2 Linked Data, Non-relational Databases and Cloud Computing

2.1.Linked Data

The World Wide Web has allowed an unprecedented amount of information to be published and shared globally. The barriers of cost to publish and retrieve the information have greatly lowered, and the interconnection provided by hyperlinks delivers the means to find more related data about a particular topic. Search engines are build on top of those characteristics to rank relevant documents and facilitate locating information to be retrieved [2]. All those factors combined contributed to the unconstrained growth of the Web [3].

These low barriers to multimedia documents, rendered human readable by browsers, are not the same to data. As more data is collected and processed, it is produced on different formats, as Excel tables, CSV files, XML or SQL databases. Three problems arise from this. First, the logic of access and parsing of those contents is different from one another. Excel data, for example, was originally a binary proprietary format, while CSV files allow for different separators and escape codes. Second, the pieces of data stored on those formats have relationships that are more complex than the simple implicit relationship between two hyperlinked documents. A hyperlink does not indicate the type of relationship a document shares with its connected part, and the relationship between pieces of data must be more expressive than just a connection. Finally, as different providers produce databases, the schema of those databases does not necessarily end up being similar to other providers working on the same kind of data. To join those databases, a translation between those schemas will need to be made, adding up the cost of making large amounts of data available.

Linked Data was proposed to lower some of those barriers and make way for this data to be available on the Web [4]. The principles of the proposal, the "Linked Data principles", are a set of rules for publishing data on the Web:

1. Use URIs as names for things;
2. Use HTTP URIs so that people can look up those names;

3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL);
4. Include links to other URIs, so that they can discover more things.

Typically Linked Data uses RDF documents to contain data [5]. The relationship between that data is also described on RDF to make typed links between the data nodes. The resulting data, described this way and available on the Web, is the Web of Data.

Since these principles of Linked Data were laid out, the public availability of information on the Semantic Web grew quickly [6]. Many organizations, institutes and others contribute to the collective Web of Data, building and publishing RDF repositories that they had available previously only on traditional models. Those repositories encompass different kinds of data, such as movie databases (e.g., LinkedMDB and iMDB), New Testament names (e.g., Semantic Bible), encyclopaedic articles (e.g., DBPedia from Wikipedia), biological taxonomy (e.g., GeoSpecies Knowledge Base), and many other examples [7].

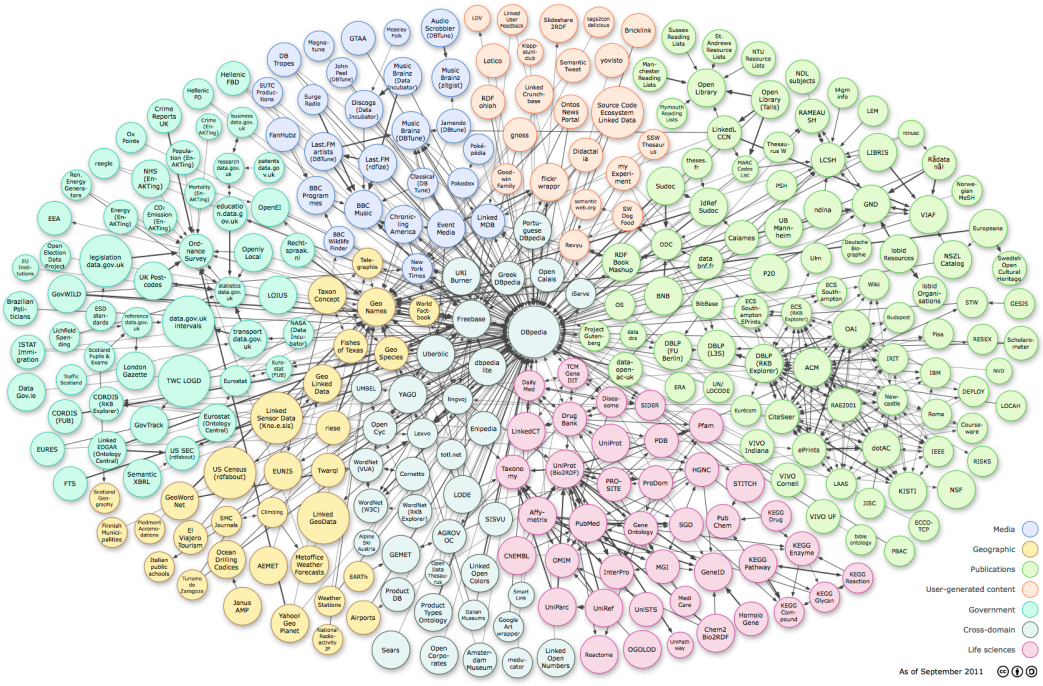

Figure 1 - The Linking Open Data Cloud Diagram [6]

2.1.1.Generic data model and graph representation

As described above, RDF is the basic format for data documents and databases that compose the Web of Data. In its most basic form, the generic data model described in RDF is a data triple in the subject-predicate-object format.

