Анализ производительности и оптимизация приложений на MySQL

Петр Зайцев
CEO Percona
29 октября 2013г

О презентации

- Краткий обзор
  - Более детально - на VIP-дне.
- Большая часть сказанного относится к большинству баз данных
  - SQL и NoSQL.

Моя цель

- Чтобы вы знали, **когда** надо оптимизировать базы данных.
- Чтобы вы знали, когда оптимизировать **базы данных**.
- Чтобы вы в принципе знали, **что** в них можно оптимизировать.

Производительность базы данных не важна для пользователя!

- Важна производительность приложения.

О чем мечтает пользователь?

- О том, чтобы приложение **быстро** откликалось на попытки пользователя взаимодействовать с ним.
- **Любого** взаимодействия.
- **Всегда.**

Что значит «быстро»?

- Все зависит от приложения и операции.
- Определяется Бизнесом.
- Ожидания по скорости ответа есть **всегда**.

Не производительностью единой…

- Высокая доступность
- Безопасность
- Легкость и скорость разработки
- Качество
- ….

«Виновата» ли база данных ?

- «Почему вы считаете, что тормозит именно база данных?»
  - «Потому что предыдущие 3 раза тормозила именно база данных».
    - Из почти анекдота

А как же знать точно?

- Инструментация
  - Самописный
  - NewRelic, AppDynamics и т.д.
  - Важно знать, из чего именно формируется время ответа.
- Мониторинг и анализ корреляций
  - когда инструментация не доступна.

Масштаб

- На пустой базе данных для одного пользователя любые запросы «летают».
- Проблемы возникают с **масштабом.**
  - Число пользователей и их активность
  - Объем данных
  - Сложность запросов

Что делать, если база данных не справляется ?

- Не все проблемы базы данных решаются непосредственно на уровне базы данных.

Где решаются проблемы производительности?

- Архитектура приложения
- Правильный выбор технологий
- «Железо»
- Операционная система и ее конфигурация
- Схема базы данных и запросы
- Конфигурация базы данных

Архитектура приложения

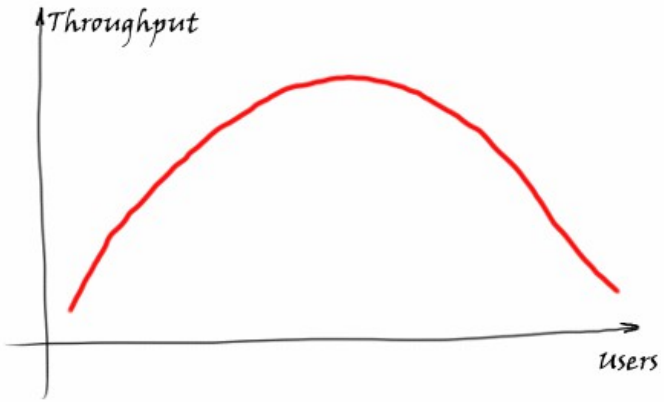
- Наиболее важно!
- Обычно эволюционирует со временем.
  - Меняется масштаб.
  - Меняются технологии.

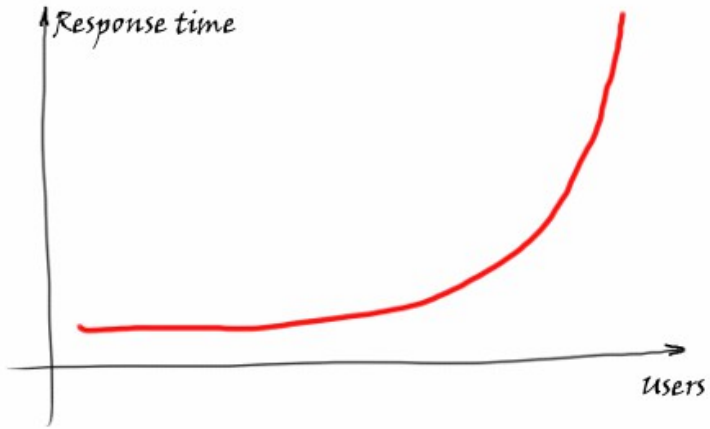
Основные шаблоны архитектуры

- Репликация
- Разделение данных на множество узлов
- Кэширование
- Избыточное хранение
- Пре-генерация данных
- Буферизация

Параллельность

- Для любой системы есть оптимальная параллельность.
- При превышении снижается эффективность.





