Cirrus:
A Serverless Framework for End-to-end ML Workflows

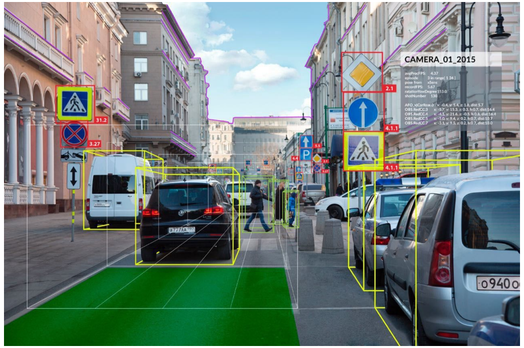
Joao Carreira, Pedro Fonseca, Alexey Tumanov, Andrew Zhang, Randy Katz

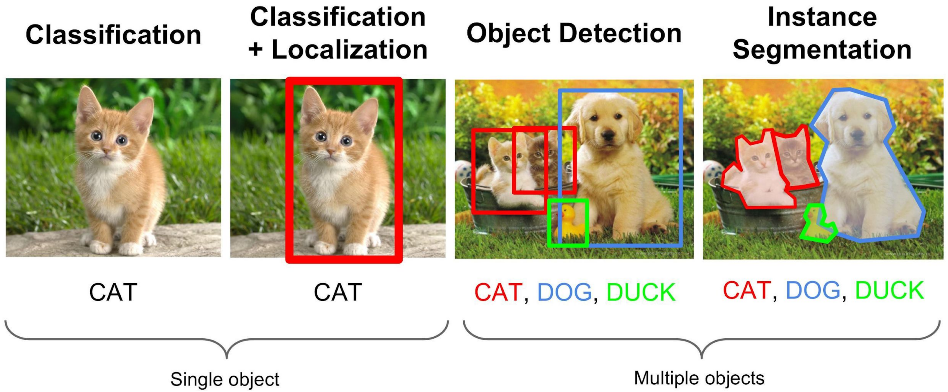
Berkeley
UNIVERSITY OF CALIFORNIA

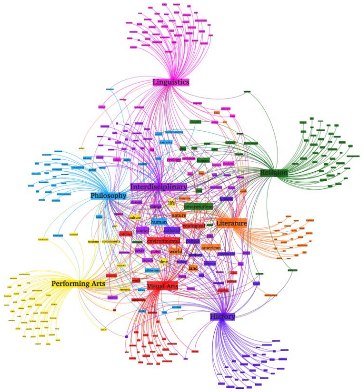
PURDUE
UNIVERSITY

Georgia Tech

Machine Learning







End-to-end ML workflows

- Modern end-to-end ML workflows are complex

End-to-end ML workflows

- Modern end-to-end ML workflows are complex
- ML workflows consist of 3 heterogeneous stages

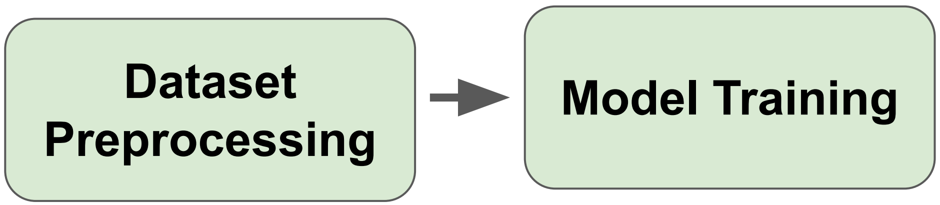
End-to-end ML workflows

- Modern end-to-end ML workflows are complex
- ML workflows consist of 3 heterogeneous stages

Dataset Preprocessing

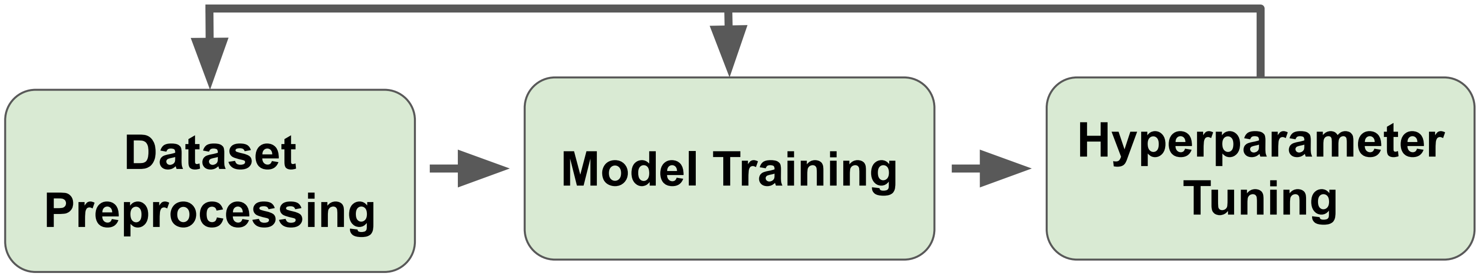
End-to-end ML workflows

- Modern end-to-end ML workflows are complex
- ML workflows consist of 3 heterogeneous stages



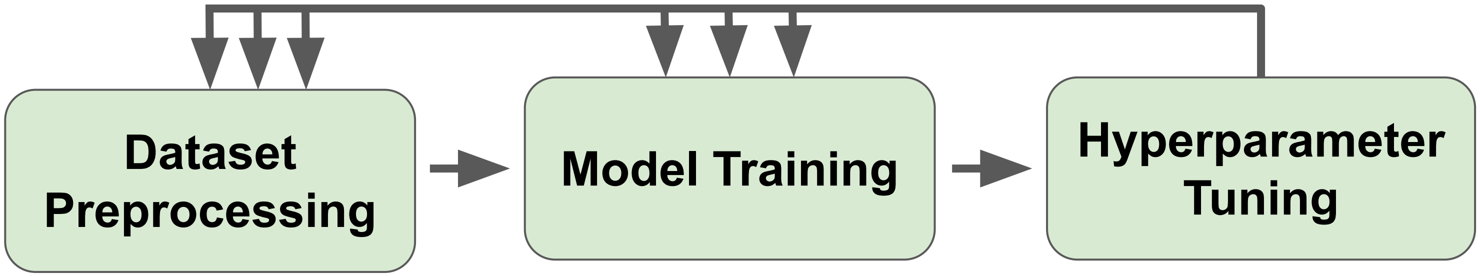
End-to-end ML workflows

- Modern end-to-end ML workflows are complex
- ML workflows consist of 3 heterogeneous stages



End-to-end ML workflows

- Modern end-to-end ML workflows are complex
- ML workflows consist of 3 heterogeneous stages



