From IoT Big Data to IoT Big Services

Amir Taherkordi, Frank Eliassen, and Geir Horn
Department of Informatics
University of Oslo, Norway
{amirhost, frank, geir.horn}@ifi.uio.no

ABSTRACT

The large-scale deployments of Internet of Things (IoT) systems have introduced several new challenges in terms of processing their data. The massive amount of IoT-generated data requires design solutions to speed up data processing, scale up with the data volume and improve data adaptability and extensibility. Beyond existing techniques for IoT data collection, filtering, and analytics, innovative *service computing* technologies are required for provisioning data-centric and scalable IoT services. This paper presents a service-oriented design model and framework for realizing scalable and efficient acquisition, processing and integration of data-centric IoT services. In this approach, data-centric IoT services are organized in a service integrating tree structure, adhering to the architecture of many large-scale IoT systems, including recent fog-based IoT computing models. A service node in the tree is called a *Big Service* and acts as an integrator, collecting data from lower level Big Services, processing them, and delivering the result to higher level IoT Big Services. The service tree thereby encapsulates required data processing functions in a hierarchical manner in order to achieve scalable and real-time data collection and processing. We have implemented the IoT Big Services framework leveraging a popular cloud-based service and data platform called Firebase, and evaluated its performance in terms of real-time requirements.

CCS Concepts

•**Networks** → *Cloud computing;* •**Software and its engineering** → *Data flow architectures; Abstraction, modeling and modularity;*

Keywords

Internet of Things; Big Services; Big Data

1. INTRODUCTION

Smart devices have facilitated the pervasive interaction and cooperation with each other through unique addressing schemes known as the Internet of Things (IoT). Such devices are deployed in different environments to collect various kinds of data, such as environmental, geographical, and logistics data. In a broader sense, connecting a multitude of sensor-equipped physical objects to the Internet generates a new *big data ecosystem*. The collection and processing of data generated by such a huge number of devices lead to unprecedented challenges in mining and processing the generated data [2], such as heterogeneity and variety of data, real-time data processing, and discovering the value of data produced by multi-purpose and shared smart devices [6]. Beyond these, sensor data is often noisy and redundant, and it has strong correlations with the IoT physical context [2].

With the exponential amount of data generated by IoT systems, innovative service computing technologies are required for provisioning data-centric, scalable and composable services—Big Services. A Big Service is defined as a managed integration of a massive, complicated series of services centered on big data [29]. Its objectives are to speed up data processing, scale up with data volume, improve the adaptability and extensibility over data diversity, and facilitate better understanding and manipulation of the data. Big Services should be designed and constructed based on the attributes of the target big data system, such as its data acquisition model, the desired correlation between data sets, and analysis techniques used. For example, in a smart city, a Big Service could be a customer-oriented service composed of various data-centric city services, e.g., an urban traffic data system responding quickly to traffic events gathered from a wide range of resources such as vehicles and traffic surveillance cameras. Large-scale IoT systems, as the dominant generator of big data sets, deserve special attention in this context as their aforementioned unique characteristics imply the need for a data-centric service modeling approach that can efficiently achieve the premise of Big Services for IoT big data.

Big data research efforts have so far been devoted to addressing inherent data processing challenges, such as data acquisition, storage, management and analysis. The services that support these features have not received sufficient attention by the research community. Recent initiatives in integrating service-oriented technologies and big data have focused on Big-Data-as-a-Service and service generated big data [32, 22]. The former encapsulates various big data storage, management, and analytics techniques into services and provides APIs for seamless system integration. Approaches on service-generated big data propose infrastructures to provide common functionality for data management and analysis. To the best of our knowledge, there exists no study specific to the design and architecture of data-centric IoT services in big data scenarios.

SAC 2017 – IoT Track, April 03-07, 2017, Marrakech, Morocco
ACM 978-1-4503-3738-0.
http://dx.doi.org/10.1145/0000000.0000000

The focus of this paper is on the service design aspect of IoT-generated big data processing platforms. In particular, we propose a Big Service design model and framework for scalable and efficient acquisition, processing, and integration of data-centric IoT services. The special characteristics of IoT big data underpin this design model, including strong time and space correlation in the sensor data, high level of noise and redundancy in datasets, and context-driven nature of data sensing and processing. The essence of the proposed approach is to structure data-centric IoT services hierarchically in a *multi-root tree* data structure where the nodes represent Big Services. A Big Service therefore acts as an integrator, collecting data from lower level Big Services, processing them, and delivering the result to its upper level Big Services. Scalability is supported through hierarchical *service scopes*, associated to the Big Services, encapsulating the required data processing functions and real-time requirements for each service scope. We leverage a popular cloud-based service and data platform called Firebase to implement the IoT Big Service framework. The evaluation results show how the structure of the tree can collectively impact the timeliness of the Big Services.

