SOFTWARE: PRACTICE AND EXPERIENCE
Softw. Pract. Exper. (2016)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/spe.2432

Analytics-as-a-service in a multi-cloud environment through semantically-enabled hierarchical data processing

Prem Prakash Jayaraman[1,*,†], Charith Perera[2], Dimitrios Georgakopoulos[1], Schahram Dustdar[3], Dhavalkumar Thakker[4] and Rajiv Ranjan[5,6]

[1]*Swinburne University of Technology, Hawthorn Campus, Melbourne, VIC 3122, Australia*
[2]*Department of Computing, The Open University, Milton Keynes, MK7 6AA, UK*
[3]*Distributed Systems Group, Vienna University of Technology, Argentinierstrasse 8/184-1, Wein A-1040, Austria*
[4]*University of Bradford, Bradford BD7 1DP, UK*
[5]*Chinese University of Geosciences, Wuhan, China*
[6]*School of Computing Science, Newcastle University, NE1 7RU, Newcastle upon Tyne, UK*

SUMMARY

A large number of cloud middleware platforms and tools are deployed to support a variety of internet-of-things (IoT) data analytics tasks. It is a common practice that such cloud platforms are only used by its owners to achieve their primary and predefined objectives, where raw and processed data are only consumed by them. However, allowing third parties to access processed data to achieve their own objectives significantly increases integration and cooperation and can also lead to innovative use of the data. Multi-cloud, privacy-aware environments facilitate such data access, allowing different parties to share processed data to reduce computation resource consumption collectively. However, there are interoperability issues in such environments that involve heterogeneous data and analytics-as-a-service providers. There is a lack of both architectural blueprints that can support such diverse, multi-cloud environments and corresponding empirical studies that show feasibility of such architectures. In this paper, we have outlined an innovative hierarchical data-processing architecture that utilises semantics at all the levels of IoT stack in multi-cloud environments. We demonstrate the feasibility of such architecture by building a system based on this architecture using *OpenIoT* as a middleware, and *Google Cloud* and *Microsoft Azure* as cloud environments. The evaluation shows that the system is scalable and has no significant limitations or overheads. Copyright © 2016 John Wiley & Sons, Ltd.

Received 29 January 2016; Revised 8 June 2016; Accepted 3 July 2016

KEY WORDS: internet of things; multi-cloud environments; big data; semantic web; data analytics

1. INTRODUCTION

Recent studies have shown that we generate 2.5 quintillion bytes of data per day [1] and this is set to explode to 40 yottabytes by 2020. This will amount to approximately 5200 GB for every person on earth. Much of these data is and will be generated from the internet of things (IoT) [2]. IoT is a part of the future internet and comprises billions of Internet-connected objects or *'things'* where each thing can sense, communicate, compute, potentially actuate and have intelligence, multi-modal interfaces, physical/virtual identities and attributes. Internet-connected objects can include wireless/wired sensors, Radio-Frequency Identification (RFID), data from social media, smart consumer appliances (TV, smart phone, etc.), smart industries (such as equipments fitted with sensors), scientific instruments (e.g. high energy physics synchrotron) and actuators. The vision

*Correspondence to: Prem Prakash Jayaraman, Swinburne University of Technology, Hawthorn Campus, Melbourne, VIC 3122, Australia.
†E-mail: prem.jayaraman@gmail.com

of IoT is to allow 'things' to be interconnected any time, anywhere, with anything and anyone, ideally using self-configured paths, networks and services. This vision has led to IoT emerging as a major producer of big data. Today, cloud technologies [3, 4] provide the ability to store and efficiently process large-scale datasets by offering a mix of software and hardware resources with modest operating costs proportional to the actual use (pay-as-you use model) [5]. It is well understood that the IoT big data applications need to process and manage streaming data from geographically distributed data sources. The cloud computing model has emerged as a suitable solution to fulfil IoT big data applications' data processing needs. The cloud essentially acts as a transparent layer between the IoT and applications providing flexibility, scalability and hiding the complexities between the two layers (IoT and applications). The fusion of cloud and IoT into 'Cloud of Things' has given rise to the following new cloud computing paradigms (but not limited to): Sensing-as-a-Service, Sensing- and Actuation-as-a-Service, Video-Surveillance-as-a-Service, Big Data Analytics-as-a-Service, Data-as-a-Service and Sensor-Event-as-a-Service. However, the integrated Cloud-of-Things approach impose several challenges right from the IoT layer including device discovery, cost-efficient communication, device management and monitoring, interoperability, quality of service and machine-to-machine (M2M) issues to the cloud layer including service discovery and delivery, big data management and analytics, cloud monitoring and orchestration, mobility issues in cloud access, privacy and security and Service Level Agreement (SLA) management. Further, the notion of *-as-a-service model will enable multiple independent operators to provide various services across the Cloud-of-Things layers that will need to be integrated based on application requirements. The prolific rise of IoT and the corresponding ecosystem will soon result in device being owned and operated by independent providers. These solutions will mostly be constrained into independent multiple-cloud provider silos. A multi-cloud environment consists of several data centres, which are geographically and topologically distributed across the Internet [6, 7]. The focus of this work is to address the challenge of facilitating multi-cloud data analytics for IoT data originating from things that are owned and operated by multiple service providers. Enabling third parties to access this data and the analytic capabilities can significantly increase the innovation and value of end-user applications. IoT big data applications that need to process and manage streaming data from multiple sources need to exploit the resources hosted across multiple cloud data centres because of following reasons [8]:

- IoT datasets and data sources can be geographically distributed; hence, moving them to a single centralised data centre could lead to high network communication overhead.
- The IoT data storage and processing needs cannot be fullfilled by the computational and storage resources offered by any single data centre. For example, in the Azure Cloud, there is a limit of 300 cores per application deployments (i.e. the maximum number of VMs that can be deployed at any instance of time). Clearly, this could lead to serious problems if the IoT datasets flow at a very high volume and velocity.
- IoT datasets may be constrained by security and legal policies; that is, data may not leave a national jurisdiction or cannot be streamed into a remote international data centre.

In this paper, we present hierarchical data analytics model for multi-cloud environments. Our proposed approach allows an end-user application to integrate and take advantage of independent infrastructure and analytics service providers. We present a use case to demonstrate the proposed hierarchical and distributed multi-cloud approach to facilitate effective and efficient sharing of analysed data across cloud providers. We use the popular open-source IoT middleware platform, namely, OpenIoT [9] to demonstrate the feasibility of our approach in multi-cloud environments. Finally, we conduct experimental evaluations on Google Cloud and Microsoft Azure platforms to establish the performance of the proposed hierarchical and distributed multi-cloud approach system.

It is important to note that our approach is not application dependant. Therefore, it can be generalised in to any application domain where only the analytical functions employed would need to be differed. Any type of analytical functions can be used on our proposed infrastructure. In this paper, we assume that all the cloud instances who engaged in a given data analytics task are trust-able and verified, before organise them into a certain hierarchical composition in order to support a given application.

DOI: 10.1002/spe

2. MOTIVATION: ANALYTICS-AS-A-SERVICE

In sensing-as-a-service [10] model, data are exchanged seamlessly among data producers (owners) and consumers via the cloud resources. Data producers are owners of the IoT devices (products) and deploy them in their environments. These IoT products sense, analyse and perform actuation to solve the needs of the data owners. While these data normally reside in individual silos, sensing-as-a-service model promotes the sharing of data (liberating data from silos) allowing data consumers to access the data using secure mechanisms. For example, a plant biologist studying the spread of certain diseases in plants may want to know the list of affected farms to better understand the trajectory of the diseases. In this case, the aim of the biologist is not to identify individual farms, but a while set of farms in specific areas. When the number of data providers and consumers increase, there is a need to develop an open data market. The data from this market may not necessarily freely available [11] (may follow the cloud computing pay-as-you-go model), but in the meta-data description, the data would be. The meta-data will enable users and other services to discover relevant data stored in data owner silos.

Analytics-as-as-Service refers to next generation IoT data processing applications where third party will be responsible for hosting IoT Analytics and data processing applications (e.g. detecting events from video camera feeds and detecting events from smart home sensors) on private/public cloud infrastructures. These analytics applications will be offered to end-users under pay-as-you-go model. Currently, such a service model is offered for cloud-based hardware (CPU, storage and network) and software (databases, message queuing systems, etc.) resources by providers such as Amazon Web Services. Providers such as SalesForce.com offers pay-as-you-go model for enterprise resource planning (ERP) and customer relationship management (CRM) applications. However, ERP and CRM applications are fundamentally different from IoT analytics applications. Moreover, Analytics-as-a-Service model introduces further complexities as there is need to describe not only the data but also the analytics performed on the data. Further, when data analytics exists as data silos within independent data owner clouds, there is a need to develop systems that can function across multiple cloud providers. Such systems will inherently require the following capabilities, namely, (i) ability to interoperate via standard interfaces, (ii) ability to describe data, (iii) support for machine to machine communication and (iv) ability to describe the analytics built on the acquired data.

