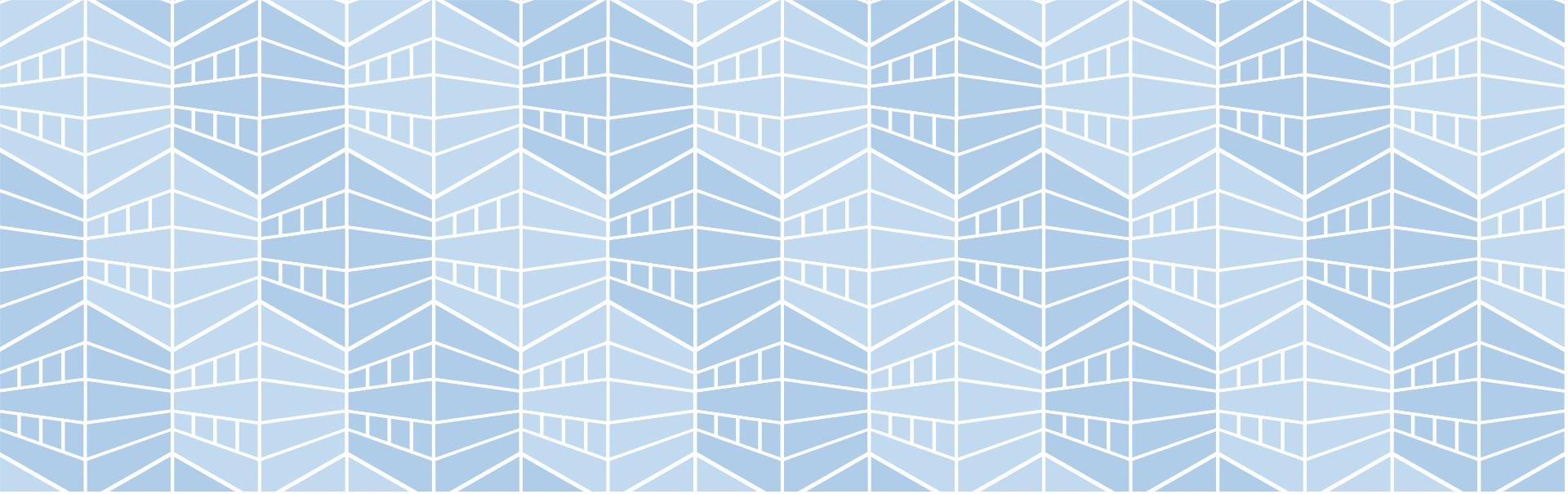


Functional Programming
in Serverless World

Lambda Days 2018 - Cracow (*23.02.2018*)

APPLISCALE

~ # whoami

(afronski)

- ✓ Software and Operations Engineer
  Appliscale

  Erlang, Elixir, Node.js
  AWS (*2 certificates*)