The rest of this paper is organized as follows. In Section 2, an overview of IoT big data systems is provided. The principles and design model of IoT Big Services is presented in Section 3, while the implementation details and the evaluation results are reported in Section 4. Then, we present related work in Section 5 and conclude the paper with future directions of research in Section 6.

2. IOT BIG DATA

Connecting huge numbers of sensor-equipped physical objects to the Internet generates a new big data ecosystem. The IoT-generated data possesses properties that conform to the big data paradigm, including *volume* in terms of generating masses of data; *variety* in the form of a mixture of structured and unstructured IoT data; and *velocity* referring to the different data generation frequencies among IoT devices, and different timeliness requirements for data delivery.

However, IoT big data differs somewhat from what is generally thought of as *big data* because of the many different types of data collected and higher real-time requirements, in addition to their noisiness and often high sampling redundancy [6]. Moreover, IoT devices are often associated to a location context and every piece of data has a timestamp. This distinguishes typical big data from IoT big data as the latter includes spatial and temporal context in the data analytics system. In terms of data value in the acquisition and transmission, only a small portion of sensor data may be valuable. For example, in emergency management applications, smart devices are in charge of monitoring natural resources (e.g., rivers and forests), and human-made resources (e.g., buildings and water pipelines) to help in predicting and detecting emergencies such as fires, flooding, water leaks, etc. However, during the data acquisition, the few data sets that detect accidents are more valuable than those only sensing normal situations [5].

2.1 An Example IoT Big Data Platform

In this section, we study *smart cities* as a relevant example of a IoT big data system. IoT data generated by environmental sensors (e.g., road traffic, parking places, noise, etc.) and smart phones is the major source of information in smart city scenarios, compared to other technological sources of city information (e.g., social media and a traffic congestion repository) [30]. Typically, citizens will use their smart phones to contribute in collecting city contextual information via social networks or participatory sensing. Based on a prediction by Gartner, 2.3 billion connected things will be used by smart cities in 2017, rising to 9.7 billion by 2020 [15]. These numbers are supported by the case of intelligent transportation in London where around 8 million trips are made in a day on tubes, heavy rail and buses, resulting in about 45 million journeys a week, 180 million a month, and a billion every half year [3].

Inspired by the general big data processing architectural model, Figure 1 presents the main components of a typical smart city big data architecture. The lowest level is dedicated to collecting, processing, filtering, and storing data. The middle layer is in charge of performing data analytics for different types of data. The highest level supports visualizing and interpreting big data based on the requirements of end-user smart city applications. The architectural framework, in fact, shows implicitly the data flow from IoT devices in the city towards the data analytics components and further up to smart city applications. In this architecture, the role of the components in the lowest level as well as Real-Time Analytics are more crucial because of tight dependency between the network and deployment model of IoT devices and data acquisition and in-network processing techniques for IoT systems. Moreover, the City Context Management layer directly influences the IoT data acquisition, processing, and delivery model.

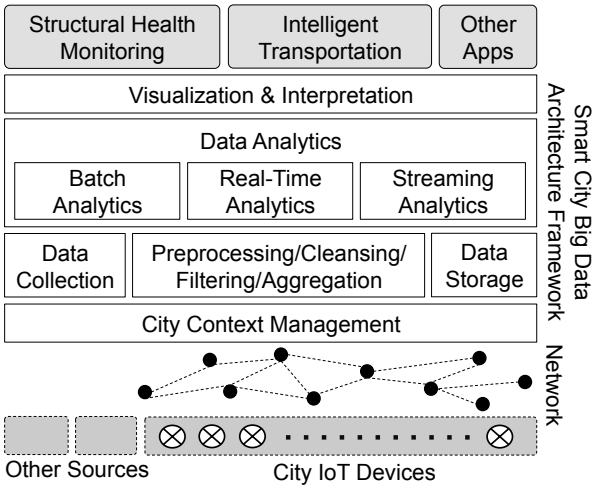

Figure 1: Overview of the smart city big data architecture

2.2 Supporting Data-Centric Services

In the above architecture, a key challenge is to provide efficient service design and integration models that realize complicated data-centric processes. The key service types that implement the big data architecture framework include [14, 7, 21, 26]:

- **Resource Access Services** are elementary or composed services that provide the basic infrastructure for collecting, preprocessing, and storing IoT data together with the associated metadata to access the data uniformly. For example, CiDAP [8] introduces *IoT-agents* from city IoT testbeds to save IoT data into NoSQL databases, including the historical data and the latest real-time data.
- **Domain-Specific Data Processing Services** enable the analysis and extraction of the data stored inside the cloud storage.