RDF can be serialized in several formats. The most recognized one is the XML syntax described alongside the RDF definition by W3C [8]. But other formats for serialization, such as Notation 3 and N-Triples, also exist with applications in different scenarios.

For example, Figure 2 is a fictional RDF document. The XML headers of that listing indicate that it is an RDF document and the syntax version it uses. The document then describes two CDs (i.e., two subjects) with information like artist and price (i.e., several predicate-objects). If represented as N-Triples, the same listing would start out without any headings about syntax, assuming the parser already knows its format. Then each line of the file would define a triple, with URIs between angle brackets, literals between quotes, and the whole line finished by a full stop, as it can be seen in the example in Figure 3.

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">

<rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist>Bob Dylan</cd:artist>
  <cd:country>USA</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>10.90</cd:price>
  <cd:year>1985</cd:year>
</rdf:Description>

<rdf:Description
rdf:about="http://www.recshop.fake/cd/Hide your heart">
  <cd:artist>Bonnie Tyler</cd:artist>
  <cd:country>UK</cd:country>
  <cd:company>CBS Records</cd:company>
  <cd:price>9.90</cd:price>
  <cd:year>1988</cd:year>
</rdf:Description>
.
.
.
</rdf:RDF>
```

Figure 2 - Two CD's listing example in RDF XML [28]

