



PERCONA  
Performance Consulting Experts

---

# Sphinx full-text search engine

November 15, 2008

OpenSQL Camp

Piotr Biel, Percona Inc

Andrew Aksyonoff, Sphinx Technologies

Peter Zaitsev, Percona Inc

# Full Text Search

---

*Full Text Search* – technique for searching words in indexed documents by examination of all keywords in stored document and matching it against keywords supplied in the query.

# External or local?

---

## Local:

- Native, easy in implementation
- No need to change environment which is pretty often problematic in hosted services

## External:

- Independent of storage system (MySQL, PostgreSQL, Oracle, XML files...)
- Works with all storage engines (MyISAM, InnoDB, Falcon)
- Ideal for minimising load on databases

# Why Sphinx?

- Great indexing / searching speed
- Scalability
- Better resources utilization with a lot of concurrent searches than traditional, native FTS
- Indexes not only text
  - Numerical attributes can be used to filter results
- Can work as SQL replacement
  - sorting, grouping
  - can be more efficient than MySQL by order of magnitude on large sets
- Online index rebuilds
- Good choice of matching modes and operators

# MySQL FTS vs Sphinx

---

## MySQL

- Available only for MyISAM tables
- Slow on moderate sized (1 GB+) collections
- Limited querying capabilities
- Can't index BLOBs

## Sphinx

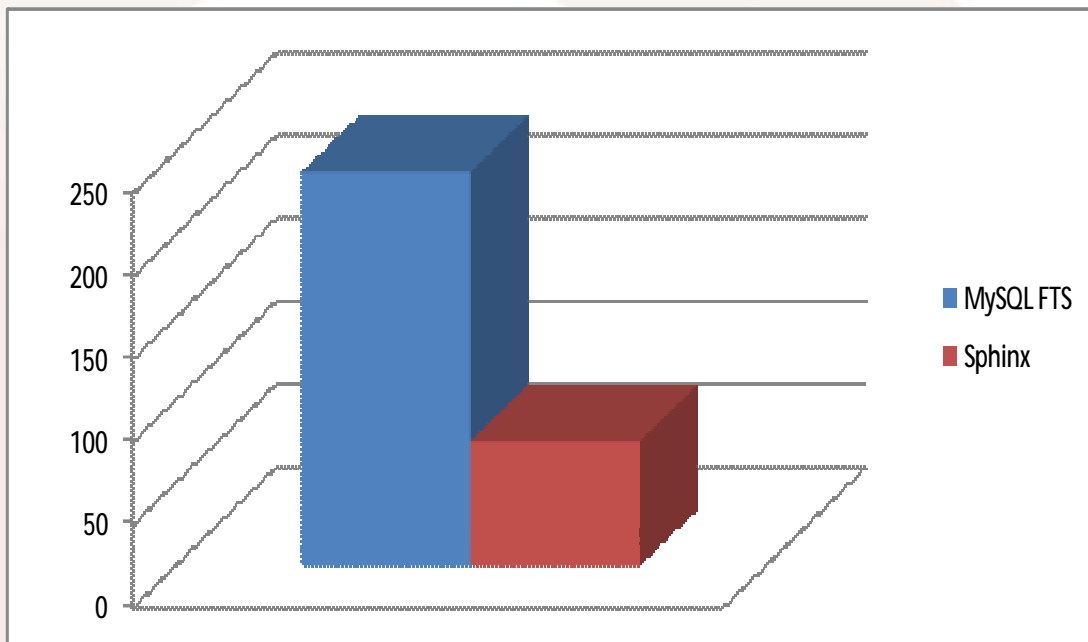
- Available for all type of tables
- Scales well
- Advanced querying capabilities
- Ability to index all data types

# Speed

Wikipedia database used for tests, 2.5M rows, 15G text.

Times are in seconds to complete.

Creating index (just on title):



MySQL: 273,7s

Sphinx: 75,1s

# Speed

Wikipedia database used for tests, 2.5M rows, 15G text.

Times are in seconds to complete.

