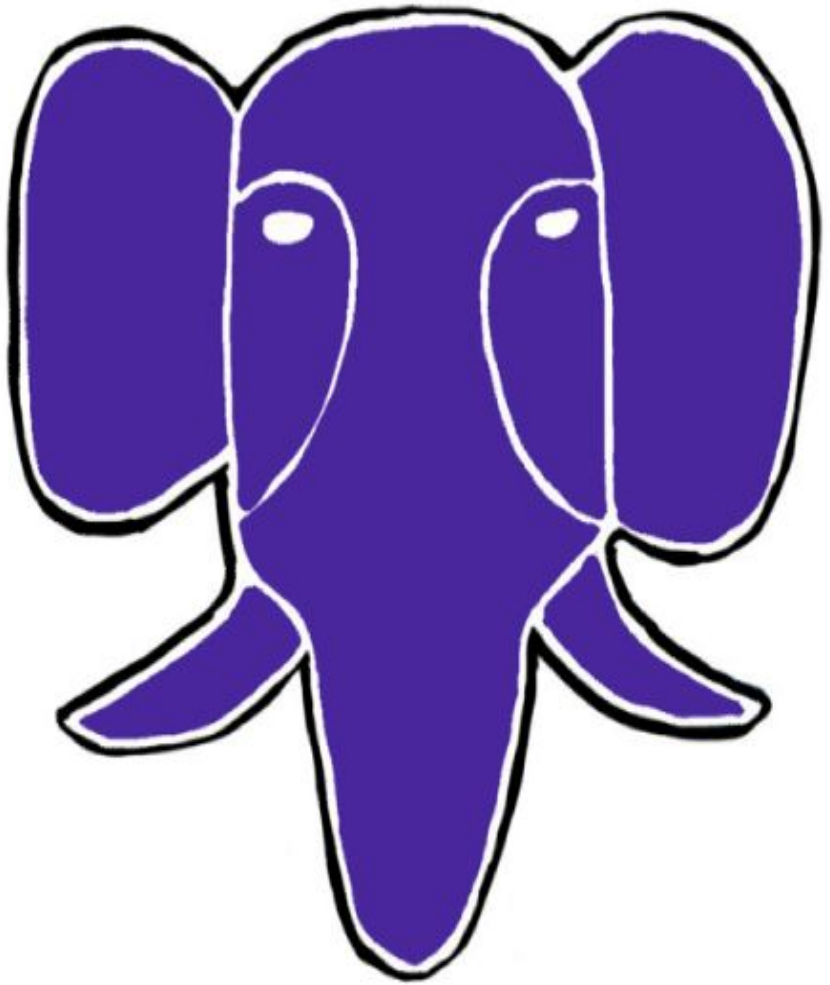
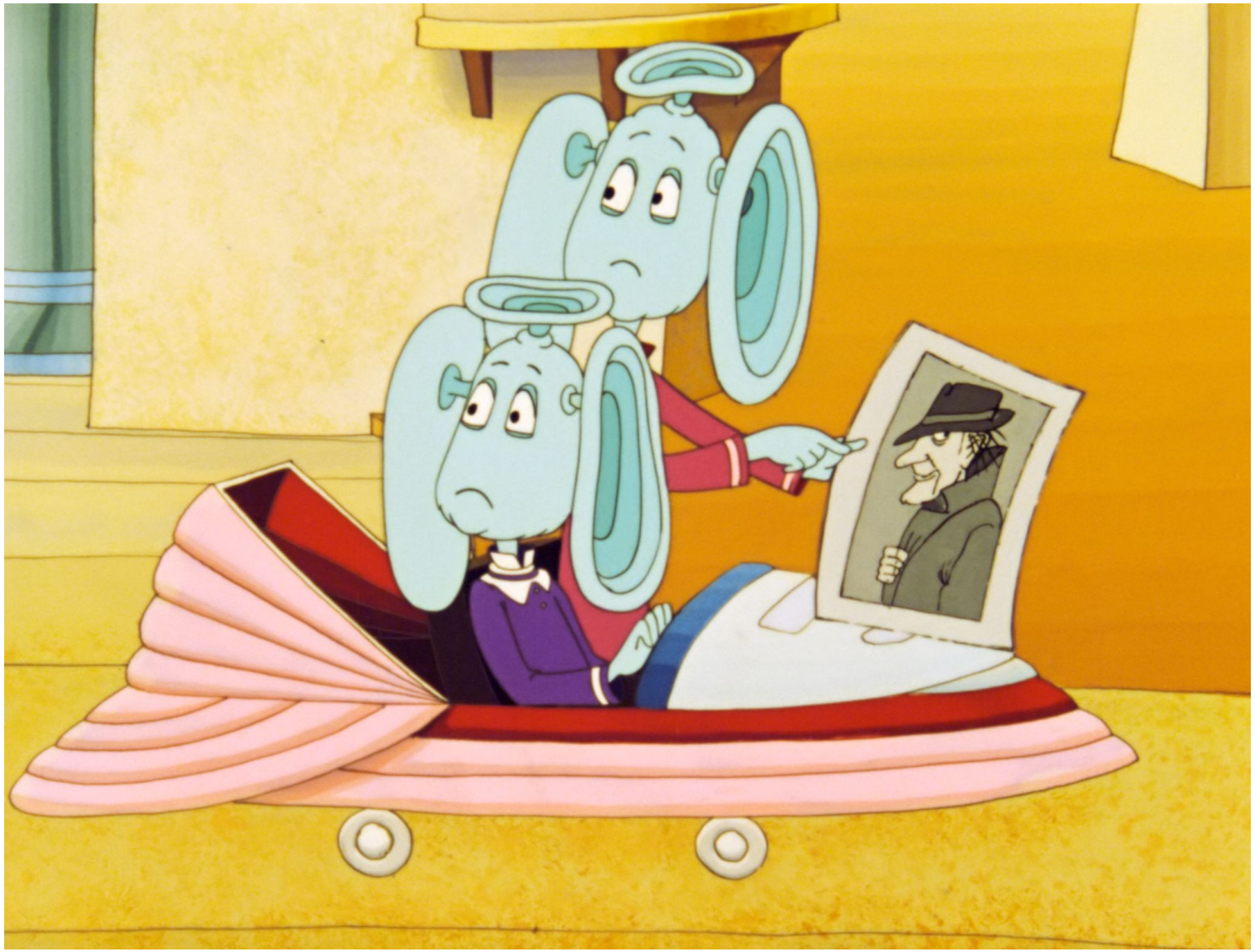
Евгений Аристов