Another advantage provided by Analytics-as-a-Service model is that it supports knowledge sharing while reducing the privacy risks. Because this model does not share raw data, it eliminates the risks associates with sharing raw data such as anonymised sharing of analysed data and enforce restrictions on data storage location. Another advantage is the savings of computational resources due to the elimination of redundant data processing. This means that when one cloud IoT platform perform a certain data processing task over data, the recipient cloud platforms do not required to perform the same data processing task again. For example, one IoT cloud platform may collect data form motion sensors and cameras to determine how much time in average a person may wait in a certain queue. Once such data processing is carried out, the recipient cloud can take average waiting time as an input. We elaborate on this example in Section 5 when we present the use-case scenario. Further, Analytics-as-a-Service model also reduces the data communication requirements. Typically, raw data are large in term of size. However, the processed data are significantly smaller that raw data. Therefore, the amount of data that need to be transferred from one cloud to another reduces drastically by saving network communication bandwidth and costs.

3. CURRENT STATE OF THE ART: PROCESSING DISTRIBUTED INTERNET OF THINGS DATA

Existing big data processing technologies and data centre infrastructures [12] have varied capabilities with respect to meeting the distributed IoT data processing challenges. In this section, we summarise capabilities of existing technologies based on the review given in our past work [8]. The proposed analytics-as-a-service model is expected to be extensively leverage these technologies. We have reviewed literature under six different themes: (i) basic data centre cloud computing infrastructure service stacks, (ii) massive data processing models and frameworks, (iii) trusted and integrated

DOI: 10.1002/spe

data management services across data centres, (iv) data-intensive workflow computing, (v) benchmarking, application kernels, standards and recommendations and (vi) sensing middleware in the Cloud.

1. **Basic data centre cloud computing infrastructure service stacks**

   Commercial or public data centres, for example, Amazon Web Services and Microsoft Azure offer computing, storage and software resources as remotely programmable cloud services via application programming interface (API). These resources are orchestrated by deploying virtualisation software/middleware stacks. It is well understood that virtualisation allows data centre providers to obtain more out-of-physical resources by allowing multiple instances of virtual cloud resources to run concurrently. For example, virtual machine orchestration systems such as Eucalyptus and Amazon EC2, image management tools such as FutureGrid image repository [13], massive data storage/file system such as Google file system (GFS), Hadoop Distributed File System and Amazon S3 and data-intensive execution framework including Amazon Elastic Map Reduce. In addition, FutureGrid[‡] and OpenStack also provide software stack definition for cloud data centres.

   On the other hand, private data centres are constructed typically by combining multiple types of software tools and services. These software can include cluster management systems such as Torque, OSCAR, VMWare's vCloud and/or vSphere suites and Simple Linux Utility for Resource Management, parallel file/storage systems such as storage area network (SAN)/Network attached storage (NAS)[§], Lustre and data management systems such as BeSTMan[¶] and dCache[‖]. Apart from, some private data centres are enabled for resource sharing with grid computing middleware, such as Globus Toolkits, Unicore and gLite. In general, access to private data centre resources is restricted to known group of application administrators and users because of stringent security and privacy concerns.

2. **Big Data processing models and frameworks**

   Big Data processing frameworks include software frameworks [14] that enable creation of Big Data application architecture [15]. These frameworks can be classified as follows:

   - Large-scale data mining frameworks (FlexGP, Apache Mahout, MLBase and Yahoo SAMOA) implement a wide range of data mining algorithms (clustering, decision trees, latent Dirichlet allocation, regression and Bayesian) to analyse massive data sets (historical and streaming) in parallel, by exploiting distributed resources.
   - Distributed message queuing frameworks (Amazon Kinesis and Apache Kafka) provide a reliable, high-throughput and low-latency system of queuing real-time streams of data.
   - Parallel and distributed data programming frameworks (Apache Hadoop and Apache Storm). Such frameworks enable development of distributed applications that deal with large sets of cloud resources to parallel process massive amounts of historical and streaming data [15, 16]. The large-scale data mining frameworks mentioned earlier are generally implemented on top of parallel and distributed data programming frameworks. Low-level distributed system management complexities (task scheduling, data staging, fault management, inter-process communication and result collection) are automatically taken care of by these frameworks.
   - Data store frameworks are categorised as NoSQL and Structured Query Language (SQL). NoSQL frameworks (MongoDB, HyperTable, Cassandra and Amazon Dynamo) support access based on transactional programming primitives, where an exact key allows search for an exact value. Such predetermined access patterns lead to better scalability and predictions of performance, which is suitable for storing large amounts of unstructured data (e.g. social media postings). SQL data stores (MySQL, SQL Server and PostGreSQL)

[‡]http://FutureGrid.org/.
[§] http://capitalhead.com/articles/san-vs-das-a-cost-analysis-of-storage-in-the-enterprise.aspx.
[§]http://wiki.lustre.org/.
[¶]https://sdm.lbl.gov/bestman/.
[‖]http://www.dcache.org/.

DOI: 10.1002/spe

manage data in relational tables, where the generic SQL can be used to manipulate data (insert, delete and update). In essence, SQL data stores are more effective than NoSQL stores, where transactional integrity (ACID properties) is a strict requirement. Future Big Data applications are likely to use both NoSQL and SQL data stores, driven by data varieties and querying needs. SQL engines (Apache Hive and Apache Pig) enable the querying of data across a variety of cloud storage resources including Amazon S3 and Hadoop Distributed File System based on structured query language.

3. **Data-intensive workflow orchestration framework**

   Typical workflow frameworks for managing scientific Big Data applications [17, 18] include Pegasus, Kepler, Taverna, Triana, Swift and Trident. Traditionally, in service computing domain, orchestration with Business Process Execution Language (BPEL) and yet another workflow language (YAWL) [19] has been extensively explored. On the other hand, service choreography has been carried out using WS-CDL[**]. More recently, orchestration frameworks such as Yet Another Resource Negotiator ([20]) and Mesos [21] have emerged for coordinating IoT data analytics workflow tasks across multiple Big Data processing frameworks (e.g. Apache Hadoop and Apache Storm).

4. **Benchmark, application kernels, standards and recommendations**

   Several benchmarks and application kernels have been developed, for example, Graph 500 (graph500.org/), Hadoop Sort[††] and Sort benchmark (sortbenchmark.org), MalStone [22], Yahoo! Cloud Serving Benchmark[‡‡], Google cluster workload[§§], TPC-H benchmarks (www.tpc.org/tpch), BigDataBench, BigBench, Hibench, PigMix, CloudSuite and GridMix powered by the needs of analysing the performance of different Big Data workloads. These benchmark suites model workloads for stress testing one or more categories of Big Data processing frameworks such as Apache Hadoop and Apache Mahout. In the current generation of framework suites, BigDataBench and BigBench are the most comprehensive ones. This is because they incorporate Big Data workload models for variety of processing frameworks including NoSQL, DBMS, SPEs and batch processing frameworks. Mainly, BigDataBench targets the application domains such as search engine, social network and e-commerce. Having said that, their are limited benchmarks and application kernels available for heterogeneous data centres and IoT data types. Specially, there is no consensus on available performance benchmarking for executing large-scale IoT applications across distributed data centres. Literally, the absence of inter-centre benchmark and standards needs to be the primary research agenda for the future. As of now, international organisations including NIST, OGF, DMTF Cloud working group, Cloud Security Alliance and Cloud Standards Customer Council are all working on cloud standards (occi-wg.org/)[¶¶].

