Exploring Data-driven Approaches to Resource Management in Serverless Systems

Alan Song and Evan Ning (PRIMES CS)

Mentors: Nikita Lazarev
Varun Gohil

PRIMES Fall Conference: October 15, 2023
What is Cloud Computing?

Apps

Cloud
How does cloud computing work?

- Actual application
- How to run the application/what is needed
- How many resources to give to the container
What is serverless?

Traditional cloud computing

Serverless computing
Serverless in Production

Since 2014…

- AWS Lambda
- IBM Cloud Functions
- Google Cloud Functions
- Azure Functions
A Challenge of Serverless

Resource Management
Quality of Service (QoS)

Imagine you are a serverless user who sends some functions to a provider. What do you care about?

Latency: how quickly? 
(milli/micro)seconds

Throughput: how many per second?
Requests Per Second (RPS)
Resource Management - A Balancing Act

Vertical Scaling
(Scaling up)

Horizontal Scaling
(Scaling out)

Quality of Service

Resource Utilization

Cloud Provider
How is resource management done in production?

Autoscaler Logic:

```python
if (cpu_util_per_container > 250m):
    scale_up()
else:
    scale_down()

if (mem_util_per_container > 256 MiB):
    scale_up()
else:
    scale_down()
```
Intelligent Scaling

Data-driven scaling system!

Use an ML model instead!
The problem with traditional ML

Everything keeps changing!
Reinforcement Learning!
Reinforcement learning can be thought of as a loop between the environment and the agent.
RL: Learning through experience

1st time seeing a Goomba

5th time seeing a Goomba

500th time seeing a Goomba
RL (Q-Learning) for Resource Management

- **Agent (scaler)**
  - Q-Table
  - Q=0.88
  - Q=0.13

- **Environment (Cloud)**
  - State (QoS requirements, resource usages, current # of containers)
  - Reward function (based on QoS metrics)

- **Action (scaling)**
  - Scale up
  - Invoke and collect QoS metrics (tail latency)
  - Read metrics

- **Reward**
  - Compute reward
  - Save experience

Q=0.88
Q=0.13

Q-Table
Deep Q-Learning for Resource Management

- **State (resource usages, current # of containers)**
  - Read metrics
  - Compute reward
- **Action (scaling)**
  - Scale up
- **Reward function (based on QoS metrics)**
- **Reward**
  - Based on QoS metrics
  - Q=0.88
  - Q=0.13

- **Agent (scaler)**
  - Q-Table
  - Deep Q Network
- **Environment (Cloud)**
  - Read metrics (tail latency)

Challenges in a serverless environment

- **Agent (scaler)**
  - State (resource usages, current # of containers)
  - Read metrics
  - Compute reward
  - Scale up
  - Save experience

- **Environment (containers)**
  - Read metrics (tail latency)

- **Reward**
  - Based on QoS metrics
  - Functions are running…

**Actions (scaling)**
- Scale up
- Read metrics
- Compute reward
- Save experience
Reusing previous experiences with a replay buffer

State (QoS requirements, resource usages, current # of containers)

Environment (Cloud)

Agent (scaler)

Deep Q Network

Reward

Action (scaling)

Compute reward

Save experience

Invoke and collect QoS metrics (tail latency)

Reward function (based on QoS metrics)

Scale up

Read metrics

Reusing previous experiences with a replay buffer

State (QoS requirements, resource usages, current # of containers)

Environment (Cloud)

Agent (scaler)

Deep Q Network

Reward

Action (scaling)

Compute reward

Save experience

Invoke and collect QoS metrics (tail latency)

Reward function (based on QoS metrics)
System Implementation
A Real Serverless Environment

Prometheus  kubernetes  vSwarm

CloudLab

hotel-app  fibonacci-python
DQN Implementation: Each Episode

1. Deploy functions
2. Get initial environment state
3. Initialize DQN

Client (main)

CloudLab Node

DQN

API Gateway

Training Loop

Written by us
DQN Implementation: Training

1. Action(s) $A = \text{Up/Down/0}$
2. Return state and compute reward
3. Scale functions and invoke
4. Return metrics and latencies
5. Compile env state and latencies

Written by us

Any Model!
Results - Horizontal Pod Autoscaling

Horizontal Scaling

Latency as QoS

- 99th tail
- 90th tail
- median

Stop learning!

Reward curve

Number of containers

function #1
function #2
function #3
Results - Vertical Pod Autoscaling

Vertical Scaling

(Scaling up)

Reward curve

CPU limit

QoS

Microsecond

Around 1000-2000

Step
Contributions and Artifacts

1. We **constructed proper infrastructure** to replicate serverless environments with different workloads.
2. We implemented **Deep Q-Learning** as a data-driven way to tackle resource management in dynamic serverless environments.
3. Github can be found here
Acknowledgments

- Nikita, Varun, Lisa
- Families
- Shoutout to CloudLab!
- PRIMES