Creating Full-text based services with Sphinx and MySQL

Oct 1st, Percona Live New York

Vladimir Fedorkov, astellar.com

About me

- Use Sphinx in production since 2006
- Performance geek
  - blog http://astellar.com
  - Twitter @vfedorkov
- Focused on LAMP stack tuning
  - Data layer is most common bottleneck

Search is important

- Keep customer satisfied
  - Let visitor find what he need
  - Even if they don't really know!
  - Make them get back to you again
- Search is the way to explore the website
  - Find something that your customer doesn’t know
  - Show your customer what you want to show

Fast search is important

- Rule 0.1 … 1 … 10
  - For users
  - For search engines
- Utilize less resources
- Use less hardware
- More functionality

Good search is important

- Relevance
- Flexibility
- Simple maintenance
- Fault tolerance

Available solutions

- Most databases has it's integrated FT engines
  - MySQL: MyISAM FullText index
    - Recently released FT support in InnoDB
- Standalone solutions
  - Solr, Lucene, Sphinx.
    - Bill Karvin from Percona did benchmarks
- External services
  - IndexDen, SearchBox, Flying Sphinx, WebSolr, …

Today's agenda

- What Sphinx can do for you
- How to integrate Sphinx into your application
- Sphinx recipes
  - Technical cookbook
  - Sphinx-based services
- When and how Sphinx could help
  - Query fine tuning
  - Features you want to hire
  - Sphinx day-to-day operations

Closer look to Sphinx

- Age: 10+ years old open source search server
  - Written on C++
  - Separate daemon, just like MySQL
- Available for Linux, Windows x86/64, Mac OS
  - Can be built on AIX, iPhone and some DSL routers
- Open source and free!
  - GPL v2

Sphinx is not

- Plugin to MySQL
  - SphinxSE is not a Sphinx itself
- SQL-based search engine
  - Non-SQL APIs are available
- Database replacement
  - Yet?

Sphinx is

- Standalone open source search server
  - With on-disk and real-time indexes
- **Fast**
  - Both indexing and search
    - See benchmarks
- Highly **scalable**
  - Local and disributed search supported
  - Scales out of the box

Sphinx records

- Searching though Billions of documents
  - Over 30,000,000,000 at Infegy
  - Over 26,000,000,000 at boardreader.com
    - over 8.6Tb indexed data across 40+ boxes
- Serves 200,000,000+ queries per day
  - craigslist.org 2,000+ QPS against 15 Sphinx boxes
- 10-1000x faster than MySQL on full-text searches
  - Even faster on faceted search queries
- Overall response time drop 4-6 times without fine tuning
  - My personal experience

Why so fast?

- Architecture designed for search
- Attributes are always stays in memory
  - If you can’t prevent full scans – do it fast!
- Grouping and sorting in fixed memory
- Attribute search blocks skipping
- Scaling out of the box

Basics

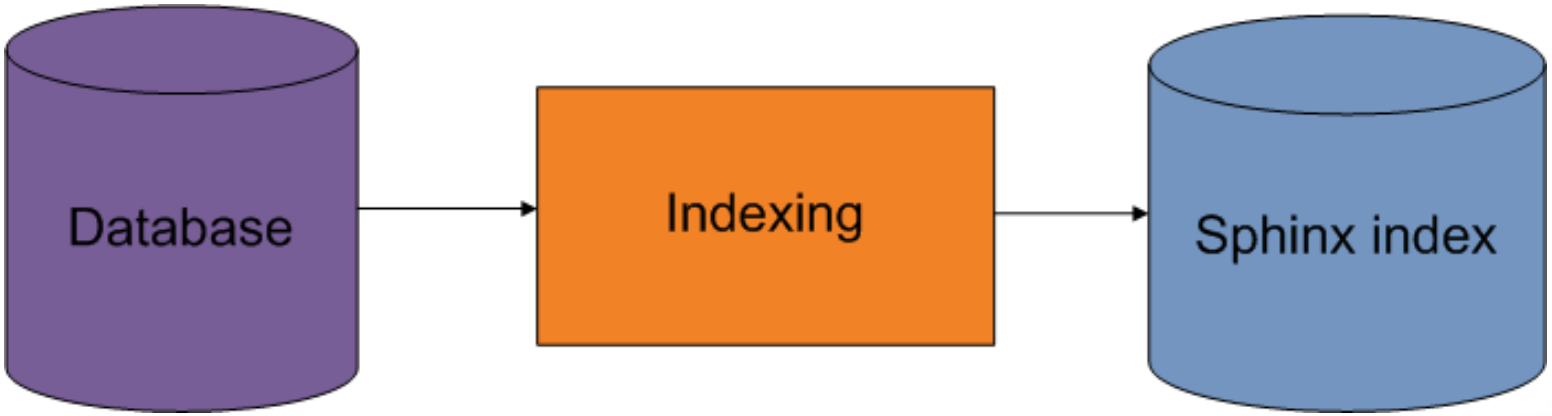
- Installation and setup
- Integration and basic search
- Faceted search
- Real-time search