5. **Sensing Middleware in the Cloud**

   Over the last few years, number of IoT cloud has been made their way in the sensing middleware marketplace. Thingworx (thingworx.com) and Xively (xively.com) are cloud-based online platforms that process, analyse and manage sensor data retrieved through a variety of different protocols. HomeOS [23] is a platform that supports home automation. HomeOS is a software platform that can be installed on a normal PC. As with the smartthings platform, applications can be installed to support different context-aware functionalities (e.g. capturing an image from a door camera and sending it to the user when someone rings the doorbell). Lab-of-things [24] is a platform built for experimental research. It allows the user to easily connect hardware sensors to the software platform and enables the collection of data and the sharing of data, codes and participants. However, most of these platforms hosted on the cloud by their owners and customers have no choice on the cloud technologies used. There are a few open source IoT platforms, developed by both research community (e.g. OpenIoT [9]) and

[**]http://www.w3.org/TR/ws-cdl-10/.
[††]http://wiki.apache.org/hadoop/Sort.
[‡‡]http://research.yahoo.com/Web_Information_Management/YCSB.
[§§]http://code.google.com/p/googleclusterdata/.
[¶¶]http://www.dmtf.org/standards/ovf.

DOI: 10.1002/spe

industrial players (e.g. WSO2IoT-wso2.com/landing/internet-of-things/) that can be hosted in any cloud available in the market today. Therefore, in this paper, we used OpenIoT as the IoT platform of choice to develop the prototypes.

4. HIERARCHICAL DATA ANALYTICS IN MULTI-CLOUDS

In this section, first, we explain what hierarchical data analysis means in multi-cloud environment and its important feature and characteristics. We then present the widely used open-source IoT platform OpenIoT and describe its features that enable multi-cloud hierarchical processing. The presented OpenIoT platform is driven by semantic web concepts and hence incorporates extensive use of ontologies to define devices and services. This feature of OpenIoT, which will be presented in detail, is the foundation for achieving the hierarchical multi-cloud data analytics model.

Let us consider Figure 1. It is important to note that hierarchical data analytics does not mean that communication network has to be hierarchical. Hierarchical data analysis can happen in any type of network. The fundamental idea is as follows. First, data are captured by leaf nodes. In Figure 1, nodes A, B, C and D can be considered as leaf notes, which are responsible for gathering data streams generated by different sources. Data sources could be hardware sensors (e.g. temperature sensor) or a virtual sensors (e.g. calling a weather service). First, the leaf nodes may analyse the data they gathered. Each node may have their own data analytical capabilities (as denoted in $a_1 \ldots a_{10}$) based on the library of data analytics tools they have access to. Once data analytics is applied by leaf node, the data are transferred to the next layer of nodes (i.e. node E and F). These nodes will run another set of analytics over the incoming data streams and generate more abstract outputs (i.e. a data stream). Finally, E and F nodes transfer their outputs to node G.

It is important to note that data processing does not follow any particular layered structure. The idea is to perform analytics in a node and pass the results onto another node to perform another set of analytics. As a result, nodes A, B, C and D do not have to be in the same layer. One stream of data may directly be sent to node A without sending them to node E if the analytics performed in node E is not required by the node A.

In both Sensing-as-a-Service model and Analytics-as-a-Service models, nodes are collecting and processing data in order to achieve their own objective. Hierarchical data analytics in multi-cloud

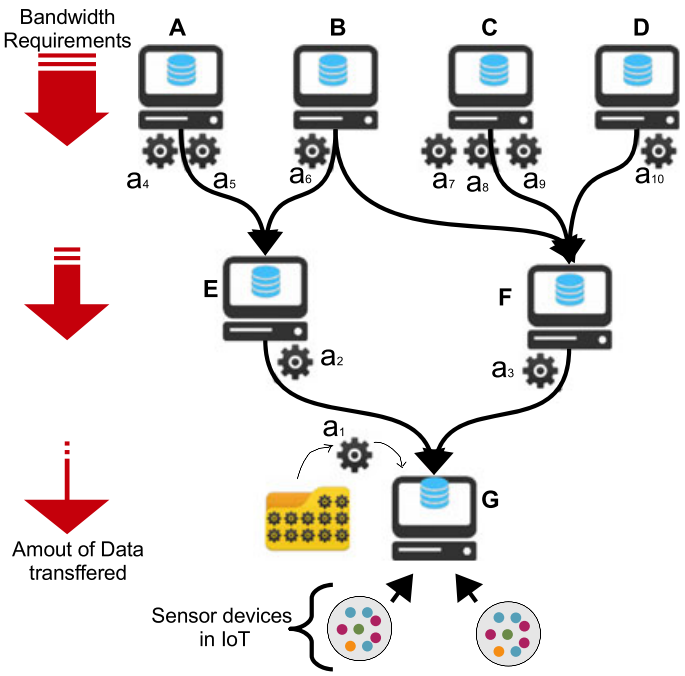


Figure 1. Theoretical view of hierarchical data analytics.

DOI: 10.1002/spe

environment occurs, when a given node does not have access to required data (e.g. node G). In such occasions, initiation node sends requests to other nodes in order to obtain access to the data it requires. Further, as shown by red arrows in Figure 1, the amount of data needs to be transferred between nodes as well as the bandwidth requirement get reduced at each layer. Primarily, the reason for this is that each layer performs some kind of analytics over the data and generates more aggregated results. For example, an average function may aggregate data over 5 min and generate a single tuple. In another instance, a function may combine sensor data from video cameras to identify the number of people entering into a certain area over an hour. Without sensing streaming video feeds, each processing node may only stream the number count to the next node in the hierarchy. The proposed model has several advantages, namely:

- It facilitates integration of services across various layers
- It allows seamless integration of data producers and consumers staying agnostic to infrastructure and technologies
- It is a platform to build complex end-user applications without owning the data production infrastructure nor the data processing tools/infrastructure
- It allows seamless discovery of service provider capabilities that can be implemented using many mechanisms including semantic discovery, probabilistic discovery and SOA-style discovery.

4.1. OpenIoT: an open source middleware for internet of things

The OpenIoT middleware [9] is a versatile blueprint architecture for collecting and processing data from Internet of Things data sources. OpenIoT provides an innovative complete IoT stack platform for IoT/cloud convergence, which enables (i) the integration and streaming of IoT data and applications within cloud computing infrastructures, (ii) the deployment of semantically interoperable applications in the cloud, (iii) the implementation of mainstream cloud computing concepts and properties in the IoT domain, including the concept of 'Sensing-as-a-Service' (i.e. on-demand, utility-based access to IoT services) and the concept of pay-as-you-go for IoT applications and (iv) handling of mobile sensors (e.g. smart phones) and associated QoS parameters (e.g. energy efficiency). OpenIoT currently uses standard communication protocols such as Transmission Control Protocol /Internet Protocol and RESTful architecture to enable communication between the different components. However, it is an open framework with support for any new protocols such as CoAP.

4.1.1. OpenIoT: architectural overview. The OpenIoT architecture is composed of seven main elements that belong to three different logical planes, as illustrated in Figure 2. These planes are the Utility/Application Plane, the Virtualised Plane and the Physical Plane, which include the following modules:

Utility/Application Plane: The utility and application plane is responsible for managing interaction with end-user applications. In particular, it provides a set of tools and interfaces that users can use to deploy IoT application on-the-fly. It comprises the following key components, namely:

- The Request Definition enables the specification of service requests to the OpenIoT platform. It comprises a set of services for specifying and formulating such requests, while also submitting them to the Global Scheduler. This component can be realised using a feature rich Graphical User Interface allowing user interaction or via APIs for machine-to-machine communication.
- The Request Presentation is responsible for visualising the outputs of an IoT service. This component creates mashups from the service description in order to facilitate presentation of analysed data.
- The Configuration and Monitoring component enables the management and configuration of functionalities over the sensors and the (OpenIoT) services that are deployed within the OpenIoT platform. Moreover, it enables the user to monitor the health of the different deployed modules.

DOI: 10.1002/spe

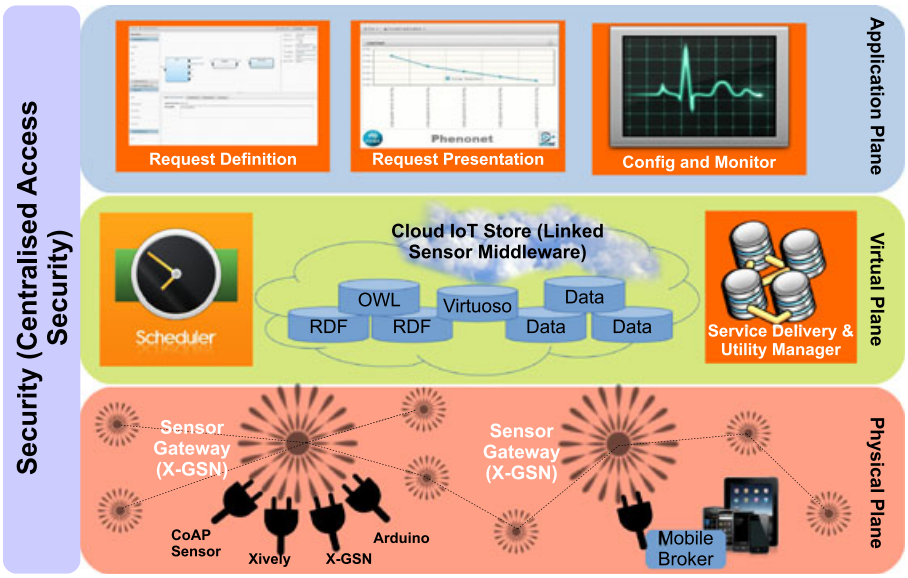


Figure 2. OpenIoT architectural overview.

