Transformation of a Legacy Airport Meteorology Application into a Serverless Cloud Application

IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI 2023)

Ondrej Habala

Institute of Informatics of the Slovak Academy of Sciences

May 26, 2023

Outline

Introduction

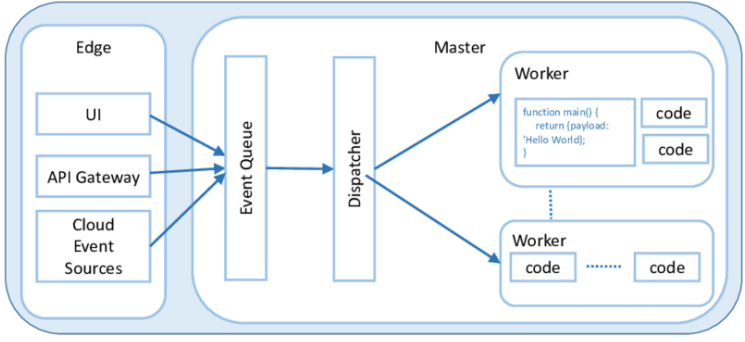
- We present part of the construction of an airport visibility meteorological application based on the Function-as-a-Service paradigm
- monolythic version already in use at airports
- our work: transformation into a server-less application
- goal of the paper: to provide other developers with a methodology on how to use FaaS systems

Motivation

- For the research partner:
  - part of long-standing research in cloud computing
  - acquisition of new know-how
  - opportunity to verify research results in real life use
- For the application developer:
  - modularization of application, allowing several different deployments (different requirements, functionality, pricing)
  - unburdens customers from hardware acquisition and maintenance
  - easier development of new functionality, even spanning new domains
  - opportunity to gain know-how in modern computing paradigms

What is Function as a Service?

- Subset of serverless computing that provides a platform allowing developers to write and deploy applications without building and maintaining the underlying infrastructure
- infrastructure management (resource provisioning, maintenance and regular update of base operating systems) are the responsibility of the Cloud provider
- developer focuses only on applicaiton code and logic



Advantages of FaaS

- Automatic scaling: functions are scaled automatically, independently, and instantaneously according to the actual demands by the cloud provider. That relieves developers from concerns of high traffic or heavy use.
- Cost efficiency: Users have to pay only for the computing resources they really use, not for idle resources that are often reserved for handling possible high demands in the typical IaaS (scaling by cloud provider).
- Quick development: developers don't have to manage infrastructure, they can focus only on the code, reducing the cost of development and the time to market.

Disadvantages of FaaS

- Potential vendor lock-in: The application codes are built on the top of a concrete FaaS platform and difficult to port to another vendor.
- Difficulties for testing: The codes are running on the top of a FaaS platform, it may make difficulties for creating local test environments for applications

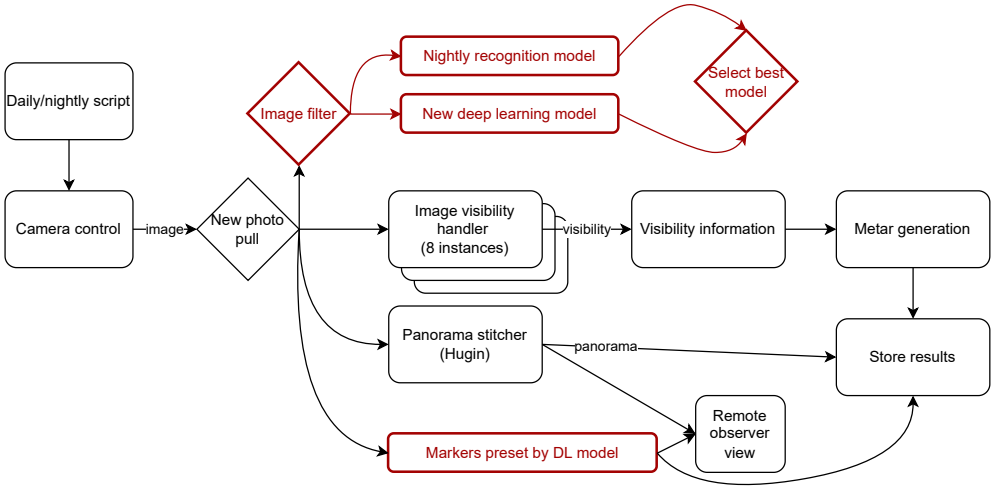
Current FaaS Frameworks

- The first commercial provider offering FaaS is Amazon AWS with **AWS Lambda** platform,
- followed by Google with **Google Cloud Functions**.
- We will focus on two open-source platforms: **Apache OpenWhisk**, originally by IBM,
- and **OpenFaaS** (by a company of the same name)

