Hello!

CS 744: PYWREN

Shivaram Venkataraman

Fall 2022

ADMINISTRIVIA

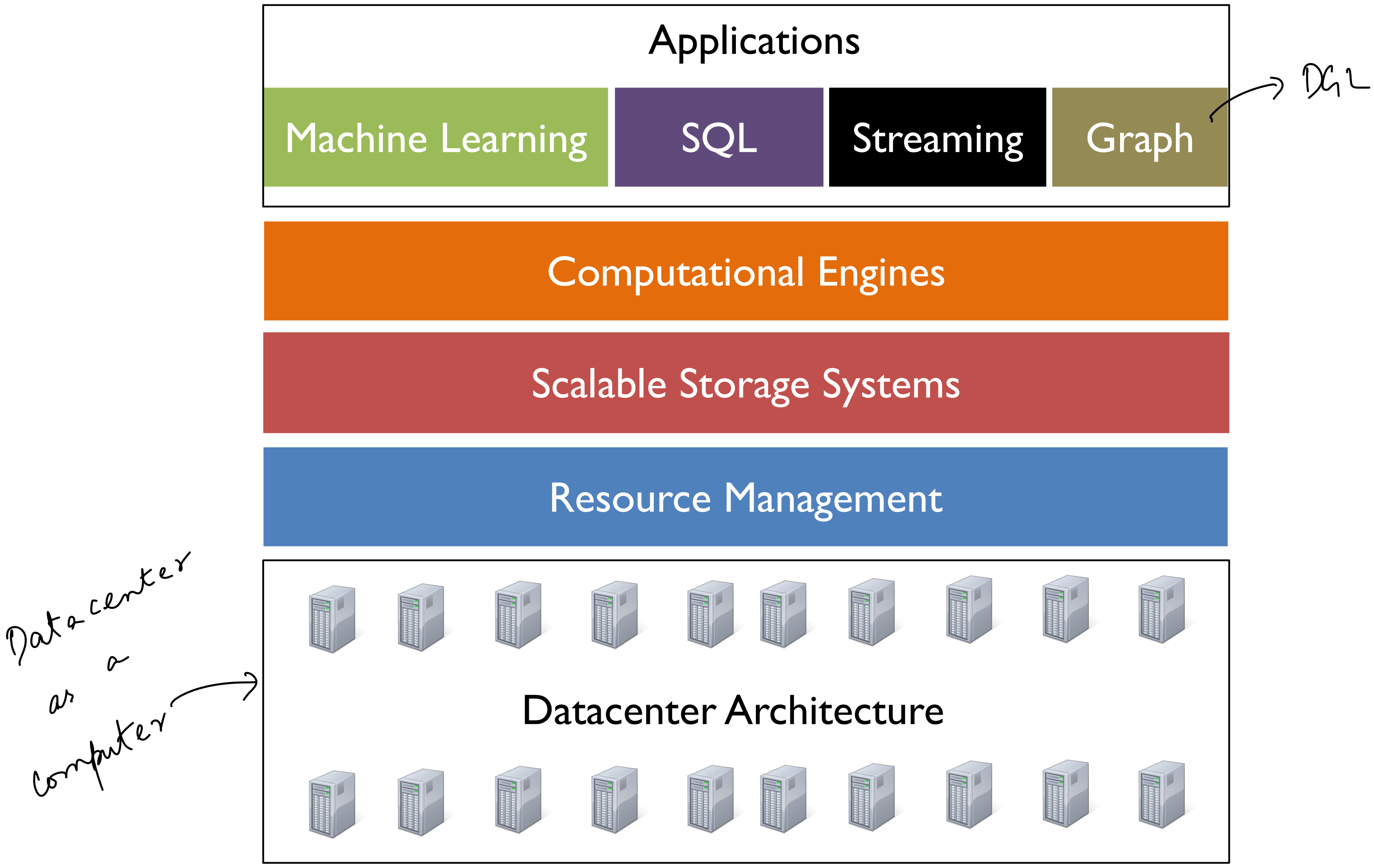
Project checkins due Nov 23rd → one page writeup
↳ how things are going

Poster presentation: Dec 13th

Final report: Dec 20th

Project grade breakdown → 30%

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


Applications
Machine Learning
SQL
Streaming
Graph
DGL
Computational Engines
Scalable Storage Systems
Resource Management
Datacenter Architecture
Datacenter as a computer