- ✓ ***Functional Miners*** co-organizer

P.S. ***Appliscale*** *is hiring!*

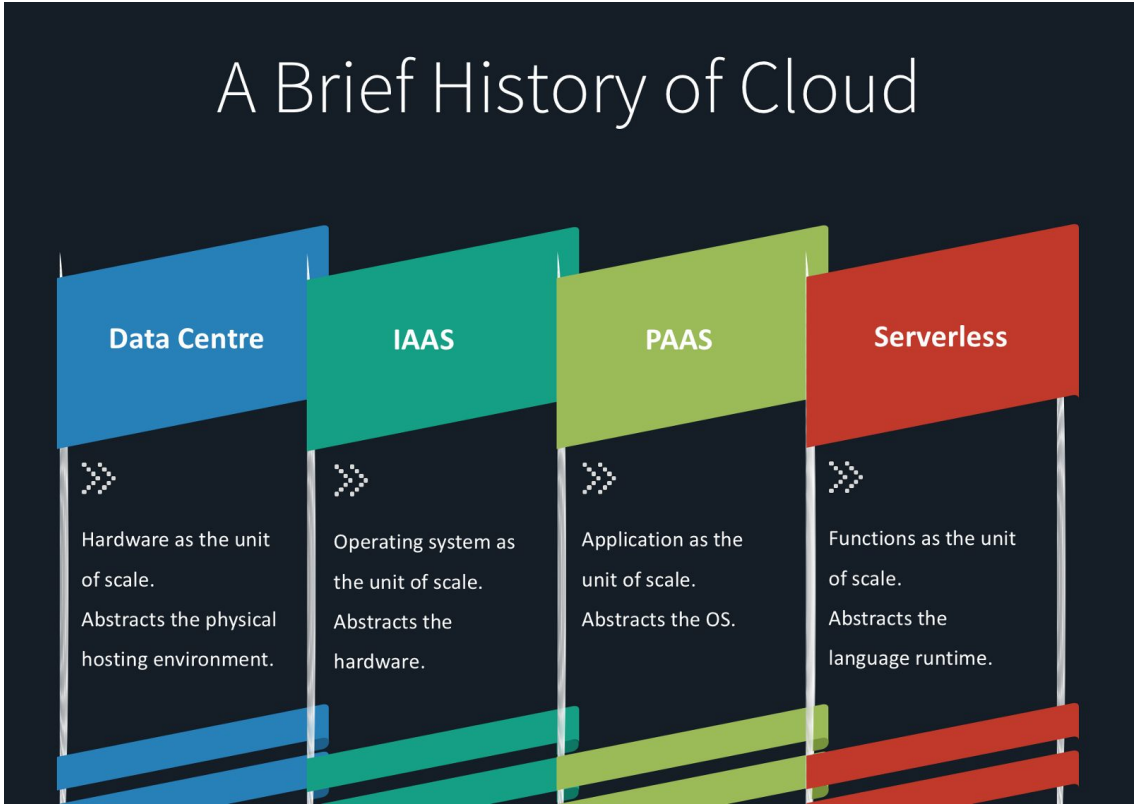
What is **serverless**?

http://celiacandthebeast.com/wp-content/uploads/2014/08/SRSLY.jpg

There are **servers**

There are

ops



read.acloud.guru/iaas-paas-serverless-the-next-big-deal-in-cloud-computing

Serverless

Back-end as a Service (**BaaS**)