Creating index (on title and content):

MySQL: +inf

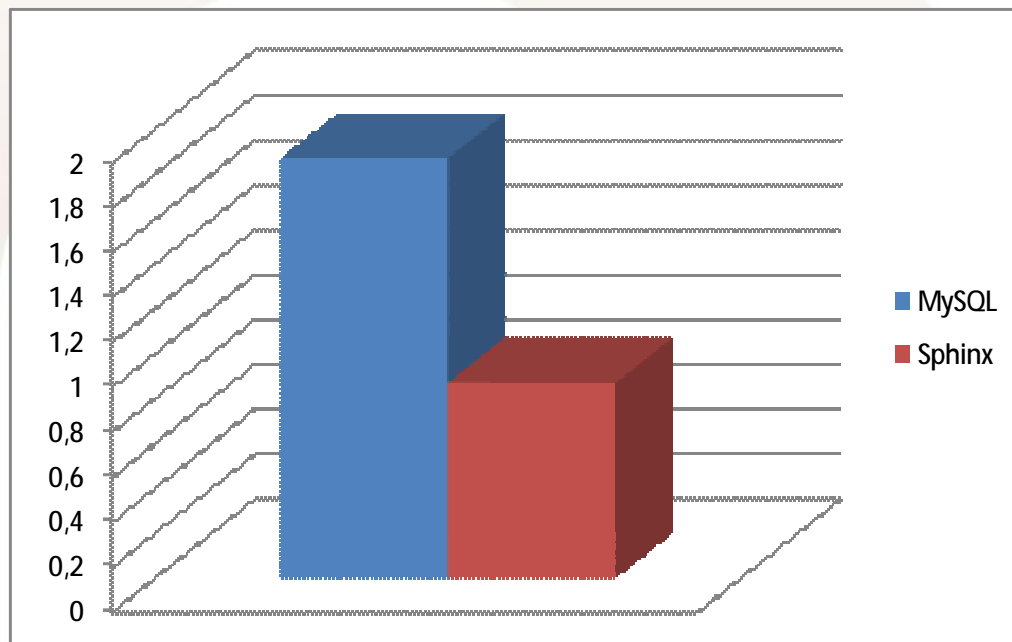
Sphinx: 1159,3s

Unfortunately, we had to kill MySQL after 6+ hours (21600s) of indexing.

# Speed

## Sorting speed

- `SELECT id FROM test1 ORDER BY touched DESC LIMIT 1000`
- `php test.php -i test1 -s touched,desc`



