UNIVERSITY of WASHINGTON

Distributed Machine Learning with a Serverless Architecture

Authors: Hao Wang, Di Niu and Baochun Li
Institution: University of Toronto, University of Alberta

Team: Zhifei Cheng, Sijin Huang, Zichao Zhang

Outline

- Paper overview
- Background/Related work
- Summary of new technology
- Key contributions
- Experimental evaluation
- Author's conclusions
- Critique: strengths, weaknesses and evaluation
- Gaps

Paper Overview

What is the problem being solved?

- ➔ Rapid growth of data sparked broad interests for **distributed machine learning systems**
- ➔ Current distributed training methods
  - ➔ Based on a dedicated cluster of physical or virtual machines have posed non-trivial cluster management overhead
  - ➔ Mismatch between the dynamically varying resource demands during a model training job and the inflexible resource provisioning

Paper Overview

Why is it a problem that is worth solving?

- ❑ Machine learning (ML) has become fundamentally important in a wide range of research areas, including computer vision, speech recognition, and natural language processing.
- ❑ Imperative need to improve the performance when training machine learning models
- ❑ Presence of larger volumes of data and increasingly complex models.

Background

Distributed ML Training and Parameter Servers

- Traditional centralized gradient descent algorithm iteratively updates after the gradients computed
- can also be parallelized by partitioning the entire training dataset among multiple workers
- One fundamental idea of current parameter server systems is to use Stale Synchronous Parallel (SSP) or Bounded Delay Synchronization of MXNet, allowing the progress of straggler workers to be out of sync with other workers to a certain extent.

Background

Serverless vs IaaS Architectures

Preliminary experiment:

Train logistic regression model with 100,000×2 matrix data on

(1) AWS Lambda with parameter server
(2) EC2 c5.2xlarge instance

	Loss Value	Time (s)	Cost ($)
20 functions	0.009725	237.40	0.019
8-core EC2	0.009779	307.87	0.029
150 functions	0.009699	50.04	0.031
X functions	0.009761	202.55	0.019

TABLE I: Loss values, time and cost of different resource configurations, after the ML training job converges.

20 functions: Serverless

8-core EC2: IaaS

Serverless training could:

- **Reduce training time** by 22.9%
- **Reduce cost** of 34.5%

Background

Resource Provisioning Policies

Preliminary experiment:

Train logistic regression model

	Loss Value	Time (s)	Cost ($)
20 functions	0.009725	237.40	0.019
8-core EC2	0.009779	307.87	0.029
150 functions	0.009699	50.04	0.031
X functions	0.009761	202.55	0.019

TABLE I: Loss values, time and cost of different resource configurations, after the ML training job converges.

Training time and cost vary with resource provisioning:

- ❏ 150 functions is faster to converge, though more costly
- ❏ X function reduces training time with the same cost as 20 functions.

X functions: 120 functions (the first epoch) + 20 functions (intermediate epochs) + 10 functions (last epoch)

Summary of New Technology

SIREN : Architecture Overview

SIREN: a new framework for distributed machine learning based only on serverless architectures

