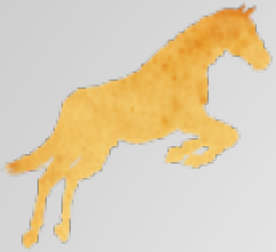
PostgreSQL and Sphinx

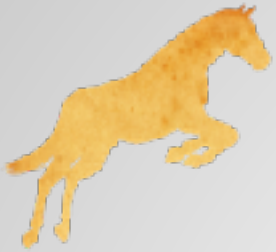
PgCon 2013

Emanuel Calvo



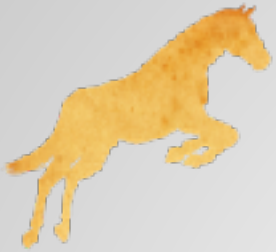
About me:

- Operational DBA at PalominoDB.
  - MySQL, Maria and PostgreSQL databases.
- Spanish Press Contact.
- Check out my LinkedIn Profile at: http://es.linkedin.com/in/ecbcbcb/



Credits

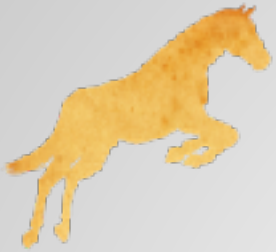
- Thanks to:
  - Andrew Atanasoff
  - Vlad Fedorkov
  - All the PalominoDB people that help out !



Palomino - Service Offerings

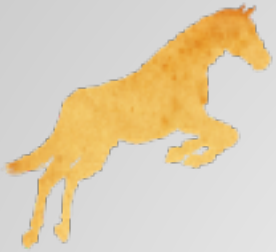
- Monthly Support:
  - Being renamed to Palomino DBA as a service.
  - Eliminating 10 hour monthly clients.
  - Discounts are based on spend per month (0-80, 81-160, 161+
  - We will be penalizing excessive paging financially.
  - Quarterly onsite day from Palomino executive, DBA and PM for clients using 80 hours or more per month.
  - Clients using 80-160 hours get 2 new relic licenses. 160 hours plus get 4.
- Adding annual support contracts:
  - Consultation as needed.
  - Emergency pages allowed.
  - Small bucket of DBA hours (8, 16 or 24)

For more information, please go to: Spreadsheet



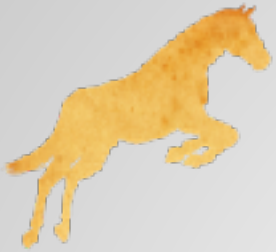
Agenda

- What we are looking for?
- FTS Native Postgres Support
  - http://www.postgresql.org/docs/9.2/static/textsearch.html
- Sphinx
  - http://sphinxsearch.com/



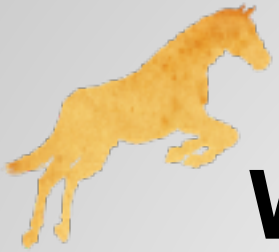
News

- http://sphinxsearch.com/blog/2013/04/01/high-availability-built-in-mirroring/



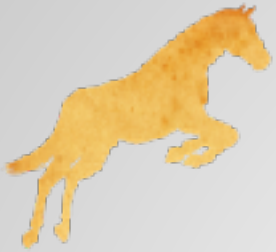
Goals of FTS

- Add complex searches using synonyms, specific operators or spellings.
  - Improving performance sacrificing accuracy.
- Reduce IO and CPU utilization.
  - Text consumes a lot of IO for read and CPU for operations.
- FTS can be handled:
  - Externally
    - using tools like Sphinx or Solr
  - Internally
    - *native FTS support*.
- Order words by relevance
- Language sensitive
- Faster than regular expressions or LIKE operands



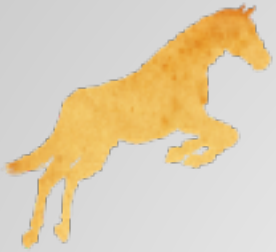
What is the purpose of the talk?

