Топ-секреты Сфинкса

Как жить с полнотекстовым и не очень поисковиком

Владимир Федорков, 2012

Что такое Сфинкс

- Демон ориентированный на быстрый поиск
  - Не библиотека, а именно отдельный демон
    - Как MySQL
- Очень хорошо масштабируется
  - Можно разложить поиск по ядрам
  - Можно разложить поиск по серверам
- Можно обращаться к Сфинксу через MySQL клиент

Самореклама

- Связался высокими нагрузками в 2006-м году и покатился.
- Примерно тогда же заразился сфинксом
- Performance geek
  - Читать твиты: twitter.com/vfedorkov
  - Смотреть фоточки: fb.com/vfedorkov
  - Читать блог http://astellar.com

Чоздесьделаете? (c) Snach

- Думаем как:
  - Сделать быстрее
  - Использовать меньше машин
  - Обслуживать больше пользователей на том же железе

Как будем делать?

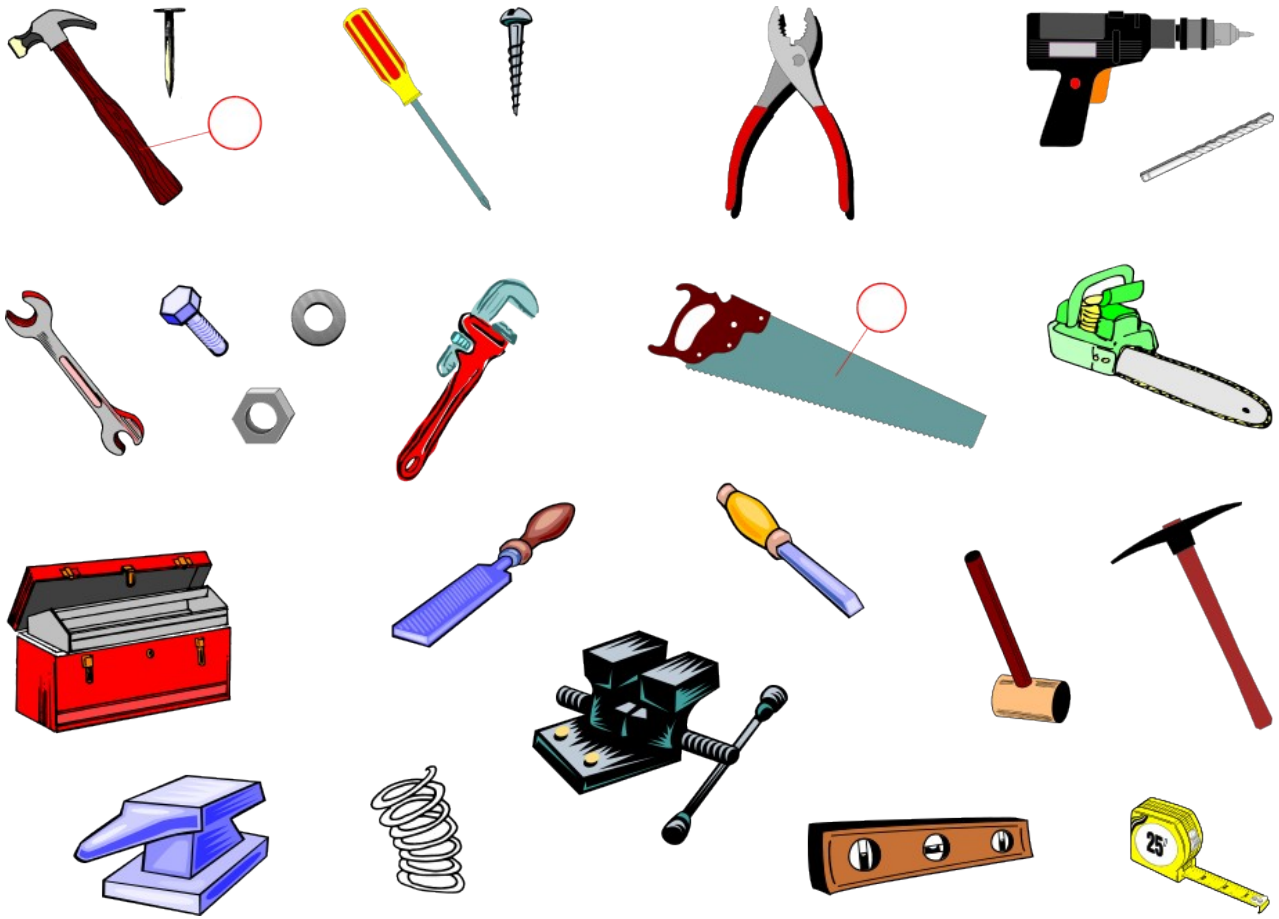
- Думать головой
  - Разбираться в инструментах которые используем
- Тюнить конфиг
  - Помогает в особо запущенных случаях
  - В остальных не помогает или делает хуже
- Тюнить запросы
- Использовать сложные трюки
  - Когда нет простых выходов

Быстро это сколько?

