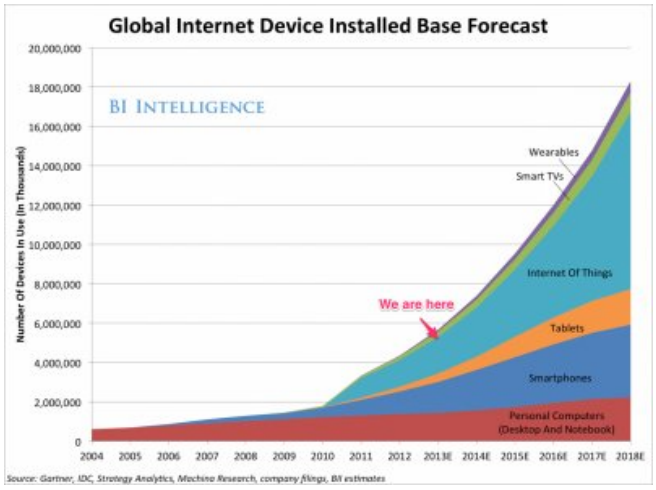
Whitepaper, 2014

Data Processing in the Internet of Things (IoT)

Table of Contents

What is the Internet of Things (IoT)?

The Internet of Things (IoT) is the concept of an ubiquitous network of devices to facilitate communication between the devices themselves, as well as between the devices and human end users. The involved devices are typically constrained devices such as RFID sensors, but more sophisticated ones like smartphones are also considered to be part of the IoT ecosystem.



Source: Business Insider Australia

The IoT is growing quickly (see also Fig. to the left) and is in fact going mainstream in verticals from manufacturing over utilities to consumer sectors such as wearables.

Processing data from IoT devices lends itself to the *Big Data approach*: this means, using scale-out techniques on commodity hardware in a schema-on-read fashion along with community-defined interfaces such as the Apache Spark API.

Why is that so? In order to develop a commercial-grade IoT application you need to be able to capture and store all the incoming sensor data to build up the historical references (volume aspect of Big Data). Then, there are dozens of data formats in use in the IoT world and none of the sensor data is relational per se (variety aspect of Big Data). Last but not least, many devices generate data at a high rate and usually we cope with data streams in an IoT context (the velocity aspect of Big Data).

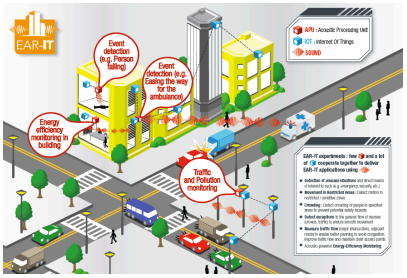
IoT Use Cases and Landscape

The use cases can be grouped along the scope of an IoT application, from a narrow scope over groups to entire organizations and areas:

Personal IoT sets the focus on a single person. A user might be wearing a smart watch (such as Android Wear or Apple Watch) measuring their heart rate and sharing this data with her doctor.

Group IoT has a scope of a fairly small group of people. For example, in the context of a smart home, this could be a family and the deployed sensors might capture temperature and lighting conditions for optimal comfort or in the context of a car, the people inside it.

The **Community IoT** takes a large group of people into account, potentially thousands or more. Typically, this is the case in the context of public infrastructure, such as smart cities, where sensors track cars and road conditions to orchestrate an efficient traffic flow.



Finally, **Industrial IoT** is the most established branch, with a scope either of an organization—such as a company operating a smart factory—or between organizations like the case in the retailer supply chain.

The IoT Landscape

As of 2014 there are a number of (commercial) entities in the realm of IoT, from the infrastructure level over the data (processing) level to the application level, including but not limited to the following:

Application level examples:

- Apple
- dweet.io
- Google
- Samsung
- wit.ai

Data (processing) level examples:

- Cisco
- IBM
- MapR Technologies

Infrastructure level examples:

- Qualcomm
- Ericsson
- Siemens

Naturally, the three levels are overlapping and do not have clear-cut boundaries. Further, some of the players are active in more than one level, for example Cisco provides infrastructure components, but also has offerings on the data (processing) level.

Requirements for an IoT Data Processing Platform

In order to process data from IoT devices at scale, a platform has to meet the following requirements:

- Native **raw data** support. Both in terms of data ingestion and processing, the platform should be able to natively deal with IoT data. Hadoop makes it possible to land the incoming data in its raw format (JSON, log files, etc.) and also, for optimization purposes, to convert data downstream to more sophisticated formats such as Apache Parquet[1].

- Support for a **variety of workload** types. IoT applications usually require that the platform supports stream processing from the get-go as well as deals with low-latency queries against semi-structured data items, at scale. Hadoop offers an append-only filesystem called HDFS to persist data. For stream data, usually

[1] http://parquet.incubator.apache.org/

messages queues such as Apache Kafka are used to buffer and feed it into stream processing systems such as Apache Storm.