- The only thing we want to show is that there are complementary tools for specific needs on full text searching.
- Native FTS is a great and powerful feature. Also, there is a great work exposed on the last PgEU 2012 about GIN index and FTS improvements.
- Is well known the flexibility and powerness provided by Postgres in terms of SQL. For some projects (specially science and GIS), this combination is critical.
- You can also use both, because are not exclusive each other.



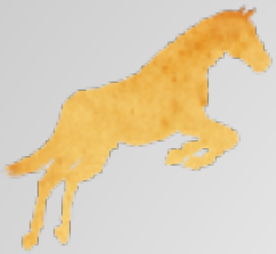
So ... Why Sphinx?

(if we already have native support for FTS)



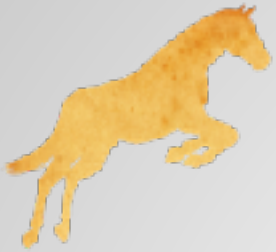
Cases of use

- Generally there is no need to integrate FTS searches in transactions.
- Isolating text searches into boxes outside the main database is optimal if we use it for general purposes.
- If you want to mix FTS searches with other data, you may use the native support.
- If you massively query for text searches you may want to use a text search, easier to scale out and split data.
- Native support is like a RT index. Sphinx allows you to have async indexing, which is faster for writes.

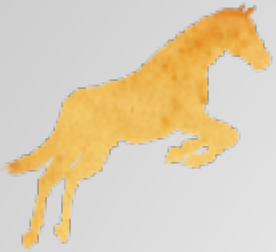


Meh, still want native support!

- Store the text externally, index on the database
  - requires superuser
- Store the whole document on the database, in an object, index on Sphinx/Solr
- Don't index everything
  - Solr /Sphinx are not databases, just index only what you want to search. Smaller indexes are faster and easy to maintain.
  - But you can have big indexes, just be aware of the --enable-id64 option if you compile
- ts_stats
  - can help you out to check your FTS configuration
- You can parse URLS, mails and whatever using ts_debug function for nun intensive operations



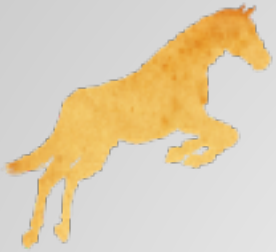
If not, let's Sphinx so.



Sphinx features

- Standalone daemon written on C++
- Highly scalable
  - Known installation consists 50+ Boxes, 20+ Billions of documents
- Extended search for text and non-full-text data
  - Optimized for faceted search
  - Snippets generation based on language settings
- Very fast
  - Keeps attributes in memory
    - See Percona benchmarks for details
- Receiving data from PostgreSQL
  - Dedicated PostgreSQL datasource type.
  - Able to feed incremental indexes.

http://sphinxsearch.com

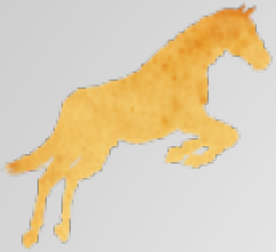


Key features - Sphinx

- Scalability & failover searches
- Extended FT language
  - Faceted search support
  - GEO-search support
- Integration and pluggable architecture
  - Dedicated PostgreSQL source, UDF support
- Morphology & stemming
- Both batch & real-time indexing is available
- Parallel snippets generation

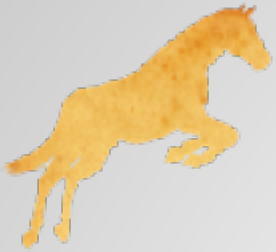
What's new on Sphinx (2.1.1 beta)

- Added AOT (new morphology library, lemmatizer) support
  - Russian only for now; English coming soon; small 10-20% indexing impact; it's all about search quality (much much better "stemming")
- Added JSON support
  - Limited support (limited subset of JSON) for now; JSON sits in a column; you're able to do thing like WHERE jsoncol.key=123 or ORDER BY or GROUP BY
- Added subselect syntax that reorders result sets, SELECT * FROM (SELECT ... ORDER BY cond1 LIMIT X) ORDER BY cond2 LIMIT Y
- Added bigram indexing, and quicker phrase searching with bigrams (bigram_index, bigram_freq_words directives)
  - improves the worst cases for social mining
- added HA support, ha_strategy, agent_mirror directives
- Added a few new geofunctions (POLY2D, GEOPOLY2D, CONTAINS)
- Added GROUP_CONCAT()
- Added OPTIMIZE INDEX rtindex, rt_merge_iops, rt_merge_maxiosize directives
- Added TRUNCATE RTINDEX statement



