INFOCOM'19, Paris, FR

Distributed Machine Learning with a Serverless Architecture

Hao Wang[1], Di Niu[2], Baochun Li[1]

[1]University of Toronto, [2]University of Alberta

UNIVERSITY OF TORONTO

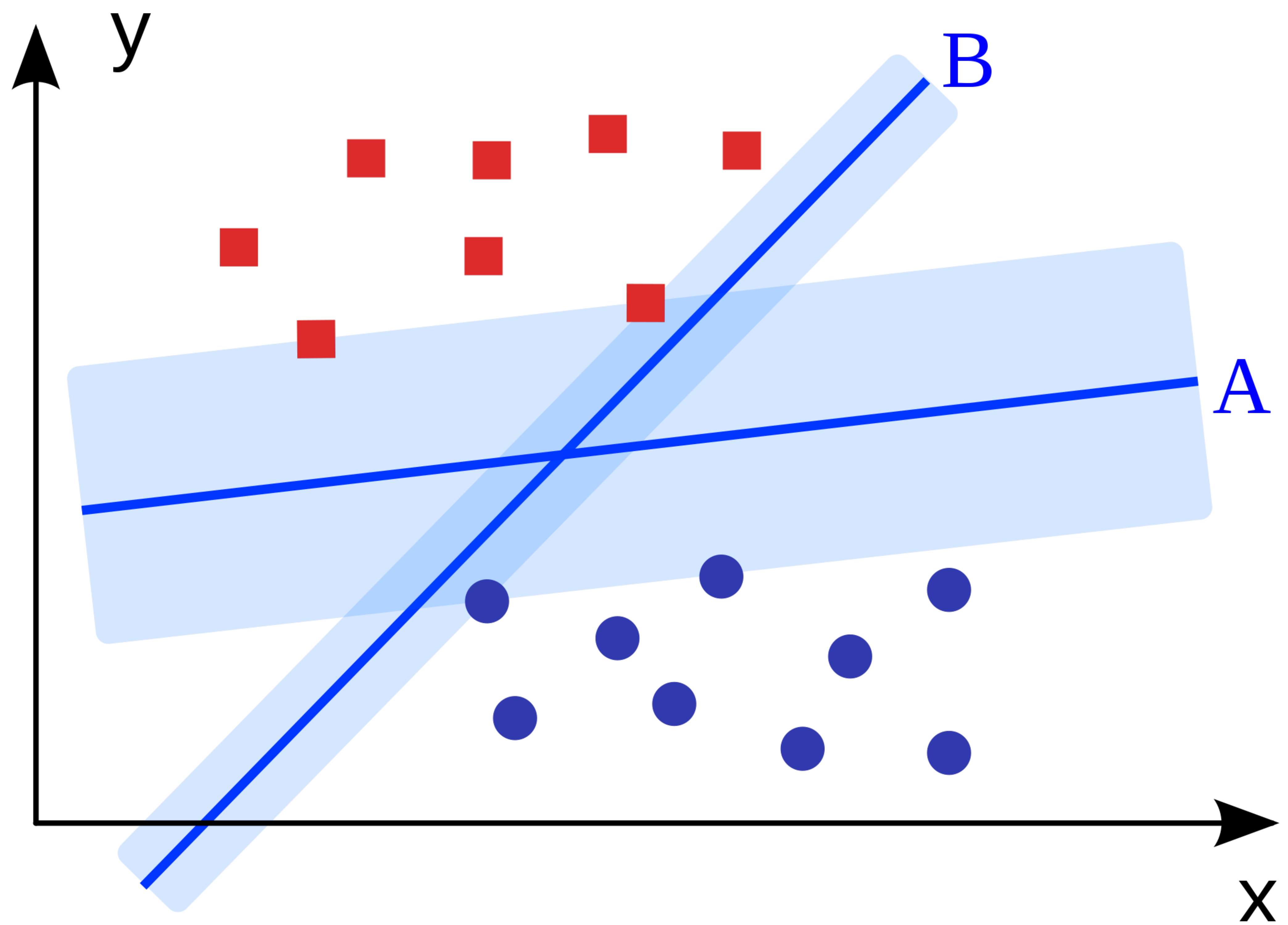
Camera 1
20:14:22
03/31/2017
Males
Females
amazon
harman/kardon
Neutral
UnHappy
Happy

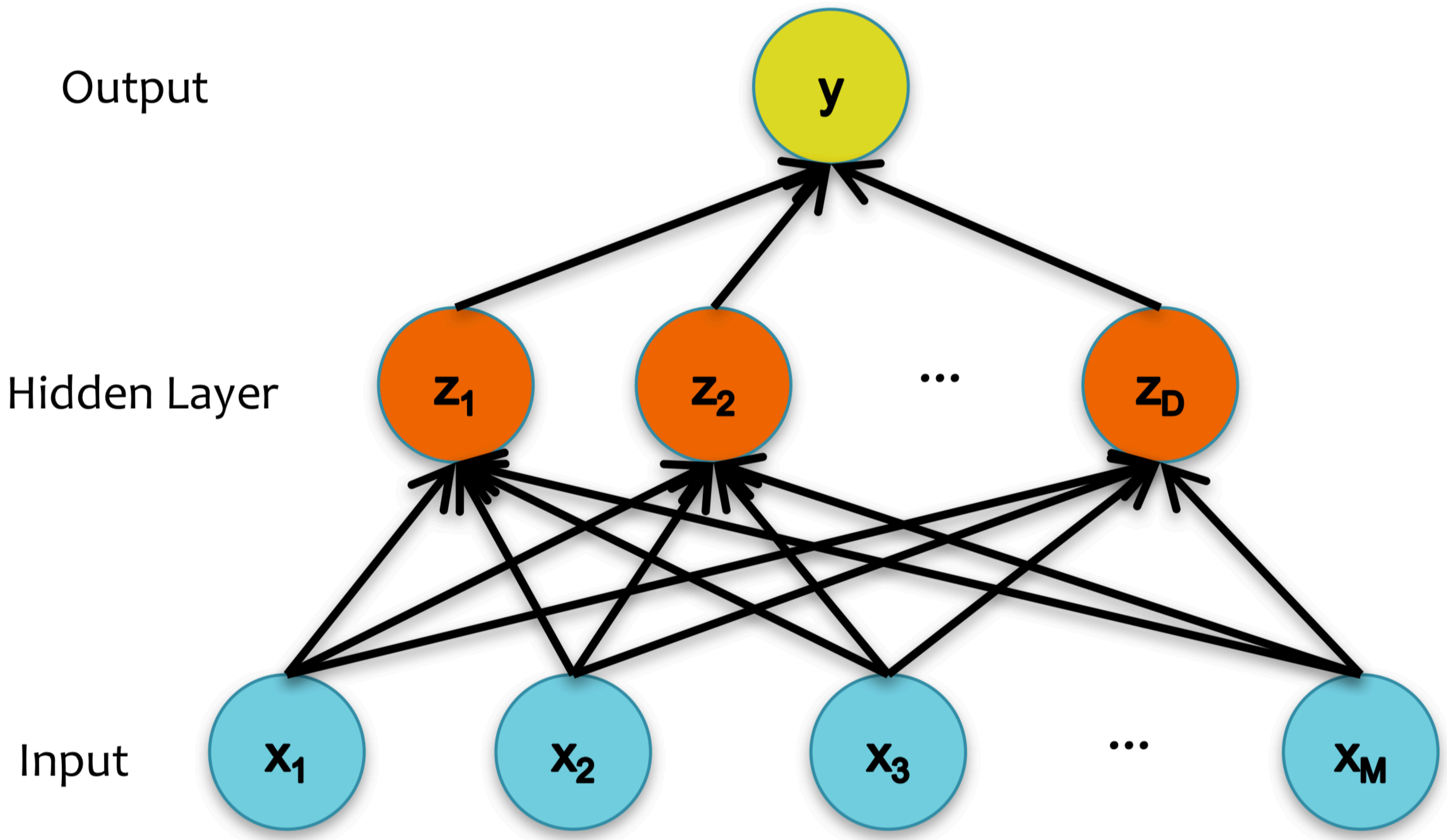
What is machine learning?

