Addressing data management on the cloud: tackling the big data challenges*

Genoveva Vargas-Solar[1]

French Council of Scientific Research, LIG-LAFMIA
681 Rue de la Passerelle, BP. 72, 38402 Saint Martin d'Hères Cedex, France
Genoveva.Vargas-Solar@imag.fr

Abstract. The increasing adoption of the cloud computing paradigm has motivated a redefinition of traditional data management methods. In particular, data storage management has been revisited, due to a growing interest in the challenges and opportunities associated to the NoSQL movement. Together with novel opportunities offered by the cloud for accessing hardware, and resources as an implicit commodity, the possibility of processing huge data collections for supporting a "science of data" has put data in the centre of the arena. In this new era of information, it seems that new research and development opportunities are emerging The objective of this paper is to discuss three aspects that seem to be in the agenda for addressing the new vague of challenges introduced by the so called big data age: determining big data collection sizes from L to XXL; providing flexible persistence support; determining and capitalizing the value of big data collections.

1 Introduction

With hardware, networks and software having been commoditized to the point that all are essentially free, it was inevitable that the trajectory toward maximum entropy would bring us the current age of Big Data [1].

Cloud computing is emerging as a relatively new approach for dealing and facilitating unlimited access to computing and storage resources for building applications. Instead of discussing whether it is really a novel approach, it is important to identify which are the real challenges introduced by two important notions associated to the cloud: unlimited access and pay as you go model. The very basic principle of cloud computing is to assume that applications accessible through a network are built upon a service oriented infrastructure dedicated to provide them with the necessary (not more not less) computing, storage and network resources. The underlying infrastructure manages such resources transparently without demanding the application to have code to manage them or to reserve more resources than those it really requires. The difference with classic approaches is that the application can have an ad hoc execution context and that the resources it consumes are not necessarily located in one machine. Instead of having one for all computer or server, the computing context is configured according to the characteristics of the application. Instead of buying one computer/server resources are provided (and bought) on demand.

* This work is partially financed by the projects S2EUNET and CASES of the FP7 IRESES program.

[1] Shomit Ghose, Big Data, The Only Business Model Tech has Left; CIO Network http://www.bigdatabytes.com/big-data-is-the-new-tech-economy/

Applications can be deployed and scaled rapidly without having to produce physical servers. A computing cloud is a self-service proposal where a credit card can purchase compute cycles, and a Web interface or API is used to create virtual machines and establish network relationships between them. The result is a Cloud provider API to create an application initial composition onto virtual machines, to define how it should scale and evolve to accommodate workload changes including self-monitoring and self-expanding applications. Together with this novel opportunity for accessing hardware, and resources as an implicit commodity, the possibility of processing huge data collections for supporting a science of data has put data in the centre of the arena. More than ever "information is power" not only for advanced business oriented decision making, but for common users with day by day requirements: the best way to go from one place to another, controlling the personal CO_2 trace, getting recommendations on leisure or financial activities. In this new era of information, it seems that new research and development opportunities are emerging.

The objective of this paper is to discuss three aspects that seem to be in the agenda for addressing the new vague of challenges introduced by the so called big data age. Accordingly, the remainder of the paper is organized as follows. Section 2 discusses the characteristics of the big data collections focussing on L, XL, and XXL sizes. Section 3 generally introduces the polyglot persistence approach intended to combine NoSQL stores for providing a scalable data management solution when performance and data volumes are important in size and when information systems have to be built on top of them. Section 4 discusses the role of economic business models related to big data and how they are integrated in the data processing and exploitation processes. Finally, Section 5 concludes the paper with final comments.

2 How big is Big?

Digital information scale has evolved together with technology advances, particularly the capacity to store "huge" data collections, but also the capacity to produce, process and exploit such collections. How big is Big? This depends on the perspective. History started with the bit – the code used by computers to store and process data–, and the notion of byte as the basic unit of computing. This unit had to evolve as it was necessary to manage larger volumes of data and new measures appeared: Kbytes, Mbytes, for instance a typical pop song measures 4 MBytes and the complete works of Shakespeare a total of 5 MBytes. Today we are easily talking about Tbytes. For example, all catalogued books in America's Library of Congress measure around 15Tbytes, but also my personal backup information produced during around two years measures more than 3-4 Tbytes (!) ... Google announces that they process 1Pbyte per hour, and the total amount of information in existence in this year is forecast to be around 1,27 ZB [2].

