Future Generation Computer Systems 32 (2014) 307–323

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Semantic-based QoS management in cloud systems: Current status and future challenges

Dimitrios Kourtesis, Jose María Alvarez-Rodríguez *, Iraklis Paraskakis

South East European Research Centre (SEERC), 24 Proxenou Koromila Street, Thessaloniki, 54622, Greece

HIGHLIGHTS

- We review the concept of Quality of Service in Cloud and Service Oriented Computing.
- We review the use of Semantics in Cloud and Service Oriented Computing.
- We review the existing techniques to deal with Big Data.
- We propose a Lambda Architecture based on Semantics and Big Data.
- We discuss and outline future challenges in semantic-based QoS management.

ARTICLE INFO

Article history:
Received 28 February 2013
Received in revised form
7 October 2013
Accepted 13 October 2013
Available online 26 October 2013

Keywords:
Cloud systems
Quality of service
Service oriented architectures
Semantics
Ontologies
Linked data
Sensor data
Big data

ABSTRACT

Cloud Computing and Service Oriented Architectures have seen a dramatic increase of the amount of applications, services, management platforms, data, etc. gaining momentum for the necessity of new complex methods and techniques to deal with the vast heterogeneity of data sources or services. In this sense Quality of Service (QoS) seeks for providing an intelligent environment of self-management components based on domain knowledge in which cloud components can be optimized easing the transition to an advanced governance environment. On the other hand, semantics and ontologies have emerged to afford a common and standard data model that eases the interoperability, integration and monitoring of knowledge-based systems. Taking into account the necessity of an interoperable and intelligent system to manage QoS in cloud-based systems and the emerging application of semantics in different domains, this paper reviews the main approaches for semantic-based QoS management as well as the principal methods, techniques and standards for processing and exploiting diverse data providing advanced real-time monitoring services. A semantic-based framework for QoS management is also outlined taking advantage of semantic technologies and distributed datastream processing techniques. Finally a discussion of existing efforts and challenges is also provided to suggest future directions.

1. Introduction

Cloud Computing [1] systems and Service Oriented Architectures (SOA) have reached a level of complexity [2,3] that implies the necessity of new methods and algorithms to automatically deal with the vast amount of data, variables, parameters, etc. that appears in this new realm for the advanced management of applications, services or resources.

In this new environment QoS is playing a relatively minor role but its importance, in a wide range of application scenarios, has likely become more crucial than ever before. The main problem lies in the complexity of designing QoS models that enables an adequate management of a distributed architecture making decisions about resource provisioning, getting feedback for the final users, etc. with the objective of avoiding existing "brute-force" solutions and overprovisioning.

Although QoS management and the concept of "Measured Service" [1] have been widely investigated [3] in the well-known grid-computing area, the emergence of the Cloud Computing paradigm brings a new set of open issues: accomplish the Service Level Agreements (SLAs), predict future workload, process large and diverse data streams/logs, make real-time decisions, reasoning and inference, dynamic adaptation and provision of resources, etc. Nowadays old-fashioned QoS management techniques (based on a combination of network resource provisioning with techniques such as admission control or active queue management) can be applied to a static environment but in the near future with the new implications for professionals [4], the challenge of providing higher

* Corresponding author. Tel.: +30 2310253477.
E-mail addresses: dkourtesis@seerc.org (D. Kourtesis), jmalvarez@seerc.org (J.M. Alvarez-Rodríguez), iparaskakis@seerc.org (I. Paraskakis).
URLs: http://www.seerc.org/ikm (D. Kourtesis), http://www.josemalvarez.es (J.M. Alvarez-Rodríguez), http://www.seerc.org/ikm (I. Paraskakis).

0167-739X/$ – see front matter
http://dx.doi.org/10.1016/j.future.2013.10.015

elasticity and dynamic adaptation cannot be accomplished with these methods.

In this sense, the features and requirements of the new cloud systems with regards to QoS [5] and the aforementioned implications in the future IT [4] match the advantages of software component and knowledge-based architectures. In fact, Autonomic Computing support for the next generation of cloud systems needs to be [3,5]: (1) Self-x management, (2) agile, flexible and reliable, (4) deployable over a multiple cloud platforms, (5) handle complexity, (6) enable collaboration and coordination and (7) cost-effective and greener (energy-efficient). Under this context, semantic technologies have emerged as an option to design and develop intelligent software components and agents to perform certain tasks on the Web and fulfill user's requirements (in this case applications). Therefore, Semantics enables machines to automatically process and enrich data from different sources and has the potential to deeply influence the further development of the Internet Economy as cloud systems also does.

On the other hand, in the Semantic Web area there is a growing commitment to process large data streams applying new stream reasoning [6,7] or complex event processing [8] (CEP) techniques. Furthermore there are research works offering cloud-based solutions to deal with Big Data [9] (e.g. analysis of social media), modeling SLAs and ECA rules with ontologies, monitoring real-time systems (e.g. traffic), sensor networks, or making decisions in a collaborative fashion [10] (e.g. clinical reasoning). The main advantage of applying semantic technologies to a specific domain lies in the standard representation of knowledge and data through a common-shared data model (RDF) and the capacity of reusing existing knowledge through ontologies (OWL). Thus, data coming from cloud systems can be automatically processed, checked for inconsistencies and used in expert systems to support self-x management activities.

This review is intended to provide researchers, developers and practitioners a summary of the current status of QoS management in Cloud Computing and SOA applying semantics. To do so, the paper is structured as follows. Next section reviews the background concepts required to a better understanding of the paper. Section 3 presents the existing works to perform QoS management using semantics; more specifically most of the ontology-based frameworks for QoS management are reviewed. Afterwards, a review of existing techniques for processing large data streams is also provided in Section 4. Section 5 outlines a framework to meet QoS requirements applying semantics and data stream processing techniques in real-time. Finally, the paper ends with a discussion of existing approaches for semantic-based QoS management, limitations, future challenges and concluding remarks.

1.1. Motivating scenario

The vast amount of cloud infrastructures, platforms and services is becoming a major challenge for technicians and IT/Business managers. According to the Gartner's hype cycle on cloud computing, one of the next big things in this environment lies in the creation of cloud brokers that can automatically fulfill the requirements of an application, user or organization in an automatic way. In this sense the accomplishment of QoS indicators, SLA agreements, ECA rules, etc. is considered a key-enabler to ease and boost the creation of cloud brokers with the aim of easing organizations to overcome their initial lack of experience and to manage the different service levels of an organization.

Therefore and following Gartner's definition, a cloud service brokerage (CSB) is a kind of service intermediation *in which a company or other entity adds value to one or more (generally public or hybrid, but possibly private) cloud services on behalf of one or more consumers of that service*. Some of the capabilities of a CSB must include: governance, community management, service enrichment and deliver, distributed quality of service, analytics and operational intelligence and SaaS and custom SaaS (to support business processes). Although its potential benefit rating is set as *High*, the market penetration is just 1%–5% *of target audience* and its maturity is still in an *Adolescent* status. According to the "MUST" capabilities of a CSB it seems that a QoS-driven CSB can really help organizations to improve or deploy the business models using a cloud infrastructure. Nevertheless a right deployment of a CSB will depend on its capability to process diverse data and perform analytical processes in an adequate fashion. Furthermore a CSB is supposed to be used for both technicians and business users to make their own decisions in different aspects: (1) select a service provider or an API and/or (2) select the "best" cloud infrastructure for organization business.

For instance, let us suppose that a developer needs to implement a new mobile application with a geocoding capabilities and a response time in terms of milliseconds. The selection of this API is not a mere process of ensuring the functionality but to track and assess "developer's quality criteria". In this sense, a CBS based on QoS management can deal with some of the questions that can arise such as: (1) Which is the "best" API for a geocoding service? (more than 54 existing geocoding APIs); (2) How can the developer compare (ranking of) different providers?; (3) How can the developer track the quality (response time) of the selected service?, etc.

Now let us turn this simple example into a generic "template" for a CBS. Let us suppose that an agent needs to (create|deploy|implement|move) a new action re-using an existing service (XaaS) under a certain set of key performance indicators (KPIs): (1) Which is the "best" provider for the service (XaaS)?; (2) How can the agent compare (ranking of) different providers? or (3) How can the agent track the quality (KPIs) of the selected service (XaaS)?.

In this scenario a CBS can clearly help "agents" to select the appropriate service and provider but it is necessary to enable a way of expressing and computing the "best" provider, comparing existing ones and tracking the selected one according to "my" quality restrictions. Due to the aforementioned tangled environment it also requires the use of the right methods to model user intentions and provider capabilities, to integrate data and compute in a standard way the "best service". As main conclusion a QoS-based CBS can be the next big thing in a Cloud Computing environment but the use of semantics and distributed processing techniques of data streams must be considered to enable the proper and efficient exploitation of data, information and knowledge in order to help agents to make decisions.

2. Background

This section presents a brief summary of the background concepts reviewed in this paper: (1) Cloud Computing and QoS; (2) Semantic Technologies and (3) Big Data, with the aim of building a common understanding of the requirements of QoS in Cloud Computing.

2.1. Quality of service in cloud systems

Cloud Computing represents the next natural step in the evolution of on-demand services and applications. Several definitions have been made but the description [1] provided by the NIST institute has reached a major consensus. These basic concepts [1] and usages [11] in a cloud environment lead us to consider that QoS is a key-enabler of the five essential characteristics [1] identified by the NIST institute and it is closely related to the concepts of Autonomic and Utility Computing [2]. As a consequence the QoS management

must play a major role in cloud environments and in the Future Internet to afford, from a quality point of view, the implementation of the "Measured Service" concept.

On the other hand, the ITUT-T Recommendation E.800 defines QoS as *collective effect of service performance that determines the degree of satisfaction by a user of the service*. Thus QoS data is a key-enabler to design, identify and put in action SLAs. It should also influence software components and applications to ensure a reliable environment for executing services. Some open issues in QoS management emerge to extend this definition including reputation-based mechanisms for service selection or dynamic adaptation of resource provisioning.

In order to facilitate the QoS management in the cloud-environment some tools, called Cloud Management Platforms (CMPs) can be found to manage the different layers of cloud-based applications but the majority of them are now focused on the IaaS and PaaS layers. The use of these platforms can help to manage the growing of cloud applications and ease the deployment and monitoring of services across public and private clouds. The six key capabilities [12] that we should look for in a CMP are: simplify complexity, manage multiple clouds, build for the future, support the whole application lifecycle, self-management (set-it and forget-it) and manage/control costs. In this sense OASIS just launched a CAMP TC to create and inter-operable protocol that cloud implementers can use to package and deploy their applications. The idea is to provide a set of REST services, at the PaaS layer, to foster an ecosystem of common tools, plug-ins, libraries and frameworks, which will allow vendors to offer greater value-add. In the particular case of QoS, the use of standards to gather data from applications can improve the process of making decisions about resource provisioning or help in saving costs among others. Although there is no a clear objective to support quality indicators it can be considered as a major effort to unify information exposed by providers and improve the creation of an integrated and inter-operable ecosystem in which existing cloud management application platforms such as RightScale, Enstratus, ScaleUp, Cloudability, Cloudyn, CloudExpress or MyGravitant can take advantage of implemented added-value services on the top of a common API.

On the other hand there is an interesting approach to manage cloud quality indicators using a policy-making perspective. In this sense public and private bodies are continuously seeking for new analytical tools and methods to assess, rank and compare their performance based on distinct indicators and dimensions with the objective of making some decision or developing a new policy. In this context the creation and use of quantitative indexes is a widely accepted practice that has been applied to various domains such as Bibliometrics and academic performance and quality (the Impact Factor by Thomson-Reuters, the H-index or the Shanghai and Webometrics rankings) the Web impact (the Webindex [13] by the Webfoundation) or Smart Cities (The European Smart Cities ranking) to name a few. Therefore policymakers as well as individuals are continuously evaluating quantitative measures to tackle or improve existing problems in different areas and support their decisions. Nevertheless the sheer mass of data now available is raising a new dynamic and challenging environment in which traditional tools are facing major problems to deal with data-sources diversity, structural issues or complex processes of estimation. According to some efforts such as the "Policy-making 2.0" within the Cross-Over project[1] that *refers to a blend of emerging and fast developing technologies that enable better, more timely and more participated decision-making*, new paradigms and tools are required to take advantage of the existing environment (open data and big data) to design and estimate actions in this dynamic context according to requirements of transparency, standardization, adaptability and extensibility among others with the aim of providing new context-aware and added-value services such as visualization that can help a deepen and broaden understanding of the impact of a policy in a more fast and efficient way. As a consequence common features and requirements can be extracted from the existing situation out: diverse data sources management, definition of metadata, structure and computation process of the index and documentation taking into account the different profiles and multicultural/multilingual character of information.