Deep Learning

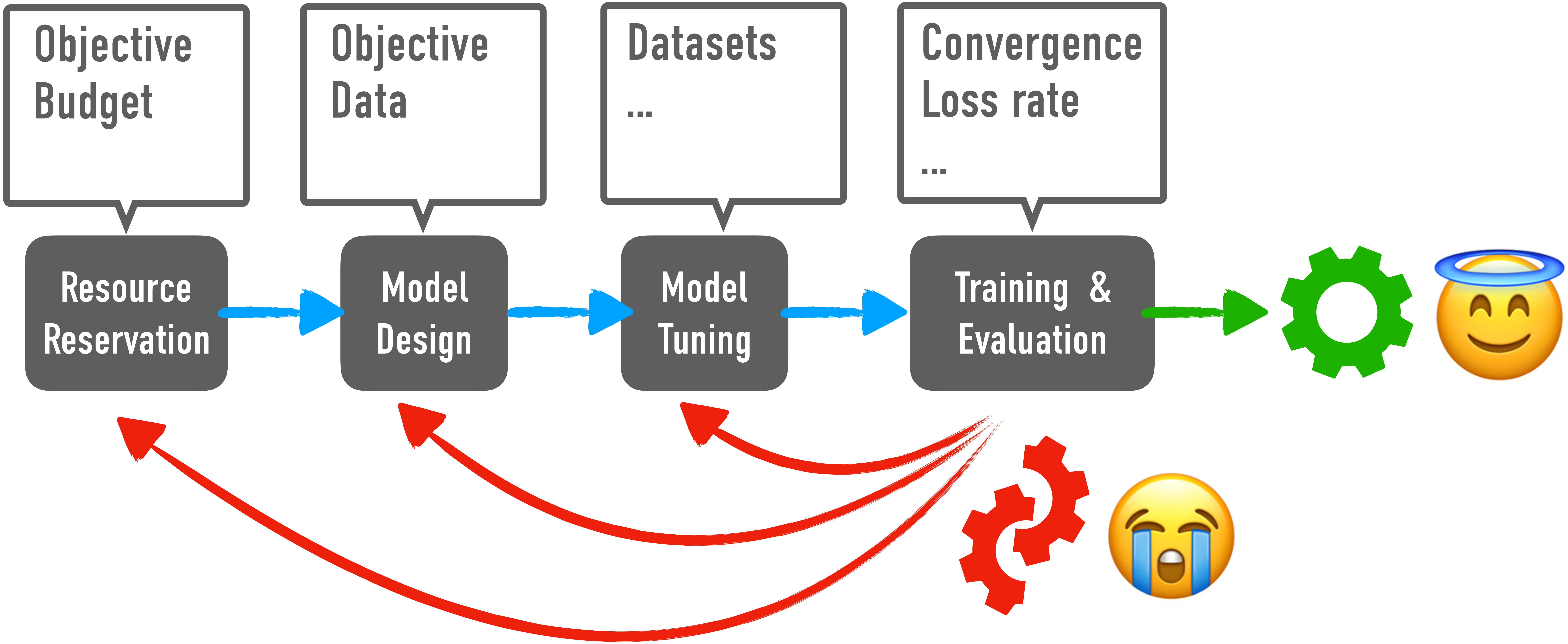
Machine Learning

Numerical optimization → Gradients



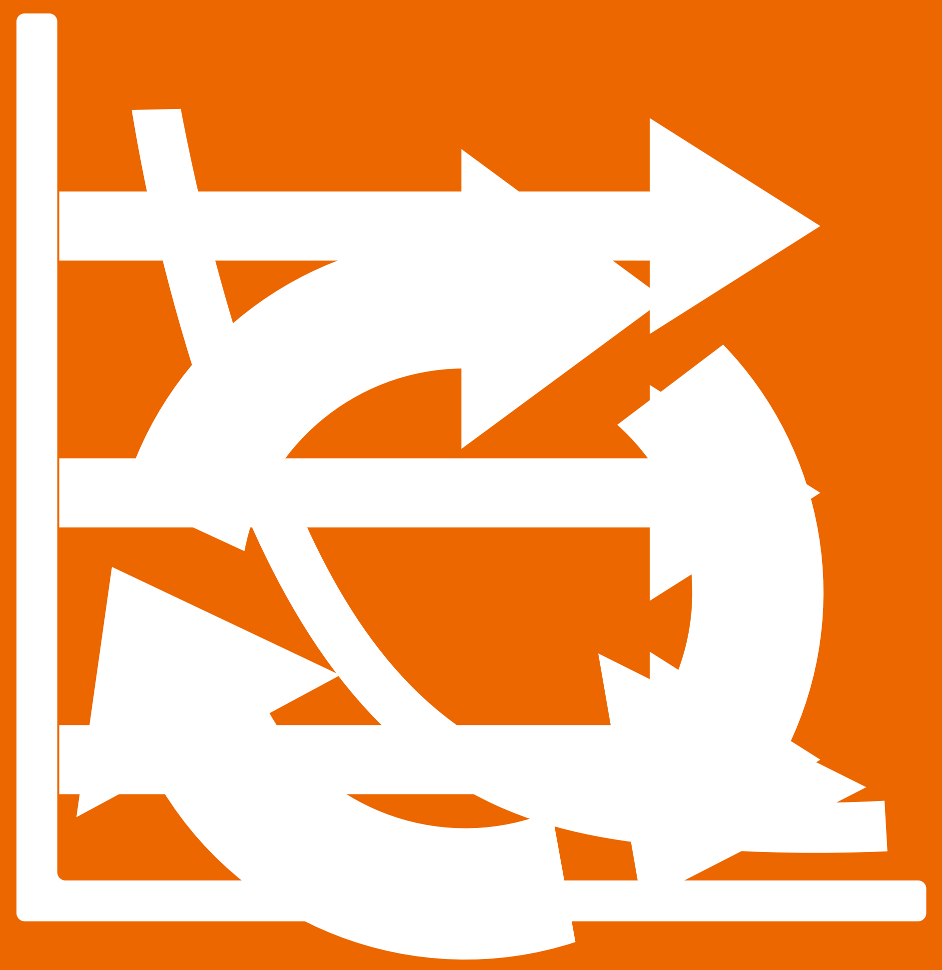


ML Workflow


Objective
Budget
Objective
Data
Datasets
...
Convergence
Loss rate
...
Resource Reservation
Model Design
Model Tuning
Training & Evaluation

Our Key Insights

- Most current ML training jobs are **data parallel**
- Model quality and resource investment have a **nonlinear** relation
- ML training is inevitably a **trial-and-error** process



Distributed ML Infrastructure

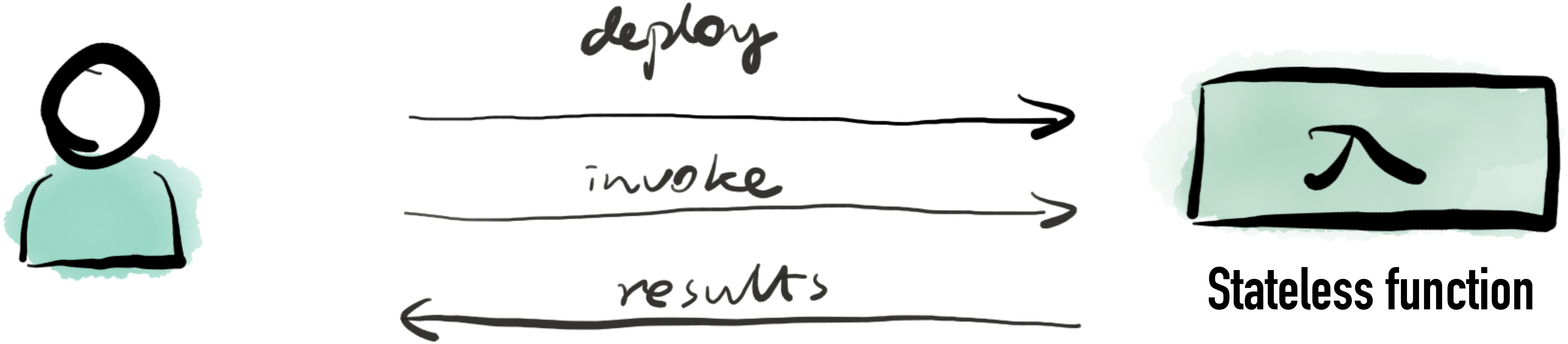
	IaaS	PaaS
Pricing	Per hour	Per hour
Maintanance	By users	By providers
Examples	AWS EC2, Google Cloud Compute ...	Azure ML Studio, Google Cloud ML Engine ...

Serverless?

	IaaS	PaaS	Serverless
Pricing	Per hour	Per hour	Per call
Maintanance	By users	By providers	By providers
Examples	AWS EC2	Azure ML Studio	AWS Lambda

Serverless Computing?

- Only input and output, no intermediate states



Go Serverless?

Pro:

1. Flexible concurrency
2. Instant response
3. Easy to deploy
4. Cheap? Runtime * MemSize

Con:

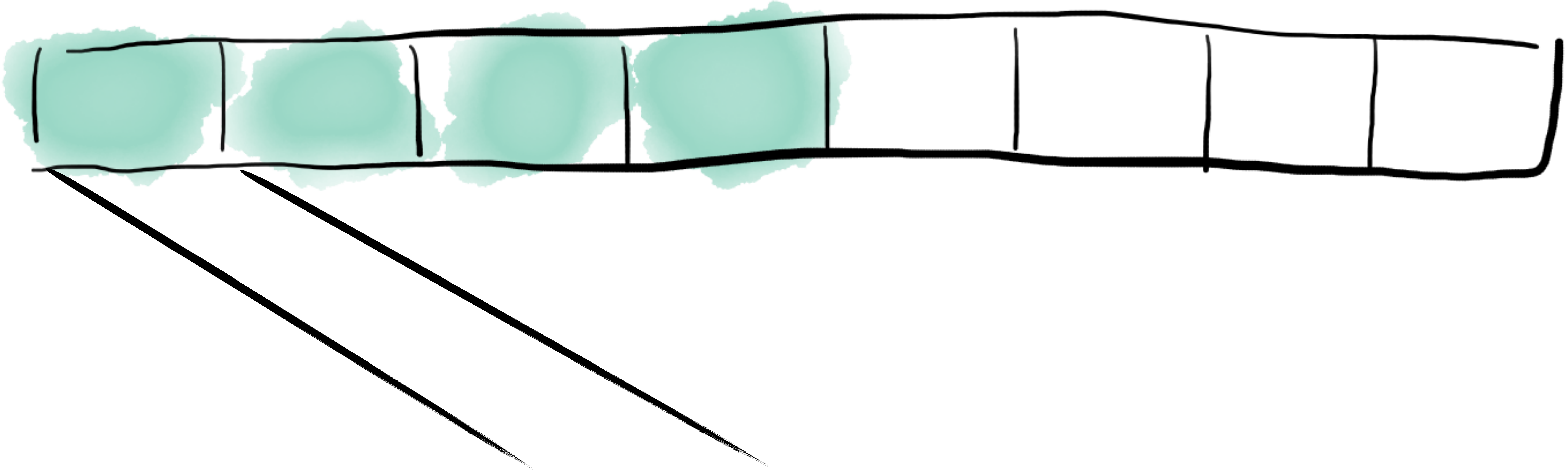
1. Execution model is too simple
2. Runtime limitations (~15min)
3. Communication overhead

ML Training on Serverless?

- MapReduce on Serverless Cloud (PyWren, [SoCC'17])
- Video processing on Serverless Cloud (Sprocket [SoCC'18])

Stochastic Gradient Descent (SGD)

