How to offload MySQL server with Sphinx

Vladimir Fedorkov, Sphinx Technologies

Percona Live, MySQL UC, Santa Clara 2012

About me

- Design and tune high-loaded apps since 2006
- Performance geek
- Blog posts writer
  - http://sphinxsearch.com/blog
  - http://astellar.com

Agenda

- What is application performance
- How to make **your** aplication faster
  - Technical cookbook
- When and where Sphinx could help
  - Query fine tuning
  - Sphinx day-to-day operations
- Interesting features you want to hire

Why are we here?

- Keep visitors satisfied
  - Performance
  - Relevance
  - Reliability

Meet the expectation

- Allow users find what they **looking for**
- … even when they don't really know
- … to keep them satisfied and return back to you

0.1 — 1 — 10

- For users
- For search engines

Application is not a solid rock

- Lots of layers
  - Apache/Nginx/Lighttpd/Tomcat/You name it
  - Perl/PHP/Python/Ruby/Java/.NET/C++/Haskel/…
  - Percona Server/MariaDB/Drizzle/MySQL/
    - PostgreSQL/MSSQL/Oracle/DB2/Firebird...
  - Memcache/MongoDB/CouchDB…
  - Sphinx/Lucene/SOLR/Elastic/IndexDen/…
  - Third party libraries and frameworks
  - Your own code

Which one to use?

What do we need

Data layer. The basement.

- MySQL to store data
  - +few replicas for failover
  - +in-memory storage for dynamic data
    - memcache, tarantool, etc
  - +application level caching
  - +sphinx for specific queries

What is Sphinx

- Age: 10+ years old open source search server
  - Separate daemon, just like MySQL
- Easy application integration via number of APIs
- You can query Sphinx via MySQL client
  - MySQL is not required!
- Highly scalable
  - local and disributed search supported
  - Scales out of the box

What is Sphinx

- Two types of engines are available
  - On-disk indexes (pull model)
  - Real-Time engine (Soft-realtime backend)
- Available for Linux, Windows x86/64, Mac OS
  - Can be built on AIX, iPhone and some DSL routers
- Open source and free!
  - GPL v2
  - Support (and consulting) is available

Isn't there any better?

- 10-1000x **faster** than MySQL on full-text searches
  - MySQL only behaves when indexes are in RAM
- 2-3x **faster** than MySQL on non-full-text scans
  - Grouping and sorting in fixed memory
  - Attribute search block skipping
- Up to 10Mb/s indexing on a single core.
- Have benchmarks saying we're slow? Let me know!

Known installations

- Over 30,000,000,000+ (yes **Billions**) documents
  - Infegy
  - 26B+ boardreader.com, over 8.6Tb indexed data across 40+ boxes
- Over 200,000,000 queries per day
  - craigslist.org 2,000 QPS against 15 Sphinx boxes
- We're open source
  - Go ahead and let us know about you!
  - http://sphinxsearch.com/info/powered/

Agenda

1. Installation and setup
2. Integration and basic search
3. Faceted search
4. Real-time search
5. Advanced search and performance tricks
6. Distributed search
7. Backup and restore

1. Installation

Sphinx 2.0.4-release downloads

Sphinx 2.0.4-release (r3135; Mar 02, 2012)

