Happy Thanksgiving!?
NEW HARDWARE MODELS
Serverless Computing

Compute Accelerators

Infiniband Networks

Non-Volatile Memory
SERVERLESS COMPUTING
**MOTIVATION: USABILITY**

### EC2 EC2 Instance Comparison

<table>
<thead>
<tr>
<th>Region</th>
<th>Size</th>
<th>Hourly Price</th>
<th>On-Demand Cost</th>
<th>Spot Price</th>
</tr>
</thead>
</table>

### Instance Details

<table>
<thead>
<tr>
<th>Name</th>
<th>Memory</th>
<th>Storage</th>
<th>Network Performance</th>
<th>Network Performance</th>
<th>Spot Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>m5.large</td>
<td>16 GB</td>
<td>80 GB</td>
<td>High</td>
<td>High</td>
<td>$0.16/GB</td>
</tr>
<tr>
<td>m5.xlarge</td>
<td>32 GB</td>
<td>320 GB</td>
<td>High</td>
<td>High</td>
<td>$0.32/GB</td>
</tr>
<tr>
<td>m5.2xlarge</td>
<td>64 GB</td>
<td>640 GB</td>
<td>High</td>
<td>High</td>
<td>$0.64/GB</td>
</tr>
<tr>
<td>m5.4xlarge</td>
<td>128 GB</td>
<td>1280 GB</td>
<td>High</td>
<td>High</td>
<td>$1.28/GB</td>
</tr>
<tr>
<td>r5.large</td>
<td>16 GB</td>
<td>320 GB</td>
<td>High</td>
<td>High</td>
<td>$0.25/GB</td>
</tr>
<tr>
<td>r5.xlarge</td>
<td>32 GB</td>
<td>640 GB</td>
<td>High</td>
<td>High</td>
<td>$0.50/GB</td>
</tr>
<tr>
<td>r5.2xlarge</td>
<td>64 GB</td>
<td>1280 GB</td>
<td>High</td>
<td>High</td>
<td>$1.00/GB</td>
</tr>
<tr>
<td>r5.4xlarge</td>
<td>128 GB</td>
<td>2560 GB</td>
<td>High</td>
<td>High</td>
<td>$2.00/GB</td>
</tr>
</tbody>
</table>

### Base Image

- **Base Image:** SUSE Linux Enterprise Server 15 Live Guest

### Spot Price

- **Spot Price:** $0.01/GB

---

**What instance type?**

**What base image?**

**How many to spin up?**

**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 cost for servers
ABSTRACTION LEVEL?

Application

Compute Framework

Hardware

Logistic Regression

Spark

Amazon EC2
CloudLab
Private Cluster
...

Application

Compute Framework
STATELESS DATA PROCESSING

Function Schedules

- Key Value Store (Low Latency)
- Blob Store (High Bandwidth)

Containers
“SERVERLESS” COMPUTING

300-900 seconds single-core

512 MB in /tmp

3GB RAM

Python, Java, node.js
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()
future = runner.map(fn, data)

Serialize func and data
Put on S3
Invoke Lambda

call

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?

<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?
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/Y9AFUpvVBA7LpKqh7
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?
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)