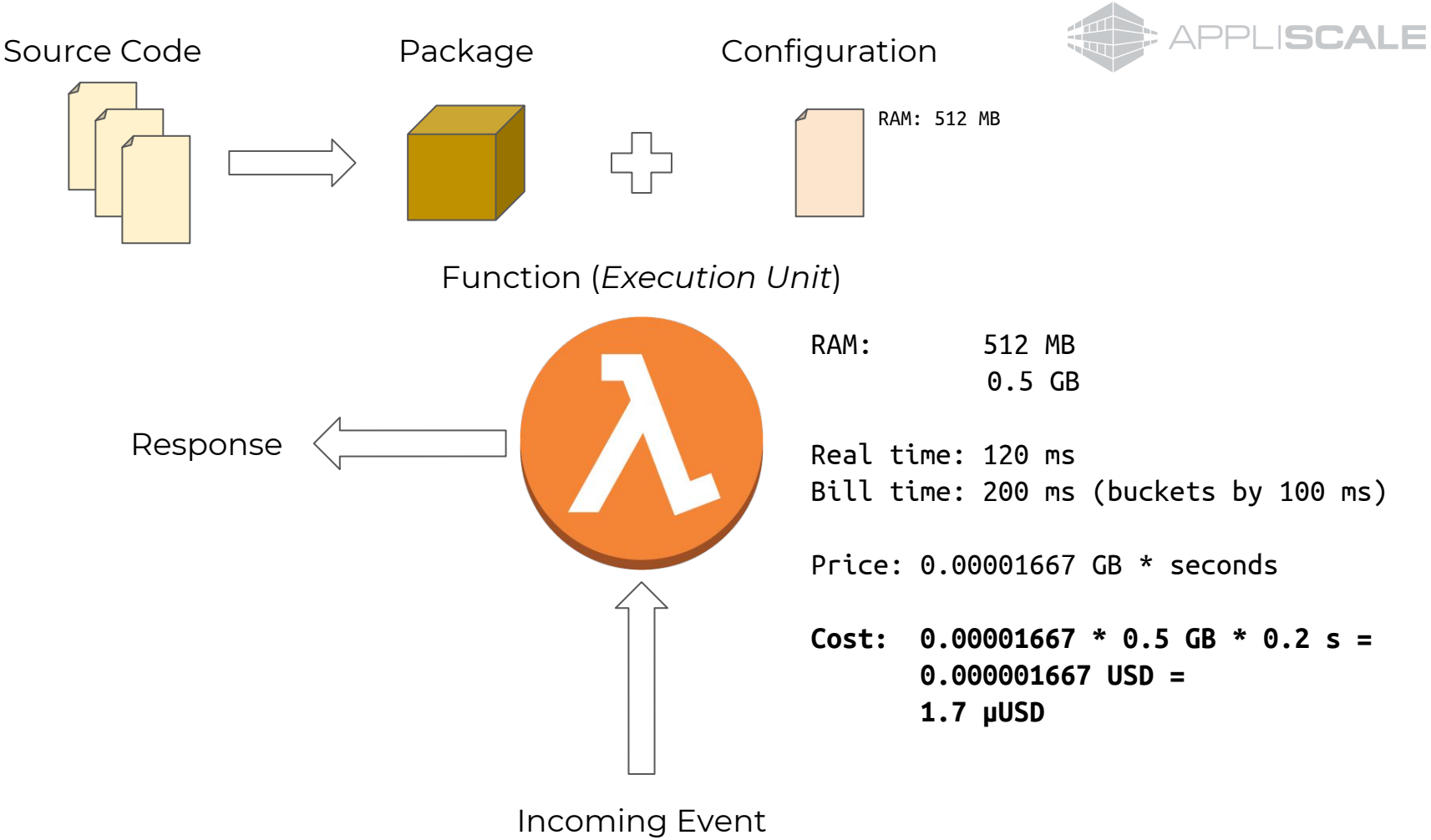
Parse

kinvey

Firebase

Function as a Service (**FaaS**)

Cloud Functions


APPLISCALE
Source Code
Package
Configuration
RAM: 512 MB
Function (Execution Unit)
Response
RAM: 512 MB
0.5 GB
Real time: 120 ms
Bill time: 200 ms (buckets by 100 ms)
Price: 0.00001667 GB * seconds
Cost: 0.00001667 * 0.5 GB * 0.2 s =
0.000001667 USD =
1.7 µUSD
Incoming Event

Why **serverless**?

APPLISCALE

Serverless Computing: Economic and Architectural Impact

Gojko Adzic
Neuri Consulting LLP
25 Southampton Buildings
London, United Kingdom WC2A 1AL
gojko@neuri.co.uk

Robert Chatley
Imperial College London
180 Queen's Gate
London, United Kingdom SW7 2AZ
rbc@imperial.ac.uk

http://www.doc.ic.ac.uk/~rbc/papers/fse-serverless-17.pdf

Gojko Adzic - Designing for the Serverless Age (GOTO 2017)

Costs

Service instance	Billable unit	Unit cost (USD)	Fail-over costs (%)	Cost of 12 x 200ms exec'ns	% reference price
Lambda (128 MB)	100 ms	$0.000000208	included	$0.000004992	24.94%
Lambda (512 MB)	100 ms	$0.000000834	included	$0.000020016	100.00%
Heroku Hobby (512 MB)	1 month	$7.00	100%	$0.0097222222	48572.25%
AWS EC2 t2.nano (512 MB)	1 hour	$0.0059	100%	$0.0118	58952.84%
AppEngine B1 (128MB)	1 hour	$0.05	100%	$0.1	499600.32%
AppEngine B4 (512MB)	1 hour	$0.20	100%	$0.4	1998401.28%

