Efficient data management for putting forward data centric sciences

Genoveva Vargas-Solar[1]

French Council of Scientific Research, LIG-LAFMIA
700 avenue Centrale, Domaine Universitaire, 38401 St Martin d Hères, France
genoveva.vargas@imag.fr
http://www.vargas-solar.com

Abstract. The novel and multidisciplinary data centric and scientific movement promises new and not yet imagined applications that rely on massive amounts of evolving data that need to be cleaned, integrated, and analysed for modelling, prediction, and critical decision making purposes. This paper explores the key challenges and opportunities for data management in this new scientific context, and discusses how data management can best contribute to data centric sciences applications through clever data science strategies.

1 Introduction

Data centric sciences are leading to different trendy applications for smart cities, homes and energy; urban computing, monitoring and student assessment from kinder garten to university, automatic health control and monitoring, finances self-management. These applications still provide partial solutions to multi-variable and multi-facet problems related to the understanding of complex systems. For example, the personalization of drugs for better attacking diseases, understanding social behaviours of individuals and societies, modelling and reproducing human skills, and processes like creativity, artistic creation, neuronal behaviour and prediction of natural changes and phenomena. The backbone of these problems are data collections that must be harvested and they must be representative enough and with specific properties so that they can serve as raw material to processing and analysis algorithms. These processes and algorithms can reproduce and understand such phenomena and systems, analyse them, and deduce valid conclusions.

Thus, it is necessary to deal with data collections characterized by their volume, production rate (velocity), variety, multiple veracity, and value. Besides these properties, data collections are also determined by the type of content they group and the conditions in which they are consumed. Paul McFedries in his article Beyond Just Big Data[11] classifies data collections into:

- thick data, which combines both quantitative and qualitative analysis;
- long data, which extends back in time hundreds or thousands of years;

– hot data, which is used constantly, meaning it must be easily and quickly accessible, and
– cold data, which is used relatively infrequently, so it can be less readily available.

These types of data collections can be consumed according to different patterns that determine the conditions in which they are accessed:

– shared by several users and accessed in a concurrent manner, with specific isolation or atomicity requirements;
– stored only for archival purposes;
– distributed for promoting parallel data processing and analytics;
– continuous processing for data flows with efficient read/writes.

Cloud and High Performance Computing have evolved to respond to emerging data management and processing challenges introduced by data centric sciences. There is a long path to go through, but the scientific community and industry has started designing the next generation of data management and processing stacks and evolve towards a data science. Data science will deal with the exascales and its associated variety, not in terms of format but in terms of different properties: hot/cold, long/cold, thick data.

On the other hand, the emergence of Big Data some years ago denoted the challenge of dealing with huge collections of heterogeneous data continuously produced and to be exploited through data analytics processes. First approaches have addressed data volume and processing scalability challenges. Solutions have addressed the problem of balancing and scheduling the delivery of computing, storage, memory, and communication resources (bandwidth and reliability) for greedy analytics and mining processes with high in-memory and computing cycles requirements.

Based on these previous results, emerging data centric sciences (e.g., data science, network science, computational science, social electronic sciences, digital humanities) develop methodologies weaving data management, greedy algorithms, and programming models that must be tuned to be deployed in different target computer architectures.

This paper focuses data collections with different uses and access patterns because these properties tend to reach limits of:

– the storage capacity (main memory, cache and disks) required for archiving data collections permanently or during a certain period, and
– the pace (computing speed) in which data must be consumed (harvested, prepared and processed).

These aspects must be addressed when dealing with data centric sciences problems. They are discussed in this paper. Accordingly the remainder of the paper is organized as follows. Section 2 gives an overview of existing approaches addressing data storage infrastructures, data analytics under a data science vision, and infrastructures providing computing capacity for processing data using greedy

algorithms. Section 3 profiles data centric sciences and introduces their particular requirements with respect to data as an enabling backbone. It gives the general lines for storing and delivering data and thereby contributing to the efficient execution of data analytics processes running on parallel environments. Finally, Section 4 concludes the paper and discusses perspectives.

2 Scaling up data processing

The emergence of Big Data has dealt with huge collections of heterogeneous data continuously produced and to be exploited through data analytics processes.

Big Data management and processing are no longer only associated to scientific applications with prediction, analytics requirements, data centric sciences also call for data aware management to address the understanding and automatic control of complex systems, to the decision making in critical and non-critical situations. Given that data centric sciences rely on data collections that stem from complex and diverse gathering processes, two data management issues are key for enabling analysis workflows:

- data storage and distribution, and
- runtime environments that provide enough computing resources for executing greedy data exploration and analytics algorithms.

