Distributed Machine Learning with Ray

Scaling AI Applications in Python with the Ray Framework

inquiries@jonathan.industries

Learning Outcomes

inquiries@jonathan.industries

Learning Outcomes

- Basics of distributed machine learning

inquiries@jonathan.industries

Learning Outcomes

- Basics of distributed machine learning
- Use cases that **Ray** is especially well-suited to solve

inquiries@jonathan.industries

Learning Outcomes

- Basics of distributed machine learning
- Use cases that **Ray** is especially well-suited to solve
- Differences between **Ray** vs. other frameworks

inquiries@jonathan.industries

Learning Outcomes

- Basics of distributed machine learning
- Use cases that **Ray** is especially well-suited to solve
- Differences between **Ray** vs. other frameworks
- Internals of **Ray** and its execution model

inquiries@jonathan.industries

Learning Outcomes

- Basics of distributed machine learning
- Use cases that **Ray** is especially well-suited to solve
- Differences between **Ray** vs. other frameworks
- Internals of **Ray** and its execution model
- Context of the larger **Ray** ecosystem.

inquiries@jonathan.industries

Getting the Materials

https://sr.ht/~hyphaebeast/ray-live-training/

inquiries@jonathan.industries

Do you have any experience with machine learning?

inquiries@jonathan.industries

Have you used a distributed computing framework?
(like Spark, Flink, Hadoop, etc.)

inquiries@jonathan.industries

Why Scale?

inquiries@jonathan.industries

A Brief History

In terms of size [of transistors] you can see that we're approaching the size of atoms which is a fundamental barrier, but it'll be two or three generations before we get that far—but that's as far out as we've ever been able to see.

— Gordon Moore (2006)

inquiries@jonathan.industries

A Brief History

We have another 10 to 20 years before we reach a fundamental limit. By then they'll be able to make bigger chips and have transistor budgets in the billions.

— Gordon Moore (2006)

inquiries@jonathan.industries

Big Data

inquiries@jonathan.industries

Big Data

- **Long Data**: Large # of instances/records/rows/etc.

inquiries@jonathan.industries

Big Data

- **Long Data**: Large # of instances/records/rows/etc.
- **Wide Data**: High dimensionality

inquiries@jonathan.industries

Big Models

inquiries@jonathan.industries

Big Models

- **Large Parameter Spaces**: Models with large number of parameters to learn

inquiries@jonathan.industries

Big Models

- **Large Parameter Spaces**: Models with large number of parameters to learn
- **Prediction Cascades**: Complex pipeline of multiple prediction tasks

inquiries@jonathan.industries

Small Latencies

inquiries@jonathan.industries

Small Latencies

- Real-time predictions (i.e. website ad serving)

inquiries@jonathan.industries

Small Latencies

- Real-time predictions (i.e. website ad serving)
- Prediction Cascades... again (i.e. self-driving cars)

inquiries@jonathan.industries

What Scale?

inquiries@jonathan.industries

What Scale?

- Memory

inquiries@jonathan.industries

What Scale?

- Memory
- Computation

inquiries@jonathan.industries

Introduction to distributed systems

Distributed programming is the art of solving the same problem that you can solve on a single computer using multiple computers.

— Mikito Takada[1]

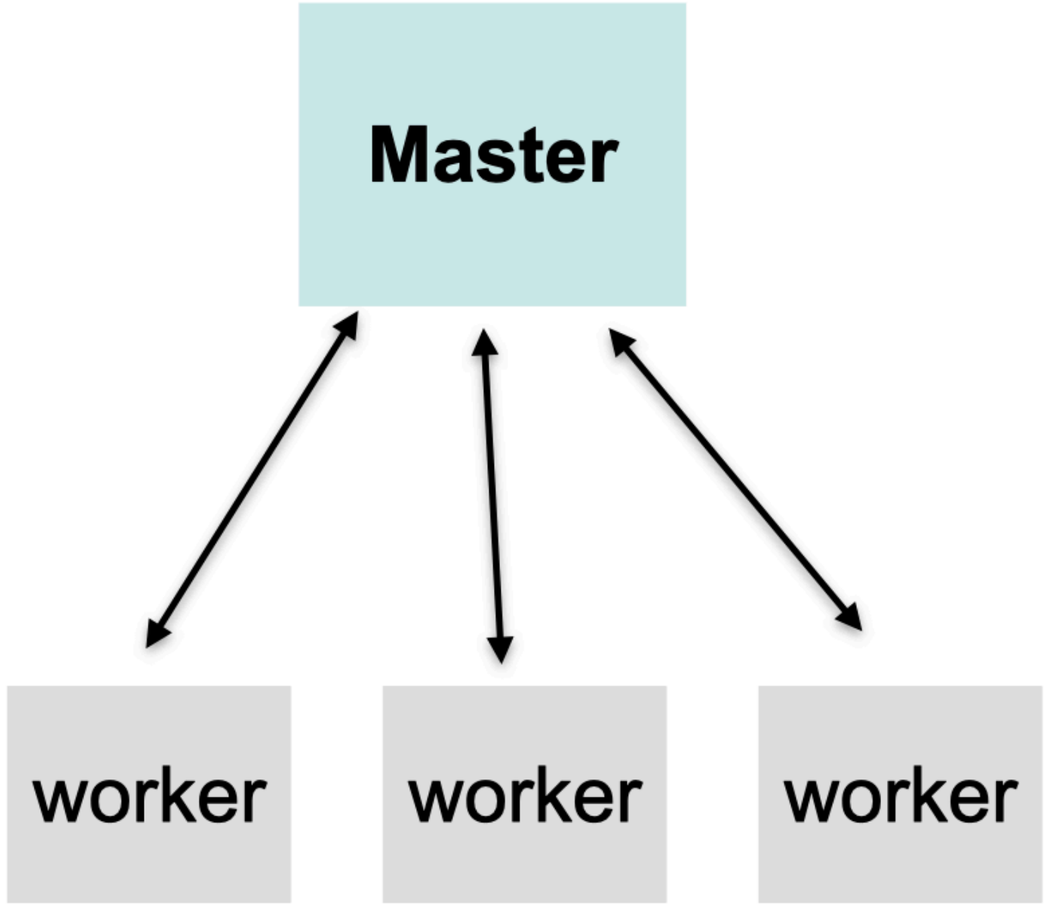
[1] Distributed systems: for fun and profit

