Greetings!

Сравнение Redis и MySQL на задаче построения игровых рейтингов

Алексей Романов, Минск
Melesta Games

Hello, gamedev!

tya

Постановка задачи

- Еженедельный турнир среди игроков по заработанным деньгам
- Глобальный рейтинг игроков
- Обновление рейтинга
- Постраничное получение рейтинга

Какой путь выбрать?

- Почитать статьи, послушать доклады опытных товарищей(например, с LVEE)
- Посмотреть результаты бенчмарков
- Написать прототип своей системы и проверить, как же она поведет себя на предполагаемой нагрузке

Средства разработки

- Python 2.7
- Twisted
- MySQL
- Redis

Почему Python и Twisted?

- Высокая скорость разработки
- Open source
- Асинхронная модель позволяет получить неплохую производительность на выходе
- Corutines в виде генераторов делают код понятным человеку(yield, inlineCallbacks)

Почему MySQL?

- Широко применяется в различных проектах компании
- Open source
- Популярная
- Обеспечивает неплохую производительность

Redis, что это?

- NoSQL СУДБ
- Open source(3-clause BSD license)
- Много классных типов данных
- Данные должны влезать в ОЗУ
- Стабильно и быстро работает
- Быстро развертывается, легко администится

Схема данных MySQL

```
CREATE TABLE `weekly_tournament` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
  `tournament_id` int(11) NOT NULL COMMENT 'Дата начала турнира',
 `person_id` int(11) NOT NULL COMMENT 'См. persons.id',
 `money_earned` int(11) NOT NULL DEFAULT '0' COMMENT 'Сколько заработано денег в турнире?',
 PRIMARY KEY (`id`),
 UNIQUE KEY `complex_key` (`tournament_id`,`person_id`),
 KEY `tourn_i` (`tournament_id`),
 KEY `person_id` (`person_id`),
 KEY `money_earned_key` (`tournament_id`,`money_earned`)
) ENGINE=InnoDB  COMMENT='Таблица для хранения результатов еженедельного турнира'
```

Схема данных Redis

- Какая ещё схема? В Redis она не нужна!
- Используем sorted set!
- Ключ — id игрока, score — текущее количество заработанных денег за неделю
- ZADD, ZREVRANGE, ZREVRANK

  http://redis.io/commands/sadd

Алгоритм работы с MySQL

- Обновление рейтинга: обычный INSERT или UPDATE в зависимости от состояния хэш-таблицы в ОЗУ
- Получение страницы рейтинга

```
SELECT person_id, money_earned FROM
weekly_tournament WHERE

tournament_id=%(tournament_id)s ORDER

BY money_earned DESC LIMIT %(offset)s, %
(page_size)s
```

Алгоритм работы с MySQL

- Обновление рейтинга: обычный INSERT или UPDATE в зависимости от состояния хэш-таблицы в ОЗУ
- Получение страницы рейтинга

```
SELECT person_id, money_earned FROM
weekly_tournament WHERE

tournament_id=%(tournament_id)s ORDER

BY money_earned DESC LIMIT %(offset)s, %
(page_size)s
```

Алгоритм работы с MySQL(ч.2)

- Получение текущей позиции игрока:

```
SELECT wo.money_earned,
(SELECT COUNT(wi.money_earned) + 1 FROM
weekly_tournament wi WHERE
wi.tournament_id=%(tournament_id)s AND
wi.money_earned > wo.money_earned) AS
rank
FROM weekly_tournament wo WHERE
wo.tournament_id=%(tournament_id)s AND wo.person_id=
%(person_id)s
```

Алгоритм работы с Redis

- Обновление, вставка:

  zincr(get_pvp_rating_name(), person_id, delta)

- Получение страницы рейтинга:

  zrevrange(get_pvp_rating_name(), start_pos, end_pos, withscores=True)

- Получение позиции игрока:

  zrevrank(get_pvp_rating_name(), person_id)