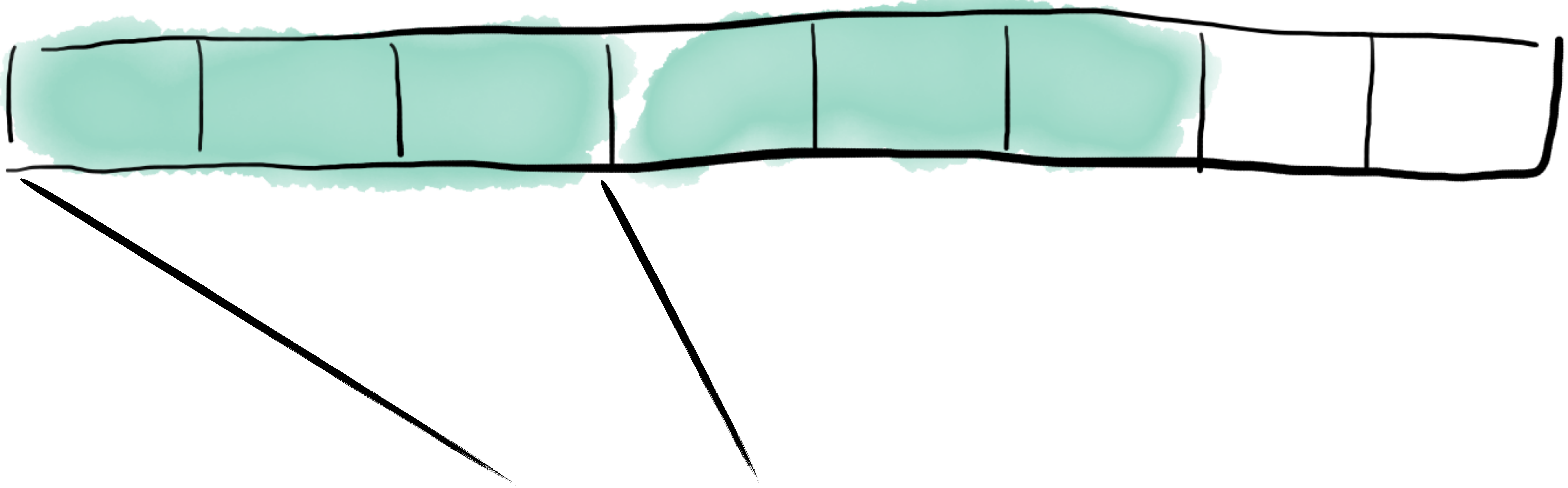
Input Samples



$$\theta_j = \theta_j + \alpha(y^i - h_\theta(x^i))x^i{}_j$$

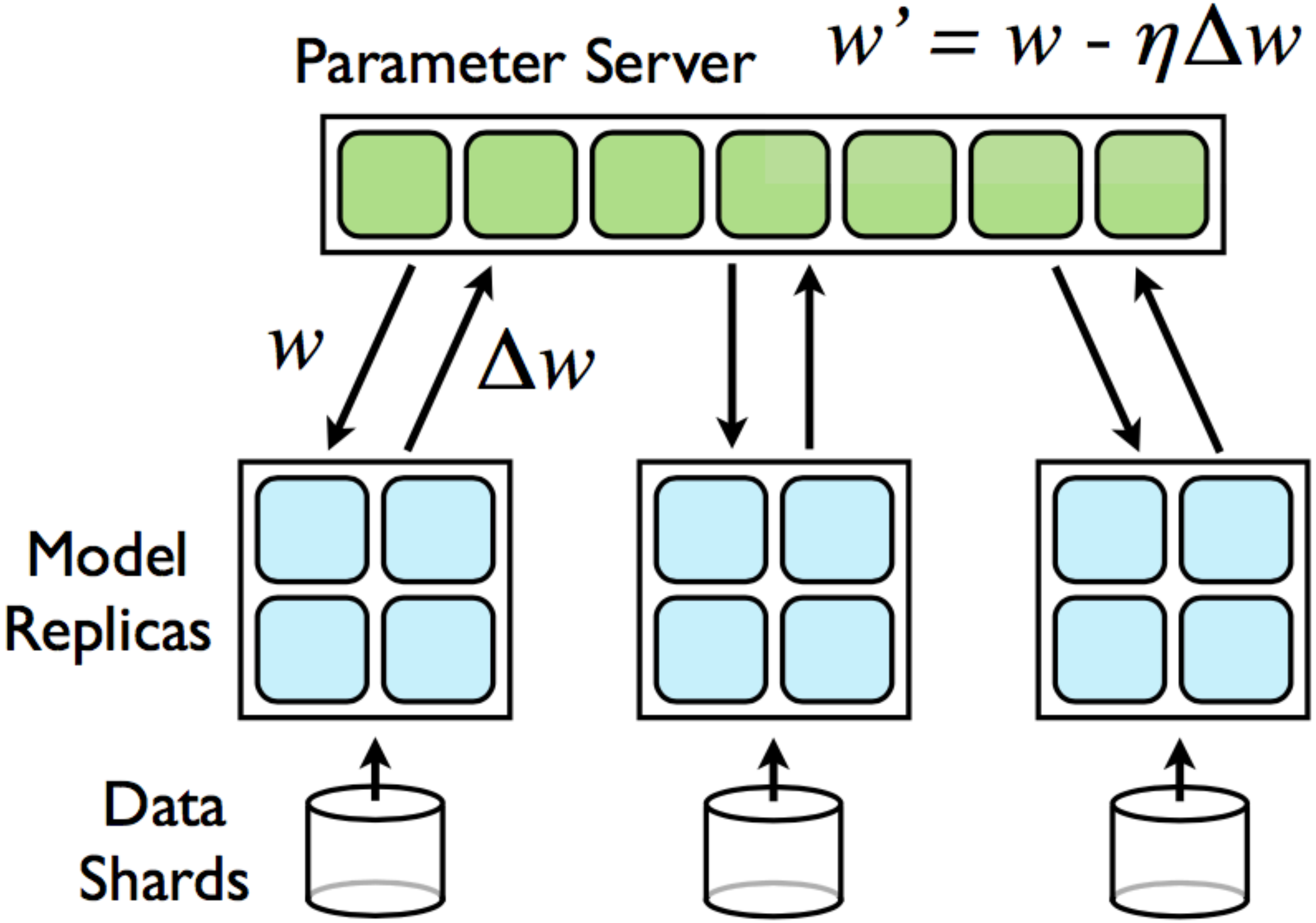
Mini-batch SGD

Input Samples



$$\theta_j = \theta_j + \frac{a}{b} \sum_{k=i}^{i+b-1} (y^k - h_\theta(x^k)) x_j^k$$

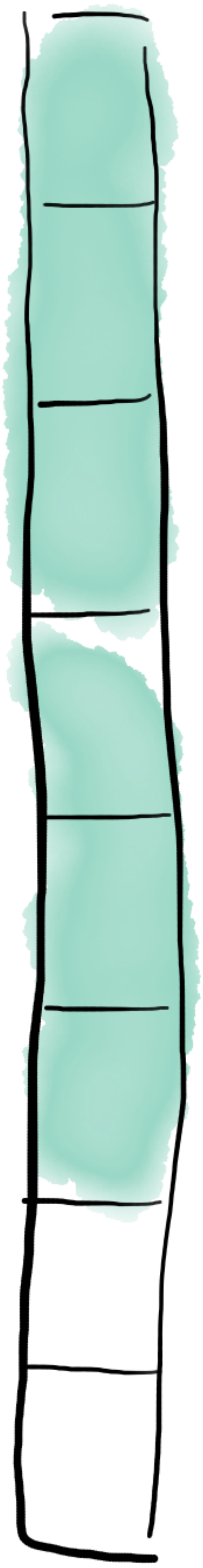
Parameter Server



- Model replicas on workers
- Servers update parameters

Li, Mu, et al. Scaling distributed machine learning with the parameter server. OSDI'14

SGD on Lambda



Input Samples

Function

$$\theta_j = \theta_j + \frac{a}{b}\sum_{k=i}^{i+b-1}(y^k - h_\theta(x^k))x_j^k$$

Function

$$\theta_j = \theta_j + \frac{a}{b}\sum_{k=i}^{i+b-1}(y^k - h_\theta(x^k))x_j^k$$

⋮

Function

$$\theta_j = \theta_j + \frac{a}{b}\sum_{k=i}^{i+b-1}(y^k - h_\theta(x^k))x_j^k$$

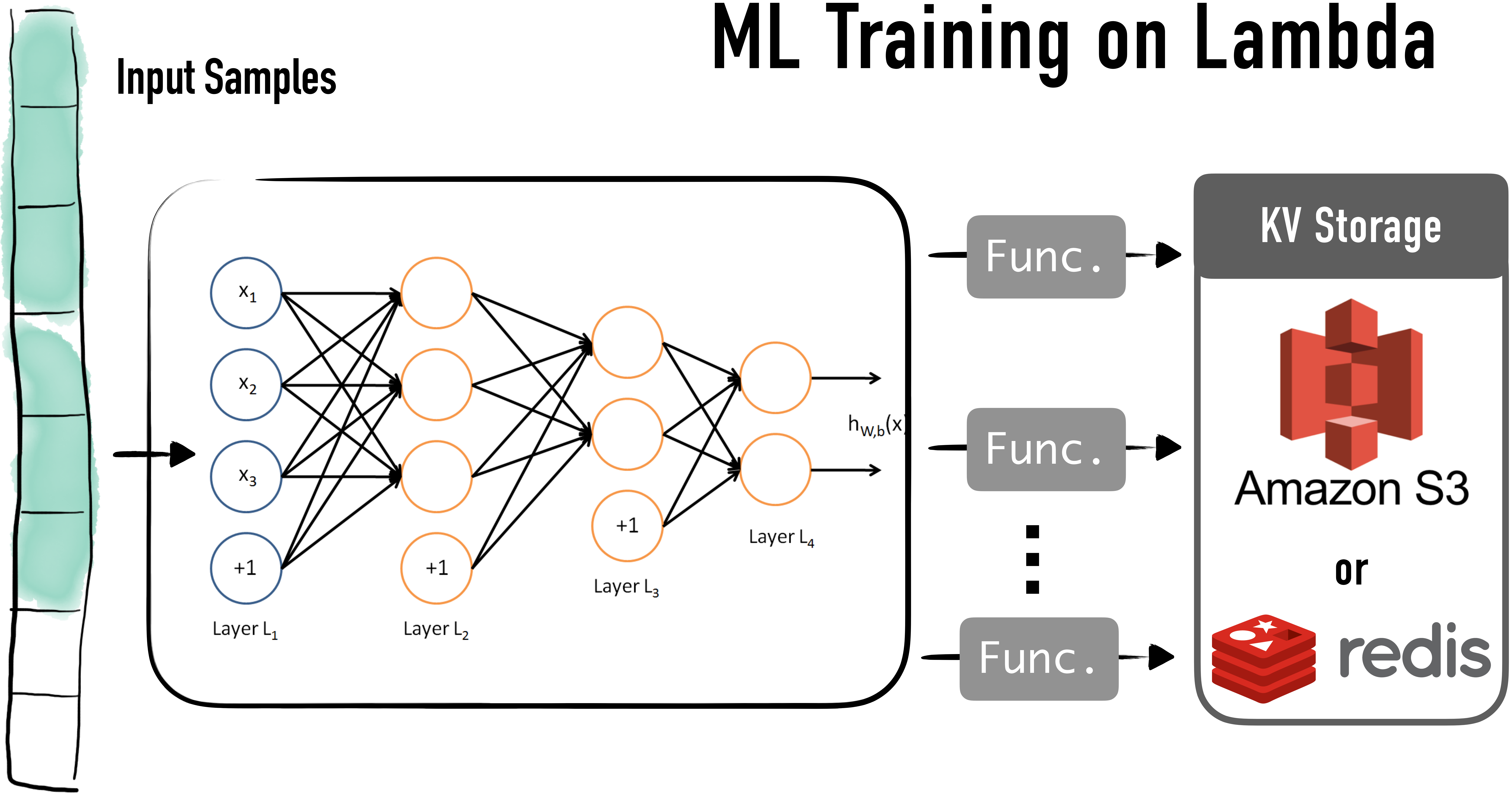
KV Storage

Amazon S3

or

redis

ML Training on Lambda



Toy Example

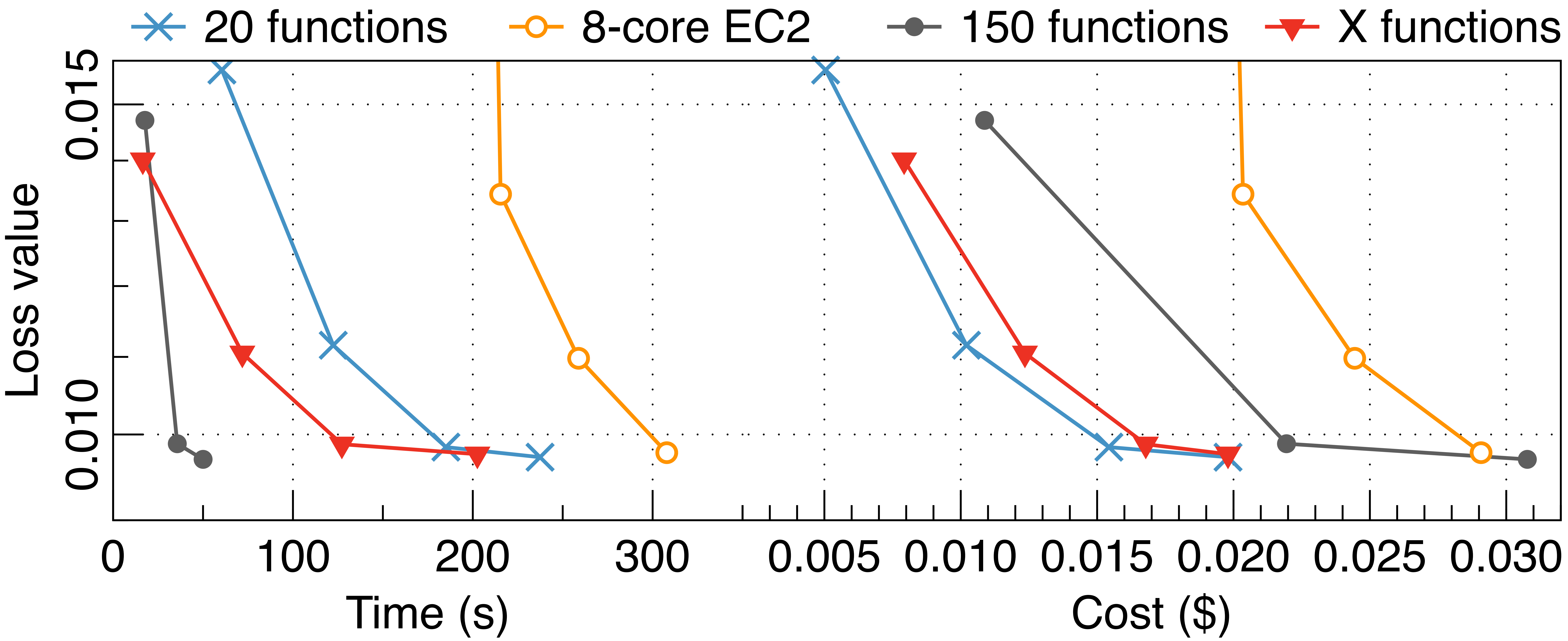
- **Workload**
  - A logistic regression model