Ограничение для стабильности

- Часто хорошо ограничить «параллельность» за счет использования очереди.

highload++ 2013

Выбор технологий

- MySQL - всего лишь один из выборов.
- Можно использовать несколько технологий одновременно.
- Примеры:
  - Memcache, Redis, Tarantool
  - Sphinx, Solr, ElasticSearch
  - MongoDB,Cassandra,Couchbase
  - Hadoop

«Железо» может многое

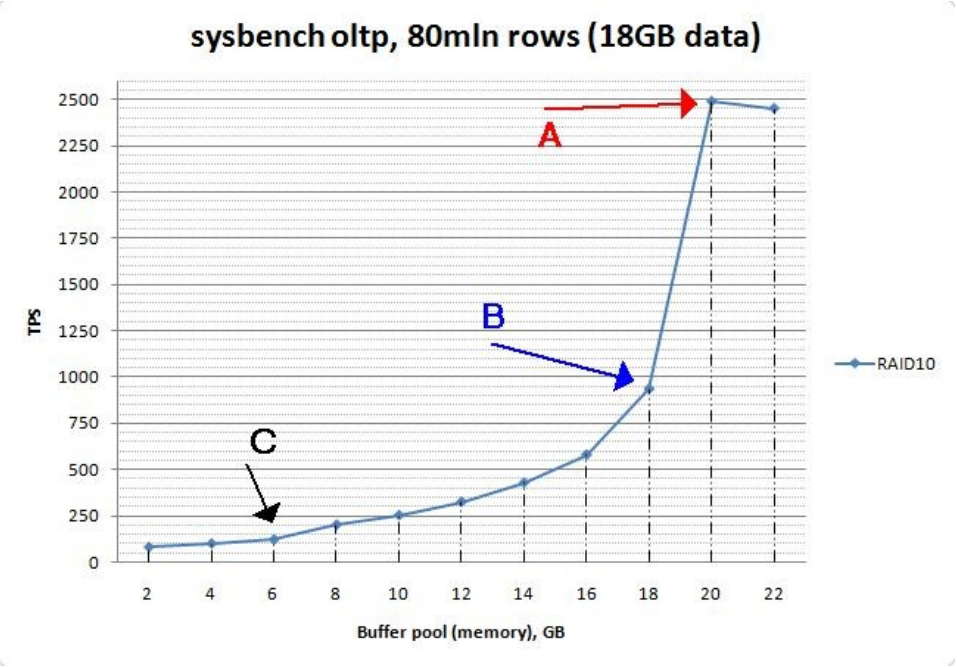
- MySQL показывает более **200 тыс.** простых запросов/сек на современном «железе».
- Может «перелопатить» более **10 млн.** строк/сек.
- Обновить или вставить более **200 тыс.** строк.
- Можем получить 64+ потоков на сервере.
- Более 1 ТБ оперативной памяти.

Основное по выбору «железа»

- Процессоры для MySQL
  - Обычно быстрота ядер важнее их числа.
- Быстрая сеть – очень важно время отклика и стабильность.
- Диски
  - Flash/SSD – отлично
  - RAID с кэшем и батарейкой - хорошо

Память

- Больше памяти – меньше нагрузки на диски.
- Часто используемые данные должны «влезать» в память.

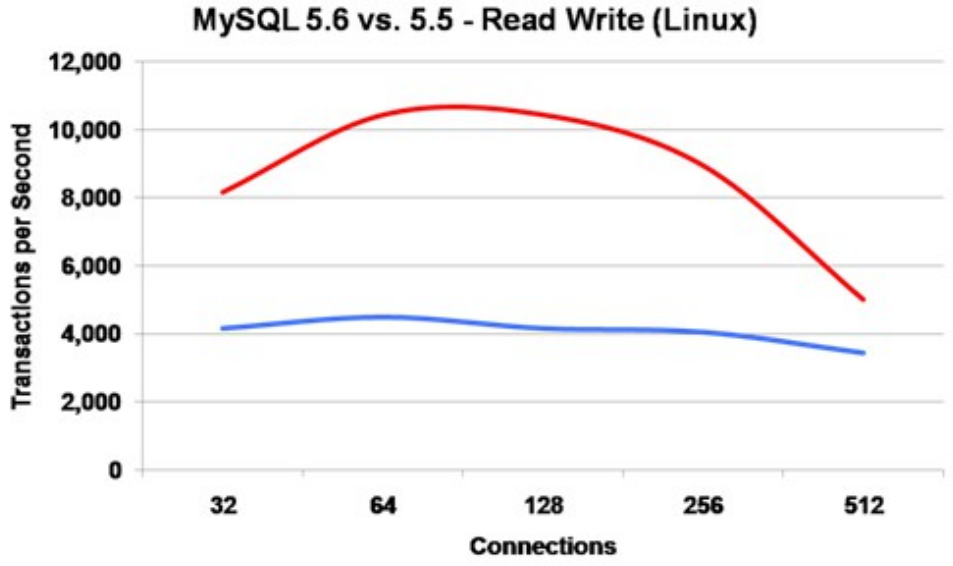


Операционная система

- Linux – наиболее типичный выбор
- Серверный дистрибутив
- Разумно новая версия для нового железа
- Масштабируемая файловая система
  - XFS или EXT4 на Linux
- Более поднобно в Webinar
  - http://bit.ly/1imDk3f