Module A: serverless cloud platform

> launches stateless functions for the ML training job

Module B: local client

> resource scheduling decisions using deep reinforcement learning

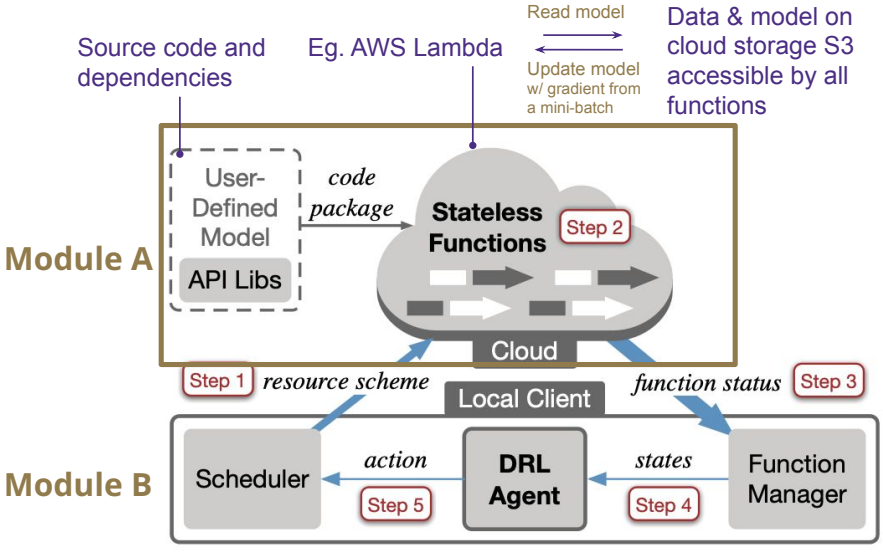


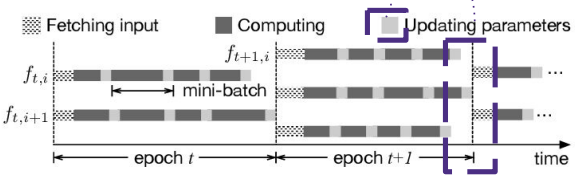
Fig. 2: The system architecture and workflow of SIREN

Summary of New Technology

SGD on Stateless Functions

Hybrid Synchronous Parallel

- **Module A**: Lambda functions **asynchronously** calculate gradient to update the target model on S3 after each **mini-batch**
- **Module B**: At the end of the epoch, fetch current states **synchronously** to update the resource scheme for the next epoch



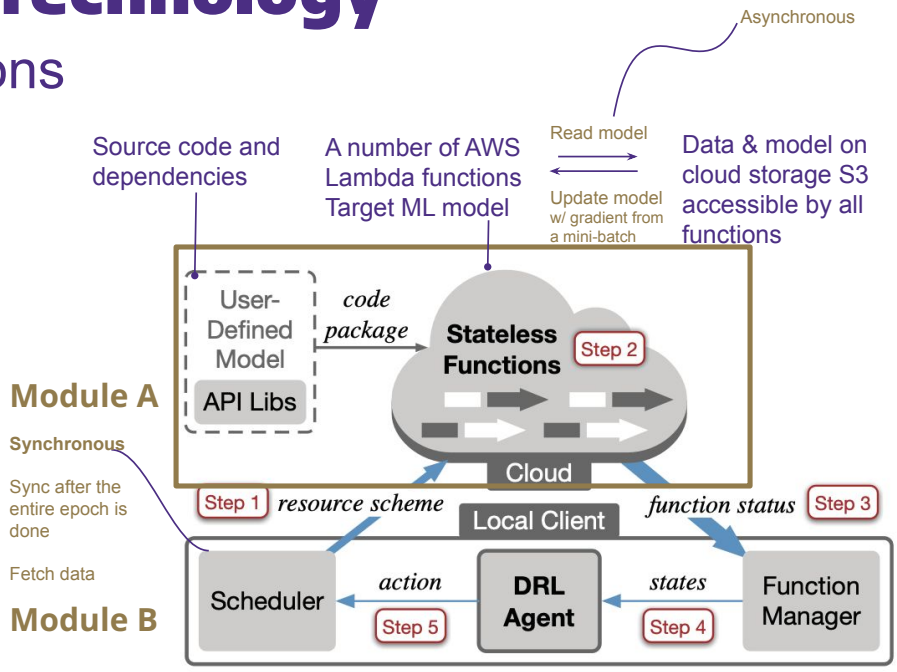


Fig. 2: The system architecture and workflow of SIREN.

