Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 56 (2015) 183 – 188
Procedia Computer Science

The 12th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2015)

Designing a data management pipeline for pervasive sensor communication systems

Jussi Ronkainen*, Antti Iivari

VTT Technical Research Centre of Finland, Kaitoväylä 1, FI-90571 Oulu, Finland

Abstract

Pervasive sensor systems offer unbounded possibilities for monitoring and tracking objects, machines, and spaces. To maximize the benefit from a sensor system, sensor data requires efficient preprocessing and analysis. Big data techniques make distributed processing of huge amounts of data fast and cost-effective, making them a practical necessity for sensor data. However, the real-time requirements and the sheer velocity and volume of data from large sensor systems require a dedicated approach to designing the data processing pipeline. This paper discusses viewpoints and requirements for designing a sensor data pipeline, with specific focus on data input, live preprocessing, and storage.

Keywords: data pipeline; sensors; iot; big data

1. Introduction

Increased processing power, low-power connectivity options and the use of IPv6 enable the development of small embedded sensors that will bring everyday objects to the Internet. Location sensing and tracking options are also available at low cost and power consumption. This makes it possible to sense and track a vast number of objects, machines, and spaces in real time, and process their data in a cloud environment. This development is often referred to as the Internet of Things (IoT), and although there is a lot of hype around the phenomenon[1], its importance will be

* Corresponding author. Tel.: +358405418892; fax: +358207227001
E-mail address: jussi.ronkainen@vtt.fi

tremendous in the near future. There are already more "things" connected to the Internet than the number of people in the world, and by 2020, the number of connected devices is expected to be around 50 billion[2].

Sensor systems are only beginning to make use of the possibilities offered by cloud storage and distributed big data analysis[3]. Real-time requirements and the velocity and volume of data from millions of sensors pose analysis challenges that are fundamentally different from the ones faced with transactional or interactional data such as shopping transactions or Twitter feeds.

To manage this enormous amount of machine-generated data from pervasive sensor communication systems and the IoT, a distributed data processing pipeline is needed for collecting, preprocessing and storing the data for further analysis. In this paper, the requirements and initial architectural designs of the main components for such a sensor data processing pipeline will be presented. First, we take a look at the characteristics of machine-generated sensor data from a data processing perspective. Then we present an overview of specific requirements the characteristics pose to a sensor data processing system. Finally, in Section 4, we present an initial architecture for a sensor data processing pipeline.

2. Big data from sensors and machines

The size of the Digital Universe is rapidly growing. A study by IDC states that the amount of data in 2013 was 4,4 Zettabytes (10^{21}) and it is expected to grow to 44 ZB by 2020 – and more than 40 percent of all that data is expected to be machine generated[4]. Big data is often described via three V's - the data comes in at high **V**elocity, the **V**olume of data is huge, and it has a high level of **V**ariety[5]. Storing and processing such data has become feasible in the recent years largely by the release of Apache Hadoop[6], the use of which has been reported in many successful cases, such as detecting fraud in banking and insurance, predicting customer churn, and modelling consumer shopping behaviour for better targeting of advertising[7,8]. The use of big data analysis methods on social media data is also a popular topic[9,10].

Typically, sensor data is structured and modest in size, but the large number of sensors creates high volumes of data. Also, the real-time nature of sensor data dictates that the velocity of data is very significant, and often causes a need to react to the data quickly. Time criticality also means that queries and analysis of the data focus on a time range. All in all, the significance of sensor data is largely in discovering and reacting to trends, which can be obtained via statistical aggregation. However, the large volume and velocity in practice necessitate the use of big data processing methods.

Big data processing methods are divided into two categories. *Batch processing* can be used for very complex and computationally intensive analytics methods such machine learning and clustering, but it requires the data to be available at the beginning of analysis. *Stream processing*, on the other hand, processes data items as soon as they become available. It facilitates real-time action on the data, as well as filtering and aggregating it for efficient storage. Both methods are essential for sensor data, but stream processing is more critical in terms of a processing pipeline as the time criticality requires more efficient mechanisms for handling data input, such as a load balancing mechanism for distributing the data streams to the processing nodes.

