Российские
интернет-технологии
2010

Improving MySQL-based applications performance with Sphinx

Maciej Dobrzański
(Мачей Добжаньски)
Percona, Inc.

Introduction

Who am I?

- Consultant at *Percona, Inc.*
- What do I do?
  - Performance audits
  - Fix broken systems
  - Design architectures
- Typically work from home

Introduction

What is Percona, Inc.?

- Consulting company
- Provides services for MySQL applications
- Develops open-source software
  - Scalability patches for InnoDB
  - **XtraDB** storage engine for MySQL
  - **Xtrabackup** – free backup solution for InnoDB/XtraDB

What is MySQL?

What is MySQL?

MySQL is...

- Open-source relational database management system
- Popular enough to assume everyone here knows it

What is Sphinx?

What is Sphinx?

A standalone full-text search engine

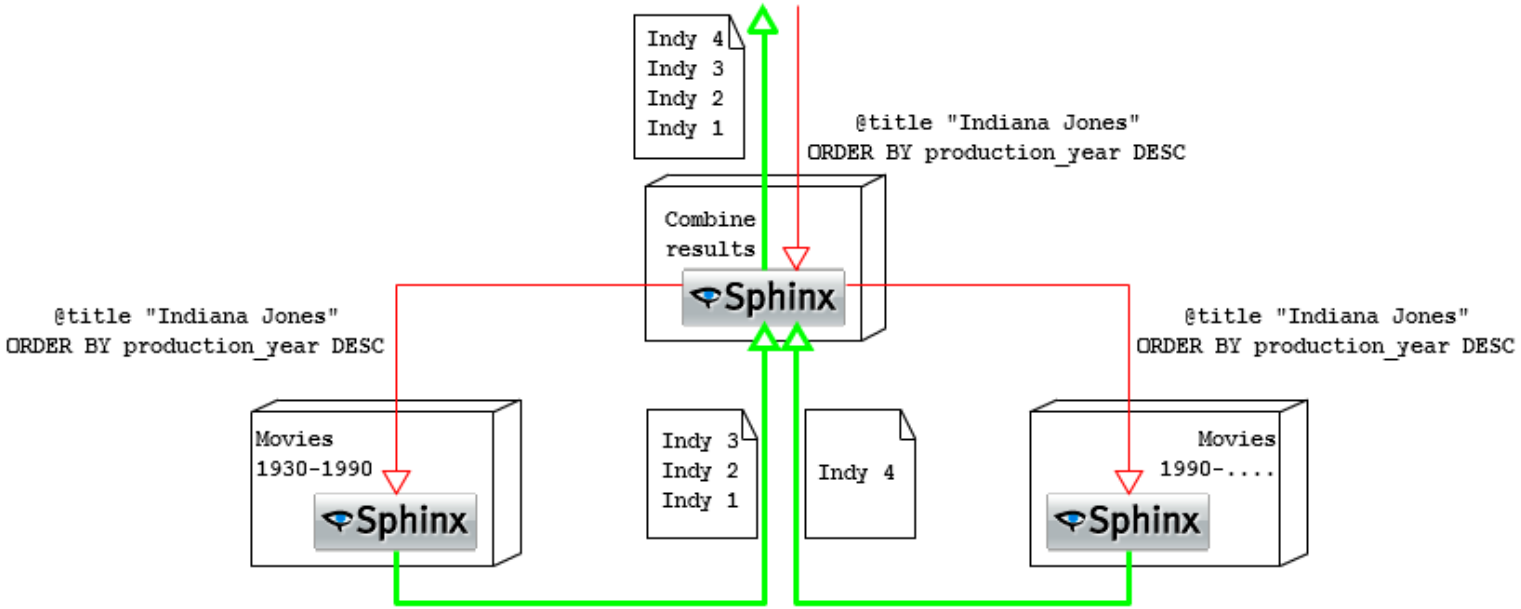
- Consists of two major applications
  - *indexer*
  - *searchd*
- More efficient than MySQL FULLTEXT
  - On larger data sets

What is Sphinx?

A standalone full-text search engine

- Can be easily scaled horizontally
  - Sphinx indexes can be distributed across many servers
  - Allows parallel searching
  - One instance becomes a dispatcher
    - Forwards queries to other instances
    - Combines results before sending them back to clients

What is Sphinx?



What is Sphinx?