Beyond the volumes of data that we can and – maybe – need to process today, it is important to consider the characteristics of producers and consumers. Some years ago, only specialized acquisitions of data were capable of dealing with really huge volumes of data: the Sloan Digital Sky Survey, the large hadron collider... The last years, with the emergence of social networks, mobile devices, sensors, the digital shadow of *"monsieur tout le monde"* measures terabytes and even petabytes – considering a "moderate" use of technology –: 75 percent of the information is generated by individuals writing documents, taking pictures, downloading music, etc. Yet, this is far less than the amount of information being created about them in the digital universe (!). Indeed, Gbytes of stored content can generate a Pbyte or more of transient data that we typically do not store.

[2] http://blog.websourcing.fr/infographie-la-vrai-taille-dinternet/

For example, we store the digital TV signals we watch but do not record voice calls that are made digital in the network backbone during the duration of a call.

This phenomenon has lead to the emergence of the term "big data" referring to *collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications*[3]. But having big data implies also challenges for capturing, curating, storing, searching, sharing, analyzing, and visualizing within a tolerable elapsed time such huge data collections. Thus, the community agrees to say that data growth challenges and opportunities are three-dimensional (3Vs model): increasing volume (amount of data), velocity (speed of data in and out), variety (range of data types and sources). According to the 3Vs model, big data *"are high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization*[4].

Challenges opened by the big data movement have to do with data processing for analyzing and discovering knowledge, for providing recommendations and understanding trends. The "thing" with big data is that it calls also for approaches that must revisit the whole information systems stack: data management functions (storage, duplication, querying, sharing), management systems architectures, and infrastructure services that can help to exploit and give interesting use and can capitalize the acquisition of such massive data collections.

3 Polyglot persistence for building database applications

The use of heterogeneous data stores within a single system is gradually becoming a common practice in application development. Modern applications tend to rely on a polyglot approach to persistence, where traditional databases, non-relational data stores, and scalable systems associated to the emerging NewSQL movement, are used simultaneously. Even if there is no standard definition of what NoSQL means [5] there are common characteristics of these systems: (i) they do not relay on the relational model and do not use the SQL languages; (ii) the tend to run on cluster architectures; (iii) they do not have a fixed schema, allowing to store data in any record. Examples include MongoDB, CouchDB, Cassandra, Hbase, and also BigTable and SimpleDB which are tied to cloud services of their providers, they fit in general operating characteristics.

The diversity of systems and models introduces decision making challenges when designing an information system. Which is the right tool for the right job? Inspired in the polyglot programming approach that states that applications should be written in a mix of languages to take advantage of different languages are suitable for tackling different problems, emerges the polyglot persistence approach. The principle is based on the idea that applications have different requirements for the different types of information they deal with. Take for instance a classic e-commerce application that deals with a catalogue, user access logs, financial information, shopping carts, and purchase transactions. These data have different management requirements, for instance, the catalogue has a lot of reads and very few writes assuming that the catalogue of a shop is more or less stable. Information about user sessions require rapid access for reads and writes but they do not need to be durable. The shopping carts need high availability across multiple locations, and can merge inconsistent writes. User activity logs imply high volume of writes on multiple nodes, while recommendations for users must provide rapid link traversals between friends, product purchases and ratings.

[3] Wikipedia
[4] ibid.
[5] The notion was introduced in a workshop in 2009 according to [4].

