Hello!

CS 744: PYWREN

Shivaram Venkataraman
Fall 2020
Project checkins due Nov 20th
In-class project presentations
Dec 8th and Dec 10th
Project grade breakdown
Intro: 5%
Mid-semester checkin: 5%
Presentation: 10%
Final Report: 10%

Tonight deadline for submitting regrade requests for Midterm 1
NEW HARDWARE MODELS

Implications → Society
of Big Data
analysis

Cloud computing evolution

Big Data Systems

Computation Engines
Sched
Storage

New hardware
SERVERLESS COMPUTING

↓

No servers??
**MOTIVATION: USABILITY**

### Data Scientist

Data Scientist

### Analysis

Analysis

#### What instance type?

What instance type?

#### What base image?

What base image?

#### How many to spin up?

How many to spin up?

#### What price? Spot?

What price? Spot?
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 short time
- Varying compute needs
- No admin or software (yes)
ABSTRACTION LEVEL?

So far in this course

Application

Compute Framework

Hardware

Logistic Regression

Spark

Amazon EC2

CloudLab

Private Cluster

...
STATELESS DATA PROCESSING

Intermediate state in Spark/MR was on local disk.

Local storage is ephemeral, so intermediate state needs to be remote!

Function, Dependencies → Function Scheduler → Key Value Store (Low Latency) → Blob Store (High Bandwidth) → Container → Compute

Resource limits
“SERVERLESS” COMPUTING

- Submit a function (lambda) to be executed

300-900 seconds single-core

512 MB in /tmp

3GB RAM

Python, Java, node.js

Provided by Cloud Provider

Some processing for each upload

Time bound

Storage

Memory

Cloud database

Lambda

Google Cloud Platform

CLOUD FUNCTIONS ALPHA

A serverless platform for building event-based microservices

Microsoft Azure

Azure Functions

Process events with a serverless architecture
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()
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

result

future.result()

poll S3
unpickle and return

your laptop

the cloud
STATELESS FUNCTIONS: WHY NOW?

What are the trade-offs?

→ Need more network I/O

All the data is read over network!

→ But network BW is pretty good! Comparable to local SSD BW!

→ Bottleneck could be S3?

<table>
<thead>
<tr>
<th>Storage Medium</th>
<th>Write Speed (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD on c3.8xlarge</td>
<td>208.73</td>
</tr>
<tr>
<td>SSD on i2.8xlarge</td>
<td>460.36</td>
</tr>
<tr>
<td>4 SSDs on i2.8xlarge</td>
<td>1768.04</td>
</tr>
<tr>
<td>S3</td>
<td>501.13</td>
</tr>
</tbody>
</table>
MAP AND REDUCE?

Input Data → Map → Shuffle phase in MR is now being done using Redis → Reduce → Output Data

Sort benchmark
same as MapReduce paper

Redis key-value store - memory

bucket keys into ranges

small files not good for blob store like S3

0 - 100 key1, key2...
[101 - 200] keys...
PARAMETER SERVERS

Use lambdas to run “workers”

Parameter server as a service?

How do you profile or measure function requirements?

→ Run function locally, use profiler?

→ Checkpoint (before time limit) and resume?

Sparse models → Ad click prediction

→ Ad click prediction

input → read

compute gradient → update

ML model stored

→ Redis or VMs, etc.

Parameter Server

Fault tolerance

Recent work!
WHEN SHOULD WE USE SERVERLESS?

Yes!
Use when we need elasticity
Use when you don’t need fine grained comm. across workers
→ Not all lambdas might be active at the same time!

Maybe not?
Not use serverless when you need local state (actors)
Iterative workloads might need state from prev. iteration
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/PAMDKmwHepmPWWDrBA
Increasing workers by 5x → compute, storage

<table>
<thead>
<tr>
<th>(workers, Redis shards)</th>
<th>invocation</th>
<th>setup</th>
<th>compute</th>
<th>S3 read/write</th>
<th>Redis read/write</th>
</tr>
</thead>
<tbody>
<tr>
<td>(100, 10)</td>
<td>1,471</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(500, 10)</td>
<td>431</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1000, 10)</td>
<td>369</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1000, 30)</td>
<td>204</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Hard to know how to choose new partitions

more shards reduces time to read/write to Redis

compute is very short compared to I/O

Increasing workers by 5x ↑ != 5x improvement
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? How would you go about addressing them?

- Mapping lambda functions → machines
  How do we do this?

- Locality? Does one lambda talk to some Redis shard?
  Can we infer it?

- When to schedule a new container / when do we reuse?

- Need to find opt configuration? Use ML?

- Resource requirements are fixed! 900s, 1 core, up to 3 GB
OPEN QUESTIONS

- Scalable scheduling: Low latency with large number of functions?

- Debugging: Correlate events across functions?

- Launch overheads: Fraction of time spent in setup (OpenLambda)

- Resource limits: 15 minute AWS Lambda (Oct 2018)

See you on Thursday!
Cold Starts

→ App side

→ Sched. side

Containers would be warm for 5 mins if you ran one within 5 mins

Azure - policy paper

ATC 2020

Memory 512MB to 3GB

VM

EBS

Block service

/home

write

mount

unmount

mount

unmount

{


}