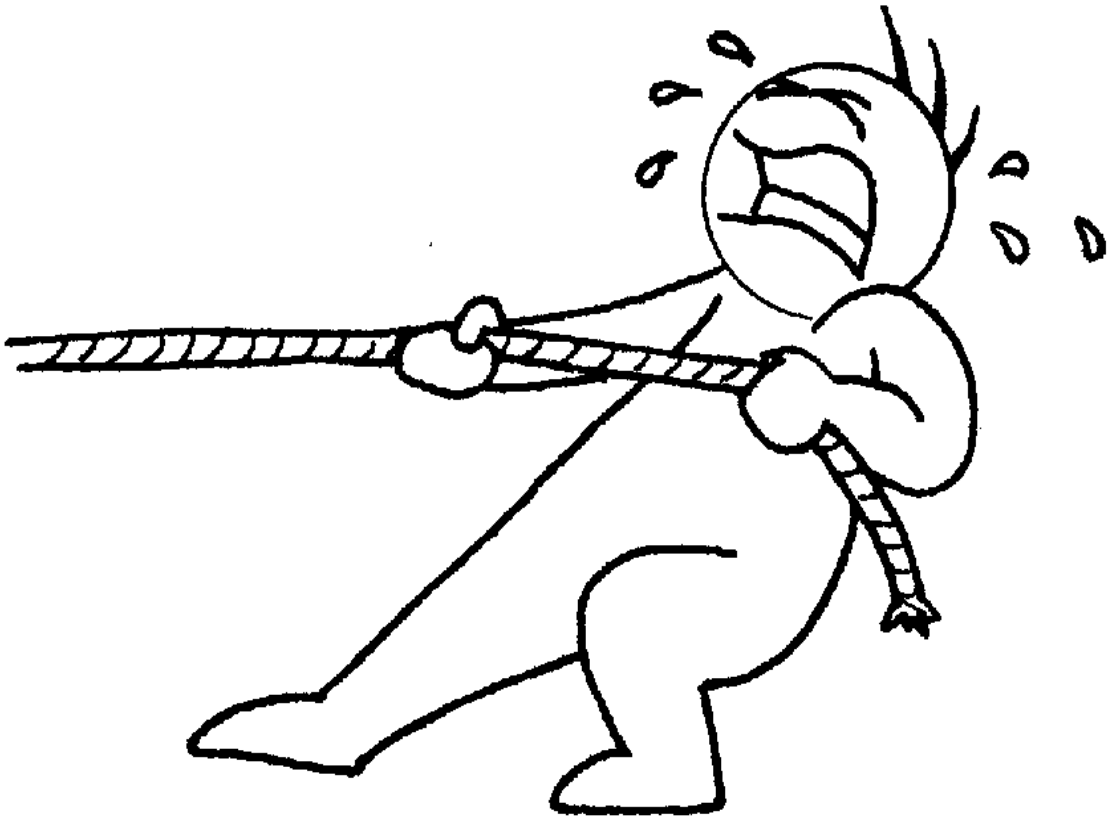
Source tarball (tar.gz)	2.0.4-release	1.8M	Download
Win32 binaries w/MySQL support	2.0.4-release	4.0M	Download
Win32 binaries w/MySQL+PostgreSQL support	2.0.4-release	5.7M	Download
Win32 binaries w/MySQL+PgSQL+libstemmer+id64 support	2.0.4-release	6.1M	Download
Ubuntu 10.04 LTS x86_64 DEB	2.0.4-release	4.5M	Download
Ubuntu 10.04 LTS i386 DEB	2.0.4-release	4.5M	Download
Ubuntu 11.10 x86_64 DEB	2.0.4-release	4.3M	Download
Ubuntu 11.10 i386 DEB	2.0.4-release	4.4M	Download
RHEL/CentOS 5.x x86_64 RPM	2.0.4-release	4.8M	Download
RHEL/CentOS 5.x i386 RPM	2.0.4-release	5.5M	Download
RHEL/CentOS 6.x x86_64 RPM	2.0.4-release	5.3M	Download

Installation: How?

- http://sphinxsearch.com/downloads/
- http://sphinxsearch.googlecode.com/svn/
  - configure && make && make install

Where to look for the data?

- MySQL
- PostgreSQL
- MSSQL
- ODBC source
- XML pipe



MySQL source

```
source data_source
{
    ...
    sql_query = \
        SELECT id, channel_id, ts, title, content \
        FROM mytable

    sql_attr_uint               = channel_id
    sql_attr_timestamp          = ts
    ...
}
```

A complete version

```
source data_source
{

    type   = mysql
    sql_host   = localhost
    sql_user   = my_user
    sql_pass   = my******
    sql_db     = test

    sql_query_pre = SET NAMES utf8
    sql_query     = SELECT id, channel_id, ts, title, content \
                          FROM mytable  \
                          WHERE id>=$start and id<=$end

    sql_attr_uint      = channel_id
    sql_attr_timestamp = ts

    sql_query_range  = SELECT MIN(id),  MAX(id) FROM mytable
    sql_range_step   = 1000
}
```