They offer query interfaces that are exposed to target application domains, e.g., traffic monitoring or weather services. Moreover, these types of services may include an event processing engine that analyses requested queries and takes real-time actions based on the application' criteria.

– **Dynamic On-Demand Services** aim at mashing up and aggregating the services offered by deployed applications. Such services are individualized or extremely diverse, and it is therefore necessary to match demand-oriented services with the customer requirements.

The integration of such massive and complicated series of data-centric services is a big challenge in developing scalable IoT big data systems. In the next section, we present our approach to this challenge through the concept of *IoT Big Services*.

3. IOT BIG SERVICES

Big Services are defined as efficient and managed integration of a massive, complicated series of data-centric services [29]. They are aimed at speeding up data processing, scaling with data volume, enhancing the adaptability and extensibility over data diversity and heterogeneity, and facilitating the interpretation of low-level data into useful knowledge to better understand and manipulate big data. IoT big data systems can benefit from Big Services in several respects. First, the high diversity of data-centric services requires scalable service integration solutions for creating diverse end-user IoT applications. Second, existing service integration models for IoT are more designed around the communication protocols and technologies [31, 11], and most data-centric services are built on data distribution middleware systems such as publish/-subscribe platforms [17]. While the former suffers from lack of scalability in complicated data-centric integration of services, the latter does not meet the goal of on-demand service integration.

Based on the above discussion, we consider the concept of Big Services from the IoT perspective and propose a novel service computing model for enabling the adoption of Big Services in large-scale IoT applications with big data processing requirements. To this end, we first highlight the IoT Big Services design requirements:

Scalable processing of data. Data-centric IoT services have to support data redundancy reduction, cleansing of noisy data, and real-time data processing and delivery. Supporting this feature for IoT services in huge deployments is a non-trivial task. In particular, an efficient service design model is required to scale with the massive number of IoT services, shared between multiple applications with different processing requirements.

Extensible design. Data-centric IoT services may be composed in different ways to fulfill varying requirements of multiple applications. In large-scale deployments, integration of such services, involved in many domain-specific service composition scenarios, is a big challenge. Thus, extensibility of the service composition model is crucial.

Supporting spatial and temporal context aspects. The two key correlated context parameters in IoT data are location and time due to the tight dependency between sensing data and the physical environment. Therefore, data-centric services need to be context-aware in terms of time and location, e.g., performing data analysis for a city district. The granularity of these contextual parameters may be very wide based on the specification of the target domain-specific service. Thus, a scalable model of spatio-temporal based service access and integration becomes a crucial requirement.

Network architecture and service model. Unlike the flat architecture of conventional networks, IoT devices are often organized into hierarchies of data access and aggregation, as is, for example the case in the recent *fog computing* based architectures for IoT systems [4]. In order to optimize the cumbersome service integration processes, we believe that data-centric IoT services should comply with the architecture of the IoT network. This not only enables efficient utilization of network resources for data processing, but also facilitates development and management of Big Services.

3.1 IoT Big Services Design Model

Our overall design idea of IoT Big Services is to create hierarchies of service access and integration, similar to a *multi-root tree* structure. We define a multi-root tree as a graph data structure that looks like a set of trees merged together, each with its own root. This choice is motivated by recent initiatives on hierarchical and end-point oriented architectures of large-scale IoT systems [4], which are, for instance, applied in smart cities [19]. This is analogous to typical graph-based service access structures. However, the hierarchical top-down access to IoT services is hardly achievable by a general graph-based modeling of IoT services. The graph-based model applies to cases in which a single IoT service needs to be associated to several contexts which are not hierarchically related, e.g., in the smart home scenario presented in [16].

The tree-based service access model enables more efficient distribution of data processing services as well as moving local data analysis processes to the *edge of the network*. Additionally, the tree structure needs to be multi-rooted in order to fulfill possibly different service composition scenarios needed by an end-user application, as well as by multiple applications sharing the same set of IoT devices. As a result, the ultimate Big Service platform may encompass several such tree structures for parallel processing of service requests by third-party applications. Figure 2 illustrates the Big Services abstraction model for a smart city system. Edges between nodes illustrate information flow from low-level smart city devices to high-level integration of sensor data at the level of a street or a municipal district.

