Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 52 (2015) 1022 – 1027

Procedia Computer Science

Workshop on Big Data and Data Mining Challenges on IoT and Pervasive Systems (BigD2M-2015)

servIoTicy and iServe: a Scalable Platform for Mining the IoT

Álvaro Villalba[a], Juan Luis Pérez[a], David Carrera[a], Carlos Pedrinaci[b], Luca Panziera[b]

[a]*Barcelona Supercomputing Center (BSC) - Universitat Politècnica de Catalunya (BarcelonaTECH), Jordi Girona 1-3, Barcelona, 08034, Spain*
[b]*Knowledge Media Institute - The Open University. Walton Hall, Milton Keynes, MK7 6AA, United Kingdom*

Abstract

In the last years, Internet of Things (IoT) and Big Data platforms are clearly converging in terms of technologies, problems and approaches. IoT ecosystems generate a vast amount of data that needs to be stored and processed, becoming a Big Data problem. In this paper we present a platform that is specifically designed for mining the information associated to the IoT, including both sensors data and meta-data. The platform is composed of two major components: servIoTicy for storing and processing data, and iServe for the publication and discovery of sensors meta-data. The former provides capabilities to ingest, transform on real time and query data generated by sensors; the latter provides capabilities to publish, discover and use sensors based on semantic information associated to them. Both components are clearly designed for scalability, as any IoT cloud deployment requires. Both servIoTicy and iServe are available as an open source projects.

Peer-review under responsibility of the Conference Program Chairs

Keywords: Internet of Things; IoT; Big Data; Analytics; Stream Processing; Semantic Search; Service Discovery

1. Introduction

In the last years, Internet of Things (IoT) and Big Data platforms are clearly converging in terms of technologies, problems and approaches. IoT ecosystems generate a vast amount of data that needs to be stored and processed, becoming a Big Data problem. IoT devices and sensors generate streams of data across a diversity of locations and protocols that in the end reach a central platform that is used to store and process it. Processing can be done in real time, with transformations and enrichment happening on-the-fly, but it can also happen after data is stored and organised in repositories. In the former case, stream processing technologies are required to operate on the data; in the latter analytics and queries are of common use.

At the same time, the IoT ecosystem growth forecast is that by 2020 more than 50 billion devices will be connected and accessible. Therefore, discovering devices will become also a Big Data challenge, and will require extremely scalable technologies to manage all the semantic data associated to them.

The above-mentioned situation implies that there is an increasing demand for advanced IoT data management and processing platforms. Such platforms require support for multiple protocols at the edge for extended connectivity with

* Corresponding author. Tel.: +34-93-405-42-81
E-mail address: alvaro.villalba@bsc.es

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2015.05.097

the objects, but also need to exhibit uniform internal data organisation and advanced data processing capabilities to fulfil the demands of the application and services that consume IoT data, as well as mechanisms to store and query semantic information associated to the devices.

Advanced streaming and analytics platforms such as servIoTicy and iServe are complex pieces of software that integrate a large set of components under the hood. They hide their complexity behind simple REST APIs and multi-protocol channels, but the reality is that their deployment and configuration is complex. The platform described by this paper is part of the developments of the COMPOSE[1] project, which aims to develop a more ambitious IoT platform, not only focused on the data management and processing part, but including other aspects such as security, discovery of objects, development tools and composition engines. The sources of the servIoTicy and iServe are freely available as an open source projects[1] on GitHub. The platform is also available for single node testing as a vagrant box, downloadable from a github repository[2].

The next sections of the paper are organised as follows: Section 2 introduces a set of abstractions defined in servIoTicy for managing data associated to objects; Section 3 introduces the general architecture and components of iServe; Section 4 describes how iServe has been adapted to index and locate IoT resources stored in servIoTicy; Finally, Section 5 goes through the related work.

2. servIoTicy: Stream Processing and Data Analytics

servIoTicy[3] is a state-of-the-art platform for hosting Internet of Things (IoT) workloads in the Cloud. It provides multi-tenant data stream processing capabilities, a REST API, data analytics, advanced queries and multi-protocol support in a combination of advanced data-centric services. ServIoTicy aims to provide a technological platform for easily creating services based on the Internet of Things (IoT), thus unleashing the full potential of an Internet of Services (IoS) based on the IoT. The main focus of servIoTicy is to provide a rich set of features to store and process data through it REST API, allowing objects, services and humans to access the information produced by the devices connected to the platform. servIoTicy allows for a real time processing of device-generated data, and enables for simple creation of data transformation pipelines using user generated logic. Unlike traditional service composition approaches, usually focused on addressing the problems of functional composition of existing services, one of the goals of the servIoTicy is to focus on data processing scalability.