- **Business continuity**. Commercial IoT applications usually come with SLAs in terms of uptime, latency and disaster recovery metrics, such as Recovery Point Objective (RPO) and Recovery Time Objective (RTO). Hence, the platform should be able to guarantee those SLAs, innately. This is especially critical in the context of IoT applications in domains such as health care, where people's lives are at stake.

- **Security & Privacy.** The platform must ensure a secure operation, including integration with existing authentication and authorization systems such as LDAP, AD, Kerberos, SAML or PAM. Last but not least, the privacy of the users must be warranted by the platform, from ACLs over data provenance support to data encryption and masking[2].

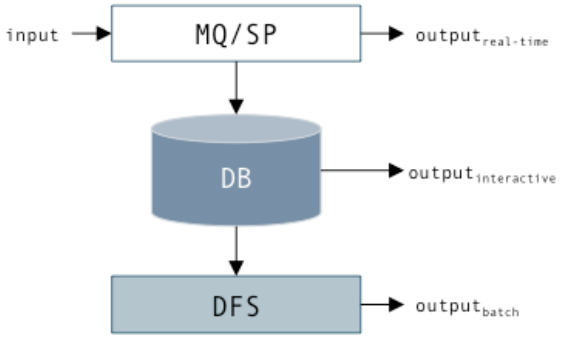
The current Hadoop architecture does not allow fulfilling above requirements out-of-the-box. While it is perfectly capable of dealing with raw IoT data, support for a wide array of workloads is rather limited. Recent upgrades in the stack, incl. YARN and Mesos, allow isolation on the compute layer, however, concerning the storage layer, HDFS only provides a flat namespace and this forces real-world apps to use separate physical clusters for different workloads.

Further, due to architectural constraints[3] of HDFS, it is not able to deal with many (small) files, in a read/write manner, also causing issues concerning business continuity (high availability and the capability to recover from disasters, meeting RPOs/RPTs).

[2] http://www.dataguise.com/
[3] https://www.usenix.org/legacy/publications/login/2010-04/openpdfs/shvachko.pdf

The Internet of Things Architecture (iot-a)

We now discuss a polyglot processing architecture, enabling us to develop and operate IoT applications, at scale: the ***I**nternet **o**f **T**hings **A**rchitecture* (or *iot-a*, for short):



The iot-a consists of three main building blocks discussed in the following.

MQ/SP … the *Message Queue/Stream Processing* block takes the `input` data and performs—depending on the application requirements—one or more of the following operations on it:

- *Buffering:* to adapt to the processing speed or throughput characteristics of downstream components, it often is necessary to buffer inbound data points. Equally, through micro-batching data points, the ingestion rate into downstream components can be increased.
- *Filtering:* ranging from simple cleansing operation to application-specific removal of certain data points.
- *Complex online processing over streams*: continuous queries, aggregates, counts, real-time machine learning algorithms, etc.

The MQ/SP can produce two kinds of `output` data: 1. real-time output, and (optionally) 2. ingestion output for downstream components (DB and/or DFS).

An example for the usage of an MQ/SP block—taken from an app of the Group IoT domain—is a heads up text message that is sent to the driver of a smart car in the event of a predicted engine malfunction. The most important aspect here is that the alert reaches the user in time, potentially preventing an accident.

DB … the *Database* block takes the data from the upstream MQ/SP and provides structured, fine-grained, low-latency access to the data points. Due to the nature of the data, the database block is typically a NoSQL solution, able to accommodate sparse data, with auto-sharding and horizontal scale-out properties. The database block `output` can is of interactive nature, with an interface provided either through a store-specific API (such as the HBase API) or through the standard interface SQL.

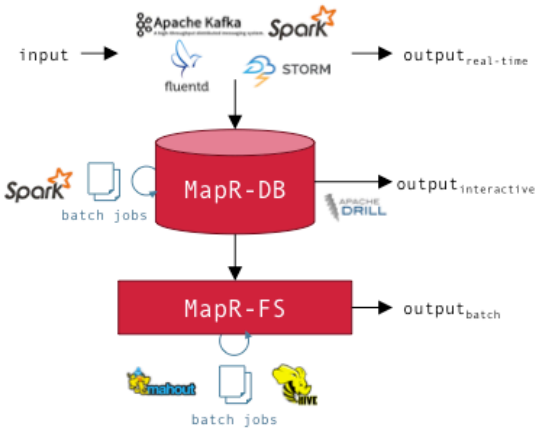
Staying with the smart car from above, an example usage of the DB block is as follows: a mechanic in a garage can, once the car owner approves a service-as-you-go, inspect the car's vital signal, asking ad-hoc questions and correlate it with other cars of the same build in order to asses potential damage and develop a repair strategy.

