Learn from Peers: PostgreSQL (i.e. Postgres)

Kian Win Ong

Survey (Part 1 of 2)

1. Why are you interested in learning Postgres?

   a. I don't know yet.
   b. I want to build a web application for a demo.
   c. I want to tune Postgresql so that my prototype is fast.
   d. I want to tune Postgresql for fair comparisons in my experiments.
   e. I have other reasons.

Survey (Part 2 of 2)

2. How well do you understand the EXPLAIN command?

   a. I have not used it before.
   b. I understand when a SQL query is using indexes.
   c. I understand the different relational operators.
   d. I understand the different join algorithms.
   e. I understand how to tune the cost optimizer.

Biography

- Web and Database lab
- Co-inventor of SQL++ query language
- Building a SQL++ database / query processor
- Reads the Postgres manual for fun =)

Outline

- Should I use a SQL database?
- Should I use Postgres?
- How can I find performance bottlenecks?
- How can I improve performance?
- Other questions

Should I use a SQL database?

- Yes, for any of the following:
  - The use case is common (e.g. online commerce)
  - Many users concurrently read + write data
  - Development productivity is a priority (e.g. prototyping, ad-hoc queries)
  - Simple abstractions are desirable
- No, for any of the following:
  - The use case is specialized (e.g. astronomy data)
  - Raw performance is a priority

SQL databases are very good at:

- Basic data analysis
  - Associations (JOIN)
  - Filtering (WHERE)
  - Ranking (ORDER BY)
  - Aggregation (GROUP BY)

- Data on disk, i.e. too big to fit in memory
  - Algorithms are chosen to minimize I/O

- Optimizing queries automatically
  - Queries are aggresively rewritten into faster, equivalent queries

- Using static analysis for optimizations (schemas)
  - Based on histograms of data distribution

SQL databases are not good (yet) at:

- Complex algorithms
  - DB/AI research on integrating machine learning algorithms into SQL (e.g. MADlib)
- Data that can fit entirely in memory
  - Startups for in-memory databases (e.g. MemSQL)
- Allowing fine-grained control of optimizations
  - Most query optimizers are customized by hints
- Allowing semi-structured data (i.e. no schema)
  - Startups for NoSQL databases that read + write JSON data (e.g. MongoDB, Couchbase)

Should I use Postgres?

- Well-designed
  - Supports common SQL features and special extensions (e.g. full text search, geometric types
  - Highly conformant to SQL 2011 standard
  - Academic origins (UC Berkeley, 1986)
  - Forked to build many commercial databases (IBM Netezza, Teradata Aster, Teradata Hadapt, Greenplum etc.)
- Open source
  - First open source release in 1996
  - Active development: new features with yearly release
  - Commercial support by EnterpriseDB

Postgres versus ...

- MySQL
  - MySQL is ranked 2nd, Postgresql is ranked 4th
    http://db-engines.com/en/ranking
  - Developers are polarized between the two
  - Both support similar features
    http://www.wikivs.com/wiki/MySQL_vs_PostgreSQL

- NoSQL databases
  - MongoDB, Cassandra, Couchbase etc.
  - Support JSON data without requiring schemas
  - Favor weaker consistency models for concurrent reads + writes
  - Require writing code instead of queries (but this is changing)
  - Benchmarks are not definitive on whether they are faster than traditional SQL databases

Postgres versus …

- Column-store databases
  - Teradata Aster, HP Vertica, MonetDB, CitusDB etc.
  - Data are clustered column-wise (instead of row-wise in traditional SQL databases)
  - Offers huge speedups (~ 5-10x) for analyzing data warehouses (OLAP)
  - DB research occurred in past 10 years
  - Acquisitions by large DB vendors occured in past 3 years
  - Will become mainstream in the next X years
    - MonetDB is an academic project that is open source
    - CitusDB is a startup with an open source column-store extension to Postgres

Postgres versus ...

- Object-Relational Mappers (ORMs)
  - Hibernate, ActiveRecord, LINQ etc.
  - Automatically translates objects + method calls into SQL queries
  - Elegantly handles simple SQL queries, but requires head scratching for moderate and complex SQL queries
  - Usually not worth the trouble

How can I find performance bottlenecks?

How can I improve performance?

- Understand how a query optimizer works
  - Postgres documentation
    - http://www.postgresql.org/docs/9.3/static/index.html
    - http://www.postgresql.org/docs/9.3/static/using-explain.html
  - Database textbooks and/or classes
    - CSE 232 in Winter
    - CSE 232B in Spring

How can I improve performance?

- Configure Postgres correctly
  - Use Postgres hosted by a vendor (e.g. AWS Relational Database Service)
  - Fix the insanely conservative defaults
    - E.g. memory buffer defaults to 128 MB
    - Use pgtune to set sensible defaults

- Investigate performance bottlenecks
  - Investigate Postgres memory buffer hit/miss ratio
  - Investigate OS disk cache hit/miss ratio

- Improve I/O latency and throughput
  - Use faster disks (e.g. SSDs)
  - Use AWS Provisioned IOPS, which supports predicatable I/O rate

How can I improve performance?

- Blood, sweat and tears
  - EXPLAIN ANALYZE, EXPLAIN BUFFERS etc.
    - http://www.postgresql.org/docs/9.3/static/sql-explain.html
  - PREPARE queries
    - http://www.postgresql.org/docs/9.3/static/sql-prepare.html
  - Use covering indexes (new in Postgres 9.2)
    - https://wiki.postgresql.org/wiki/Index-only_scans
  - Understand an optimizer assumes that two conditions are not correlated when estimating selectivities
  - Pre-compute results and store them in tables (if necessary)
  - Create a tablespace on a ramdisk to keep important tables in memory
  - Each row has a 27 byte overhead, thus Postgres is inefficient when caching rows of small width

Current research: SQL++

- SQL + JSON
  - Familiar SQL syntax
  - Inputs/outputs JSON (and more)
  - Extends SQL for semi-structured data
    - Optional schemas, nesting, ordering, heterogeneity etc.

- Unifying language for both old and new databases
  - SQL
  - SQL-on-Hadoop: Hive, Pig, Jaql
  - NoSQL: MongoDB, Cassandra, Couchbase, JSONiq
  - Others: Google BigQuery, AsterixDB, MongoJDBC

- http://forward.ucsd.edu