SOSCON 2018
Edge Computing
Smart Home | Smart Factory | Smart City

Samsung R&D - India | IoT | Ashok Babu Channa
Date: October 17-18
Contents

I Intro: Edge?
II Edge Schema
III EdgeX Foundry
   Architecture Style
   Messaging Framework
Conclusion
Contents

I  Intro : Edge ?
II Edge Schema
III EdgeX Foundry
   Architecture Style
   Messaging Framework
Conclusion

SOSCON 2018
SAMSUNG OPEN SOURCE CONFERENCE 2018
Technology Innovation wave

**Devices**
Extracting signals from Sensors and Things and connect to network

**Cloud**
Global access and unlimited computing

**Edge**
Logical extreme of network for Intelligent computing

**AI**
Intelligence and inference at cloud or edge
Why Edge? Need

- Data Privacy?
  - Data Security
  - Regulations

- Slow response time?
  - Network Latency
  - Encrypt/Decrypt

- Too much Bandwidth?
  1.5 GB for one person per day

(1ZB = 10^{21} \text{ bytes})
## Edge | Solution

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Cloud</th>
<th>Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geo-distribution</td>
<td>Centralized</td>
<td>Distributed</td>
</tr>
<tr>
<td>Distance client and server</td>
<td>Multiple hops</td>
<td>One hop</td>
</tr>
<tr>
<td>Latency</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Delay Jitter</td>
<td>High</td>
<td>Very low</td>
</tr>
<tr>
<td>Location awareness</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Support mobility</td>
<td>Limited</td>
<td>Supported</td>
</tr>
<tr>
<td>Location of service</td>
<td>Within the Internet</td>
<td>At the edge</td>
</tr>
</tbody>
</table>

**IDC:** By 2020, 45% of IoT data used and stored @Edge

Contents

I  Intro : Edge ?
II  Edge Schema
III  EdgeX Foundry
   Architecture Style
   Messaging Framework
Conclusion

SOSCON 2018
SAMSUNG OPEN SOURCE CONFERENCE 2018
**Multi-Tier Edge:**
- Number of deployed services and functionality increases higher in Tier
- An operator is able to constitute their optimized Edge under given use cases and requirements.

<table>
<thead>
<tr>
<th>Field Devices</th>
<th>Simple Edge GWs</th>
<th>Intelligent Edge GWs</th>
<th>Edge Servers</th>
<th>Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collecting Edge</strong></td>
<td>• Ingestion for local generated data</td>
<td>• Front-line data filtering for redundancy, query for stream data</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Computing Edge</strong></td>
<td>• Data pre-processing and analytics at the Edge with collected data</td>
<td>• Basic ML/CEP for reacting to local events (e.g. alert security when intruder detected)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Storage Edge</strong></td>
<td>• Aggregated data for analytics of overall performance</td>
<td>• Streaming data from all process lines, more complex analytics</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Portfolio Level**
- Deep learning in the cloud to optimize manufacturing processes across entire enterprise portfolio
Edge Platform | Coverage

**Storage Edge**
- Distributed data storing
- Metadata management

**Computing Edge**
- Data processing/analysis
- Machine learning

**Collecting Edge**
- Data collecting
- Stream data query

**EDGE Platform / Device**
- Server
- PC, Mobile
- ARTIK
- Raspberry Pi

**Service deployment**

**Operation Sol.**
- Docker Registry
- CI / CD
- Tool
- Login
- Admin / Operator

**Commit**
- Service Developer
### Edge Platform | Goal

<table>
<thead>
<tr>
<th>Microservice Architecture</th>
<th>Data Collect/Transfer/Storage Tech.</th>
<th>Data Analytics at Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ <strong>Docker-based Microservice</strong></td>
<td>□ Securing <strong>Standardized data modeling</strong> for data transferring analytics improvement</td>
<td>□ <strong>Data Processing Framework</strong> for Quality/ Process from generated Data through devices</td>
</tr>
<tr>
<td>Architecture for Platform-agnostic service dev. And legacy service employment</td>
<td>□ <strong>Real-time data streaming and message framework</strong> for collection/analytics of large-scale data</td>
<td>□ <strong>Stream data query tech.</strong> for real-time data filtering &amp; event processing for analytics and inference</td>
</tr>
<tr>
<td>□ <strong>Lightweight Edge device</strong> for establishing testbeds powered by Samsung Edge P/F</td>
<td>Examples:</td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td>Examples:</td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td><img src="image1.png" alt="docker" /></td>
<td><img src="image2.png" alt="AutomationML" /></td>
</tr>
<tr>
<td></td>
<td><img src="image3.png" alt="OMQ" /></td>
<td><img src="image4.png" alt="TensorFlow" /></td>
</tr>
</tbody>
</table>

