- Pick up midterm papers after class
- Course Projects: sign up for final round of meetings!
NEW HARDWARE MODELS
Serverless Computing

Compute Accelerators

Infiniband Networks

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

**What instance type?**

**What base image?**

**How many to spin up?**

**What price? Spot?**
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 durations
- Varying compute needs
- No admin overhead (yes)

Why is there no “cloud button”?
ABSTRACTION LEVEL?

Application -> Compute Framework -> Hardware

Logistic Regression
Spark
Amazon EC2
CloudLab
Private Cluster
...

Application
Compute Framework
“SERVERLESS” COMPUTING

900 seconds single-core

512 MB in /tmp

3GB RAM

Python, Java, node.js
PYWREN ARCHITECTURE

Stateless functions

Move state to elastic storage systems!
```python
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]
```
future = runner.map(fn, data)

- Serialize func and data
- Put on S3
- Invoke Lambda

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

the cloud
STATELESS FUNCTIONS: WHY NOW?

What are the trade-offs?

- Fine-grained scheduling
- Fault Tolerance
- Disk Locality

<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 → λ → sort partition → λ → Output Data

Range partition
PARAMETER SERVERS

Use lambdas to run “workers”

Parameter server as a service?
WHEN SHOULD WE USE SERVERLESS?

Yes!

Embarrassingly parallel

Fine-grained compute

Varying degree of parallelism

Maybe not?

Fine-grained dependencies

Shuffle/Locality intensive?

Hard to decompose
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)
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