Welcome back!

CS 744: PYWREN

Shivaram Venkataraman

Spring 2024

ADMINISTRIVIA

Project checkins due today! → ~1-2 pages

Poster presentation: May 2

Final report: May 7

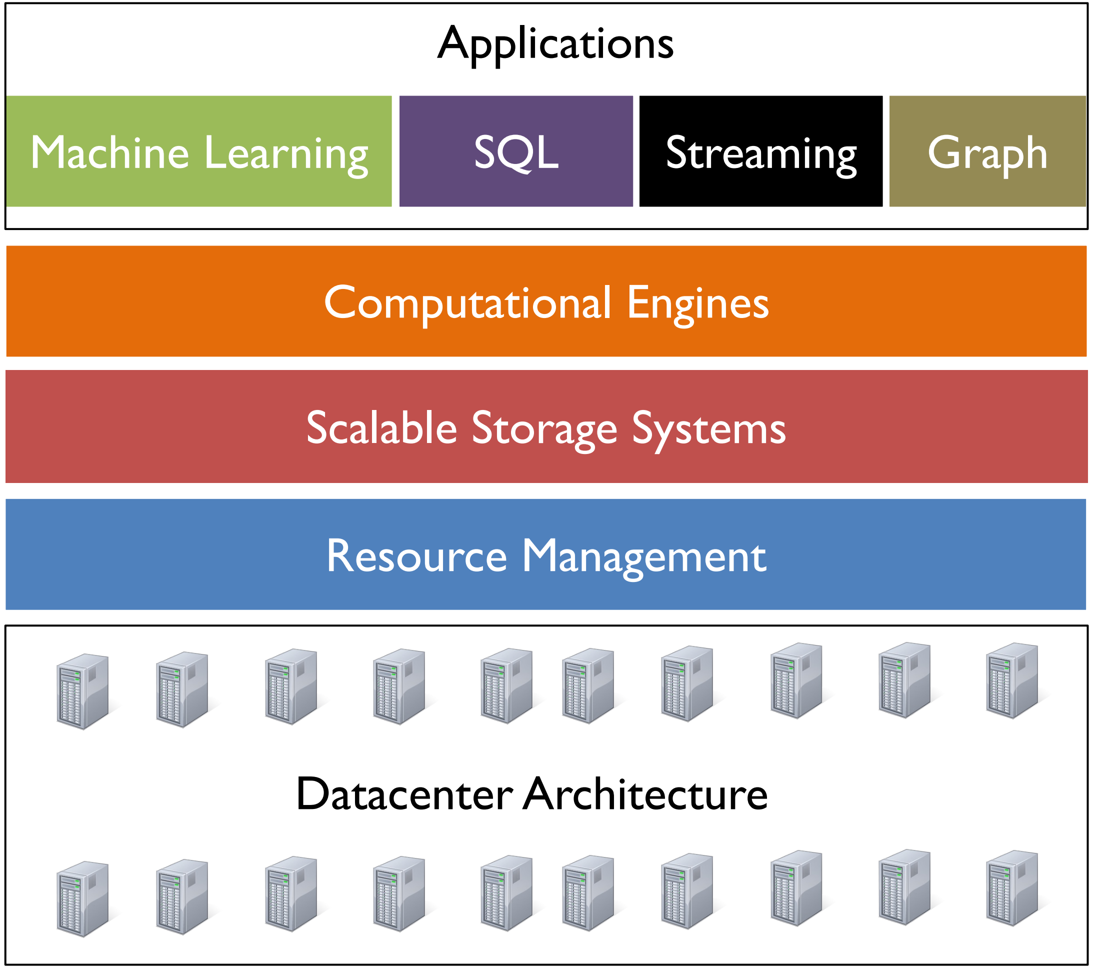
Project grade breakdown

- Intro: 5%
- Mid-semester checkin: 5%
- Poster: 10%
- Final Report: 10%

→ Upload template

Regrade requests

midterm 2 practice exams!


Applications
Machine Learning
SQL
Streaming
Graph
Computational Engines
Scalable Storage Systems
Resource Management
Datacenter Architecture

NEW DATA, HARDWARE MODELS

Cloud computing provider

Serverless Computing

Accelerator design
ML workloads

TPU paper

Compute Accelerators

Infiniband Networks

Non-Volatile Memory

HeMem

↳ fast DRAM
persistent

SERVERLESS COMPUTING

↳ there are actually servers!

