Visual-textual framework for serverless computation: a Luna Language approach

Piotr Moczurad, Maciej Malawski
Workshop on Serverless Computing 2018
Zürich, 20.12.2018
The evolution of an architecture
The next step: serverless

A function is a first-class unit that is billed and deployed separately.

It seems to be the finest level of granularity we can achieve in cloud computing.
The trend

- Subdivide computations
- Separate responsibilities
- Reduce cost
- Increase performance
- Increase scalability
- Increase complexity…

Value Axis

Complexity

Monolith  Services  Microservices  Functions

Image source: https://spectreattack.com/

Copyright: AGH University of Science and Technology, The Luna Team, 2018
Look again...

How do we go about visualising the serverless architecture?
Visualisation

Ideally: a **graph**.

**Node**: a **function**.

**Edge**: the **flow of data** between functions.
Visual-textual programming
Data flow graph

uri "http://www.ttss.krakow.pl/intern"

Text

output
Adjustable levels of abstraction
Every node corresponds to one line of code
def main:
    x = 14
    y = 24

Main.luna > main

x
14

y
24

24
Benefits

• Clear pipeline in form of a readable graph

• Manageable complexity: correctness & productivity

• Communication backed by the compiler
Luna + Serverless
Ideal world

Functions executing remotely (e.g. on AWS Lambda) *indistinguishable* from local asynchronous functions.
### Rationale

<table>
<thead>
<tr>
<th><strong>Local</strong> async function:</th>
<th><strong>Remote</strong> function:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Doesn’t return immediately</td>
<td>• Doesn’t return immediately</td>
</tr>
<tr>
<td>• Can fail (interrupted?)</td>
<td>• Can fail (network failure?)</td>
</tr>
</tbody>
</table>

This can be captured by the same result type:

Future[T]
The configuration is abstracted away. Sensible, overridable defaults are what makes the work efficient.

```python
aws = AWS.init
aws1 = aws.setRole "myRole..."
```
Defining a function

PoC stage: take the function code in JS and create a Lambda function.

```
jsCode = "exports.handler = async (event) => { return 'Hello' + event.name + '!'; };"

code = LambdaFunctionCode.fromText "helloFunc" jsCode
createFunction1 = aws.createFunction code
```
Invoking a function

Call a remote function similarly to a regular method.

Two flavours:

\[
\text{sync} :: \text{Payload} \rightarrow \text{Result}
\]

\[
\text{async} :: \text{Payload} \rightarrow \text{Future Result}
\]

\[
\text{payload} = \text{JSON}.\text{empty} . \text{Insert} \text{ "name" "Peter"}
\]

\[
\text{helloFunc}.\text{invoke} \text{ payload}
\]
Function return value

We can chain operations on values that are not-quite-there-yet. (Thank you, monads!)

```
futureRes = invokeFun . flatMap extractRP . await . get
```
Utilities

Caching and lookup of remote functions

Creating the function many times is the thing to avoid.
Under the hood

• Part of the Luna Language Standard Library

• Crucial parts written in Haskell, API wrapper in Luna

• Leverages the Amazonka library [1] and its Amazonka Lambda extension [2]

Performance

- Performance was not a design goal! (Programmer productivity was)
- Incidentally, the performance is comparable to Haskell and Node.js:

<table>
<thead>
<tr>
<th></th>
<th>Luna</th>
<th>Haskell</th>
<th>Node.js</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sync</strong> Mean [s]</td>
<td>35.12</td>
<td>37.17</td>
<td>37.94</td>
</tr>
<tr>
<td>StdDev [s]</td>
<td>0.38</td>
<td>0.79</td>
<td>0.81</td>
</tr>
<tr>
<td><strong>Async</strong> Mean [s]</td>
<td>34.55</td>
<td>33.32</td>
<td>34.35</td>
</tr>
<tr>
<td>StdDev [s]</td>
<td>0.81</td>
<td>2.67</td>
<td>0.58</td>
</tr>
</tbody>
</table>
Future work

- Enable the deployment of functions written in Luna (!)
- Support other cloud providers
- Develop a more sound typing scheme for calls and responses
- Develop a formal model for proving the correctness of Serverless applications
Closing remarks

• Long way to go until serverless functions are supported as a first-class citizens in a programming language but we are getting there.

• Serverless and functional are a promising match!

• Visual solutions for serverless are necessary: a visual language provides that out-of-the-box.
Get in touch!

https://www.icsr.agh.edu.pl/

• GitHub: github.com/luna
• Website: luna-lang.org
• Chat: chat.luna-lang.org

• GitHub: https://github.com/piotrMocz/
• Twitter: https://twitter.com/PMoczurad
• Mail: piotr.moczurad@gmail.com