Application - Motivation

- Visibility is a crucial element in the safety of all kinds of transport
- Almost 50% of all aircraft accidents is due to weather conditions
- main cause of weather-related aviation accidents is reduced visibility
- better visibility information also leads to better traffic management
  - reduced fuel consumption
  - reduced flight delays

Application - Architecture

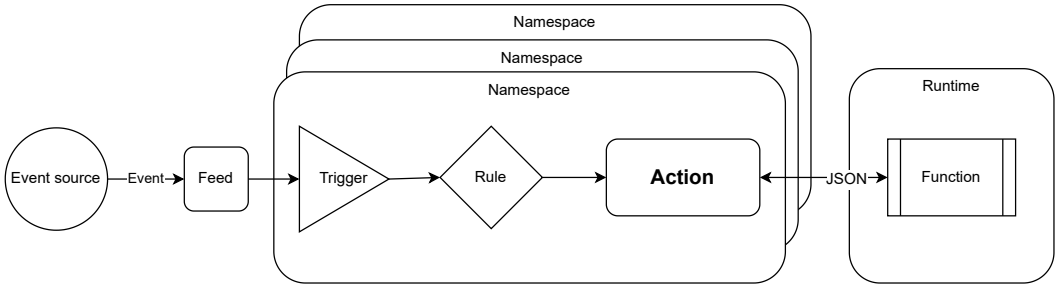


OpenWhisk - Origin and Use

- free and open implementation of the Function-as-a-Service paradigm
- tied to AWS Lambda service, first presented in November 2014
- started by Rodric Rabbah of IBM Research
- initially developed at IBM Research as Whisk
- later renamed to OpenWhisk, made OSS and transferred to the Apache Software Foundation Incubator
- can be installed in several ways, we have used Kubernetes
  - Helm chart for OpenWhisk is available
  - local installation of wsk command-line tool also necessary on the developer's machine

OpenWhisk - Programming Model

- an event-driven system
- event from an even source feeds into a trigger
- a trigger uses rules to execute an action
- uses REST API to accept new events
- accepts functions in several languages (Java, Python, PHP, Go, Ruby...) as well as black-box code



OpenWhisk - Application

- application will be implemented according to the architecture shown above
- so far implemented:
  - ImageVisibilityHander, as a Java action
  - Visibility info from 8 xmls, also as a Java action
  - Panorama stitch, as a black-box docker action, since we use a 3rd party software (Hugin[1])
- In the case of the Panorama stitch action, we have created a specific docker image, based on OpenWhisk's Docker Runtime image, adding the Hugin software for panorama-stitching
- Panorama stitch could also be done as a Python action with a custom docker image for OpenWhisk's Python actions

[1] Hugin - Panorama Photo Stitcher, `https://hugin.sourceforge.io/`

Panorama Stich as black-box action

- use of 3rd party tool - Hugin - requires a black-box action
- OpenWhisk support is available - template docker image
- template has Docker Runtime proxy - listens to incoming action calls
- Our modified docker image contains Hugin and a Python script to configure it
- needed modifications to configuration - too small memory limits

