Метод параллельного выполнения запросов к системе управления базами данных PostgreSQL в пределах одного соединения

Студент: Платонова Ольга Сергеевна

Группа: ИУ7-85Б

Руководитель: Филиппов Михаил Владимирович,
к.т.н., доцент кафедры ИУ-7

Консультант: Гаврилова Юлия Михайловна

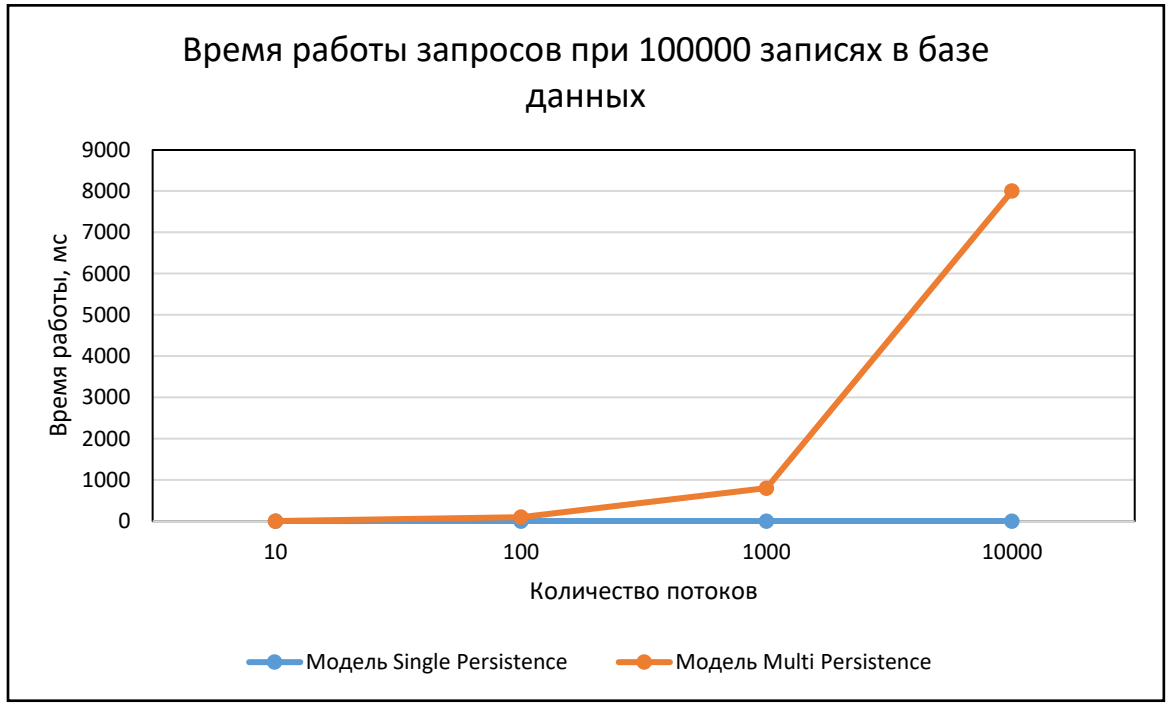
Цель и задачи работы

Цель — разработать метод параллельного выполнения запросов к СУБД PostgreSQL в пределах одного соединения.

Задачи:

- Выполнить анализ предметной области и существующих методов выполнения запросов в MPP системах
- Разработать метод параллельного выполнения запросов к СУБД PostgreSQL в пределах одного соединения
- Реализовать программный модуль для СУБД PostgreSQL
- Выполнить сравнительный анализ стандартных методов обработки запросов к СУБД PostgreSQL с реализуемым методом.

Формализация задачи



Доступ к БД объемом 100.000 записей:

- многопоточная программа примерно в 1000 раз работает быстрее;
- однопоточная программа показывает нестабильную работу на больших данных.

Операция подключения — одна из самых дорогостоящих (процесс подключения к БД занимает от 2 до 3 МБ).