The architecture of servIoTicy is composed of different elements. The Front-End of platform is a Web Tier that implements the REST API that sits at the core of servIoTicy. The API contains parts of the logic of the Service Objects and Data Processing Pipelines, related to authentication, data storage and data retrieval actions. The Stream Processing Topology is responsible for the execution of the code associated to Data Processing pipes as well as the forwarding of data across Service Objects and to external entities (e.g. external subscribers that want data forwarded on real-time using a push model on top of MQTT or STOMP). Finally, the data Back-End includes the Data Store that provides scalable, distributed and fault-tolerant properties to servIoTicy, and the Indexing Engine that provides search capabilities across sensors data using different criteria, like timestamps, string patterns or geo-location.

At the core of the servIoTicy runtime there is a novel technique to dynamically construct data stream processing topologies based on user-supplied codes. These topologies are built on-the-fly using a data subscription model defined by the applications that consume data. Each user-defined processing unit is called a Service Object, and each Service Object consumes input data streams and may produce output streams that others can consume. Data streams can originate in real-world devices or they can the outputs of Service Objects deployed in the platform.

Several abstractions are used in servIoTicy to embrace the different entities involved in the existence of IoT ecosystems.

- Web Object: Web Objects are physical objects sitting on the edge of the servIoTicy and capable of keeping for example HTTP-based bi-directional communications, such that the object will be able to both send data to

[1] servIoTicy: https://github.com/serviotic; iServe: https://github.com/kmi/iserve
[2] https://github.com/servioticy/servioticy-vagrant
[3] servIoTicy.com

the platform and receive activation requests and notifications. Not all such objects will support the same set of operations, but a minimum subset will have to be guaranteed to make them usable servIoTicy.

- Service Object: Service Objects are standard internal servIoTicy representations of Web Objects. This entity serves mainly for data management purposes and has a well-defined and closed API[4] that provides search capabilities across sensors data using different criteria, like timestamps, string patterns or geo-location. ServIoTicy, in an effort to embrace as many IoT transports as possible, allows Web Objects to interact with their representatives in the Platform (the Service Objects) using a set of well-known protocols: HTTP, STOMP[2] over TCP, STOMP over WebSockets[3], and MQTT[4] over TCP.
- Data Processing Pipeline: A Data Processing Pipeline is a data service and aggregation mechanism, which relies on the data processing and management back-end component to provide complex computations resulting from subscriptions to different Service Objects as data sources. This construct can support pseudo-real time data stream transformations, combined with queries concerning historical data. Data analytics code defined by the user may be provided as well. Just like a Service Object, this entity serves mainly for data management purposes and has a well-defined and closed API.
- Subscription: Data subscriptions are a mechanism in servIoTicy that allow Service Objects, Data Processing Pipelines and external data consumers to get data updates automatically and asynchronously forwarded for further processing.
- Sensor Update: Sensor Updates are the unit of data sent by a Web Object to its Service Object. It contains the different synchronously sensed values and a timestamp that is maintained all over the pipelines. A subscription or a query to a Service Object will get the data in this format.

3. iServe: Mining Sematic Meta-data

iServe, previously introduced in[5], is an open source platform[5] that unifies the publication, discovery, and use of Web Services (e.g., WSDL services), Web APIs (e.g., Twitter API, Flickr API, etc), and Things available on the Web. iServe is, to the best of our knowledge, the first platform to provide this level of support homogeneously across types of devices and software interfaces, providing a convenient one-stop-shop for the location and use of distributed software building blocks as necessary for creating novel Web and IoT applications.

iServe exploits and expands state of the art research and development from a number of fields. From a data acquisition perspective, the platform includes several import plugins providing the platform with the ability to ingest, process, and index interface descriptions in several formalisms including WSDL, SAWSDL, Swagger[6], OWL-S and WSMO-Lite to name a few[7]. Additionally, Machine Learning techniques are used to support the automated discovery of Web APIs over the Web by automatically identifying Web pages that provide technical documentation of these APIs[8]. Subsequently specific Web Mining techniques have been devised to automatically extract important features from these descriptions such as the functionality provided, the endpoints, etc[9]. The aforementioned processes are fed by a Web crawl, namely CommonCrawl[6], which is processed using a combination of Hadoop and Mahout jobs to support their efficient processing.

Once imported, all descriptions are homogenised into a common representation expressed in terms of an ontology called the Minimal Service Model (MSM)[5]. This ontology essentially provides a common ground for describing i) the overall structure of the interface exposed by a Web API or a *Thing* including aspects like the operations or resources exposed, ii) human-oriented information such as a text-based descriptions about the component at hand, and iii) machine-oriented descriptions of the components including semantic annotations about the type of component, the functionality offered, or the semantics of the data manipulated.