Installation: How?

- From binary packages
  - http://sphinxsearch.com/downloads/
- From source
  - http://sphinxsearch.googlecode.com/svn/
  - configure && make && make install
    - Make sure to use --enable-id64
    - for huge document collection
    - already included in pre-compiled packages

Initial configuration: indexing

- Where to look for data?
- How to process it?
- Where to store index?



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/idx

      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
}
```

Running indexer

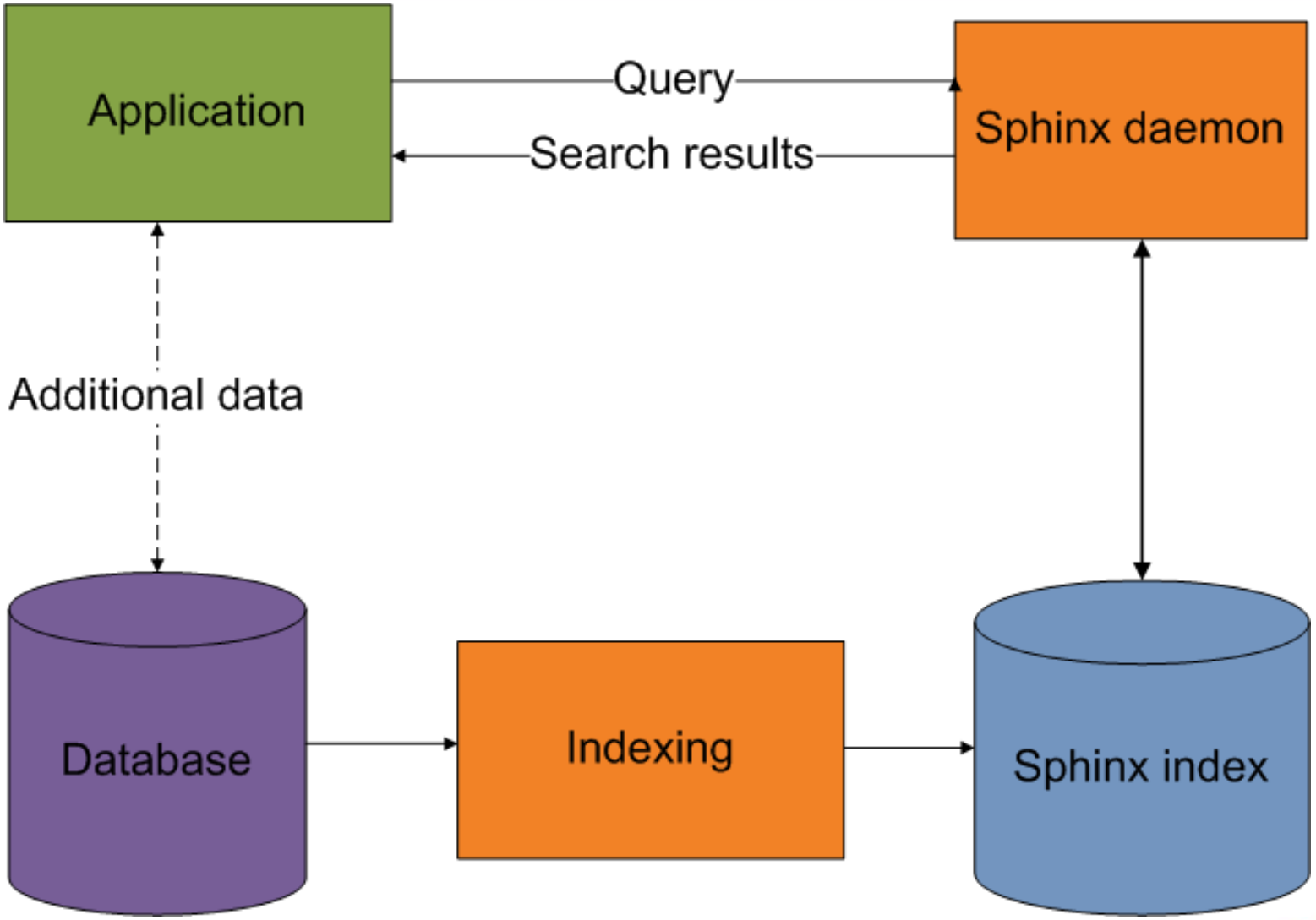
```
$ ./indexer my_sphinx_index
Sphinx 2.0.2-dev (r2824)
Copyright (c) 2001-2010, Andrew Aksyonoff
Copyright (c) 2008-2010, Sphinx Technologies Inc (http://sph...

