Unveiling objects in Big Data - model based approach

Amir Khatibi[1], Fabio Porto[1], Angelo Clarlili[2], Paulo Pires[3], Patrick Valduriez[4], Dennis Shasha[5]

LNCC[1]
Petrópolis, RJ, Brazil
Khatibi, fporto@lncc.br

EMC Research[2]
Ilha do Fundão, RJ, Brazil
Angelo.ciarlini@emc.com

UFRJ, DCC[3]
Ilha do Fundão, RJ, Brazil

INRIA[4]
Montpellier, France
Patrick.valduriez@inria.fr

NYU[5]
New York, USA
Dennis.shasha@cs.nyu.edu

Abstract Big data processing is expected to empower decision-making as more information becomes accessible to analytical tools. In this paper, we argue that the data deluge produced by the Big Data phenomenon blurs, amongst billions of dataset elements, high-level objects that can only be perceived once adequate composition models are in place. We argue that identifying such objects is relevant for various disciplines and we provide an example in astronomy. We present a mathematical and computational model for the raised problem and provide a first implementation using a parallel architecture.

Keywords Hidden Objects, Big Data, model, approximate sample query, patterns in Big Data.

1) Introduction

In recent years, Big Data has become a new ubiquitous term. It is considered to transform science, engineering, medicine, health care, finance, business, and ultimately society itself. The phenomenon is fostered by a conjunction of factors like reduced costs on persistent storage; ubiquitous access to Internet; deployment of high throughput instruments, and continuous sensor based monitoring. In other words, the impact of the Big Data phenomenon spans most areas from business to science. The availability of information in accessible digital format pushes the development of new approaches that shall enable enriched and improved decision making in every area.

In this paper, we explore one aspect of Big Data that has been given little attention. It is based on the subtile observation that some important information may be blurred in huge datasets. In this context,

objects of interest may emerge as a result of processing the big data files by means of some composition semantics.

Such objects of interest may have been subject to lower level capturing mechanisms, as in the case of sensors, discretizing a phenomenon in space and time. Recovering the original objects from datasets, in which they appear in a transformed version, requires big data analyses.

In order to exemplify this observation, consider the scenarios below:

Example 1. Astronomy catalogues hold billions of sky objects from a region in sky. An astronomer may be interested in elements composing more complex structures, such as constellations or galaxy clusters (Allen, 2011). In this context, a complex structure is blurred amongst billions of individual sky objects. Recovering it requires composing the elements of the catalogue that match some composition criterion.

Example 2. Environmental sensor data is another area with huge datasets of time-series measurements recording, such as: temperature, humidity, wind. Likewise, the primitive objects (i.e. individual measurements) are irrelevant till the hidden patterns inside the sea of data are revealed. For instance, a spatial aggregation of time-series presenting rain level of a region can lead to a conclusion about the occurrence of storms in that region.

Example 3. In seismic interpretation, a huge seismic dataset holds billions of seismic traces, which, for each position in space, present a list of values corresponding to the amplitude of a seismic wave in increasingly deeper depths (i.e. seismic traces). Similarly, a seismic interpreter tries to extract meaning out of huge seismic dataset by finding higher-level seismic objects such as: faults, salt domes, etc. Those *features* may be obtained from the seismic dataset through a smart combination of low-level seismic traces. Indeed, aggregation of seismic traces in a special manner can convey meaning to the user in terms of real seismic objects of interest.

In the above examples, the objects of interest are more complex than the elements held by the target dataset. In the first example, one is looking for a constellation on catalogues of sky objects, whereas in the second one, the weather condition on a region is inferred from multiple sensor measurements. Finally, in the seismic example, a seismic higher-level feature is searched for within a dataset of seismic traces.

Thus, in this paper we first raise this particular aspect of big data, regarding the emergence of higher-level abstractions from a huge dataset of millions of occurrences of elements of the same nature. Next, we discuss possible efficient strategies to search for such elements in huge datasets.

