Architecture for Internet of Things Analytical Ecosystem

Andrzej Ratkowski[1]

[1] Warsaw University of Technology,
Institute of Control and Computation Engineering
A.Ratkowski@elka.pw.edu.pl

Abstract. Data analysis is a crucial part of Internet of Things concept. Smart objects generate significant amount of data that should be processed in order to utilize full potential of IoT. The paper presents proposed architecture that enables creating data analysis ecosystem for Internet of Things (IoT). The purpose of that architecture is to separate data analysis problem from orchestration of smart objects. Another important benefit of the ecosystem is to share analysis logic for IoT without revealing analytical algorithms.

Keywords: Internet of Things, IoT, data analysis for IoT.

1 Introduction

Internet of Things (IoT) as a vision of a world-wide network of interconnected objects [1] is a very promising concept however the concept is still in an early stage of development. IoT was defined by Haller [2] as "*a world where physical objects are seamlessly integrated into the information network, and where the physical objects can become active participants in business processes. Services are available to interact with these 'smart objects' over the Internet, query and change their state and any information associated with them, taking into account security and privacy issues*". It is estimated, that quantity of IoT objects potentially interconnected via information network, will be 10 to 100 times larger than it is now in contemporary Internet of PCs and mobile devices [3]. Such population of devices will produce wide stream of data that should be analyzed and utilized to orchestrate objects behavior. Hence analysis should be performed in near-real time and data flow should be processed fast and efficiently. This is the motivation to elaborate specific architecture that will support data flow and analysis. Important feature of the proposed architecture is entity that possess knowledge on analytical algorithms should be able to share that knowledge without revealing algorithms itself.

adfa, p. 1, 2011.

2 State of the art

Data analysis for IoT domain is a vital research problem. It is believed that the key to real usability of IoT paradigm is ability to provide real time data from various different distributed sources to other machines, smart entities and people for a variety of services [4]. Data analysis area for IoT covers data gathering and exchange [5], analysis algorithms used for data abstraction [6], semantic representation and cognitive analysis [7], data privacy and security [8]. Finally data analysis result could be also useful in orchestration of smart objects [9].

Literature review shows that data analysis for IoT is a complex issue, hence analysis should be separated from other aspects. It would be also economic to reuse algorithms without repeating effort on algorithm elaboration by using algorithm sharing ecosystem similar to mobile application ecosystems like Google Play [10].

Due to the amount and heterogeneity of data generated by smart objects it is recommended to use Big Data technologies [11] that support processing of data that have volume, velocity and variety properties [12].

Reference architecture for IoT was elaborated by IoT-A project [13] however the architecture does not address detailed data analysis aspect.

3 Analytical Ecosystem Vision

3.1 IoT Ecosystem Use Case

Following example illustrates use case of IoT ecosystem. A home is equipped with smart home installation based on IoT principles. The installation consist of sensors (light, temperature, air-quality, human presence etc.) and actuators (light switches, heating, air-conditioning etc.). Installation uses also data gathered from home residents smartphones and social services (facebook etc.). All sensors and actuators are connected to a local hub that coordinates their actions. The local hub is a computer system that stores set of simple rules that control actuators on the basis of facts gathered from sensors. Example of rules: if a person is present in the room and temperature drops below 20 C then turn on heating; if a house is empty and temperature drops below 15 C then turn on heating; etc.

The problem is how to generate set of rules with a view to comfort of living, energy saving, security and so on. It can be assumed that there is a data analytic algorithm *shRuleGenerator* possibly based on heuristics, that can gather data received by sensors and generate such rules. The algorithm can infer the model of the environment (house) on the basis of received data. For example, heat capacity of each room – analyzing heating/warming rate; or personal heat perception – analyzing feedback on temperature feeling from residents smartphones.

Such algorithm should be the same for certain type of houses (for flats, detached houses), so every owner of such IoT smart home installation do not have to develop his

own algorithm but can use general one available in analytical ecosystem. Local hub collects historical data from sensors and sends them periodically to analytical ecosystem

Analytical ecosystem enables exchange of algorithms between consumers and suppliers. Analytical ecosystem provides consumer with CPU power and data storage for algorithms execution. Consumer does not know algorithm but is able to execute it on provided data. Algorithm supplier does not know data feed by consumer but is aware of usage of algorithms.

