Опыт миграции высоконагруженных игровых проектов с MySQL на PostgreSQL

PGConf.Russia 2018

@mail.ru

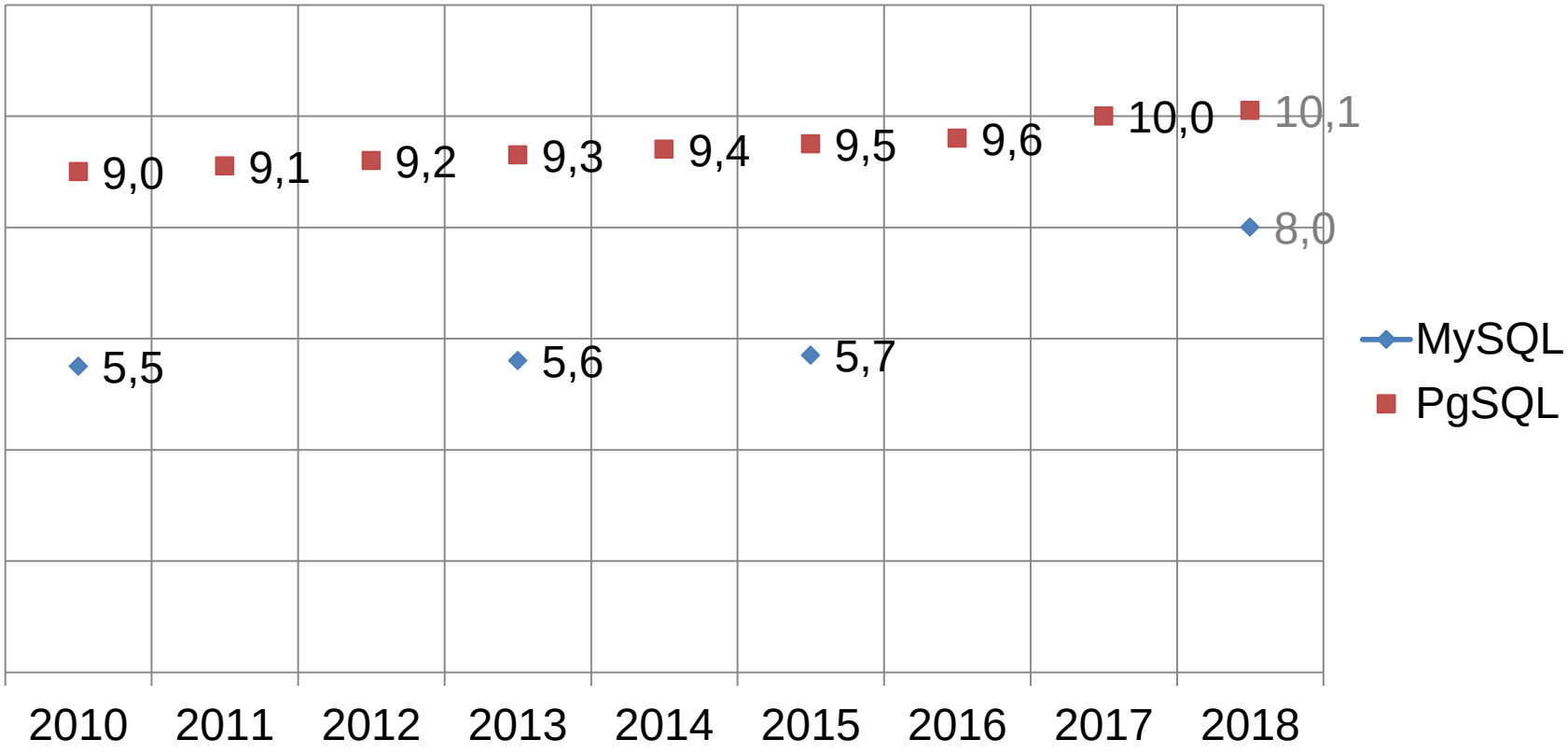
Характеристики проектов

Характеристика	Проект 1	Проект 2
Несколько серверов	■	✘
DB size	40-200GB	150GB
Memory	32-64 GB	64 GB
Storage system	4-6 SAS RAID 10	4 SSD RAID 10
Минимальное время отклика	■	✘
QPS	1000	5000
Total users	5M+	5M+