How to process. Index config.

```
index my_sphinx_index
{
    source              = data_source
    path                = /my/index/path/my_index

    html_strip          = 1

    morphology          = stem_en
    stopwords           = stopwords.txt
    charset_type        = utf-8
}
```

Indexer configuration

```
indexer
{
        mem_limit   = 512M
        max_iops    = 40
        max_iosize  = 1048576
}
```

Configuring searchd

```
searchd
{
        listen = localhost:9312
        listen = localhost:9306:mysql4

        query_log               = query.log
        query_log_format = sphinxql

        pid_file                = searchd.pid
}
```

Integration

Just like MySQL

```
$ mysql -h 0 -P 9306
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 2.1.0-id64-dev (r3028)

Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.

mysql>
```

But not quite!

```
mysql> SELECT *
    -> FROM lj1m
    -> WHERE MATCH('I love Sphinx')
    -> LIMIT 5
    -> OPTION field_weights=(title=100, content=1);
+---------+--------+------------+------------+
| id      | weight | channel_id | ts         |
+---------+--------+------------+------------+
| 7637682 | 101652 |     358842 | 1112905663 |
| 6598265 | 101612 |     454928 | 1102858275 |
| 6941386 | 101612 |     424983 | 1076253605 |
| 6913297 | 101584 |     419235 | 1087685912 |
| 7139957 |   1667 |     403287 | 1078242789 |
+---------+--------+------------+------------+
5 rows in set (0.00 sec)
```

What's different?

- Meta fields @weight, @group, @count
- No full-text fields in output
  - So far
  - Requires additional lookup to fetch data
- MySQL query become primary key lookup
  - WHERE id IN (33, 9, 12, …, 17, 5)
    - Good for caching
- Adding nodes is transparent for the application
  - zero downtime or less ;-)

SQL & SphinxQL

- WITHIN GROUP ORDER BY
- OPTION support for fine tuning
  - weights, matches and query time control
- SHOW META query information
- CALL SNIPPETS let you create snippets
- CALL KEYWORDS for statistics

Looking at the Manual

- Integers
  - Int 32bit unsigned (only)
  - Int 64bit signed (only)
  - Set of integers (Multi-Value-Attribute, MVA)
  - Limited ints using bitcount
- Floats
- Strings
- Timestamps
- MVA (Multi Value Attributes)

Query speed against 8m rows

```
mysql> SELECT id, ...
    -> FROM myisam_table
    -> WHERE MATCH(title, content_ft)
    -> AGAINST ('I love sphinx') LIMIT 10;
...
10 rows in set (1.18 sec)

mysql> SELECT * FROM sphinx_index
    -> WHERE MATCH('I love Sphinx') LIMIT 10;
...
10 rows in set (0.05 sec)
```

Only one way?

There's more

- API
  - PHP, Python, Java, Ruby, C is included in distro
  - .NET, Rails (via Thinking Sphinx) is available
- SphinxSE
  - Prebuilt into MariaDB

Sphinx API

```
<?php
require ( "sphinxapi.php" ); //from sphinx distro
$cl->SetServer ( $host, $port );
$cl = new SphinxClient();
$res = $cl->Query ( "my first query", “my_sphinx_index" );
var_dump ( $res );

?>
```

- More in api/test.php

3. Facets? You bet!

```
mysql> SELECT *, YEAR(ts) as yr
    -> FROM lj1m
    -> WHERE MATCH('I love Sphinx')
    -> GROUP BY yr
    -> ORDER BY yr DESC
    -> LIMIT 5
    -> OPTION field_weights=(title=100, content=1);
+---------+--------+------------+------------+------+----------+--------+
| id      | weight | channel_id | ts         | yr   | @groupby | @count |
+---------+--------+------------+------------+------+----------+--------+
| 7637682 | 101652 |     358842 | 1112905663 | 2005 |     2005 |     14 |
| 6598265 | 101612 |     454928 | 1102858275 | 2004 |     2004 |     27 |
| 7139960 |   1642 |     403287 | 1070220903 | 2003 |     2003 |      8 |
| 5340114 |   1612 |     537694 | 1020213442 | 2002 |     2002 |      1 |
| 5744405 |   1588 |     507895 |  995415111 | 2001 |     2001 |      1 |
+---------+--------+------------+------------+------+----------+--------+
5 rows in set (0.00 sec)
```