MOTIVATION: USABILITY

Usability
↳ Snowflake

Price with each of them

What instance type?

What base image?

How many to spin up?

What price? Spot?

Instances | EC2 | RDS | ElastiCache | Redshift | OpenSearch

Save 50%+ on AWS with Autopilot →

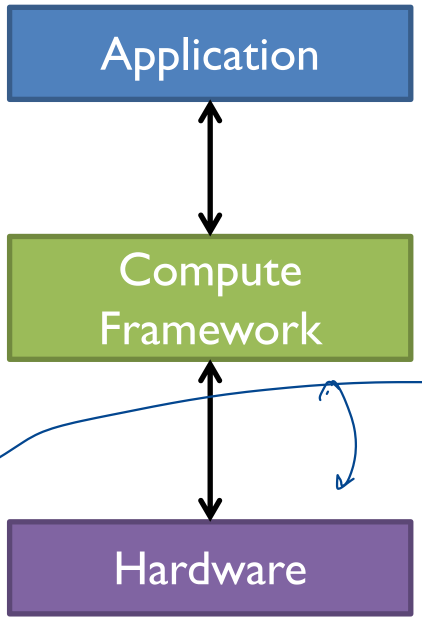
Region: US East (N. Virginia) · Pricing Unit: Instance · Cost: Hourly · Reserved: 1-year - No Upfront · Columns · Compare Selected · Clear Filters · Expor

Name	API Name	Instance Memory	vCPUs	Instance Storage	Ne
Filter..	Filter..	Min Mem: 0	Min vCPUs: 0	Min Storage: 0	F
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	

Why is there no "cloud button"?
Electric Eng
MRI
When to use the Cloud ?
Data
- Large amounts of data. Can't store locally
- Shared data across users
- Long term storage
Compute

ABSTRACTION LEVEL ?

Language Integrated Queries



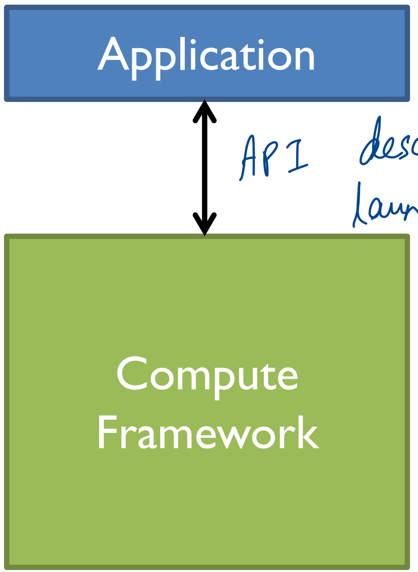
PageRank

Logistic Regression

API

Spark / PyTorch Configure

Amazon EC2
CloudLab
Private Cluster
…



API describe launch cloud

"SERVERLESS" COMPUTING

"Tasks"

↳ Small tasks for short time period

1GB memory
1GB disk
↔ 15 mins

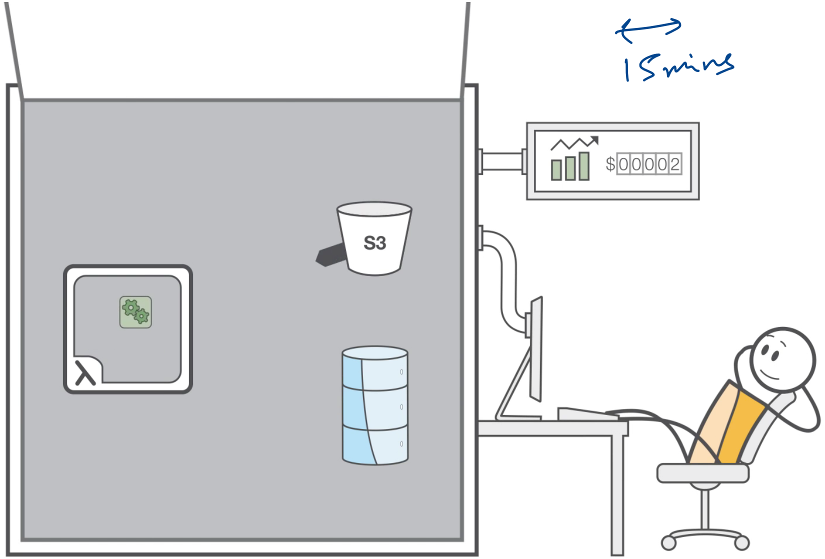
Pay ~ ms granularity
~15 mins
Duration

~~300~~ 900 seconds single-core

~~512~~ 512-10240 MB in /tmp   disk

~~3~~ 10GB RAM   Memory

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



Support for containers

- Pay when task is running
- Really fast to launch

lambda

Google Cloud Platform
CLOUD FUNCTIONS ALPHA
A serverless platform for building event-based microservices

