

# Распределенная Архитектура LAMP приложений



Петр Зайцев  
Директор, Percona Ltd.  
[pz@mysqlperformanceblog.com](mailto:pz@mysqlperformanceblog.com)

# Немного о Докладчике

- Persona Ltd – Консалтинг в области производительности MySQL LAMP
- <http://www.mysqlperformanceblog.com>
- MySQL Inc – Консалтинг, Поддержка, Работа с партнерами по вопросам производительности
- SpyLOG.RU – Один из основателей, Тех. Директор в далеком 1999

# Немного о Докладе

- Введение в Архитектуру LAMP приложений
  - Что можно успеть рассказать за 20 минут
- Принципы построения архитектур
- Проблемы роста
- Успешные архитектуры крупных проектов
- Фокус на MySQL

# Два типа Приложений

- Скромные стартапы
  - Один человек с классной идеей
  - Начинаются с одного сервера, простая архитектура
  - Требуется быстрый рост в случае успеха
- Рожденные крупными
  - Yahoo открывает новый проект
  - Строятся для большой нагрузки с первого дня

# Типичные этапы масштабирования MySQL

- Одна Машина
- Репликация
  - Для надежности и производительности
- Распределение по ролям/типам данных
- Горизонтальный партишенинг

# Просто репликация

- Плюсы
  - Просто использовать – пишем на мастер читаем со слейвов
  - Надежность – можно использовать один из слейвов если мастер падает
- Минусы
  - Запись не масштабируется
  - Много слейвов – много копий данных
  - Не эффективное использование кэша

# Оптимизация использования кэша



- Обращение к разным слайдам за разными данными
  - четные пользователи на одном не четные на другом
- Надо учесть что кэшируются страницы
- Средняя эффективность особенно при интенсивной записи

# Разделение Ролей

- Выделение слейвов для полнотекстового поиска
  - Репликация Innodb->MyISAM
- Выделение других тяжелых запросов
- Улучшение использование кэша
- При перегрузке страдают только эти функции системы

# Разделение данных

- Схема разбивается на независимые модули (нет или мало joins между)
- Каждый хранится на отдельном сервере/группе
  - Сессии (если используется база данных)
  - Логгинг
  - Биллинг
- Часто один тип объектов отвечает за большинство нагрузки

# Горизонтальный Партишенинг

- Данные разбиваются на много «кластеров»
- Спец кластер содержит таблицу приписки
- Возможно требуется несколько копий данных с разным партишенингом
- Серьезно усложняет приложение
- Уровень абстракции внутренний или Web Service

# Как организовать «Кластеры»

- Master-Slave
  - Нужно клонировать после падения мастера
- Master-Master
  - Можно легко переключать роли
- Master-N-Slaves
  - Сложнее переключать. Больше копий
  - Не так падает производительность при отказе одного сервера
- Master+DRDB/SAN

# Множество Дата Центров

- Можно использовать Master-Master + несколько слейвов для каждого при 2х центрах
  - Аккуратная политика записи чтобы не было конфликтов
- «Circular Replication» - сложна и не надежна
- «Ручная» репликация для большего числа или спец трюки с репликацией

# Как использовать Слейв ?

- На слейве данные не актуальные
  - Задержка плавает от миллисекунд до минут
- Разные техники использования
  - Запросы не критичные к задержке
  - Сессии без записи могут читать старые данные
  - Если объект давно не обновлялся его состояние можно читать со слейва

# Кэширование Кэширование Кэширование

- Лучшая оптимизация операции – ее исключения
- Лучше всего если веб сервер вообще не получит запрос
- Если получит пусть он будет статическим
- Если динамический то пусть не трогает базу
- Если трогает базу то пусть не требует чтения с диска

# Кэширование

- Правильно сконфигурированный Expire для веб сервера. Иногда Server Side Proху
- Прегенеренный статический контент
  - Можно создавать по запросу и кэшировать
- Кэширование статических длоков страниц
- Кэширование объектов/данных/запросов

# Где кэшировать

- Memcached
  - Распределенный сетевой кэш – легко и дешево наращивать
  - Хранилище сессий
- APC/Eaccelerator/XCache
  - Быстрый доступ в разделяемой памяти
  - Хорош для часто используемых объектов малого объема
- Диск – Долговременное кэширование больших объемов

# Работа с Web Уровнем

- Легко решать проблемы производительности наращиванием серверов
- Можно использовать дешевые сервера так как сбои не критичны
- Часто настроен не оптимально

# Keep-Alive и медленные клиенты

- Использование отдельных серверов для картинок и статики
  - Nginx, lighttpd
- Использование их же как reverse-проxy чтобы не держать apache процесс долго
  - Экономия памяти
- FastCGI
- Если нет Web IO редко нужно более 20 параллельных процессов

# Картинки фильмы ИТД

- «Content Distribution Networks»
  - АКАМАИ, S3 ИТД
- Или Собственные системы хранения
- Отдавать непосредственно с сервера хранения (быстрее чем NFS ИТД)
- Ручное дублирование по серверам
  - Нужен аккуратный протокол
  - MogiloFS - OpenSource от LiveJournal
- Или Глобальные Файловые Системы,  
SAN

# Приглашаем работать с нами



- Вам интересны вопросы производительности ?
- Вы отлично знаете MySQL и Unix/Linux ?
- PHP, Perl, Ruby или Java
- Владеете английским языком
- Самостоятельны в решении задач
- Свяжитесь с нами
  - [jobmysql@mysqlperformanceblog.com](mailto:jobmysql@mysqlperformanceblog.com)