ML workflows are **interactive** and **iterative**

Provisioning ML workflows

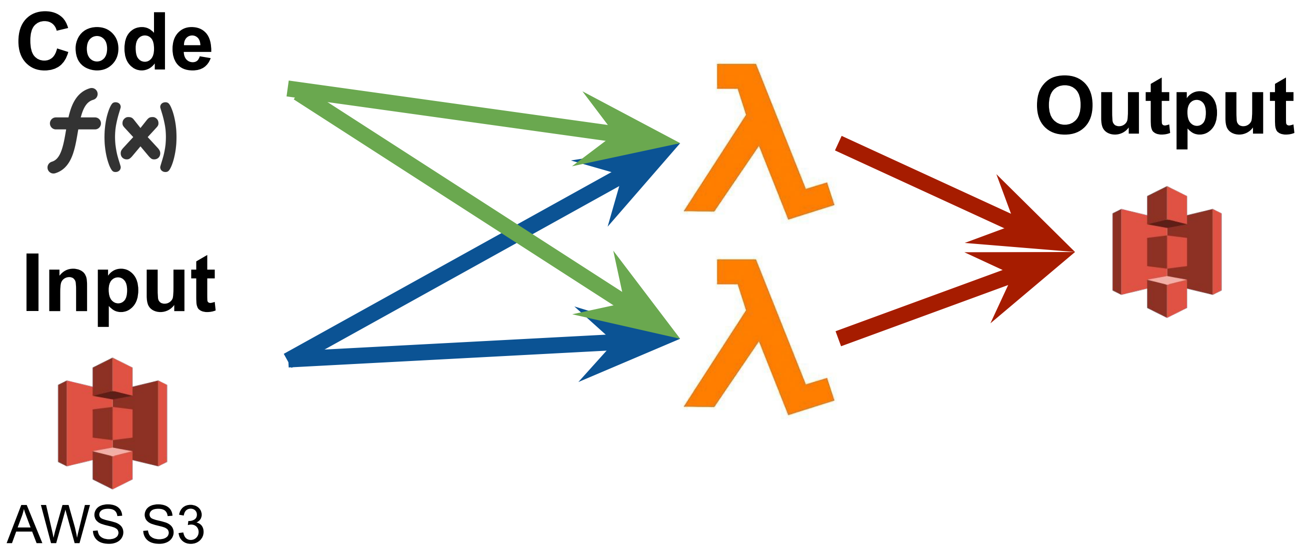
Provisioning ML workflows is challenging

Hard to accurately estimate resource demands of each stage

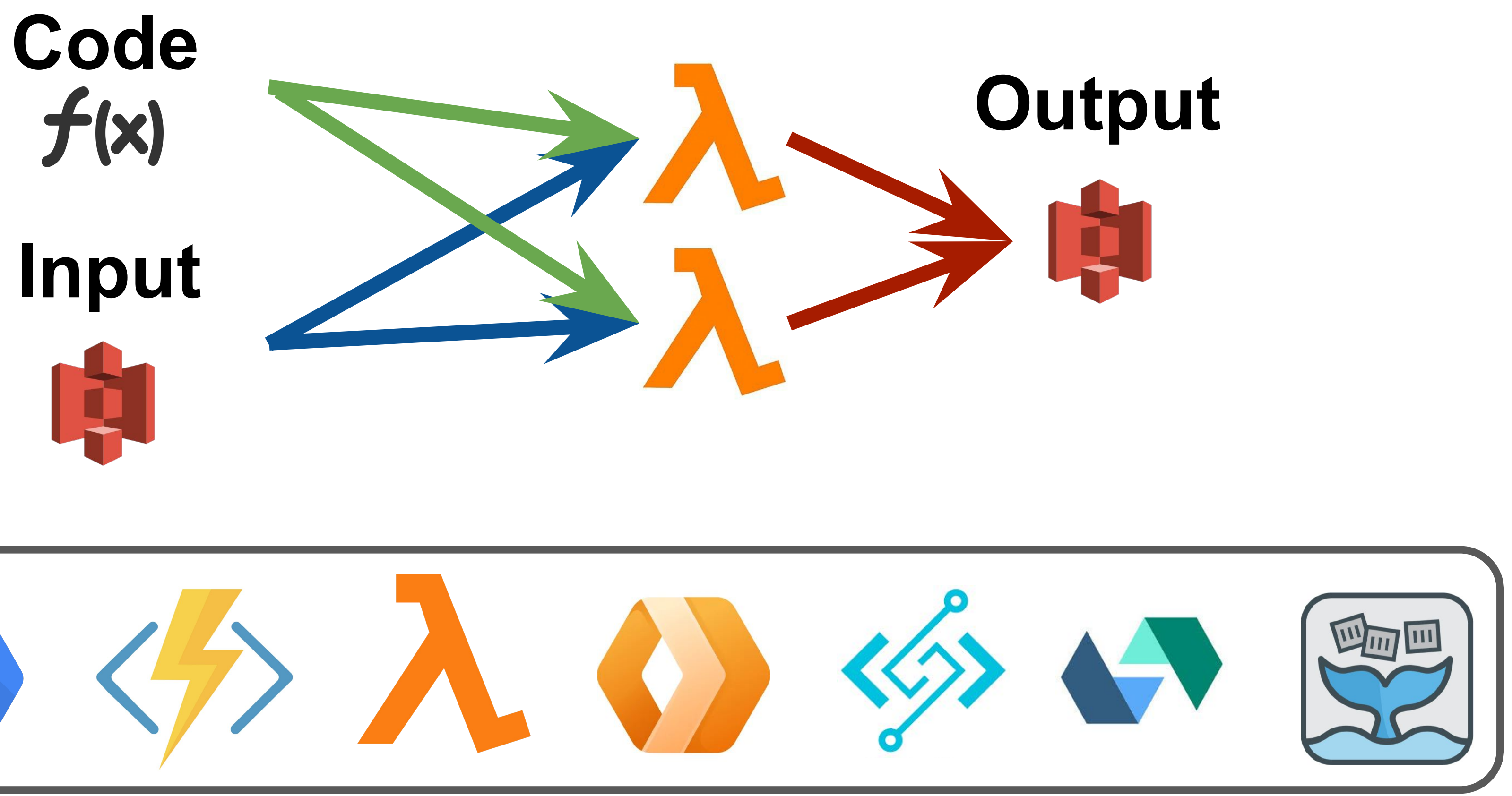
Data scientists have limited systems expertise

- Complex infrastructure management **detracts from ML work**
- **Resource waste** due to overprovisioning of resources

Serverless computing



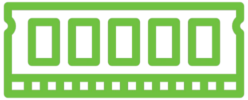
Serverless computing



Serverless computing benefits

Tight provisioning of resources

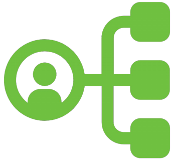
Fine-grained billing



Fine-grained resources

High elasticity

Simplifying infrastructure management



Automatic resource configuration / provisioning / maintenance

Challenges of serverless

Small local memory and storage

Low bandwidth and no P2P communication

Limited lambda package size

Lack of fast shared storage

Short-lived and unpredictable launch times

Existing approaches

Serverless Frameworks

Machine Learning Frameworks

Existing approaches

Serverless Frameworks

corral PyWren faast.js

Limit. Pkg size

Download dependencies from S3

No fast storage

High-latency communication through S3

Unpred. launch

Stragglers

Machine Learning Frameworks

Existing approaches

<u>Serverless Frameworks</u>

corral PyWren faast.js

Limit. Pkg size — Download dependencies from S3

No fast storage — High-latency communication through S3

Unpred. launch — Stragglers

<u>Machine Learning Frameworks</u>

TensorFlow Apache Spark

Small mem. — Unable to launch runtimes in lambdas

No P2P comm. — No ring/tree reduces
No driver-to-worker comm.