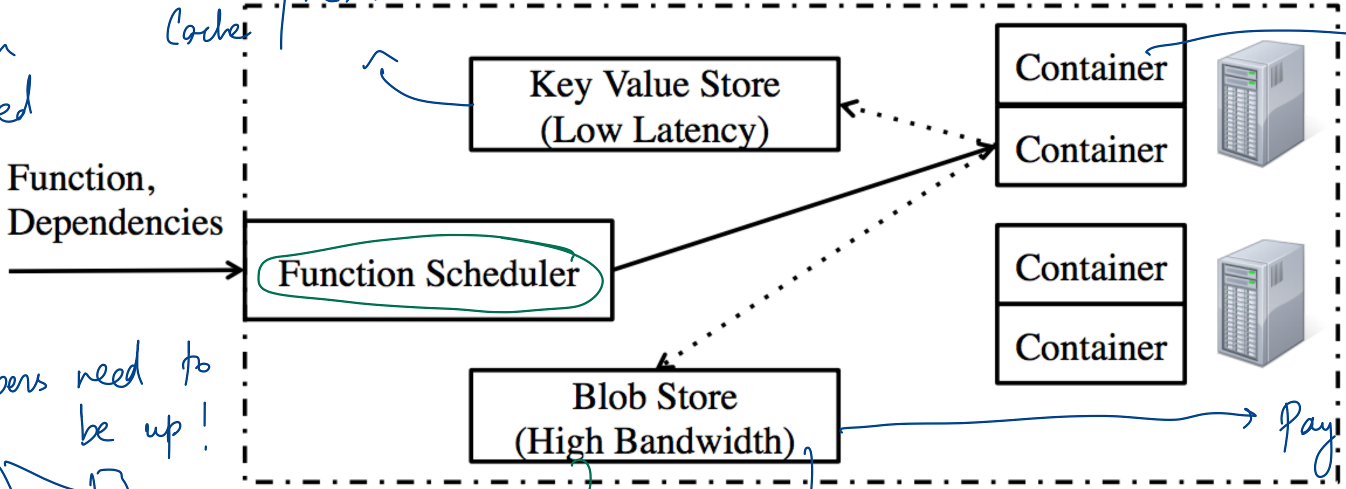
Microsoft Azure
Azure Functions
Process events with a serverless code architecture

"STATELESS" DATA PROCESSING

State
→ Comp. graph (lineage)
→ Intermediate data → on disks colocated with tasks

↳ No State inside the Serverless instance!
→ task short / limited resources

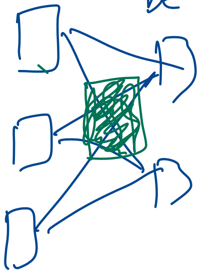
Elastic Cache | Redis



Serverless functions

Pay as you go

Mappers need to be up!



inputs and outputs

Storage system

S3 → intermediate data

PYWREN API

pip install pywren

```
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]
```

import scipy
scipy..

nothing about cloud / instances etc.

local function

list of inputs

similar semantics to Python map

handle that you can use to wait or retrieve result

The output is as expected:

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

PYWREN: HOW IT WORKS

cloud pickle

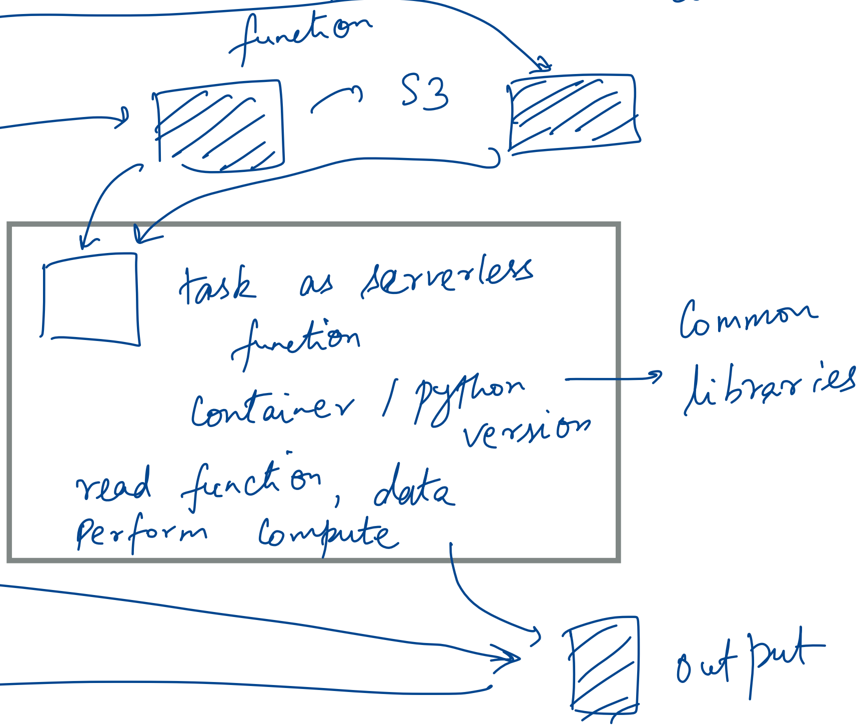
Config file → docker container

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

Serialize fn

upload

Invoke serverless function from cloud API




```
future.result()
```

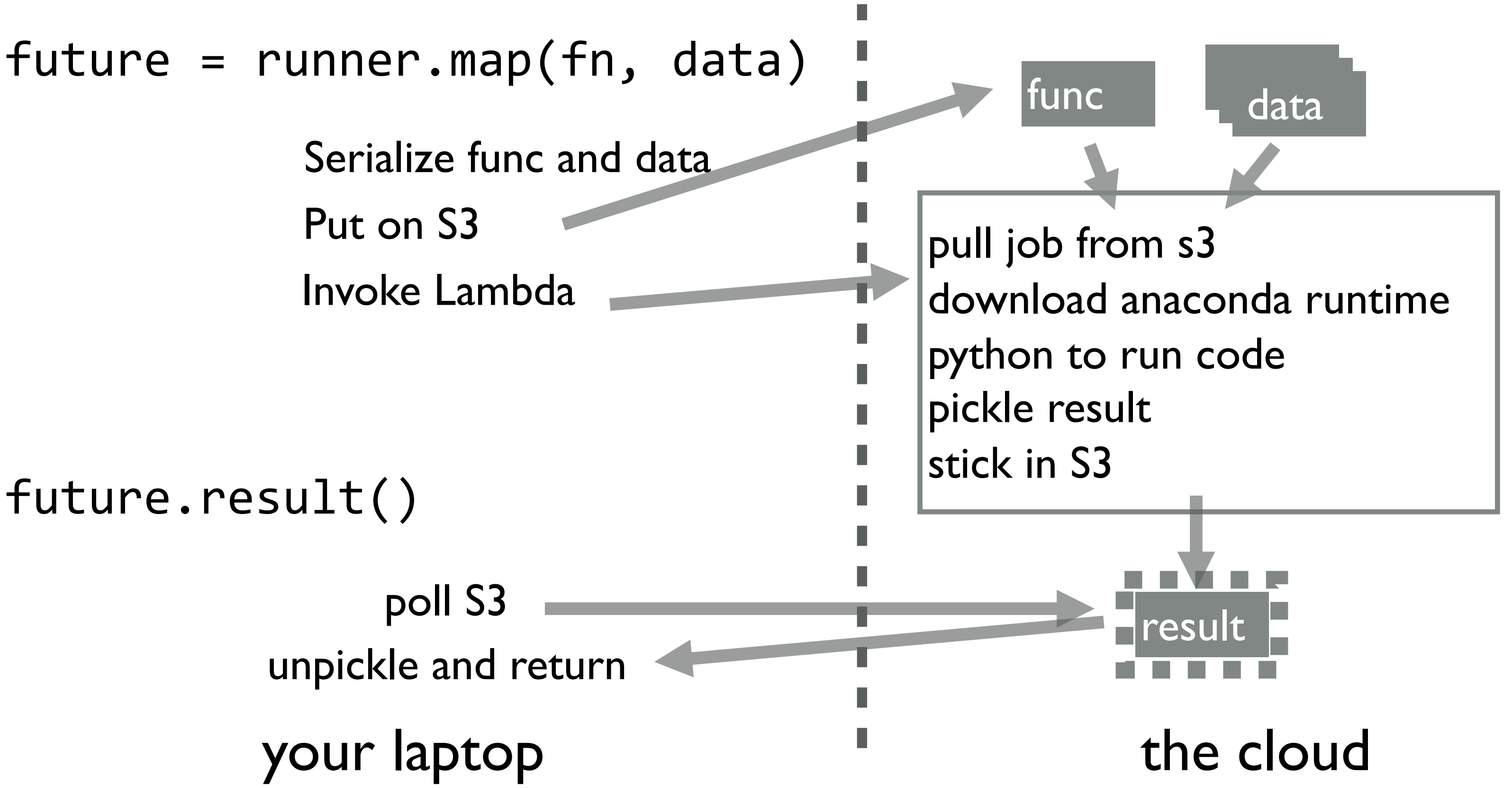
Polling

deserialize

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
result
