Serverless Computation with OpenLambda

Scott Hendrickson, Stephen Sturdevant, Tyler Harter, Venkateshwaran Venkataramani†, Andrea C. Arpac-Dusseau, Remzi H. Arpac-Dusseau

† Unaffiliated
Web development in the cloud

**CDN**: static content (e.g., JavaScript)

**Compute**: dynamic logic (e.g., Python)

**Storage**: application data

Amazon Cloudfront → Amazon EC2 → DynamoDB

RPCs → Queries
Web development in the cloud

**CDN:** static content (e.g., JavaScript)

**Compute:** dynamic logic (e.g., Python)

**Storage:** application data

- **Amazon CloudFront** (CDN)
- **Amazon EC2** (VMs)
- **DynamoDB** (Storage)

- RPCs
- Queries
Web development in the cloud

**CDN**: static content (e.g., JavaScript)

**Compute**: dynamic logic (e.g., Python)

**Storage**: application data

compute is evolving
Web development in the cloud

**CDN**: static content (e.g., JavaScript)

**Compute**: dynamic logic (e.g., Python)

**Storage**: application data

RPCs

Queries

(Containers)

compute is evolving
Web development in the cloud

**CDN**: static content (e.g., JavaScript)

**Compute**: dynamic logic (e.g., Python)

**Storage**: application data

(Lambdas)

compute is evolving
Web development in the cloud

**CDN:** static content (e.g., JavaScript)

**Compute:** dynamic logic (e.g., Python)

**Storage:** application data

prior to the Lambda model, cloud compute was neither *elastic* nor *pay-as-you-go*
Outline

Evolution of compute

**Non-conventional virtualization**

Lambda model

Why OpenLambda?

Conclusion
How to virtualize compute?
Classic web stack

RPCs

Application

Server

OS

Hardware
Classic web stack

RPCs

Application

Server

OS

Hardware

weak virtualization
1st generation: virtual machines
1st generation: virtual machines
1st generation: virtual machines

advantages:
- very flexible
- use any OS

problems:
- interposition
- is RAM used? (ballooning)
- redundancy (e.g., FS journal)
2nd generation: containers

advantages:
- centralized view
- init H/W once
Are containers good enough?
Container case studies

**Literature**: Google Borg

- Internal container platform [1]
- 25 second median startup
- 80% of time spent on package installation
- matters for flash crowds, load balance, interactive development, etc

[1] Large-scale cluster management at Google with Borg.  
Container case studies

**Literature**: Google Borg

- Internal container platform [1]
- 25 second median startup
- 80% of time spent on package installation
- matters for flash crowds, load balance, interactive development, etc

**Experimental**: Amazon Elastic Beanstalk

- Autoscaling cloud service
- Build applications as containerized servers, service RPCs
- Rules dictate when to start/stop (various factors)

[1] Large-scale cluster management at Google with Borg.
Interesting “autoscaling” rule
Experiment

Simulate a small short burst

- Maintain **100 concurrent requests**
- Use **200 ms** of compute per request
- Run for **1 minute**
Container Case Study: Elastic Beanstalk
Container Case Study: Elastic Beanstalk

![Graph showing latency and percent of requests](image)
Conclusion: Elastic Beanstalk does not scale quickly enough to handle load bursts.
**Conclusion**: Elastic Beanstalk does not scale quickly enough to handle load bursts.
Why should it take minutes (or even seconds) to execute scripts that are <1000s of LOC?
2nd generation: containers

Advantages:
- Centralized view
- Init H/W once

Problems:
- Large deployment bundle
- Server spinup
2nd generation: containers

RPCs

Application
Server
OS
Hardware

virtual OS
3rd generation: Lambdas

RPCs → Application → Application

Server and Runtime

OS

Hardware

virtual servers
3rd generation: Lambdas

serverless computing
3rd generation: Lambdas

Decompose application
3rd generation: Lambdas

- **advantages:**
  - very fast startup
  - agile deployment
  - share memory

- **problems:**
  - not flexible
Lambda elasticity

Repeat ElasticBS experiment
  • Maintain 100 concurrent requests
  • Spin 200 ms per request
  • Run for 1 minute
Lambda elasticity

The graph shows the percent of requests (y-axis) over latency (x-axis). The AWS Lambda curve increases gradually until it reaches 100% at around 2 seconds of latency. The Elastic BS curve shows a sudden 100% increase after 20 seconds of latency.
Lambda elasticity

**Conclusion**: Lambdas are highly elastic
Lambda elasticity

Conclusion: Lambdas are highly elastic (though a little slow)
Outline

Evolution of compute

Non-conventional virtualization

**Lambda model**

Why OpenLambda?