Virtualised Plane: The virtual plane is responsible to bridge the device layer (physical) to the application layer. The virtual plane in most cases is deployed on cloud environments and is responsible for providing core functionalities and services to the physical and application layer. Note that the cloud infrastructure could be either a public infrastructure (such as the Amazon Elastic Compute Cloud (EC2)) or a private infrastructure (e.g. a private cloud deployed based on Open Stack (http://www.openstack.org/)). It comprises the following components

- The Directory Service (Linked Sensor Middleware Light (LSM-Light)-Light) keeps information about all the sensors and services that are available in the OpenIoT platform. It also provides the means (i.e. services) for registering sensors and services with the directory, as well as for the look-up (i.e. discovery) of sensors and services. The architecture specifies the use of semantically annotated descriptions of sensors as part of its directory service. This component is developed by extending the World Wide Web Consortium (W3C) Semantic Sensor Network (SSN) ontology [9] allowing representation of both sensors and their corresponding services, respectively. The directory service can be characterised as a sensor cloud, given that it primarily supports storage and management of sensor data streams (and of their metadata). This component of OpenIoT is vital to the relational of the proposed hierarchical multi-cloud data analytics approach and will be discussed in detail in the following section.
- The Global Scheduler processes all the requests for on-demand deployment of services and ensures their proper access to the resources (e.g. data streams). This component undertakes the task of parsing the service request and accordingly discovering the sensors that can contribute to its fulfilment. It also selects the resources, that is, sensors that will support the service deployment, while also performing the relevant reservations of resources.
- The Service Delivery & Utility Manager (SDUM) performs a dual role. On one hand, it combines the data streams as indicated by service workflow description, in order to deliver the requested service. To this end, this component makes use of the service description and the resources identified and reserved by the (Global) Scheduler component. On the other hand, this component acts as a service metering facility, which keeps track of utility metrics for each individual service. This allows utility-based metering to facilitate the development of application using service provided by disparate providers.

Physical Plane: The physical plane refers to the devices deployed in the physical environment. This can include real hardware sensors and virtual sensors. This layer is responsible for managing interactions between the device layer and the upper layers (virtual and application). This layer enables both sensing and actuation capabilities. This layer comprises the following component:

DOI: 10.1002/spe

- The Sensor Middleware (Gateway) collects, filters and combines data streams stemming from virtual sensors (e.g. signal processing algorithms, information fusion algorithms and social media data streams) or physical sensing devices (such as temperature sensors, humidity sensors and weather stations). This middleware acts as a hub between the OpenIoT platform and the physical world, since it enables access to information stemming from the real world. Furthermore, it facilitates the interfacing to a variety of physical and virtual sensors such as IETF COAP compliant sensors (i.e. sensors providing RESTful interfaces), data streams from other IoT platforms (such as https://xively.com) and social networks (such as Twitter). Among the main characteristics of the sensor middleware is its ability to stream W3 SSN compliant sensor data in the cloud. The Sensor Middleware is deployed on the basis of one or more distributed instances (nodes), which may belong to different administrative entities. The prototype implementation of the OpenIoT platform uses an enhanced/extended version of the GSN middleware (namely X-GSN, which is currently as a module of the OpenIoT open source project). However, other sensor middleware platforms could be also used in alternative implementations and deployments of the OpenIoT architecture.

Security Plane: The security plane cuts across the OpenIoT architecture stack ensuring an end-to-end security mechanism. The platform uses a token-based authentication system supported by role-based access control for authentication, authorisation and identity management.

4.2. Hierarchical multi-cloud data analytics using openIot

The OpenIoT system is driven by semantic web technologies. It extensively uses an enhanced version of the W3C SSN ontology, namely, OpenIoT ontology [25] to for semantics annotation of data at each layer of the IoT stack, that is, device layer, virtual layer and the application layers. OpenIoT exploits other semantic web technologies such as Linked Data[26] for dynamically linking related sensor data sets with corresponding services and vice-versa and Resource Description Framework (RDF), Web Ontology Language and Simple Protocol and RDF Query Language (SPARQL) for semantic modelling, representation, storage and retrieval of sensors and services. In this section, we will present the features of the OpenIoT architecture that enables the realisation of multi-cloud data analytics applications.

The virtual layer services, namely, LSM-Light, Scheduler and SDUM, are at the heart of the OpenIoT architecture that enables the following capabilities, namely: (i) ability to register sensors with semantic descriptions, (ii) Ability to register service that are composed by the user/application and (iii) a discovery service that enables semantic discovery of sensors and service. A *service* in OpenIoT is defined as a specification that defines the set of analytical operation to be performed on a stream of sensor data and the respective visual presentation.

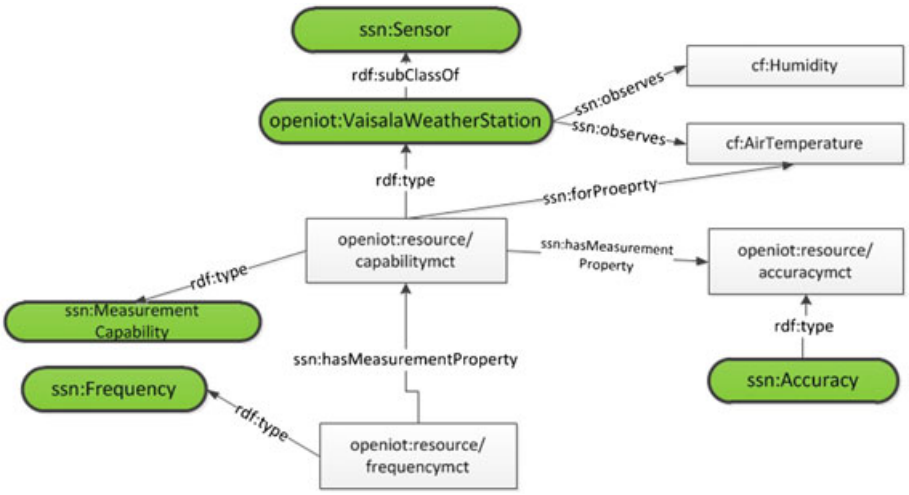


Figure 3. Sensor description based on SSN.

DOI: 10.1002/spe

Description of Devices: The OpenIoT Ontology extends the W3C SSN ontology enabling it to describe and register devices (sensors and things) with the virtual layer. Figure 3 presents an example of a partial sensor description. The RDF later describes a sensor namely a *Vaisala Weather Station* that has the capability to measure temperature and humidity.

Description of Services: The OpenIoT Service Description specification (OSDSpec) is capable of describing in detail the service composed by the user/application. The OSDSpec is modelled in the OpenIoT ontology and is stored/managed by the directory service and scheduler components of the virtual layer. This OSDSpec allows the service to be described in detail including query control features such as query schedule and permissions on the query. Listing 1 is an example of an OpenIoT OSDSpec.