Анализ предметной области

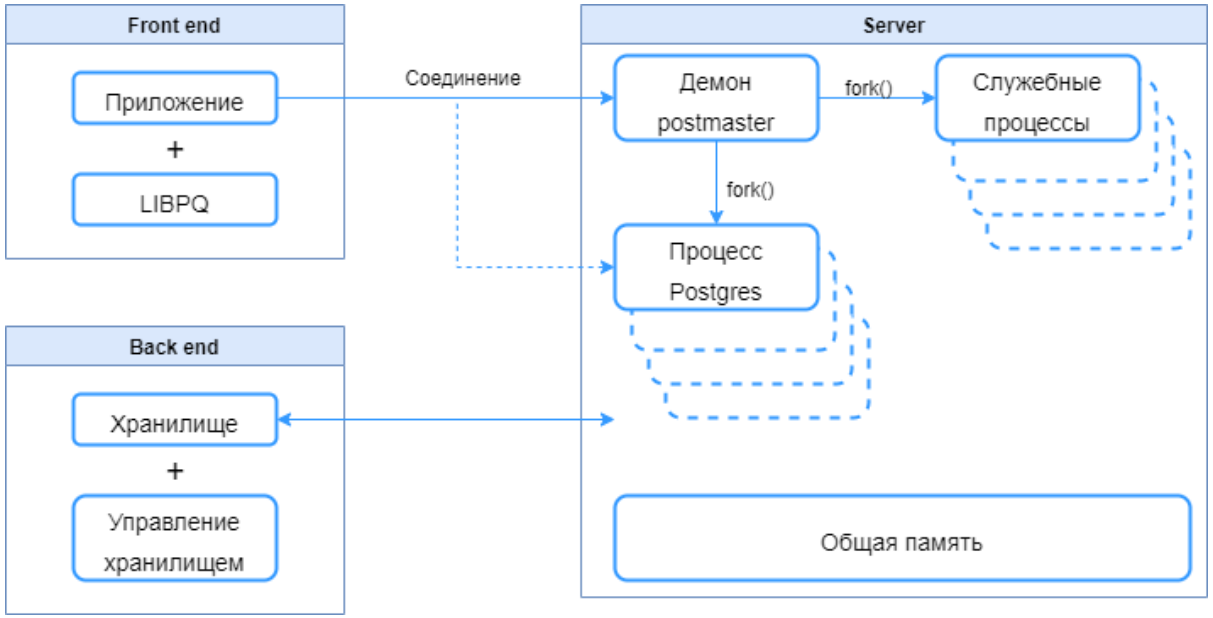
PostgreSQL (14.2):

- доступность исходного кода;
- кроссплатформенность.

Рейтинг	СУБД	Модель БД
1.	Oracle	Реляционная
2.	MySQL	Реляционная
3.	Microsoft SQL Server	Реляционная
4.	PostgreSQL	Реляционная
5.	MongoDB	Документная
6.	Redis	«Ключ-значение»
7.	IBM Db2	Реляционная
8.	Elasticsearch	Поисковая система
9.	Microsoft Access	Реляционная
10.	SQLite	Реляционная

«Два потока не должны пытаться одновременно работать с одним объектом PGconn.
В частности, не допускается параллельное выполнение команд из разных потоков через один объект соединения»

Анализ предметной области



Выделяют 3 основные подсистемы:

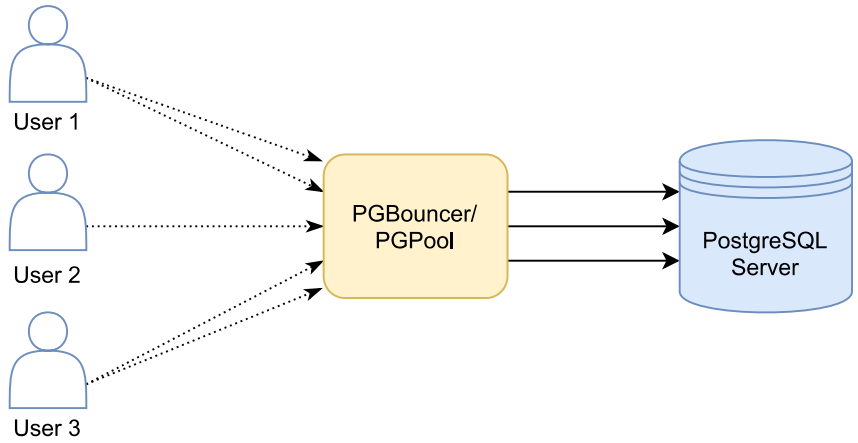
- клиентская часть
- серверная часть
- хранилище данных

Анализ существующих решений. Пул соединений

Повышение производительности, когда стоимость и скорость инициализации экземпляра высоки, а количество одновременно используемых объектов в любой момент времени является низким.

В PostgreSQL отсутствует встроенный пул соединений, однако допускается использование внешнего.

- **Пул на основе libpq**
- **Пул в качестве внешней службы**

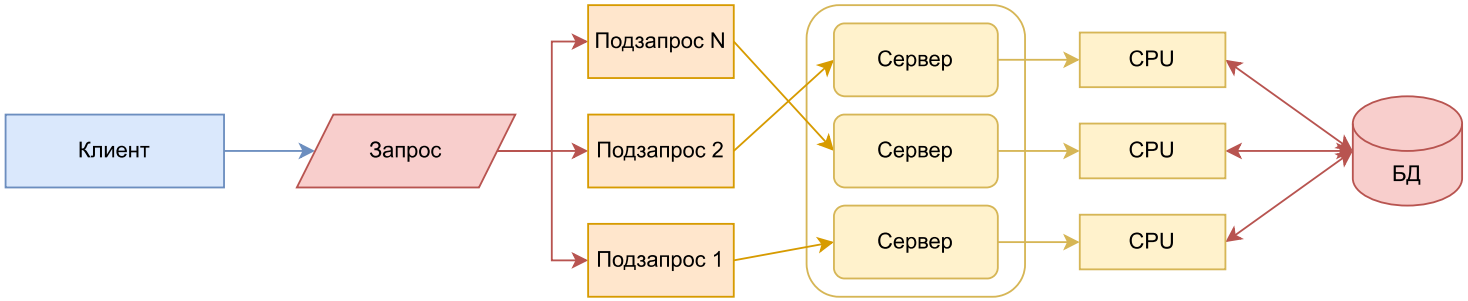


Анализ существующих решений. Пул соединений

Пул на основе libpq	Пул в качестве внешней службы	Встроенный пул
Увеличение пропускной способности транзакции до 60%		
Необходимость поддержки отдельного пула соединения для каждой БД		
Ограничение максимального количества одновременных подключений к БД		Доступен только в коммерческой версии
Затраты на разработку	Сложность конфигурации пула	
Сложность встраиваемости в код	Отсутствие кода ошибки	
	Однопоточная реализация службы	

Анализ существующих решений. Параллельное выполнение запроса

Распараллеливание — возможность построения таких планов запросов, которые будут задействовать несколько ядер.



