XII международная конференция
CEE-SECR / РАЗРАБОТКА ПО

28 - 29 октября, Москва

2016 CEE-SEC(R)

Сильные стороны MySQL для высоконагруженных проектов

Алексей Копытов

О себе

- разработчик MySQL в MySQL AB/Sun/Oracle
  2004—2010
- разработчик, руководитель проекта в Percona
  2010—2015
- http://github/akopytov/sysbench
  2004—н. в.
- MySQL эксперт, Аурига

О чём доклад?

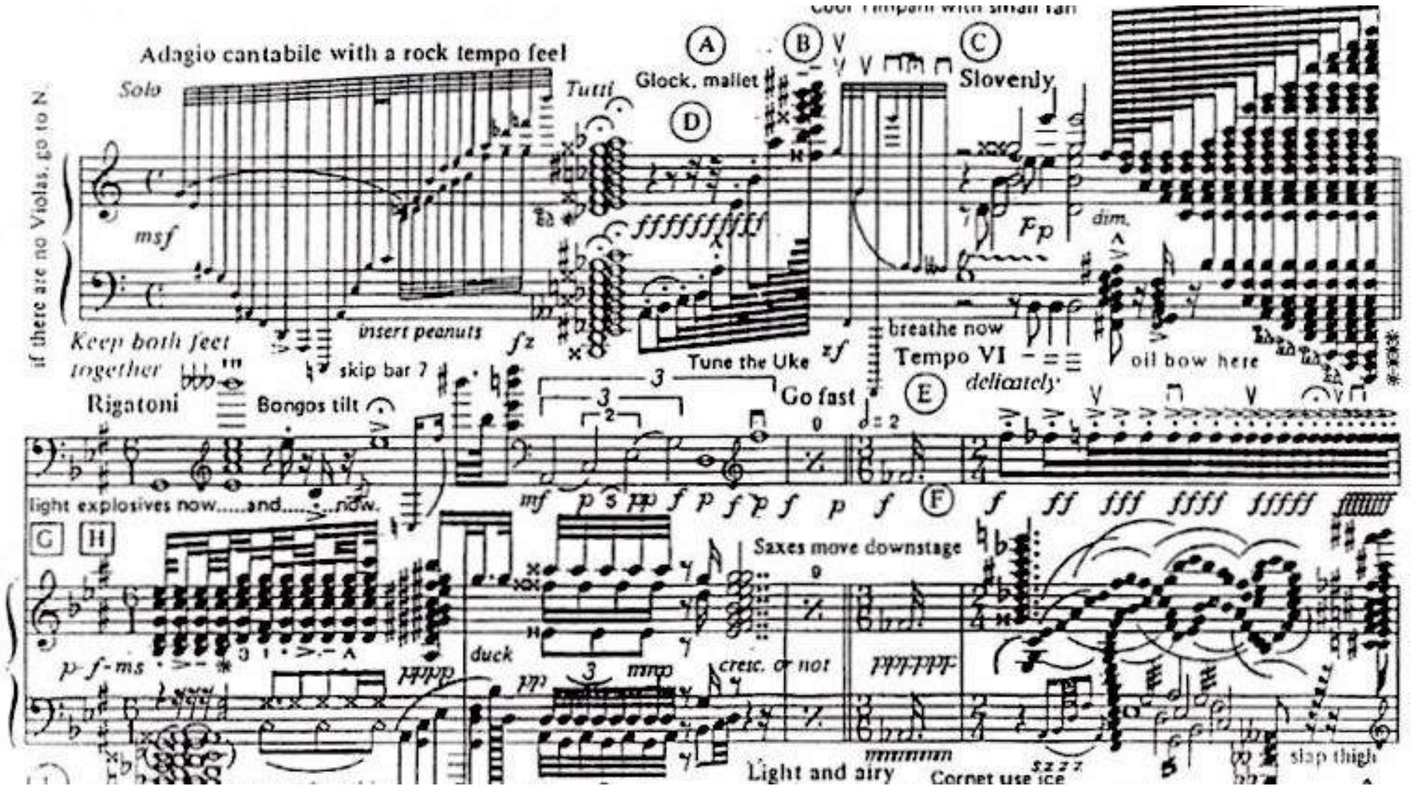
- о сильных сторонах MySQL
- о возможностях MySQL, отсутствующих в PostgreSQL
- почему крупнейшие веб-проекты используют MySQL?

