Федоров Андрей
Oracle

PGDAY'17 RUSSIA
КОНФЕРЕНЦИЯ ПО БАЗАМ ДАННЫХ

Что нового в MySQL 8.0?

Санкт-Петербург, 2017

ORACLE

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

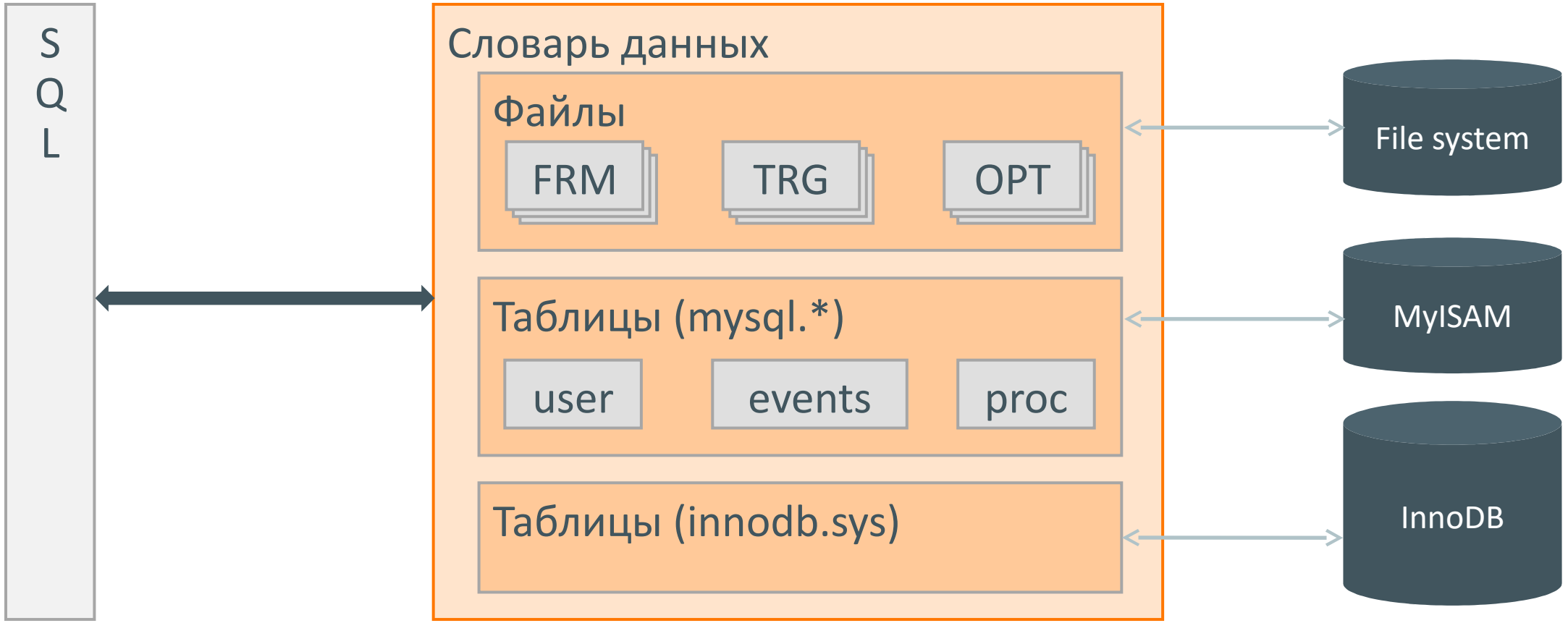
Актуальные версии MySQL

- **5.7 - стабильная ветка**
  - **Свежая версия - 5.7.18 (2017-04-10)**
  - **Доступны на dev.mysql.com/downloads/mysql/**
- **8.0 – активная разработка**
  - **Development Milestone Release - 8.0.1 (2017-04-10)**
  - **Доступны там же**
- **MySQL Labs версии**
  - **8.0.1 + unified data-dictionary (2017-04-21)**
  - **Доступны на labs.mysql.com/**
- **Rapid плагины**
  - **X (с 5.7.12), Group Replication (с 5.7.17)**
  - **Распространяются вместе со стабильными или DMR версиями**

MySQL 8.0 с высоты птичьего полета

- Новый словарь данных и поддержка атомарного DDL
- Табличные выражения (CTE), оконные функции
- Убывающие и невидимые индексы
- Улучшения в поддержке Unicode
- Гибкая работа с блокировками (SKIP LOCKED/NO WAIT)
- Роли, улучшения в системе привилегий
- Улучшения в репликации

