

CS 744: PYWREN

Shivaram Venkataraman

Spring 2025

ADMINISTRIVIA

Poster presentation: May 1

Final report: May 8

Project grade breakdown

Intro: 5%

Mid-semester checkin: 5%

Poster: 10%

Final Report: 10%

Applications

Machine Learning

SQL

Streaming

Graph

Computational Engines

Scalable Storage Systems

Resource Management



Datacenter Architecture



MICRO-SERVICES, NEW HARDWARE, ML FOR SYSTEMS

SERVERLESS COMPUTING

MOTIVATION: USABILITY

What instance type?

What base image?

How many to spin up?

What price? Spot?

The screenshot shows the AWS Management Console for EC2 Instance Types. The interface includes a navigation bar with 'Instances' selected, and sub-tabs for 'EC2', 'RDS', 'ElastiCache', 'Redshift', and 'OpenSearch'. A purple banner at the top right promotes 'Save 50%+ on AWS with Autopilot'. Below the navigation, there are filters for Region (US East (N. Virginia)), Pricing Unit (Instance), Cost (Hourly), and Reserved (1-year - No Upfront). There are also buttons for 'Columns', 'Compare Selected', and 'Clear Filters'. The main content is a table of instance types with columns for Name, API Name, Instance Memory, vCPUs, and Instance Storage. The table is filtered to show instance types with a minimum of 0 vCPUs and 0 Instance Memory. The instance types listed include C5 High-CPU Double Extra Large, C5 High-CPU Extra Large, M6A 24xlarge, M5DN Extra Large, C5 High-CPU Metal, C6A Eight Extra Large, D3EN 12xlarge, D3EN Eight Extra Large, R5AD 16xlarge, M5A Double Extra Large, M5N Metal, C6ID Eight Extra Large, M5AD Double Extra Large, and M6ID Extra Large.

Name	API Name	Instance Memory	vCPUs	Instance Storage
C5 High-CPU Double Extra Large	c5d.2xlarge	16.0 GiB	8 vCPUs	200 GB NVMe SSD
C5 High-CPU Extra Large	c5d.xlarge	8.0 GiB	4 vCPUs	100 GB NVMe SSD
M6A 24xlarge	m6a.24xlarge	384.0 GiB	96 vCPUs	EBS only
M5DN Extra Large	m5dn.xlarge	16.0 GiB	4 vCPUs	150 GB NVMe SSD
C5 High-CPU Metal	c5.metal	192.0 GiB	96 vCPUs	EBS only
C6A Eight Extra Large	c6a.8xlarge	64.0 GiB	32 vCPUs	EBS only
D3EN 12xlarge	d3en.12xlarge	192.0 GiB	48 vCPUs	335520 GB (24 * 13980 GB HDD)
D3EN Eight Extra Large	d3en.8xlarge	128.0 GiB	32 vCPUs	223680 GB (16 * 13980 GB HDD)
R5AD 16xlarge	r5ad.16xlarge	512.0 GiB	64 vCPUs	2400 GB (4 * 600 GB NVMe SSD)
M5A Double Extra Large	m5a.2xlarge	32.0 GiB	8 vCPUs	EBS only
M5N Metal	m5n.metal	384.0 GiB	96 vCPUs	EBS only
C6ID Eight Extra Large	c6id.8xlarge	64.0 GiB	32 vCPUs	1900 GB NVMe SSD
M5AD Double Extra Large	m5ad.2xlarge	32.0 GiB	8 vCPUs	300 GB NVMe SSD
M6ID Extra Large	m6id.xlarge	16.0 GiB	4 vCPUs	237 GB NVMe SSD

When to use the Cloud ?