Black-box action runtime script

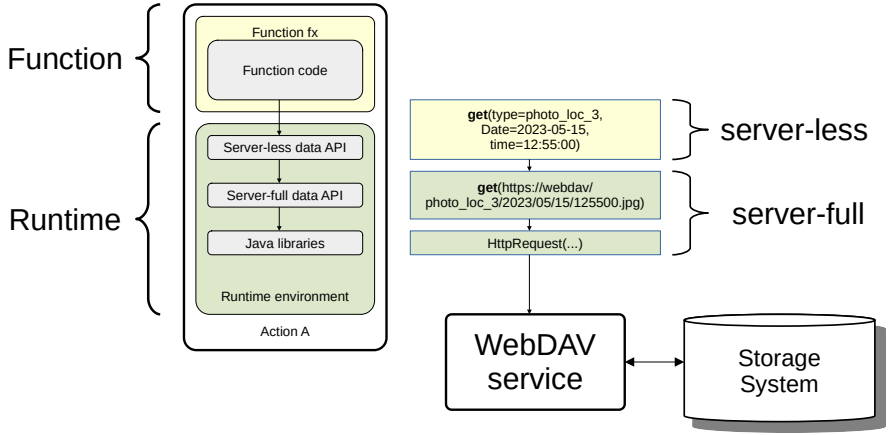
- receives the JSON input structure from the Docker Runtime proxy
- decodes it into function parameters, including input image file URLs
- downloads the input images,
- runs Hugin to do panorama stitching of the downloaded images
- uploads the resulting panorama image to the cloud storage of the application, and
- finishes, indicating success or failure in the function result.

Java runtime modifications

- inclusion of 3rd party libraries required during runtime
- adding any static data, in our case configuration files for visibility algorithms
- built on OpenFaaS which provides automatic function deployment
- a storage client library - for accessing the WebDAV storage of the system
- upper layer of the storage client library is metadata-based

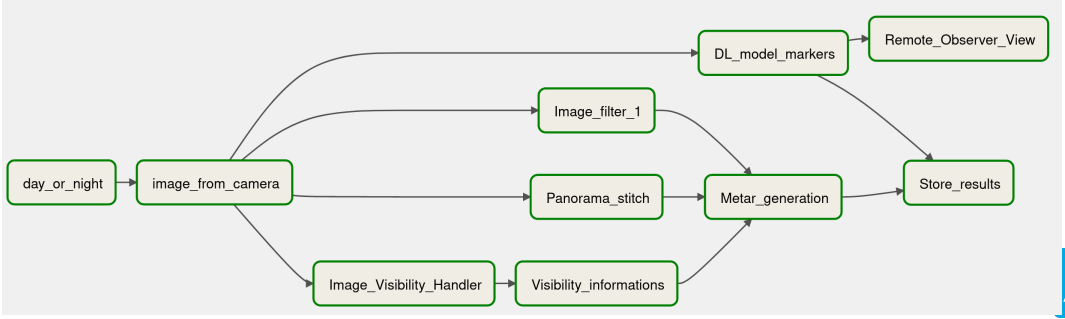
Storage architecture

- a server-less layer API
- a server-full layer API
- storage service



Airflow

- open source framework for lightweight serverless functions management
- amalgamates individual tasks into a workflow which is expressed as a directed acyclic graph
- in our case the tasks are serverless functions
- the resulting workflow characterizes the relations between its tasks which also defines their execution order.

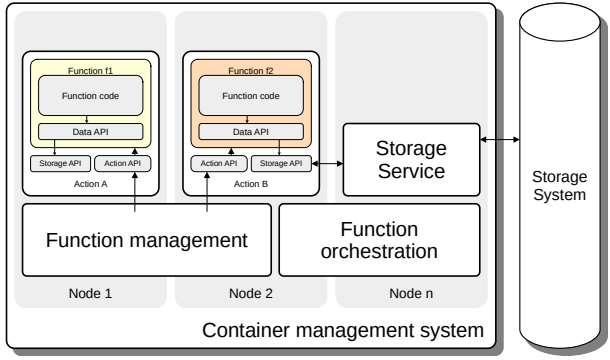


Summary

- continuing work on the transformation of a monolithic application into the serverless cloud domain (FaaS)
- we have chosen the open source OpenWhisk platform, it allows the event-driven execution of actions
- created a custom docker image for black-box action (panorama stitching using Hugin)
- created a custom Java docker image including additional libraries, config data, storage client library

Future Work

- transformation of additional parts of the application into FaaS actions
- use of workflow manager to tie together the application workflows
- more complete description of the methodology of porting legacy applications to FaaS
- a generalized server-less application architecture



Thank you!

For questions please write to ondrej.habala@savba.sk