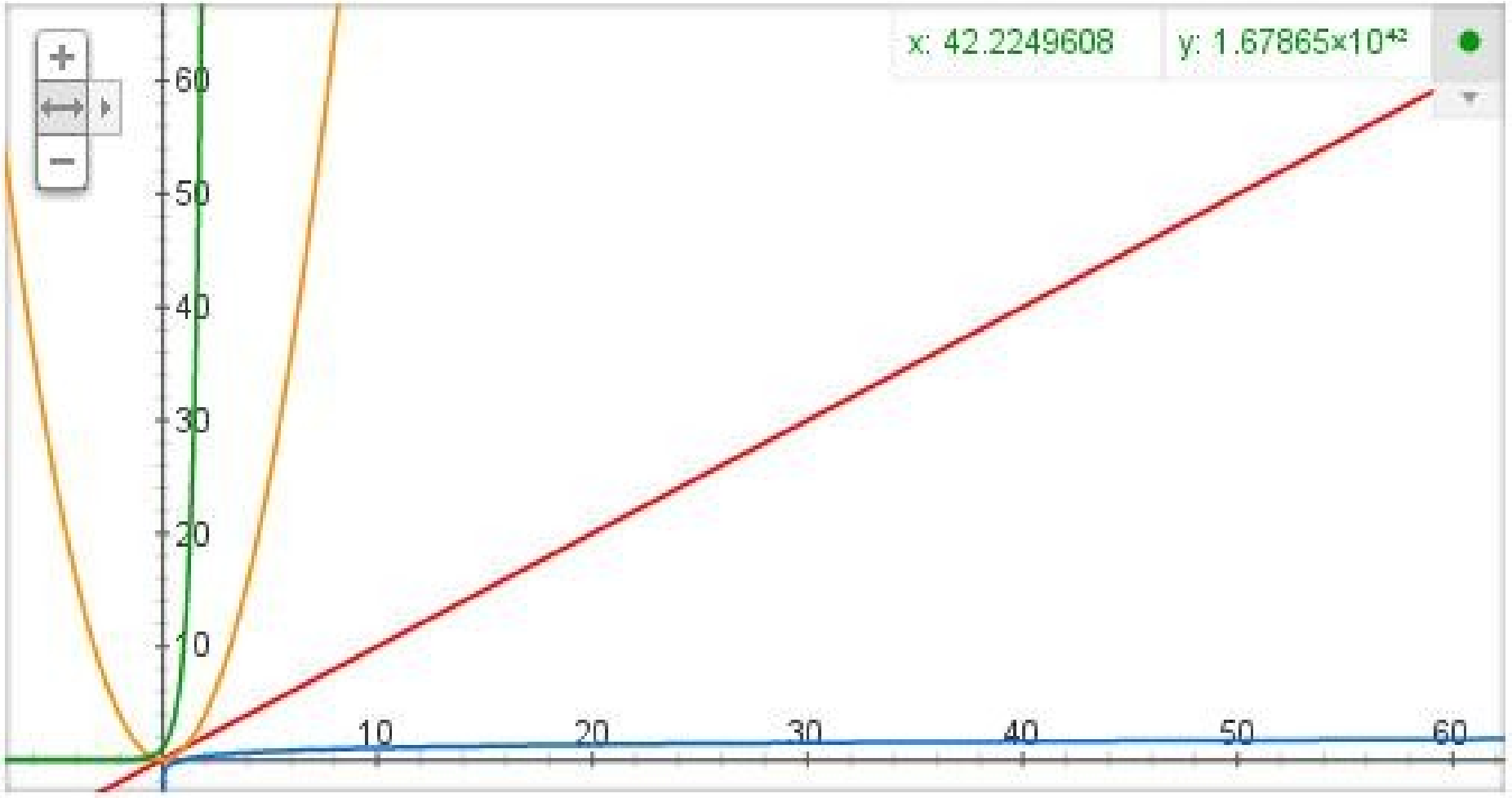
- Получение очков игрока: zscore

  zscore(get_pvp_rating_name(), person_id)

https://github.com/drednout/redis_vs_mysql

Асимптотический анализ, введение

Graph for log(x), x, x^2, 10^x



Анализ кода

MySQL:

- Можно оценить только очень приблизительно, нет информации
- Либо экспериментальным путём

Анализ кода, ч.2

А вот в Redis все иначе!

