PALOMINODB

OPERATIONAL EXCELLENCE
FOR DATABASES

MySQL Anti-Queries and Sphinx Search

Percona Live, MySQL Users Conference
Santa Clara, 2013

Vlad Fedorkov
www.palominodb.com

Let’s meet each other

- Performance geek
- Working for PalominoDB
  - http://www.palominodb.com
- Twitter @vfedorkov
- Now fighting with clouds

What is MySQL

- Database we love
- Database we use
  - So sometimes we hate it too
  - Depends on how much data you have
- Most important
  - Database that improving
    - Thanks to Percona, Oracle and MariaDB!
    - Special thanks to MySQL community
- Does it work in every case?

MySQL today

- Software
  - Can’t complain about concurrency anymore
  - Warm up is much-much better today
  - Optimizer doing much better now
  - Replication is now multi-threaded
- Environment
  - Flash storage makes IO very fast and concurrent
  - Cheap and fast RAM
  - Awesome cloud services available
  - Galera, Tungsten, MySQL cluster

YOU WERE BORN TO BE REAL, NOT TO BE PERFECT.

What's wrong with MySQL?

- One-by-one:
  - ACID compliance is inconveniently slow
  - B-Tree index is not perfect for some queries
  - Query execution is single threaded
  - Full-text indexes are not that fast
  - Replication wants to keep data consistent
  - lot more inconvenient things just ask me!
- So maybe some queries are bad, not MySQL itself?

MySQL as a microscope

- Excellent for specific operations
  - Very fast at primary key lookup
  - Fast enough on index lookups
    - some range scans too
- Makes sure your data is safe
- Optimized for transactional (InnoDB) workload

Full-table scans

- SELECT * FROM table WHERE myfunc(a) = 5
- SELECT * FROM table WHERE a NOT IN (1,2,3)
  - WHERE a <> 5
- SELECT * FROM table WHERE c LIKE ‘%ing’
- SELECT COUNT(*) FROM …
- Reading the whole table is expensive
- Buffer pool flooded with rarely used data
- Indexes are not helpful

Inefficient index usage

- SELECT * FROM table WHERE enabled=0
  - archived, deleted
- SELECT * FROM table WHERE sex=1
  - WHERE age_range=5
- SELECT * FROM table WHERE state=CA
  - WHERE city='New York'
- Better than full-table scan but still inefficient

Some sorting operations

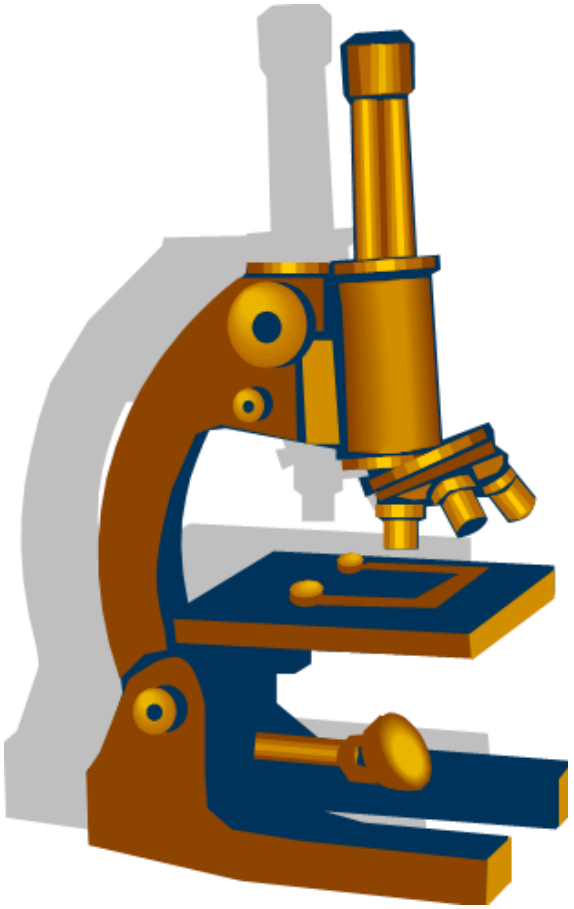
- ORDER BY against non indexed field
- ORDER BY a DESC, b ASC, c DESC
- ORDER BY RAND() ASC
  - ORDER BY RAND() DESC works much better

Temporary table operations

- Some operations forces MySQL to create
  - In memory table
  - On disk table
- Subqueries
- GROUP BY
- DISTINCT + ORDER BY
  - If you have TEXT or BLOB field MySQL will create on disk table at all times

These queries like nails

C’mon, you can do that!!!