Опасения при переходе

- Декодирование WAL
- Фоновая работа vacuum
- Скорость CRUD-операций

Развитие MySQL vs PostgreSQL



ALTER TABLE в MySQL медленный

В большем числе случаев приводит к пересозданию таблицы

```
ALTER TABLE tbl ADD COLUMN c1 int;
```

MySQL	PgSQL
O(N)	O(1)

IDENTITY в MySQL

- IDENTITY –последовательность уникальных идентификаторов
- IDENTITY в MySQL называется AUTO_INCREMENT
- Значения AUTO_INCREMENT в реализации MySQL в некоторых случаях могут повторяться

Документация MySQL, избранное

If you specify an `AUTO_INCREMENT` column for an `InnoDB` table, the table handle in the `InnoDB` data dictionary contains a special counter called the auto-increment counter that is used in assigning new values for the column. This counter is stored only in main memory, not on disk.
To initialize an auto-increment counter after a server restart, `InnoDB` executes the equivalent of the following statement on the first insert into a table containing an `AUTO_INCREMENT` column.

Документация MySQL, избранное

If you specify an `AUTO_INCREMENT` column for an `InnoDB` table, the table handle in the `InnoDB` data dictionary contains a special counter called the auto-increment counter that is used in assigning new values for the column. This counter is stored only in main memory, not on disk.
To initialize an auto-increment counter after a server restart, `InnoDB` executes the equivalent of the following statement on the first insert into a table containing an `AUTO_INCREMENT` column.

Документация MySQL, избранное

```
CREATE TABLE test(id integer check (id > 0));

INSERT INTO test VALUES(-1);
```

Документация MySQL, избранное

The CHECK clause is parsed but ignored by all storage engines. See Section 1.8.2.3, “Foreign Key Differences”.

Вы знаете SQL?

```
CREATE TABLE test (id1 integer, id2 integer);

INSERT INTO test VALUES (1,2);

UPDATE test SET id1 = id2, id2 = id1;

SELECT * FROM test;
```

Нетранзакционный DDL

- Если все идет хорошо, то все закончится хорошо
- Если что-то пошло не так, то нужен будет backup
- В MySQL 8 обещают транзакционность на уровне STATEMENT

Репликация MySQL vs PostgreSQL

- Асинхронная репликация
- Statement-based
- Row-based

Документация MySQL, избранное

```
CREATE TABLE t2 LIKE t1;
ALTER TABLE t2 ADD id INT AUTO_INCREMENT PRIMARY KEY;
INSERT INTO t2
SELECT * FROM t1 ORDER BY col1, col2;
```

To guarantee the same ordering on both master and slave, the ORDER BY clause must name *all* columns of t1.

Неявные индексы на foreign key

MySQL	PgSQL
■	✖

Если есть нагрузочное тестирование на реальных данных, то об этом станет известно заранее, даже если Вы об этом не знали/забыли

JSON

MySQL 5.6	MySQL 5.7	PgSQL 9.3+	PgSQL 9.4+
Varchar + StoredProcedures	JSON	JSON	JSONB

JSONB может давать overhead в объеме

- добавление 1 числа в массив дает +12 байт к размеру jsonb
- каждое увеличение разрядности числа кратное 4-м дает еще +2 байта
- SELECT pg_column_size('{“test1":[1,2,3,4,5]}'::jsonb)

JSONB может давать overhead в объеме

- добавление 1 числа в массив дает +12 байт к размеру jsonb
- каждое увеличение разрядности числа кратное 4-м дает еще +2 байта
- SELECT pg_column_size('{“test1":[1,2,3,4,5]}'::jsonb)
- 88 байт
- зато быстрее работать

Отсутствие кластерного индекса

- Серьезно портит жизнь
- CLUSTER table_name USING index_name немного спасает ситуацию
- Блокировки ☹
- pg_repack спасает ситуацию

Отсутствие unsigned значений

- В MySQL они есть
- В Postgres их нет ☹
- Расширить диапазон
- Жить можно вместе, но не очень удобно

JDBC PostgreSQL driver + Unix Domain Sockets

- Нет поддержки Unix Domain Socket
- Сделали свой патч
- Получили 10-15% прироста производительности на наших нагрузках

Ошибка в тестах

- Запустили тесты к драйверу
- Тесты упали
- Вызов Statement.cancel() приводил к зависанию внутри драйвера