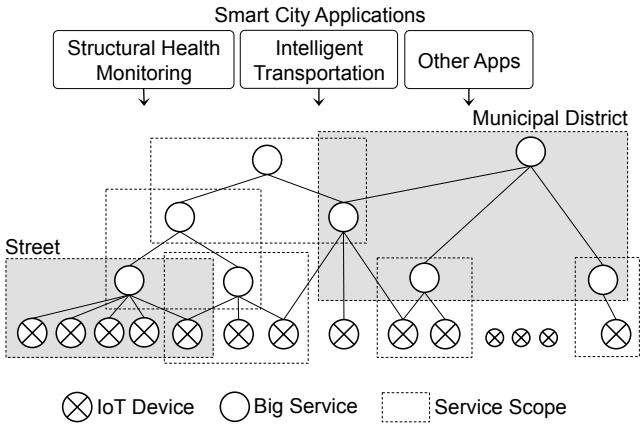

Figure 2: Overall abstraction model for IoT Big Services

In the following we will make design choices that are aimed to meet the aforementioned requirements.

3.1.1 Service Scope

Each Big Service is associated to a *service scope* and all service scopes are linked together in a hierarchical manner, called *Service Scopes Hierarchy* (SSH), according to a given system constraint, e.g., contextual properties such as location-based service delivery in smart cities. In this view, each Big Service is deployed either on intermediate network nodes or on the back-end powerful server machines that may even be hosted in the Cloud. Within each service scope, the associated Big Service (cf. Figure 2) is mainly in charge of integrating services exposed by the services at the next level down the tree, processing the services' data, and sending the output to the service node at the level above, as shown in Figure 3. The lower level nodes can either inherit the composition logic and data processing functions from the level above, or define their own functions for data processing. A common way of modeling the SSH is to use the context model of the given IoT system, similar to the context model in [9].

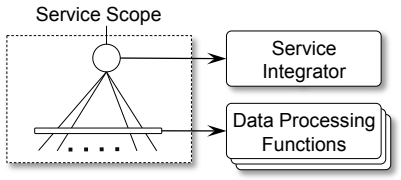

Figure 3: The Service Scope concept

As an example, Figure 4 illustrates the levels of information processing for smart pipeline monitoring [27]. Pipelines are essential infrastructure components in cities. At the lowest level, small computing nodes or edge devices are used to communicate with numerous optical fibers, measuring the temperature along the pipeline and identifying potential threat patterns on the incoming data streams. Likewise, at city districts and community levels, relevant computing platforms are used to analyze measurements at various locations and identify potential hazardous events, respectively.

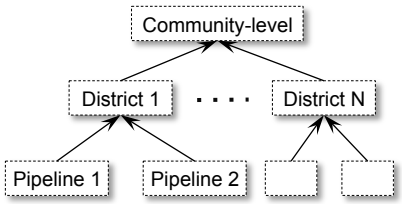

Figure 4: A sample SSH for smart pipeline monitoring

3.1.2 Domain-Specific Service Scopes

Based on the above design choice, each end-user application can specify its own SSH. For example, in the smart city case, the intelligent transport system and the emergency management system may create different integration models for the city sensors. In this way, domain-specific Big Services include their own data processing functions as well, e.g., for data redundancy reduction, or for removing noisy data. The implementation of such functions could differ from one application to another based on several factors such as the expected accuracy level of data or the volume of the processed data. Note that a SSH need not to be built from scratch. Rather, new applications can exploit part of the SSH of other running applications in order to avoid creation of redundant Big Services. This implies that the IoT Big Services framework should support discovery of existing Big Services. If a relevant service is discovered, the Big Services framework will provide an instance of the fetched service to interested applications. The mechanism for discovering Big Services is not within the scope of this paper.

3.1.3 On-Demand Big Service Creation

The top level service scopes are those created by end-user applications, integrating domain-specific Big Services and employing the necessary data processing functions. Thus, several SSHs may be considered for the integration based on the diversity of on-demand Big Services. For example, in a smart city, an on-demand Big Service is the composition of various domain-specific services in the city: the urban traffic data system can be coupled with the public transport tracking system to enable quick response to traffic events.

