Lessons for the optimizer from TPC-DS benchmark

Sergei Petrunia
Query Optimizer developer
MariaDB Corporation

2019 MariaDB Developers Unconference
New York

The goals

1. Want to evaluate/measure the query optimizer
2. Hard to do, optimizer should handle
   - Different query patterns
   - Different data distributions, etc
3. How does one do it anyway?
   - Look at benchmarks
   - Or “optimizer part” of the benchmarks

Benchmarks

1. sysbench
   - Popular
   - Does only basic queries, few query patterns
2. DBT-3 (aka TPC-H)
   - 6 tables, 22 analytic queries
   - Was used to see some optimizer problems
   - Limited:
     - Uniform data distribution, uncorrelated columns
     - ...

TPC-DS benchmark

- Obsoletes DBT-3 benchmark
- Richer dataset
  - 25 Tables, 99 queries
  - Non-uniform data distributions
- Uses advanced SQL features
  - 32 queries use CTE
  - 27 queries use Window Functions
  - etc
- Could not really run it until MariaDB 10.2 (or MySQL 8)

MariaDB still can’t run all of TPC-DS

- 2 Queries: FULL OUTER JOIN
- 10 Queries: ROLLUP + ORDER BY problem (MDEV-17807)

```
select
 ...
group by
  a,b,c with rollup
order by
  a,b,c

ERROR 1221 (HY000): Incorrect usage of CUBE/ROLLUP
and ORDER BY
```

- ~20 more queries have fixable problems
  - “Every derived table must have an alias”, etc

Oracle MySQL and TPC-DS

- ROLLUP + ORDER BY is supported since 8.0.12
- Doesn’t support FULL OUTER JOIN (2 queries)
- Doesn’t support EXCEPT (1 query)
- Doesn’t support INTERSECT (3 queries)

Running queries from TPC-DS

- Data generator creates CSV files
  - Adjust #define for MySQL/MariaDB
- Query generator produces “streams” from templates
  - A set of QueryNNN.tpl files
  - A stream is a text file with one instance of each of the 99 queries
  - One can add hooks at query start/end
- Queries have a few typos
- There’s no tool to run queries/measure time
  - Note that the read queries are a subset of benchmark (TpCX$)

Getting it to run

- A collection of scripts at https://github.com/spetrunia/tpcds-run-tool
- The goal is a fully-automated run
  - MariaDB, MySQL, PostgreSQL
- Because we need to play with settings/options

Test runs done

- The dataset
  - Scale=1
  - 1.2 GB CSV files
  - 6 GB when loaded
- The Queries
  - 10..20 “Streams”
- Tuning
  - Innodb_buffer_pool=8G (50% RAM)
  - shared_buffers = 4G (25% RAM)

Test results

Test results

- ...

Test results

- … a bit inconclusive – query times varied across my runs (?)
- Time to run one stream = 20 min – 2 hours
- Searching for the source of randomness
  - Started to work on full automation
    - (did I run ANALYZE? Did I have correct with my.cnf parameters?)
  - Started to look at **rngseed** in dataset/query generator