Unpred. launch — Precludes MPI

Cirrus: a framework for serverless end-to-end ML workflows

Cirrus: design principles

1) Addressing serverless challenges

Low memory

Limited package size

No P2P communication

No fast storage

Short lifetimes and unpredictable launch

Ultra-lightweight runtime + data prefetching

High-perf. data store (parameter-server and KV)

Robust handling of lambda termination

Cirrus: design principles

2) Achieving benefits for end-to-end ML

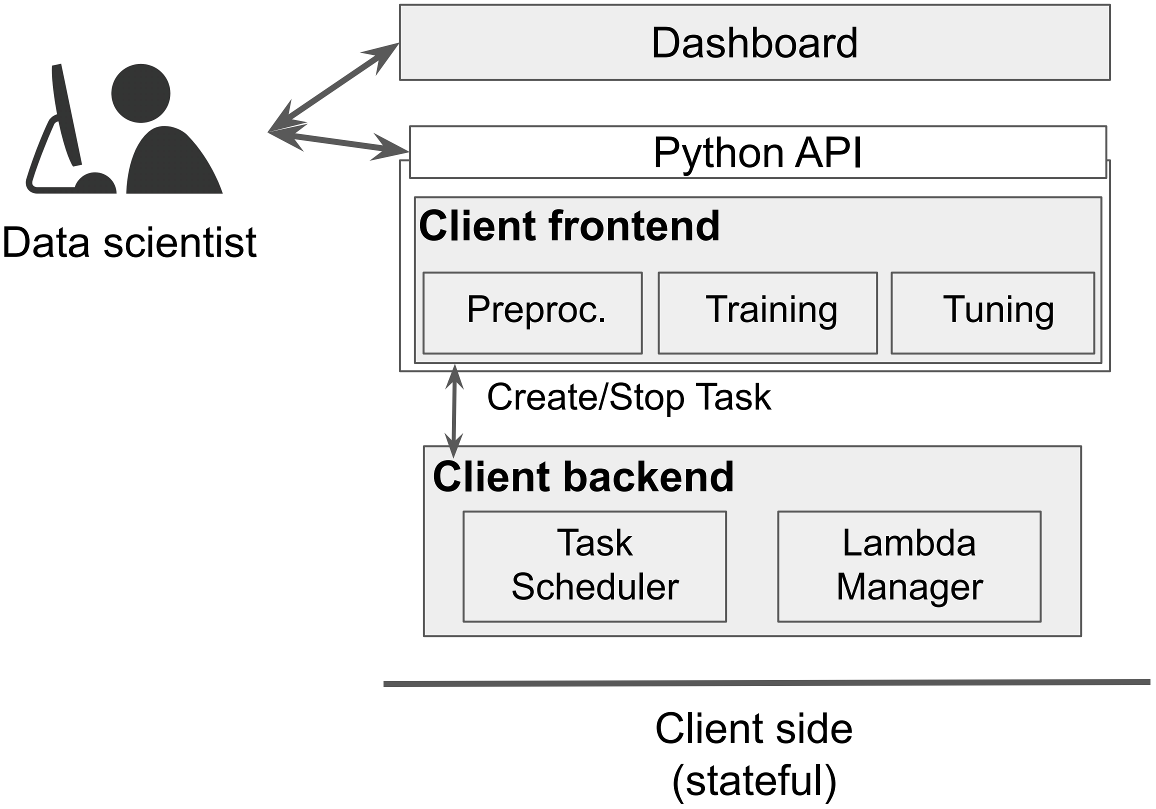
Tight provisioning of resources

Per-stage fine-grained variable agile scalability

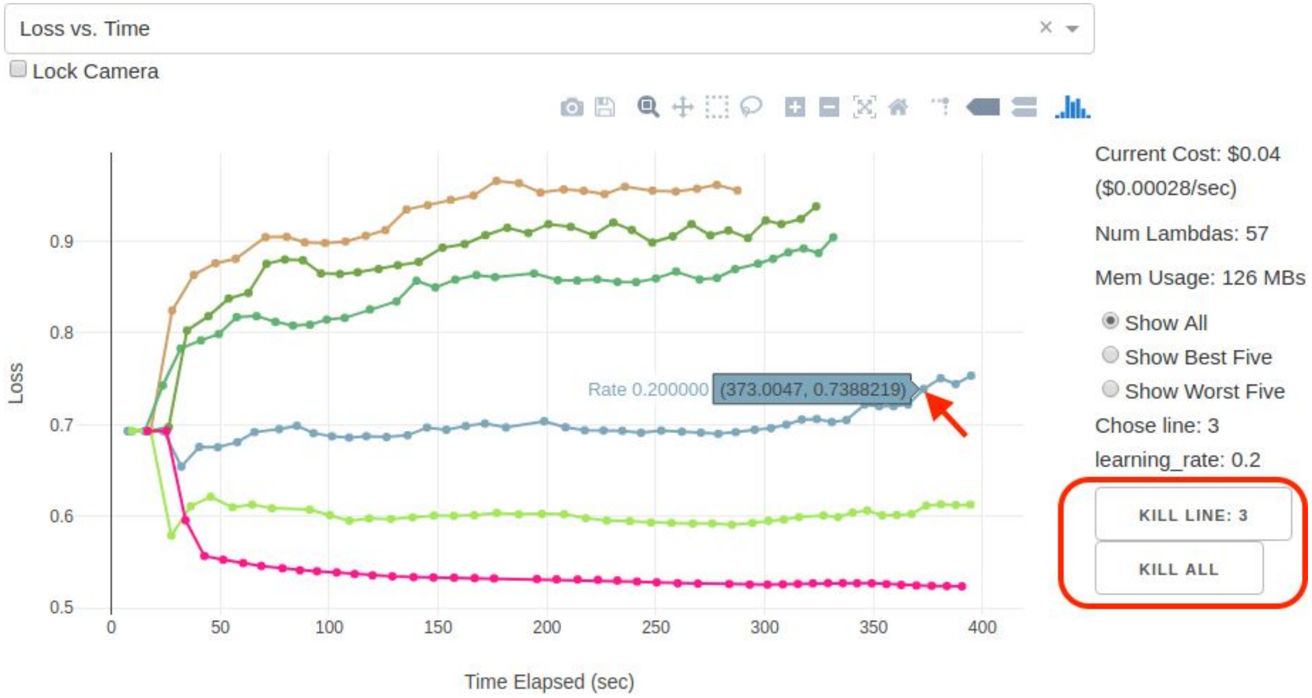
Simplifying infrastructure management

High-level API supports end-to-end ML

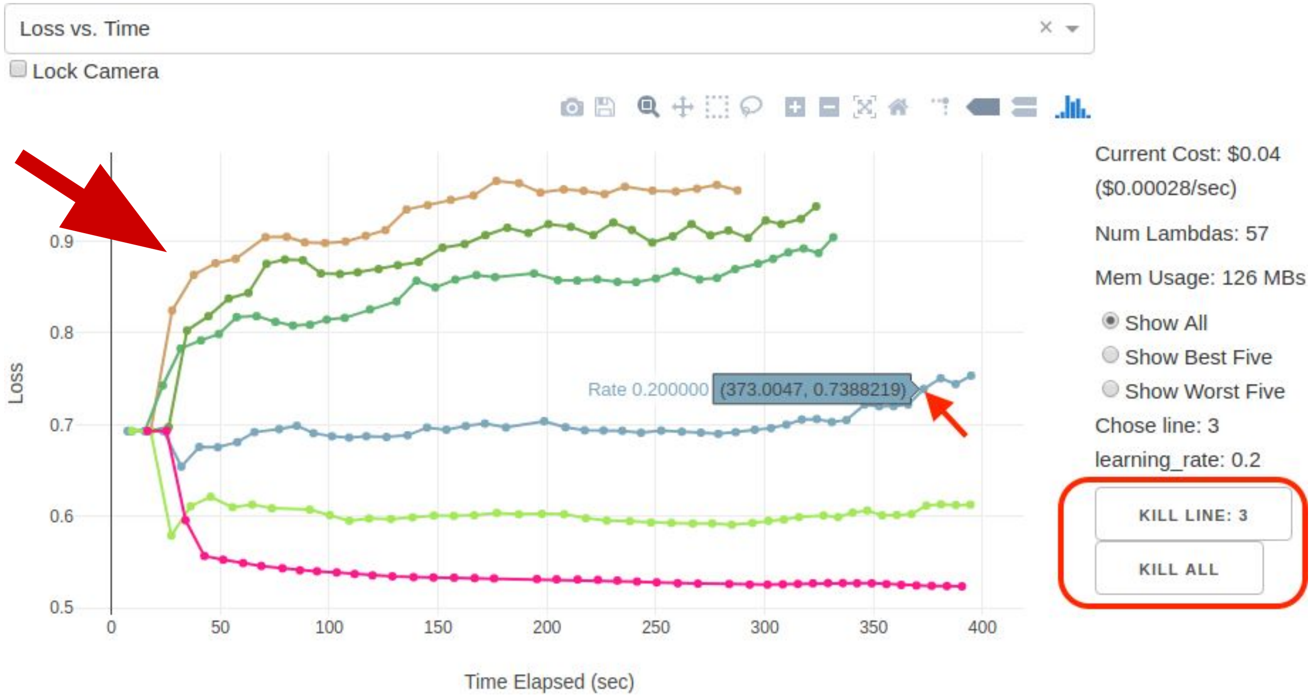
Cirrus architecture (client side)



