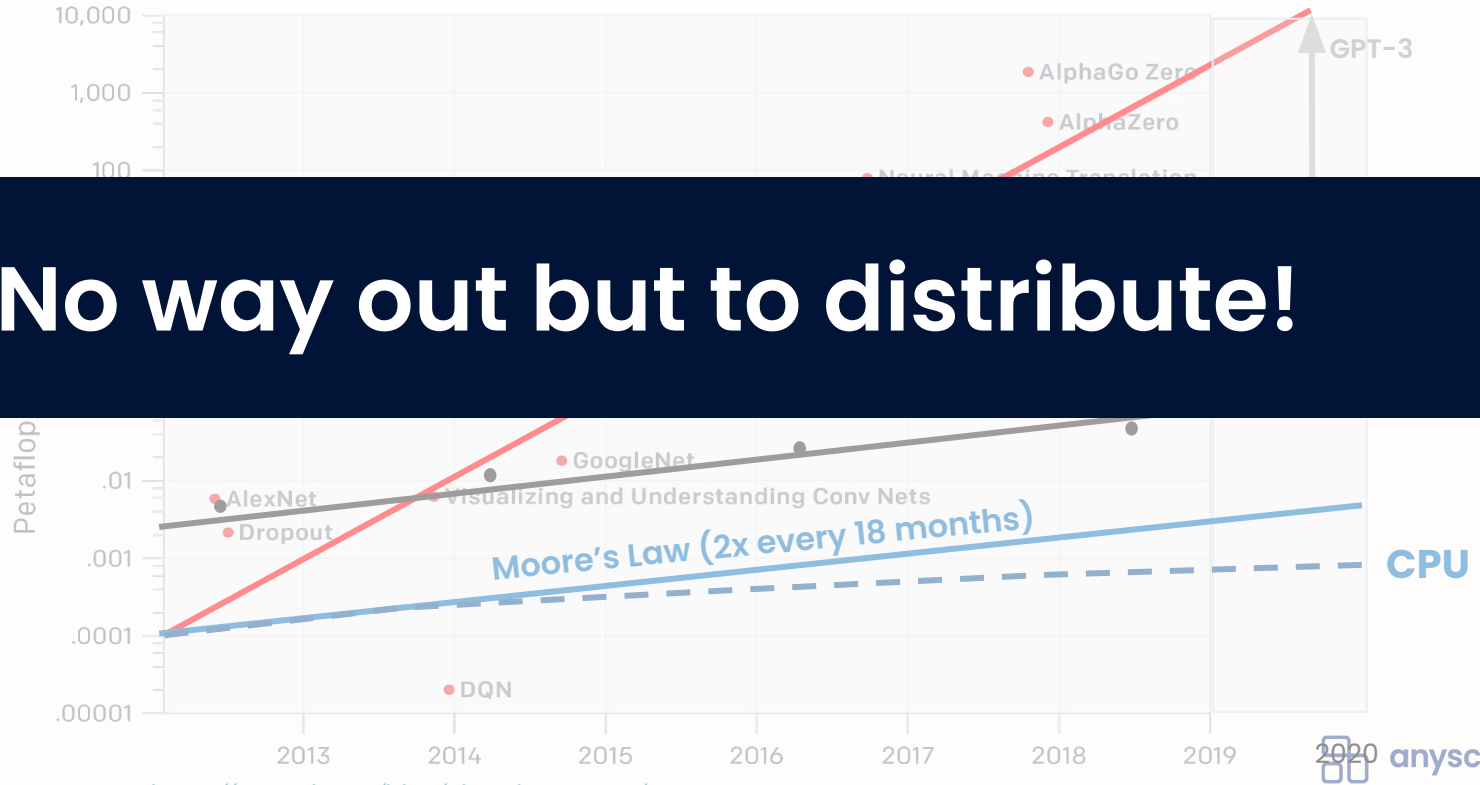


https://openai.com/blog/ai-and-compute/

RAY

anyscale

Specialized hardware is also not enough



No way out but to distribute!