using config file './sphinx.conf'...
indexing index 'my_sphinx_index'...
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
```

Index files

```
$ ls -lah idx*
-rw-r--r-- 1 vlad vlad  12M 2010-12-22 09:01 idx.spa
-rw-r--r-- 1 vlad vlad 334M 2010-12-22 09:01 idx.spd
-rw-r--r-- 1 vlad vlad  438 2010-12-22 09:01 idx.sph
-rw-r--r-- 1 vlad vlad  13M 2010-12-22 09:01 idx.spi
-rw-r--r-- 1 vlad vlad    0 2010-12-22 09:01 idx.spk
-rw-r--r-- 1 vlad vlad    0 2011-05-13 09:25 idx.spl
-rw-r--r-- 1 vlad vlad    0 2010-12-22 09:01 idx.spm
-rw-r--r-- 1 vlad vlad 111M 2010-12-22 09:01 idx.spp
-rw-r--r-- 1 vlad vlad    1 2010-12-22 09:01 idx.sps
$
```

Initial configuration: Run search query



ToDo

- Run the daemon
- Connect to daemon
- Send search query
- Read results

Configuring searchd

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

    query_log           = query.log
    query_log_format    = sphinxql

    pid_file            = searchd.pid
}
```

Running sphinx daemon!

```
$ ../bin/searchd -c sphinx.conf
Sphinx 2.0.2-dev (r2824)
Copyright (c) 2001-2010, Andrew Aksyonoff
Copyright (c) 2008-2010, Sphinx Technologies
  Inc (http://sphinxsearch.com)

using config file 'sphinx.conf'...
listening on 127.0.0.1:9312
listening on 127.0.0.1:9306
precaching index `idx'
precached 1 indexes in 0.028 sec
```

Connecting to Sphinx

- Sphinx API
  - PHP, Python, Java, Ruby, C is included in distro
  - .NET, Rails (via Thinking Sphinx) via third party libs
- SphinxQL
  - MySQL-compatible protocol
- SphinxSE
  - Storage engine for MySQL

Sphinx API

```
<?php
require ( "sphinxapi.php" ); //from sphinx distro
…
$cl = new SphinxClient();
…
$res = $cl->Query ( “I love Sphinx", “idx" );

//error processing is here

var_dump ( $res );
?>
```

The results (query “I love Sphinx”)

```
["error"]=> "", ["warning"]=> "", ["status"]=> 0
["fields"]=> array(3) { "title", "content" }
["attrs"]=> array(2) { "channel_id" =>  1, "ts"=> 2 }
["matches"]=> array(20) { … }
["total"]=> string(2) "51"
["total_found"]=> string(2) "51"
["time"]=> string(5) "0.006"
["words"]=> array(2) {
  ["love"]=> {“docs"} =>"227990", "hits"=>"472541"}
  ["sphinx"]=>{"docs"=>”114", "hits"=>"178"}
}
```

What is document match?

