HighLoad++

Очевидное – невероятное

(Sphinx tips-n-tricks)

Андрей Аксенов, 2008

Кто здесь?

- Слово Sphinx похоже на search
  - Потому что такое же количество букв
- Бесплатный, открытый поисковой движок
  - Специально обучен индексировать БД
  - Специально обучен искать по тексту
  - Еще умеет исполнять SQL-style запросы (не специально, само приползло)

Соло на баяне, на тему…

- Обзор – скучно!
- Документация – скучно!
- Внутренняя архитектура – тоже скучно!
  - См. доклад и видео с PHPconf’08
  - См. доклад с Highload’08 без ++
- Поэтому…

Как забить шуруп микроскопом

UnFAQ

- Как иногда можно ускорить индексацию MySQL
- Как иногда нужно замедлять индексацию
- Как индексировать результат работы MySQL SP
- Как бороться с MyISAM locks
- Как бороться с расходом памяти PgSQL client
- Как правильно обновлять версию в бою
- Как делать всякое со строками, не имея строк
- Как искать точные совпадения формы (а не стема)
- Как искать точные совпадения слова (а не маски)
- Как ранжировать полные совпадения поля повыше
- Как эмулировать regexp для wordforms
- Как искать по индексам с разными схемами
- Как и зачем делать SQL-style запросы
- Как искать связанные (related) документы
- Как делать исправление опечаток (suggestions)
- Как все это не делать

1. Как ускорять индексацию

- Заставляем протокол сжимать данные
  - mysql_connect_flags=32
  - До +20% к общему (!) времени на 100 Mbps линке
  - Может навредить на 1 Gbps линке
- Отключаем query cache
  - sql_query_pre = SET SESSION query_cache_type=off
- Переносим UNCOMPRESS на клиент (0.9.9+)
  - unpack_mysqlcompress = bodyc

2. Как замедлять индексацию

- Тормозим выборки
  - Понаехали тут, DB сервер не резиновый!
  - sql_ranged_throttle=100
- Тормозим indexer IO
  - max_iops=40 # типичный винт успевает ~100
  - max_iosize=1048576 # для эстетов

3. Как индексировать MySQL SP

- Опять магические флажки в протоколе:
  mysql_connect_flags=131074
- См. mysql_com.h
  CLIENT_MULTI_STATEMENT = 65536
  CLIENT_MULTI_RESULTS = 131072
  CLIENT_FOUND_ROWS = 2
- Почему работает? Видимо, такие процедуры…

4. Как бороться с MyISAM locks

- SELECT * FROM table – это удар в солнечное
- Понятное решение

```
sql_query_range = 1000
sql_query = SELECT …
    WHERE id>= $start AND id<=$end
```

- Надо помнить – это **НЕ** число строк!
- При сквозной нумерации бывает

5. Как бороться с PgSQL client

- SELECT * FROM table – это удар в мозг (RAM)
- Причина другая – мощный дизайн клиента
- Норовит вынуть ВЕСЬ result set сразу
- Понятное решение

  sql_query_range = 1000

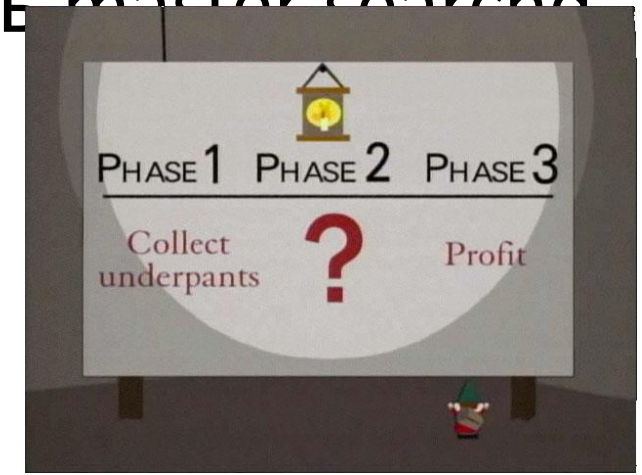
6. Как правильно обновляться

- Sphinx давно и успешно обратно совместим
  - Умеет читать старые конфиги
  - Умеет читать старые индексы
  - Умеет говорить со старыми клиентами
  - API давно не меняется (планируем на 2009й)
- Но даже титановый шарик можно сломать

6. Как правильно обновляться

- Обновить и перезапустить agent searchd(s)
- Обновить и перезапустить master searchd
- Обновить indexer
- Обновить API

  …
- PROFIT!!!