Conclusion
Lambda model

Run user handlers in response to events
  • web requests (RPC handlers)
  • database updates (triggers)
  • scheduled events (cron jobs)

Pay per function invocation
  • actually pay-as-you-go
  • no charge for idle time between calls
  • e.g., charge $actual\_time \times memory\_cap$
Share everything

Share server pool between customers
  • Any worker can execute any handler
  • No spinup time
  • Less switching

Encourage specific runtime (C#, Node.js, Python)
  • Minimize network copying
  • Code will be in resident in memory
Multi-node architecture

load balancers

Load Balancer

Load Balancer

...

handler store

workers

Python

Server

Python

Server

...

...
Multi-node architecture

load balancers

Load Balancer

... 

Load Balancer

handler store

H1

H2

developer

upload code

workers

Python

Server

Python

Server

...
Multi-node architecture

**load balancers**
- Load Balancer
- ... (multiple load balancers)
- Load Balancer

**handler store**
- Small: H1, H2

**workers**
- Large: Python, Server
- Small: Python, Server (three more instances)

...
Multi-node architecture

load balancers

user

Load Balancer

RPC

Handler store

H1

H2

... workers

Python

Server

Python

Server

...
Multi-node architecture

load balancers

user

RPC

Load Balancer

... 

Load Balancer

workers

Python

Server

handler store

H1

H2

Python

Server

...
Multi-node architecture

load balancers

user → Load Balancer

Load Balancer → ...

Load Balancer

handler store

H1, H2

workers

H2 → Python

Python → Server

Server

...

RPC
Multi-node architecture

load balancers

- Load Balancer
- Load Balancer
- ...

user

RPC

handler store

H1

H2

workers

Python

Server

H2

sandbox

Python

Server

...

...
Multi-node architecture

load balancers

user

RPC

Load Balancer

Load Balancer

handler store

H1 H2

workers

H2 sandbox

Python

Server

Python

Server
Outline

Evolution of compute

Non-conventional virtualization

Lambda model

Why OpenLambda?

Conclusion
Need for open source serverless

Many research areas

- Applications, tools, distributed systems, execution engines
- Evaluate ideas by **building**, not just simulating
Need for open source serverless

Many research areas
  • Applications, tools, distributed systems, execution engines
  • Evaluate ideas by **building**, not just simulating

First implementations are proprietary

- AWS Lambda
- Google Cloud Functions
Need for open source serverless

Many research areas
- Applications, tools, distributed systems, execution engines
- Evaluate ideas by building, not just simulating

First implementations are proprietary
- AWS Lambda
- Google Cloud Functions

OpenLambda: explore further-reaching techniques
- Goal: enable academic research on Lambdas
- Storage awareness, kernel support, RPC inspection
- ...

...
Need for open source serverless

Many research areas
- Applications, tools, distributed systems, execution engines
- Evaluate ideas by building, not just simulating

First implementations are proprietary
- AWS Lambda
- Google Cloud Functions

OpenLambda: explore further-reaching techniques
- Goal: enable academic research on Lambdas
- Storage awareness, kernel support, RPC inspection
- ...

Other recent open-source implementations
- Azure Functions
- IBM OpenWhisk
OpenLambda research topics

**Workloads**
- Workload studies
- Benchmarks
- Versioning+dependencies
- Code characteristics
- Package management

**Distributed systems**
- Databases
- Load balancing
- Scatter gather patterns
- Sessions and streams

**Tools**
- Debugging
- Monetary cost optimization
- Porting legacy applications

**Execution engines**
- Sandboxing
- Containers
- Just-in-time interpreters
OpenLambda research topics

**load balancers**

- Load Balancer
- Load Balancer
- ...

**workers**

- H2
- Python
- Server
- ...

**handler store**

- H1
- H2

**user**

- RPC

**developer**

- upload code
OpenLambda research topics

load balancers

handler store

workers

user

RPC

workloads

Load Balancer

Load Balancer

... 