- Document id
  - Not a document itself!
    - You will need to retrieve text fields back from database
      - Or cache
- Document weight
- Non-FT attribute values
  - For each attributes

Document match example

```
["id"]=> int(6598265)
["weight"]=> string(3) "101"
["attrs"]=>  array(2) {
      ["channel_id"]=> int(454928)
      ["ts"]=> int(1102858275)
}
```

Sphinx API complete example

```
require ( "sphinxapi.php" );
$cl = new SphinxClient ();
$cl->SetServer ( $host, $port );
$cl->SetArrayResult ( true );
$cl->SetWeights ( array ( 100, 1 ) );
$cl->SetFilter ( … );
$cl->SetMatchMode ( … );
$cl->SetRankingMode ( … );
$res = $cl->Query ( «I love sphinx», «idx»);
```

Per-Field weights

- SetWeights is outdated
- Use SetFieldWeights instead

SetFieldWeights("titile" => 100, "content" => 1)

- Document weight = “title” * 100 + “content”
- Works on per-query basis
- Per-index weights are also tunable

SetMatchMode

- SPH_MATCH_ALL
- SPH_MATCH_ANY
- SPH_MATCH_PHRASE
- SPH_MATCH_BOOLEAN
- SPH_MATCH_FULLSCAN
- SPH_MATCH_EXTENDED

SetRankingMode

- SPH_RANK_PROXIMITY_BM25 (default)
- SPH_RANK_BM25
- SPH_RANK_NONE
- SPH_RANK_WORDCOUNT
- SPH_RANK_PROXIMITY
- SPH_RANK_FIELDMASK
- SPH_RANK_SPH04
- SPH_RANK_EXPR

Expression based ranker (SPH_RANK_EXPR)

- Document-level
  - bm25
  - max_lcs, field_mask, doc_word_count
- Field-level
  - LCS (Longest Common Subsequence)
  - hit_count, word_count, tf_idf
- External attributes
  - Customer rating, reviews, popularity

Search stages

- Full-text matches lookup
- NON-FT Filtering
- Sorting & grouping
- Returning result set
- Aggregating results
  - For distributed search

Full-Text filtering

- And, Or
  - hello | world, hello & world
- Not
  - hello -world
- Per-field search
  - @title hello @body world
- Field combination
  - @(title, body) hello world
- Search within first N
  - @body[50] hello
- Phrase search
  - “hello world”
- Per-field weights
- Proximity search
  - “hello world”~10
- Distance support
  - hello NEAR/10 world
- Quorum matching
  - "the world is a wonderful place"/3
- Exact form modifier
  - “raining =cats and =dogs”
- Strict order
- Document structure support
  - Sentence
  - Zone
  - Paragraph

Non-Full-text filtering

- Number of API calls:
  - SetFilter(), SetFilterRange(), SetFilterFloatRange()
- Works for non-full-text attributes
- Same as WHERE
  - a = 5, a < 5, a > 5, a BETWEEN 3 AND 5
- Applies AFTER full-text filters

Non full-text attribute types

- Integers
  - Int 32bit unsigned (only)
  - Int 64bit signed (only)
  - Set of integers (Multi-Value-Attribute, MVA)
  - Limited ints using bitcount
- Floats
  - Includes float pairs for GEO anchors
- Strings
- Timestamps

Grouping

- SetGroupBy API call

```
<?php
require ( "sphinxapi.php" );
…
//initializing search
…
$cl->SetGroupBy ( "ts", SPH_GROUPBY_YEAR, "@group desc" );
$res = $cl->Query ( "I love sphinx",“idx");
var_dump ( $res );
?>
```

Grouping result (single match)

```
["id"]=> 7637682
["weight"]=> 404652
["attrs"]=>
array(4) {
  ["channel_id"]=> 358842
  ["ts"]=>  1112905663
  ["@groupby"]=> 2005
  ["@count"]=> 14
}
```

Grouping results

[0] ["@groupby"]=>2005, ["@count"]=> 14

[1] ["@groupby"]=>2004, ["@count"]=> 27

[2] ["@groupby"]=>2003, ["@count"]=> 8

[3] ["@groupby"]=>2002, ["@count"]=> 1

API error processing

```
$res = $cl->Query ($q, $index);
if ( $res===false )
{
   $sph_error = $cl->GetLastError();
} else if ($cl->GetLastWarning()){
    …
}
```

More functionality

- SetFilter & SetFilterRange
- SetGeoAnchor
- SetSortMode
- SetIndexWeights
- Multiquery support
- BuildExcerpts