4. Real Time engine

- Same SphinxQL and API
  - But you need to **insert** data from the application
- Keeps all the data (chunks) in memory
  - Saves disk chunks
- Uses binary logs for crash safety
- Different index type in sphinx.conf

RT config

```
index rt
{
        type                    = rt
        rt_mem_limit            = 512M
        rt_field                = title
        rt_field                = content
        rt_attr_uint            = channel_id
        rt_attr_timestamp = ts
}
```

RT — Memory Utilization

- Memory allocation by RT engine
  - rt_mem_limit
    - Default is 32Mb
- Disk chunks
  - Static
  - Places on disk

RT — Disk chunks sample

sphinx_rt.kill
sphinx_rt.lock
sphinx_rt.meta
sphinx_rt.ram
sphinx_rt.0.spa
sphinx_rt.0.spd
sphinx_rt.0.sph
sphinx_rt.0.spi
sphinx_rt.0.spk
sphinx_rt.0.spm
sphinx_rt.0.spp
sphinx_rt.0.sps

What to do?

- Keep RT_Mem_Limit big enough
  - Use 64bit Sphinx version
- Reload data when needed
- Keep all static data away from RealTime
  - Use ATTACH INDEX if needed

5. I want it faster!

- Profile
- Scale
- Optimize
- Compact

Remove high-frequency words

- «i», «a», «the», «of», etc
  - Sometime is waste of memory and space
- Create stopwords file
  - *Indexer <index> —buildstops <output.txt> <N>*
    - *You'll need to build an ondisk index for that*
  - http://astellar.com/downloads/stopwords.txt
- Could be used to eliminate «adult words»

Decrease max_matches

- All docuemnts will still be searched
  - Only best results will be returned
- Even Google does 1000!

Use «lightweight» rankers

- SPH_RANK_NONE
  - Fastest, implements boolean search
- SPH_RANK_WORDCOUNT
- SPH_RANK_PROXIMITY

Use custom ranking

- SPH_RANK_SPH04
  - Actially slower but more relevent in some cases
- SPH_RANK_EXPR
  - Allow you to build your own ranker

Available ranking factors

- Document-level
  - bm25
  - max_lcs, field_mask, doc_word_count
- Field-level
  - LCS (Longest Common Subsequence)
  - hit_count, word_count, tf_idf
  - More :)

Extended search syntax

- And, Or, Not
  - hello | world, hello & world, hello -world
- Per-field search
  - @title hello @body world
- Field combination
  - @(title, body) hello world
- Search within first N chars
  - @body[50] hello

Phrase and proximity

- Phrase search
  - “hello world”
- Proximity search
  - “hello world”~10
- Distance support
  - hello NEAR/10 world
- Quorum matching
  - "the world is a wonderful place"/3

Even more

- Exact form modifier
  - “raining =cats and =dogs”
- Strict order
  - aaa << bbb << ccc
- field-start and field-end
  - ^hello world$
- Sentence / Zone / Paragraph

Full-text search in non-FT data

- Meta keywords search sometimes faster
  - __META_AUTHOR_ID_3235
  - __META_AUTHOR_NAME_Kelby
- Use sql_joinned_field
- Doesn't support ranges

Replacing «A*»-type search

- First letter search
  - __ARTIST_A, __ARTIST_B, __ARTIST_C, …
- Static ranges emulation with meta_keywords
  - __MY_RANGE_0, __MY_RANGE_1, …
- Not flexiable, but fast

Geodistance

- GEODIST(Lat, Long, Lat2, Long2) in Sphinx
  - Two pairs of float values (Latitude, Longitude)

```
SELECT *,
GEODIST(docs_lat, doc_long, %d1, %d2) as dist,
FROM sphinx_index
ORDER BY dist DESC
LIMIT 0, 20
```