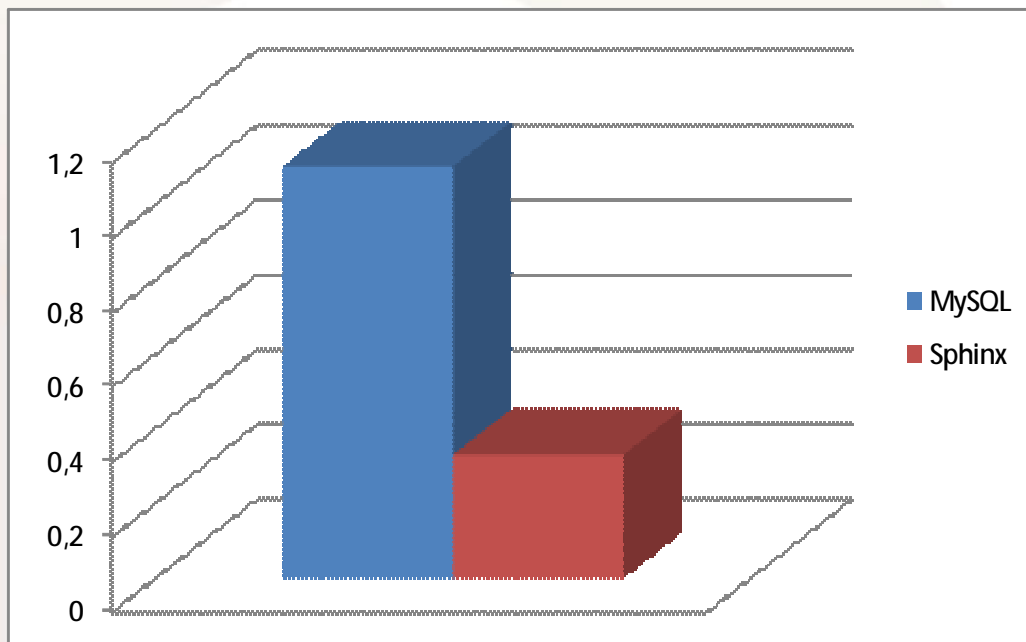
MySQL: 1,87s

Sphinx: 0,87s

# Speed

## Grouping speed

- `SELECT flet, COUNT(*) AS q FROM test1 GROUP BY flet ORDER BY q DESC`
- `php test.php -i test1 -g flet -gs @count,desc`



MySQL: 1,10s

Sphinx: 0,33s

# Speed (methodology)

Sorting and grouping speed — what do we measure?

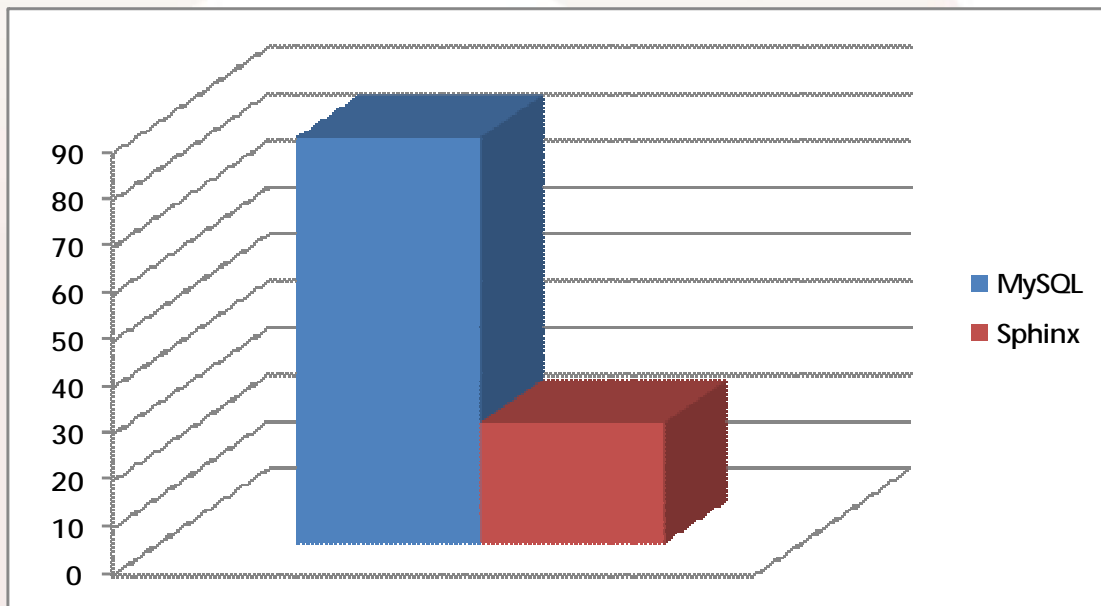
- We create trimmed down MySQL table
- We benchmark it against similar Sphinx index
- We benchmark full-scan + ORDER BY + GROUP BY
- Covering index, of course, makes MySQL lot faster?
  - WRONG, it does not. Still got 1.87 sec on sorting
  - WRONG, it hurts!!! Got 1.74 sec on grouping
- Original data, of course, makes MySQL lot slower
  - 66.0+ sec for both sorting and grouping, IO bound

# Speed

FTS speed - 2000 queries running in 8 threads

Against just 100K rows of Wiki for MySQL (time)

Against complete 2500K rows for Sphinx



MySQL: 86,56s x25!