```
<http://fake/cd/Empire Burlesque> <http://fake/cd/artist> "Bob Dylan" .
<http://fake/cd/Empire Burlesque> <http://fake/cd/country> "USA" .
<http://fake/cd/Empire Burlesque> <http://fake/cd/company> "Columbia" .
<http://fake/cd/Empire Burlesque> <http://fake/cd/price> "10.90" .
<http://fake/cd/Empire Burlesque> <http://fake/cd/year> "1985" .
<http://fake/cd/Hide your heart> <http://fake/cd/artist> "Bonnie Tyler" .
<http://fake/cd/Hide your heart> <http://fake/cd/country> "UK" .
<http://fake/cd/Hide your heart> <http://fake/cd/company> "CBD Records" .
<http://fake/cd/Hide your heart> <http://fake/cd/price> "9.90" .
<http://fake/cd/Hide your heart> <http://fake/cd/year> "1988" .
```

Figure 3 - Two CD's example listing in N-Triples

Also defined in the W3C conceptualization is the graph data model to represent the generic data model. In this representation, a subject-predicate-object triple expresses a node-arc-node link, as seen on Figure 4. The subject and the object are nodes of that link, and the predicate is a directed arc in the link connecting the subject node to the predicate node. The whole set of triples, expressed in this manner, results in a RDF graph that can be visually represented as a traditional graph.

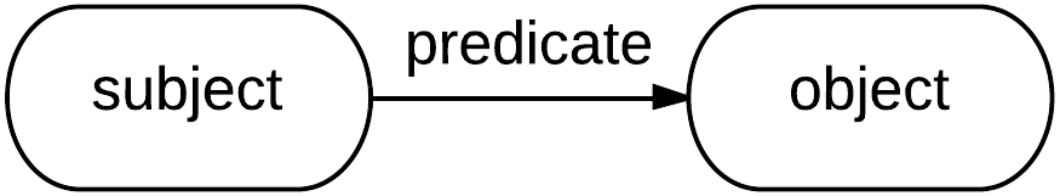

Figure 4 - Subject-Predicate-Object graph representation

The graph representation presents several advantages. First, the visual description of the information can convey the information for immediate human consumption, whereas the original triples set didn't. The visual display can adapt graph visualization techniques to arrange nodes by proximity of their relations, weight nodes with more relations from them, and clustering nodes among other techniques [9]. For data processing, the RDF graph representation allows the application of distributed computing for storage and retrieval of its data, which will be explored in latter sections.

2.2.Non-relational databases

Large amounts of data are being produced and stored through various means and a large proportion of that in an unstructured or semi-structured way. However, the availability of that data is an issue for some reasons.

First, the current dominant storage and retrieval paradigm of data in corporate and government environments is the relational model, and the task to

adapt semi-structured data to a fully functional relational model is cumbersome, with questionable results [10]. The resulting database schemas might be either too deconstructed and fall back to the basic RDF schema (e.g., a triples schema on SQL), or result in too many levels of hierarchy to avoid being too vague (e.g., a city list that relates to a locations list that relates to a entities list). In both cases, it means that deep knowledge of the original domain rules is still needed to use the database.

Second, unstructured data, in particular those in binary format (e.g., video), tends to be used in units that are similar to those in which they are stored (e.g., video contained in files). As such, the task to locate and retrieve those units is usually very simple, requiring little to no use of the relational model itself. The real problems in those cases are on storage and later processing of those units of data.

Finally, the availability of that data on current internet scale, requiring low response times even under the regime of tens of thousands of parallel queries, is a challenge to conventional relational model. One of the strongest set properties of relational databases is the ACID - atomicity, consistency, isolation and durability. Those properties guarantee that database transactions are processed reliably. Assuring all those properties, at the same time on parallel and distributed database nodes on top of replication and load balancing techniques, is time consuming and often only a subset of them is enough.

2.2.1. NoSQL

Non-relational databases have been used and studied since the late 1960s, but recently a renewed focus has been given to them under the label of NoSQL. The term covers a broad range of database types that will be described below. NoSQL is often interpreted as "not only SQL" to avoid the unintentional meaning of being against SQL and relational databases. Instead it is meant to be the use of non-relational databases for problems less suited for relational databases.

The set of properties commonly seen at modern non-relational databases is described as BASE - Basically Available, Soft-state and Eventual consistency - in contrast with relational ACID properties. In short, a trade-off is made, giving

away some of ACID properties to regain availability and performance. That trade-off is described under the CAP-Theorem. CAP is acronym standing for:

- Consistency: the nodes of the system are expected to be all at a consistent and equivalent state after the execution of an operation.
- Availability: the system is expected to be highly available and is able to continuously answer any kind of operations, even if some nodes are unavailable.
- Partition Tolerance: in case of a network split, it creates isolation between the nodes of the system, and continuous operation is expected after that network partition and eventual restoration.

The CAP-Theorem states that only two of those three characteristics can be chosen for a shared-data system. Based on this, non-relational databases compared to relational databases choose to have availability and partition tolerance over consistency. As such, a very common description used for NoSQL databases is that they are eventually consistent and inherently scalable, even though not all of them choose those characteristics from the CAP set.

2.2.2.Non-relational databases types

Non-relational databases can be categorized by the model of data they represent. These types are just guidelines for better understanding how they work, and some databases are a hybrid of those models. The most basic types are Key-Value, Document and Graph. On top of those are more specific types as Tabular, Object, Multi-value, Tuple etc. The basic types can be described as:

- Key-Value: typically consists of data indexed by a string key that is used to retrieve a unit of data of a primitive type (i.e., an integer or another string). Inherently schema-less, it depends on the application to properly use the data in a meaningful way. Most implementations add functionality over the simplistic model, such as more complex primitive types (e.g., Redis' data structures) or customizable synchronization and replication model (e.g., Cassandra's clusters-token-replication implementable factors).
- Document: is the type of store which the central concept is a multi-valued document or object. In comparison with Key-Value stores,

