CS 744: RAY

Shivaram Venkataraman

Fall 2019

ADMINISTRIVIA

- Assignment 1 Grades
- Assignment 2 due on Fri
- Course Project emails

Machine Learning

Bismarck

Supervised learning, Unified Interface
Shared memory, Model fits in memory

Parameter Server

Large datasets, large models (PB scale)
Consistency model, Fault tolerance

Tensorflow

Need for flexible programming model
Dataflow graph
Heterogeneous accelerators

→ CNNs
→ language model
→ PS tasks

WORKLOADS

Bismarck

→ Convex optimization

→ Small datasets

simple models - SVMs, Logistic reg:

Parameter Server

→ Larger datasets → sparse data

→ High dim parameters → Ad Click model

Tensorflow

↳ Dense features

↳ Advanced models Supervised learning

Deep learning

↳ Image Classification Inceptionv3

Supervised learning

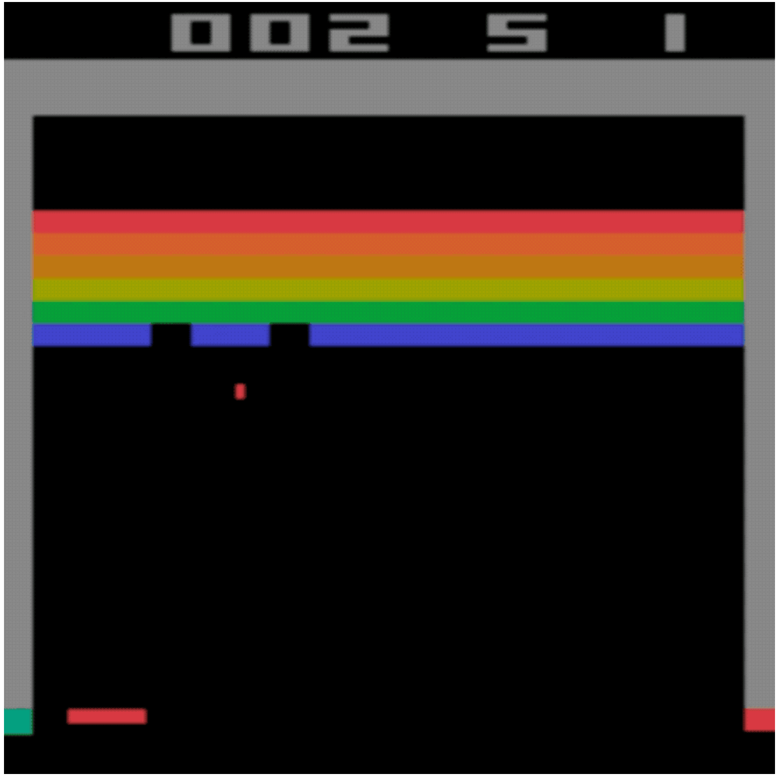
Training Data labels

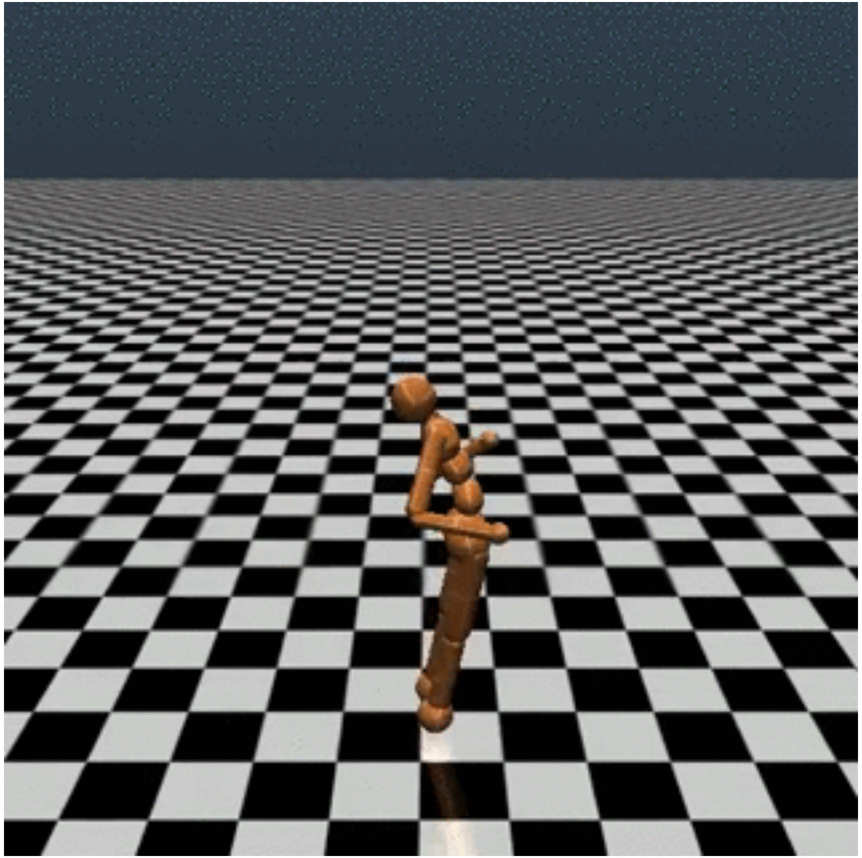
Model

$\longrightarrow$

fit

REINFORCEMENT LEARNING


002
5
1



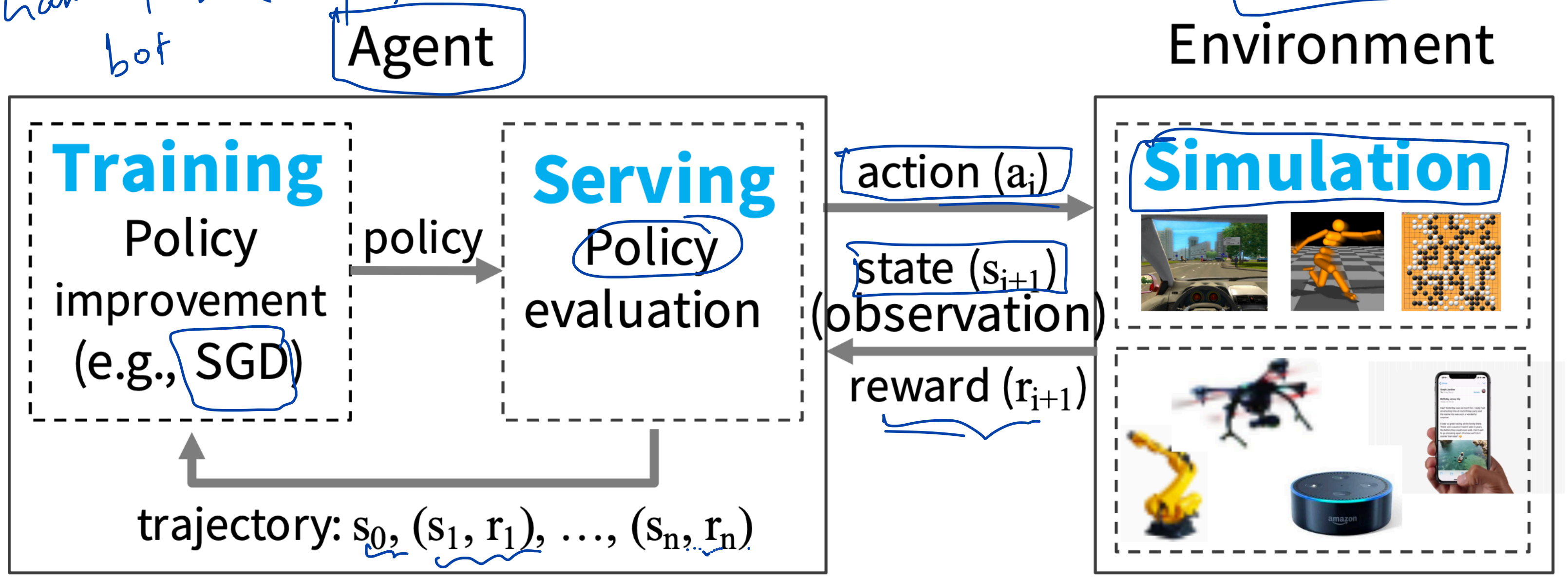
RL SETUP

State → Agent (Policy) → action

reward

Robot

Game playing bot



Agent

Environment

Training

Policy improvement (e.g., SGD)

policy

Serving

Policy evaluation

action (a_i)

state (s_{i+1}) (observation)

reward (r_{i+1})

