Performance issues related to migration from Oracle

Pavel Stěhule

Pavel Stěhule

- PostgreSQL extensions
  - `Orafce`, `plpgsql_check`
- Pager `pspg`
- Some patches to Postgres
  - Variadic arguments, default parameters,
    mixed and named notation for passing parameters
  - Functions `format`, `xmltable`, ...
- Some patches to PLpgSQL
  - `RETURN QUERY`
  - `GET STACKED DIAGNOSTICS`

Performance issues

- Are expected
- When you migrate Oracle‘s optimized application to Postgres
- It‘s different database

Possible problems

- Different storage
- Different planner
- Different optimizer
- Different executor

Different storage

- Oracle like Postgres uses MGA, but there is diffent implementatin
- Postgres – fast rollback, slower UPDATE
- Oracle – slow rollback, faster UPDATE

Long transactions

- Postgres needs VACUUM
- Extra long transactions can block VACUUM
- Close all transactions when it is possible immediately (Postgres is much more sensitive on too long transactions (days).

Different storage

- UPDATE is more expensive on Postgres
- UPDATE on Postgres touch all indexes on table
- Don‘t update same values in cycle, when it is possible
- Reduce number if indexes how much it is possible

Planner

- Oracle try to fix developer‘s bugs or ORM bugs (reduce useles selfjoins, ..)
- PostgreSQL doesn‘t do this
- Fix queries, write queries well

Planner

- Postgres know nothing about functions (functions are black box)
- Oracle (MSSQL) has support for some functions (COALESCE, ...)

EXAMPLE

```
– CLEAN BUT WITH PERCENTUAL ESTIMATION
SELECT * FROM tab WHERE coalesce(col, 0) = 0;

– WITH STATISTICS BASED ESTIMATION
SELECT * FROM tab WHERE col IS NULL OR col = 0;

– some strange is in design – mix different states together
```

Optimizer

- Postgres optimizer is fast by default, but little bit lazy
- Oracle optimizer is slower, but returns better plans – the problems with speed are solved by plan cache (other source of problems)

Optimizer

- Is possible to change limits of PostgreSQL optimizer
  - FROM_COLLAPSE_LIMIT
  - JOIN_COLLAPSE_LIMIT
  - GEQO_THRESHOLD
  - RANDOM_PAGE_COST
  - EFFECTIVE_CACHE_SIZE
  - WORK_MEM

Attention

- Unrealistics configuration can do problems

Executor

How to fix it

- Be creative
- Run VACCUM ANALYZE first
- Maybe VACUUM FULL
- Detect a problem and try to ask in mailing list
- PostgreSQL indexes are strong
  - Partial indexes,
  - Functional indexes (statistics)
  -

Indexes

```
CREATE INDEX ON tab (id) WHERE state = ‘active‘;

CREATE INDEX on tab((upper(name)));
```

Check

- pg_stat_user_indexes
- pg_stat_user_tables
- EXPLAIN SELECT ...

Tools

- Slow query detection
  - pg_stat_statement
  - log_min_duration_statement
  - auto_explain
- Reports
  - PgFouine
  - PgBager
- Sharing plans
  - Explain.depecs.com

Question

```
CREATE TABLE foo(a int);
ANALYZE foo;

EXPLAIN SELECT * FROM foo;

— how much rows are expected in EXPLAIN?
```

Why some queries are fast on Oracle and not on PG?

- Different planners (estimators)
  some patterns are better supported by Oracle (nvl, aggpush down, ..), some ugly patterns are supported by O.
- Different optimizers
  from_collapse_limit, join_collapse_limit, GEQO
- Missing implicit plan cache on Postgres
- Slower start of queries on Postgres
  due simple and more dynamic implementation
- Extra intensive UPDATEs are slower on Postgres