here the database is expected to have functionalities that can be used to query or modify data over fields of a document instead of making use only of indexed keys. This type of store can be further categorized by the way objects are organized and grouped on its database, such as collections, tags or hierarchies. Also, they can be differentiated by the underlying document representation, such as JSON, multi-value rows or XML datasets. Examples of different functionalities are map-reduce support over queried data (e.g., MongoDB's aggregation and data processing) and bi-directional data replication and synchronization (e.g., CouchDB's partition, offline and reunion operations).

- Graph: is the kind of database designed for storing and querying data nodes and relations interconnecting these nodes, i.e., data that can be well represented as a graph.

2.2.3.Data Sharding

Data sharding is a common approach, upon in which effective use of non-relational databases is based. This approach consists of relying on the application to indicate how data should be distributed among the many nodes that compose the database system network. The rationale behind this is that, even though the database can use some natural keys to better distribute load and optimize data retrieval, these are never going to be as efficient as those designed taking the application into consideration.

The support to sharding varies between databases. Some do not give any support, relying entirely on the application (e.g., Redis). Other give limited access to parameterize default sharding algorithms provided by the database (e.g., MongoDB) [11]. Finally, there are databases with extension points to plug in custom made modules that dictate to the database how to distribute, rebalance, and replicate data based, not only on keys, but on information provided by the database about its network topology (e.g., Cassandra).

2.3.Cloud Computing

Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources, such as networks, servers, storage, applications, and servers, which can be rapidly provisioned and released with minimal effort or service provider interaction [12]. The emergence of the current model is historically associated with the evolution of Amazon's system architecture. Amazon began their business selling books online, but quickly expanded their offers to other retail options and, as demand grew, so did their computational needs. But demand is highly seasonal, peaking between Thanksgiving and Christmas. In the traditional model of capacity planning, the allocated resources to Amazon's systems had to target the peaks of access, leaving the investment in the allocated hardware underused during most of the year. Realizing this, Amazon began to remodel their architecture towards an on-demand computational shared pool of resources. In 2006, Amazon began renting their excess computational resources [13] with the easy to remember description: "Sometimes you need a lot of processing power, and sometimes you need just a little. Sometimes you need a lot, but you only need it for a limited amount of time."

That model quickly evolved to the current availability of public clouds by providers other than Amazon's Web Services (AWS). Services such as Netflix and Dropbox explore those resources extensively, basing their whole operations on third-party's public cloud infrastructure. The impact of the new model is changing industries, such as television [14] and providing a special boost to e-Science.

2.3.1.MapReduce

One of the most useful techniques for cloud computing based systems is MapReduce [15]. This programming model is well suited for large processing and generating large datasets, and several computational tasks can be expressed in this model with some adaptations. Systems specified to use this model have a *map* function that should process a key-value pair and generate a set of intermediate key-value pairs that can be processed in parallel, and they also have a *reduce* function to merge the intermediate results.

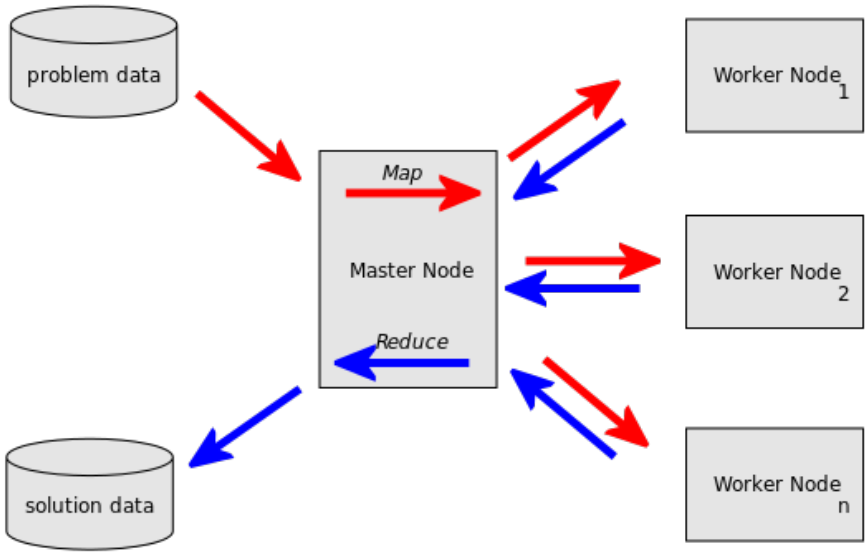

Figure 5 - MapReduce Overview (Wikimedia Commons)

Once a computational problem is adapted and expressed in this model, it is automatically ready to parallel execution on large clusters of computers. There are several benefits of this approach: allowing programmers to write parallel and distributed systems to easily use the resources of clusters; making use of heterogeneous, commodity machines instead of relying on homogeneous clusters; providing a uniform approach to implement distributed systems, which allow the development of frameworks to automate and support the execution of applications built to the MapReduce model.

This last benefit has been extensively explored. Any general Job Queues and Workers framework can enable a MapReduce application. The model consists on implementing a user defined *perform* method and then building up a pool of worker processes on several machines. Each of those workers periodically access a shared job queue, marks a job and uses the jobs associates inputs to execute its own *perform* method. Once he finishes, he clears the job from the queue and takes another one. A monitoring process can check if workers are active and set jobs as failed or retry them if staled. Resque is an example of such framework and is employed on many computational problems for regular background jobs execution and also on MapReduce [16].

Even though MapReduce can be executed in any general Job Queues framework, several specialized frameworks now exist, and they have grown to cover other aspects of distributed systems with modules to handle distributed filesystems, databases, and to load balancing and provisioning. Hadoop is an example of such framework and is employed by several organizations [17].

2.4.Summary

In this chapter, we reviewed the two areas of software engineering involved on this dissertation, Linked Data and Cloud Computing. First, we discussed the foundation ideas behind Linked Data and their impact on the unprecedented availability of data on the Web of Data. We then skimmed over the general Linked Data RDF model and discussed non-relational databases, how they differ from relational databases and how they scale well while still having all the necessary characteristics to store large RDF datasets. We finally describe how the Cloud Computing is currently evolving and explored the MapReduce technique that enables an uniform and accessible way to harness the computational power available on the public clouds, using very simple Job Queues and Workers frameworks or specialized frameworks. In the next chapter, we provide an overview on Linked Data keyword search and discuss a simplified definition of the tensor-based approach to search on large RDF datasets.