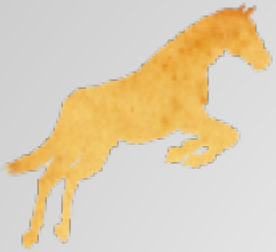
How to feed Sphinx?

- Mysql
- Postgres
- MsSQL
- OBDC
- XML pipe



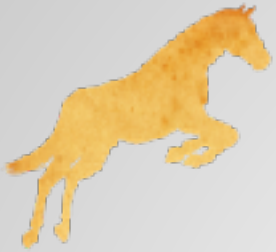
Configuration

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



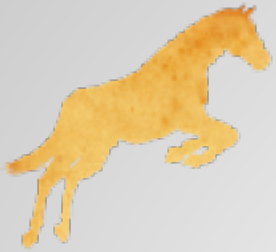
Sections

- Source
  - Where I get the data?
- index
  - How I process the data?
- indexer
  - Resources I'll use to index
  - max_limit, max_iops
- search
  - How I provide the data?



Full text extensions

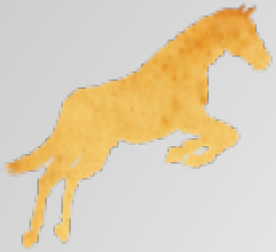
- And, Or
  - hello | world, hello & world
- Not
  - hello -world
- Per-field search
  - @title hello @body world
- Field combination
  - @(title, body) hello world
- Search within first N
  - @body[50] hello
- Phrase search
  - “hello world”
- Per-field weights
- Proximity search
  - “hello world”~10
- 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



Connection

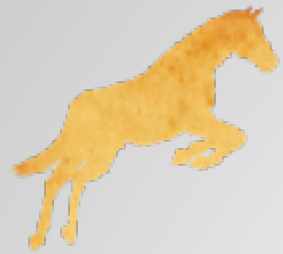
- API to searchd
  - PHP, Ruby, Python, etc
- SphinxSE
  - Storage Engine for MySQL
- SphinxQL
  - Using Mysql client
  - Using client library
  - Require a different port from search daemon

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



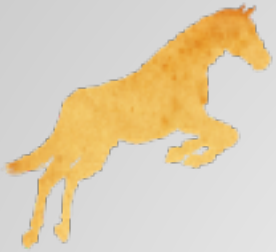
SphinxQL

- 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

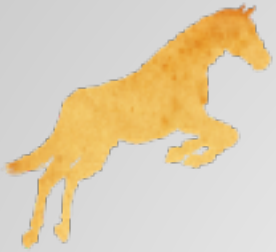
```
mysql> SELECT …, YEAR(ts) as yr
-> FROM sphinx_index
-> WHERE MATCH('I love Sphinx')
-> GROUP BY yr
-> WITHIN GROUP ORDER BY rating DESC
-> ORDER BY yr DESC
-> LIMIT 5
-> OPTION field_weights=(title=100, content=1);
+---------+--------+------------+------------+------+----------+--------+
| id | 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 |
+---------+--------+------------+------------+------+----------+--------+
```



Other features

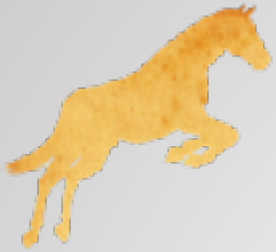
- GEO distance
- Range
  - Price ranges (items, offers)
  - Date range (blog posts and news articles)
  - Ratings, review points
  - INTERVAL(field, x0, x1, …, xN)

