Outline

- Background
- Research Questions
- Experimental Workloads
- Experiments/Evaluation
- Conclusions
Serverless Computing

Why Serverless Computing?
Many features of distributed systems, that are challenging to deliver, are provided automatically

...they are built into the platform
Serverless Platforms

- AWS Lambda
- Azure Functions
- IBM Cloud Functions
- Google Cloud Functions
- Apache OpenWhisk
- Fn (Oracle)

AWS Lambda
Serverless Computing Platform

Serverless Computing
Deploy Applications Without Fiddling With Servers

Image from: https://mobisoftinfotech.com/resources/blog/serverless-computing-deploy-applications-without-fiddling-with-servers/
### Smith Waterman Example

- Applies dynamic programming to find best local sequencing alignment of two DNA/RNA samples
  - Embarrassingly parallel, each task can run in isolation
  - Use case for GPU acceleration

**Example:** Compare 20,336 protein sequences
- Python client, C execution engine

- Intel i5-7200U 2.5 GHz laptop client (2-core, 4-HT): 8.7 hrs

- AWS Lambda, same laptop as client: 2.2 minutes
  - Partitions 20,336 sequences into 41 sets
  - Execution cost: $\sim 82\$ (237x speed-up)
Going Serverless: Evaluating the Potential of Serverless Computing for Environmental Modelling Application Hosting

Vendor architectural lock-in

- Serverless software architecture requires external services/components

Client

- Lambda is triggered
- API Gateway
- S3
- DYNAMODB

Example: Weather Application

- Front-end code for weather app hosted in S3
- User clicks on link to get local weather information
- App makes REST API call to endpoint
- Lambda runs code to retrieve local weather information and returns data back to user

- Increased dependencies $\rightarrow$ increased hosting costs

Images credit: aws.amazon.com
Pricing Obfuscation

• **VM pricing:** hourly rental pricing, billed to nearest second intuitive...

• **Serverless Computing:**

  **AWS Lambda Pricing**
  **FREE TIER:**
  - first 1,000,000 function calls/month → FREE
  - first 400 GB-sec/month → FREE
  
  • Afterwards: $0.0000002 per request
  $0.000000208 to rent 128MB / 100-ms

Memory Reservation Question...

• Lambda memory reserved for functions
• UI provides “slider bar” to set function’s memory allocation
• CPU power coupled to slider bar:
  “every doubling of memory, doubles CPU...”
• But how much memory do model services require?
Service Composition

- How should model code be composed for deployment to serverless computing platforms?

  - Recommended practice: Decompose into many microservices
  - Platform limits: code + libraries ~256MB
  - How does composition impact the number of function invocations, and memory utilization?

Infrastructure Freeze/Thaw Cycle

- Unused infrastructure is deprecated
  - But after how long?
- Infrastructure: VMs, “containers”
- Provider-COLD / VM-COLD
  - “Container” images - built/transferred to VMs
- Container-COLD
  - Image cached on VM
- Container-WARM
  - “Container” running on VM
Serverless Computing Challenges for Environmental Modelling

- Vendor architectural lock-in
- Pricing obfuscation
- Memory reservation
- Service composition
- Infrastructure freeze/thaw cycle

Outline

- Background
- Research Questions
- Experimental Workloads
- Experiments/Evaluation
- Conclusions
Research Questions

Precipitation Runoff Modeling System (PRMS) on AWS Lambda:

RQ1: **Infrastructure**
What are the performance implications of memory reservation size?

RQ2: **Scaling Performance**
How does performance change when increasing the number of concurrent requests?

---

Research Questions - 2

Precipitation Runoff Modeling System (PRMS) on AWS Lambda:

RQ3: **Cost**
What are the costs of hosting model services using AWS Lambda, a serverless computing cloud platform?
Outline

- Background
- Research Questions
- **Experimental Workloads**
- Experiments/Evaluation
- Conclusions

AWS Lambda

PRMS Modeling Service

- PRMS: deterministic, distributed-parameter model
- Evaluate impact of combinations of precipitation, climate, and land use on stream flow and general basin hydrology (Leavesley et al., 1983)
- Java based PRMS, Object Modelling System (OMS) 3.0
- Approximately ~11,000 lines of code
- Model service is 18.35 MB compressed as a Java JAR file
- Data files hosted using Amazon S3 (object storage)

**Goal:** quantitate performance and cost implications of memory reservation size and scaling for model service deployment to AWS Lambda
Serverless Computing: An Investigation of Factors Influencing Microservice Performance

Wes Lloyd, Shruti Ramesh, Swetha Chinthalapati, Lan Ly, Shrideep Pallickara

April 20, 2018

Institute of Technology, University of Washington, Tacoma, Washington USA

IC2E 2018: IEEE International Conference on Cloud Engineering

Available at: https://goo.gl/tZvfCH

PRMS Lambda Testing

Client:
c4.2xlarge or c4.8xlarge
(8 core) (36 core)