Many additional features beyond just full-text search

- Indexable attributes for non-FTS filtering
  - numerical, multi-value and now also text
  - Example: limit results to rows which have `article_score>=2`
- Sorting results by an attribute or an expression
  - Example: `@weight+(article_score)*0.1`

What is Sphinx?

Many additional features beyond just full-text search

- Grouping results by an attribute
  - Additional support for timestamp attributes
  - Returns also row count per group – may be approximate
- Calculating expressions
  - Much faster than in MySQL as per recent benchmarks

What is Sphinx?

Anything else?

- On-line re-indexing
- Live index updates
- Extensive API available for many programming languages
  - PHP
  - Python
  - Java
  - many more

What is Sphinx?

There's even more!

- **SphinxQL** – MySQL server protocol compatible
  - Connect with **any MySQL client**
    - command line
    - API call, e.g. mysql_connect()
  - Run SQL-like queries

What is Sphinx?

Example use of SphinxQL

```
garfield ~ $ mysql -h 0 -P 9316
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 0.9.9-release (r2117)

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

mysql> SELECT *, (@weight*2) AS double_weight FROM movies
    -> WHERE MATCH('james bond') GROUP BY movie_id LIMIT 1;
+--------+--------+----------+-----------------+---------------+----------+--------+
| id     | weight | movie_id | production_year | double_weight | @groupby | @count |
+--------+--------+----------+-----------------+---------------+----------+--------+
| 452646 |   4675 |   262633 |            2003 |          9350 |   262633 |     65 |
+--------+--------+----------+-----------------+---------------+----------+--------+
```

How does Sphinx work with MySQL?

How does Sphinx work with MySQL?

Sphinx is **external application**; not part of MYSQL

- Uses own data files
- ***Needs*** memory
- Has to be queried separately
  - Sphinx API
  - SphinxQL
  - Sphinx Storage Engine for MySQL

How does Sphinx work with MySQL?

Sphinx is external application; not part of MySQL

- Updating Sphinx indexes has to be done separately too
  - Periodic data re-indexing with *indexer*
    - Some information may be outdated for a while
    - Can be optimized through re-indexing the latest changes only
  - Live index updates from applications
    - Applications need to write twice to both MySQL and Sphinx
    - Available only for attributes; full-text updates to come

How does MySQL work with Sphinx?

Example data source for Sphinx index

```
sql_query = SELECT mi.id, mi.movie_id, t.production_year,
   t.title, mi.info FROM movie_info mi JOIN title t
   ON t.id = mi.movie_id
sql_attr_uint                   = movie_id
sql_attr_uint                   = production_year
```

- Notice the source can be **any valid SQL query**
  - Uses joins to denormalize data for Sphinx
- Two integer attributes – movie_id and production_year

How does Sphinx work with MySQL?

Sphinx is not a full database (yet?)

- It's primarily a search engine
- It can return values stored as attributes, e.g: *movie_id, production_year*
- …but not any full-text searchable columns
- Results from Sphinx can be used to fetch full details from database

Important facts to know about MySQL

Important facts to know about MySQL

Uses B-TREE indexes to improve search performance

- Works great for equality operator (=)
- …and small range lookups: >, >=, <, <=, IN (list), LIKE
  - Range size relative to table size, not an absolute value
  - Large range often turns into plain scan

IMPORTANT FACTS TO KNOW ABOUT MYSQL

MySQL can use *any left-most part* of an index

- INDEX (a, b, c) can fully optimize both:

  (1) SELECT * FROM T WHERE a=9

  (2) SELECT * FROM T WHERE a=9 AND b IN (1,2) AND c=4

  …but **not** any of:

  (3) SELECT * FROM T WHERE b=7 AND c=1

  (4) SELECT * FROM T WHERE a=9 AND c=2 (may still use index for a=9 only)

- No good indexes means you may need a new one

Important facts to know about MySQL

Each index slows down writes to a table

- Index is an organized structure, it has to be maintained
- There can't be too many or performance will suffer

MySQL can typically use only one index per query

- There are rare exceptions – index merge optimizations
- Merges are often not good enough – an observation

Important facts to know about MySQL

These work great in MySQL

- Index optimized searching
  - A query which uses indexes **efficiently** is fast enough
  - B-TREE lookups are typically very efficient
  - FULLTEXT indexes can be the exception
- Index optimized sorting and grouping
  - Rows are read in the proper order