3.2 Requirements

Architecture and data model has to satisfy following functional and non-functional requirements of analytical ecosystem:

RQ1. All services of analytical ecosystem for consumer and supplier are available through APIs.
RQ2. Supplier is able to provide and register algorithms.
RQ3. Algorithm is expressed in a convenient way in language preferred by supplier.
RQ4. Algorithm can execute (call) other algorithms.
RQ5. Consumer is able to find desirable algorithm.
RQ6. Consumer is able to execute desirable algorithm on provided data
RQ7. Consumer knows semantics of algorithm but does not know algorithms itself.
RQ8. Supplier does not know data that are feed by consumer.
RQ9. Both consumer and supplier are able to get information about usage of algorithms (number of executions).
RQ10. Consumer is billed for number of processed records, storage usage and algorithm execution.
RQ11. Supplier is billed for CPU used by his algorithms.
RQ12. Supplier is rewarded for usage of his algorithms.
RQ13. Analytical ecosystem supports semantic transformation and normalization of data feed by consumer.
RQ14. Analytical ecosystem provides anonimisation and privacy security mechanisms.
RQ15. Architecture is scalable in order to serve changing stream of execution of algorithms and to store input data for algorithms.

4 Analytical Ecosystem Architecture

4.1 Functional Architecture

Fig. 1. depicts functional architecture of analytical ecosystem. Architecture consist of seven modules. Each module is characterized in next sections. Fulfillment of requirements is marked with (RQxx) mark.

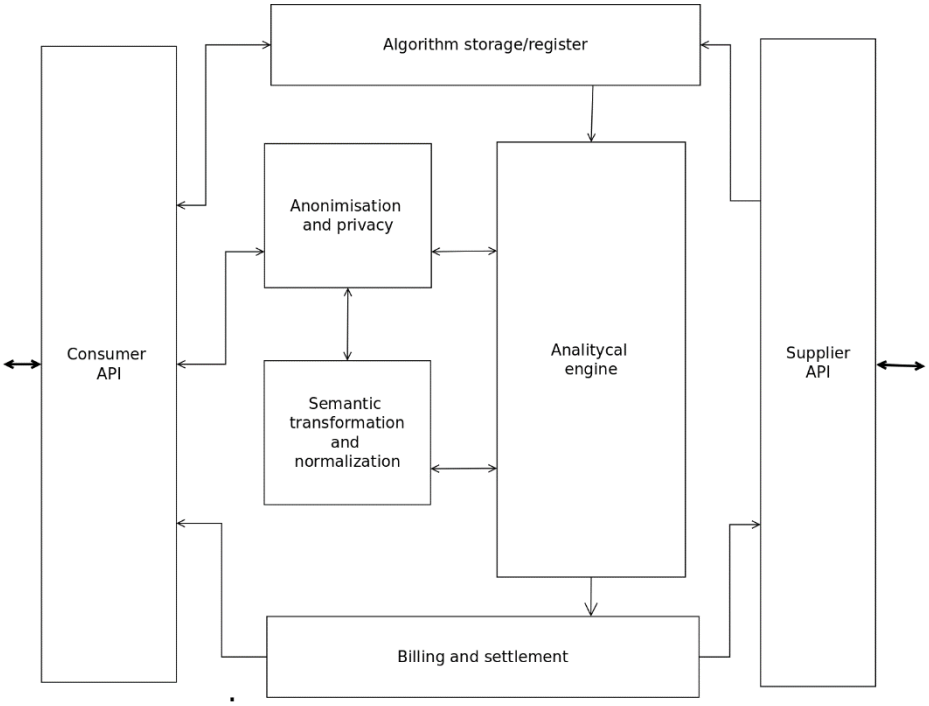

Fig. 1. Analytical ecosystem functional architecture

Consumer API.