SphinxQL

- Same functionality as for APIs
- Speed is the same
- SQL-based
- Works without MySQL
  - MySQL client library required
    - mysqli
- The only way to push data to the Real-Time index
- Required different port
- Almost same syntax as in MySQL
  - But not exactly the same

SphinxQL. Search 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

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

Sphinx

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 idx
    -> 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
  - 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

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

Group by example

```
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)
```

Good search is not plain search

- Drill-down (narrow search, faceted search)
- Typos correction
- Search string autocompletion
- Related documents

Search by GEO-Distance

- Bumping up local results
  - Requires coordinates for each document
  - Two pairs of float values (Latitude, Longitude)
- GEODIST(Lat, Long, Lat2, Long2) in Sphinx

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

Search within range

- Price ranges (items, offers)
- Date range (blog posts and news articles)
- Ratings, review points
- 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;
```

Misspells correction service

- Provides correct search phrase
  - "Did you mean" service
- Allows to replace user's search on the fly
  - if we're sure it's a typo
    - "ophone", "uphone", etc
  - Saves time and makes website look smart
- Based on your actual database
  - Effective if you DO have correct words in index

Bundled solution

- Helper script is located in /misc/suggest/
  - suggest.conf includes required Sphinx index
  - suggest.php is an actual implementation
- Requires PHP and MySQL to work
- Based on the tri-grams & levenshtein function

Limitations and features

- Provided as a showcase, not a complete service
- Doesn’t work with UTF8
  - PHP function limitation
- Based on your actual database
  - Index required rebuild as you have new data
- Script is only provides you word-by-word correction
- Works better in combination with autocompletion service

Autocompletion service

- Suggest search queries as user types
  - Show most popular queries
  - Promote searches that leads to desired pages
  - Might include misspells correction

Autocompletion implementation

- Enable prefix indexing
  - Set min_prefix_len and prefix_fields
- Use pre-built index with search prases
  - Based on user's input
  - Based on document statistics
- Use star search: MATCH ('ipho*')
  - It's sometimes wise to delay search until 3-4 letters has typed

Related search

- Improving visitor experience
  - Providing easier access to useful pages
  - Keep customer on the website
  - Increasing sales and server's load average
- Based on documents similarity
  - Different for shopping items and texts
  - Ends up in data mining

Related search implementation

- Uses main Sphinx index
- Basic implementation uses quorum operator
  - “Sony NEX-5N”/2
  - “Mitt Romney wonders why airplane windows don’t open”/2
- Next step: use custom ranking
- Next step: enable statistics
  - Keywords/Phrases
  - Shopping experience
- Next step: use internal information

Excerpts (snippets)

- **BuildExcerpts() or CALL SNIPPETS**
- Options
  - before_match ()
  - after_match ()
  - chunk_separator (…)
  - limit
  - around
  - force_all_words

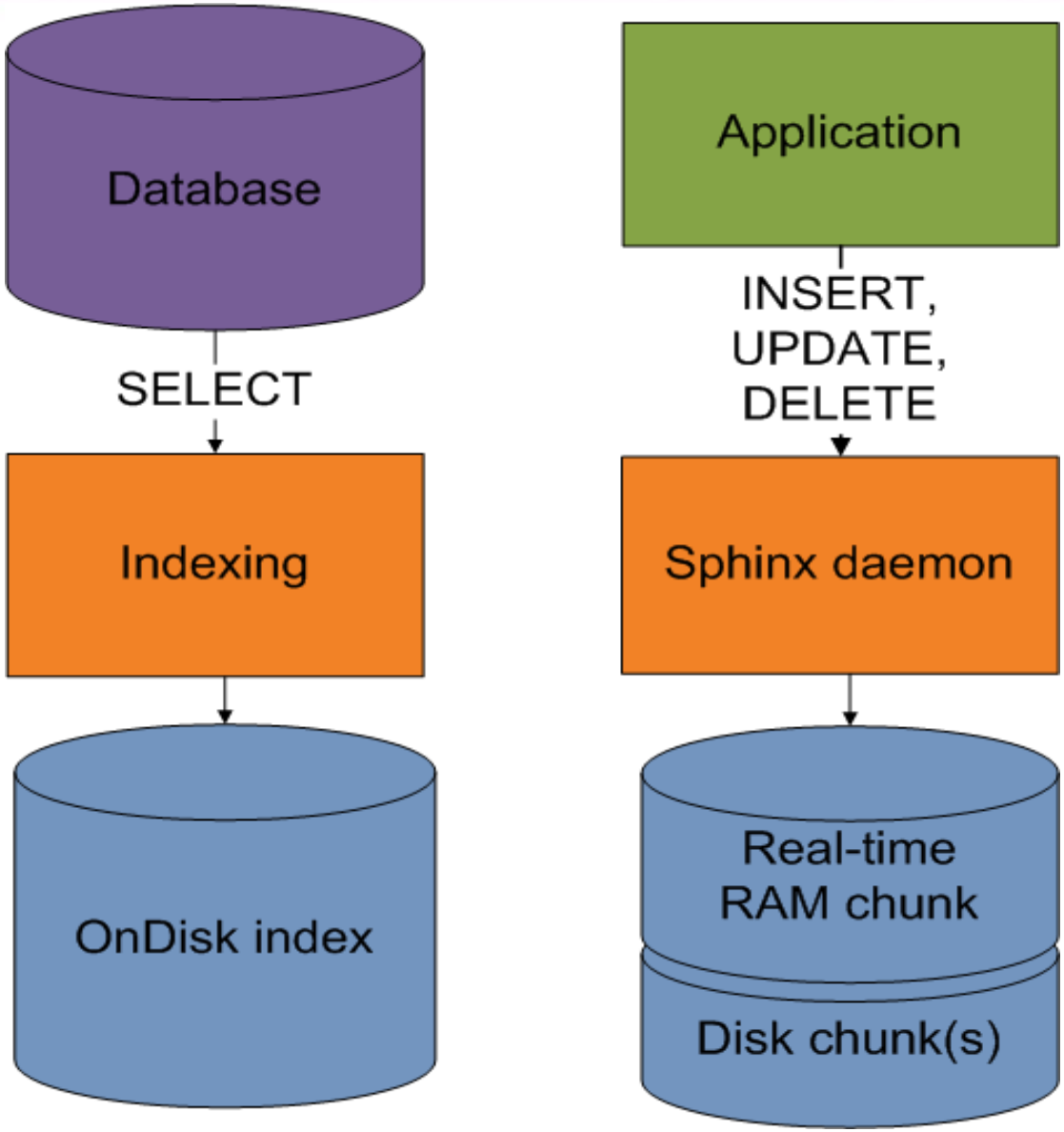
Steps for better search

- Basic search
  - Installation
  - Indexing
  - Integration
- Facets
- Extended services
  - Autocompletion
  - Misspells
  - Related
- Excerpts

Performance

- Minimize indexing delay
- Speed up search response time

On disk vs Real-time indexes



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

Real Time engine config