- **AWS Lambda**
  - 20 functions
  - 150 functions
  - X functions (dynamic # of func.)
  - S3 storage

- **EC2 c5.2xlarge**
  - 8 CPUs, 16GB mem
  - Local storage

Toy Example

- Loss value v.s. training time
- Loss value v.s. monetary cost



Toy Example

	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

Slowest, no cheap
Fastest, expensive
Fast, cheap

X functions:

- The first epoch: 120 functions
- The last epoch: 10 functions
- Intermediate epochs: 20 functions

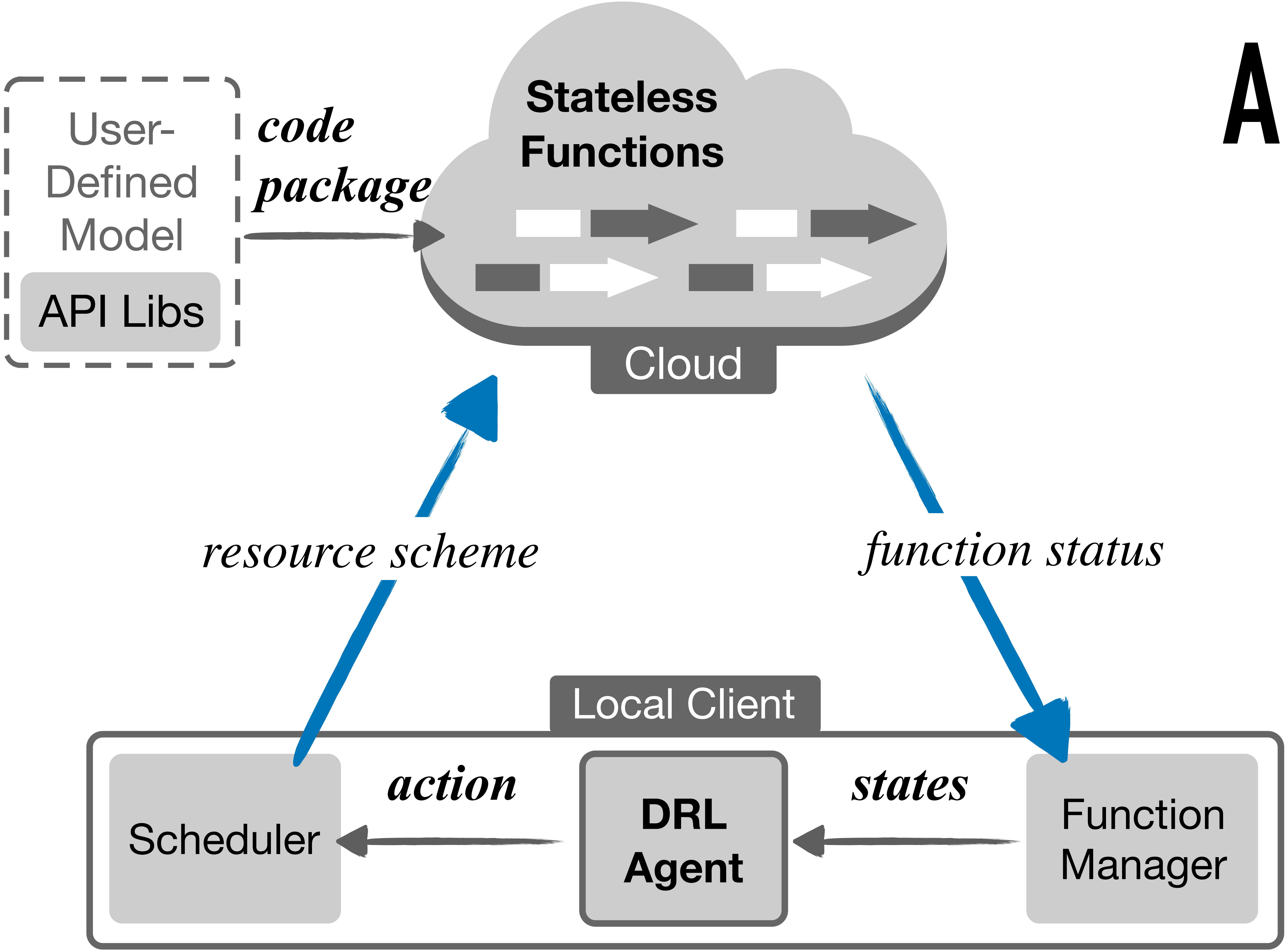
Challenges

- **Functions on Serverless**
  - Limitation on performance and deployment
- **Dynamic Resource Provisioning**
  - Speed v.s. cost (given a budget, how fast could be?)

