Schema inference for massive JSON data sets

Dario Colazzo, LAMSADE
Univertité Paris Dauphine
dario.colazzo@dauphine.fr

Carlo Sartiani, DIME
Università della Basilicata
sartiani@gmail.com

Context

Cloud computing is a novel and very popular computing paradigm that aims at building extremely scalable and elastic applications working on huge datasets. This paradigm is based on the idea of using hundreds or thousands of low-end, unreliable, and cheap machines connected through standard network switches. The most popular incarnation of this paradigm is the Map/Reduce architecture, first introduced by Google and then adopted by companies like Facebook and Amazon.

While cloud computing applications can be written in standard programming languages like Java or C++, there has been much work in the development of special-purpose languages for the Cloud, such as Pig Latin [1] or Sawzall [2], that hide low-level details.

Cloud-specific languages are essentially untyped, and no guarantees about the correctness of complex jobs can be obtained at static time. This is particularly frustrating when multiple jobs form a single complex workflow processing huge datasets, as it happens in clustering and machine learning computations. In this case, a statically undetected mismatch between the output of a job J_i and the input of the next job J_{i+1}, or between the output returned by Map worker and the expected input of reducers, can lead to incomplete and/or erroneous results, hence wasting large amounts of CPU time. These mismatches are very hard to detect for the programmer, as she usually has to manually inspect the logs of Map and reduce workers.

It is therefore crucial to extend cloud-specific languages with the typing mechanisms offered by modern programming languages. An *important* step towards this direction is the design of a data typing algorithm that could automatically infer a schema for a dataset or assist the programmer in designing it ; key properties of the inferred types should be *succinctness* and *precision*, which are not easy to ensure in this setting.

Project

We aim at devising, studying and implementing a schema inference algorithm for large data sets represented by means of JSON [3], one of the most used data formats currently used in Big Data applications. In order to make schema inference passible in the presence of huge JSON collections, techniques in the MapReduce ecosystems can be used. In a preliminary work [4], we defined a possible MapReduce algorithm for this problem. However, implementing the algorithm in the Java-based Hadoop system [5] revealed to be tedious as Java has poor support for symbolic manipulation, which is crucial for schema inference. Also, this work is still at a very preliminary stage and many important aspects still need to be dealt with.

In this internship project, in order to tackle the problem of efficiently inferring schemas from huge JSON collections we aim at using Spark [6], a recent system enabling general-purpose, large-scale data processing. Spark allows for running programs written in the Scala [7] programming language, which is particularly suitable for symbolic manipulations performed by schema inference algorithms. Also, Spark outperforms Hadoop-MapRedcue in many contexts, and we expect that this holds in our setting.

Références

[1] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, and Andrew Tomkins. Pig latin : a not-so-foreign language for data processing. In Jason Tsong-Li Wang, editor, *SIGMOD Conference*, pages 1099–1110. ACM, 2008.

[2] Rob Pike, Sean Dorward, Robert Griesemer, and Sean Quinlan. Interpreting the data : Parallel analysis with sawzall. *Scientific Programming*, 13(4) :277–298, 2005.

[3] Javascript object notation (JSON). `http://json.org`.

[4] Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. Typing massive json datasets. In *International Workshop on Cross-model Language Design and Implementation (XLDI, affiliated with ICFP)*, 2012.

[5] Hadoop. `http://hadoop.apache.org/`.

[6] Spark. `https://spark.apache.org`.

[7] Scala. `http://www.scala-lang.org`.