The next sections discuss these topics.

2.1 Data storage and distribution

Data management in many data analytics workflows underlines the need to reduce overhead when data are read, updated, and put into storage supports (memory, cache) as stated by the RUM conjecture proposed in [3]. Several platforms address some aspect of the problem like Big Data stacks [6, 1]; data processing environments (*e.g.,* Hadoop, Spark, CaffeonSpark); data stores dealing with the CAP (consistency, atomicity and partition tolerance) theorem (*e.g.,* NoSQL's); and distributed file systems (*e.g.,* HDFS). The principle is to define API's (application programming interface) to be used by programs to interact with distributed data management, processing, and analytics layers that can cope with distributed and parallel architectures.

Objects and components persistence has been an important issue addressed already by consolidated middleware such as JBOSS and PiJAMA. The new exascale requirements introduced by greedy processes often related to Big Data processing has introduced objects persistence issues again. Particularly for the "passivation" (storage of the execution state and environment of a component) of greedy processes execution in the presence of failures. Instead of loosing costly processes (in time and computing resources) they can be stored and the restarted in without loosing already executed tasks.

In order for exascale and/or Big Data systems to deliver the needed I/O performance, new storage devices such as NVRAM or Storage Class Memories

(SCM) need to be included into the storage/memory hierarchy. Given that the nature of these new devices are closer to memory than to storage (low latencies, high bandwidth, and byte-addressable interface) using them as block devices for a file system does not seem to be the best option. DataClay[5] proposes object storage to enable both the programmer, and DataClay, to take full advantage of the coming high-performance and byte-addressable storage devices. Today, given the lack of such devices, DataClay performs a mapping of such abstractions to key-value stores such as Kinetic drives from Seagate[1].

Data structures and associated functions are sometimes more important for some requirements rather than non functional properties like RUM or CAP. Non-relational databases have emerged as solutions when dealing with huge data sets and massive query work load. These systems have been redesigned to achieve scalability and availability at the cost of providing only a reduced set of low-level data management functions, thus forcing the client application to take care of complex logic.

The large spectrum of data persistence and management solutions are adapted for addressing workloads associated with Big Data volumes; and either simple read write operations or with more complex data processing tasks. The challenge today is choosing the right data management combination of tools for variable application requirements and architecture characteristics. Plasticity of solutions is from our point of view the most important property of such tools combination.

Once data has been stored, possibly under a distributed strategy for addressing storage space and data availability, analytics processes can run on top of the whole collection or some sample of it. These processes require very often to be parallelized in order to process the whole data collections in reasonable time. Many data centric science solutions require this type of parallel settings. Thus, there exist different types of parallel runtime environments that enable to deploy analytics processes with different degrees of transparency.

2.2 Parallel runtime environments

Today maybe because of the emergence of Big Data and greedy algorithms and applications requiring computing resources, parallel architectures have come back in the arena. There are different kinds of computing, memory, and storage resources providers that adopt their own method for delivering such resources for executing programs. According to [9] there are three categories of resources provision: PaaS frameworks, programming models for computing intensive workloads, and programming models for Big Data.

Platform-as-a-Service (PaaS) layers offer APIs to write applications. For example, in the Microsoft Azure Cloud programming model applications are structured according to roles, which use APIs to communicate (queues) and to access persistent storage (blobs and tables). Microsoft Generic Worker proposes a mechanism to develop a Worker Role that eases the porting of legacy code in the

[1] http://www.seagate.com

Azure platform [13]. Google App Engine provides libraries to invoke external services and queue units of work (tasks) for execution; furthermore, it allows to run applications programmed in the Map-Reduce model. Data transfer and synchronization are handled automatically by the runtime. Environments for computing workload intensive applications use in general the bag of tasks execution model conceiving an application as composed of independent parallel tasks. For example, the Cloud BigJob, Amazon EC2, Eucalyptus, Nimbus Clouds, and ProActive that offer a resource manager developed to mix Cloud and Grid resources [12, 2]. Map-Reduce programming is maybe the most prominent programming model for data intensive applications. Map-Reduce based runtime environments provide good performance on cloud architectures above all on data analytics tasks working on large data collections. Microsoft Daytona[2] proposes an iterative Map-Reduce runtime for Windows Azure to support data analytics and machine learning algorithms. Twister [7] is an enhanced Map-Reduce runtime with an extended programming model for iterative Map-Reduce computations. Hadoop [4] is the most popular open source implementation of Map-Reduce on top of the Hadoop Distributed File System (HDFS). The use of Hadoop avoids the lock into a specific platform allowing to execute the same Map-Reduce application on any Hadoop compliant service, as the Amazon Elastic Map-Reduce[3].

