A Microservice Approach to IoT Edge Computing

Jim White - Dell Technologies
Agenda

• The inherent challenges of IoT
• Introduce EdgeX – a microservice architecture for the edge
• Addressing 5 key challenges of the edge & how microservices help
• Microservices no silver bullet – challenges microservice architecture bring
• Resource list for more info, call to action
Who is this guy?

• Jim White
  • Dell Technologies IoT Solutions Division – Distinguished Engineer
  • Team Lead of the IoT Platform Development Team
  • Chief architect and lead developer of Project Fuse
    • Dell’s original IoT platform project that became EdgeX Foundry
    • Yes – I wrote the first line(s) of code for EdgeX (apologies in advance)
  • EdgeX Foundry …
    • Technical Steering Committee member
    • Ad hoc and unofficial lead architect
Why is IoT hard to do?

- Heterogeneity of platforms
  - Diverse collection of OS and OS variants
    - Linux, Unix, Windows, VxWorks, embedded and RTOS, …
  - Various Hardware (Intel, AMD, ARM, …)
  - Cloud, gateway, smart thing (the “Fog continuum”)

- Thing protocol soup
  - Industrial: BACNet, Modbus, OPC-UA,…
  - Wireless: BLE, Z-Wave, Zigbee,…
  - Message: MQTT, AMQP, DDS, …

- Variety of cloud platforms
  - Azure IoT Hub, AWS IoT Platform, Google IoT Core, IBM Watson IoT Platform, …

- Add your favorite selection of…
  - Applications, edge analytics/intelligence, security, system management, …

- Difficulties in determining where to start

---

IoT is a post doctorate in all we know and have done in computing for the last 30-40 years

- Networks/protocols
- Mobile computing
- Distributed compute
- Cloud compute
- AI/Machine learning
- …
Introducing EdgeX Foundry

• An open source, vendor neutral project (and ecosystem)
• A microservice, loosely coupled software framework for IoT edge computing
• Hardware and OS agnostic
  • Remain agnostic with regard to microservice implementation
  • Many of the microservices were in Java and are now in Go
  • C/C++ is envisioned for south side connectors and to address real time needs
  • JavaScript for UI
• Goal: enable and encourage growth in IoT solutions
  • The community builds and maintains common building blocks and APIs
  • Plenty of room for adding value and getting a return on investment
  • Allowing best-of-breed solutions
A Brief EdgeX History

• Chartered by Dell IoT marketing in July 2015
  • A Dell Client CTO incubation project (Project Fuse)
• Designed to meet interoperable and connectivity concerns at the IoT edge
• Started with over 125,000 lines of Dell code
• Entered into open source through the Linux Foundation on April 24, 2017
  • Started with nearly 50 founding member organizations; today we have more than 75
• Release Cadence: 2 formal releases a year

<table>
<thead>
<tr>
<th>2017</th>
<th>2018</th>
<th>2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct</td>
<td>Nov</td>
<td>Dec</td>
</tr>
<tr>
<td>Jan</td>
<td>Feb</td>
<td>Mar</td>
</tr>
<tr>
<td>Apr</td>
<td>May</td>
<td>Jun</td>
</tr>
<tr>
<td>July</td>
<td>Aug</td>
<td>Sep</td>
</tr>
<tr>
<td>Oct</td>
<td>Nov</td>
<td>Dec</td>
</tr>
<tr>
<td>Jan</td>
<td>Feb</td>
<td>Mar</td>
</tr>
<tr>
<td>Apr</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

‘Barcelona’ Release (Released Oct 20 2017)

‘California’ Release (June 2018)

‘Delhi’ Release (Oct 2018)

‘Edinburgh’ Release (Apr 2019)
Disclaimer

• I am not here to sell you on EdgeX Foundry (the product or org)
  • Although that would be a nice by-product 😊

• I am here to present a microservice based solution to solve some of the more challenging IoT problems
  • The EdgeX implementation helps to demonstrate (validate?) the concept
  • Consider the architecture in your decision making
  • Use our lessons learned where you can
  • Replicate/duplicate if you must
  • Join us if you can
    • If you think the approach correct and you don’t feel like starting from scratch
EdgeX Primer - How it works

- A collection of a dozen+ microservices
  - Written in multiple languages (Java, Go, C, … we are polyglot believers!!)
  - Several commonly used library projects (common domain objects, client libraries, etc.)