Following this perspective of creating a quantitative index the Cloud Computing community [14,15] and some of the big players have launched some relevant indexes such as: the Service Measurement Index (SMI) by the Cloud Services Measurement Initiative Consortium (CSMIC) consortium, the Cisco Global Cloud Index (GCI) by Cisco, the CSC Cloud Usage index or the VMWare Cloud index.

As final remark, the creation of a quantitative index of QoS indicators seems to be a promising approach because it represents a joint effort to establish a common set of KPI's and it can be applied to the different cloud types and models. Furthermore the participation of big players validates and ensures their potential development. Nevertheless, as other existing quantitative indexes, the structure and the computing process are completely closed and there is not any framework to automatically compute observations. Apart from that it would also be necessary the definition of an API for exposing quality indicators that could be part of existing CAMPs.

2.2. Semantic technologies

The Semantic Web area, coined by Tim Berners-Lee in 2001, has experienced recent times a growing commitment from both academia and industrial areas with the objective of elevating the meaning of web information resources through a common and shared data model (graphs) and an underlying semantics based on different logic formalisms (ontologies). The Resource Description Framework (RDF), based on a graph model, and the Web Ontology Language (OWL), designed to formalize and model domain knowledge, are the two main *ingredients* to reuse information and data in a knowledge-based realm. Thus data, information and knowledge can be easily shared, exchanged and linked to other knowledge-based systems and databases through the use URIs, more specifically HTTP-URIs, with the aim of overcoming the data heterogeneities, lack of standard knowledge representation and interoperability issues. As a practical view of the Semantic Web, the Linked Data initiative emerges to create a large and distributed database on the Web. In order to reach this major objective the publication of information and data under a common data model (RDF) with a specific formal query language (SPARQL) provides the required building blocks to turn the Web of documents into a real database or "Web of Data". Research works are focused in two main areas: (1) production/publishing and (2) consumption of Linked Data. In the first case data quality, conformance, provenance and trust, description of datasets and entity reconciliation issues are becoming major objectives since a mass of amount data is already available through RDF repositories and SPARQL endpoints.

On the other hand, consumption of Linked Data is being addressed to provide new ways of data visualization, faceted browsing, execution of distributed queries and scalable reasoning processes, searching and data exploitation. Currently there is also a growing commitment to publish a vast amount of existing statistical databases. In this sense, the "RDF Data Cube Vocabulary" a W3C Working Draft document, and its predecessor the "Statistical Core Vocabulary" (SCOVO), are shared efforts to represent statistical

[1] http://www.crossover-project.eu/.

Table 1
Features for selecting a semantic-based QoS model.

Feature	Definition	Type
Language	This feature indicates how data is modeled	Word/sentence associated
Reasoning	The model enables some kind of reasoning process	Yes/No and Word/sentence associated
Accessibility	The model can be easily accessed in different context	Likert scale
Adaptability	The model can be easily configured to meet new requirements	Likert scale
Auditability	The model provides a mechanism to know how it is working	Likert scale
Extensibility	The model can be easily extended	Likert scale
Flexibility	The model can be configured on-demand adding/removing features	Likert scale
Interoperability	The model can be integrated with third-parties	Likert scale
Portability	The model can be easily move to different architectures	Likert scale
Usability	The model can be easily configured and exploited	Likert scale
Standards	The model is based in the (re) use of standards and vocabularies (compliance)	Likert scale
Licensing	The type of license	Word/sentence associated
Maturity	The model presents a good level of maturity, development or presence	Likert scale
Update	The model is frequently updated	Likert scale

data in RDF reusing parts (the cube model) of the "Statistical Data and Metadata Exchange Vocabulary" (SDMX). In this view some works are also emerging to mainly publish statistical data following the concepts of the LOD initiative covering statistical analysis of linked data, statistical data publication, survey data publication or quantitative indexes structure and metadata [13] among others.

All the aforementioned works must be considered in order to re-use existing vocabularies and datasets to address the challenges of creating meta-described data, information and knowledge. Mainly semantics allows us to model logical restrictions on data while linked data enables the publication of new data and information under a set of principles to boost their re-use and automatic processing through machine-readable formats and access protocols with the aim of boosting a new wave of professionals [16].

2.3. Big data

Recent times have seen the emergence of new applications to deal with "Big Data" that usually includes the processing of large datasets and vast amounts of data coming from different sources with the objective of extracting "the most of data" to support decision processes. These tools are focused on capturing, curating, managing and processing data in a certain slot of time.

Therefore systems [17] that require real-time, search or high-frequency trading in a certain context such as smart cities, advertising or social networks are moving to this kind of Big Data systems to be able to process large volumes of data in highly scalable and streaming fashion. Existing tools and frameworks use or implement a streaming strategy of partitioning the input data into fixed-size segments as MapReduce-based frameworks do but the main drawback of this approach lies in the latency (it is proportional to the length of the segment plus the overhead required to do the segmentation and initiate the processing of new jobs). In this case the size of the segment is a key-decision to get an optimal data-processing system. Nevertheless new architectures such as the "Lambda Architecture" [18] minimizes this issue adding different layers of processing to operate with data streams in real-time.

The evolving Big Data Community is unleashing the potential of these tools to drive innovation through the creation of new platforms with more and more analysis capabilities that try to fulfill both market and research areas. Forrester [19] has outlined the importance of this new rise of big data as an opportunity to increase corporate knowledge and get competitive advantages with regards to competitors making faster and better decisions. In this sense it seems clear that the use of predictive analytics to find patterns in data represents a new market of opportunities and a real development of a new data-based economy. Nevertheless Forrester also presents a set of requirements that an organization must address: (1) understand data from a variety of sources; (2) create the predictive model; (3) prepare the data; (4) evaluate the model; (5) deploy the model and (6) monitor the effectiveness of the model. Finally they have also created a set of 51 criteria to evaluate the current offering, strategy and market presence of these monitoring tools with the aim of obtaining a quantitative measure of existing large vendors.

Since the core concepts of this review are presented, it is clear that semantic web technologies offer a standard and unified way for representing information and data coming from cloud-based applications. QoS management processes can take advantage of this situation building expert systems that exploit this data to support the aforementioned five key-characteristics of Cloud Computing providing an intelligent and flexible environment for self-managed applications in which both profiles technicians and business users can use semantics and real-time systems as a tool for supporting their decisions.

3. Semantic-based QoS management in cloud systems

In this section a literature review of main ontology-based frameworks for QoS management is presented. After that an empirical evaluation of some selected features, see Table 1, is also outlined to finally present a summary, see Table 2, of the most relevant approaches for semantic-based QoS management.

3.1. Ontology-based frameworks for QoS management

An ontology-based resource description is proposed in [20,21] to solve problems with regards to the difficulty of resource information management, no standard definitions of resource requirements and the difficulty of guaranteeing compatibility of resource allocation. In this sense, there are also works trying to produce a global ontology by merging existing ontologies of resource groups [22]. On the other hand, authors in [23] propose the Semantically-Enhanced Resource Allocator (SERA), a scheduling system using customer requests with the ability of re-scheduling requests based on their priorities and considering advanced reservations.

In [24] a Rule Based Resource Manager is proposed for a cloud hybrid environment with the objective of increasing the scalability (on-demand) of private clouds. This work also sets up the execution time for public and private cloud in order to fulfill requests selected different services. In this case, the methodology is applied to the IaaS layer to access resources on demand enabling the scale up of private clouds with a cost-effective.

The SITIO platform [25] gathers the emerging concepts of SaaS, semantic technologies, Business Process Modeling and Cloud Computing to foster the dramatic evolution of a new platform oriented towards interoperability and cost reduction. SITIO is defined

Table 2
Summary of ontology-based frameworks for QoS management.

Model/Feature	Language	Reasoning	Access.	Adapt.	Audit.	Extens.	Flex.	Interoper.	Port.	Usability	Standards	Licensing	Maturity	Update
SERA [23]	OWL	Y	1	3	4	3	2	4	3	3	4	–	2	2
OReSS [59]	OWL	Y	1	2	3	4	4	2	2	3	4	–	2	2
SITIO [25]	OWL + Rules	Y	1	4	4	3	4	3	4	4	4	–	2	2
Cloud Recommender [26]	–	1	4	3	4	2	2	3	2	4	3	–	2	1
FP7 4WARD [27]	OWL	Y	3	2	4	2	2	3	2	3	3	–	1	1
SRC [28]	OWL	Y	1	3	3	4	2	2	3	3	3	–	2	2
Cloudle [27]	OWL	Y	1	2	4	3	2	2	3	2	3	–	2	1
IRPS [30]	OWL + RDQL	Y	1	4	3	4	3	4	2	3	2	–	1	1
RASIC [31]	–	–	1	4	4	3	4	4	4	3	3	-	1	1
SAMM [33]	OWL + ESPER	Y	1	4	4	4	4	4	4	3	3	–	1	1
QoSMONaaS [34]	OWL + CEP + SLA	Y	1	3	4	4	4	4	3	4	3	–	1	1
IRMOS [39]	–	–	1	3	4	4	4	2	2	2	3	–	2	2
QoS model [40]	WSQM	–	1	4	2	4	4	4	2	2	4	–	1	1
eCloudManager [42]	RDF	Y	1	3	3	3	3	3	3	2	4	–	1	1
mOSAIC [44]	OWL-S + SPARQL	Y	1	4	4	4	4	3	3	4	4	–	1	1
Q-Clouds [45]	MIMO	–	1	3	3	4	4	3	3	3	3	–	1	1
QAComPS [47]	OWL2 + SAWSDL	Y	1	3	3	4	4	4	4	3	4	–	1	1
ServiceRank [52]	–	–	1	3	3	2	3	34	3	2	2	–	2	2
onQoS [53]	OWL + onQoS-QL	Y	1	3	3	3	4	4	4	3	2	–	1	1
QoS & SLAs [48]	–	–	1	4	4	4	4	4	4	4	3	–	2	2
SOA & Dependability [49]	–	–	1	4	4	4	4	4	4	4	3	–	2	2
QoS Taxonomy [50]	–	–	1	3	3	3	3	3	3	3	3	–	1	1

as a platform for reliable, privacy-aware, secure and cost-efficient semantics-based Software-as-a-Service Creation, Integration and Management. Its main contribution lies in a methodology for annotating services using old-fashioned semantic web services techniques. Although the SITIO platform applies semantics to solve interoperability problems it is only focused on the description of web services functionality and capabilities.

A declarative system called CloudRecommender is presented in [26] through an unified and formalized domain model capable of describing infrastructure services such as Amazon, Microsoft Azure, GoGrid, etc. A prototype is also introduced to show the main benefits of this approach: (1) a recommender with the capability of estimating costs across multiple providers, (2) the aid in the selection of cloud services and (3) an user-friendly service interface based on widgets that maps user requirements (form inputs) to available infrastructure services. As future work authors suggest the use of the recommender to support the selection of more cloud service types such as PaaS services including run-time features.

In [27] authors review three concepts developed in the context of the FP7 4WARD project with the objective of demonstrating their potential impact on QoS management: network virtualization (to decouple network from infrastructure and overcome "one-size-fits-all"), generic path semantic resource management (a QoS profile to overcome inadequacies of the traditional layered network model) and in-network management (to incorporate QoS management capabilities in network elements). All of them are novel and promising concepts that are being targeted at handling QoS issues and are supposed to be relevant for enabling a new dynamic, flexible, adaptable and scalable cloud environment.

