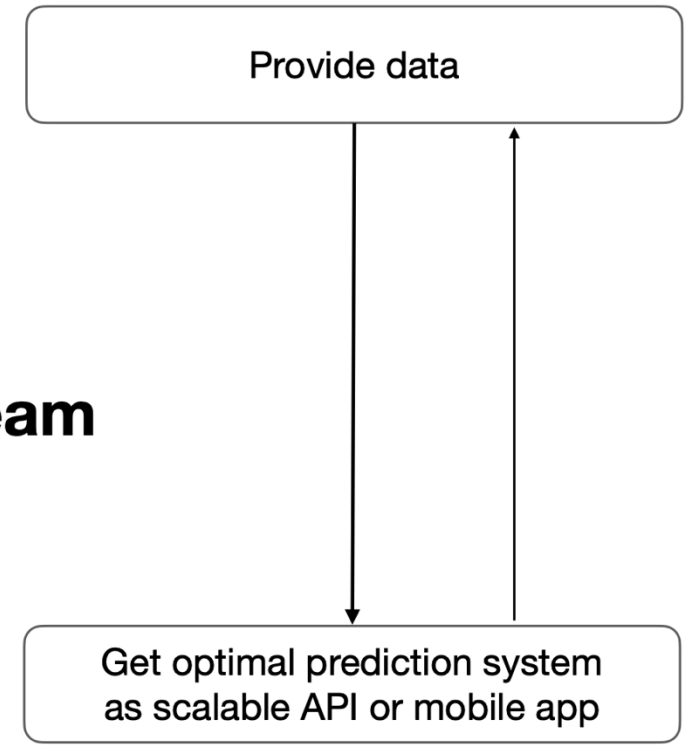
[COMSE6998-015] Fall 2024 Introduction to Deep Learning and LLM based Generative AI Systems

Lecture 3

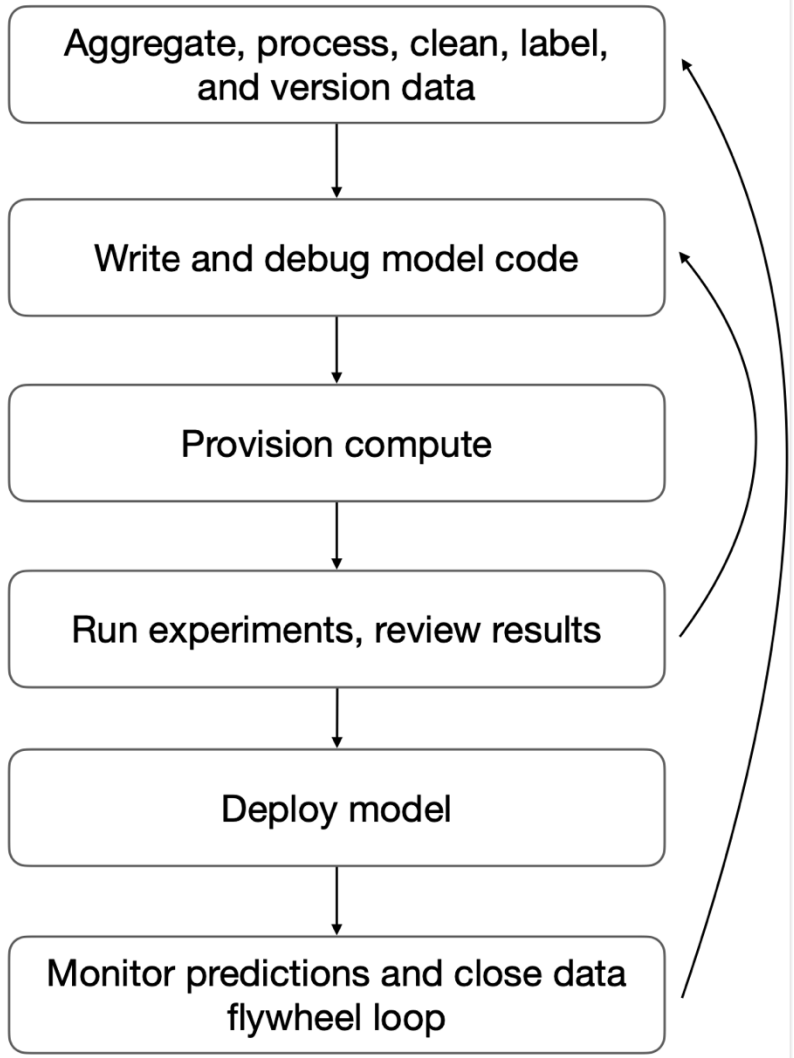
Today's Agenda

- Machine Learning System Stack
- Resource Management
- Slurm, Docker, Kubernetes
- Cloud based ML platforms from AWS, Microsoft, Google
- Ray
- TorchX

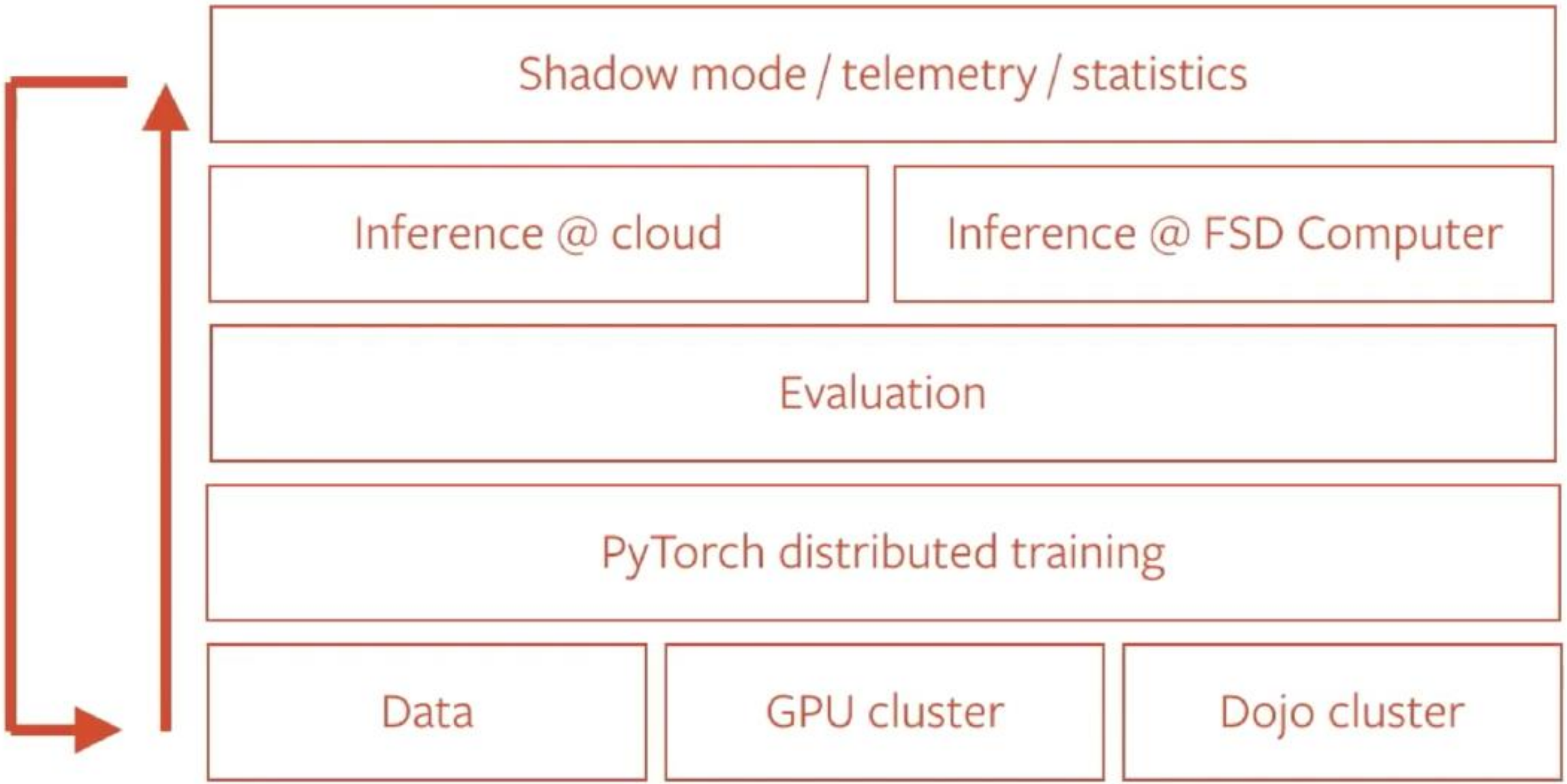
Dream

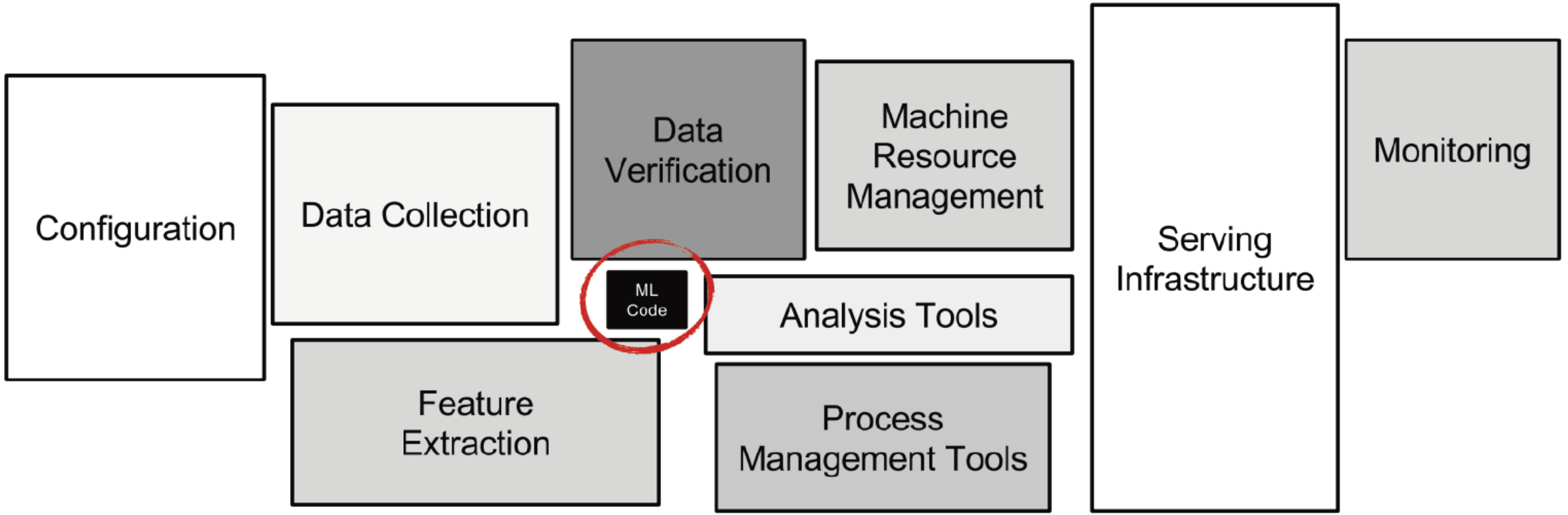


Reality



"OPERATION VACATION"





Machine Learning:
The High-Interest Credit Card of Technical Debt

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov,
Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young

Amazon SageMaker · gradient by Paperspace · FLOYD · DOMINO DATA LAB

"All-in-one"

Data

- Versioning: DVC, Pachyderm, LIQUIDATA
- Labeling: figure eight, scale, Aquarium
- Processing: Spark, DAGSTER
- Exploration: pandas, RAPIDS, dbt
- Data Lake / Warehouse: snowflake, databricks
- Sources: S3, LOG, Parquet

Training/Evaluation

- Frameworks & Distributed Training: fast.ai, PYTORCH, RAY, K, HOROVOD
- Hyperparameter Tuning: SIGOPT, Determined AI, Weights & Biases, tune
- Experiment Management: Weights & Biases, comet, TensorBoard, Neptune, mlflow
- Resource Management: Determined AI, slurm
- Software Engineering: jupyter, Streamlit, git
- Compute: CoreWeave, Lambda

Deployment

- Feature Store: tecton
- Monitoring: fiddler
- Edge: ML, NVIDIA TensorRT, TensorFlow Lite, ONNX
- Web: ALGORITHMIA, SELDON
- CI / Testing: Buildkite, great_expectations
- or · or

Compute needs

Development

Function

- Writing code
- Debugging models
- Looking at results

Desiderata

- Quickly compile models and run training
- Nice-to-have: use GUI

Solutions

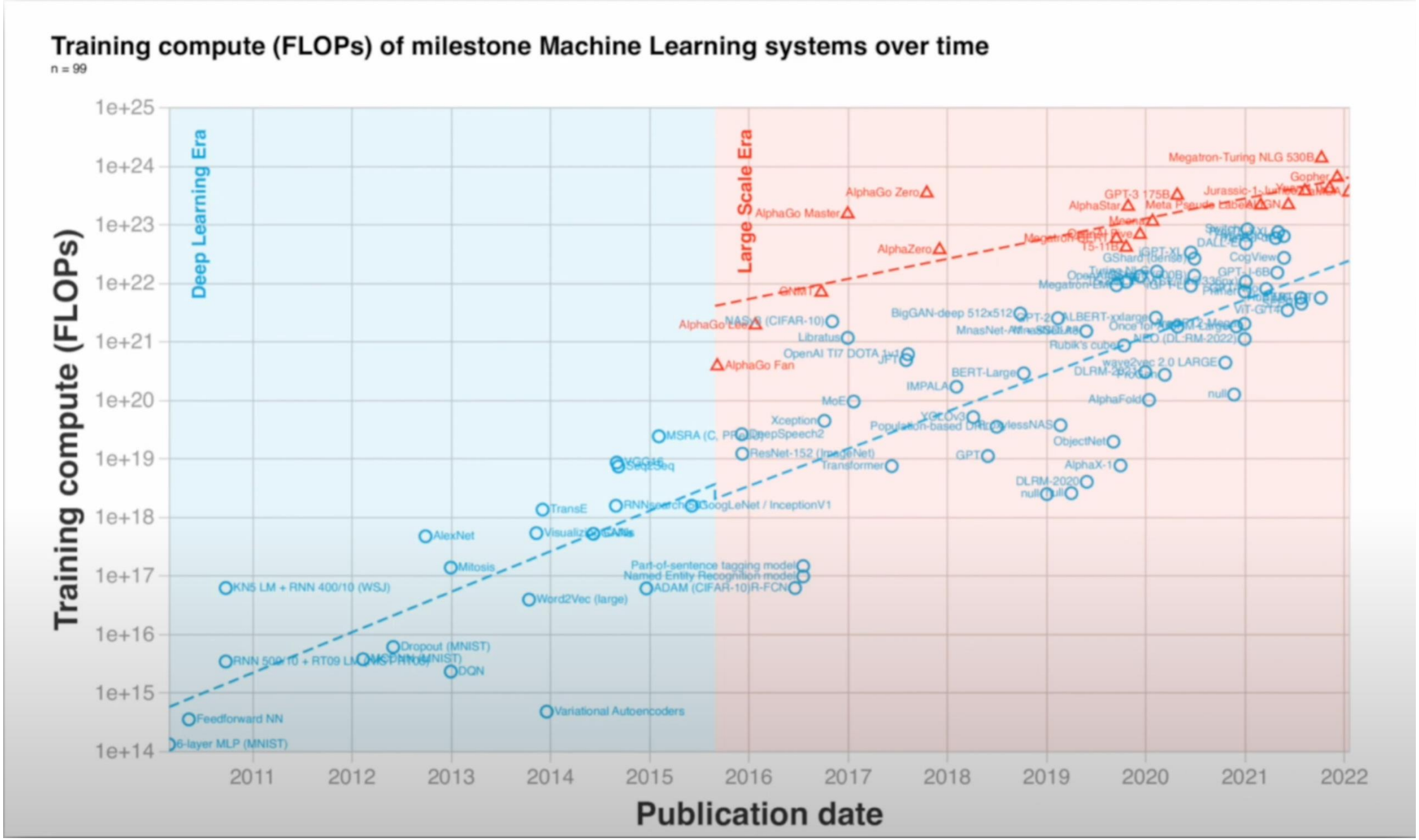
- Desktop with 1-4 GPUs
- Cloud instance with 1-4 GPUs

 or 

Training/Evaluation

- **Function**
  - Model architecture / hyperparam search
  - Training large models
- **Desiderata**
  - Easy to launch experiments and review results
- **Solutions**
  - Desktop with 4 GPUs
  - Private cluster of GPU machines
  - Cloud cluster of GPU instances

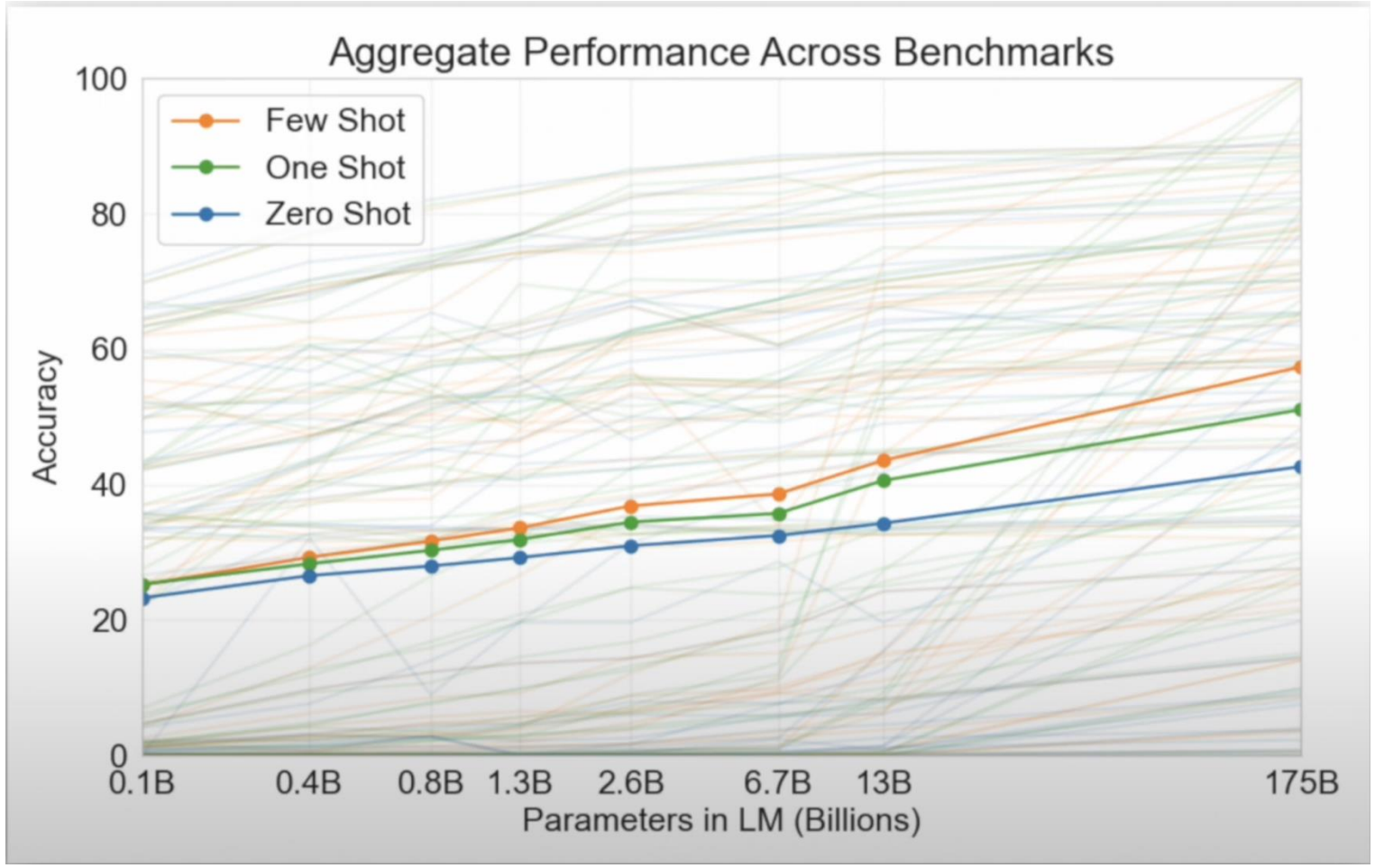
 or 



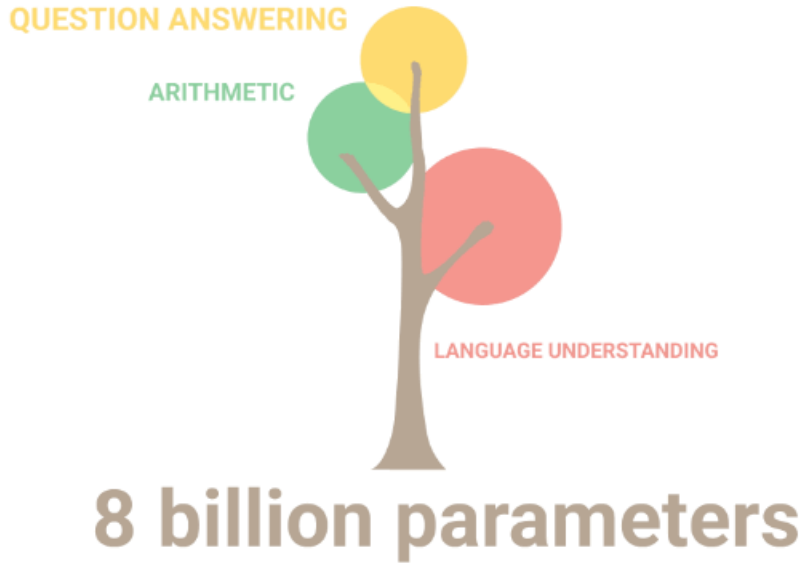
Compute Trends across Three Eras of Machine Learning, https://arxiv.org/pdf/2202.05924

