Spock: Exploiting Serverless Functions for SLO and Cost Aware Resource Procurement in Public Cloud

Jashwant Gunasekaran, Prashanth Thinakaran, Mahmut Kandemir, Bhuvan Urgaonkar, George Kesidis, Chita Das

Computer Science and Engineering
The Pennsylvania State University
The Last Supper of cloud clients

WE JUST GOT THE AWS BILL FOR THIS MONTH

But.. I have the money to pay for it this time

Probably our private key is compromised

But I turned off my VM Instances
That Last AWS bill
Spock: Cost Aware Resource Procurement in Public Clouds using Serverless
Whose problem are we solving?
Outline

• Elastic Web Services

• VM-based Resource Procurement

• Serverless Functions

• Cost of VMs vs Cloud Functions

• Spock Hybrid Elastic Scaling

• Implementation and Evaluation

• Results
Elastic Web Services

- Short lived queries
- Strict SLO
- Varying resource demands
- Stateless

Resources Required
- acquired/released on demand
- Average to Peak ratio is high

Typical example?
ML based web services
ML Inference Engine

User Queries → Inference endpoints → Elastic Load Balancer → Procured Resources

Service 1

Service 2

Service N
Outline

• Elastic Web Services

• VM-based Resource Procurement

• Serverless Functions

• Spock Hybrid Elastic Scaling

• Implementation and Evaluation

• Results
VM-based Procurement

Users → API Gateway → Load Balancer → Model serving VMs (EC2 instances) → Model Archive

Elastic Block store
VM-based Procurement

- Initial pool of active VMs
- Procure more VMs on demand
- Autoscaling during request surge
Disadvantages

- Very long VM startup times (5s-50s)

Possible alternative/s?

- Under-provisioned during sudden surge
Outline

• Elastic Web Services

• VM-based Resource Procurement

• **Serverless Functions**

• Spock Hybrid Elastic Scaling

• Implementation and Evaluation

• Results
Serverless Functions
Serverless Functions

- Pay per second
- Cost efficient
- Scale instantaneously
- Intermittent SLA violations

But, is serverless a panacea?
Constant arrival rate

- Constant arrival rate
- Cost compared under iso-performance
- All requests have similar SLA compliance
- VMs are 100% utilized
Varying arrival rate

- Trace based arrival rate
- Each request is an ML inference for caffenet-model

Cost-effective Solution?

- All requests have similar SLA compliance
- VMs are provisioned for the peak request rate
SPOCK

- Use serverless functions along with VMS
  - Reduce SLO violations during request surge
  - Reduce intermittent over-provisioning VMs
Key Motivation

- It is non-trivial to predict the peak request rate at any given time period.
- Provisioning VMs for the peak demands would always lead to higher cost of deployment. While, under provisioning VMs leads to severe SLO violations for queries.
- Using serverless functions would overcome the SLO violation problem. However, it is not cost effective.
Outline

• Elastic Web Services

• VM-based Resource Procurement

• Serverless Functions

• Spock Hybrid Elastic Scaling

• Implementation and Evaluation

• Results
Spock Scheme

• Schedule queries on VM’s if available
• If VM’s are fully utilized, redirect queries to lambda functions
• Spawn a new VM in the meantime
• After spin-up incoming requests are sent to new VMs
• Scale down VMs after three minutes of inactivity
Two Scaling Policies

• **Reactive**
  • Spin-up new VMs as when request surge occurs

Let's see an example

• Using moving window linear regression predict request every minute
• Spin up new VMs based on prediction
Spock resource procurement

- **Scale out**
- **Scale in**

<table>
<thead>
<tr>
<th>Time (hundreds of sec)</th>
<th>Request rate per sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
</tr>
<tr>
<td>4</td>
<td>250</td>
</tr>
<tr>
<td>10</td>
<td>375</td>
</tr>
<tr>
<td>15</td>
<td>500</td>
</tr>
</tbody>
</table>

- Lambda
- VM
Overall Design of Spock

User Applications

Scalability

Resource Manager

Load Monitor

Load Balancer

Resource Status

Predicted Load

Query Complete

Resource Required

Queries

Query Assigned

Instance Created

VM

MODEL 1

MODEL 2

MODEL 3

MODEL 4
Outline

- Elastic Web Services
- VM-based Resource Procurement
- Serverless Functions
- Spock Hybrid Elastic Scaling
- Implementation and Evaluation
- Results
Evaluation

- Two traces used to generate ML inference workload
Evaluation

- Mxnet Framework
- AWS resources
- Pretrained ML models on imagenet dataset

<table>
<thead>
<tr>
<th>Query Type</th>
<th>Memory Required (GB)</th>
<th>Memory Allocated (GB)</th>
<th>Average Execution (ms)</th>
<th>Requests vCPU for VMs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caffenet</td>
<td>1.024</td>
<td>3.072</td>
<td>300</td>
<td>4</td>
</tr>
<tr>
<td>Googlenet</td>
<td>0.456</td>
<td>2.048</td>
<td>450</td>
<td>3</td>
</tr>
<tr>
<td>Squeezenet</td>
<td>0.154</td>
<td>2.048</td>
<td>130</td>
<td>6</td>
</tr>
<tr>
<td>Resnet-18</td>
<td>0.304</td>
<td>3.072</td>
<td>320</td>
<td>3</td>
</tr>
<tr>
<td>Resnet-200</td>
<td>1.024</td>
<td>3.072</td>
<td>956</td>
<td>1</td>
</tr>
<tr>
<td>Resnext-50</td>
<td>0.645</td>
<td>3.072</td>
<td>560</td>
<td>2</td>
</tr>
</tbody>
</table>
Evaluation

• Two scaling policies
  • Predictive
  • Reactive
• Three resource procurement schemes
  • Autoscale
  • X-autoscale
  • Spock
Outline

• Elastic Web Services
• VM-based Resource Procurement
• Serverless Functions
• Spock Hybrid Elastic Scaling
• Implementation and Evaluation
• Results
Berkely Trace Results

![Chart showing normalized cost and SLO violations for different strategies (autoscale, X-autoscale, and Spock) for two Mixes (Mix-1 and Mix-2). The chart displays the comparison between cost and SLO violations across these strategies.](image-url)
WITS Trace Results

![Graph showing normalized cost and SLO violations for different scenarios. The x-axis represents different platforms: autoscale, X-autoscale, Spock. The y-axis for normalized cost ranges from 0 to 2, and for SLO violations, it ranges from 0 to 8. The graph compares Mix-1, Mix-2, and SLO Violation.]
Spock Prediction Accuracy

![Graph 1: Request rate per sec vs. Time (hundreds of sec)]

![Graph 2: Request rate per sec vs. Time (hundreds of sec)]
Spock Resource Procurement

Scale out  Scale in  --- Request rate

Request rate per sec

VM

Time (hundreds of sec)

Scale out  Scale in  --- Request rate

Request rate per sec

VM

VM

VM

VM
Questions?