inquiries@jonathan.industries

Local

Single Machine

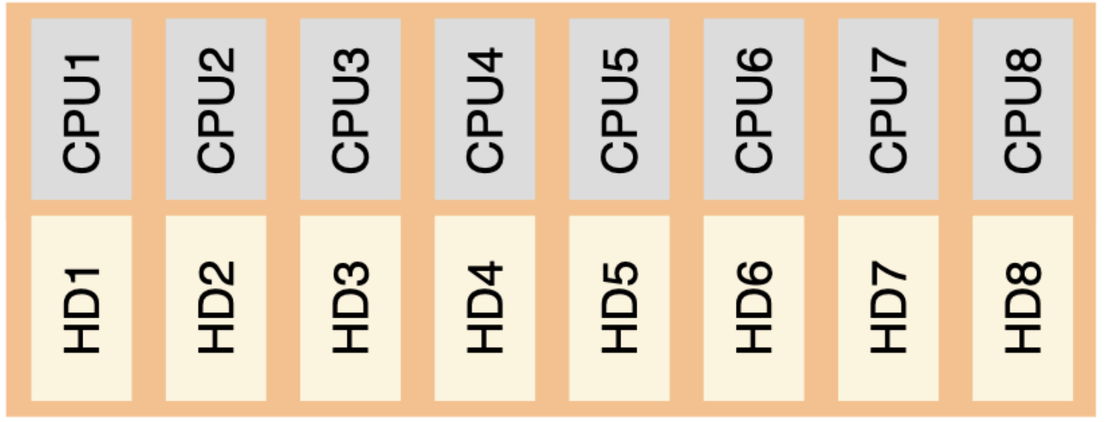
Distributed



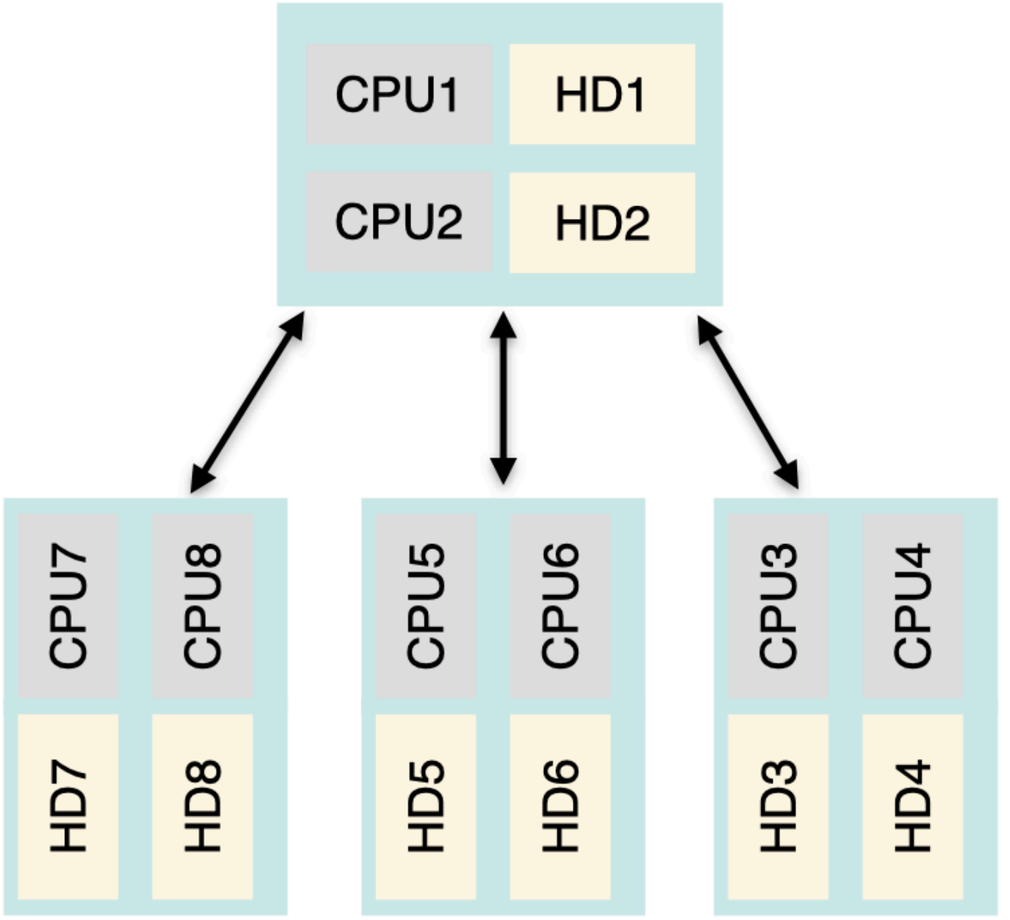
inquiries@jonathan.industries

How to Distribute

Local



Distributed

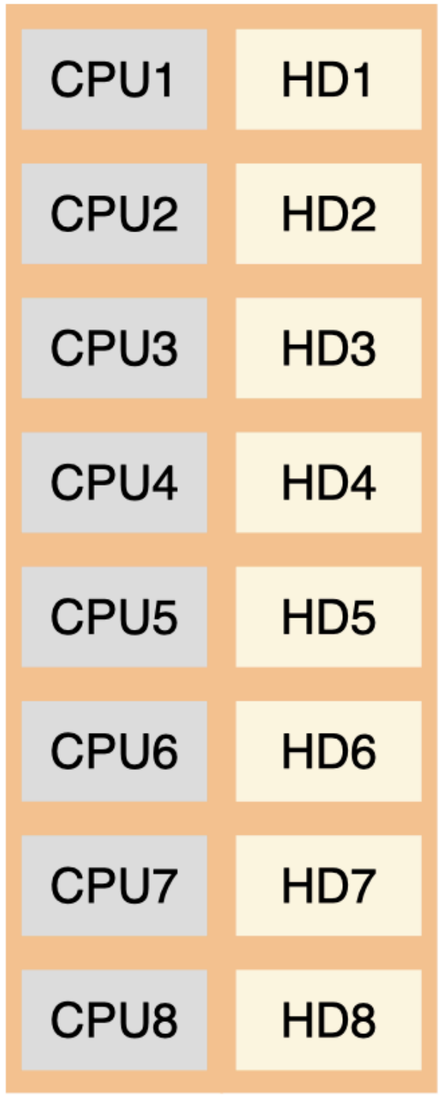


Storage: Hard drives

Computation: CPUs

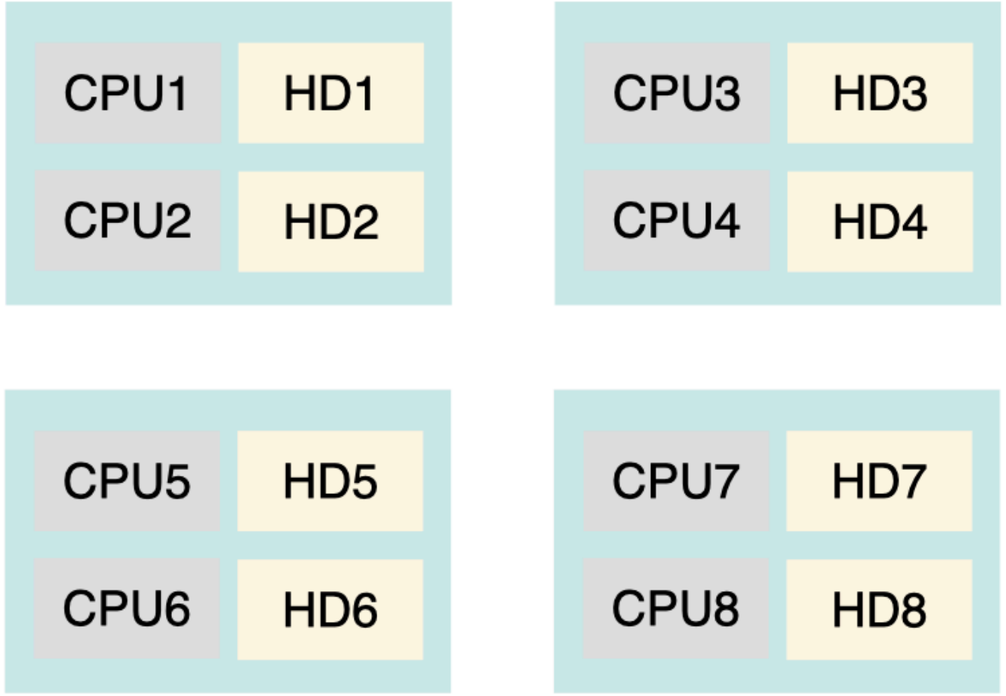
inquiries@jonathan.industries

Scale Up versus Out



Up

Out



inquiries@jonathan.industries

Streaming Algorithms

we won't talk about these now....

inquiries@jonathan.industries

Paralellism

inquiries@jonathan.industries

Paralellism

- **Data Parallelism**: Split instances between machines

inquiries@jonathan.industries

Paralellism

- **Data Parallelism**: Split instances between machines
- **Model Paralellism**: Split parameters between machines

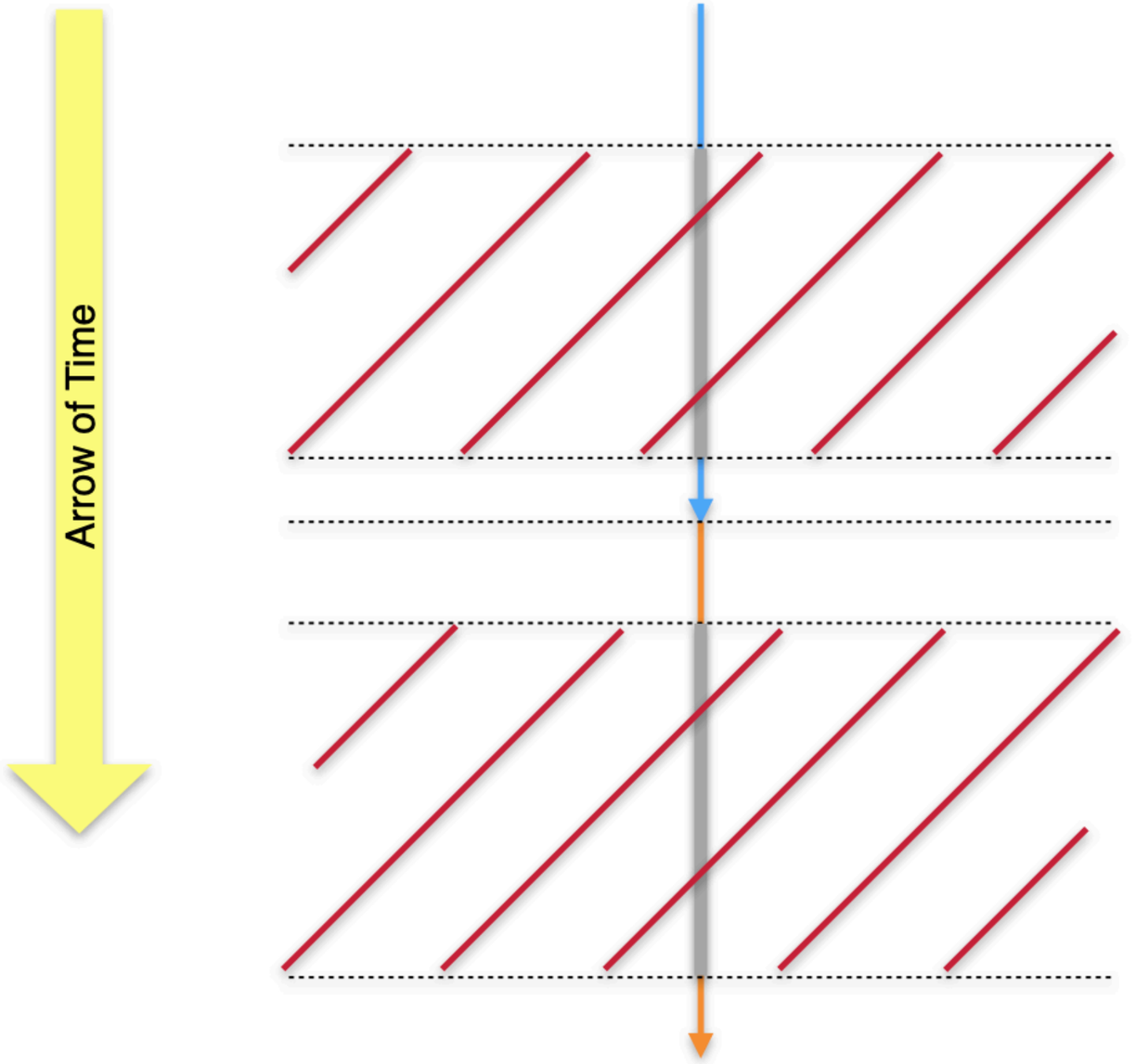
inquiries@jonathan.industries

Paralellism

- **Data Parallelism**: Split instances between machines
- **Model Paralellism**: Split parameters between machines
- **Task Paralellism**: Split algorithms between machines