Summary of New Technology

Deep Reinforcement Learning (DRL) for Dynamic Resource Provisioning

Objective:

Minimize **sum of training time**,

subject to a **cost budget**

Method:

Each epoch the DRL decides provisioning scheme (n, m)

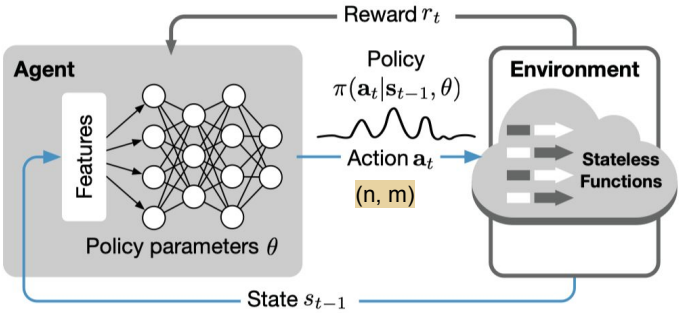


Fig. 4: DRL with policy represented by a DNN.

DRL Input: state of previous epoch

- Loss value
- avg data fetching, computing and parameter updating time, completion time
- avg mem and CPU utilization
- remaining budget

DRL Output: provisioning scheme

- n: number of concurrent functions
- m: memory size of each function

Summary of New Technology

Deep Reinforcement Learning (DRL) for Dynamic Resource Provisioning

Reward of epoch:

- Longer training time, less reward
- Extra positive reward if training converged within budget, negative otherwise

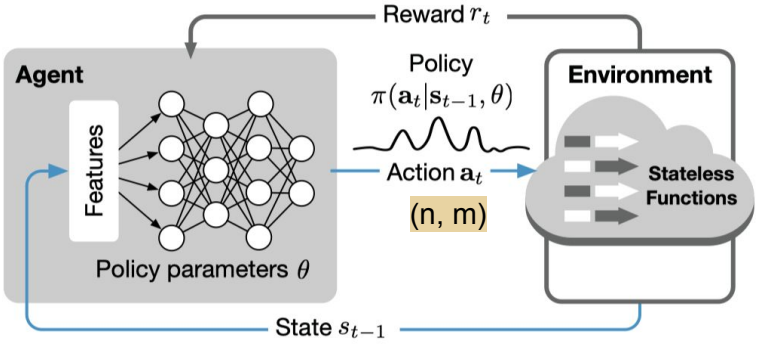
Training DRL agent:

Phase 1: Forward

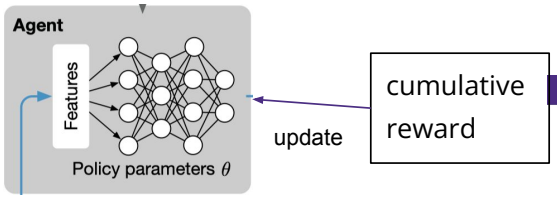
- Each epoch the DRL predicts provisioning scheme (n, m)
- Invoke Lambda following scheme (n, m)
- Collect reward of current epoch

Phase 2: Backward

- After forward phase finished for all epochs
- Use gradient descent on DRL params to pursue higher cumulative reward



Phase 1: Forward



Phase 2: Backward

Key Contributions

Distributed machine learning framework based on serverless architecture

- ❑ Reduce complexity to manage and maintain computing clusters
- ❑ Resources can be easily scaled up or down
- ❑ Reduce cost

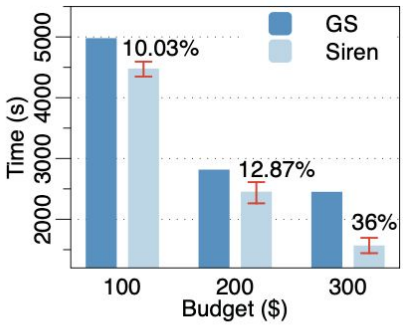
DRL-based scheduler

- ❑ learns the best way to achieve a balanced trade off between model training quality and cost

Experimental Evaluation

Simulation

- Mini-batched SGD algorithm with SIREN
- Simulation environment: OpenAI Gym
- Baseline methods: grid search over a particular number of functions
- SIREN reduces the training time compared to grid search, given the same cost



(a) Comparison on ML training time.