Cloud platforms for IoT. The past couple of years have seen the birth of several commercial platforms for IoT services, covering service levels from Software-as-a-Service to Infrastructure-as-a-Service. Many established vendors such as Microsoft, HP, IBM and Oracle also target the IoT landscape. The commercial platforms provide cloud-oriented solutions and middleware for a wide range of IoT aspects such as device management, directory services, communication and data storage[11]. However, commercial platforms carry a price tag and may promote vendor lock-in. In the quickly changing IoT landscape, they may also suffer from delays in supporting new protocols, tools and data formats.

One alternative are open source IoT platforms. Many of them aim for a similar approach to the commercial ones, i.e., to cover the entire range of IoT aspects. Like their commercial counterparts, they often propose their own communication or middleware solutions, which requires tailoring the sensor software, and thus limits their adoption. Examples of open source IoT platforms are DeviceHive[12], Kaa[13], Nimbits[14], and OpenIoT[15].

One significant shortcoming in both commercial and existing open-source platforms is, however, that they are typically focused more at bringing sensors and other IoT devices to the cloud than offering the maximum potential for utilizing their data. Thus, there is a clear need to focus on the efficient data processing for very large pervasive sensor systems.

3. Data processing requirements in pervasive sensor systems

In pervasive sensor systems, the number of sensors may be huge, and the data rates high. For example, a Boeing 787 may generate half a terabyte of sensor data per flight[16]. Processing and storing a large volume of data from a large number of heterogeneous sources in real time poses significant challenges. Solving these challenges efficiently mandates a system that supports multiple data sources, protocols and data formats. The system also needs high horizontal scalability in data input, preprocessing, streaming analysis, and storage. The following sections focus on the specific requirements for sensor data input, preprocessing, and storage, which we find the most crucial components in enabling high-velocity, high-volume sensor data handling.

Data input. Obtaining sensor data is typically done via either a publish/subscribe messaging, direct polling from the sensor, or via a centralized blackboard for posting and reading values. Each approach has its benefits, but the point is that a data processing pipeline should facilitate all three methods. This dictates that the pipeline needs a configurable protocol front-end for a wide range of data collection mechanisms and communication protocols.

As a whole, the protocol landscape for sensor systems is fragmented. On the physical layer there are various wired and wireless options that make use of a host of competing link layer protocols that in turn can carry a variety of transport protocols. From a data storage and processing point of view, however, the relevant protocols are where the data messages are handled, which is the the session or communication layer. The most important communication layer architecture style is the representational state transfer (REST) model, which is popular also in machine-to-machine communications, such as in OneM2M[17]. However, communication in sensor systems is often limited by the constrained processing capabilities and limited energy storage in sensor devices. Thus, several IoT specific communications protocols have been developed that consider both processing capability and energy consumption. In our experience, the most important ones are the REST-based CoAP (Constrained Application Protocol)[18], the publish/subscribe-style MQTT (Message Queue Telemetry Transport)[19], the message-oriented, XML-based XMPP (extensible messaging and presence protocol)[20], and the open standard AMQP (Advanced Message Queuing Protocol)[21] which supports both point-to-point and publish/subscribe communication. It is, however, likely that new protocols emerge with the advent of new IoT sensor and communication technologies, mandating that a sensor data processing system needs to have very flexible protocol and data format support.

Stream processing. Pervasive sensor systems have real-time requirements that require the data processing pipeline to be able to react to ingress data as it arrives. This may include identifying values that are outside a given control range, calculating statistics of the data, and running analysis such as clustering, regression, classification, and machine learning. The analysis and related decision-making are likely to also require real-time access to auxiliary data such as other sensors, historical data, learning models, control ranges, or classification parameters. The objective of the process, often called *complex event processing*, is to gain actionable information that cannot be extracted from any of the sources alone[22]. Stream processing is also very important in order to aggregate and preprocess the large volume of data for storage. Preprocessing may require handling gaps or inconsistencies in the incoming data, and discarding erroneous data.

Data storage. Another big question is the storage of all the data generated by potentially millions of sensors. Given that sensor data is usually structured, using a database is a natural option, with in-built data management, indexing, caching and querying. However, the database needs to be highly performant, given the large datasets and high incoming data rates - and enable the use of different data formats. The technology of choice is thus often NoSQL, which addresses both the difficulties faced by traditional relational databases with large datasets, and new requirements set by cloud environments, such as scalability, elasticity, fault tolerance, and availability[23].