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

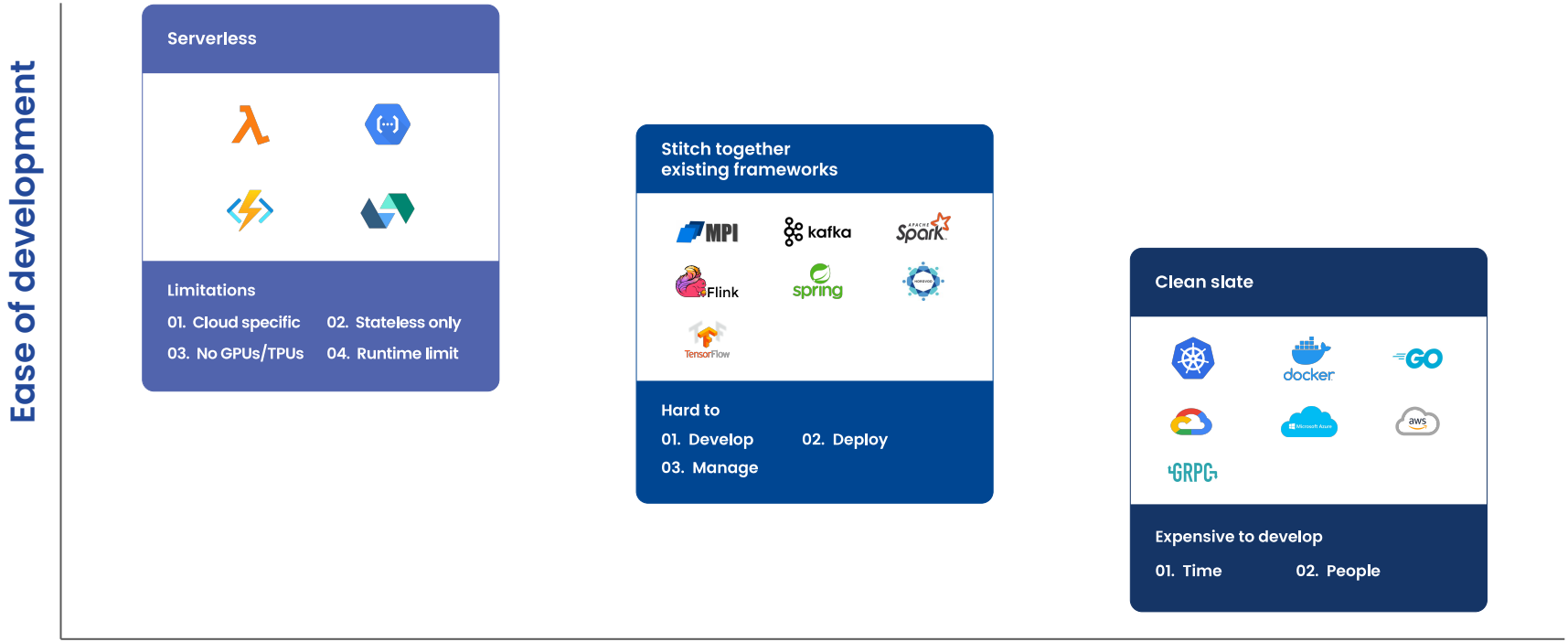
RAY

anyscale

Why Ray?

- Machine learning is pervasive in every domain
- Distributed machine learning is becoming a necessity
- **Distributed systems are notoriously hard**

Existing solutions have may tradeoffs



RAY

anyscale

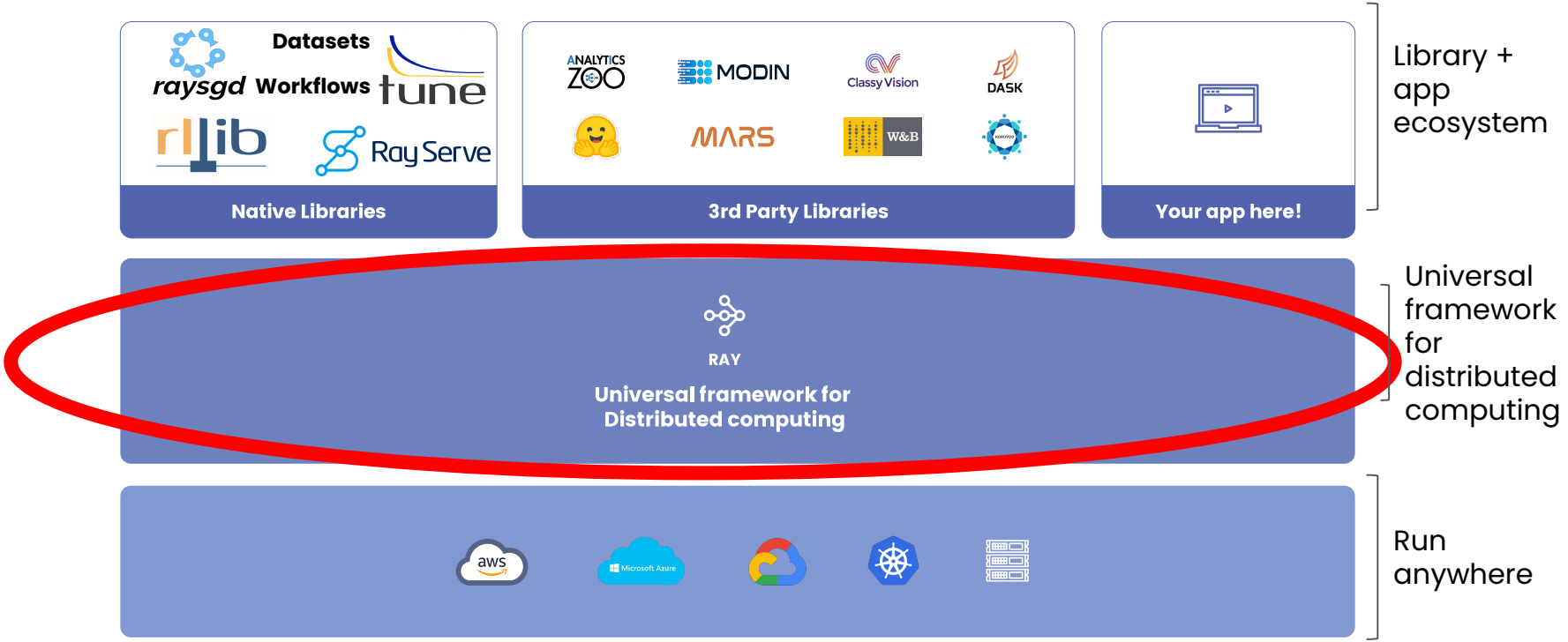
Why Ray?