Authors in [28] aim to provide a suitable service cater to discover consumer service requests including functional requirements and non-functional properties. They propose a service registry model named "SRC" which is an extension of a keyword based service registry model. The SRC is deployed as a cloud application to provide a behavior-aware and QoS aware service discovery storing both semantic descriptors of web services and QoS feedback. This data is processed using a Map/Reduce mechanism. Basically it is a matchmaking service based on the WSDL descriptions taking advantage of using OWL for simple annotations of functional and non-functional properties. The main drawback of this approach lies in the necessity of ensuring synchronized multiple copies of OWL definitions on all nodes.

In [29] a search engine and an architecture for cloud systems (Cloudle) is outlined to semantically look up services according to a user profile. Two ontologies (T-Box and A-Box) have been also designed in order to assist this similarity reasoning process and are used to improve the accuracy of results. The main finding of this study is that an enriched ontology can improve the selection of cloud services. However all concepts, properties, etc. are defined by the authors from the scratch without any reuse of existing standard concepts.

In [30] authors introduce the system Inter-cloud Resource Provisioning System (IRPS) to accomplish the requirements of a customer providing additional resources in a federated cloud system. This system schedules some tasks to allocate resources by using semantics and an inference engine; more specifically they use Sesame and RQL to query over RDF instead of the approach in [23] where Jena is used. Although the approach of running semantics in a federated environment is powerful and the use of RDF can solved interoperability problems, the distributed execution of queries is still under study.

In [31] a framework called Reference Architecture for Semantically Inter-operable Clouds (RASIC) is presented to facilitate the management of inter-cloud components and to provide reliable end to end services that meet SLA requirements. This work tries to capture the concepts and attributes of resources in a cloud environment using semantics to address the problem of semantic interoperability between heterogeneous cooperating clouds.

A cloud computing ontology is proposed in [32] to ease the semantic identification, discovery and access to cloud services. Authors create ontologies and taxonomies trying to capture existing concepts and relationships in a cloud environment. Basically, they are focused on service discovery and selection according to functional and non-functional properties.

A semantic-based monitoring and management system (SAMM) is presented in [33]. This system shows a novel approach to automatic infrastructure scaling, based on the observation of business-related metrics with the objective of offering on-demand resource provisioning capabilities and high flexibility to manage cloud systems. By using ontologies to describe resources and metrics

available for observation, SAMM provides capabilities to express different system architectures and monitoring facilities. The architecture is based on OSGi including a decision-making module based on the Esper event processing engine. The main outcome of this work lies in the possibility of dynamically increasing the amount of resources taking into account both business and technical issues. This tool is also supposed to support cloud stacks such as OpenStack or OpenNebula.

In [34] authors present QoSMONaaS (Quality of Service MONitoring as a Service), a QoS monitoring facility built on top of the SRT-15 platform (a cloud-oriented and CEP-based system). In particular they present the main components of QoSMONaaS: (1) a semantic model; (2) a SLA analyzer; (3) a KPI Meter; (4) a Breach Detector and (5) a Violation Certifier. This work addresses the monitoring problem using a CEP-based approach. Finally, authors also point out the possibility of combining statistical and logical reasoning to make predictions in a QoS aware cloud environment.

Author presents in [35] a three-year research about QoS in Service Oriented Architectures in which his main contributions [36–38] consists in: (1) design and development of a real-time SOA with QoS negotiation and management capabilities; (2) design and development of a QoS registry to support the QoS management of adaptive service-oriented real-time applications including functional behaviors and to predict the future performance based on data already collected; (3) a methodology to support QoS management for virtualized services deployed in Service Oriented Infrastructures (SOIs) and (4) design and development of a service-oriented, flexible and adaptable middleware for QoS configuration and management of Wireless Sensor Networks (WSNs). Although this work perfectly tackles some of the challenges in a cloud environment it is mainly focused on the intermixing of real-time techniques with SOAs, whilst other aspects typical of SOA-based approaches to software design such as semantics are not provided.

In [39] authors present their experiences while developing the IRMOS platform (a real-time cloud computing infrastructure developed in the context of the IRMOS European Project). The main outcome of this work lies in the advance in the state-of-the-art in SLAs and, in particular, the expression of requirements in the language of the application domain: user's needs are dynamically translated to infrastructure requirements in fine grained SLAs and a real-time method has been designed to evaluate and mitigate violations in SLAs. Although it is a promising platform there is no information about how this platform models the SLAs, resources or the QoS features.

Authors introduce in [40] a QoS model to provide the appropriate ground for QoS engineering in Service Oriented Computing (SOC). The model is focused on emerging QoS features related to the dynamics of service environments such as user mobility and context of application services. In this case, the use of semantics emerges to represent and enrich QoS features making use of the Web Service Quality Model (WSQM).

In [41] author makes a review of the marriage between clouds and agents discussing how this can be done and which scientific areas and issues must be involved to carry out research works for producing intelligent cloud services. The main focus of this article is the convergence between multi-agent systems that need a reliable infrastructure and cloud computing that needs intelligent software with dynamic, flexible and autonomous behavior.

A study of the semantic technologies for enterprise cloud management is presented in [42]. Authors present the suite eCloudManager to address the topics of data integration, collaborative documentation and annotation, intelligent information access and analytics. One of the main conclusions is that a RDF approach can improve data integration in highly heterogeneous and changing enterprise environments in which complex event processing and reasoning can be key-processes to enable a smart environment.

Following a similar approach to [40] authors present in [43] a QoS-aware service composition that enables the fulfilling of complex user tasks while meeting QoS constraints. One challenging issue in this topic is the selection of the best set of services (NP-hard problem) to compose and meeting global QoS constraints defined by the user. The main outcome of this work is an algorithm guided by a heuristic that provides the appropriate ground for QoS composition in dynamic service environments.

The mOSAIC [44] platform for multiple clouds uses ontologies and semantics for providing a unified description of cloud components, interfaces, SLAs, QoS, APIs and requirements. The main objective of this platform is to enable a semantic framework in which reasoning processes can be carried out as well as SPARQL queries for discovering, selecting and matchmaking services. Semantic technologies are applied to describe services using OWL-S.

The Q-Clouds system [45] is a QoS-aware control framework that tunes (applying an on-line control feedback to build a MIMO-multi-input–multi-output model) resource allocations to mitigate performance interference effects. The main contribution of this work is a system to provide assurance in performance issues applying a MIMO model for capturing interference effects and driving a closed loop resource management controller.

Authors in [46] introduce a trust model for efficient reconfiguration and allocation of computing resources satisfying user requests. This model collects and analyzes reliability based on historical information gathered from a cloud data center. Thus the model is provided and validated against different datasets but no semantics is used in any process.

QAComPS, a quality-aware federated computational semantic web service for computational modelers, is presented in [47] to provide a federated QABroker based on ontologies and making use of OWL2 features. Basically they perform a matchmaking reasoning process for discovering and selecting services according to a set of characteristics to meet user requirements. A SAWSDL interface is also published to transfer semantic annotation to/from the QAComPS service and QABroker.

Authors present in [48] an approach (a three-step method) to map SLAs and QoS requirements of business processes. They formalize the capabilities and requirements to finally compare them with the objective of detecting performance or reliability gaps. This method is evaluated as a dynamic technique to accommodate and improve the performance of individual services deployed in a grid or a cloud computing infrastructure. In [49] an application of a SLA management is proposed to address dependability in a SOA lifecycle. Authors describe the concepts and formalisms of each lifecycle stage (Model, Assemble, Deploy, and Manage). The final objective of this approach is to meet user requirements offering optimized levels of dependability.

A taxonomy of QoS management and Service Selection Methodologies in cloud computing is presented in [50]. This survey reviews the current status of QoS in web services and purposes a taxonomy to model the resources on cloud computing environments. The main objective of this work is to provide a taxonomy for service selection in grid computing, SOA and cloud computing as well as define QoS characteristics such as user preferences, QoS source, context, web service attributes, semantic descriptions of web services, fuzzy preferences, roles, etc. Finally the taxonomy is tested using the Analytic Hierarchy Process (AHP) technique to make decisions about service selection.

In [51] authors make a proposal for inter-cloud exchanges (XMPP) and cataloging of computing resources (ontology). They perform queries via a SPARQL endpoint to select the components to be exchanged. The main aim of this work is to provide a federated cloud environment but it is still an early stage of development.

ServiceRank [52] is a new ranking method which considers quality of service aspects (such as response time and availability)

Table 3
Features for selecting a technique of datastream processing.

Feature	Definition	Type
Data model	This feature indicates how data is represented	Word/sentence associated
Programming Language	The programming language used to implement this tool/technique	Word/sentence associated
Reasoning	The tool enables some kind of reasoning process	Yes/No and Word/sentence associated
Formats	The input formats that the technique can process	Word/sentence associated
Data integration	The tool enables the consumption of different datasources	Yes/No
Architecture	The type of architecture for data processing	Word/sentence associated
Programming Model	The type of programming model for data processing	Word/sentence associated
API	The tool provides and API to manage all processes	Yes/No
Accessibility	The tool can be easily used in different context	Likert scale
Adaptability	The tool can be easily configured to meet new requirements	Likert scale
Auditability	The tool provides a mechanism to know how it is working	Likert scale
Elasticity	The tool can be adjusted its resource consumption to meet demand	Likert scale
Extensibility	The tool can be easily extended	Likert scale
Flexibility	The tool can be configured on-demand adding/removing features	Likert scale
Interoperability	The tool can be integrated with third-parties	Likert scale
Portability	The tool can be easily move to different architectures	Likert scale
Usability	The tool can be easily configured	Likert scale
Security	The tool can be used under a protocol for secure communication	Likert scale
Standards	The tool is based in the use of standards (compliance)	Likert scale
Off-line processing	The technique enables off-line/batch processing	Yes/No
Real-time processing	The technique enables real-time processing	Yes/No
Storage	The database to store data	Word/sentence associated
Query	The formal query language to access to results	Yes/No
Dashboard	The technique provides a graphical tool to manage processes, export data, etc.	Likert scale
Licensing and pricing	The type of license	Word/sentence associated
Maturity	The tool has reached a good level of maturity or development	Likert scale
Presence	The tool has reached a good level of presence in existing products	Likert scale
Update and evolution	The tool is frequently updated	Likert scale
Certification	The tool provides courses and other methods to learn its use	Likert scale
Financial support	The tool is supported for investors/venture capital, etc.	Likert scale

as well as social perspectives of services (such as how they invoke each other via service composition). Authors present this new algorithm that has been implemented on SOAlive, a platform for creating and managing services and situational applications. The main outcome of this work is the combination of QoS metrics with social aspects but no semantics is applied in any of the process to select services.

Other QoS ontology, onQoS, is presented in [53]. Authors make a study of the impact of Semantics for the management of QoS requirements in service-based applications and they also present the aforementioned ontology, its role for specifying service requirements and the onQoS-QL language to support queries for service discovery. Finally, in [54] authors present a proposal for an ontology-driven approach to self-management of cloud application platforms using the MAPE-K reference model and another ontology-based framework for policy-driven governance in cloud application platforms is also presented in [55].

3.2. Summary and evaluation

This section presents a summary/questionnaire of the most relevant aforementioned semantic approaches for QoS management with the aim of establishing an intuitive but empirical comparison among the different QoS models. The evaluation of "quality" in ontologies is not a mere process as some works [56,57] have already demonstrated. That is why we summarize these models according to a set of features, see Table 1. This list is inspired in previous works but not exclusive in order to remark characteristics to take into account when we want to re-use or extend some of the existing QoS models. Each feature is evaluated following the next approaches:

- Open ended questions using a word/sentence associated to the feature.
- Multiple choice using the Likert scale [58] value: 1-Strongly disagree, 2-Disagree, 3-Neither agree nor disagree, 4-Agree and 5-Strongly agree.
- Closed ended questions with a Yes (Y)/No (N) value.
- Finally the symbol "-" is used to represent those unknown/missing/not applicable features.