7. Как работать со строками…

- …не имея поддержки строковых атрибутов?
- А что значит работать?
  - Искать точное совпадение (WHERE str='abc')
  - Сортировать (ORDER BY str)
  - Группировать (GROUP BY str)

7. Как работать со строками…

- Все, кроме сортировки, можно делать с CRC
- Коллизии? MD5 + sql_attr_bigint (0.9.9+)
- Сортировать можно по sql_attr_str2ordinal
  - Но сломается при UNION индексов в рантайме
- Сортировать можно по первым N байтам

8. Как искать точную форму

- В случае индексов со стеммингом?
- 0.9.8 – делаем 2 индекса и…

  dog | (_iamexact “dog jump”)
- 0.9.9+ – опция index_exact_words

  dog | “=dog =jump”

  dog | “dog =jump”

9. Как бустить точное слово

- В случае prefix/infix индексов?
- Вариант 1. Магия в запросе
  highload | *highload*
- Вариант 2. Два индекса (но поможет, только если **все** слова совпали)
  $client->SetIndexWeights (…);
- Вариант 3. Дописать спец-фичу

10. Как бустить совпадение поля

- Вариант 1. CRC32 + expr sort
  - $cl->SetSortMode (SPH_SORT_EXPR, **“@weight+IF(fieldcrc=XXX,1,0)”**)
- Вариант 2. Добавить маркеры
  - $cl->Query (**“_begin test query _end”**);
  - Удар по скорости, тк. _begin/_end будут везде
- Вариант 3. Дописать спец-фичу

11. Как эмулировать regex forms

- Sphinx умеет wordforms
- Но вы **НЕ ХОТИТЕ**, чтобы **там** были regexes

11. Как эмулировать regex forms

- Однако иногда вы таки хотите regexes
  - eeeeeeeeek -> eek, hiiiiiiighload -> highload
  - Или разные варианты записи SKU
- Кошерно – предобработка вне Sphinx
  - Можно при индексации, см. xmlpipe2
  - Но лучше при вставке в базу (меньше нагрузка при индексации)

12. Как скрещивать ежа и ужа

- Т.е. одновременно искать по индексам с разными схемами?
- Минимизация схемы результата – вернет атрибуты, которые есть во **всех** индексах
- Фильтры по несуществующим атрибутам – будут **тихо** подавлены

12. Как скрещивать ежа и ужа

- Поиск по несуществующим полям – вернет ошибку, но
- Если запрос начинается с @@relaxed – “невозможные” части будут отброшены
  - @title hello @author vasya
  - **@@relaxed** @title hello @author vasya

13. Как делать SQL-style запросы

- А, главное, зачем?
  - Иногда быстрее, чем база (см. селективность)
  - Иногда удобнее размазать по ядрам/машинам
- Запрос – пустая строка (форсирует full scan)
- Индекс – должен быть с docinfo=extern

13. Как делать SQL-style запросы

- Когда можно, отключайте ранжирование
  - $client->SetRankingMode (SPH_RANK_ NONE);
- Используйте ключевые слова вместо фильтров для высоко-селективных фильтров
  - $client->Query (“_authorid123”);
- Не используйте для низко-селективных!
- Не злоупотребляйте max_matches
- Аккуратнее с группировкой, она намеренно неточная, когда групп много (для скорости)

Disclaimer

- Пункт 13 – это и было “вкратце про тюнинг”

14. Как искать related документы

- Серебряной пули нет, только мелкая дробь
- Можно искать title и использовать кворум
  - “Red Hat chases Redmond with HPC play”/3
  - Порог кворума выбирать ревчутьем
- Можно выбирать “интересные” слова

14. Как искать related документы

- Интересные слова поможет выбирать BuildKeywords() – вернет статистику
- Можно анализировать статистику во времени, отдельными запросами (zeitgeist)
- Можно склеивать синонимы wordforms-ами
  - Redmond > Microsoft
- Sphinx в целом (пока?) не коробочное решение, однако – поможет, чем сможет

15. Как делать suggestions

- Или “когда не хватает aspell”
- Иногда хочется по локальному словарю
  - Что советовать на слово **Camara**?
  - В английском словаре, наверное, **Camera**
  - На авто-сайте, наверное, **Camaro** (Chevrolet)
  - В списке русских городов, наверное, **Samara**
  - В бразильском yellow pages, наверное, **ничего**!