Simulation

trajectory: $s_0, (s_1, r_1), \ldots, (s_n, r_n)$

improve Policy given sequence of state, reward

RL REQUIREMENTS

Stateful ≡ Game engine
+
Stateless

Simulation rollout

↳ Tasks could be of varying length 1ms → seconds / mins

Training Tasks are deterministic (state, action) → reward

Dynamic execution ≡ iteration structure
depends on current iteration

Serving

↳ very low latency ≡ Parallelism

Rollouts happen in parallel

RAY API

f1 : f.remote (args 1)

Tasks

normal python function

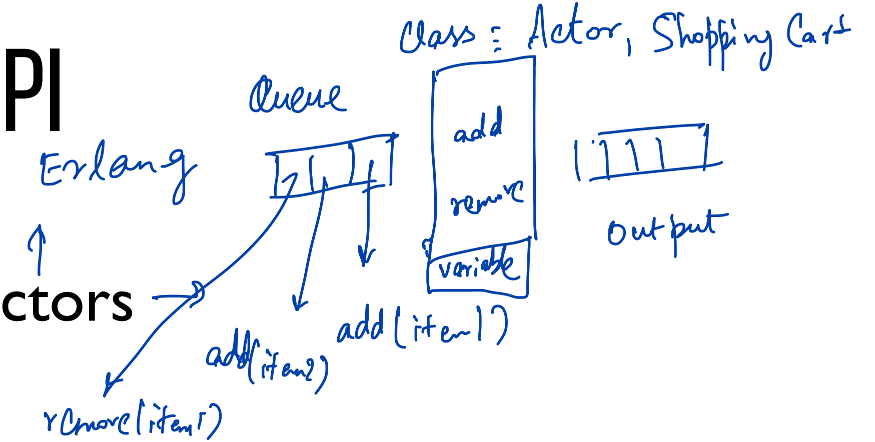
```
futures = f.remote(args)
```

Task which will run f with args

Handle to result of task

Actors

Erlang



```
actor = Class.remote(args)
futures = actor.method.remote(args)
```

= actor. method. remote (args 2)

f (args) before f (args 1)

```
objects = ray.get(futures)
ready = ray.wait(futures, k,timeout)
```

Stateless tasks

COMPUTATION MODEL

Vertices:

Data.

Tasks/ Actors

def task()

= task1.remote(a)