From sensor data point of view, columnar databases are an important subset of NoSQL that enable efficient fetching of a limited set of attributes from a dataset, because they store the data as a columns instead of rows[24]. Another interesting storage technology are time series databases, which address specific issues related to time series that are difficult in relational databases, such as queries for historical data and time zone conversions. Time series databases also offer data transformations for time series, as well as automatic calculation of basic statistics.

4. Initial structure of the processing architecture for sensor data

Our objective is to focus on the maximal effectiveness of processing sensor data, while avoiding an attempt to build a one-size-fits-all solution. Rather, we aim to enable the use of various optimal combinations of tools for different needs. This requirement for flexibility, combined with the aforementioned issues with commercial frameworks, naturally lends to using open source tools. The next paragraphs discuss various tools that we have chosen as a starting point for a flexible sensor data pipeline, focusing on data input, preprocessing, and storage. Naturally, we anticipate the following toolset to change over time as both the big data and IoT worlds are evolving rapidly. In any case, scalability and load balancing are key requirements in selecting each component in order to handle potentially millions of sensors.

Data input. Drawing from the findings above, three criteria for data input are above others: flexible protocol support, ability to provide data preprocessing, and good database integration. Based on these criteria, we have chosen three candidates for messaging and data input. The first one, Apache Flume[25], focuses on a flexible architecture, enabling the use of a variety of data sources and sinks. Flume uses an agent model where events from a source are queued until consumed. The second tool is Fluentd[26], which has a similar philosophy to Flume in that it supports a wide variety of input and output sources and formats by user-definable plugins. Fluentd also offers incoming message filtering, buffering, and routing. The final candidate are messaging frameworks such as ZeroMQ, RabbitMQ, NSQ, and Netty, which are optimized for high performance message handling. It should be noted, however, that no single tool is likely to cover all data input needs, and tools likely have to be picked and mixed on a case by case basis.

Preprocessing. Data aggregation is a very common preprocessing procedure for sensor data, due to the large volume and monotony of the data[27]. More sophisticated stream processing needs, on the other hand, vary greatly by application. We identify four tools as candidates for performing stream processing in a sensor data pipeline. Apache Samza[28] claims a simple API, fault tolerance, scalability, and support for various messaging and execution frameworks. Apache Storm[29], on the other hand, advertises integration with any message queuing system and any database. It has wide programming language support, with many reported uses, including real time analytics and machine learning. The third tool is Apache Flink[30], a big data analysis framework with a streaming API that is claimed to provide high throughput and low latency, with builtin support for RabbitMQ, Flume, and ZeroMQ, as well as user defined sources. Finally, Spark Streaming[31] is a part of the popular Spark big data framework, with extensive messaging support and well reported use cases. However, unlike the other frameworks, it processes data in micro batches rather than as individual items.

Data storage. Performance and scalability requirements shift the choice of data storage towards column-oriented NoSQL databases. We identify the following three databases as especially useful for sensor data. The first one is Apache HBase[32], which is one of the most popular columnar databases around. It offers real time access to very large tables up to billions of rows and millions of columns, with time-series support available via KairosDB[33]. Our second choice is Apache Cassandra[34], which is also widely used. Cassandra focuses on high performance, linear scalability and high availability. Like Hbase, Cassandra can be augmented with time-series operation. The final candidate, InfluxDB[35], is highly interesting in that it is a time-series database out of the box, and thus has built-in time-centric operators, and support for automatic raw data disposal to save space.

Based on the previous chapters, an overall structure and component functionality of a sensor data pipeline is presented in Figure 1. Our aim is to construct a sensor data processing pipeline whose components will scale to cover very large numbers of sensors and high volumes of data, while targeting all relevant parts of sensor data processing. In this paper, we cover data input, preprocessing and storage, which we consider the most critical parts for high-throughput, high-volume data. However, we acknowledge the importance of the steps that follow; the data needs to be made actionable via analysis. Therefore, the components for the input, preprocessing and storage need to be selected with further emphasis on their compatibility towards distributed data processing and data analytics tools, which will bring out the real value of the sensor data.