For data centric sciences parallel tasks, strategies for dealing with data persistence, efficient read and write in memory operations, data preparation (transformation, cleaning, collocation) must be integrated within the runtime environments. These environments must provide interfaces, either API or annotations to enable programmers to transparently tag their data with their associated properties. It will be up to the runtime environment to provide data management services or integrate existing ones that can ensure efficient data I/O thanks to well adapted operations with underlying processes for providing the right data in the right moment and in the right place.

3 Supporting data centric sciences experiments

More than ever, researchers in all disciplines find themselves wading through more and more kinds of data. Frequently, there is no standard system for storing, organizing, or analysing this data. Doing so requires a set of skills researchers must largely learn independently. In recent years, data science has emerged as the field that exists at the intersection of math and statistics knowledge, expertise in a science discipline, and so-called hacking skills, or computer programming ability. Yet, data science is more of an emerging interdisciplinary philosophy, a wide-ranging modus operandi that entails a cultural shift in the academic community. The term means something different to every data scientist, and in a time when all researchers create, contribute to, and share information that describes how we live and interact with our surroundings in unprecedented detail, all researchers are data scientists. The philosophy is promising and opens new

[2] http://research.microsoft.com/en-us/projects/daytona/

[3] Amazon elastic Map-Reduce. http://aws.amazon.com/documentation/elasticmapreduce/

possibilities to develop methodologies that take advantage of information and communication technologies that find themselves new challenges and scientific opportunities in other sciences included in the data science umbrella.

The rapid evolution of computer architectures delivering an increasing amount of resources have opened new opportunities for data centric sciences. As said before, these sciences address complex problems modelled by several related variables that interact as complex systems with plenty of constraints (climate change, traffic control, social behavior in crowds, simulation of biological processes and social phenomena, prediction of diseases). For addressing these kind problems, data centric sciences perform experiments that weave data management, with greedy artificial intelligence and data mining algorithms and computing architectures services.

We consider that efficient data management is the backbone of these experiments. For us, it is no longer pertinent to reason with respect to a set of computing, storage, and memory resources. Instead, it is necessary to conceive algorithms and processes considering an unlimited set of resources usable via a *pay as U go* model, energy consumption or services reputation and provenance models. Rather than designing processes and algorithms considering as threshold the resources availability, computing service providers (e.g. cloud providers) change this vision and rather take into consideration the economic cost of the processes vs. the resources they consume.

It is necessary to develop novel strategies and tools for storing and delivering terabytes of on-line data collections consisting of billions of records, optimizing the consumption of resources while reducing the overhead of exploiting them by parallel programs.

3.1 Data analytics for data science

Methods for querying and mining Big Data are fundamentally different from traditional statistical analysis on small samples. Big Data are often noisy, dynamic, heterogeneous, inter-related, and untrustworthy. Nevertheless, even noisy Big Data could be more valuable than tiny samples. Indeed, general statistics obtained from frequent patterns and correlation analysis usually overpower individual fluctuations and often disclose more reliable hidden patterns and knowledge.

Big Data forces to view data mathematically (e.g., measures, values distribution) first and establish a context for it later. For instance, how can researchers use statistical tools and computer technologies to identify meaningful patterns of information? How shall significant data correlations be interpreted? What is the role of traditional forms of scientific theorizing and analytic models in assessing data? What you really want to be doing is looking at the whole data set in ways that tell you things and answers questions that you are not asking. All these questions call for well-adapted infrastructures that can efficiently organize data, evaluate and optimize queries, and execute algorithms that require important computing and memory resources.

Big Data has enabled the next generation of interactive data analysis with real-time answers. In the future, queries towards Big Data will be automatically generated for content creation on websites, to populate hot-lists or recommendations, and to provide an ad-hoc analysis of data sets to decide whether to keep or to discard them [8]. Scaling complex query processing techniques to terabytes while enabling interactive response times is a major open research problem today.