- Machine learning is pervasive in every domain
- Distributed machine learning is becoming a necessity
- Distributed systems are notoriously hard

Ray's vision:

Make distributed computing accessible to every developer

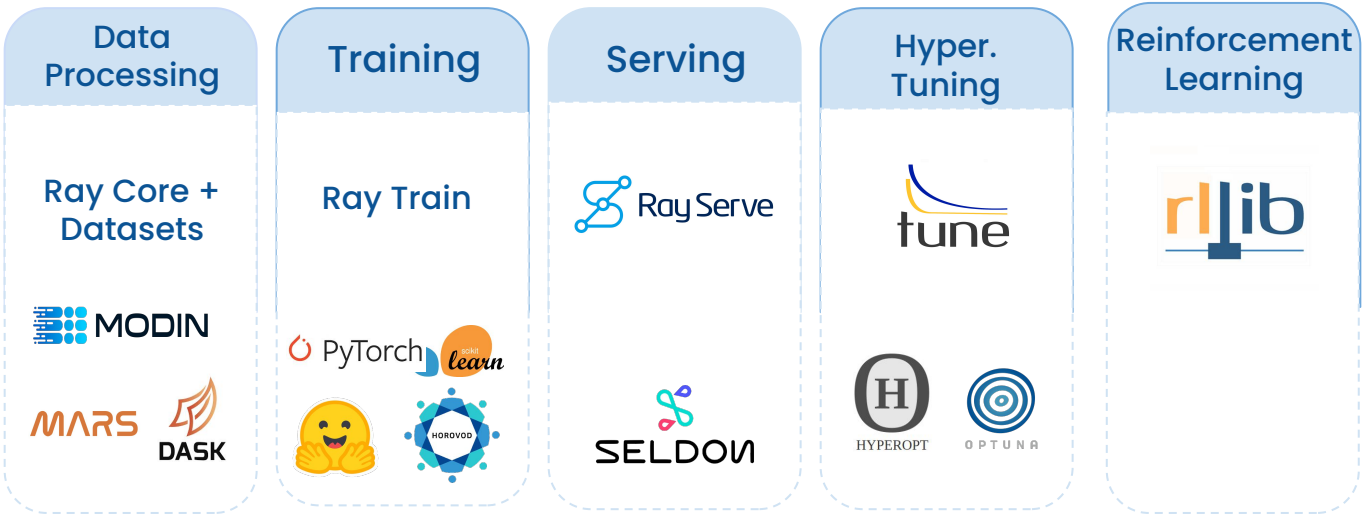
The Ray Layered Cake and Ecosystem



RAY

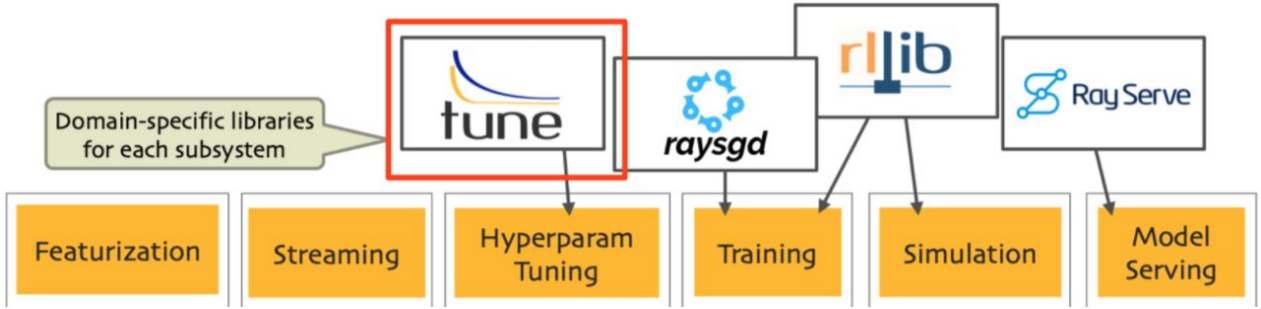
anyscale

Rich ecosystem for scaling ML workloads



Integrate Ray only based on your needs!