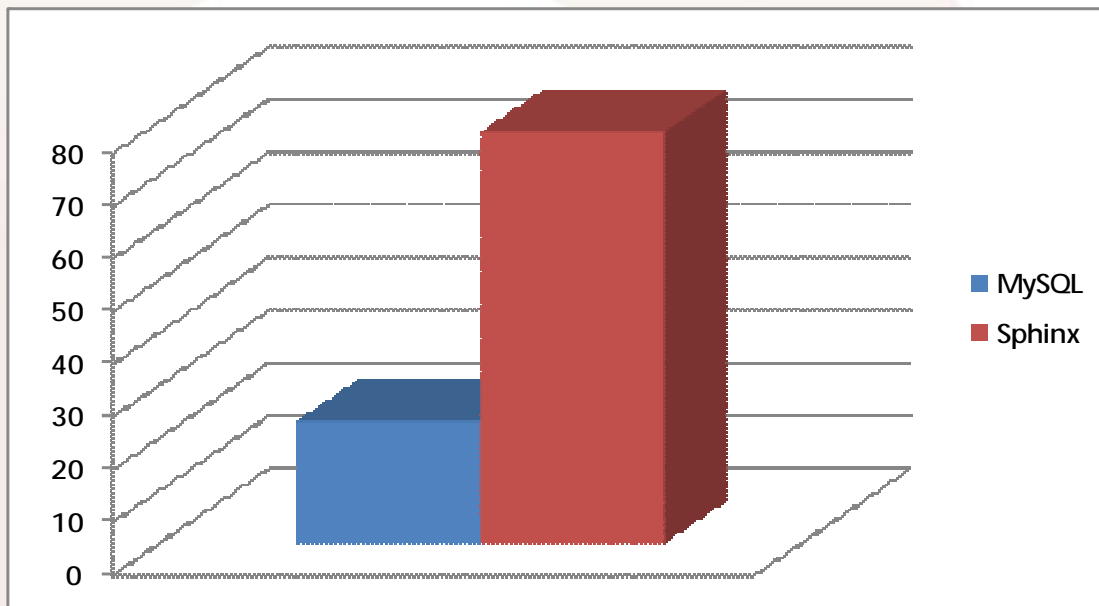
Sphinx: 25,66s

# Speed

FTS speed - 2000 queries running in 8 threads

Against just 100K rows of Wiki for MySQL (time)

Against complete 2500K rows for Sphinx



MySQL: 23,1 qps /25

Sphinx: 77,95 qps

# Scalability

---

- **Distributed searches**
  - Many servers can be used
  - Many CPUs/cores within single server can be used
- **Distributed indexes are fully transparent for end-users**
  - Virtual distributed index overlays many physical indexes, either local or remote
- **Great local resources utilization (CPUs)**

# How is it organized?

---

***Searchd*** – standalone daemon running in system, responsible for answering client queries

- Filtering – WHERE analogue
- Sorting – ORDER BY analogue
- Grouping – GROUP BY analogue

***Indexer*** – tool to build indexes

- Fetching documents and splitting into separate words
- Processing fetched results

# Talking to Sphinx

---

- Using APIs
- PHP, Perl, Python, Ruby, pure-C, C++, C#, Haskell...
- SphinxSE – MySQL storage engine dedicated for communication with Sphinx
  - Useful when there's no native API port
  - Also lets you juggle huge datasets directly on MySQL side
  - Without fetching them to client then sending to MySQL

# New features in Sphinx 0.9.9

---

## Why?

- 0.9.9 is the current development version
- Beta will be published shortly (this weekend?)
- Let's see what's bleeding at the edge!

## Overall

- 34 new features of different caliber
- 10 are major changes
- had a hard time selecting best N...

# Top-10 new features

## (1) added select-list feature w/full expressions support

- lets you specify specific columns and expressions to fetch
- compute and fetch arbitrary number of arbitrary expressions
- computed columns can be used for filtering, sorting, grouping
- expressions are currently 2-4x slower than native (!) code:

benchmarking expressions

run 1: int-eval 49.7M/sec, flt-eval 46.3M/sec, native 129.7M/sec

run 2: flt-eval 28.2M/sec, native 108.6M/sec

run 3: flt-eval 269.2M/sec, native 309.9M/sec

- further optimizations planned (JIT native code for expressions)

# Top-10 new features

---

## (2) added arbitrary brackets/negations nesting support to query language

- query parser was rewritten from scratch
- query still must be "computable"
- implicit lists of documents such as "foo|-bar" are not allowed
- they usually indicate a programming or querying mistake anyway...

