Predicting the Costs of Serverless Workflows

Simon Eismann  
University of Würzburg  
@simon_eismann

Johannes Grohmann  
University of Würzburg

Erwin van Eyk  
Vrije Universiteit  
@erwinvaneyk

Nikolas Herbst  
University of Würzburg  
@HerbstNikolas

Samuel Kounev  
University of Würzburg  
@skounev

https://se.informatik.uni-wuerzburg.de
What are serverless functions?

1. Upload code
2. Setup triggers to run code in response to events
3. Code is executed on-demand with continuous scaling
4. Pay for used time with sub-second metering

Predicting the Costs of Serverless Workflows

@simon_eismann
Pay-per-use makes estimating costs challenging

- Cost of serverless functions depends on [1, 2]:
  - Response time rounded to nearest 100ms
  - Function size (allocated memory/CPU)
  - Static overhead per execution

- Moreover, function response time depends on input [3]
  - Function execution in a different context changes cost
  - Makes estimation of costs for workflows challenging

- Existing approaches for cost estimation [4, 5, 6]:
  - Describe the response time as a static mean
  - Require user to estimate response time
Predicting the Costs of Serverless Workflows

@simon_eismann

Summary

Problem

• Estimating the expected costs of serverless workflows is challenging
• Input influences function response time

Idea

• Build predictive model for workflow costs from production monitoring

Benefit

• Guides decision between serverless and traditional hosting
• Enables comparison of workflow alternatives
• First step towards fully automated workflow optimization
Overview

Continuous Model Learning

Serverless Functions → Monitoring Data → Model Trainer → Function Model Repository

Monitoring Data Repository

Workflow Prediction Engine

Monte-Carlo Simulator

Workflow Model

Workflow Cost Prediction

1. upload workflow structure
2. fetch function models
3. build workflow model
4. simulate workflow model
5. return cost estimate

Predicting the Costs of Serverless Workflows
@simon_eismann
Accurate cost prediction requires predicting the response time distribution of a function, not just its mean response time.
Predicting the Function Response Time Distribution

- Gaussian mixture models model distribution as linear combination of gaussian kernels [7]
- Gaussian mixture models can approximate any distribution assuming sufficient kernels
- Mixture density networks use DNN to parameterize mixture distribution [8]
Predicting the Costs of Serverless Workflows

@simon_eismann

**Approach**

1. Model Workflow Structure
2. Integrate MDNs
3. Identify next node
4. Monte-Carlo simulation
5. Repeat steps 3+4
6. Calculate costs
Evaluation

RQ1: Can we accurately predict the distribution of the response time and the output parameters of a serverless function?

RQ2: Can we accurately predict the costs of a previously unobserved workflow?

RQ3: What is the required time for model training and workflow cost prediction?
Case Study

Five functions:
- Text to speech
- Audio format conversion
- Profanity detection
- Censor audio segments
- Compress audio file

Two Workflow alternatives:
Can we accurately predict the distribution of the response time and the output parameters of a serverless function?
RQ 1 – Numerical Results

Can we accurately predict the distribution of the response time and the output parameters of a serverless function?

Normalized, relative Wasserstein metric [9, 10]

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameter</th>
<th>1 kernel</th>
<th>2 kernels</th>
<th>3 kernels</th>
<th>4 kernels</th>
<th>5 kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text2Speech</td>
<td>Response time</td>
<td>5.3%</td>
<td>4.2%</td>
<td>4.1%</td>
<td>6.4%</td>
<td>4.5%</td>
</tr>
<tr>
<td>Text2Speech</td>
<td>FileSize</td>
<td>0.6%</td>
<td>0.3%</td>
<td>1.1%</td>
<td>0.4%</td>
<td>0.6%</td>
</tr>
<tr>
<td>Conversion</td>
<td>Response time</td>
<td>13.2%</td>
<td>38.3%</td>
<td>3.4%</td>
<td>3.3%</td>
<td>3.3%</td>
</tr>
<tr>
<td>Conversion</td>
<td>FileSize</td>
<td>0.9%</td>
<td>1.2%</td>
<td>7.8%</td>
<td>9.0%</td>
<td>16.4%</td>
</tr>
<tr>
<td>Compression</td>
<td>Response time</td>
<td>13.1%</td>
<td>4.3%</td>
<td>5.2%</td>
<td>4.4%</td>
<td>3.6%</td>
</tr>
<tr>
<td>Compression</td>
<td>FileSize</td>
<td>0.2%</td>
<td>1.7%</td>
<td>0.4%</td>
<td>0.2%</td>
<td>3.5%</td>
</tr>
<tr>
<td>ProfanityDet</td>
<td>Response time</td>
<td>38.7%</td>
<td>32.9%</td>
<td>12.8%</td>
<td>9.4%</td>
<td>4.6%</td>
</tr>
<tr>
<td>ProfanityDet</td>
<td>ProfanityCount</td>
<td>14.5%</td>
<td>69.0%</td>
<td>12.8%</td>
<td>12.3%</td>
<td>14.0%</td>
</tr>
<tr>
<td>Censor</td>
<td>Response time</td>
<td>9.5%</td>
<td>10.1%</td>
<td>8.5%</td>
<td>8.2%</td>
<td>9.1%</td>
</tr>
<tr>
<td>Censor</td>
<td>FileSize</td>
<td>1.0%</td>
<td>0.6%</td>
<td>0.7%</td>
<td>1.5%</td>
<td>7.9%</td>
</tr>
</tbody>
</table>