At a first glance and according to results in Table 2 it seems that most of the QoS models based on semantics are not publicly available. Although in most of them some uses of OWL or SPARQL are very promising to define profiles, restrictions, SLAs, ECA rules, etc. there is no way of re-using them avoiding one of the main principles of semantic web technologies. In most of cases they have been developed within the execution of some research project but, at the moment, are not up-to-date. Apart from that the number of QoS models implies that there is no consensus in building a common set of QoS indicators and, maybe, this is one of the reasons that prevents the real deployment of quality-based methods. On the other hand, the use of the Likert scale to evaluate some features presents some disadvantages it can serve as a guide to ranking different approaches. As a final remark, semantic technologies have been applied to QoS but violating some of the basic principles: (1) re-use of existing vocabularies and (2) build on the top of a common and shared understanding. Further steps in this area should be the creation of a common set of quality indicators and their representation using semantic web technologies with the aim of enabling a better re-use of knowledge and dissemination. Although it is not easy to reach an agreement in a broad field like QoS some minimal common definitions should be established. In this sense, the initiatives presented in Section 2.1 can be a step-forward to boost the application of QoS in real environments.

4. Techniques for datastream processing

In this section a literature review of main datastream processing techniques and tools is presented. After that an empirical evaluation of some selected features, see Table 3, is also outlined to finally present a summary, see Tables 4 and 5, of the most relevant approaches for semantic-based QoS management.

Table 4
Summary of selected tools and techniques for data stream processing (I).

Tool/Feature	IEEE 1451	EEML [62]	SWE [61]	SSN [66]	Semantic Streams [64]	Storm [18]	Impala [68]	SPARK [69]	Druid [73]	S4 [72]	MapR [74]
Data model	TEDS[a]	EEML	SensorML	OWL/RDF	–	Topology	–	–	–	–	–
Programming Language	–	Java	–	-	–	Java and JVM-based languages	Java	Scala, Java & Python.	Java	Java	Java
Reasoning	–	–	–	Y	Y	N	N	N	N	N	
Formats	1451.x	EEML	SensorML	RDF syntax[b]	–	–	Sequence Files (Snappy, Avro, Gzip, Bzip, LZO etc.)	Hadoop Input Formats	–	–	Hadoop Input Formats
Data integration	Y	Y	Y	Y	Y	N	N	N	N	N	N
Architecture	S/D[c]	S/D	S/D	S/D	S/D	D	D	D	D	D	D
Programming Model	Service	Service	Service	–	–	MapReduce[d]	MapReduce	MapReduce	MapReduce	EP[e]	MapReduce
API	1451.x	Pachube	SOS[f] and SPS[g]	–	Y	Y	Y	Y	Y	Y	Y
Accessibility	4	4	4	5	2	5	5	5	5	5	5
Adaptability	4	4	5	5	4	5	5	5	4	5	5
Auditability	3	3	3	3	3	5	5	4	3	4	4
Elasticity	4	4	4	5	4	5	5	5	4	5	4
Extensibility	4	4	4	5	4	5	5	5	4	5	5
Flexibility	4	5	5	5	4	5	5	5	4	5	5
Interoperability	4	5	5	5	4	4	4	4	3	4	4
Portability	4	5	3	5	4	5	5	5	3	5	5
Usability	3	4	4	1	4	4	5	4	3	4	4
Security	3	3	3	3	3	4	5	3	3	3	4
Standards	5	4	5	5	3	3	4	3	3	4	4
Off-line processing	3	3	3	–	–	2	4	4	4	3	5
Real-time processing	3	4	4	–	4	5	54	5	5	5	4
Storage	3	3	3	3	3	4 (DFS[h])	HDFS	HDFS	–	DFS	HDFS
Query	–	Y	Y	Y (SPARQL)	Y	Y (Trident)	Hive SQL	Hive SQL (iif Shark)	Y (SQL-based)	–	Hive SQL/Pig
Dashboard	3	4	4	1	4	2	5	3	3	3	4
Licensing and pricing	Open	Open	Fee on TGAR[i]	W3C	–	EPL v1.0	Apache	Apache	GPL	Apache	M3 Edition is free, M5 & M7 commercial licenses
Maturity	4	4	4	2	4	5	5	4	4	5	4
Presence	4	4	5	2	2	4	5	4	5	5	5
Update and evolution	4	4	5	4	2	5	5	5	5	4	5
Certification	4	4	5	2	2	3	5	4	3	4	3
Financial support	4	4	5	2	3	4	5	4 (Apache Spark)	4	4 (Apache S4)	5

[a] Transducer electronic data sheet.
[b] RDF/XML normative syntax, Turtle, N3, etc.
[c] Sensor/Distributed (cluster).
[d] It is based on MapReduce in the sense of batch processing.
[e] Event processing.
[f] Sensor Observation Service.
[g] Sensor Planning Service.
[h] Distributed file system.
[i] Total gross annual revenue.

Table 5
Summary of selected tools and techniques for data stream processing (II).

Tool/Feature	Streaming SPARQL [6]	C-SPARQL [7]	CQELS [60]	EP-SPARQL [8]	WebPIE [77]	QueryPIE [78]	SAOR [79]	Pig SPARQL [83]	Hadoop SPARQL [84]	H2RDF [86]
Data model	OWL/RDF	OWL/RDF		OWL/RDF	–	–	OWL/RDF	–	–	–
Programming Language	Java	Java	Java	Java	Java	Java	Java	Java	Java	Java
Reasoning	N	N	N	Y	Y (RDFS/OWL-Horst backward)	Y (RDFS/OWL-Horst backward)	Y (OWL/RDFS)	N	N	N
Formats	RDF syntax	RDF syntax	RDF syntax	RDF syntax	Hadoop Input Formats	Hadoop Input Formats	–	Hadoop Input Formats	Hadoop Input Formats	RDF syntax
Data integration	Y	Y	Y	Y	N	N	N	N	N	Y
Architecture	C[a]	C	D	C	D	D	D	D	D	D
Programming Model	MapReduce	Events	Events	Events	MapReduce	MapReduce	–	MapReduce	MapReduce	MapReduce
API	N	Y	Y	Y	N	N	N	N	N	N
Accessibility	3	4	4	4	4	4	4	3	2	3
Adaptability	4	3	4	4	3	4	4	3	3	3
Auditability	2	2	3	3	3	3	3	2	2	3
Elasticity	3	2	4	4	3	4	4	4	4	4
Extensibility	3	2	4	4	4	4	3	3	3	4
Flexibility	2	2	3	4	3	3	3	4	3	4
Interoperability	4	3	4	4	4	4	4	3	3	4
Portability	4	3	4	4	4	4	4	4	3	4
Usability	2	2	3	4	4	4	4	4	3	4
Security	3	2	3	3	3	3	3	3	3	3
Standards	4	4	4	4	4	4	4	3	3	4
Off-line processing	4	2	2	2	4	4	4	4	4	4
Real-time processing	2	4	4	4	2	2	2	2	2	3
Storage	M[b]	M	–	M	HDFS	HDFS	–	HDFS	HDFS	HDFS
Query	Y (SPARQL)	Y (SPARQL)	Y (CQELS language)	Y (SPARQL)	Y (SPARQL)	Y (SPARQL)	–	Y (SPARQL/Pig)	Y (SPARQL)	Y (SPARQL)
Dashboard	N	N		N	N	N	N	N	N	N
Licensing and pricing	–	CC BY-NC-SA 3.0	LGPLv3.0	GPL (Etalis Engine)	Apache	Apache	–	–	–	GPL
Maturity	2	3	4	4	4	4	4	3	2	3
Presence	2	3	3	4	2	2	3	2	2	2
Update and evolution	2	2	4	4	2	2	3	3	2	3
Certification	1	1	3	3	2	2	2	3	2	3
Financial support	1	1	3	3	2	2	3	3	–	3

[a] Centralized.
[b] Memory.

The increasing number of applications and services on the cloud is creating an across computer platform to support new methods in different disciplines such as life sciences or engineering disciplines. The stream model for data intensive monitoring and analysis can potentially benefit these applications but the processing of data in real time must ensure that data is gathered continuously (24/7), large volumes of data are properly addressed taking into account that data sources are distributed and is often not feasible to store all data for processing at a later time, thereby, requiring real-time analysis.

On the other hand, a sensor can be defined as a data source which can produces a sequence of data items over time [60]. This definition leads us to a new environment in which any device can continuously produce data and we can take advantage of this information to deliver advanced services on monitoring distinct domains such as traffic, logistics or supply chain management, etc. Nevertheless this environment requires new techniques and models to deal with this vast amount of data making the integration and interoperability among applications possible.

In the particular context of Cloud Computing each resource, application or service can be seen as a data sensor producing data and information about their current status that can be processed in an automatic way with the objective of enabling self-management cloud systems and applications. Nevertheless some problems arise to manage a stream of data: storage, semantics, inference and querying, among others. As authors present in [60] traditional data stream management systems (DSMS) assumes stream data has a bigger impact in query performance than static metadata. In the case of Linked Data this situation is more dramatic due to the existence of different vocabularies and the possibility of having new RDF triples in execution time that are not modeled by static metadata. Existing triple stores do not manage this situation and assume RDF data as static. In a cloud system environment this assumption is no longer hold and a method to query and make decisions in real-time processing stream data must be provided in order to deliver a reactive knowledge-based system. More specifically in the QoS management context, the need of dynamic adaptation of resources is dramatically growing up due to the necessity of adjusting costs and optimizing the reservation of cloud resources. Thus the QoS management can be seen as a motivating scenario to apply stream reasoning over data coming from cloud resources.

First efforts to address data stream challenges were made in the IEEE 1451 definition, the Radiation Detection Standards (ANSI N42), the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) [61] or the Extended Environments Markup Language [62] (EEML). They followed a domain-specific approach that cannot be easily applied in other contexts. A standard with a broader application was the Sensor Web Enablement standard which was implemented by the 52° North project, the NASA/JPL project and the European Space Agency among others. Efforts with regards to Linked Data such as "SensorBase" enables the publication of sensor data via HTTP POST as well as Pachube that offers a Restful interface for querying stream real-time sensor data in different formats such as JSON or EEML. Nevertheless the common and main drawback of these approaches lies in the lack of semantic descriptions to describe data streams.

In [63] authors proposed the use of RDFa to annotate ontological concepts and properties to SWE by using XLink. Thus the sensor information can support semantic functionalities and new services can be implemented on the top of RDF. Furthermore Whitehouse et al. launched the SemanticStreams project [64] based on Prolog rules to allow user queries on the semantics of sensor and in [65] authors propose to use OWL to represent data stream and provide a semantic context in which the composition of applications could be easily done. However they assume [60] that semantic descriptions are already available and have enough quality. Finally the W3C Semantic Sensor Network Incubator Group has proposed the SSN ontology [66] to answer the need of a domain-independent and end-to-end model for sensing application by merging sensor focused (e.g. SensorML), observation-focused (e.g. observation and measurement) and system focused views. It also covers subdomains which are sensor-specific such as the sensing principles and capabilities and can be used to define how a sensor will perform in a particular context. Nevertheless further steps imply their real adoption in a Linked Sensor Data context to bridge the new era of Internet of Things and Internet of Services fostering their potential adoption in the OGC community.

Following the review of approaches to mix semantic technologies and stream data processing is presented with special focus on storage, stream processing, analytics and some applications:

- In the case of storage, RDF triple stores such as Sesame, Apache Jena (Joseki and Fuseki), RISC-3X, YARS2, Oracle Semantic, OWLim, OpenLink Virtuoso have gained importance in recent times due to the deployment of the Linked Open Data initiative. Most of them are based on a SQL-backend and offer distinct services such as basic reasoning processes and a query service via a SPARQL endpoint. The main focus of these repositories is the scalability in terms of the number of triples they can manage. They are adequate when data do not change frequently and a better performance can be reached applying well-known techniques of indexing. Moreover the current initiative of NoSQL databases brings the opportunity of using new representation and retrieving models (key-value, document oriented, column-family or graph) to deal with the vast amount of data in contexts such as Social Media or, like in our case of study, data coming from dozen of applications.