IMPORTANT FACTS TO KNOW ABOUT MYSQL

These can cause problems in MySQL

- Full table scans
  - No index is used
  - Query reads entire table row by row checking for matches
- Large scans related to poor selectivity
  - An index is used, but it is not selective enough
  - MySQL has to read a lot of rows and reject many of them

Important facts to know about MySQL

These can cause problems in MySQL

- Search on many combinations of columns in a single table
  - Each combination may require new index
  - Can't have too many indexes in table at the same time
- Handling multi-value properties in searches
  - Keywords, tags
  - Such queries often can't be optimized very well

IMPORTANT FACTS TO KNOW ABOUT MYSQL

These can cause problems in MySQL

- Sorting or grouping not done through indexes
  - Requires rewriting rows into temporary storage
  - ***At least*** one additional pass over results to complete
  - LIMIT ***does not work*** until ***all*** matches are found ***and*** sorted/grouped

Important facts to know about MySQL

Indexes and data may be cached in memory

- key_buffer and filesystem cache for MyISAM tables
- innodb_buffer_pool for InnoDB tables
- No guarantees what is in RAM
  - MySQL has no option to lock certain data in buffers

Important facts to know about MySQL

Full-text support in MySQL

- Available through FULLTEXT keys
- Only supported by MyISAM engine
  - MyISAM uses *table level locking*
  - May become a showstopper for busy databases
- Cannot be used together with any other index
  - Even index merge will not work

Important facts to know about Sphinx

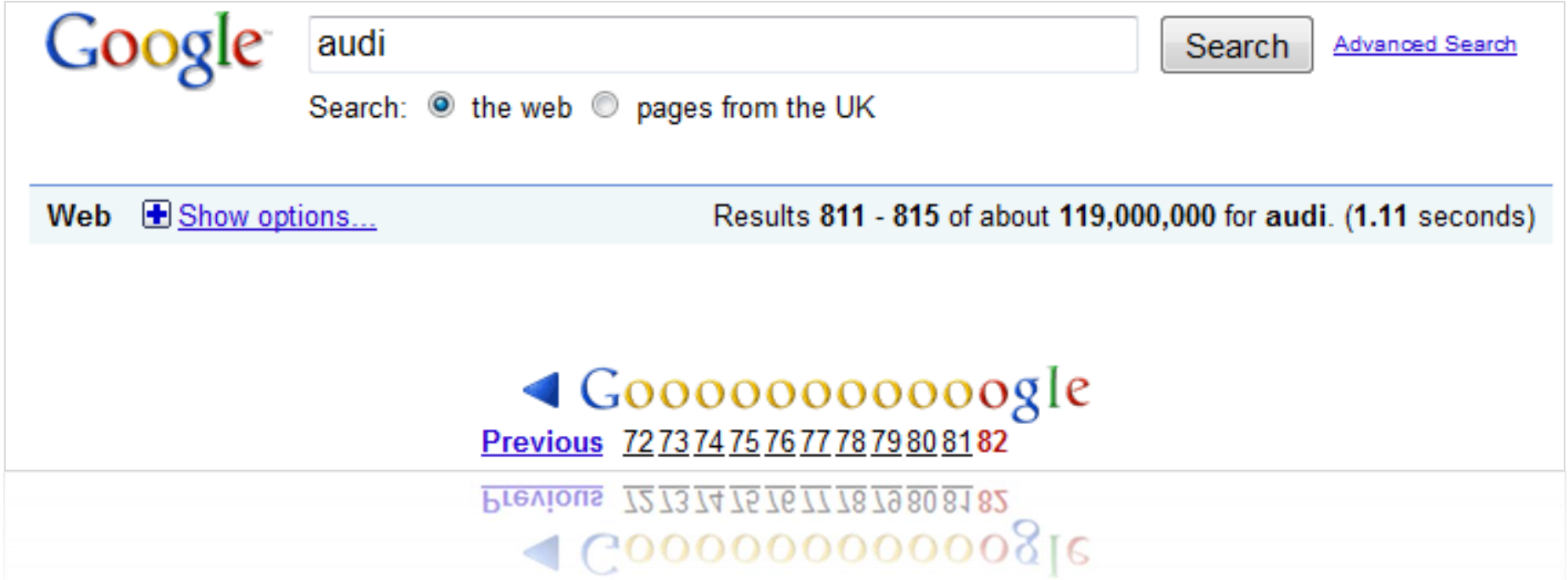
Important facts to know about Sphinx

Search remembers no more than *max_matches* results

```
| total          | 1000  |
| total_found    | 2255  |
```