Открытый урок
idle VS idle in transaction



https://aristov.tech



Правила вебинара

Задаем вопрос в чат

Можно общаться голосом, для этого поставьте знак ? в чат

Вопросы вижу, отвечу в момент логической паузы

Если остались вопросы, можно написать мне через сайт aristov.tech

https://aristov.tech

Маршрут вебинара

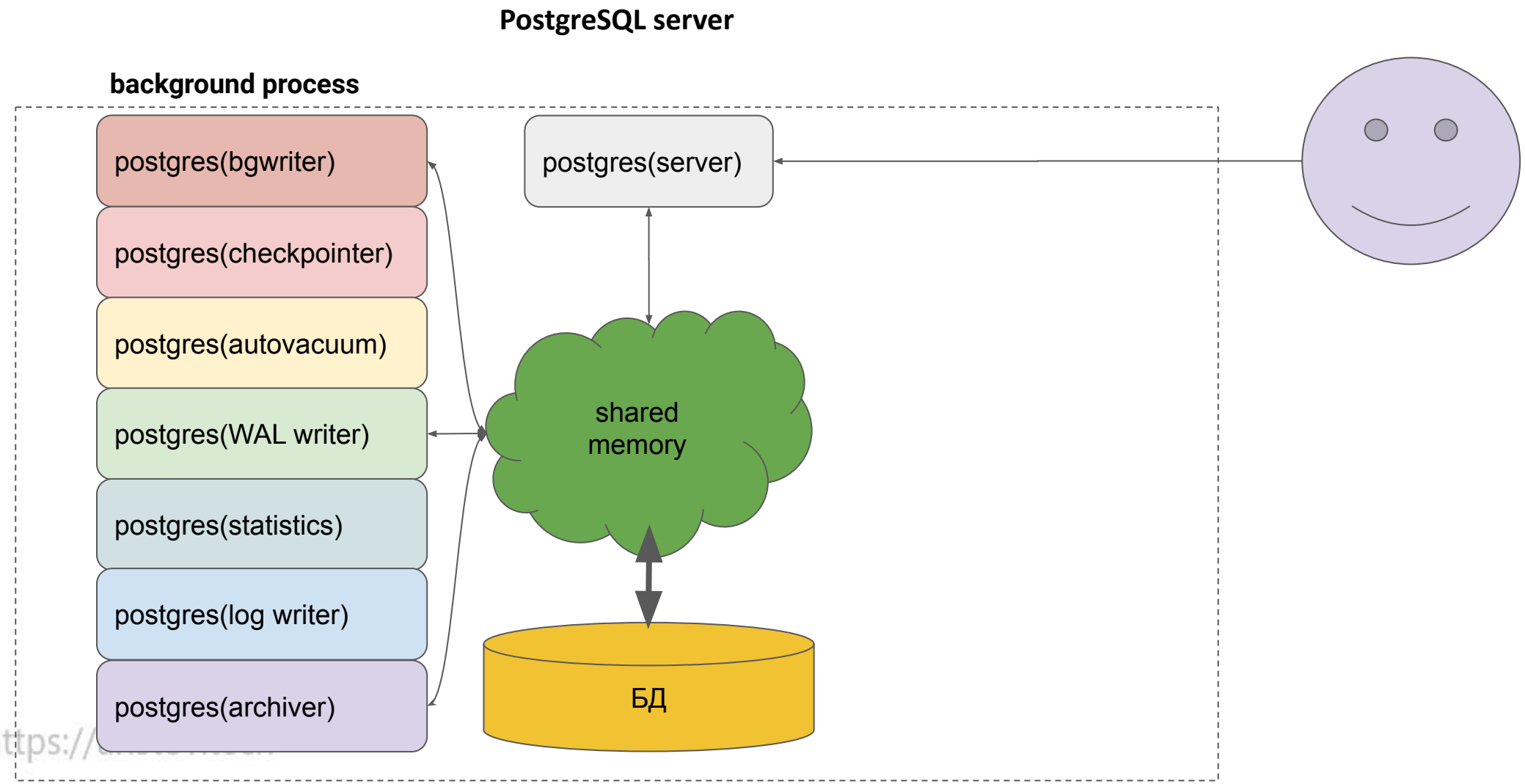
- вспоминаем работу с памятью и процессами
- проблематика долгих транзакций
- idle VS idle in transaction
- подводные камни этих вариантов
- практические исследования
- немного о проекте aristov.tech
- розыгрыш ***новой книги*** и скидки на курс