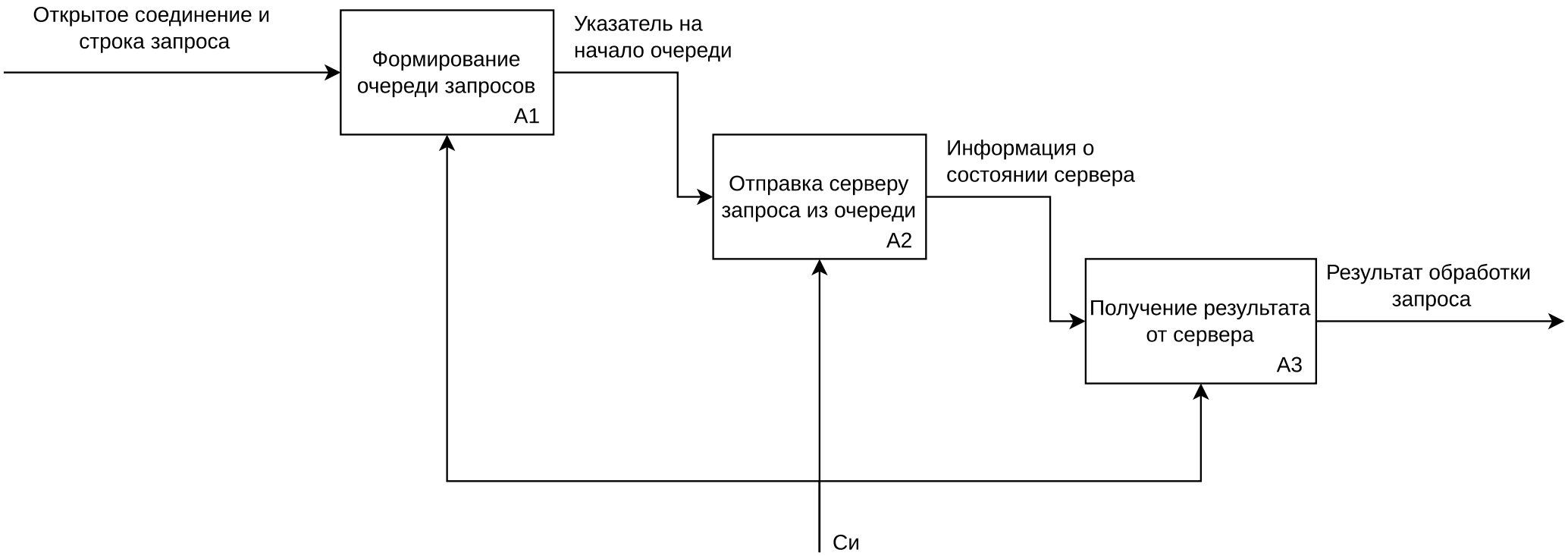
Выбор плана:

- рассмотрение всевозможных вариантов для получения одного и того же результата;
- оценка каждого варианта для выбора самого дешевого.

<u>Недостатки метода:</u>

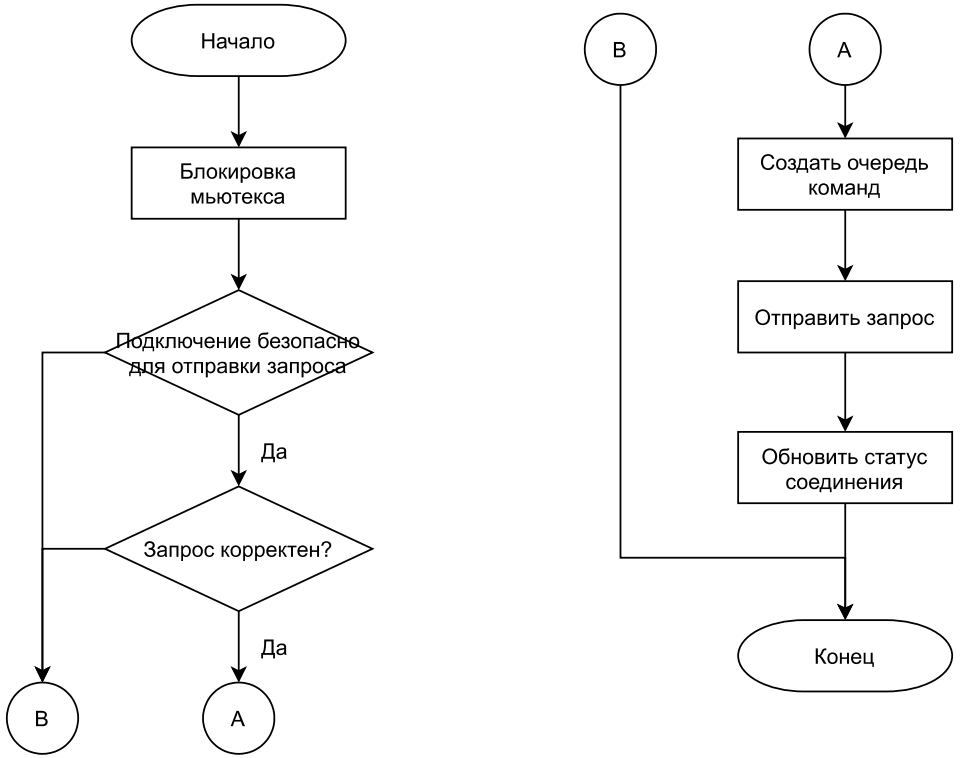
- применим к малому числу запросов;
- может быть снижена производительность.