Смотрим в документацию:

zadd — O(log(N))

zrevrange - O(log(N)+M)

zrevrank - O(log(N))

zscore — O(1)

N — количество игроков, M — размер страницы рейтинга

Методика эксперимента

- Базовые тесты: insert_test, update_test, select_test
- Каждый эксперимент проводится 5 раз
- Проводится мониторинг процессов redis, mysql раз в 1 секунду(CPU%, CPU User/System, I/O)
- Данные собирается в текстовые файлы, а затем агрегируется скриптами

Инструменты для измерений

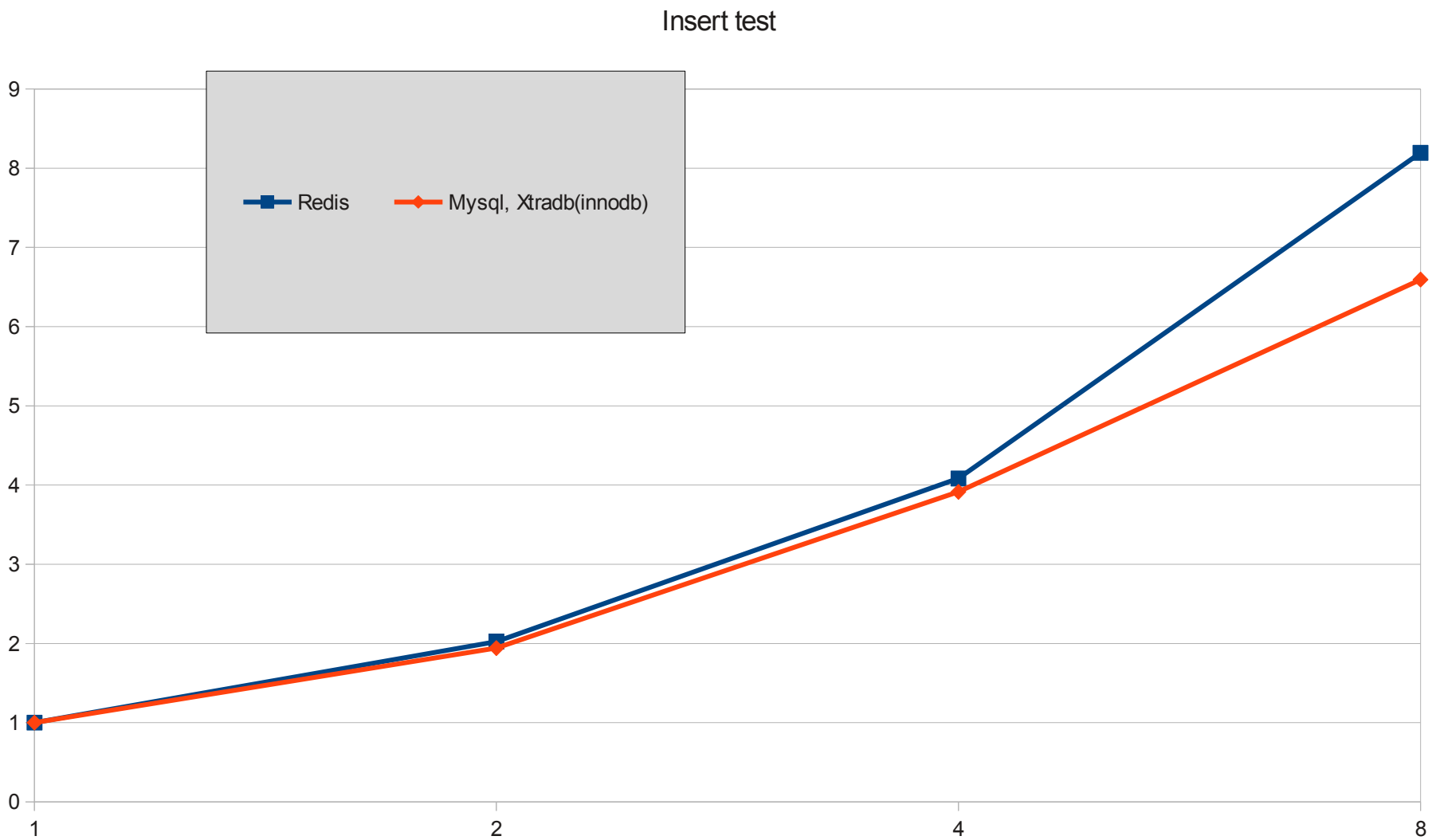
- test_weekly_tournament.py
- monitor_process.py
- Вспомогательные Python-скрипты: aggregate_monitor_log.py, avg_time.py
- Вспомогательные шелл-скрипты
- Zsh, du, grep и др.

