Function-as-a-Service Application
Service Composition: Implications for a Natural Language Processing Application

Mohammadbagher Fotouhi, Derek Chen
Wes Lloyd

December 9, 2019

School of Engineering and Technology,
University of Washington, Tacoma, Washington USA

WOSC 2019: 5th IEEE Workshop on Serverless Computing
Outline

- Background
- Research Questions
- Experimental Implementation
- Experiments/Evaluation
- Conclusions
Natural Language Processing

How can computers be used to understand speech?

Image from: https://aliz.ai/natural-language-processing-a-short-introduction-to-get-you-started//
NLP Dialogue modeling components

- Intent Tracking
  - Determines what the user wants
- Policy Management
  - Choose the agent action
- Text Generation
  - Generate the actual text
NLP Dialogue modeling components

• Considering a scenario where a user asks: “What is Milad’s phone number?”
  • Intent tracker -> Question
  • Policy Management -> To answer
  • Text generator -> “The number is 123-456-7890”

• These phases include an initialization and inference step
Serverless Computing

Deploy Applications Without Fiddling With Servers

Image from: https://mobisoftinfotech.com/resources/blog/serverless-computing-deploy-applications-without-fiddling-with-servers/
Serverless Computing

- Function-as-a-Service (FaaS) platforms
  - New cloud computing delivery model that provides a compelling approach for hosting applications
  - Bring us closer to the idea of instantaneous scalability
- Our goals - research implications of:
  - Memory reservation
  - Service composition
  - Adjustment of neural network weights
  - In the context of NLP application deployment
Memory Reservation

- Lambda memory reserved for functions
- UI provides “slider bar” to set function’s memory allocation
- Resource capacity (CPU, disk, network) coupled to slider bar:
  “every doubling of memory, doubles CPU…”
- How does memory allocation affect performance?
Infrastructure Freeze/Thaw Cycle

- Unused infrastructure is deprecated
  - But after how long?
- AWS Lambda: Bare-metal hosts, firecracker micro-VMs
- Three infrastructure states:
  - **Fully COLD (Cloud Provider/Host)**
    - Function package transferred to hosts
  - **Runtime environment COLD**
    - Function package cached on Host
    - No function instance or micro-VM
  - **WARM (firecracker micro-VM)**
    - Function instances/micro-VMs ready
Service Composition

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

Switchboard / Asynchronous

- Fully aggregated (Switchboard) and fully disaggregated (Service isolation) composition

- Platform limits: code + libraries ~250MB

- How does service composition affect freeze/thaw cycle and impact performance?
Outline

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

RQ1: MEMORY: How does the FaaS function memory reservation size impact application performance?

RQ2: COMPOSITION: How does service composition of microservices impact the application performance?
Research Questions - 2

RQ3: **NN-WEIGHTS:** How does varying the neural network weights impact the performance of the NLP application?

RQ4: **FREEZ THAW LIFE CYCLE:** How does the service composition of our NLP application impact the freeze-thaw life cycle?
Outline

- Background
- Research Questions
- Implementation
- Experiments/Evaluation
- Conclusions
## Aws lambda Inference functions

<table>
<thead>
<tr>
<th>Function ID</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Initialize Intent Tracker</td>
<td>Text preprocessing and create sentence embedding</td>
</tr>
<tr>
<td>F2</td>
<td>Run Intent Tracker</td>
<td>Load the weights and predict user intent</td>
</tr>
<tr>
<td>F3</td>
<td>Initialize Policy Manager</td>
<td>Create action embedding</td>
</tr>
<tr>
<td>F4</td>
<td>Run Policy Manager</td>
<td>Load the weights and predict agent action</td>
</tr>
<tr>
<td>F5</td>
<td>Initialize Text Generator</td>
<td>Create the generated output embeddings</td>
</tr>
<tr>
<td>F6</td>
<td>Run Text Generator</td>
<td>Load the weights and create final text output</td>
</tr>
</tbody>
</table>
Switchboard architecture

- Aggregated all 6 microservices in one package
- Client initiates pipeline
- Switchboard routine accepts calls and routes internally
Full service isolation architecture

- Fully decomposed functions as independent microservices
- Cloud provider provisions separate runtime containers
Application Implementation

- Disseminate neural network models with AWS S3
- AWS CLI based client for submitting requests
- Leveraged AWS EC2’s Python Cloud9 IDE to identify and compose dependencies
- Packaged dependencies as ZIP for inclusion in Lambda FaaS function deployment
- Conformed to package size limitations (<250MB)
Outline

• Background
• Research Questions
• Experimental Workloads
• Experiments/Evaluation
• Conclusions
NN-Weights

How does varying the neural network weights impact the performance of the NLP application?
Runtime performance
Switchboard

Switchboard Run-time

- run_text_generator
- init_text_generator
- run_policy_manager
- init_policy_manager
- run_intent_tracker
- init_intent_tracker

Seconds

Number of Samples

3 10 30 100 300 1000

3 10 30 100 300 1000

0 25 50 75 100

average of 8 runs

WOSC 2019: FaaS Application Service Composition: Implications for an NLP Application
Runtime performance
Service Isolation

Service Isolation Run-time

Seconds

Number of Samples

3 10 30 100 300 1000

run_text_genet init_text_genet run_policy_manag
int_policy_manag run_intent_track init_intent_tracker

Average of 8 runs
Memory

How does the FaaS function memory reservation size impact application performance?
Memory Utilization Switchboard

Switchboard Memory Limits

- Run It
- Run PM
- Run Tg

Max Memory used (MB)

<table>
<thead>
<tr>
<th>Sample Size</th>
<th>Run It</th>
<th>Run PM</th>
<th>Run Tg</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Memory Utilization
Service isolation

Service Isolation Memory Limits
- Run IT
- Run PM
- Run TG

Max Memory used (MB)

Samples

December 9, 2019
WOSC 2019: FaaS Application Service Composition: Implications for an NLP Application
Composition

How does service composition of microservices impact the application performance?
Performance Comparison

Service Isolation VS Switchboard end to end Run-time

- Service Isolation
- Switchboard

Total Run-time (Seconds)

Samples

Memory sizes tested: 192, 256, 384, 512 MB
Outline

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

• Switchboard architecture minimized cold starts
• Switchboard performed more efficiently over larger input dataset sizes vs. service isolation
  • 14.75% faster for 1,000 samples
  • 17.3% increase in throughput
• When inferencing just 3 samples, the service isolation architecture was faster
  • 36.96% faster for 3 samples
  • 58% increase in throughput

• full service isolation not always optimal
Questions