Транзакции и работа с памятью

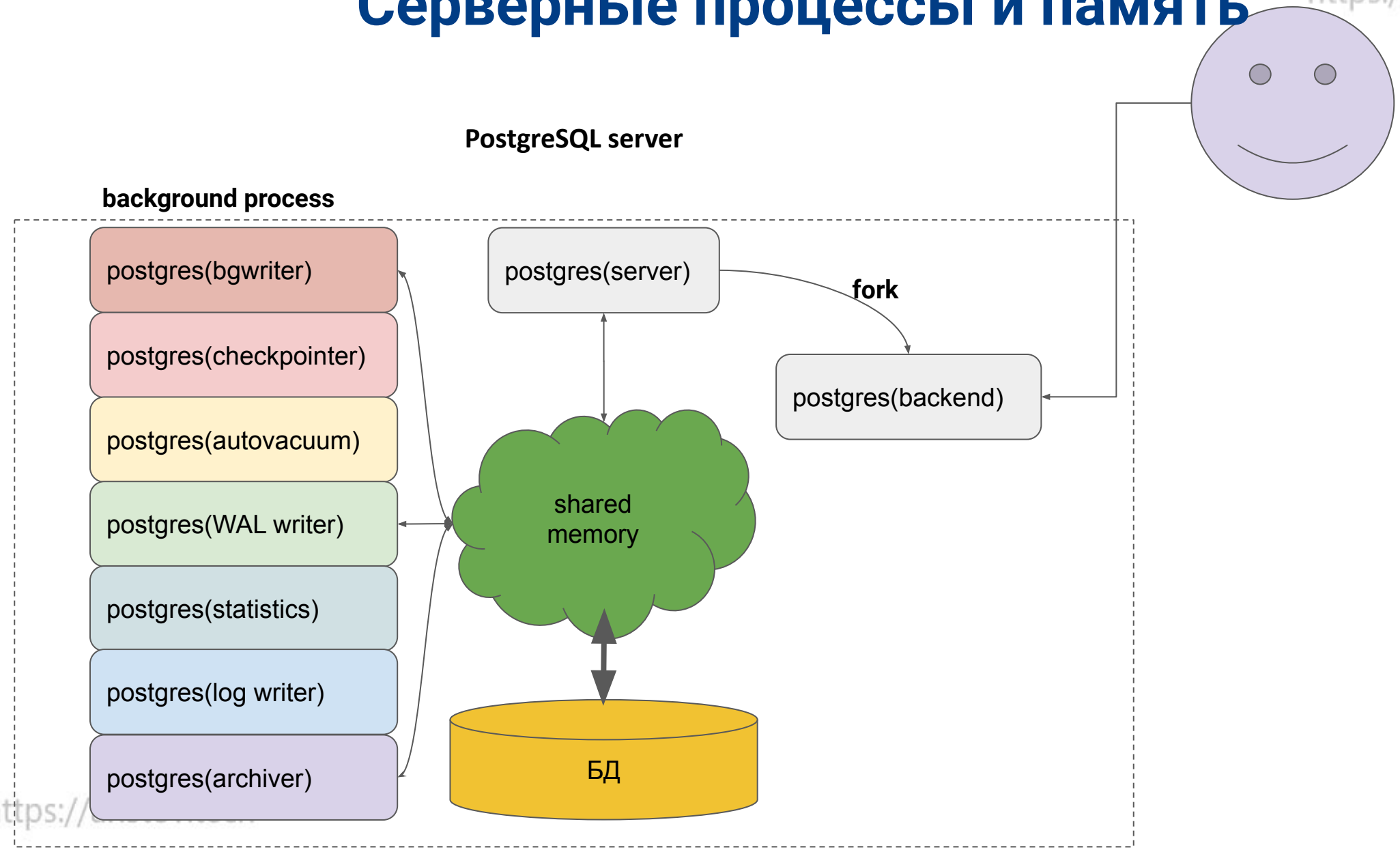
https://aristov.tech

Серверные процессы и память

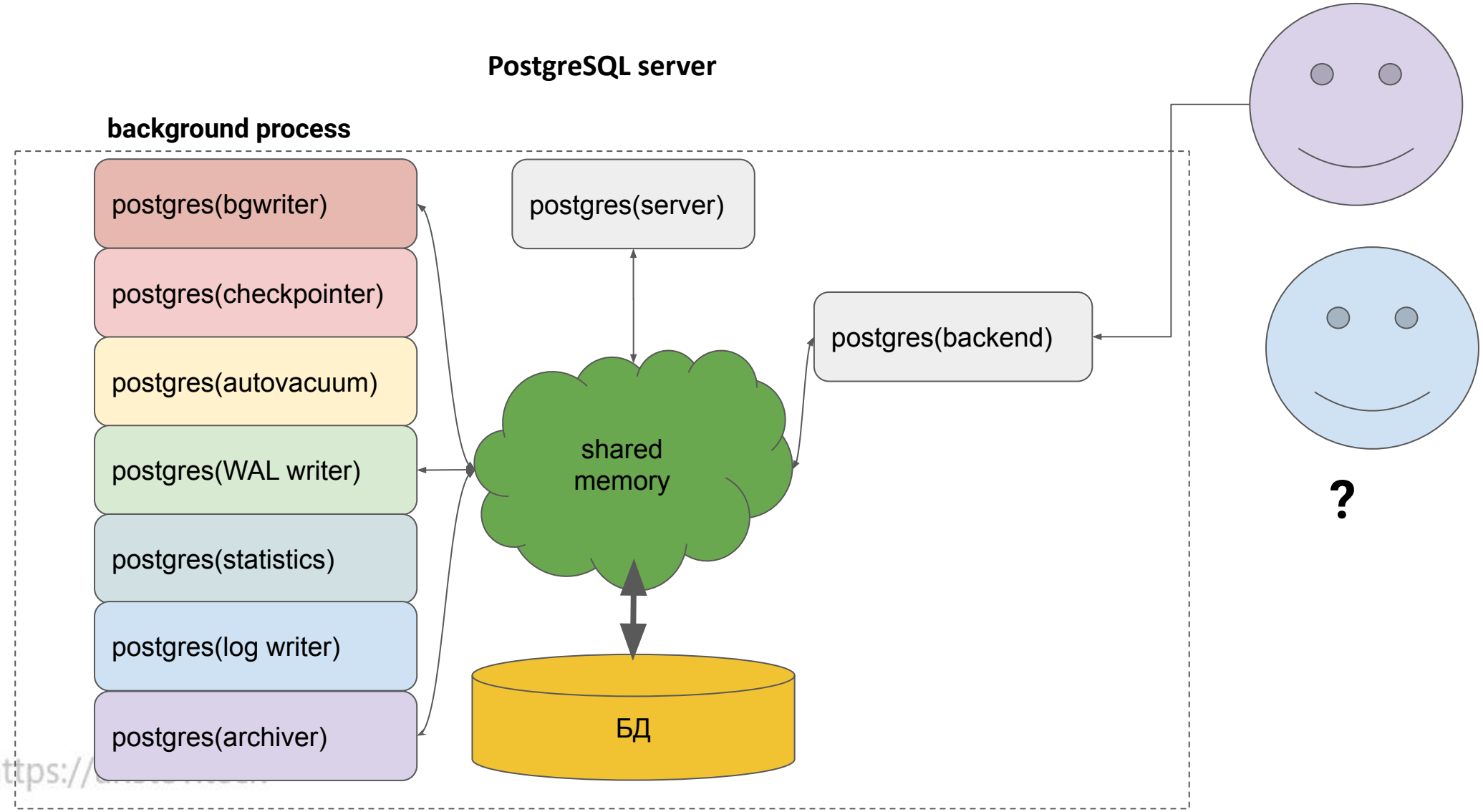
?