Every Web Service, Web API, and Thing are homogeneously described as *services* expressed in terms of MSM. iServe exposes services publicly following the Linked Data principles, which essentially dictate that every piece of data should be given an HTTP URI which, when looked up, should offer useful information using standards like RDF and SPARQL[10]. Additionally, data should be linked to other relevant resources therefore allowing humans and computers

[4] docs.serviotic y.com
[5] http://iserve.kmi.open.ac.uk
[6] http://commoncrawl.org

to discover additional information. In a nutshell, adopting Linked Data principles enables both humans and machines to seamlessly retrieve and process the descriptions of the software components indexed by iServe in a convenient manner.

After import, iServe tries to enrich the data with external information. Currently, for instance, iServe attempts to identify the provider of the service, and tries to interlink it with two large Linked Data datasets, i.e., DBPedia and Yago, so that we can leverage external information about the provider to assess, for instance, how trust worthy or stable a service might be. Similarly, we enrich the descriptions with annotations to Schema.org Actions, providing in this manner a coarse-grained classification for the registered services in terms of a widely used schema.

The descriptions gathered by iServe represent the common grounds upon which the platform offers advanced support for the discovery and use of the registered components. In particular, iServe implements a range of search and retrieval facilities allowing users to filter and rank all registered services according to several criteria. First and foremost the platform relies on Lucene for supporting text-based search over the entities registered. Additionally, the platform offers state-of-the-art semantic discovery by reasoning over semantic annotations whenever they exist may they be general classifications (e.g., with Schema.org) or semantic annotations of inputs and outputs. These discovery features are enhanced with further facilities ranking the results by taking into account other aspects such as the popularity of a given component or how active the community behind it is.

The entire functionality of iServe is offered both through a human-oriented Web interface based on Elda[7], as well as through a RESTful API offering a convenient integration point to external applications.

4. Indexing the IoT ecosystem: iServe on servIoTicy

Thus far, most data acquisition effort in iServe has been devoted to locating and indexing services and public Web APIs. The increasing popularity of the Internet of Things is, however, at the origin of an outstanding proliferation of Internet accessible sensors and actuators. Although, this newer kind of distributed components typically relies on different protocols and technologies (e.g., ZigBee), most often in an attempt to simplify and promote their use, they end up being exposed on the Web either directly or through gateways and cloud services such as Xively and ServIoTicy. In the light of this proliferation of Things on the Web, we have also started providing support for them.

In a first instance we are targeting cloud services for they typically expose large numbers of sensors and actuators. In these cases the data and actuations offered by Things are all exposed through one common Web API, which is indeed dependent on the platform at hand. Although iServe already provides advanced support for Web APIs, it is worth noting that the handling of IoT platforms should be approached differently. In particular, in the case of a "traditional" Web API, e.g., Flickr's API, the Web API as a whole can be understood as one service providing a more or less large set of operations or resources that are closely related, e.g., all methods are about the same collection of images, and the data is all exposed by the same provider. In the case of cloud platforms for the Internet of Things, each and everyone of the Things exposed is best considered as a single stand alone entity. After all, a temperature sensor in London has hardly any relationship with a traffic sensor in Rome, and they should therefore be handled as distinct and separately manageable entities. In fact, for the case of cloud services for the Internet of Things, perhaps the main and often unique relationship is that they are exposed through the same platform using a common API scheme.

Following this approach, we have developed a targeted crawler and importer for ServIoTicy, see 1. In a nutshell the crawler uses a dedicated ServIoTicy API to list all Things and obtain their descriptions. The ServIoTicy descriptions are then used to generate a corresponding Thing-centric Swagger description, whereby every stream and actuation is a resource offering the typical CRUD methods all correctly grounded into ServIoTicy's Web API. This approach enables users to have fine-grained (stream level) interactions with Things. Additionally, we generate at this stage basic metadata such as names, and descriptions for methods and resources. After this transformation, we import all Swagger descriptions in iServe which stores and indexes them and subsequently transforms them into MSM for ulterior semantic enrichment and exploitation within advanced discovery algorithms.