Capitalize on Ray Ecosystem



Library + app ecosystem

RAY

Universal framework for distributed computing

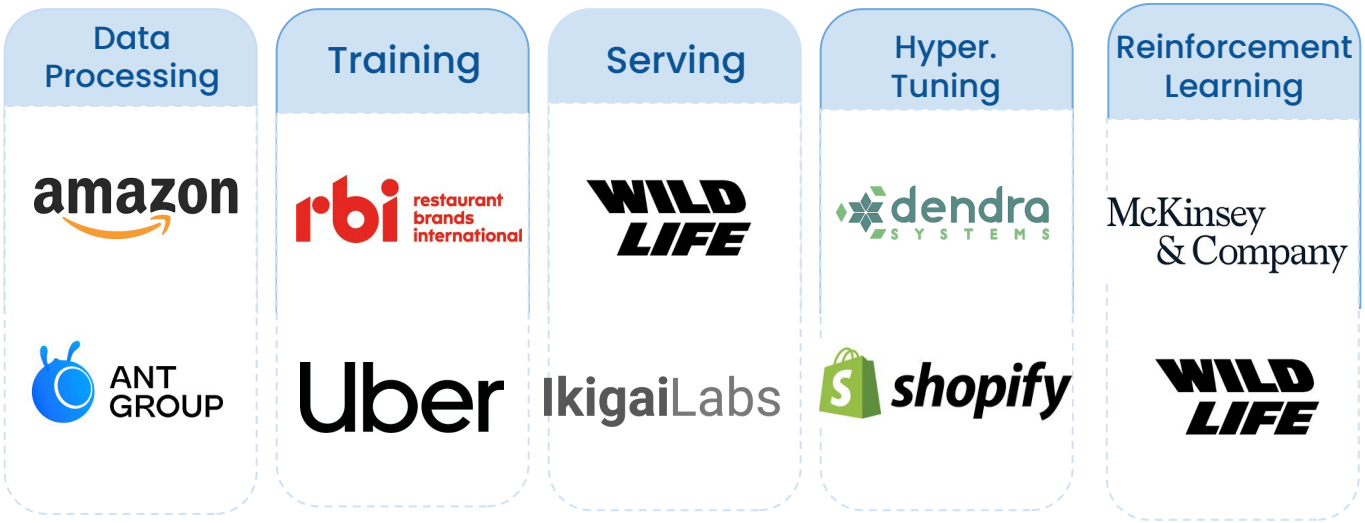
aws
Microsoft Azure

Run anywhere

anyscale

Companies scaling ML with Ray

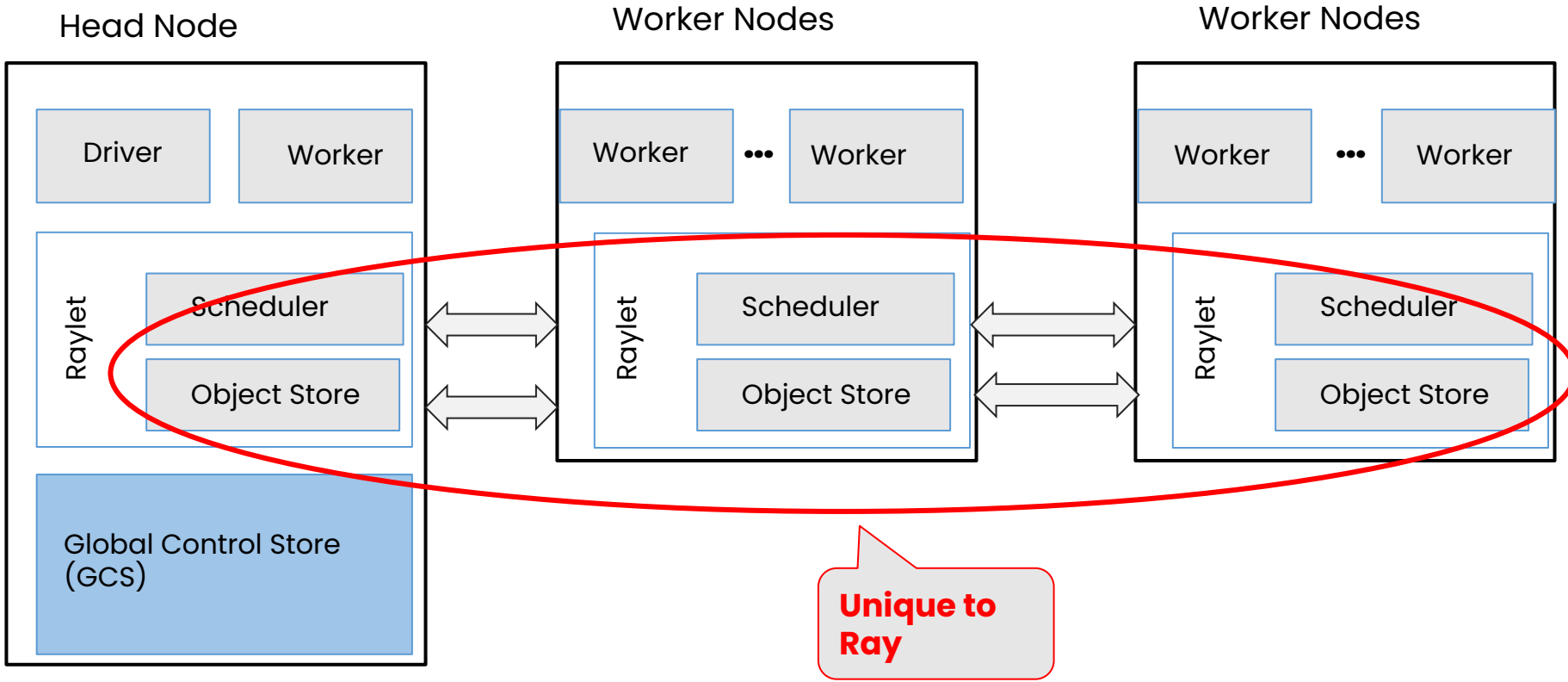
Companies scaling ML with Ray

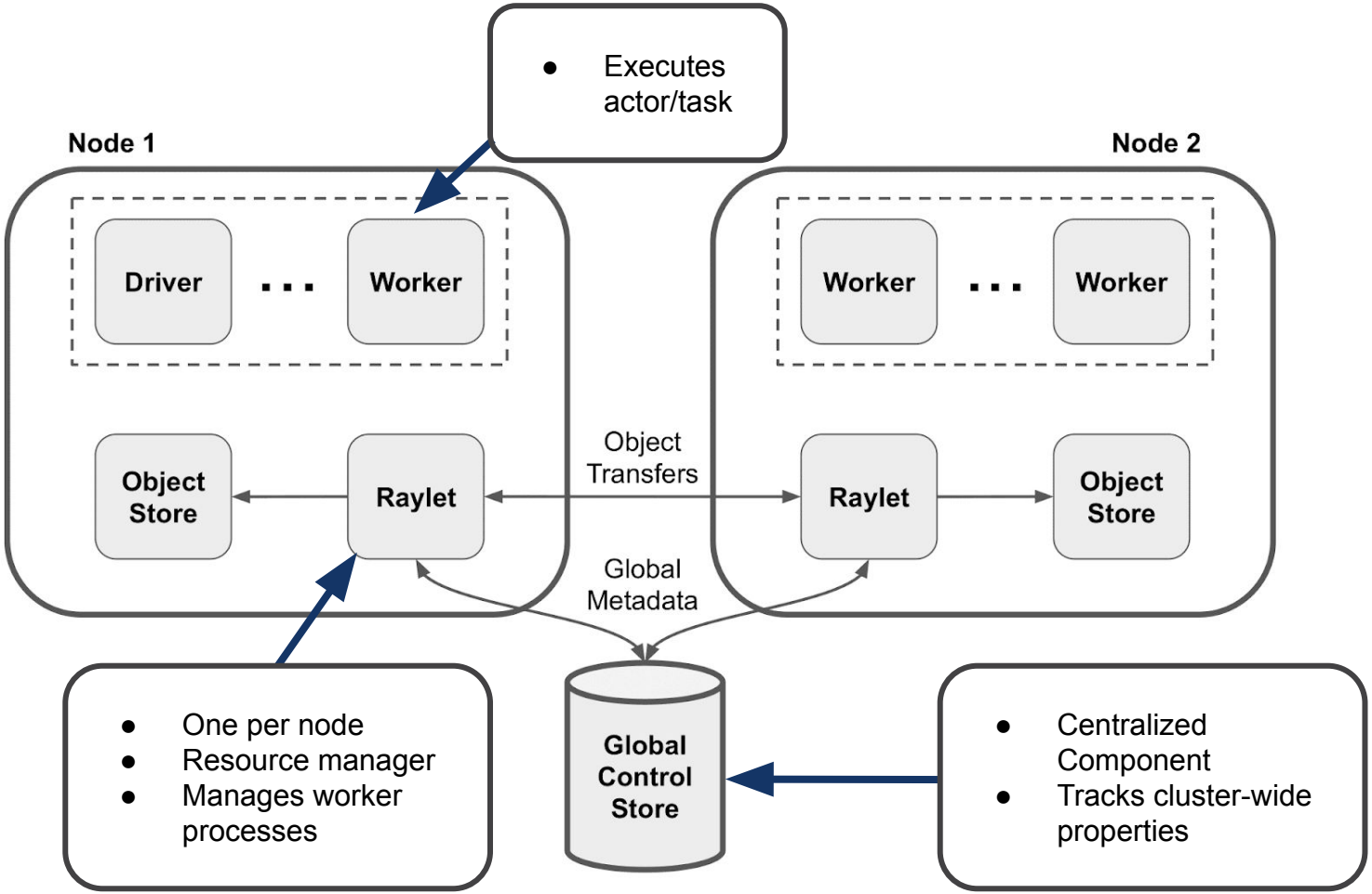


- https://eng.uber.com/horovod-ray/
- https://www.anyscale.com/blog/wildlife-studios-serves-in-game-offers-3x-faster-at-1-10th-the-cost-with-ray
- https://www.ikigailabs.com/blog/how-ikigai-labs-serves-interactive-ai-workflows-at-scale-using-ray-serve

Ray Architecture & Components

What does Ray Cluster Looks Like …

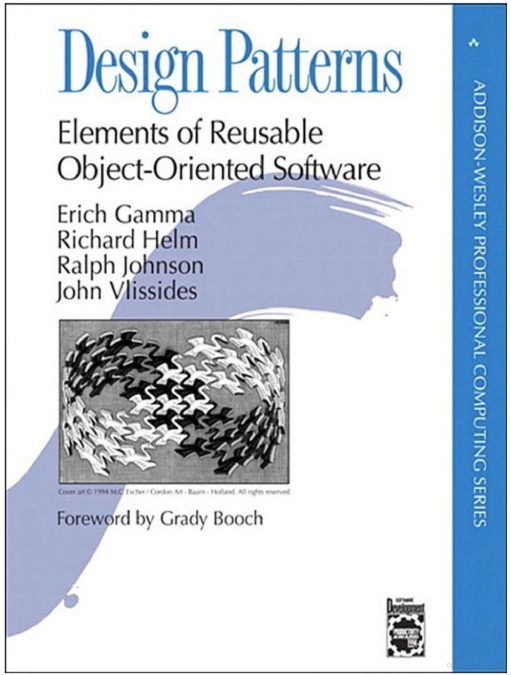



Executes actor/task
Node 1
Node 2
Driver
Worker
Worker
Worker
Object Store
Raylet
Object Transfers
Raylet
Object Store
Global Metadata
One per node
Resource manager
Manages worker processes
Global Control Store
Centralized Component
Tracks cluster-wide properties

Ray Distributed Design Patterns & APIs

Ray Basic Design Patterns

- Ray Parallel Tasks
  - Functions as stateless units of execution
  - Functions distributed across a clusters as tasks
- Ray Objects or Futures
  - Distributed (immutable) Object stored in cluster
  - Retrievable when available
  - Enable asynchronous execution of
- Ray Actors
  - Stateful service on a cluster
  - Message passing and maintains state



1. Patterns for Parallel Programming
2. Ray Design Patterns

Python → Ray Basic Patterns

Function → Task → Node

Class → Actor → Node

Object → (Distributed immutable) Object → Node

RAY

anyscale

Function

```
def read_array(file):
    # read ndarray “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

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

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

anyscale

Function → Task

```
@ray.remote
def read_array(file):
    # read ndarray “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)
```

Class → Actor

```
@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()
```

anyscale

Function → Task

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

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

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

Class → Actor