Why? Bigger model, better accuracy

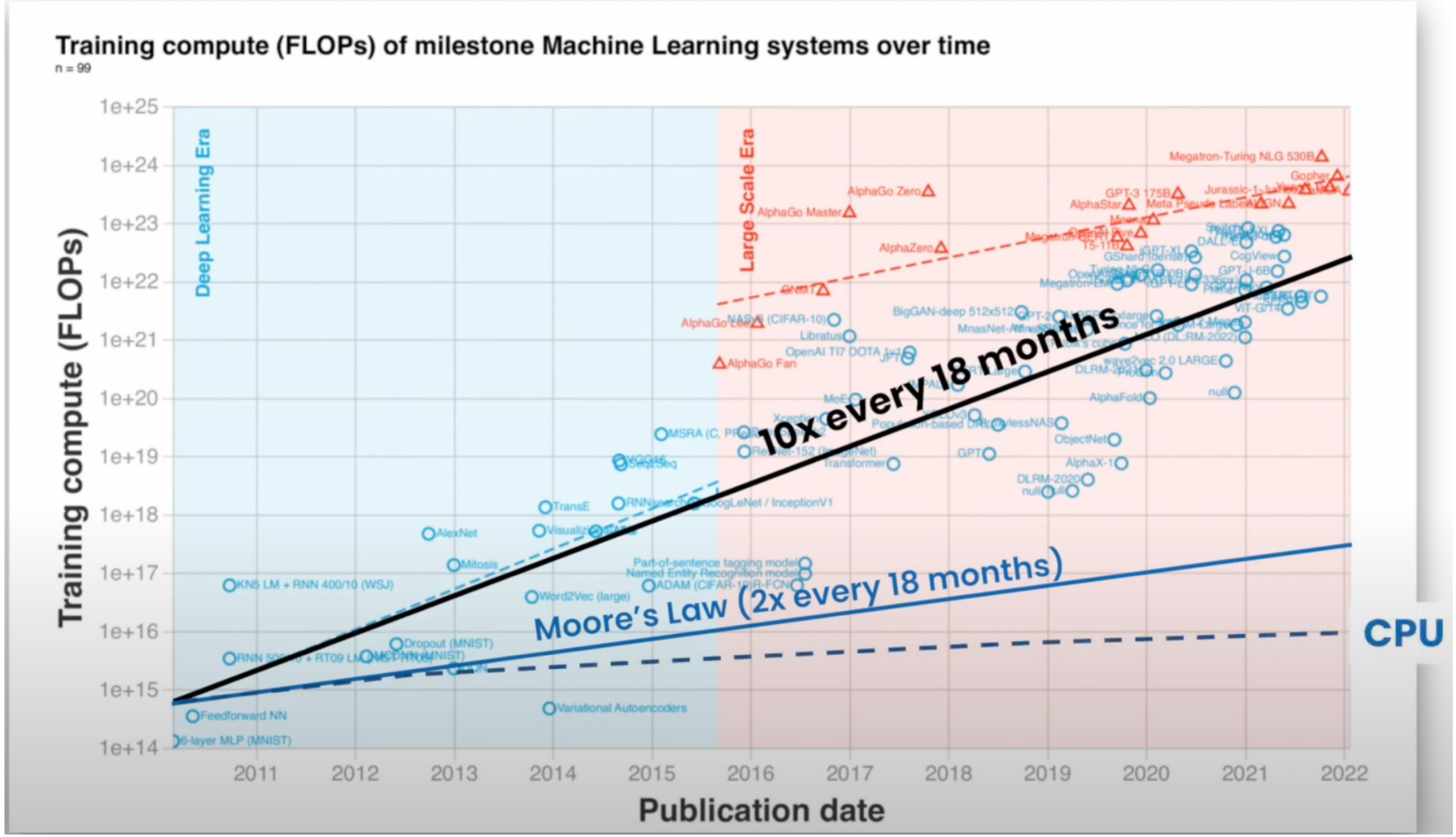
Language Models are Few Shot Learners, https://splab.sdu.edu.cn/GPT3.pdf



Why? Emergence of Foundation Models



Pathways Language Model (PaLM): Scaling to 540 Billion Parameters for Breakthrough Performance, https://research.google/blog/pathways-language-model-palm-scaling-to-540-billion-parameters-for-breakthrough-performance/



Compute Trends across Three Eras of Machine Learning, https://arxiv.org/pdf/2202.05924

So,  or  ?

LLM Models GPU Requirements for Pre-Training

Model	Developer	Parameters	GPU Requirements for Training
LLaMA 2	Meta AI	7B to 70B	7B: At least 24GB VRAM 65B: Multi-GPU setup, each with 160GB+ VRAM (e.g., 2x-4x NVIDIA A100 or H100)
Bloom	Hugging Face and collaborators	176B	Multi-GPU setup, each with at least 40GB VRAM (e.g., NVIDIA A100 or H100)
BERT	Google AI	110M to 340M	8GB to 16GB VRAM (e.g., NVIDIA RTX 3080 or 3090)
Falcon	Independent / Community-driven	Tens of billions	12GB to 24GB VRAM (e.g., RTX 4080 for smaller models, RTX 4090 or RTX 6000 Ada for larger variants)
Zephyr	Academic collaboration	~7B	16GB VRAM for basic tasks, 24GB+ VRAM recommended (e.g., NVIDIA RTX 4090)
Mistral 7B	OpenAI-inspired, community-driven	7B	At least 24GB VRAM (e.g., RTX 6000 Ada or A100)
Phi 2	Academic collaboration	Varies	12GB to 24GB VRAM for small to medium models (e.g., RTX 4080 or 4090) Larger models: RTX 6000 Ada or A100 recommended
MPT 7B	Tech companies and academic partners	7B	At least 16GB VRAM (e.g., RTX 4080 or 4090) RTX 6000 Ada for more demanding applications
Alpaca	OpenAI and partners	Varies (focus on smaller models)	As low as 8GB VRAM for basic use 16GB+ VRAM recommended for optimal performance (e.g., RTX 4080 or 4090)

Cloud GPU Resources for LLM Fine-tuning and Inference

Fine-tuning on Cloud GPUs

- Accessible GPU instances from cloud providers
- Cost-effective hourly rentals (e.g., Google Cloud T4 GPU: $0.35/hour)
- Suitable for fine-tuning smaller models like BERT

Inference on Cloud GPUs

- Lower computational requirements than training
- Flexible GPU options based on model size and performance needs
- On-demand usage for cost optimization

Practical Examples

- Fine-tuning BERT: $10-$50 per session on T4 GPU
- Inference with Mistral 7B: $1-$5 per hour on T4/V100 GPU

Benefits for Users

- Experimentation without long-term hardware investment
- Scalability for various project sizes
- Access to latest GPU models

ML Platforms

- Google Colab: Free GPU access with limitations
- Kaggle Kernels: Free GPU hours for competitions
- Hugging Face: Easy-to-use interfaces for model deployment

Cost Management

- Spot instances: Up to 90% cost reduction
- Preemptible VMs: Discounts for flexible workloads
- Off-peak scheduling for lower costs

Key Takeaway: Cloud GPU resources have democratized LLM work, allowing individuals and small teams to experiment with and deploy AI models efficiently and cost-effectively.

Quad PC vs. Spot Instances

Length of trial in experiment (hours)	**6**
Number of trials in experiment	16
Total GPU hours for experiment	96
Cost of 4x RTX 2080 Ti machine	$10,000.00
Time to run experiment on 4x machine	24 hours
Time to run experiment on V100 spot instances	6 hours
Cost of provisioning enough pre-emptible V100s	$96.00
Number of experiments that equal cost of 4x	104

How to think about it: cloud enables quicker experiments.

Amazon SageMaker · gradient by Paperspace · FLOYD · DOMINO DATA LAB

"All-in-one"

Data

- **Versioning**: Pachyderm, DVC, LIQUIDATA
- **Labeling**: figure eight, scale, Aquarium
- **Processing**: Apache Spark, DAGSTER
- **Exploration**: pandas, RAPIDS, dbt
- **Data Lake / Warehouse**: snowflake, databricks
- **Sources**: S3, LOG, Parquet

Training/Evaluation

- **Frameworks & Distributed Training**: fast.ai, PYTORCH, RAY, HOROVOD, K
- **Hyperparameter Tuning**: SIGOPT, Determined AI, Weights & Biases, tune
- **Experiment Management**: Weights & Biases, comet, TensorBoard, Neptune Machine Learning Lab, mlflow
- **Resource Management**: Determined AI, PY, slurm workload manager
- **Software Engineering**: jupyter, Streamlit, git
- **Compute**: CoreWeave, Lambda

Deployment

- **Feature Store**: tecton
- **Monitoring**: fiddler
- **Edge**: ML, NVIDIA TensorRT, TensorFlow Lite, ONNX
- **Web**: ALGORITHMIA, SELDON
- **CI / Testing**: Buildkite, great_expectations
- or · or

Resource Management

Function

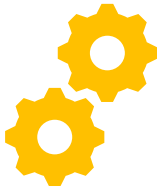
Multi-tenancy: multiple people share the resources.

Using multiple GPUs/machines

Running different environments

Goal

Easy to launch a batch of experiments, with proper dependencies and resource allocations



Solutions

Python Scripts

SLURM

Docker + Kubernetes

Software specialized for ML use cases

Scripts or slurm

- Problem we're solving: allocate free resources to programs
- Can be scripted pretty easily
- Even better, use old-school cluster job scheduler
  - Job defines necessary resources, gets queued

```python
class GPUManager(object):
    def __init__(self, verbose: bool=False):
        self.lock_manager = Redlock([{"host": "localhost", "port": 6379, "db": 0}, ])
        self.verbose = verbose

    def get_free_gpu(self):
        """
        If some GPUs are available, try reserving one by checking out an exclusive redis lock.
        If none available or can't get lock, sleep and check again.
        """
        while True:
            gpu_ind = self._get_free_gpu()
            if gpu_ind is not None:
                return gpu_ind
            if self.verbose:
                print(f'pid {os.getpid()} sleeping')
            time.sleep(GPU_LOCK_TIMEOUT / 1000)

    def _get_free_gpu(self):
        available_gpu_inds = [
            gpu.index
            for gpu in gpustat.GPUStatCollection.new_query()
            if not gpu.processes
        ]
        if available_gpu_inds:
            gpu_ind = np.random.choice(available_gpu_inds)
            if self.verbose:
                print(f'pid {os.getpid()} picking gpu {gpu_ind}')
            if self.lock_manager.lock(f'gpu_{gpu_ind}', GPU_LOCK_TIMEOUT):
                return int(gpu_ind)
            if self.verbose:
                print(f'pid {os.getpid()} couldnt get lock')
        return None
```

train.sh

```
./train.py --gpu=-1 --fc_size=128
./train.py --gpu=-1 --fc_size=256
./train.py --gpu=-1 --fc_size=512
./train.py --gpu=-1 --fc_size=1024
./train.py --gpu=-1 --fc_size=2048
./train.py --gpu=-1 --fc_size=4096
./train.py --gpu=-1 --fc_size=8192
```

```
sergeyk@thefarm:~$ parallel -j8 :::: train.sh
```

Scripts or slurm

workload manager

- Problem we're solving: allocate free resources to programs
- Can be scripted pretty easily
- Even better, use old-school cluster job scheduler
  - Job defines necessary resources, gets queued

```
class GPUManager(object):
    def __init__(self, verbose: bool=False):
        self.lock_manager = Redlock([{"host": "localhost", "port": 6379, "db": 0}, ])
        self.verbose = verbose
```

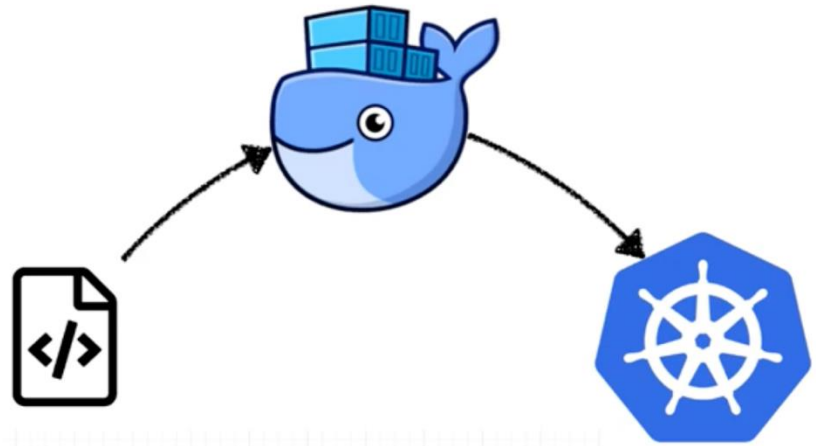
Job-file (schedule job with sbatch, check status with squeue -u <Username>):

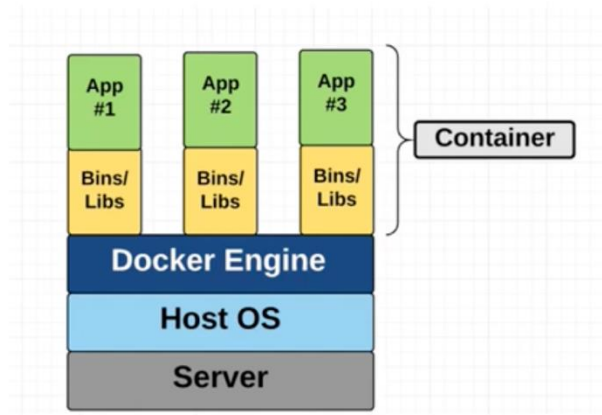
```
#!/bin/bash
#SBATCH --gres=gpu:1
#SBATCH --mem=10000
#SBATCH -p gpu2    # K80 GPUs on Haswell node
#SBATCH --time=01:00:00

## with Theano (using configs from above)
module purge # purge if you already have modules loaded
module load modenv/eb
module load Keras

srun python mnist_cnn.py


## with Tensorflow
module purge
module load modenv/eb
module load Keras
module load tensorflow
# if you see 'broken pipe error's (might happen in interactive session af
module load h5py/2.6.0-intel-2016.03-GCC-5.3-Python-3.5.2-HDF5-1.8.17-ser

export KERAS_BACKEND=tensorflow      # configure Keras to use tensorflow

srun python mnist_cnn.py
```

```
./train.py --gpu=-1 --fc_size=8192
```

```
sergeyk@thefarm:~$ parallel -j8 :::: train.sh
```

Docker + Kubernetes

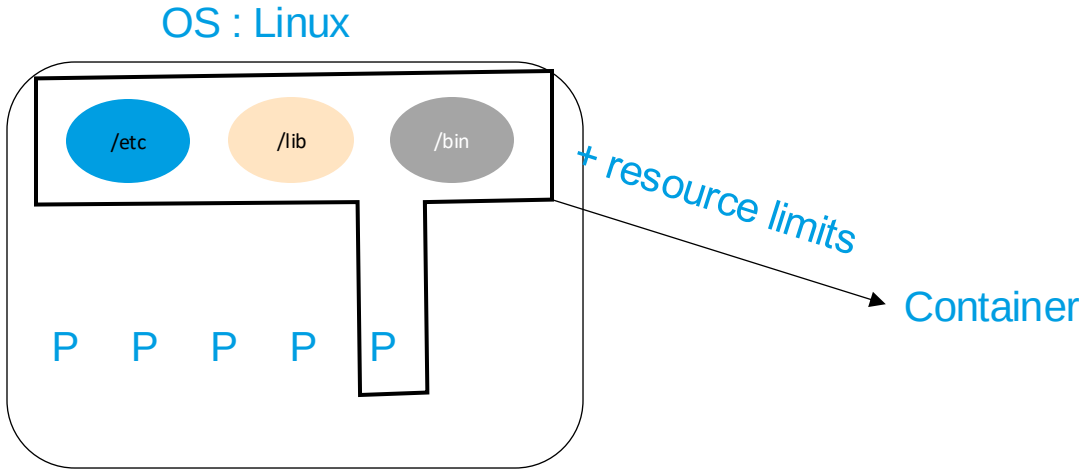


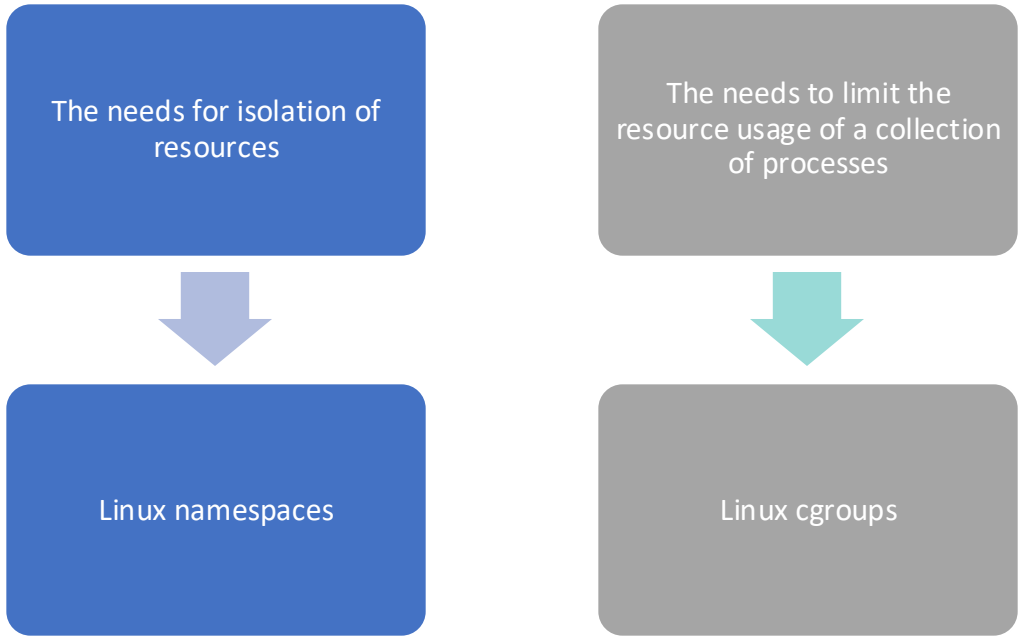


- Kubernetes is a way to run many docker containers on top of a cluster

What are Containers?