This integration, on the one hand, provides ServIoTicy with high-level search functionality over the Things it hosts. On the other hand, it also enables 3rd party application developers to rely on iServe for discovering and using Web Services, Web APIs, and Things homogeneously. For instance, users can directly use iServe's interactive documentation

[7] https://github.com/epimorphics/elda

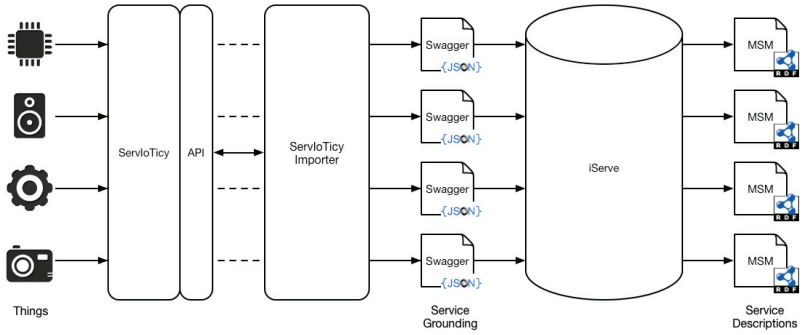

Fig. 1. Targeted crawling and indexing of ServIoTicy.

for the services it indexes in order to test Things from ServIoTicy or they can directly use these Things within editing environments such as NodeRed through a dedicated generic node[8].

The ServIoTicy importer (see Figure 1) interacts with ServIoTicy APIs in order to build a Swagger description for each service, which provides access to functionalities of a specific thing. The process of Swagger creation is based on three steps. The first one is the identification of the things that are registered in ServIoTicy, then the skeleton of a Swagger description is built for each of them. The second step is the extraction of the thing metadata, such as name and descriptions in natural language. This information becomes the metadata of each Swagger service, which will be exploited to perform free text search over things. The third step focuses on stream and actuation modelling. Each stream and actuation is transformed in an Web API described in terms of the Swagger specification. A Swagger API represents a main RESTful resource of the service. We choose to list streams and actuations as static service resources for two objectives: (i) to allow developers to browse the complete list of streams and actuations by accessing Swagger descriptions and avoiding additional interaction with ServIoTicy; (ii) to enable discovery of streams and actuation as discrete reusable operations. Stream and actuation names are used as Swagger APIs descriptions in order to enable free text search on operations.

After the import, Swagger descriptions are stored in iServe and transformed into semantic descriptions according to the Minimal Service Model (MSM). Each Swagger description is mapped to a MSM service, each operation of Swagger API is converted to MSM operations and each parameter becomes a MSM message part. The resulting MSM descriptions enables advanced semantic discovery of things as services.

5. Related Work

Data Centric view of the IoT is not something new for servIoTicy as it was widely covered in the survey presented in[11]. What servIoTicy uniquely provides is an open source solution that challenges the features of commercial solutions such as Xively[12] and Evrythng[13], while extending their capabilities with the ability to inject user-defined code into its stream processing runtime. There are other open source platforms for IoT in the market similar to servIoTicy, but they are focused on other aspects of the Internet of Things and do not provide the capability to deploy user codes to be used for real time data processing. Examples of projects in this domain are DeviceHive[14], Devicehub.net[15], IoT Toolkit[16], Mango[17], Nimbits[18], OpenRemote[19], SiteWhere[20] and ThingSpeak[21].

The discovery of reusable services has been subject of much research in Service-Oriented Computing (SOC). Although it was not particularly successful, UDDI[22] is perhaps the best-known effort to support the publication and discovery of services on the Web. A major reason for the lack of success of UDDI was the fact that, although these registries are relatively complex, they do not support expressive queries, limiting their usefulness[23]. Semantic Web Services researchers have long tried to overcome the limitations of Web service descriptions by enriching them with semantic annotations, see[24,25,26].

[8] See https://github.com/kmi/node-red-contrib-swagger

The vast majority of the service discovery initiatives are predicated upon the availability of WSDL Web services, and these have turned out not to be prevalent on the Web[27]. The world of services on the Web has recently seen a major evolution with the advent and proliferation of Web APIs and RESTful services[28]. Little progress has, however, been done towards better supporting their discovery and use the main example being ProgrammableWeb. This platform, however, although valuable is essentially based on manual contributions by users and provides fairly limited search capabilities that prevent its systematic use for developing distributed applications.

The results and expertise obtained through iServe on SOC research allow us to overcome limits existing solutions for discovery of Things. Commercial IoT platform, such as Xilvely[12], and domain-specific sensors catalogs, such as the Esonet data portal[29] implement discovery of Things exclusively through text-based search. State-of-the-art discovery approaches based on Semantic Web technologies[30,31,32] require manual annotation of sensor metadata and do not support discovery of external services and Web APIs.