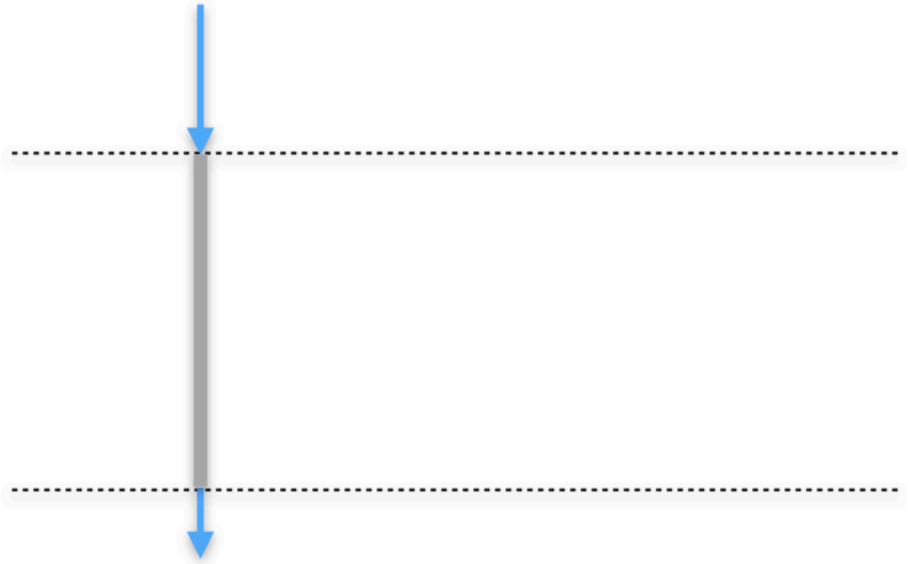
inquiries@jonathan.industries

Sequential



inquiries@jonathan.industries

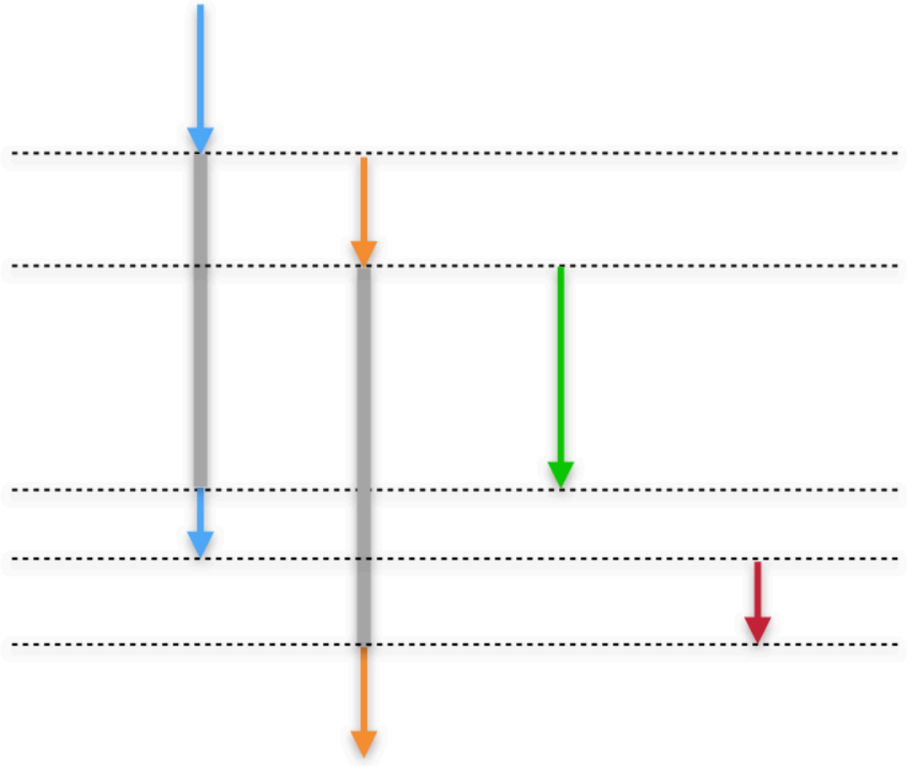
Concurrent



Parallel

inquiries@jonathan.industries

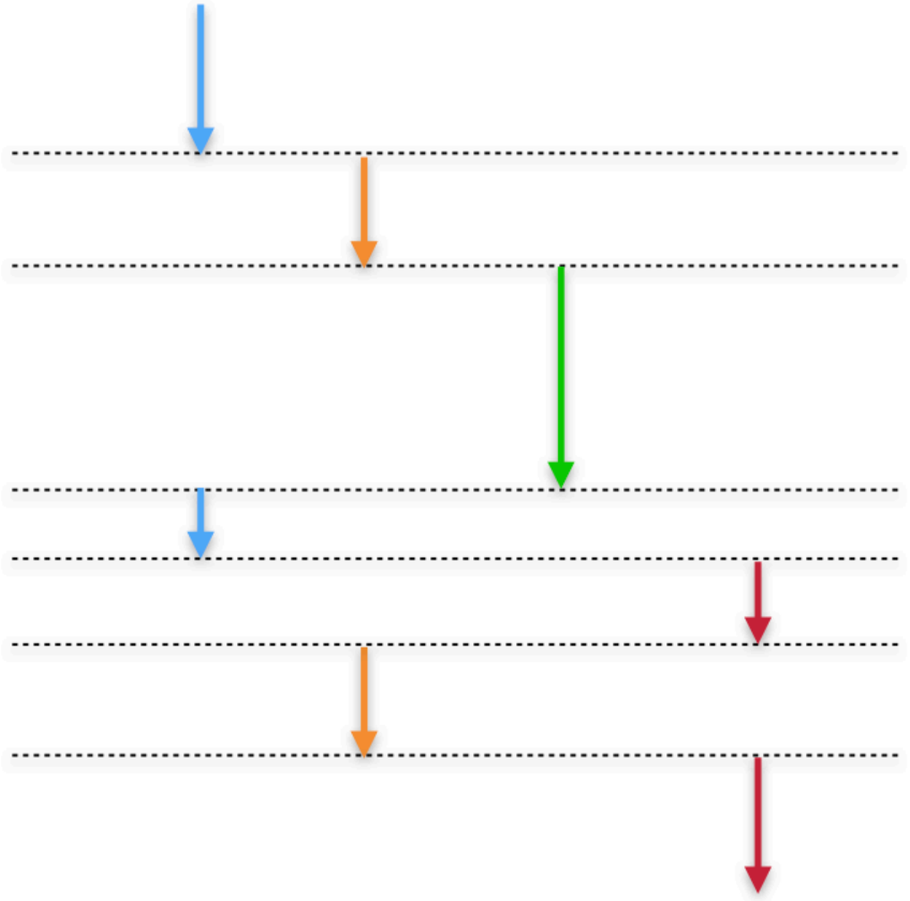
Concurrent



Parallel

inquiries@jonathan.industries

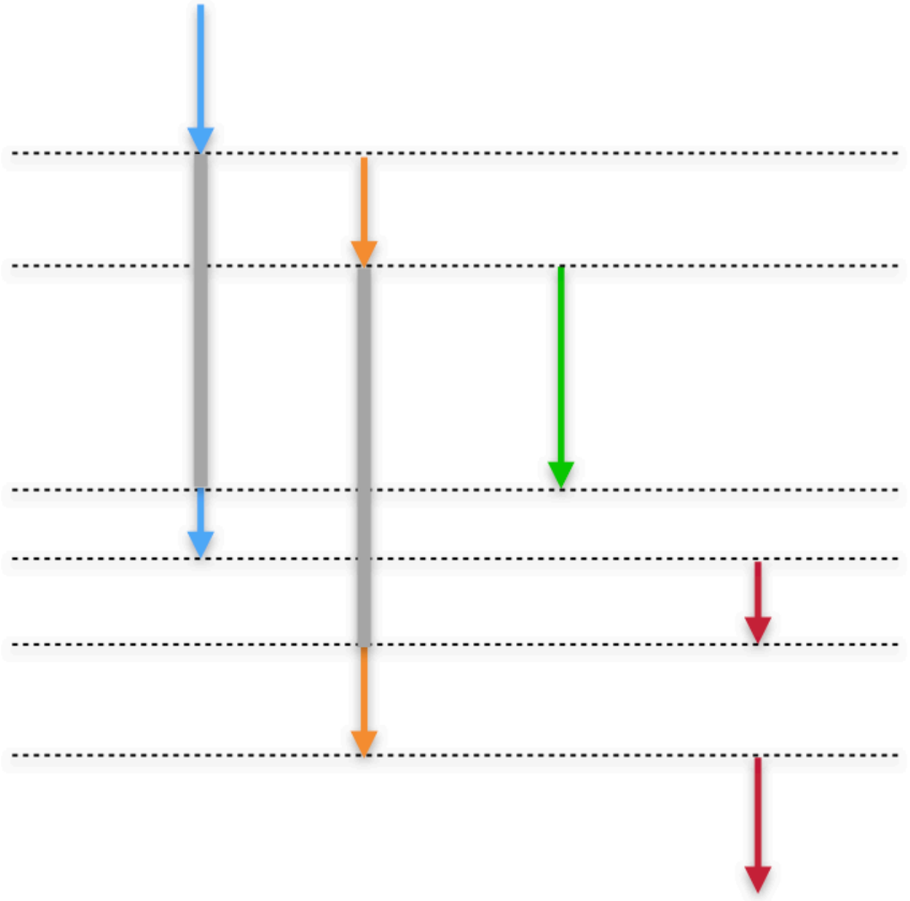
Concurrent



Parallel

inquiries@jonathan.industries

Concurrent



Parallel

inquiries@jonathan.industries

Problem Solved?

inquiries@jonathan.industries

Problem Solved?

- Coordination: what happens when and to whom? And how do you synchronize these events

inquiries@jonathan.industries

Problem Solved?

- *Coordination*: what happens when and to whom? And how do you synchronize these events
- *Communication*: how do the different nodes in your system talk to one another? And how do you talk to your nodes?

inquiries@jonathan.industries

Problem Solved?

- *Coordination*: what happens when and to whom? And how do you synchronize these events
- *Communication*: how do the different nodes in your system talk to one another? And how do you talk to your nodes?
- *Fault Tolerance*: is your system robust to network and machine failures (because they will happen)?

inquiries@jonathan.industries

This is what Ray helps us with 👇

- *Coordination*: what happens when and to whom? And how do you synchronize these events
- *Communication*: how do the different nodes in your system talk to one another? And how do you talk to your nodes?
- *Fault Tolerance*: is your system robust to network and machine failures (because they will happen)?

inquiries@jonathan.industries

CAP Theorem[2]

Consistency: all nodes see the same data at the same time

Availability: every request receives a response about whether it succeeded or failed

Partition tolerance: the system continues to operate despite arbitrary partitioning due to network failures

[2] Choose two: except you can't really sacrifice *P*!

inquiries@jonathan.industries

Break

inquiries@jonathan.industries

Why Ray?

inquiries@jonathan.industries

Why Ray?

- *Distributed computing framework* meant for a new generation of AI applications.

inquiries@jonathan.industries

Why Ray?

- *Distributed computing framework* meant for a new generation of AI applications.
- Emerged from *RL applications*

inquiries@jonathan.industries

Why Ray?

- *Distributed computing framework* meant for a new generation of AI applications.
- Emerged from *RL applications*
- *More generic* programming model than Spark/Flink/Hadoop/etc.

inquiries@jonathan.industries

Why Ray?

- *Distributed computing framework* meant for a new generation of AI applications.
- Emerged from *RL applications*
- *More generic* programming model than Spark/Flink/Hadoop/etc.
- More *fault tolerant* than MPI

inquiries@jonathan.industries

multiprocess meets RPC....

inquiries@jonathan.industries

The Ray programming model

inquiries@jonathan.industries

The Ray programming model

- Tasks (remote functions)

inquiries@jonathan.industries

The Ray programming model

- Tasks (remote functions)
- Actors (remote classes)

inquiries@jonathan.industries

Tasks

```
@ray.remote
def index(url):
    return requests.get(url)

futures = [index.remote(site) for site in internet]
print(ray.get(futures))
```


inquiries@jonathan.industries

Live Code

inquiries@jonathan.industries

Actors

```
@ray.remote
class Child(object):
    def __init__(self):
        self.name = random.choice(baby_names)
        self.age = 1

    def grow(self):
        self.age += 1

    def greet(self):
        return (
            f'My name is {self.name} '
            f'and I am {self.age} years old'
        )
```

```
children = [Child.remote() for i in range(4)]
[c.grow.remote() for c in children]
futures = [c.greet.remote() for c in children]
print(ray.get(futures))
```


inquiries@jonathan.industries

Live Code

inquiries@jonathan.industries

Break

inquiries@jonathan.industries

Distributed Training Architectures

inquiries@jonathan.industries

Distributed Training Architectures

- Parallelized Stochastic Gradient Descent (2010)

inquiries@jonathan.industries

Distributed Training Architectures

- Parallelized Stochastic Gradient Descent (2010)
- Hogwild! (2011)

inquiries@jonathan.industries

Distributed Training Architectures

- Parallelized Stochastic Gradient Descent (2010)
- Hogwild! (2011)
- Downpour SGD (DistBelief) (2012)

inquiries@jonathan.industries

Distributed Training Architectures

- Parallelized Stochastic Gradient Descent (2010)
- Hogwild! (2011)
- Downpour SGD (DistBelief) (2012)
- AllReduce (2014)

