Sensor Observation Streams within Cloud-based IoT Platforms: Challenges and Directions

Antoine Auger, Ernesto Exposito, Emmanuel Lochin
ISAE-SUPAERO, Université de Toulouse, France

antoine.auger@isae.fr

09/03/2017
Track B: Internet of Things
Motivation

- Internet of Things (IoT) and data explosion

- Many “IoT platforms” for many use cases providing different services and/or APIs

- “Black box approach” so far
  Which requirements? Which software solutions? Why?

- Common challenges when building software platforms, need to explain:
  - Concepts, technical issues
  - How to cope with them, good practices
  - … Not just describe final results!
• **We focus on:**
  – **IoT platforms**: ingest information from *Things*, create value from data, services exposed to users or applications
  – **Cloud-based**: available on demand via the Internet, Software as a Service model (SaaS), other non-functional requirements
  – **Sensor observation streams**: physical-observed phenomena or virtual-occurred events

• **Contributions of the paper:**
  – Insights for researchers who are interested in IoT platforms capable of handling sensor observation streams
  – Takeaways for developers
Motivation

• “Lessons learned” coming from:
  – State of the Art
  – Study of available tools/components for IoT platforms
  – Attended conferences (in particular, see [Kamienski2016])
  – Own experience: development of an Integration platform for QoI Assessment as a Service (iQAS) [Auger2017]

• Disclaimer:
  – Not exhaustive!
  – May contain subjective thoughts…
1. Observation-related challenges  *What?*
   Observation streams, levels, quality

2. Implementation considerations  *What solution? How?*
   Observation collection, processing and consumption

3. Non-functional requirements  *With what features?*
   Adaptation, scalability, availability
Observation-related challenges

• We assume that an observation is *something* produced by a sensor (either physical or virtual)

• IoT platforms are asked to create value from received observations
  – Data-centric systems

• Heterogeneous observations:
  – Level
  – Quality
  – Sensing rate
  – Coverage
  – ...
Observation-related challenges

Observation levels

- Taxonomies to denote different observation levels
  - Useful to estimate the level of complexity required to process and “understand” observations by final consumers

An augmented version of the “DIKW ladder” of Sheth [Sheth2016]
Observation-related challenges

Observation quality

• Several quality dimensions:
  – Quality of Service (QoS)
  – Quality of Information (QoI) [Bisdikian2013]
  – Context, Quality of Context (QoC)
  – Quality of Knowledge (QoK)

In short:
✓ All dimensions not always needed
✓ Attributes to consider may vary according to use cases (delay, obs. lifetime, accuracy, trust, …)
✓ Recent works consider mainly QoS-QoI (e.g., FP7 CityPulse)
Observation-related challenges

Unbounded observation streams

• Operations on streams ≠ databases

• Observation streams
  – Unbounded
  – Heterogeneous
  – Real-time

• Reactive Streams
  – http://www.reactive-streams.org
  – Asynchronous stream processing with non-blocking back pressure
  – Java APIs
Implementation considerations

• A jungle of IoT platforms
  – Open-source solutions, projects:
    FP7 INSIGHT, FP7 OpenIoT, FP7 CityPulse, H2020 OrganiCity, …
  – Commercial or proprietary solutions (SaaS):
    AWS IoT, IBM Watson IoT, …

• Researchers may want to add new features:
  – By extending existing solutions: requires a learning phase
  – By developing new prototypes from scratch (preferred)

• Most popular data architectures:
  – Lambda Architecture http://lambda-architecture.net
  – Kappa Architecture http://www.kappa-architecture.com
Implementation considerations

Kappa Architecture

Cloud-based IoT platform

Observation ingestion

Speed (real-time) layer

Batch (offline) layer

Serving layer

Distributed storage

Lambda Architecture

Cloud-based IoT platform

Observation ingestion

Observation sources

Observation consumers

Observation collection

Observation processing

Observation consumption

Serving layer

Distributed storage

Job version n

Job version n+1

Unified Stream Processing layer

Result n

Result n+1

Serving layer

Distributed storage
Observation collection

• Virtual Sensor Containers (VSCs)
  – For preliminary tests
  – To assess platform scalability
  – APIs to modify sensor behavior on the fly

• Apache Nifi
  – [https://nifi.apache.org](https://nifi.apache.org)
  – Open source, web-based GUI
  – Run in local/cluster mode
  – Supports directed graphs of data routing, transformation, and system mediation logic
  – Allows you to develop your own processors
Observation processing