NEW DATA, HARDWARE MODELS

Cloud computing

↳ data analytics stack

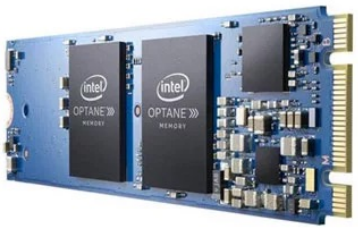
Serverless Computing

Machine Learning

New System designs

Compute Accelerators

Infiniband Networks



Non-Volatile Memory

SERVERLESS COMPUTING

↳ not actually without servers

MOTIVATION: USABILITY

→ user who is not a CS major use this?

Amazon EC2
Virtual machine
instances

User

→ **What instance type?**

What base image?

How many to spin up?

What price? Spot?

3 machine or 100 machines?

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	

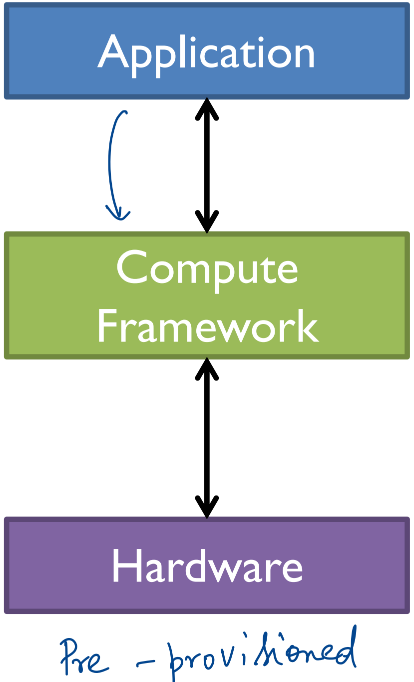
CPU

Mem

335 TB


Cloud
EE grads
MRI
images
Push a button
Why is there no "cloud button"?
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

ABSTRACTION LEVEL ?



Deep Learning Training

Logistic Regression

Word Count / Sort

Spark / PyTorch

Amazon EC2

CloudLab

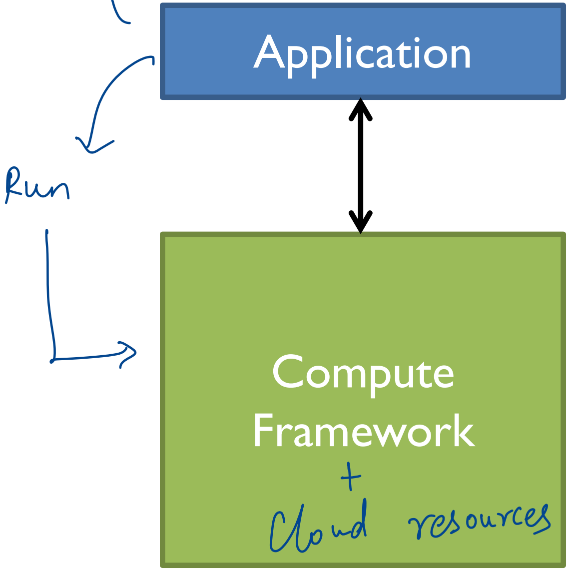
Private Cluster

…

allocate, downloading, configuration etc.