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



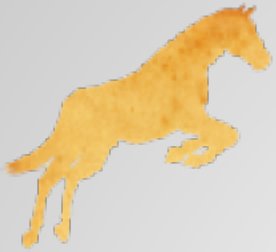
Misspell service

- 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



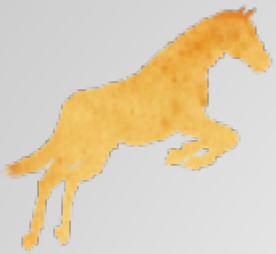
Autocompletion service

- 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



Sphinx - Postgres compilation

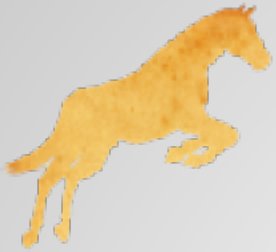
[root@ip-10-55-83-238 ~]# yum install gcc-c++.noarch
[root@ip-10-55-83-238 sphinx-2.0.6-release]# ./configure --prefix=/usr/local/sphinx --without-mysql --with-pgsql --with-pgsql-includes=/usr/local/pgsql/include/ --with-pgsql-libs=/usr/local/pgsql/lib --enable-id64

[root@ip-10-48-139-161 sphinx-2.1.1-beta]# make install -j2

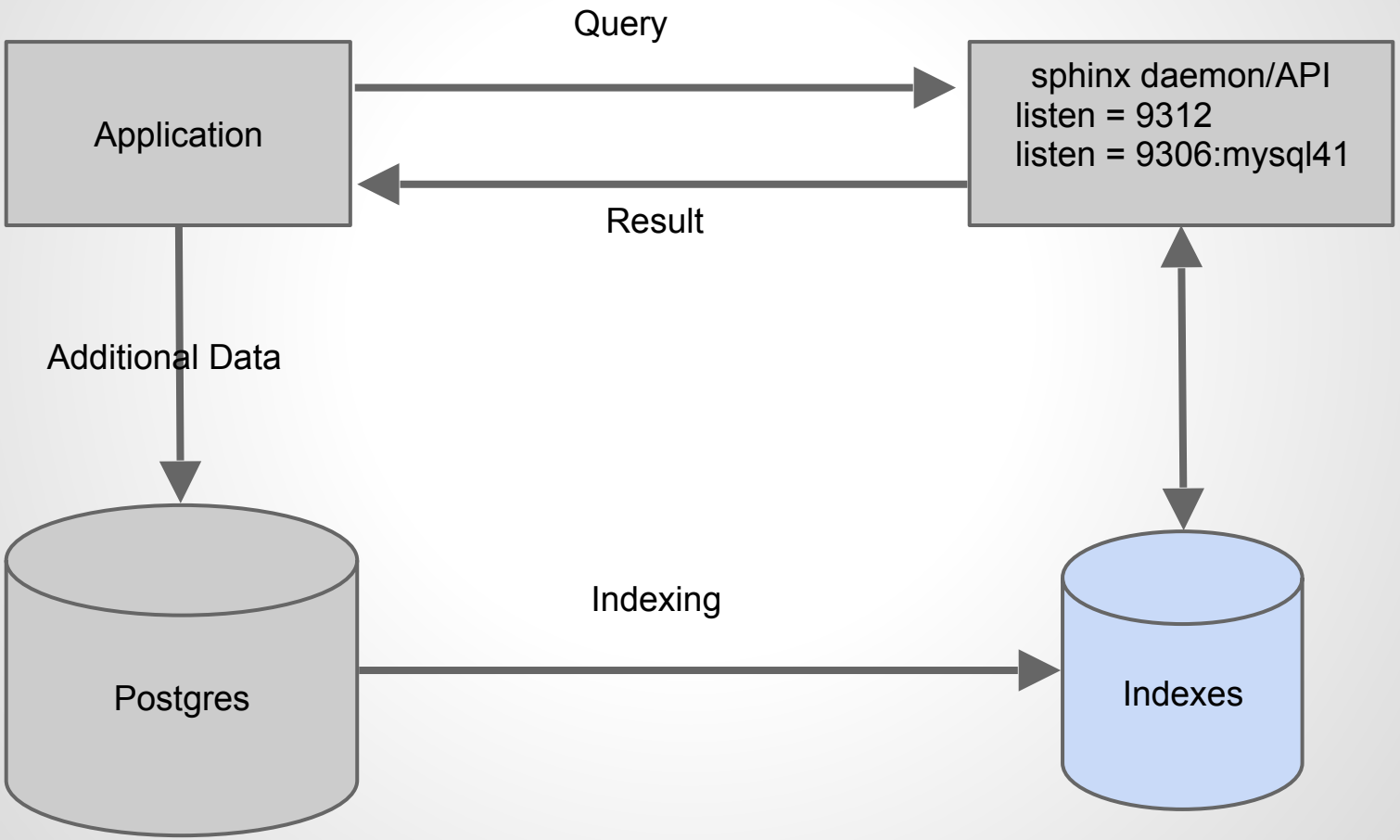
[root@ip-10-48-139-161 sphinx]# cat /etc/ld.so.conf.d/pgsql92-i386.conf
/usr/local/pgsql/lib/
[root@ip-10-48-139-161 sphinx]# ldconfig

