CS 744: PYWREN

Shivaram Venkataraman
Fall 2019
Happy Thanksgiving!?
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?
Why is there no "cloud button"?
ABSTRACTION LEVEL?

- Application
  - Compute Framework
    - Hardware
    - Logistic Regression
      - SQL query
      - Spark
      - Tensorflow
      - Amazon EC2
      - CloudLab
      - Private Cluster
      - ...
STATELESS DATA PROCESSING

- No server storing intermediate state
- Functional program, linear scalability?
- Key Value Store (Low Latency)
- Blob Store (High Bandwidth)
- Function Scheduler

Assign or spin up containers
“SERVERLESS” COMPUTING

300-900 seconds single-core
512 MB in /tmp
3GB RAM
Python, Java, node.js
```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]
```

Cloud pickle in 2010s

Closure capture

PySpark
PYWREN: HOW IT WORKS

future = runner.map(fn, data)

data

fn

S3

future.result()

results

your laptop

the cloud
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

result

worker fails

worker

stateless
What are the trade-offs?

- Overhead from fetching state
  - Locality
- Network BW is pretty good!
- Blob stores - large file size
  - Fine grained computation → time vs memory
- Rate limits!

<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>

Network BW
MAP AND REDUCE?

Input Data

Sort Benchmark

ephemeral KV store
better support for small files

partition range → too many small files

Output Data

lambda

lambda

sorting
PARAMETER SERVERS

Use lambdas to run "workers"

Parameter server as a service?

Fault Tolerance

Sharding, replication 99.99% uptime

Consistency: Wait for some iterations

Storage Keys Version

to 100 to 10

Parameter Server

goto update

goto get

WHEN SHOULD WE USE SERVERLESS?

Yes!
- Prototyping → not much config
- Good for testing
- Streaming workflow
- Varying degree of parallelism

Maybe not?
- Don’t need elasticity
- Too frequent coordination
- Non-deterministic
- Latency might include spin up?
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
S3 reads/write parallelism & scale

500 → 1000 redis increases

Compute is small most of our time going in I/O

setup is a constant initially could become significant with scale

workers, Redis shards

(100, 10)

(500, 10)

(1000, 10)

(1000, 30)

invocation
setup
S3 read/write
compute
Redis read/write

0 300 600 900 1200 1500 1800
time (seconds)

Same 1000 workers, compute increases?
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?

Challenges
- Lots of requests, scheduler shouldn't be a bottleneck
- Sharing property
- Low latency, SLA
- Information gap

Cloud

User

How many requests
Where do you place the cluster of machines

Containers
Techniques

- Exposing more information \(\rightarrow\) Usability!
  - Profiling

- Guaranteed allocation = new model
  - Extend 15 mins

- Migration can be useful \(\rightarrow\) 15 mins
  - Containers

- Hierarchical schedulers
  - Per region, scheduler \(\rightarrow\) Global scheduler
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)