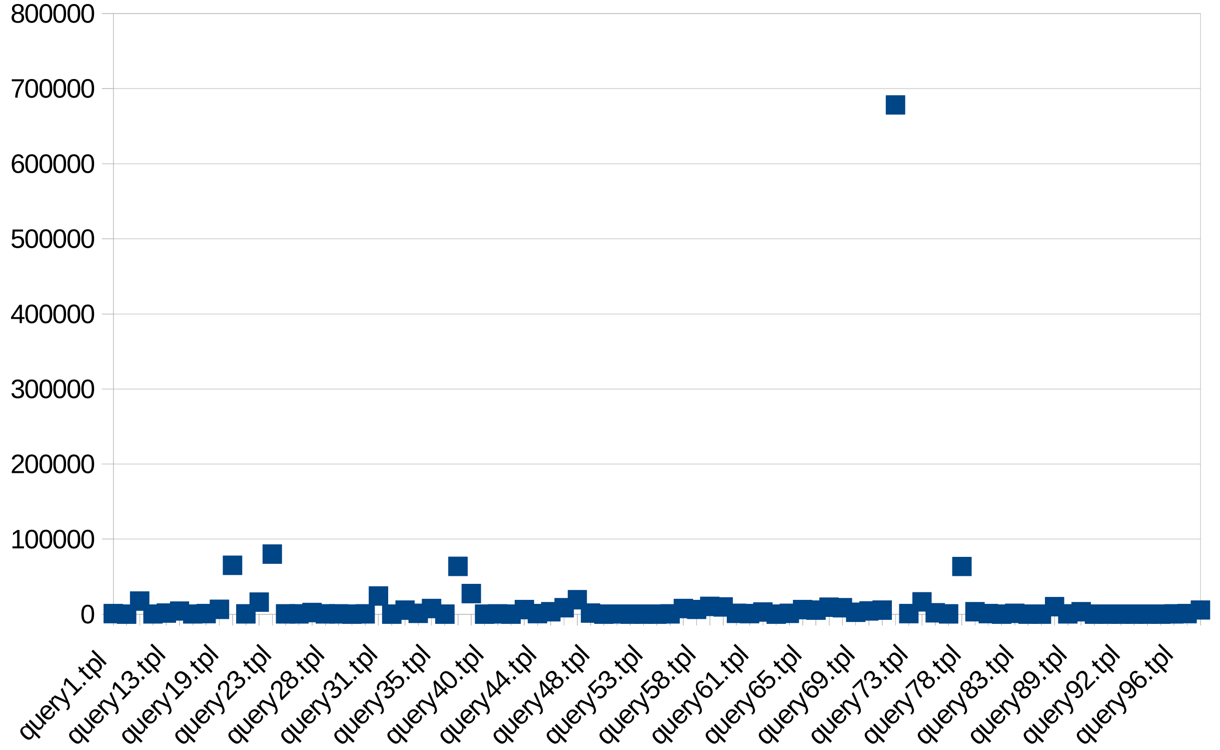
MariaDB/MySQL

MariaDB 10.2, 10.4, MySQL 8

- Scale=1, 6.1 GB data, 8G buffer pool
- **rngseed**=1234 for both
- Benchmark takes ~20 min
- Query times are very non-uniform

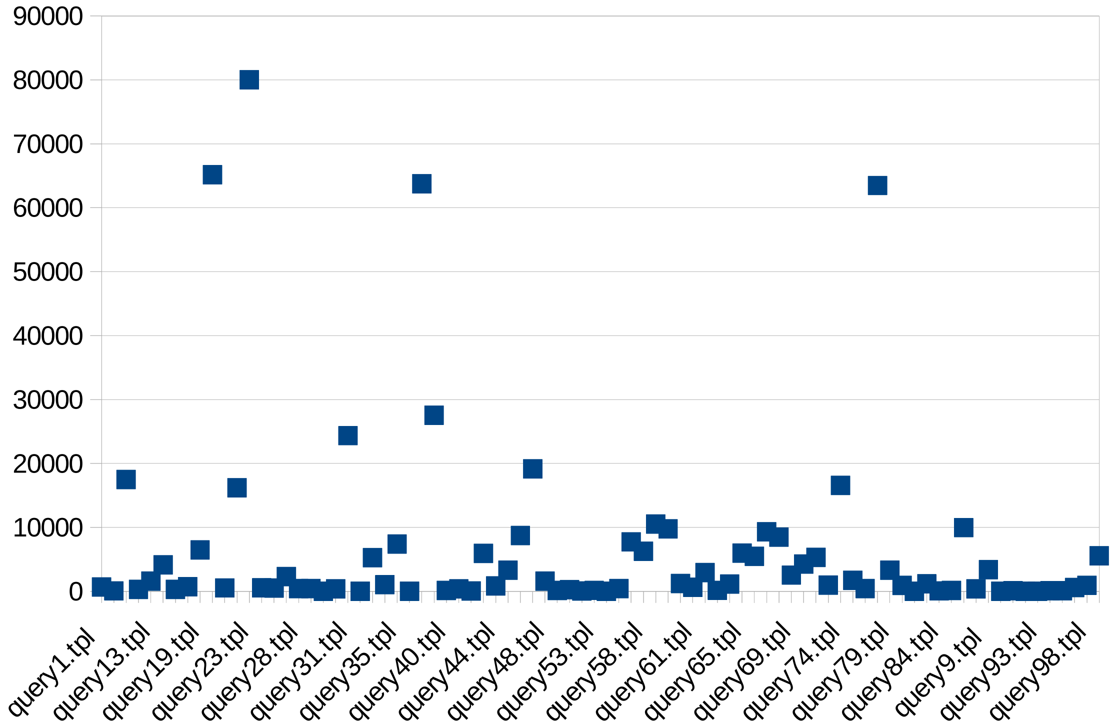
```
+--------------+----------------+
| query_name   |   QueryTime_ms |
+--------------+----------------+
| query72.tpl  |        678,321 |
| query23.tpl  |         80,025 |
| query2.tpl   |         65,156 |
| query39.tpl  |         63,761 |
| query78.tpl  |         63,473 |
| query4.tpl   |         27,549 |
| query31.tpl  |         24,344 |
| query47.tpl  |         19,156 |
| query11.tpl  |         17,484 |
| query74.tpl  |         16,571 |
| query21.tpl  |         16,212 |
| query59.tpl  |         10,522 |
| query88.tpl  |          9,965 |
```