Language Integrated

simple programming model



"SERVERLESS" COMPUTING

fixed size containers for fixed time duration

Ephemeral compute resources

→ Fixed configuration

~~300~~ 900 seconds single-core

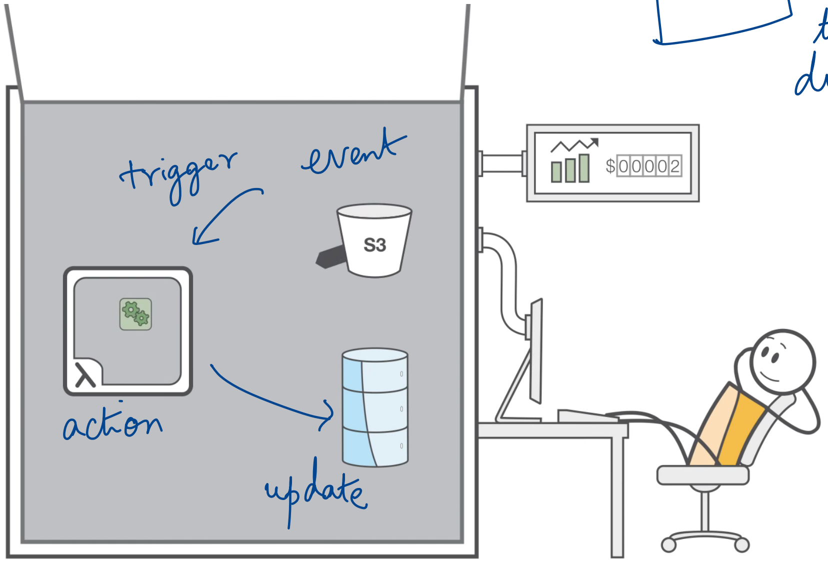
~~512~~ 512-10240 MB in /tmp → disk space

~~3~~ 10GB RAM → upto fixed memory

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

Support for containers

Docker to pull in dependencies



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

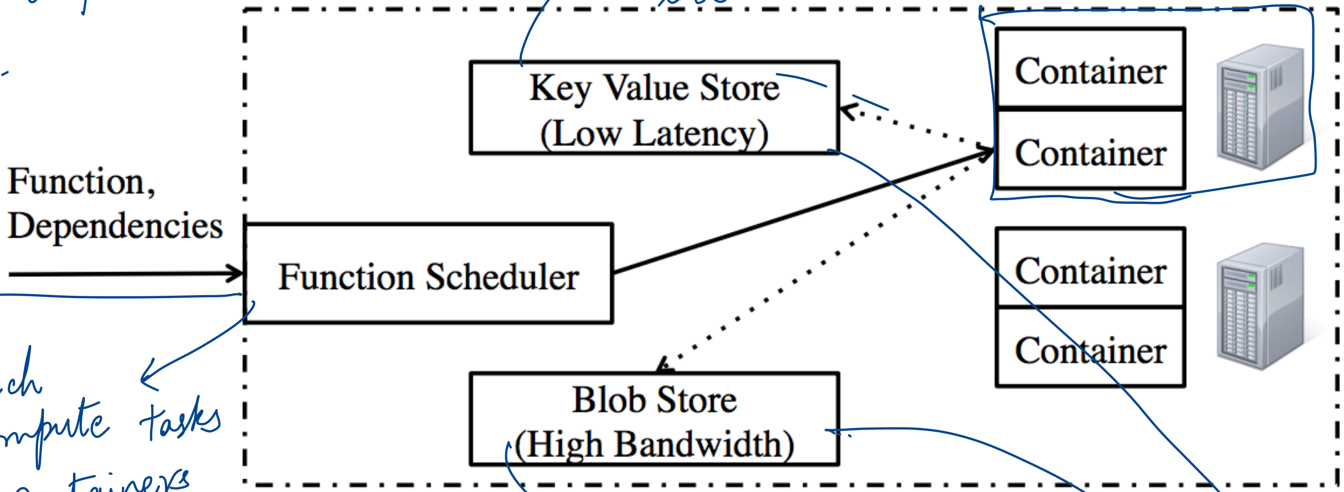
Usable data analytics →

← Serverless computing ↳ (900s)

Operators cannot save state
↳ each time output persisted.

storage engines
intermediate state

compute layer
↓
operators



launch compute tasks on containers

inputs / outputs

long running services

Function Scheduler

Application side → can run on your laptop

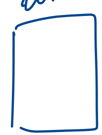
↳ Tracking which operator should be run next

MR() → map() finished | start reduce tasks

launch container()

Infrastructure side (AWS)

clever decision
- to minimize cost etc.



which machine

resources that are use

configuration: cloud API key
S3 bucket
...