Edges

- - -> Control edge

→ Data edge Data vertex Args

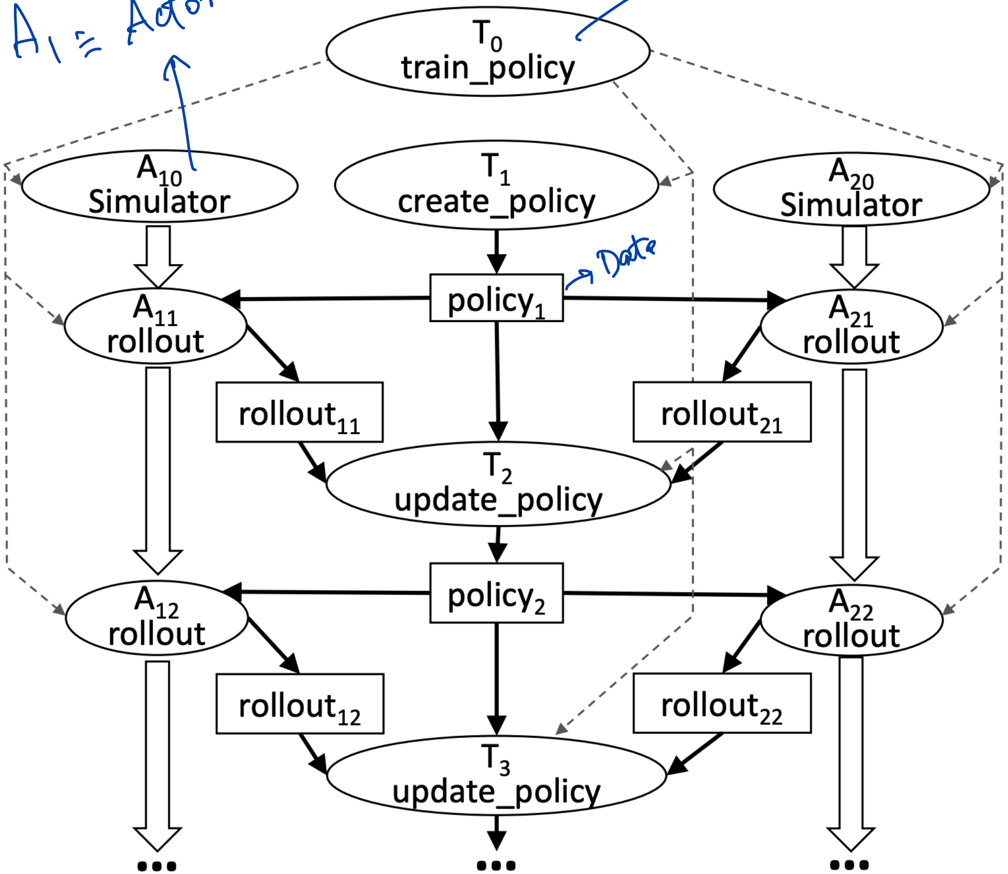
Task/ Actor

⟹ Stateful edge to ensure

Actor methods ordering

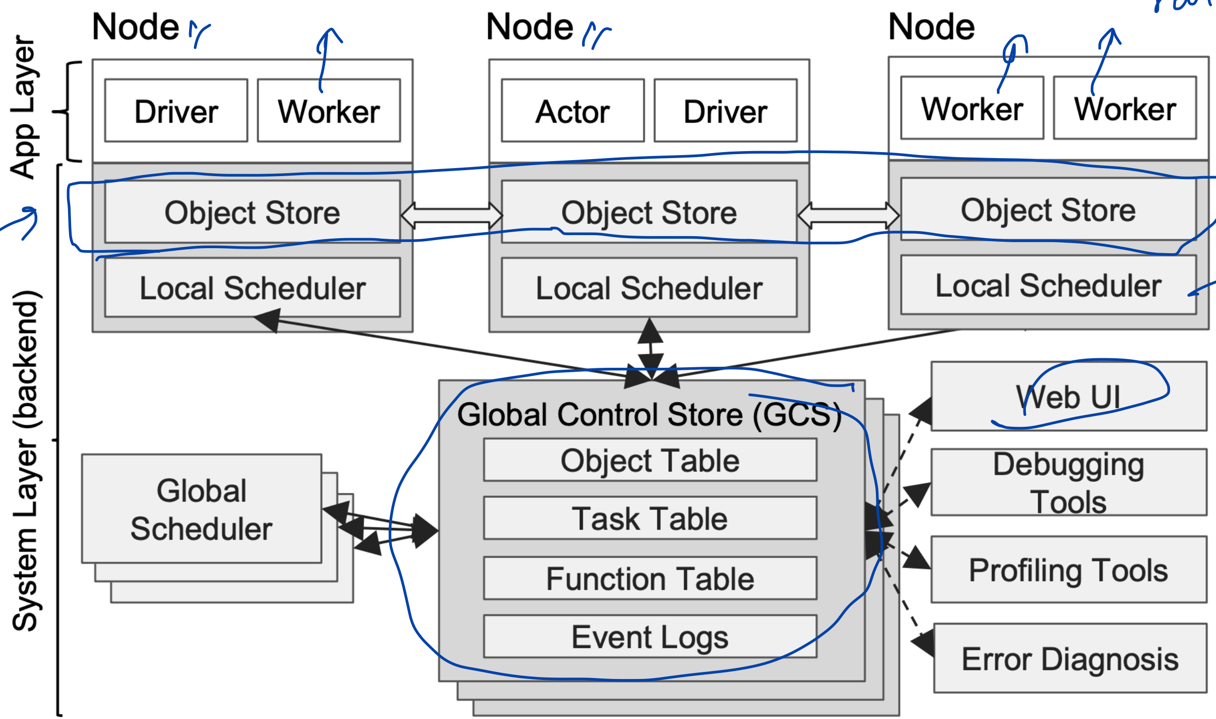
A1 = Actor

Task



Data

ARCHITECTURE



" Data plane "