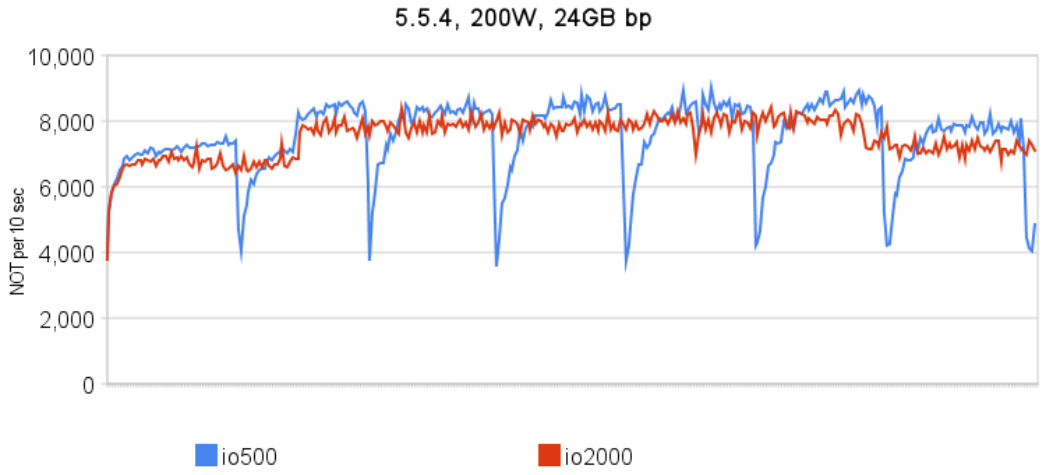
Версии MySQL

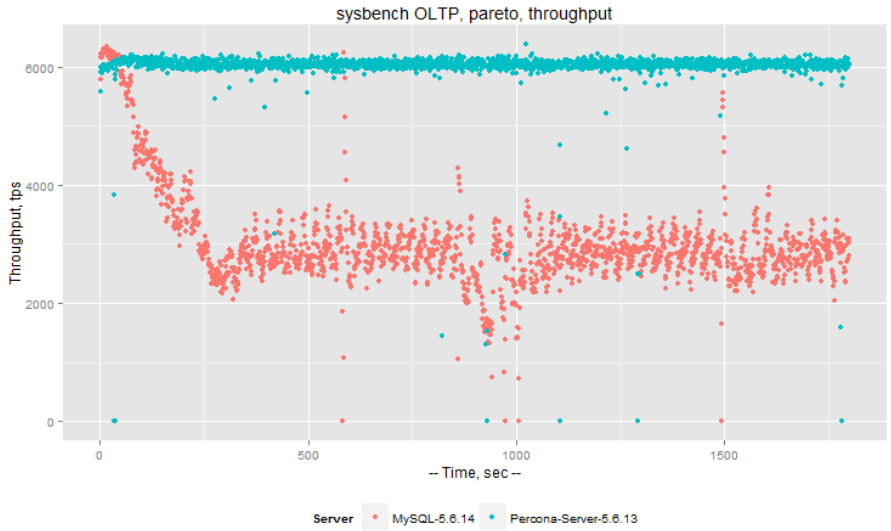
- Каждая новая версия масштабируется все лучше
  - Правда, часто за счет производительности при одном пользователе



Следите за стабильностью

- Важно для реальных систем
- Редко увидите в маркетинговых материалах





highload++ 2013

Конфигурация базы данных

- Конфигурировать MySQL Нужно
  - Умолчания не предназначены для больших серверов
  - Наиболее важно – использование памяти
  - **innodb_buffer_pool_size**
  - **innodb_log_file_size=256M**
  - **innodb_fush_method=O_DIRECT**
  - **innodb_flush_log_at_trx_commit**

Запросы и структура баз данных

- Думайте о проблемах, а не просто о запросе
- На них нужно смотреть **вместе**
- Важно понимать, как база данных исполняет запрос
- Ваш друг в MySQL – **EXPLAIN**
  - http://dev.mysql.com/doc/refman/5.6/en/using-explain.html
- Не все сводится к индексам, но с них полезно

Вместо заключения

- Знайте, какая производительность вам нужна.
- Понимайте, во что «упирается» система.
- Знайте о вариантах оптимизации.
- Оптимизируйте **то, что** разумно, и **когда** это разумно.

Спасибо!

Зайцев Петр

pz@percona.com

@PeterZaitsev

http://www.linkedin.com/in/peterzaitsev

http://www.percona.com