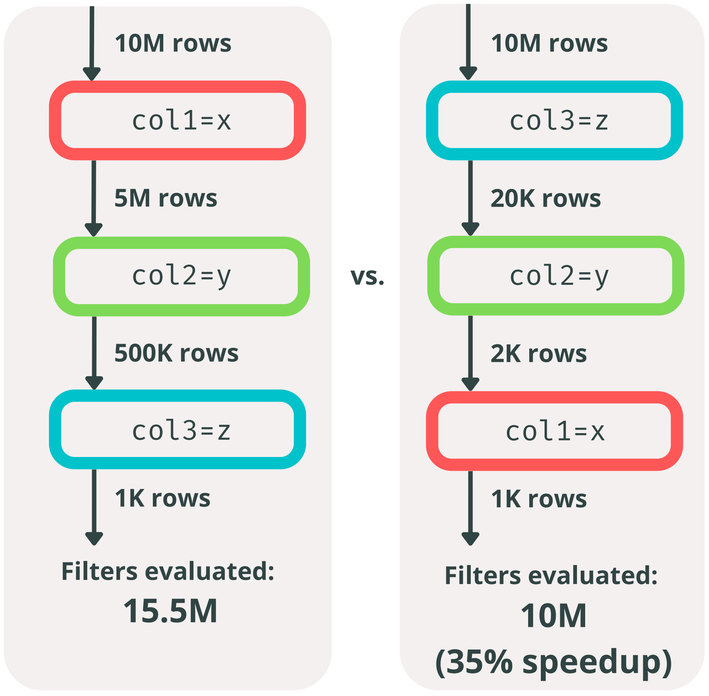
Adaptive Filter Reordering for JIT-Compiled Queries in PostgreSQL

Kai Franz (kfranz@andrew.cmu.edu)
Advised by Todd C. Mowry and Andrew Pavlo

Motivation

- Filtered SELECT database queries
  - Conjunction of several filter clauses
    - e.g. **`WHERE col1=x`**
  - Evaluation order affects query latency
- Optimal ordering is data-dependent
  - Often unknown until runtime

```
SELECT * FROM foo
  WHERE col1=x
    AND col2=y
    AND col3=z
```

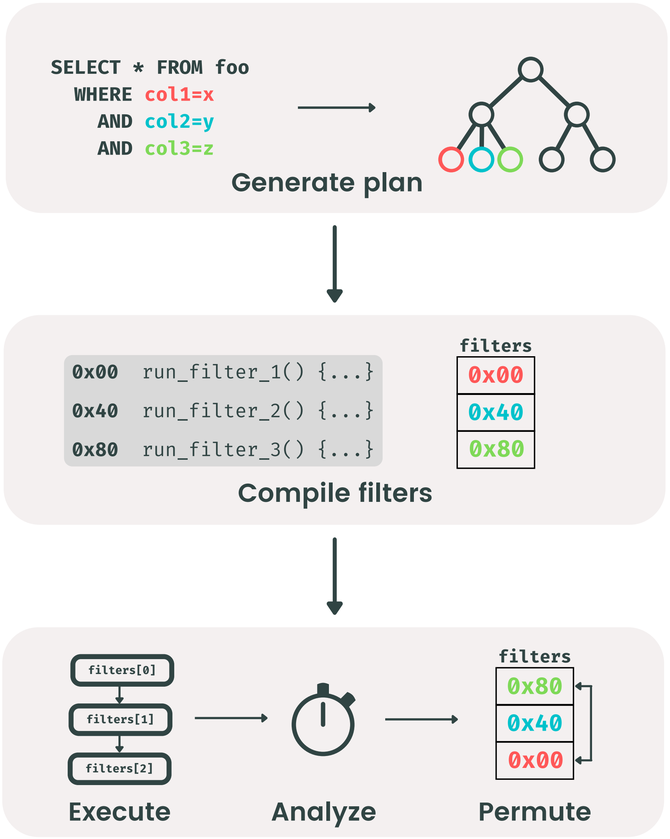


References

1. Menon, Prashanth, et al. “Permutable Compiled Queries.” Proceedings of the VLDB Endowment, vol. 14, no. 2, 2020, pp. 101–113., https://doi.org/10.14778/3425879.3425882.

Permutable Compiled Queries[1]

- Compile each filter as a separate function
- Keep table of pointers to each filter
- Periodically gather statistics for each filter
  - Execution time
  - Selectivity
- Reorder table entries based on statistics
- Allows for runtime permutation
- Avoids expensive recompilation

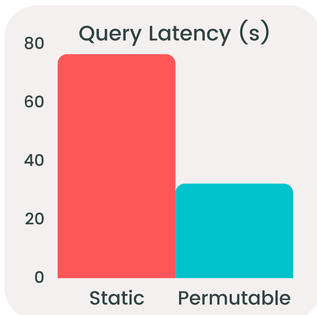


Implementation

- Modify PostgreSQL's JIT compiler
  - Subexpressions of qual expressions
  - Build separate functions in LLVM
- First time expression is evaluated
  - Compile LLVM module
  - Initialize filter table
- Modify PostgreSQL executor
  - Add filter manager that executes filters as usual
    - Collects runtime statistics
    - Permutes filter table

Evaluation

- Microbenchmark: in-memory table with 3 64-bit integer columns, 10M rows
- Query: 3 filters with identical execution time
  - Varying selectivities
    - 88%, 92%, 0.5%
- Adaptive reordering finds optimal filter ordering that static optimizer misses



Future Work

- More runtime optimizations
  - Adaptive Aggregations
  - Adaptive Joins
- Reuse of compiled query plans

Carnegie Mellon University