… or use something more convenient?

What do we need?

- Avoid full-scans as much as we can
  - Make them faster when we can’t get rid of them
- Avoid slow IO operations
  - Put most data to memory
    - It is cheap now

Let’s meet Sphinx

- Stand alone daemon
  - Written in C++ and very fast
- Works without MySQL
  - At all
- Can be connected using mysql client
  - PHP/Ruby/JDBC/.NET/etc
- Runs on lots of platforms
  - Linux/Windows/Mac/everywhere

Sphinx

Pros

- Highly scalable
- Keeps vital data in memory
- Very fast in scans
- Powerful full-text syntax
- High availability support

Cons (yet)

- Don’t have B-tree indexes
- No out-of-the-box replication from MySQL
- Not completely ACID

Full-table scans

- Sphinx keeps attributes in memory at all times
- It scale queries
  - Well actually you have to tell how
    - But only once
- Limit amount of documents to walk though
- You can utilize Full-Text search power
  - As a nice bonus ☺

Full-Text functions

- 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
- Sentence / Zone / Paragraph
- Custom documents weighting & ranking

SphinxQL.

```
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>
```

What is SphinxQL?

- SQL-based query language in Sphinx
- Works without MySQL!
  - MySQL client library required
    - JDCB, .NET, PHP, Ruby
- Required different port
- Almost same syntax as in MySQL
  - But not exactly the same

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

Extended syntax

```
mysql> SELECT …, YEAR(ts) as yr
    -> FROM sphinx_index
    -> WHERE MATCH('I love Sphinx')
    -> GROUP BY yr
    -> WITHIN GROUP ORDER BY rating DESC
    -> 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)
```

GEO-Distance support

- 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
```

Range support

- 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;
```

Extended services

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

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

Autocompletion service

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

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

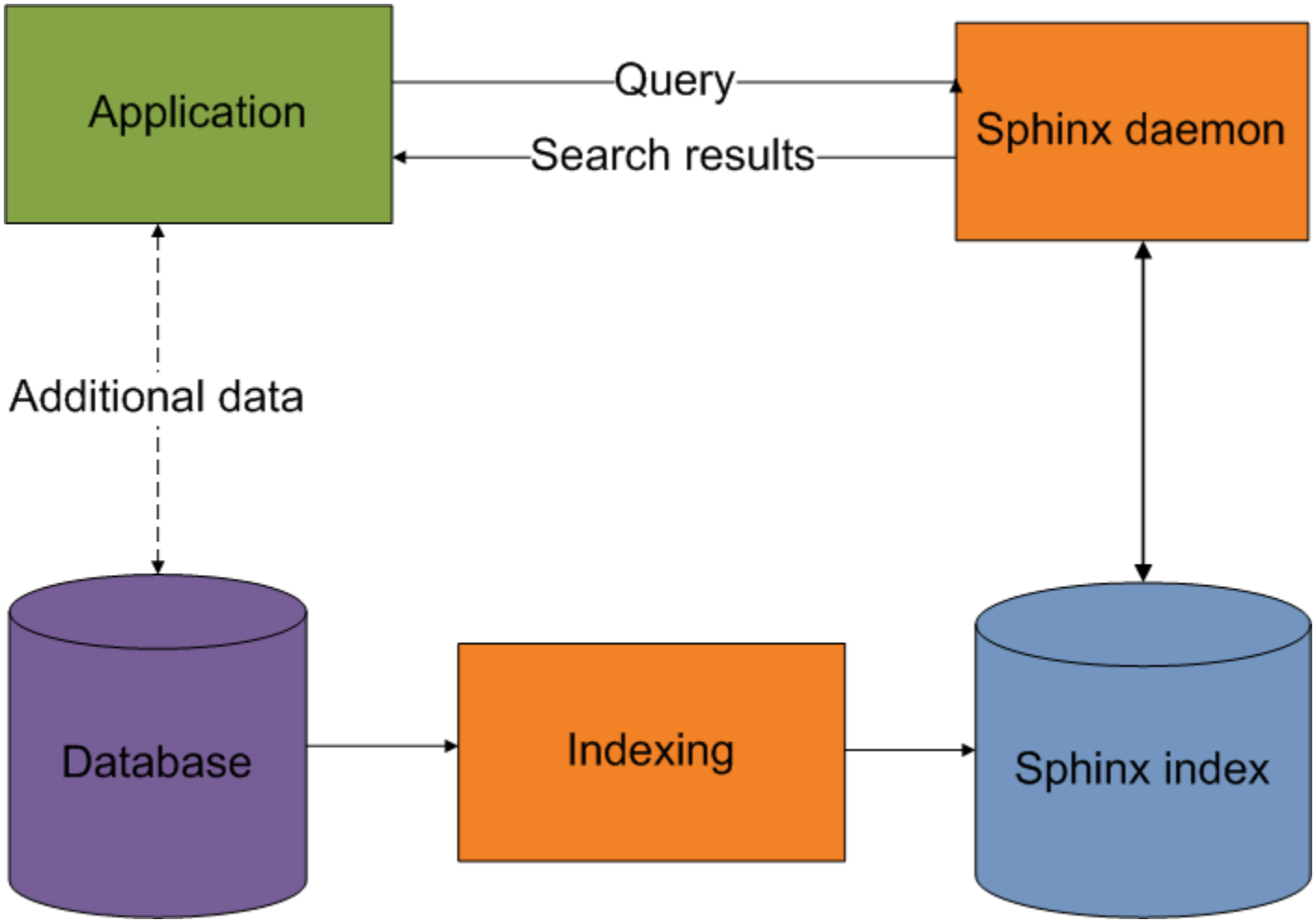
Excerpts (snippets)