Analytical pipelines can often involve multiple steps, with built-in assumptions. By studying how best to capture, store, and query provenance, it is possible to create an infrastructure to interpret analytical results and to repeat the analysis with different assumptions, parameters, or data sets. Frequently, it is data exploration and visualization that allow Big Data to unleash its true impact. Visualization can help to produce and comprehend insights from Big Data. Visual.ly, Tableau, Vizify, D3.js, R, are simple and powerful tools for quickly discovering new things in increasingly large datasets.

In parallel to data science addressing (Big) Data under different perspectives, emerges computational science uses advanced computing capabilities to:

- understand and solve complex problems fusing numerical and non-numerical algorithms and modelling and simulation tools;
- computer and information science that develop advanced hardware, software, networking, and data management systems needed to solve computationally demanding problems; and
- computing infrastructure that supports science and engineering problem solving.

In practice, it is the application of computer simulation and other forms of computation from numerical analysis and theoretical computer science to solve problems in various scientific disciplines, for example the one reports in [10] about the application of Computational Science in Archaeology. A collection of problems and solutions in computational science can be found in [14]. Computational science techniques are being applied to perform digital humanities research that are exported for experiment reproducibility. For example, the action Museum 2.0 that explores ways that web 2.0 philosophies can be applied in museum design. Museums share their data collections for digital social computational scientist to perform research and visualize them.

3.2 Data collections sharding and storage for data centric sciences

Data sharding has its origins in centralized systems that had to partition files, either because the file was too big for one disk, or because the file access rate could not be supported by a single disk. Relational distributed databases use data sharding when they place relation fragments at different network sites. Data sharding allows parallel database systems to exploit the I/O bandwidth of multiple disks by reading and writing them in parallel. Relational DBMS implement strategies for sharding data (i.e., tuples): round robin seems appropriate

for processes accessing the relation by sequentially scanning all of it on each query, hash-partitioning seems suited for sequential and associative access to data avoiding the overhead of starting queries on multiple disks.

While sharding is a simple concept that is easy to implement, it raises several physical database design issues. Each data collection must have a sharding strategy and a set of disk fragments. Increasing the degree of sharding usually reduces the response time for an individual query and increases the overall throughput of the system. In parallel DBMS for sequential scans, the response time decreases because more processors and disks are used to execute the query; for associative scans, the response time improves because fewer tuples are stored at each node, and hence the size of the index that must be searched decreases.

The NoSQL momentum introduces new datasets storage possibilities. Graph, key-value, multidimensional records stores can provide storage solutions with efficient read/write operations possibilities. Sharding can be also guided by the structure of data, and it can be coped to ad-hoc stores that can ensure persistency and retrieval. This implies also a tuning effort of different NoSQL stores that should then provide efficient that reads/writes with specific degrees of availability, fault tolerance, and consistency. NoSQL stores rely on automatic replication mechanisms. Reads and data querying implemented at the application level are then executed by applying Map-Reduce execution models. Sharded data collections are thus the key to parallel executions.

The NoSQL approach leads to performant ad-hoc per problem solutions. The DBMS approaches, in contrast, promote one-fits all solutions where sharding strategies can be tuned. What is true, it that the pertinence of sharding strategies depends on analysis and processing requirements and available storage and memory resources. In both cases, data collections sharding requires effort, expertise, and a lot of testing for finding the best balance to achieve good data processing and analysis performance. We believe that sharding data collections must be based on clever data organizations and indexing on file systems, data stores, caches, or memories to reduce effort and ensure performant exploitation (retrieval, aggregation, analysis).

Therefore, automatic and elastic data collections sharding toolsare necessary to parametrize data access and exploitation by parallel programs willing to be performant and scale-up in different target architectures:

- Delivering of datasets along distributed process units avoiding bottlenecks.
  - Different data structures used to reduce the overhead associated to read, updates and persistence requests (RUM conjecture) on different target architectures.
  - New strategies and algorithms to enable the transfer of large data sets between distributed processing units. We will explore techniques such as data streaming and process transfer instead of bulk data transfer, as potential solutions.
  - Ensure properties like availability, consistency, and partition tolerance (CAP theorem) that can be reinforced in an adaptable and dynamic manner.

– Assessing and predicting storage (disk, cache, memory) resources for optimally delivering data and scaling up parallel processing.

With the new computing, storage, and memory requirements stemming from data centric science problems, the use of cluster oriented architectures providing such resources has increased and is somehow democratized particularly with the emergence of the cloud. Data management requirements have to be revisited they vary from simple storage coupled with read/write requirements with reasonable data processing performance needs to real critical applications dealing with huge data collections to be processed by greedy algorithms. In such settings it is possible to exploit parallelism for processing data by increasing availability and storage reliability thanks to duplication and collocation.

