Proactive Serverless Function Resource Management

Sixth International Workshop on Serverless Computing (WoSC6) 2020

Erika Hunhoff, Shazal Irshad, Vijay Thurimella*, Ali Tariq, Eric Rozner

University of Colorado Boulder, *Thrive, Inc
Outline

- Background
- Freshen Design
- Evaluation
- Discussion
- Questions
IDCat Serverless Application
UserID: Erika

OpenWhisk

function1

IDCat Serverless Application
UserID: Erika

Function Trigger

OpenWhisk

function1

“Give me the ML model!”

“Here it is”

Cat Model Database

IDCat Serverless Application
IDCat Serverless Application

UserID: Erika

Function Trigger

OpenWhisk

function1 does work!

"Give me the ML model!"

"Here it is"

Cat Model Database
UserID: Erika

OpenWhisk

“Give me the ML model!”

“Here it is”

Store “Erika’s Cat, Brown Tabby”

“Saved!”

IDCat Serverless Application
UserID: Erika

Result: “Success!”

"Give me the ML model!"

"Here it is"

"Saved!"

IDCat Serverless Application
UserID: Erika

Function Trigger

Result: “Success!”

OpenWhisk

“Give me the ML model!”

“Here it is”

Store “Erika’s Cat, Brown Tabby”

“Saved!”

Cat Model Database

Cat Database

Improve Efficiency?
UserID: Erika

OpenWhisk

“Give me the ML model!”

“Here it is”

“Saved!”

Store “Erika’s Cat, Brown Tabby”

Result: “Success!”

Function Trigger

Cat Model Database

Cat Database

Improve Efficiency?
UserID: Erika

Microservice

Request Handler

Request

Result: “Success!”

“Give me the ML model!”

Store “Erika’s Cat, Brown Tabby”

“Here it is”

“Saved!”

Cat Model Database

Cat Database

Long-lived connection or connection pool

IDCat Microservice
Runtime Reuse in Serverless
Runtime Reuse in Serverless
Runtime Reuse in Serverless
Runtime Reuse in Serverless
Runtime Reuse in Serverless – Room for Improvement?
Outline

Background

*Freshen Design*

Evaluation

Discussion

Questions
We propose a new serverless runtime primitive, *freshen*, as a mechanism to enable proactive serverless function resource management.
Time=0

/init

init

/run

function1

*Freshen* Design
Freshen Design

Time=0

/init

/freshen

/run

init

freshen

function1
Time=0

/init

/freshen

/run

/init

/freshen

/function1

Prediction window

**Freshen Design**
Freshen Design

Policy Options:
- Prediction
- Concurrency
- Forced blocking
<table>
<thead>
<tr>
<th></th>
<th>Reuse</th>
<th>Dynamic State</th>
<th>Proactive</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Init Phase</strong></td>
<td>☑️</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Function Code</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Runtime Reuse</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Freshen</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Time=0

/init

init

/run

function1

/run

function1

<table>
<thead>
<tr>
<th></th>
<th>Reuse</th>
<th>Dynamic State</th>
<th>Proactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init Phase</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Function Code</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Runtime Reuse</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Freshen</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reuse | Dynamic State | Proactive
--- | --- | ---
Init Phase | ✓ | ✗ | ✗
Function Code | ✗ | ✓ | ✗
Runtime Reuse | ✓ | ✓ | ✗

**Freshen**
Reuse | Dynamic State | Proactive
--- | --- | ---
Init Phase | ✓ | ✗ | ✗
Function Code | ✗ | ✓ | ✗
Runtime Reuse | ✓ | ✓ | ✗
Freshen | ✓ | ✓ | ✓
Serverless Function Prediction

Prediction useful for many reasons (scheduling, resource utilization, coldstart avoidance, etc.)
Some cases are easier to predict, e.g., chained functions

Prediction useful for many reasons (scheduling, resource utilization, coldstart avoidance, etc.)
Some cases are easier to predict, e.g., chained functions

Many applications consist of multiple functions

Prediction useful for many reasons (scheduling, resource utilization, coldstart avoidance, etc.)
Serverless Function Prediction

Prediction useful for many reasons (scheduling, resource utilization, coldstart avoidance, etc.)

Some cases are easier to predict, e.g., chained functions

Many applications consist of multiple functions

May be infrastructure overheads

<table>
<thead>
<tr>
<th>Trigger Service</th>
<th>Delay (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step Functions</td>
<td>0.064</td>
</tr>
<tr>
<td>Direct (Boto3)</td>
<td>0.060</td>
</tr>
<tr>
<td>SNS Pub/Sub</td>
<td>0.253</td>
</tr>
<tr>
<td>S3 bucket</td>
<td>1.282</td>
</tr>
</tbody>
</table>
What Can Freshen Do?
What Can **Freshen** Do?

UserID: Erika

Function Trigger

Result: “Success!”

OpenWhisk

“Give me the ML model!”

“Here it is”

Store “Erika’s Cat, Brown Tabby”

“Saved!”

Cat Model Database

Cat Database
What Can **Freshen** Do?

User ID: Erika

**Function Trigger**

**Result:** “Success!”

OpenWhisk

- “Give me the ML model!”
- “Here it is”
- Store “Erika’s Cat, Brown Tabby”
- “Saved!”

**PREFETCH**

- **Cat Model Database**
- **Cat Database**
What Can **Freshen** Do?

UserID: Erika

Function Trigger

Result: “Success!”

OpenWhisk

“Give me the ML model!”

“Here it is”

Store “Erika’s Cat, Brown Tabby”

“Saved!”

WARM CONNECTION

Cat Model Database

Cat Database
UserID: Erika

Function Trigger

Function1

OpenWhisk

"Give me the ML model!"

"Here it is"

Store "Erika’s Cat, Brown Tabby"

"Saved!"

Result: “Success!”

Cat Model Database

Cat Database

What Can Freshen Do?

MORE?
Freshen Motivation

Reduces latency to access data
**Freshen** Motivation

TCP connections warmed send traffic more efficiently
Discussion

Connection state manipulation

- How to access?
- Beyond TCP

Function prediction

Who is responsible for \textit{freshen}?

Other \textit{freshen} actions

• Memory allocation?
• Caches?
• Things we have not yet thought of?
Discussion

- Connection state manipulation
- Function prediction
- Who is responsible for *freshen*?
- Other *freshen* actions

Who is responsible for freshen?
Memory allocation?
Caches?
Things we have not yet thought of?

Beyond TCP
Connection state manipulation

Developer
Libraries
Inference
Discussion

- Connection state manipulation
- Function prediction
- Who is responsible for `freshen`?
  - Developer
  - Libraries
  - Inference
- Other `freshen` actions
- How to access?
- Beyond TCP
- Scalability
- Generality
- Developer
- Libraries
- Inference
- Memory allocation?
- Caches?
- Things we have not yet thought of?
Discussion

- How to access?
- Beyond TCP
- Connection state manipulation
- Function prediction
- Scalability
- Generality
- Function prediction
- Developer
- Libraries
- Inference
- Who is responsible for *freshen*?
- Other *freshen* actions
- Memory allocation?
- Caches?
- Things we have not yet thought of?

44
We propose a new serverless runtime primitive, *freshen*, as a mechanism to enable proactive serverless function resource management.

Erika Hunhoff
erika.hunhoff@colorado.edu