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

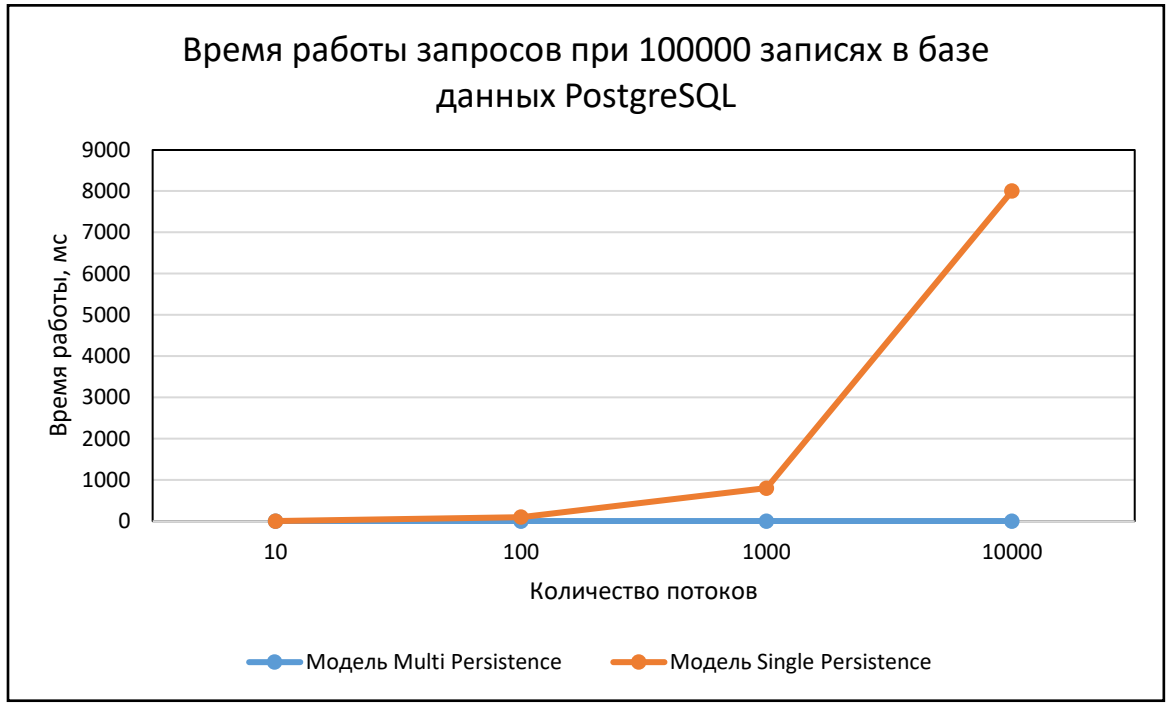
Цель и задачи работы

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 разработанный метод;
- Выполнить сравнительный анализ времени работы метода и различных реализаций выполнения запросов.

Введение в предметную область



Доступ к БД объемом 100.000 записей:

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

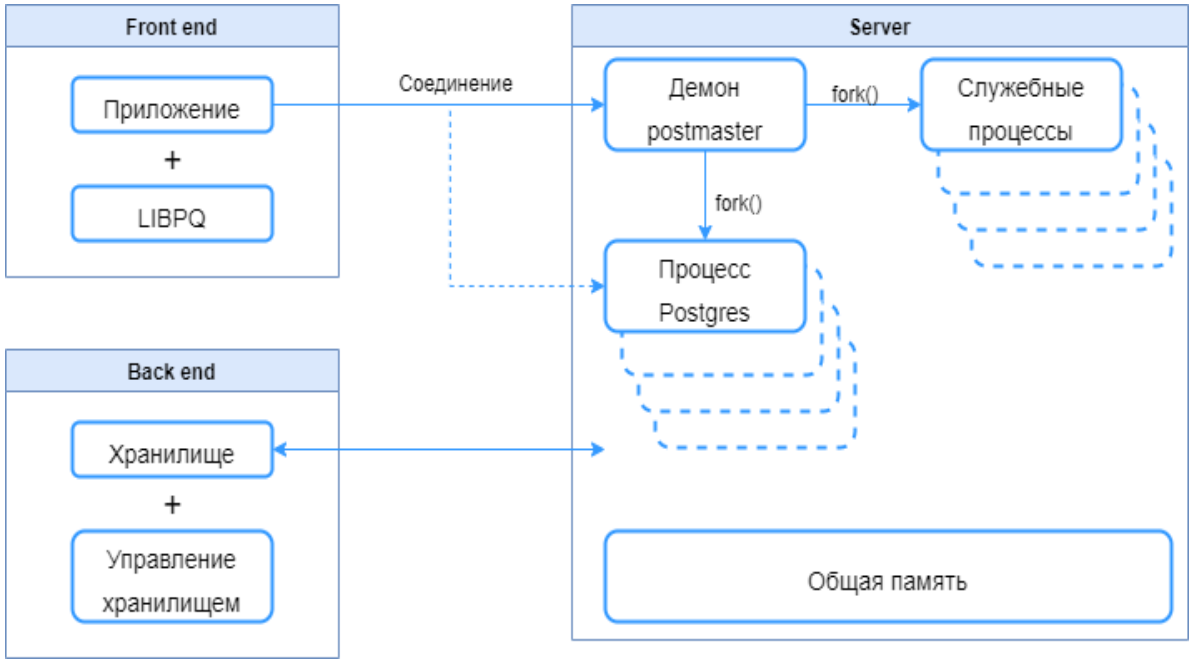
Операция подключения — одна из самых дорогостоящих (процесс подключения к БД занимает от 2 до 3 МБ).

Анализ предметной области

Рейтинг	СУБД	Модель БД
1.	Oracle	Реляционная
2.	MySQL	Реляционная
3.	Microsoft SQL Server	Реляционная
4.	PostgreSQL	Реляционная
5.	MongoDB	Документная

PostgreSQL (14.2):

- доступность исходного кода;
- кроссплатформенность.