inquiries@jonathan.industries

Distributed Training Architectures

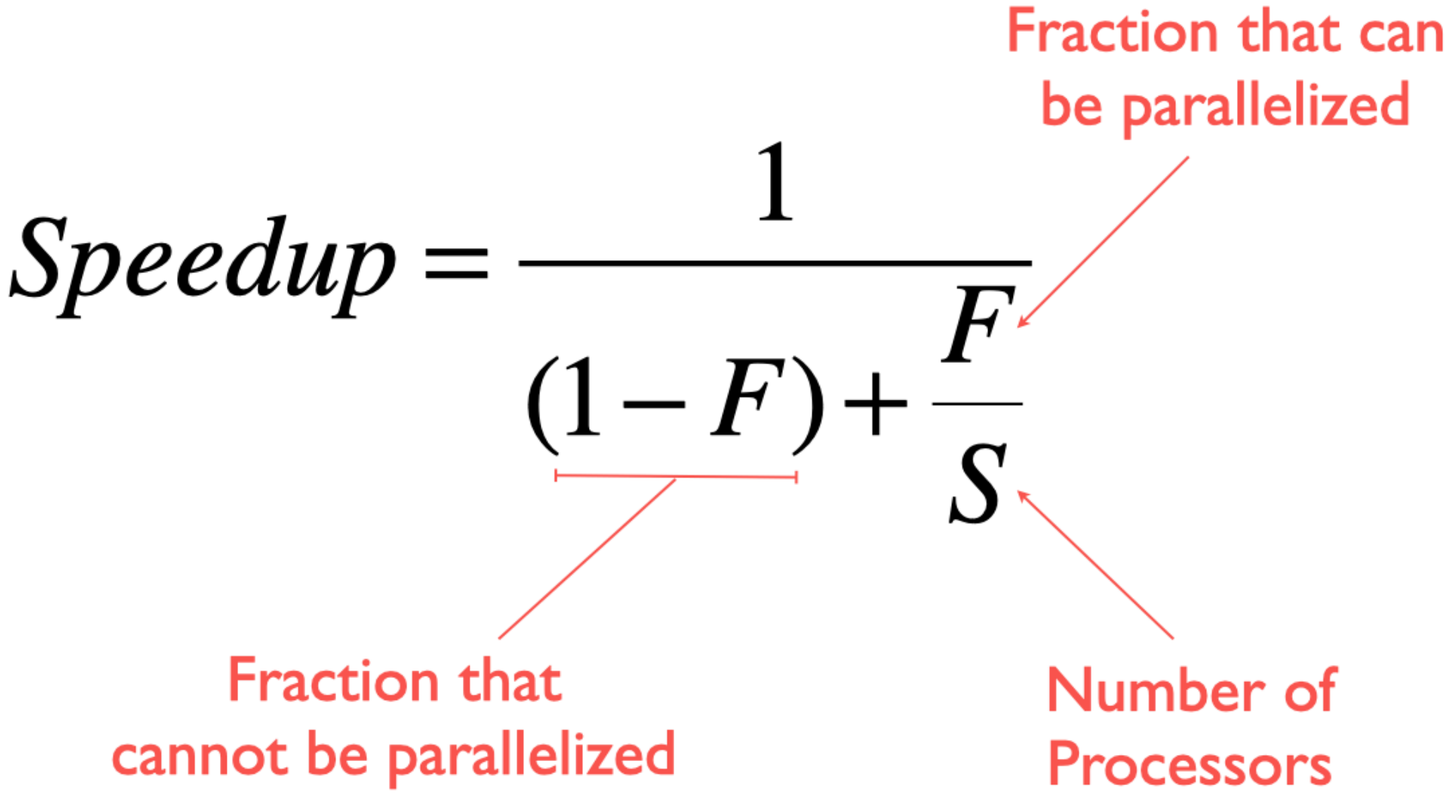
- Parallelized Stochastic Gradient Descent (2010)
- Hogwild! (2011)
- Downpour SGD (DistBelief) (2012)
- AllReduce (2014)
- Parameter Server (2014)

inquiries@jonathan.industries

Will paralellization help me?

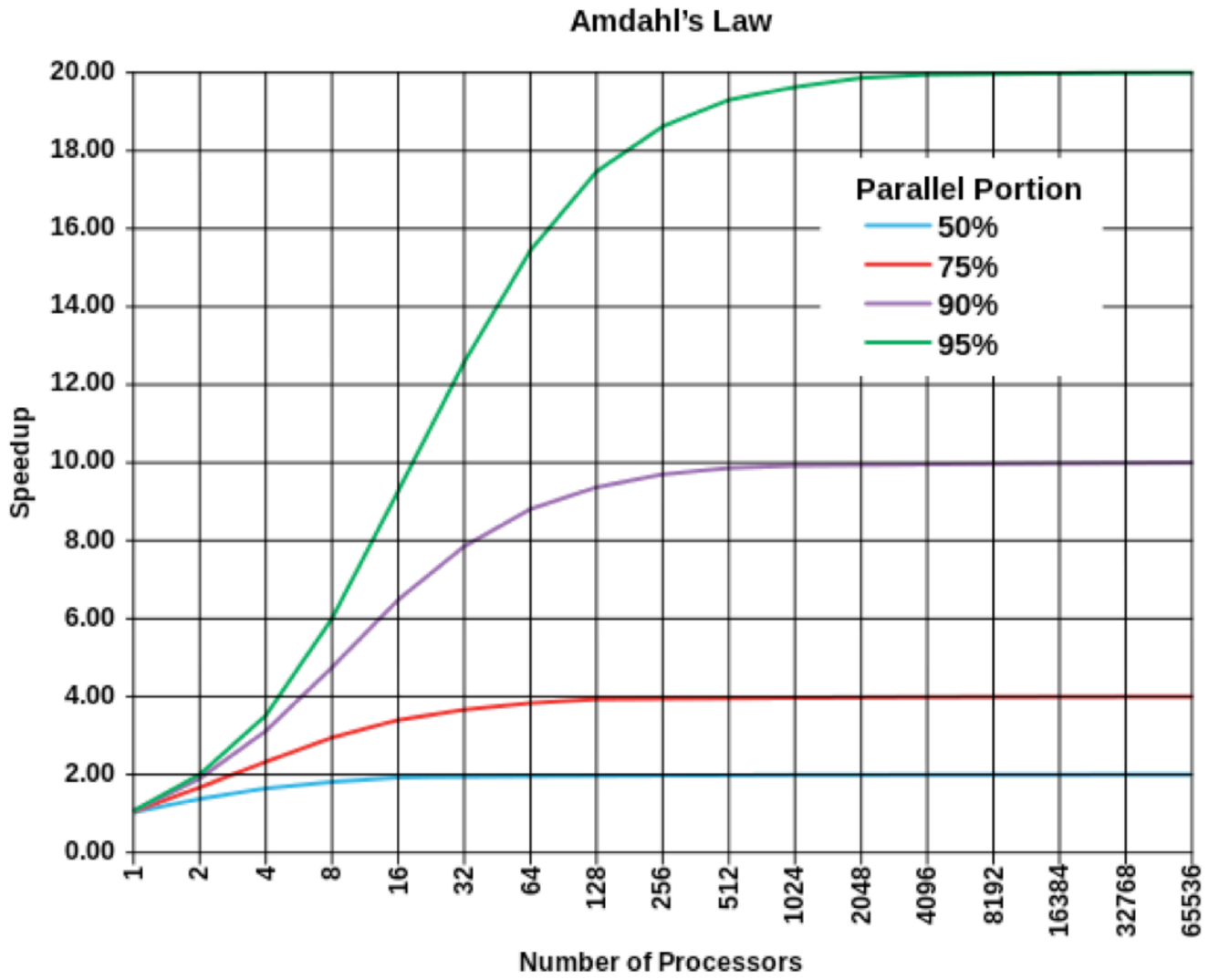
inquiries@jonathan.industries

Amdahl's Law



inquiries@jonathan.industries

Amdahl's Law



inquiries@jonathan.industries

Amdahl's Law

20x improvement
25% of the time

Fraction that can be parallelized

$$Speedup = \frac{1}{(1 - 0.25) + \frac{0.25}{20}}$$

Fraction that cannot be parallelized

Number of Processors

inquiries@jonathan.industries

Amdahl's Law

20x improvement
25% of the time

Only 1.3x Speedup!

$$Speedup = \frac{1}{0.75 + \frac{0.25}{20}} = 1.31$$

Even with 20 processors

inquiries@jonathan.industries

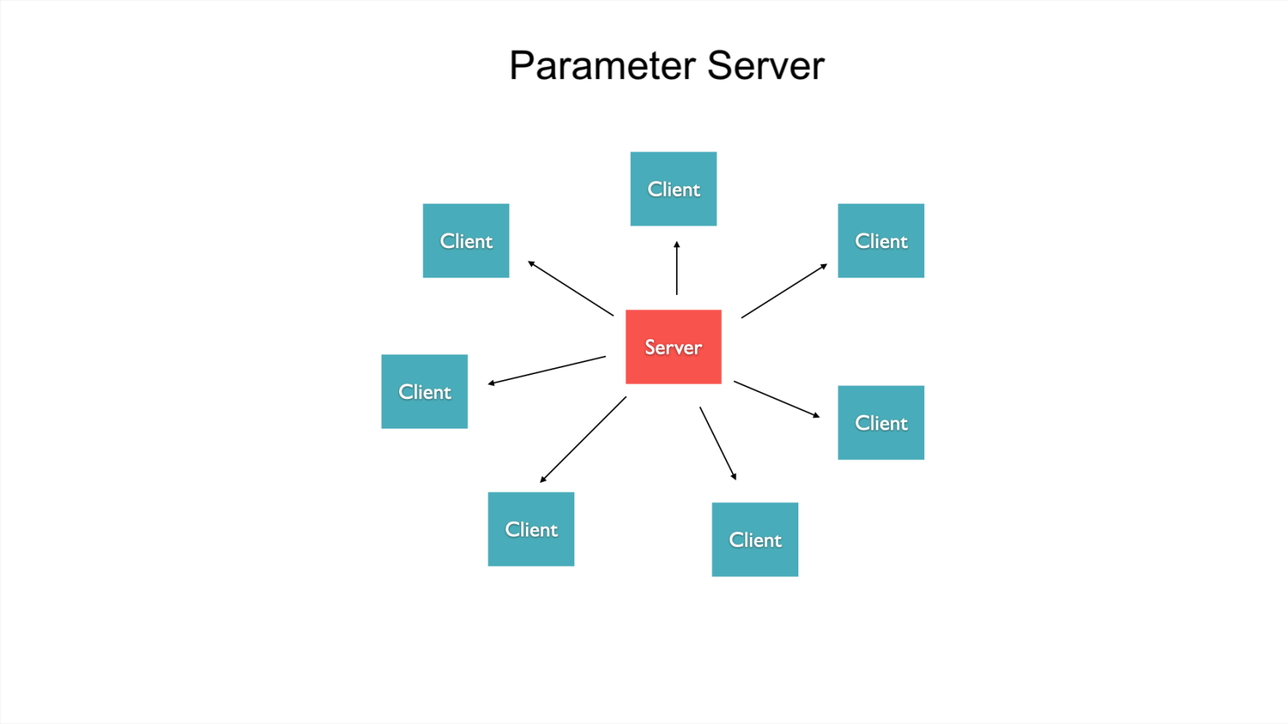
Moral?

