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/

Limiting interference of code generation to analytical query execution through remote JIT-ing

Keywords: Analytical query processing, just-in-time code generation, remote execution

Problem: To achieve efficient analytical query execution, modern analytical DBMS engines have utilized code generation [3, 1] to reduce the overheads of query execution. Through Just-In-Time (JIT) compilation, these engines produce code tailored to each incoming query and minimize the overheads of interpretation and intermediate result materialization during query execution. Nevertheless, previous studies [2] show that code generation itself can be responsible for a significant percentage of the query-to-result time.

Project: As a solution, Kohn et al. [2] proposed adaptive code generation, to decompose code generation into multiple steps with each step being more time consuming that the previous one but also producing more efficient code. For small queries, the code generated by the first "fast" steps are executed, while for long-running queries, the code from the first steps is incrementally replaced by more optimized code generated by the latter steps.
Unfortunately, adaptive code generation requires the compilation to continue even after query execution starts, causing interference to the actual execution. Even in the absence of adaptive JIT compilation, overlapping code generation with query execution of previous queries can cause interfere and result into performance degradation.
This project is composed of two parts. In the first part, the student will analyse the characteristics of code generation with respect to the usage of system resources like memory bandwidth, cache utilization and processing resources as well as quantify the performance degradation caused by overlapped code generation and query execution. The analysis will take place in Proteus [1], an in-house analytical DBMS engine. In the second part, the student will decompose the system to allow remote code generation. Thus, providing isolation between the query execution and code generation by running code generation in a different machine, possibly much smaller, like a desktop/laptop or even an embedded processor.

Plan:

1. Micro-benchmark the interference between code generation and execute queries
2. Micro-benchmark the result of isolating into a different socket/as a different process
3. Using LLVM's remote JITing, generate code in one machine and run it on another machine
4. Modify DBMS to allow a low-footprint version of it to run in a low-end machine for JITing
5. Measure the impact on latency as well as on total throughput of queries and micro-benchmark.

Supervisor: Prof. Anastasia Ailamaki, anastasia.ailamaki@epfl.ch
Responsible collaborator(s): Hamish Nicholson, hamish.nicholson@epfl.ch

Duration: 14 weeks

References

[1] M. Karpathiotakis, I. Alagiannis, and A. Ailamaki. Fast Queries Over Heterogeneous Data Through Engine Customization. PVLDB, 2016.
[2] A. Kohn, V. Leis, and T. Neumann. Adaptive execution of compiled queries. In ICDE, 2018.
[3] T. Neumann. Efficiently Compiling Efficient Query Plans for Modern Hardware. PVLDB, 2011.