

Sphinx 2009

Vocals: Peter Zaitsev, Percona
Lyrics: Andrew Aksyonoff, Sphinx



Who are you?

- Sphinx – FOSS full-text search engine



<http://sphinxsearch.com>

Who are you?

- Sphinx – FOSS full-text search engine
- Why Sphinx?
 - Quick indexing and searching
 - Good relevance (we think)
 - Scales well
 - Can improve non-fulltext (!) queries
 - Lots of other features

The talk?

- Assume you know zee basics
 - Or can Google 'em up!
- Bleeding project news
 - Changelog read aloud indeed, kind of
 - Except most stuff so bleeding there's yet no changelog you could actually read

Overall

- Three branches right now
- 0.9.9 – feature frozen long ago
- 0.9.9-release planned in 1-2 weeks, busy fixing and testing now
- 0.9.10 – current trunk, alpha to be frozen after 0.9.9-release
- 1.x – trunk + Real Time Updates, alpha to go public after and/or with 0.9.10-alpha

New stuff in 0.9.9

- Biggest one: SphinxQL and MySQL protocol

```
$ mysql -P 9306
```

```
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)
```

SphinxQL via MySQL protocol

- You can talk SQL to Sphinx now!
 - Using regular MySQL client libs or CLI client
 - Yeah, `mysql_connect()` from PHP **will** work
- No dependencies on MySQL for that
 - No need to use SphinxSE, it's plain old **searchd**
 - No need to link `searchd` against `mysqld` (awww)
 - No need for client libs on Sphinx box, even

SphinxQL features

- 0.9.9 allows half-blown SELECT syntax
 - Without JOINS etc, but
 - With arbitrary arithmetic expressions
 - With WHERE (limited)
 - With ORDER BY
 - With GROUP BY
- Trick to work around limited WHERE:
 - SELECT *, a OR b AS q WHERE q=1

Other cool 0.9.9 features

- Aggregate functions (AVG, MIN, MAX, SUM)
- ODBC and unixODBC support
 - pull data from MS SQL, Oracle, Firebird, ... easily
- Internal counters
 - New searchd `-cpustats` switch
 - New `SHOW STATUS` query and `Status()` API call
- Merge runs in fixed RAM now (finally)
- 20-30 more features (see changelog)

Forthcoming in 0.9.10

- New MPMs (multi-processing models)
 - We had none (`--console`) and `fork` (implicit)
 - We added **prefork** and **threads**
- Prefork already used in prod
- Nice savings on system time under lots of tiny super-quick queries

Forthcoming in 0.9.10

- Added **string attributes**
 - sql_attr_string: just store as attr
 - sql_field_string: store but also FT index
 - Stored in RAM, just as MVA
 - Storage/retrieval only for now
 - No ORDER BY, GROUP BY, WHERE just yet (sponsors welcome, email us for a quote)

Forthcoming in 0.9.10

- Added `indextool --check`
 - Index integrity checking tool
 - To help check for HW failures, indexer bugs, maybe add a level of protection for prod, etc
 - Fun fact: per-keyword occurrence count can be computed using 3 (three) different routes
 - We do all 3 independently and verify them

Forthcoming in 0.9.10

- Optimized 1-keyword queries again (got lost in 0.9.9)
- Optimized attr handling (less internal copying)
- Optimized MySQL protocol vs 0.9.9
- Optimized common subqueries within a batch (added a cache)
- Optimized index loading on startup (moved block-level attr subindex to indexer)

Forthcoming in 0.9.10

- Added `blend_chars` (handle AT&T and C++ without specifying prerecorded exceptions)
- Added hitless indexes and partially hitless indexes
- Added `sql_joined_field` (when you can't `GROUP_CONCAT`)
- Added query-based snippet highlighting
- ...and about 10 other features

Last but not least, 1.11-alpha

- With **online full-text index updates**
 - Attribute updates are already online for a while!
- Hybrid approach
 - RT chunk that lives in RAM
 - Online updates/inserts touch RT chunk
 - Once RAM is exhausted, it gets dumped to disk
 - Strictly limited RAM use, as usual

1.x by the numbers

- INSERT single ~ 1 KB row and COMMIT = $\sim 1-2$ ms/avg
- Yes, that means 1 ms and it's searchable
- Faster if you COMMIT entire row blocks
- Indexing 500K rows one-by-one about 2x slower than regular indexing
- COMMIT not durable yet (no crash recovery; cf alpha)