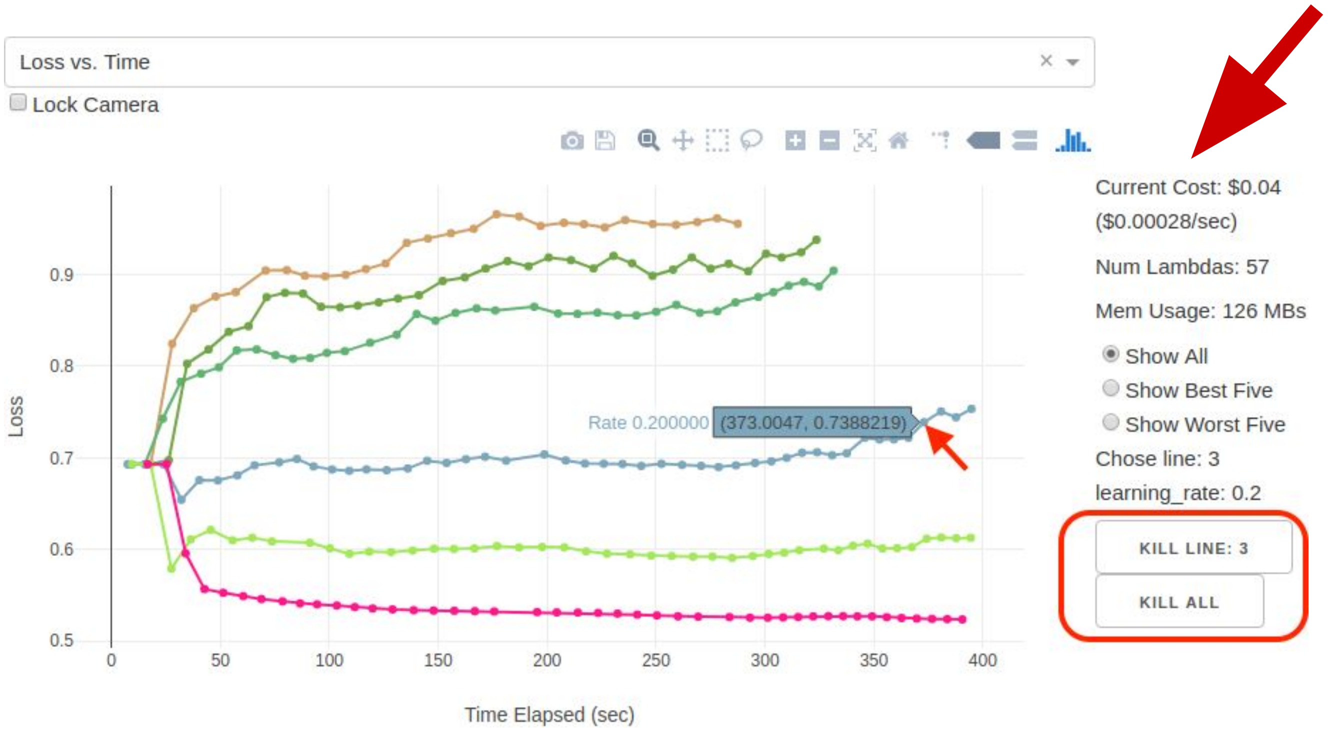
Cirrus Dashboard



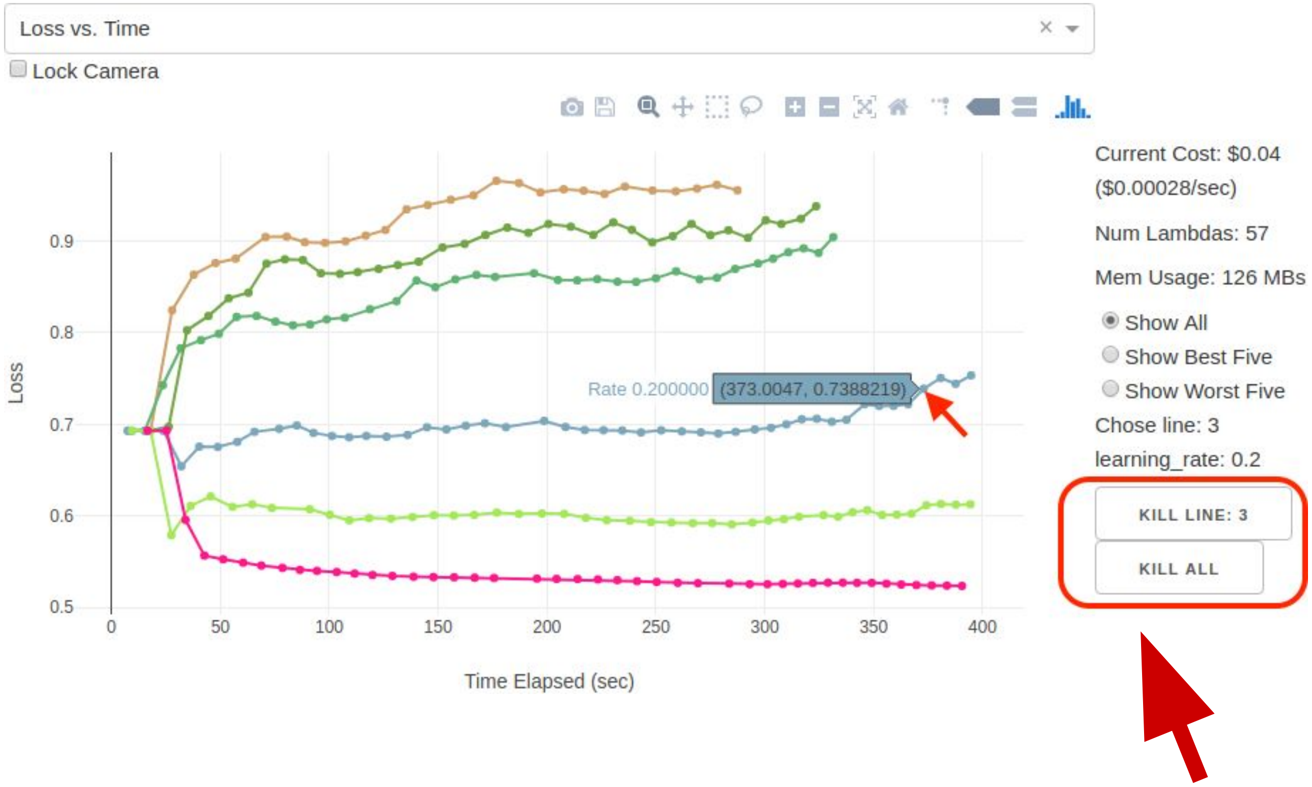
Cirrus Dashboard



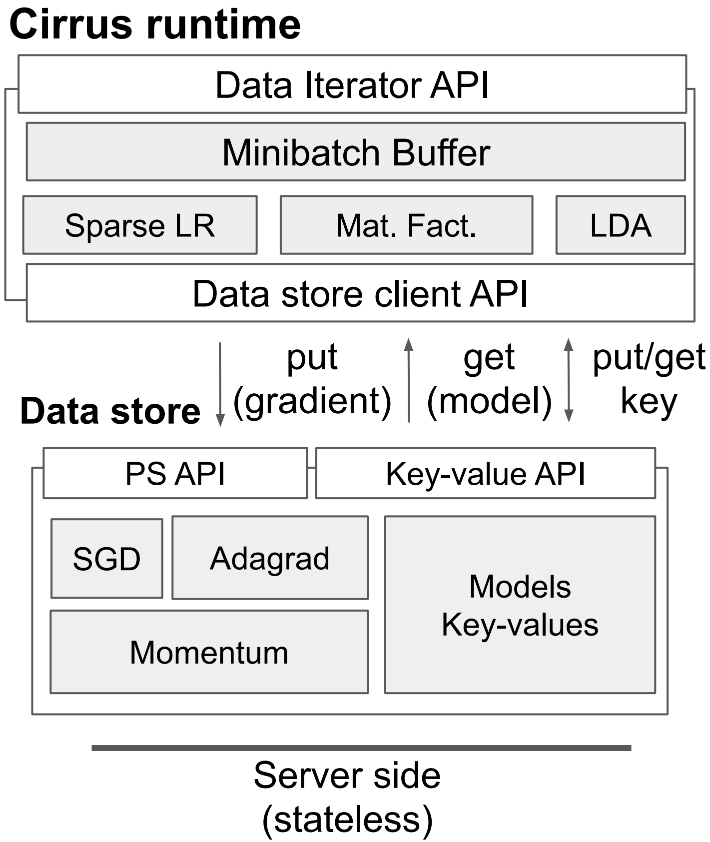
Cirrus Dashboard



Cirrus Dashboard



Cirrus architecture (server side)