3.1.4 Service Interaction Models

In the IoT Big Service model, we envisage two communication models: *push-based* and *pull-based*. Big Services under the push-based communication model are rather general-purpose Big Services that can be used across different complex business processes with domain-specific service composition requirements. As an example, consider a weather forecaster offering event-based weather services, which in turn may be a Big Service composed of several lower level weather reporting services. The pull model enables provisioning of on-demand Big Services, allowing customer applications to request services exactly at the time they require it. An example of a pull-based service request from the smart city domain is an application that needs the energy consumption behavior of a specific city district to create a diagram that subsequently will be displayed to the user. We adopt the *orchestration* composition model within each node of SSH, where a Big Service communicates with all its lower level services to implement the orchestration. We choose orchestration because [10]: *i)* often smart devices provide services independent from each other, leading to the use of orchestration, and *ii)* communication complexity between nodes is avoided since peer-to-peer interaction between nodes is not needed.

3.1.5 Big Services Framework

The Big Services framework provides the basic services required to realize creation and composition of Big Services. In particular, it is in charge of implementing the Big Services composition plans described by the developer using pre-defined service orchestration techniques. This implies that the framework should first be able to discover the services needed for the composition based on their scopes, as discussed before. In addition, the framework supports execution of data analysis functions defined for a Big Service. The remaining text of this section, as well as the implementation details in the next section further clarify the role of the Big Services framework and its services.

3.2 Structure of Big Services

Listing 1 shows the pseudo-code of a Big Service and its main methods. To create a Big Service, we need to specify the `ServiceScope`, as well as the `TypedOrchestPlan`. The latter is the key implementation feature, defining a template orchestration plan based on the type of involved services, e.g., `TrafficSrvType`→ `WeatherInfoSrvType`→... . The Big Service framework first invokes service discovery to find lower level Big Services, then apply the defined `TypedOrchestPlan` to create the concrete service composition plan. When a Big Service receives a request from other services in the system, it executes the service composition

and returns the result to the requesting service.

For example, in Listing 1, we assume that the Big Service is implemented as a RESTful service, thus, implementing the `onGet()` method. Execution of a composition includes sending invocations to each individual lower level Big Services and applying the appropriate data processing functions during the composition. This may be performed after or during the execution of the composite service, depending on the data processing logic. For example, the `analyzeRedundancy` method should be called after collecting traffic service information from different city districts. The key design technique is that the data processing functions and real-time concerns in lower level services can be inherited from the higher level Big Service, enabling scalable distribution and execution of data processing functions.

```
public class MyBigService {
  ServiceScope myScope;
  Set<BigService> childServices;
  OrchstPlan myPlan;
  public MyBigService(ServiceScope aServiceScope,
                      TypedOrchestPlan aTypedPlan){
    myScope=aServiceScope;
    //based on SubServiceScopes:
    childServices=discover(myScope);
    myPlan=buildOrchPlan(aTypedPlan,childServices);
  }
  //e.g., for a RESTful Big Service
  public Data onGet(){
    Data output=executePlan(); //using myPlan
    return output;
  }
  public Data executePlan(){
    //call each childService using myPlan
    //call associated data processing functions,
    //e.g., analyzeRedundancy()
  }
  //data processing functions:
  public Set<DataItem> analyzeRedundancy(){...}
}
```

Listing 1: Pseudo-code of structure of a Big Service

4. IMPLEMENTATION AND EVALUATION

The realization of the presented IoT Big Service framework largely depends on the target system, including the IoT network architecture and the leveraged service computing technologies. Beyond that, the implementation of our framework involves provisioning a generic tree data structure for populating IoT services and creating Big Services. The ideal way to realize the latter is to investigate existing cloud computing platforms for storing and maintaining service references and build the framework on a platform that meets the design goal of hierarchical IoT Big Service provisioning. Firebase [13] is an efficient and appropriate platform for this purpose. Firebase is a cloud-based, real-time back-end system that supports the realization of various data processing features. For example, a mobile application can store a reference to a given service and also upload and add its current service information. Firebase enables a high-level decoupling between service producers and service consumers, while the structure for describing services plays a key role in providing a meaningful perception of individual service data items stored in Firebase.

The framework is implemented as a Node.js[1] server application, deployed in the Cloud and integrated with Firebase. We have developed the framework's components as Node.js modules, deployed within a Ubuntu 12.04 instance which is set up in Amazon EC2. Node.js is an open-source, cross-platform JavaScript runtime environment for developing server-side web applications. The developed framework architecturally resides between Firebase and the end-user application. It processes the hierarchical service access tree of existing IoT Big Services, e.g., domain-specific services. Then, it can create Big Services using the tree of already existing Big Services or atomic services. The core of the implementation is that, in each Firebase node, the framework maintains the URL of a RESTful web service, as well as the real-time and historical data of that service. Services are organized according to the SSH defined by the developer. For example, under the path of `city/traffic/highway1`, the service `http://ipaddress:port/services/smoke` is located to report the smoke level of a highway, where `http://ipaddress:port` denotes a domain-specific server which provides information about city sensors, including the historical data. Likewise, `city/traffic/district1` contains another service for reporting the average temperature. At the level of `city/traffic`, a new Big Service can be defined to integrate services from smoke and temperature and report the result back to the caller, e.g., other IT systems.