your laptop
the cloud

STATELESS FUNCTIONS: WHY NOW ?

What are the trade-offs ?

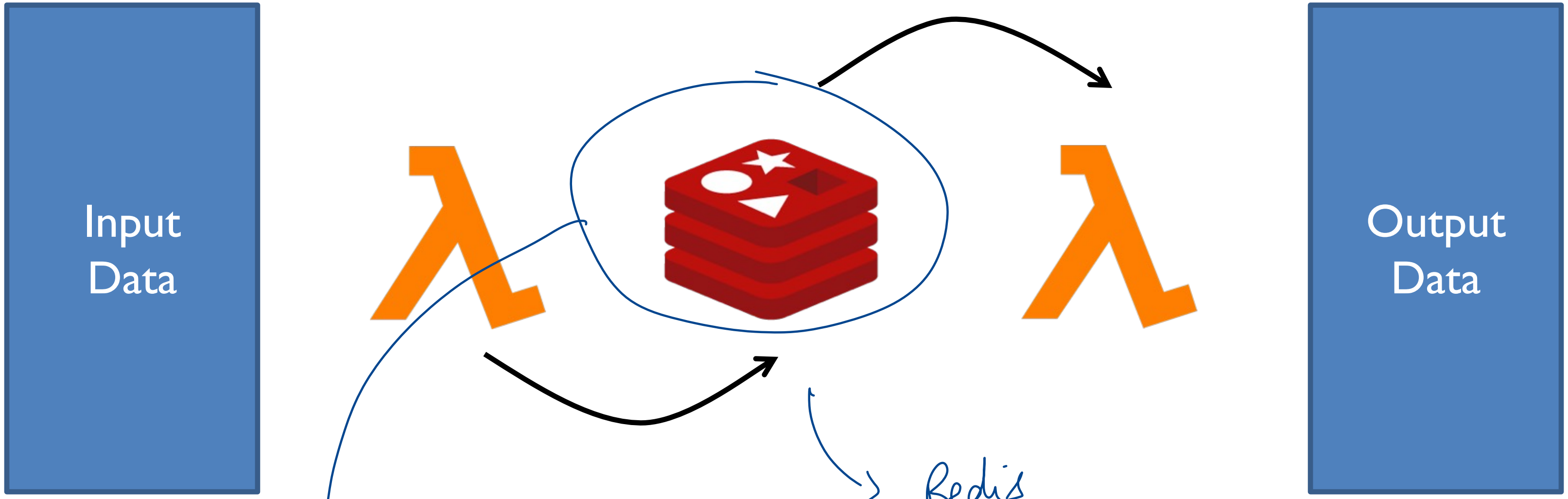
- All reads & writes goes to S3 !
- Network is competitive with local SSD storage
  - ↳ slower than 4 x SSDs

Throughput / Bandwidth

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 ?

Sort
↳ 1000 maps → output from each mapper to each reducer → 1 million files
1000 reducers



Better storage systems ?

