



Meet the Sphinx

Andrew Aksyonoff // Sphinx Technologies Inc.

DORS/CLUC 2010



Who are you?

- Sphinx – FOSS full-text search engine



<http://sphinxsearch.com>



Why Sphinx?

- Easy
- Fast
- Relevant
- Scales
- Feature rich
- Not just full-text

Why not MySQL/Lucene/Solr/...?

- Use them – if they **do** work for you
- MySQL starts to crawl at 100K+ rows
- Lucene speed/functionality are better, but
 - We've got faster indexing
 - We've got faster, more relevant searching (*)
 - We're better at “SQL style” queries
 - We can do Java but don't require a Java stack ;)

Gimme numbers!

- 10-1000x (!!!) vs MySQL on full-text searches
 - MySQL only behaves when indexes are in RAM
- 2-3x vs MySQL on non-full-text (!) scans
 - On a single core
 - Because of less overheads (but less dynamic)
- 2-4x faster than Lucene on full-text searches
 - Our (aged) internal benchmarks, YMMV

That thing about relevance

- Once of the reasons behind Sphinx actually...
- The usual way, so-called BM25 function
 - Accounts keyword frequencies, good!
 - Ignores keyword positions, bad :(
 - "to be or not to be", "i feel you" etc queries
- Sphinx adds phrase based ranking
 - Boosts (sub) phrase matches
 - Perfect match is guaranteed to be ranked #1

Scales as in?

- Over 3,000,000,000 (yes billion) documents
 - boardreader.com, over 1M q/day too
- Over 50,000,000 queries/day
 - craigslist.org, 20-30 GB docs
- Powers Craigslist, Meetup, Slashdot, WikiMapia, and a few thousand other sites
 - We might power something big here in Croatia, with open source you never know :)

Coming up next in this talk...

- Sphinx architecture/workflow introduction
 - For web developers (aka Sphinx users)
 - Visit the workshop tomorrow for deeper details
- Project news 2010
 - For developer bosses 8-)
 - What are we up to lately

Architecture (aka engine insides)



Sphinx workflow

- You index – using indexer program
- You search – using searchd program
- There are data sources (what and where?)
- There are indexes
 - What data sources to index
 - How to process the incoming text
 - Where in the FS to put the index files

So how do I index?

- Define a source – basically an SQL query
- Define an index – basically FS path + misc text processing settings
- Run indexer
- ...
- **PROFIT!!!**

Show me yours, I'll show you mine (config file, that is)

```
source test1
{
    sql_query = SELECT id, title, descr, added, price \
        FROM products
    sql_attr_timestamp = added
    sql_attr_float = price
}

index test1
{
    source = test1
    path = /my/index/store/test1
}
```



indexer in action

(patent-pending 80x25 screenshot)

```
$ ./indexer lj
Sphinx 1.10.1-dev (4c7aaa426b6a)
Copyright (c) 2001-2010, Andrew Aksyonoff
Copyright (c) 2008-2010, Sphinx Technologies Inc (http://sph...

using config file './sphinx.conf'...
indexing index 'lj'...
collected 999944 docs, 1318.1 MB
sorted 224.2 Mhits, 100.0% done
total 999944 docs, 1318101119 bytes
total 158.080 sec, 8338160 bytes/sec, 6325.53 docs/sec
total 33 reads, 4.671 sec, 17032.9 kb/call avg, 141.5 msec/call
total 361 writes, 20.889 sec, 3566.1 kb/call avg, 57.8 msec/call
```

And then how do I search?



And then how do I search?

- Run searchd
- Connect to searchd and query it
 - SphinxAPI (native ports for PHP, Python, Perl, Ruby, Java, C#, Haskell...)
 - SphinxSE
 - SphinxQL

And then how do I search?

- SphinxAPI?

```
<?php

require ( "sphinxapi.php" );
$client = new SphinxClient ();
$res = $client->Query ( "my first query", "test1" );
var_dump ( $res );

// wham, bam, searching kinda done
```

And then how do I search?

- SphinxSE?
- Plug-in it into your MySQL instance, and then

```
SELECT *
FROM sphinxsetable s
JOIN products p ON p.id=s.id
WHERE s.query='@title ipod'
ORDER BY p.price ASC

// or better!
... WHERE s.query='@title ipod;sort=attr_asc:price';
```

And then how do I search?

- SphinxQL?
 - Our own implementation of MySQL protocol
 - Our own SQL parser
 - **MySQL not required!!!**
 - Any **client** library (eg. PHP's) should suffice



SphinxQL FTW

```
$ 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)
```

What the searcher can do

- SphinxQL explains it pretty good now
- It supports arbitrary expressions in SELECT
- It supports WHERE, ORDER BY, GROUP BY, COUNT, AVG/MIN/MAX/SUM
- It supports our own extensions
 - Such as OPTION ranker=bm25
 - Such as WITHIN GROUP ORDER BY ...

What's beyond basic searching?

- Many features of different caliber, actually
 - MVAs, tokenizing settings, wordforms, 1-grams, HTML processing, throttling, geosearching, arbitrary expressions, prefix/infix indexing, tuning the ranking...
- Let's quickly cover two frequently used ones
- Both optimization related

Chapter 2. Project news 2010



Non-tech stuff

- Sales/support phone
 - SHAMELESS AD, call us at +1 888 333 1345 and hire us to implement your search
- Launched public SVN
- Launched 0.9.9 release
 - Brings SphinxQL, aggregate functions, ODBC, persistent connections, official IANA ports ...
- Launched a blog recently

Tech stuff

- A bunch of tricky, but rather minor stuff
 - Better stats // better constant optimizer // new FlushAttributes() call // precaching progress bar // per-hit payload indexing // searchd --logdebug // --htmlstrip tool // force_all_words feature // ...
 - sql_joined_field // query-based snippets // blended characters // keyword expansion // SPH04 ranker // hitless indexing // common subquery cache // ...

Anything for the REST of us?!

- Major features that we've added last year (and that are possible to comprehend!!!):
 - Index checking tool
 - Optimized index format (1.5x less size)
 - Prefork and Threads support (Fork-only before)
 - Improved parallel searching (dist_threads)
 - String attributes support
 - **Real Time indexes** (alpha!)

RT indexes

- SVN-only yet, but production-tested already
- Lets you update data on the fly
- Formally, they are “soft-realtime”
 - As in, most of the writes are very quick
 - But, not guaranteed to complete in fixed time
- Indexing-wise, just INSERT rows (literally)
- Searching-wise, transparent to the app

RT indexing/searching

- Indexing is SphinxQL only
 - `mysql_connect()` to Sphinx instead of MySQL
 - `mysql_query()` and do INSERT/REPLACE as usual
- Searching is transparent
 - SphinxAPI / SphinxSE / SphinxQL all work
 - We now prefer SELECT now that we have SphinxQL :)

RT performance

- Depends **a lot** on data, settings, patterns...
- 1000+ INSERT/sec with single (!) rows
- 5000+ INSERT/sec with batches
- 200+ SELECT/sec
- 500 MB test collection, ~1KB/row, no binlog
- Optimizations TBD, performance to improve :)
- YMMV

Summary (what is all this about?!)



Summary

- Sphinx is a free, open, easy, fast, relevant, scalable ... [3 more pages of self-promotion censored] ... full-text engine
- It works, it scales, it evolves
- Interested in search? Give it a try
- Looking for search-related help? Give us a call
- More nitty gritty tech details at a workshop!



Questions?



<http://sphinxsearch.com>