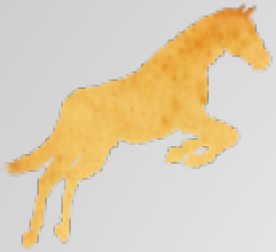
[root@ip-10-55-83-238 sphinx]# /opt/pg/bin/psql -Upostgres -hmaster test < etc/example-pg.sql

NOTE: works for Postgres-XC also



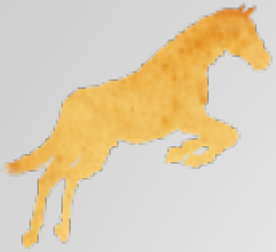
Sphinx - Basic index host





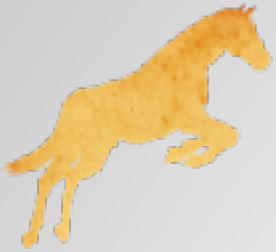
Sphinx - Daemon

- For speed
  - to offload main database
  - to make particular queries faster
    - Actually most of search-related
- For failover
  - It happens to best of us!
- For extended functionality
  - Morphology & stemming
  - Autocomplete, “do you mean” and “Similar items”



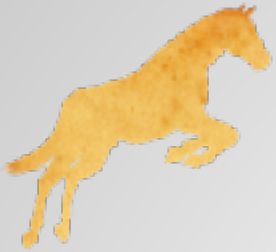
Sphinx - Daemon

```
[root@ip-10-48-139-161 sphinx]# bin/searchd --config etc/postgres.conf
Sphinx 2.1.1-id64-beta (rel21-r3701)
Copyright (c) 2001-2013, Andrew Aksyonoff
Copyright (c) 2008-2013, Sphinx Technologies Inc (http://sphinxsearch.com)


using config file 'etc/postgres.conf'...
listening on all interfaces, port=9312
listening on all interfaces, port=9306
precaching index 'test1'
precaching index 'src1_base'
precaching index 'src1_incr'
 precached 3 indexes in 0.002 sec
```



Sphinx - Indexer

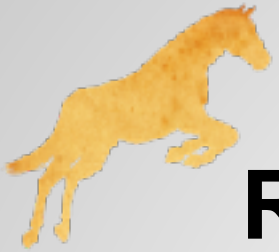
```
[root@ip-10-48-139-161 sphinx]# bin/indexer --all --config etc/postgres.conf
using config file 'etc/postgres.conf'...
indexing index 'test1'...
collected 4 docs, 0.0 MB
sorted 0.0 Mhits, 100.0% done
total 4 docs, 193 bytes
total 0.012 sec, 15186 bytes/sec, 314.73 docs/sec
indexing index 'src1_base'...
collected 4 docs, 0.0 MB
sorted 0.0 Mhits, 100.0% done
total 4 docs, 193 bytes
total 0.010 sec, 17975 bytes/sec, 372.54 docs/sec
indexing index 'src1_incr'...
collected 4 docs, 0.0 MB
sorted 0.0 Mhits, 100.0% done
total 4 docs, 193 bytes
total 0.033 sec, 5708 bytes/sec, 118.31 docs/sec
total 9 reads, 0.000 sec, 0.1 kb/call avg, 0.0 msec/call avg
total 30 writes, 0.000 sec, 0.1 kb/call avg, 0.0 msec/call avg
```



