Serverless Edge Orchestration

September 25 - 27, 2018
Amsterdam, The Netherlands
Eden Rozin
CPO, Product Management
AT&T Tel-Aviv, Israel
Edge Computing

• Key enabler for 5G
• Decentralized architecture
• Latency issue mitigation
• Essential for IOT
However...

- Distributed across 1000s of locations
- Limited space & real estate
- Limited cooling and power
- Scarce computing resources
- Significant workload support
- Runs 3rd party software
## Orchestration on Edge vs Central Cloud Challenges

<table>
<thead>
<tr>
<th></th>
<th>Edge</th>
<th>Central Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Location of application components</strong></td>
<td>Location of nodes plays significant role in application blueprint</td>
<td>Pretty much location-agnostic</td>
</tr>
<tr>
<td><strong>Mobility of workloads</strong></td>
<td>Workload transition from one node to the other</td>
<td>Static unless there is a cloud node failure</td>
</tr>
<tr>
<td><strong>Workload dynamic</strong></td>
<td>Various applications need to run at various times to serve different needs</td>
<td>Static workload most of the time. One you deploy a service, it is there forever</td>
</tr>
<tr>
<td><strong>Architecture heterogeneity</strong></td>
<td>Edge is made of different nodes, various sizes, vendors and technologies. Large, small, PNFs, Akraino, Green Grass, Azure Edge, etc.</td>
<td>Mostly homogeneous. If it is Openstack, AWS or Azure, it is the same Cloud OS for all nodes, and diversity is considerably small</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>Latency and distance from the end consumer plays major role</td>
<td>Most central cloud apps are not latency-sensitive</td>
</tr>
<tr>
<td><strong>Availability of resources</strong></td>
<td>Edge nodes are small; availability of resources for application is not guaranteed</td>
<td>Availability of resources is pretty much guaranteed. This is one of the basic principles of any cloud</td>
</tr>
</tbody>
</table>
### Monitoring on Edge vs Central Cloud Challenges

<table>
<thead>
<tr>
<th></th>
<th>Edge</th>
<th>Central Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Distributed data collection</strong></td>
<td>Collection needs to be done from thousands of distributed nodes across the network</td>
<td>Everything is centralized and collected to a central DB</td>
</tr>
<tr>
<td><strong>Architecture heterogeneity</strong></td>
<td>Edge is made of different vendors; each has its own metrics and APIs</td>
<td>Each cloud vendor has its own collection and monitoring framework (OS Ceilometer, AWS CloudWatch, etc.)</td>
</tr>
<tr>
<td><strong>Distributed root cause analysis</strong></td>
<td>Identification of the root cause and its impact on the service in distributed environment</td>
<td>Although it’s complicated, it’s still simpler than doing it on the edge network</td>
</tr>
<tr>
<td><strong>Distributed closed loop</strong></td>
<td>Location and latency take major role in recovery, mitigation plan</td>
<td>Recovery is much simpler. Most of the time it’s to spin up another instance</td>
</tr>
</tbody>
</table>
## Data Management on Edge vs Central Cloud Challenges

<table>
<thead>
<tr>
<th></th>
<th>Edge</th>
<th>Central Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supporting ACID (transactions)</td>
<td>Distribution and partition of the edge is a challenge for every transactional DB</td>
<td>Everything is in one place; just install SQL DB</td>
</tr>
<tr>
<td>High availability of DB</td>
<td>Replication of DB is not practical in most cases</td>
<td>No problem having any H/A solution on central cloud</td>
</tr>
<tr>
<td>Latency</td>
<td>Latency requirements prevent using a DB on central cloud; DB needs to be local to the apps</td>
<td>Apps are close to the DB in central cloud, no latency issues</td>
</tr>
<tr>
<td>Mobility/Availability of data on the edge nodes</td>
<td>The environment is dynamic so all data needs to be available to all nodes although it is distributed</td>
<td>No such issue in central cloud</td>
</tr>
</tbody>
</table>
Edge Operating System Manifesto

• Treat the Edge as one big distributed compute
• Harness distribution for availability and reliability
• Data is available anywhere on the Edge network
• Execute workload anywhere on the Edge network
• Intelligent resource management
• Location-sensitive workload orchestration
• Expand application beyond Edge boundaries (Public Cloud, DC, etc.)
• No single point of failure
Akraino Edge Stack

- The industry adopted cloud native for edge
- Containers have smaller footprint than VMs
- Improved resource utilization
- Micro-services architecture

However...

- Integration of new micro-service is complicated
- Permanent allocation of resources
- Container is still larger execution unit
Introducing Serverless
FaaS (Function-as-a-Service)

- Functions are the unit of deployment and scaling
- No machines, VMs, or containers visible in the programming model
- Permanent storage lives elsewhere (SLE)
- Scales per request; Users cannot over- or under-provision capacity
- Never pay for idle (no cold servers/containers or their costs)
- Implicitly fault-tolerant because functions can run anywhere
- Bring Your Own Code (BYOC)
- Metrics and logging are a universal right
Functions in a Nutshell

Events:
- Triggers: Data path event, Timer, OSS/BSS, etc.

Serverless Function Router:
- Network events create triggers that result in a function launch.
- Function executes per transaction and exits on completion.
- Functions hold no internal state, rely on external resources for persistent data.
- Function can trigger other functions creating function chains.

Resources:
- Function
- Persistent object storage
- Messaging

FaaS Platform:
- FaaS platform maps triggers to functions, provisions and executes functions.
- The FaaS platform typically related on clustering for scale-in / scale-out.
## What is Serverless good for?

### GOOD 🌟
- Data collection & enrichment
- Mobile backend (Control Plane)
- File processing
- Web backend
- IoT Backend
- Stream processing

### NOT GOOD 🔴
- Long running persistent processes
- Network traffic processing (routers, gateways, firewalls)
- Databases
What did we built?
Akraino Based Serverless Edge Node with IoT Gateway
ONAP SDC, SO Orchestration and Monitoring Infrastructure
Intelligent Transport System (ITS)

Congestion avoidance system
Re-route connected cars to alternative routes:

1. Function deployment for each car vendor
2. Function mobility
3. Manual scale-out to accommodate load
Detailed Demo Architecture

Load orchestration

Visualization

Central Cloud

Cloudify Micro Services

Monitoring Micro Services

Visualization Micro Services

Event Load Generator Micro Services

Subscriber Management Micro Services

kubernetes Docker Cluster

FaaS orchestration, Kubernetes orchestration

Event Generator

MQTT

IOT Gateway

FaaS API

Fass Platform

kubernetes Docker Cluster

Subscriber persistence (Endpoint shadow) Mobility across edge POPs

Mazda Function

Toyota Function

Ford Function

Edge Cloud POP
Modeling the Serverless Edge Stack using ONAP SDC
Edgility Code Contribution to Akraino
Thank You!

Contact Details: er434w@att.com