- Containers are an abstraction at the app layer that
  - package code and dependencies together.
  - each running as isolated processes in user space.
  - Multiple containers can run on the same machine and share the OS kernel with other containers,
  - Containers are basically leveraging linux namespaces and cgroups for isolation of processes execution and resource limitations respectively

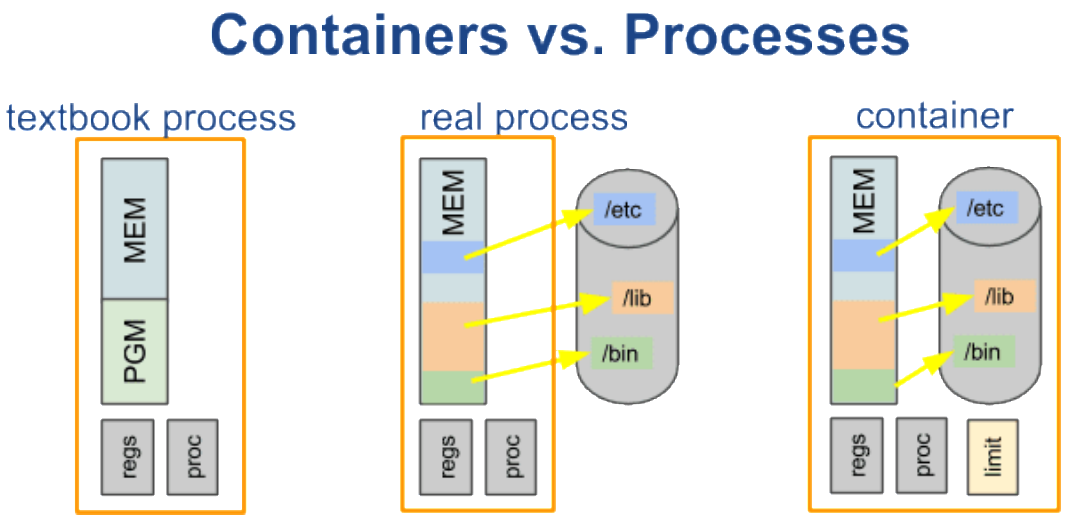




Tutorial: https://www.youtube.com/watch?v=EnJ7qX9fkcU
https://jvns.ca/blog/2016/10/10/what-even-is-a-container/

Containers vs. Processes

- Textbook process
  - own address space
  - own program
  - own CPU state
  - own process table entry
- Real process:
  - memory mapped from the filesystem into the process address space
  - consists of dozens of shared libraries, programs and files that are shared.
  - super-dependent on its filesystem environment (e.g. locale info, shared libs and files)
- Container
  - an encapsulation of one program with all its dependencies.
  - with limits on the amount of resources the program can use.



Docker

- Docker is a tool designed to make it easier to *create*, *deploy*, and *run* applications by using containers.
  - **Docker image**: A Docker image is a file, comprised of multiple layers, used to execute code in a Docker container.
  - **Dockerfile**: The Dockerfile is essentially a set of build instructions to build the image. A Dockerfile is a text document used to indicate to Docker a base image, the Docker settings you need, and a list of commands you would like to have executed to prepare and start your new container.
- Docker can associate a seccomp profile with the container using the **--security-opt** parameter.

Web Application

Layer 9 : CMD *Start Pintail.ai website*
Layer 8 : RUN *Commands to set up Pintail.ai Website*
Layer 7 : ADD *Pintail.ai binaries*
Layer 6 : FROM *Node.js - Apline Linux*

Application Framework

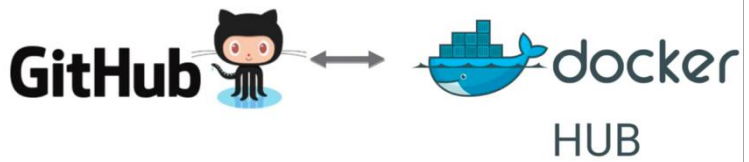
Layer 5 : RUN *Commands to set up Node.js*
Layer 4 : ADD *Node.js binaries*
Layer 3 : FROM *Alpine Linux*

Operating System

Layer 2 : RUN *Commands to set up OS*
Layer 1 : ADD *Operating System binaries*
Layer 0 : FROM *Scratch*

Docker image repositories:

- Private: IBM Kubernetes Registry
- Public: Docker Hub



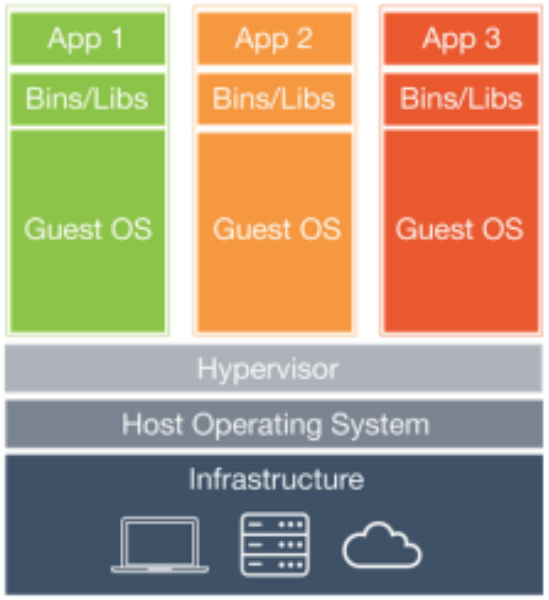
Container vs Virtual Machine

- Virtual machines
  - have a full OS
  - own memory management
  - own device drivers
  - own daemons
  - own binaries, libraries, and applications

- Containers
  - Share the host's OS kernel
  - Share the binaries and libraries i read-only mode.

How is this different from virtual machines?

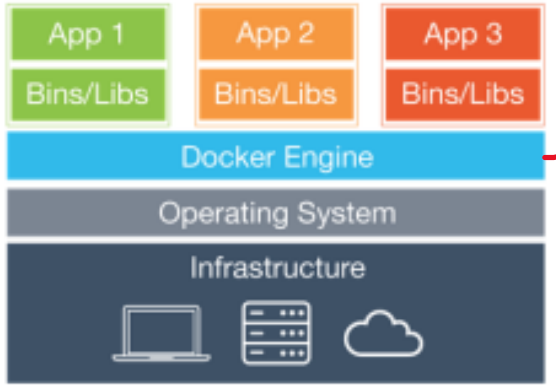
Containers have similar resource isolation and allocation benefits as virtual machines but a different architectural approach allows them to be much more portable and efficient.



Virtual Machines

Each virtual machines includes the application, the necessary binaries and libraries and an entire guest operating system - all of which may be tens of GBs in size.

software that hosts the containers



Containers

Containers include the application and all of its dependencies, but share the kernel with other containers. They run as an isolated process in userspace on the host operating system. They're also not tied to any specific infrastructure – Docker containers run on any computer, on any infrastructure and in any cloud.

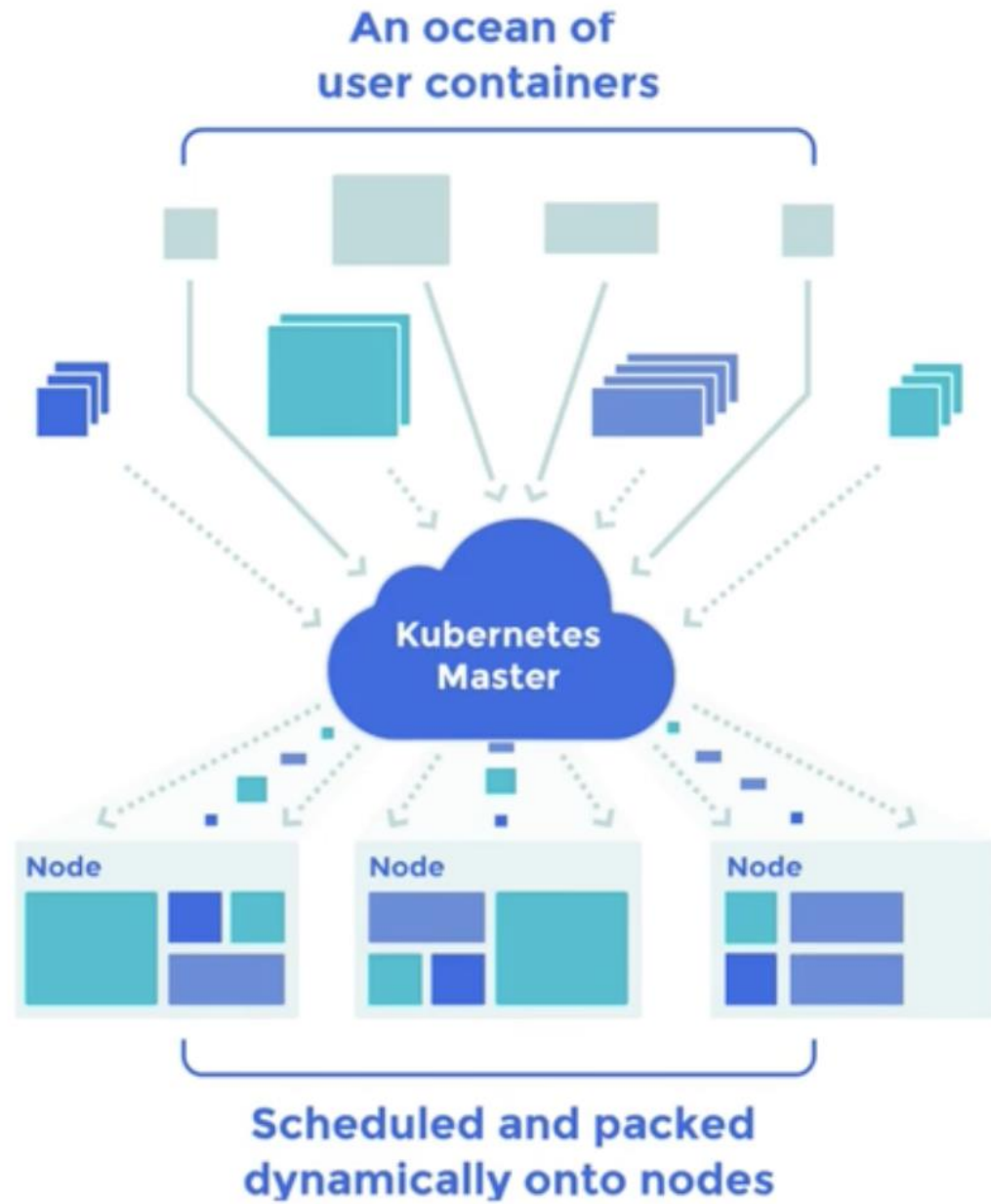
	CONTAINER BENEFITS	VIRTUAL MACHINE BENEFITS
Consistent Runtime Environment	✓	✓
Application Sandboxing	✓	✓
Small Size on Disk	✓	
Low Overhead	✓	

Benefits of using Container

- Containerization allows our development teams to
  - move fast
    - Decoupling from environment
    - Compared to VMs, containers are far more lightweight.
  - deploy software efficiently
    - Consistent environment
    - Run Anywhere
    - Isolation
    - Easy version control
    - Agile development
  - and operate at an unprecedented scale
    - Kubernetes: Production-Grade Container Orchestration

Docker + Kubernetes

- Kubernetes is a way to run many docker containers on top of a cluster



What is Kubernetes (K8s)?

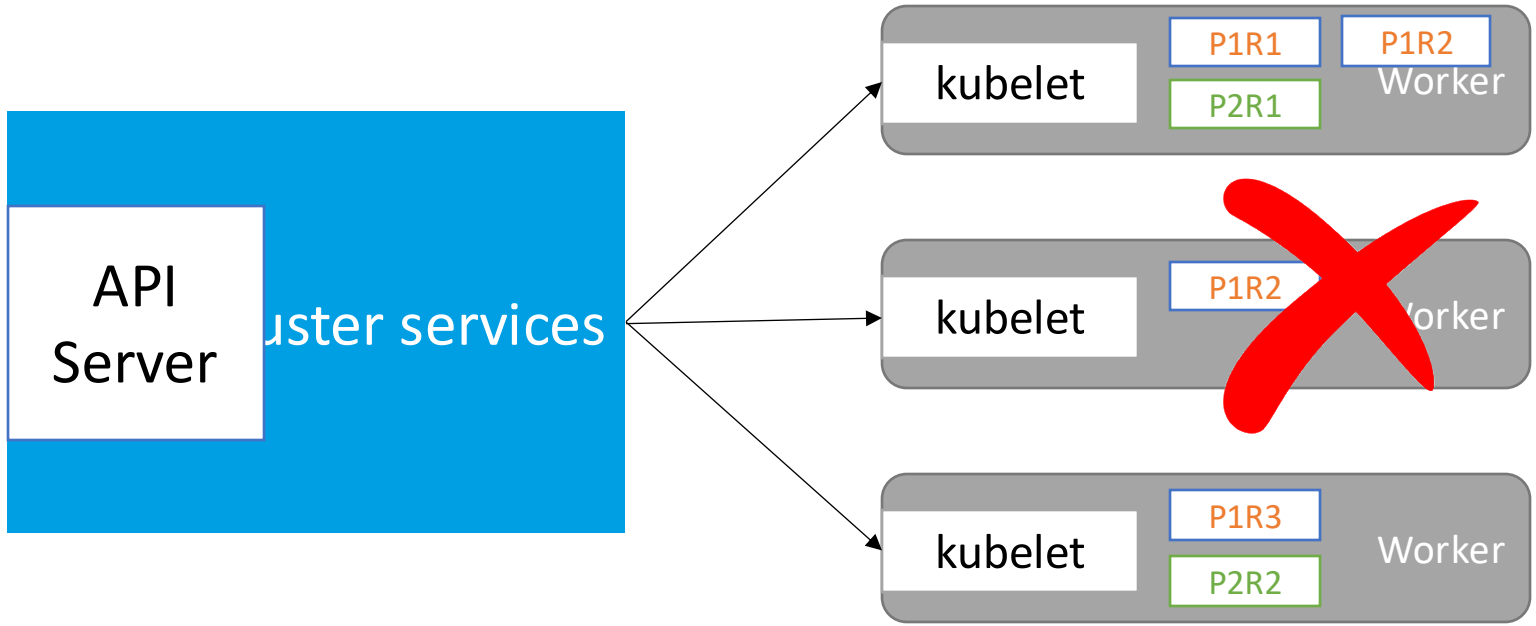
- Kubernetes is an *open source system* for managing containerized applications across *multiple hosts*, providing basic mechanisms for *deployment*, *maintenance*, and *scaling* of applications.

Kubernetes cluster services → "Desired State Management"

App1.yaml

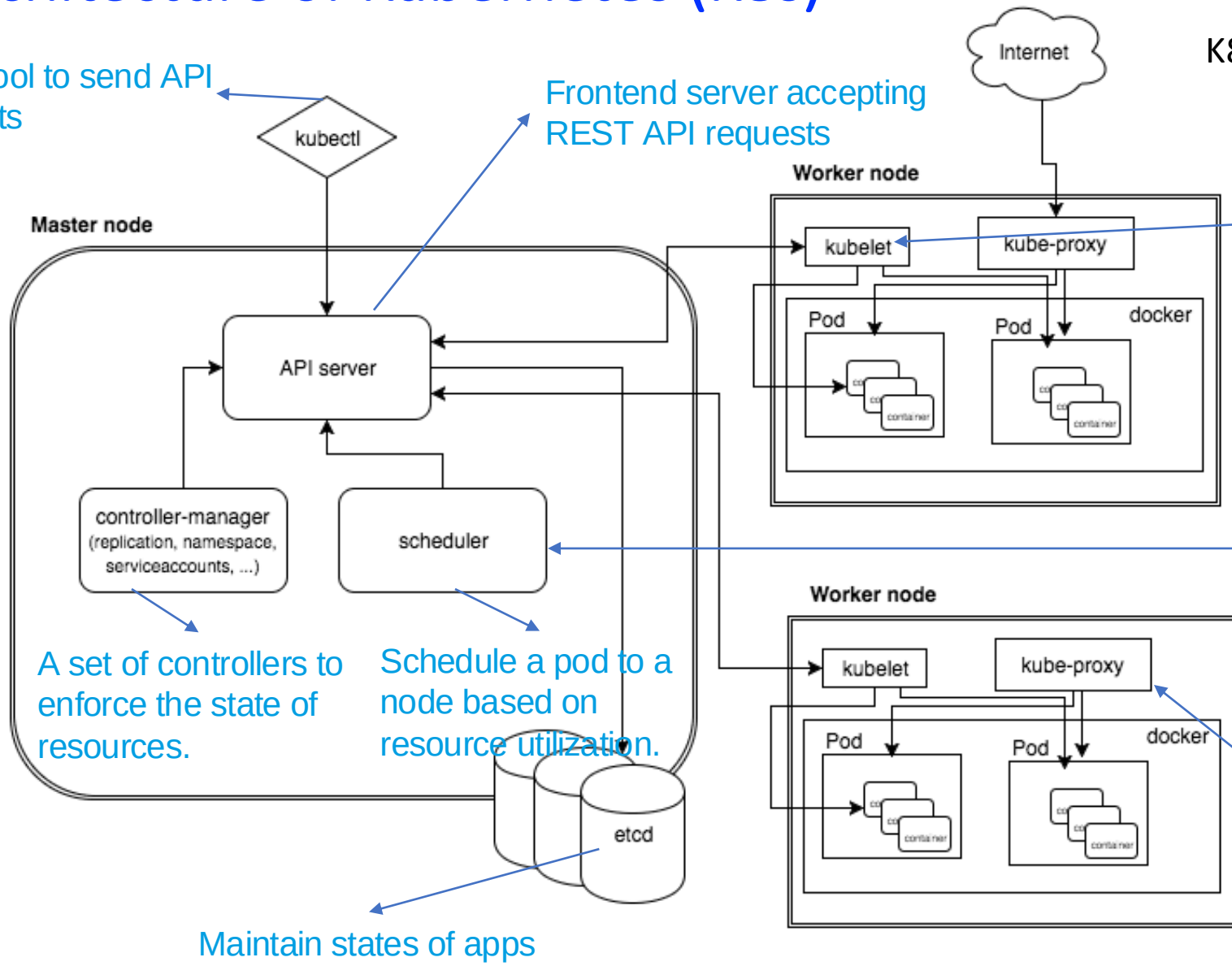
Deployment
Pod 1
- container img 1
- container img 2
- Replica -> 3

Pod 2
- container img 3
- replicas = 2



https://kubernetes.io/

Architecture of Kubernetes (K8s)



K8s follows a client-server architecture

- Ensure pods heathy and running
- Reports to the master to report the health of the host.

Scheduler assigns a node to the pod by
(1) filtering the nodes that satisfy the pod resource requirements and other predicate constraints,
(2) ranking the candidate nodes based on priority functions, and
(3) selecting the node with the highest rank.

- Deal with individual host subnetting
- Expose services to the external world.
- Request forwarding across isolated networks in a cluster.

Cloud Services running in Container Cloud

- Internal Customers of IBM Cloud Kubernetes Service: *Launched on May 23, 2017*
  - IBM Watson (Conversation, NLC, sentiment analysis, etc.)
  - IBM Deep Learning as a Service (DLaaS)
- Google: *Launched in 2015*
  - Almost everything from YouTube to Gmail.
  - Google Kubernetes Engine: https://cloud.google.com/kubernetes-engine/
- Alibaba:
  - 90% of their main business, especially e-commerce like Taobao and Tmall
  - Alibaba Cloud Container Service for Kubernetes: https://www.alibabacloud.com/product/kubernetes
- AWS: *Nov. 29, 2017*
  - Amazon Elastic Container Service for Kubernetes: https://aws.amazon.com/eks/
- Azure: *Oct. 27, 2017*
  - Azure Kubernetes service: https://azure.microsoft.com/en-us/services/kubernetes-service/

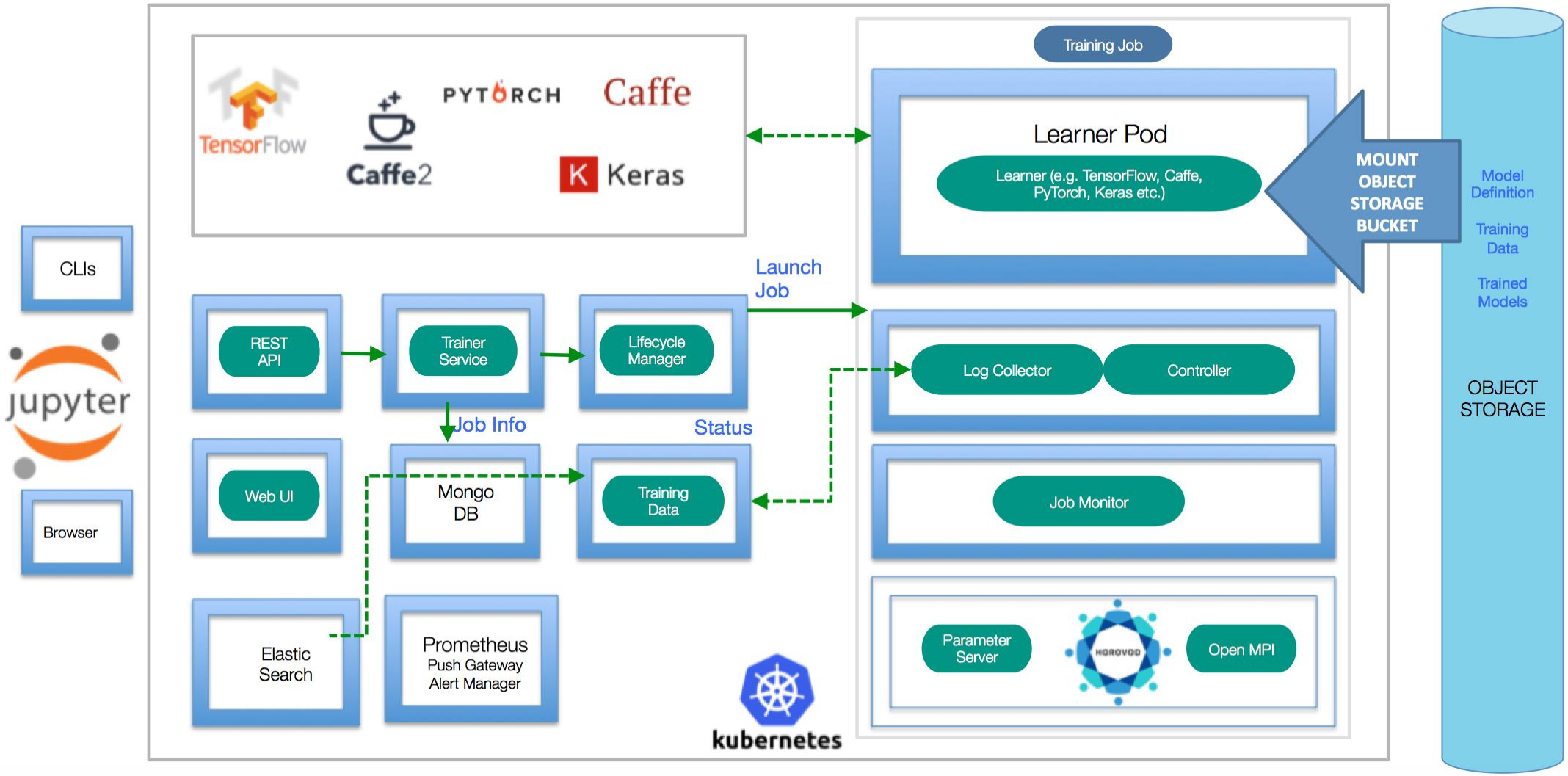
Platform specialized for DL training workloads

- “Why do we need a platform specialized for DL training workloads”?
- Can I build my own platform over K8s ?
- You need to understand low level details of K8s
- You need to keep track of job life-cycle including deployment, monitoring, termination of associated pods using K8s APIs.