- Event Stream Processing (ESP)

- Pick one framework according to your needs
  - Final choice may depend on many features [ESPcomparison]
Implementation considerations

Observation consumption

- **Message Brokers**
  - “Shock absorbing” technology
  - Compliant with Reactive Streams initiative

- **Trend: Kafka for the IoT**
  - Developed by LinkedIn, open sourced in 2011

- **Benchmarks for comparison but biases**

- **Real bottlenecks: final consumers**

<table>
<thead>
<tr>
<th>RabbitMQ</th>
<th>ActiveMQ</th>
<th>Kafka</th>
</tr>
</thead>
<tbody>
<tr>
<td>First release</td>
<td>2007</td>
<td>2012</td>
</tr>
<tr>
<td>Solution based on</td>
<td>AMQP</td>
<td>JMS</td>
</tr>
<tr>
<td>Distributed</td>
<td>yes (cluster)</td>
<td>yes (Zookeeper cluster)</td>
</tr>
<tr>
<td>Exchange types</td>
<td>Queues, Topics</td>
<td>Queues, Topics</td>
</tr>
<tr>
<td>Routing support</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Written in</td>
<td>Erlang</td>
<td>Java</td>
</tr>
<tr>
<td>Producer performance (messages/sec.)</td>
<td>25000</td>
<td>2000</td>
</tr>
<tr>
<td>Consumer performance (messages/sec.)</td>
<td>4800</td>
<td>5000</td>
</tr>
</tbody>
</table>
Non-functional requirements

- Functional requirements depend mainly on use cases.
- Some non-functional (NF) requirements are recurrent regarding Cloud-based IoT platforms.
- Among them, some NF requirements may be fulfill by relying on the Cloud Computing paradigm.
Non-functional requirements

Platform adaptation

- Autonomic Computing [Kephart2003]
  - IBM MAPE-K control loop
    - Monitor
    - Analyze
    - Plan
    - Execute

- IoT platforms act as mediators
  - Sensor capabilities
  - Consumer needs
  - Service Level Agreements (SLAs)

- Adaptation is needed
  - E.g., self-configuration, self-optimization, self-healing, …
Non-functional requirements

Platform scalability

- **Vertical scalability**
  - Allocate more resources
  - VM migration

- **Horizontal scalability**
  - Load balancing
  - Cluster

- **Scalability ≠ elasticity**
  - Scalability = add resources on-demand
  - Elasticity = add or remove resources on-demand

Taken from [http://pudgylogic.blogspot.fr/2016/01/horizontal-vs-vertical-scaling.html](http://pudgylogic.blogspot.fr/2016/01/horizontal-vs-vertical-scaling.html)
Non-functional requirements

Platform availability

• CAP theorem [Brewer2000] proposed by Eric Brewer
  – “You can only have 2 of 3 features among”:
    • Consistency
    • Availability
    • Tolerance to network Partitions

• CAP theorem has evolved [Brewer2012]:
  – Network partitions can be handled
  – Tradeoff instead of exclusive choice (latency considerations)

• IoT platforms raise new issues:
  – Platform availability may depend on the Cloud Provider (SLA)
  – Sensors availability
  – Is data consistency critical?
Other challenges

Many other challenges such as:

- General architecture for the IoT [FP7 IoT-A]
- Interoperability [FP7 Fiesta-IoT] [FP7 OpenIoT]
- Observation collection (e.g., CoAP protocol)
- Development tools to create IoT applications or services
- Specific challenges for Smart City platforms (many stakeholders)
- “Big Data” challenges: Volume, Value, Velocity, Variety, Veracity and more…

=> Not covered in this paper
Conclusions and perspectives

- **IoT data explosion**

- **A jungle of IoT platforms but:**
  - Complex to add new features to them, requires a learning phase
  - SaaS = “Black box approach”
  - Need to sensitize researchers and developers about concepts, issues and good practices

- **In this paper, we focus on sensor observation streams within Cloud-based IoT platforms**
  - Observation-related challenges
  - Implementation considerations
  - NF requirements and benefits of using Cloud Computing

[ESPcomparison] Available at: https://www.madewithtea.com/images/streaming.png


Sensor Observation Streams within Cloud-based IoT Platforms: Challenges and Directions

Antoine Auger
antoine.auger@isae.fr

#ICIN2017

This research was supported in part by the French Ministry of Defense through a financial support of the Direction Générale de l'Armement (DGA).