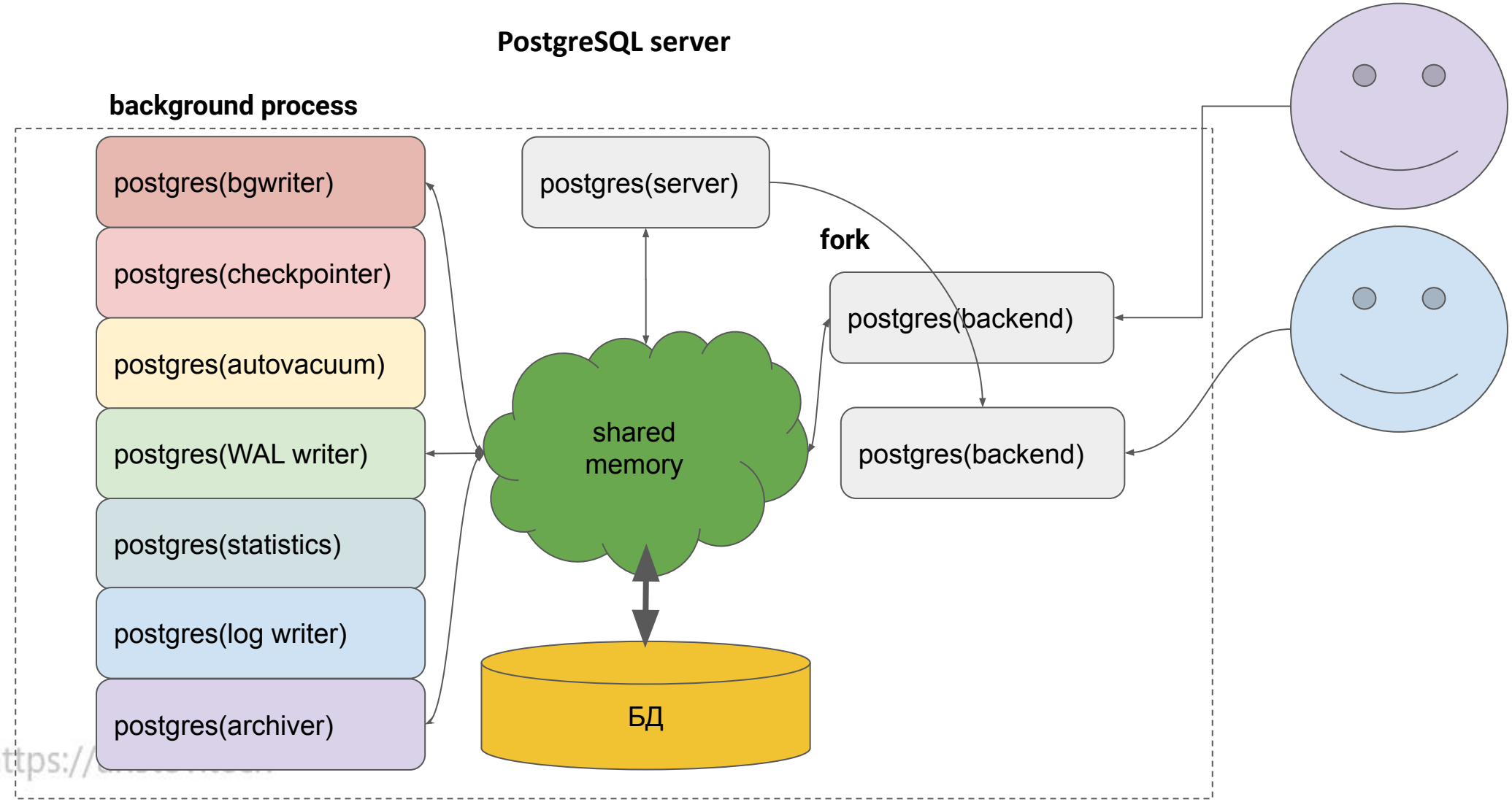
Серверные процессы и память



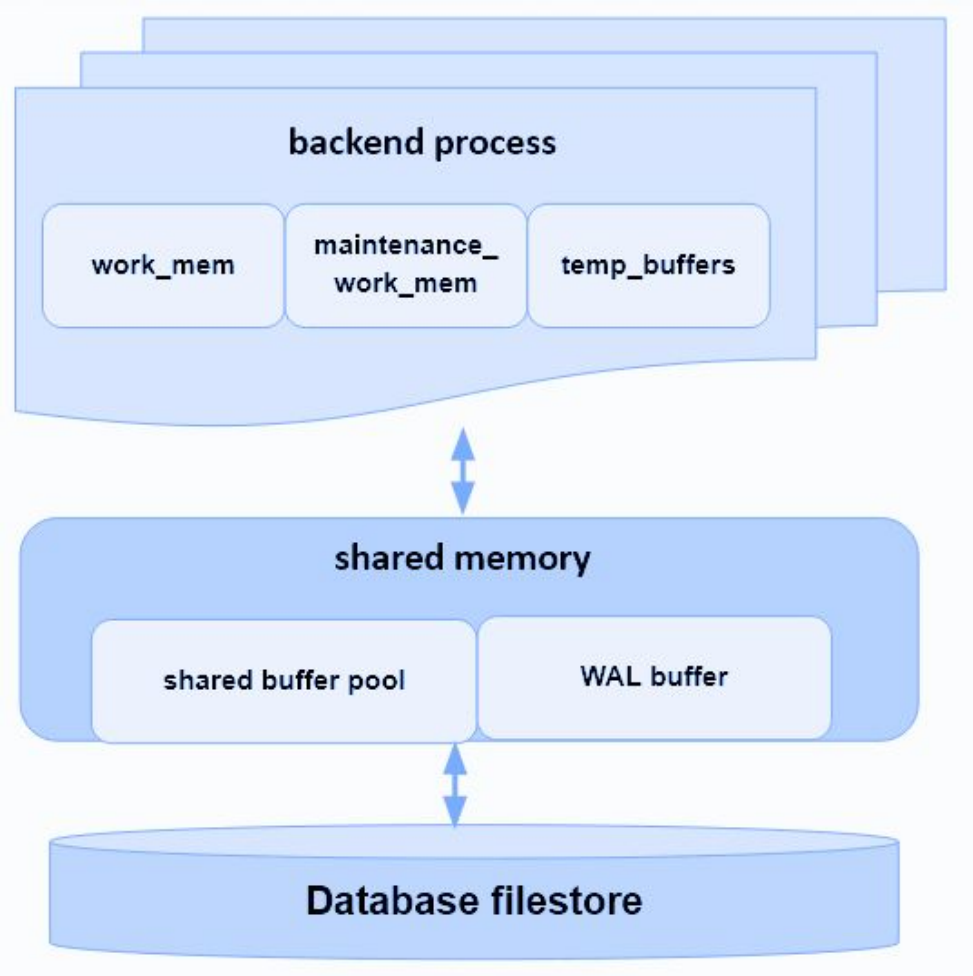
Серверные процессы и память



Серверные процессы и память



Кроме это выделяется память для каждой сессии



- принадлежит **КАЖДОМУ** backend процессу
- **work_mem (4 MB)**

  эта память используется на этапе выполнения запроса
- **maintenance_work_mem (64MB)**