Segments and Ranges

- Grouping results by
  - Price ranges (items, offers)
  - Date range (blog posts and news articles)
  - Ratings (product reviews)
- INTERVAL(field, x0, x1, …, xN)

```
SELECT
    INTERVAL(item_price, 0, 20, 50, 90) as range, @count
FROM my_sphinx_products GROUP BY range
ORDER BY range ASC;
```

Segments: Results example

```
+-------+--------+-------+---------+
| id    | weight | range | @count  |
+-------+--------+-------+---------+
| 34545 |      1 |     1 |     654 |
| 75836 |      1 |     2 |     379 |
| 94862 |      1 |     3 |      14 |
+-------+--------+-------+---------+
3 rows in set (0.00 sec)
```

Performance tricks: MVA

- MVA stands for Mlti Value Attributes
  - Array of 32/64bit integers
  - Supported by Real-Time and ondisk indexes
- Useful for shopping categories, page tags, linked documents or items
- Avioding JOIN on MySQL side

Indexing tricks: sql_joined_field

- Emulates GROUP_CONCAT
- Replaces JOIN while indexing
- Could store several text values from another table into one field.

```
sql_joined_field = dynamic_fields from query; \
    SELECT doc_id,  field_value \
    FROM  dynamic_fields_values \
    ORDER BY doc_id ASC
```

Reduce database size

- sql_file_field
  - Keeps huge text collections out of database.
  - sql_file_field = path_to_text_file
  - max_file_field_buffer needs to be set properly

Multiquery

- Saves time on common RT part
- Useful for Full-Text queries and faceted search
- AddQuery(…) API call
  - SphinxQL also supports it

6. Scale!

- Use more than one index
- Keep static data in on-disk indexes
  - Daily/Weekly reindexing
- Use 2/4/8 shards
  - It'll be 2/4/8 times faster
- Spread data across servers

Scaling: data sources

```
source source1
{
   …
   sql_query          = SELECT id, channel_id, ts, title, content
   FROM ljposts WHERE id>=$start and id<=$end
   sql_query_range    = SELECT 1, 7765020
   sql_attr_uint      = channel_id
   sql_attr_timestamp = ts
   …
}

source source2 : lj_source1
{
    sql_query_range   = SELECT 7765020, 10425075
}
```

Scaling: local indexes

```
index ondisk_index1
{
  source          = source1
  path            = /path/to/ondisk_index1
  stopwords       = stopwords.txt
  charset_type    = utf-8
}

index ondisk_index2 : ondisk_index1
{
  source          = source2
  path            = /path/to/ondisk_index2
}
```

Scaling: distributed index

```
index my_distribited_index1
{
  type         = distributed
  local        = ondisk_index1
  local        = ondisk_index2
  local        = ondisk_index3
  local        = ondisk_index4
}
…
  dist_threads = 4
…
```

Scalling: multi-box configuration

```
index my_distribited_index2
{
   type     = distributed
   agent    = 192.168.100.51:9312:ondisk_index1
   agent    = 192.168.100.52:9312:ondisk_index2
   agent    = 192.168.100.53:9312:rt_index
}
```

Know your queries

- Add extended query logging
  - *query_log_format = sphinxql*
- Enable performance counters
  - *./searchd –iostats –cpustats*
- Add application-level profiling

6. Backup & Restore

- OnDisk indexes are simply plain files
- Use FLUSH RTINDEX to dump data
- Use ATTACH INDEX

Crash safety

- Tune binlog_flush according to your hardware
- Set rt_flush_period
  - There still no *guarantees*, daemon will decide on his own.
  - Do FLUSH RTINDEX <rt_index_name>
- And backup your data!

Other best practices

- Keep Sphinx updated
  - http://sphinxsearch.com/downloads/release/
- Perform regular index checks
  - ./indextool –check <indexname>

Even more in:

- Visit our booth and ask questions!
- «Introduction to search with Sphinx» by Andrew Askyonoff
  - Email me to vlad@sphinxsearch.com for discount
- Invite us to speak!
  - Ping me via email vlad@sphinxsearch.com or twit to @vfedorkov

Feel free to ask questions :)

Thank you!