Словарь данных до 8.0: Метаданные хранятся в разных местах



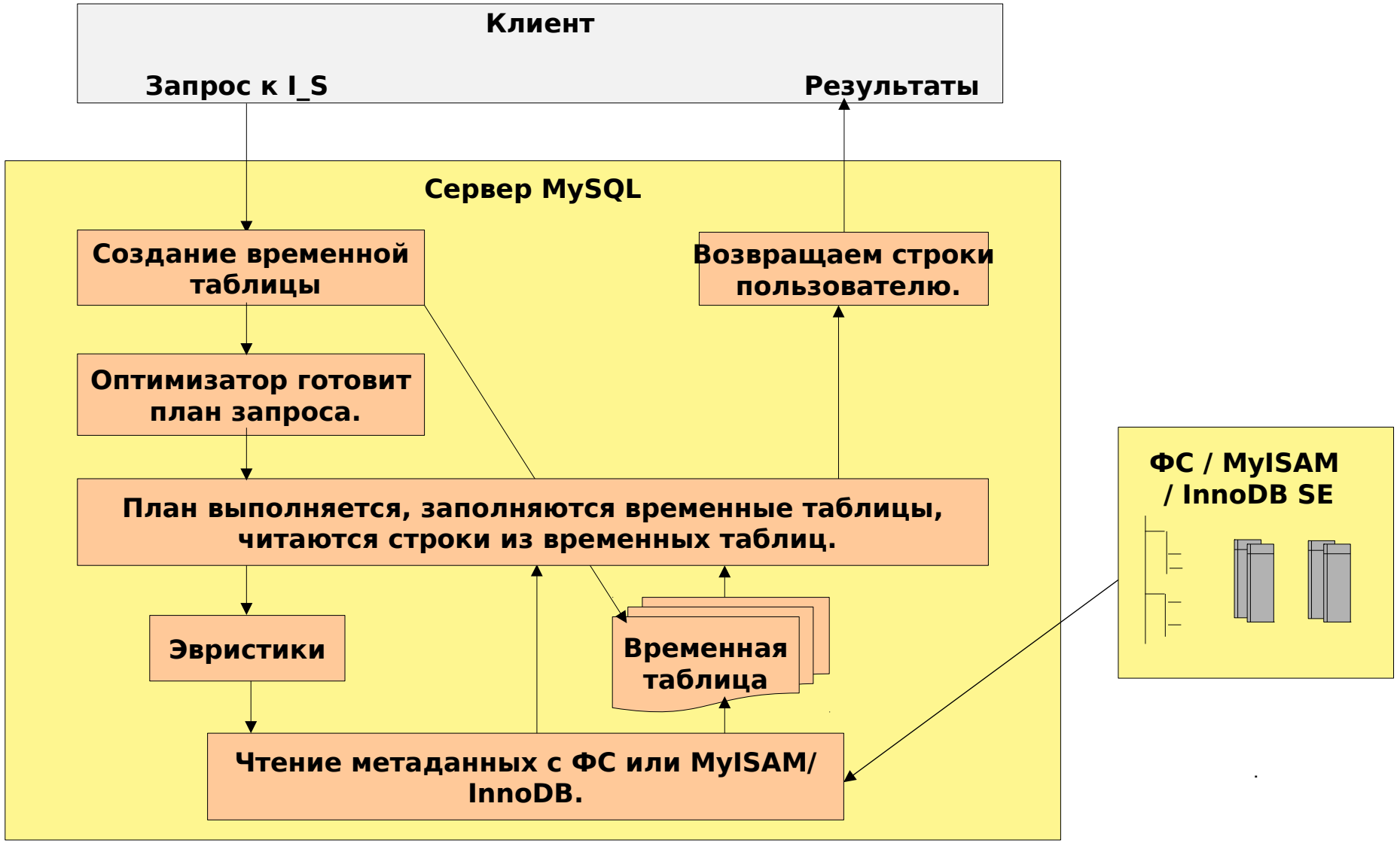
Словарь данных до 8.0: Проблемы

- INFORMATION_SCHEMA медленная
- Конфликты из-за нетранзакционного хранения МД
- Конфликты между МД в InnoDB и в ядре сервера
- Трудно реализовать атомарный и транзакционный DDL
- Проблемы с репликацией неатомарного DDL
- Сложно расширять

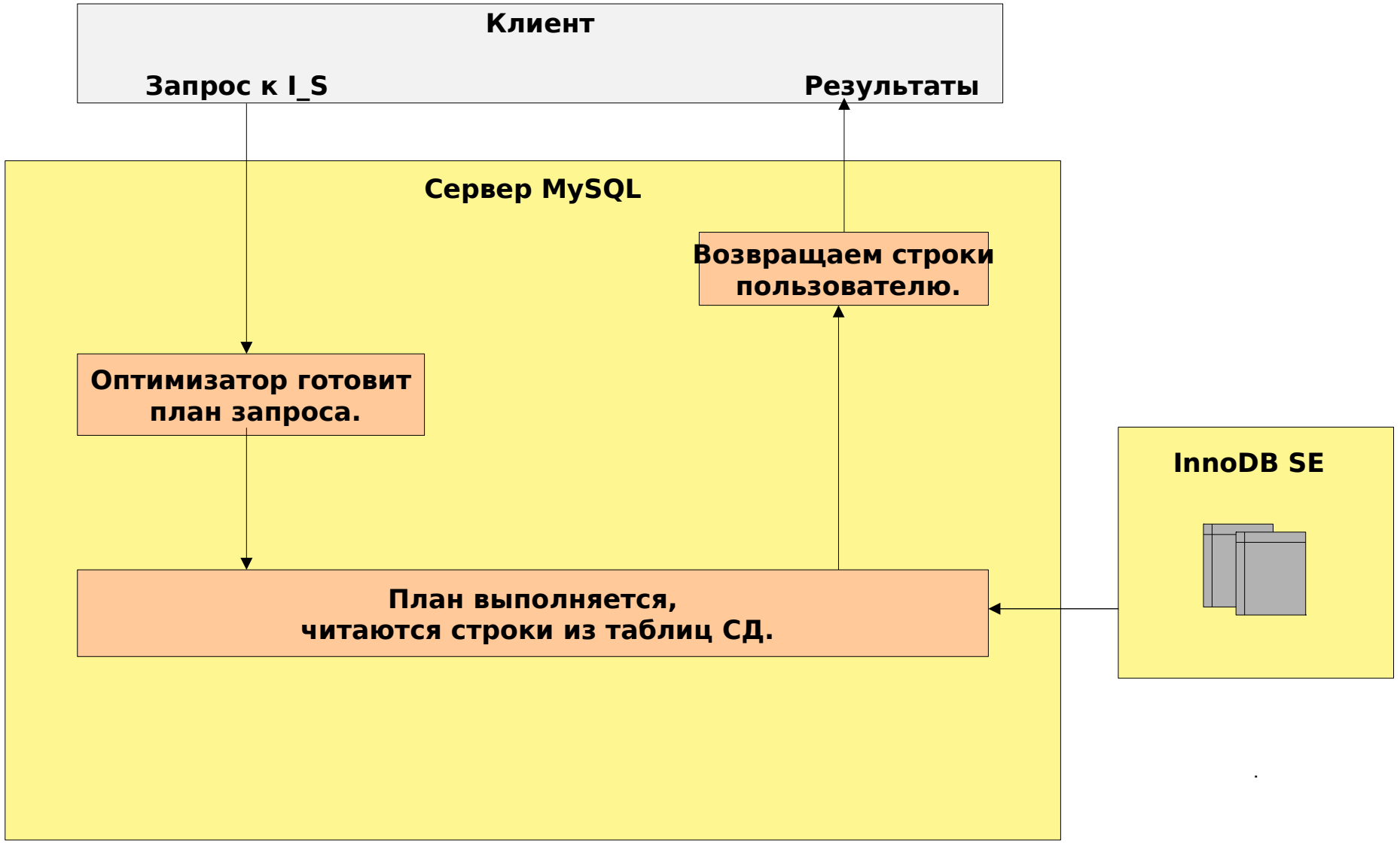
Новый словарь данных

- Единое хранилище для всех метаданных – таблицы
  - Транзакционность
  - INFORMATION_SCHEMA – просто представления
- Единое API для ядра сервера, SE и плагинов
- Единообразное API для разных объектов
- Расширяемость
  - Продуман upgrade
  - Поддержка плагинов
- Поддержка disaster recovery (SDI)