Data source flow (from DBs)

- Connection to the database is established
- Pre-query, is executed to perform any necessary initial setup, such as setting per-connection encoding with MySQL;
- main query is executed and the rows it returns are indexed;
- Post-query is executed to perform any necessary cleanup;
- connection to the database is closed;

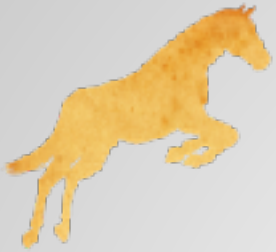
- indexer does the sorting phase (to be pedantic, index-type specific post-processing);
- connection to the database is established again;
- post-index query, is executed to perform any necessary final cleanup;
- connection to the database is closed again.



Range query for the main_query

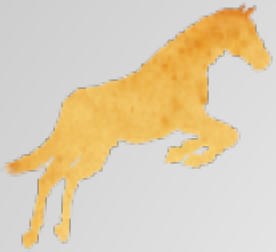
```
sql_query_range = SELECT
    MIN(id),MAX(id) FROM
    documents
sql_range_step = 1000
sql_query = SELECT * FROM
    documents WHERE id>=$start
    AND id<=$end
```

- Avoids huge result set transfer across the network or undesirable seqscans on the database.
- You can avoid min and max agg fxs using a table with those values updated by a trigger or by a async process.



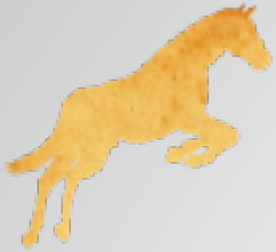
Delta indexing

- Don't update the whole data, just the new records (Main+delta)
  - You need to create your own "checkpoint" table. Update it once your indexing is finished.
- Consider merging big indexes instead