PYWREN API

python add.py

add.py

```
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]
```

library

function

function

list of inputs

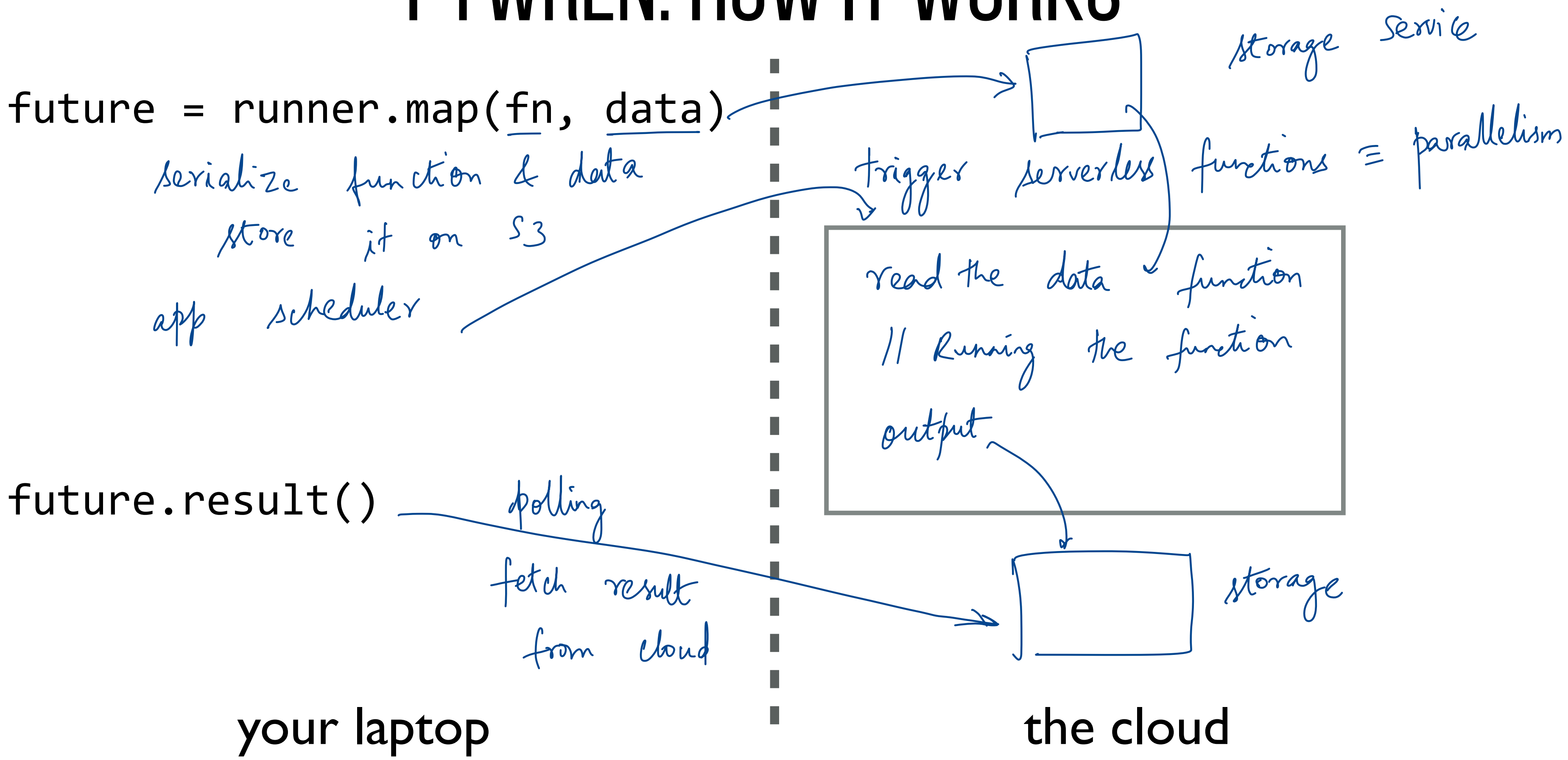
Go from laptop to cloud and back

The output is as expected:

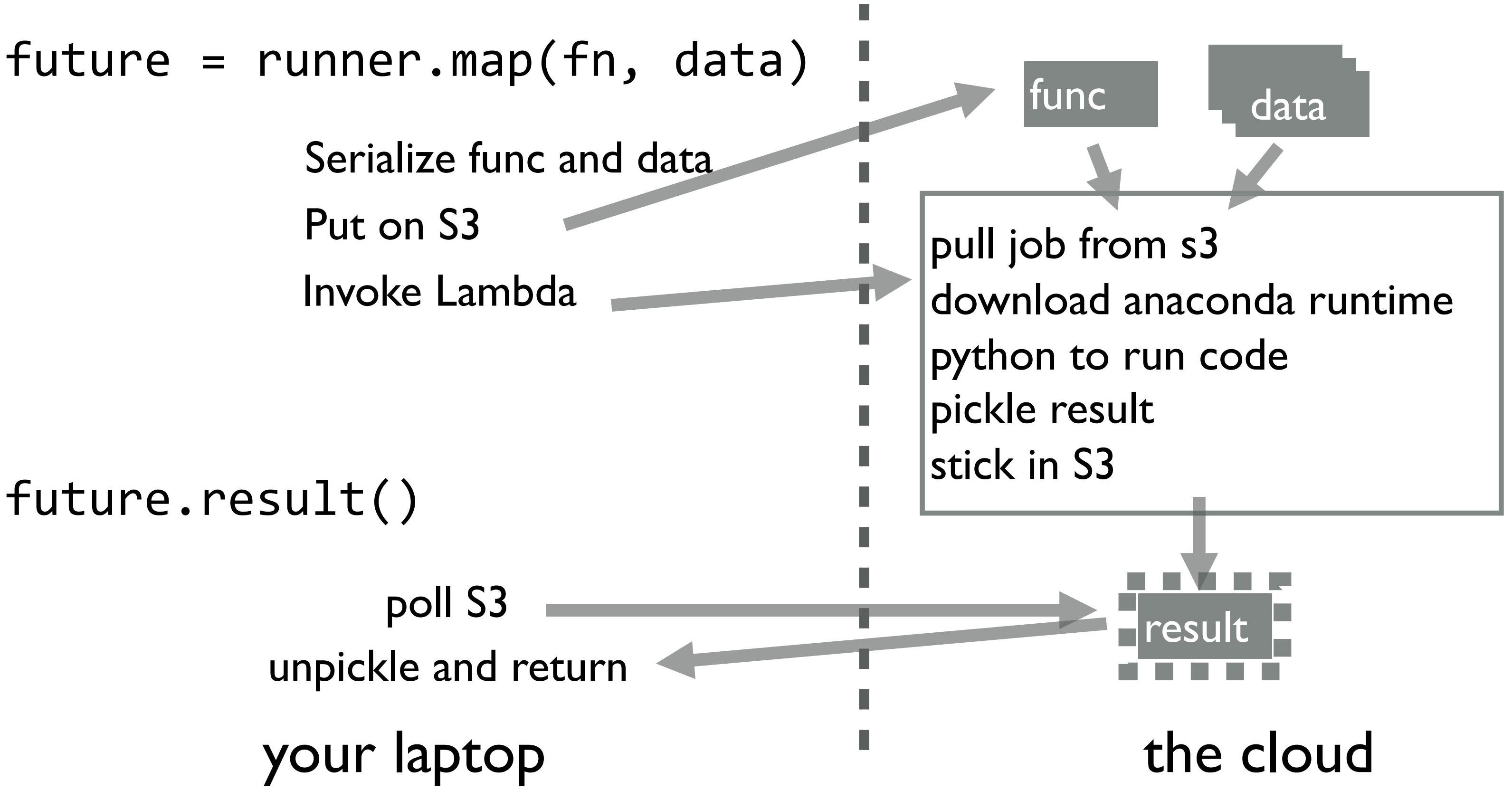
2000 tasks → incremental results

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

PYWREN: HOW IT WORKS



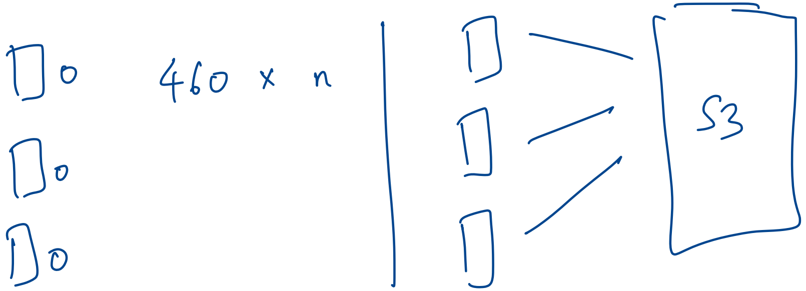
HOW IT WORKS


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 ?