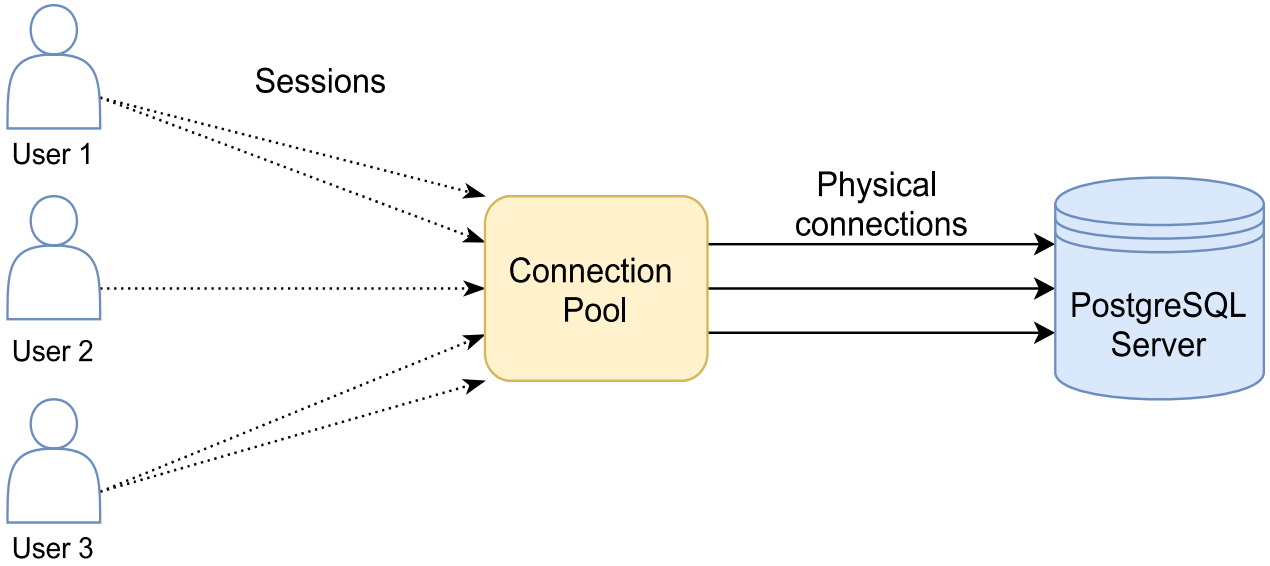
Архитектура PostgreSQL

Пул соединений

Пул соединений — набор открытых и готовых к использованию соединений с базой данных.

Оптимизации процесса подключения к базе данных: многократное использование