INFORMATION_SCHEMA в 5.7

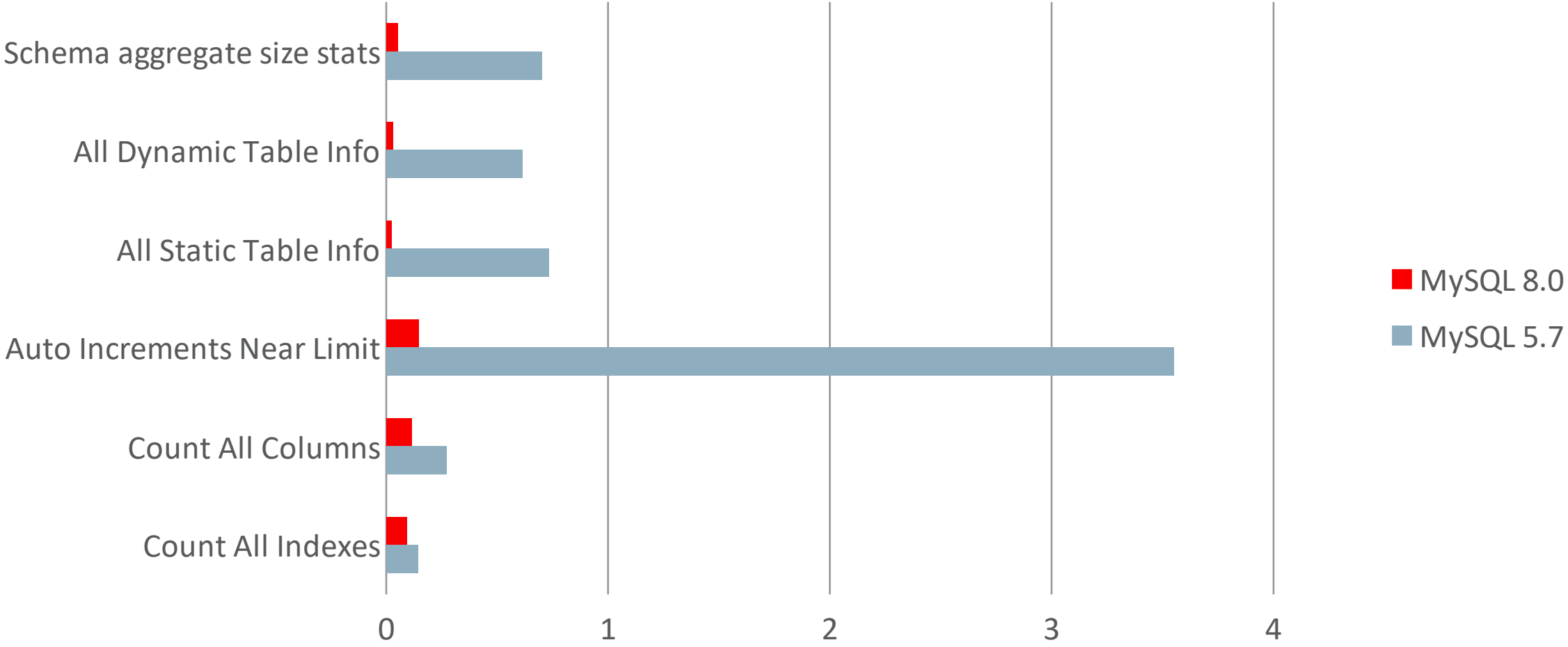


INFORMATION_SCHEMA в 8.0



Время выполнения запросов к INFORMATION_SCHEMA в 8.0

(секунды, 100 БД по 50 таблиц)

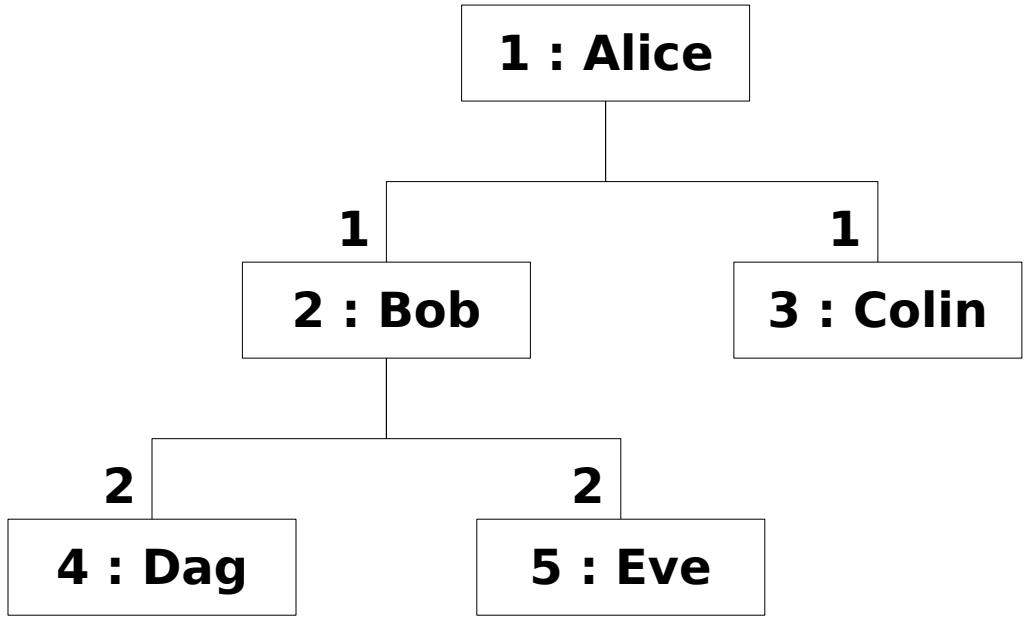


Атомарный DDL

В разработке

- Семантика все или ничего при ошибках
  - Лучше соответствует ожиданиям пользователей
  - Проще реплицировать
- Согласованность информации при падении сервера
  - Словарь данных
  - SE
  - Binary log
- Нужна поддержка в SE (InnoDB, NDB)