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

Installation: How?

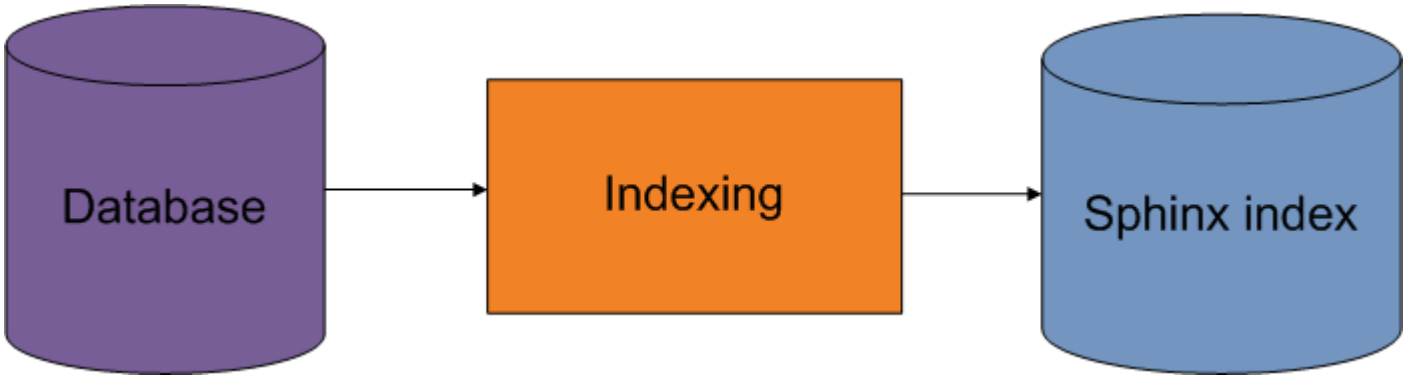
- From binary packages
  - http://sphinxsearch.com/downloads/
- From source
  - http://sphinxsearch.googlecode.com/svn/
  - configure && make && make install
    - Make sure to use --enable-id64
    - for huge document collection
    - already included in pre-compiled packages

