CS 744: RAY

Shivaram Venkataraman
Fall 2019
- Assignment 1 Grades
- Assignment 2 due on Fri
- Course Project emails
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
WORKLOADS

Bismarck
- Convex optimization
- Small datasets
  - Simple models - SVMs, logistic reg.

Parameter Server
- Larger datasets
- Sparse data
- High dim parameters
- Ad Circle model

Tensorflow
- Dense features
- Advanced models
  - Supervised learning
- Deep learning
  - Image classification
  - Inception v3
REINFORCEMENT LEARNING
RL SETUP

Training
Policy improvement (e.g., SGD)

Serving
Policy evaluation

Simulate
Agent action

Environment
Action (a_i)
State (s_{i+1})
Observation
Reward (r_{i+1})

trajectory: s_0, (s_1, r_1), ..., (s_n, r_n)

Improve policy given sequence of state, reward
RL REQUIREMENTS

Simulation
- Stateful vs. Stateless
  - Stateful = Game engine
  - Stateless

Training
- Tasks could be of varying length (e.g., seconds vs. minutes)
- 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

Tasks

- Normal Python function

futures = f.remote(args)

Task which with run
f with args

Handle to result of task

Objects

objects = ray.get(futures)

Stateless Tasks

Actors

- Class.remote(args)

actor = Class.remote(args)

futures = actor.method.remote(args)

- actor.method.remote(args)

f(args) before f(args)

Ready

ready = ray.wait(futures, k, timeout)
Vertices:
- Data
- Tasks / Actors

Edges:
- Control edge
- Data edge
- Stateful edge

```
def task(a):
    task1.remote(a)
```

Diagram:
- $T_0$ train_policy
- $T_1$ create_policy
- $T_2$ update_policy
- $T_3$ update_policy
- $A_1$ Actor
- $A_2$ Actor
- $A_10$ Simulator
- $A_11$ rollout
- $A_12$ rollout
- $A_20$ Simulator
- $A_21$ rollout
- $A_22$ rollout

States:
- $\text{policy}_1$
- $\text{policy}_2$
- $\text{rollout}_{11}$
- $\text{rollout}_{21}$
- $\text{rollout}_{12}$
- $\text{rollout}_{22}$

...
GLOBAL CONTROL STORE

Object table
- List of objects
- Their locations
- Namenode metadata

Task table
- Lineage, tasks created
- Edge in Compute graph

Function table
- Code blocks that are running

Externalized State
- Bottlenecks not in global system
- Sharding
- Fault tolerance
- Chain replication
RAY SCHEDULER

Diagram showing the components of the Ray scheduler, including:
- Driver
- Worker
- Object Store
- Local Scheduler
- Global Scheduler
- Global Control Store
FAULT TOLERANCE

Tasks
- lineage from GCS

Actors
- "checkpointing"

GCS
- sharded, replicated

Scheduler
- Stateless

Driver
- computation fails
- GCS -> extra resources
- scalability

Object: Shard
- multiple schedulers
- or backup sched.
DISCUSSION

https://forms.gle/QQyLbwjAufjNXWnr6
Consider you are implementing two tasks: a deep learning model training and a sorting application. When will you use tasks vs actors and why?

**Sorting:**
- **Actors:**
  - Wait for other tasks?

**Model Training:**
- **Parameters:**
  - "Make" don't need to broadcast

**Tasks:**
- You can parallelize and stateless operations
- Flexible sync
- Fine-grained recovery
Why is read latency going down? One of the replicas is down, warming up of new replica. Lower than before. New replica is somehow closer or has more resources.

Latency (μs)

Time since start (s)

Write
Read
Node dead

10^4
10^3
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