BASH: GNU Parallel
Multi-thread client script "partest"

Fixed-availability zone: EC2 client / Lambda server us-east-1e

Max service duration: < 30 seconds
Memory: 256 to 3008MB

Up to 100 concurrent synchronous requests
Results of each thread traced individually
AWS Lambda Testing

Client:
c4.2xlarge or c4.8xlarge (8 core) (36 core)

Automatic Metrics Collection:
- New vs. Recycled Containers/VMs
- # of requests per container/VM
- Avg. performance per container/VM
  - Avg. performance workload
  - Standard deviation of requests per container/VM
- Container Identification
  - UUID \(\rightarrow\) /tmp file
- VM Identification
  - btime \(\rightarrow\) /proc/stat
- Linux CPU metrics

Outline

- Background
- Research Questions
- Experimental Workloads
- Experiments/Evaluation
- Conclusions
RQ-1: Infrastructure

*Infrastructure*
What are the performance implications of memory reservation size?

---

RQ-1: AWS Lambda Memory Reservation Size

<table>
<thead>
<tr>
<th>PRMS AWS Lambda Performance (100 concurrent requests)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Speedup (256 → 3008 MB):</td>
</tr>
<tr>
<td>4.3 X 8-vCPU client</td>
</tr>
<tr>
<td>10.1 X 36-vCPU client</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Execution time (ms)</th>
<th>c4.2xlarge client</th>
<th>c4.8xlarge client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedup @ 256MB</td>
<td>4.3x</td>
<td>10.1x</td>
</tr>
<tr>
<td>Speedup @ 1024MB</td>
<td>1.3x</td>
<td>1.9x</td>
</tr>
<tr>
<td>Speedup @ 1536MB</td>
<td>1.14x</td>
<td>1.4x</td>
</tr>
<tr>
<td>Speedup @ 2048MB</td>
<td>1.06x</td>
<td>1.2x</td>
</tr>
</tbody>
</table>

June 27, 2018
RQ-1: AWS Lambda

Memory Reservation Size - Infrastructure

c4.2xlarge – average of 8 runs

- Many more VMs available when memory > 1536 MB
- 8 vCPU client struggles to generate 100 concurrent requests @ >= 1024MB

Memory Reservation Size (MB)

# of containers / VMs

Higher memory size guarantees access to more VMs
- c4.2xlarge client: 3.3x more VMs (low to high)
- c4.8xlarge client: 6.8x more VMs (low to high)

AWS Lambda Hosting Infrastructure - PRMS Service

AWS Lambda - Service Load Balancing

c4.2xlarge – average of 8 runs

- Higher memory size guarantees access to more VMs
- c4.2xlarge client: 3.3x more VMs (low to high)
- c4.8xlarge client: 6.8x more VMs (low to high)
RQ-2: Scaling Performance

How does performance change when increasing the number of concurrent users?

*(scaling-up, totally cold, and warm)*

---

RQ-2: AWS Lambda
PRMS Scaling Performance

When slowly increasing the number of clients, performance stabilizes after ~15-20 concurrent clients.
RQ-2: AWS Lambda Infrastructure for Scaling

Load Balancing:
- @512MB: 5/6 requests per VMs
- @1664MB (<82): 2 requests per VM
- @1664MB (>82): 6 requests per VM

RQ-2: AWS Lambda Cold Scaling Performance

To Do:
- @1664MB Cold Scaling Test
- @3008MB Cold Scaling Test
  (estimate ~16sec initialization)

@3008MB model execution is 3.1x faster than @512MB, is initialization ??
RQ-3: Hosting Costs

What are the costs of hosting PRMS using AWS Lambda serverless computing?

RQ-3: VM (EC2) Hosting

1,000,000 PRMS runs

- Using a 2 vCPU c4.large EC2 VM
- Estimated time: 347.2 hours, **14.46 days**
  - Assume average exe time of 2.5 sec/run
- Hosting cost @ 10¢/hour = **$34.72**
RQ-3: AWS Lambda Hosting