Cirrus evaluation

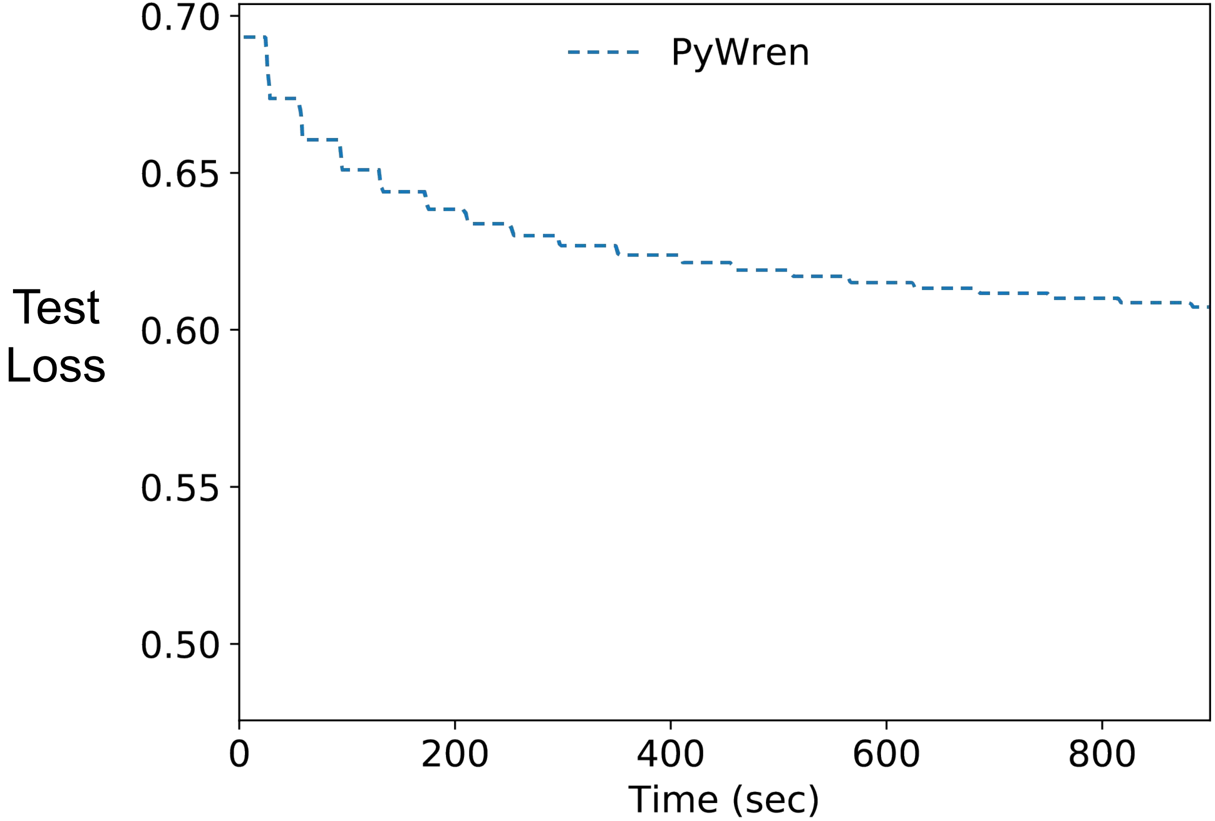
1. Cirrus provides benefits by specializing both for serverless and end-to-end ML

2. We show that Cirrus outperforms a state-of-the-art serverless system: PyWren

Evaluation setup

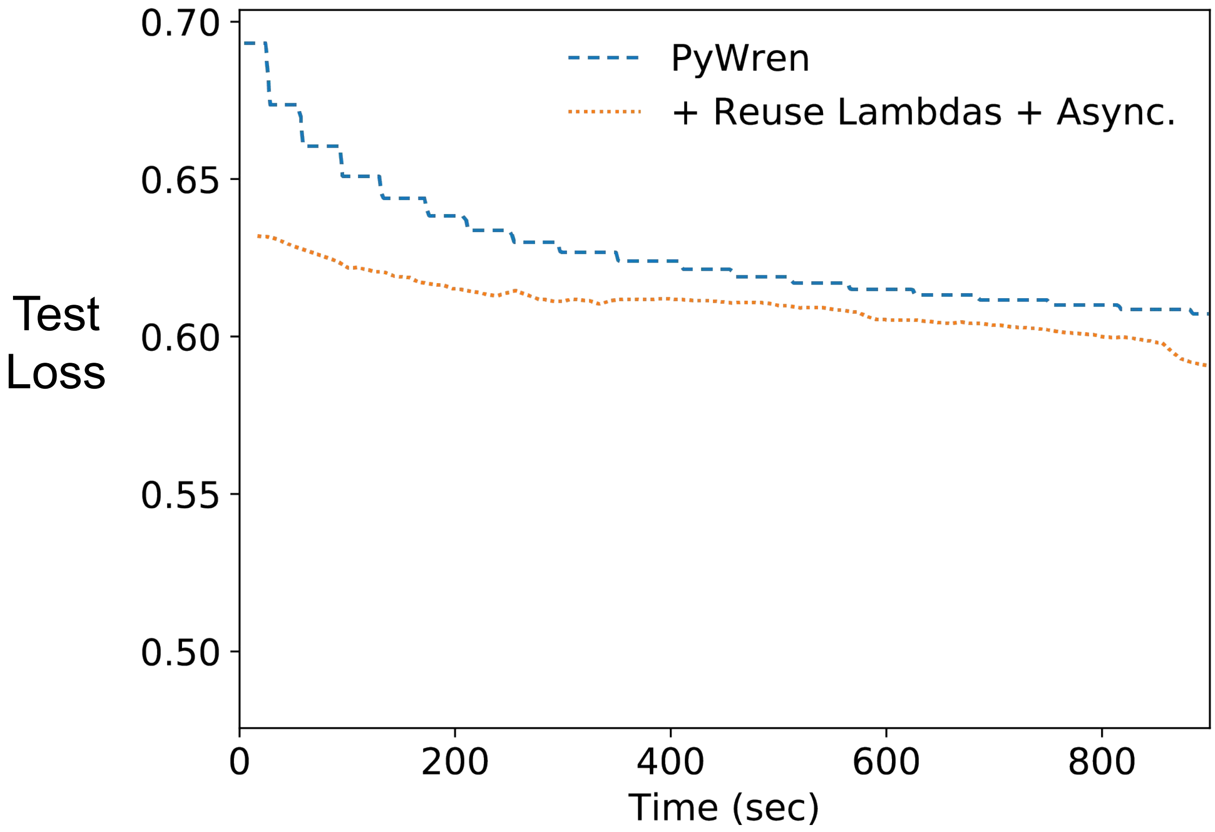
1. Deployment: AWS Lambdas (3GB of mem.)
2. Benchmark: async. distributed SGD Sparse Logistic Regression task
3. Dataset: Criteo Dataset (a dataset of display ads)
4. PyWren:
    a. Baseline: iterative synchronous SGD training using AWS S3 to store gradients and model
    b. + 3 incremental optimizations
5. Cirrus: 2 modes (with/without prefetching)