The Big Service developer has to describe the service composition plan on each Big Service node of Firebase. Since our framework is developed in Node.js, we have used CHORUS.JS[2] service orchestration platform in our implementation. CHORUS.JS is a light-weight and less resource demanding implementation of the Business Process Execution Language (BPEL) [20] orchestration engine, which is an important feature for our work considering the possible high number of services integrated as a IoT Big Service. In CHORUS.JS, a composition plan is defined as a Chorus Orchestration Definition (CORD) file. Then, the CHORUS.JS development tool generates the corresponding Node.js application in real-time. To support the IoT Big Service specification, we have extended CHORUS.JS to support definition of data processing functions. For example, the code snippet of a CORD in Listing 2 shows how to invoke `calculateSmoke` with a data processing function as parameter.

```
environment dev {
process EmergencyRoute version 1  {
  var input:em_level
  sequence main {
    receive emergencyLevel {
       ->input
    }
    input->values
    invoke calculateSmoke {
       smoke.calculate() -> area
    }
    ...
  }
}
```

Listing 2: A CORD code snippet with the support for defining data processing functions

We evaluate our proposed design model and framework in the context of the smart Fujisawa application developed in the ClouT project [24]. Since one key performance metric is ensuring timeliness for Big Services bounded to a temporal context, the evaluation is mainly focused on the following notification delays: *i)* Big Service-level delay: investigating the delay imposed in service no-

[1]http://www.nodejs.org

[2]http://www.chorusjs.com

tifications with respect to the number of data entries stored under a node of SSH, and *ii)* tree-level delay: evaluating the behavior of Big Service notification as contextual hierarchies of SSH grow in depth. For both of these scenarios, we use a local server version of Firebase [12], rather than using the cloud-based version of Firebase [13]. This will allow to assess the above performance metrics more accurately on a desktop machine as a worst case setting.

The first experiment is devoted to investigating the overhead of the size of a node in a given SSH. By size, we refer to the number of data sets stored under a SSH node—an IoT Big Service. The high frequency generation of data by an IoT service will cause a larger number of IoT data sets to be associated to the service (as historical IoT data). This requires a careful evaluation of notification delay when subscribed to the data generated by a Big Service. Figure 5 depicts the notification delay as the number of data sets increases from 0 to 2000. The linear behavior of delay overhead indicates that the size of historical service data should be limited to an upper bound corresponding to an acceptable notification delay. It should be noted that the cloud version of Firebase behaves more efficiently than the local server version, resulting in flatter increase in notification delay.

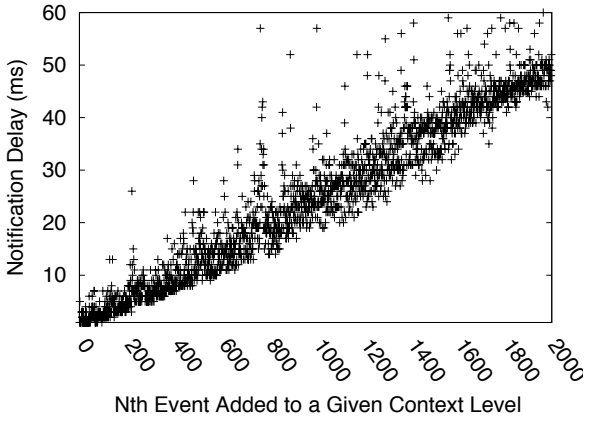

Figure 5: Notification delay with respect to the number of events added to a given node of SSH

The other performance metric is related to the depth of a Big Service in the SSH. The evaluation goal is to find out if hierarchical context levels to access a Big Service can influence the notification delay. To this end, we setup a seven-level SSH and measure the notification delay when listening to a change at the first level up to the seventh level. Figure 6 illustrates the obtained result. As shown, higher context levels lead to a longer delay, though the observed delays are negligible. The reason for this overhead is that when data is fetched for a node in Firebase, all of its lower level nodes will be fetched as well. Again, it should be noted that this result is obtained on the local Firebase server as the worst case scenario. In the cloud version of this storage platform, the trend of the notification delay in Figure 6 will become quite flat, though Firebase allows us to nest data up to 32 levels deep.