Query#72 dominates



aDB

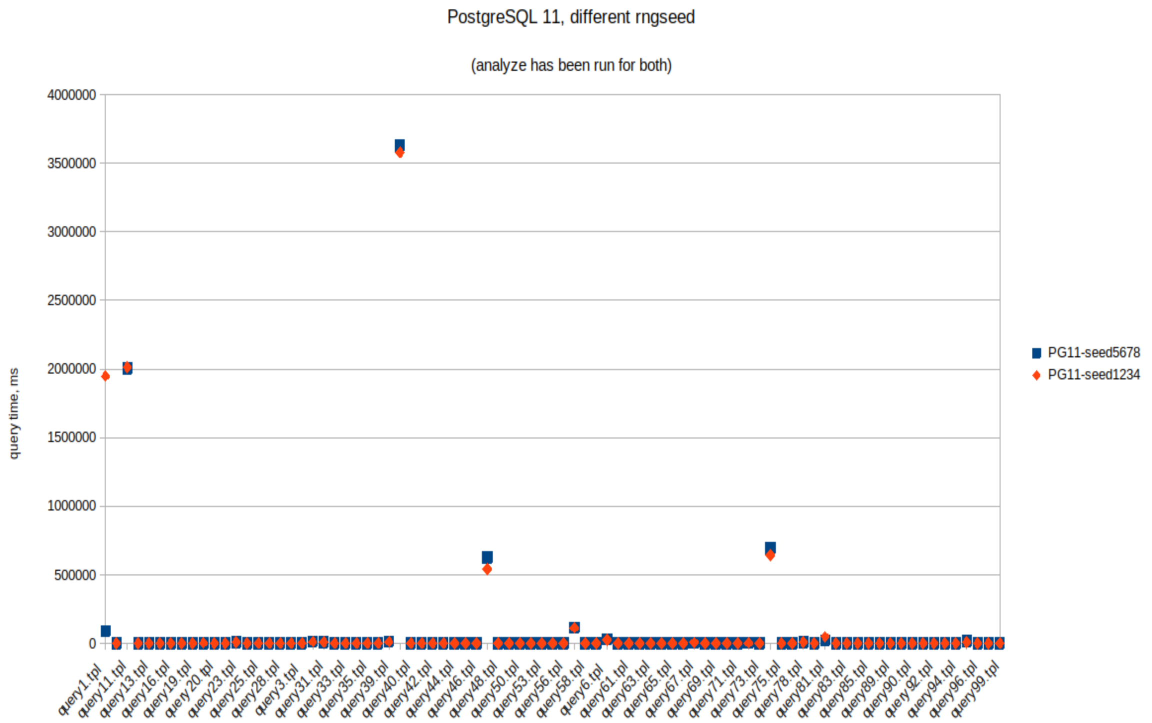
Without Query #72



PostgreSQL 11

PostgreSQL 11

- There was a “fast” run
- Showing results from the last two runs (both where “slow”)
  - **rngseed**=5678 for both – 121 min
  - **rngseed**=1234 (data), **rngseed**=4321 (query) – 145..154 min.



Heaviest queries in the run

```
+-------------+-----------------+-----------------+---------+
| query_name  |   PG11-seed5678 |   PG11-seed1234 | X       |
+-------------+-----------------+-----------------+---------+
| query4.tpl  |       3,628,830 |       3,578,944 | 1.0139 |
| query11.tpl |       2,004,392 |       2,013,597 | 0.9954 |
| query1.tpl  |          87,981 |       1,947,624 | 0.0452 |
| query74.tpl |         693,784 |         641,696 | 1.0812 |
| query47.tpl |         624,717 |         539,941 | 1.1570 |
| query57.tpl |         116,570 |         112,472 | 1.0364 |
| query81.tpl |          22,089 |          47,366 | 0.4663 |
| query6.tpl  |          27,896 |          27,009 | 1.0328 |
| query30.tpl |          11,214 |          11,171 | 1.0038 |
| query39.tpl |          10,803 |          10,702 | 1.0094 |
| query95.tpl |          16,418 |          10,065 | 1.6312 |
```

- Execution time varies
- Is this a query optimizer issue?
- Or different constants in a skewed dataset?

- Do we need a “representative collection of datasets”?
  - Check N datasets?