Cirrus outperforms vanilla serverless



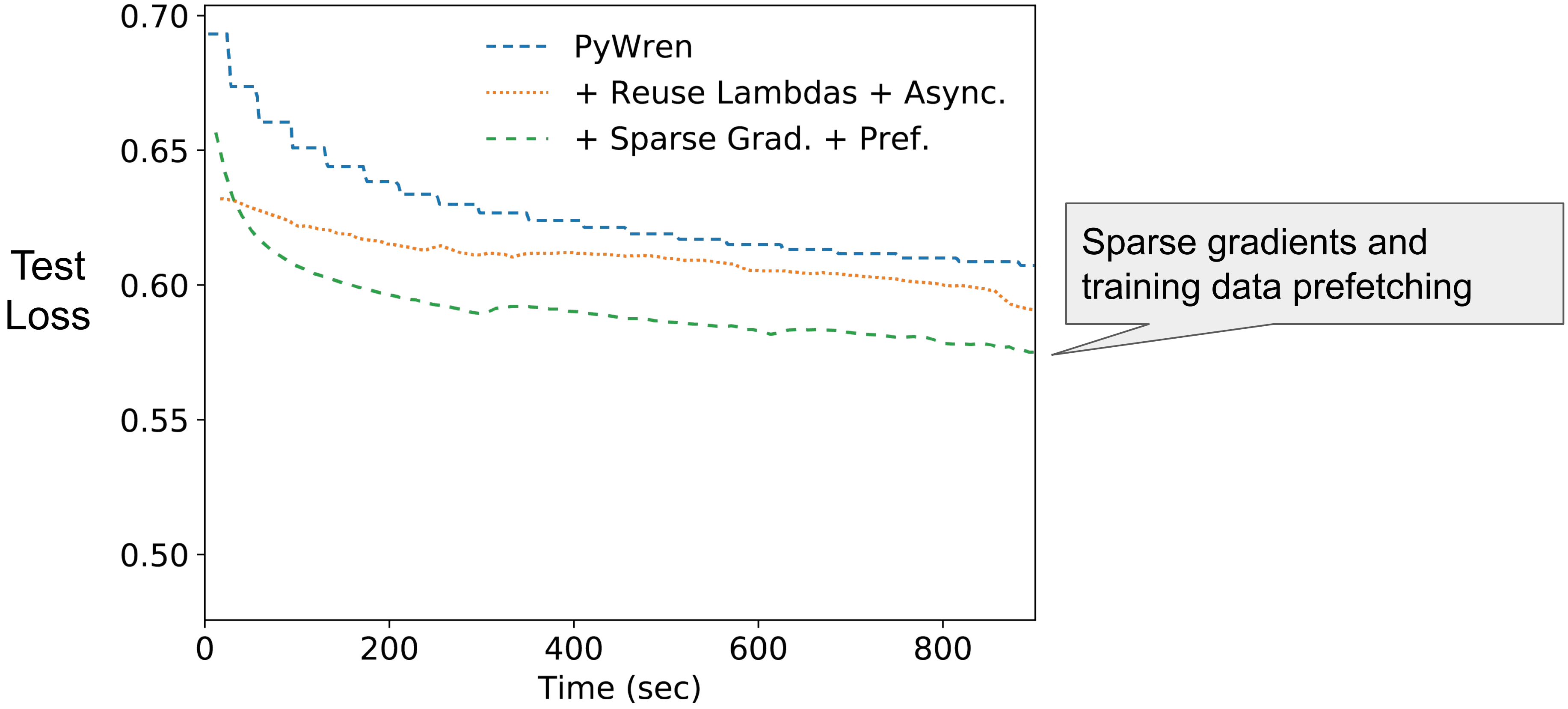
Synchronous SGD training suffers from stragglers