Конфигурация стенда

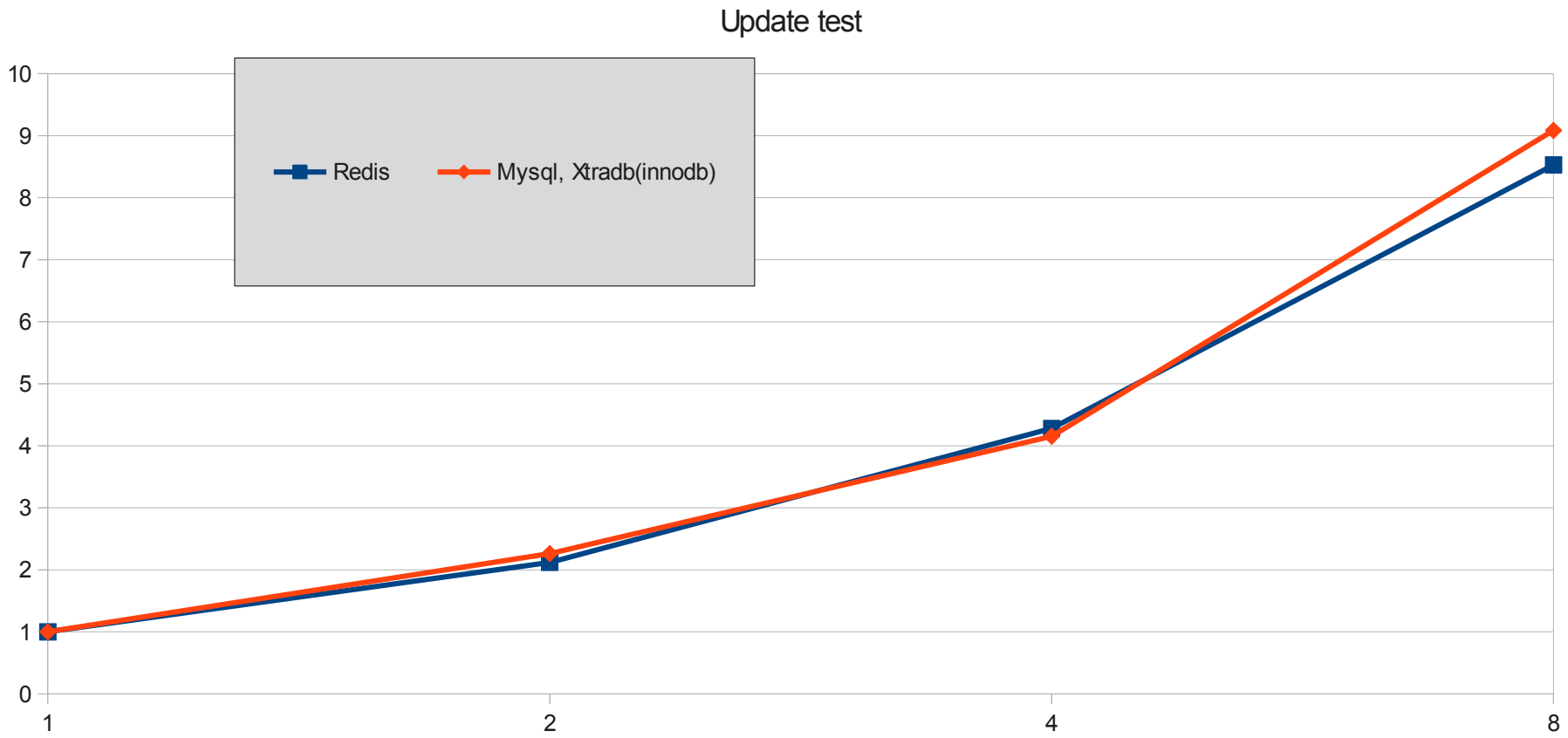
- Intel Core i5 4 физ ядра
- 8 GiB RAM

- Arch Linux, am64
- MySQL MariaDB 5.5.31
- Redis 2.6.13

Тест на различных объёмах данных, Insert

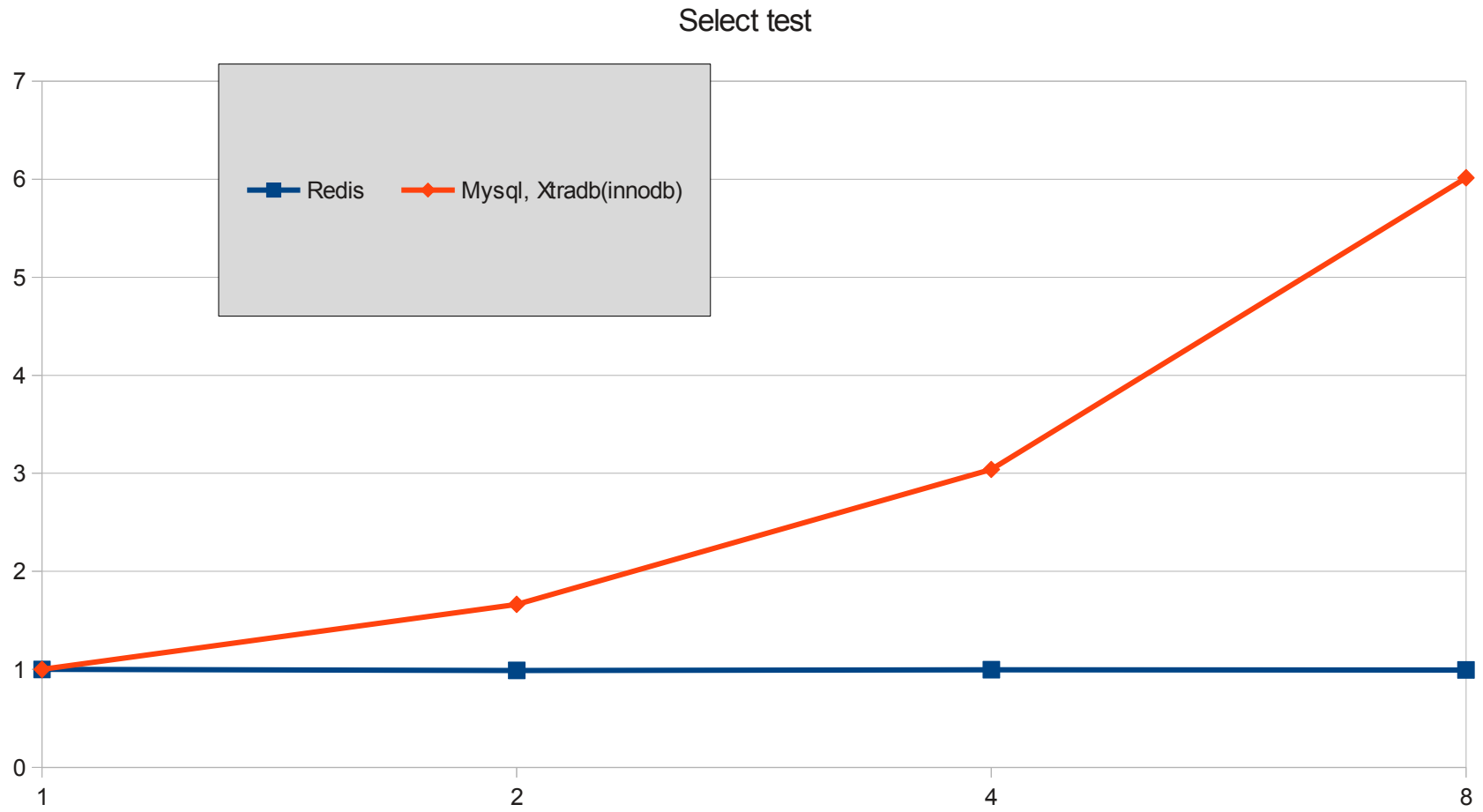


