



PERCONA
Performance Consulting Experts

Full Text Search with Sphinx

OSCON 2009

Peter Zaitsev, Percona Inc

Andrew Aksyonoff, Sphinx
Technologies Inc.

Sphinx in a nutshell

- Free, open-source full-text search engine
- Fast indexing and searching
- Scales well
- Lots of other (unique) features



<http://sphinxsearch.com>

Some figures

- Biggest Sphinx instance – **boardreader.com**, with **3,000,000,000** documents
 - Yes, that's billions
 - Over 3 TB of source text (documents aren't tweets)
- Busiest Sphinx instance – **craigslist.org**, with **50,000,000 queries/day**
- Sphinx also powers searches on a number of other top-N sites such as Meetup, Wikimapia, DailyMotion, Scribd, etc

Plain old features

- Sphinx does everything you would expect from a search engine...
 - Quick indexing
 - Quick searching
 - Distributed searching
 - Full-text query language (field limits, phrase queries, etc)
 - Per-field weights
 - Non-text search constraints (WHERE authorid=123)
 - Snippets builder
 - Native APIs for a bunch of languages – PHP, Perl, Python, Ruby, Java, etc

Shiny new features

- ...but it also has a number of not-your-typical-run-of-the-mill features
 - Different relevance rankers
 - Multi-queries (with advanced optimizations)
 - Support for non-full-text “full scan” queries
 - Advanced result set processing (WHERE, ORDER BY, GROUP BY, expressions)
 - SELECT syntax support (SQL plus own extensions)
 - MySQL wire protocol support (in searchd server)

Rankers

- Let you choose different relevance ranking functions
 - Slower but with phrase match boost (default)
 - Or faster but w/o phrase match boost (BM25)
 - Or skip ranking at all for boolean searching
- 8 existing rankers at this point
- New rankers will be gradually developed
 - Generic ones to generally improve searching quality
 - Custom ones for specific customer tasks

Multi-queries

- Let you pass a batch of queries to Sphinx
- That lets searchd perform batch optimizations
- Superset of so-called faceted searching
- Common query optimization (added in 0.9.8)
 - When only sorting/grouping modes differ, expensive full text query is only computed once
 - Particularly important for faceted searching
- Subtree caching (added in bleeding edge 0.9.10)
 - When full text queries have common parts, those common parts (subtrees) are only computed once and cached

Scan queries

- Sphinx can store additional attributes attached to every searchable document – as in SQL columns
- Sphinx can do search result set filtering, sorting, grouping – as in SQL WHERE, ORDER, GROUP
- So it was natural to allow doing this w/o text query
- If you keep the full text query empty, Sphinx will enumerate all indexed documents – and apply filtering, sorting, grouping, etc
- Faster than MySQL on some kinds of queries
- And easier to parallelize, too

SELECT support

- Anything that can be done with single table SELECT using an SQL server – can be done to full text search (or scan) result using Sphinx
 - Plus a few special perks (on the next slide)
- It can compute arbitrary arithmetic expressions ($a*b+c*\log(@weight+1.234)-5$)
- It can do WHERE, ORDER BY, GROUP BY, COUNT(*)/COUNT(DISTINCT), SUM()/AVG()...
- There still are certain syntax kinks to iron out – however all the core functionality is there

SELECT support – special perks

- Perk 1 – pretty quick expressions engine
- MySQL
 - Integer: 21.2 M/sec
 - Float: 4.0 M/sec

```
mysql> set @a:=1; set @b:=2; select benchmark(100000000,@a*@b+3);  
1 row in set (4.73 sec)
```

```
mysql> set @a:=1.2; set @b:=3.4; select benchmark(100000000,@a*@b+5.6);  
1 row in set (24.52 sec)
```

- Sphinx
 - Integer: 51.7 M/sec
 - Float: 24.0 M/sec

SELECT support – special perks

- Perk 2 – pretty quick full scan + sort/group
- On an identical 2.5 Mrow test table
- `SELECT id FROM test1 ORDER BY touched DESC`
 - MySQL: 1.87 sec
 - Sphinx: 0.87 sec
- `SELECT flet, COUNT(*) AS q FROM test1 GROUP BY flet ORDER BY q DESC`
 - MySQL: 1.10 sec
 - Sphinx: 0.33 sec
- Details in Piotr Biel's paper from OpenSQLCamp

SELECT support – special perks

- Perk 3 – WITHIN GROUP ORDER BY extension
 - Lets you control specifically which “best” row to choose to represent the entire group when doing GROUP BY
 - For example GROUP BY gid ORDER BY date WITHIN GROUP ORDER BY @weight DESC will order the result set by date, and pick the most relevant rows within groups
- Perk 4 – enforces LIMIT and always optimizes for it
 - Queries that scan 10M rows and return 1K best rows will store at most 1K rows in RAM any given moment
 - Known deficiency in MySQL unless you do an index scan

MySQL wire protocol

- We used SQL in the examples a lot
- And as mentioned Sphinx can run it, literally
- But moreover, starting from 0.9.9 you can use any MySQL client to talk to searchd
- Because we now support MySQL wire protocol in addition to Sphinx protocol

```
searchd { # sphinx.conf section
  listen = localhost:3312:sphinx # native protocol
  listen = localhost:3307:mysql41 # mysql protocol
  ...
```

MySQL wire protocol

```
$ mysql -P 3307
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 0.9.9-dev (r1734)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SELECT * FROM test1 WHERE MATCH('test')
-> ORDER BY group_id ASC OPTION ranker=bm25;
+-----+-----+-----+-----+
| id    | weight | group_id | date_added |
+-----+-----+-----+-----+
| 4    | 1442  | 2       | 1231721236 |
| 2    | 2421  | 123     | 1231721236 |
| 1    | 2421  | 456     | 1231721236 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Roadmap

- Even more cool stuff coming in the future
- Short term
 - Support for string attributes
 - Preforked workers (helps “lots of quick queries” scenario)
 - Some index size/speed optimizations
- Mid term
 - Realtime full-text index updates (real-time attribute updates are already in place)

Support and Consulting

- By Sphinx Technologies
 - Annual Support and Consulting Services
 - Custom feature implementation
 - <http://www.sphinxsearch.com>
- By Percona
 - Support and Consulting
 - <http://www.percona.com>

Questions?

