



The SOLID architecture for real-time management of big semantic data[☆]



Miguel A. Martínez-Prieto^{a,*}, Carlos E. Cuesta^b, Mario Arias^c, Javier D. Fernández^{a,d}

^a DataWeb Research, Department of Computer Science, University of Valladolid, E.U. Informática (Segovia) & E.T.S. Informática (Valladolid), Spain

^b VORTIC3 Research, E.T.S. Ingeniería Informática, Rey Juan Carlos University, Madrid, Spain

^c Insight Centre for Data Analytics, NUI Galway, Ireland

^d Ontology Engineering Group (OEG), Technical University of Madrid, Spain

HIGHLIGHTS

- We propose an architecture (SOLID) for managing big semantic data in real-time.
- Specific big data and real-time responsibilities are isolated in dedicated layers.
- A dynamic pipe-filter solution is introduced for addressing query responsibilities.
- SOLID leverages RDF/HDT features to obtain the most compressed representations.
- The SOLID prototype performs competitive respect to the most prominent triplestores.

ARTICLE INFO

Article history:

Received 31 August 2013

Received in revised form

26 September 2014

Accepted 8 October 2014

Available online 22 October 2014

Keywords:

Tiered architecture

Big semantic data

Real-time

RDF/HDT triplestores

ABSTRACT

Big Data management has become a critical task in many application systems, which usually rely on heavyweight batch processes to manage such large amounts of data. However, batch architectures are not an adequate choice for designing real-time systems in which data updates and reads must be satisfied with very low latency. Thus, gathering and consuming high volumes of data at high velocities is an emerging challenge which we specifically address in the scope of innovative scenarios based on semantic data (RDF) management. The Linked Open Data initiative or emergent projects in the Internet of Things are examples of such scenarios. This paper describes a new architecture (referred to as SOLID) which separates the complexities of Big Semantic Data storage and indexing from real-time data acquisition and consumption. This decision relies on the use of two optimized datastores which respectively store historical (big) data and run-time data. It ensures efficient volume management and high processing velocity, but adds the need of coordinating both datastores. SOLID proposes a 3-tiered architecture in which each responsibility is specifically addressed. Besides its theoretical description, we also propose and evaluate a SOLID prototype built on top of binary RDF and state-of-the-art triplestores. Our experimental numbers report that SOLID achieves large savings in data storage (it uses up to 5 times less space than the compared triplestores), while provides efficient SPARQL resolution over the Big Semantic Data (in the order of 10–20 ms for the studied queries). These experiments also show that SOLID ensures low-latency operations because data effectively managed in real-time remain small, so do not suffer Big Data issues.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

Although Big Data is one of the buzzwords in the current technological landscape, there is no consensus in its definition. A widely accepted meaning says that Big Data is “*when the size of the data itself becomes part of the problem*” [1]. However, this definition basically states the most obvious dimension of Big Data: **volume**, but obviates **velocity** and **variety**. The convergence of these *three*

[☆] A preliminary and significantly less detailed version of this research appeared in Proc. of ECSA'2013 (Cuesta (2013) [12]).

* Corresponding author.

E-mail addresses: migumar2@infor.uva.es (M.A. Martínez-Prieto), carlos.cuesta@urjc.es (C.E. Cuesta), mario.arias@insight-centre.org (M. Arias), jfergar@infor.uva.es (J.D. Fernández).

V's [2] comprises the primitive (and most accepted) Big Data characterization:

Volume: large amounts of data are gathered and stored in massive datasets created for different uses and purposes. According to the statistics from the International Data Corporation,¹ the amount of digital data in 2012 is ten times higher than the existing in 2007. Thus, data production grows faster than computational power, so storage is the first scalability challenge in Big Data management since preserving the data must be our first responsibility. It is worth noting that storage decisions impact in data retrieval, processing and analysis.

Velocity: means how data flow, at high rates, in increasingly distributed scenarios. Two data streams can be distinguished. On the one hand, streams of new data (potentially generated in different ways and sources) being progressively integrated into existing (big) datasets. According to the IBM report [3], 2.5 quintillion bytes of data are produced every day. On the other hand, we consider (potentially large) streams of query results produced from user requests. Thus, velocity means how fast data is produced, demanded and served in real-time.

Variety: refers to the various degrees of structure (or lack thereof) within the Big Data [4]. Zikopoulos et al. [5] state that 80% of the world data is unstructured and these are growing at 15 times the rate of structured information. This dimension is motivated by the fact that Big Data may integrate multiple sources: e.g. any kind of sensor network, web logs, government data, social networks, etc. Obviously, each source describes its own semantics, resulting in different data schemes which are hardly integrable in a single model. Thus, Big Data variety demands a logical model enabling effective data integration regardless of their structure.

Although these three V's provide a good Big Data description, the volume dimension deserves to be revised to increase the scope of our work. One could think in terabytes, petabytes or exabytes when talking about Big Data, but few gigabytes may be enough to collapse an application running on a mobile device or even in a personal computer. Thus, the term Big Data is used, in this paper, to refer to any dataset whose size is greater than the resources available for its storage and processing in a given computational configuration.