In memory data node

run tasks
run actors ...

scheduling is "distributed"

GLOBAL CONTROL STORE

Object table

↳ list of object their locations | Namenode metadata

Task table

↳ Lineage, tasks created edge in Comp. graph

Function table

↳ Code blocks that are running

↳

Externalized State

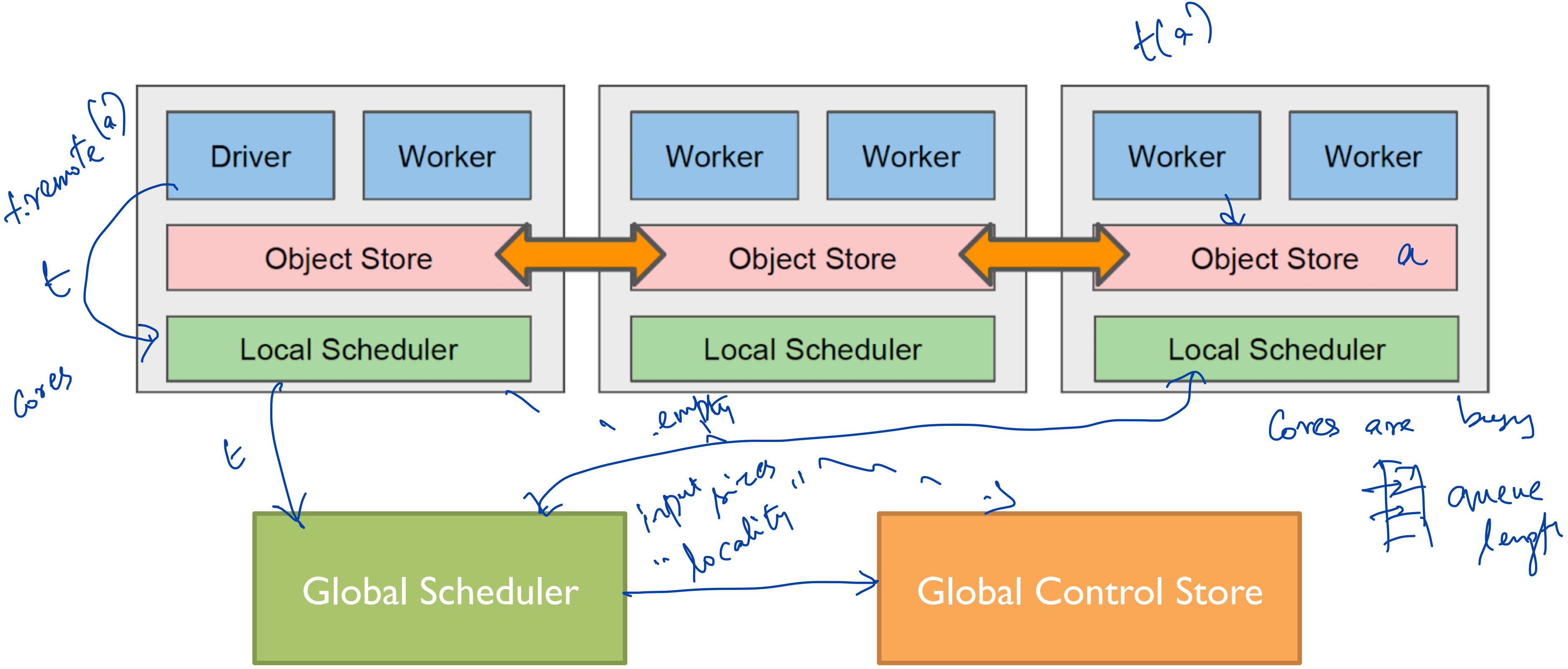
↳ Bottlenecks not in global system

↳ Sharding

Fault tolerance

↳ Chain replication

RAY SCHEDULER



FAULT TOLERANCE

Tasks

↳ lineage from GCS

Actors

↳ "checkpointing"

GCS

↳ sharded, replicated

Scheduler → Stateless, multiple schedulers or backup sched.