Работа с иерархическими данными до 8.0: Модель списка соседей

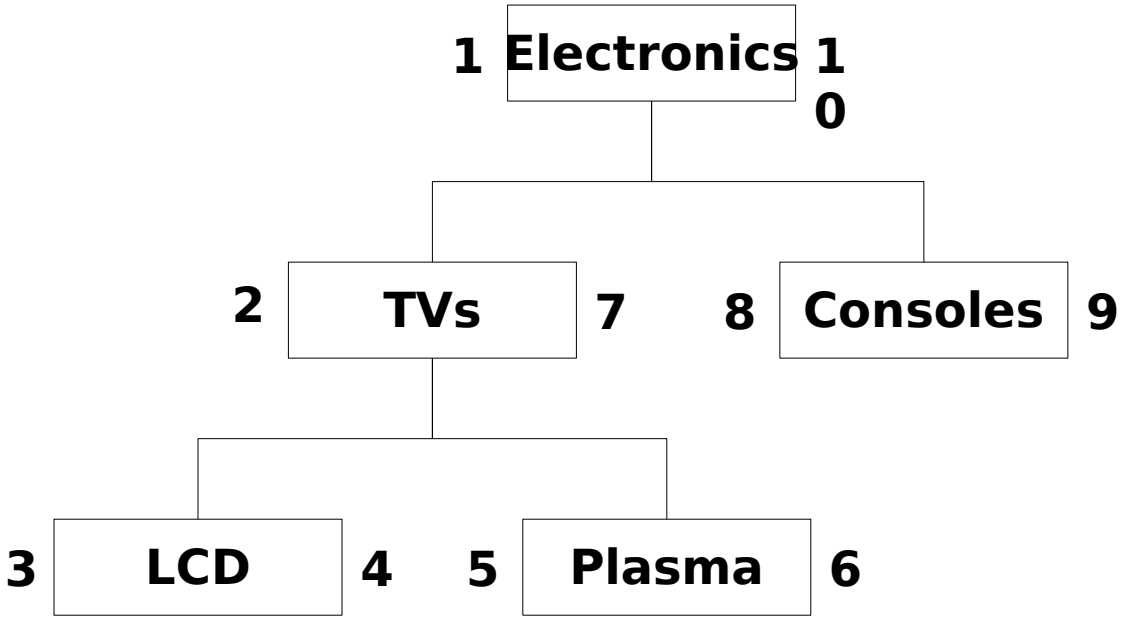


ID	NAME	MANAGER_ID
1	Alice	NULL
2	Bob	1
3	Colin	1
4	Dag	2
5	Eve	2

Проблемы:

- Проблемы с высотой дерева
- Приходится писать процедуры

Работа с иерархическими данными до 8.0: Модель вложенных множеств



NAME	LFT	RGT
Electronics	1	10
TVs	2	7
Consoles	8	9
LCD	3	4
Plasma	5	6

Проблемы:

- Нетривиальная реализация
- Сложные вставка и удаление

Работа с иерархическими данными в 8.0: Рекурсивные табличные выражения

```
WITH RECURSIVE employee_levels (id, name, level) AS
(
  SELECT id, name, 0
    FROM employees
    WHERE manager_id IS NULL
  UNION ALL
  SELECT e.id, e.name, el.level + 1
    FROM employee_levels AS el JOIN employees AS e
      ON el.id = e.manager_id
)
SELECT * FROM employee_leves ORDER BY level;
```

Табличные выражения: не только рекурсия!

```
SELECT s.name, r.total_revenue FROM supplier AS s, revenue0 AS r
WHERE s.id = r.supplier_id AND
      r.total_revenue = (SELECT MAX(total_revenue) FROM revenue0)
ORDER BY s.name;

WITH revenue0(supplier_id , total_revenue) AS (
    SELECT supplier_id, SUM(item_price * items) FROM orders
    ...
    GROUP BY supplier_id )
SELECT s.name, r.total_revenue FROM supplier AS s, revenue0 AS r
WHERE s.id = r.supplier_id AND
      r.total_revenue = (SELECT MAX(total_revenue) FROM revenue0)
ORDER BY s.name;
```

Оконные функции

Проблема:

- **До версии 8.0 сложно и иногда невозможно считать агрегатные функции от группы из текущей и связанных с нею строк (окна) без группировки всех строк в группе.**

Решение:

- **Оконные функции**

Оконные функции: пример

В разработке

```
SELECT e.name, d.name AS dep_name, salary,
       RANK() OVER (PARTITION BY d.id ORDER BY salary DESC) rank_in_dep
FROM employees e, departments d
WHERE e.department = d.id ORDER BY e.name;
```

NAME	DEP_NAME	SALARY	RANK_IN_DEP
Alice	B	50000	1
Donald	A	20000	2
Ingrid	B	40000	2
Mary	A	15000	3
Sebastian	B	40000	2
William	A	30000	1

Убывающие индексы до 8.0

- Поддержка синтаксиса
- Создается обычный возрастающий индекс
- Проблемы:
  - Сканирование в обратном порядке дороже
  - Нельзя использовать индекс когда нужно разное направление обхода по разным колонкам индекса

Убывающие индексы в 8.0

- **Поддерживаются InnoDB**
- **Оптимизатор умеет их правильно использовать**
  - **Есть ограничения на оптимизации**

```
CREATE TABLE t (c1 INT, c2 INT,
                INDEX idx1 (c1 ASC, c2 DESC));

SELECT … ORDER BY c1 ASC, c2 DESC; # без filesort
```

Невидимые индексы

Что это такое ?