Is K8s Enough for Data scientists and ML Developers ?

- Too “low-level” for use by data scientists
  - Training job vs pods, deployment, replica sets …
- DL training workloads need specialized scheduling algorithms for better performance, not natively supported by K8s
- Cannot track DL specific job status: DOWNLOADING, PROCESSING, STORING, HALTED, RESUMED

Typical Life Cycle of ML on Cloud



Fabric for Deep Learning (FfDL), https://github.com/IBM/FfDL

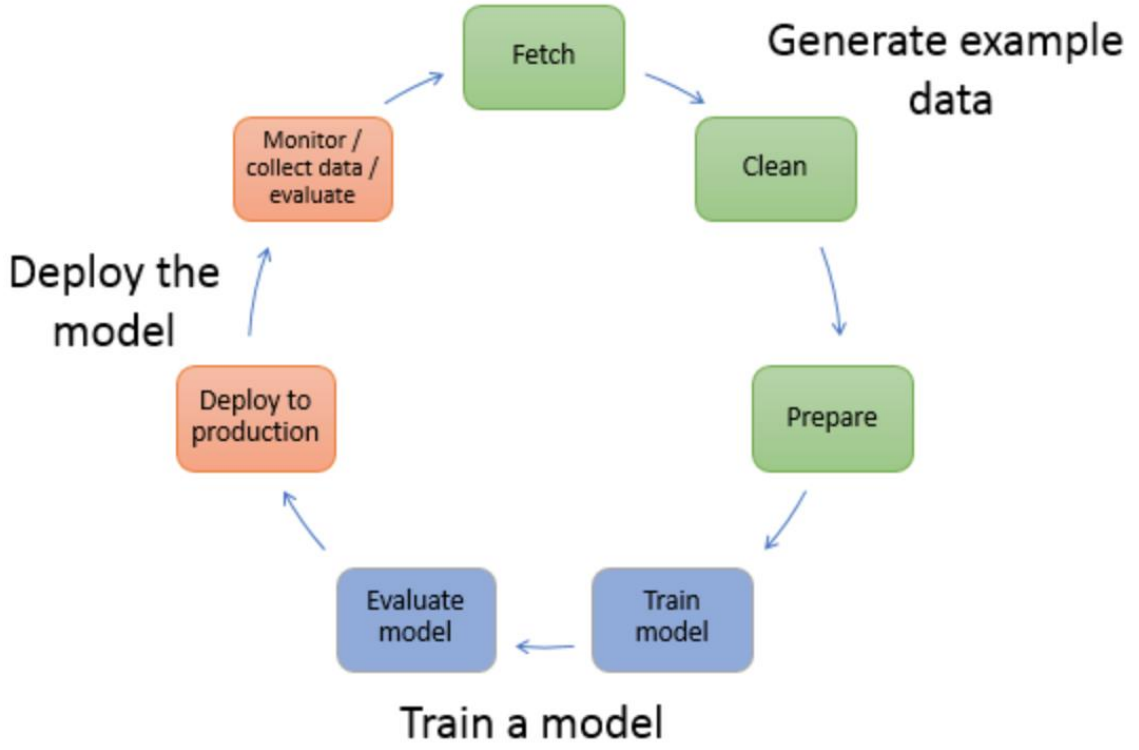
Cloud based Machine Learning Services

- IBM Watson Studio: https://www.ibm.com/products/watson-studio
- Amazon Sagemaker: https://aws.amazon.com/sagemaker
- Microsoft Azure Machine Learning : https://azure.com/ml
- Google Vertex AI Platform: https://cloud.google.com/vertex-ai/

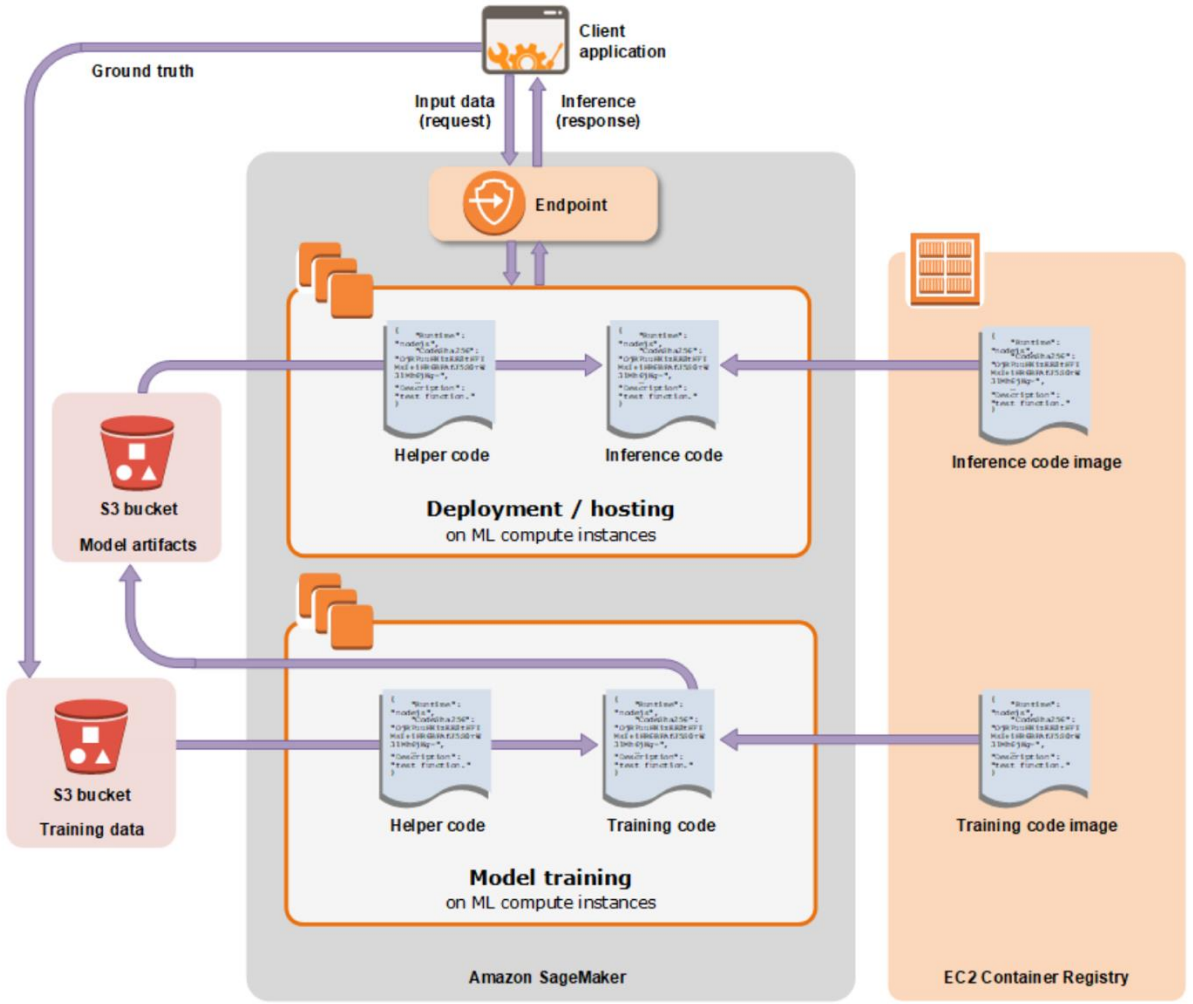
Amazon Sagemaker

- Fully managed machine learning service by Amazon
- Supports:
  - Quick and easy building and training of ML models
  - Model deployment in production-ready hosted environment

Typical ML Workflow



Train with Amazon Sagemaker



Training algorithm options

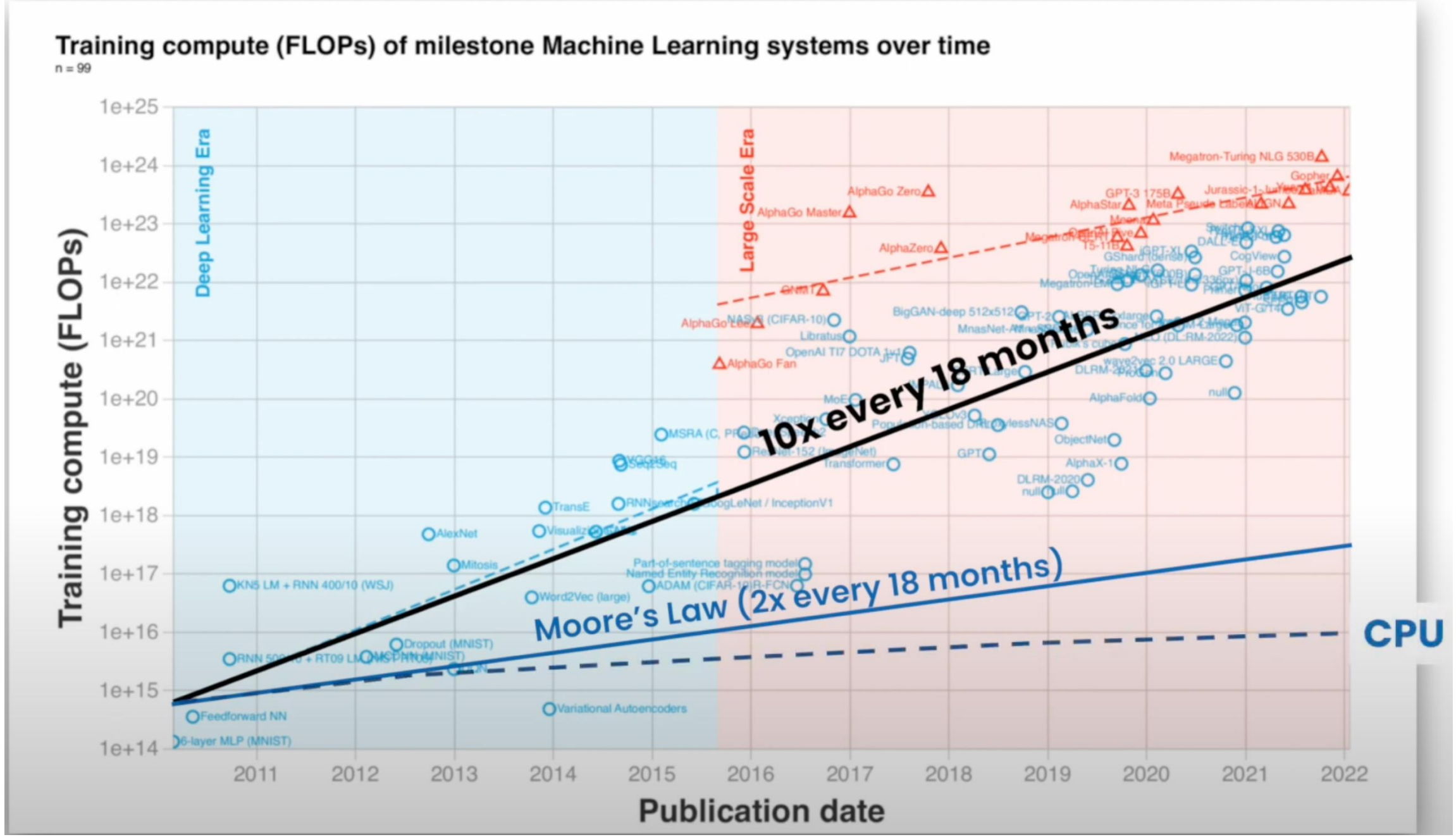
- Use an out-of-the-box algorithm provided by Amazon
  - Pre-built docker images
- Use Apache Spark MLLib with Amazon Sagemaker
- Custom python code to train with DL frameworks
  - Tensorflow and Apache MXNet
- Use your own custom algorithm in any programming language and framework
  - Package as a docker image and register it
- Use an algorithm from AWS Marketplace

https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Ray: A framework for Scaling AI workloads