Тест на различных объёмах данных, Update



1 – 100K, 2 — 200K, 4 — 400K, 8 — 800K в рейтинге

Тест на различных объёмах данных, Select



Сводная таблица с асимптотической оценкой

Асимптотический анализ операций над рейтингом		
Операция	Redis	MySQL
Добавить новую запись	O(log(n))	~O(1)
Обновить существующую запись	O(log(n))	~O(1)
Получить страницу рейтинга	O(log(n))	~O(n)

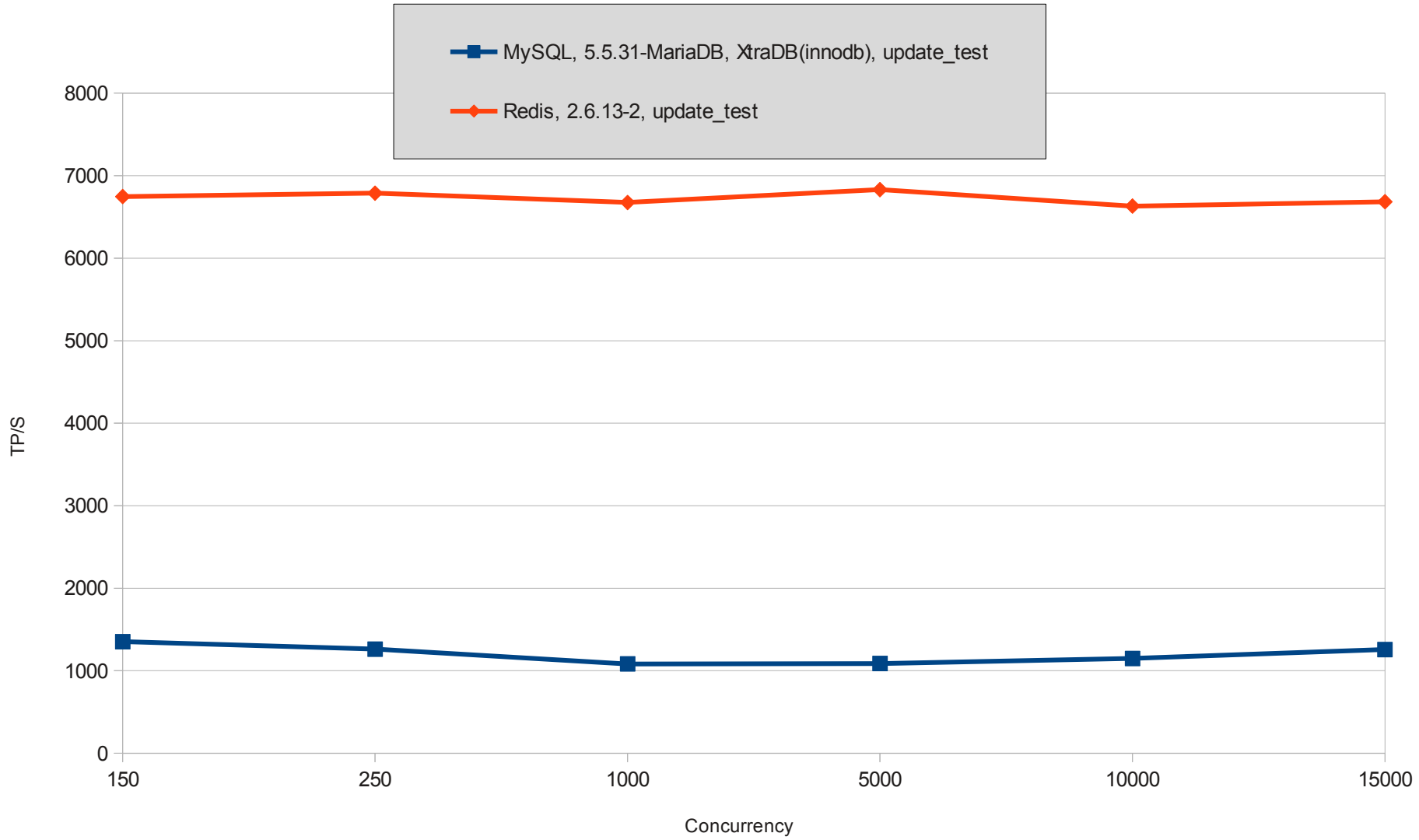
Немного о цифрах...