- **Невидимы для оптимизатора**
- **Но обновляются DML (сравните с DISABLED индексами)**

Нужно для:

- **“Удаление” индексов с мгновенным восстановлением**
- **Пошаговое добавление индексов**

Невидимые индексы: пример

- **Думаем что индекс не используется:**

```
ALTER TABLE country ALTER INDEX c INVISIBLE;
```

- **Ошиблись! Быстро возвращаем индекс назад:**

```
ALTER TABLE country ALTER INDEX c VISIBLE;
```

- **Все верно. Можно удалить индекс:**

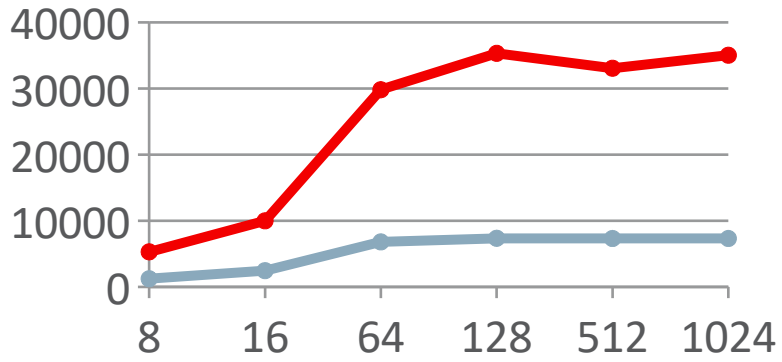
```
ALTER TABLE country DROP INDEX c;
```

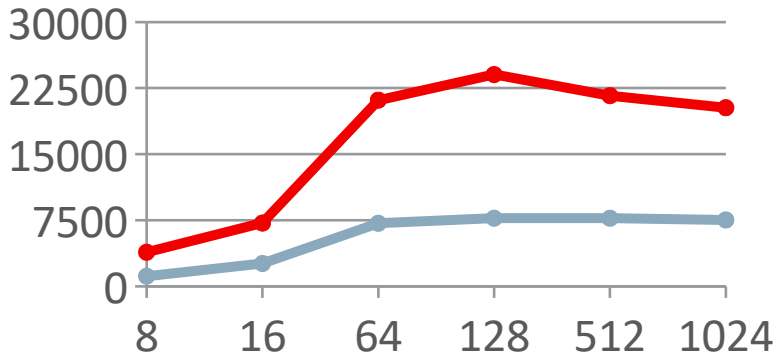
Unicode

- **Utf8mb4 – по умолчанию в 8.0!**
- **Поддержка Unicode 9.0**
- **Новые accent-sensitive и case-sensitive collation**
- **Collation специфичные для различных языков**
  - **В том числе для японского!**

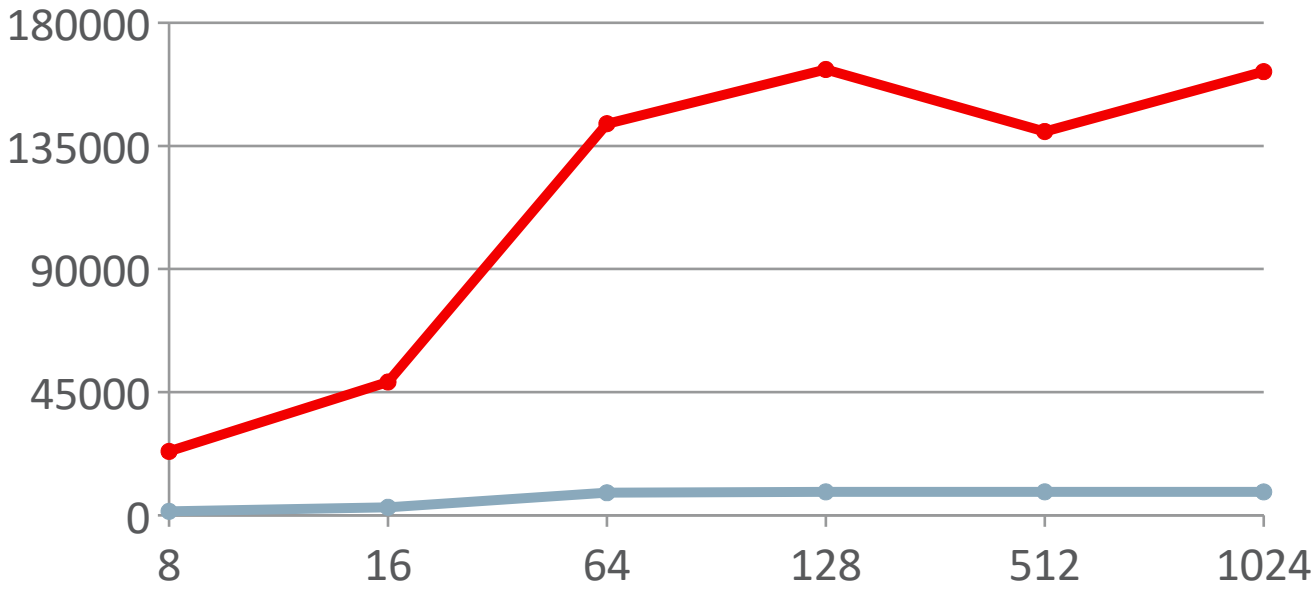
- **Оптимизация производительности utf8mb4**

Оптимизация utf8mb4: 8.0 vs 5.7





+300-350% в OLTP RO
+176-233% в OLTP RW
+1500-1800% в SELECT_DISTINCT_RANGES



Гибкая работа с блокировками

Проблема:

Хочется пропускать заблокированные строки:

- **Выборка из таблицы заданий**
- **Резервирование ресурса**

Решение:

```
SELECT … FOR UPDATE SKIP LOCKED;
```

Гибкая работа с блокировками (детали)