  Furthermore Apache Hadoop and related projects built on top of this Map/Reduce framework such as Apache Pig, Hive, Mahout, Cassandra or Zookeeper enable a distributed processing environment of large datasets across clusters of computers using simple programming models. The storage and processing models of large datasets is not a mere question and it must be carefully planned to avoid scalability problems. In this context there are also some projects providing data flows such as Storm [18] (a distributed and fault-tolerant real-time computation environment), most of these efforts are boosted by the main social media sites.
- There is also a growing set of both commercial and open tools that claims to provide large scale and distributed data stream processing, inspired in Apache Drill [67]. All of them use their own internal data models, cluster processing technique, formal query languages and different kinds of predictive analytics. In this context Impala [68], SPARK [69], Sparrow [70], Shark [71], S4 [72], Druid [73] or MapR [74] technologies to name a few deliver such predictive services on big data through off-line and real-time queries. Usually they also take advantage of: (1) programming languages with functional and parallel capabilities such as Scala, Clojure, Erlang or Python; (2) NoSQL stores and (3) MapReduce-based frameworks. Finally an advanced dashboard to manage jobs is commonly provided as a graphical interface for controlling the distributed execution and the results of predictive analysis functions.
- Data stream management systems (DSMS) represents other way to manage data streams GraphgCQ, Amazon/Cougar, Aurora, Gigascope, Hancock, Niagara, OpenCQ, Stream, Stream Mill, Tapestry, Tribeca, Streambase, Coral8, Apama or Truviso among others were implemented to overcome the limitation of traditional databases when continuous data updates appear. They are based on different data models and their implicit semantics is not based on static restrictions such as the ACID principles. The main objective, in these cases, lies in getting a better

performance when huge amounts of data should be managed. According to [60] and taking into account there is no benchmark available about their support of Linked Data streams these management systems are not suitable for processing RDF streams.

- StreamingSPARQL [6] and C-SPARQL [7] purpose extensions to the existing SPARQL language to register queries in a time frame or sliding windows. StreamSPARQL presents a rather simple query evaluation model without taking into account performance issues whilst C-SPARQL enables an execution framework built of top of existing stream data management and triple storage systems. In that sense C-SPARQL is supposed to be more stable than StreamSPARQL and it also provides a service for splitting the continuous queries into two groups: static and dynamic which are merged through orchestration service binding the required facts when new RDF triples arise. Authors also outline [75] all features that stream reasoning should address such as the use of continuous semantics instead of "static" semantics. In [60] authors present the Continuous Query Evaluation over Linked Streams (CQELS) project to unify data coming from both the Linked Open Data cloud and sensor stream data. They focus on continuous queries that is, queries that are registered in the system, and executed every time new data matches their criteria considering the queries themselves as input parameters. Finally a complete study of stream processing in the cloud is also depicted in [76].
- On the other hand, the CEP community is concerned with timely detection of compound events in streams of simple events. Nevertheless event processing cannot combine streams with background knowledge and cannot perform reasoning tasks. Although the use of event processing engines is widely accepted and technology such as Drools Fusion is well-known and used in production environments the use of semantic web technologies can perfectly manage background knowledge, mix different data streams and execute reasoning processes. That is why the conjunction of event processing for dealing with a vast amount of data and semantics to manage several streams and background knowledge can improve the exploitation of diverse data streams. In this context EP-SPARQL [8] has been designed as an extension of the existing SPARQL language to support complex events and stream reasoning. The main challenge of this work is to take advantage of real-time data, and recognizes important situations of interest in a timely fashion. Authors conclude that EP-SPARQL specifies complex events by temporarily situating real-time streaming data, and uses background ontologies to enable stream reasoning.
- The problem of querying large datasets and performing semantic reasoning is being addressed in WebPIE [77] and QueryPIE [78] which were prototypes implemented in the context of the LarKC project. These approaches show how to use a Map/Reduce framework for building an inference engine. WebPIE is focused on the calculation of the transitive closure of OWL semantics, more specifically RDFS and OWL-Horst. This inference engine proposes a scalable technique to parallelize OWL Horst forward inference over 100 billion of RDF triples. The main drawback of this approach lies in the necessity of executing the whole reasoning process when new facts arise that is why it cannot be considered adequate for stream reasoning. On the other hand and with the objective of processing dynamic data QueryPIE offers a backward inference engine to execute queries over large datasets performing a reasoning process before retrieving results. In this context of reasoning over large datasets, in [79–81] a system to compute the closure of RDF graphs is also presented but it only supports a fragment of OWL Horst to enable efficient materialization and (live) query processing on the Linked Data realm.
- As we have seen in the previous works the implementation of SPARQL extensions to be executed in a distributed environment is currently a good approach to deal with a mass of data and provide a scalable environment for the Semantic Web. In this sense, authors in [82] make an implementation of a SPARQL distributed engine to split data and queries into different nodes and to get local-optimizations. In [83] authors propose a translation of SPARQL queries into Pig Latin for the scalable processing of complex queries on very large RDF datasets. They introduce PigSPARQL, a system which processes the SPARQL queries as Map/Reduce jobs, and make an evaluation using the benchmark SP2Bench (a specific SPARQL performance benchmark) getting pretty good results. Following a similar approach HadoopSPARQL [84] is a tool that allows user to submit multiple queries at the same time. These queries are handled by an algorithm that is in charge of creating and distributing sub queries using Map/Reduce jobs. In [85] authors present a work to store RDF in HDFS (Hadoop File System) and an algorithm to ask SPARQL queries. H2RDF [86], Jena-HBase [87] and the proposal in [88] are similar works trying to manage large RDF datasets and execute distributed SPARQL queries.
- The use of the R statistics package is widely accepted in the Big Data community to make decisions, predictions or analysis and perform statistical methods on large datasets. In the context of SPARQL, there is a specific package, `r-sparql`, to work with R that enables the connection to existing RDF endpoints from the R processor. The analysis and exploitation of large datasets is not easy to solve and data mining frameworks as Weka or Apache Mahout offer scalable methods and algorithms to detect patterns, etc. in data. The main drawback of these tools lies in the lack of integration with semantic technologies. Another problem arises when a large RDF dataset must be processed, mainly due to the URIs.
- Regarding streaming applications, in [89] two applications are described and used as benchmarks in the data mining domain. The first one, CluStream, is a cluster evolving data streams [90] that groups similar objects or data points from a given set into clusters. They address the problem of clustering data when data is arriving from continuous streams and changing over the time. Firstly, mini-clusters are created to compute statistical information. After that a modified version of the K-means algorithm is applied to create the final output of clusters. The second algorithm studies the frequency of occurring items in a distributed data stream, Approx-Freq-Counts [91]. Two parameters are defined: support and error, each node observes the frequencies of item sets in each stream and periodically sends this information to a parent node. These two approaches are considered to solve the problem of resource provisioning for data stream applications in virtualized or cloud environments finding dynamic patterns when data arrive. The resource provisioning algorithm correctly converges to the optimal CPU allocation based on the data arrival rate and computational needs. The algorithm identifies over-flow and under-flow conditions and converges to the same level, irrespective of the initial allocation. The main advantage of this approach is that the system can automatically tune itself based on resource needs.
- Finally in [92] authors present the ElasticStream system that dynamically allocates computational resources on the cloud in an elastic manner for a data stream processing application. To minimize the charges for using the Cloud environment while satisfying the SLA, they formulate a linear programming problem to optimize the costs as a trade-off between the application's latency and charges. A system is also implemented to assign or remove computational resources dynamically on the top of the data stream middleware. Authors also conclude that their approach could save up to 80% of the costs while maintaining the

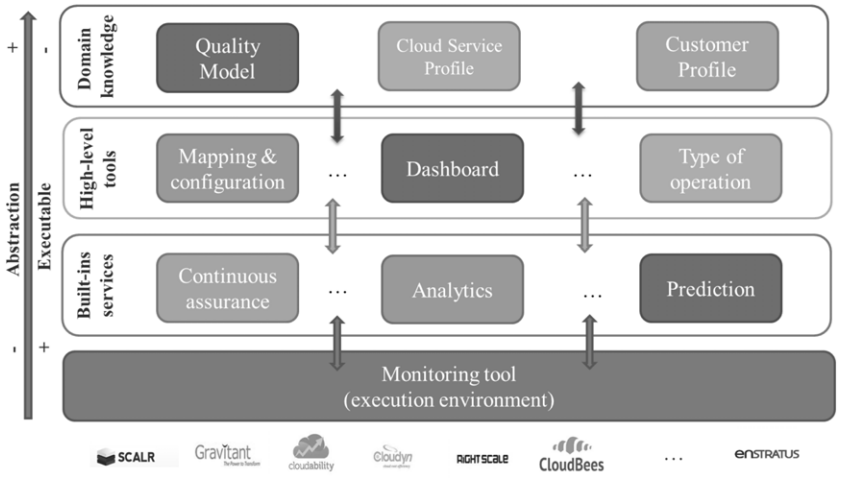

Fig. 1. QoS Framework.

application's latency in comparison to a naïve approach. Nevertheless they remain as future work the prediction of data rate and the inclusion of new features in the platform to a better stream data management. This approach does not use any semantic feature to address the needs of managing a cloud environment and it is just based on an optimization algorithm to select adequately the allocation of resources in the cloud.

4.1. Summary and evaluation

This section presents a summary of the most relevant aforementioned techniques with the aim of establishing a fact-sheet and a tool for making decisions when a technique has to be selected for some use. We have created a set of features, see Table 3, to list those references that present some evidence for validating its application and contribution to datastream and big data processing. This list is not exclusive but remarks main characteristics to take into account when a monitoring tool/technique is required. Each feature is evaluated following the next approaches:

- Open ended questions using a word/sentence associated to the feature, for instance "MapReduce" or "SQL".
- Multiple choice using the Likert scale [58] value: 1-Strongly disagree, 2-Disagree, 3-Neither agree nor disagree, 4-Agree and 5-Strongly agree.
- Closed ended questions with a Yes (Y)/No (N) value.
- Finally the symbol "-" is used to represent those unknown/missing/not applicable features.

According to Tables 4 and 5 it seems that there are two main approaches:

- Datastream tools and techniques (with a high-level of maturity) that offer a distributed MapReduce-based framework (with a high-level language) on the top of some kind of NoSQL store to provide off-line and real-time datastream processing.
- Semantic-based frameworks that offer a better interoperability and integration but less performance and stability (in terms of maturity and support).

In fact the introduction of semantics in existing "Big Data" tools is becoming a major challenges due to vendors perfectly know the advantages of providing a more standardized solution. Nevertheless, at the moment, a MapReduce-based solution such as Storm or Impala is clearly a good option to deal with a vast amount of data but sacrificing semantics. Finally there is a new "Dataland" of start-ups creating "Big Data" solutions, this financial support represents a growing and manifest commitment to boost a new data-based economy. Nevertheless the conjunction or addition of semantics in existing tools can dramatically improve their adoption and uptake for business users since technicians perfectly know how to use these big data frameworks to implement their solutions.

5. A framework for semantic-based QoS management and datastream processing in cloud systems

This section introduces an architecture and execution environment to put together semantics and existing techniques for dealing with data streams in real-time. It is based on the architecture defined in [18] by Nathan Marz that defines a general data system as a system that runs arbitrary functions on arbitrary data. This definition follows the next equation $query = f(all\ data)$ which is the basis of all systems. The Lambda Architecture defines then a clear set of principles to build robust and scalable data systems obeying the aforementioned equation. Basically three main design principles can be found:

- Human fault-tolerance. The system is unsusceptible to data loss or data corruption because at scale it could be irreparable.
- Data immutability. Data is stored in a raw form to be immutable and for perpetuity.
- Re-computation. Following the two previous principles it is always possible to re-compute results by performing a function on the raw data.