inquiries@jonathan.industries

It doesn't how many machines you throw at a problem... at some point it is just not going to run any faster

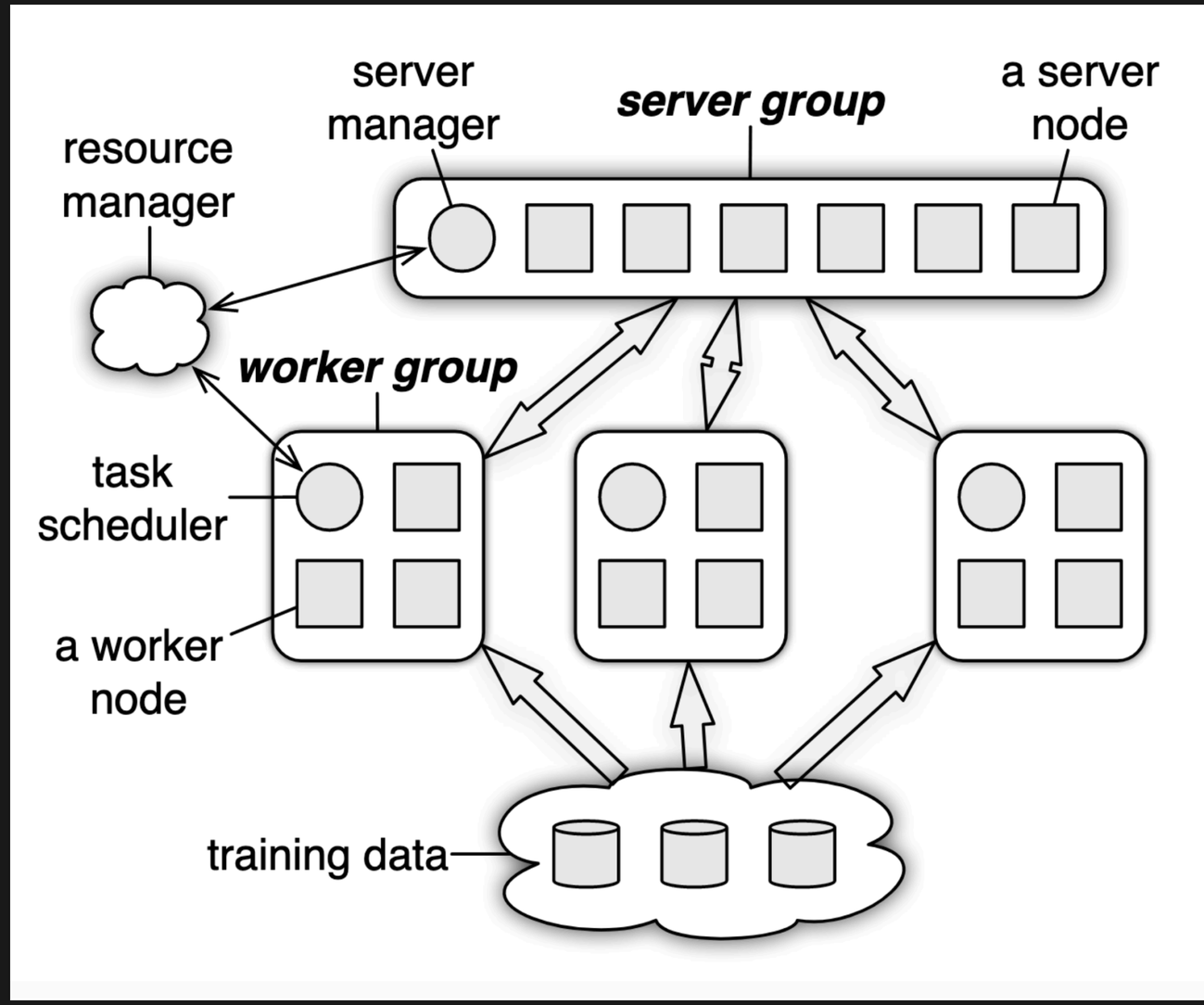
inquiries@jonathan.industries

Distributed training with the parameter server



inquiries@jonathan.industries

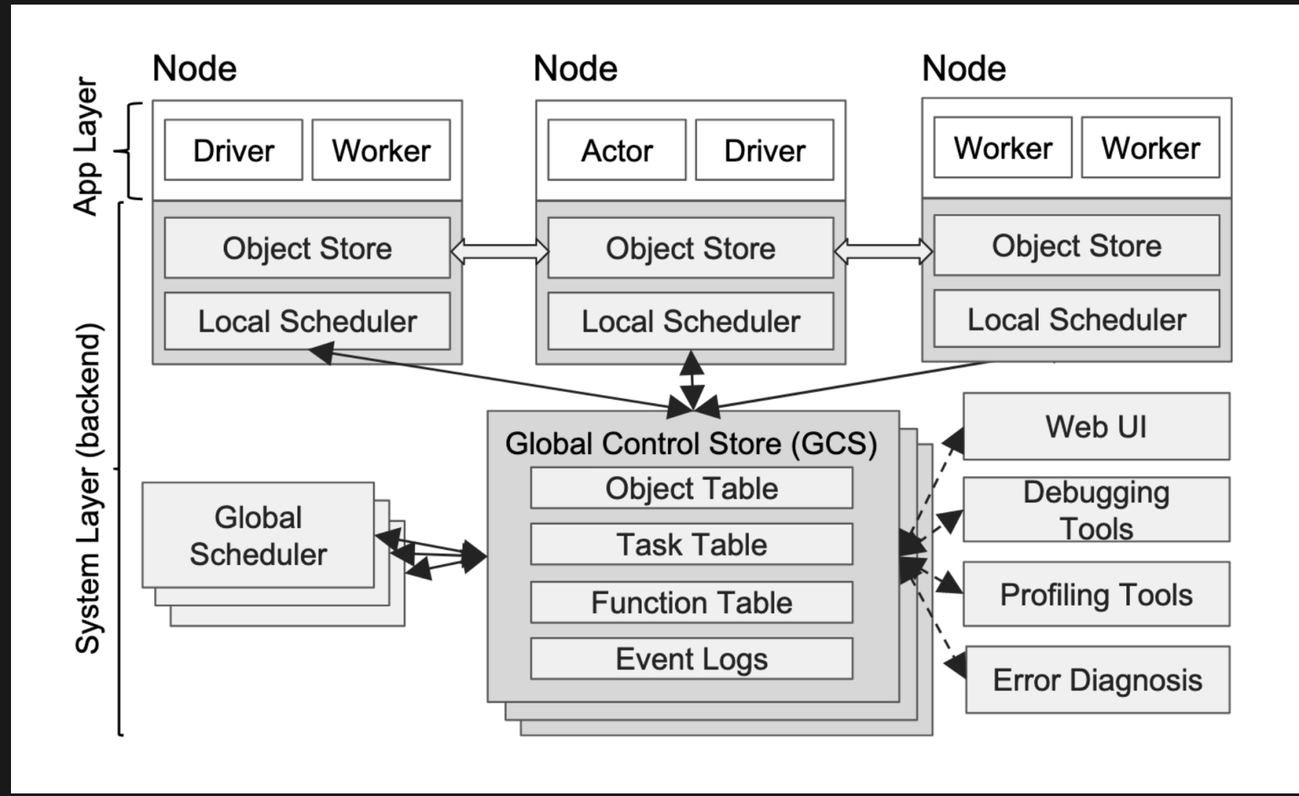
System rather than algorithm[3]



[3] Scaling Distributed Machine Learning with the Parameter Server

inquiries@jonathan.industries

Ray internals[4]