Siren

- Hybrid Synchronous Parallel (HSP)
- Experience-Driven Resource Scheduler

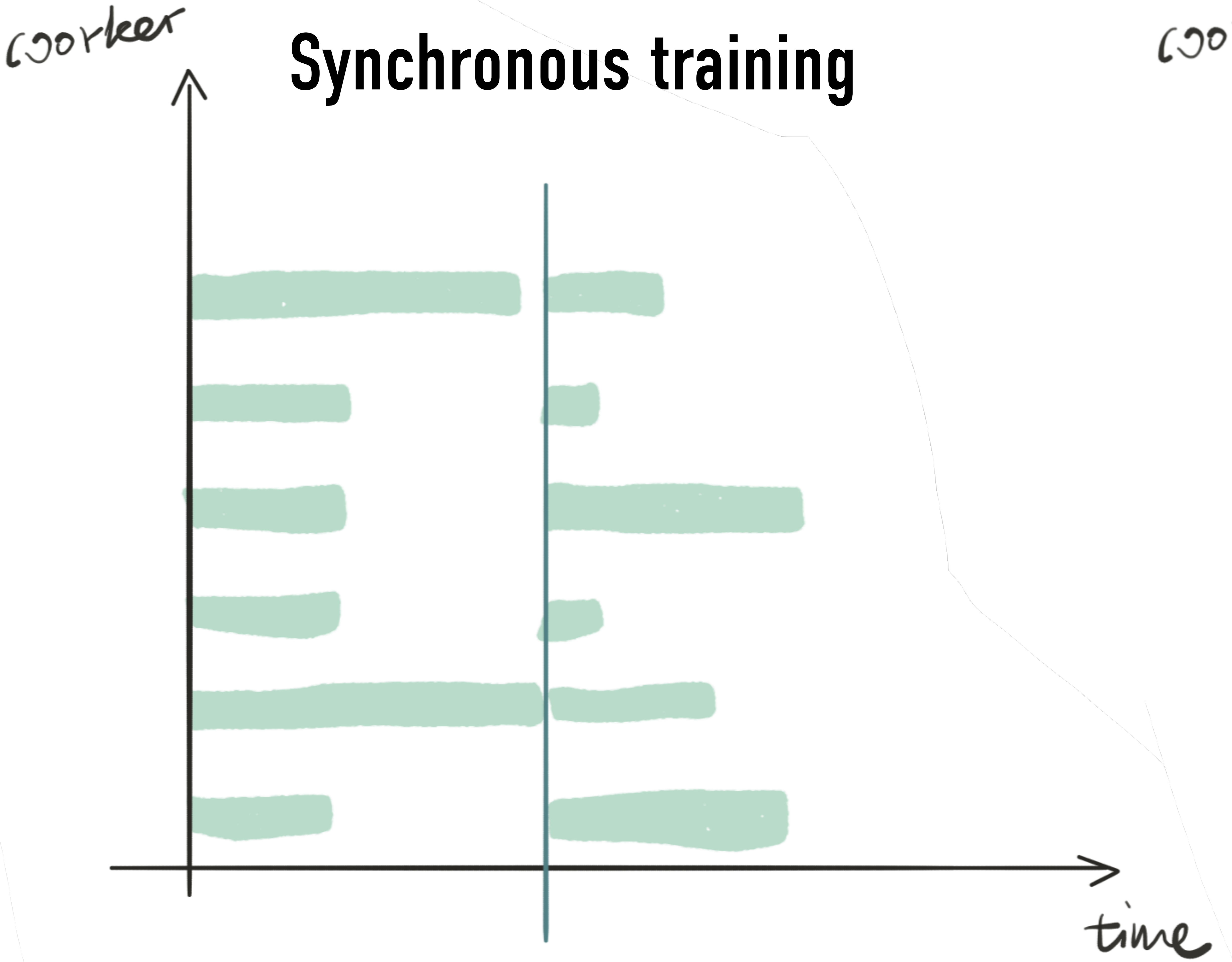
Architecture



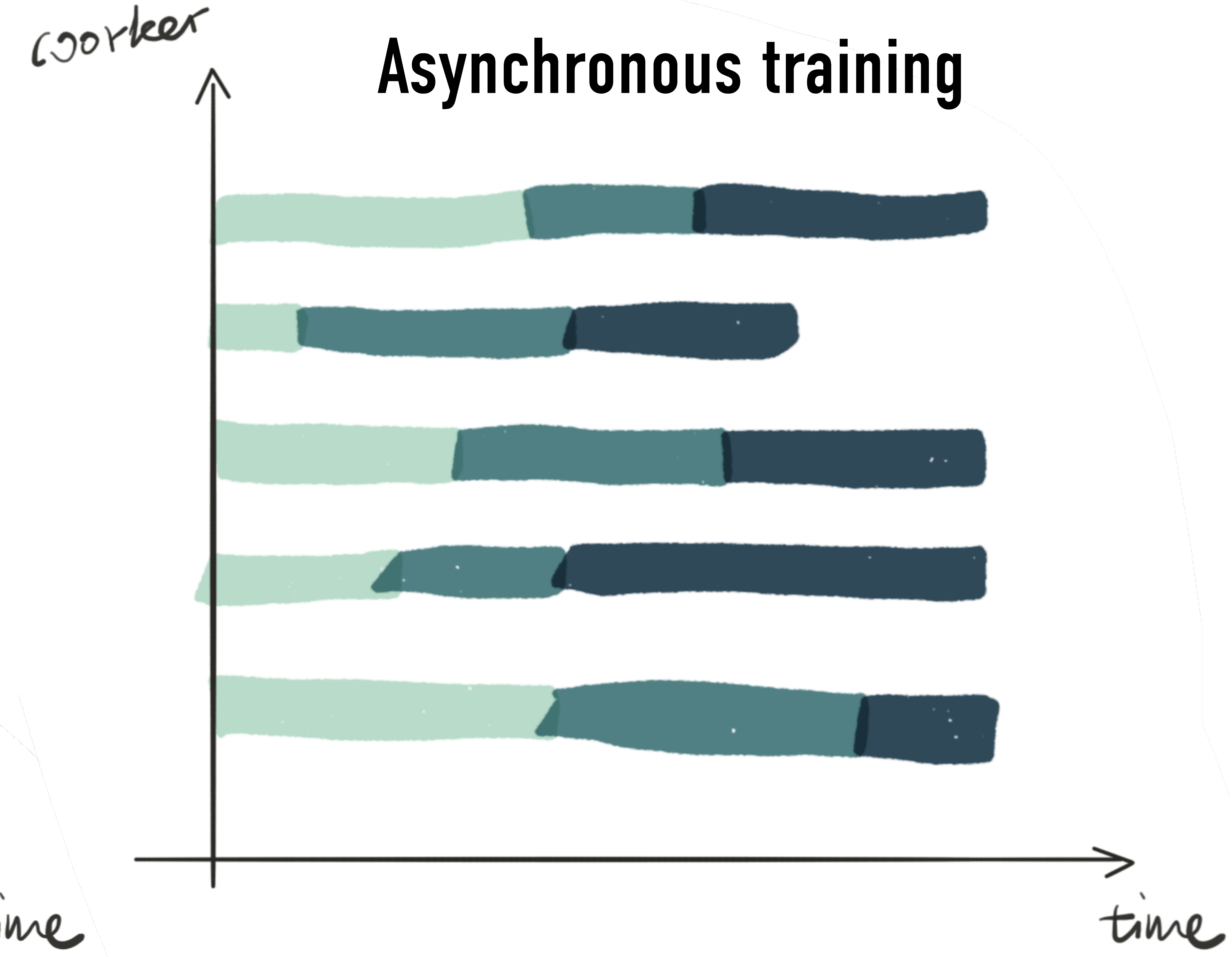
Enforce Parallelism on Siren

Synchronous or Asynchronous

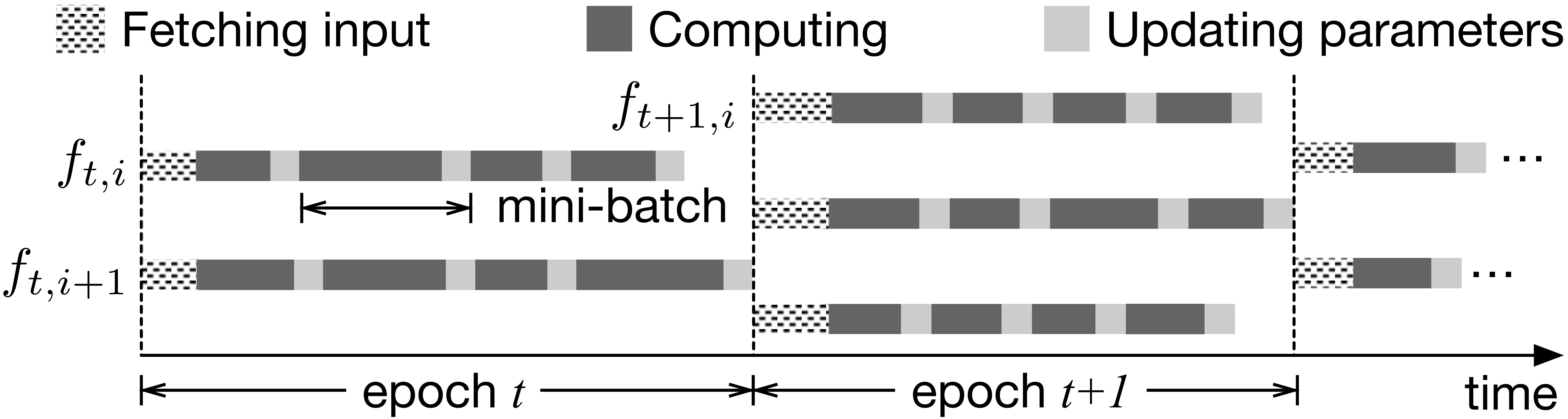
Synchronous training



Asynchronous training



Hybrid Synchronous Parallel (HSP)



Experience-Driven Scheduler

Toy Example - Find the X

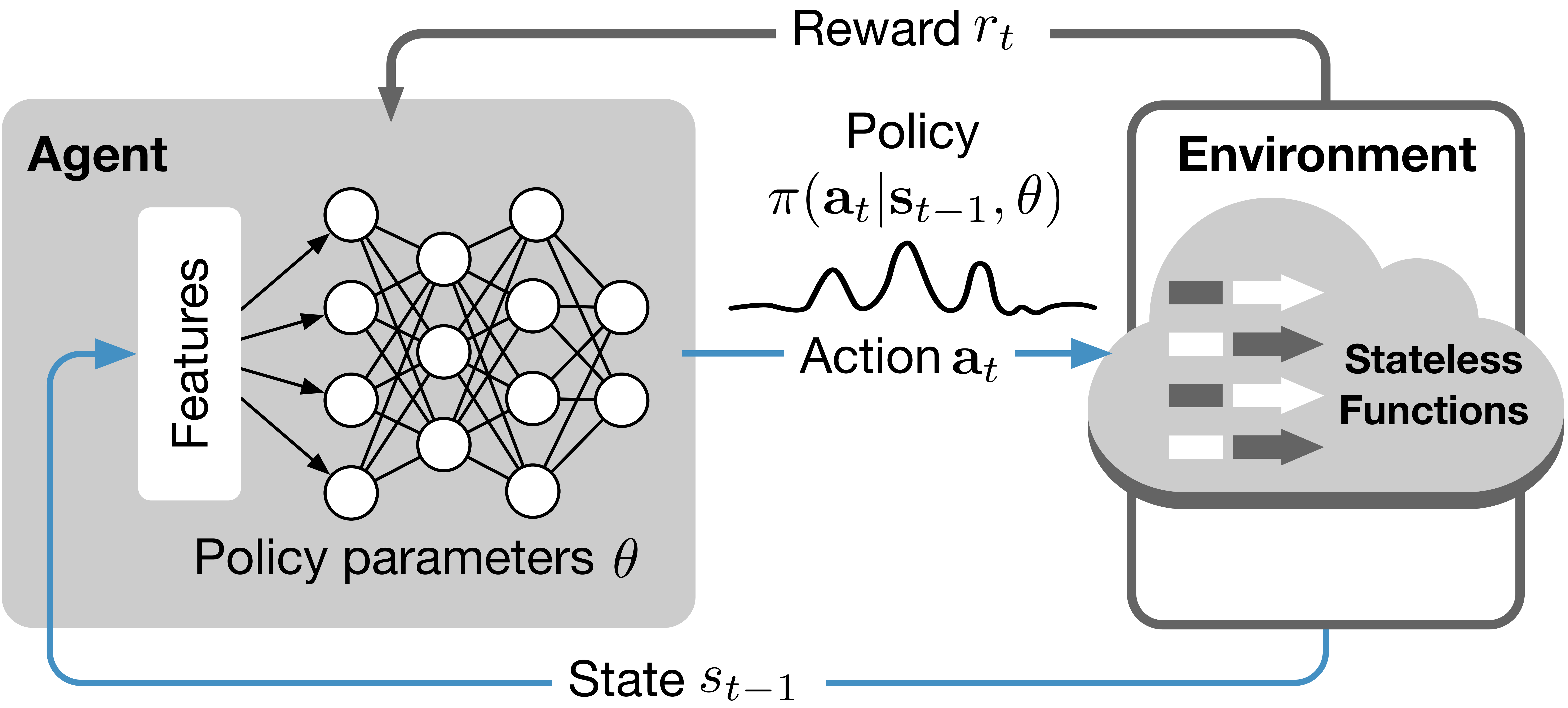
	Loss Value	Time (s)	Cost ($)
20 functions	0.009725	237.40	0.019
8-core EC2	0.009779	307.87	0.029
150 functions	0.009699	50.04	0.031
X functions	0.009761	202.55	0.019

Slowest, no cheap

Fastest, expensive

Fast, cheap

Deep Reinforcement Learning



State

$$\mathbf{s}_t = (t, \ell_t, P_t, P_t^F, P_t^C, P_t^U, u_t, w_t, b_t)$$

t	the epoch index of the training workload
$P_{t,i}^F$	the time period for function $f_{t,i}$ fetching input data
$P_{t,i}^C$	the time period for function $f_{t,i}$ computing gradients
$P_{t,i}^U$	the time period for function $f_{t,i}$ updating parameters
P_t	the whole time period of the epoch t
ℓ_t	the loss value achieved at the end of the epoch t
b_t	the remaining budget at epoch t
u_t	the average memory utilization observed in epoch t
w_t	the average CPU utilization observed in epoch t

Action

n_t	the number of concurrent functions in epoch t
m_t	the memory size of each function in epoch t

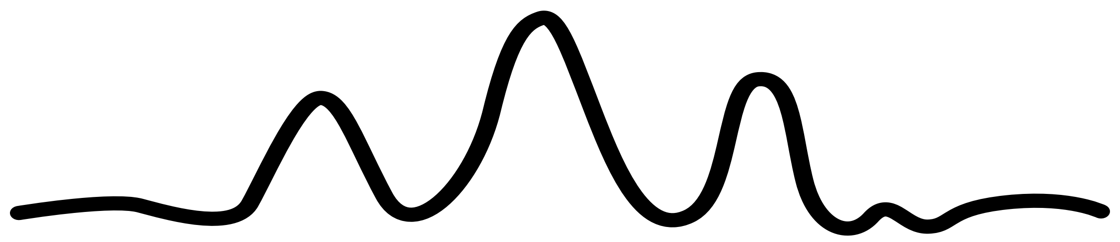
$$\mathbf{a}_t = (n_t, m_t) \qquad n_t, m_t \in \mathbb{Z}^+$$

$n_t \times m_t$ choices ~ 138,000 actions on AWS

Approximating with Gaussian distribution

Policy

$\pi(\mathbf{a}_t | \mathbf{s}_{t-1}, \theta)$



$$\pi(\mathbf{a} \,|\, \mathbf{s}, \theta) = \frac{1}{\sigma(\mathbf{s}, \theta)\sqrt{2\pi}} \exp\left(-\frac{(\mathbf{a} - \mu(\mathbf{s}, \theta))^2}{2\sigma(\mathbf{s}, \theta)^2} \right)$$

