On the design of a Globally Distributed, Locally Compressed Knowledge Base System

Olivier Curé
LIGM (UMR 8049), CNRS, ENPC, ESIEE, UPEM, Marne-la-Vallee, France.
olivier.cure@u-pem.fr

1. INTRODUCTION

Most big data processing are addressed by distributing the data and the execution of programs over a cluster of commodity hardware. These processing are generally handled by a cluster computing engine based on the MapReduce approach, *e.g.*, Apache Hadoop. We consider that many big data related operations can be performed on a single machine if sufficient effort is put into compressing the data and ensuring that the programs can efficiently manipulate the compressed data, ideally in a decompression-free manner. Of course, very large data sets may not fit on a single machine and in these cases, distributing the compressed data over a cluster of machines is necessary. This potentially has the advantage of requiring less computing nodes but still being able to maintain performance for similar jobs.

We qualify this approach as **Globally Distributed, Locally Compressed** (GDLC) and in this paper, we provide insights on using it to design a new breed of graph database systems. This popular member of the NoSQL store family is usually modeled either by the so-called property or RDF (Resource Description Framework) data models. We consider that the design of this GDLC graph store is a good opportunity to take the best of these two models, *i.e.*, enabling attributes on graph edges, providing a declarative query language based on SPARQL and/or (Open)Cypher (in opposition to the procedural Gremlin language) and proposing inference services using vocabularies associated to the data sets (hence referring to a knowledge base (KB) rather than a graph store). Moreover, we consider that guaranteeing scalability, fault-tolerance and high availability properties as well as benefiting from a set of diverse libraries, *e.g.*, machine learning, graph and stream processing, is a must have for such systems.

2. RESEARCH OPPORTUNITIES

We consider four interconnected research opportunities:

Compression. Recently, several important theoretical results have been obtained on highly compressed data structures, *e.g.*, the family of so-called Succinct Data Structures (SDS). Some of these data structures, *e.g.*, wavelet trees, FM-Index, provide a high compression rate together with decompression-free operations, *e.g.*, access, rank and select. In order to represent graph-structured data, combinations of these data structures may be required. Moreover, some of these data structures, *e.g.*, wavelet trees, can be used as indexes. We consider that the proper composition of these structures can motivate a single index approach for these KBs, as opposed to multiple indexes RDF stores, *e.g.*, RDF-3X and its 15 indexes.

Partitioning. A main goal is to limit network communication between computing nodes during query processing and reasoning operations. Compression has a positive impact of data shuffling (by reducing the volume of exchanged data). Nevertheless, advanced automatic partitioning approaches such as the analysis of query workloads, graph processing, *e.g.*, connected components or graph partitioner, or machine learning, *e.g.*, k-means, should replace the classical, often manual, sharding solutions.

Query processing. Combinations of SDS operations can be used to match the expressiveness of the main constructors of the SPARQL and (Open)Cypher query languages. Efficient scheduling of these operations is a first step toward optimizing query processing. It is also important to consider the impact of processing updates at both the data set and the vocabulary levels. Note that the efficient management of dynamic SDS is an open problem.

Reasoning. In the KB context, reasoning is tightly related to query processing. In general, two approaches are considered: materializing all inferences in the store or reformulating queries. Both have pros and cons. Finding a trade-off between them can have an important impact on the overall performance of query processing.

3. CONCLUSION

The GDLC graph store approach is a step forward on the design of innovative knowledge base systems. We started to work on a first prototype that is based on the Apache Spark engine. It thus permits to enjoy libraries such as MLlib, GraphX and Spark streaming. In particular, we have implemented a semantic-aware encoding system that supports inferences for the RDFS++ vocabulary (RDFS plus sameAs, inverse and transitive properties), identified optimization techniques for SPARQL on top Spark as well as a partitioning approach combining hashing and query workload analysis. We believe that we can lean on these positive results to pursuit the vision of a GDLC knowledge base.