Consumer API is a set of functions that provides consumer with all ecosystem services (RQ1). It consists of following function subsets:

- algorithm register querying and evaluation
- execute algorithm with provided data (RQ6)
- feed data and receive result of execution (RQ6)
- storage management
- billing and accounting information (RQ9)

Table 1. Main consumer API functions

function	specification
algorithmSet findAlgorithm(criteriaSet)	Queries register based on criteria, returns set of algorithms that meets supplied criteria
void scoreAlgorithm(algorithmId, score)	Scores algorithm
setId createDataSet()	Creates new data set identified by *setId*
void feedData(setId, entityCollection, size)	Feeds data set *setId* with collection of entities
resId executeAlgorithm(setId, algorithmId)	Executes algorithm *algorithmId* on data set *setId* , returns id of result set
resultSet getResultSet(resId)	Fetches result set *resId*
void releaseDataSet(setId)	Empty and free space occupied by data set *setId*
void releaseResultSet(resId)	Empty and free space occupied by result set *setId*
account getCurrentAccount()	Returns account balance and list of billing items.
billingData getBillingDetails(resId)	Returns detailed information on billing item.

Supplier API.

Supplier API provides supplier with all ecosystem services (RQ1). It consists of following function subsets:

- algorithm registering (RQ2)
- billing and accounting information (RQ9)
- algorithms usage statistics and CPU usage

Table 2. Main supplier API functions

function	specification
algorithmId registerAlgorithm(algorithmCode, algorithmDescription)	Registers algorithm and labels it with *algorithmDescription*
void unregisterAlgorithm(algorithmId)	Deletes algorithm *algoritmId* from register
void updateAlgorithm(algorithmId, algorithmCode, algorithmDescription)	Updates previously registered algorithm.
retCode testAlgorithm(algorithmId)	Tests syntax correctness of registered algorithm
account getCurrentAccount()	Returns current account state.
billingData getBillingDetails(algorithmId)	Returns billing data for specified algorithm.

Algorithm storage/register.

Algorithm storage/register is responsible for registering by supplier and sharing for consumer of analytical algorithms (RQ2, RQ5). Algorithms are defined by name, semantical description, price per execution and analytical algorithm. Algorithm itself can be expressed in language chosen from set of languages supported by ecosystem (RQ3). Among those languages we can find general purpose languages, (e.g. Python, Scala) and data analysis oriented (e.g. R, Pig Latin). Algorithm storage also provides data about usage popularity of algorithm, average CPU usage and reputation based on consumers evaluation scores.

Billing and accounting.

Billing and accounting module holds and provides data on mutual settlements between supplier, consumer and ecosystem itself. Billing and accounting module is feed with data from analytical engine on algorithms execution and resources used by every execution. On the basis of received data billing and accounting module calculates charges from consumer and rewards for supplier for executions of his algorithms(RQ10, RQ11, RQ12).

Semantic transformation/normalization.

The module is responsible for mapping data feed by consumer into semantically normalized form. For example smart home installation feed raw readouts from temperature sensors. Such readouts should be transformed into more synthetic information on temperature in each room. Semantic transformation/normalization module can provide such mapping. The mapping should use relatively simple algorithms in order to be fast and consume little CPU resources (RQ13).

Privacy and security.

Privacy and security module scrambles data that can be used to identify analyzed objects or contains sensible information (e.g. localization data). The module hashes such data and stores mapping of hashed data into primary data. Module is also responsible for decoding hashed result of analysis into primary data. Scrambling should be CPU effective (RQ14).

Analytical engine.

Analytical engine is core element of the ecosystem. It is responsible for executing algorithms and allocating CPU resources for execution. The analytical engine is based on MapReduce[14] programming model (RQ15). Analytical engine ensures isolation of

data between different consumers through multi-tenancy architectural pattern[15] . Internal structure of analytical engine is depicted on fig. 2.

Manager element is responsible for coordination of algorithm execution, requests scheduling and calculation of executions and resource utilization. *Interpreter* translates algorithm that was provided by supplier in desired language, to set of *map* and *reduce* functions. *Manager* After translation *MapReduce* component is requested by *Manager* to execute algorithm. *MapReduce* is also responsible for resources allocation.