- `SKIP LOCKED` **– пропустить заблокированные строки**
- **Используется совместно с:**
  - `FOR UPDATE`
  - `FOR SHARE` **(замена** `LOCK IN SHARE MODE`**)**
- `FOR UPDATE OF <table_name> SKIP LOCKED`

- `NOWAIT` **– выдать ошибку вместо ожидания**

Роли

- **Именованные коллекции привилегий**
  - **Могут включать другие роли**
- **Упрощают управление привилегиями пользователей**
  - **Проще структура привилегий**
  - **Легко добавлять и отбирать привилегии**
- **Можно активировать/деактивировать для сессии**
- **Можно назначить несколько активных ролей по умолчанию**
- `ROLES_GRAPHML()`

Динамические привилегии

Проблема: `SUPER` **привилегия перегружена**

Решение: Динамические привилегии

- **Изначально привилегии добавляемые плагинами**
- **Глобального уровня**
- **Также замена** `SUPER`**:**

  `AUDIT_ADMIN, BINLOG_ADMIN, CONNECTION_ADMIN,`
  `ENCRYPTION_KEY_ADMIN, FIREWALL_ADMIN,...`

Репликация: оптимизация взаимодействия между I/O и SQL потоками

Проблема:

Репликация отстает или неэффективна

Решение:

Вынести запись и чтение в/из relay log из критических секций

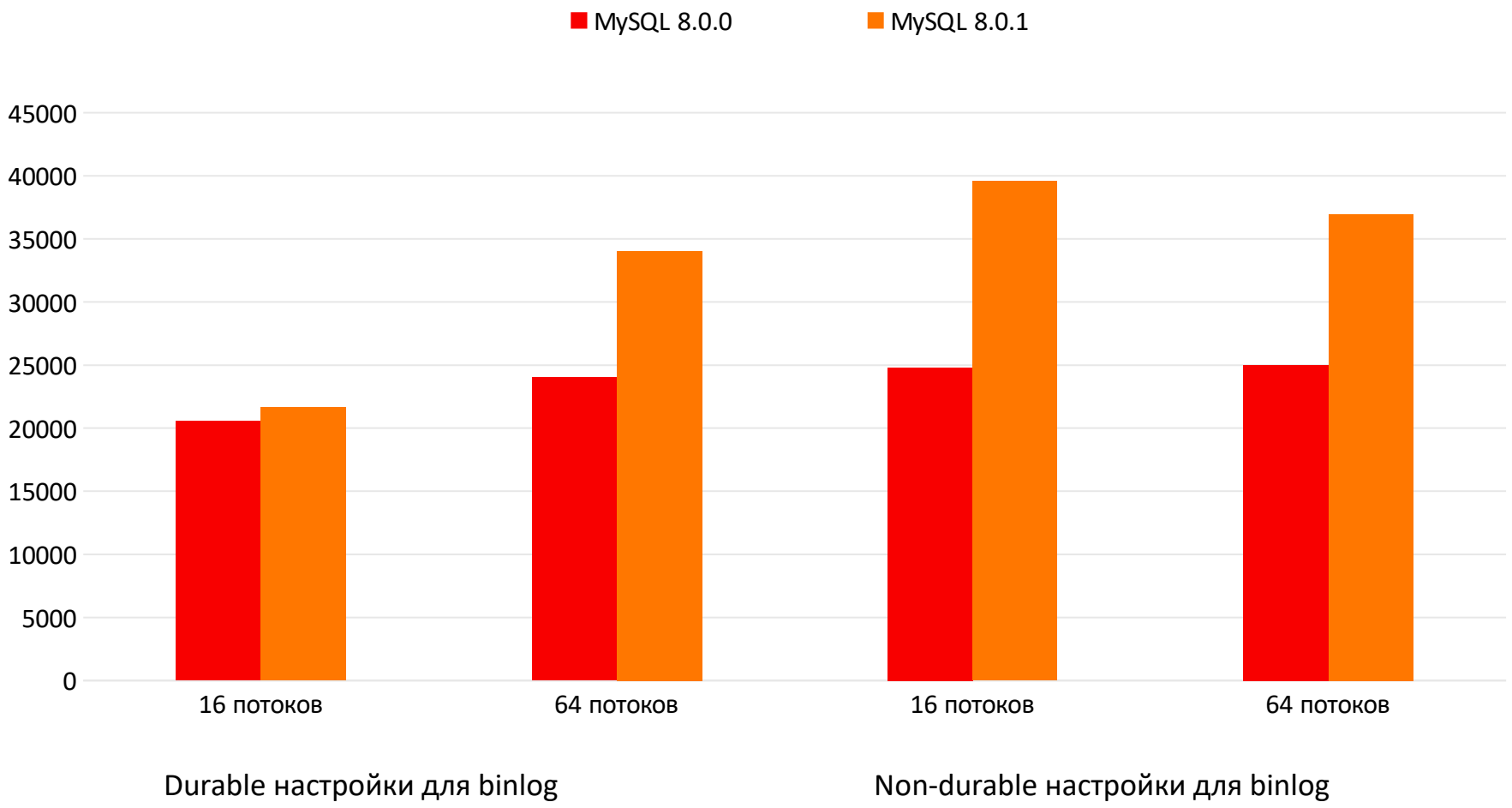
Меньше держим блокировки

Потоки ждут доступа к relay log меньше

Выше производительность

Результаты оптимизации взаимодействия между I/O и SQL потоками

(Sysbench Update Indexed: применение транзакций/сек на реплике, RBR)



Репликация: параллелизм на основе использования write-set

Проблема:

- **Репликация отстает или неэффективна**
- **Параллельная репликация на основе логических временных меток не помогает**
  - **Малое число или одно соединение на мастере**
  - **Сложная топология приводит к потере параллелизма**