In order to depict a semantic-based framework for QoS management that can leverage existing datastream processing tools such as those presented in Section 4 and improve/boost the user experience (technicians and business users), we present in Fig. 1 the building blocks to implement a semantic "WatchDog" [93] cloud pattern. This framework is built upon any cloud to manage and control QoS indicators. As any knowledge-based system it comprises a knowledge layer in which ontologies are used to represent a QoS model and the quality functions to be deployed on the monitoring tool. The formalization of a QoS model is not an easy task since a lot of tries have been already done, see Section 3. Nevertheless the effort of the CSMIC consortium in the creation of the SMI index, see Section 2.1, represents a good starting point to unify quality indicators and metrics. Thus the proposed framework can be seen as an implementation of the SMI index but modeling the index (structure and computation process) using semantic web technologies such as SPARQL and the RDF Data Cube vocabulary. The second layer outlines some required capabilities to support lowering and lifting processes [94] between the abstract QoS model and the real execution in the monitoring tool. Since SPARQL queries can

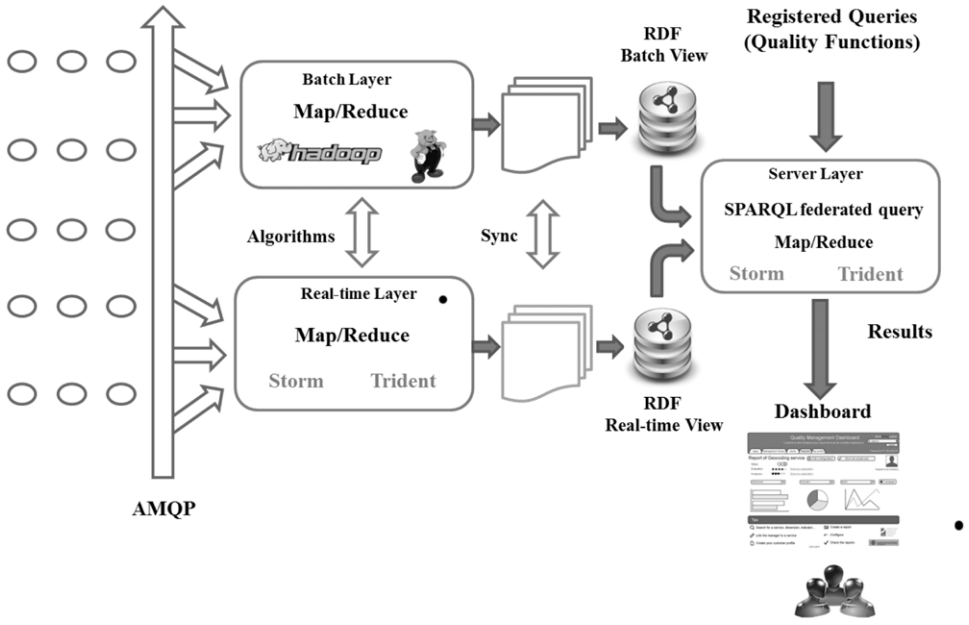

Fig. 2. A semantic-based Lambda Architecture for real-time processing of diverse datastreams.

be translated into SQL (or Pig [83]) queries it is obvious than any execution framework supporting a SQL-based language (or Apache Hadoop) can execute our semantic model generating new observations in RDF. Furthermore a dashboard is also provided to manage quality indicators. In this case tools such as CubeViz enable the automatic visualization of RDF Data Cube observations easing the creation and customization of the graphical interface.

On the other hand Fig. 2 outlines the layers of the Lambda architecture (in this case the monitoring tool) with an adaptation to support semantics.

- **Batch layer.** It contains immutable, constantly growing dataset stored on a distributed file system like HDFS. This layer receives as input the raw data coming from a queue. Data is then computed by means of some function and results are finally exposed as batch views. As an extra step and with the aim of supporting the use of semantics these final views are promoted as RDF and thus SPARQL queries can be executed on the computed results. This approach of adding an extra layer to ease queries over batch views has been already addressed using SQL as a formal query language, e.g. SploutSQL. In this case SPARQL and RDF are selected to support the implicit creation of queries from a semantic model. This layer is usually implemented using a MapReduce-based framework such as Apache Hadoop or a more high-level framework such as Apache Pig. As a final remark the off-line execution of the algorithms or functions eases the pre-computation of results when large dataset processing is required.
- **Serving layer.** The responsibility of this layer is to manage both batch and real-time views with the aim of providing a way of querying and merging both views and populate results to be consumed by a third-service. In this case, this layer is in charge of executing the SPARQL queries (using a federated extension [95] such as SPARQL-FedX [96]) and publish the results as RDF. At a first glance this job does not require random writers but must support batch updates and random reads. In this case the implementation of joining results is also designed as a Storm/Trident topology to enable and distributed real-time updates.
- **Speed layer.** It deals with new data to compensate or decrease the latency of the batch layer. Functions are deployed on some stream processing system such as Storm, S4 or Spark and results are finally exposed as RDF. The functionality provided by this layer is mainly the same as the batch-layer but removing the latency and affording a real-time query system. Once data is also processed and available in the batch view a synchronization process must remove data from the real-time views. In this case Apache Zookeeper can be used to keep synchronization in a distributed system.

The "semantized" Lambda Architecture using RDF and SPARQL on top of the batch and real-time views enables the possibility of handling the complexity of Big Data systems by defining a clear set of principles. More specifically the use of semantics serves to integrate data under a common data model that can be queried using a formal query language such as SPARQL. Furthermore immutability, human fault-tolerance and re-computation basic principles that can be easily adopted with the Hadoop platform. Finally and depending on the real-time requirements of the system some parts can be omitted and integrated in a further stage.

6. Discussion and future challenges

Despite the growing interest in Cloud Computing and the hype of this paradigm for the creation of new era of applications, a real advanced cloud management environment is far from being fully developed. There are many open issues to be solved and technology to ease the transition from traditional developments and applications to a cloud-based environment is still under development. With regards to QoS, there are a lot of initiatives and efforts trying to model and manage functional and non-functional properties in an intelligent fashion. Nevertheless the lack of standards for unifying information and data is preventing the deployment of advanced techniques for QoS management. In the case of semantic technologies, works in different areas are emerging to solve interoperability and integration problems in distributed environments. More specifically, the creation of knowledge-based systems applying semantic-based techniques as stream reasoning and CEP are currently being developed to deal mainly with Big Data problems

in the context of social media or e-Government. Following a list of questions/answers are provided to discuss the current status and future challenges in the topics covered in this paper:

- Which dimensions and metrics should be taken into account to manage QoS in Cloud systems?

  There are a big variety of QoS dimensions to ensure in a cloud system. The methods to ensure reliability, security and trust should be modeled and discovered in automatic ways. In this specific case it is also required to take into account user feedback to evaluate the real quality and trust of a service. Moreover, depending on the cloud layer, specific QoS characteristics should be defined to collect the requirements of each particular case. Currently QoS approaches are mainly focused on web service discovery and selection but new ranking methods [97] and reactive control systems taking into QoS features should be deployed to provide an intelligent cloud infrastructure.
- Which is the "best" approach to tackle the QoS management in Cloud systems?

  From a policy-making perspective the use of quantitative indexes is a widely accepted practice to compile in just one value a set of indicators. In this sense the on-going works presented in Section 2.1 are a nice starting point, more specifically the SMI index seems to be a clear candidate to assess the quality of cloud services (XaaS). Nevertheless it is necessary to find a method to: (1) integrate different data sources; (2) model the index structure and its computation and (3) provide proper documentation in a multilingual environment with different user profiles and intentions. In this sense, semantics can help to address these requirements through a common and shared data model with implicit support for multilingual documentation. In fact if a quantitative index is modeled using semantics it can be deployed and transformed to an existing monitoring tool since RDF, OWL and SPARQL semantics is clearly defined and enable transformations to other formal models or languages such as SQL.
- How QoS can leverage semantics?

  According to Section 3 the big variety of ontologies, OWL models, etc. that have been designed in recent years imply a tangled set of options that should be unified to provide an unique view of what QoS should be and cover. Apart from that the use of semantics is not clear, in some cases reasoning processes are used to discover services but others just define an ontology as a proposal to provide a formal model without any real application. A clear semantic-based architecture should be defined containing: (1) the adequate definitions of functional and non-functional properties and (2) different perspectives (technical and business).

  Furthermore semantics can help to increase the reliability in Cloud Computing providing the building blocks and models for an advanced, standardized and inter-operable QoS management. In the same way, Cloud Computing can help semantic technologies to be more scalable and flexible making use of the web as infrastructure to create large-scale data-intensive batch applications.
- Are semantic technologies able to enable scalable predictive analytical processes?

  With regards to Semantic Web and reasoning, there is a growing community trying to provide technology for supporting intelligent systems in the new Web of Data. As a consequence the necessity of dealing with Big Data problems and data coming from different sources is stimulating the creation of new approaches to reuse existing technology such as Apache Hadoop in the context of querying large datasets. Therefore the main application of the Semantic Web principles lies in the unification of data and the execution of reasoning processes to validate data and infer new facts. Nevertheless, the existing logic formalisms available in OWL such as DL, FOL, F-Logic, etc. do not seem to be a solution to tackle the challenge of modeling dynamic domains that is why some works regarding Continuous Semantics are emerging.
- How Linked Data can help to a better QoS management experience?

  Currently this initiative has been successfully applied to information retrieval systems or in the creation of rich user interfaces. Nevertheless, the expectations of linking different datasets to enrich information are growing as a manner for delivering more intelligent services. To apply Linked Data in the Cloud Computing environment we should ensure that any resource to be monitored has an URI, its data is coded into RDF according to a formal model, an API or endpoint is accessible for fetching data using pulling, pushing or triggering techniques and the methods for graph processing and reasoning are efficient and scalable under real-time constraints.
- How can I select a monitoring tool? Should I design and implement a new one from the scratch?

  Following the review in Section 4 there is already a proven set of tools to analyze big data. Most of them are based on an internal model, a formal query language and a NoSQL-based storage to perform queries or analysis in real-time. Nevertheless an extra layer of semantics based on standards such as RDF or SPARQL is still an open issue. An effort to expose processed data as a SPARQL endpoint it is completely possible since a similar approach have been reached using SQL (e.g. SploutSQL). In that sense a good approach can be to re-use the effort of existing monitoring tools but adding a RDF layer, see Section 5, to leverage the two main advantages of semantics: integration and interoperability.
- Which are the key-factors to select a monitoring tool?

  Depending on the context and the requirements of the problem (real-time, predictive analytics, type of license, etc.) some different characteristics must be evaluated but, at least, we should ensure if the monitoring tool is able to provide the next services: (1) stream processing and incremental calculation of statistics; (2) paralleling processing; (3) a dashboard for data exploration or integration with other existing visualization tools; (4) data import capabilities; (5) extensibility; (6) use of standards and (6) in general, a reporting tool to extract summaries. In fact, the selection of a monitoring tool does not differ from any other kind of software but the two first points must be carefully evaluated. Finally the selection must be aligned with a business/research strategy.
- Which is the next big thing in QoS management, Semantics and Big Data?

  In the case of QoS management, if we assume we are able to access different key performance indicators and perform some analysis then the scenario presented in Section 1.1, Cloud Brokerage, can be efficiently address. Nevertheless the complete automation of a broker service is still an utopia since human-validation is required to make strategic decisions. Furthermore it is necessary to define a QoS API, maybe using the SMI indicators, to be able to implement a real Cloud Quality Management Platform. For instance, SalesForce offers[2] some RSS feeds to check the status of their services, a similar approach should be followed for other providers to boost the user experience and trust and create a real cloud service market.

  On the other hand semantic technologies and Linked Data are now focused on addressing some challenges in data quality, provenance, trust, large dataset processing, entity reconciliation, searching or inference to name a few.

[2] http://trust.salesforce.com.

Finally the use of Big Data techniques and tools is already in the market and an evolving community is generating more and more approaches to provide more faster and scalable analysis processes. The design of algorithms taking advantage of a distributed environment, the possibility of integrating diverse data, the use of standards and the creation of more enriched visualization tools are some open issues that must be addressed to improve and bring these tools to public at large boosting a new data-based economy.

