A JIT for PySQLite

Carl Friedrich Bolz
Software Development Team
King's College London

April 24, 2015

While DataBase Management Systems (DBMSs) are highly optimised, the communications between a user program and a DBMS are often costly: traditional DBMSs require inter-process calls; embedded DBMSs normally prevent Just-In-Time (JIT) compilation across the boundary between user program and database.

As an exploratory case study, we took SQLite, an embedded DBMS, and replaced part of the C code in its core interpreter loop with RPython code, leaving the rest of the database implementation intact. Together with the RPython JIT, this makes it possible to generate machine code for the execution of SQL queries in SQLite on the fly.

This has two effects: on the one hand, SQL query execution is speed up, on average by about 10% on the TPC-H benchmark set. On the other hand, this makes it possible to drastically speed up the integration with PyPy, the Python interpreter written in RPython. When using this jitted SQLite, the JIT will dynamically inline SQL query execution into the surrounding Python code, optimizing the data passing between Python and the database. Equivalently, SQL functions and aggregates that are written in Python can be inlined into SQL code. In this way, our SQLite implementation has a more dramatic effect on queries from Python to SQPyte, which are speed up by a factor of 2.9 on average.

These results suggest that applying JIT technology to database, particularly in their interaction with other languages, holds a lot of performance potential. In addition, the results show the benefits of using high-level frameworks for JIT construction: the SQLite JIT, together with the integration into PyPy was realized in just a few person-months, and is only about 5000 LOC large.