	Redis 2.6.13-2, concurrency: 1				
Название теста	Время, с	CPU, %	CPU,	I/O, read count	I/O, write count
Insert test(вставка 100000 записей)	25,2	25,28	31,33	35504797	7359284
Update testt(обновление 100000 записей)	25,7	25,85	31,71	34140717	8243913
Select test(получение 3000 страниц рейти	2,4	20,87	2,4	2698142	3433165
	MySQL 5.5.31-MariaDB XtraDB(innodb), concurrency: 1				
Название теста	Время, с	CPU, %	CPU,	I/O, read count	I/O, write count
Insert test(вставка 100000 записей)	113,1	20,74	238,1	83672040	820308156
Update testt(обновление 100000 записей)	125,2	40,96	281,9	80303027	1899345656
Select test(получение 3000 страниц рейти	559,5	121,82	2774	7357505380	10036059374
Select test с правильным индексом	60,75	34,74	110,3	92215574	7514047

Обобщение цифр

На **Insert test** redis быстрее MySQL в	4.488095	раза
На **Update test** redis быстрее MySQL в	4.871595	раза
На **Select test** redis быстрее MySQL в	233.125	раза
В случае с постоением правильного индек		
На **Select test** redis быстрее MySQL в	25.3125	раза

Тест на различном уровне конкуррентности



Насчет памяти...

Параметр	MySQL 5.5. Xtradb(innodb)	Redis 2.6.13	
Размер файлов на диске(KiB)	126989	11556	
Размер данных в памяти(KiB)	115248	11556	
Данные в MySQL занимают в	10,9890100381	больше раз места на диске, чем в Redis	
Данные в MySQL занимают в	9,9730010384	больше раз места в ОЗУ, чем в Redis	
10**6 участников			

Итог по цифрам

- На операциях вставки и обновления Redis быстрее MySQL в 5 раз
- На операции получения страницы рейтинга и ранга игрока выйгрыш Redis по производительности до 25 раз
- На конкуррентом тесте картина удовлетворительная для обоих решений
- По памяти и дисковому пространству Redis выигрывает в 10 раз

Итого

- Sorted set — правильная структура данных для задач построения рейтингов
- Команды к Redis писать проще, чем запросы к MySQL
- Redis по результатам бенчмарков опережает MySQL для задачи построения глобального рейтинга
- Все равно от MySQL никуда не денешься

The end

Спасибо за внимание!

Thank you for your attention!

Дзякую за увагу!

https://github.com/drednout/redis_vs_mysql