```
@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()
id4 = c.inc.remote()
id5 = c.inc.remote()
```

anyscale

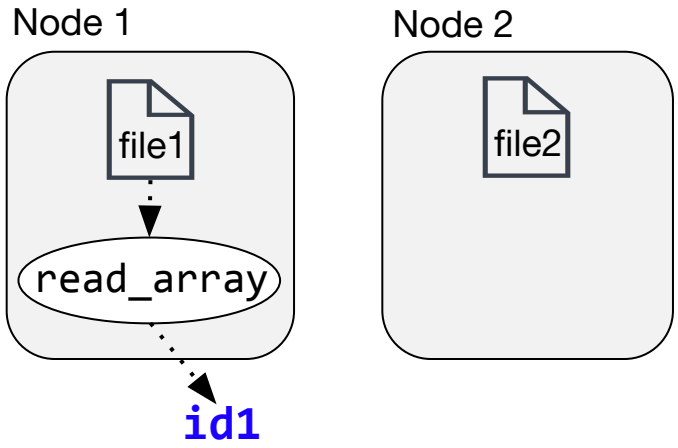
Task API

Blue variables are ObjectRef IDs (similar to futures)

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

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

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```



Return id1 (future) immediately, before **read_array()** finishes

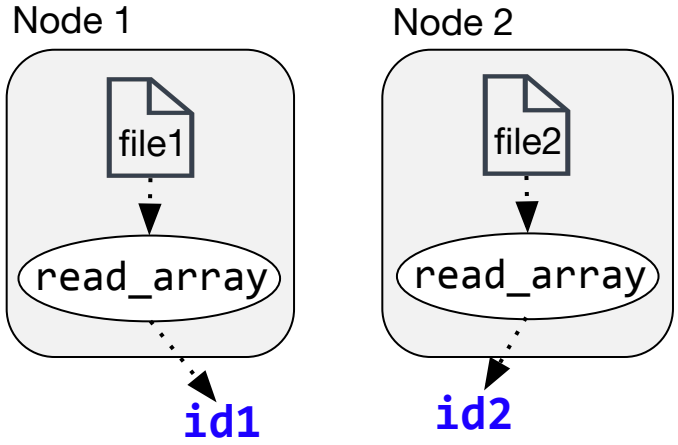
Task API

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

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

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

Blue variables are Object IDs (similar to futures)



Dynamic task graph: build at runtime

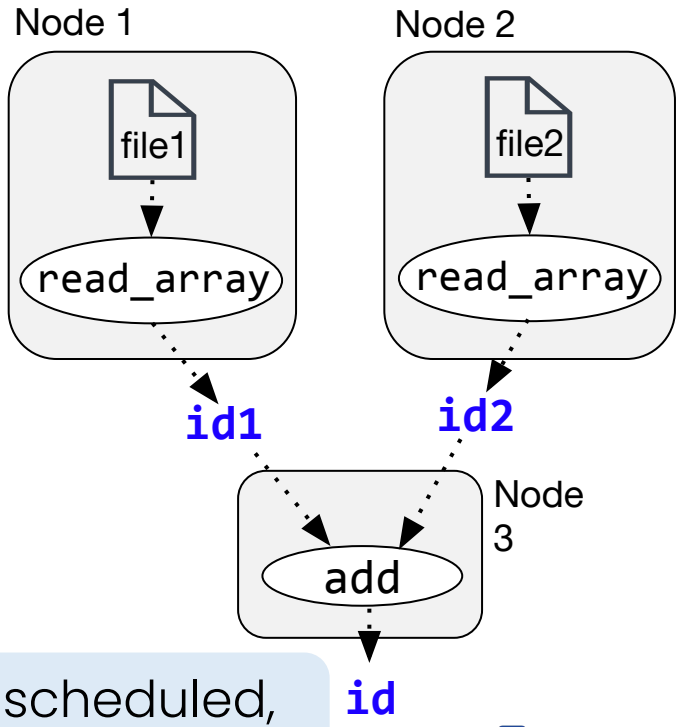
Task API

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

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

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

Blue variables are Object IDs (similar to futures)



Every task scheduled, but not finished yet

anyscale

Task API

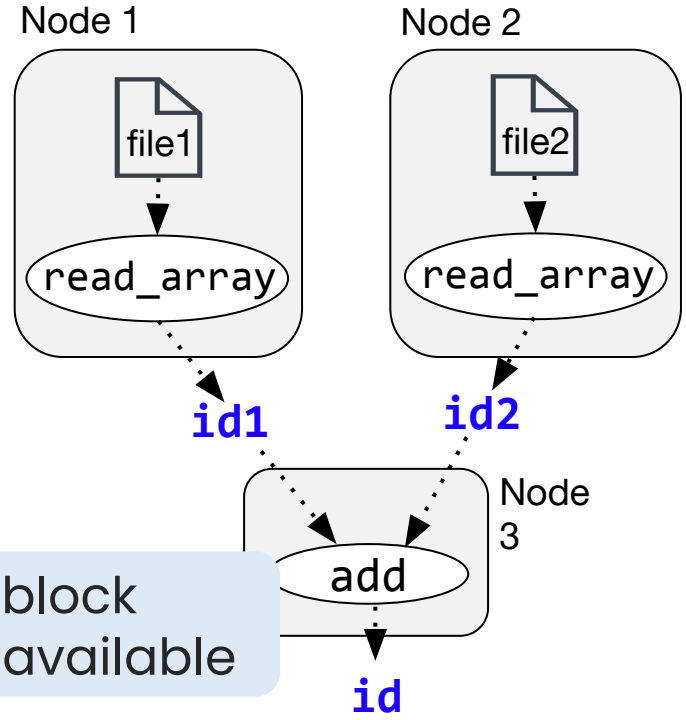
Blue variables are Object IDs (similar to futures)

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

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

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

ray.get() block until result available



anyscale

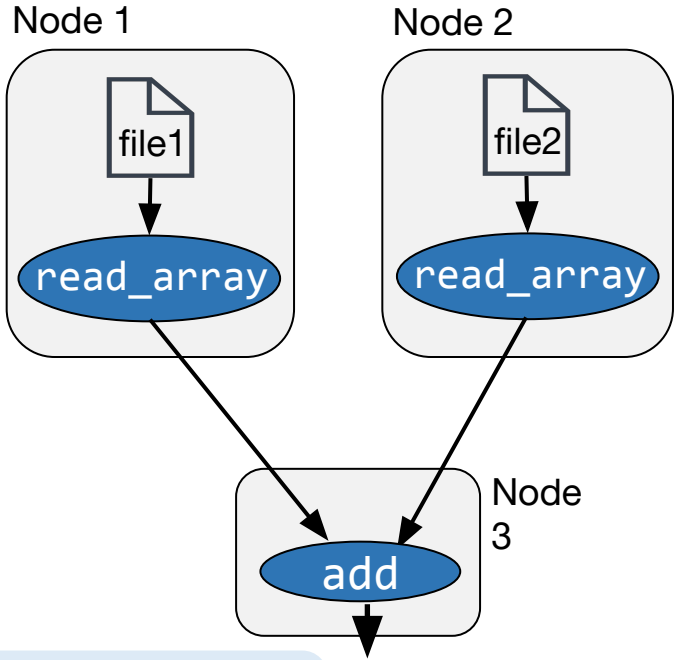
Task API

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

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

id1 = read_array.remote(file1)
id2 = read_array.remote(file2)
id = add.remote(id1, id2)
sum = ray.get(id)
```