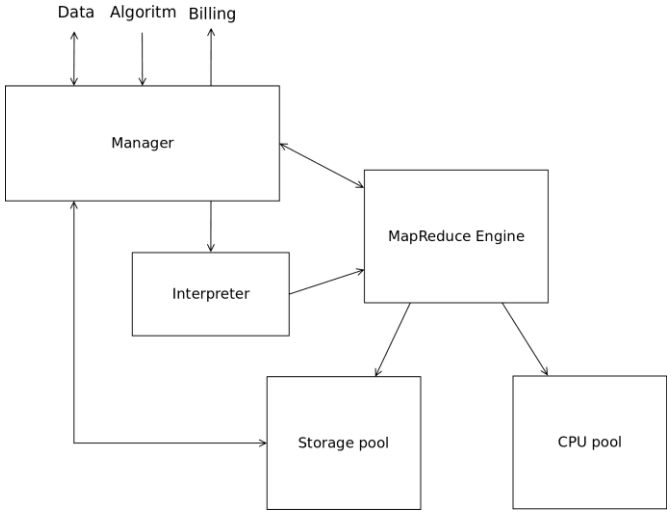

Fig. 2. Analytical engine internal structure

4.2 Data model

Fig. 3 presents data model of analytical ecosystem. Data model organizes information on algorithms (*Algorithm*). Property of algorithm entity is algorithm code expressed in certain language. Each algorithm is also characterized by input and output data structure (*DataFormat)*. All algorithms are referenced in directory (*AlgorithmDirectory*) that can be queried in order to find fitting algorithm.

Data model stores data on algorithms execution (*AlgorithmExecution*) as well as input and output data sets (*DataSet, ResultSet*) for each execution. Input and output data format has to be consistent with proper data formats associated with algorithm. Data model stores data on resources consumed by algorithm executions (*Resources*) and settlements between algorithm ecosystem, consumer and supplier (*Billing, BillingRecord*). Data model does not cover information on algorithm semantics. This aspect is handed over to supplier.