  используется служебными операциями типа VACUUM и REINDEX
- **temp_buffers (8 MB)**

  используется на этапе выполнения для хранения временных таблиц

Нюансы

- принадлежит КАЖДОМУ backend процессу
- **work_mem (4 MB)**

  эта память используется на этапе выполнения запроса для сортировок строк, например ORDER BY и DISTINCT - выделяться может неоднократно!!!

  https://www.postgresql.org/docs/current/runtime-config-resource.html
- **maintenance_work_mem (64MB)**

  используется служебными операциями типа VACUUM и REINDEX

  ➢ **выделяется только** при использовании команд обслуживания в сессии
- **temp_buffers (8 MB)** используется на этапе выполнения для хранения временных таблиц -

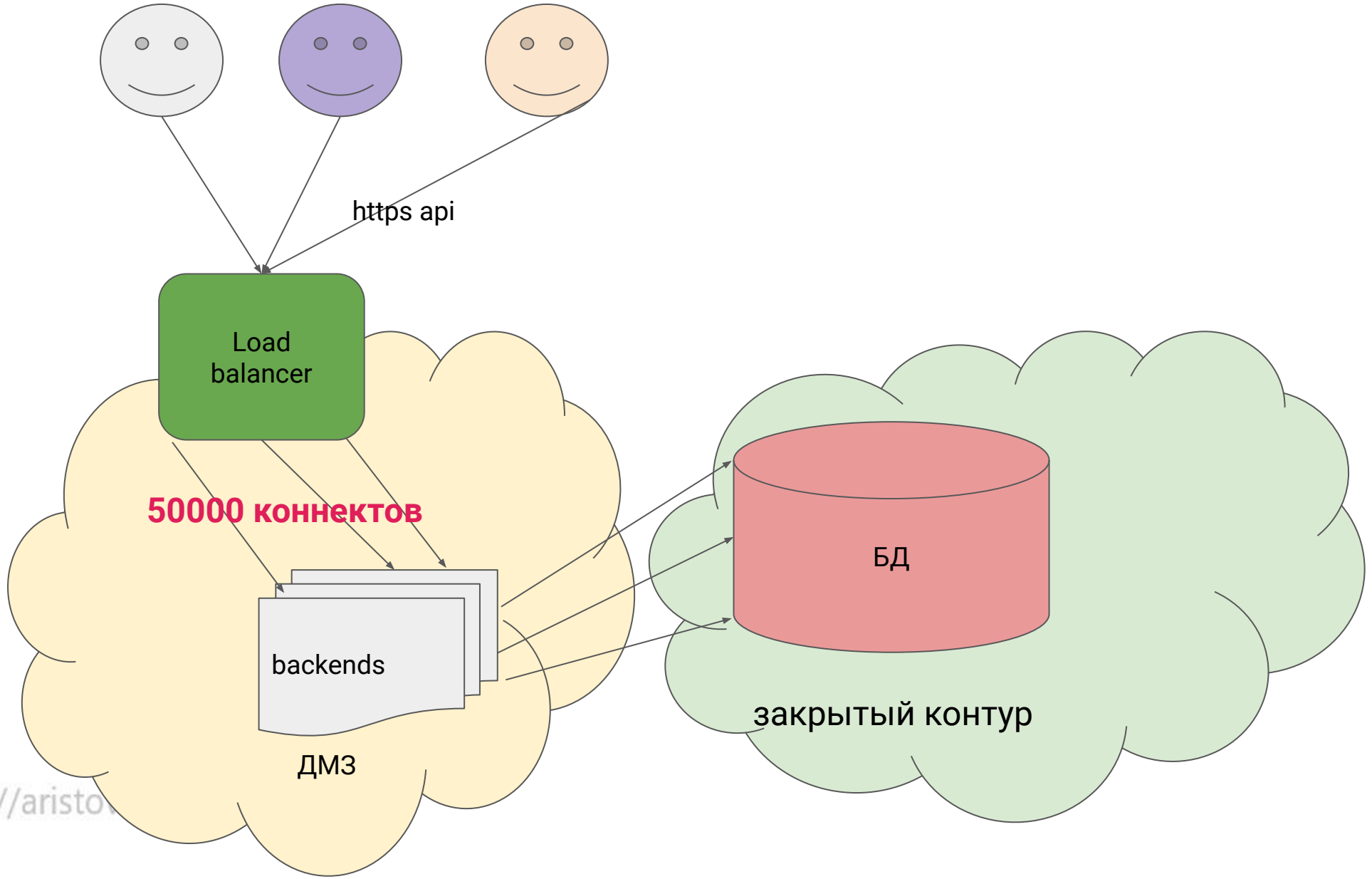
  При превышении **work_mem** или **temp_buffers** - дальше идем temp tablespace.