7. Conclusions and future work

An important body of work has been done during the last five years regarding the deployment of cloud infrastructures, services and applications. As a result, numerous SaaS, PaaS and IaaS providers can be found as well as cloud management platforms to ease the management of these infrastructures. In the same way, the definition of QoS features in SOA has became a major research topic with a lot of derived works trying to model and manage applying different approaches as ontologies, rule based systems, etc. The arrival of new techniques for processing real-time large and diverse datastreams from different data sources to deliver ontology-based expert systems is very promising and the Cloud Computing and the QoS management areas must take advantage of these methods to exploit cloud infrastructures saving costs, being efficient and greener. Further steps require the definition of a QoS model for cloud infrastructures and the design of scalable infrastructures for connecting and managing resources on the cloud. The market and business opportunities of this new realm are encouraging the research and the collaboration between the academic and industrial areas fostering both the existing open issues and the generated know-how among all interested parties.

Acknowledgments

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7-PEOPLE-2010-ITN) under grant agreement no. 264840 and developed in the context of the workpackage 4 and more specifically under the project "Quality Management in Service-based systems and Cloud Applications".

References

[1] P. Mell, T. Grance, The NIST Definition of Cloud Computing, Tech. Rep. 800-145, National Institute of Standards and Technology (NIST), Gaithersburg, MD (September).
[2] M.C. Huebscher, J.A. McCann, A survey of autonomic computing degrees, models, and applications, ACM Comput. Surv. 40 (3) (2008) 7:1–7:28.
[3] J. Conejero, L. Tomás, B. Caminero, C. Carrion, Multilevel SLA-based QoS support in grids, in: Proceedings of the 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications, ISPA'12, 2012, pp. 239–246.
[4] R.C. Palacios, E. Fernandes, M. Sabbagh, A. de Amescua Seco, Human and intellectual capital management in the cloud: software vendor perspective, J. UCS 18 (11) (2012) 1544–1557.
[5] J.M. Pedersen, M.T. Riaz, J.C. Junior, B. Dubalski, D. Ledzinski, A. Patel, Assessing measurements of QoS for global cloud computing services, in: Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, DASC'11, 2011, pp. 682–689.
[6] A. Bolles, M. Grawunder, J. Jacobi, Streaming SPARQL extending SPARQL to process data streams, in: Proceedings of the 5th European Semantic Web Conference on the Semantic Web: Research and Applications, ESWC'08, 2008, pp. 448–462.
[7] D.F. Barbieri, D. Braga, S. Ceri, M. Grossniklaus, An execution environment for C-SPARQL queries, in: Proceedings of the 13th International Conference on Extending Database Technology, EDBT'10, 2010, pp. 441–452.
[8] D. Anicic, P. Fodor, S. Rudolph, N. Stojanovic, EP-SPARQL: a unified language for event processing and stream reasoning, in: Proceedings of the 20th International Conference on World Wide Web, WWW'11, 2011, pp. 635–644.
[9] W. Fan, A. Bifet, Mining big data: current status, and forecast to the future, SIGKDD Explor. Newsl. 14 (2) (2013) 1–5.
[10] A. Rodríguez-González, J. Torres-Niño, G. Hernández-Chan, E. Jiménez-Domingo, J.M. Alvarez-Rodríguez, Using agents to parallelize a medical reasoning system based on ontologies and description logics as an application case, Expert Syst. Appl. 39 (18) (2012) 13085–13092.
[11] A. Milenkoski, A. Iosup, S. Kounev, K. Sachs, P. Rygielski, J. Ding, W. Cirne, F. Rosenberg, Cloud Usage Patterns: A Formalism for Description of Cloud Usage Scenarios, Tech. Rep., SPEC Research Group — Cloud Working Group, 2013.
[12] N. Kephart, Six must-have cloud management features, July 2012.
[13] J.M.A. Rodríguez, J. Clement, J.E.L. Gayo, H. Farhan, P. Ordoñez De Pablos, Publishing Statistical Data following the Linked Open Data Principles: The Web Index Project, IGI Global, 2013, pp. 199–226. URL http://www.igi-global.com/chapter/publishing-statistical-data-following-linked/77207/.
[14] M. Maiya, S. Dasari, R. Yadav, S. Shivaprasad, D. Milojicic, Quantifying manageability of cloud platforms, in: Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, CLOUD'12, 2012, pp. 993–995.
[15] M. Klems, D. Bermbach, R. Weinert, A runtime quality measurement framework for cloud database service systems, in: QUATIC, 2012, pp. 38–46.
[16] R.C. Palacios, J.L. Sánchez-Cervantes, G. Alor-Hernández, A.R. González, Linked data: perspectives for IT professionals, IJHCITP 3 (3) (2012) 1–12.
[17] R. Akerkar, Big Data Computing, first ed., Taylor & Francis Group/CRC Press, 2013.
[18] N. Marz, J. Warren, Big Data: Principles and Best Practices of Scalable Realtime Data Systems, first ed., Manning Publications Co., 2013.
[19] M. Gualtieri, The Forrester WaveTM: Big Data Predictive Analytics Solutions, Q1 2013, Report, Forrester Inc., 2013, URL http://www.forbes.com/sites/danwoods/2011/10/21/big-data-technology-evaluation-checklist/.
[20] A.M. Pernas, M.A.R. Dantas, Using ontology for description of grid resources, in: Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications, HPCS'05, 2005, pp. 223–229.
[21] D. Armstrong, K. Djemame, Towards quality of service in the cloud, in: Proc. of the 25th UK Performance Engineering Workshop, 2009.
[22] J.G.R.C. Lopes, A.C.M.A. Melo, M.A.R. Dantas, C.G. Ralha, A proposal and evaluation of a mechanism for grid ontology merge, in: Proceedings of the 20th International Symposium on High-Performance Computing in an Advanced Collaborative Environment, HPCS'06, 2006, pp. 2–.
[23] J. Ejarque, M.d. Palol, I. Goiri, F. Julia, J. Guitart, J. Torres, R.M. Badia, Using semantics for resource allocation in computing service providers, in: Proceedings of the 2008 IEEE International Conference on Services Computing—Volume 2, SCC'08, 2008, pp. 583–587.
[24] R. Grewal, P. Pateriya, A rule-based approach for effective resource provisioning in hybrid cloud environment, in: S. Patnaik, P. Tripathy, S. Naik (Eds.), New Paradigms in Internet Computing, in: Advances in Intelligent Systems and Computing, vol. 203, Springer, Berlin Heidelberg, 2013, pp. 41–57.
[25] F. Garcia-Sanchez, E. Fernandez-Breis, R. Valencia-Garcia, E. Jimenez, J.M. Gomez, J. Torres-Niño, D. Martinez-Maqueda, Adding semantics to software-as-a-service and cloud computing, W. Trans. on Comp. 9 (2) (2010) 154–163.
[26] M. Zhang, R. Ranjan, S. Nepal, M. Menzel, A. Haller, A declarative recommender system for cloud infrastructure services selection, in: GECON, 2012, pp. 102–113.
[27] J. Carapinha, R. Bless, C. Werle, K. Miller, V. Dobrota, A. Rus, H. Grob-Lipski, H. Roessler, Quality of service in the future internet, in: Kaleidoscope: Beyond the Internet? — Innovations for Future Networks and Services, 2010 ITU-T, 2010, pp. 1–8.
[28] H. peng Chen, S. chong Li, SRC: a service registry on cloud providing behavior-aware and QoS-aware service discovery, in: SOCA, 2010, pp. 1–4.
[29] J. Kang, K. Sim, Cloudle: an ontology-enhanced cloud service search engine, in: D. Chiu, L. Bellatreche, H. Sasaki, H.-f. Leung, S.-C. Cheung, H. Hu, J. Shao (Eds.), Web Information Systems Engineering — WISE 2010 Workshops, in: Lecture Notes in Computer Science, vol. 6724, Springer, Berlin Heidelberg, 2011, pp. 416–427.
[30] V. Nelson, V. Uma, Semantic based resource provisioning and scheduling in inter-cloud environment, in: Recent Trends in Information Technology (ICRTIT), 2012 International Conference on, 2012, pp. 250–254. http://dx.doi.org/10.1109/ICRTIT.2012.6206823.
[31] R. Buyya, R. Ranjan, R.N. Calheiros, InterCloud: utility-oriented federation of cloud computing environments for scaling of application services, in: Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing—Volume Part I, ICA3PP'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 13–31.
[32] G.O. Cortázar, J.J. Samper-Zapater, F. García-Sánchez, Adding Semantics to Cloud Computing to Enhance Service Discovery and Access, Tech. Rep., Spanish Ministry of Economy, 2012.
[33] Scalable Resource Provisioning in the Cloud Using Business Metrics, 2011.
[34] G. Cicotti, L. Coppolino, R. Cristaldi, S. D'Antonio, L. Romano, QoS monitoring in a cloud services environment: the SRT-15 approach, in: M. Alexander, P. D'Ambra, A. Belloum, G. Bosilca, M. Cannataro, M. Danelutto, B. Martino, M. Gerndt, E. Jeannot, R. Namyst, J. Roman, S. Scott, J. Traff, G. Vallée, J. Weidendorfer (Eds.), Euro-Par 2011: Parallel Processing Workshops, in: Lecture Notes in Computer Science, vol. 7155, Springer, Berlin Heidelberg, 2012, pp. 15–24.
[35] G. Anastasi, Quality of service management in service oriented architectures, 2011.
[36] T. Cucinotta A. Mancina G.F. Anastasi G. Lipari L. Mangeruca R. Checcozzo F. Rusina.