```
index rt
{
        type                    = rt
        rt_mem_limit            = 512M
        rt_field                = title
        rt_field                = content
        sql_attr_uint           = channel_id
        sql_attr_timestamp      = ts
}
```

Crash safety

- Tune binlog_flush according to your hardware
- Set rt_flush_period
  - There still no guarantees, daemon will decide on hisown.
- Do backup your data

RT — Memory Utilization

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

RT — Disk chunks

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 does it mean?

- Search is slowing down
- Disk chunks are static
  - Disk space used inefficiently
  - Removed and updated documents stays in index
- Many unnecessary IO requests

What to do?

- Set up monitoring
- 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

Speeding up the search

- Profile
- Scale
- Optimize
- Compact

Remove high-frequency keywords

- «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 right rankers

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

Use faster Full-text search for non-text data

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

Speeding up high-selective queries

- 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
- Offloading MySQL

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

Distributed search

- 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, …
      sql_query_range    = SELECT 1, 7765020
      sql_attr_uint      = channel_id
      sql_attr_timestamp = ts
      …
}

source source2 : 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 load

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

Backup & Restore

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

Some best practices

- Do monitor your services
  - Will help in both ways
- Perform regular health check
  - ./indextool –check <indexname>
  - Could also optimize killlists in RT

Where to get more

- Catch me on local meetups and conferences
  - Invite me to speak
- Follow me on twitter @vfedorkov
- Looking for better search?
- Need a faster website?
- Looking for private training for your team?
  - Send me an email to support@astellar.com

Thank you!

Twitter @vfedorkov

Website: http://astellar.com