Architecture sample



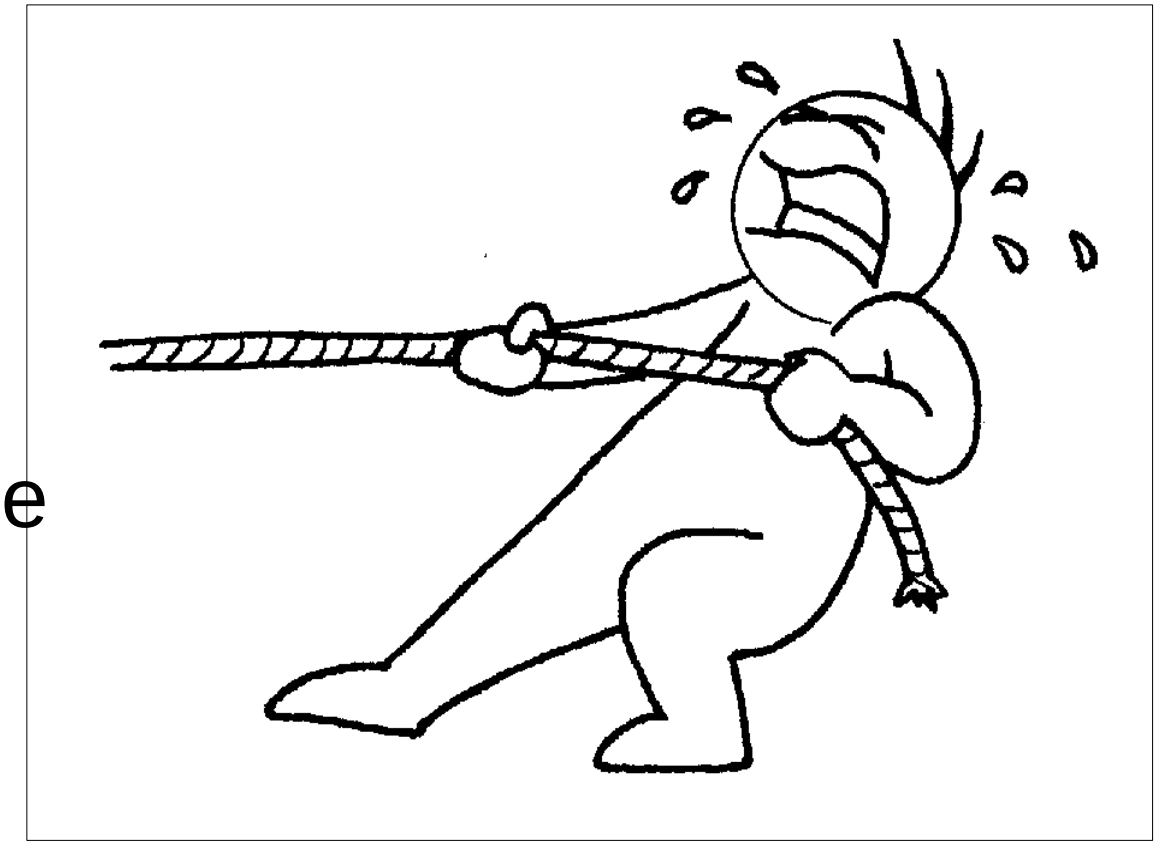
Initial configuration: indexing

- 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
$
```

Next Steps

- Run the daemon
- Connect to daemon
- Run search query

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

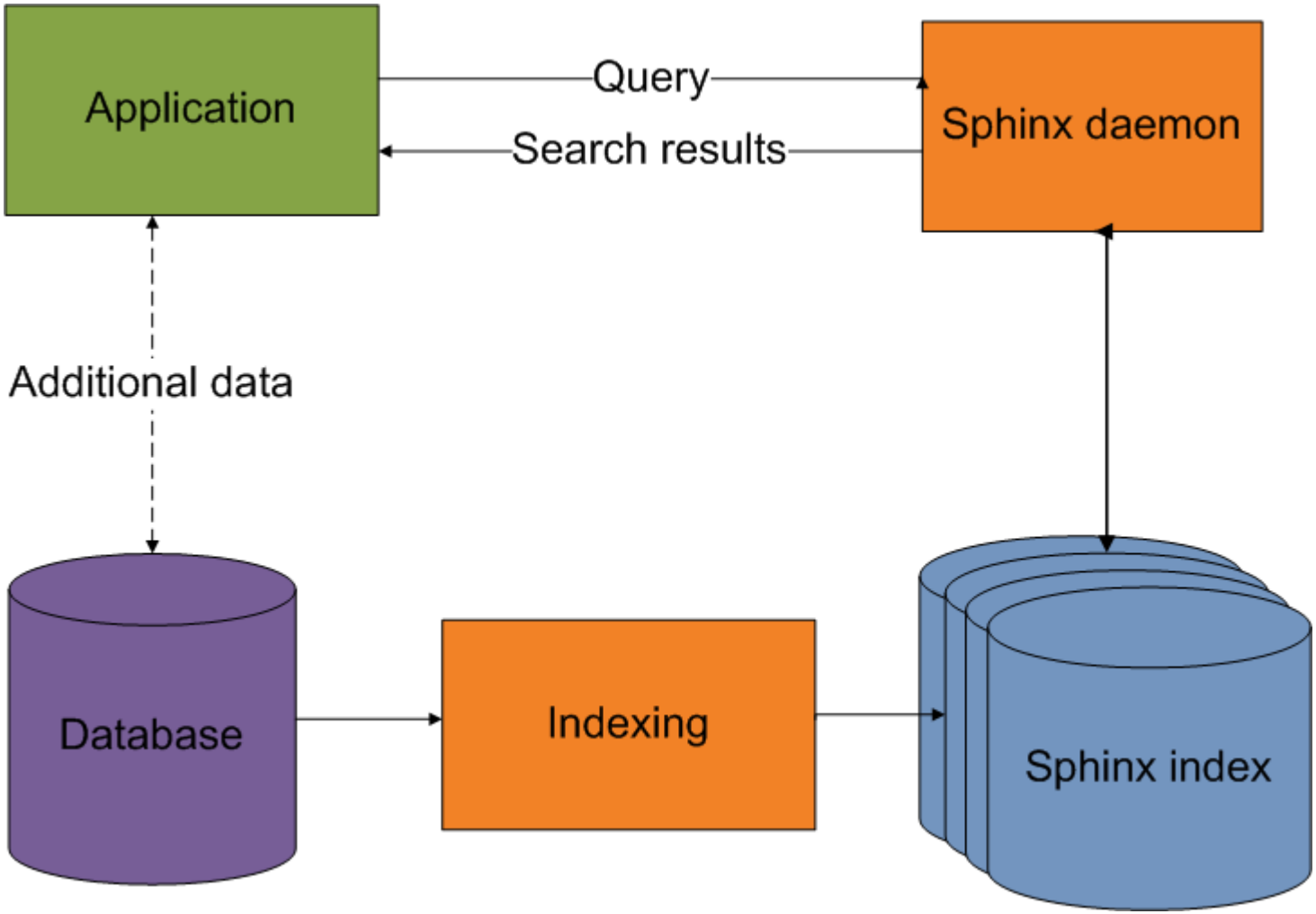
Sphinx applications

- Find relevant documents
  - Items in store(s)
  - Articles in blog/forum/news/etc website(s)
  - Pictures or photos
    - By text, description, GEO-data, publish time, etc
  - Friends
    - In social networks or dating websites
- Offload main database from heavy queries
- Build advanced search and search-based services

Another way to speed up is scaling

- Combine diffrent indexes
  - Main + Delta
  - Ondisk + RT
  - Distributed and local
    - Don't forget about dist_threads!
- Use parallel indexing

OnDisk indexes



Bright side of scaling

- Faster search
- Better load control
- Hardware utilization
- High availability

Dark side

- Hardware faults
- Network issues
- Balancing issues
  - Search time related to slowest search chunk
- Complicated operations

ToDo for attendees

- Please rate this talk!
- Submit your feedback to @vfedorkov
- Visit PalominoDB booth!
- Andrew Aksenoff's talk about new Sphinx features, don't miss it.
  - Tomorrow, 4:30pm - 5:20pm @ Ballroom E
- Questions!

Thank you!

Twitter @vfedorkov
Website: http://palominodb.com