[4] Ray: A Distributed Framework for Emerging AI Applications

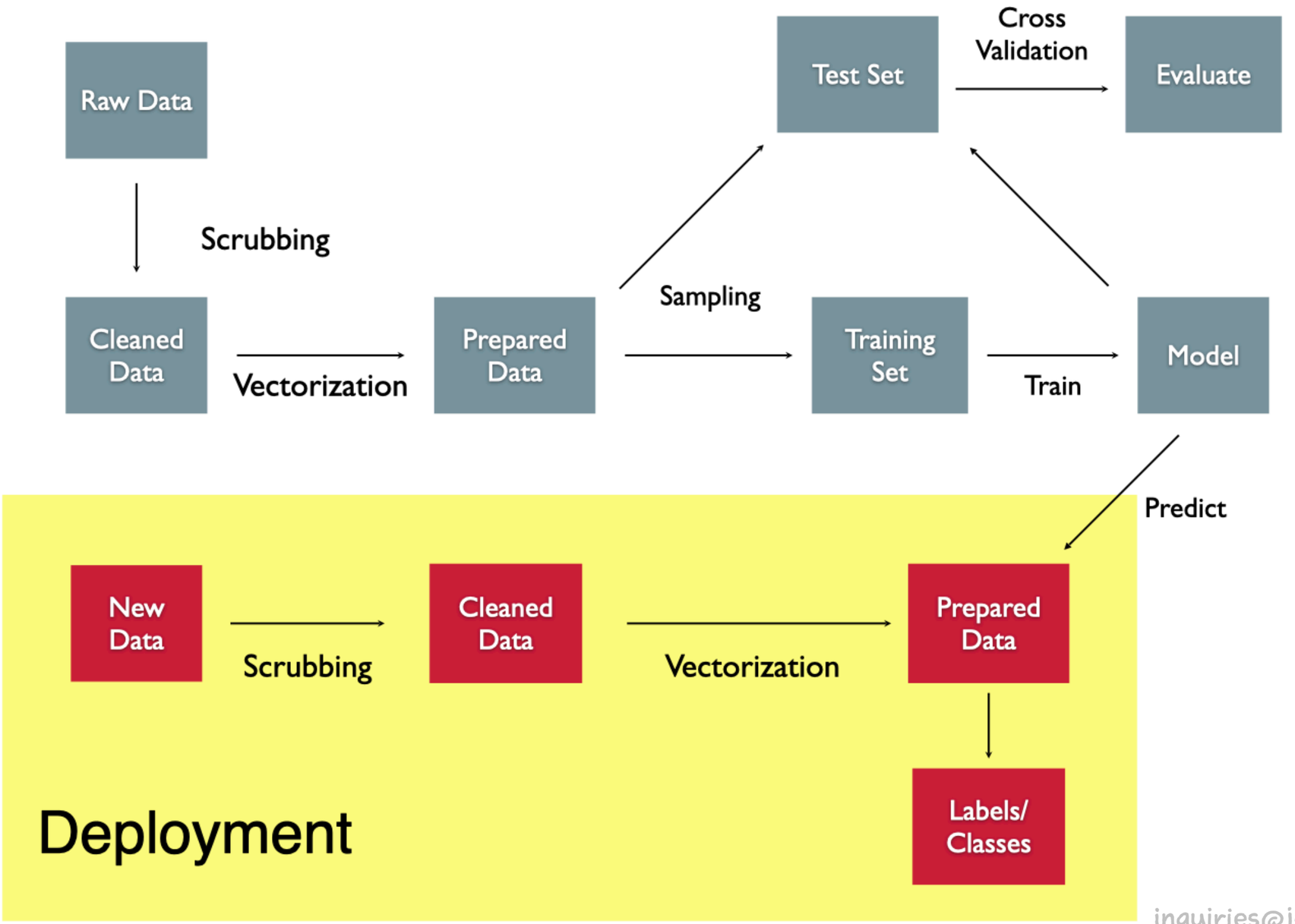
inquiries@jonathan.industries

Live Code

inquiries@jonathan.industries

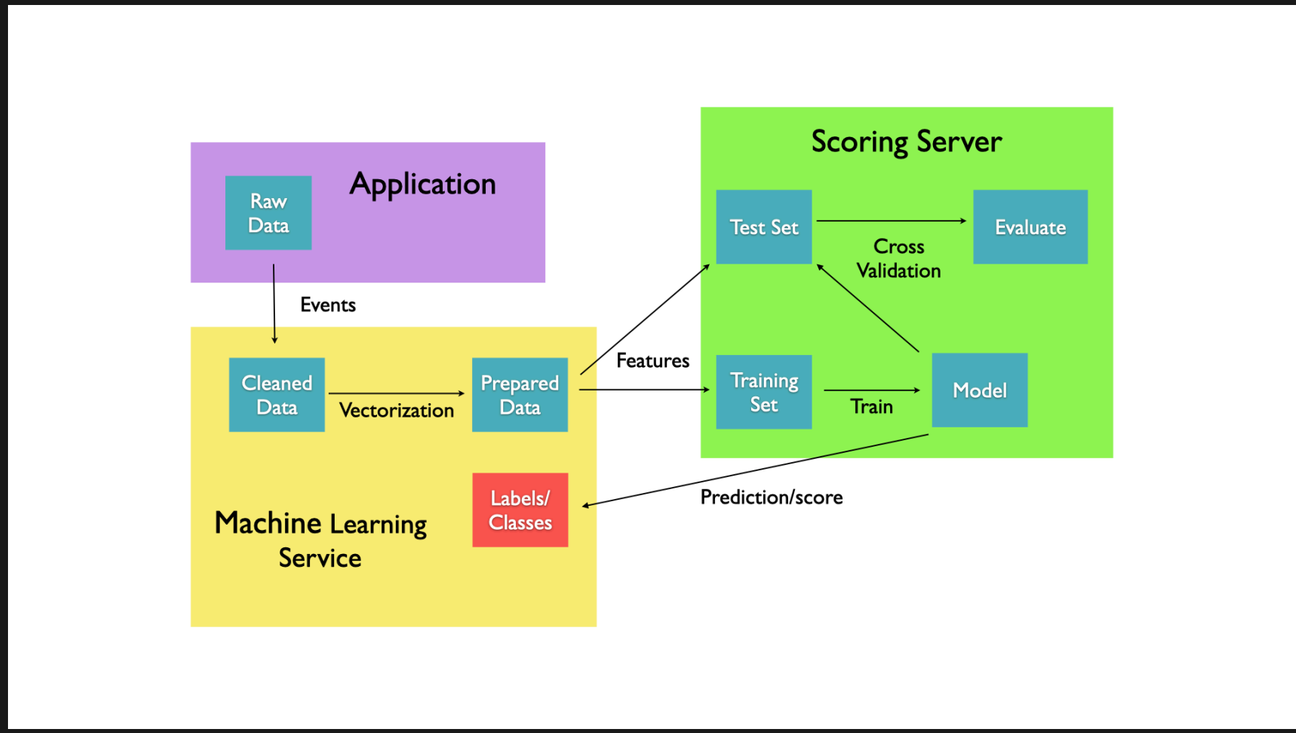
Deploying ML

inquiries@jonathan.industries



inquiries@jonathan.industries

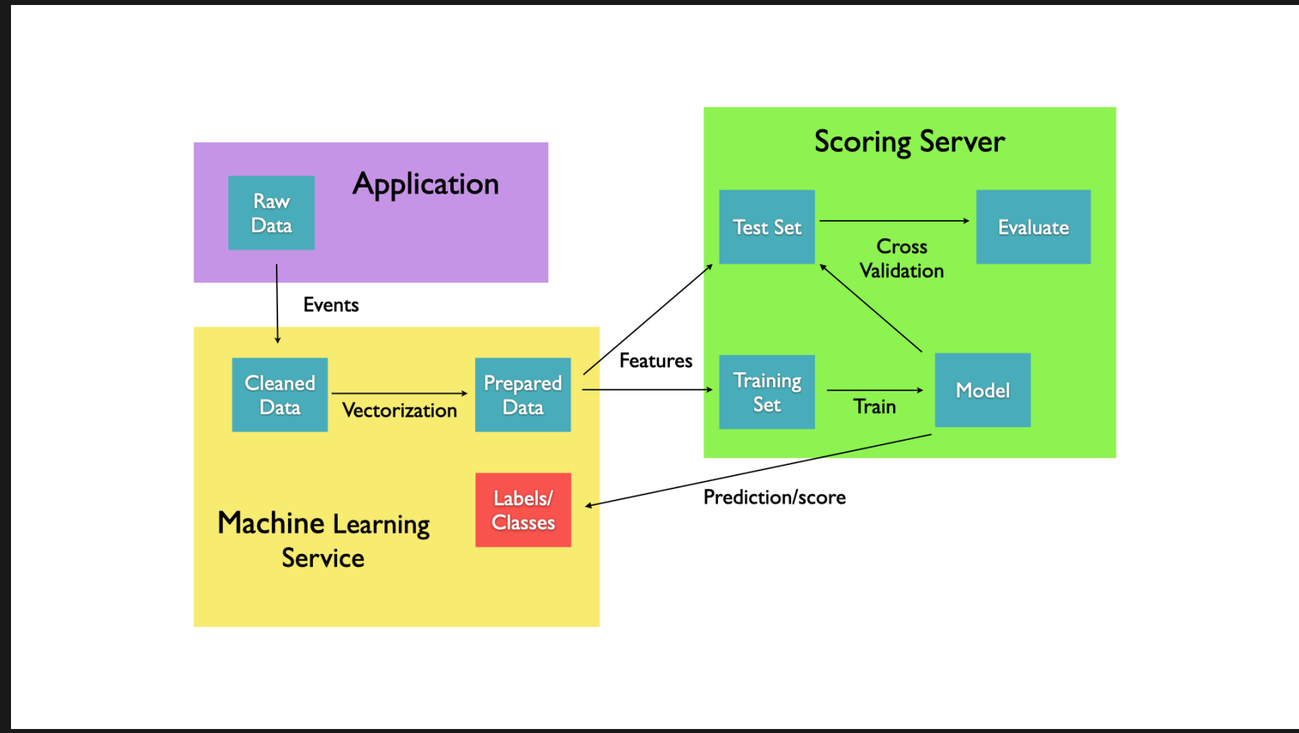
ML infrastructure is varied



inquiries@jonathan.industries

ML infrastructure is varied

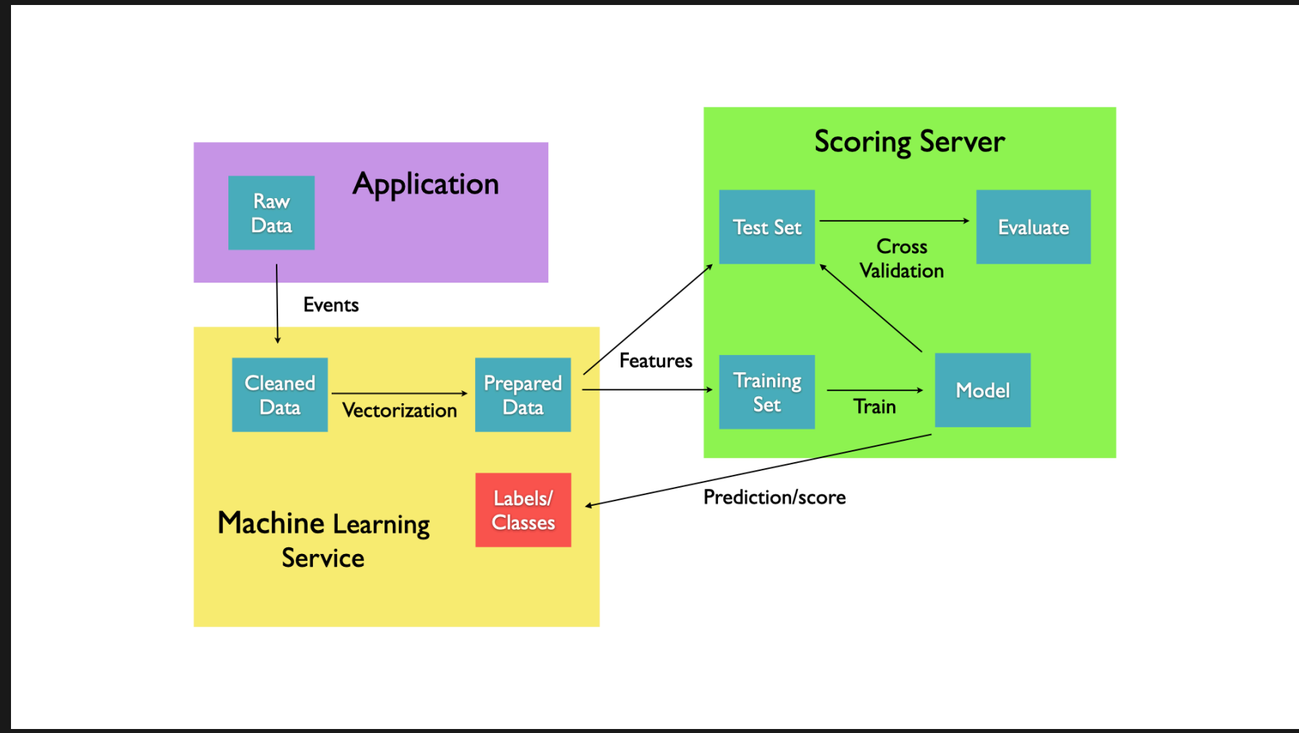
- MLaaS (SageMaker, Google AutoML, etc.)