Discovery and Invocation of Devices and Services

Once the devices and services are registered with the virtual plane, namely, the directory service, the directory service along with the scheduler and SDUM are used to discover and invoke

Listing 1. Sample OpenIoT service specification

```
<?xml version="1.0" encoding="UTF-8"?>
<osd:OSDSpec xmlns:st="http://www.w3.org/2007/SPARQL/
    protocol-types#"
xmlns:vbr="http://www.w3.org/2007/SPARQL/results#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:osd="http://www.openiot.eu/osdspec"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <osd:OAMO name="name0">
      <osd:OSMO name="name1">
         <osd:queryControls>
         <osd:QuerySchedule>
      </osd:QuerySchedule>
      <osd:reportIfEmpty>false</osd:reportIfEmpty>
      </osd:queryControls>
      <osd:requestPresentation>
         <osd:widget widgetID="http://www.oxygenxml.com/">
             <osd:presentationAttr name="name2" value="
                 value0"/>
             <osd:presentationAttr name="name3" value="
                 value1"/>
         </osd:widget>
         <osd:widget widgetID="http://www.oxygenxml.com/">
             <osd:presentationAttr name="name4" value="
                 value2"/>
             <osd:presentationAttr name="name5" value="
                 value3"/>
         </osd:widget>
      </osd:requestPresentation>
      <st:query-request>
         <query>query0</query>
      </st:query-request>
      <st:query-request>
         <query>query1</query>
      </st:query-request>
      </osd:OSMO>
  </osd:OAMO>
</osd:OSDSpec>
```


DOI: 10.1002/spe

Listing 2. Sample device discovery query

```
SELECT ?graphNode_2197552479500_sensorId
FROM <http://openiot.eu/OpenIoT/sensormeta#>
WHERE
{
?graphNode_2197552479500_sensorId <http://www.w3.org/1999/02/22-
    rdf-syntax-ns#type> <http://demo.org/ns#TestType> .
<http://demo.org/ns#TestType> <http://www.w3.org/2000/01/rdf-
    schema#subClassOf> <http://purl.oclc.org/NET/ssnx/ssn#Sensor>
     .
?graphNode_2197552479500_sensorId <http://www.loa-cnr.it/
    ontologies/DUL.owl#hasLocation> ?graphNode_2197552479500_loc
    .
?graphNode_2197552479500_loc geo:geometry ?
    graphNode_2197552479500_geo .
?graphNode_2197552479500_loc geo:lat ?graphNode_2197552479500_lat
    .
?graphNode_2197552479500_loc geo:long ?
    graphNode_2197552479500_lon .
FILTER (<bif:st_intersects>(?graphNode_2197552479500_geo, <
    bif:st_{p}oint>( 6.635227203369141, 46.52119378179781), 15)).
}
```

composed services. Listing 2 presents a sample SPARQL query that is used to perform semantic discovery for devices (things) within a given location. The query also takes additional parameters such as *SensorType* and *SensorClass* to perform more efficient discovery. The discovery service is also used to discover services, for example, an analytic service offered by a service provider. Together, the virtual planes enable application to discover services offered by independent sensor infrastructure owners and analytics service providers.

The virtual plane components also provide API interfaces to invoke the discovered services. The key contribution of the proposed multi-cloud model is to promote interoperability among different data and analytic service providers. This is achieved by the discovery service combined with the API allowing the development of the multi-cloud data analytics applications.

5. EXPERIMENTATIONS AND EVALUATIONS

In this section, we present a real-world use-case scenario where we demonstrate the importance of hierarchical data processing in multi-cloud environments. Then, we describe the experimental test-bed implemented using the OpenIoT system in order to validate the feasibility and conduct performance evaluations.

5.1. A case study

TrueLeisure is company that operates different types of entertainment attractions. Among them, they have franchised their amusement park chain. As depicted in Figure 4, currently, Amusement parks are located in United States, United Kingdom and Australia. These amusement parks are fully owned and operated by the franchisees. However, *TrueLeisure* continuously monitor and assess the service qualities and several other aspects of each of the amusements part. *TrueLeisure* takes these assessment seriously as their brand image is dependent on the quality of the services provided by the franchisees.

Jane is a data analyst overseeing the quality assessment tasks of amusement parks at *TrueLeisure*. She is responsible for continuously monitoring the service quality parameters. In addition to Jane,

DOI: 10.1002/spe

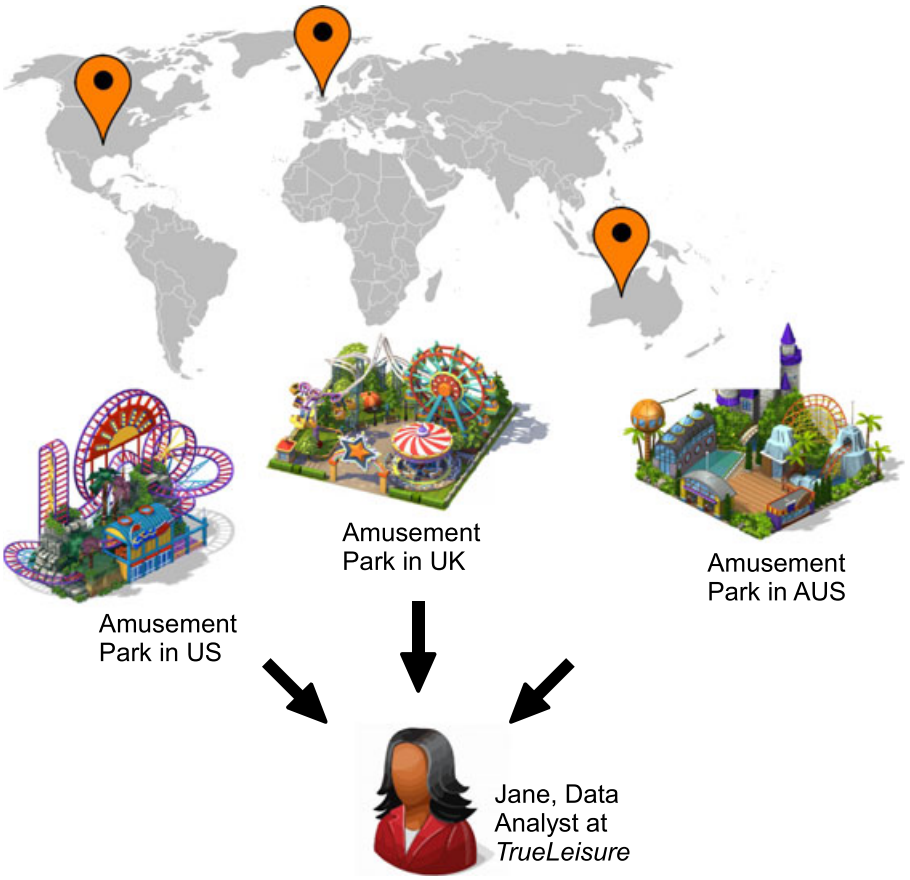


Figure 4. A case study: service quality monitoring of amusement park chain.

each of the franchisees also has their own data analysis and quality control division where they also monitor their own quality parameters. All the amusements parks are augmented with a large number of sensors that collects various types information such as environmental parameters (e.g. temperature, humidity and pressure), crowd movements, usage and demand of each rides and attractions and operational status of machinery used in the amusement part. Each of the amusement parks has deployed their own IoT platforms to which sensors are connected. Conceptually, a query would look like SELECT AVG(WaitingTime) FROM the USA, UK and Australia. The importance of this type of abstraction is that Jane does not need to know how to find waiting times in each location where each location may employ different technological means to acquire different types of sensors data to derive waiting times.

One of the important service quality parameter is *'waiting time'*. This is a main contribution factor towards customer satisfaction. Local quality assessment team continuously measures the crowd waiting time of each ride and attraction within their own amusement park. The raw data generated by sensors such as motion sensors, cameras, Bluetooth beacons and RFID tags are used to calculate these waiting times. By measuring waiting times, local data analysis team can recommend their operational division about any bottleneck within the park, so the management can take necessary actions to eliminate those to increase customer satisfaction. From Jane's perspective, who is responsible for overseeing entire portfolio of amusement parks at *TrueLeisure*, she is only interested in the *big picture*. That means Jane would like to create a single parameter of waiting time (i.e. overall waiting time) by combining individual waiting times (i.e. individual waiting time for each ride or attraction) together. As a results, she will have three measures where each represents waiting time of each amusement park locates in the USA, UK and Australia. By plotting these measures in a line chart , Jane can view how waiting time varies in real time. Jane will report these high-level

DOI: 10.1002/spe

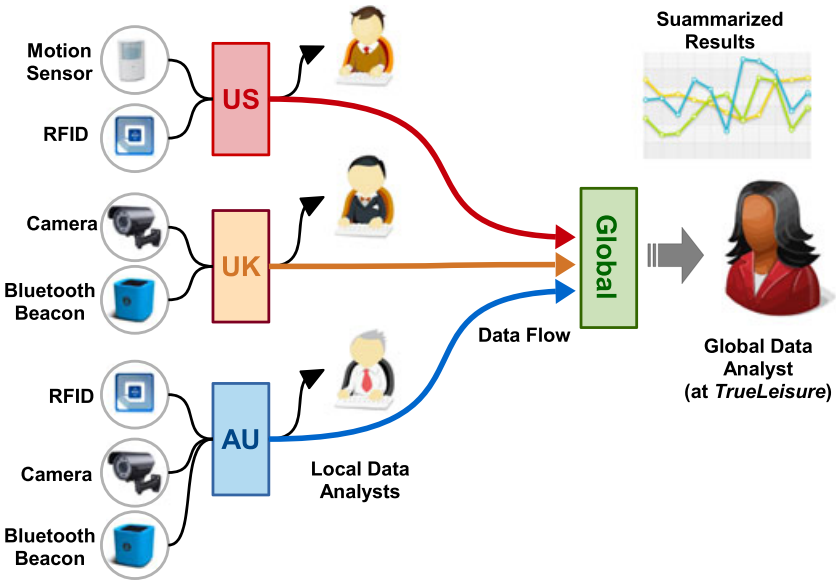


Figure 5. Data flow in hierarchical data processing.