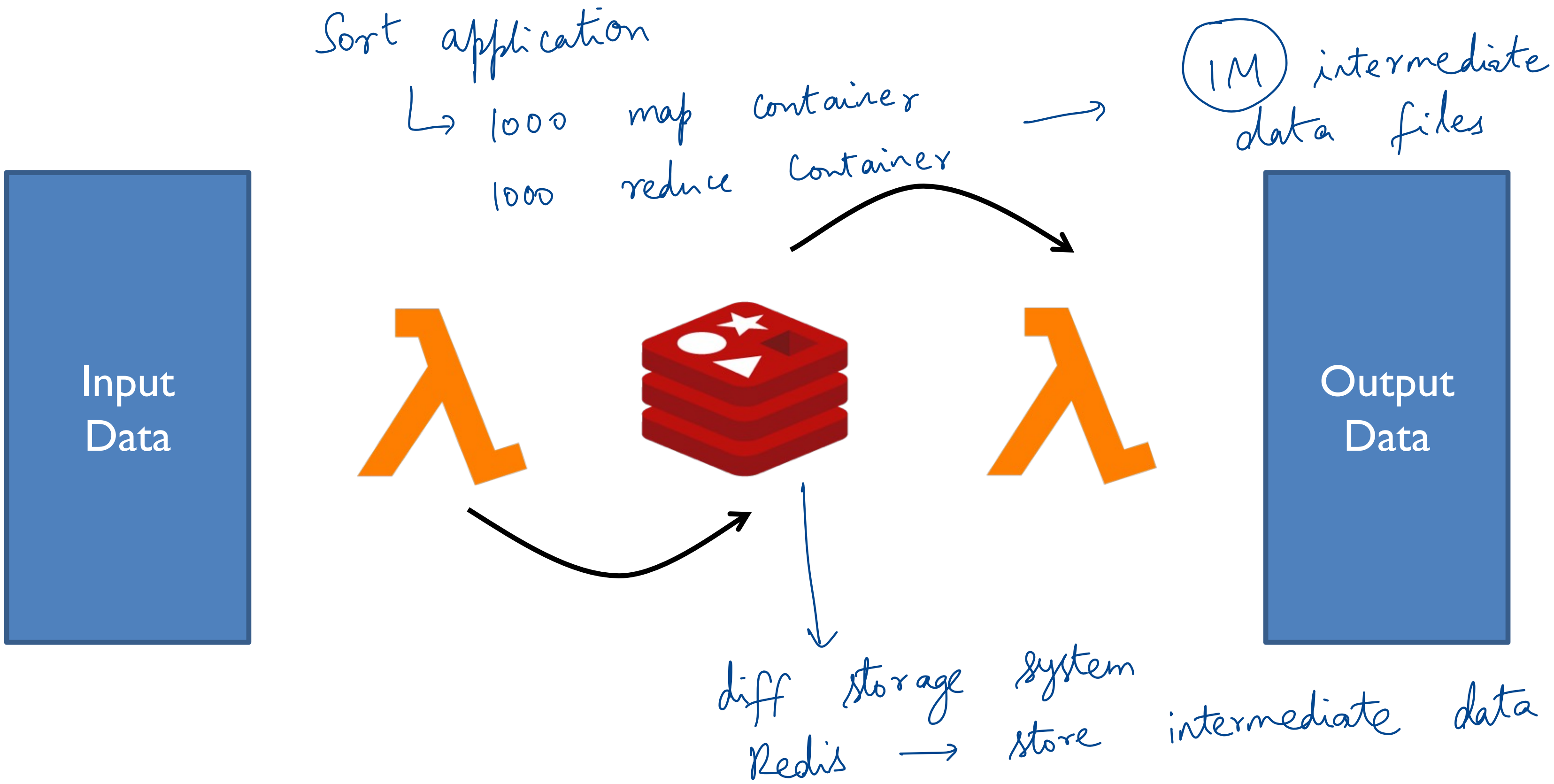
What are the trade-offs ?

- All data accesses are remote accesses
- Network / remote access is competitive to reading data from SSD → slower than 4 SSDs



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 ?