- Гугл всех разбаловал
- «0.1 — 1 — 10» никто не отменял
- Всегда ищем где можно сэкономить
  - Кеширование рулит!
  - Преагрегированные данные рулят!
- Нехорошо, но хотя бы быстро
  - Отдаем частичные результаты
    - Полные можем отдать позже

Что именно тормозит?

- Что угодно из:
  - Apache/Nginx/Lighttpd/Tomcat/Resin/…
  - Perl/PHP/Python/Ruby/Java/.NET/C++/Haskel/…
  - Percona Server/MariaDB/Drizzle/MySQL
  - Memcache/MongoDB/CouchDB…
  - Sphinx/Lucene/Elastic/IndexDen/…
  - Чужие библиотеки и фреймворки
  - Самое обидное — собственный же код
    - Часто потому что микроскопом забивать гвозди получается не очень ловко!



Почему Сфинкс?

- Один из инструментов
  - В грамотных руках умеет творить чудеса
- Два типа индексации
  - Дисковые индексы («ручная» индексация)
  - Real-Time движок
- Работает под Linux, Windows x86/64, Mac OS
  - Собирается на iPhone-ах
  - И даже на DSL модемах

Чем лучше? (минута славы)

- 10-1000x **быстрее** чем MyISAM на полнотекстовых запросах
  - MyISAM приемлем только пока данные помещаются в память
- 2-3x **быстрее** чем MySQL на не-полнотекстовых запросах
  - Группировка и сортировка выполняются в фиксированном объеме памяти
  - Умеет пропускать блоки документов при поиске
- До 10Mb/s индексации на single core.

Немного теории

- Запрос можно послать через:
  - Родные API: PHP, Java, C, Rythion
  - Сторонние API: Rails, .NET
  - SphinxQL
- Возвращает идентификаторы документов
  - WHERE id IN (33, 9, 12, …, 17, 5)
  - На стороне базы обычно срабатывает первичный ключ
    - Удобно для кеширования

SphinxQL

- SQL-подобный язык запросов
- Работает через MySQL библиотеки для почти всех языков
  - Энтузиасты недавно сделали для Delphi
- Похож на MySQL
- Но не идентичен
- Meta fields @weight, @group, @count
  - Полнотекстовые поля не возвращаются
    - Пока

SQL и SphinxQL

- WITHIN GROUP ORDER BY
- OPTION
  - weights, matches and query time control
- SHOW META
- CALL SNIPPETS
- CALL KEYWORDS

Что говорит документация?

- 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)

Про скорость (8M документов)

```
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)
```

Как быстре?

- Перенести часть запросов с БД на Sphinx
  - Все полнотекстовые
    - Остальные надо смотреть
- Уменьшить объем данных
- Выкинуть ненужную функциональность
  - В том числе на стороне сфинкса!

Выкинуть все лишние слова

- «i», «a», «the», «of», etc
  - Безжалостно тормозят и жрут ресурсы
- Сделать файлик со стоп-словами
  - *Indexer <index> —buildstops <output.txt> <N>*
    - *Нужен on-disk индекс*
  - http://astellar.com/downloads/stopwords.txt
- Выкидывает «взрослые слова» с гарантией

Уменьшаем max_matches

- Все документы все равно будут просмотрены.
  - cutoff ограничивает к-во документов
  - max_query_time — ограничивает время запроса
- Не льсти себе.
  - даже гугл показывает только 1000 результатов!

Включаем «легкие» ранкеры

- SPH_RANK_NONE
  - Самый быстрый. Boolean search.
- SPH_RANK_WORDCOUNT
  - Мега-ранкер, умеет считать слова
- SPH_RANK_PROXIMITY
  - Медленее, но понимает дистанцию

Еще ранкеры

- SPH_RANK_SPH04
  - Медленный, но хороший
- SPH_RANK_EXPR
  - Если существующие ранкеры не подходят

Чудесный мир RANK_EXPR

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

FT быстрее атрибутов

- 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

Все еще быстрее

- Phrase search
  - “hello world”
- Proximity search
  - “hello world”~10
- Distance support
  - hello NEAR/10 world
- Quorum matching (/1 = OR)
  - "the world is a wonderful place"/3

Просто круто

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

Ускоряем поиск по атрибутам

- Превращаем автора в ключевое слово!
  - __META_AUTHOR_ID_3235
  - __META_AUTHOR_NAME_Kelby
- Работает с sql_joinned_field
- Не поддерживает > < ← это операторы, а не мимими смайлик!