Причины

- обострение дискуссий за последнее время
- много мифов, неквалифицированной критики
- FUD, необъективность лидеров PostgreSQL сообщества:
  - «MySQL — проприетарщина!»
  - «В MySQL нет транзакций!»
  - «Нет причин использовать MySQL!»

Детали

- СУБД — сложные программы
- сравнивать их ещё сложнее
- не утонуть в деталях — главная проблема



Главное

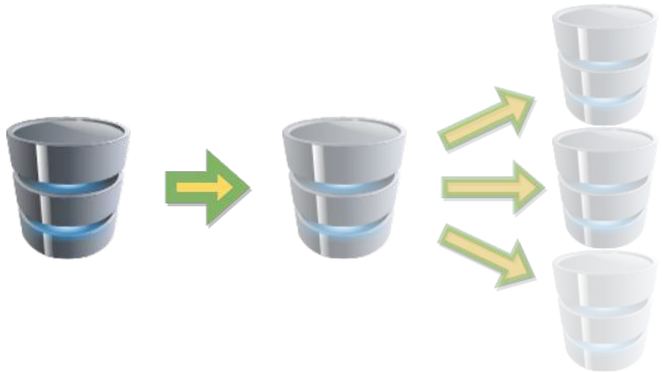
Сильные стороны MySQL:

- репликация и кластерные технологии
- оптимизация чтения, записи и хранения данных
- альтернативные движки
- NoSQL интерфейс

Репликация

Основа любых нагруженных проектов:

- горизонтальное масштабирование
- высокая доступность
- географическая распределённость
- описание изменений: логическая / физическая
- доставка изменений:
  синхронная / полусинхронная / асинхронная



Физическая репликация

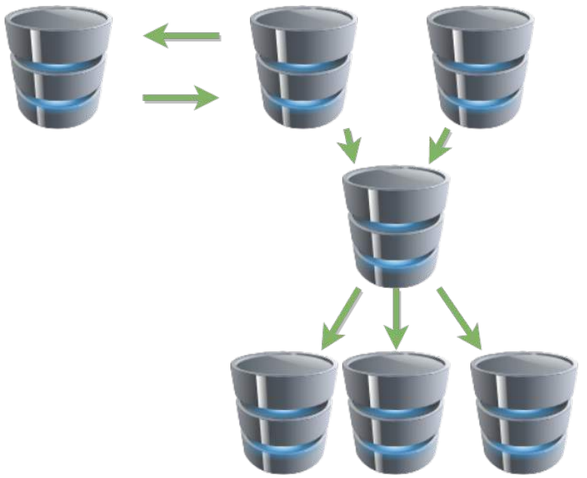
- изменения **всех** файлов данных
- точная копия данных на реплике
- встроена в PostgreSQL с 2010г.
- отсутствует в MySQL

Логическая репликация

- описание изменений на высоком уровне (строки, запросы)
- встроена в MySQL с 2000г.
- много сторонних решений в PostgreSQL, все с недостатками:

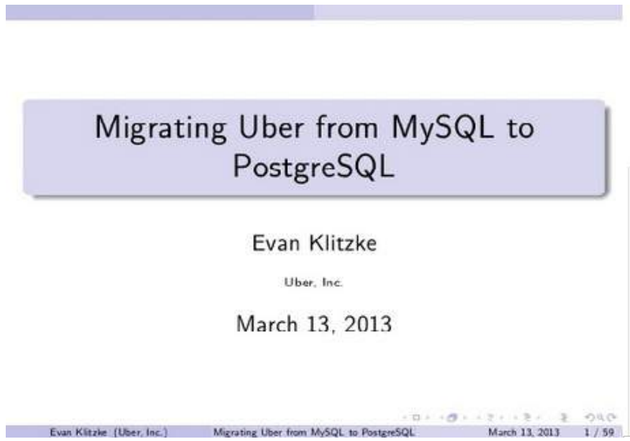
Плюсы логической репликации:

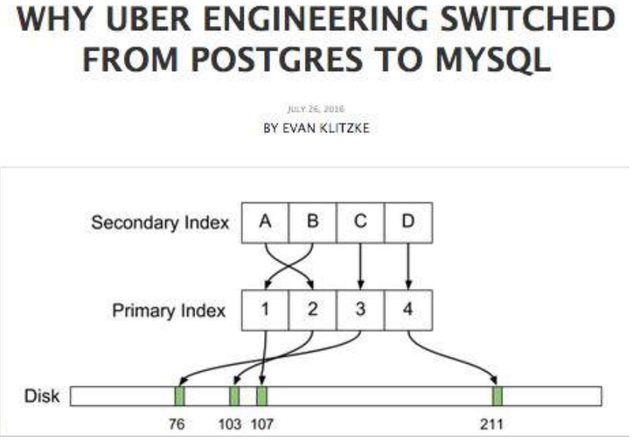
- независимость от физической структуры данных
- позволяет иметь разные схемы на мастере и реплике (при условии обратной совместимости)
- нет ограничений на чтение с реплик
- сложные топологии: каскадная, multi-source, multi-master, и т. д.
- временные таблицы
- частичная репликация
- компактность



Минусы логической репликации:

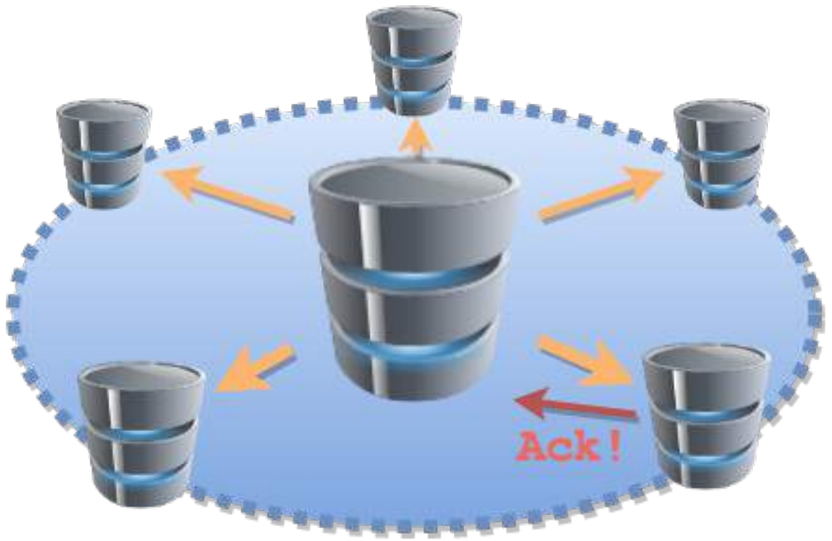
- требует больше ресурсов на реплике
  - популярный в сообществе PostgreSQL факт
  - пока не появился отчёт Uber:
    - неэффективная репликация в PostgreSQL
    - проблемы с MVCC на репликах





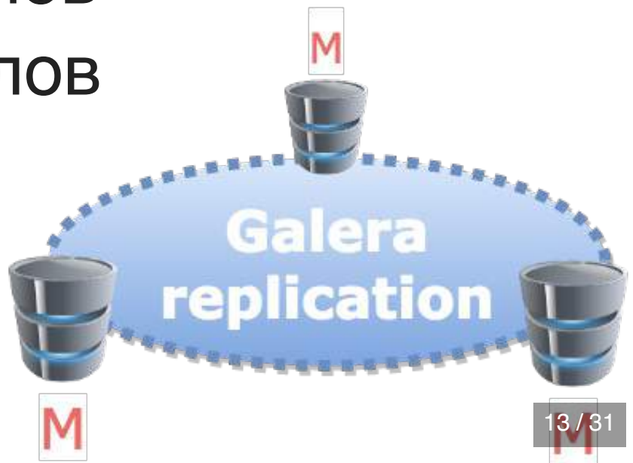
Полусинхронная репликация:

- разработана в Google в 2007г., развивается в Facebook, Alibaba для High Availabiliy кластеров
- коммит на мастере гарантирует получение данных **хотя бы одной** из реплик
- **нет аналога в PostgreSQL**
  (synchronous_standby_names — не аналог)



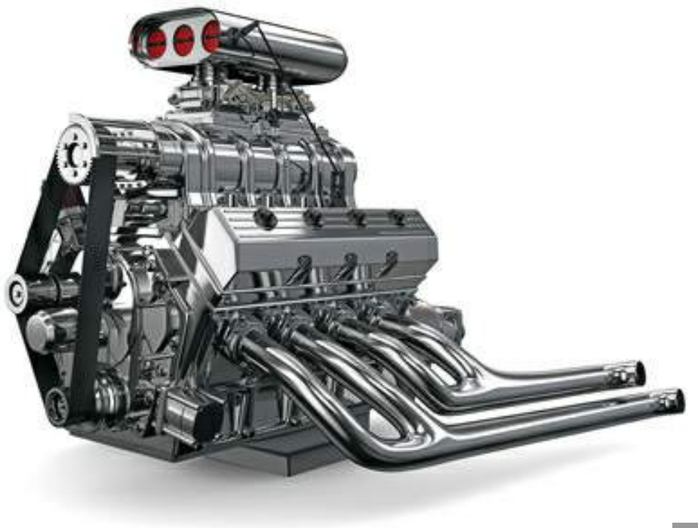
Galera Cluster

- внешняя библиотека от Codership, в разработке с 2007г.
- параллельная синхронная multi-master репликация
- включена в MariaDB, Percona XtraDB Cluster
- масштабирование чтений (всегда локальны)
- нет централизованного управления / единой точки отказа
- автоматическое включение/исключения узлов
- автоматическое создание/пересоздание узлов (node provisioning)
- **нет аналогов в PostgreSQL**



Движки хранения:

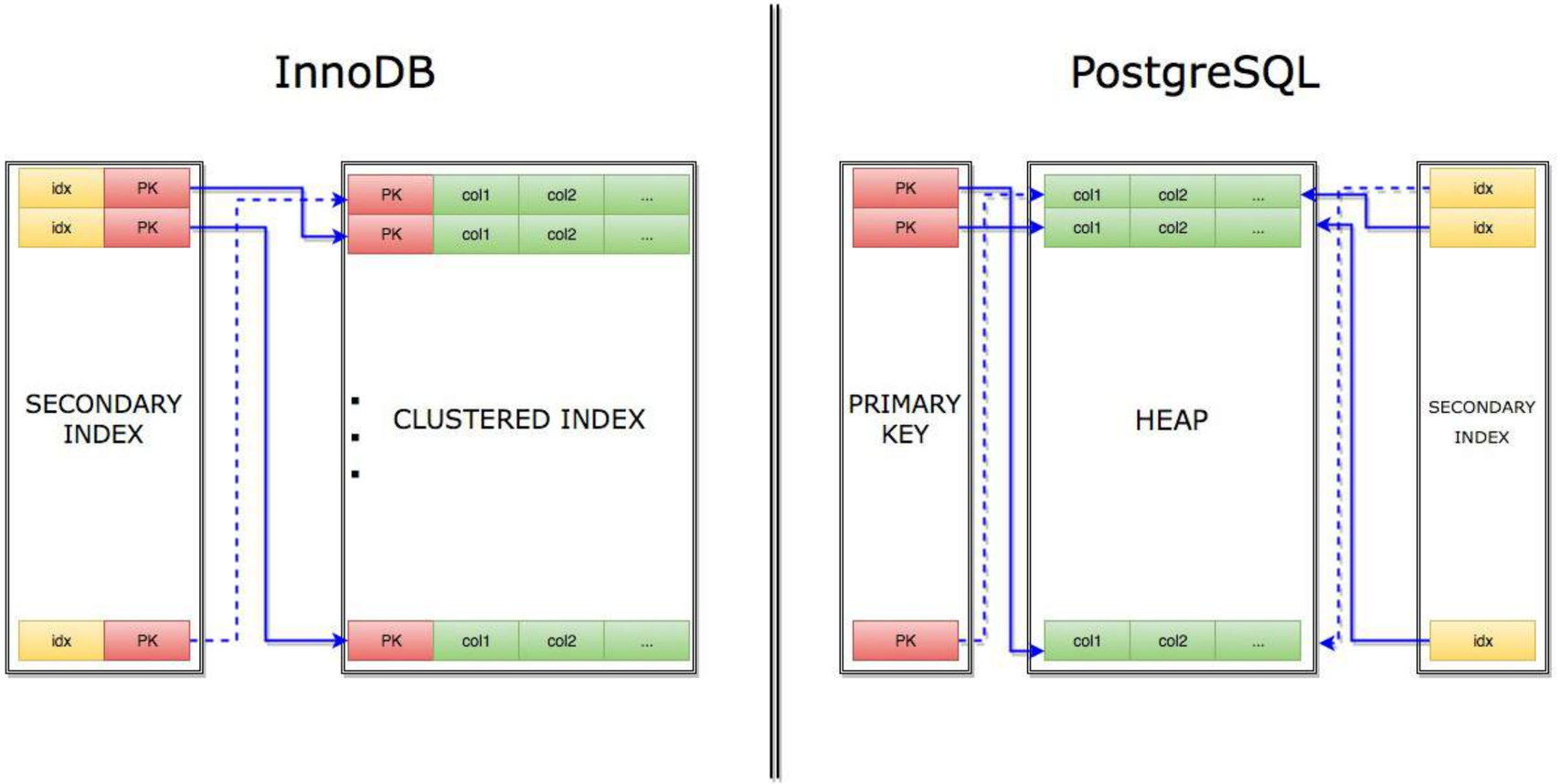
- концепция похожа на VFS в Unix
- абстрагируют физическое представление данных/индексов от ядра выполнения запросов
- могут хранить данные:
  - в оптимизированном для определённых нагрузок виде
  - на диске
  - в памяти
  - на другом СУБД сервере
  - в распределённом кластере



Движки хранения: InnoDB

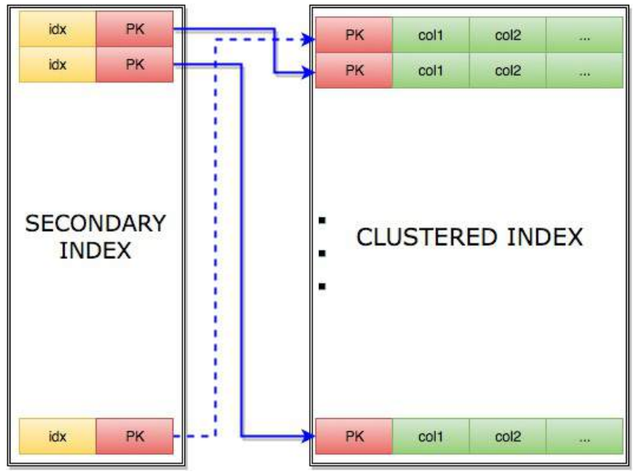
- «рабочая лошадка» современного интернета
- возможно самая обкатанная и оптимизированная реализация B+Tree

InnoDB: кластеризованные индексы



InnoDB: **особенности кластеризованного индекса:**

- запросы по первичному ключу очень быстрые
- скан первичного ключа обычно приводит к последовательному чтению с диска
- первичный ключ является покрывающим индексом для любых запросов
- вторичные индексы являются покрывающими для своих + PK колонок



Кластеризованные индексы в PostgreSQL:

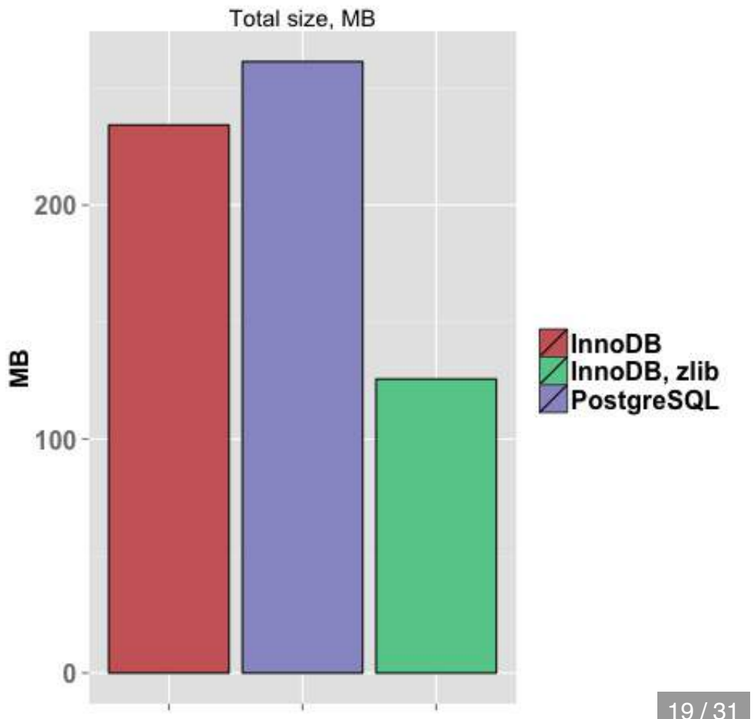
- отсутствуют
- старый и популярный пункт в PostgreSQL TODO

Многие типичные для веб операции:

- медленнее
- приводят к «распуханию» таблиц
- создают проблемы для репликации

Компрессия данных

- MySQL/InnoDB:
  - интенсивно используется интернет-гигантами (Facebook и пр.)
  - постраничная компрессия (данные + индексы)
  - кэшируются результаты компрессии / декомпрессии
- PostgreSQL (TOAST):
  - только для отдельных записей (>2 KB)
  - только для полей переменной длины
  - только для данных
  - нет кэширования / буферизации



InnoDB: поддержка O_DIRECT

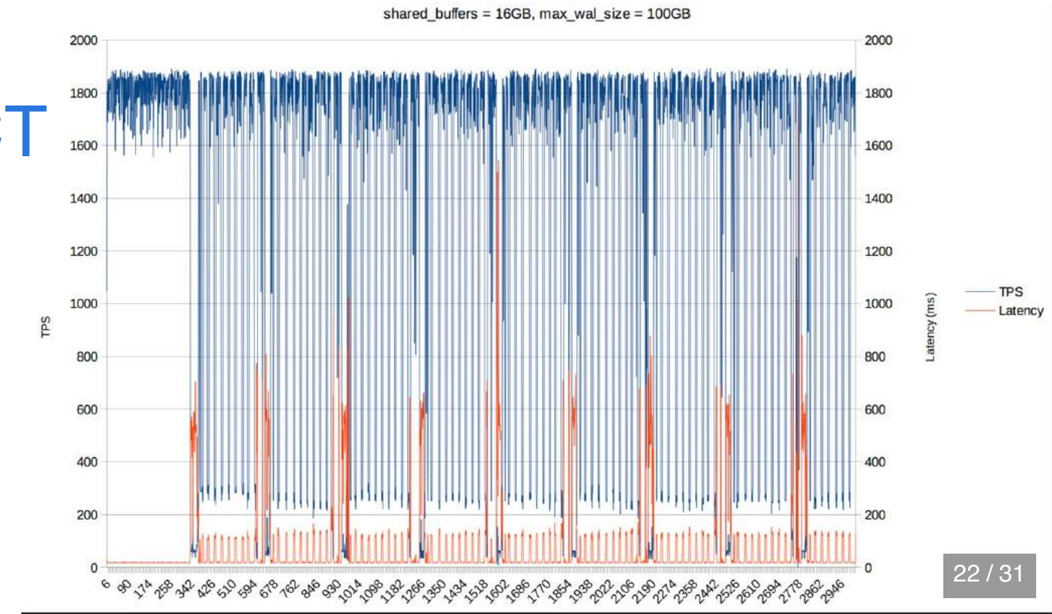
- чтение/запись данных в обход кэша ядра
- более рациональное использование памяти
- нет накладных расходов на двойное кэширование/буферизацию
- более тонкий контроль над записью на диск
- появилась в 2003г.