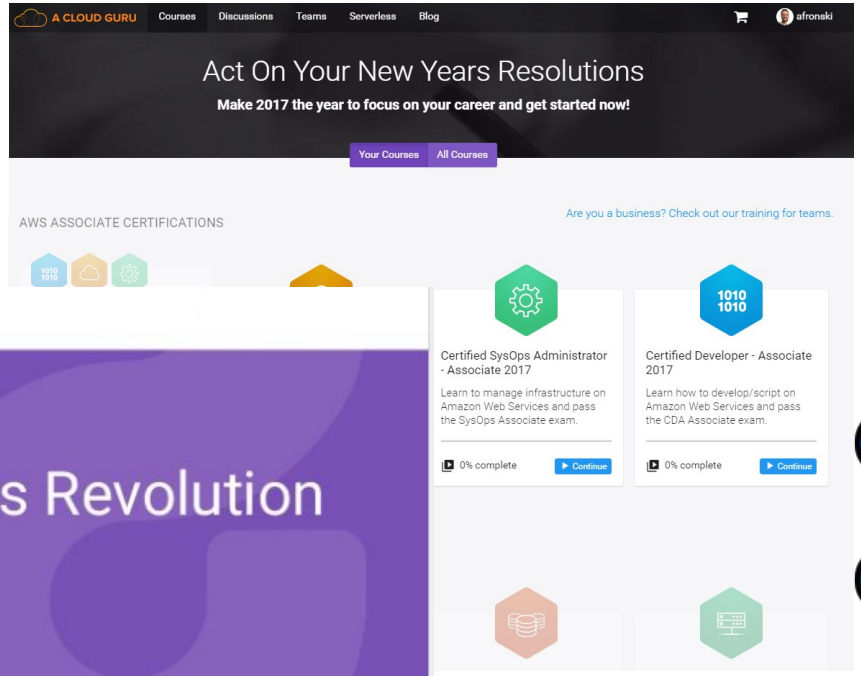
Architecture

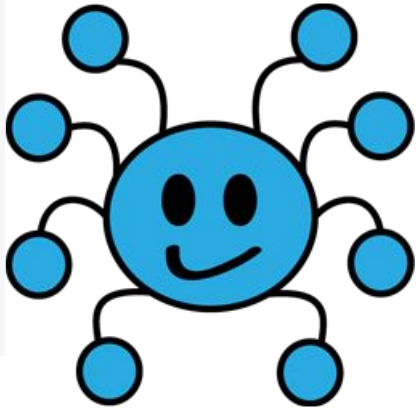
[...] breaking the monolith into functions that could be independently deployed, meant that they were **better able to** split the team up to work on more things in **parallel**, and to deploy each feature **separately**.

Operations

[...] so their estimate is that moving to Lambda gave an **operational cost reduction** of **greater than 95%** for a comparable amount of compute.

