DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

JIT Query Operators for Complex Data using GraalVM

Keywords: Raw Data, GraalVM, Truffle, Just-In-Time Code Generation

Motivation: Scientific and industrial datasets are growing in volume and complexity. In the past, most datasets consisted primarily of tabular data and were well served by relational database systems. Nowadays, datasets combine tabular data with hierarchical data (e.g., JSON, XML) and array data (e.g., from IoT or machine-learning). Moreover, this data is often semi-structured, i.e., fields may be present or missing arbitrarily in the data, or the types stored in each field may vary inside the collection. Developing a high-performance query engine for such data presents novel challenges. The goal of this thesis is to implement a system design that copes efficiently with such data and high-performance.

Approach: Developing a query engine is an intricate balance of systems engineering as it requires trade-offs between delivering a high-performance implementation while ensuring ease of implementation and code maintainability for the core query engine. When datasets were "simpler" - data had a fixed, well-defined tabular structured - the focus was in developing a full-stack query engine. Code maintainability was a lesser concern because the problem being solved was smaller in scope. The research literature [1][2] contains many examples of techniques used in this situation, which focus on low-level techniques to achieve high performance. These techniques include query compilation or the use of vectorization CPU instructions. However, the types of data have changed, and these techniques, while remaining valid, don't cope with complex data formats nor with semi-structured data. Moreover, they are incredibly hard to implement in practice or to extend with new operations or requirements - to the point where the implementation overhead dwarfs the potential performance benefit. Therefore, new system engineering approaches are required. Our approach is to leverage techniques used in just-in-time compilation for the development of high-performance query engines.

Just-in-time compilation

JITs are a popular implementation technique for interpreted and dynamic languages. Java/JVM was the first widely popular example. More recently, web browsers, which require fast performance in Javascript, have also made significant advances in JITs. The idea of just-in-time compilation is simple: when the compiler knows the type of data that is actually flowing through the system, it can best generate code to cope with that data. If assumptions change - e.g., the data types change - then the compiler can adapt

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

automatically. While JITs have an underlying overhead, they also can support optimizations that are impossible in other scenarios. For instance, given an array of floating-point numbers being read from a file that must be summed, the compiler could detect - after some iterations - that all numbers are actually "ints" and have no floating-point precision, and therefore generate SIMD-friendly code for the operation based on ints, resulting in a faster query processing.

Just-in-time query operators

Implementing a JIT is a complex task, but fortunately there are various JITs that can be used as foundational frameworks. The most interesting of this is GraalVM, in particular its combination with the Truffle sub-project. Truffle is a library for building abstract syntax tree interpreters. Truffle allows users to implement an AST interpreter, which can be then specialized at runtime to produce highly-efficient code. Truffle, presumably, solves the other part of the problem: code maintainability of the core engine, and extensibility.

Goal:

The goal in this project is to implement a small subset of query operators - e.g. Scan, Filter, Projection - for a few common formats, such as CSV and JSON, in Truffle. These operators will be developed as Truffle interpreted nodes, and subsequently they will be specialized by the framework at runtime. By supporting CSV, we can assess the performance in "classical tabular data", and compare performance with other existing systems. By supporting JSON, we can support more complex hierarchical data.

Requirements:

- Familiarity with database architecture desired but not required.
- Familiarity with compilers desired but not required.
- Familiarity with GraalVM or Truffle desired but not required.

Milestones:

1. Define simple AST query operators to implement, jointly with the project advisor.
2. Implement query operators in Truffle.
3. Performance analysis for varying data distribution and structure parameters.
4. Compare with alternative implementations - interpreted, as well as hand-coded.
5. Presentation

Timeline:

To be defined

DIAS: Data-Intensive Applications and Systems Laboratory
School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne
Building BC, Station 14
CH-1015 Lausanne
URL: http://dias.epfl.ch/

References:

1. P. Boncz et al. "MonetDB/X100: Hyper-Pipelining Query Execution". In CIDR 2005.
2. T. Kersten et al. "Everything You Always Wanted to Know About Compiled and Vectorized Queries But Were Afraid to Ask". In PVLDB 2018.

References: Prof. Anastasia Ailamaki, anastasia.ailamaki@epfl.ch
Responsible collaborators: Miguel Branco, miguel@raw-labs.com
Periklis Chrysogelos periklis.chrysogelos@epfl.ch
Panagiotis Sioulas panagiotis.sioulas@epfl.ch
Duration: 6 months