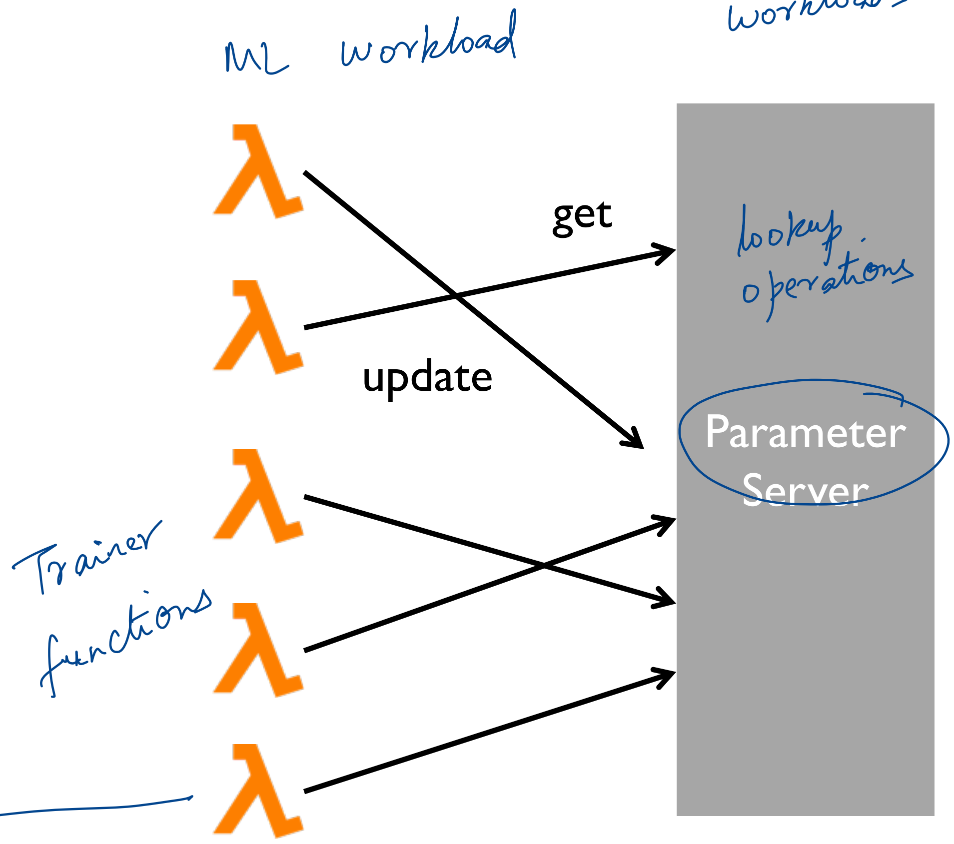
~1M KV pairs

Redis
↳ key value pair for each shuffle output

PARAMETER SERVERS

Use lambdas to run "workers"

Parameter server as a service ?



GPU workers ?

No.

WHEN SHOULD WE USE SERVERLESS ?

Yes!

Image processing

↳ embarrassingly parallel

every key on its

own

↳ trigger when new

images arrive

Maybe not ?