- EdgeX data flow:
  - Sensor data is collected by a **Device Service** from a thing
  - Data is passed to the **Core Services** for local persistence
  - Data is then passed to **Export Services** for transformation, formatting, filtering and can then be sent “north” to enterprise/cloud systems
  - Data is then available for edge analysis and can trigger device actuation through Command service

- REST communications between the service
  - Some services exchange data via message bus (core data to export services and rules engine)

- Microservices are deployed via Docker and Docker Compose
It's 102°C. Stop the machine.
Performance Targets

• The target is to run all of EdgeX on a Raspberry Pi 3 type of device
  • 1 GB RAM, 64bit CPU, at least 32GB storage space

• Additional “developer community” targets
  • Startup in 1 minute or less (post OS boot)
  • Latency for one piece of data from data ingestion to actuation will be < 1 second

• Remaining OS and Hardware agnostic
  • Windows, Linux, *nix, …
  • Intel/Arm 64/Arm 32

Current #'s

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Footprint</td>
<td>76 MB</td>
</tr>
<tr>
<td>Footprint with container</td>
<td>113 MB</td>
</tr>
<tr>
<td>Memory (idle)</td>
<td>26 MB</td>
</tr>
<tr>
<td>Memory with 100 devices</td>
<td>40 MB</td>
</tr>
<tr>
<td>Startup time</td>
<td>&lt; 10 sec</td>
</tr>
</tbody>
</table>

(without DB or device services)
The Challenges of IoT

• #1 Dealing with the diversity
  • Dealing with the diversity of device connectivity protocols
  • Working with multiple cloud and enterprise systems
  • Dealing with multiple IoT data models and data formats

• #2 Incorporating any analytics package

• #3 Allowing for the continual improvements and upgrades of parts of the IoT solution

• #4 Respond to evolving business needs and technological advancements (how to make a ROI in IoT)

• #5 Addressing limited resources in an IoT environment
Problem 1: protocols, models, and formats

• IoT is inherently a multi-transform problem
• Communicating across multiple protocols, using different data models and formats (JSON, XML, etc.)
  • From “thing” to edge platform (like a gateway)
  • From edge to application layer or cloud
  • Sometimes from thing to cloud
  • To/from analytics applications
Transformation on the “South Side”

• The world of OT protocol soup
  • Modbus, BACNet, Profibus, CANBus, OPC-UA…

• Consumer and traditional IT protocols are also entering the mix
  • BLE, Zigbee, ZWave, MQTT, SNMP, …

• New “things” & protocols are being added all the time
  • You can never keep up with them all

• EdgeX Device Services transform from “thing” protocols and data to common Core Data (micro) Service
Device Services

- Core Data
- Core Command
- Device Service (per protocol)
- REST/JSON
- Driver
- Sensor/Device Protocol
- Sensor
- SDK

"Southbound" devices, sensors, and actuators
Transformation on the “North Side”

• North side endpoints need data…
  • Filtered
  • Transformed (data model of choice)
  • Enriched (add device metadata, location, etc.)
  • Formatted (XML, JSON, CSV, …)
  • Compressed, Encrypted, etc.

