pg_pathman – extension for PostgreSQL partitioning

Alexander Korotkov, Ildar Musin

Postgres Professional

2016

Why should we work around constraint exclusion mechanism?

Constrain exclusion mechanism problems we work around with pg_pathman:

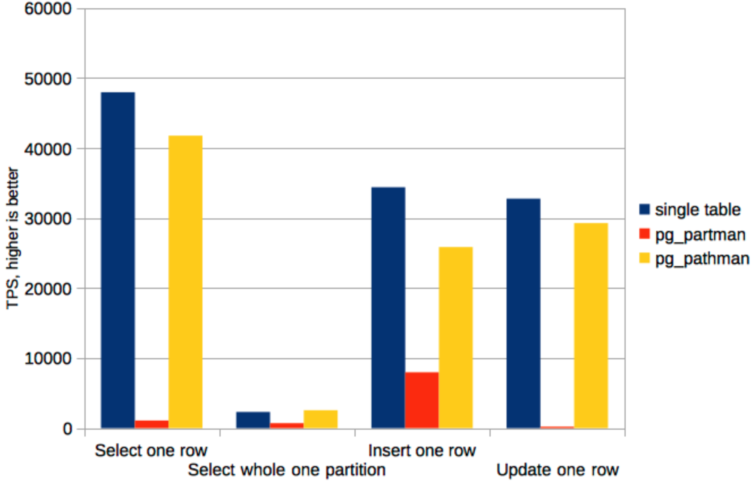
- Very slow planning on many partitions: iterate each partition during query planning;
- Query conditions are pushed “as is” to the partitions;
- Support of limited types of partitioning. For instance, HASH-partitioning isn’t supported;
- Don’t support runtime partitions selection.

What does pg_pathman do better?

- Fast partition selection
- Better filter condition processing
- Runtime partitions selection

We're working on making it a part of future declarative partitioning.

Fast partition selection



http://akorotkov.github.io/blog/2016/03/18/pg_pathman-update-delete-benchmark/

Better filter condition processing: pg_partman

```
# EXPLAIN SELECT * FROM test WHERE
  (ts >= '2015-02-01' AND ts < '2015-03-15') OR
  (ts >= '2015-05-15' AND ts < '2015-07-01');
 Append  (cost=0.00..5028.22 rows=128059 width=41)
   ->  Seq Scan on test  (cost=0.00..0.00 rows=1 width=40)
         Filter: (((ts >= '2015-02-01 00:00:00'::timestamp without time zone)
   ->  Seq Scan on test_2  (cost=0.00..1183.40 rows=40320 width=41)
         Filter: (((ts >= '2015-02-01 00:00:00'::timestamp without time zone)
   ->  Bitmap Heap Scan on test_3  (cost=444.46..1266.02 rows=20178 width=41)
         Recheck Cond: (((ts >= '2015-02-01 00:00:00'::timestamp without time
         ->  BitmapOr  (cost=444.46..444.46 rows=20178 width=0)
               ->  Bitmap Index Scan on test_3_ts_idx  (cost=0.00..430.07 row
                     Index Cond: ((ts >= '2015-02-01 00:00:00'::timestamp wit
               ->  Bitmap Index Scan on test_3_ts_idx  (cost=0.00..4.30 rows=
                     Index Cond: ((ts >= '2015-05-15 00:00:00'::timestamp wit
   ->  Seq Scan on test_5  (cost=0.00..1310.80 rows=24360 width=41)
         Filter: (((ts >= '2015-02-01 00:00:00'::timestamp without time zone)
   ->  Seq Scan on test_6  (cost=0.00..1268.00 rows=43200 width=41)
         Filter: (((ts >= '2015-02-01 00:00:00'::timestamp without time zone)
```

Better filter condition processing: pg_pathman

```
# EXPLAIN SELECT * FROM test WHERE
  (ts >= ’2015-02-01’ AND ts < ’2015-03-15’) OR
  (ts >= ’2015-05-15’ AND ts < ’2015-07-01’);

 Append  (cost=0.00..3248.59 rows=0 width=0)
   ->  Seq Scan on test_2  (cost=0.00..780.20 rows=0 width=0)
   ->  Index Scan using test_3_ts_idx on test_3  (cost=0.29..767.99 rows=0 wi
         Index Cond: (ts < ’2015-03-15 00:00:00’::timestamp without time zone
   ->  Seq Scan on test_5  (cost=0.00..864.40 rows=0 width=0)
         Filter: (ts >= ’2015-05-15 00:00:00’::timestamp without time zone)
   ->  Seq Scan on test_6  (cost=0.00..836.00 rows=0 width=0)
(7 rows)
```

Nested loop join: WITHOUT RuntimeAppend node

```
# EXPLAIN ANALYZE SELECT * FROM q JOIN journal j ON q.dt = j.dt;
-------------------------------------------------------------------------
 Hash Join  (cost=27.50..35978.91 rows=1052270 width=56) (actual time=0.696..
   Hash Cond: (j.dt = q.dt)
   ->  Append  (cost=0.00..21482.70 rows=1052270 width=49) (actual time=0.008
         ->  Seq Scan on journal_1 j  (cost=0.00..58.80 rows=2880 width=49) (
         ->  Seq Scan on journal_2 j_1  (cost=0.00..58.80 rows=2880 width=49)
.....................................................................
         ->  Seq Scan on journal_366 j_365  (cost=0.00..20.70 rows=1070 width
   ->  Hash  (cost=15.00..15.00 rows=1000 width=8) (actual time=0.249..0.249
         Buckets: 1024  Batches: 1  Memory Usage: 48kB
         ->  Seq Scan on q  (cost=0.00..15.00 rows=1000 width=8) (actual time
 Planning time: 28.302 ms
 Execution time: 277.620 ms
(374 rows)
```

Nested loop join: WITH RuntimeAppend node

```
# EXPLAIN ANALYZE SELECT * FROM q JOIN journal j ON q.dt = j.dt;
-----------------------------------------------------------------------
 Nested Loop  (cost=0.28..482.15 rows=1052270 width=56) (actual time=0.043..4
   ->  Seq Scan on q  (cost=0.00..15.00 rows=1000 width=8) (actual time=0.008
   ->  Custom Scan (RuntimeAppend)  (cost=0.28..0.46 rows=1 width=49) (actual
         ->  Index Scan using journal_349_dt_idx on journal_349 j  (cost=0.28
               Index Cond: (dt = q.dt)
         ->  Index Scan using journal_342_dt_idx on journal_342 j  (cost=0.28
               Index Cond: (dt = q.dt)
.....................................................................
         ->  Index Scan using journal_353_dt_idx on journal_353 j  (cost=0.28
               Index Cond: (dt = q.dt)
 Planning time: 29.631 ms
 Execution time: 5.304 ms
(495 rows)
```

Stay tuned, be involved

- pg_pathman is available at github: `https://github.com/postgrespro/pg_pathman`
- News are published in my blog: `http://akorotkov.github.io/blog/categories/pg-pathman/`
- Current status is beta, not production ready. Any help with testing is very welcome.

Thank you for attention!