long running ML training

↳

SUMMARY

Motivation: Usability of big data analytics

Approach: Language-integrated cloud computing

Features

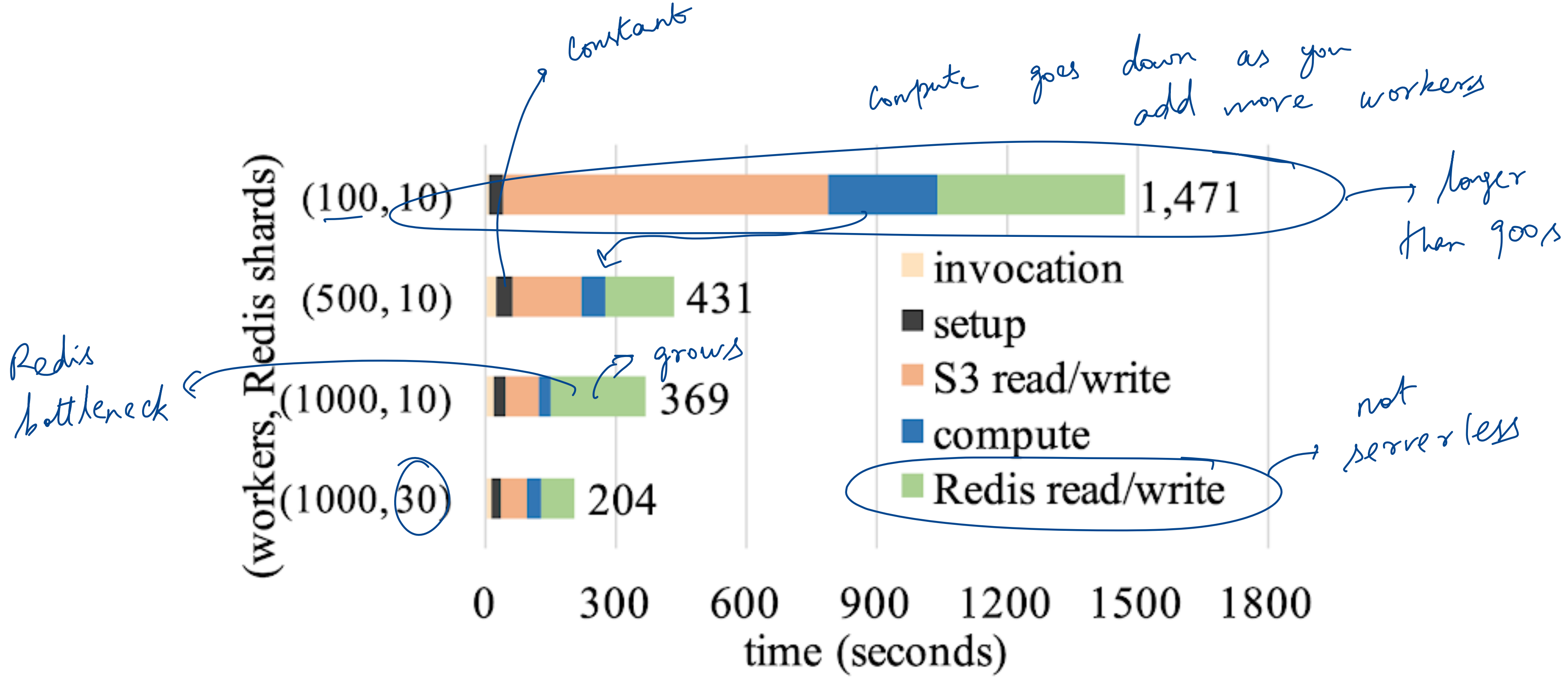
- Breakdown computation into stateless functions
- Schedule on serverless containers
- Use external storage for state management

Open question on scheduling, overheads

DISCUSSION

https://forms.gle/cEvaUK4JR65Ykp7p9

+ when to use serverless



(workers, Redis shards)
(100, 10)
1,471
(500, 10)
431
(1000, 10)
369
(1000, 30)
204
invocation
setup
S3 read/write
compute
Redis read/write
0
300
600
900
1200
1500
1800
time (seconds)
Constants
Compute goes down as you add more workers
longer than 900s
grows
Redis bottleneck
not serverless

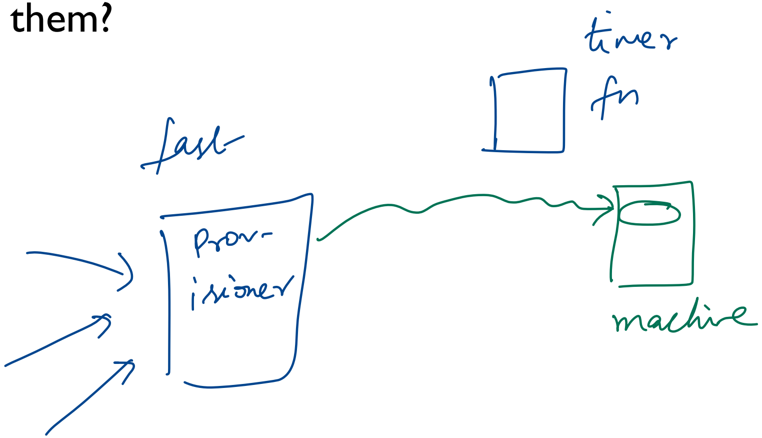
Consider you are a cloud provider (e.g., AWS) implementing support for serverless. What could be some of the new challenges in scheduling these workloads compared to schedulers we have studied in this class ? How would you go about addressing them?

Challenges & opportunities

→ large number of tasks

- max duration of tasks is 900s

↳ get nice scheduling properties



→ Pre-allocated or pre-provisioned

↳ Algorithms / mechanisms to handle bursts

NEXT UP

Next steps:

- Mid-semester project check-in
- TPU next