соединений. Особенно заметно повышение производительности, когда размер пула не превосходит количества потоков.

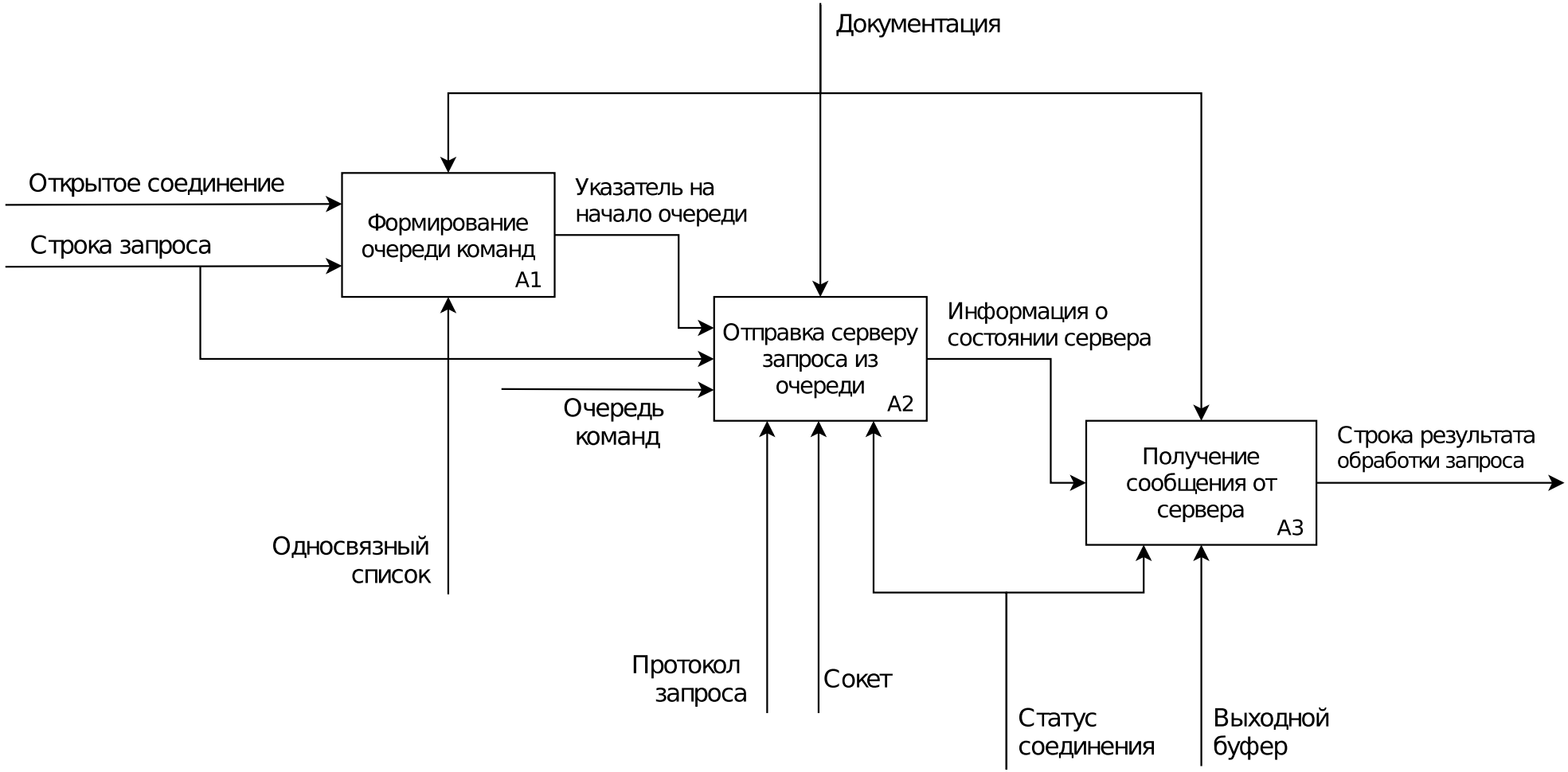
- **Встроенный пул**
- **Пул на основе libpq**
- **Пул в качестве внешней службы**



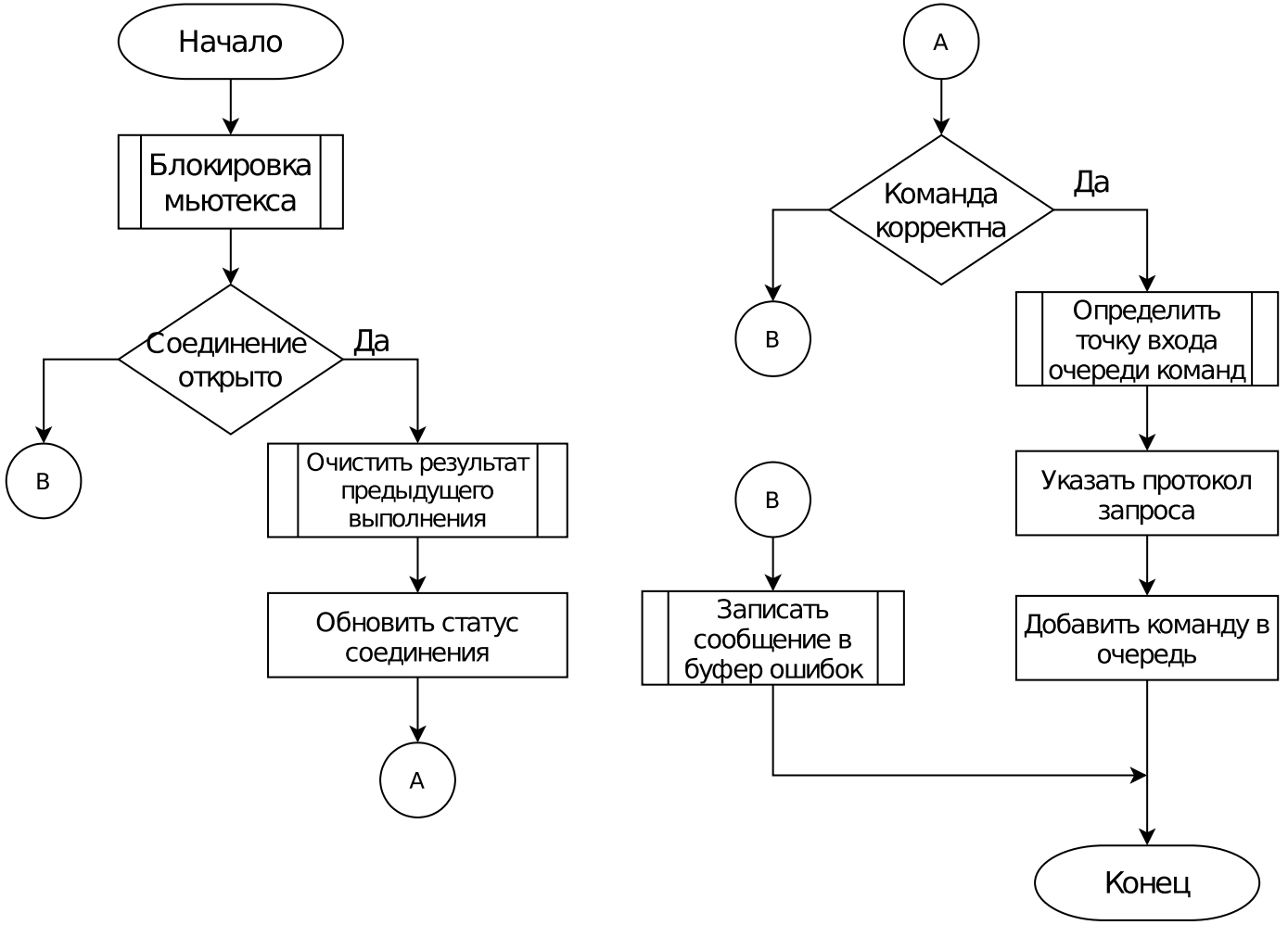
Анализ существующих решений

Вид пула / Критерий	Пул на основе libpq	Пул в качестве внешней службы	Встроенный пул
Максимальный размер пула по умолчанию	100	100	32767
Наличие лицензии	Нет	Нет	Да
Необходимость поддержки отдельного пула для каждой БД	Да	Да	Да
Необходимость самостоятельной разработки и поддержки	Да	Нет	Нет

Функциональная модель разрабатываемого программного комплекса



Формирование очереди команд и отправка серверу запроса из очереди



Для каждого соединения определен выходной буфер данных, которые еще не были отправлены на сервер.

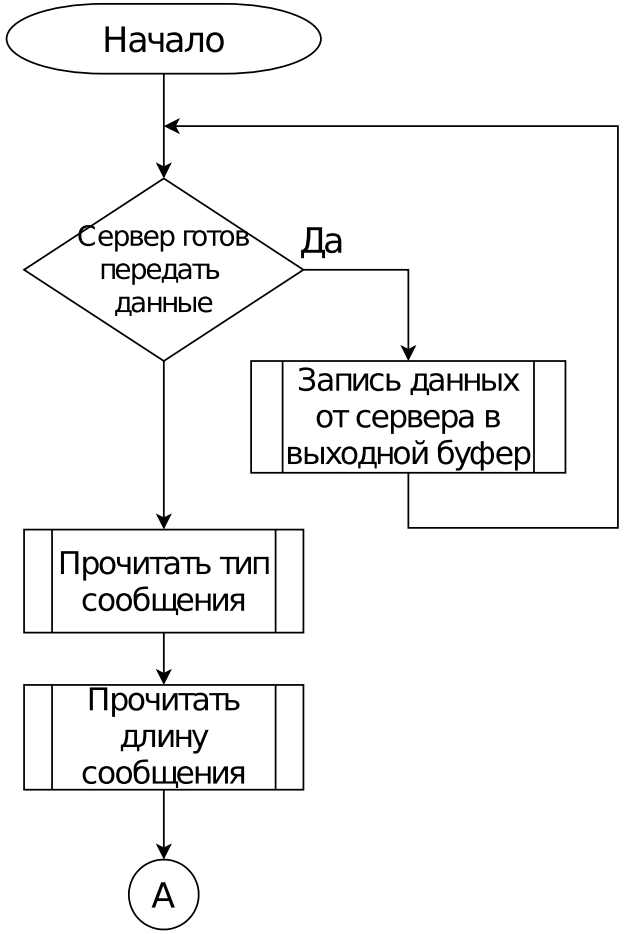
Query

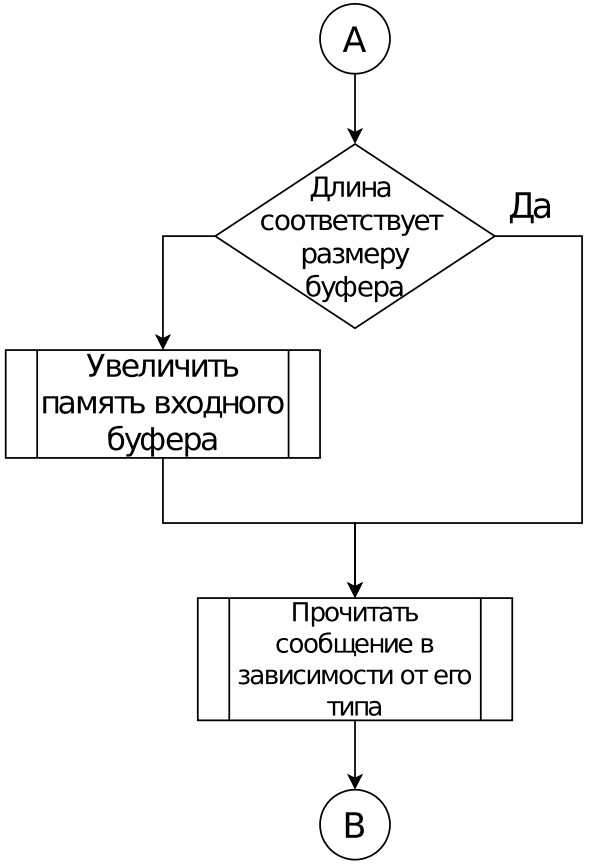
'Q'	int32 len	str query

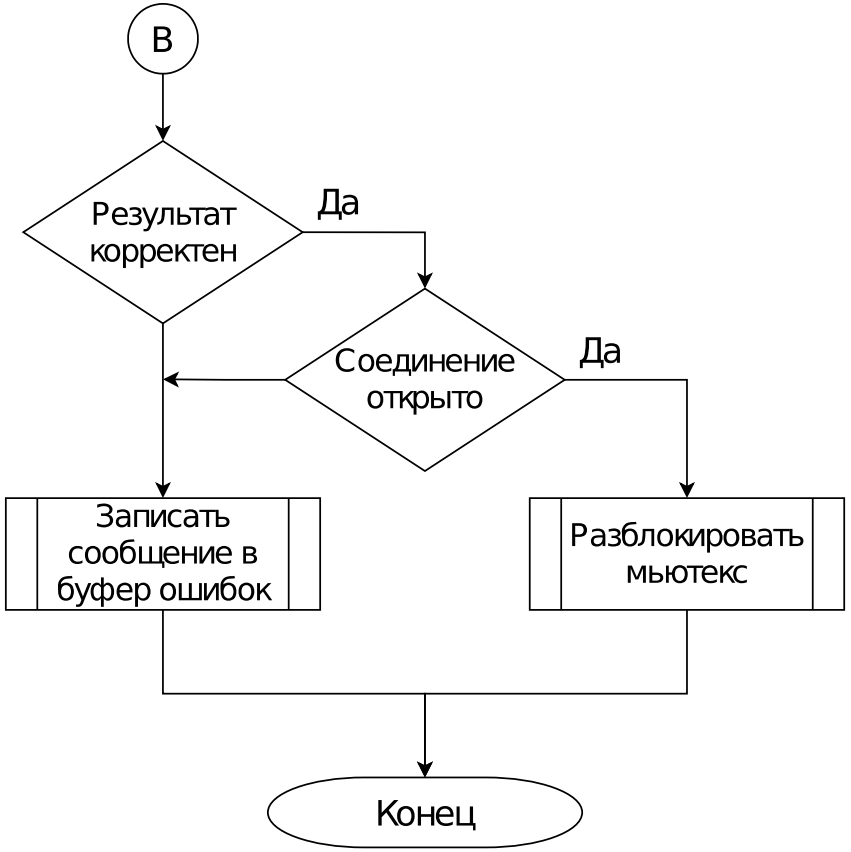
Отправка сообщения:

- размер выходного буфера достиг 8 Кб (исключает отправку маленьких пакетов);
- была вызвана функция *fflush()*.