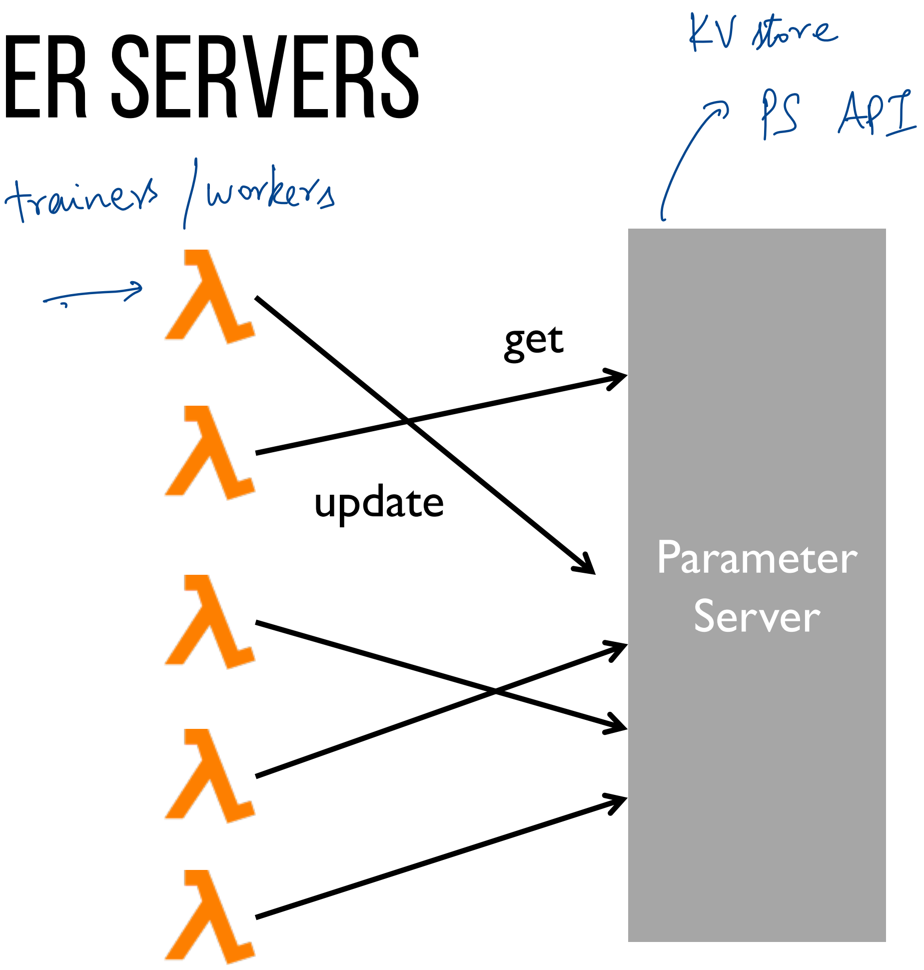
PARAMETER SERVERS

Sparse ML models
↳ subset parameters for 1 iteration

Use lambdas to run "workers"

Parameter server as a service ?

Lambda / Container
limited CPU / memory resources.



WHEN SHOULD WE USE SERVERLESS ?

Yes!	Maybe not ?
- Tasks are independent ↳ random, seed in parallel	- Long running tasks checkpoint overheads
- When latency is not a concern (service provider queue)	- Iterative algorithms reuse outputs from memory
- Cluster utilization is low	- Code / Dependencies → long install and initialize

Shuffle intensive ?