- Other results are *ignored* before sending them to client
- Saves some CPU and RAM
- All results are often unnecessary
- **Accuracy costs**

IMPORTANT FACTS TO KNOW ABOUT SPHINX



Important facts to know about Sphinx

Grouping is done in fixed memory

- Results *may* be approximate
  - When number of matches exceeds *max_matches*
- Inaccuracy depends on *max_matches* setting
  - The larger the more accurate grouping results
  - Growing *max_matches* can reduce performance
- Accuracy costs

Important facts to know about Sphinx

MySQL

```
SELECT ..., COUNT(1) _c
   FROM movie_info
WHERE
   MATCH (info)
   AGAINST ('"story"'
        IN BOOLEAN MODE)
   GROUP BY movie_id
   ORDER BY _c DESC LIMIT 4
```

Sphinx (uses SphinxQL)

```
SELECT *
   FROM movies
WHERE
   MATCH ('@info "story"')
GROUP BY movie_id
ORDER BY @count DESC 4
```

Important facts to know about Sphinx

MySQL

```
+----------+-----------+
| movie_id | COUNT(1)  |
+----------+-----------+
|    30372 |        15 |
|   855624 |        13 |
|   590071 |        13 |
|   143384 |        12 |
+----------+-----------+
```

Sphinx

```
+----------+--------+
| movie_id | @count |
+----------+--------+
|    30372 |     15 |
|   855624 |     13 |
|   143384 |     12 |
|   590071 |     12 |
+----------+--------+
```

Important facts to know about Sphinx

Full copy of attributes is always kept in RAM

- If attribute storage was set to ‘extern’ – the typical use
- Preloaded on start
- Never read from disk again once Sphinx is up
- Guarantees certain performance
- Calculate the storage requirements properly
  - Sphinx may want to allocate too much memory

Important facts to know about Sphinx

Sphinx stores rows in blocks

- 64 rows per block
- Meta data contains (min, max) range of every attribute
- Allows quick rejection when filtering by attributes
  - No need to scan every row individually

MySQL v Sphinx
Performance

Full-text search performance

Uses full IMDB database

Imported into MySQL and indexed with Sphinx

FULL-TEXT SEARCH PERFORMANCE

MySQL

```
SELECT COUNT(1)
   FROM movie_info
WHERE
   MATCH (info)
   AGAINST ('"james bond"'
        IN BOOLEAN MODE)
```

Sphinx (uses SphinxQL)

```
SELECT *
   FROM movies
WHERE
   MATCH ('@info "james
   bond"')
```

FULL-TEXT SEARCH PERFORMANCE

MySQL

```
+----------+
| COUNT(1) |
+----------+
|     2255 |
+----------+
1 row in set (0.13 sec)
```

Sphinx

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| total         | 1000  |
| total_found   | 2255  |
| time          | 0.003 |
...
```

Scan performance

Uses full IMDB database

Imported into MySQL and indexed with Sphinx

Scan performance

MySQL

```
SELECT COUNT(1)
   FROM title
WHERE
   production_year >= 1990
   AND
   production_year <= 2000
```

No index on `production_year`

Sphinx (uses SphinxQL)

```
SELECT *
   FROM titles
WHERE
   production_year >= 1990
   AND
   production_year <= 2000
```

SCAN PERFORMANCE

MySQL

```
+----------+
| COUNT(1) |
+----------+
|   239203 |
+----------+
1 row in set (1.09 sec)
```

Sphinx

```
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| total         | 1000   |
| total_found   | 239203 |
| time          | 0.051  |
...
```

More complex case search by keywords

Uses full IMDB database

Imported into MySQL and indexed with Sphinx

SEARCH BY KEYWORDS

MySQL

```
SELECT t.id FROM title t
   JOIN movie_keyword mk
   ON mk.movie_id = t.id
   JOIN keyword k
   ON k.id = mk.keyword_id
WHERE
   k.keyword IN ('beautiful-
   woman', 'women', 'murder')
GROUP BY t.id ORDER BY
   production_year DESC LIMIT 3
```

Sphinx (uses SphinxQL)