APPLISCALE





serverless

EA

Coca-Cola

NORDSTROM

Expedia

REUTERS

Technology

APPLISCALE

Scala

fλ

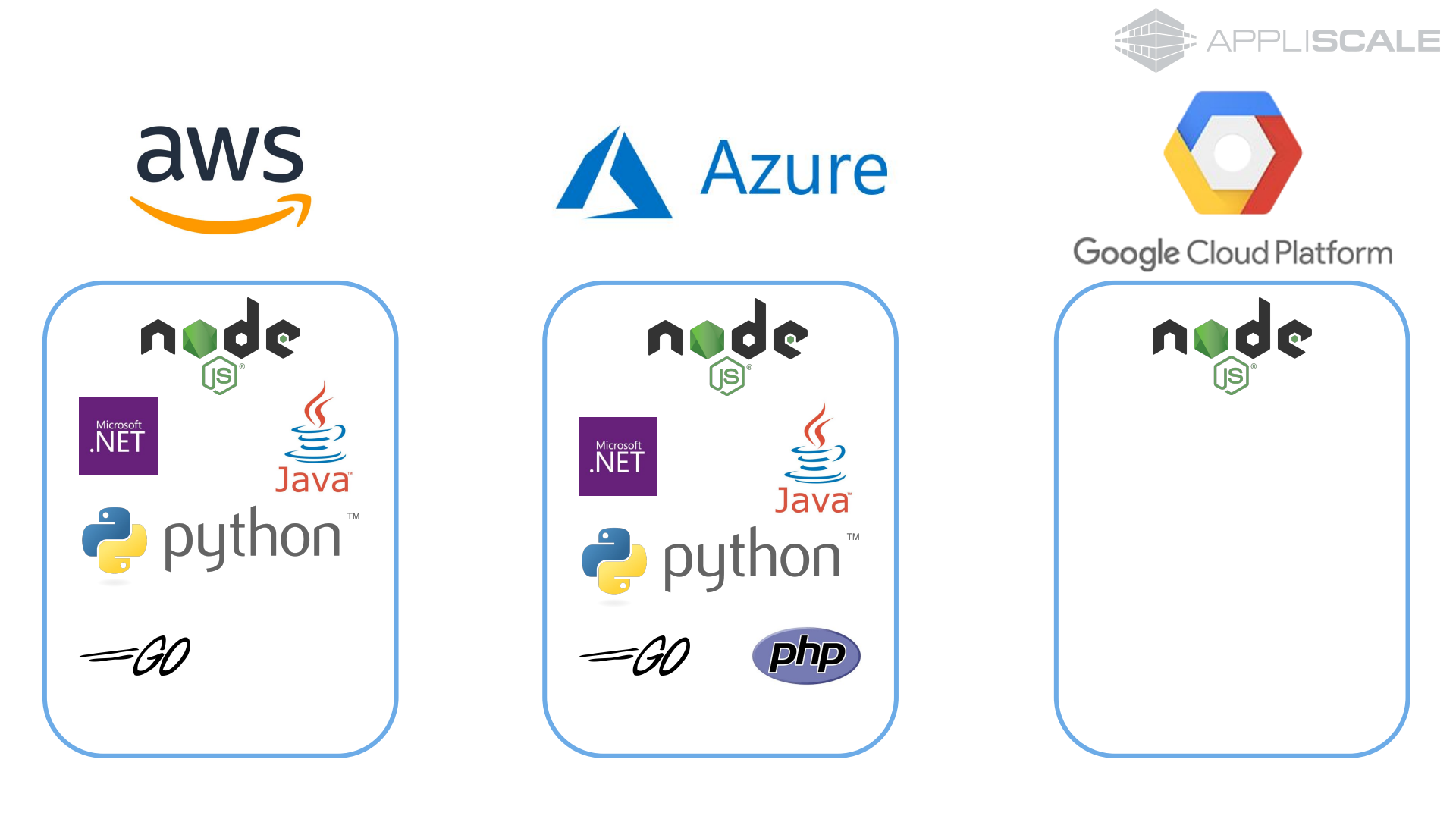
elixir

ERLANG

cljs

F#

PURESCRIPT


APPLISCALE
aws
node
Microsoft .NET
Java
python
GO
Azure
node
Microsoft .NET
Java
python
GO
php
Google Cloud Platform
node

Shims

```
const spawn = require("child_process").spawn;

exports.handler = function(event, context) {
  process.env["PATH"] = process.env["PATH"] + ":" + process.env["LAMBDA_TASK_ROOT"]
  process.env["LD_LIBRARY_PATH"] = process.env["LAMBDA_TASK_ROOT"]
  const main = spawn("./exec", { stdio: ["pipe", "pipe", process.stderr] });

  main.stdout.on("data", function(data) {
    context.done(null, data.toString());
  });

  main.on("close", function(code) {
    context.done(null, code);
  });

  main.on("exit", function(code){
    context.done(null, code);
  });

  main.on("error", function(err) {
    context.done(null, err);
  });

  main.stdin.write(JSON.stringify({ event, context }) + "\n");
}
```

Using Node.js *process* API to spawn an executable

Handling events reported by subprocess and passing results to the *AWS Lambda* handler