Reward

At each epoch t,

$$r_t = -\beta P_t,\ t = 1,\ldots,T-1$$

regularizer

At the final epoch T,

$$r_T = \begin{cases} -\beta P_T + C & \text{if } \ell_T \leqslant \mathcal{L} \text{ and } b_T \geqslant 0, \\ -\beta P_T - C & \text{otherwise.} \end{cases}$$

reach the expected loss value,
or use up all budget

a constant as the final reward/penalty

Training

Maximize cumulative discounted reward:

$$\max \sum_{t=1}^{T} \underbrace{\gamma^t}_{\text{discount factor}} r_t, \ \gamma \in (0,1]$$

discount factor

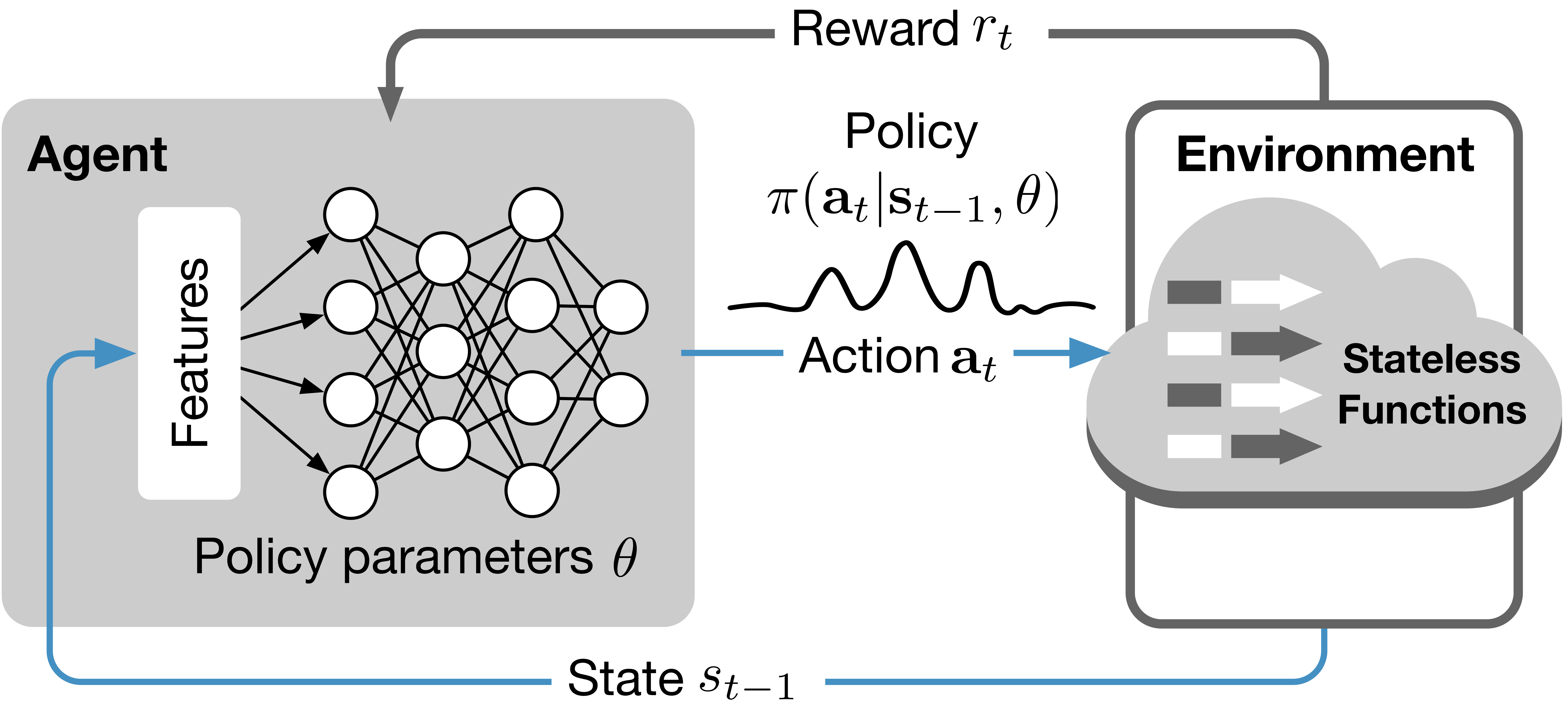
Policy gradient:

$$\nabla_\theta \mathbb{E}_\pi \left[\sum_{t=1}^{T} \gamma^t r_t \right] = \mathbb{E}_\pi [\nabla_\theta \ln \overbrace{\pi(\mathbf{a} \mid \mathbf{s}, \theta)}^{\text{policy function}} \underbrace{q_\pi(\mathbf{s}, \mathbf{a})}_{\text{expected reward with } \mathbf{a} \text{ and } \mathbf{s}}]$$

policy function

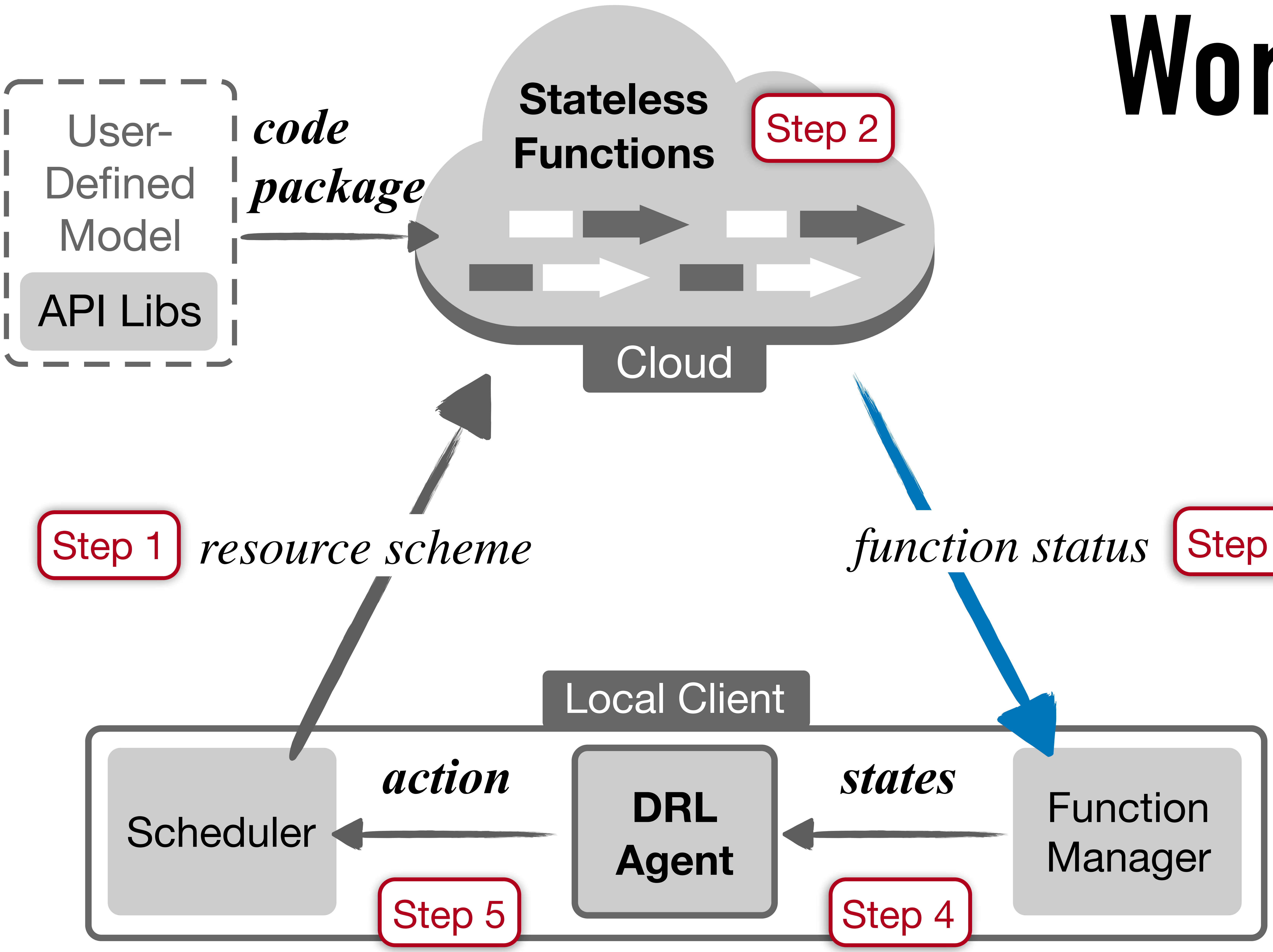
expected reward with $\mathbf{a}$ and $\mathbf{s}$

DRL



Reward r_t
Agent
Features
Policy parameters θ
Policy
$\pi(\mathbf{a}_t|\mathbf{s}_{t-1},\theta)$
Action $\mathbf{a}_t$
Environment
Stateless
Functions
State s_{t-1}

Workflow


User-Defined Model
API Libs
code package
Stateless Functions
Step 2
Cloud
Step 1
resource scheme
function status
Step 3
Local Client
Scheduler
action
DRL Agent
states
Function Manager
Step 5
Step 4