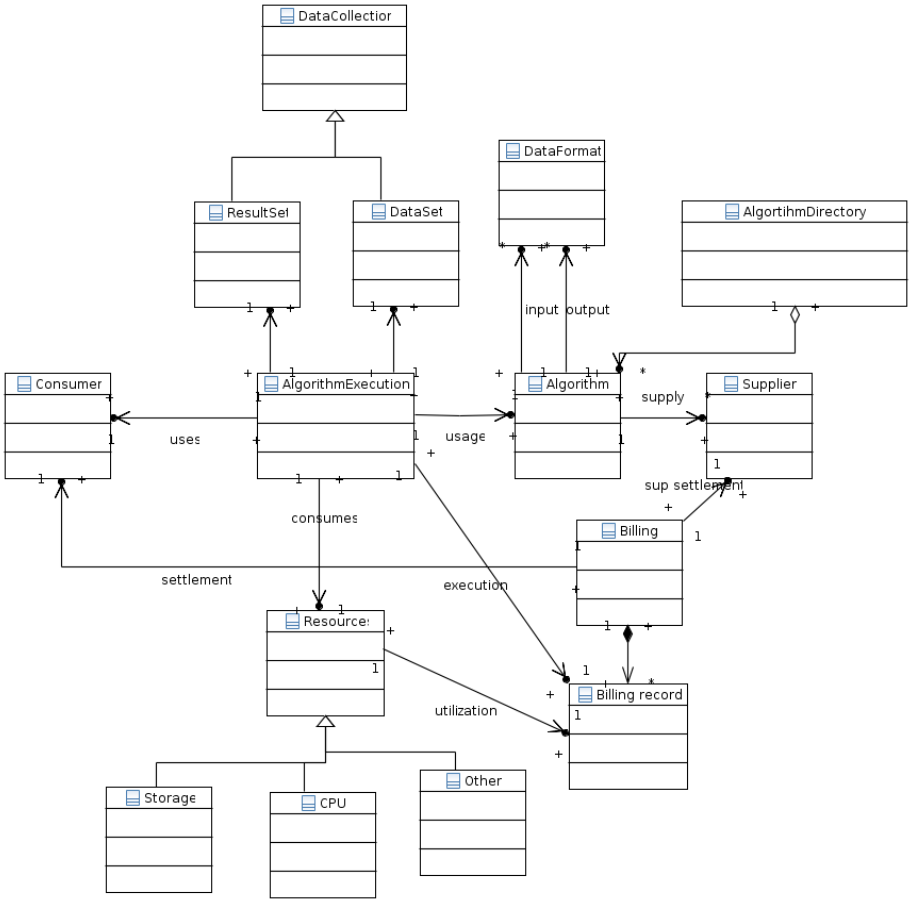

Fig. 3. Data model of analytical ecosystem for IoT

5 Summary and further work

Presented architecture fulfills requirements stated in section 3.2. Analytical ecosystem allows sharing of algorithms between supplier and customer without compromising security of data feed by customer and without revealing algorithm code. Such defined analytical ecosystem plays *trusted third party* role in interactions between algorithm supplier and customer.

Despite of the fact that proposed architecture is on certain maturity leve, following aspects require more detailed consideration:

- physical architecture that covers allocation of modules to physical resources (servers, storages etc.),
- interfaces of architectural modules,

- operational model of *Interpreter* module that will enable using different languages to express algorithms and to execute other algorithms (according to requirement RQ4),
- detailed design of consumer and supplier API including protocols and semantics of functions calling.

The next step of validation of ecosystem concept will be preparing working prototype of full functional ecosystem. The working prototype will be utilized to verify feasibility and usability of ecosystem.

References

1. Santucci G., Lange S. Internet of things in 2020 a Roadmap for the Future, joint EU-EPoSS workshop report (2008)
2. Haller S., The Things in the Internet of Things, Proceedings of Internet of Things Conference (2010)
3. Cicconi J. At&T Cisco IBSG et al. The Internet of Things Infographic (2012)
4. Aggarwal C.,Ashish N, Sheth A. The Internet of Things: A Survey from the Data-Centric Perspective. In:Managing and Mining SensorData. Springer (2013)
5. Abdelwahab, S., Hamdaoui, B., Guizani, M., & Rayes, A., Enabling smart cloud services through remote sensing: An internet of everything enabler. Internet of Things Journal, IEEE, 1(3), 276-288. (2014).
6. Ganz, F.; Puschmann, D.; Barnaghi, P.; Carrez, F., A Practical Evaluation of Information Processing and Abstraction Techniques for the Internet of Things, in Internet of Things Journal, IEEE , vol.2, no.4, pp.340-354, Aug. (2015)
7. Wu, Q., Ding, G., Xu, Y., Feng, S., Du, Z., Wang, J., & Long, K., Cognitive Internet of things: A new paradigm beyond connection. Internet of Things Journal, IEEE, 1(2), 129-143. (2014)
8. Keoh, S. L., Kumar, S. S., & Tschofenig, H., Securing the internet of things: A standardization perspective. Internet of Things Journal, IEEE, 1(3), 265-27 (2014)
9. Haller, S., & Magerkurth, C. (2011, March). The real-time enterprise: Iot-enabled business processes. In IETF IAB Workshop on Interconnecting Smart Objects with the Internet.
10. Google Inc. Play application store, https://play.google.com/ (access 2015)
11. Marz N., Warren J. Big Data: Principles and Best Practices of Scalable Realtime Data Systems (1st ed.). Manning Publications Co., Greenwich, CT, USA. (2015)
12. Russom, P. Big data analytics. TDWI Best Practices Report, Fourth Quarter (2011).
13. Bassi A. et al. Final Architectural Reference Model for the IoT v3.0 (IoT-A), www.iot-a.eu, (2011)
14. Dean, J., & Ghemawat, S. MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113. (2008)
15. Kwok, T., Nguyen, T., & Lam, L. (2008, July). A software as a service with multi-tenancy support for an electronic contract management application. In Services Computing,. SCC'08. IEEE International Conference on (Vol. 2, pp. 179-186). IEEE. (2008)