Data

- Large amounts of data. Can't store locally
- Shared data across users
- Long term storage

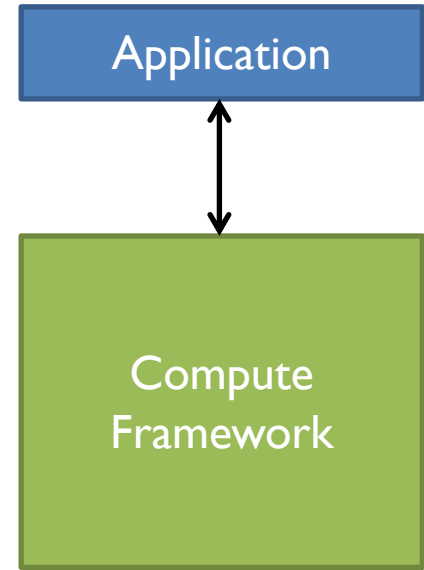
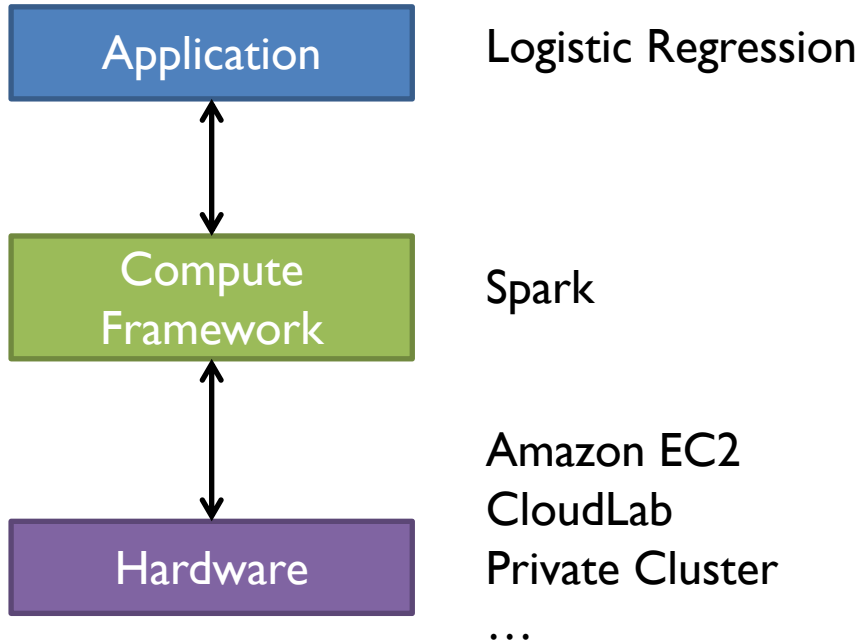
Compute

- Need lots of CPUs for shared (VMs)
- Varying compute requirements
- No admin overhead

Why is there no
"cloud button"?



ABSTRACTION LEVEL ?



“SERVERLESS” COMPUTING

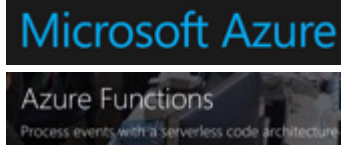
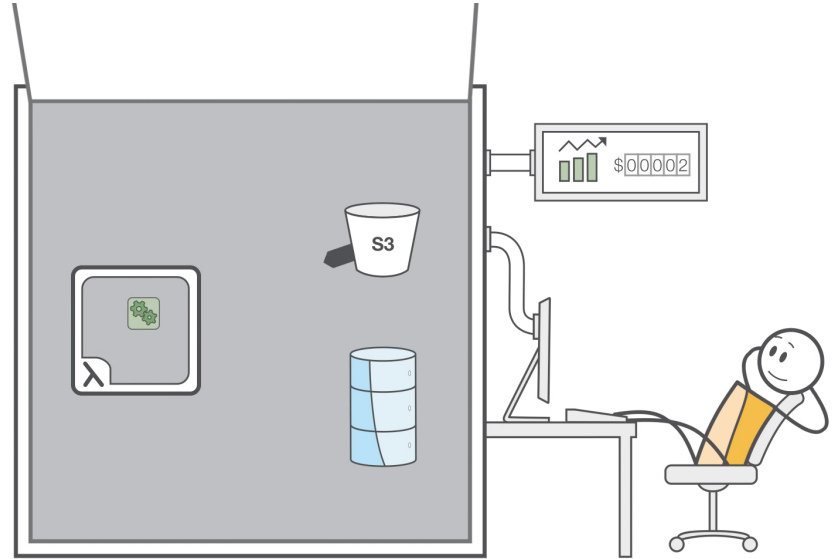
300-900 seconds single-core