The rest of this paper is organized as follows. Section 2 presents the problem formulation. Next, section 3 presents a use case scenario in astronomy. Section 4 discusses the proposed solution and section 5 presents the implementation and experimented results. In section 6 we mention the related works. Finally, section 7 concludes.

2) Problem Formulation

In this section, we formally introduce the problem of Unveiling objects in Big Data.

2.1) Problem Description

We define the problem of unveiling objects in Big Data as emerging objects in huge datasets composed of basic elements of the same nature that agree on a composition model. The specification of the object is given by a *Sample Query*.

A **Sample Query** specifies the elements that shall compose higher-level objects, and a composition model.

In Example 1, for instance, a constellation defines a *sample query*. The latter specifies the object characteristics that determine each of its components, and a compositional model based on their spatial correlation.

In this context, the problem of unveiling objects can be more precisely formulated as the search for objects in Big Data matching a given sample query.

2.2) Problem Statement

In this section, the problem is mathematically formulated.

Definition 1. A Big Dataset *D is defined as* $D = \{e_1, e_2, \ldots, e_n\}$ in which each e_i, $1 <= i <= n$ is an element of a domain *Dom*. Moreover, for each element $e_i \in D$, $e_i = [atr_1, atr_2, \ldots, atr_m]$, such that atr_j, $1 <= j <= m$, describes a characteristic of e_i.

Definition 2. A sample query Q=[*E,F-shape*]*, such that E*=$\{e_1, e_2, \ldots, e_k\}$ defines the elements that compose a shape of interest, and F-shape is a binary function F-shape(E)$\rightarrow Boolean$.

Definition 3. A matching function F-element$= (E, D) \rightarrow \mathbb{R}$, computes the similarity between two elements q_i, e_j, such that $q_i \in Q.E$ and $e_j \in D$.

Problem statement: Given a Big dataset *D*, with elements in a domain Dom, a sample query *Q*, a matching threshold th_1 and a shape threshold th_2, identify $S=\{s_1, s_2, \ldots, s_l\}$, such that $s_i \subseteq D$, and $|s_i| = |Q.E|$, for all $1 <= i <= l$, and for each $e_j \in s_i$, for all $1 <= i <= l$, it exists $q \in Q.E$, such that $F - element(e_j, q) > th_1$ and F-shape(s_i).

2.3) Mathematical Interpretation

In this section, a general mathematical model is proposed for solving the problem of searching for objects matching the sample query.

The strategy for solving this problem consists of two functions: 1) *F-element* that independently evaluates each element in the dataset against the elements in the sample query and finds a set of matches. *2) F-shape* that constructs possible combination of matched elements according to the model, to form similar shapes.

After executing the *F-element* function over elements of a dataset, we obtain a distance value between an element of the sample query and each one from the dataset, indicating their distance to the desired element in the model. The computed distance is used as a matching cost for every element in the dataset. The *F-shape* function identifies compositions, from individual dataset elements, as defined by the sample query model.

2.4) Computer Science Interpretation

Given the Big Data nature of the problem, partitioning the dataset into smaller units is a must. In this context, the implementation of the two above discussed functions can be mapped into the well-known parallel program paradigm MapReduce (Dean, 2004).

The MapReduce paradigm has been designed to be implemented by a system running on a shared-nothing cluster architecture.

A MapReduce program is composed of a *Map ()* and a *Reduce ()* procedure. The first applies its associated function on each element of a dataset, whereas the latter produces a final output by aggregating the results of the first Map function. In the context of the Unveiling objects problem, the *Map* function performs the behavior of the *F-element function* and the *Reduce* procedure implements the *F-shape function, respectively*.

3) Use case– Unveiling Objects in Science

The concept behind unveiling objects in big datasets is applicable to different areas such as sensor data, seismic data, biology, etc. In this paper, we aim to show one of its applications in the area of astronomy. In the following, first we describe the astronomy context, presenting the dataset and its characteristics. Subsequently, the unveiling objects functions adopted for the astronomy scenario are presented.