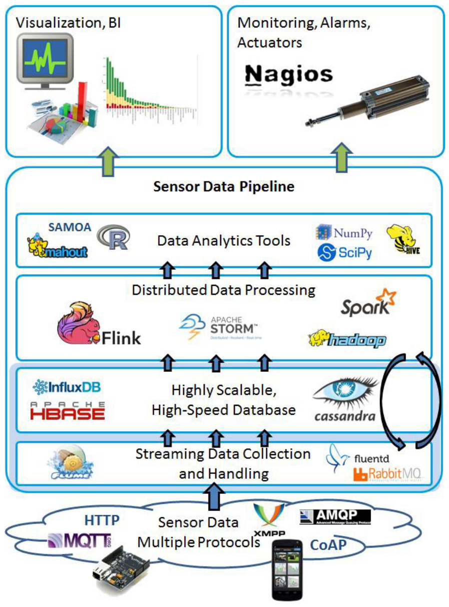

Figure 1. A high-level overview of the sensor data pipeline and its components.

In Figure 1, the bottom layer is the data collection stage that will consume data from a wide range of sensors, with support for at least HTTP/REST, MQTT, CoAP, AMQP and XMPP. Data preprocessing is done in the collection tools, e.g., Flume, or by piping the data to a stream processing tool such as Flink, indicated by a curved arrow in Figure 1. The next stage is storage, which will be done in a columnar database with support for time series operation. After that follows distributed data processing, which is largely used for aggregating and classifying large datasets. While out of scope for this paper, we anticipate at least the Apache tools Flink, Storm, Spark, and Hadoop to be on the list of supported frameworks. The tools cover both batch and streaming analysis. Finally, dedicated data analytics tools may optionally be used in addition to distributed data processing tools for complex data analysis. The types of tools we intend to support include general analytics, machine learning and complex big data querying.

After the pipeline has done its job, the final step is the extraction of business case or application domain specific insights and value, or create actions based on the analysis methods and processing of the data that has been harnessed for use by the pipeline. While essential to getting value from the data, these stages are well covered by existing tools and thus outside the scope of the data management pipeline presented in this paper.

5. Summary

In this paper, we have presented a high-level structure for the components and functionality of a sensor data management pipeline for large-scale pervasive environments of interconnected embedded devices and cloud-enabled back-ends. The main components of the pipeline and its technical requirements are based on the current state of sensor and IoT technology, and characteristics of sensor data. The structure of the pipeline was specifically designed from the point of view of realistic sensor communications and large-scale machine-generated data, for which the application of analytics and data management in the scope of current big data solutions is still a field with a wealth of untapped potential.

As the key role and function of the pipeline is collecting, preprocessing and storing sensor data, the actual considerations related to data science and analytics are left as the subject matter of another paper. It should be noted, however, that due to the choices made in the design of the pipeline, the integration of various big data processing frameworks should be, in most cases, a trivial matter.

In the next phase of the work, we will prototype the pipeline with a set of selected technologies, as presented in this paper. Tests and trials will be run against the implemented pipeline component chain with realistic large-scale sensor data that will be generated by IoT devices and sensors, both real and simulated.

Acknowledgements

Much of the research presented in this paper has been conducted under DIGILE's Need for Speed (N4S)[36] -program funded by Tekes, to which organizations the authors wish to express their gratitude.