Any kind of architecture designed for Big Data management [6] must consider all the dimensions above. However, its intended purpose restricts which one is first addressed. For instance, storage could be more critical for a mobile application than for another one running on a powerful server, whereas data retrieval velocity is a priority for a real-time application but not so critical for a batch-based process.

In this paper, we focus on an architecture able to gather, store, and expose (for consumption) Big Data in real-time. However, computing arbitrary functions on an arbitrary dataset in real-time is a daunting problem [7], and the desired architecture must address two main challenges:

1. It must store and expose big datasets containing *historical data*. This challenge demands a datastore able to scale with volume requirements and also providing fast data retrieval.
2. It must gather and expose *run-time data*. In this case, a datastore must be optimized to process frequent new data (incremental updates) that must be consumable in real-time.

Although each challenge can be independently addressed, their efficient integration remains a challenge by itself because scalability and real-time Big Data management have conflicting needs. Our current proposal tackles the problem of reconciling *historical*

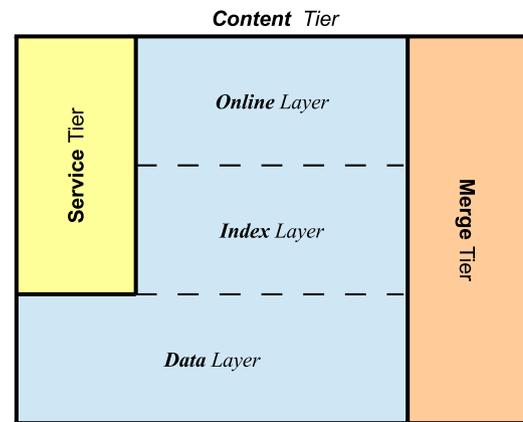


Fig. 1. The SOLID architecture.

and *real-time* data requirements with a divide-and-conquer strategy where both responsibilities are treated separately, providing specific mediators between them. These mediators must consider that the full dataset comprises run-time and historical data, so both datastores can be coordinately accessed for resolving potential data consumption functions in an efficient manner.

Our proposed architecture is designed on a strong assumption: *In the presence of heterogeneous data, the more data are integrated and managed under a common model, the more interesting knowledge may be generated*, increasing the resulting dataset value. In fact, **value** is usually referred to as the fourth V of Big Data. Thus, as explained before, effective data integration demands a single logical data model which allows various levels of structure to be managed within a dataset. To do so, we choose a graph-based model because it can reach higher levels of *variety* before data become unwieldy, allowing more data to be linked and queried together [8]. Among all graph-based models, we use the Resource Description Framework (RDF) [9], one of the cornerstone of the Semantic Web [10], whose basic principles are materialized by the emergent Web of Data [11]. RDF reaches all its potential when it is used in conjunction with vocabularies providing data semantics, i.e., giving meaning to each different schema, thereby facilitating their efficient integration under a common umbrella. Although this improves *variety* management, the use of RDF as logical model also restricts how data are finally structured, stored, and accessed. Thus, *it also influences volume and velocity dimensions*.

Under these considerations, we design our proposal, SOLID (*Service-OnLine-Index-Data architecture* [12]), as a *high-performance architecture for real-time systems managing Big Semantic Data*, i.e., *big RDF datasets*. SOLID tackles this scenario through a tiered configuration which separates the complexities of Big Semantic Data storage and indexing from real-time data acquisition and consumption. This decision is based on the foundations of the *Lambda* architecture [7], but SOLID adapts them for dealing with the specific RDF needs. Fig. 1 illustrates the SOLID configuration. The content tier is the central data core and groups three layers. On the one hand, the Big Semantic Data is preserved on a data layer which realizes a datastore for serialized RDF. Besides storing the historical data, this data layer provides functionality for basic RDF retrieval. The index layer is built on top of the data layer and is responsible for providing indexed fast access to the Big Semantic Data. This ensures efficient query resolution, typically through SPARQL [13], the de facto standard for querying RDF. Therefore, the combination of these two layers brings a competitive solution for the first aforementioned challenge: storing and serving Big Semantic Data containing historical data. On the other hand, the online layer deals with the real-time issues. This layer realizes a more complex datastore which gathers run-time data and exposes them for efficient

¹ <http://www.idc.com/>.

متن کامل مقاله

دریافت فوری ←

ISIArticles

مرجع مقالات تخصصی ایران

- ✓ امکان دانلود نسخه تمام متن مقالات انگلیسی
- ✓ امکان دانلود نسخه ترجمه شده مقالات
- ✓ پذیرش سفارش ترجمه تخصصی
- ✓ امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
- ✓ امکان دانلود رایگان ۲ صفحه اول هر مقاله
- ✓ امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
- ✓ دانلود فوری مقاله پس از پرداخت آنلاین
- ✓ پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات