ACM SIGMOD Programming Contest 2018

Quickstep Team* @ University of Wisconsin-Madison

Jianqiao Zhu, Zuyu Zhang, Dylan Bacon, Jignesh M. Patel (advisor)

{jianqiao, zuyu, dbacon, jignesh}@cs.wisc.edu

1. Contest Overview

Task: Evaluate as fast as possible batches of SPJA (Selection-Projection-Join-Aggregation) queries on a set of immutable relations.

Each query involves up to 4 relations. Aggregate functions are always SUM without GROUP BY.

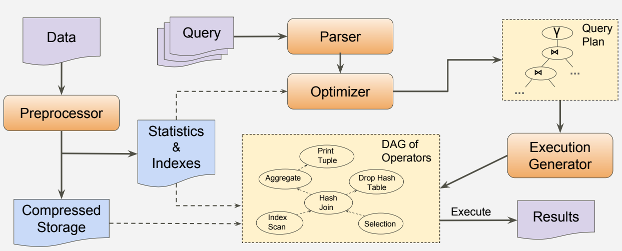
Testing machine configuration:

2x Intel Xeon E5-2660 v2 (2.2 GHz), 20 cores / 40 hyperthreads, 256 GB DDR3 RAM

2. Our Approach

The main design philosophy is that no single data structure / algorithm wins. Thus from the high level perspective we build a system with clearly abstracted modules to address this complexity of supporting a rich collection of *indexes*, *specialized operators*, and *parallel execution strategies*.

System Architecture:



Key Mechanisms:

- No "silver bullet" operator / index -- need to implement a rich collection of them and design proper rules to choose.
- Apply aggressive operator fusion to avoid the cost of materializing intermediate results.

3. Preprocessing

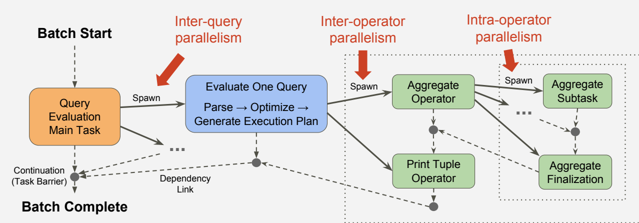
- Calculate min/max values of each column. Meanwhile compress the column if possible (simply truncating the leading zeros).
- Build existence bitmap for each column, and use it to count the number of unique values in the column.
- Use existence maps to figure out containment relationship among columns.
- Build various indexes on each relation (*primary key index*, *foreign key index*, *count vector index*) based on relation size.

4. Optimizer

- ~20 optimization passes.
- Some of the optimization rules: *filter pushdown*, *early projection*, *range propagation*, *predicate simplification*, *semi-join elimination*, *common aggregate-expression elimination*.
- Heuristic-based join order optimization.
- Identify the shape of multi-relation joins (*multi-way join*, *linear join*, *star join)* and apply corresponding specialized operators.

5. Execution

- Build a scheduler that supports dynamic task DAG spawning to fully utilize CPU resources.



- Implement a collection of relational operators. Some regular operators are: *select*, *hash join*, *sort merge join*, *index scan*, *index lookup join*.
- When applicable, fuse multiple binary joins and the top-level aggregation into one multi-relation join-aggregate operator. Example operators are *multiway-join-aggregate* and *linear-join-aggregate*.

6. Contest Workload and Results

Dataset	Small	Medium	Large	X-Large
Size	9.4 MB	89.5 MB	3.9 GB	6.6 GB
# Relations	14	12	29	34
# Queries	50	146	50	146
# Batches	5	21	5	20
Execution Time (s)	0.027	0.133	0.547	1.475

* This work was supported in part by CRISP, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, and grant FA8650-15-C-7562 also from DARPA.