5. RELATED WORK

In early vertically provided IoT services, the developer had to implement the entire service lifecycle, from requirement analysis to hardware selection and service integration. The emergence of IoT cloud integration solutions has moved some of these challenges to

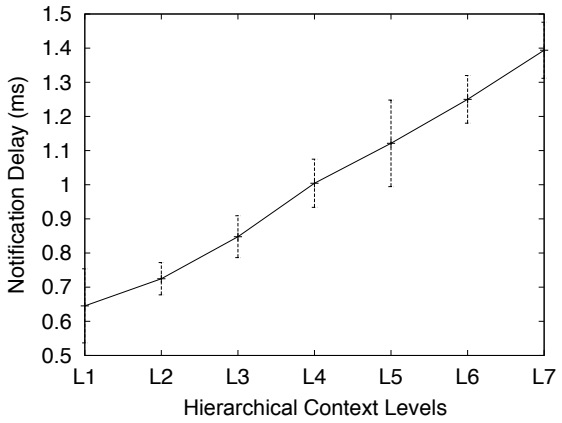

Figure 6: Notification delay increases linearly with respect to the context node level in SSH

the Cloud and hence facilitated IoT service delivery. Below, we investigate the existing IoT service provisioning approaches in the context of our work.

Traditional Web Service technologies (e.g., Simple Object Access Protocol–SOAP) have been the first work category in integrating IoT service with enterprise systems, e.g., SOCRADES [18] exploits the Device Profile for Web Services (DPWS) standards to enable Web service integration and eventing on resource-constrained devices. In [28], IoT services are composed through a data flow graph, addressing scalability in composing large-scale IoT networks through approximately-optimal composition, based on the concepts of expansion and mapping. These types of contributions are devoted to core technological challenges for service composition which is different from the focus of this paper.

Some authors have utilized Cloud features to deliver IoT services by mapping various components of clouds and IoT to the three layers of the Cloud (IaaS, PaaS and SaaS). At the PaaS level, IoT resources and cloud computing infrastructures are mashed up for applications, which is then delivered through SaaS and its on-demand service orchestration feature [25]. A common technique is to exploit the multi-tenant character of cloud computing to enable virtual vertical services instead of physically isolated vertical services [23]. In [1], The Cloud of Things (CoT) is presented to implement indexing and querying services of things, i.e., heterogeneous resources aggregated according to a given thing-like semantics and provided to end users, SaaS providers, and others as a virtual service. In [33], a cloud-based IoT platform is proposed to accommodate IaaS, PaaS, and SaaS for accelerating the development of IoT applications. Cloud-based IoT Web mashup is the other similar technique to compose a new service from existing cloud-based IoT services [18]. However, contributions in this category mainly deal with efficient usage of cloud features in IoT service delivery.

Considering IoT Big data, the existing results aim to address inherent data processing challenges, such as data acquisition, storage, management and analysis, while the services supporting such features have not been among the prioritized research concerns. Big Data-as-a-Service and service-generated big data [32, 22] are the most recent initiatives in integrating service-oriented technologies and big data. The former is aimed at encapsulating the above big data processing aspects into services and providing the associated APIs for seamless integration. The latter approach proposes the

infrastructures to provide common functionality for managing and analyzing different types of service-generated big data. However, to the best of our knowledge, there is no study devoted to the design and architecture of data-centric IoT services in big data scenarios.

6. CONCLUSIONS AND FUTURE WORK

The exponentially increasing amount of data generated by IoT systems implies the need for provisioning scalable data centric services that can be freely composed. Such services are referred to as Big Services. This paper has proposed a IoT Big Service design model and framework for scalable and efficient processing and integration of data centric IoT services. In this approach, each Big Service is associated to a service scope and all service scopes are linked together in a hierarchical manner, based on the contextual parameters of the given IoT system. The Big Service associated to a service scope is in charge of integrating services exposed by its subordinate services, processing the services' data, and sending the output to the aggregating service. We have developed the proposed framework over a popular cloud-based data storage and application platform, called Firebase. As part of our future work, we aim at investigating further the design details of the proposed framework, such as interaction models between Big Services, and Big Services discovery. The other future direction is how to maintain the Big Services tree structure when services are prone to failures.

7. REFERENCES

[1] *Enabling the Cloud of Things*, 2012.

[2] C. C. Aggarwal, N. Ashish, and A. Sheth. *The Internet of Things: A Survey from the Data-Centric Perspective*. 2013.