O_DIRECT в PostgreSQL:

- только для WAL, не для данных
- неэффективное использование памяти (shared_buffers = ~25% RAM)
- двойная буферизация
- излишняя работа для контрольных сумм (в будущем: шифрования, компрессии, и т.д.)

InnoDB: поддержка O_DIRECT

- MySQL:
  - огромная работа в Percona и Oracle по сглаживанию скачков TPS/latency при интенсивной записи:
    - fuzzy checkpointing, adaptive flushing, parallel flushing, parallel doublewrite
- PostgreSQL:
  - сложно сделать без O_DIRECT
  - “requires lots of performance work on our side”
    – Andres Freund



Движки: MyRocks, TokuDB

MyRocks и TokuDB:

- оптимизированы на запись и для SSD устройств
- MyRocks – LSM-деревья, Facebook
- TokuDB – «фрактальные» индексы, Percona
- более компактное представление данных на диске
- продвинутые возможности компрессии
- низкий write amplification по сравнению с InnoDB
- множественные кластеризованные индексы (TokuDB)
- ничего похожего по характеристикам в PostgreSQL

Движки: NDB

- in-memory кластер с опциональным чекпойнтингом на диск
- автоматические шардинг, failover, recovery
- active-active/multi-master репликация

Типичные области применения:

- телекоммуникации (данные абонента)
- платёжные, финансовые системы
- PayPal: гео-распределённый кластер на 100 TB для обнаружения мошенничества (fraud detection)

Физические резервные копии:

- важны при добавлении узлов в кластер
- MySQL:
  - Percona XtraBackup
- PostgreSQL:
  - pg_basebackup
  - barman
  - pg_arman
  - pgBackRest
  - по функциональности – XtraBackup 6-7 лет назад

key/value (NoSQL) API:

- Сэкономить время на:
  - разбор SQL
  - открытие, блокировку таблиц
  - построение плана выполнения
- MySQL:
  - HandlerSocket (сторонний плагин)
  - memcached (встроенное)
  - NDB API (только для NDB)
- PostgreSQL:
  - **нет аналогов**

За кадром:

- потоки и процессы
- встроенный пул соединений (MariaDB / Percona)
- масштабируемость при большом количестве соединений
- поддержка асинхронного ввода/вывода
- компактность данных на диске
- проблема IO amplification
- прозрачное для клиентов шифрование данных
- поддержка сохранения/восстановления состояния кэша

За кадром (2):

- декларативное секционирование данных
- поддержка кодировок
- возможности оптимизатора запросов
- динамические переменные
- виртуальные колонки
- компрессия соединений
- MySQL Embedded: сервер в виде встраиваемой библиотеки
- встроенный планировщик заданий (event scheduler)

Выводы:

PostgreSQL — замечательная СУБД, но:

- есть много причин использовать MySQL
- многие важные для крупных веб-проектов возможности MySQL отсутствуют в PostgreSQL до сих пор
- реализация их может растянуться на годы (если не десятилетия)

Выводы:

- MySQL скорее всего останется
  «самой популярной open source СУБД для веб»
- а PostgreSQL —
  «самой продвинутой open source СУБД»
- выбор СУБД — сложный вопрос
- бегите от людей, которые предлагают простые ответы!
- Uber: “Don’t rely on hearsay. Don’t believe in hype!”

Спасибо!

- Эти слайды: http://kaamos.me/talks/secr2016
- Отчёт Uber о переходе на MySQL: https://eng.uber.com/mysql-migration
- TokuDB: http://bit.ly/2dx7aYt
- MyRocks: http://bit.ly/2cRgS8y
- Galera Cluster for MySQL: http://galeracluster.com/
- MySQL Group Replication: http://bit.ly/2dx6UwZ
- MySQL NDB Cluster: https://www.mysql.com/products/cluster/