inquiries@jonathan.industries

ML infrastructure is varied

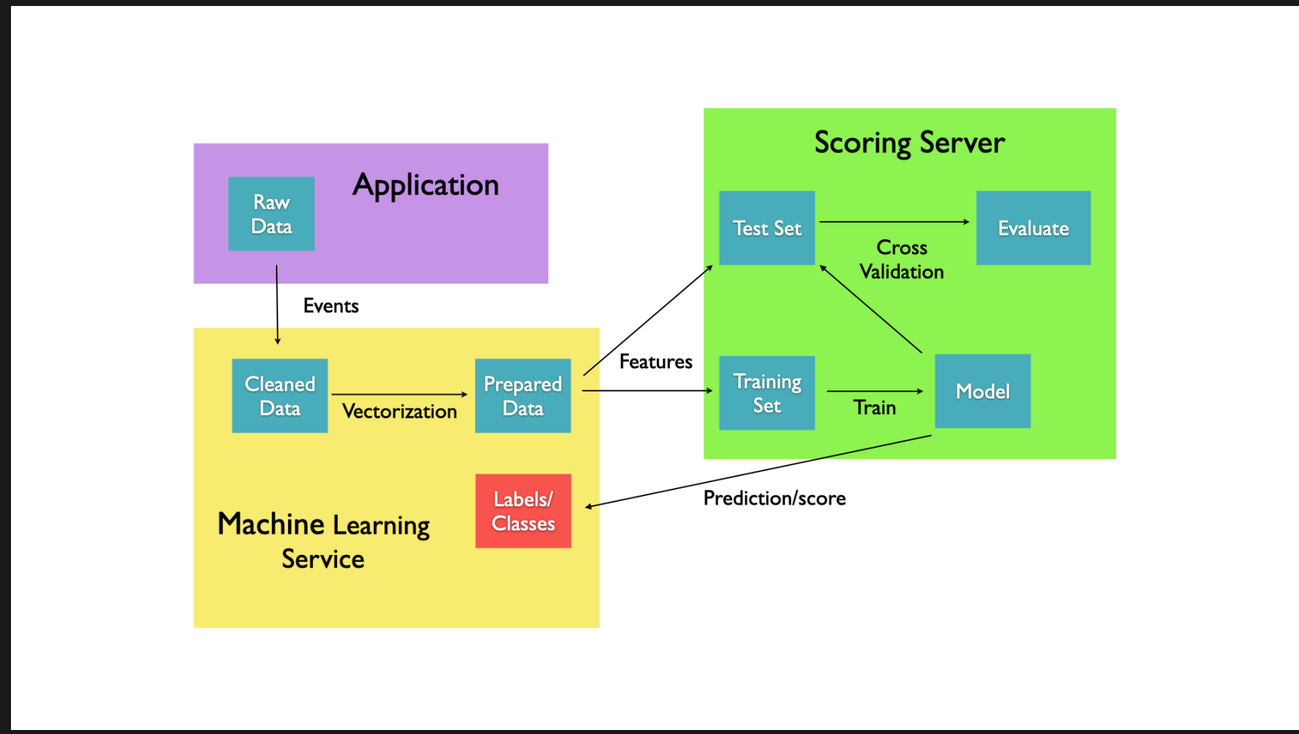
- MLaaS (SageMaker, Google AutoML, etc.)
- Containers (Fargate, GKE, etc.)



inquiries@jonathan.industries

ML infrastructure is varied

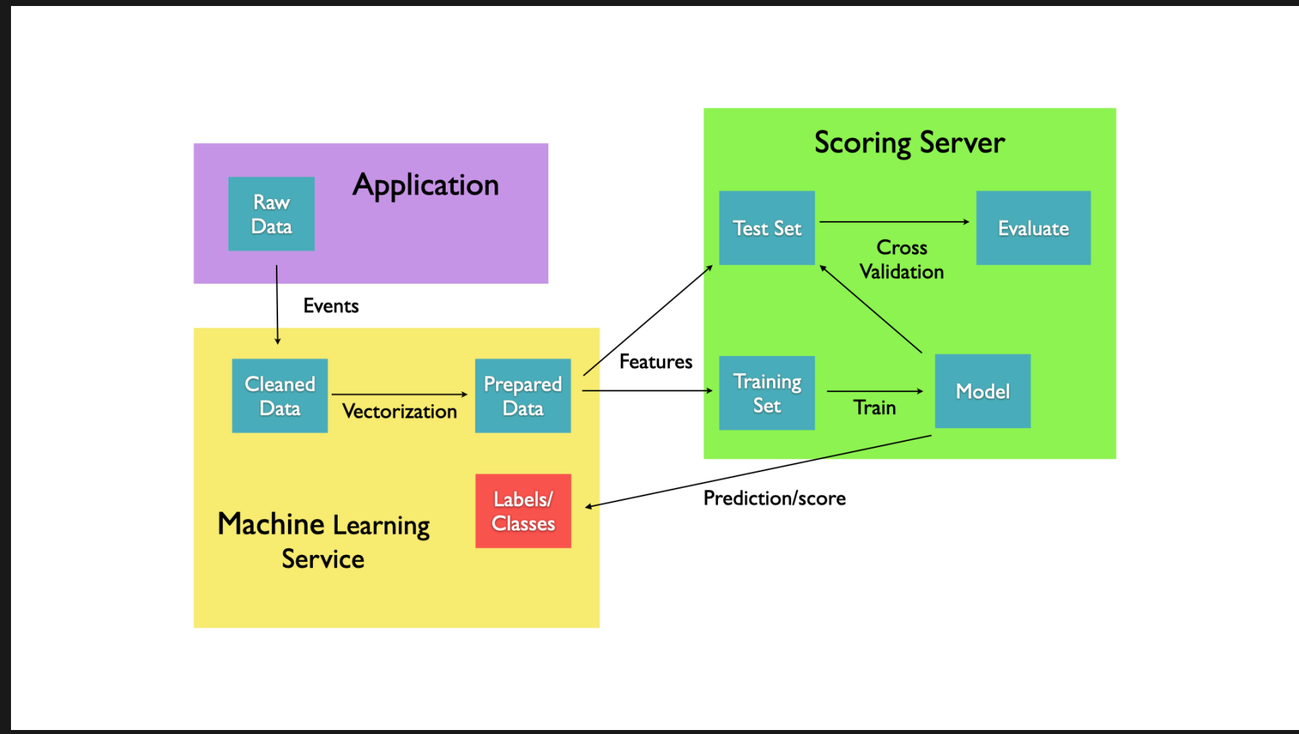
- MLaaS (SageMaker, Google AutoML, etc.)
- Containers (Fargate, GKE, etc.)
- IaaS (Ec2, GCP, etc.)