**Examples:**

- Data Collect/Transfer/Storage Tech.
  - Securing **Standardized data modeling** for data transferring analytics improvement
  - **Real-time data streaming and message framework** for collection/analytics of large-scale data

- **Docker-based Microservice** Architecture for Platform-agnostic service dev. And legacy service employment
- **Lightweight Edge device** for establishing testbeds powered by Samsung Edge P/F

**Examples:**

- **Docker**
- **OMQ**
- **AutomationML**
- **TensorFlow**
Edge Platform | Functions

- **Edge S/W Platform for real-time data analytics and collection**
  - “One Total Platform” that be able to run on **Multi-Tier Edge devices** → Dev. Efficiency ↑
    * All-in-One PC, VISA, Local Server, etc.
  - Data analytics at Edge → Save the cost of data management, real-time operation, security ↑

**Smart Services**
- Monitor / Control
- Abnormality Detection
- Outage Analysis/Prediction
- Process Optimization

**Cloud S/W Platform**
- Historical Data Analytics

**Device S/W Platform**
- Machine/ Sensors / Things Control
EdgeX Open Source & Samsung Contributions
EdgeX Foundry™ is a vendor-neutral open source project hosted by The Linux Foundation building a common open framework for IoT edge computing.

At the heart of the project is an interoperability framework hosted within a full hardware- and OS-agnostic reference software platform to enable an ecosystem of plug-and-play components that unifies the marketplace and accelerates the deployment of IoT solutions.

Architected to be agnostic to silicon (e.g., x86, ARM), OS (e.g., Linux, Windows, Mac OS), and application environment (e.g., Java, JavaScript, Python, Go Lang, C/C++) to support customer preferences for differentiation.
- Project progress since its launch from April in 2017
  - Considerable Momentum, now over 65 ecosystem members with most recent additions of Samsung at Platinum and Thales, Cavium, and Wanxiang Group at Silver. More in flight.
Containerized architecture: A common open framework for Edge Platform

- **Device**: Defines common means to create device interfaces using preferred connectivity protocols
- **Core**: Ensures interoperability between northbound and southbound micro services through common APIs
- **Supporting**: Provide edge analytics and intelligence, from logging, scheduling, and data clean up (scrubbing)
- **Export**: Interface with cloud services with client registration and off-gateway management
- **Security**: Protect the data and command of devices, sensors, and other IoT objects managed by EdgeX
- **System Mgmt**: Provide the installation, upgrade, start, stop, and monitoring of micro services and BIOS, OS, and other gateway-related S/W
EdgeX | What we have done

Best set of combination for achieving the maximum “Data Intelligence”
Micro Services | Server architecture

Monolithic

- User Interface
- Business Logic
- Data Access

Micro Services

- User Interface
- Business Logic
- API Gateway
- Micro Services

Advantages: Scalability, Availability and Easy deployment
Messing @ Edge

Synchronous:
1. MS makes a request.
2. MS waits for a response.
3. MS gets the response and does something.

Asynchronous:
1. MS makes a request.
2. MS continues working.
3. MS gets the response.
4. MS does something.

Publisher/Subscriber (Pub/Sub):
1. Publisher sends a message.
2. Subscriber continues working.
3. Subscriber subscribes to the message.
4. Subscriber continues working.
Messaging Framework | Architecture

External Clients (External edge, cloud, etc)

API
JAVA, CPP, GO, C, Node.js

Messaging Framework

Messaging
Publisher, Subscriber
ZeroMQ

Distribution
Data Model (AML), Registration
Protocol Abstract, Device Service SDK
OPC-UA Protocol, Milo library

Device Service (OPC-UA)

Storage
Interface (REST), DB Manager
DB (NoSQL/RDBMS)

Legend
- Functional Component
- Component Module
- External Component
- OPC-UA Protocol

Event Driven Messaging
Data Model & Device Services
Storage
Messaging Framework | Scenario

- Micro services (Publish) send topic name and endpoint info to Topic Name Service.
- Micro services (subscriber) can discover topic names via Topic Name Service.
- Topic based data streaming between micro services using ezMQ.
- A standard data model (AutomationML) can be transferred.
Re-cap

**Edge:**
- Why Edge - Need | Expectations
- Edge Schema – Multi Edge and 3-Tier
- Edge Platform – Goal | Functions

**EdgeX & Samsung OS Contribution**
- Edge Architecture – Micro Services
- Edge Messaging – Architecture | Deployment
References

EDGEXFOUNDRY™
https://www.edgexfoundry.org/

Samsung Smart Factory @ EdgeX
https://wiki.edgexfoundry.org/display/FA/Smart+Factory+Project

Samsung Open Source for EdgeX
https://github.com/mgjeong/
Questions ?
THANK YOU