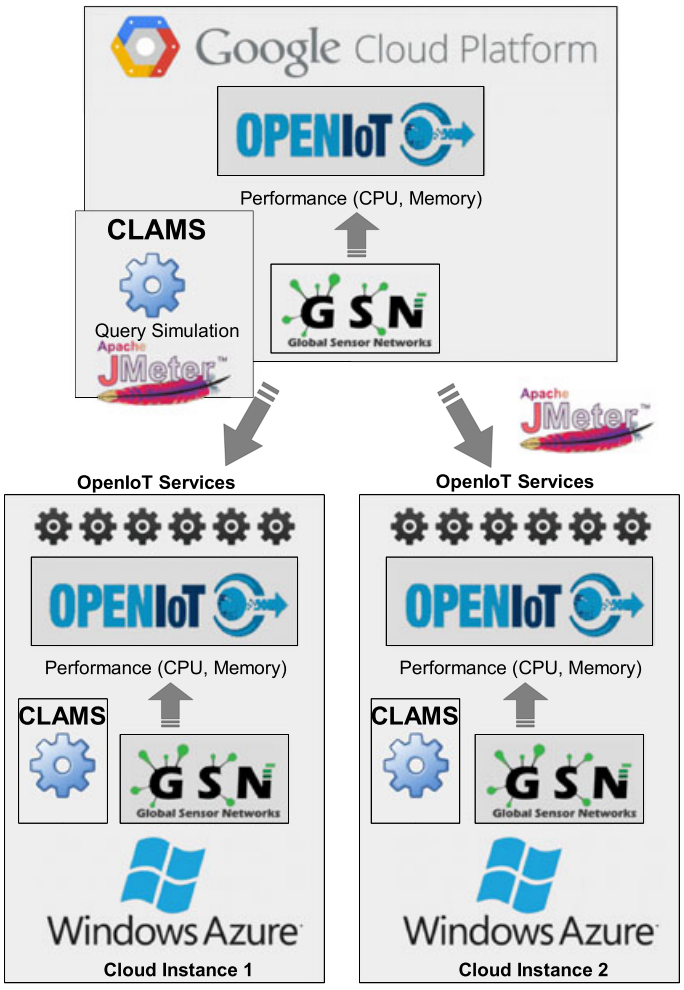


Figure 6. Experimental testbed.

DOI: 10.1002/spe

Table I. OpenIoT implementation details.

ServerName	Location/zone	Configuration
OpenIoT-1-Azure	Australia East	Standard Instance, A3(4 cores, 7-GB memory)
OpenIoT-2-Azure	Australia East	Standard Instance, A2(2 cores, 3.5-GB memory)
OpenIoT-1-Google	asia-east1-a	n1-standard-2 (2 vCPUs, 7.5-GB memory)

measures to her corporate management, so *TrueLeisure* can discuss with their franchises on future development of their theme parks efficiently and effectively. Figure 5 illustrate how data are being collected, processed and transferred in such a scenario using the proposed hierarchical data analysis in a multi-cloud environment. This scenario is a typical example of data producers, analysis service providers and data consumers operating and managing their own infrastructure (each theme park) and applications integrating these services to address specific requirements (Jane interested in overall performance of each theme park).

5.2. Experimental setup

The experimental testbed is presented in Figure 6. The analytics service at each level was implemented using the OpenIoT platform. The OpenIoT components presented in Section 4.1.1 have been implemented using Java J2EE framework using the Virtuoso RDF triplestore[27]. For more details on the implementation of OpenIoT, refer to www.openiot.eu.

The OpenIoT system was deployed on two instances of Microsoft Azure servers and one instance of a Google Cloud Server. Table I provides a summary of the server configurations. To test the performance of the system under load, we used Apache JMeter [lll] to generate user queries. The OpenIoT instance on windows azure is connected to the sensor platforms producing the data. For experimental purposes, we used a test dataset collected from publicly available weather and pollution data from the year 2014. The total amount of data in the virutoso triple store is around 10 million triples.

5.3. Experiment description

To evaluate the performance of the proposed hierarchical data analytics system using the implemented OpenIoT system on multi-cloud environments, we conduct two experiments. The OpenIoT instance on the Google Cloud (OpenIoT-1-Google) fetches data from the two OpenIoT instances on Windows Azure cloud. The OpenIoT-1-Google server fuses data from the two Azure instances to provide a combined analysis of the data to the end-user. To measure the performance of the system, we use Cross-Layer Multi-Cloud Application Monitoring-as-a-service (CLAMS) [5], a multi-cloud multi-layer performance monitoring framework. CLAMS enables a deep understanding of the performance of each individual component of our hierarchical data analytics systems deployed across the cloud layers, for example, IaaS and PaaS. CLAMS addresses the gaps in existing cloud monitoring tools inability to monitor application deployed in multi-cloud provider environments.

Experiment 1 - Streaming Data: A key to the realisation of the multi-cloud hierarchical data analytics model is its ability to handle streaming data. In this experiment, we use different two cloud configurations, namely, OpenIoT-1-Azure and OpenIoT-2-Azure. We test the stream data performance by increasing the number of sensors from 1 to 10. Each sensor produces 5 data streams including temperature, humidity, carbon monoxide, pressure and noise. So in total, when 10 sensors are active, the system handles around 50 data streams. The streaming rate is fixed at 1 data point/second. The data generated is time series data, that is, a combination of timestamps associated with data points (double).

Experiment 2 - Distributed Hierarchical Query Performance: In this experiment, we measure the response time for query processing. The queries are generated from the Google Cloud OpenIoT instance and are processed distributed by the Azure instances of OpenIoT.

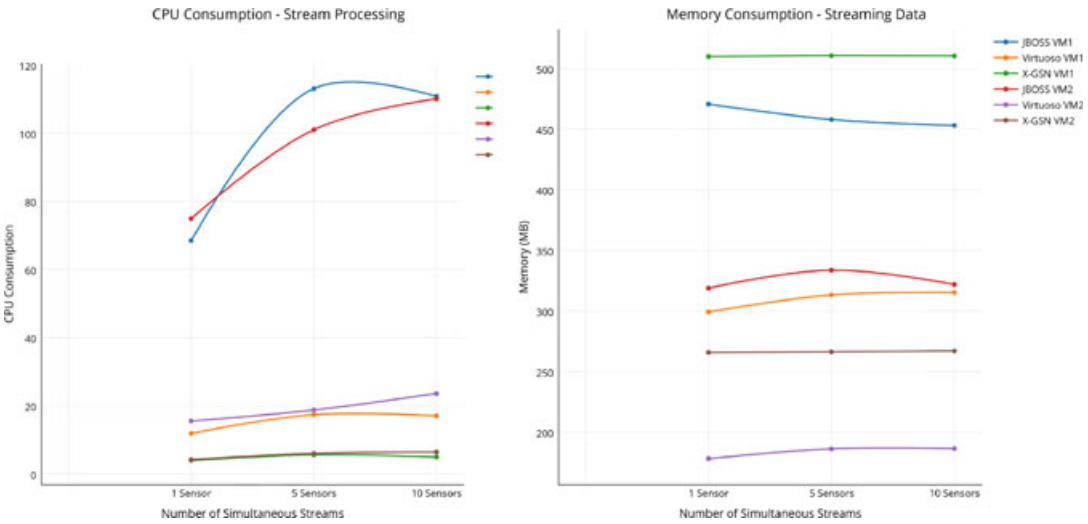
[lll]http://jmeter.apache.org/.

DOI: 10.1002/spe

In both experiments, we also compute the total CPU and memory consumption of each of the OpenIoT component. This provides us with fine grained understanding of the system's performance under load. Each experimental run was repeated three times and the results presented here are the average of these outcomes.