DFS … the *Distributed File System* block takes the data from either the DB block or directly from the MQ/SP block and performs batch jobs—usually aggregations and reporting type of jobs—over the entire dataset. This might include combining the data from IoT devices with other data sources, such as those delivering customer or product data or potentially unstructured ones, for example PDF documents.

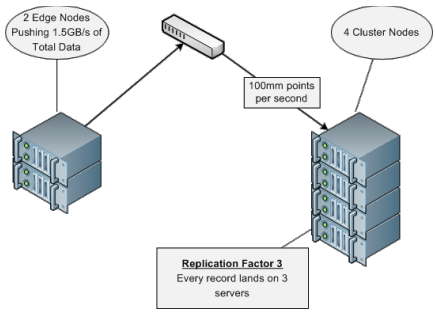
Again in the context of the smart car, an example usage of the DFS block goes as so: the car owner has access to a Web site where, on a weekly basis, metrics about the car, are made available in order for the owner to assess the overall health and performance of her vehicle.

MapR's Offering

The MapR Data Platform[4] offers to instantiate the *iot-a* in a straightforward, cost-sensitive and reliable way, meeting and exceeding the above listed requirements. The following is an example configuration:



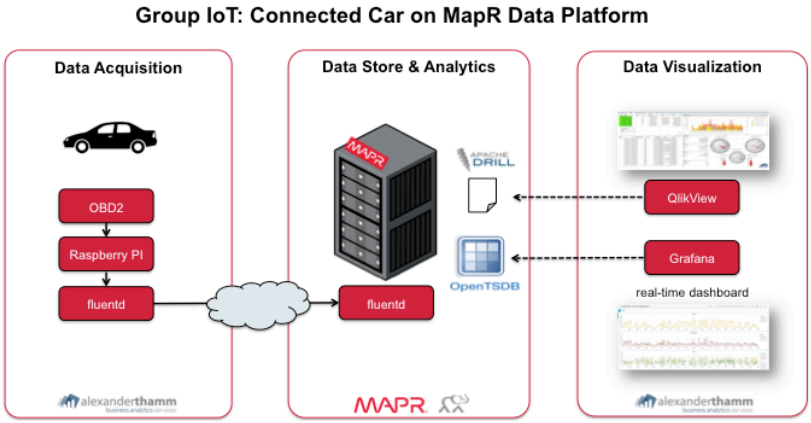
Time series play a central role in IoT applications and hence we've made sure that they are treated as first-class citizens in our offering. Through contributing[5] to OpenTSDB—a popular, HBase-based time series database that scales to billions of data points—we enable IoT applications to ingest more than 100 million data points per second.



[4] http://doc.mapr.com/display/RelNotes/Release+Notes
[5] https://www.mapr.com/blog/loading-time-series-database-100-million-points-second

We have a number of customers, from manufacturers, utilities over to governmental institutions and retailers that have already successfully deployed IoT applications based on the iot-a, using the MapR Data Platform.

Let's now have a closer look at a concrete example from the automotive sector: the connected car.



Together with our German-based partner Alexander Thamm GmbH [6] we have implemented an end-to-end system in a period of two weeks time. This proof of concept utilised data captured in a car through the OBD2 connector to show real-time driving characteristics. We streamed the data using an in-car Raspberry Pi into OpenTSDB via fluentd, driving the dashboard powered by Grafana. The inbound stream of data was landed on MapR-FS and we then used Apache Drill along with QlikView for an ad-hoc analytics visualization dashboard.

[6] http://alexanderthamm.com/

Conclusion

In this whitepaper we've reviewed IoT use cases and the IoT landscape and listed requirements for platforms processing IoT data. Further, we have discussed the IoT reference architecture, iot-a. We have demonstrated how to apply it in the context of the MapR offering and provided examples how customers successfully have deployed solutions using the MapR Data Platform, based on the iot-a.

If you have any questions concerning IoT or want to request a quote for servicing the design or implementation of an IoT application for your use case, please contact us via:

iot@maprtech.com

MapR delivers on the promise of Hadoop with a proven, enterprise-grade platform that supports a broad set of mission-critical and real-time production uses. MapR brings unprecedented dependability, ease-of-use and world-record speed to Hadoop, NoSQL, database and streaming applications in one unified big data platform. More than 500 customers use MapR across financial services, retail, media, healthcare, manufacturing, telecommunications and government organizations as well as by leading Fortune 100 and Web 2.0 companies. Amazon, Cisco, Google and HP are part of the broad MapR partner ecosystem. Investors include Google Capital, Qualcomm Ventures, Lightspeed Venture Partners, Mayfield Fund, New Enterprise Associates (NEA), and Redpoint Ventures. MapR is based in San Jose, CA.