Case insensitive поиск

```
-- MySQL
SELECT * FROM tab WHERE col= ‘TEXT’;
-- PgSQL 9.2-
SELECT * FROM tab WHERE lower(col) =
lower(‘TEXT’);
-- PgSQL 9.3+
CREATE EXTENSION citext;
SELECT * FROM tab WHERE col= ‘TEXT’;
```

Частичные индексы и null поля

- Частичные индексы на наших данных позволяют заметно уменьшить объем используемой памяти
- Представление null полей
- Экономим 10%

Представление строки в MySQL

```
CREATE TABLE T
(field1 varchar(3), field2 varchar(3), field3 varchar(3))
Type=InnoDB;
INSERT INTO T VALUES ('PP', 'PP', NULL);
```

```
97 17 15 13 0C 06      Field Start Offsets
                  …
50 50                  Field1 'PP'
50 50                  Field2 'PP'
```

```
CREATE TABLE t1 (f1 smallint,  f2 bigint);

INSERT INTO t1 VALUES (1,2);

SELECT pg_column_size(t1) FROM t1 LIMIT 1;
```

```
CREATE TABLE t1 (f1 smallint,  f2 bigint);

INSERT INTO t1 VALUES (1,2);

SELECT pg_column_size(t1) FROM t1 LIMIT 1;
```

40

```
CREATE TABLE t2 (f2 bigint, f1 smallint);

INSERT INTO t2 VALUES (1,2);

SELECT pg_column_size(t2) FROM t2 LIMIT 1;
```

34

```
CREATE EXTENSION pageinspect;

-- CREATE TABLE t1 (f1 smallint, f2 bigint);
SELECT * FROM heap_page_items(get_raw_page('t1', 0));
01 00 00 00 00 00 00 00
02 00 00 00 00 00 00 00

-- CREATE TABLE t2 ( f2 bigint, f1 smallint);
SELECT * FROM heap_page_items(get_raw_page('t2', 0));
01 00 01 00 00 00 00 00
02 00
```

```
--CREATE TABLE t1 (f1 smallint, f2 bigint, f3 varchar(32));
--INSERT INTO t1 VALUES (1,2,'PP');
SELECT * FROM heap_page_items(get_raw_page('t1', 0));
01 00 00 00 00 00 00 00
02 00 07 50 50

--CREATE TABLE t2 (f3 varchar(32), f1 smallint, f2 bigint);
--INSERT INTO t2 VALUES ('PP',1,2,);
SELECT * FROM heap_page_items(get_raw_page('t2', 0));
07 50 50 00 00 00 00 00
01 00 00 00 00 00 00 00
02 00 00 00 00 00 00 00
```

Прогрев базы

- pg_prewarm
- pg_hibernator

Миграция

	Проект 1	Проект 2
Время на миграцию	4ч.	2ч.
Структура баз	Совпадает	Разная
Порядок миграции	По очереди	За один раз
Бизнес-логика в преобразовании данных	✘	■
Метод копирования	Код на Java	MySQL -> CSV; PgSQL <- COPY FROM CSV + “Ленивая” Java миграция
Фактическое время копирования	<1ч.	<1ч.
Утилита для обратной миграции	■	✘

UPDATE TABLE + VACUUM

- На боевой сервер ушел конверт, который делает UPDATE большой таблице
- Таблица распухла
- По нашим метрикам нагрузка ощутимо поднялась, пока не сделали VACUUM

Проблемы с Vacuum

- Vacuum не так страшен, как его малюют
- Обычно под нагрузкой неиспользуемого места 10-20%
- Периодически делается pg_repack или VACUUM FULL
- В некоторых случаях таблицы все же прилично раздувает.

Collation e ё

- Приходит баг «В игре не работает поиск предметов»
- На самом деле все работает и даже лучше, так как поиск стал различать е и ё.
- Хотим все как было. Срочно.
- «Нормируем» все слова, заменяя ё на е.

Collation e ë

- Приходит баг «В игре не работает поиск предметов»
- На самом деле все работает и даже лучше, так как поиск стал различать е и ё.
- Хотим все как было. Срочно.
- «Нормируем» все слова, заменяя ё на е.
- Не только ё **ß->ss ç -> c ğ -> g ı->i ş -> s é->e è->e ï->i î->i**

СПАСИБО!

Локшин Марк

Старший программист

m.lokshin@corp.mail.ru

@mail.ru®

www.mail.ru