Ускоряем поиск по каталогу

- __ARTIST_A, __ARTIST_B, __ARTIST_C, …
  - Добавляем ключевые слова в отдельное поле
- Эмулируем статические диапазоны ключевыми словами
  - __MY_RANGE_0, __MY_RANGE_1, …
  - Неудобно, зато быстро!

Подробнее про sql_joined_field

- Сохраняет данные из таблицы в индекс
  - Отдельным запросом
- Эмулирует GROUP_CONCAT в БД где его нет (напримр MSSQL)
- Заменяет JOIN при индексации
  - Для каждого поля будет свой запрос

Геопоиск

- 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
```

Сегменты

- Удобно для группировки результатов
  - По диапазонам цен
  - По датам (новости, статьи в блоги)
  - По рейтингам
- Использование:
  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;
```

MVA

- MVA stands for Mlti Value Attributes
  - 32/64bit integers
  - Поддерживается RT и ondisk индексами с 2.x
- Удобно для хранения категорий, тегов, сопустствующих товаров
  - Упрощает фильтрацию
    - Ключевые слова все равно быстрее

Уменьшаем объем БД

- sql_file_field
  - Читает данные прямо из файл-системы
  - sql_file_field = path_to_text_file
  - Не забывайте про max_file_field_buffer

Мультизапросы

- Ускоряет запросы с общими полнотекстовыми частями
- Использование для фасетов - обязательно
- Работает через API вызов AddQuery(…)
  - Доступно в SphinxQL
    - mysqli точно поддерживает

Real Time engine

- Запросы через работают одинакого
  - Данные нужно добавлять через INSERT
- Держит данные в памяти
  - Когда память кончается пишет на диск
- Пишет бинарные логи, устойчив к крешам
- Отдельно описывается в sphinx.conf

RT config

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

RT — Память

- Выделение памяти контролируется с помощью rt_mem_limit
  - Внимание! По умолчанию только 32Mb
- Дисковые чанки
  - Не изменяются
  - Данные из них не удаляются

RT — Пример индекса

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

Что делать?

- Увеличить RT_Mem_Limit
  - Использовать 64bit Sphinx
- Пересоздать индекс с нуля
  - Использовать ATTACH INDEX
- Не писать лишнего!
  - Убрать архивные данные в ondisk индесы

Что еще делать?

- Использовать несколько индексов
  - Ondisk и RT можно комбинировать!
- Сделать несколько ondisk индексов
  - Текущие данные (час/день/неделя)
  - Архивные данные в отдельном(-ых) индексе(-ах)
- Шардим данные, товарищи!
  - 4 индекса будут почти в 4 раза быстрее чем один!
    - Возможно даже на одном сервере!

Масштабирование. Пример.

```
source lj_source
{
    ...
    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 lj_source2 : lj_source
{
   sql_query_range      = SELECT 7765020, 10425075
}
```

Локальные индексы

```
index ondisk_index1
{
    source          = lj_source1
    path            = /path/to/ondisk_index1
    stopwords       = stopwords.txt
    charset_type    = utf-8
}

index ondisk_index2 : ondisk_index1
{
    source          = lj_source2
    path            = /path/to/ondisk_index2
}
```

Распределенный индекс

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

Несколько серверов

```
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
}
```

Как со всем этим жить?

- Логгировать запросы
  - *query_log_format = sphinxql*
- Включить счетчики
  - *./searchd –iostats –cpustats*
- Профайлить приложение

Сохраняем вемя и нервы!

- Не забываем про резервное копирование!
- OnDisk индексы это просто файлы
- RT индексы — тоже, но не совсем
  - FLUSH RTINDEX для сброса данных на диск
- Используем ATTACH INDEX для быстрого восстановления

Crash safety

- Тюним binlog_flush под наши требования
- Задаем rt_flush_period
  - Никто ничего не гарантирует, демон все решает сам :)

Что еще имеет сделать?

- Обновить сфинкс
  - http://sphinxsearch.com/downloads/release/
- Проверять индексы на ошибки
  - ./indextool –check <indexname>
- Смотреть в query log, много думать

Время кончается, но есть выход

- Документация
  - http://sphinxsearch.com/docs/
- Книга «Introduction to search with Sphinx» by Andrew Askyonoff
  - сегодня, но на английском
- Мастеркласс
  - завтра но, на русском
- Пишите на vlad@astellar.com

ВОПРОСЫ?