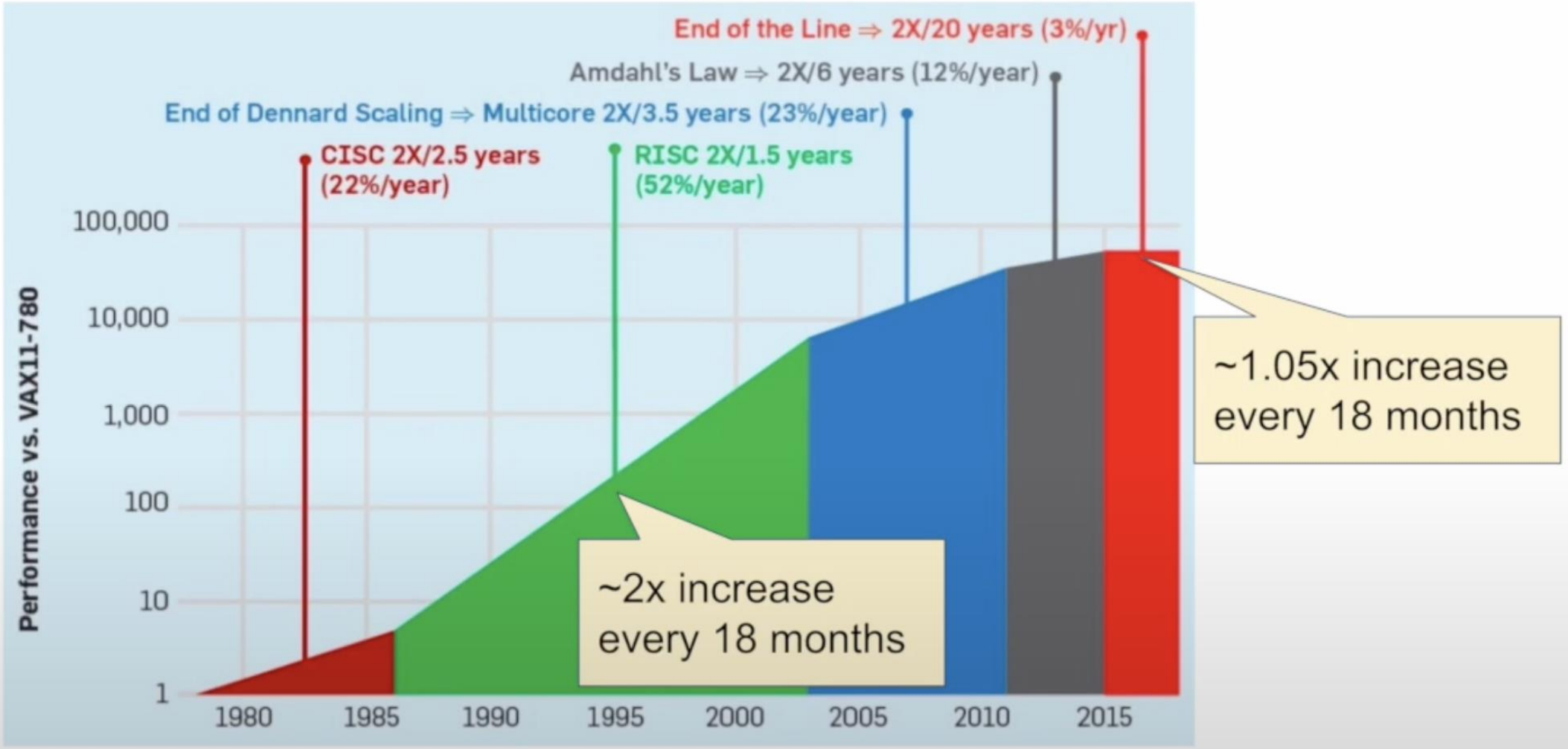
AI demands grow much faster …

… than a single node/processor capabilities

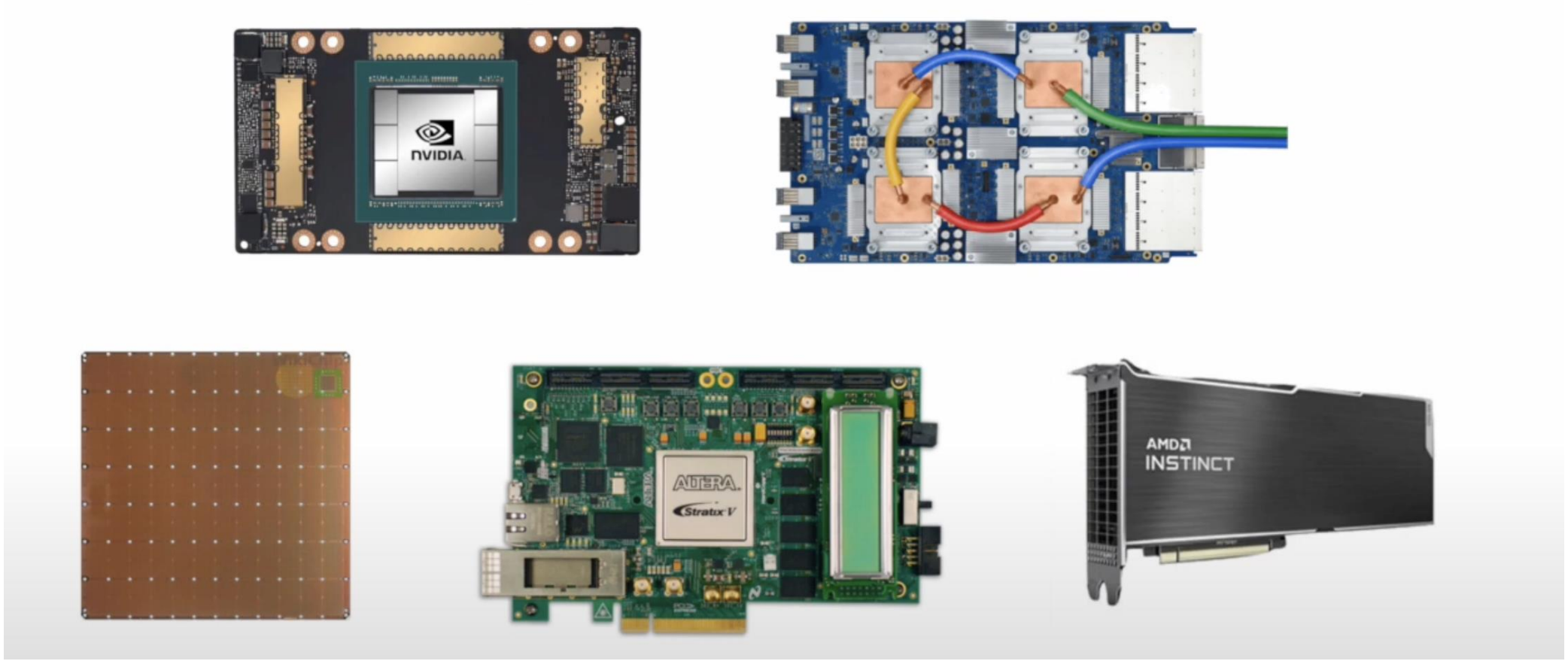


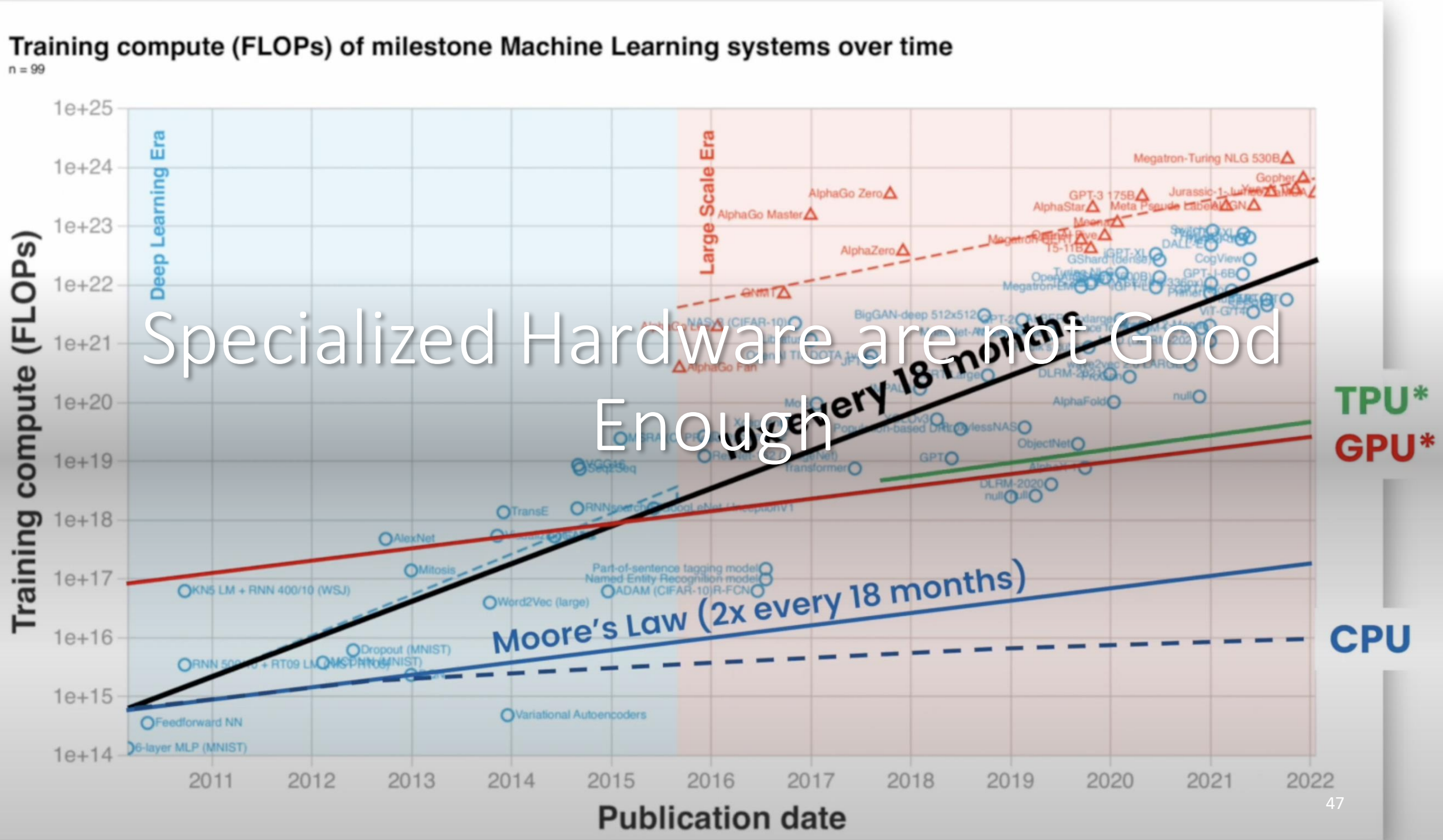
Compute Trends across Three Eras of Machine Learning, https://arxiv.org/pdf/2202.05924

Moore's Law coming to an end



What about specialized hardware?





Specialized Hardware are not Good Enough

Not so fast...

- The Scaling Law still holds
- But can no longer increase model size 10x/year (not enough GPUs!)
- Expected to grow just 2-3x/year

Article • May 28, 2023

OpenAI's plans according to Sam Altman

Raza Habib

https://website-754fwhahs-humanloopml.vercel.app/blog/open_ai_talk

Even if model sizes would stop growing …

… it would take decades for specialized hardware to catch up!

Example:

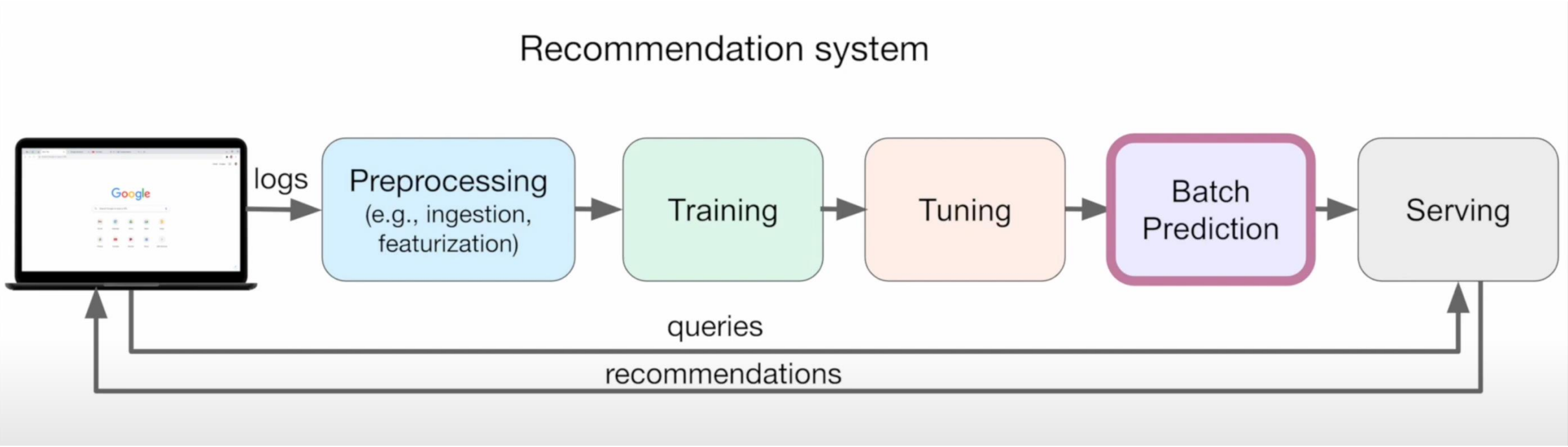
Google's PaLM takes 6144 TPU v4 to train

Assuming doubling performance every 18 months, it would take

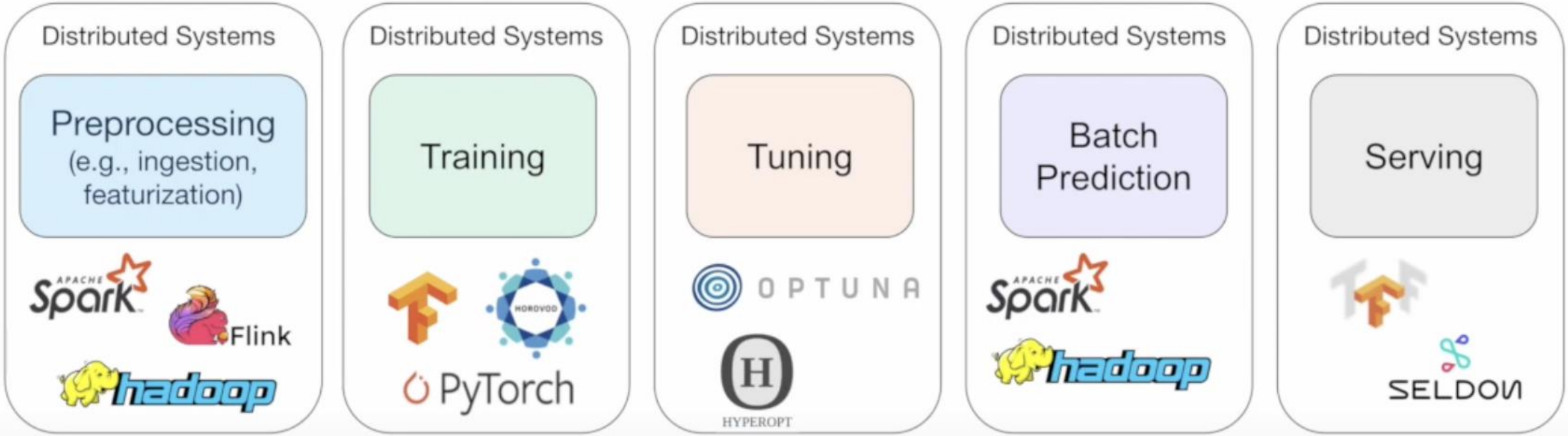
~19 years to train on a single chip

No way but to
distribute AI workloads
and this will remain **true** for
foreseeable future!

The anatomy of an end-to-end ML application



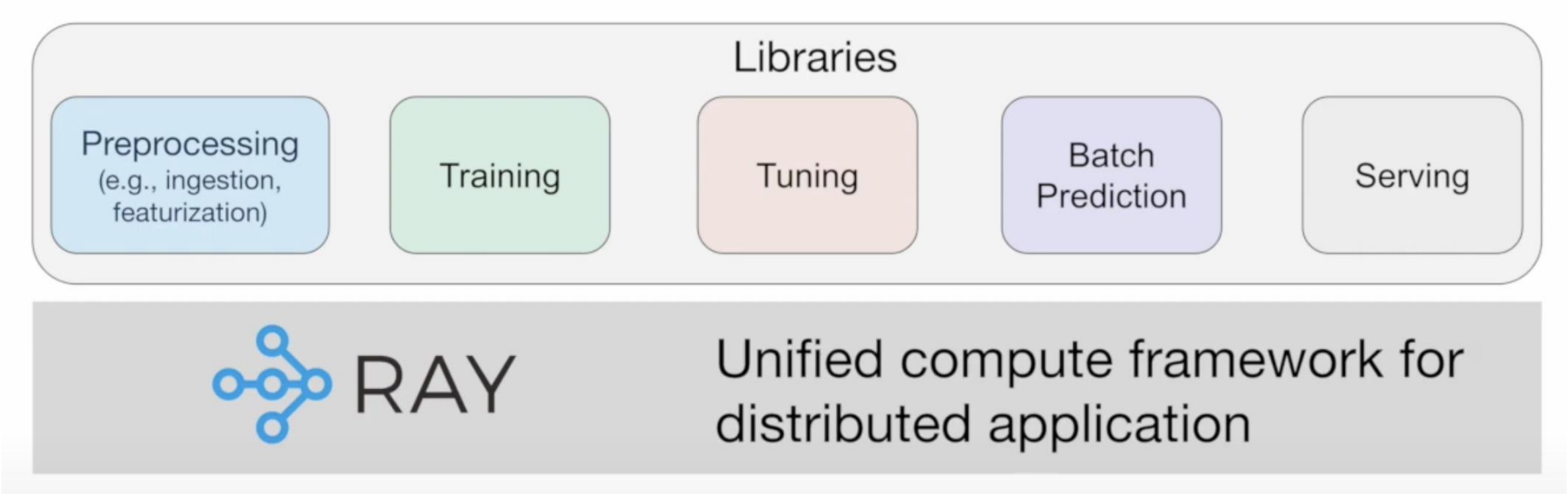
Challenge: need to scale every stage!



Need to stitch together a bunch of disparate systems

- **Hard** to develop
- **Hard** to deploy
- **Hard** to manage
- Slow

Ray is designed to support all these workloads



Ray: a short history

2016: Started as a class project

Berkeley
UNIVERSITY OF CALIFORNIA

- Initial goal: scale distributed deep neural network training and reinforcement learning

2017: RLlib and Ray Tune released

2019: Anyscale founded (company behind Ray)

2020: Ray v1.0 release; Ray Serve released

2022: Ray v2.0; AI Runtime (AIR) alpha released

Minimalist API

ray.init()	Initialize Ray context.
@ray.remote	Function or class decorator specifying that the function will be executed as a task or the class as an actor in a different process.
.remote	Postfix to every remote function, remote class declaration, or invocation of a remote class method. Remote operations are *asynchronous*.
ray.put()	Store object in object store, and return its ID. This ID can be used to pass object as an argument to any remote function or method call. This is a *synchronous* operation.
ray.get()	Return an object or list of objects from the object ID or list of object IDs. This is a *synchronous* (i.e., blocking) operation.
ray.wait()	From a list of object IDs returns (1) the list of IDs of the objects that are ready, and (2) the list of IDs of the objects that are not ready yet.

The **FAST** Compute Model

Futures: reference to objects (possible not created yet)

Actors: remote class instance (object)

Shared in-memory distributed object store

Tasks: remote functions

Python example

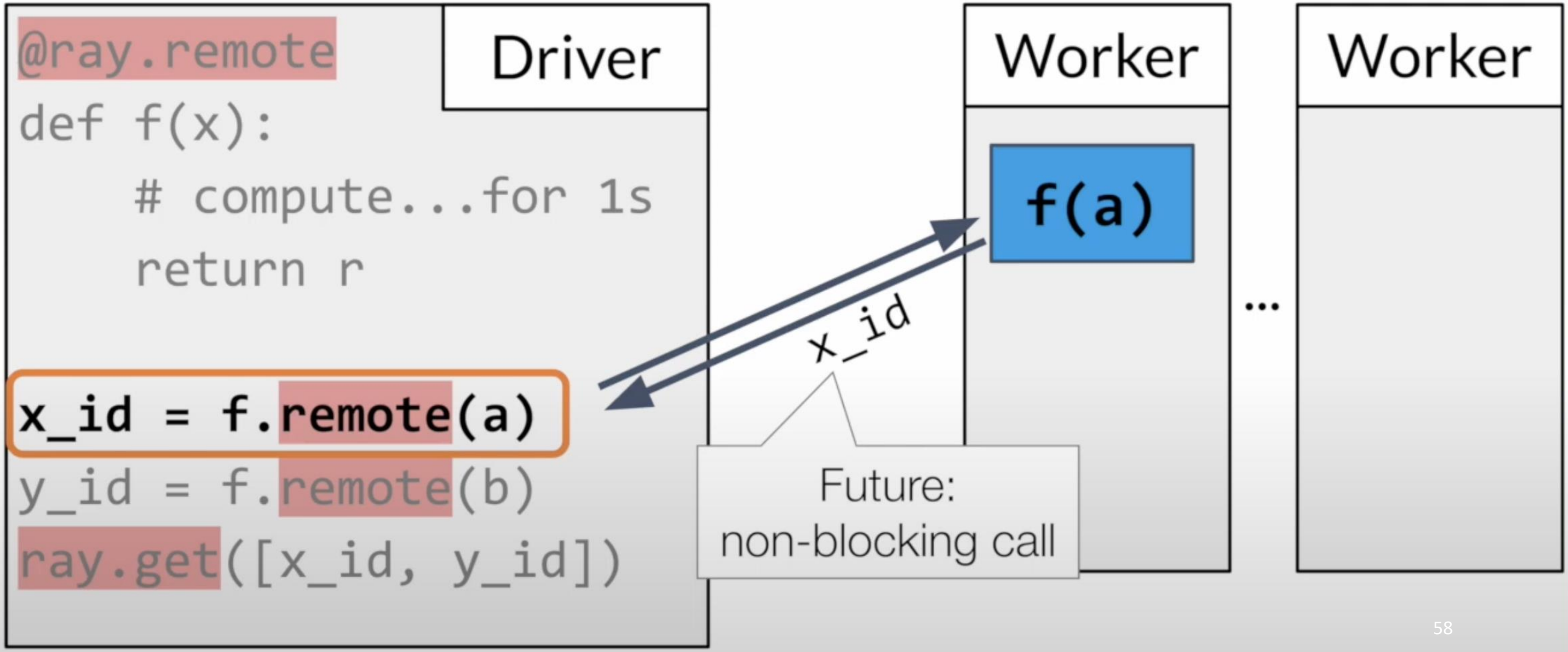
```
def f(x):
    # compute...for 1s
    return r

x = f(a)
y = f(b)
```