Astronomy surveys are interested in investigating regions of the sky. By means of some capturing instrument, such as an optical telescope, sky objects are identified and registered in a large table of sky objects, named after sky catalog.

Surveys make possible statistical studies of large number of objects and enable interesting or rare examples of phenomena to be found, which can then be studied in greater detail. An astronomy catalogue is a dataset that contains a list of celestial objects and their characteristics, like position, flux, magnitude and color. Their spatial coordinates, assigned according to a celestial sphere coordinate system, are used as objects identification. An object positioning is given by is right-ascension (ra) and declination (dec) values. The former assumes values between 0 and 360 degrees, whereas the latter measures its distance from equator between -90 and +90 degrees.

Thus, a sky catalogue can be modeled as a relation, as follows:

Cat (ra,dec, flux, photo-z,u,g,r,i,z,…) (1)

The attributes u, g, r, i, z refer to the magnitude of light emitted by an object. Their values are measured in logarithmic units, through various wavebands, from the ultraviolet to the infrared. The photo-z attribute corresponds to an estimation of the redshift, a measure of the objects distance from earth.

Studying astronomy dataset have relevance in a very broad range of astronomical fields, including the solar system, stars, the interstellar medium, the structure of galaxies and cosmology. To unveil a group of sky elements with special properties or finding a certain sky structure with interesting information in an astronomy dataset, we need some powerful tools capable of revealing those implicit high

level objects out of explicit individual sky object elements.

4) Proposed solution

Thus, given the problem as formulated in section 2.4, the strategy for identifying the set S of shapes matching a sample query Q consists of applying two nested functions:

S= F-shape (F-element (Q.E, D)) (2)

F-element: looks for every element of the query independently in the dataset *D* and finds a set of matches for those elements with different distances with respect to Q.E, indicating their difference to the desired elements in the query. The metric applied to compute the distance between two elements can be varied by the nature of every application domain and the dimensionality of its dataset. A list of some frequently used distance metrics includes: Euclidean distance, Dynamic Time Warping (Berndt, 1994), Hausdorff distance (Huttenlocher, 1993) and Manhattan distance (Krause, 1987).

F-shape: constructs the possible combination of elements out of sets of matches according to some restrictions. These restrictions verify some relationships among the matched elements, such as ordering or distances. Similarly to the previous function, we can employ different distance metrics as well. The composition of possible shapes can be modeled as a hypergraph, in which the hyper nodes are sets of matches of each query element and hyper edges are the relationships between these sets (figure 1). There is an edge between two hyper nodes if their elements obey the same relationship between the corresponding elements in the query; furthermore, if the order of those corresponding elements is vital, the edge would be a directed edge. *F-shape* looks for paths in this directed hypergraph that pass through all hyper nodes.

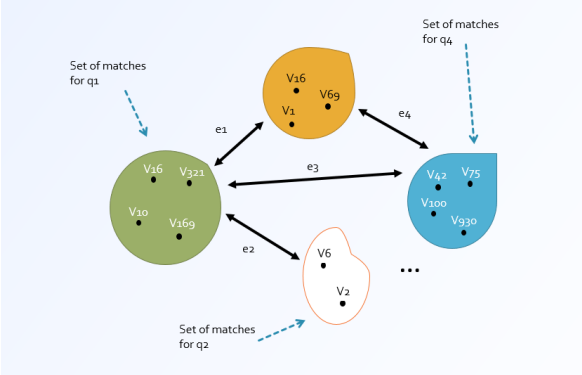

Figure 1. *q*: model elements are the low level objects in the sample query. *V*: hyper nodes are set of matches for every element of query. *e*: hyper edges are relations between the corresponding elements of query.