Fig. 5: A comparison between SIREN and the grid search for the best number of functions.

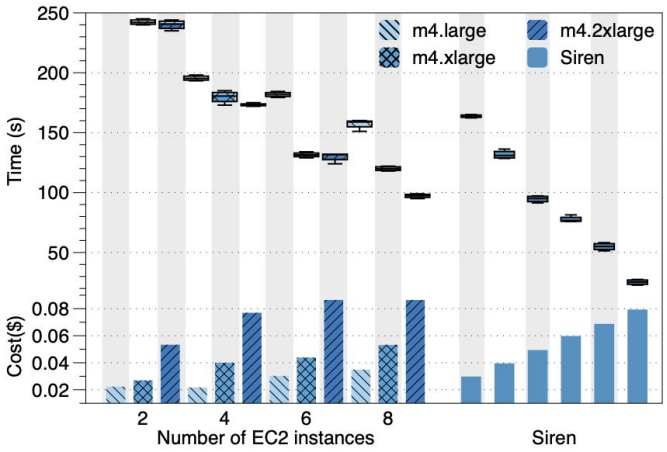
	Function #	Cost ($)	Time (s)
Grid Search	828	299.89	2452.3
SIREN	652 – 892	299.92	1569.5
Grid Search	482	199.67	2816.9
SIREN	355 – 597	199.73	2454.4
Grid Search	138	99.99	4979.7
SIREN	56 – 258	99.82	4480.4
Grid Search	3000	47.76	Fail
SIREN	1293 – 2995	49.82	Fail

TABLE III: A comparison between SIREN and the grid search for the best number of functions under different budgets.

Experimental Evaluation

Testbed

- Completion time and cost
  - EC2: non-linear
  - SIREN: shorter completion time, more cost
- SIREN on AWS Lambda vs. MXNet on EC2 clusters
- Training task: LeNet model
- EC2 instances:
  - m4.large (2 vCPU, 8GB memory, $0.1/h)
  - m4.xlarge (4 vCPU, 16GB memory, $0.2/h)
  - m4.2xlarge (8 vCPU, 32GB memory, $0.4/h)



(a) Completion time and cost of EC2 and SIREN under different settings.

Fig. 6: Training LeNet on the MNIST dataset by SIREN and by MXNet on EC2.

Experimental Evaluation

SIREN is faster than EC2 clusters at the same cost

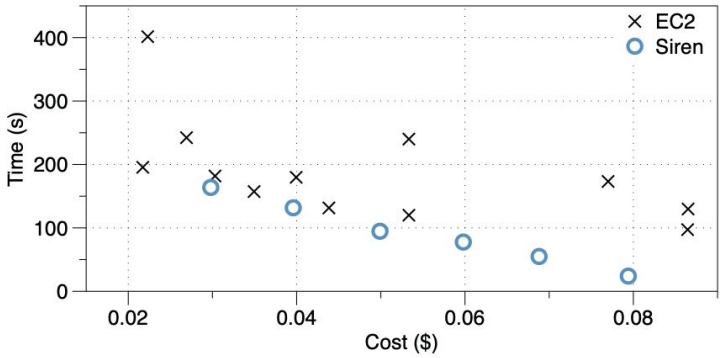


Fig. 7: Training completion time and cost of EC2 clusters and SIREN.

Experimental Evaluation

Workload:

SIREN reduces training time for these three models at the same cost.