4 Conclusion and perspectives

Digging into data today requires data science and data engineering methodologies that can apply information and communication technologies in the best way to address the challenges introduced by the characteristics of data collections (big, continuous, multi-form, and multimedia with different veracity degrees) and that require computing resources in order to be exploited. Yet, data collections are not only a digital artifact, they represent content determined by the conditions in which it has been authored, produced, collected, and digitalized, by its provenance, by the intention behind its exploitation, and by the conditions and rules in which it can be reused. These characteristics have to be exhibited and be accessible to (social) scientists wishing to use them as raw material to perform research and analysis.

It is no longer possible to practice digital social sciences without being supported by data science and engineering to avoid empirical use of technology to maintain and curate data collections. The way algorithms and technology are used has to avoid technological bias pollution of results. If some data processing and cleaning is necessary because of mathematical or technical reasons, this has to be known by the scientist performing the analysis, and it has to be reported in the results obtained. The mathematical and technical conditions in which analysis and visualisation of results are done must be considered in the interpretation of those results. This will ensure the credibility of the experiment performed on digital data using ICT and it will provide an objective perception of the results and interpretations. Data science and engineering do not have sense without having concrete problems with explicit requirements, rules and expectations. The challenge is to have tools that can change their preferences towards data management and analytics and provide elastic strategies for implementing these operations. Such strategies should evolve as data acquire different structures and semantics as a result of the data processing operations applied on them.

References

1. A. Alexandrov, R. Bergmann, S. Ewen, J.C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, et al. The stratosphere platform for big data analytics. *The VLDB Journal*, 23(6):939–964, 2014.
2. Brian Amedro, Françoise Baude, Denis Caromel, Christian Delbé, Imen Filali, Fabrice Huet, Elton Mathias, and Oleg Smirnov. An efficient framework for running applications on clusters, grids, and clouds. In *Cloud Computing*, pages 163–178. Springer, 2010.
3. M. Athanassoulis, M. Kester, L. Maas, R. Stoica, S. Idreos, A. Ailamaki, and M. Callaghan. Designing access methods: The rum conjecture. In *International Conference on Extending Database Technology (EDBT)*, 2016.
4. Dhruba Borthakur. The hadoop distributed file system: Architecture and design. *Hadoop Project Website*, 11(2007):21, 2007.
5. Toni Cortes, Anna Queralt, Jonathan Mart, and Jesus Labarta. Dataclay: towards usable and shareable storage. *http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/*.
6. Matthew Franklin. The berkeley data analytics stack: Present and future. In *Big Data, 2013 IEEE International Conference on*, pages 2–3. IEEE, 2013.
7. Thilina Gunarathne, Bingjing Zhang, Tak-Lon Wu, and Judy Qiu. Scalable parallel computing on clouds using twister4azure iterative mapreduce. *Future Generation Computer Systems*, 29(4):1035–1048, 2013.
8. Stratos Idreos, Ioannis Alagiannis, Ryan Johnson, and Anastasia Ailamaki. Here are my data files. here are my queries. where are my results? In *Proceedings of 5th Biennial Conference on Innovative Data Systems Research*, number EPFL-CONF-161489, 2011.
9. Francesc Lordan, Enric Tejedor, Jorge Ejarque, Roger Rafanell, Javier Alvarez, Fabrizio Marozzo, Daniele Lezzi, Raül Sirvent, Domenico Talia, and Rosa M Badia. Servicess: An interoperable programming framework for the cloud. *Journal of grid computing*, 12(1):67–91, 2014.
10. Ben Marwick. Computational reproducibility in archaeological research: basic principles and a case study of their implementation. *Journal of Archaeological Method and Theory*, pages 1–27, 2016.
11. Paul McFedries. Beyond just big data. *http://spectrum.ieee.org/computing/software/*.
12. Junjie Peng, Xuejun Zhang, Zhou Lei, Bofeng Zhang, Wu Zhang, and Qing Li. Comparison of several cloud computing platforms. In *Information Science and Engineering (ISISE), 2009 Second International Symposium on*, pages 23–27. IEEE, 2009.
13. Yogesh Simmhan, Catharine Van Ingen, Girish Subramanian, and Jie Li. Bridging the gap between desktop and the cloud for escience applications. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 474–481. IEEE, 2010.
14. WH Steeb, Y Hardy, A Hardy, and R Stoop. Problems and solutions in scientific computing with c++ and java simulations world scientific publishing. 2004.