• This is classic EAI (enterprise application integration)
  • aka pipe & filter architecture

• EdgeX Export Services take the data from Core Data and get it to the applications/cloud
Export Services

- Core Data
- Export Client
- Export Distro
- Export Services
  - HTTP(s) endpoint
  - MQTT(s) endpoint
  - 0MQ endpoint
  - Azure IoT Hub
  - Google IoT Core

- Filter(s)
- XML
- JSON
- CSV
- Compress
- Encrypt
Problem 2: edge analytics

- Intelligence or analytics at the edge is critical
  - It’s too expensive to transport all the data “north”
  - It’s too expensive to store all the data “north”
  - It’s too late to react to a problem from the cloud
  - Your devices/sensors are not always connected to the “north”

- How smart does your edge platform need to be?
  - Simple rule engine smart?
  - Complex event process (CEP) smart?
  - Machine learning/AI smart?

- The edge platform must be flexible enough to incorporate different capability
- EdgeX’s analytic service can wrap and isolate the edge analytic capability
Rules Engine Service

EdgeX Reference Implementation

3rd Party Value Add
Problem 3: Fast Continual Improvements

- Microservice architecture allows for continual improvements, future break throughs
  - Ex: device services that do their own device discovery
  - Ex: streaming analytics over core data/analytics services
- Upgrades to microservices without impact to others
  - Example: upgraded the config/registry service that used Consul 0.8 to Consul 1.0
- Improve performance over time, so they fit on more constrained devices
  - Ex: Moving from Java to Go for massive performance and footprint improvements
- Grow them over time and distribute/migrate them to the cloud
  - Ex: Machine Learning analytic services with a local edge agent
- Promote best of breed solutions
- Allow specialization to occur
Problem 4: Differentiation and Value Add

• EdgeX was created with commercialization in mind!!!

• Allow for value additions
  • RedHat style commercial support packages (IoTech)
  • Improved data synchronization between edge and cloud (MongoDB)
  • Edge analytics customized for the vertical or use case (many orgs)
  • Security features to protect trust devices, secure the data, etc. (RSA)

• Allow for low/no value commodity to be taken care of by a community
  • Ex: open logging probably suffices for a large part of the user base

• Allow specialization to demand higher price
  • Ex: IoTech creating a real time extension on the south side for embedded systems
  • Ex: Aicas using Jamaica and other technologies to run faster/smaller Java microservices

• Provide incentive to commercial (even competing) companies and reason to support an open source effort
Problem 5: How to maximize the use of resources

• From sensor to cloud, there exist a continuum of
  • Compute
  • Storage
  • Networking/connectivity
  • Management
  • Security

• How can all these resources be maximized?
  • Sensors will get more compute and storage
  • Pipes to the cloud may get bigger
Microservice Distribution

- Microservices can live where they can get the resources they need
- With a tendency to push to the south
  - Latency needs
  - Storage and transportation costs
  - Disconnected modes
- Allow the microservices to adapt to the use case
- Requires extremely loose coupling
- In some uses, microservices might be collapsed or combined
EdgeX Flexible Deployment Possibilities
Microservices challenges

• Microservices offers aid to addressing some of the significant IoT issues
  • A microservices architecture inherently introduces challenges

• Performance
  • More microservices = more communications
  • More communications = more latency concerns

• Orchestration
  • Deploying microservices (especially when distributed across platforms)
  • Managing/updating microservices
  • Configuring (how to provide platform dependent/environmentally dependent configuration to each service)
  • Registering (how does one service know where to go to get another service)
  • Getting status/health (where microservices are dependent on one another, how do you know a microservice is up and ok)

• More points of failure

• Security – more interfaces and endpoints to secure
Addressing Microservice Challenges

• Performance
  • Combined/collapsed services on occasion
  • Real time versions or components are being developed (IoTech EdgeX RT)

• A bevy of products can help with deployment and orchestration
  • Docker, Compose, Snappy, Kubernetes, Mesos, Swarm, …

• Combine some services to reduce points of failure

• Security services offer some of the most opportunity for 3rd party value add
  • Ex: take advantage of hardware root of trust
  • Ex: distributed ledgers/blockchain