  Разберем проблематику на курсе по оптимизации

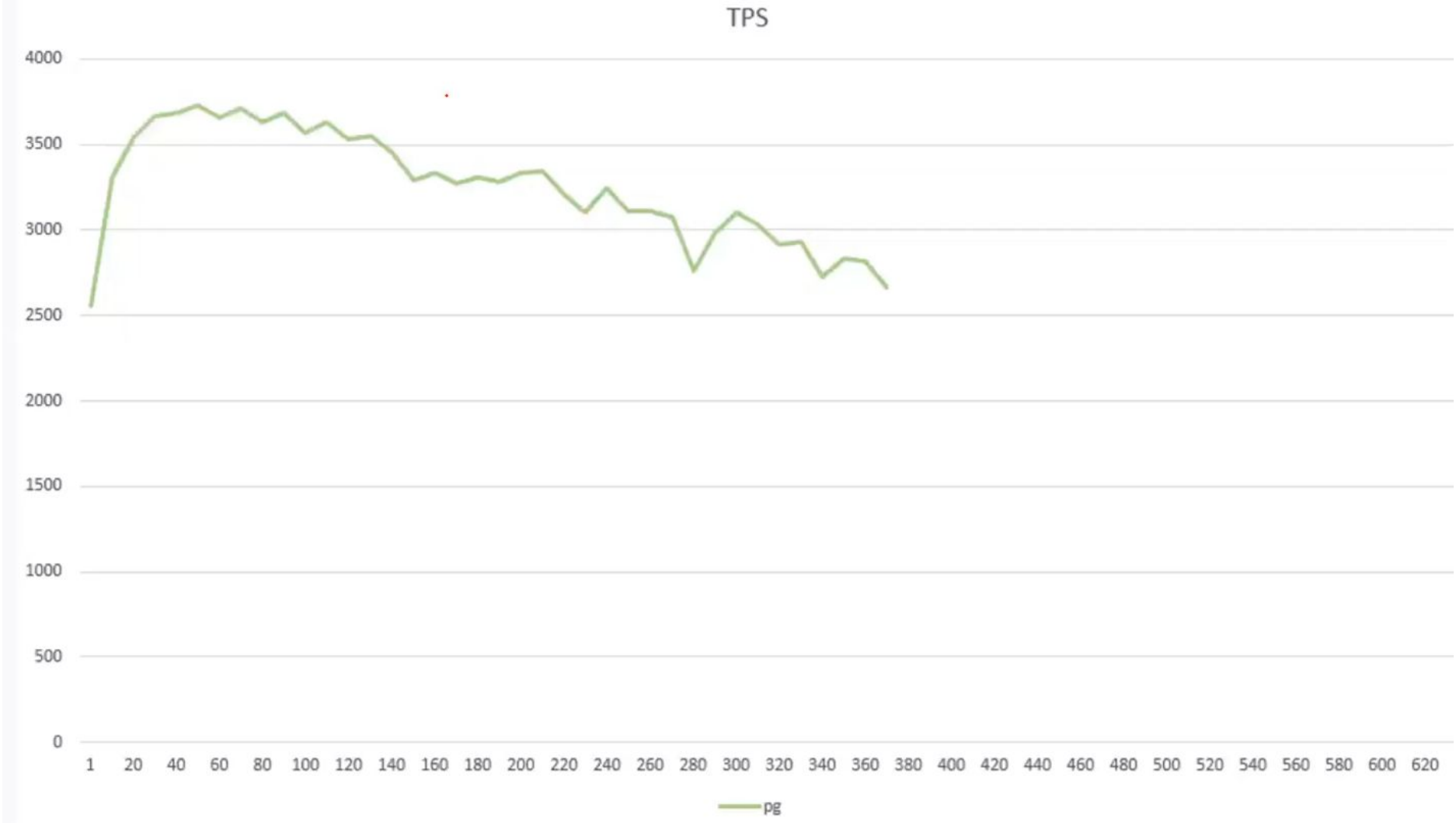
Connecting

https://aristov.tech

Чем плохо много коннектов?