Cirrus outperforms vanilla serverless

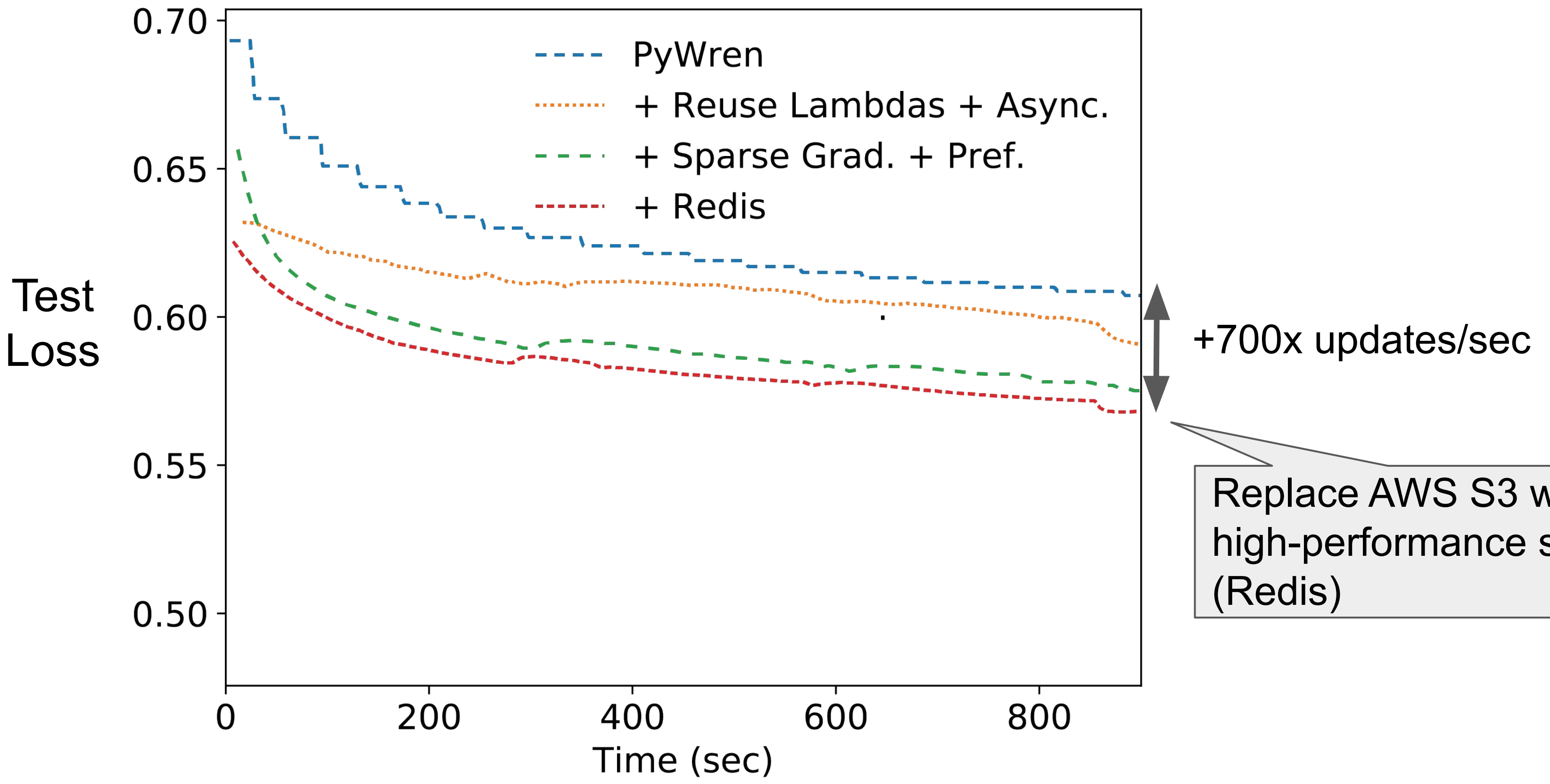


- Multiple SGD iterations on each lambda invocation
- Asynchronous SGD

Cirrus outperforms vanilla serverless

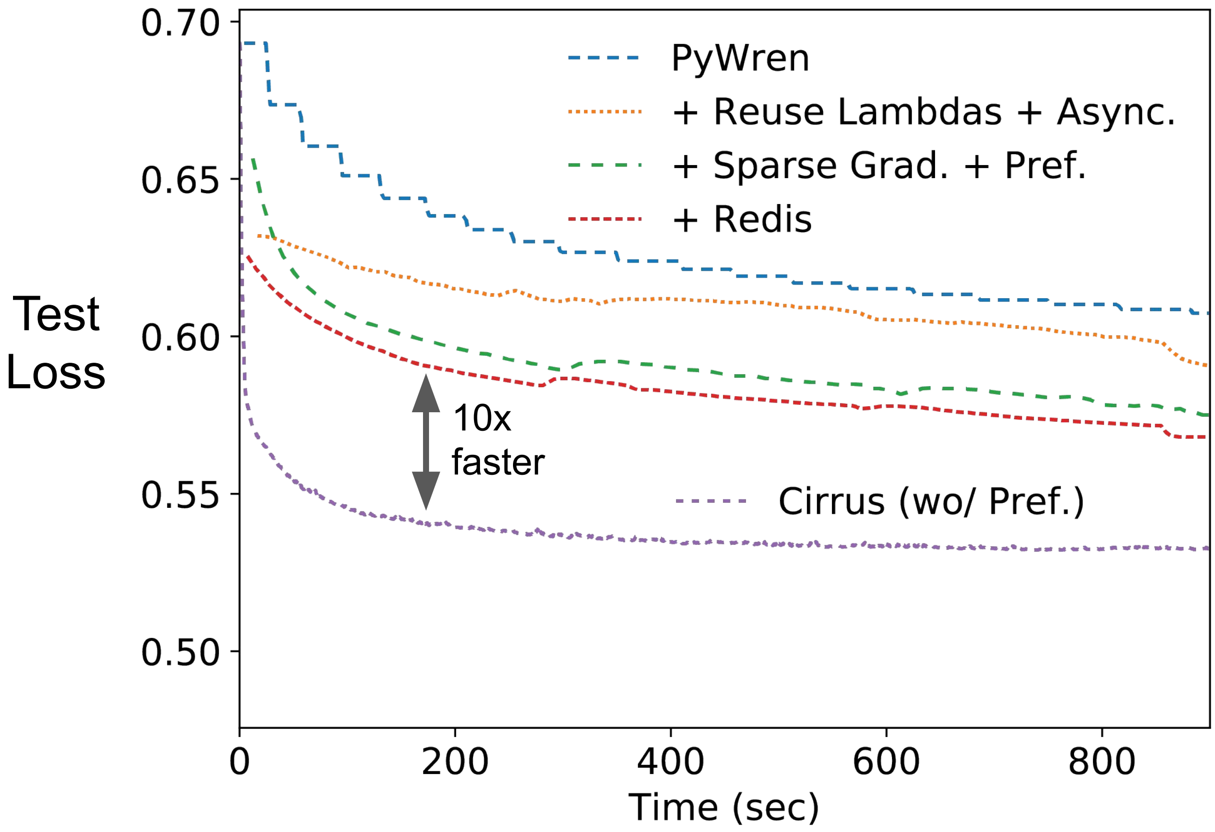


Cirrus outperforms vanilla serverless



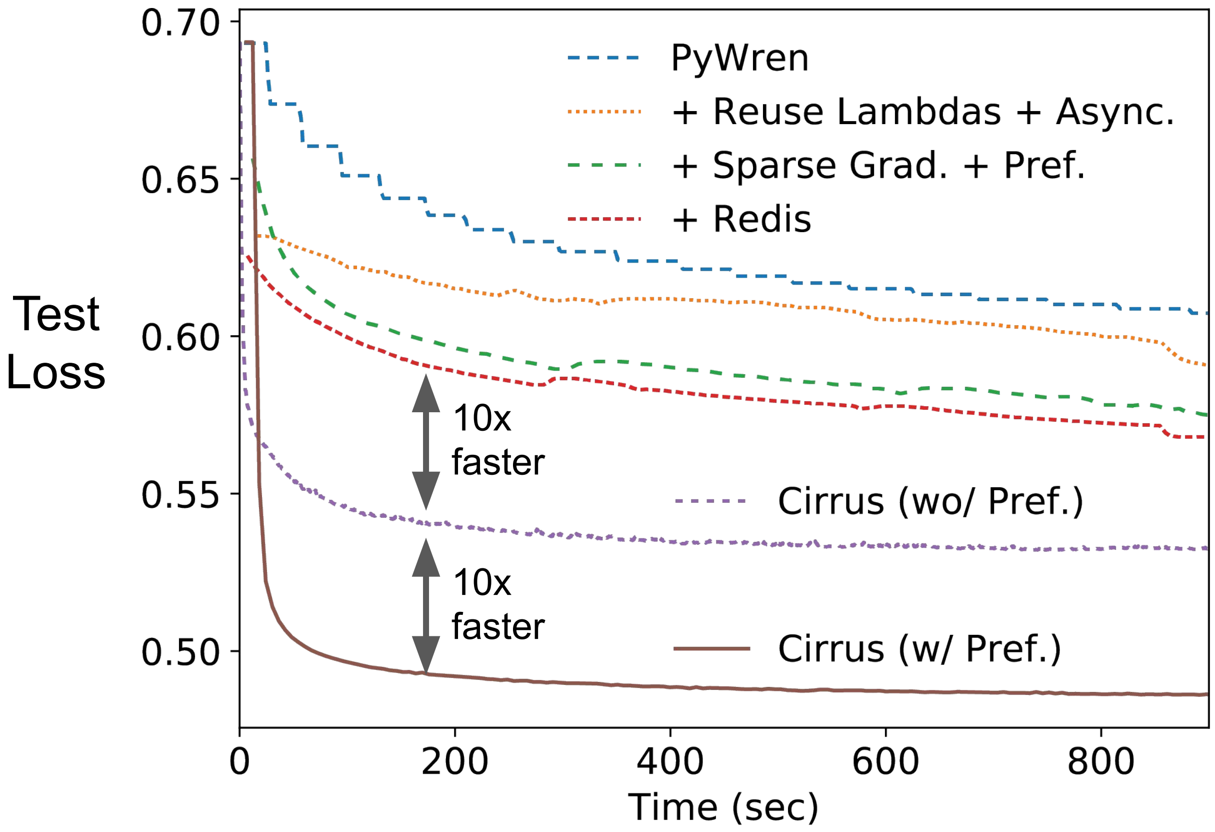
Replace AWS S3 with high-performance store (Redis)

Cirrus outperforms vanilla serverless



Cirrus without training data prefetching

Cirrus outperforms vanilla serverless



Cirrus with training data prefetching

Conclusion

1. End-to-end ML workflows:
   a. time-consuming infrastructure management
   b. resource overprovisioning
2. Cirrus -- serverless end-to-end ML framework:
   a. simplify deployment of ML workflows
   b. per-stage provisioning of resources
3. Cirrus outperforms existing serverless solutions by specializing for <u>serverless</u> and <u>ML</u>

Thank you!

github.com/ucbrise/cirrus

@jccarreira