References

1. Mandler, B., Antonelli, F., Kleinfeld, R., Pedrinaci, C., Carrera, D., Gugliotta, A., et al. Compose – a journey from the internet of things to the internet of services. *2013 27th International Conference on Advanced Information Networking and Applications Workshops* 2013; **0**:1217–1222. doi:http://doi.ieeecomputersociety.org/10.1109/WAINA.2013.116.
2. Stomp. 2015. URL: `http://stomp.github.io`.
3. The webSocket API. 2015. URL: `http://dev.w3.org/html5/websockets`.
4. MQTT. 2015. URL: `http://mqtt.org`.
5. Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecký, J., Domingue, J.. iServe: a Linked Services Publishing Platform. In: *Proceedings of Ontology Repositories and Editors for the Semantic Web at 7th ESWC*. 2010, .
6. Swagger API specification. 2015. URL: `https://github.com/swagger-api/swagger-spec`.
7. Pedrinaci, C., Sheth, A., Domingue, J.. Semantic Web Services. In: *Handbook of Semantic Web Technologies*. Springer; 2011, .
8. Lin, C., He, Y., Pedrinaci, C., Domingue, J.. Feature lda: a supervised topic model for automatic detection of web api documentations from the web. In: *The 11th International Semantic Web Conference (ISWC)*. Boston, USA; 2012, .
9. Ly, P.A., Pedrinaci, C., Domingue, J.. Automated information extraction from web APIs documentation. In: *WISE'12: Proceedings of the 13th international conference on Web Information Systems Engineering*. Springer-Verlag; 2012, p. 497–511.
10. Prud'hommeaux, E., Seaborne, A.. SPARQL Query Language for RDF. Tech. Rep.; W3C; 2008. Available at `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`.
11. Qin, Y., Sheng, Q.Z., Falkner, N.J.G., Dustdar, S., Wang, H., Vasilakos, A.V.. When things matter: A data-centric view of the internet of things. *CoRR* 2014;**abs/1407.2704**. URL: `http://arxiv.org/abs/1407.2704`.
12. Xively. 2015. URL: `xively.com`.
13. evrythng. 2015. URL: `evrythng.com`.
14. DeviceHive. 2015. URL: `http://www.devicehive.com/`.
15. Devicehub. 2015. URL: `http://devicehub.net`.
16. IoT Toolkit. 2015. URL: `iot-toolkit.com`.
17. Mango. 2015. URL: `http://forum.infiniteautomation.com/`.
18. Nimbits. 2015. URL: `http://www.nimbits.com`.
19. OpenRemote. 2015. URL: `http://www.openremote.com`.
20. SiteWhere. 2015. URL: `http://www.sitewhere.org`.
21. ThingSpeak. 2015. URL: `https://thingspeak.com/`.
22. Hately, L.C.A., von Riegen, C., Rogers, T.. UDDI Specification Version 3.0.2. 2004.
23. Pilioura, T., Tsalgatidou, A.. Unified publication and discovery of semantic web services. *ACM Transactions on the Web (TWEB)* 2009; **3**(3):11.
24. Kawamura, T., De Blasio, J.A., Hasegawa, T., Paolucci, M., Sycara, K.. Public deployment of semantic service matchmaker with uddi business registry. In: *The Semantic Web–ISWC 2004*. Springer; 2004, p. 752–766.
25. Kiefer, C., Bernstein, A.. *The creation and evaluation of isparql strategies for matchmaking*. Springer; 2008.
26. Klusch, M., Kapahnke, P., Zinnikus, I.. Sawsdl-mx2: A machine-learning approach for integrating semantic web service matchmaking variants. In: *Web Services, 2009. ICWS 2009. IEEE International Conference on*. IEEE; 2009, p. 335–342.
27. Pedrinaci, C., Domingue, J.. Toward the next wave of services: Linked Services for the Web of data. *Journal of Universal Computer Science* 2010;**16**(13):1694–1719.
28. Richardson, L., Ruby, S.. *RESTful web services*. " O'Reilly Media, Inc."; 2008.
29. Esonet data portal. 2015. URL: `http://dataportals.pangaea.de/esonet/`.
30. Pschorr, J., Henson, C., Patni, H., Sheth, A.. Sensor discovery on linked data. Tech. Rep.; Kno.e.sis Center; 2010. URL: `http://knoesis.org/library/resource.php?id=851`.
31. Le-Phuoc, D., Quoc, H.N.M., Parreira, J.X., Hauswirth, M.. The linked sensor middleware–connecting the real world and the semantic web. *Proceedings of the Semantic Web Challenge* 2011;.
32. Perera, C., Zaslavsky, A., Christen, P., Compton, M., Georgakopoulos, D.. Context-aware sensor search, selection and ranking model for internet of things middleware. In: *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*; vol. 1. IEEE; 2013, p. 314–322.