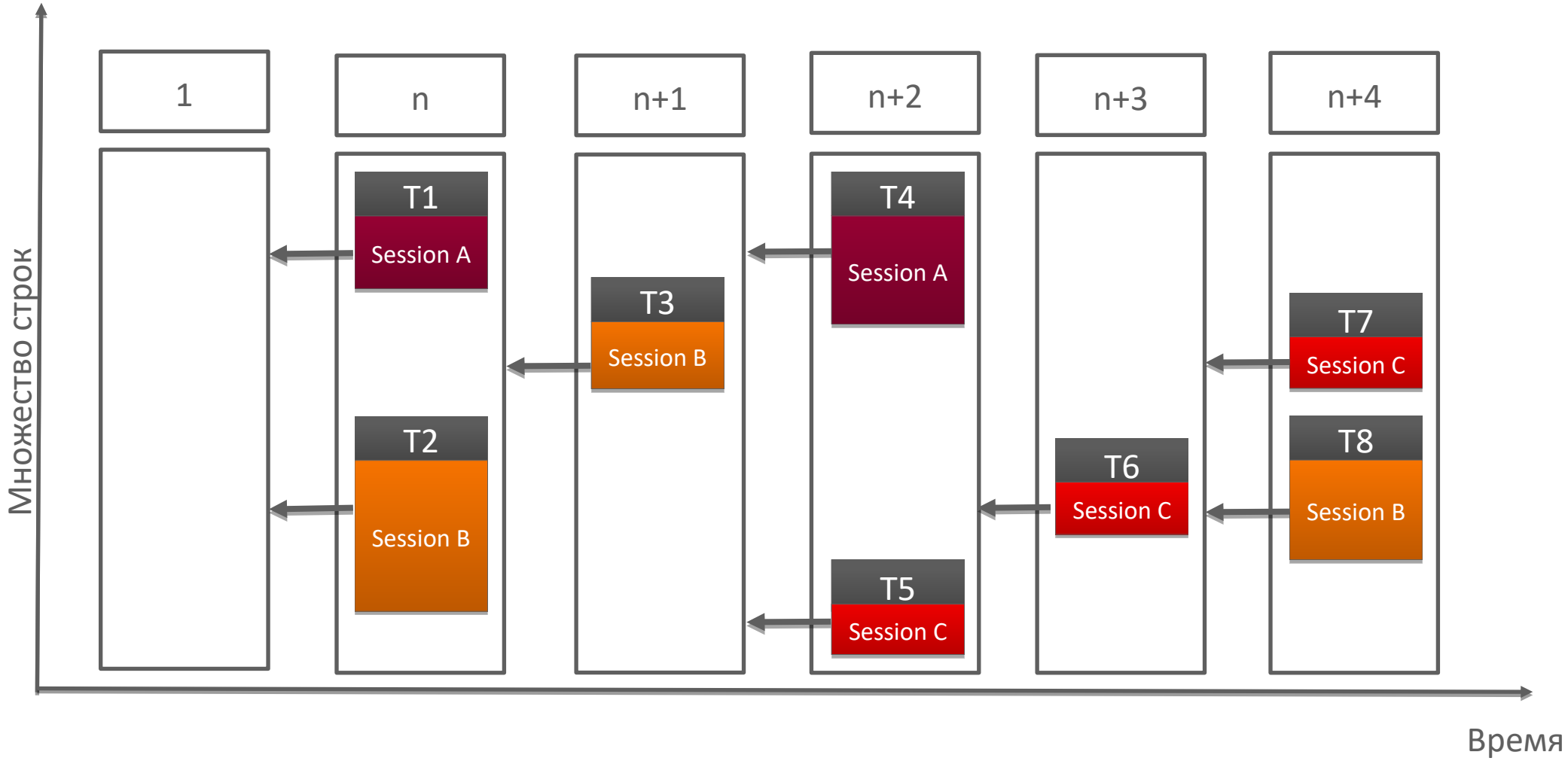
Решение:

- **Использовать параллелизм на основе write-set транзакций**

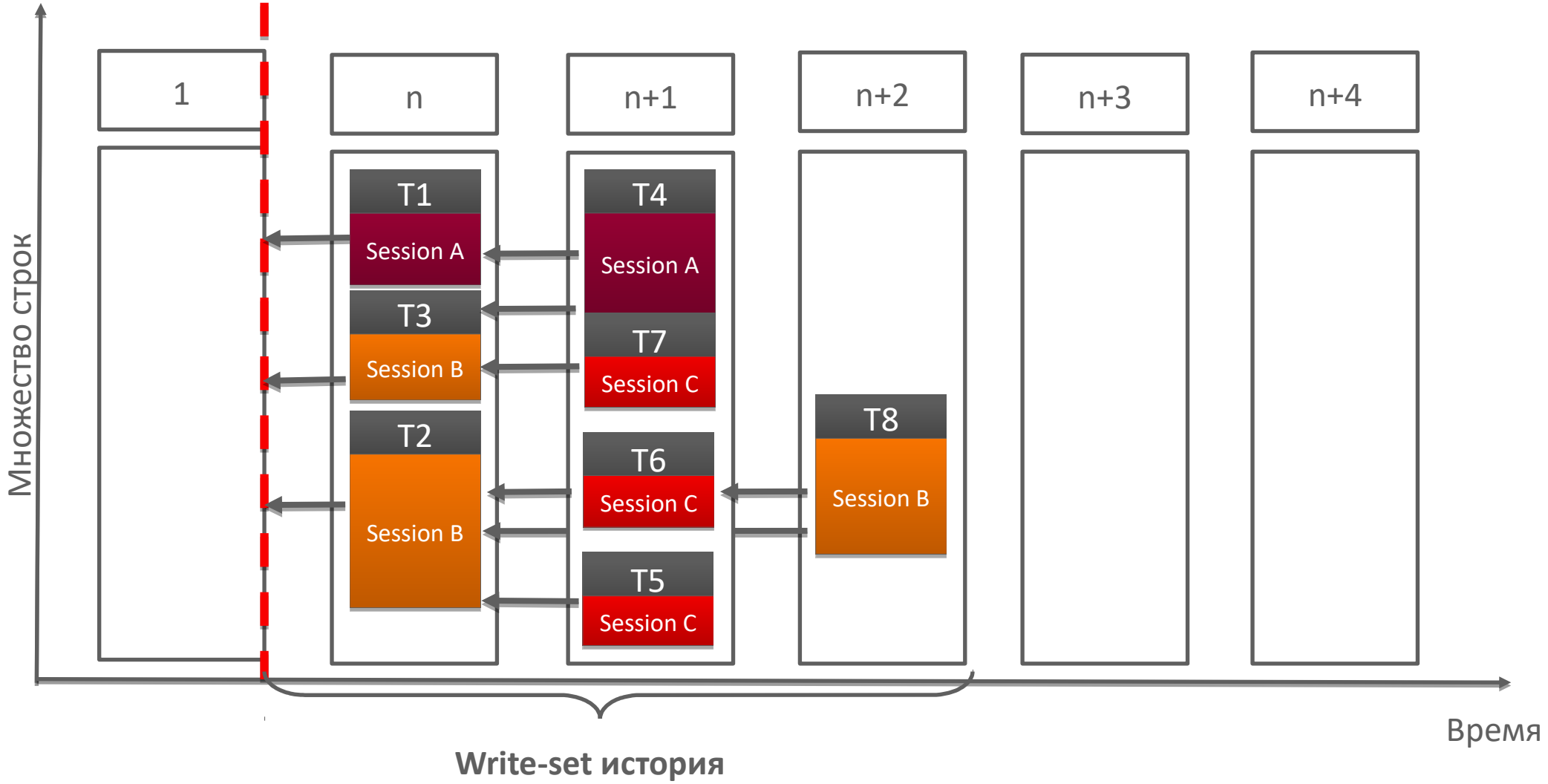
Репликация: параллелизм на основе использования write-set