Sending *input arguments* to the spawned subprocess

APPLISCALE

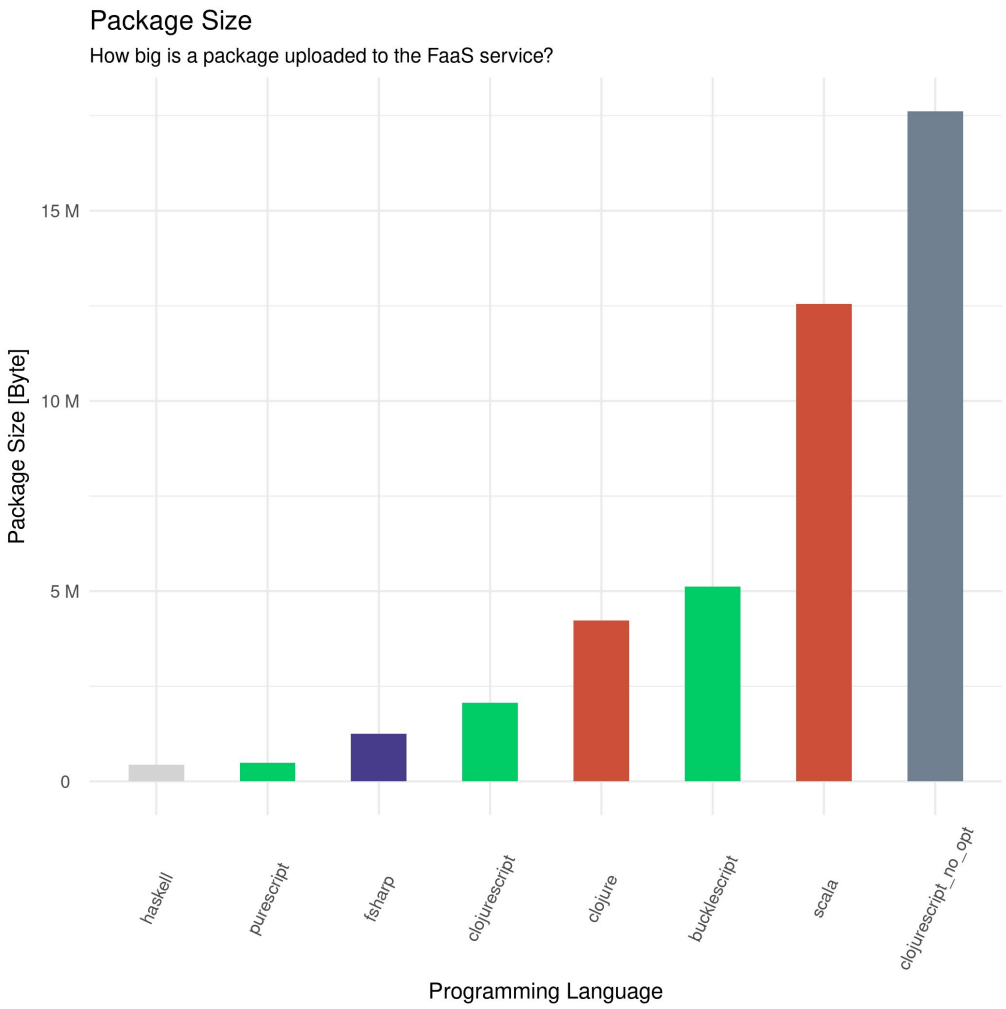
Scala

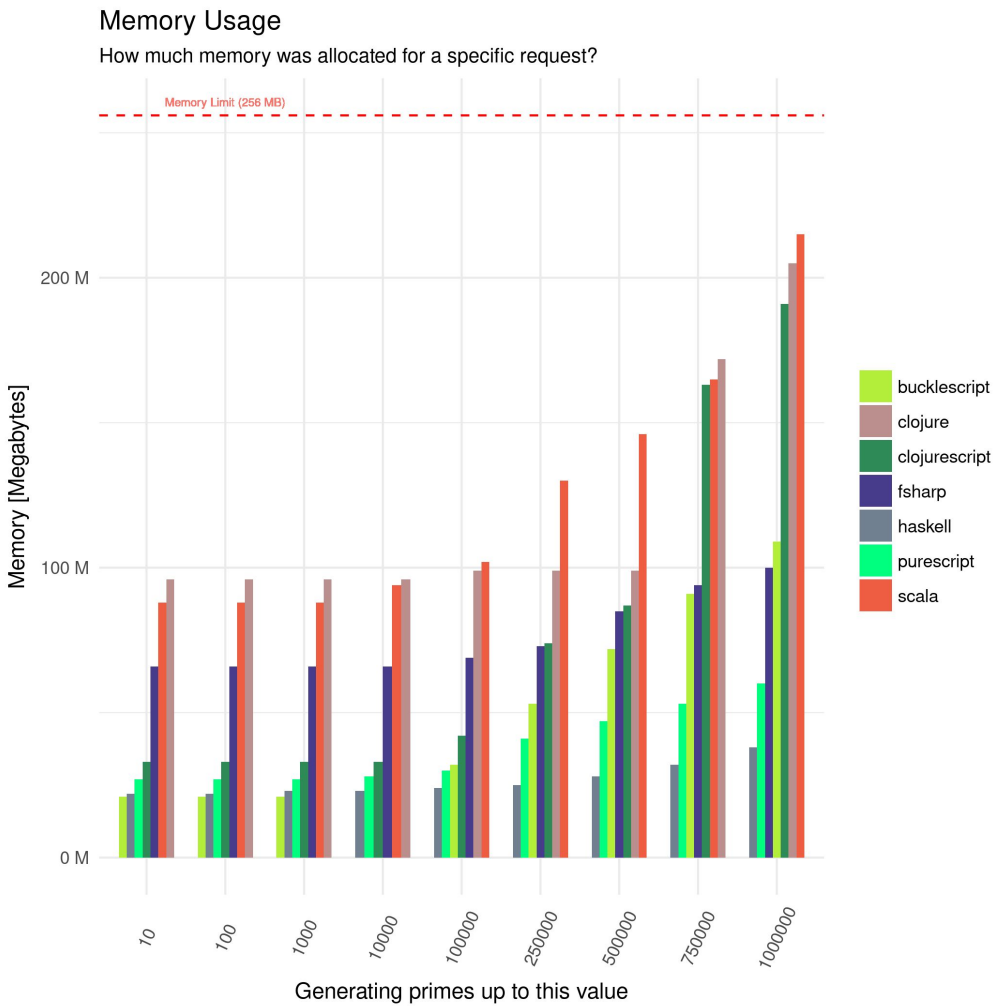
elixir

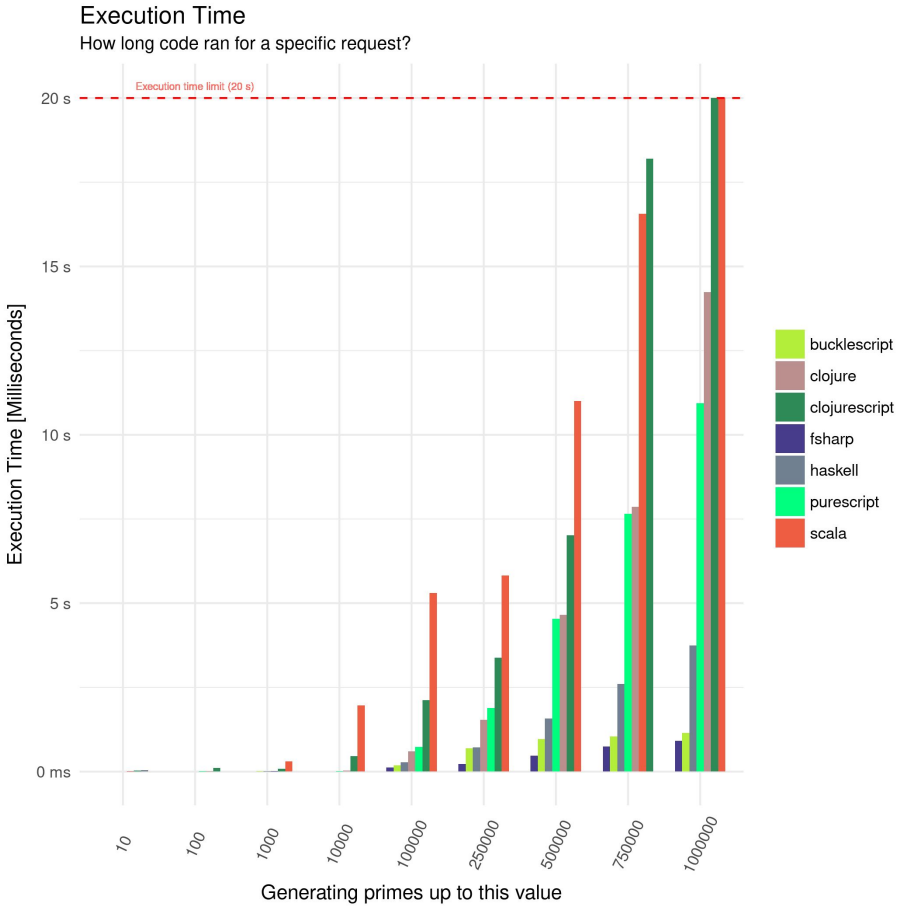
Measurements

Test Scenarios

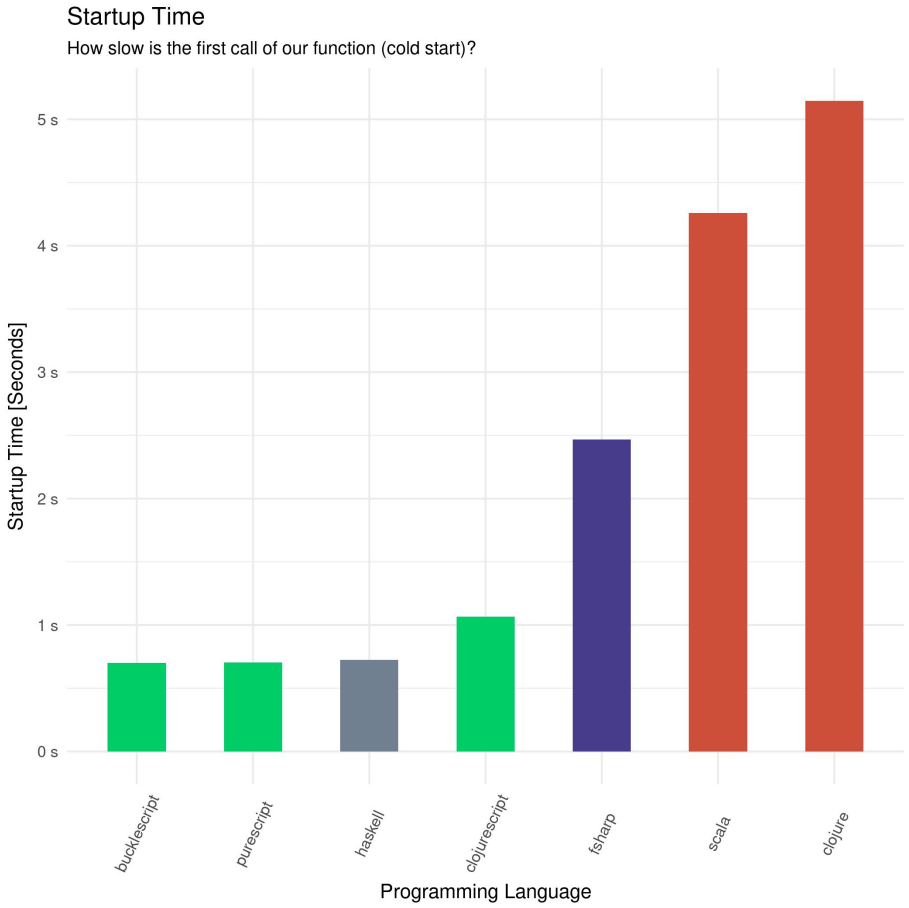
1. **Two endpoints**:
   a. Echoing incoming body.
   b. Sieve of Eranthoses (*naive implementation*).
2. **Package Size**.
3. **Memory Usage** (dependent on input argument).
4. **Execution Time** (dependent on input argument).
5. **Startup Time** (*cold start*, no work done).


Package Size
How big is a package uploaded to the FaaS service?
Package Size [Byte]
15 M
10 M
5 M
0
haskell
purescript
fsharp
clojurescript
clojure
bucklescript
scala
clojurescript_no_opt
Programming Language


Memory Usage
How much memory was allocated for a specific request?
Memory Limit (256 MB)
200 M
100 M
0 M
Memory [Megabytes]
10
100