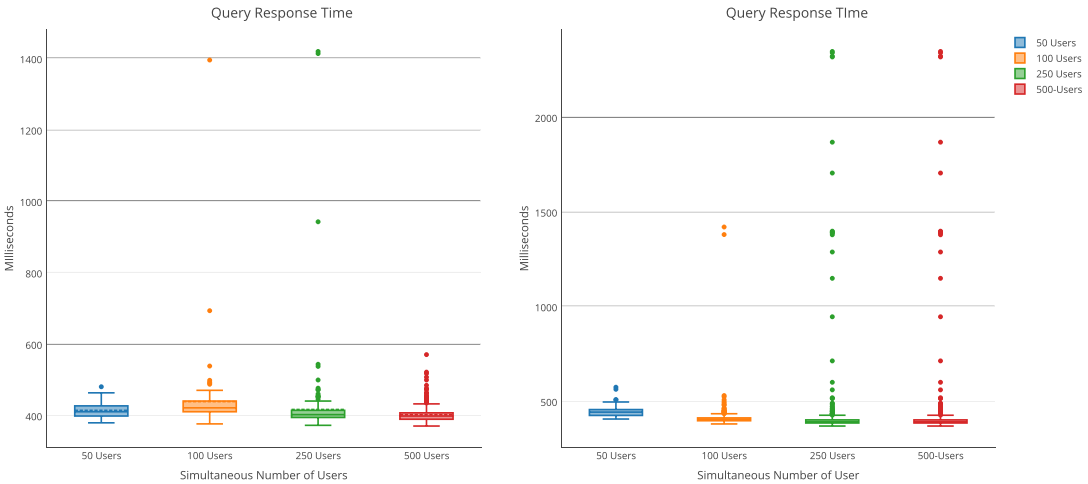
5.4. *Experimental results*

Experiment 1- Streaming Data Performance: Figure 7 presents the outcomes of our experiment. The three components that are measured here include JBOSS (hosting all the OpenIoT modules), Virtuoso (the datastore) and X-GSN (the streaming engine connecting sensors to the OpenIoT platform). The results show some interesting observations including CPU consumption of over 100%. This is because in multi-core CPU, when more than one core is used, the CPU consumption goes over 100. For example, in a four core CPU, the maximum CPU consumption as reported by CLAMS could be a maximum of 400%. The VM1 refers to the Azure-1 instance while the



(a) 1a: CPU Consumption (b) 1b: Memory Consumption

Figure 7. Streaming sata performance. (a) 1a: CPU consumption and (b) 1b: memory consumption.



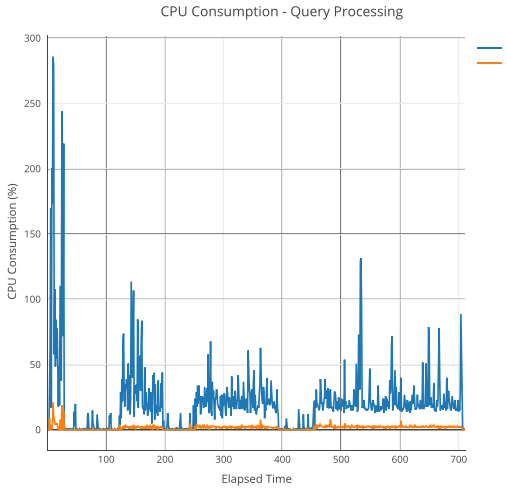
(a) 1a: Response Time - OpenIoT-1-Azure (b) 1b: Response Time - OpenIoT-2-Azure

Figure 8. Query tesponse times. (a) 1a: response time – OpenIoT-1-Azure and (b) 1b: response time – OpenIoT-2-Azure.

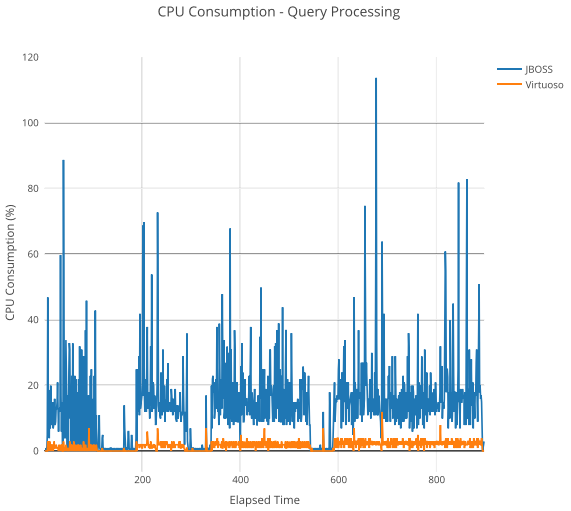
DOI: 10.1002/spe

VM2 refers to Azure-2 instance. Overall, for managing 50 data streams (10 sensors) at the rate of 1 s, the system performs significantly well without any major bottlenecks. Because the memory consumption of the JBOSS is controlled by the JVM, a trend of higher memory consumption for VM1 can be noted. This is due to the higher memory availability (7 GB) on VM1 as compared with VM2 (3.5 GB).

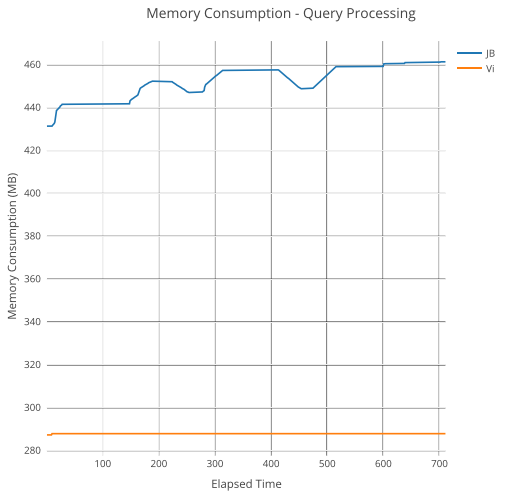
Experiment 2- Distributed Hierarchical Query Performance: Figure 8 presents the outcome of query response times on the two Azure configuration. The queries originated from the Google Cloud OpenIoT instance. In general, the overall query response time is very good in the order of 400–450 ms with number of parallel users increasing from 50 to 500. As expected, the Azure 1 instance that has more memory and CPU cores performs better than the Azure 2 instance. The interesting result here is that the response time decreases as number of users increase. This is something we suspect to be associated with how the JVM will allocate memory when the load on the system increases. This outcome is consistent with the outcomes from both the Azure configurations.



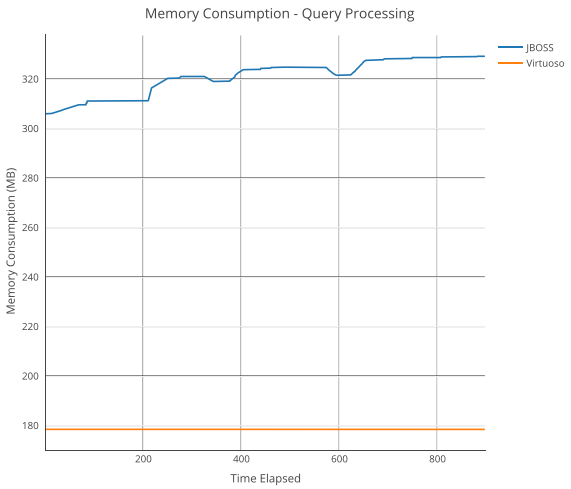
(a) 1a: CPU Consumption - OpenIoT-1-Azure



(b) 1b: CPU Consumption - OpenIoT-2-Azure



(c) 1a: Memory Consumption - OpenIoT-1-Azure



(d) 1b: Memory Consumption - OpenIoT-2-Azure

Figure 9. Hierarchical query processing performance. (a) 1a: CPU consumption – OpenIoT-1-Azure, (b) 1b: CPU consumption – OpenIoT-2-Azure, (c) 1a: memory consumption – OpenIoT-1-Azure and (d) 1b: memory consumption – OpenIoT-2-Azure.

DOI: 10.1002/spe

Figure 9 presents the CPU and memory consumption of both the Azure 1 and Azure 2 instances while processing the queries from the Google Cloud instance. As described earlier, because of the higher configuration of Azure 1, we note that the JBOSS component of OpenIoT in Azure 1 consumes upto 300% CPU. The same outcomes is observed with the Memory consumption of JBOSS on each of the instance.

The experimental outcomes validate the following key contributes of the paper. (i) It is feasible to deploy a hierarchical data analytics system where the various systems could be owned by different providers. (ii) Using device and service discovery, we can compose multi-cloud data analytics applications. (iii) The performance of such a system implemented using the widely used OpenIoT system is scalable and does not show any significant limitations or overheads.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a novel, hierarchical data processing architecture suitable for multi-cloud environments. This architecture provides flexibility to different parties who host their own cloud IoT platforms to share processed data to reduce computation resource consumption collectively. This also reduces the risks associated in sharing raw data. Such low privacy risks encourage data owners to share their data with third parties where they will use such data for secondary objectives. The demonstrated system is semantically inter-operable. Such interoperability allows different instances deployed in multi-cloud environments to work together to collectively analyse data to achieve a common objective through hierarchical data processing. This was demonstrated in this paper by real-world implementation of the OpenIoT system on Azure and Google cloud platforms. Finally, the experimental results validate the scalability of our proposed multi-cloud data analytics approach. Moreover, experimental outcomes also show that the system does not impose any significant limitations or overheads. Our next step is to develop a complimentary performance model for such hierarchical data processing in multi-cloud environments for autonomous provisioning of cloud resources.

ACKNOWLEDGEMENT

Charith Perera's work is supported by European Research Council Advanced Grant 291652 (ASAP).

REFERENCES

1. Pepper R, Garrity J. The internet of everything: how the network unleashes the benefits of big data. *Technical Report*, CISCO, 2014. Available at: http://blogs.cisco.com/wp-content/uploads/GITR-2014-Cisco-Chapter.pdf [last accessed January 2016].
2. Perera C, Zaslavsky A, Christen P, Georgakopoulos D. Context aware computing for the internet of things: a survey. *IEEE Communications Surveys Tutorials* 2013; **16**(1):414–454.
3. Wang L, Chen D, Zhao J, Tao J. Resource management of distributed virtual machines. *International Journal of Ad Hoc and Ubiquitous Computing* 2012; **10**(2):96–111. DOI: 10.1504/IJAHUC.2012.048261.
4. Wang L, von Laszewski G, Younge A, He X, Kunze M, Tao J, Fu C. Cloud computing: a perspective study. *New Generation Computing* 2010; **28**(2):137–146. DOI: 10.1007/s00354-008-0081-5.
5. Alhamazani K, Ranjan R, Mitra K, Rabhi F, Jayaraman P, Khan S, Guabtni A, Bhatnagar V. An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art. *Computing* 2015; **97**(4): 357–377. DOI: 10.1007/s00607-014-0398-5.
6. Song W, Wang L, Ranjan R, Kolodziej J, Chen D. Towards modeling large-scale data flows in a multidatacenter computing system with petri net. *IEEE Systems Journal* 2015; **9**(2):416–426. DOI: 10.1109/JSYST.2013. 2283954.
7. Wang L, Ma Y, Zomaya A, Ranjan R, Chen D. A parallel file system with application-aware data layout policies for massive remote sensing image processing in digital earth. *IEEE Transactions on Parallel and Distributed Systems* 2015June; **26**(6):1497–1508. DOI: 10.1109/TPDS.2014.2322362.
8. Wang L, Ranjan R. Processing distributed internet of things data in clouds. *IEEE Cloud Computing* 2015; **2**(1): 76–80. DOI: 10.1109/MCC.2015.14.
9. Soldatos J, Kefalakis N, Hauswirth M, Serrano M, Calbimonte J-P, Riahi M, Aberer K, Jayaraman P, Zaslavsky A, Žrko I, Skorin-Kapov L, Herzog R. Openiot: Open Source Internet-of-things in the Cloud. In *Interoperability and Open-Source Solutions for the Internet of Things*, vol. 9001, Lecture Notes in Computer Science. Springer International Publishing, 2015; 13–25.

DOI: 10.1002/spe

10. Perera C, Zaslavsky A, Christen P, Georgakopoulos D. Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies (ETT)* 2014; **25**(1):81–93. DOI: 10.1002/ett.2704.
11. Perera C, Zaslavsky A. Improve the sustainability of internet of things through trading-based value creation. *2014 IEEE World Forum on Internet of things (WF-Iot)*, Seoul, 2014; 135–140.
12. Wang L, Chen D, Hu Y, Ma Y, Wang J. Towards enabling cyberinfrastructure as a service in clouds. *Electrical and Computer Engineering* January 2013; **39**(1):3–14. DOI: 10.1016/j.compeleceng.2012.05.001.
13. Diaz J, von Laszewski G, Wang F, Younge A, Fox G. Futuregrid image repository: a generic catalog and storage system for heterogeneous virtual machine images. *2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*, Athens, 2011; 560–564.
14. Chen D, Hu Y, Cai C, Zeng K, Li X. Brain big data processing with massively parallel computing technology: challenges and opportunities. *Software: Practice and Experience* 2016:n/a–n/a. DOI: 10.1002/spe.2418.
15. Deng Z, Wu X, Wang L, Chen X, Ranjan R, Zomaya A, Chen D. Parallel processing of dynamic continuous queries over streaming data flows. *IEEE Transactions on Parallel and Distributed Systems* 2015; **26**(3):834–846. DOI: 10.1109/TPDS.2014.2311811.
16. WangL, Geng H, Liu P, Lu K,Kolodziej J, Ranjan R, Zomaya AY. Particle swarm optimization based dictionary learning for remote sensing big data. *Knowledge-Based Systems* 2015;**79**(0):43–50. DOI: 10.1016/j.knosys.2014.10.004.
17. Hu Y, Wang L, Liu Y, Chen D, Li X. Towards an efficient multi-way factorization of multi-dimensional big data across a gpu cluster. *2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, Chengdu, 2015; 18–24.
18. Chen D, Li X, Wang L, Khan SU, Wang J, Zeng K, Cai C. Fast and scalable multi-way analysis of massive neural data. *IEEE Transactions on Computers* 2015; **64**(3):707–719. DOI: 10.1109/TC.2013.2295806.
19. Ouyang C, Adams M, ter Hofstede A. Yet Another Workflow Language: Concepts, Tool Support and Application. In *Handbook of Research on Business Process Modeling*, Cardoso J, van der Aalst W (eds). IGI Global: Germany, 2009; 91–121.
20. Vavilapalli VK, Murthy AC, Douglas C, Agarwal S, Konar M, Evans R, Graves T, Lowe J, Shah H, Seth S, Saha B, Curino C, O'Malley O, Radia S, Reed B, Baldeschwieler E. Apache hadoop yarn: yet another resource negotiator. *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, ACM: New York, NY, USA, 2013; 5:1–5:16. DOI: 10.1145/2523616.2523633.
21. Hindman B, Konwinski A, Zaharia M, Ghodsi A, Joseph AD, Katz R, Shenker S, Stoica I. Mesos: a platform for fine-grained resource sharing in the data center. *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, USENIX Association: Berkeley, CA, USA, 2011; 295–308.
22. Bennett C, Grossman R, MalStone JS. A benchmark for data intensive computing, Open Cloud Consortium, 2009.
23. Dixon C, Mahajan R, Agarwal S, Brush AJ, Lee B, Saroiu S, Bahl P. An operating system for the home. *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, USENIX Association: Berkeley, CA, USA, 2012; 25–25.
24. Brush AB, Filippov E, Huang D, Jung J, Mahajan R, Martinez F, Mazhar K, Phanishayee A, Samuel A, Scott J, Singh RP. Lab of things: A platform for conducting studies with connected devices in multiple homes, UbiComp '13 Adjunct: ACM, New York, NY, USA, 2013; 35–38. DOI: 10.1145/2494091.2502068.
25. Soldatos J, Kefalakis N, Hauswirth M, Serrano M, Calbimonte J-P, Riahi M, Aberer K, Jayaraman PP, Zaslavsky A, Žarko IP, Skorin-Kapov L, Herzog R. *Interoperability and Open-source Solutions for the Internet of Things: International Workshop, FP7 OpenIot Project, Held in Conjunction with SoftCOM 2014, Split, Croatia, September 18, 2014, Invited Papers*, chap. OpenIoT: Open Source Internet-of-Things in the Cloud. Springer International Publishing: Cham, 2015; 13–25, DOI: 10.1007/978-3-319-16546-2_3.
26. Bizer C, Heath T, Berners-Lee T. Linked data-the story so far. *Semantic Services, Interoperability and Web Applications: Emerging Concepts. IGI Global* 2011:205–227. DOI:10.4018/978-1-60960-593-3.ch008.
27. Thakker D, Osman T, Gohil S, Lakin P. A Pragmatic Approach to Semantic Repositories Benchmarking. In *The Semantic Web: Research and Applications*. Springer-Verlag: Berlin, Heidelberg, 2010; 379–393.