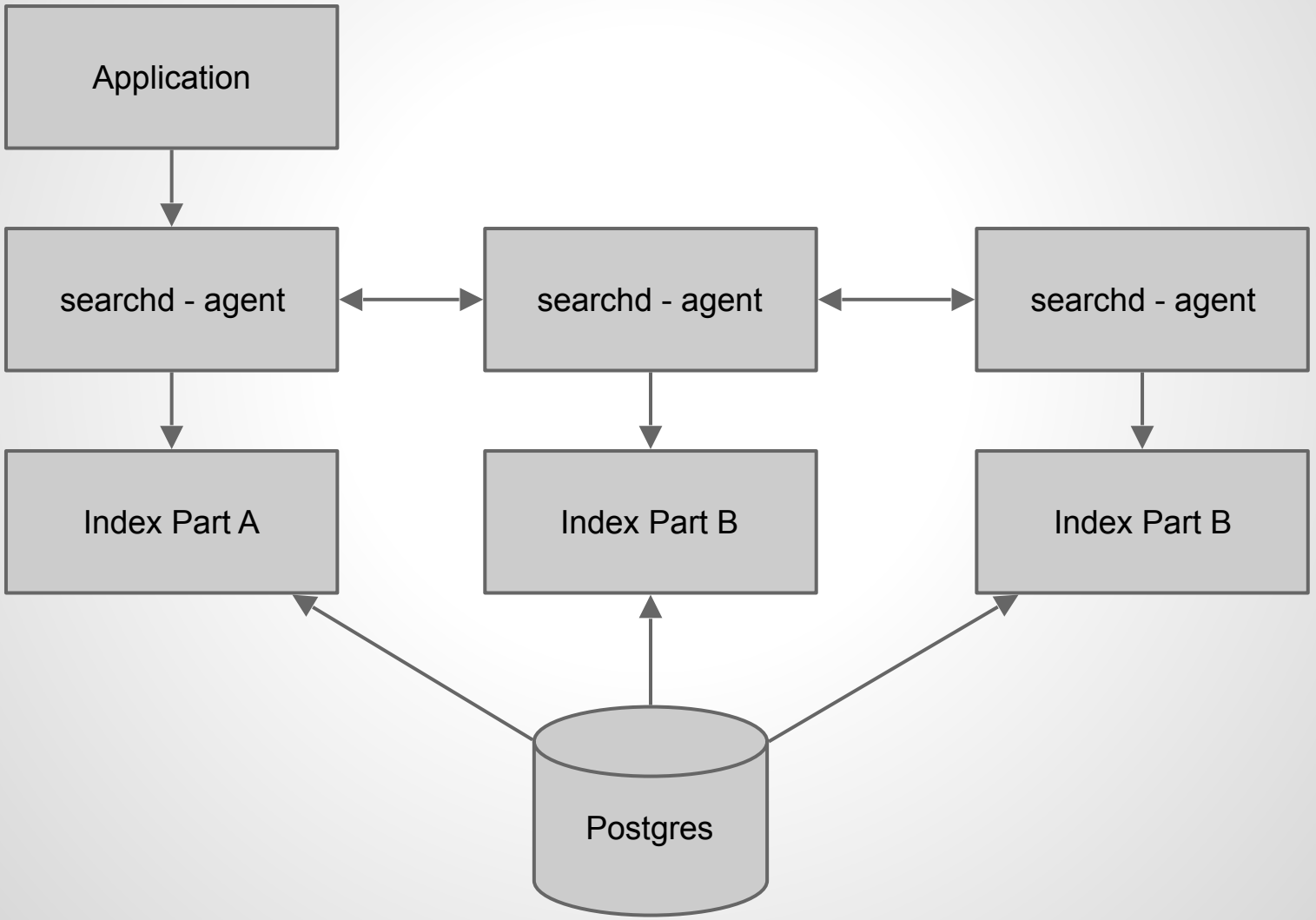


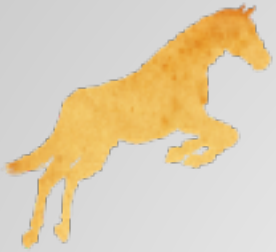
Distributed searches

- The key idea is to horizontally partition (HP) searched data accross search nodes and then process it in parallel.
- The partitioning should be done manually
  - setup several instances of Sphinx programs (indexer and searchd) on different servers;
  - make the instances index (and search) different parts of data;
  - configure a special distributed index on some of the searchd instances;
  - and query this index.
- Queries are managed by agents (pointers to networked indexes)



Distributed searches

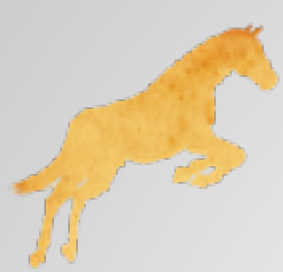




Agent mirrors

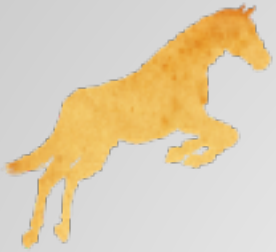
- It allows to do failover searches.

```
# sharding index over 4 servers total
# in just 2 chunks but with 2 failover mirrors for
    each chunk
# box1, box2 carry chunk1 as local
# box3, box4 carry chunk2 as local


# config on box1, box2
agent = box3:9312|box4:9312:chunk2


# config on box3, box4
agent = box1:9312|box2:9312:chunk1
```