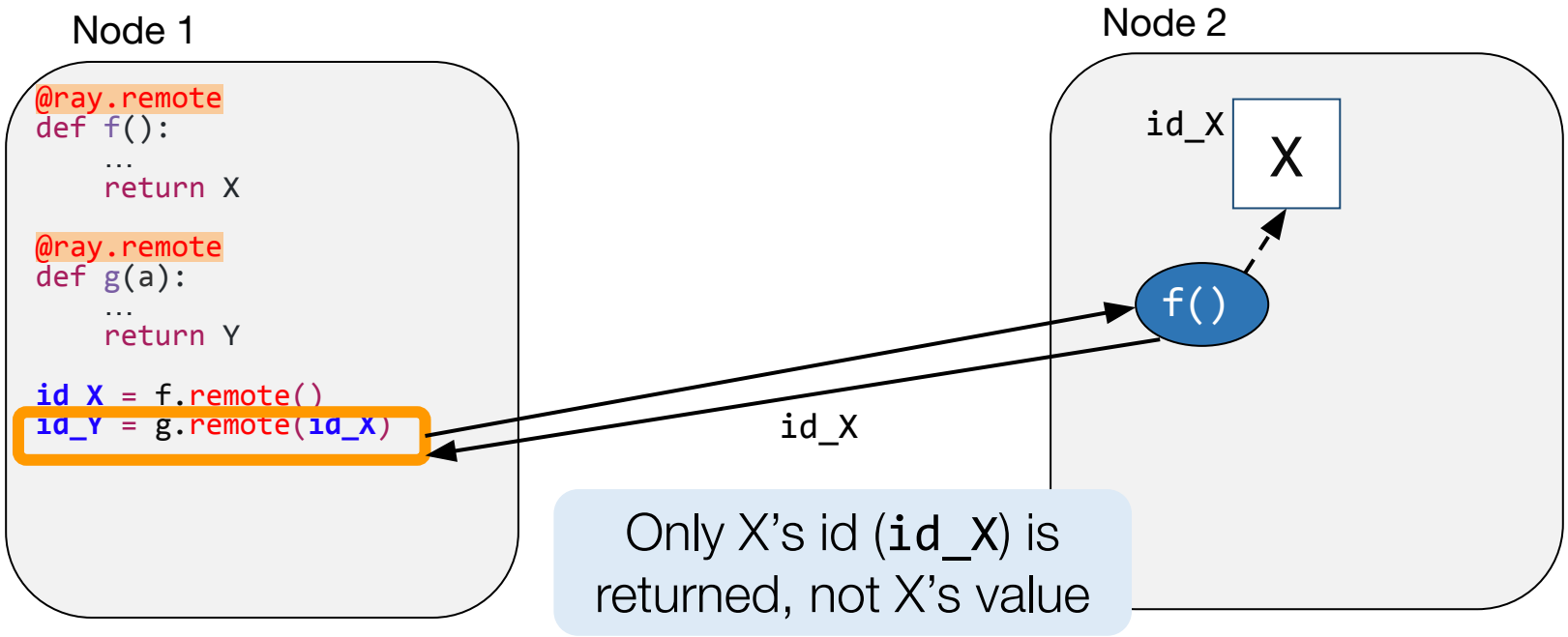
Blue variables are Object IDs (similar to futures)