H1

H2

distributed systems

developer

tools

upload code

Python

Server

H2 sandbox

execution engine
Understanding Lambda workloads

**Collaborate** with industry, measurement studies
- e.g., Azure Functions

**Build** LambdaBench
- Everybody joining builds an application
- Ticketing, calendar, autocomplete, OCR, flash card, stock alert, blog, and scientific compute applications

**Trace** RPC calls (e.g., AJAX) of existing apps

Gmail:
OpenLambda research topics

load balancers

- Load Balancer
- Load Balancer
- ...

handler store

- H1
- H2

workers

- H2 sandbox
- Python
- Server

user

RPC

workloads

developer

tools

upload code

distributed systems

execution engine

...
OpenLambda research topics

load balancers

user

RPC

workloads

Load Balancer

Load Balancer

handler store

developer

tools

upload

code

H1 H2

distributed systems

workers

H2

sandbox

Python

Server

execution engine

...
Developer tools

Portability
• E.g., can Django apps run on Lambdas?

Debugging
• Understand Lambda flows, may be a complex graph

Optimizing expense
• Hard with containers: how to share 1-hour server time across requests?
• With Lambdas: know cost of every RPC and query
• Show where money is going
OpenLambda research topics

load balancers

user
RPC
workloads

Load Balancer
Load Balancer

handler store

developer
upload code
tools

distributed systems

workers

H2 sandbox
Python
Server

execution engine

OpenLambda research topics
OpenLambda research topics

load balancers

- Load Balancer
- ...
- Load Balancer

handler store

- H1
- H2

workers

- H2
- sandbox
- Python
- Server

execution engine

user

workloads

RPC

developer

tools

upload code

distributed systems
Building locality-aware Lambdas

Use deep inspection of RPCs for routing

• Working with gRPC group
• GSOC project (Stephen Sturdevant)
Building locality-aware Lambdas

Use deep inspection of RPCs for routing
  • Working with gRPC group
  • GSOC project (Stephen Sturdevant)

Locality factors
  • code locality
  • data locality
  • session locality
Building locality-aware Lambdas

Use deep inspection of RPCs for routing
  • Working with gRPC group
  • GSOC project (Stephen Sturdevant)

Locality factors
  • code locality
  • data locality
  • session locality

Load Balancer

Lambda workers
Building locality-aware Lambdas

Use deep inspection of RPCs for routing
- Working with gRPC group
- GSOC project (Stephen Sturdevant)

Locality factors
- code locality
- data locality
- session locality

Lambda workers
- A - numpy
- B - numpy

Load Balancer

call B (uses numpy)
Building locality-aware Lambdas

Use deep inspection of RPCs for routing
- Working with gRPC group
- GSOC project (Stephen Sturdevant)

Locality factors
- code locality
- data locality
- session locality

Load Balancer

Lambda workers

DB shard: A-M (keys)

DB shard: N-Z (keys)

call B

predict X

local query
Building locality-aware Lambdas

Use deep inspection of RPCs for routing
- Working with gRPC group
- GSOC project (Stephen Sturdevant)

Locality factors
- code locality
- data locality
- session locality

Lambda workers

browser

Load Balancer

web socket

Lambda Engine

session service

Lambda Engine

session service
OpenLambda research topics

Load Balancers

Server

Python

workers

H2 sandbox

H2

execution engine

user

RPC

workloads

Load Balancer

Load Balancer

handler store

H1

H2

developer

tools

upload code

distributed systems

OpenLambda research topics

Load Balancers

Server

Python

workers

H2 sandbox

H2

execution engine

user

RPC

workloads

Load Balancer

Load Balancer

handler store

H1

H2

developer

tools

upload code

distributed systems
OpenLambda research topics

load balancers

user
RPC
workloads

Load Balancer
...
Load Balancer

handler store

H1 H2
developer
tools
upload
code
distributed systems

workers

H2 sandbox
Python
Server
execution engine
...

RPC
workloads
Minimizing latency

![Graph showing latency distribution with AWS Lambda and Elastic BS]
Minimizing latency

How can we reduce base latency?
Execution engine

Sandboxing

• Process VMs (e.g., JVM): how to mostly initialize?
• Containers: how to speed up restart and optimize pausing?
Execution engine

Sandboxing
- Process VMs (e.g., JVM): how to mostly initialize?
- Containers: how to speed up **restart** and optimize **pausing**?

Language runtimes
- Challenge: **code warms up** over time
- How to share dynamic optimizations?

![Diagram showing code execution and optimization](image)
Outline

Evolution of compute

Non-conventional virtualization

Lambda model

Why OpenLambda?

Conclusion
Conclusion

Lambdas finally deliver on promises of the cloud

• finally pay-as-you-go
• finally elastic
• will fundamentally change how people build scalable applications

New challenges in every area of systems

• scheduling, isolation, languages, debugging, tools, storage, …

Getting involved

• contribute at https://github.com/open-lambda
• site: https://open-lambda.org/