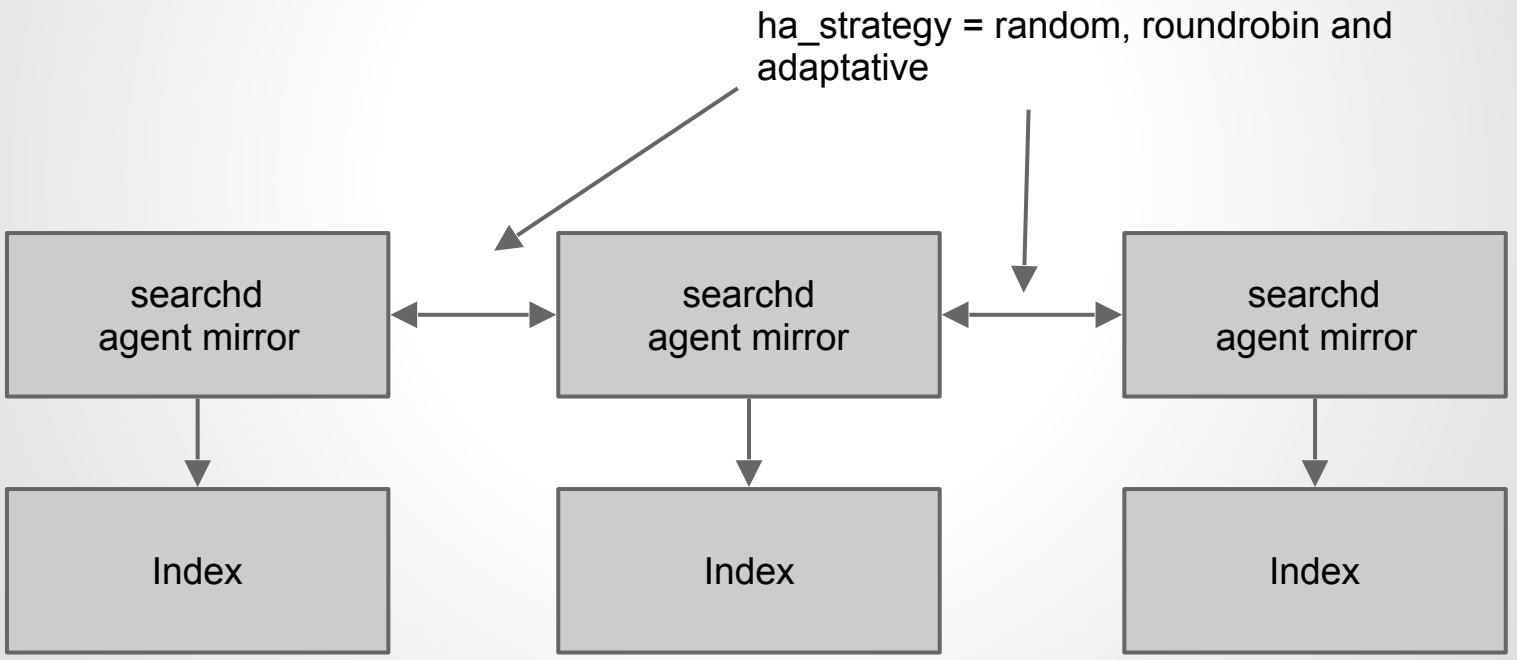


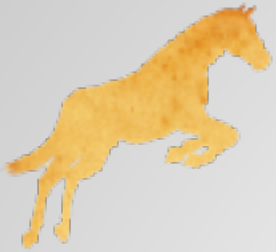
Mirror example

```
index src1_base : test1
{
      source = src1_base
      path = /usr/local/sphinx/data/scr1_base


      type                    = distributed
      local                   = test1


      # remote agent
        # multiple remote agents may be specified
        # syntax for TCP connections is 'hostname:port:index1,[index2[,...]]'
        # syntax for local UNIX connections is '/path/to/socket:index1,[index2[,...]]'
      agent                   = sphinx2:9312:test1
      agent                   = sphinx1:9312:test1


}
```



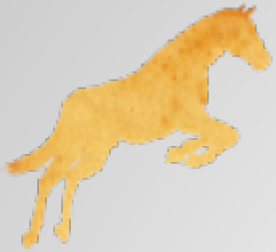
Agent mirrors





Foreign Data Wrapper?

- A data wrapper will allow to retrieve data directly from any Sphinx service, from the database.
- Until today there is no project started, but hopefully soon.



Thanks!

Contact us!
We are hiring *remotely* \o/ !
emanuel@palominodb.com