Task graph executed to compute `sum`

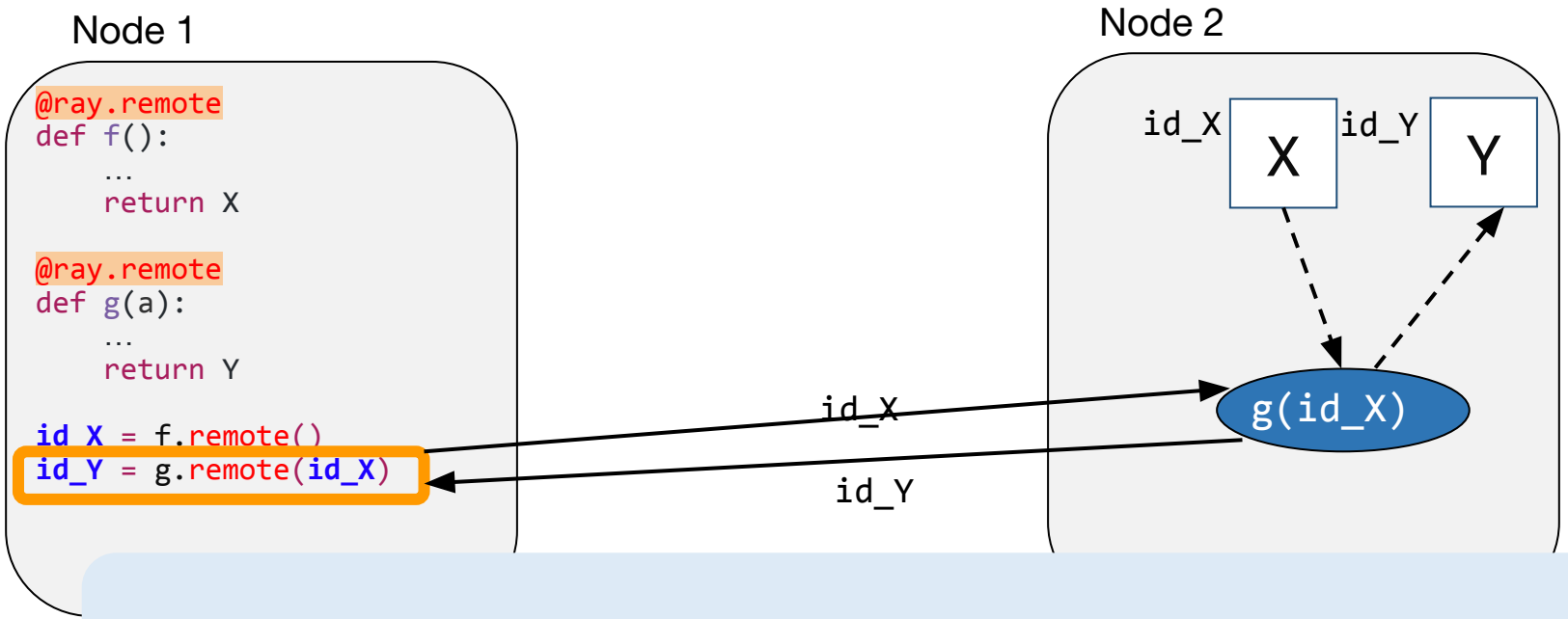
anyscale

Distributed Immutable object store



anyscale

Distributed object store



`g(id_X)` is scheduled on same node, so `X` is never transferred

Takeaways

- Distributed computing is a necessity & norm
- Scale your ML workloads with Ray Libraries
- Ray's vision: make distributed programming simple
- Don't have to be distributed systems expert. Just use @ray.remote :)

RAY

Meetup In 2022

https://tinyurl.com/ray-meetup-jan2022

Jan 19 th @ 6:00 PM PST

Demo

https://tinyurl.com/ray-core-demo

anyscale

You might be interested in …

- How to scale ML workloads with Ray
- How to Effectively tune HPO with Ray Tune
- How to Scale Model Serving with Ray Serve
- Scalable RL training & serving in the cloud with Ray
- Let us know what you are looking for ….

Start learning Ray and contributing …

Getting Started:

Documentation (docs.ray.io)
Quick start example, reference guides, etc

Join Ray Meetup
Revive in Jan 2022. Publish recording to the members
https://www.meetup.com/Bay-Area-Ray-Meetup/

Forums (discuss.ray.io)
Learn / share with broader Ray community, including core team

Ray Slack
Connect with the Ray team and community

Social Media (@raydistrtibuted, @anyscalecompute)
Follow us on Twitter and linkedIn

GitHub
*Check out sources, file an issue, become a contributor, give us a **Star** :)*
https://github.com/ray-project/ray

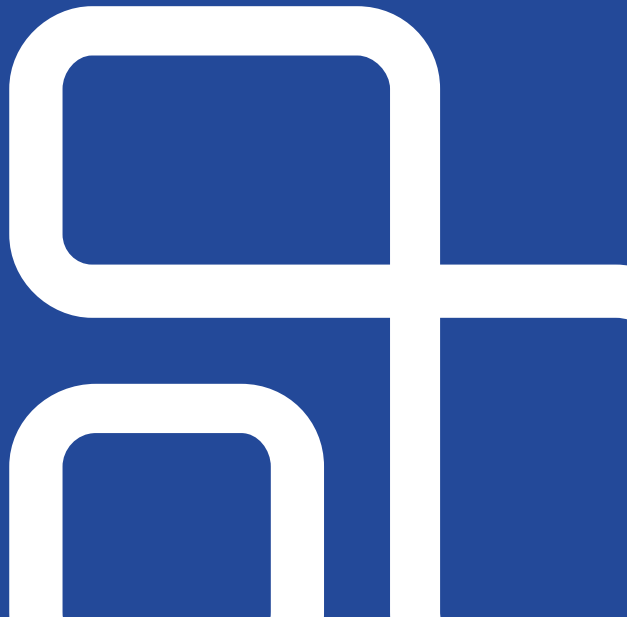
https://github.com/dmatrix/ray-core-tutorial

Thank you!

Let's stay in touch:

jules@anyscale.com
https://www.linkedin.com/in/dmatrix/

@2twitme



anyscale

Extras

ray.wait() non-blocking usage

```
c = Counter.remote()
increment_refs = [c.inc.remote() for _ in range(5)]
while len(increment_refs) > 0:
   return_n = 2 if len(increment_refs) > 1 else 1
   ready_refs, remaining_refs = ray.wait(increment_refs,
                                         num_returns=return_n,
                                            timeout=10.0)

   if len(ready_refs > 0:
      print(ray.get(ready_refs))
   # Update the remaining ones
   increment_refs = remaining_refs
```

```
[1, 2]
[3, 4]
[5]
```

Object Ownership