15. Как делать suggestions

- Построить личный частотный словарь

  indexer --buildstops dict.txt 1000000 --buildfreqs

- Затем искать слова в нем
  - Например, сделать словарь для aspell
  - Например, обыскивать биграммы и триграммы
    - Тем же Sphinx?
  - Учитывать частоты, по ним сортировать

16. Как все это **не** делать

- Есть секретный код, привожу PHP вариант

```
while ( !mail ( str_replace ( "(at)", chr(64),
    "support(at)sphinxsearch.com" ), // (*)
    "HALP!!!!11",
    "Are you available for a consulting gig?",
    "From: johndoe@mycompany.com"
    ) );
```

(*) или http://sphinxsearch.com/contact.html

Вопросы?

http://sphinxsearch.com

А теперь – бонус-трек

- Как работает поиск?
- Для каждого локального индекса
  - Строим список кандидатов
  - Фильтруем (аналог WHERE)
  - Ранжируем (считаем веса документов)
  - Сортируем (аналог ORDER BY)
  - Группируем (аналог GROUP BY)
- Склеиваем результаты по всем индексам

Цена булева поиска

- Построение списка кандидатов
  - 1 ключевое слово – 1+ IO (список документов)
  - Булевы операции над списками документов
  - Стоимость пропорциональна (~) длине списков
  - Т.е., сумме частот **всех** ключевых слов
  - При поиске фраз итп, еще и операции над списками позиций слов – более 2x IO/CPU
- Мораль
  - “The Who” – очень плохая музыка

Цена фильтрации

- Дефолтный режим хранения, docinfo=extern
  - Атрибуты хранятся в отдельном файле (.spa)
  - Загружаются в RAM при старте searchd
  - Хэш по docid и затем бинарный поиск
- Фильтры перебираются линейно
- Стоимость ~ числу кандидатов, умноженному на число фильтров

Цена ранжирования

- Прямая – зависит от ранкера
  - SPH_RANK_NONE вообще ничего не стоит
  - SPH_RANK_DEFAULT учитывает позиции слов, дорого
  - Стоимость ~ числу **результатов**
- Косвенная – наводится в сортировке
  - Важно для бенчмарков

Цена сортировки

- Стоимость ~ числу **результатов**
- Еще зависит от критерия сортировки
  - Документы придут в порядке @id asc
  - Поэтому по @id asc очень дешево сортировать!
- Еще зависит от max_matches
  - Чем больше, тем хуже
  - 1-10K нормально, 100K много, 10-20 мало

Так оптимизировать-то как?

- Где можно, ранжируйте попроще
  - Сортировка не по весу? Ранжировать не надо
- Можно вкомпилировать ф-ю сортировки
  - См. src/sphinxcustomsort.inl + @custom
- Можно (редко) оптимизировать сортировку
  - Например, если есть корреляция между @id и timestamp

А еще как?

- Вместо высоко-селективных (“редких”) фильтров – делайте ключевые слова
- Вместо низко-селективных (“частых”) ключевых слов – делайте фильтры
- Benchmark, benchmark, benchmark

Ну а еще как?!

- Мульти-запросы
- Всегда экономит round-trip
- Иногда оптимизируются внутри
- Особо частый случай – когда отличаются только режимы сортировки/группировки
- Это, кстати, как раз т.н. “фасеточный”

Надеюсь, все?

- Конечно
- Конечно, НЕТ
- partitioning, cutoff, max_query_time, block level rejects, index level rejects…
- consulting (да, это самореклама)
- Вот теперь у нас **точно** кончилось время

Вопросы-2.0?

http://sphinxsearch.com