[3] M. Batty. Big data, smart cities and city planning. *Dialogues in Human Geography*, 3(3):274–279, 2013.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proc. of 1st Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, 2012.

[5] D. Chen et al. Natural disaster monitoring with wireless sensor networks: A case study of data-intensive applications upon low-cost scalable systems. *Mobile Networks and App.*, 18(5), 2013.

[6] M. Chen, S. Mao, and Y. Liu. Big data: A survey. *Mobile Networks and Applications*, 19(2), 2014.

[7] B. Cheng et al. Building a big data platform for smart cities: Experience and lessons from santander. In *Big Data, IEEE Inter. Cong. on*, 2015.

[8] B. Cheng et al. Building a big data platform for smart cities: Experience and lessons from santander. In *2015 IEEE International Congress on Big Data*, 2015.

[9] K. Cho et al. Hicon: a hierarchical context monitoring and composition framework for next-generation context-aware services. *IEEE Network*, 22(4), 2008.

[10] K. Dar, A. Taherkordi, R. Rouvoy, and F. Eliassen. Adaptable service composition for very-large-scale internet of things systems. In *Proceedings of the 8th Middleware Doctoral Symposium*, MDS '11. ACM, 2011.

[11] L. de Souza et al. Socrades: A web service based shop floor integration infrastructure. In *The Internet of Things*, volume 4952 of *Lecture Notes in Computer Science*. Springer, 2008.

[12] L. Firebase. http://www.npmjs.com/package/firebase-server.

[13] Firebase Cloud Platform. http://www.firebase.com/.

[14] C. Formisano et al. The advantages of iot and cloud applied to smart cities. In *Future Internet of Things and Cloud (FiCloud), 3rd Int. Conf. on*, 2015.

[15] Gartner. Forecast: Internet of things — endpoints and associated services, worldwide. Technical report, 2015.

[16] T. Gu, X. H. Wang, H. K. Pung, and D. Q. Zhang. An ontology-based context model in intelligent environments. In *In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2004.

[17] U. Hunkeler et al. Mqtt-s – a publish/subscribe protocol for wireless sensor networks. In *Communication Systems Software and Middleware and Workshops, COMSWARE 3rd Inter. Conf. on*, 2008.

[18] J. Im, S. Kim, and D. Kim. Iot mashup as a service: Cloud-based mashup service for the internet of things. In *Services Computing (SCC), 2013 IEEE International Conference on*, 2013.

[19] J. Jin et al. An information framework for creating a smart city through internet of things. *IEEE Internet of Things Journal*, 1(2), 2014.

[20] M. B. Juric. *Business Process Execution Language for Web Services BPEL and BPEL4WS 2Nd Edition*. Packt Publishing, 2006.

[21] Z. Khan, A. Anjum, and S. L. Kiani. Cloud based big data analytics for smart future cities. In *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, 2013.

[22] T. Le Dinh, T.-C. Phan, T. Bui, and M. C. Vu. *A Service-Oriented Framework for Big Data-Driven Knowledge Management Systems*. Springer, 2016.

[23] F. Li et al. Efficient and scalable iot service delivery on cloud. In *IEEE Cloud*, 2013.

[24] M. Maggio et al. D4.1 preliminary report of city application developments and field trials. Technical report, FP7 ClouT project, 2014.

[25] C. Perera et al. Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies*, 25, 2014.

[26] R. Petrolo et al. Towards a smart city based on cloud of things. In *2014 ACM Inter. Workshop on Wireless and Mobile Technologies for Smart Cities*, WiMobCity '14, 2014.

[27] B. Tang et al. A hierarchical distributed fog computing architecture for big data analysis in smart cities. In *Proceedings of the ASE BigData & SocialInformatics 2015*. ACM, 2015.

[28] T. Teixeira et al. *Service Oriented Middleware for the Internet of Things: A Perspective*. 2011.

[29] X. Xu, Q. Z. Sheng, L.-J. Zhang, Y. Fan, and S. Dustdar. From big data to big service. *Computer*, 48(7), 2015.

[30] A. Zanella, N. Bui, et al. Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1), 2014.

[31] D. Zeng, S. Guo, and Z. Cheng. The web of things: A survey (invited paper). *Journal of Communications*, 6(6), 2011.

[32] Z. Zheng et al. Service-generated big data and big data-as-a-service: An overview. In *Big Data (BigData Congress), IEEE Cong. on*, 2013.

[33] J. Zhou et al. Cloudthings: A common architecture for integrating the internet of things with cloud computing. In *Computer Supported Cooperative Work in Design (CSCWD), IEEE 17th Inter. Conf. on*, 2013.