512-10240 MB in /tmp

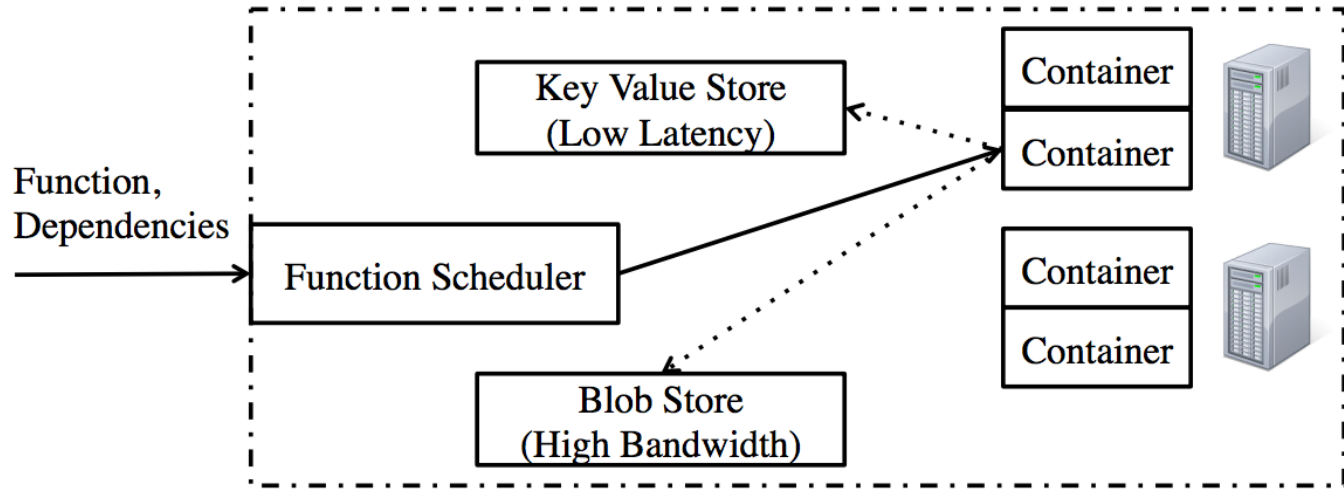
3-10GB RAM

Python, Java, node.js, Ruby, Go etc.

Support for containers



STATELESS DATA PROCESSING



PYWREN API

```
import pywren
import numpy as np

def addone(x):
    return x + 1

wrenexec = pywren.default_executor()
xlist = np.arange(10)
futures = wrenexec.map(addone, xlist)

print [f.result() for f in futures]
```

The output is as expected:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

PYWREN: HOW IT WORKS

```
future = runner.map(fn, data)
```

```
future.result()
```