• Cloud and system management tools have addressed many of the points of failure issues
  • Provides many know solutions to take advantage of
Now Backed by 70+ Members

With more in process!
Current Status

- EdgeX California Release on track for release at the end of June 2018. Key features include:
  - Initial security building blocks (reverse proxy, secure store)
  - Most services transitioned from Java to Go (exception: device services and SDK)
  - Dramatically improved performance, resource usage, and footprint (~7x reduction in size)
    - Already hitting our system performance targets
  - Additional “northbound” connectors
  - Improved documentation (documentation treated more like code in its management)
  - Arm 64 support
  - Blackbox testing for all services
  - Improved continuous integration
- Technical Steering Committee meet in Palo Alto, June 4-6
  - Scoped next release (code named Delhi) due Oct 2018
  - Roadmapped future releases (Edinburgh – Apr 2019, Fuji – Oct 2019)
- Current membership: ~70 companies/organizations
  - Code contributions from ~40 developers
## EdgeX Releases

<table>
<thead>
<tr>
<th>2017</th>
<th>2018</th>
<th>2019</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oct</td>
<td>Nov</td>
<td>Dec</td>
</tr>
<tr>
<td>‘Barcelona’ Release</td>
<td>‘California Preview’ Release</td>
<td>‘Delhi’ Release</td>
</tr>
<tr>
<td>- Improved fit and finish, formalized Core Service APIs, additional Device and Export Services, test apparatus</td>
<td>- Drop-in Go Lang microservice replacements demonstrating reduced footprint and higher performance</td>
<td>- Additional security and first manageability capabilities</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- First integration of security</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Run in &lt; 1 GB RAM, come up in &lt; 30 sec, &lt; 1 second actuation latency</td>
</tr>
<tr>
<td></td>
<td>‘California’ Release</td>
<td>‘Edinburgh’ Release</td>
</tr>
<tr>
<td>(Released June 2018)</td>
<td>(Apr 2019)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Go / C device service SDKs</td>
<td>- Certification Program</td>
</tr>
<tr>
<td></td>
<td>- EdgeX UI</td>
<td>- Improved and more scalable northbound connectors</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Southbound connectors to common protocol devices</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ARM 32 support</td>
</tr>
<tr>
<td></td>
<td>‘Fuji’ Release</td>
<td></td>
</tr>
<tr>
<td>(Oct 2019)</td>
<td></td>
<td>- Load balancing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Multi-host EdgeX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Additional security and system management capability</td>
</tr>
</tbody>
</table>

 edgedxfoundry.org | @edgexfoundry
Delhi Major Themes & Objectives

• Smaller development cycle (due to California length) so scope has to match
• High level scope
  • Initial System Management APIs and agent
  • Device Service SDKs (Go/C) & at least one example device service
  • The next wave of security features
    • Access control lists to grant access to appropriate services, and improved security service bootstrapping
  • Improve testing
    • Better/more unit, complete black box and add performance testing
  • Refactored and improved Go Lang microservices
  • Design and architecture work in advance of Edinburgh release
    • Options and implementation plan for database replacement
    • Design and implementation plans for export service replacement with application services
  • An EdgeX UI suitable for demos and smaller installations
Call to action

- We could use your help!
- There are plenty of places to contribute to EdgeX
  - Additional southside connectivity
  - Additional northside connectivity
  - Replacement and refactor work
  - Security & microservice management work
  - Checkout Github Issues and our roadmap for more
Key Project Links

• Access the code:
  • https://github.com/edgexfoundry

• Access the technical documentation:
  • https://wiki.edgexfoundry.org

• EdgeX Blog:
  • https://www.edgexfoundry.org/news/blog/

• Join an email distribution:
  • https://lists.edgexfoundry.org/mailman/listinfo

• Join the Rocket Chat:
  • https://chat.edgexfoundry.org/home
Thanks!

james_white2@dell.com