1 second

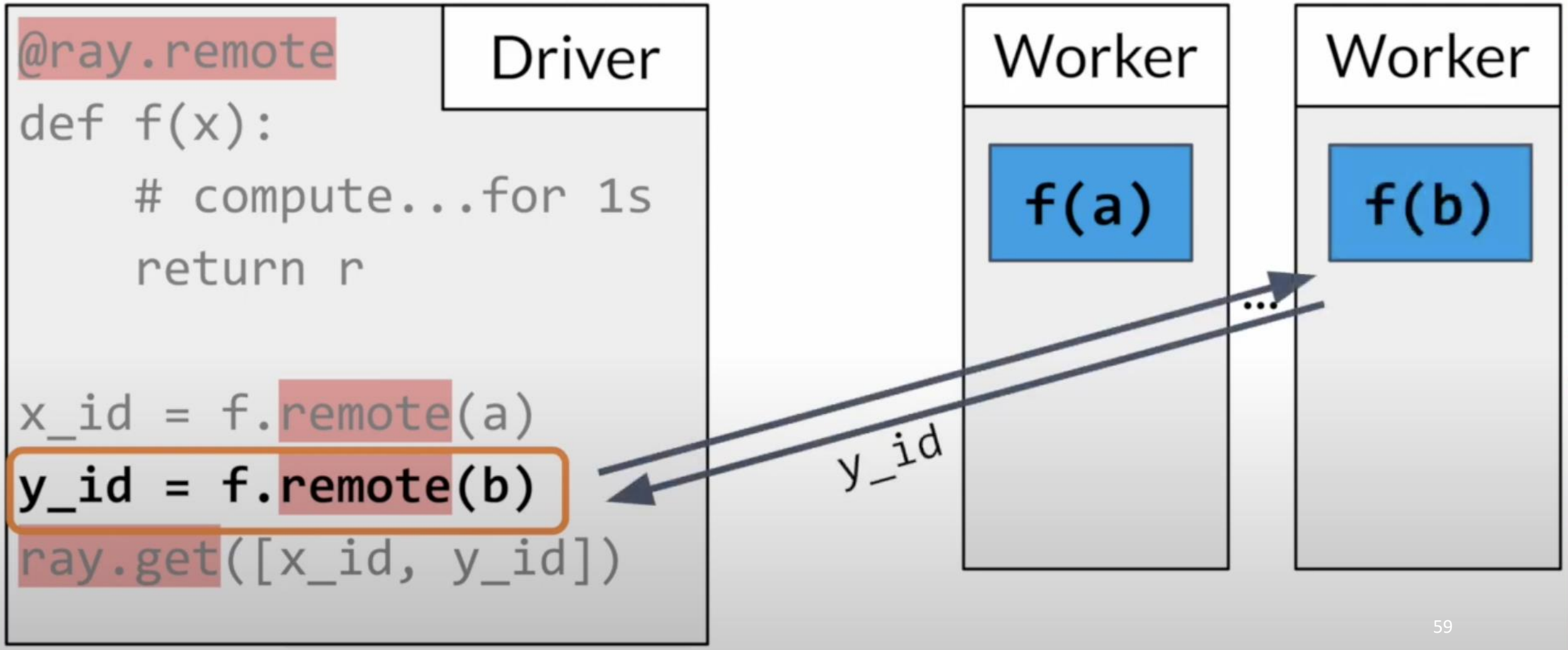
1 second

2 seconds

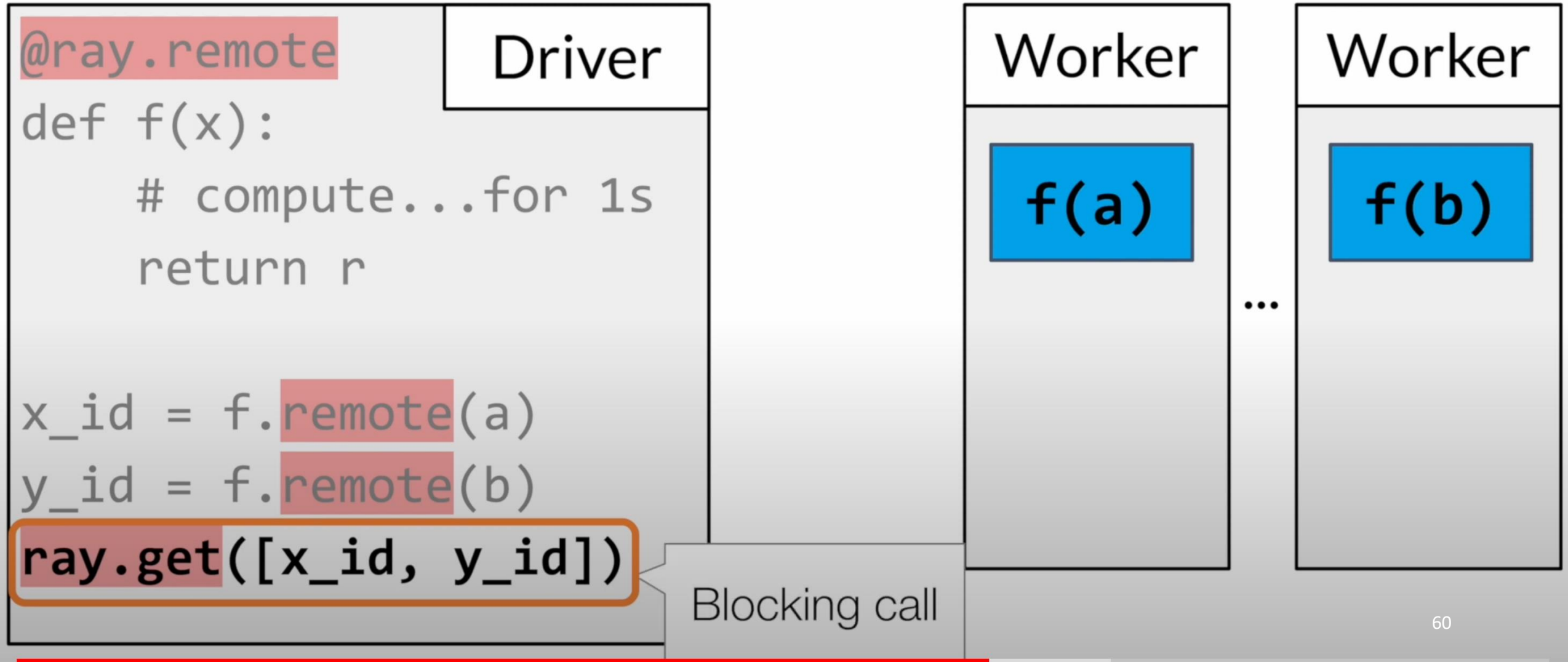
Ray: Function → Task



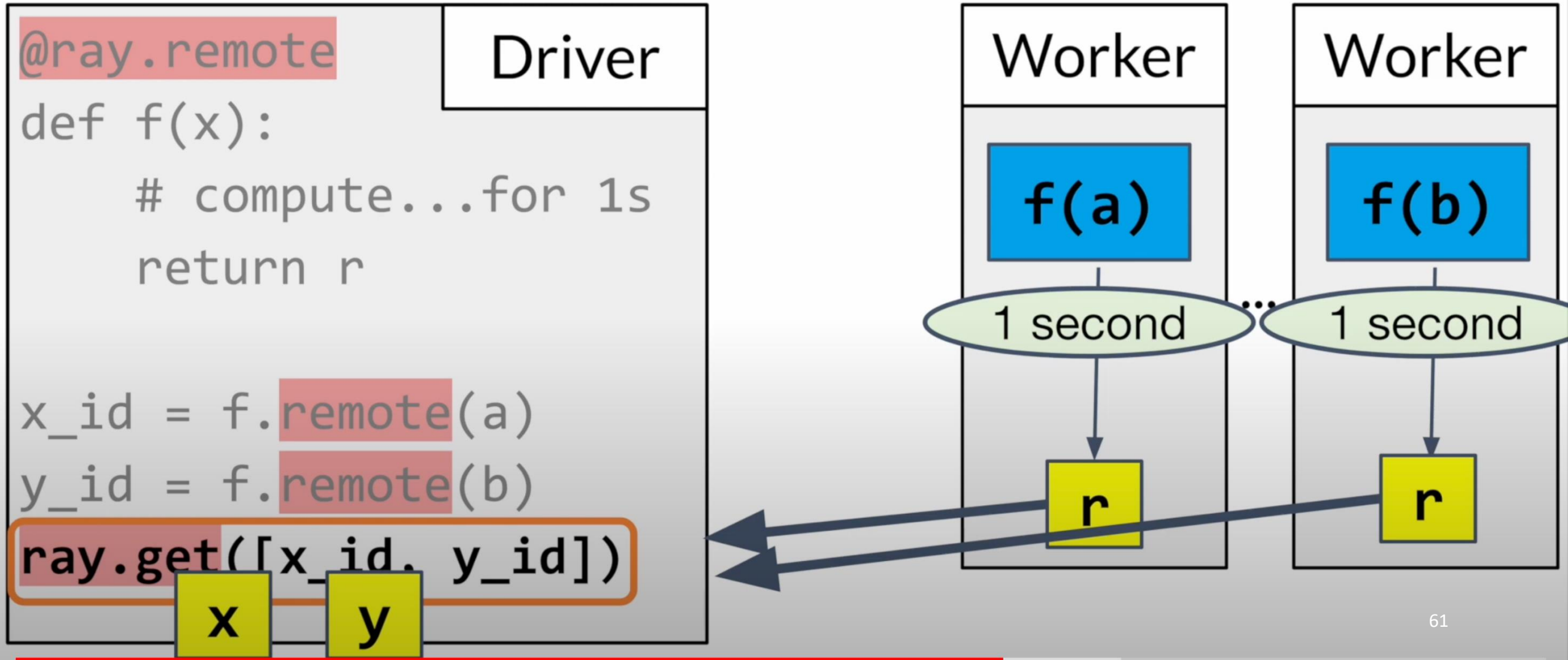
Ray: Function → Task



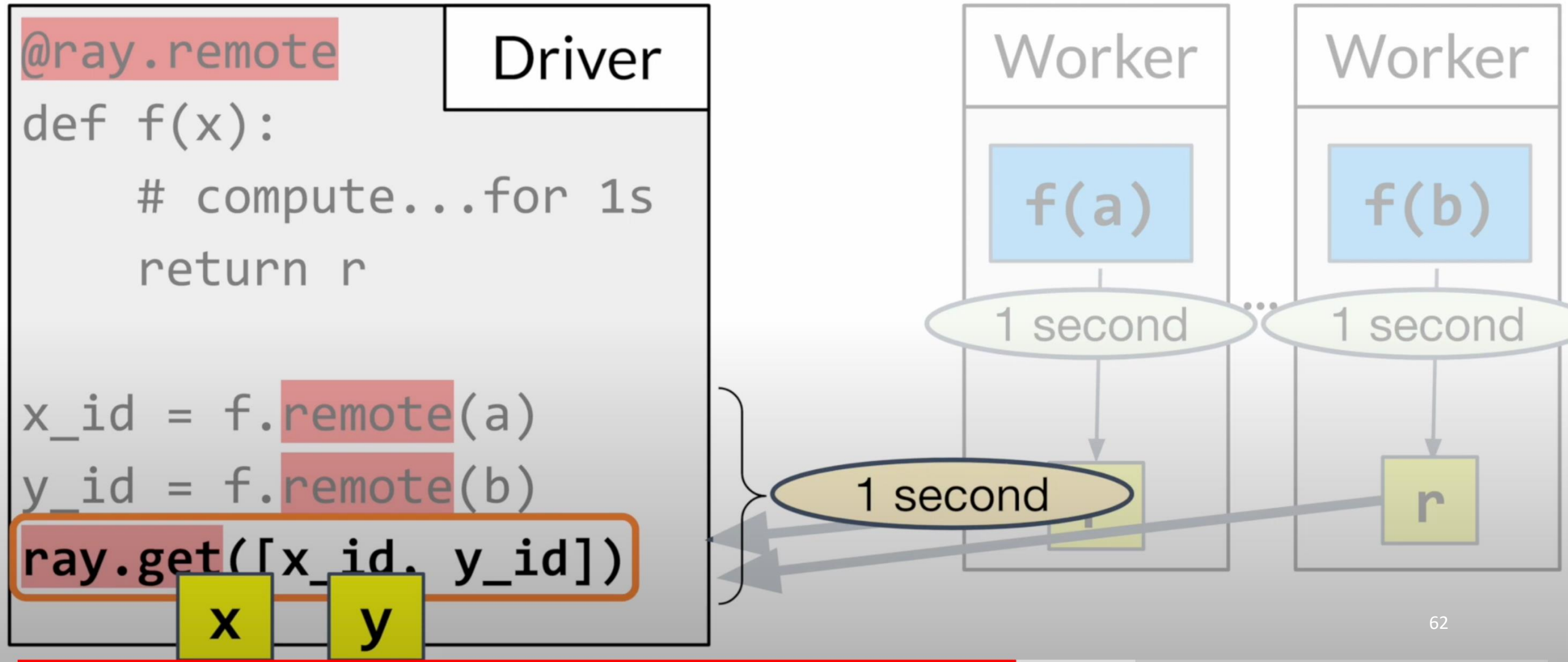
Ray: Function → Task



Ray: Function → Task



Ray: Function → Task



Python 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()
```

Ray: Class → Actor

```
@ray.remote
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()
```

Python Class

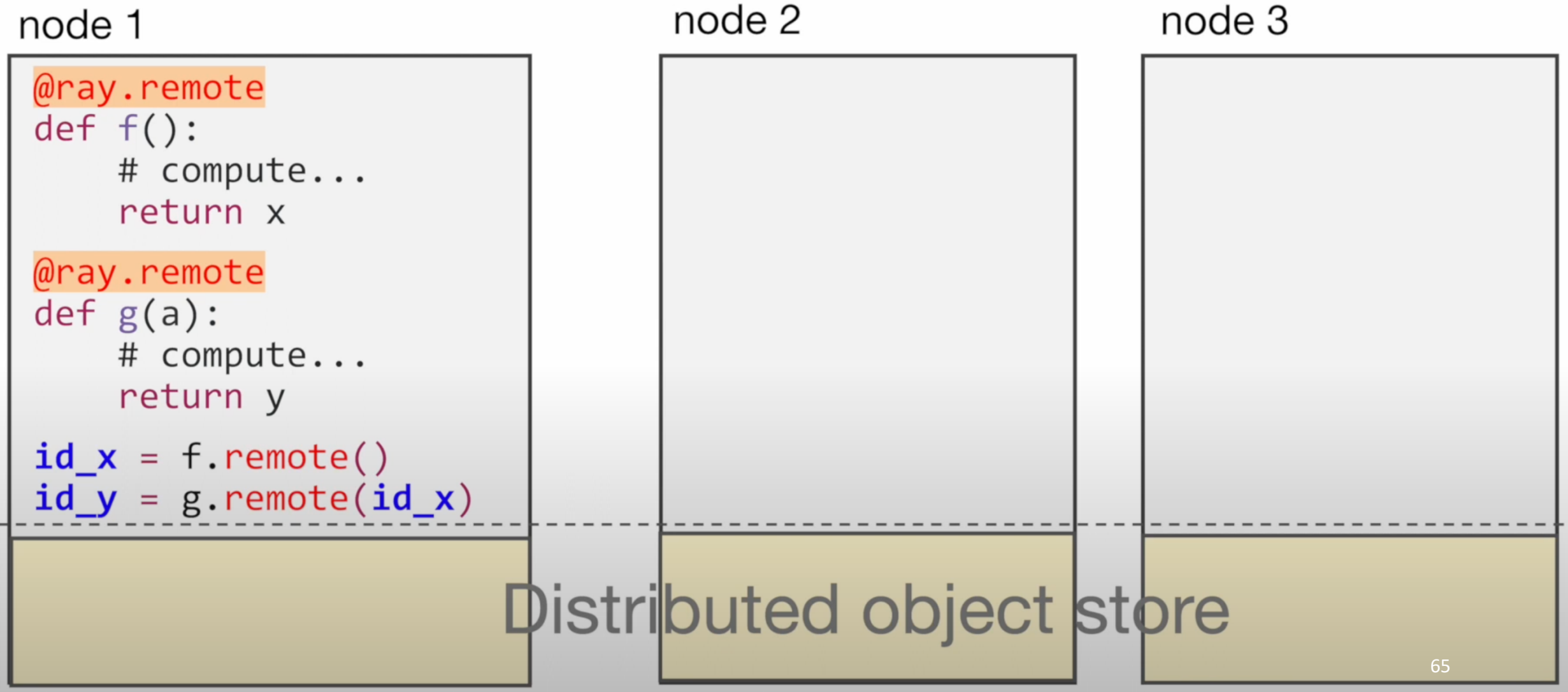
```
class Counter(object):
    def __init__(self):
        self.value = 0
    def inc(self):
        self.value += 1
        return self.value