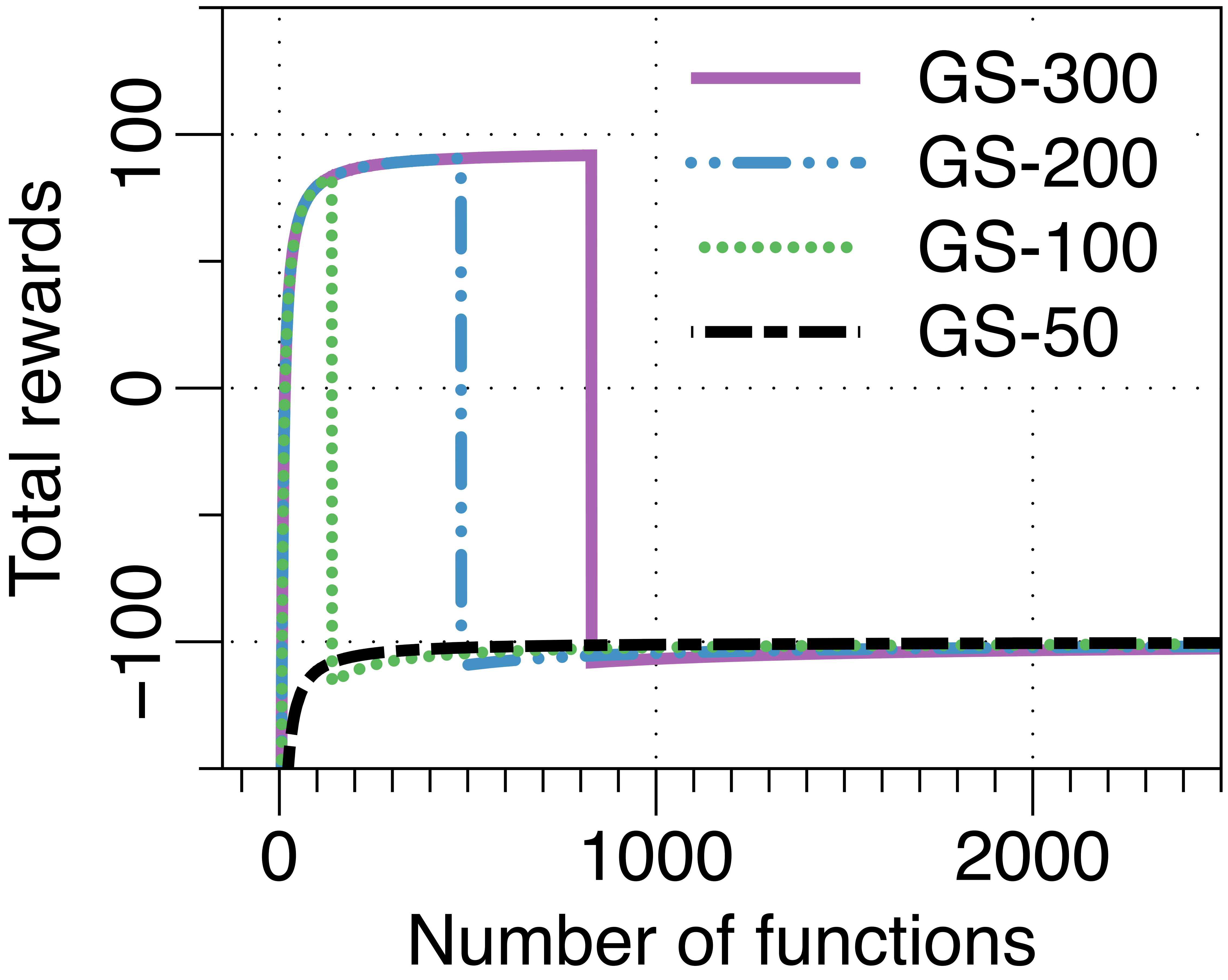
Evaluation

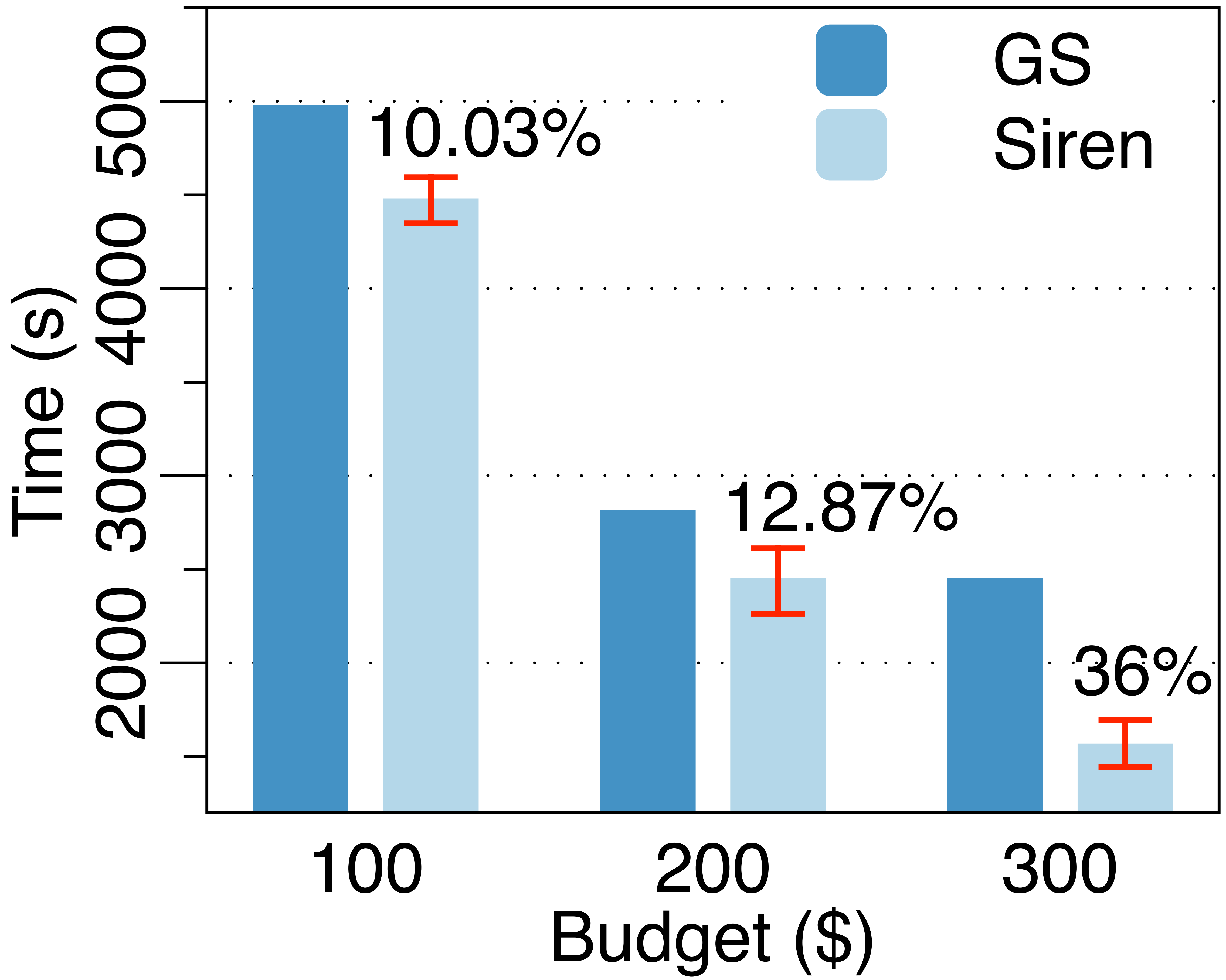
- Simulation: OpenAI Gym
- Testbed: AWS Lambda + AWS EC2
- 44.3% ⬇ on job completion time

Simulation - overview

- **Workload**: mini-batched SGD algorithms
- **Goal**: DRL agent v.s. Grid search (# of functions)