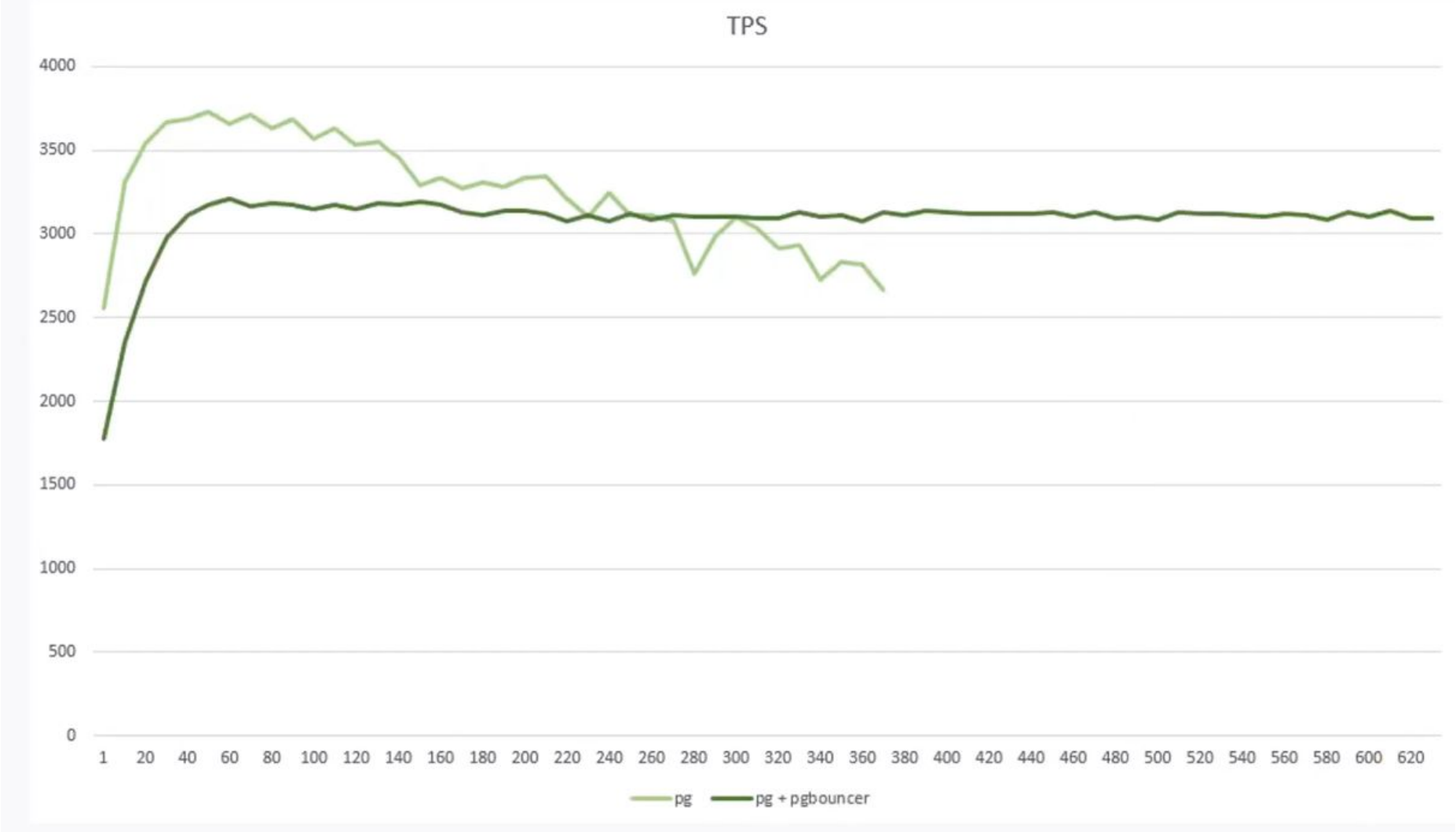


600++ conn



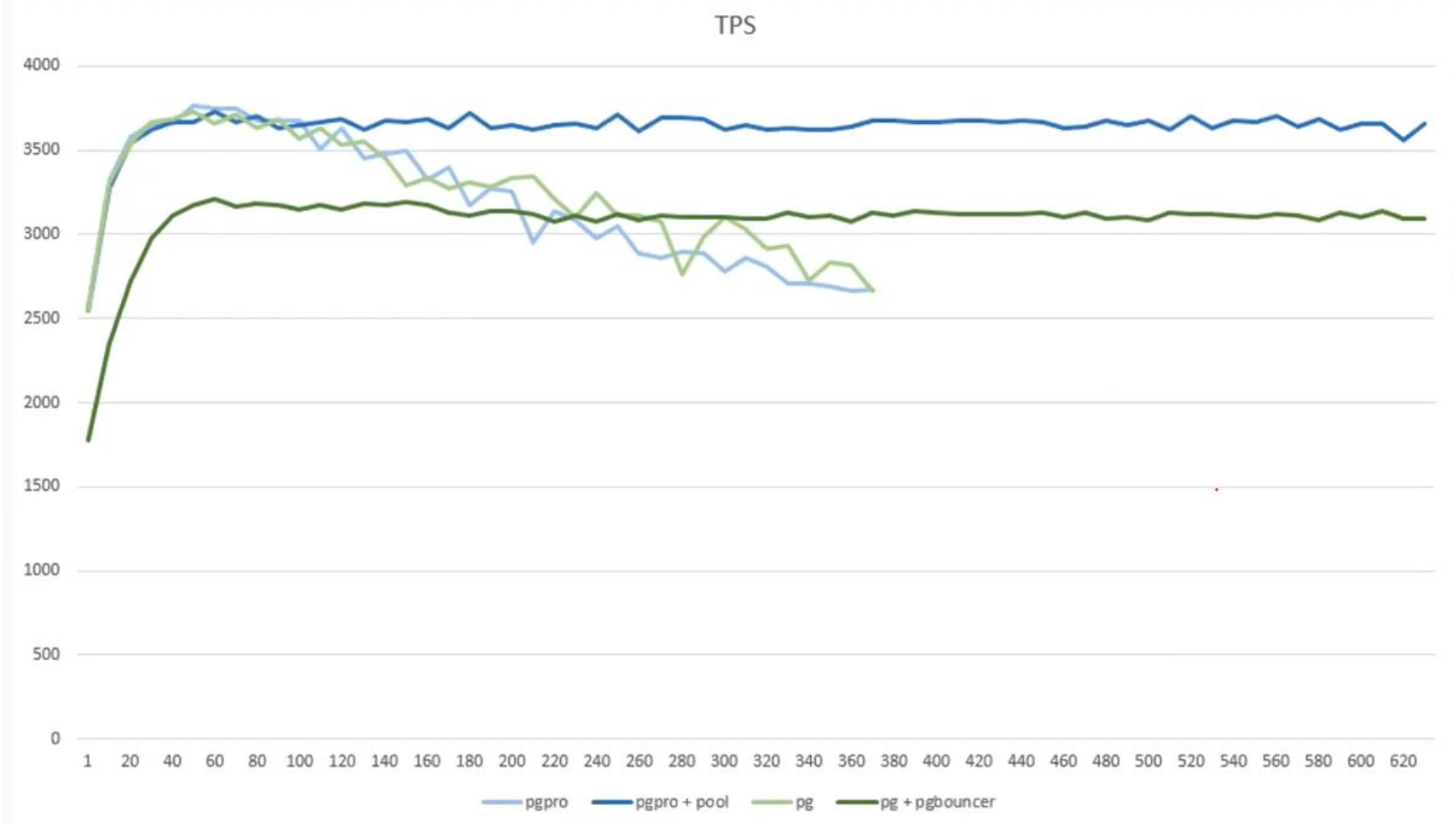
https://aristov.tech

600++ conn



https://aristov.tech

600++ conn



https://aristov.tech

https://postgrespro.ru/docs/enterprise/16/connection-pooler-configuration

Проблемы:

1. коннект в Постгресе очень дорог
2. без пулконнектора (pgbouncer, pgpool-II,pgagroal,odyssey) жить тяжело - но необходимо учитывать overhead и особенности каждого пулера
3. возникает идея - если держать постоянные коннекты (не тратить время на fork процессов)?

idle VS idle in transaction

https://aristov.tech

idle VS idle in transaction

Список процессов:

SELECT * from pg_stat_activity;

Наши ситуации на сегодня:

idle - есть коннект, транзакций нет

Создание транзакций внутри коннекта тоже **не бесплатно**.

Существует аффект при ручном создании транзакций (не AUTOCOMMIT) - падение производительности до 2х раз на коротких SELECT.

Решение - создание долгих транзакций, а внутри уже не зависящие друг от друга запросы (oracle way).

Проблему решает включенный AUTOCOMMIT и виртуальные транзакции (подробные исследования на курсе, а также разница между COMMIT vs ROLLBACK)

idle VS idle in transaction

idle in transaction - транзакция идёт, но висит и ждёт чего-то (блокировки) /кого-то (вручную держим открытую транзакцию).