4.1) Astronomy Implementation

In this section, we elaborate on two functions F-element and F-shape customized to the astronomy application domain. The functions express the criteria for selecting elements of the dataset that match the sample query. The result is composed of elements of an astronomy dataset that describe high-level sky object similar to the sample query, representing a sky complex structure, such as a constellation, a solar system, etc. The description of the higher-level object of interest is indirect and is obtained through marking a set of low-level objects (i.e. elements) in the astronomy dataset. Once the sample query has been defined, it is used in the definition of the unveiling functions, F-element and F-shape.

In this scenario, F-element is a function that evaluates a similarity based predicate on attribute values of each element of the

dataset, comparing with elements of the query. In the astronomy scenario discussed here, a predicate is defined on the value of the *flux* attribute. Ounce the whole astronomy dataset has been evaluated, the F-element function places matched elements in buckets. Each bucket holds elements matching with a sample query element. Accordingly, the F-shape function constructs the possible combinations of elements out of buckets produced by F-element, using a nested-join algorithm (Elmasri, 1989). The join criterion considers the defined restrictions of sky structure, such as spatial distances, to form shapes similar to the sky model. In the astronomy scenario, the model establishes a spatial correlation among elements of the sample query. The composition therein can be assessed by computing the Euclidean distance considering the position of objects as specified by the values of their coordinate in right-ascension (ra) and declination (dec). The pairwise comparison between the space correlation in the model and that produced by joining matched elements in buckets produce candidate solutions.

The details of this use case accompanying the experiments are given in the section 5.

5) Implementation

We adopt the MapReduce model to introduce the two functions used for unveiling objects in big data. MapReduce is a parallel programming paradigm (see Section 2.4). Various software implementations exist, such as Apache Hadoop that materialize the paradigm into a system. Such systems allow developers to write programs that process massive amounts of unstructured data in parallel across a distributed cluster of processors or stand-alone computers. In the following, we describe our implementation in Hadoop. The description will consider the astronomy scenario presented in section 3:

5.1) Hadoop Implementation

This section presents a MapReduce solution to the Unveiling Objects in Big Data problem using a cluster environment: 1) Map function is invoked for each element of the dataset to check whether it matches with elements of the sample query. Indeed, the Map function checks all the matches for every record of dataset in one traversal of the big dataset and then partitions the results between reducers which then will run the Reduce function. The partitioning function operates on the declination (dec) value of every sky object. It splits the sky plane into equal intervals according to the dec value (-90 to +90) of sky objects divided by the number of available slaves in the cluster. In this fashion, we try to pass approximately the same amount of matches to every reducer. 2) Reduce function firstly materializes the input matches by putting them into the separate buckets according their matched element; as a result, every bucket contains all the matches to the corresponding element of the query stored into the disk; secondly, it produces the set of sky objects matching the model by joining the elements in the buckets using the nested-join operation (Elmasri, 1989); finally, it outputs the solutions which passed the join spatial conditions. For the sake of simplicity, in the current implementation, we didn't consider possible solutions in the boundaries of partitions. Instead, we simply look for

solutions with element from the same partition.

5.2) Experimental Results

In this section, we present our experimental results in the context of astronomy data. To run our tests over Map Reduce functions, we used a clustered framework with the following configuration (table 1):

- Programming Language: Java

Property	Master	Slaves Type 1	Slaves Type 2
CPU	Intel Xeon E5 2420, 2.2Ghz	Intel Xeon E5 2620, 2.00Ghz	Intel Xeon E5 2420, 2.2Ghz
# Logical CPUs	16	2	6
# Cores	16*6	2*2	6*6
RAM	10 GB	8 GB	4 GB
Disk	200 GB	200 GB	200 GB
# virtual system of this type	1	4	2

Table 1: Hadoop master and slaves configurations

In figure 2, we show the results comparing the execution time using different dataset [1]sizes over models of sizes 3, 5 and 7 elements. Furthermore, we varied the catalogue size from 0.5 GB to 10 GB.

