

# AIOPG vs ASYNCPG

Email: [virmir49@gmail.com](mailto:virmir49@gmail.com)

Telegram: @virmir49

Алексей Фирсов

# Очень коротко обо мне

Head of Python Department в S7 TechLab.

TeamLead / TechLead в продукте речевой и текстовой аналитики для быстрого реагирования ботом на проблему клиента S7 Airlines.

# S7 TechLab

Мы агрегируем компетенции в областях анализа данных, машинного обучения, блокчейн-технологий, разработки и вопросах инженерии данных.

# Сравнение AIOPG vs ASYNCPG



# На самом деле

## Стек aiorg

- aiorg
- psycopg
- libpq
- postgresql

# На самом деле

## Стек aiorg

- aiorg
- psycopg
- libpq
- postgresql

## Стек asyncpg

- asyncpg
- postgresql

# Что обычно говорят

## AIOPG

- Текстовый протокол
- Работает с pgbouncer
- Нет prepared statement

## ASYNCPG

- Бинарный протокол
- “Не работает” с pgbouncer
- prepared statement

# Давайте тогда так

## Стек aiorg

- aiorg
- psycopg
- libpq
- pgbouncer
- postgresql

## Стек asyncpg

- asyncpg
- pgbouncer
- postgresql



# flow postgresql

Простой запрос

<https://goo.gl/aqhnRE>



Расширенный  
запрос

<https://goo.gl/n2Lqac>



# Простой запрос

## Query:

**Byte1 - Q** Указывает, что это простой запрос.

**Int32** - Длина сообщения в байтах.

**String** - Строка запроса.

# Простой запрос пример

```
1  INSERT INTO t1 (id) VALUES(1);  
2  SELECT * FROM t1 WHERE id = 1;  
3  INSERT INTO t2 (id) VALUES(2);  
4
```

# Расширенный запрос

- Подготовленные операторы (Parse and OID)
- Порталы (Describe)\*
- Bind
- Execute
- Sync

# Расширенный запрос

- Подготовленные операторы (Parse and OID)
- Порталы (Describe)\*
- Bind
- Execute
- Sync

ЭТО НЕ  
ВСЕ.

# Parse (P) и OID type

- Parse - передает запрос строкой
- Параметры имеют свой формат
- Желательно указывать типы (OID)
- Типы могут определяться автоматически

# Пример Parse (P) и OID type

1 `SELECT * FROM t1 WHERE x = $1::bigint;`

2

3

.....

4

5 `SELECT * FROM t1 WHERE x = $1::boolean;`

6

# Порталы и Describe (D)

- SELECT и открытый курсор
- WITH в python.
- D - задать имя портала\*



# Bind - Параметры к запросу

- Текстовый формат параметров
- Бинарный формат параметров
- Формат ответа -  
бинарный/текстовый
- Формат ответа выбирается неявно.

# Execute и Sync

Execute - запрос готов, нужно  
ВЫПОЛНИТЬ.

Sync - Завершает сессию  
BEGIN/COMMIT.

# Кэшируемый запрос

- Prepare( Parse/Describe/Sync )
- Execute( Bind(D)/Execute(D)/Sync )
- Execute( Bind(D)/Execute(D)/Sync )

# Обычный запрос

- Execute(Parse/Bind/Execute/Sync)
- Execute(Parse/Bind/Execute/Sync)
- Execute(Parse/Bind/Execute/Sync)

# Итог по postgresql

- Простой текстовый запрос
- Расширенный flow с/без кэша
- Расширенный flow - запрос string
- Расширенный flow - парам.  
text/binary

# pgbouncer

Управляет пулом соединений к PostgreSQL

- session - по сути прокся
- transaction - после завершения тран.
- statement - после завершения запроса.

# pgbouncer - session

Умеет работать с prepared statement.

**Нет особых преимуществ**

Хочется transaction

statement/transaction - prepared

**Невозможно**



# transaction

- Prepare( Parse/Describe/Sync ) **time**

---

- Execute( Bind(D)/Execute(D)/Sync )

```
/*  
 * extended protocol allows server (and thus pooler)  
 * to buffer packets until sync or flush is sent by client  
 */
```

```
case 'P':          /* Parse */  
case 'E':          /* Execute */  
case 'C':          /* Close */  
case 'B':          /* Bind */  
case 'D':          /* Describe */  
case 'd':          /* CopyData(F/B) */  
    break;
```

<https://goo.gl/eLtNH1>



# statement/transaction

- Execute(Parse/Bind/Execute/Sync)
- Execute(Parse/Bind/Execute/Sync)

Работает!

# asyncpg - не работает так

Холива



<https://goo.gl/C2gUzB>

Asyncpg-форк



<https://goo.gl/v21JQn>

# libpq

Интерфейс PostgreSQL для  
программирования приложений на  
языке C

Execute(Parse/Bind/Describe/Execute/Sync)

```
static int send_query_bigdata(DbConn *db)
```

```
{
```

```
    const char *values[1];
```

```
    int lengths[1];
```

```
    int fmts[1];
```

```
    int arglen;
```

```
    char *q = "select $1::text";
```

```
    arglen = random() % bulk_data_max;
```

```
    db->_arglen = arglen;
```

```
    values[0] = bulk_data + bulk_data_max - arglen;
```

```
    lengths[0] = arglen;
```

```
    fmts[0] = 1;
```

```
    return PQsendQueryParams(db->con, q, 1, NULL, values, lengths, fmts, 1);
```

```
}
```

<https://goo.gl/mGQh7>



# Ну и вот - кажется побеждаем!



# Рsуsорг - не поддерживается!

Холивар 2010

Холивар 2013

Г



<https://goo.gl/GX3879>

Г



<https://goo.gl/XsKLcE>



# Psycorg - форк

Я еще не  
пробовал

Нет  
поддержки



<https://goo.gl/GTxPqY>

# Подводя итог

- Проблема ответственности
- Преграды на пути

# Подводя итог

- Проблема ответственности
- Преграды на пути
- Скорость - **важный** показатель, **но не главный**

# Вопросы?

- С вами был Алексей Фирсов
- Занимаюсь программированием ~ лет 10-11
- Последние ~ лет 6-8 работаю с Python
- Участник сообщества [aio-alibs](#)
- Один из основных maintainers [aio-alibs/aioapg](#)  
Email: [virmir49@gmail.com](mailto:virmir49@gmail.com)

Telegram: @virmir49