SUMMARY

Motivation: Usability of big data analytics

Approach: Language-integrated cloud computing

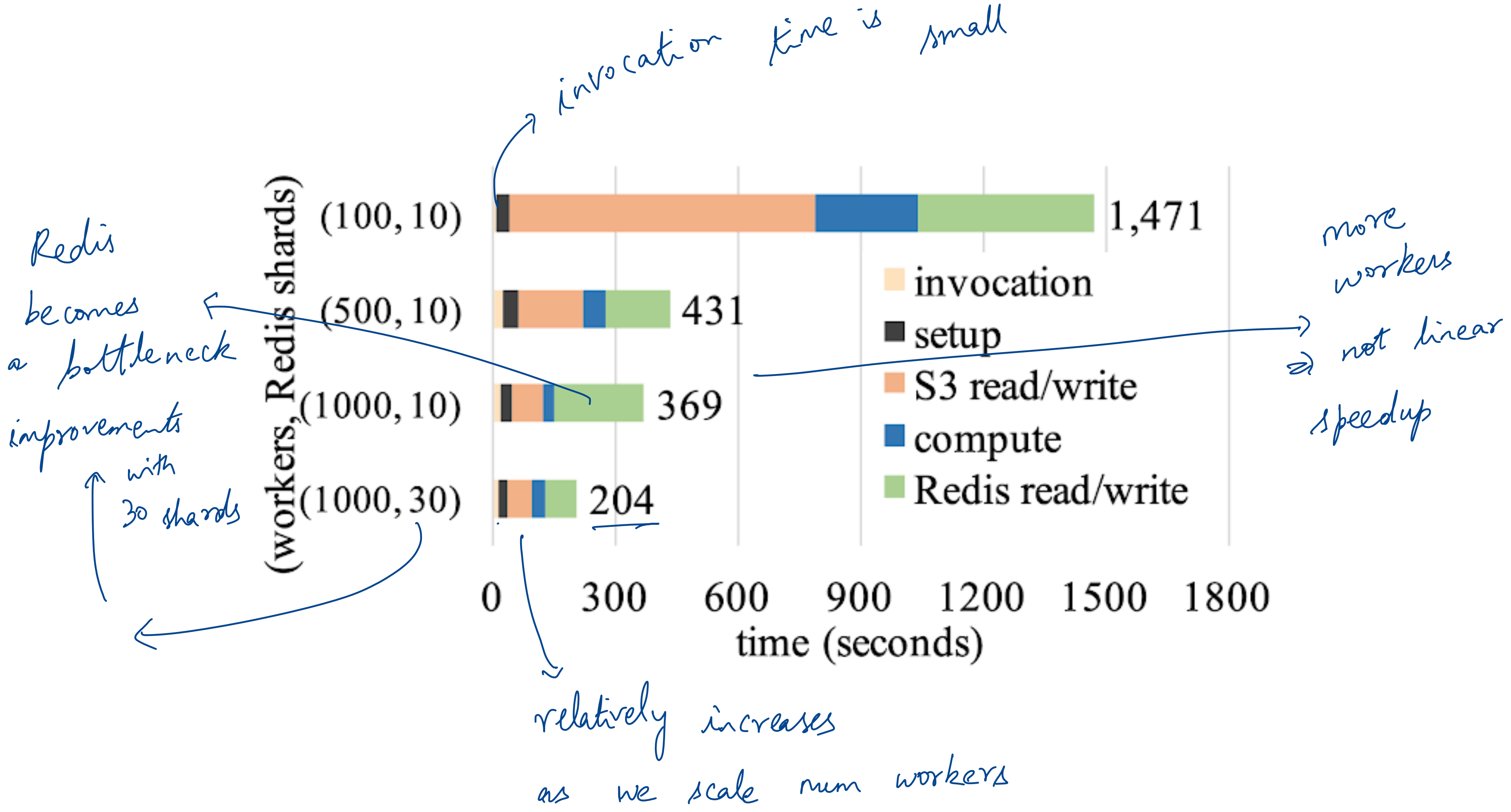
Features

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

Open question on scheduling, overheads

DISCUSSION

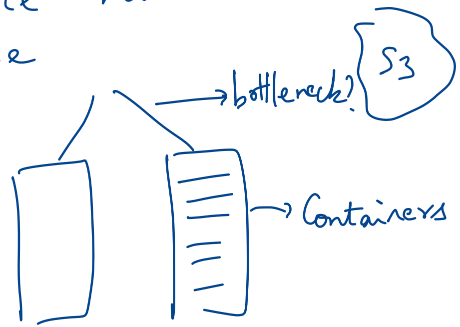
https://forms.gle/GF1kkME52tvDRdDC8


(workers, Redis shards)
(100, 10)
(500, 10)
(1000, 10)
(1000, 30)
1,471
431
369
204
invocation
setup
S3 read/write
compute
Redis read/write
0
300
600
900
1200
1500
1800
time (seconds)
invocation time is small
Redis becomes a bottleneck
improvements with 30 shards
more workers ⇒ not linear speedup
relatively increases as we scale num workers

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?

Infrastructure scheduler

→ all containers " same compute " → balance network bandwidth use



→ Fairness ?

↳ SLO in terms of waiting time ?

→ Pricing → higher tier for better responsiveness

- Base Images might vary → very light / heavy → Distribute images

NEXT UP

Happy Thanksgiving!

Next steps:

- Mid-semester project check-in, Nov 23rd
- After break: Owl, TPU papers
- Midterm 2, Dec 6^{th}