One may observe an interesting duality as an effect of the F-element and F-shape functions with respect to the size of the model, i.e. the number of elements in the sample query. As the latter increases, the number of F-element invocation also increases, per dataset records, potentially increasing the number of matched records. Conversely, as the model size increases, it becomes more constrained, reducing the potential number of candidate solutions. This is indeed observed in Figure 3.

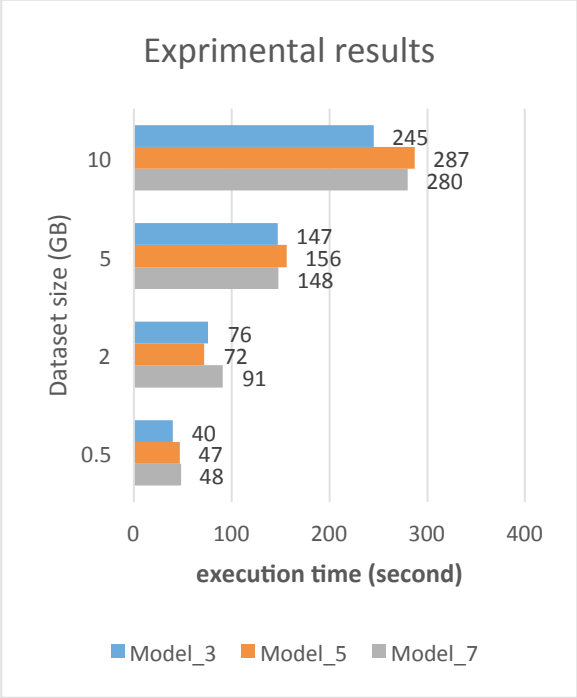

Figure 2. Experimental results

In figure 3, we compare the number of matches versus number of solutions in a dataset size of 500 MB; as it is shown, by increasing the number of elements in the model, the number of matches increases and conversely the number of solutions decreases. We observed the same behavior in other dataset sizes, as well. This is due to the fact that by adding more elements to the model, we add more conditions to our join operator, when constructing solutions, which decreases the probability of a combination to be a solution.

[1] All the datasets has been queried and downloaded from the Sloan Digital Sky Survey (SDSS) - http://skyserver.sdss.org/

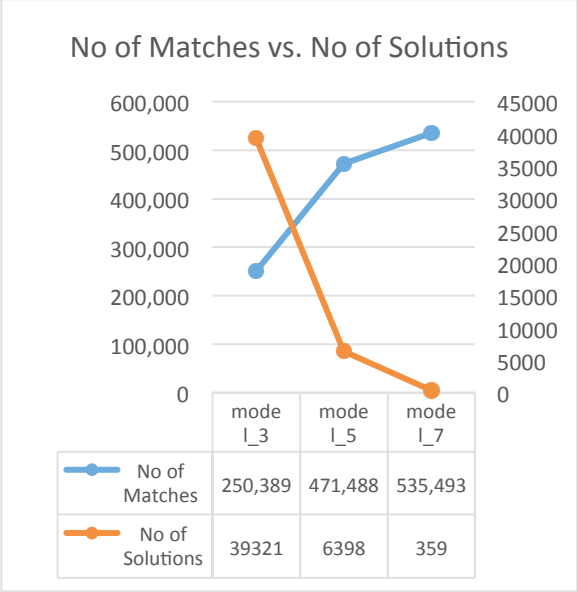

Figure 3. The growth in the number of matches and solutions by increasing the model size.

6) Related works

Some previous work has investigated pattern queries over graphs. In (Zou, 2009), to solve the pattern match query in graph databases, the authors transform the vertices into points in a vector space via graph embedding techniques, converting a pattern match query into a distance-based multi-way join problem over the converted vector space and finally they process the multi-way join operation. In (Zou, 2012), the authors answer the pattern queries through graph embedding. They define the problem as finding the shortest path in a graph.