При этом:

- держит блокировки: select * from pg_locks;
- память занята под work_mem + temp_buffers

idle VS idle in transaction

А что же может пойти не так? Спросим у экспертов:

https://dba.stackexchange.com/questions/332402/is-harmfulness-of-idle-in-transaction-connections-a-myth

Database has long running idle in transaction connection - Amazon Aurora

> A transaction in the idle in transaction state can hold locks that block other queries. **It can also prevent VACUUM (including autovacuum) from cleaning up dead rows, leading to index or table bloat or transaction ID wraparound**.

https://www.cybertec-postgresql.com/en/idle_in_transaction_session_timeout-terminating-idle-transactions-in-postgresql/

> A long transaction is actually not a problem – the problem starts if a long transaction and many small changes have to exist. **Remember: The long transaction can cause VACUUM to not clean out your dead rows**.

https://aristov.tech

idle VS idle in transaction

[Why wouldn't VACUUM ANALYZE clear all dead tuples? - Database Administrators Stack Exchange]

> It is not really long-lived transactions, but **long lived *snapshots***. Certainly a long running select or insert statement will do that. **For isolation levels higher than read-committed, the whole transaction will retain the snapshot until it is down**, so if some opens a repeatable read transaction and then goes on vacation without committing it, that would be a problem. Hung-up prepared transactions will as well (if you don't know what a prepared transaction is, then you probably aren't using them).

Даже Павел Лузанов из PostgresPro:

https://www.cybertec-postgresql.com/en/idle_in_transaction_session_timeout-terminating-idle-transactions-in-postgresql/#comment-3846188971

> I believe that example of a long transaction is true only for Repeatable Read (or Serializable) isolation level. But by default BEGIN used Read Commited. So, after SELECT in the first session finished, VACUUM will remove dead rows in a table after subsequent UPDATE, DELETE commands in the session 2.

idle VS idle in transaction

Давайте разбираться

Caveats

Гипотезы для проверки:

1. затруднение обслуживающих процессов (vacuum)
   a. при наличии только операций чтения
   b. при наличии пишущих транзакций
   c. а если повысить уровень изоляции транзакций?
2. что с таблицей блокировок и pg_stat_activity?
3. повышенное потребление памяти?
4. снижение производительности?
5. освобождается ли work_mem после завершения запроса? транзакции?

Практика

https://aristov.tech

VACUUM blocks

The transaction holds a ***snapshot*** of the database. A snapshot is a data structure that determines which other transactions are visible to a certain transactions. Snapshots are held open

a. as long as an SQL statement is running (so a long running query can block VACUUM progress)

b. while there is a cursor open

c. on the REPEATABLE READ or SERIALIZABLE isolation level, for the whole duration of the transaction

https://www.cybertec-postgresql.com/en/reasons-why-vacuum-wont-remove-dead-rows/

Не забываем про проблему Transaction ID wraparound - заморозка и всё с этим связанное

Что ещё

https://aristov.tech

Caveats

Параметр old_snapshot_threshold = -1

В теории должен БЫЛ решить проблему со старыми снепшотами при долгих транзакциях, но все как обычно:
https://www.postgresql.org/message-id/20230213204507.b7k3fiorgwrahsjx%40awork3.anarazel.de

Проблему признали, но так никто и не пофиксил (обещают в 17 версии)

Исходный код с долгой обработкой:

https://github.com/postgres/postgres/blob/REL_13_STABLE/src/backend/utils/time/snapmgr.c#L1808C33-L1808C33

Пример из жизни: 100+ ядер и долгие транзакции - до 5 раз падение производительности

Итоги:

Вопреки мнению большинства экспертов долгие транзакции могут сэкономить время на создание транзакций и при этом не мешать обслуживающим процессам, но:

a. выделение постоянной памяти (work_mem + temp_buffers)
b. разросшаяся таблица pg_stat_activity + pg_locks
c. работает для уровня изоляции read committed
d. периодически необходимо пересоздавать транзакцию
e. при обрыве соединения, если были не только читающие запросы - можем потерять изменения - всё таки лучше пишущие транзакции не делать длинными
    i. плюс будут мешать другим транзакциям
    ii. плюс необходимо учитывать области видимости незавершенных транзакций!

Это всё необходимо учитывать, если вы решили пойти по пути долгих транзакций!!

А есть ли плюсы, кроме эфемерной экономии на создании транзакции?

Моя рекомендация - объединять в транзакции логически зависимые апдейты, а селекты доверить автокоммиту и виртуальным транзакциям.

Проект aristov.tech

aristov.tech

Книга по 14 Постгресу/ по оптимизации 16 Постгреса (18 мая) https://aristov.tech/#orderbook
Эксклюзивный **курс** по Оптимизации Постгреса 3 поток
https://aristov.tech/blog/kurs-po-optimizaczii-postgresql-2-0/
Со всеми **отзывами** без цензуры можно ознакомится по ссылке
https://aristov.tech/blog/otzivi-kurs/
Блог с популярными темами https://aristov.tech/blog
ТГ **канал** с новостями блога и проекта https://t.me/aristov_tech
Ютуб канал с интересными видео https://www.youtube.com/@aristovtech
Моя **группа** ДБА https://t.me/dbaristov
Курс SQL с 0 до джуна https://aristov.tech/blog/kurs-sql-c-0/

Также за прошедшее время был запущен проект **менторинга** в котором уже участвует 15 лучших экспертов в отрасли ДБА/DevOps и разработки.
Ознакомится с преподавателями и проектом https://aristov.tech/mentorship
В направлении крутых **вакансий** пока 4 предложений от партнёров с доходом до **600!**
https://aristov.tech/blog/vakansii-ot-partnerov-aristov-tech/
Всегда можно со мной связаться через сайт для консультация, аудита вашего проекта, менторинга, обучения и многого другого

PostgreSQL 16: лучшие практики оптимизации

PostgreSQL 16: лучшие практики оптимизации

18 мая авторский вечер в Москве. Время и место будут позже объявлены в группе

https://github.com/aeuge/postgres16book/tree/main

aristov.tech

Розыгрыш *новой* книги и скидки 30% на курс (только очные варианты)

регистрируемся по форме:

https://forms.gle/BamQm6Q94WDD5ypVA

Сам розыгрыш в db-fiddle

https://www.db-fiddle.com/f/43wbuUzv7YUAcS3aypkvvC/1

https://aristov.tech

ДЗ

https://aristov.tech

ДЗ

1. Подписаться на канал
2. Подписаться на Ютуб
3. Вступить в мою группу ДБА
4. Посещать открытые уроки
5. Прокачаться с помощью менторинга
6. Прийти на курс %)
7. Устроиться на одну из вакансий 450+

Спасибо за внимание!

Следующий открытый урок в июне. Подробнее информация о дате и времени проведения, теме на моём сайте https://aristov.tech

Видео ОУ на ютуб, материалы занятия загружу после обработки и опубликую в своём новостном телеграм канале https://t.me/aristov_tech

Аристов Евгений