1000
10000
100000
250000
500000
750000
1000000
Generating primes up to this value
bucklescript
clojure
clojurescript
fsharp
haskell
purescript
scala



"Situated" Programs (from Rich Hickey's keynote at Clojure/Conj 2017)



How long does AWS Lambda keep your idle functions around before a cold start?

Constraints

Constraints

1. **No knowledge** about container and its reuse.
2. New approach requires **new paths**.
3. Limits **everywhere**!
4. Provider **limitations**.
5. **VM optimizations** required from the get go.
6. Beware the **toolchain**!

Let’s **recap**!

Safe default:

F# on Azure or AWS

(.NET Core 2.0)

APPLISCALE

Serverless Computing: Economic and Architectural Impact

Gojko Adzic
Neuri Consulting LLP
25 Southampton Buildings
London, United Kingdom WC2A 1AL
gojko@neuri.co.uk

Robert Chatley
Imperial College London
180 Queen's Gate
London, United Kingdom SW7 2AZ
rbc@imperial.ac.uk

http://www.doc.ic.ac.uk/~rbc/papers/fse-serverless-17.pdf

Gojko Adzic - Designing for the Serverless Age (GOTO 2017)

Context is **King**

λ

Thank you!

Questions?

APPLISCALE

1. Our company - Appliscale and job offers.
2. Me - afronski.pl and my talks.
3. **Functional Miners** meetup (facebook, twitter, github, email).
4. Serverless Architecture, Serverless (Martin Fowler's articles).
5. AWS Lambda, Google Cloud Functions, Azure Functions.
6. Container Reuse in AWS Lambda.
7. Running Executables in AWS Lambda.
8. Serverless Computing: Economic and Architectural Impact.
9. Why the JVM is a Good Choice for Serverless Computing.
10. Droplr Serverless Revolution.
11. Optimizing Enterprise Economics with Serverless.
12. Should I use *AWS Lambda* or *EC2*?
13. AWS Lambda support for .NET Core 2.0.
14. **Repository** with examples, scripts, and measurements.

References