Our work is based on their work but in a more general approach, which makes it applicable to any area of Big Data and not just over graph databases. Due to this generalization, in our algorithm instead of looking for the same labels within vertices in a graph, we use a similarity approach to measure the similarity of matched points. Indeed, the main difference in our work is that every point in our pattern query has a set of specified attributes that should be matched with the same points in the dataset through a similarity function.

7) Conclusion

In this paper, we presented the unveiling objects in Big Data problem that discovers high-level objects that are blurred in Big data sets. We propose an approach for unveiling high-level objects in Big Data, using two nested functions. The problem is explored using an astronomy scenario, where complex structures, such as constellations are identified out of a catalogue of astronomy objects.

The problem is modeled trough the implementation of two functions: F-element and F-shape. Their composition implements the desired semantics, according to the target application problem. Given the Big Data nature of the problem, the functions are implemented in a state of the art parallel programing model, Map Reduce, enabling robust and efficient computation.

The results of this implementation for different query and dataset sizes are discussed. The first version of our implementation that is presented in this paper was a proof for the functionality of out theory; obviously, there are possible improvements to our functions that we will scrutinize in the future works: 1) ordering the buckets according their sizes; if we check the spatial-distance conditions between the join elements as soon as possible, this will reduce the number of computations by processing the joins efficiently. 2) Early pruning of branches if total-cost gets bigger than defined threshold; in other words, we can define a condition to calculate the current total cost

and if it got bigger than threshold, the program breaks the rest of joins for that combination. Our estimation is that by applying the above ideas, we avoid from many useless computations. In addition, by utilizing buffer management techniques like hash piping, we can avoid from huge I/O operations in the phase of materializing the intermediate results.

References

1. Allen, S., Evrard, A., Mantz, A., 2011, Cosmological Parameters from Observations of Galaxy Clusters, Annual Review of Astronomy and Astrophysics, Vol. 49, pp. 409-470.
2. Berndt, D., Clifford, J., 1994, Using Dynamic Time Warping to Find Patterns in Time Series, KDD workshop.
3. Ciarlini,A., Porto,F., Khatibi,A., Dias, J., 2015,Methods and apparatus for parallel evaluation of patternqueries over large n-dimensional datasets to identify features of interest, patented andapproved by United States Patent and Trademark Office.
4. Dean, J., Ghemawat, S., 2004, MapReduce: Simplified Data Processing on Large Clusters, 6th Symposium on Operating System Design and Implementation, San Francisco, USA.
5. Elmasri, R., Navathe, Sh., Fundamentals of Database Systems, book published by Pearson Education, chapter 5, 1989.
6. Freire, V., De Macedo, J., Porto, F., Akbarinia, R., 2014, NACluster: A Non- Supervised Clustering Algorithm for Matching Multi Catalogues,IEEE e-Science Workshop, Guaruja, SP, Brazil.
7. Huttenlocher, D., Klanderman, G., Rucklidge, W, 1993, Comparing images using the Hausdorff distance, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15.
8. Khatibi, A., Porto, F., 2014, Unveiling objects in Big Data - using similarity approach, poster in International workshop Many Faces of Distances, University of UNICAMP, Campinas, Brazil.
9. Krause, E., 1987, Taxicab Geometry: An Adventure in Non-Euclidean Geometry, Dover Books on Mathematics.
10. Han, J., Haihong, E., Le, G., Du, L., 2011, Survey on NoSQL database, 6th International Conference on Pervasive Computing and Applications (ICPCA).
11. Zou,L., Chen, L., Tamer Özsu, M., Zhao, D., 2009, Distance-Join: Pattern Match Query in a Large Graph Database, VLDB 2009, pp. 886-897.
12. Zou,L., Chen, L., Tamer Özsu, M., Zhao, D., 2012, Answering pattern match queries in large graph databasesvia graph embedding, VLDB Journal 2012, pp. 97–120.