Driver

↳ computation fails

GCS → extra resources

↳ scalability

Object : Shard

DISCUSSION

https://forms.gle/QQyLbwjAufJNXWnr6

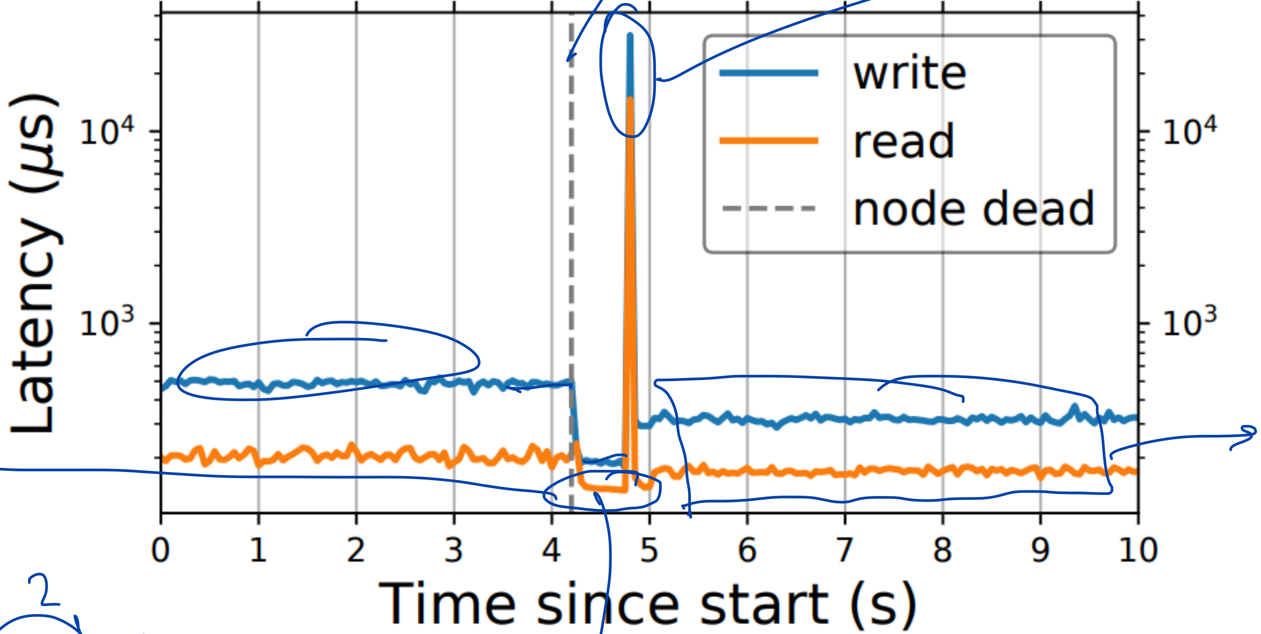
Consider you are implementing two ~~task~~ apps: a deep learning model training and a sorting application. When will use tasks vs actors and why ?

	Actors	Tasks
Sorting:	wait for other tasks?	You can parallelize and stateless operations ✓
Model training	Parameters are "state" don't need to broadcast ✓	Flexible sync. fine grained recovery?

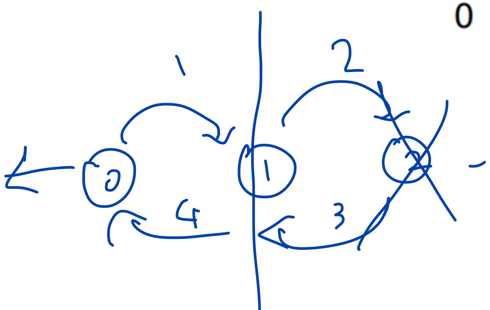
Why is read latency going down?

one of replicas GCS is down

warming up of new replica



lower than before



latency goes down!

→ New replica is somehow closer? or has more resources?

Considering AllReduce using MPI as the baseline parallel programming task. Discuss the improvements made by MapReduce, Spark over MPI and discuss if/how Ray further contributes to the comparison.

NEXT STEPS

Next class: Clipper

Assignment 2 due this week!

Course project