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

Ray: Class → Actor

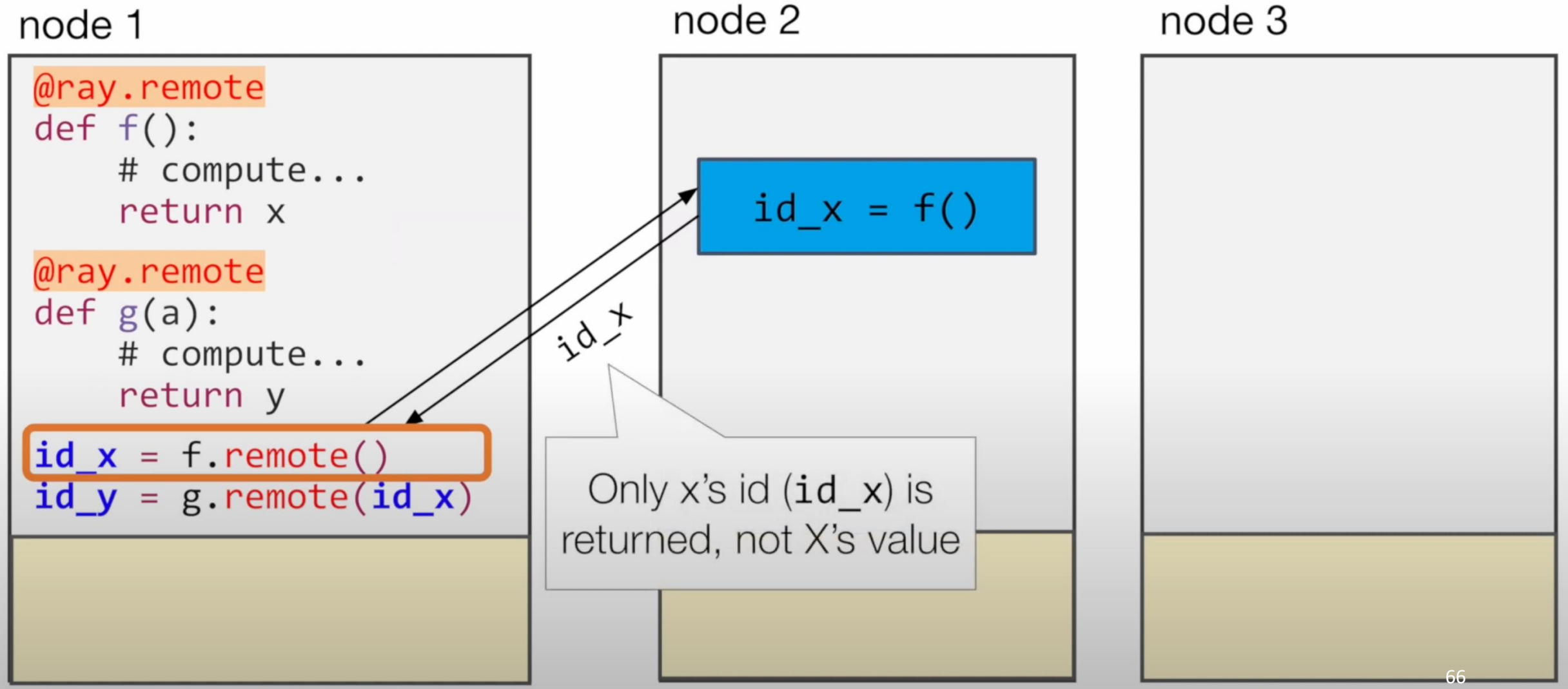
```
@ray.remote(num_cpus=2, num_gpus=1)
class Counter(object):
    def inc(self):
        self.value += 1
        return self.value

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

can specify resource demands;
support heterogeneous hardware

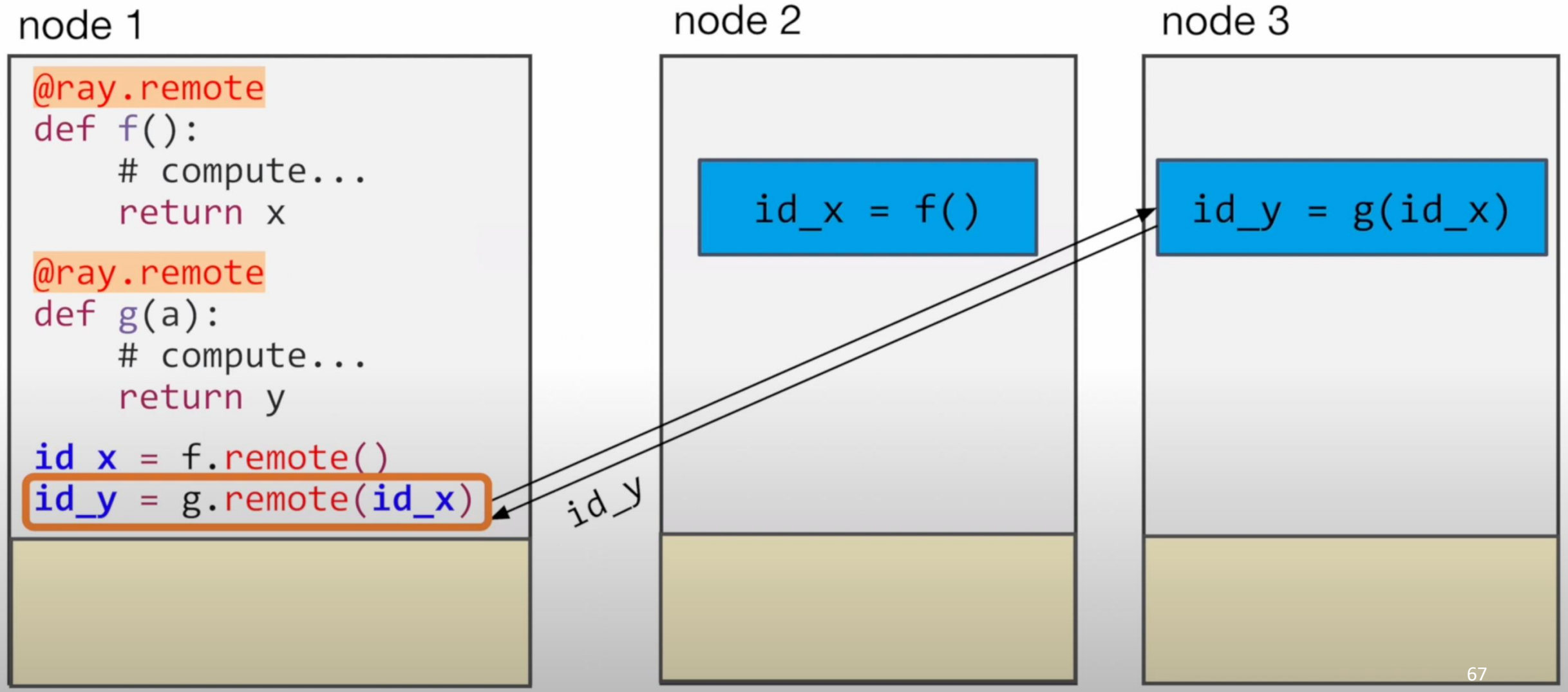
Shared in-memory object store



Shard in-memory object store



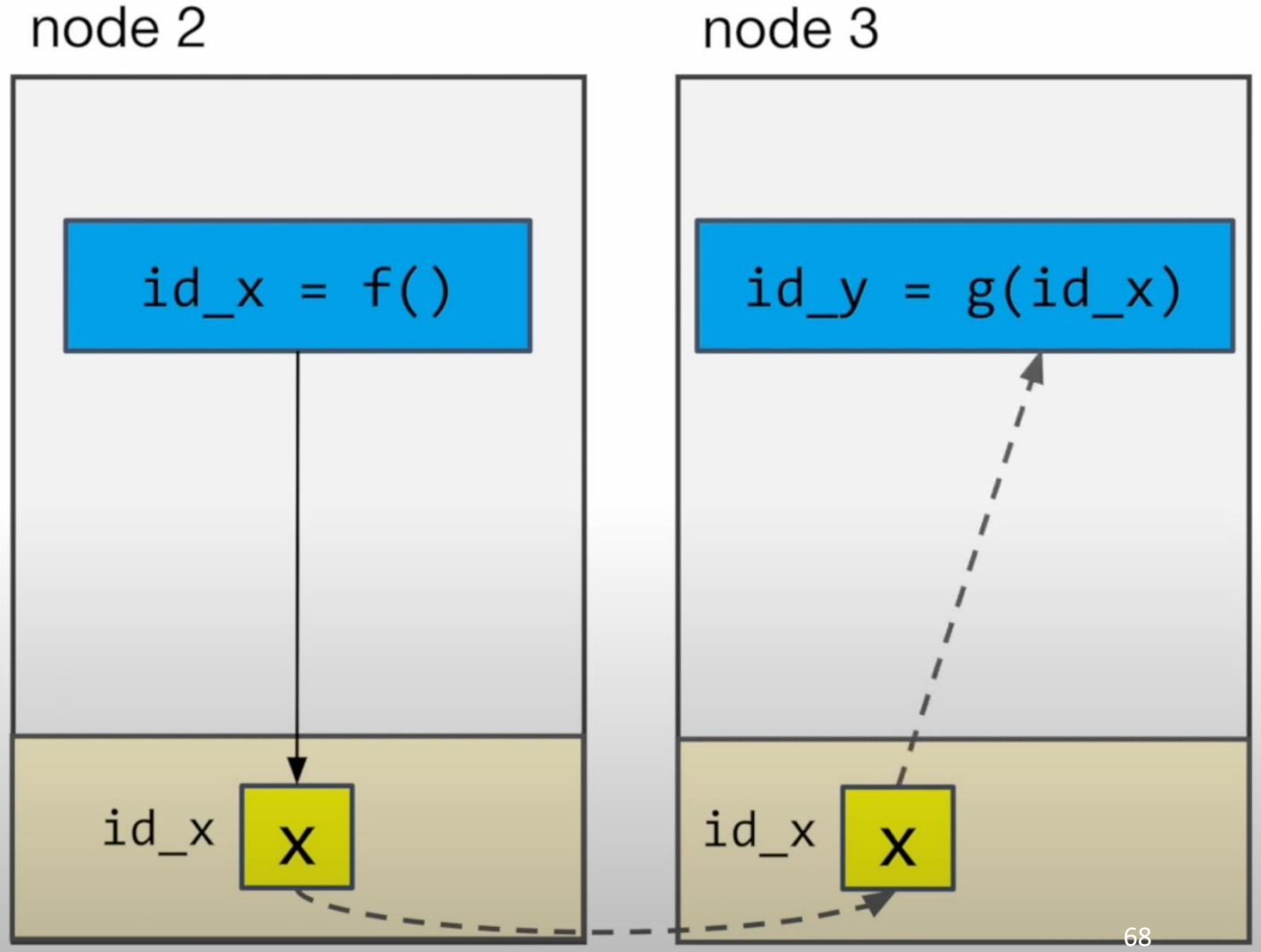
Shard in-memory object store



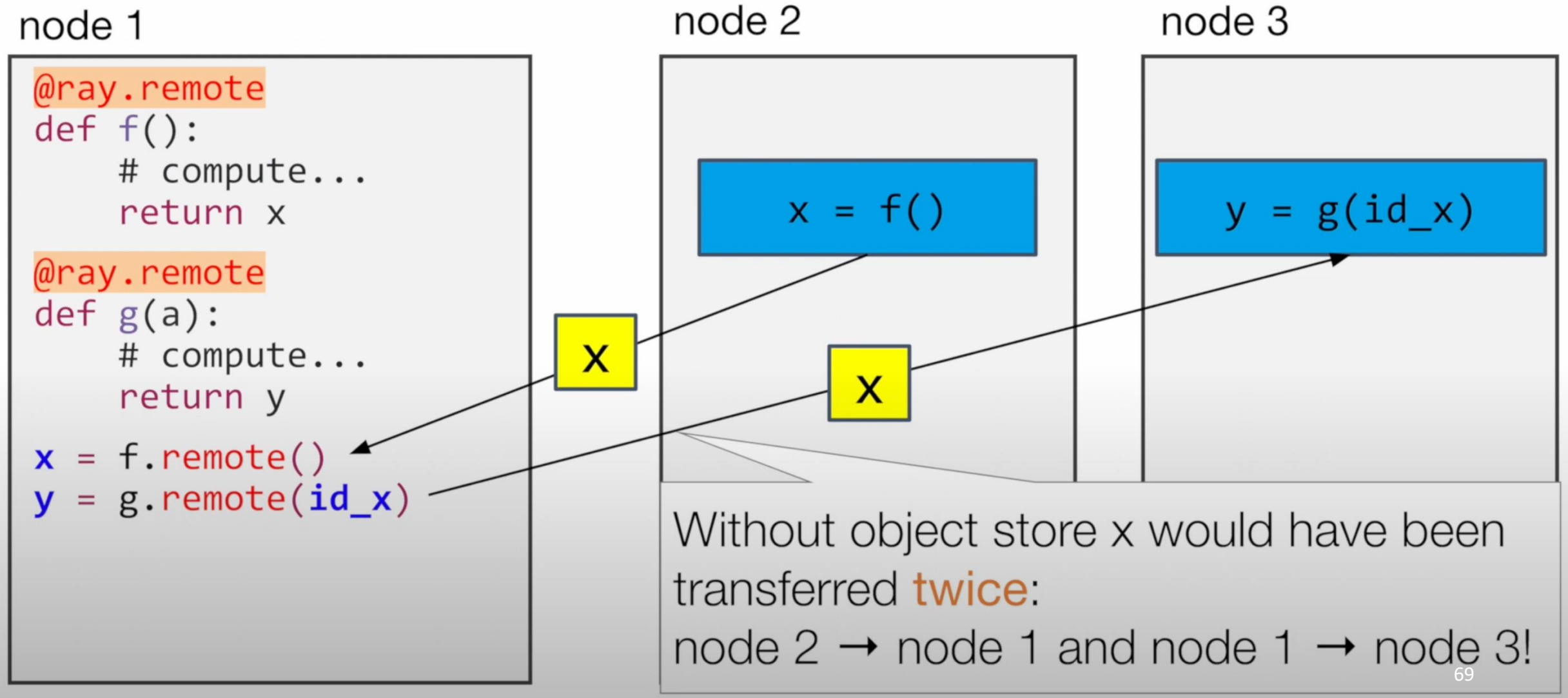
Shard in-memory object store

node 1

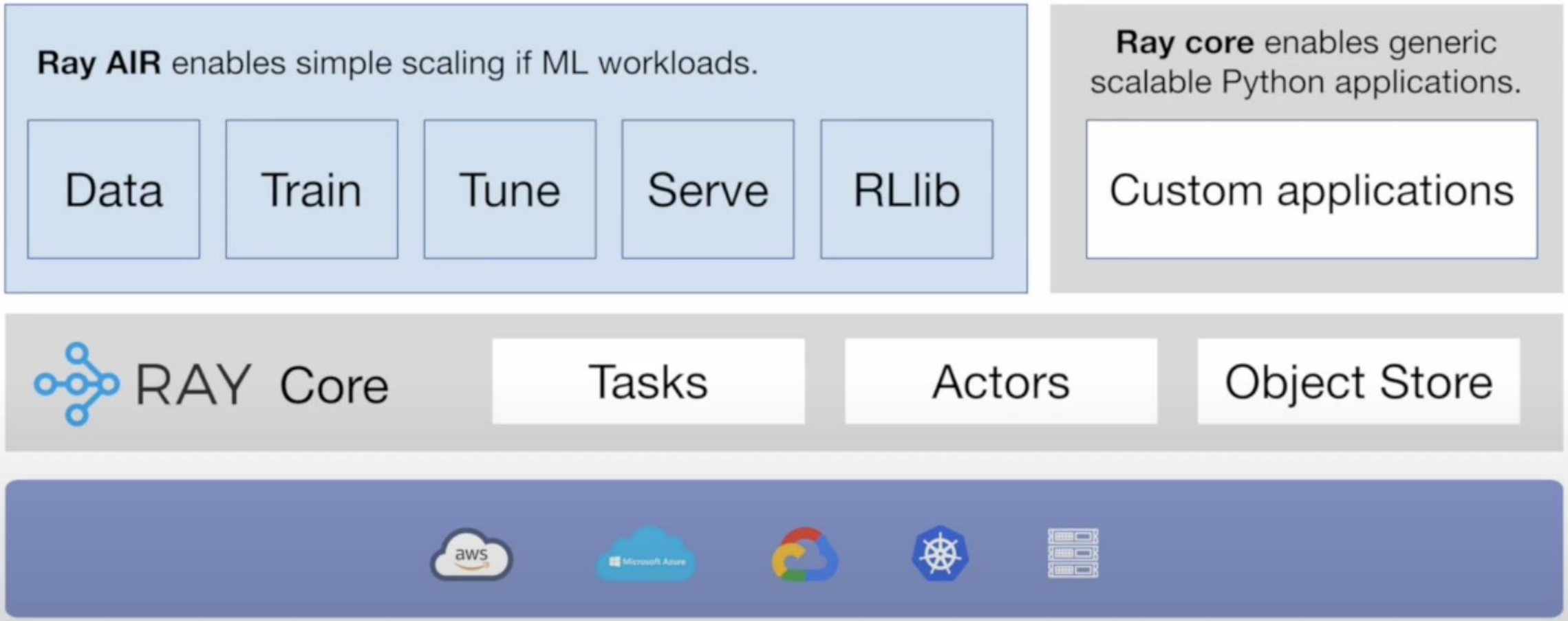
```
@ray.remote
def f():
    # compute...
    return x

@ray.remote
def g(a):
    # compute...
    return y

id_x = f.remote()
id_y = g.remote(id_x)
```



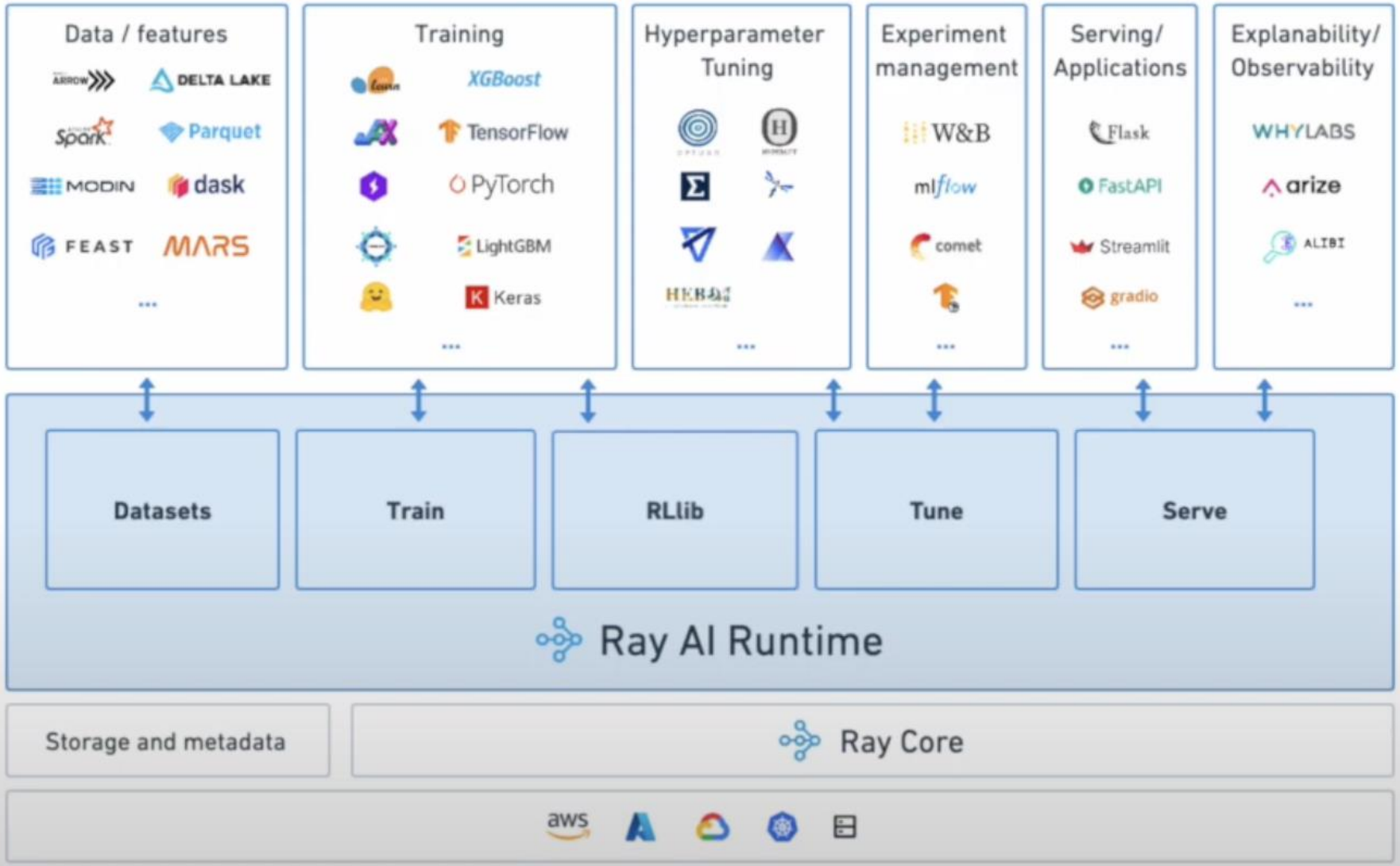
Shard in-memory object store



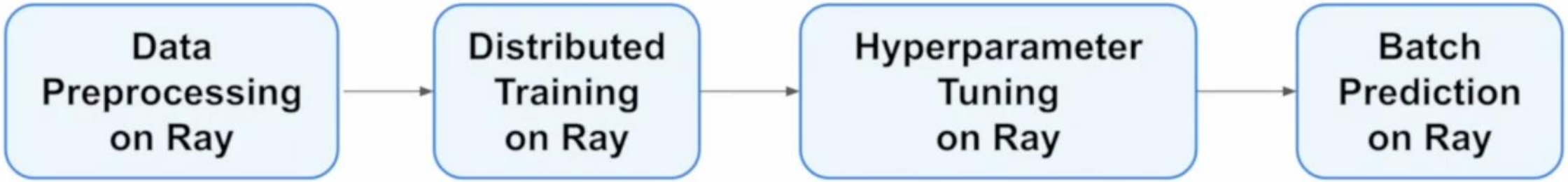
Ray Ecosystem



Ray AI Runtime (AIR): scalable runtime for E2E ML apps



Using Ray AIR to Scale "Classic" E2E DL Workflow



Using Ray AIR to Scale "Classic" E2E DL Workflows

```
dataset = ray.data.read_csv(...)
train_ds, valid_ds = train_test_split(
    dataset, test_size=0.3)
test_ds = valid_ds.drop_columns(["target"])
preprocessor = StandardScaler(columns=["mean radius"])
```

Scalable Data Preprocessing (Ray Data)

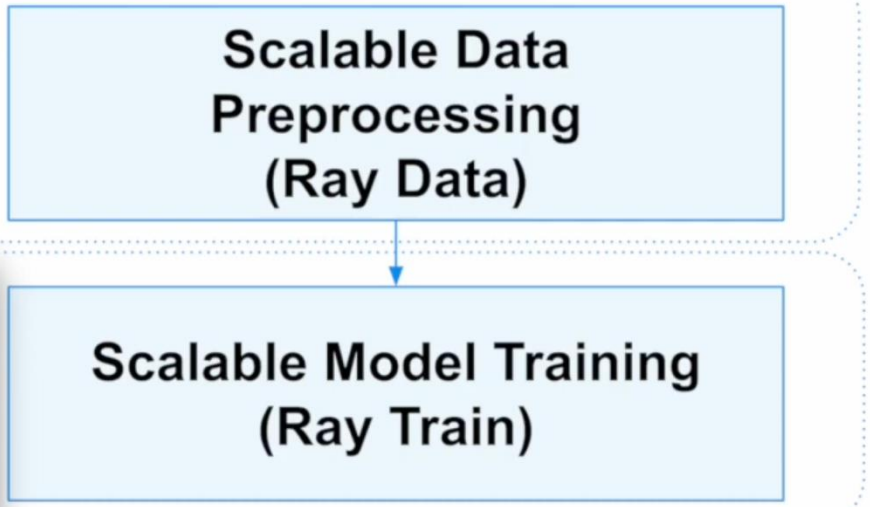
Ray Data library built for ML tasks

- Efficiently load distributed data from MB to TB scale
- Preprocessors for unified training <> inference

Using Ray AIR to Scale "Classic" E2E DL Workflow

```
dataset = ray.data.read_csv(...)
train_ds, valid_ds = train_test_split(
    dataset, test_size=0.3)
test_ds = valid_ds.drop_columns(["target"])
preprocessor = StandardScaler(columns=["mean radius"])
```

```
trainer = ray.train.huggingface.TransformersTrainer(
    scaling_config=ScalingConfig(num_workers=128),
    label_column="target",
    datasets=dict(train=train_ds, valid=valid_ds},
    preprocessor=preprocessor)
result = trainer.fit()
```

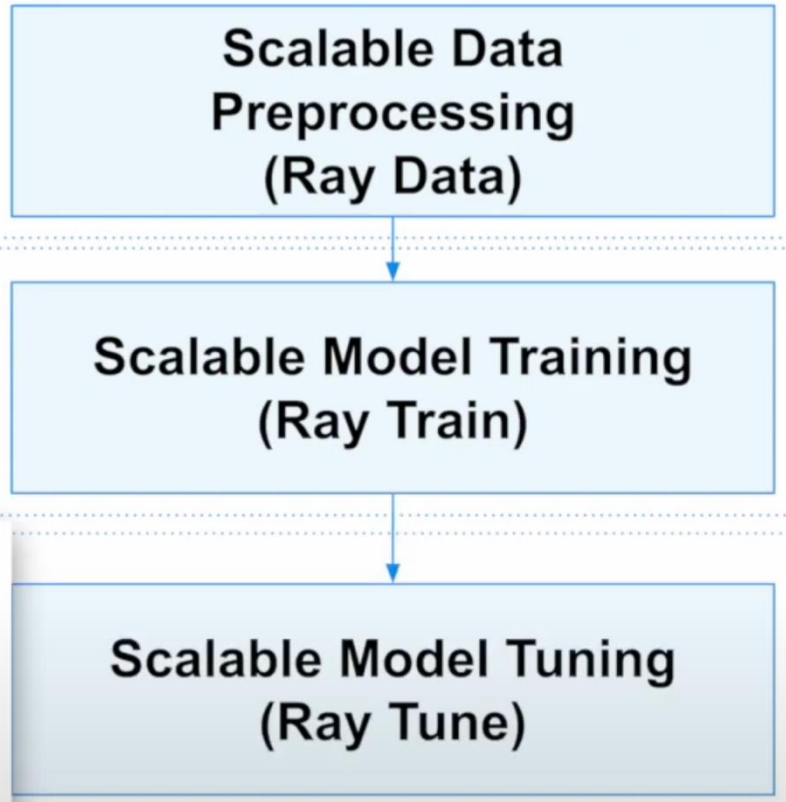


Using Ray AIR to Scale "Classic" E2E DL Workflows

```
dataset = ray.data.read_csv(...)
train_ds, valid_ds = train_test_split(
    dataset, test_size=0.3)
test_ds = valid_ds.drop_columns(["target"])
preprocessor = StandardScaler(columns=["mean radius"])
```

```
trainer = ray.train.huggingface.TransformersTrainer(
   scaling_config=ScalingConfig(num_workers=128),
   label_column="target",
   datasets=dict(train=train_ds, valid=valid_ds},
   preprocessor=preprocessor)
result = trainer.fit()
```

```
tuner = ray.tune.Tuner(
   trainer,
   param_space={"params": {"lr": tune.loguniform(1e-5, 5e-4)}},
   tune_config=TuneConfig(
      num_samples=5, metric="logloss", mode="min"),
)
checkpoint = tuner.fit().get_best_result().checkpoint
```



Using Ray AIR to Scale "Classic" E2E DL Workflows