We can accurately predict the response time and output parameter distributions of serverless functions.
## RQ 2 - Results

Can we accurately predict the costs of a previously unobserved workflow?

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Metric</th>
<th>Invocations</th>
<th>CPU Time</th>
<th>Memory Time</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow1</td>
<td>Measured cost [ct]</td>
<td>$2.00 \times 10^{-6}$</td>
<td>$8.6 \times 10^{-5}$</td>
<td>$1.40 \times 10^{-5}$</td>
<td>$1.02 \times 10^{-4}$</td>
</tr>
<tr>
<td>Workflow1</td>
<td>Predicted cost [ct]</td>
<td>$1.79 \times 10^{-6}$</td>
<td>$9.42 \times 10^{-5}$</td>
<td>$1.40 \times 10^{-5}$</td>
<td>$1.10 \times 10^{-4}$</td>
</tr>
<tr>
<td>Workflow1</td>
<td>Relative prediction error</td>
<td>10.5%</td>
<td>9.5%</td>
<td>0.0%</td>
<td>7.8%</td>
</tr>
<tr>
<td>Workflow2</td>
<td>Measured cost [ct]</td>
<td>$2.00 \times 10^{-6}$</td>
<td>$3.80 \times 10^{-5}$</td>
<td>$6.00 \times 10^{-6}$</td>
<td>$4.60 \times 10^{-5}$</td>
</tr>
<tr>
<td>Workflow2</td>
<td>Predicted cost [ct]</td>
<td>$1.79 \times 10^{-6}$</td>
<td>$3.76 \times 10^{-5}$</td>
<td>$5.60 \times 10^{-6}$</td>
<td>$4.50 \times 10^{-5}$</td>
</tr>
<tr>
<td>Workflow2</td>
<td>Relative prediction error</td>
<td>10.5%</td>
<td>1.0%</td>
<td>6.7%</td>
<td>2.2%</td>
</tr>
</tbody>
</table>

The proposed approach can accurately predict the execution cost of previously unobserved workflow.
RQ 3 - Results

What is the required time for training and workflow prediction? Is the overhead feasible for a production environment?

Training time for all models with hyper-parameter optimization

<table>
<thead>
<tr>
<th>Serverless function</th>
<th>Training time [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text2Speech</td>
<td>10</td>
</tr>
<tr>
<td>Conversion</td>
<td>12</td>
</tr>
<tr>
<td>Compression</td>
<td>15</td>
</tr>
<tr>
<td>ProfanityDet</td>
<td>18</td>
</tr>
<tr>
<td>Censor</td>
<td>20</td>
</tr>
</tbody>
</table>

Prediction time

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Prediction time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow A</td>
<td>16.34s ± 0.30s</td>
</tr>
<tr>
<td>Workflow B</td>
<td>14.20s ± 0.03s</td>
</tr>
</tbody>
</table>

We consider the time requirements of using our approach in production feasible.
Performance measurements

Wrapped in docker container for platform independent execution

Requires only google cloud access keys as input

Fully automated performance measurements

Available online at:
https://doi.org/10.5281/zenodo.3582707

Data set and analysis

Measurement data of serverless functions in public cloud

Scripts to reproduce any analysis, table or figure from the manuscript

1-click reproduction of the results as a CodeOcean Capsule

Available online at:
https://doi.org/10.5281/zenodo.3582707
Cost of serverless functions depends on:
- Response time rounded to nearest 100ms
- Function size (allocated memory/CPU)
- Static overhead per execution

Moreover, function response time depends on input:
- Function execution in a different context can occur different costs
- Makes estimation of costs for workflows challenging

Existing approaches for cost estimation [2,3]:
- Describe the response time as a static mean
- Require user to estimate response time

Gaussian mixture models model distribution as linear combination of gaussian kernels [4]

Gaussian mixture models can approximate any distribution assuming sufficient kernels

Mixture density networks use DNN to parameterize mixture distribution [5]
References