Compare most heavy queries

MariaDB

```
+-------------+---------------+
| query_name  |  QueryTime_ms |
+-------------+---------------+
| query72.tpl |       678,321 |
| query23.tpl |        80,025 |
| query2.tpl  |        65,156 |
| query39.tpl |        63,761 |
| query78.tpl |        63,473 |
| query4.tpl  |        27,549 |
| query31.tpl |        24,344 |
| query47.tpl |        19,156 |
| query11.tpl |        17,484 |
| query74.tpl |        16,571 |
| query21.tpl |        16,212 |
| query59.tpl |        10,522 |
```

PostgreSQL

```
+-------------+-----------------+-----------------+--------+
| query_name  |   PG11-seed5678 |   PG11-seed1234 | X      |
+-------------+-----------------+-----------------+--------+
| query4.tpl  |       3,628,830 |       3,578,944 | 1.0139 |
| query11.tpl |       2,004,392 |       2,013,597 | 0.9954 |
| query1.tpl  |          87,981 |       1,947,624 | 0.0452 |
| query74.tpl |         693,784 |         641,696 | 1.0812 |
| query47.tpl |         624,717 |         539,941 | 1.1570 |
| query57.tpl |         116,570 |         112,472 | 1.0364 |
| query81.tpl |          22,089 |          47,366 | 0.4663 |
| query6.tpl  |          27,896 |          27,009 | 1.0328 |
| query30.tpl |          11,214 |          11,171 | 1.0038 |
| query39.tpl |          10,803 |          10,702 | 1.0094 |
| query95.tpl |          16,418 |          10,065 | 1.6312 |
`
```

- Some queries are present in both lists, but some are only in one.
- Not clear

Observations about the benchmark

- **rngseed** on the dataset matters A LOT
  - What is a representative set of rngseed values?
- **rngseed** on query streams – much less
- Hardware?
- Queries are not equal
  - Heavy vs lightweight queries
  - Is SUM(query_time) an adequate metric?
    - Wont see that a fast query got 10x slower

Other observations

- Both DBT-3 and TPC-DS workloads are relevant for the optimizer
  - Condition selectivities
  - Semi-join optimizations
  - …
- But don’t match the optimizer issues we see
  - ORDER BY … LIMIT optimization
  - Long IN-list
  - …

Extra: parallel query in PG?

Extra – PostgreSQL 11, parallel query?

- Trying on a run with both rngseed=5678:
- Parallel settings

```
max_parallel_workers_per_gather=8  (the default was 2)
dynamic_shared_memory_type=posix
show max_worker_processes= 8
```

- Results
  - Only saw one core to be occupied
  - The run still took 121 min, didin’t see any speedup

Try a parallel query

```
select
  sum(inv_quantity_on_hand*i_current_price)
from
  inventory, item
where
  i_item_sk=inv_item_sk;
```

- max_parallel_workers_per_gather=0

```
                                  QUERY PLAN
---------------------------------------------------------------------------------
 Aggregate  (cost=301495.25..301495.26 rows=1 width=32)
   ->  Hash Join  (cost=1635.00..213408.54 rows=11744894 width=10)
         Hash Cond: (inventory.inv_item_sk = item.i_item_sk)
         ->  Seq Scan on inventory  (cost=0.00..180935.94 rows=11744894 width=8)
         ->  Hash  (cost=1410.00..1410.00 rows=18000 width=10)
               ->  Seq Scan on item  (cost=0.00..1410.00 rows=18000 width=10)
```

Try a parallel query

```
select
  sum(inv_quantity_on_hand*i_current_price)
from
  inventory, item
where
  i_item_sk=inv_item_sk;
```

- max_parallel_workers_per_gather=8

```
                                            QUERY PLAN
----------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=125048.98..125048.99 rows=1 width=32)
   ->  Gather  (cost=125048.55..125048.96 rows=4 width=32)
         Workers Planned: 4
         ->  Partial Aggregate  (cost=124048.55..124048.56 rows=1 width=32)
               ->  Parallel Hash Join  (cost=1468.23..102026.87 rows=2936224 width=10)
                     Hash Cond: (inventory.inv_item_sk = item.i_item_sk)
                     ->  Parallel Seq Scan on inventory  (cost=0.00..92849.24 rows=2936224 width=8)
                     ->  Parallel Hash  (cost=1335.88..1335.88 rows=10588 width=10)
                           ->  Parallel Seq Scan on item  (cost=0.00..1335.88 rows=10588 width=10)
```

Try a parallel query

```
select
  sum(inv_quantity_on_hand*i_current_price)
from
  inventory, item
where
  i_item_sk=inv_item_sk;
```

- Results
  - max_parallel_workers_per_gather=8: 1.0 sec
  - max_parallel_workers_per_gather=0: 3.8 sec
- Didn’t see anything like that in TPC-DS benchmark

Thanks!

MariaDB