Thus, according to the polyglot persistence approach, any decent sized enterprise will have a variety of different data storage technologies for different kinds of data a new strategic enterprise application should no longer be built assuming a relational persistence support the relational option might be the right one - but programmers should seriously look at other alternatives. In the example, the catalog and, user sessions data can be stored in a key-value store like Redis [6] or MongoDB [7]; the shopping cart with loosely structured data can be store in Riak [8], recommendations in Neo4J [9] graph model support and user activity logs in column oriented data store as Cassandra [10]. Financial data and reporting would be supported by a relational DBMS. The choices illustrated here are more or less intuitive. The interested reader can refer to the characteristics of these systems in [2]. The objective here is also to show that it should be a "complex" decision making tuning process to choose the suited store for the data managed by applications.

According to [4], polyglot persistence is pertinent for applications essentially composing and serving web pages that only look up page elements by ID, that have no need for transactions, and no need to share their database. A problem like this is much better suited to a key-value store than the corporate relational hammer they had to use. Applications scaling to lots of traffic gets harder and harder to do with vertical scaling. Many NoSQL databases are designed to operate over clusters so they can tackle larger volumes of traffic and data than is realistic with a single server.

Nonetheless, we believe that the combination of these heterogeneous data stores, flexible schemas and non-standard APIs, represent an added complexity for application developers. Considering that schema-less data stores do not enforce any particular schema and generally lack a data dictionary from where to recover the structure of the entities that they manage [4], software developers need to analyze the application source code (including field names, types, etc.) of systems that rely on these data stores, in order to discover and maintain the overall applications external schemas [3]. However, this manual analysis can be an error-prone activity, due to the combination of different programming styles, APIs and development environments. As part of an effective database change management process, the recovered schemas also need to be adequately maintained, so as to allow the tracking of changes and the replication of the database structure, both between different versions of the application, and on multiple deployment environments [5].

Essentially, we consider that schemas are shifted from the database to the application source code. To help overcome this situation we propose ExSchema [11], an automated approach and its implementation to discover data schemas from polyglot persistence applications, based on source code analysis techniques. We integrate our discovery process with Git repositories, in order to analyze and maintain the evolution of data schemas over a period of time. Besides inspecting the Java API provided by these data stores, we also support a subset of the Spring Data projects associated to each of them. The discovery mechanisms provided by ExSchema [1] are based on source code analysis techniques, particularly on API usage, and are available as an Eclipse plugin. The source code analysis techniques currently implemented by ExSchema focus on insert and update operations of the supported APIs. However, for future work we would like to include the analysis of the queries defined in polyglot persistence applications.

[6] http://redis.io
[7] http://www.mongodb.org
[8] http://basho.com
[9] http://www.neo4j.org
[10] http://cassandra.apache.org
[11] http://code.google.com/p/exschema/

4 Big data value

In the information era users have to make decisions and generate knowledge by dealing with huge amounts of data. Putting order to data collections takes time and people and organizations invest energy finding, retrieving and, organizing data of different provenances and qualities. Effort in creating these data collections (links collections, images, documents) and then in summarizing and generating information out of them is rarely shared. These collections of curated data often remain in the logs and histories of personal computers. A good number of person/hours are somehow wasted or at least they are not capitalized once the data has been used or not. This is the case in todays social networks experience of lambda users. For example take Facebooks timeline, where people spend time organizing their important events on time, by defining sorts of checkpoints. Then, others can exploit this information without letting authors have any turnovers on their data organization effort.

Consider instead a scenario where data tagged with comments on consumers experiences and opinions about their quality and usefulness, are exported as a data market with an associated cost model. For example, defining a price on the number of free access that can be executed on them, poker like pay to see model where the market exposes its catalogue but shares data of different qualities according to specific fees or subscriptions, define a cost per recurrent access to be executed on RSS data depending on a data/loss rate, define other costs depending on the results are made persistent or not, open or not, continuously available or not. Research on data processing and brokering are promising given the explosion of huge amounts of data largely distributed and produced by different means, and the requirements to consume them to have the right information, at the right place, at the right moment, with the right cost.