[37] K. Konstanteli, D. Kyriazis, T. Varvarigou, T. Cucinotta, G. Anastasi, Real-time guarantees in flexible advance reservations, in: Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference—Volume 02, COMPSAC'09, 2009, pp. 67–72.
[38] T. Cucinotta, G. Anastasi, L. Abeni, Respecting temporal constraints in virtualised services, in: COMPSAC (2), 2009, pp. 73–78.
[39] T. Cucinotta, S. Gogouvitis, K. Konstanteli, SLAs in virtualized cloud computing infrastructures with QoS assurance, in: Proceedings of the International Workshop on eContracting in the Clouds, co-located with the eChallenges 2011 Conference, 2011.
[40] N.B. Mabrouk, N. Georgantas, V. Issarny, A semantic end-to-end QoS model for dynamic service oriented environments, in: Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems, PESOS'09, 2009, pp. 34–41.
[41] D. Talia, Cloud Computing and Software Agents: Towards Cloud Intelligent Services, in: WOA, 2011, pp. 2–6.
[42] P. Haase, T. Mathäß, M. Schmidt, A. Eberhart, U. Walther, Semantic technologies for enterprise cloud management, in: Proceedings of the 9th International Semantic Web Conference on The Semantic Web—Volume Part II, ISWC'10, 2010, pp. 98–113.
[43] N.B. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, V. Issarny, QoS-aware service composition in dynamic service oriented environments, in: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware, Middleware'09, 2009, pp. 7:1–7:20.
[44] G. Cretella, B. Di Martino, V. Stankovski, Using the mOSAIC's semantic engine to design and develop civil engineering cloud applications, in: Proceedings of the 14th International Conference on Information Integration and Web-based Applications & Services, IIWAS'12, 2012, pp. 378–386.
[45] R. Nathuji, A. Kansal, A. Ghaffarkhah, Q-clouds: managing performance interference effects for QoS-aware clouds, in: Proceedings of the 5th European Conference on Computer Systems, EuroSys'10, 2010, pp. 237–250.
[46] H. Kim, H. Lee, W. Kim, Y. Kim, A Trust Evaluation Model for QoS Guarantee in Cloud Systems, IJGUC 3 (1).
[47] P.M. Dew, S. Nizamani, QAComPS: a quality-aware federated computational semantic web service for computational modellers.
URL http://weblidi.info.unlp.edu.ar/worldcomp2011-mirror/SWW3720.pdf.
[48] V. Stantchev, C. Schröpfer, Negotiating and enforcing QoS and SLAs in grid and cloud computing, in: Advances in Grid and Pervasive Computing, Springer, 2009, pp. 25–35.
[49] V. Stantchev, M. Malek, Addressing dependability throughout the soa life cycle, Services Computing, IEEE Transactions on 4 (2) (2011) 85–95.
[50] A. Dastjerdi, R. Buyya, A taxonomy of QoS management and service selection methodologies for cloud computing (2011).
URL http://www.cloudbus.org/papers/QoSTaxonomyCloud2011.pdf.
[51] D. Bernstein, D. Vij, Using semantic web ontology for intercloud directories and exchanges, in: International Conference on Internet Computing, 2010, pp. 18–24.
[52] Q. Wu, A. Iyengar, R. Subramanian, I. Rouvellou, I. Silva-Lepe, T. Mikalsen, Combining quality of service and social information for ranking services, in: Proceedings of the 7th International Joint Conference on Service-Oriented Computing, ICSOC-ServiceWave'09, 2009, pp. 561–575.
[53] G. Damiano, E. Giallonardo, E. Zimeo, onQoS-QL: a query language for QoS-based service selection and ranking, in: E. Nitto, M. Ripeanu (Eds.), Service-Oriented Computing — ICSOC 2007 Workshops, 2009, pp. 115–127.
[54] R. Dautov, D. Kourtesis, I. Paraskakis, M. Stannett, Addressing self-management in cloud platforms: a semantic sensor web approach, in: Proceedings of the 2013 International Workshop on Hot Topics in Cloud Services, HotTopiCS'13, 2013, pp. 11–18.
[55] D. Kourtesis, I. Paraskakis, A registry and repository system supporting cloud application platform governance, in: ICSOC Workshops, 2011, pp. 255–256.
[56] M. d'Aquin, A. Schlicht, H. Stuckenschmidt, M. Sabou, Ontology modularization for knowledge selection: experiments and evaluations, in: DEXA, 2007, pp. 874–883.
[57] M. Sabou, M. Fernández, E. Motta, Evaluating semantic relations by exploring ontologies on the semantic web, in: NLDB, 2009, pp. 269–280.
[58] G. Albaum, The Likert scale revisited, Journal of Market Research Society 39 (1997) 331–348.
[59] H. Yoo, C. Hur, S. Kim, Y. Kim, An ontology-based resource selection service on science cloud, in: D. Ślęzak, T.-H. Kim, S.S. Yau, O. Gervasi, B.-H. Kang (Eds.), Grid and Distributed Computing, in: Communications in Computer and Information Science, vol. 63, 2009, p. 221.
[60] D. Le-Phuoc, J.X. Parreira, M. Hausenblas, M. Hauswirth, Unifying Stream Data and Linked Open Data, Tech. Rep., DERI, 2010.
[61] O. Consortium, Sensor Web Enablement (May 2013).
URL http://www.opengeospatial.org/projects/groups/sensorweb.
[62] EEML, Extended Environments Markup Language (EEML) (May 2013). URL http://www.eeml.org.
[63] A. Sheth, C. Henson, S.S. Sahoo, Semantic sensor web, IEEE Internet Computing 12 (4) (2008) 78–83.
[64] K. Whitehouse, F. Zhao, J. Liu, Semantic streams: a framework for composable semantic interpretation of sensor data, in: Proceedings of the Third European conference on Wireless Sensor Networks, EWSN'06, 2006, pp. 5–20.
[65] E. Bouillet, M. Feblowitz, Z. Liu, A. Ranganathan, A. Riabov, F. Ye, A semantics-based middleware for utilizing heterogeneous sensor networks, in: Proceedings of the 3rd IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS'07, 2007, pp. 174–188.
[66] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, V. Huang, K. Janowicz, W.D. Kelsey, D. Le Phuoc, L. Lefort, M. Leggieri, H. Neuhaus, A. Nikolov, K. Page, A. Passant, A. Sheth, K. Taylor, Ontology paper: the SSN ontology of the W3C semantic sensor network incubator group, Web Semant. 17 (2012) 25–32.
[67] M. Hausenblas, J. Nadeau, Apache drill: interactive Ad-Hoc analysis at scale, Big Data.
[68] C. Inc., Cloudera impala: real-time queries in apache hadoop (May 2013). URL http://www.cloudera.com/content/cloudera/en/products/cdh/impala.html.
[69] M. Zaharia, T. Das, H. Li, S. Shenker, I. Stoica, Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters, in: Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing, USENIX Association, 2012, pp. 10–10.
[70] K. Ousterhout, P. Wendell, M. Zaharia, I. STO-ICA, Sparrow: Scalable Scheduling for Sub-Second Parallel Jobs, Tech. Rep. UCB/EECS-2013-29, EECS Department, University of California, Berkeley, 2013.
[71] R.S. Xin, J. Rosen, M. Zaharia, M.J. Franklin, S. Shenker, I. Stoica, Shark: SQL and rich analytics at scale, in: SIGMOD Conference, 2013, pp. 13–24.
[72] L. Neumeyer, B. Robbins, A. Nair, A. Kesari, S4: distributed stream computing platform, in: Data Mining Workshops, ICDMW, 2010 IEEE International Conference on, IEEE, 2010, pp. 170–177.
[73] F. Yang, E. Tschetter, G. Merlino, N. Ray, X. Léauté, D. Ganguli, Druid: A Real-time Analytical Data Store.
[74] M. Inc., MapR: Apache Hadoop Solutions For Big Data, May 2013. URL http://www.mapr.com/.
[75] E.D.V. Jeff, Z. Pan, Stream Reasoning For Linked Data. Tutorial at SemTech 2012, 2012. URL http://streamreasoning.org/events/sr4ld2011.
[76] N. Backman, R. Fonseca, U. Çetintemel, Managing parallelism for stream processing in the cloud, in: Proceedings of the 1st International Workshop on Hot Topics in Cloud Data Processing, HotCDP'12, 2012, pp. 1:1–1:5.
[77] J. Urbani, S. Kotoulas, J. Maassen, F. van Harmelen, H.E. Bal, WebPIE: a web-scale parallel inference engine using MapReduce, J. Web Sem. 10 (2012) 59–75.
[78] J. Urbani, F. van Harmelen, S. Schlobach, H. Bal, QueryPIE: backward reasoning for OWL horst over very large knowledge bases, in: Proceedings of the 10th International Conference on The Semantic Web—Volume Part I, ISWC'11, 2011, pp. 730–745.
[79] A. Hogan, A. Harth, A. Polleres, Scalable authoritative OWL reasoning for the web, Int. J. Semantic Web Inf. Syst. 5 (2) (2009) 49–90.
[80] J. Umbrich, A. Hogan, A. Polleres, S. Decker, Improving the recall of live linked data querying through reasoning, in: Proceedings of the 6th International Conference on Web Reasoning and Rule Systems, RR'12, 2012, pp. 188–204.
[81] J. Umbrich, M. Karnstedt, A. Hogan, J.X. Parreira, Freshening up while staying fast: towards hybrid SPARQL queries, in: Proceedings of the 18th International Conference on Knowledge Engineering and Knowledge Management, EKAW'12, 2012, pp. 164–174.
[82] J. Huang, D.J. Abadi, K. Ren, Scalable SPARQL querying of large RDF graphs, PVLDB 4 (11) (2011) 1123–1134.
[83] A. Schätzle, M. Przyjaciel-Zablocki, G. Lausen, PigSPARQL: mapping SPARQL to Pig Latin, in: Proceedings of the International Workshop on Semantic Web Information Management, SWIM'11, 2011, pp. 4:1–4:8.
[84] C. Liu, J. Qu, G. Qi, H. Wang, Y. Yu, Hadoopsparql: a hadoop-based engine for multiple sparql query answering, in: Proceedings of th 9th Extended Semantic Web Conference, 2012.
[85] M. Farhan Husain, P. Doshi, L. Khan, B. Thuraisingham, Storage and retrieval of large RDF graph using Hadoop and MapReduce, in: Proceedings of the 1st International Conference on Cloud Computing, CloudCom'09, 2009, pp. 680–686.
[86] N. Papailiou, I. Konstantinou, D. Tsoumakos, N. Koziris, H2RDF: adaptive query processing on RDF data in the cloud, in: Proceedings of the 21st International Conference Companion on World Wide Web, WWW'12 Companion, 2012, pp. 397–400.
[87] V. Khadilkar, M. Kantarcioglu, B.M. Thuraisingham, P. Castagna, Jena-hbase: a distributed, scalable and effcient rdf triple store, in: International Semantic Web Conference, Posters & Demos, 2012.
[88] J.J.H. Maindonald, Data Analysis and Graphics Using R [Electronic Resource]: An Example-Based Approach, Vol. 10, Cambridge University Press, 2007.
[89] S. Vijayakumar, Q. Zhu, G. Agrawal, Dynamic resource provisioning for data streaming applications in a cloud environment, in: CloudCom, 2010, pp. 441–448.
[90] C.C. Aggarwal, J. Han, J. Wang, P.S. Yu, A framework for clustering evolving data streams, in: Proceedings of the 29th International Conference on Very Large Data Bases—Volume 29, VLDB'03, 2003, pp. 81–92.
[91] S. Schneider, H. Andrade, B. Gedik, A. Biem, K.-L. Wu, Elastic scaling of data parallel operators in stream processing, in: Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing, IPDPS'09, 2009, pp. 1–12.
[92] A. Ishii, T. Suzumura, Elastic stream computing with clouds, in: Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, CLOUD'11, 2011, pp. 195–202.

[93] C.C. Patterns, Watchdog: high availability with unreliable compute nodes, 2013. URL http://cloudcomputingpatterns.org/?page_id=278.
[94] M.G. Rodríguez, J.M.Á. Rodríguez, D.B. Muñoz, L.P. Paredes, J.E.L. Gayo, P.O. de Pablos, Towards a practical solution for data grounding in a semantic web services environment, J. UCS 18 (11) (2012) 1576–1597.
[95] N.A. Rakhmawati, J. Umbrich, M. Karnstedt, A. Hasnain, M. Hausenblas, Querying over federated SPARQL endpoints — a state of the art survey, CoRR abs/1306.1723.
[96] A. Schwarte, P. Haase, K. Hose, R. Schenkel, M. Schmidt, FedX: a federation layer for distributed query processing on linked open data, in: ESWC (2), 2011, pp. 481–486.
[97] S.K. Garg, S. Versteeg, R. Buyya, A framework for ranking of cloud computing services, Future Generation Comp. Syst. 29 (4) (2013) 1012–1023.

Dimitrios Kourtesis holds a B.Sc. degree in Computer Science and an M.Sc. degree in Software Engineering and Telecommunications from the University of Sheffield, and is currently studying for a Ph.D. in Computer Science. Before joining SEERC as a research associate in 2006 he was working in the electronic media industry for a period of six years. His primary expertise is in the areas of Semantic Web technologies for Web service description and discovery, formal methods for service testing and monitoring, governance and quality assurance in cloud application platforms, and ontology-driven information systems. He has been involved in a number of research projects pertaining to these areas and has published and served as reviewer in related conferences and journals.

Jose María Alvarez Rodríguez holds a Ph.D. (2012) in e-Procurement, Linked Data and Semantics by the University of Oviedo. From 2005 to 2010 he worked at the R&D Department within CTIC Foundation. He has also participated in more than 15 research projects and has published more than 40 publications and research works. He also held a position as part-time Assistant Professor from 2008 to 2012 at the Department of Computer Science within the University of Oviedo. He has been also rewarded with a HPC2-Europe Transnational Access Programme grant at SARA (Netherlands) and now holds a position as Marie Curie ER at SEERC (Greece) within the RELATE-ITN FP7 project.

Iraklis Paraskakis is a Senior Research Officer at the South East European Research Centre (SEERC), and coordinator of the Information & Knowledge Management research cluster. He is also Senior Lecturer in the Department of Computer Science at CITY College, Greece, and Associate Lecturer in the Department of Informatics at the Hellenic Open University. He holds a Ph.D. in Information Technology and Education from the Open University (UK), and an M.Sc. in Analysis, Design and Management of Information Systems from London School of Economics. His research interests are in the areas of educational informatics, information systems, and knowledge management. He has a number of publications in related conferences and journals, and has participated in several successfully completed and on-going EU projects in these areas.