Ключевые этапы работы метода

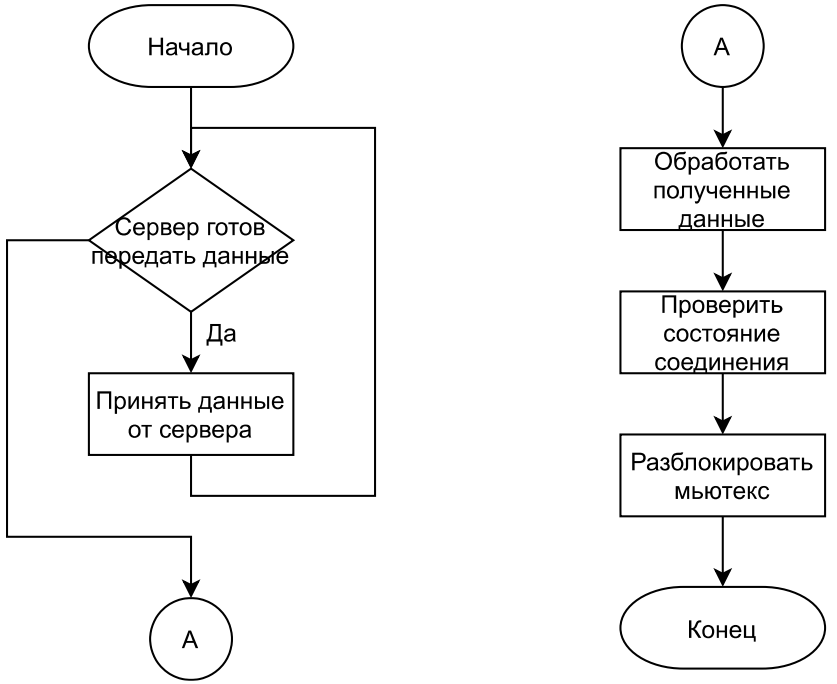


Ключевые этапы работы метода

Отправка запроса серверу



Получение результата от сервера



Внешний модуль

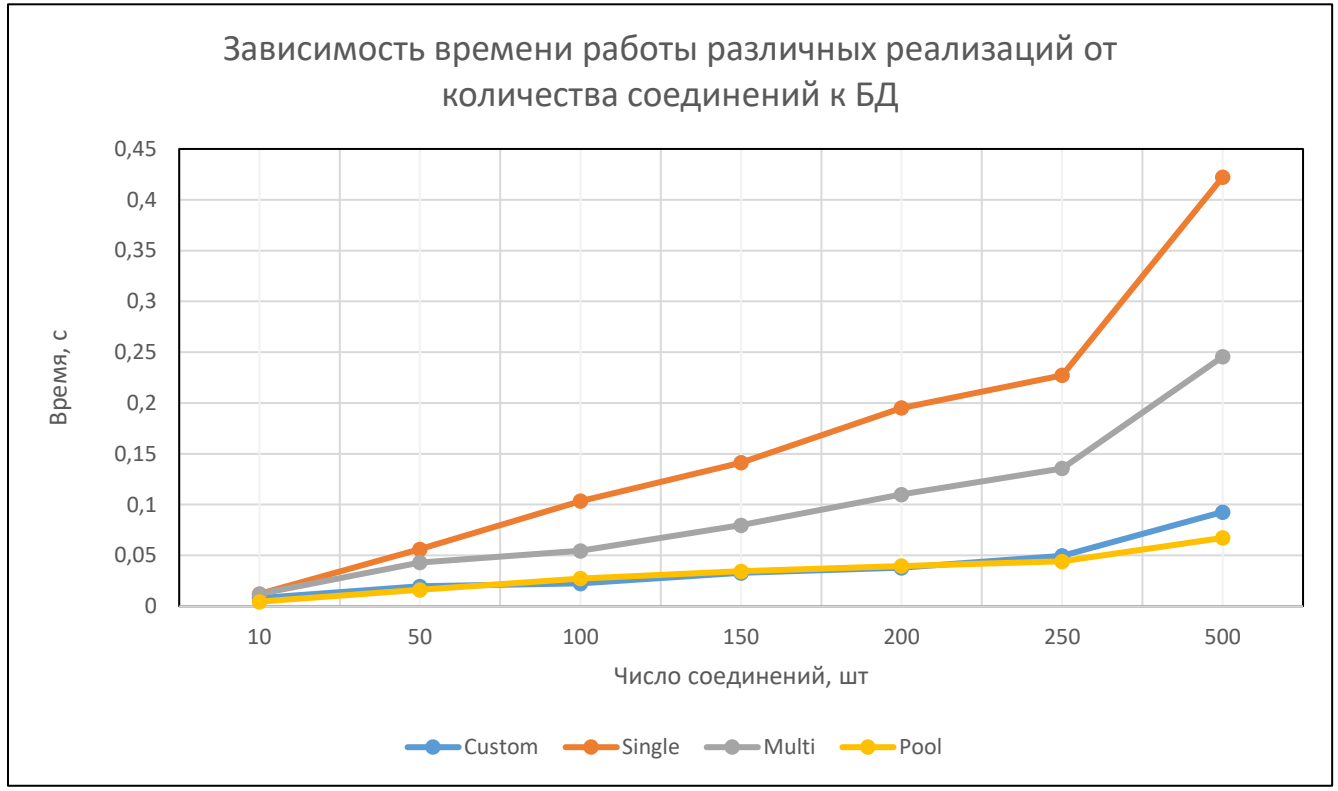
Внешний модуль, используя интерфейс командной строки, предоставляет пользователю возможность выбора запускаемой реализации.

Пользователю доступны следующие реализации:

- однопоточная;
- многопоточная;
- с использованием внешнего пула;
- с использованием разработанного метода.

Внешний пул был разработан с использованием умных указателей для предотвращения возможной утечки ресурсов. Сам пул был реализован в качестве очереди соединений: в конец добавлялись свободные соединения, работа с которыми была завершена.