Our vision is that it is necessary to see data management that goes beyond accessing timely and costly ready to use data sources. It should be seen as an effort that implies economically capitalizing the effort of going out for hunting data sources and services of different qualities and stemming from different processing processes, curating and delivering them. We shall call this environment a data market because we will assume that data brokers have an associated cost model. These brokers can be then contacted for accessing to data that can be processed for building new data collections that can be then made available in the data market. The key issues here are: (i) being able to associate a cost model for the data market, i.e., associate a cost to raw data and to processed data according on the amount of data and processing resources used for treating it, for instance; (ii) then being able to combine these cost model and the consumer expectations (service level agreement) with processing resources cost required by data processing; (iii) providing data management and brokering processing mechanisms under ad hoc business models.

The objective is to tackle economy-oriented data brokering process of large tagged data collections exported by data markets. Propose a novel data brokering process that can coordinate the access to data markets guided by an economical model that can provide strategies guaranteeing data access, processing and delivery based on ad hoc business models.

In our vision, economy oriented data brokering will be tuned by the cost of accessing data markets, and the cost of using resources for brokering data. The challenge will be to manage the computation of results versus the cost of accessing completely of partially such results according to their completeness, data delivery frequency (continuous or one shot), their duration, and their quality:

- Completeness: a sample of resulting data that can be completed according to an economic model: guided by predefined fees (1M 10euros, 10M 15 euros), the user specifies whether to buy data to complete results.
- Data delivery frequency: new data can be delivered, while a data market proposes new data. It is up to the user to subscribe according to different fees.
- Duration: volatile/persistent results produced out of series of queries can be used for building a new data market. In this case the owner can require accessing and buying storage services for dealing with her data markets and exporting them as paying services. The associated cost of such service will depend on the QoS and the kind of Service level agreements that the service can honor.
- Content quality: data provenance, data freshness and degree of aggregation.

5 Conclusions

An important observation to do is that in today's perspectives introduced by architectures like the cloud, and by movements like big data, there is an underlying economic model that guides directly the way they are addressed. This has not been a common practice in previous eras, but today the "pay as U go", the iTunes economic models have become an important variable of (big) data production, consumption and processing.

Which is the value to obtain from big data? Big data is not a "thing but instead a dynamic/activity that crosses many IT borders. Big data is not only about the original content stored or being consumed but also about the information around its consumption. Big data technologies describe a new generation of technologies and architectures, designed to economically extract value from very large volumes of a wide variety of data, by enabling high-velocity capture, discovery, and/or analysis [12]. Even if technology has helped by driving the cost of creating, capturing, managing, and storing information the prime interest is economic: the trick is to generate value by extracting the right information from the digital universe.

The key is how quickly data can be turned in to currency by analyzing patterns and spotting relationships/trends that enable decisions to be made faster with more precision and confidence; identifying actions and bits of information that are out of compliance with company policies can avoid millions in fines; proactively reducing the amount of data you pay (18,750 USD/gigabyte to review in eDiscovery) by identifying only the relevant pieces of information; optimizing storage by deleting or offloading non-critical assets to cheaper cloud storage thus saving millions in archive solutions [13].

The good news are that current technology is not adequate to cope with such large amounts of data (requiring massively parallel software running on tens, hundreds, or even thousands of servers). Expertise is required in a wide range of topics including: machine architectures (HPC), service-oriented-architecture distributed/cloud computing, operating and database systems software engineering, algorithmic techniques and networking. This expertise must be put together with economic business models into the next generation of database management services conceived to address the big data challenge.

[12] http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf
[13] ibid.

References

1. Juan Castrejón, Genoveva Vargas-Solar, Christine Collet, and Rafael Lozano. Model-driven cloud data storage. In *Proceedings of CloudMe*, 2012.
2. Rick Cattell. Scalable sql and nosql data stores. *SIGMOD Record*, 39(4):12–27, 2010.
3. Jean-Luc Hainaut, Jean Henrard, Vincent Englebert, Didier Roland, and Jean-Marc Hick. Database reverse engineering. In *Encyclopedia of Database Systems*, pages 723–728. 2009.
4. Fowler M. and Sadalage P. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. 2012.
5. C. Mullins. *Database Administration: The Complete Guide to DBA Practices and Procedures*. 2012.