



Function as a Service / Serverless Computing

COSC349—Cloud Computing Architecture

David Evers

Learning objectives

- Explain what **Function as a Service** (FaaS) is
- Contrast FaaS with IaaS / PaaS / SaaS
- Indicate why FaaS might suit **Internet of Things** apps.
- Understand why FaaS systems are **stateless & reactive**
- Sketch how FaaS is supporting rise of **edge computing**

Function as a Service (FaaS)

- FaaS is a common form of **serverless computing**
 - *i.e.*, trend away from needing to considering servers at all
 - Tenant's focus is instead just **providing functions** to execute
 - *c.f.*, **lambda functions**: anonymous functions passed to functions
 - Variety of popular languages available for writing functions
- FaaS embodies **data flow programming**
 - *i.e.*, transformations to data happen when data is ready
 - *e.g.*, change to a spreadsheet cell—also data flow programming

Distributed stream processing systems

- Contrast DBs with **distributed stream processing** systems
 - Databases run queries when instructed to do so
- Stream processing systems define a **data flow graph**
 - **Sources**—tuples are emitted from them
 - **Operators**—nodes that transform n input streams to m outputs
 - **Sinks**—tuples are output or stored
- Computing is **triggered** by new tuples appearing
- Many high-quality, scale-out, open source DSPSs:
 - Apache Storm, Apache Spark Streaming, Apache Flink, ...

Event-driven programming

- In IaaS, VM is yours, so your code is **always running**
- In PaaS, still usually a sense that your code is active
 - PaaS auto-scales the server instances that run your code
- In FaaS, your code operates in **a reactive style**
 - Reactive programming typically relies on callbacks
 - Some sort of shared event dispatcher **issues callbacks**
 - You do not need to be aware of server instances at all
- Of course servers still need to run your code...
 - FaaS may have wide **variance in function execution latency**

AWS Lambda—public FaaS cloud

- **AWS Lambda** (2014) was first successful FaaS, then
 - Google Cloud Functions, Microsoft Azure Functions, ...
 - Apache OpenWhisk (open source—initiated by IBM), ...
- AWS Lambda **supports many programming languages:**
 - Python, Java, Node.js, Go, Ruby, and C# (.Net core)
 - Other effects can use call-outs to Linux executables
- Aims for **millisecond startup latency**
 - Caching will likely mean significant speed-up from recent use

Pricing for AWS Lambda

- Pricing based on number of **requests** and their **duration**
- **Request cost** is \$0.20 per million per month (US East)
 - ... but the first million requests per month are free
- **Duration cost** is \$0.0000166667 per GB-second (x86)
 - So involves both time and allocated memory you've chosen
 - ... but the first 400,000 GB-seconds per month are free
 - Memory allocation can be as low as 128 MB
 - (So the free tier will go a long way, for small-scale applications)

AWS Lambda event sources

- Typical use cases for FaaS include reactions such as:
 - Objects are updated or added to an **S3 bucket**
 - Updates are made to an **Amazon database** platform
 - **Sensor readings** arrive to the cloud (we will discuss IoT soon...)
- Only want to pay **when your code is running**:
 - Avoid paying for overheads like time to start/stop VM
 - Avoid paying to keep VMs in a ready state to handle requests
- Lambda well suits **app-specific interlinking** of AWS tools

Internet of Things (IoT)

- IoT embodies ambition of **all devices being networked**
 - Devices including cars, streetlights, toasters, wireless sensors,...
 - (But many IoT devices have low-quality security engineering)
- For sensor networks, want to **offload data processing**
 - e.g., extend lifetime of battery-powered devices
- Often sensor data will be **disseminated periodically**
 - FaaS facilitates app-specific data checks and transformations
 - Provides a reliable endpoint for real-world devices to interact with

Edge computing

- Cloud computing suits many types of jobs
 - However some data processing **needs distribution**
 - (Maintain history's central/decentralised computing oscillation?)
- Edge computing is half-way **between cloud and IoT**
 - Often full-size computing devices, widely distributed
 - ... but not at cluster scale (so **not scale-out** edge computing)
- CloudFront CDN can run AWS Lambda functions
 - e.g., **personalise web content** within any AWS Region's DC

Stateless functions

- Often a requirement that **transformers are stateless**
 - Easy to run in parallel when invocations are independent
 - Greatly helps scaling up applications
 - Fault tolerance: can **re-run failed functions** safely
 - Also can run a set of functions to check for **consistent answer**
 - Can add function inputs to compensate for statelessness
- Likely to require **reengineering of legacy apps**
 - Most apps' functions are not purely reactive and stateless
 - ... but ideal to use stateless design and let cloud optimise state