Model	Dataset
LeNet	MNIST
convolutional neural network (CNN)	Sentiment-analysis of a movie review
Linear classification	Avazu click-through prediction dataset

m4.2xlarge x8 cluster vs. SIREN

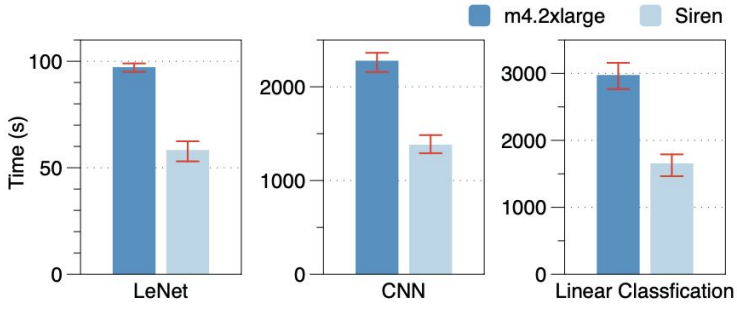


Fig. 8: A comparison between SIREN and EC2 under the same cost budget for different models.

Conclusions

- ❏ Design and implementation of SIREN
- ❏ Design a DRL-based scheduler
- ❏ Implement a prototype of SIREN based on AWS Lambda
- ❏ Evaluate SIREN with ML models
- ❏ Compared SIREN with ML training jobs on AWS EC2

Critique

Strengths

> Innovatively proposed SIREN, an *asynchronous* distributed ML (DRL) framework based on ***serverless*** architecture.

> Reduce model training time up to **44%** compared to traditional ML training benchmarks on AWS EC2 at the same cost.

Critique

Weaknesses

- Lacks enough detail to reproduce the result.
- Lacks clarification on whether the scheduling DRL model is universal that can be applied to any ML model with minimum modification or it need to be tailored case by case.

Critique

Evaluation

- Lack of consistency in cost comparison. Eg. excluding the EC2 provisioning time and lambda free quotas.
- Lack of inclusion other cost besides memory. Eg GPU config & storage.
- Lack of launch overhead such as cold vs. warm invocations.
- Lack of proof of scalability in real-world in terms of Dataset size and complexity. Eg. training LeNet on the MNIST dataset.
- Would be more solid if the paper could provide the cost of DRL scheduling model at the local client in terms of GPU, CPU, runtime and memory.

Gaps

> Potentially deployment on Fargate using CaaS will offer more flexibility in memory, CPU, and GPU setting. Currently on AWS Lambda, there is an upper limit for execution time of 15 minutes. Memory limit up to 30 GB.

> The author could also investigate on the comparison of cost given same training time.

Gaps

> The datasets, which are loaded in every learning task, are stored in the Amazon S3 object storage. An alternative would be the AWS Elastic File System (EFS) which can be mounted directly for AWS Lambda calls.

Amazon S3, Amazon EFS, and AWS EBS Pricing Table

Features	Amazon S3	Amazon EBS	Amazon EFS
Storage Cost (US-East, for example)	Amazon S3 pricing: First 50 TB / Month $0.023 per GB Next 450 TB / Month $0.022 per GB Over 500 TB / Month $0.021 per GB	Amazon EBS pricing: General Purpose SSD (gp2) $0.10 per GB-month Provisioned IOPS SSD (io1) $0.125 per GB-month ($0.065 per provisioned IOPS-month) Throughput Optimized HDD (st1) $0.045 per GB-month Cold HDD (sc1) volumes $0.025 per GB-month	Amazon EFS pricing: $0.30 GB-month ($6.00 per provisioned MB/s-month)
Storage Size	No limit on number of objects	Maximum storage size of 16 TB	No limitation on the size of the file system
File Size Limitation	Individual Amazon S3 objects can range from a minimum of 0 bytes to a maximum of 5TB	No limitation on file size in EBS disk	Single files have a maximum size of 47.9TiB

Questions?

UNIVERSITY *of* WASHINGTON

THANK YOU

Team: Zhifei Cheng, Sijin Huang, Zichao Zhang