```
SELECT *
   FROM keywords
WHERE
   MATCH
   ('@keywords
        ("beautiful-woman"|
        "women"|"murder")')
ORDER BY production_year DESC
   LIMIT 3
```

SEARCH BY KEYWORDS

MySQL

```
+--------+
| id     |
+--------+
| 561959 |
|  74273 |
| 344814 |
+--------+
3 rows in set (1.84 sec)
```

Sphinx

```
+--------+
| id     |
+--------+
| 561959 |
|  74273 |
| 344814 |
+--------+
time = 0.015
```

Search by keywords

Sphinx returns

- Values of the indexed attrubites
- Meta information about search and results
- No text
  - Recent version can actually store and return short strings
  - But only defined as attributes, not full-text searchable

SEARCH BY KEYWORDS

Use that information to fetch full details from MySQL

```
mysql> SELECT t.id, t.title FROM title t WHERE
       t.id IN(561959, 74273, 344814)
+--------+----------------------------------------+
| id     | title                                  |
+--------+----------------------------------------+
|  74273 | Blue Silence                           |
| 344814 | Marvin: The Life Story of Marvin Gaye  |
| 561959 | The Red Man's View                     |
+--------+----------------------------------------+
```

SEARCH BY KEYWORDS

MySQL

```
+--------+--------------------+
| id     | title              |
+--------+--------------------+
|  74273 | Blue Silence       |
| 344814 | Marvin: The Li...  |
| 561959 | The Red Man's ...  |
+--------+--------------------+
```

Sphinx

```
+--------+-----------------+
| id     | production_year |
+--------+-----------------+
| 561959 |            2014 |
|  74273 |            2013 |
| 344814 |            2012 |
+--------+-----------------+
```

Notice MySQL returned rows in different order!

Search by keywords

The order in SQL can only be guaranteed with ORDER BY!

What is the solution?

- Append `ORDER BY production_year DESC`
  - applies to only small number of rows, so it's probably okay

or

- Remember the order of Sphinx results in application
- Restore it after reveiving data from MySQL

SEARCH BY KEYWORDS

What if „keywords” were numerical identifiers?

- Create „fake keywords” and index them as text
- Convert numbers into strings when building index

```
sql_query = SELECT t.id,
GROUP_CONCAT(CONCAT('KEY_', mk.keyword_id))
FROM title t JOIN movie_keyword mk ON t.id = mk.movie_id
GROUP BY t.id
```

- Run full-text searches using strings such as "KEY_1234"

FLEXIBLE SEARCH

FLEXIBLE SEARCH

A data structure describing user profile

```
CREATE TABLE `members` (
   `user_id` int(10) unsigned,
   `user_firstname` varchar(50) unsigned,
   `user_surname` varchar(50) unsigned,
   `user_dob` date unsigned,
   `user_lastvisit` datetime unsigned,
   `user_datetime` datetime unsigned,
   `user_bio` unsigned,
   `user_hasphoto` tinyint(2) unsigned,
   `user_hasvideo` tinyint(2) unsigned,
   ...
```

Flexible search

Flexible search typically means

- Search conditions may involve any number of columns in any combination
- Sorting may be done on one of many columns as well

Often impossible to add all necessary indexes in MySQL

FLEXIBLE SEARCH

Many columns may have very low cardinality

- Example: user_gender
- MySQL would not even consider using index for such column

It may be very difficult to make it work fast in MySQL

- When tables or traffic are *large enough*

Flexible search

How does Sphinx help?

- Scans are optimized
- Optimizations apply to all columns
- Possibility to use „fake keywords”
- Data can be split across several instances
  - Parallel search
  - No extra application logic necessary to combine results

SUMMARY

Summary

Sphinx can be of great help to many MySQL-based apps

- Developed to work better where MySQL performs poorly
  - Text search
  - Large scans
  - Filtering on many combinations of columns
  - Handling multi-value properties

Summary

Sphinx can be of great help to any MySQL-based apps

- Comes with features that can actually replace database
- Easily scalable
- Actively developed
- You can sponsor development and have features you need done soon
  - No need to wait long until some functionality „appears”

Sphinx

http://www.sphinxsearch.com/

Percona Consulting

http://www.percona.com/

THANK YOU!