```
dataset = ray.data.read_csv(...)
train_ds, valid_ds = train_test_split(
    dataset, test_size=0.3)
test_ds = valid_ds.drop_columns(["target"])
preprocessor = StandardScaler(columns=["mean radius"])
```

Scalable Data Preprocessing (Ray Data)

```
trainer = ray.train.huggingface.TransformersTrainer(
   scaling_config=ScalingConfig(num_workers=128),
   label_column="target",
   datasets=dict(train=train_ds, valid=valid_ds},
   preprocessor=preprocessor)
result = trainer.fit()
```

Scalable Model Training (Ray Train)

```
tuner = ray.tune.Tuner(
   trainer,
   param_space={"params": {"lr": tune.loguniform(1e-5, 5e-4)}},
   tune_config=TuneConfig(
      num_samples=5, metric="logloss", mode="min"),
)
checkpoint = tuner.fit().get_best_result().checkpoint
```

Scalable Model Tuning (Ray Tune)

```
batch_predictor = BatchPredictor.from_checkpoint(
    checkpoint, XGBoostPredictor)
predicted_probabilities = batch_predictor.predict(test_ds)
predicted_probabilities.show()
```

Scalable Batch Prediction (Predictors)

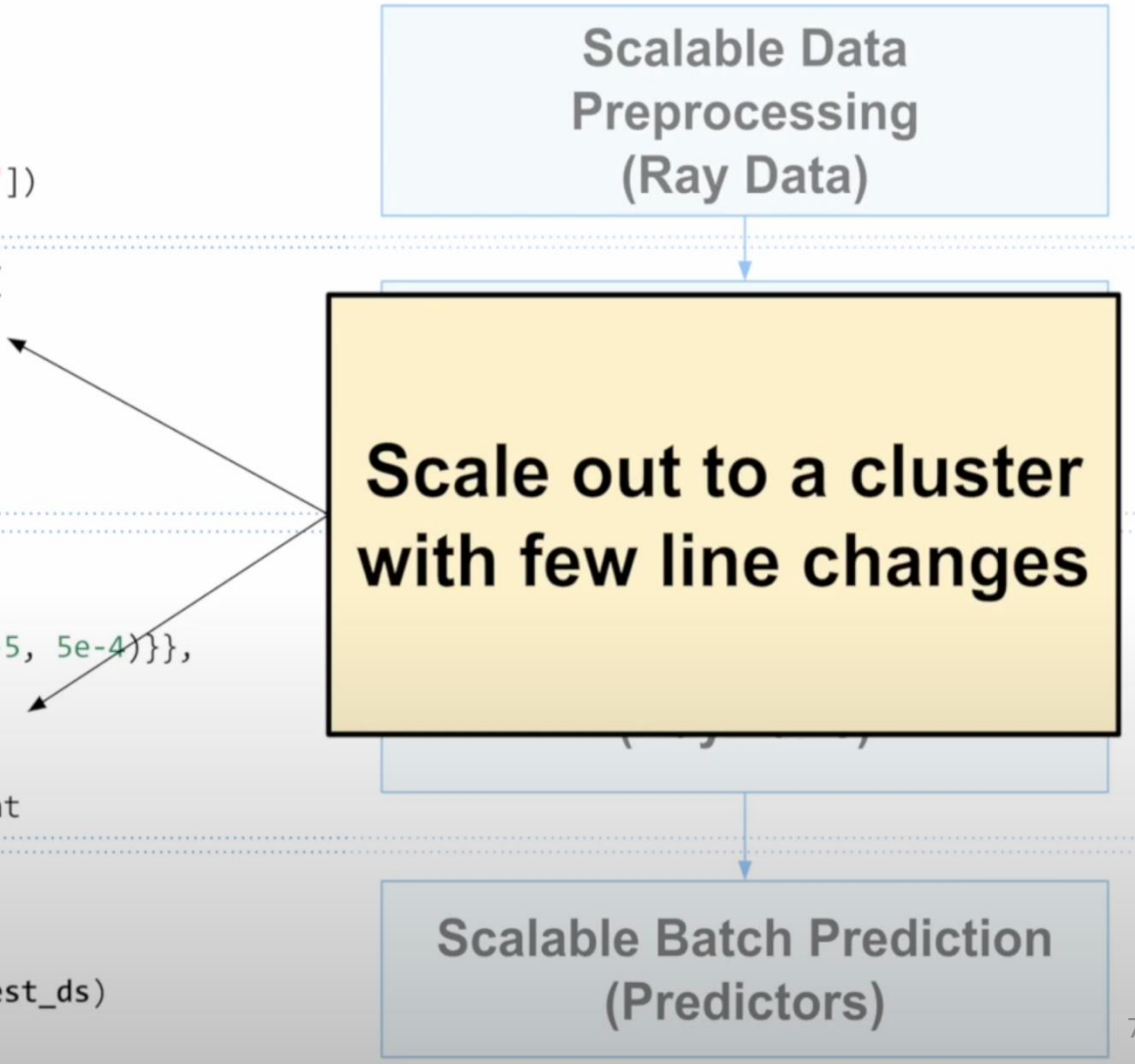
Using Ray AIR to Scale "Classic" E2E DL Workflows

```
dataset = ray.data.read_csv(...)
train_ds, valid_ds = train_test_split(
    dataset, test_size=0.3)
test_ds = valid_ds.drop_columns(["target"])
preprocessor = StandardScaler(columns=["mean radius"])
```

```
trainer = ray.train.huggingface.TransformersTrainer(
   scaling_config=ScalingConfig(num_workers=128),
   label_column="target",
   datasets=dict(train=train_ds, valid=valid_ds},
   preprocessor=preprocessor)
result = trainer.fit()
```

```
tuner = ray.tune.Tuner(
   trainer,
   param_space={"params": {"lr": tune.loguniform(1e-5, 5e-4)}},
   tune_config=TuneConfig(
      num_samples=5, metric="logloss", mode="min"),
)
checkpoint = tuner.fit().get_best_result().checkpoint
```

```
batch_predictor = BatchPredictor.from_checkpoint(
    checkpoint, XGBoostPredictor)
predicted_probabilities = batch_predictor.predict(test_ds)
predicted_probabilities.show()
```

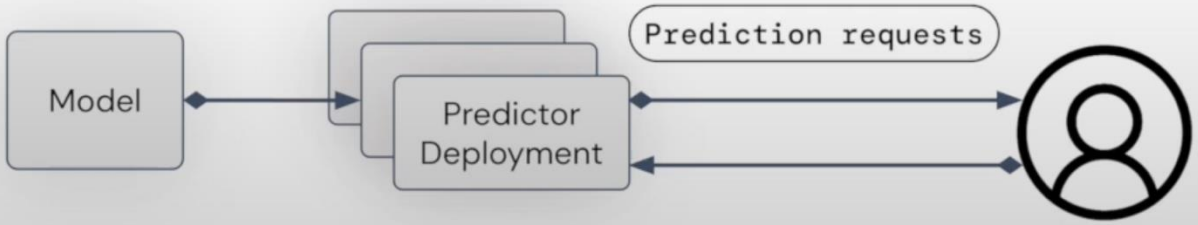


Scalable Online Inference

with Ray Serve

- Deploy single models as HA inference services in Ray
- Build multi-model pipelines with custom business logic

```
deployment = PredictorDeployment.options(
name="TransformerService")

deployment.deploy(TorchPredictor, checkpoint, ...)

print(deployment.url)
```



TorchX

Taking PyTorch in Production

Introduction

TorchX is an SDK for building and deploying ML apps from R&D to production

01 DEFINE OR CHOOSE

Define your own components (job defs) using the **torchx.specs** API or choose one of the builtin **torchx.components** and use them out-of-the-box.

02 RUN AS A JOB

Components can be run as a single node or distributed job in one of the supported schedulers or plugin your own scheduler

03 CONVERT TO PIPELINE

Components can be run in a production pipeline by converting them into pipeline stages of one of the supported ML pipeline orchestrators

Components

- **Component** - python function that parameterizes an AppDef
- **AppDef** - job definition data class
- **Builtin components**:
  - Use out-of-the-box
  - Compose to customize
  - Examples: dist.ddp, serve, hpo, etc

```python
from torchx.specs import AppDef, Role, named_resources
import torchx.components.dist as dist

def train(
    *script_args:str,
) -> AppDef:
    return AppDef(
      name="train",
      roles=[
        Role(
          name="worker",
          image="my/docker_img",
          entrypoint="my/trainer/main.py",
          args=*script_args,
          env={"RANK":"0", "WORLD_SIZE":"1"},
          resource=named_resources["aws_p3.8xlarge"],
        )
      ])

def train_dist(
    nnodes: int,
    nproc_per_node: int,
    *script_args:str,
) -> AppDef:
    return dist.ddp(
        *script_args,
        image="my/docker_img",
        script="my/trainer/main.py",
        j=f"{nnodes}x{nproc_per_node}",
        h="aws_p3.8xlarge")
```

TorchX Run

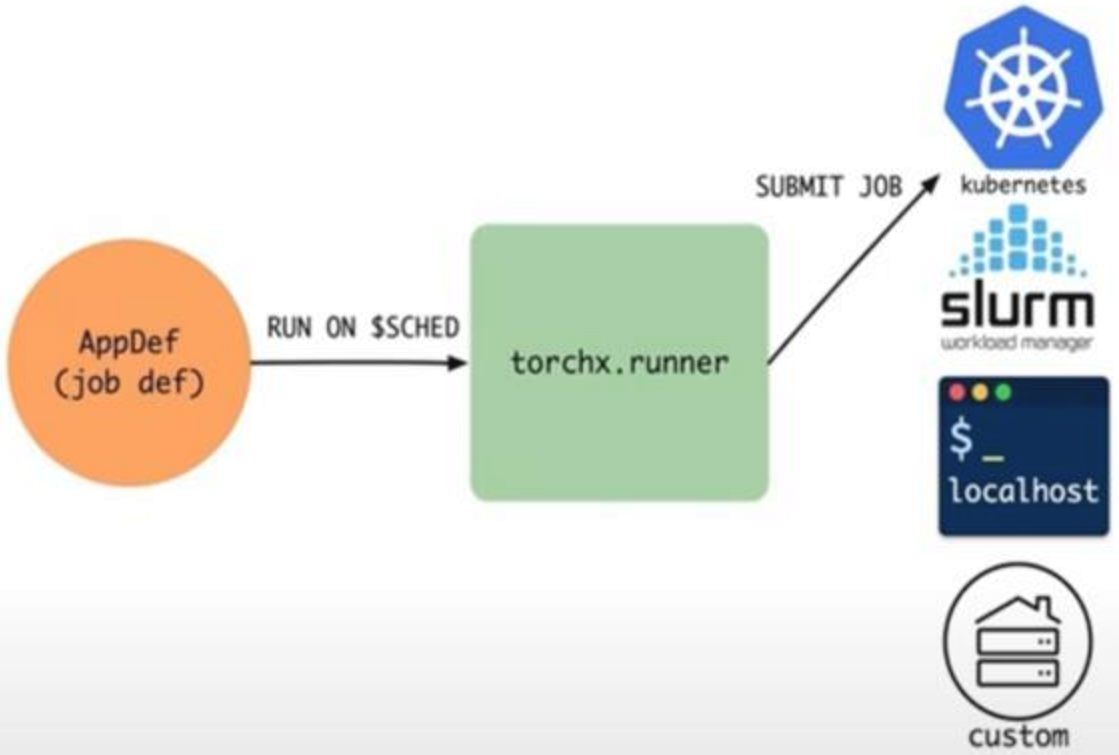
- Run a custom component

```
$ torchx run -s kubernetes ~/component.py:train_dist
    --nnodes 4
    --nprocs_per_node 8
    foo bar
```

- OR a builtin component

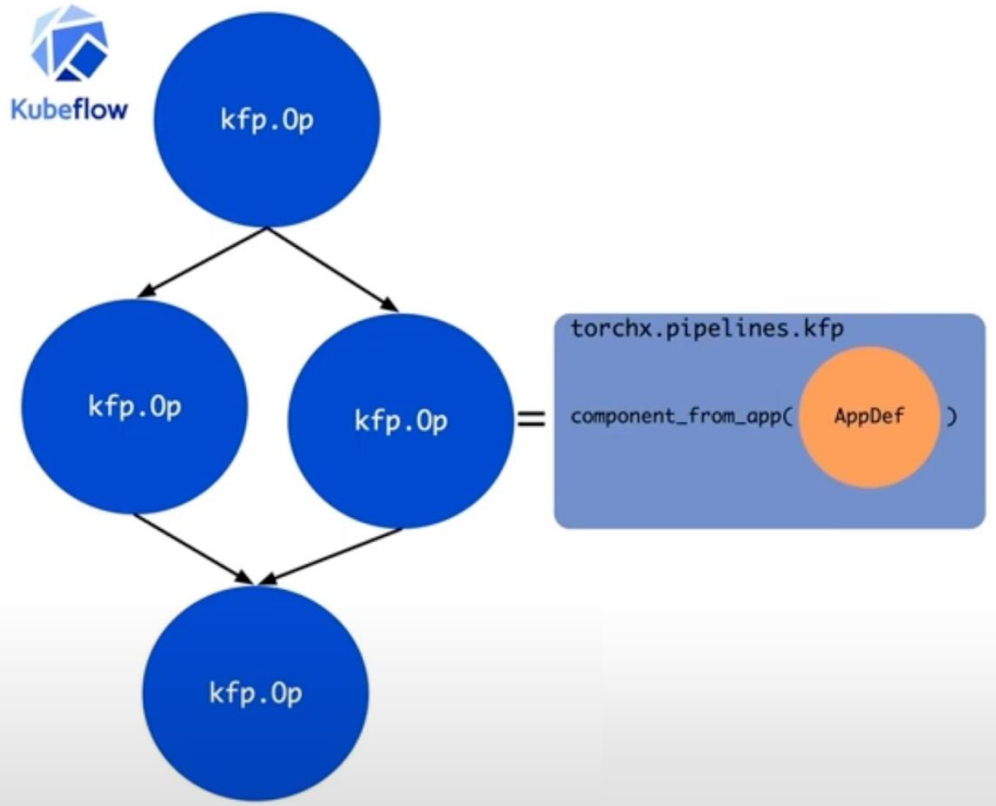
```
$ torchx run -s kubernetes dist.ddp
    -j 4x8
    -h aws_p3.8xlarge
    --image my/docker_img
    --script my/trainer/main.py
    foo bar
```

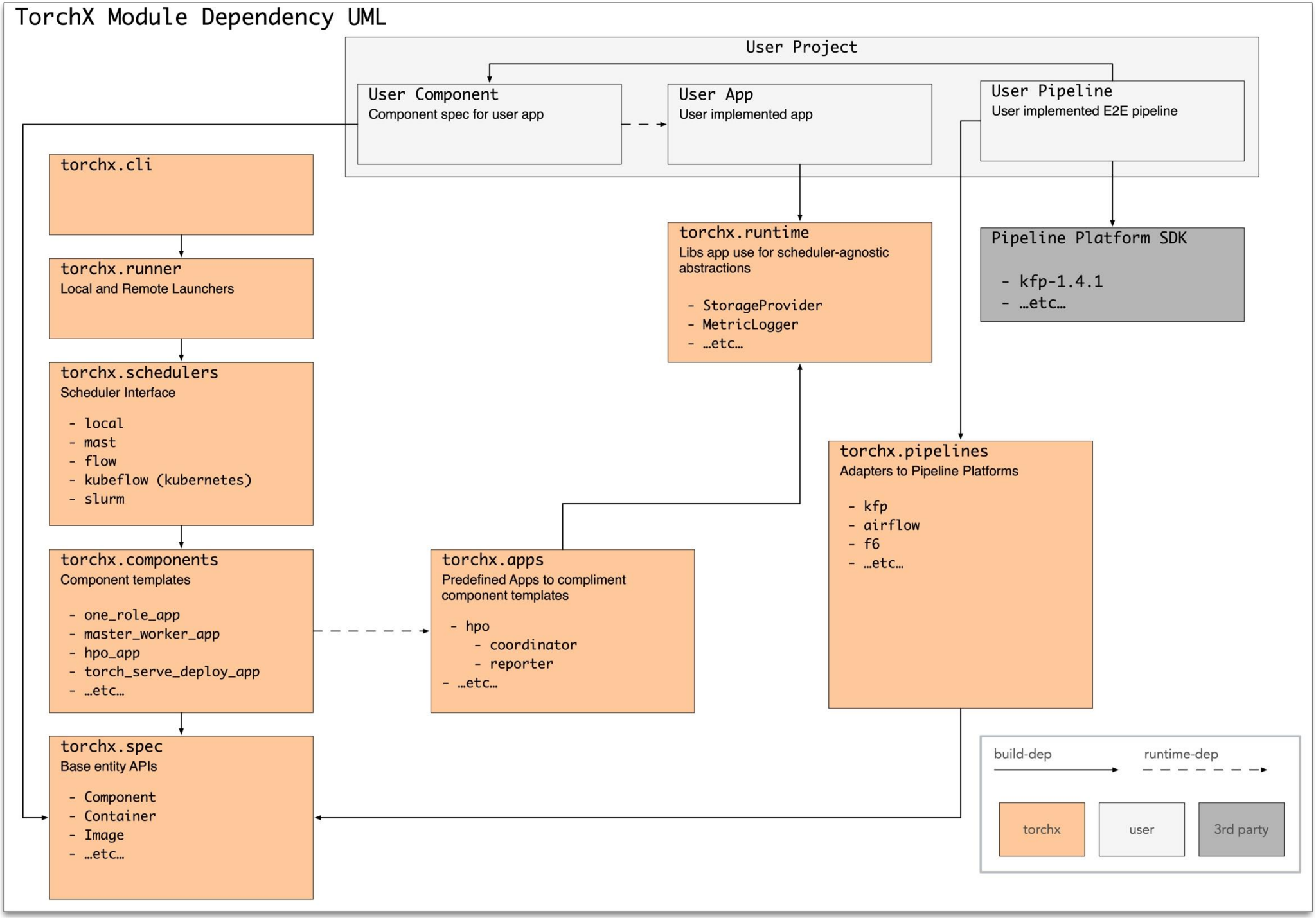
- Job runs locally or on one of the supported schedulers
- Easy to add a custom scheduler



TorchX Pipelines

- Run components as a stage in an ML pipeline
- Use **`torchx.pipelines`** adapter to convert a component into a pipeline op
- Define DAG (op dependencies) in the pipeline orchestrator's DSL not TorchX
- Can mix legacy pipeline ops with TorchX components




TorchX Module Dependency UML
User Project
User Component
Component spec for user app
User App
User implemented app
User Pipeline
User implemented E2E pipeline
torchx.cli
torchx.runner
Local and Remote Launchers
torchx.schedulers
Scheduler Interface
- local
- mast
- flow
- kubeflow (kubernetes)
- slurm
torchx.components
Component templates
- one_role_app
- master_worker_app
- hpo_app
- torch_serve_deploy_app
- …etc…
torchx.spec
Base entity APIs
- Component
- Container
- Image
- …etc…
torchx.runtime
Libs app use for scheduler-agnostic abstractions
- StorageProvider
- MetricLogger
- …etc…
Pipeline Platform SDK
- kfp-1.4.1
- …etc…
torchx.apps
Predefined Apps to compliment component templates
- hpo
- coordinator
- reporter
- …etc…
torchx.pipelines
Adapters to Pipeline Platforms
- kfp
- airflow
- f6
- …etc…
build-dep
runtime-dep
torchx
user
3rd party

Reference Papers

- Compute Trends across Three Eras of Machine Learning, https://arxiv.org/pdf/2202.05924
- Language Models are Few Shot Learners, https://splab.sdu.edu.cn/GPT3.pdf
- Pathways Language Model (PaLM): Scaling to 540 Billion Parameters for Breakthrough Performance,https://research.google/blog/pathways-language-model-palm-scaling-to-540-billion-parameters-for-breakthrough-performance/
- Moritz, Philipp, et al. "Ray: A distributed framework for emerging {AI} applications." *13th USENIX symposium on operating systems design and implementation (OSDI 18)*. 2018.

Blogs, Videos, Code Links

- Janakiram. 10 KEY ATTRIBUTES OF CLOUD-NATIVE APPLICATIONS
- AWS Documentation. How Amazon Sagemaker Works
- James Quigley. Microservices, Docker, and Kubernetes
- Ben Corrie. What is a container?. Video
- Microsoft Azure. Kubernetes Learning Path. A hands-on course.
- TorchX Quickstart.
- Kubeflow. Getting Started with Kubeflow