1,000,000 PRMS runs

<table>
<thead>
<tr>
<th>Memory MB</th>
<th>GB-sec/run</th>
<th>Runs-free tier</th>
<th>GB-sec/1,000,000 runs</th>
<th>Lambda Cost</th>
<th>Execution hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>6.53</td>
<td>61.268</td>
<td>6,528,655</td>
<td>$107.62</td>
<td>7.25</td>
</tr>
<tr>
<td>384</td>
<td>4.90</td>
<td>81.674</td>
<td>4,897,523</td>
<td>$80.14</td>
<td>3.63</td>
</tr>
<tr>
<td>512</td>
<td>4.08</td>
<td>98.120</td>
<td>4,076,625</td>
<td>$66.20</td>
<td>2.26</td>
</tr>
<tr>
<td>640</td>
<td>3.30</td>
<td>92.973</td>
<td>4,032,330</td>
<td>$70.04</td>
<td>1.91</td>
</tr>
<tr>
<td>768</td>
<td>4.51</td>
<td>88.669</td>
<td>4,511,183</td>
<td>$73.59</td>
<td>1.67</td>
</tr>
<tr>
<td>896</td>
<td>4.52</td>
<td>88.488</td>
<td>4,520,364</td>
<td>$73.75</td>
<td>1.44</td>
</tr>
<tr>
<td>1024</td>
<td>4.95</td>
<td>80.742</td>
<td>4,954,080</td>
<td>$81.09</td>
<td>1.38</td>
</tr>
<tr>
<td>1152</td>
<td>5.12</td>
<td>78.140</td>
<td>5,119,043</td>
<td>$83.88</td>
<td>1.26</td>
</tr>
<tr>
<td>1280</td>
<td>5.18</td>
<td>77.213</td>
<td>5,180,475</td>
<td>$84.92</td>
<td>1.15</td>
</tr>
<tr>
<td>1408</td>
<td>5.34</td>
<td>74.897</td>
<td>5,340,679</td>
<td>$87.62</td>
<td>1.08</td>
</tr>
<tr>
<td>1536</td>
<td>5.39</td>
<td>74.254</td>
<td>5,386,950</td>
<td>$88.40</td>
<td>1.00</td>
</tr>
<tr>
<td>1664</td>
<td>5.67</td>
<td>70.582</td>
<td>5,667,171</td>
<td>$93.13</td>
<td>0.97</td>
</tr>
<tr>
<td>1792</td>
<td>5.78</td>
<td>69.192</td>
<td>5,701,055</td>
<td>$95.05</td>
<td>0.92</td>
</tr>
<tr>
<td>1920</td>
<td>6.10</td>
<td>66.607</td>
<td>6,096,919</td>
<td>$100.36</td>
<td>0.90</td>
</tr>
<tr>
<td>2048</td>
<td>6.33</td>
<td>63.209</td>
<td>6,328,240</td>
<td>$104.25</td>
<td>0.88</td>
</tr>
<tr>
<td>2176</td>
<td>6.58</td>
<td>60.748</td>
<td>6,584,525</td>
<td>$108.56</td>
<td>0.86</td>
</tr>
<tr>
<td>2368</td>
<td>6.69</td>
<td>59.761</td>
<td>6,693,277</td>
<td>$110.38</td>
<td>0.80</td>
</tr>
<tr>
<td>2496</td>
<td>6.69</td>
<td>59.756</td>
<td>6,693,911</td>
<td>$110.39</td>
<td>0.76</td>
</tr>
<tr>
<td>2624</td>
<td>6.92</td>
<td>57.827</td>
<td>6,917,187</td>
<td>$114.14</td>
<td>0.75</td>
</tr>
<tr>
<td>2752</td>
<td>7.38</td>
<td>54.212</td>
<td>7,378,504</td>
<td>$121.88</td>
<td>0.76</td>
</tr>
<tr>
<td>2880</td>
<td>7.56</td>
<td>52.931</td>
<td>7,557,019</td>
<td>$124.87</td>
<td>0.75</td>
</tr>
<tr>
<td>3008</td>
<td>7.56</td>
<td>52.909</td>
<td>7,560,214</td>
<td>$124.92</td>
<td>0.71</td>
</tr>
</tbody>
</table>

AWS Lambda @ 512MB Enables execution of 1,000,000 PRMS model runs in **2.26 hours** @ 1,000 runs/cycle - for **$66.20**

With no setup (creation of VMs)
Outline

- Background
- Research Questions
- Experimental Workloads
- Experiments/Evaluation
- Conclusions

Conclusions

- **RQ-1 Memory Reservation Size:**
  - Increasing to 3GB provided a **10x speedup**
  - ~7x more VMs leveraged at high memory

- **RQ-2 Scaling Performance:**
  - Slow scale up: stable performance stabilizes after ~15-20 concurrent clients.
  - COLD performance slow at low memory settings
Conclusions - 2

- **RQ-3 Cost**: 1,000,000 PRMS model runs
  - Traditional 2-core VM: **14.5 days, $35**
  - AWS Lambda 512MB: **~2.3 hours, $66**
  - AWS Lambda 3008MB: **42 minutes, $125**
  - **No VM/docker configuration/setup**

June 27, 2018

Questions