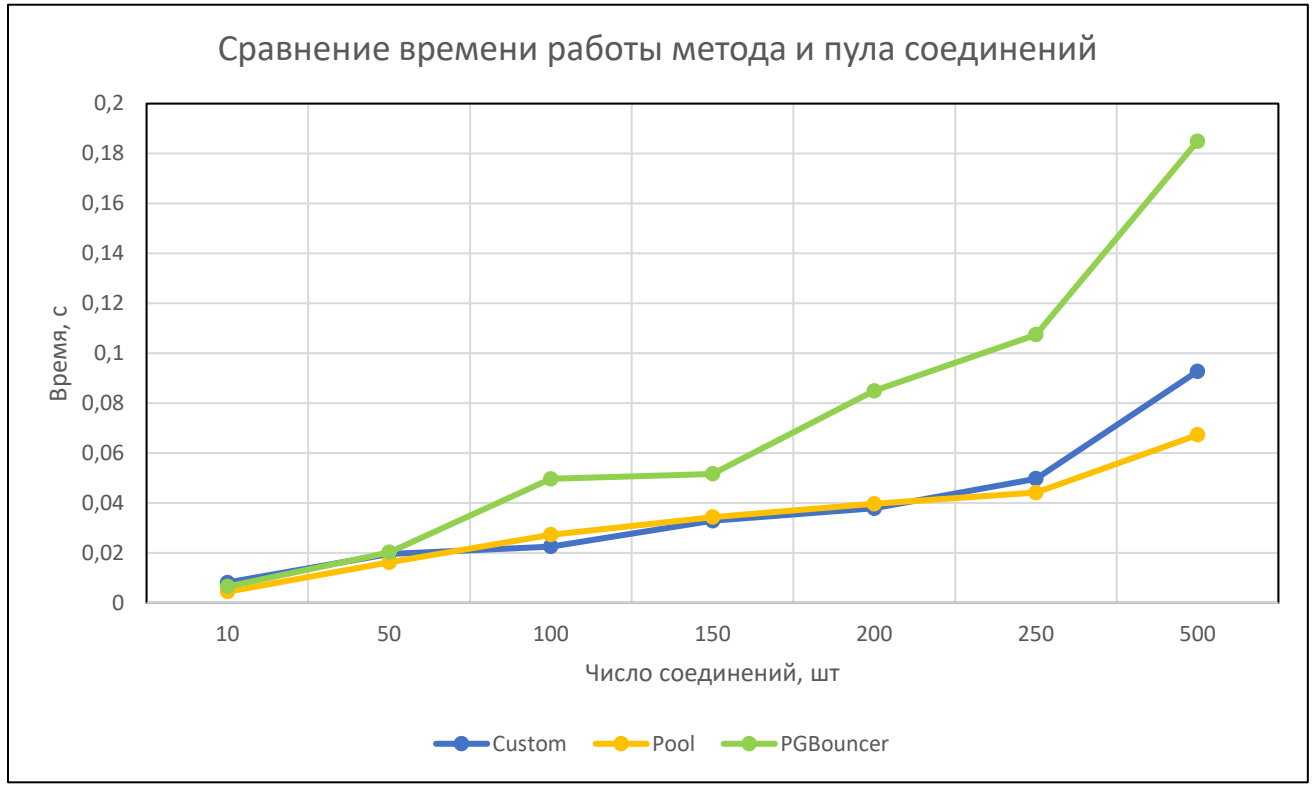
Выполнение простого запроса без нагрузки БД



Сравнение времени выполнения простого запроса для 4 реализаций:

1. последовательная;
2. параллельная;
3. реализация с использованием внешнего пула соединений;
4. реализация с использованием разработанного метода.

Сравнение разработанного метода с пулом соединений



Сравнение времени работы пула, использующего библиотеку libpq и пула, реализованного в качестве внешней службы (PGBouncer), с разработанным методом.

Анализ памяти

Сравнение затрат памяти для каждой реализации в случае создания 10 соединений и выполнения простого запроса.

Реализация	Число раз выделения памяти	Суммарный объем используемой памяти
Однопоточная	729	588,870 байт
Многопоточная	812	593,508 байт
Внешний пул	831	586,212 байт
Разработанный метод	182	180,794 байт

Заключение

Цель достигнута: разработан метод параллельного выполнения запросов к СУБД PostgreSQL в пределах одного соединения.

Поставленные задачи решены:

- Выполнен анализ предметной области и существующих методов выполнения запросов в MPP системах
- Разработан метод параллельного выполнения запросов к СУБД PostgreSQL в пределах одного соединения
- Реализован программный модуль для СУБД PostgreSQL
- Выполнен сравнительный анализ стандартных методов обработки запросов к СУБД PostgreSQL с реализуемым методом

Предложенный метод рекомендуется к применению.

Дальнейшее развитие

- Конкатенация сообщений об ошибке в функции получения результата *PQgetResultThread()*.
- Обеспечение безопасности данных при передаче соединения в созданные потоки.