Einreichung für: BTW 2017,
Geplant als Veröffentlichung innerhalb der Lecture Notes in Informatics (LNI)

Invest Once, Save a Million Times — LLVM-based Expression Compilation in PostgreSQL

(Invited Demonstration)

Dennis Butterstein[1] Torsten Grust[1]

We demonstrate how a surgical change to PostgreSQL's evaluation strategy for SQL expressions can have a noticeable impact on overall query runtime performance. (This is an abridged version of [BG16], originally demonstrated at VLDB 2016.)

The evaluation of scalar or Boolean expressions often takes the backseat in a discussion of query processing although table scans, filters, aggregates, projections, and even joins depend on it. SQL expression evaluation in vanilla PostgreSQL is based on the database kernel's family of `Exec···` functions that form an interpreter. During query runtime, this interpreter repeatedly walks expression trees whose inner nodes represent SQL operators while leaves carry literals or table column references. This evaluation strategy has long been identified as CPU-intensive and outright wasteful on modern computing and memory architectures, leading to frequent pipeline flushes and instruction cache pollution. The resulting interpretation overhead is significant and may dominate all other tasks of the query processor: PostgreSQL indeed spends the *lion share of execution time* on expression computation when it processes selected TPC-H queries.

The present work is an exploration of how PostgreSQL can benefit if we **trade SQL expression interpretation for compilation** and thus turn repeated run time effort into a one-time query compile time task. Cornerstones of the approach are:

- The PostgreSQL query optimizer itself remains unchanged—expressions are compiled after planning and just before query execution starts. We adopt a non-invasive approach that—outside of expression evaluation—retains PostgreSQL's Volcano-style pipeline query processor. Compiled and interpreted expression evaluation coexist; both can contribute to the execution of the same query.
- The just-in-time compilation of expressions is based on the LLVM compiler infrastructure which comes in shape of a library that we link with the original PostgreSQL code—LLVM offers high-quality code generation at low compilation times.
- We invest in code size to save runtime effort—a tradeoff that the SQL expression context admits but traditional compilers for programming languages would shy away from.

The resulting runtime gains are made tangible in terms of an interactive demonstration setup that allows cursory as well as deeper looks under the hood of PostgreSQL 9, both pre- and post-surgery.

References

[BG16] Butterstein, D.; Grust, T.: Precision Performance Surgery for PostgreSQL—LLVM-Based Expression Compilation, Just in Time, *Proceedings of the VLDB Endowment* 9/13, 2016.

[1] Universität Tübingen, Department of Computer Science, Database Systems Research Group, Tübingen, Germany, ⟨`firstname`⟩.⟨`lastname`⟩`@uni-tuebingen.de`