References

1. Gartner: Internet of Things has reached hype peak - Network World n.d. http://www.networkworld.com/article/2464007/cloud-computing/gartner-internet-of-things-has-reached-hype-peak.html (accessed April 21, 2015).
2. Cisco Visualization — The Internet of Things n.d. http://share.cisco.com/internet-of-things.html (accessed February 6, 2015).
3. Xiao F, Zhang C, Han Z. Big Data in Ubiquitous Wireless Sensor Networks. *Int J Distrib Sens Netw* 2014;**2014**:2.
4. Data Growth, Business Opportunities, and the IT Imperatives —Rich Data and the Increasing Value of the Internet of Things n.d. http://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm (accessed February 10, 2015).
5. Deja VVVu: Others Claiming Gartner's Construct for Big Data - Doug Laney n.d. http://blogs.gartner.com/doug-laney/deja-vvvue-others-claiming-gartners-volume-velocity-variety-construct-for-big-data/ (accessed April 24, 2015).
6. The history of Hadoop: From 4 nodes to the future of data — Tech News and Analysis n.d. https://gigaom.com/2013/03/04/the-history-of-hadoop-from-4-nodes-to-the-future-of-data/1/ (accessed February 10, 2015).
7. Provost F, Fawcett T. Data Science and its Relationship to Big Data and Data-Driven Decision Making. *Big Data* 2013;**1**:51–9.
8. Baesens B. *Analytics in a Big Data World: The Essential Guide to Data Science and its Applications.* 1st ed. Wiley; 2014.
9. Russell MA. *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More.* O'Reilly Media, Inc.; 2013.
10. Sumbaly R, Kreps J, Shah S. The big data ecosystem at LinkedIn. Proc. 2013 ACM SIGMOD Int. Conf. Manag. Data, New York, New York, USA: ACM; 2013, p. 1125–34.
11. Mineraud J, Mazhelis O, Su X, Tarkoma S. A gap analysis of Internet-of-Things platforms. ArXiv Prepr ArXiv150201181 2015.
12. DeviceHive - M2M, Machine-to-Machine Communication Framework n.d. http://devicehive.com/ (accessed March 6, 2015).
13. Kaa Open Source IoT Platform n.d. http://www.kaaproject.org/ (accessed March 6, 2015).
14. Nimbits website n.d. http://www.nimbits.com/index.jsp (accessed March 6, 2015).
15. OpenIotOrg/openiot. GitHub n.d. https://github.com/OpenIotOrg/openiot (accessed March 6, 2015).
16. Boeing 787s to create half a terabyte of data per flight, says Virgin Atlantic. Computerworld UK n.d. http://www.computerworlduk.com/news/infrastructure/3433595/boeing-787s-to-create-half-a-terabyte-of-data-per-flight-says-virgin-atlantic/ (accessed March 6, 2015).
17. oneM2M - Leadership Team n.d. http://www.onem2m.org/ (accessed April 20, 2015).
18. CoAP — Constrained Application Protocol — Specification n.d. http://coap.technology/spec.html (accessed March 6, 2015).
19. MQTT n.d. http://mqtt.org/ (accessed March 6, 2015).
20. The XMPP Standards Foundation n.d. http://xmpp.org/ (accessed March 6, 2015).
21. AMQP n.d. http://www.amqp.org/ (accessed March 6, 2015).
22. Wu E, Diao Y, Rizvi S. High-performance Complex Event Processing over Streams. Proc. 2006 ACM SIGMOD Int. Conf. Manag. Data, New York, NY, USA: ACM; 2006, p. 407–18.
23. Grolinger K, Higashino WA, Tiwari A, Capretz MA. Data management in cloud environments: NoSQL and NewSQL data stores. J Cloud Comput Adv Syst Appl 2013;2:22.
24. Abadi DJ, Boncz PB, Harizopoulos S. Column-oriented database systems. Proc VLDB Endow 2009;2:1664–5.
25. Flume 1.5.2 User Guide — Apache Flume n.d. http://flume.apache.org/FlumeUserGuide.html (accessed March 19, 2015).
26. What is Fluentd? — Fluentd n.d. http://www.fluentd.org/architecture (accessed March 19, 2015).
27. Krishnamachari B, Estrin D, Wicker S. The impact of data aggregation in wireless sensor networks. Distrib. Comput. Syst. Workshop 2002 Proc. 22nd Int. Conf. On, 2002, p. 575–8.
28. Samza n.d. http://samza.apache.org/ (accessed March 19, 2015).
29. Storm, distributed and fault-tolerant realtime computation n.d. https://storm.apache.org/ (accessed March 19, 2015).
30. Apache Flink: Scalable batch and stream processing n.d. https://flink.apache.org/ (accessed March 19, 2015).
31. Spark Streaming — Apache Spark n.d. https://spark.apache.org/streaming/ (accessed March 19, 2015).
32. HBase – Apache HBase^{TM} Home n.d. http://hbase.apache.org/ (accessed March 19, 2015).
33. KairosDB n.d. https://kairosdb.github.io/ (accessed April 17, 2015).
34. The Apache Cassandra Project n.d. http://cassandra.apache.org/ (accessed April 17, 2015).
35. InfluxDB - Open Source Time Series, Metrics, and Analytics Database n.d. http://influxdb.com/ (accessed April 17, 2015).
36. Digile N4S n.d. http://www.n4s.fi/en/ (accessed April 21, 2015).