## (3) added config reload on SIGHUP

- lets you add and remove new indexes on the fly
- also, all index settings are now stored within index
- (much) less index/config inconsistency issues

# Top-10 new features

---

(4) added signed 64bit attrs support (sql\_attr\_bigint directive)

- support means support
- filtering, sorting, groupby, expressions, everything should work

(5) added persistent connections, UNIX-socket, and multi-interface support (Open(), Close(), listen)

- self-explanatory (less TCP pressure, more security, etc)
- pconns are not (yet?) used by master searchd instances when talking to remote agents, though
- something to add for HP/HA... sponsors are welcome :)

# Top-10 new features

## (6) added kill-list support

- new `sql_query_killlist` directive
- lets you eliminate "phantom results" from older indexes
- fetches a list of documents to remove from previous results
- example:

main index

doc 1 title is "hello world"

doc 2 title is "hello world reloaded"

doc 3 title is "hello world revolutions"

delta index:

doc 2 gets deleted

doc 3 title becomes "sample program"

# Top-10 new features

## (kill-lists, continued)

- querying for "hello" will return documents 2 and 3
  - doc 2 could be suppressed by runtime "deleted" flag update... ugly
  - doc 3 from "main" (!) could not be suppressed at all (phantom result)
- solution? kill-lists attached to "delta" index
- delta (!) kill-list contains ids 2 and 3
- how that works?
- result set after searching "main" will be 1,2,3
- then delta kill-list only keeps result 1 and removes 2,3
- then delta search would add new matches to result set (if any)

# Top-10 new features

---

- (7) added MS SQL (aka SQL Server) source type support
- (8) added `index_exact_words` feature, and exact form operator
  - because sometimes you want to partially suppress stemming
  - query “your =business” will match “business of yours”
  - but not “are you busy” any more
- (9) added inplace inversion of `.spa` and `.spp`  
(`inplace_enable`, 1.5-2x less disk space for indexing)
- (10) improved excerpts speed (upto 50x faster!)
  - not exactly totally new, but major improvements...

# Other 24 features at a glance

---

- added `min_stemming_len`
- indexer-side column unpacker (`unpack_mysqlcompress`)
- builtin Czech stemmer (`morphology=stem_cz`)
- on-disk SPI support, trade RAM for IO (`ondisk_dict`)
- indexer now prints out IO stats
- HTML stripper now skips PIs (such as `<?php ... ?>`)
- `IsConnectError()` API call (API errors vs remote errors)
- `int64` expressions and `BIGINT()` cast (lets avoid 32bit wraparounds when computing  $A*B$ )

# Other 24 features at a glance

---

- star-syntax support in BuildExcerpts()
- IDIV(), NOW(), INTERVAL(), IN() functions
- index-level early-reject based on filters
- MVA updates feature (mva\_updates\_pool directive)
- multiforms support (multiple source words can be mapped to single destination word)
- removed legacy matching code, everything runs using new V2 engine now
- field position limit (syntax: @title[50] hello world)
- periodic .spa flush (attr\_flush\_period directive)

# Other 24 features at a glance

---

- periodic .spa flush (attr\_flush\_period directive)
- per-query attribute overrides (see SetOverride() call)
- duplicate log messages filter in searchd
- --nodetach debugging switch in searchd
- blackhole agents support in searchd (agent\_blackhole)
- max\_filters, max\_filter\_values (were hardcoded)
- crash handler for debugging (crash\_log\_path)
- status variables support in SphinxSE
- max\_packet\_size (was hardcoded)

# Thank you

---

Sphinx website

<http://sphinxsearch.com>

Percona website

<http://percona.com>