1.x by the numbers

- Not quite benchmarked yet, but
- In-RAM chunk searching: close to regular (cached regular index is pretty fast)
- On-disk chunk searching: identical, because regular index format is reused for those
- In theory, should be slower on big indexes because of fragmentation. Never tried yet

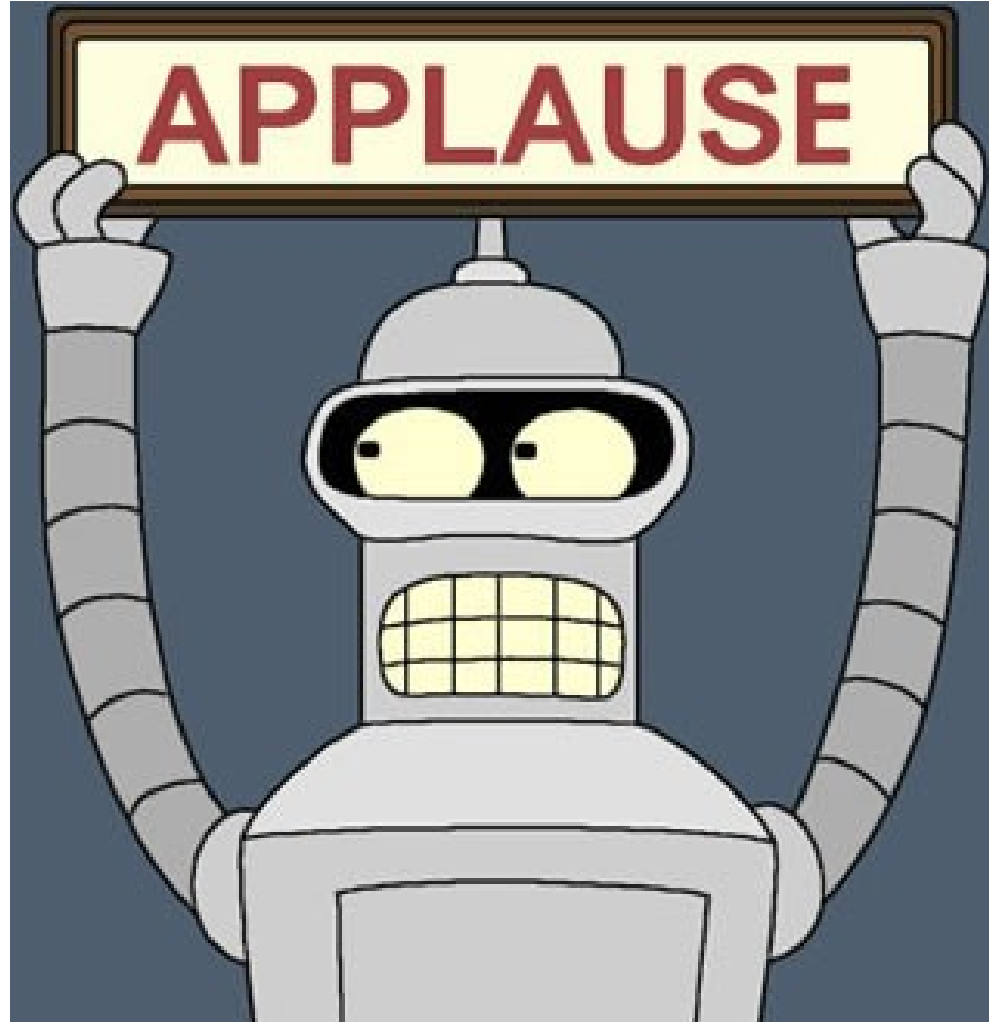
1.x interface

- Only via MySQL protocol for now
 - Already got INSERT, REPLACE, SELECT, and DELETE ... WHERE id=X
 - Trivial to add UPDATE attr WHERE id=X
 - Will **not** add UPDATE single_text_field
 - would need to store all source text data
 - can't enforce that much of a constraint
 - Will add UPDATE all_text_fields though
 - Will add WHERE cond in DELETE and UPDATE



Ah, almost forgot!

- Got official port assignments from IANA
- API on port 9312, SphinxQL on 9306 now





Questions?



<http://sphinxsearch.com>