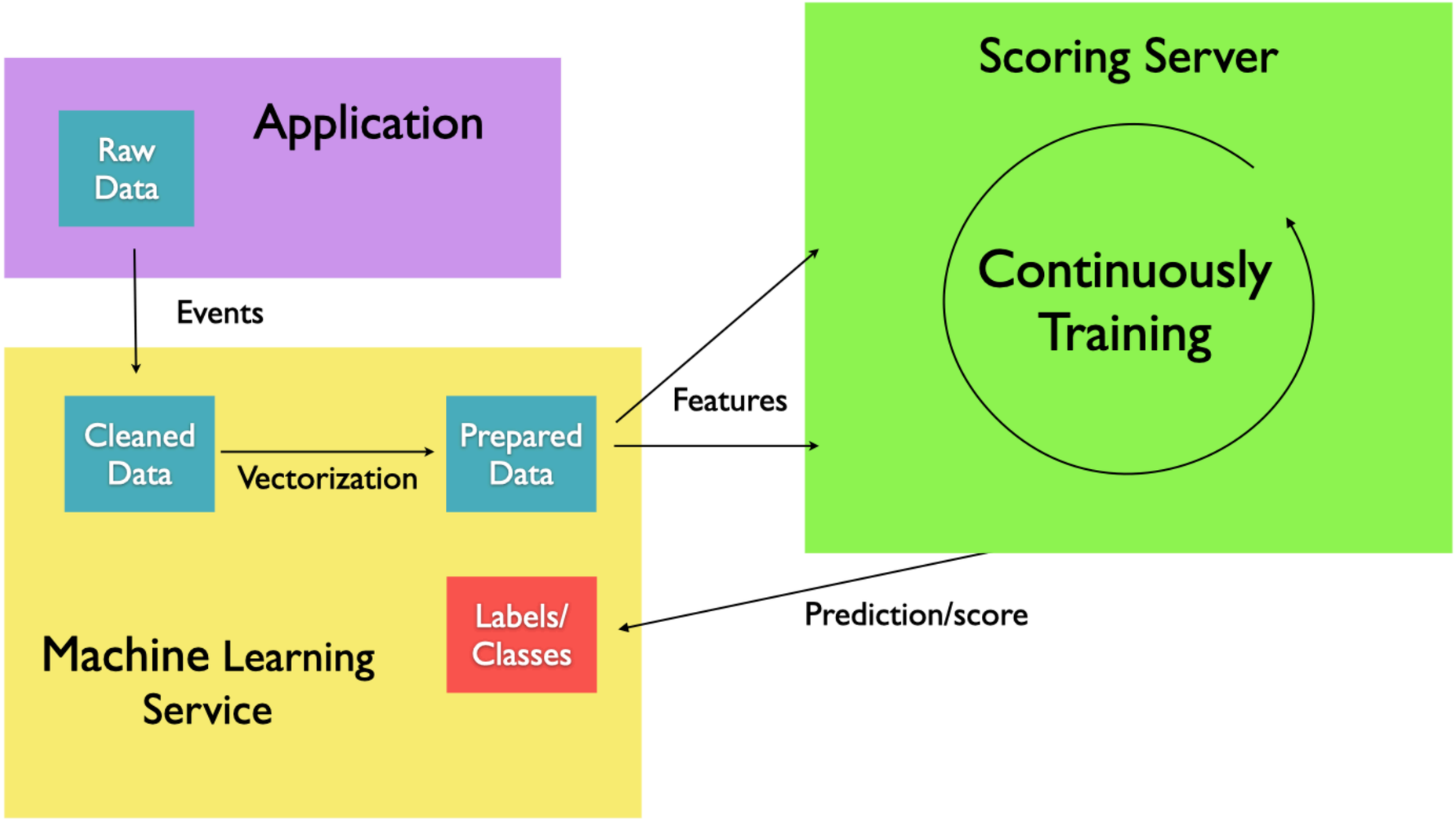
inquiries@jonathan.industries

ML infrastructure is varied

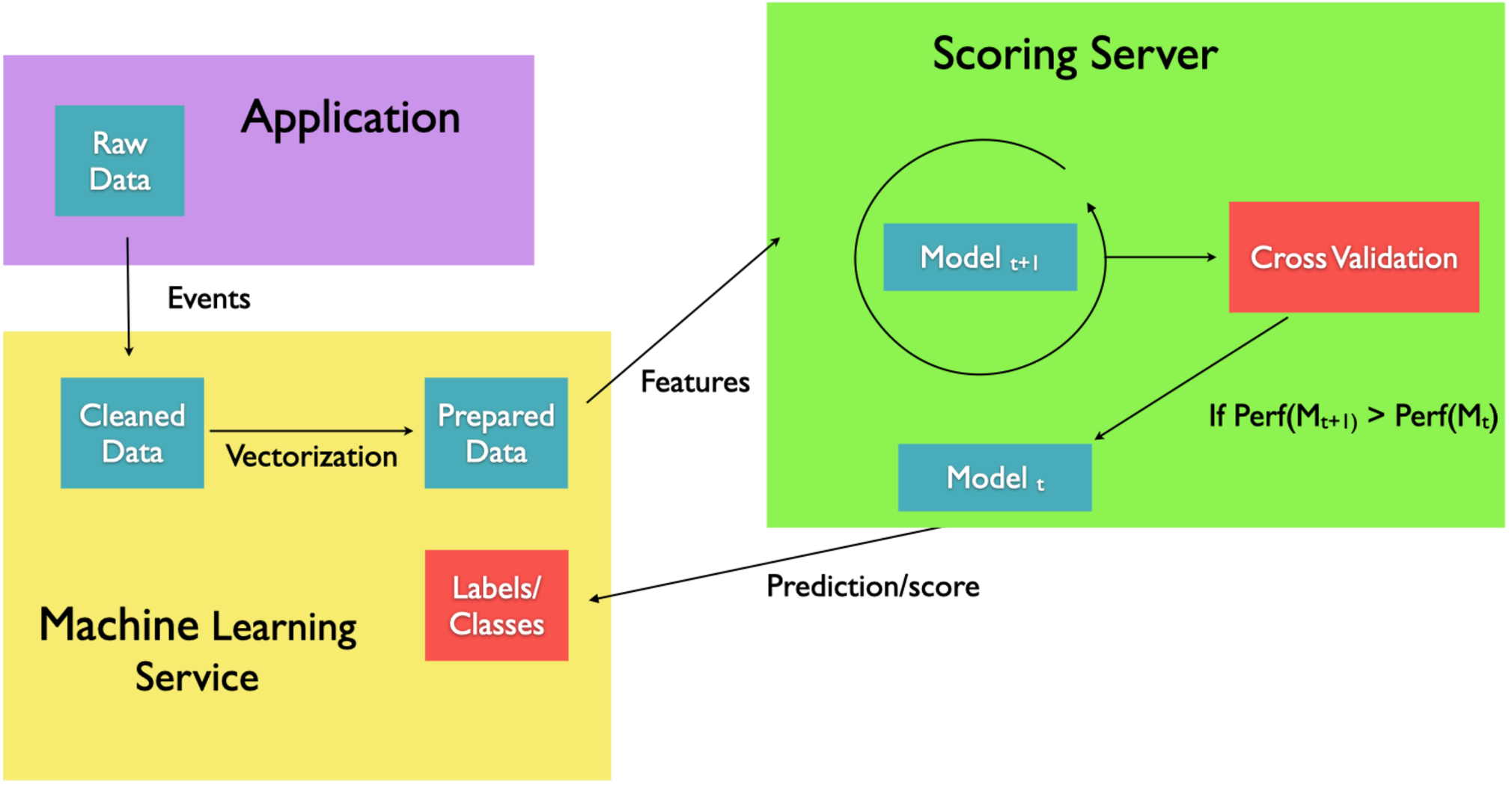
- MLaaS (SageMaker, Google AutoML, etc.)
- Containers (Fargate, GKE, etc.)
- IaaS (Ec2, GCP, etc.)
- On premise/data center



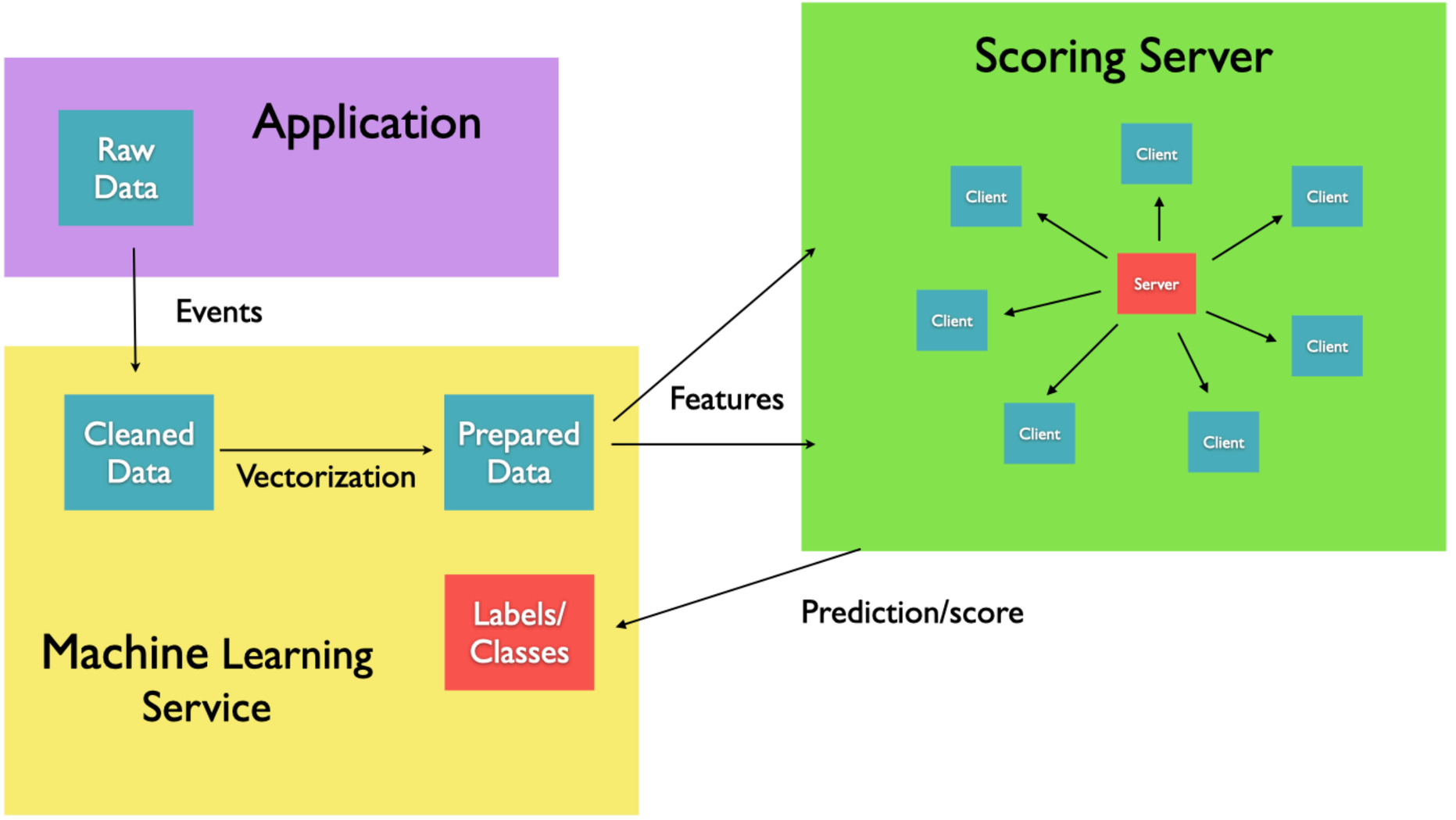
inquiries@jonathan.industries



inquiries@jonathan.industries



inquiries@jonathan.industries



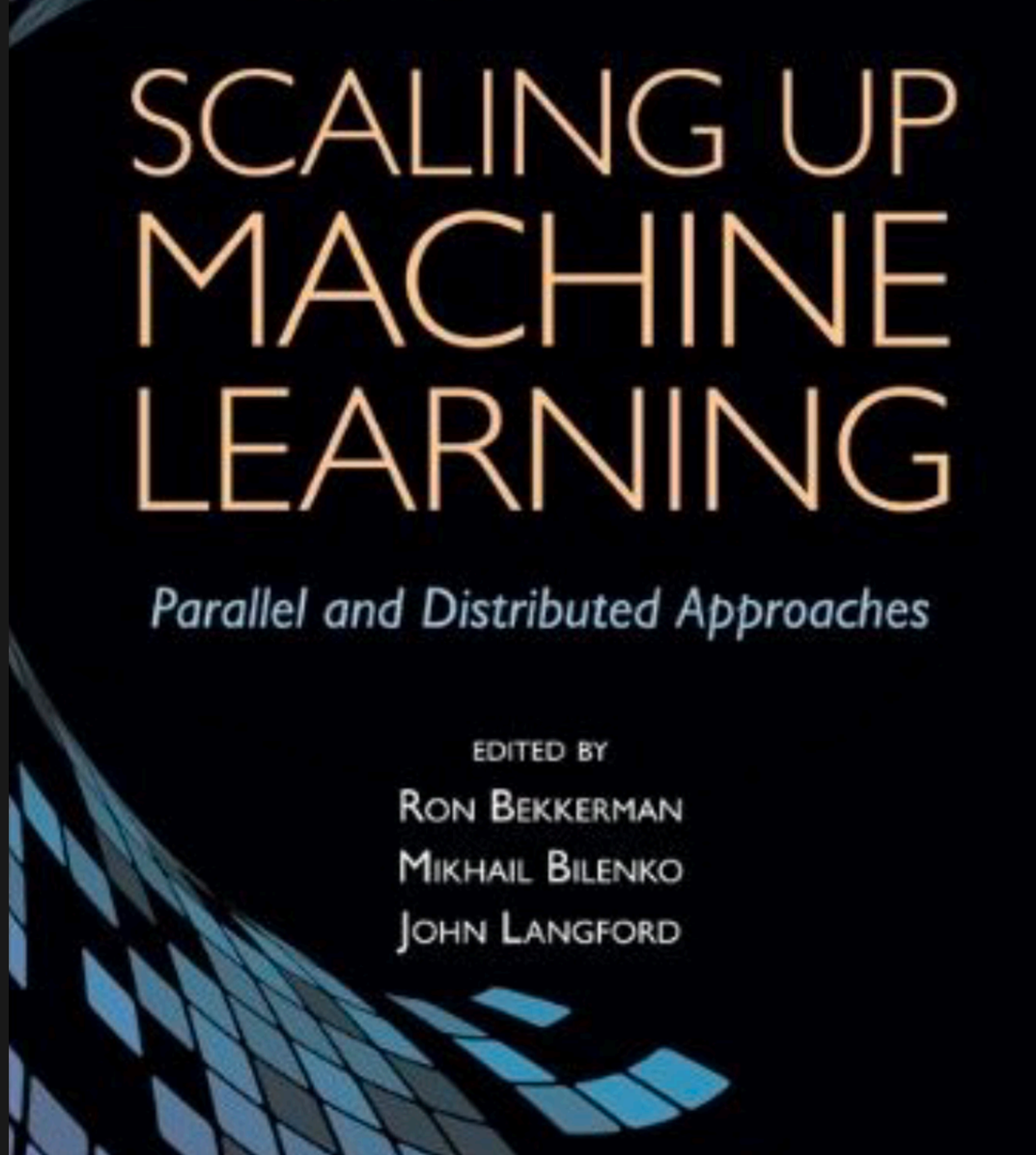
inquiries@jonathan.industries

Next Steps

inquiries@jonathan.industries

Learning More DistML

Scaling up Machine Learning



Future Live Training

inquiries@jonathan.industries

Future Live Training

- Bayesian Hyperparameter Optimization (Sept. 10)

inquiries@jonathan.industries

Q & A

Materials, mailing list, questions, feedback 👇

https://sr.ht/~hyphaebeast/ray-live-training/

inquiries@jonathan.industries