- Write-set транзакции – множество строк измененных транзакцией
- Write-set – средство обнаружить завимости
- Транзакции с непересекающимися write-set можно применять паралелльно
- Параллелизм на основе write-set расширяет механизм на основе логических временных меток
- Зависимости определяются на мастере и выражаются при помощи логических временных меток в binary log

Параллельная репликация до версии 8.0: на основе логических временных меток

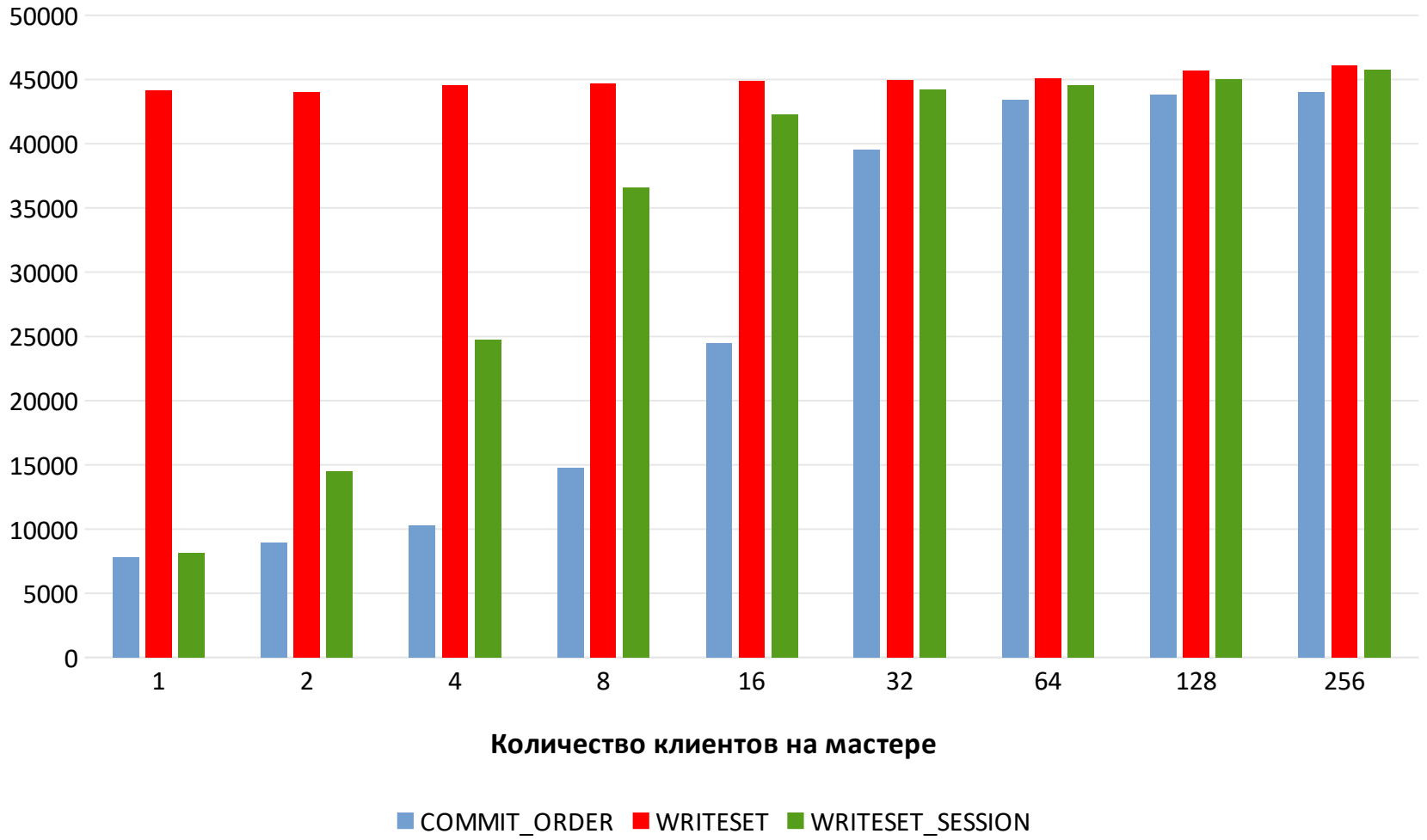


Параллельная репликация начиная с 8.0: на основе использования write-set



Результаты использования write-set параллелизма:

(Sysbench Update Indexed: применение транзакций/сек на реплике)



Спасибо!