your laptop

the cloud



HOW IT WORKS

```
future = runner.map(fn, data)
```

Serialize func and data

Put on S3

Invoke Lambda

func

data

pull job from s3

download anaconda runtime

python to run code

pickle result

stick in S3

```
future.result()
```

poll S3

unpickle and return

your laptop

result

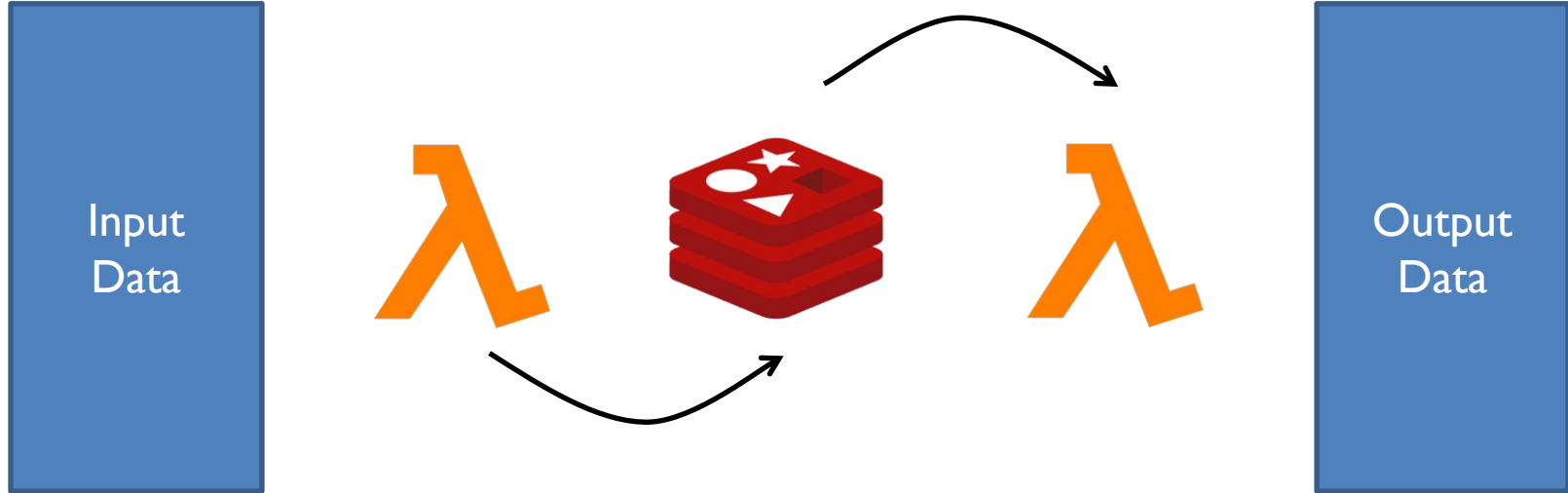
the cloud

STATELESS FUNCTIONS: WHY NOW ?

What are the trade-offs ?

Storage Medium	Write Speed (MB/s)
SSD on c3.8xlarge	208.73
SSD on i2.8xlarge	460.36
4 SSDs on i2.8xlarge	1768.04
S3	501.13

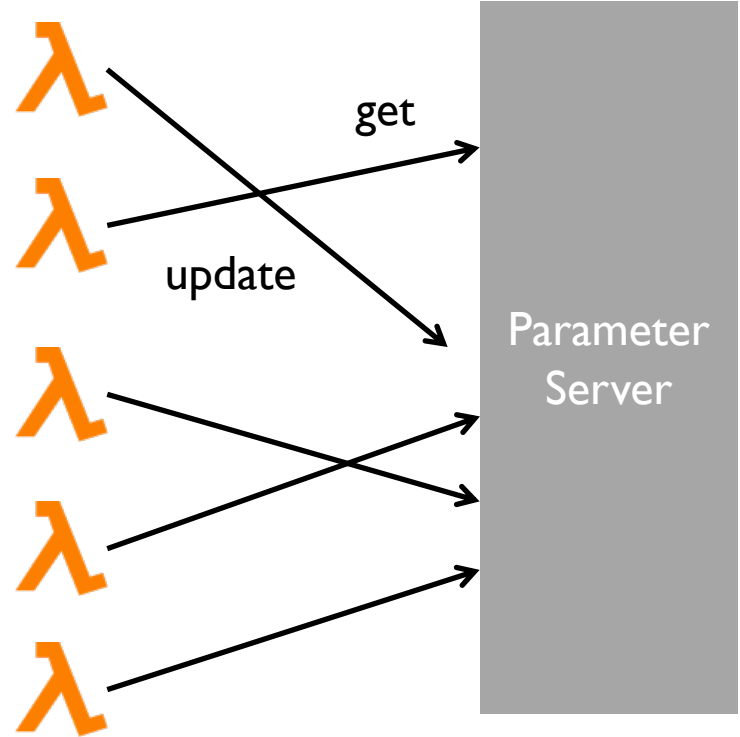
MAP AND REDUCE ?



PARAMETER SERVERS

Use lambdas to run “workers”

Parameter server as a service ?



WHEN SHOULD WE USE SERVERLESS ?

Yes!

Maybe not ?

SUMMARY

Motivation: Usability of big data analytics

Approach: Language-integrated cloud computing

Features

- Breakdown computation into stateless functions
- Schedule on serverless containers
- Use external storage for state management

Open question on scheduling, overheads



DISCUSSION

<https://forms.gle/2RhPBaIXQC9XpFVw8>

Consider a recommendation model trained on a graph where we use 2-hop neighbors. What are some challenges in using BagPipe-style ideas for such a workload?

Consider you are a cloud provider (e.g., AWS) implementing support for serverless. What could be some of the new challenges in scheduling these workloads compared to schedulers we have studied in this class? How would you go about addressing them?

NEXT UP

Next steps:

- TPU next