Simulation - grid search

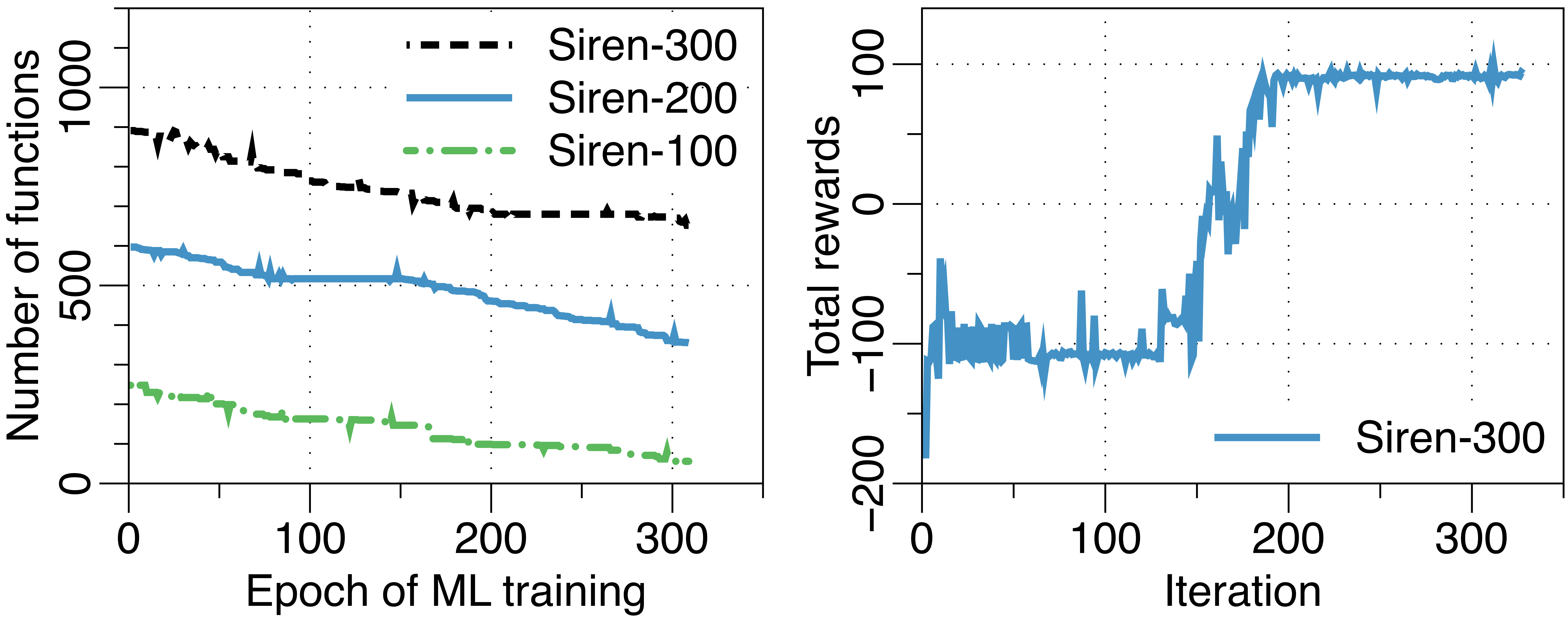




Simulation

	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

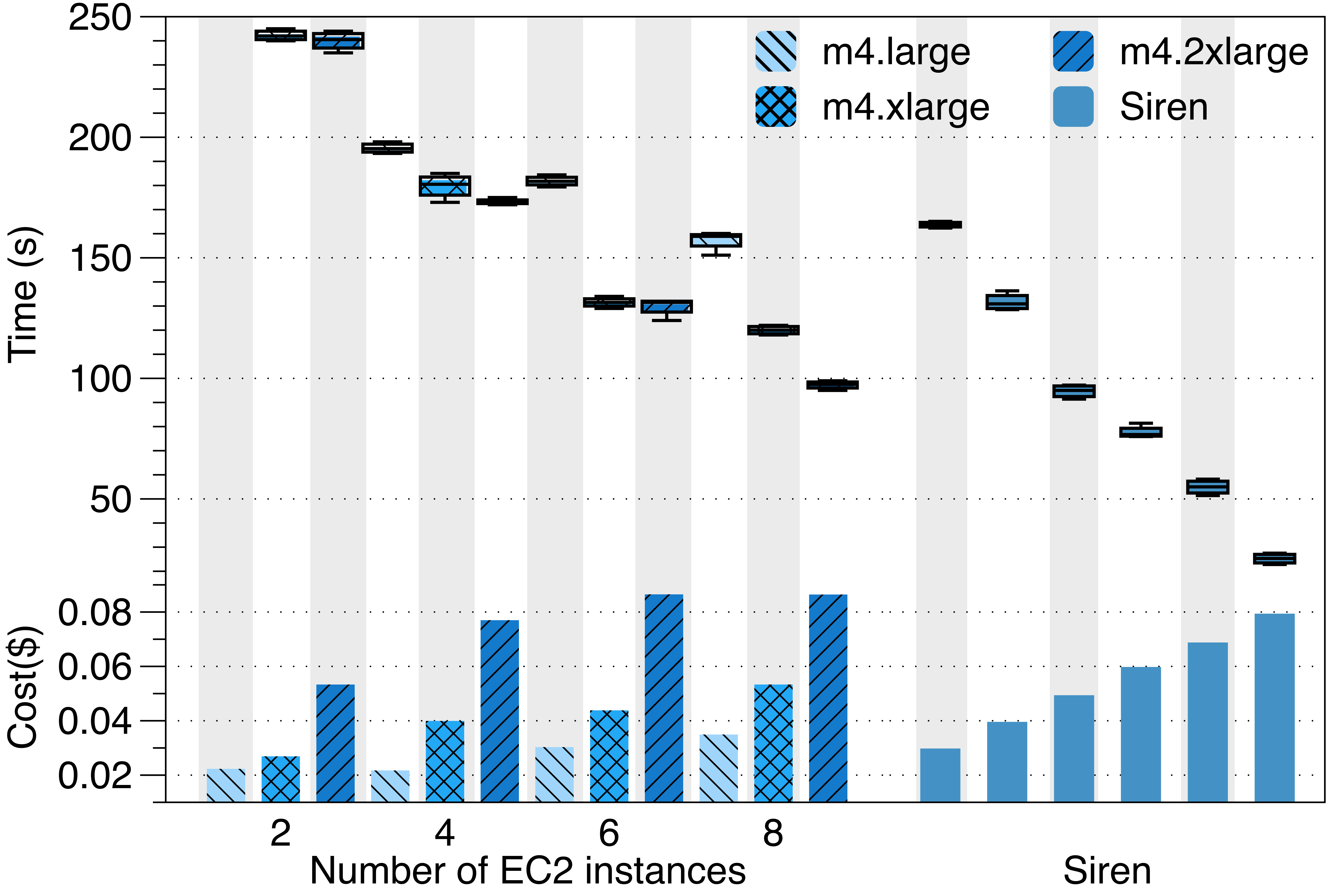
Simulation - DRL training



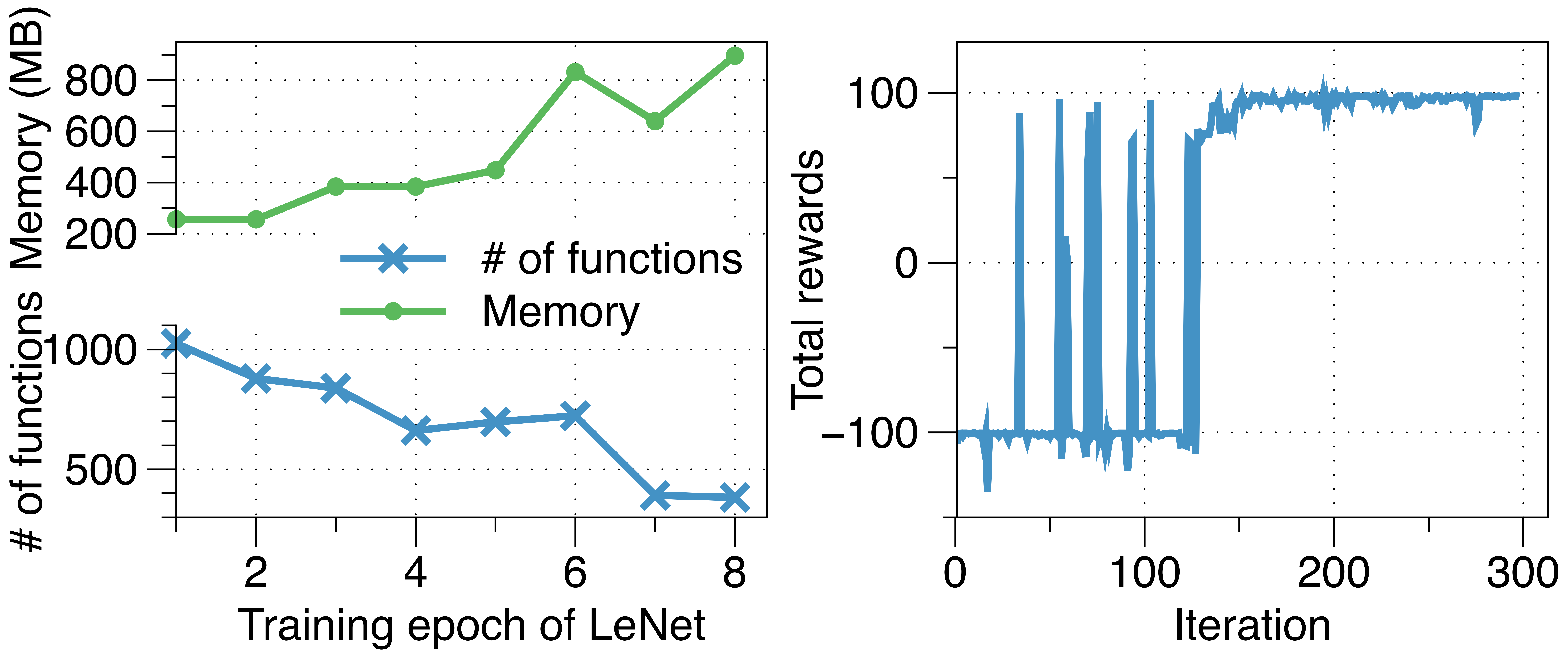
Testbed

- **Siren on AWS Lambda v.s. MXNet on EC2**
  - m4.large: 2 vCPU, 8GB memory, \$0.1/hr
  - m4.xlarge: 4 vCPU, 16GB memory, \$0.2/hr
  - m4.2xlarge: 8 vCPU, 32GB memory, \$0.4/hr
- **Workload**
  - LeNet on MNIST
  - CNN on movie review
  - Linear Classification on click-through prediction dataset

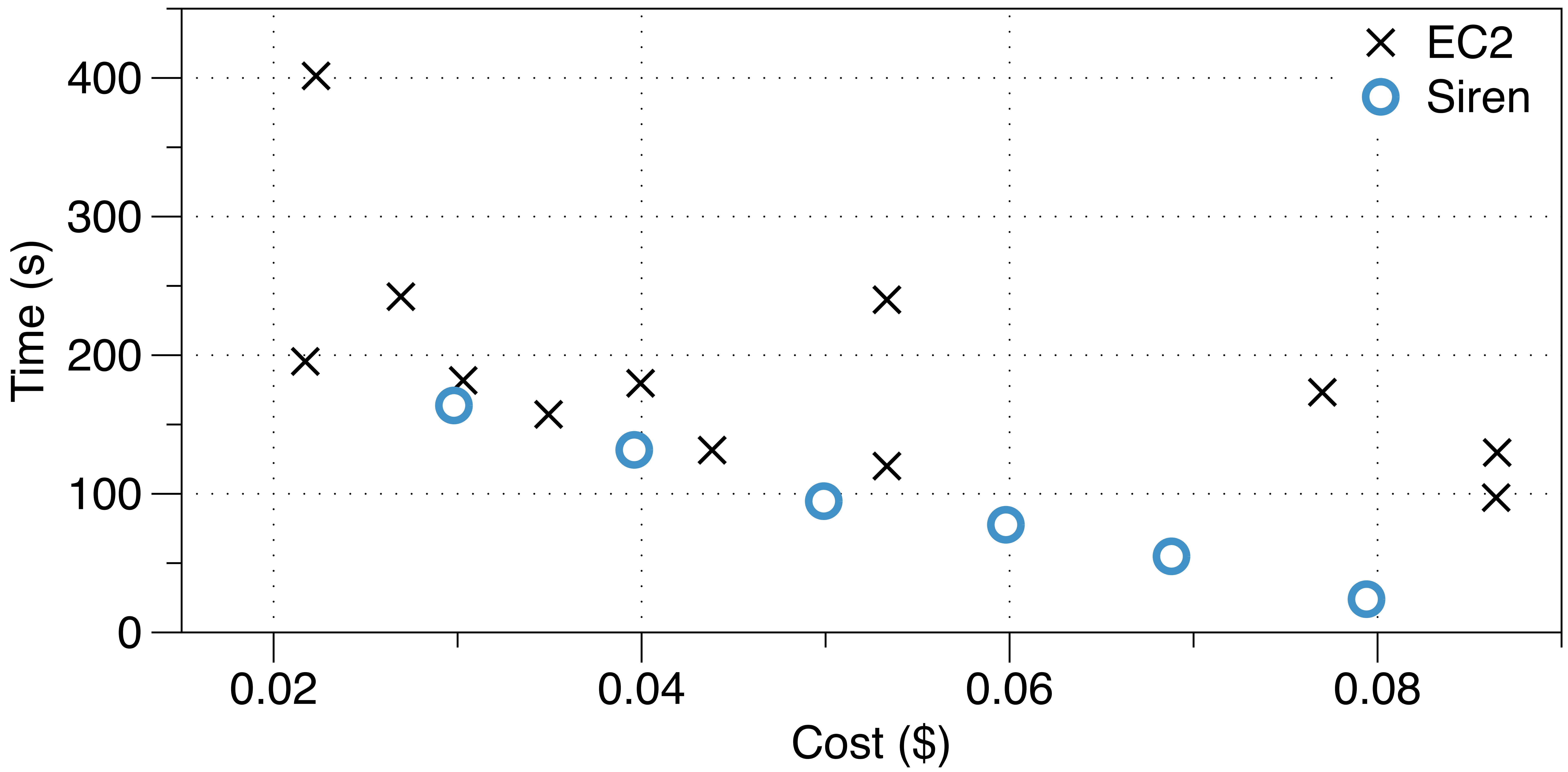
Testbed - Siren and EC2 on LeNet



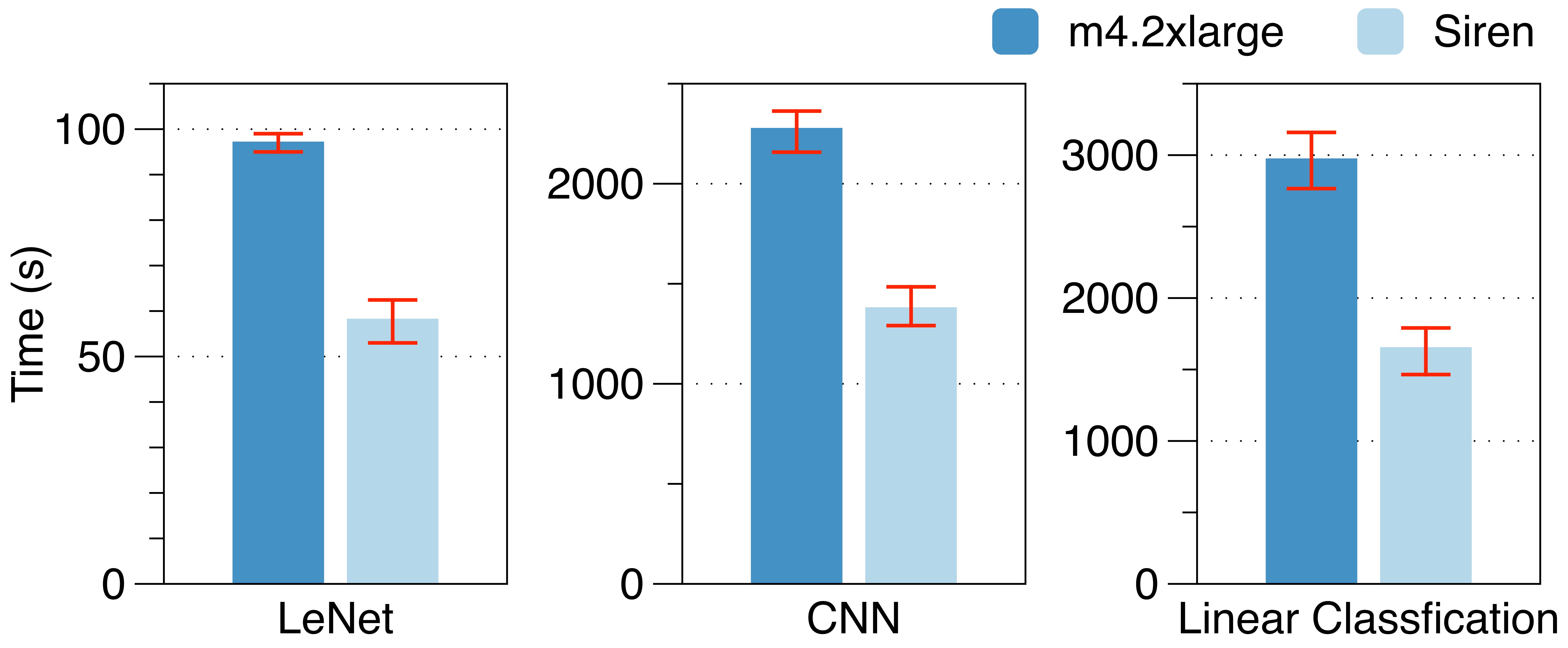
Testbed - DRL training



Testbed - time v.s. cost



Testbed - given the same cost



Conclusion

- **Siren: Distributed Machine Learning with a Serverless Architecture**
  - Hybrid Synchronous Parallel (HSP)
  - Experience-Driven Resource Scheduler

- **Evaluation**
  - Simulation & Testbed
  - 44.3% ⬇ on job completion time


Q&A
Thank You