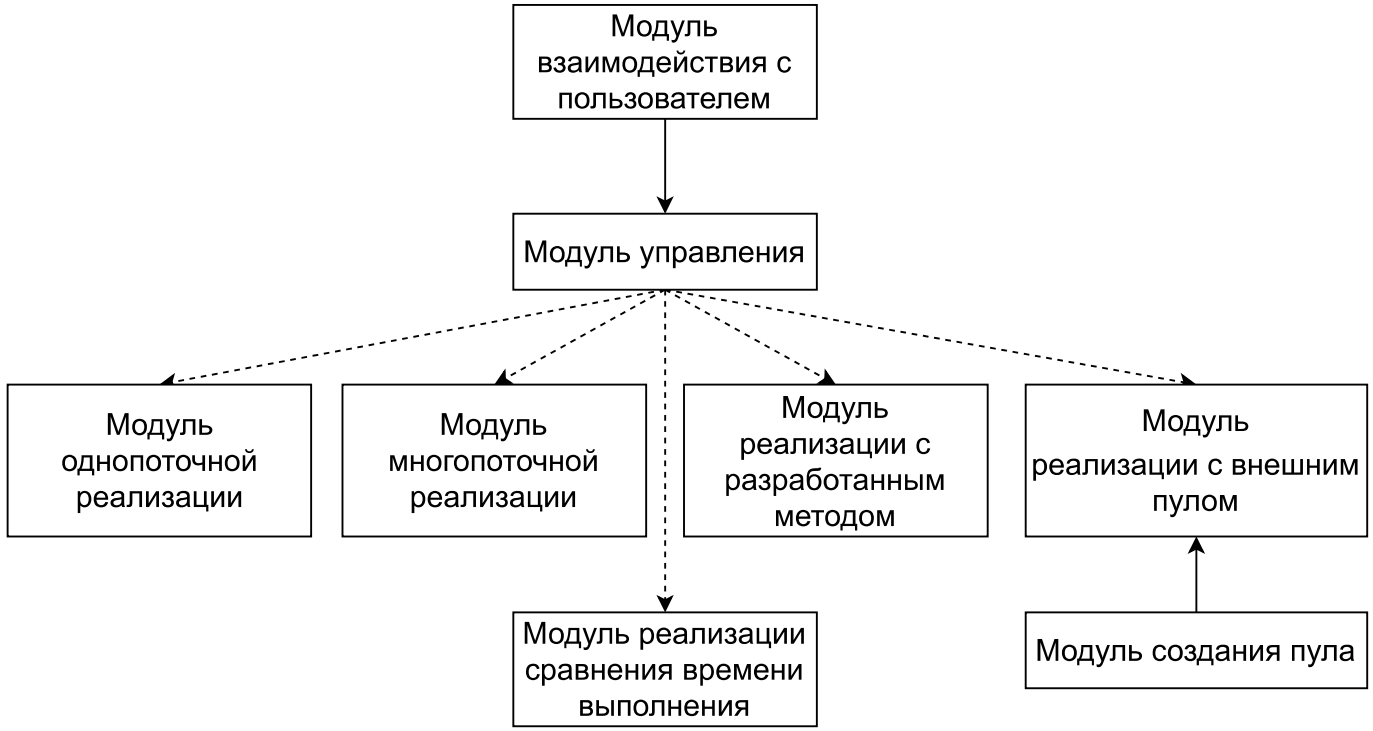
Получение сообщения от сервера







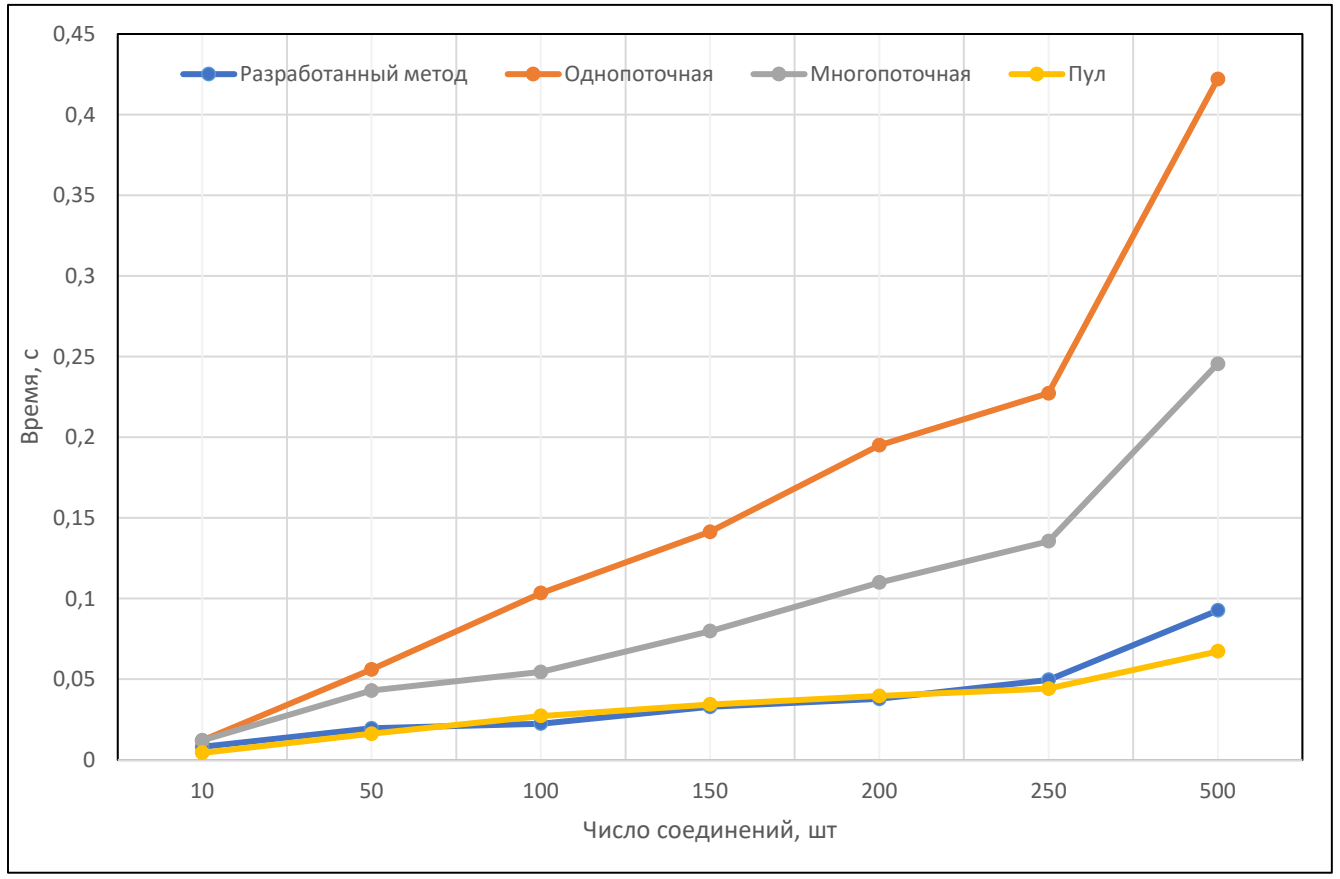
Внешний модуль взаимодействия с разработанным методом



Внешний пул был разработан с использованием умных указателей для предотвращения возможной утечки ресурсов.

Сам пул был реализован в качестве очереди соединений.

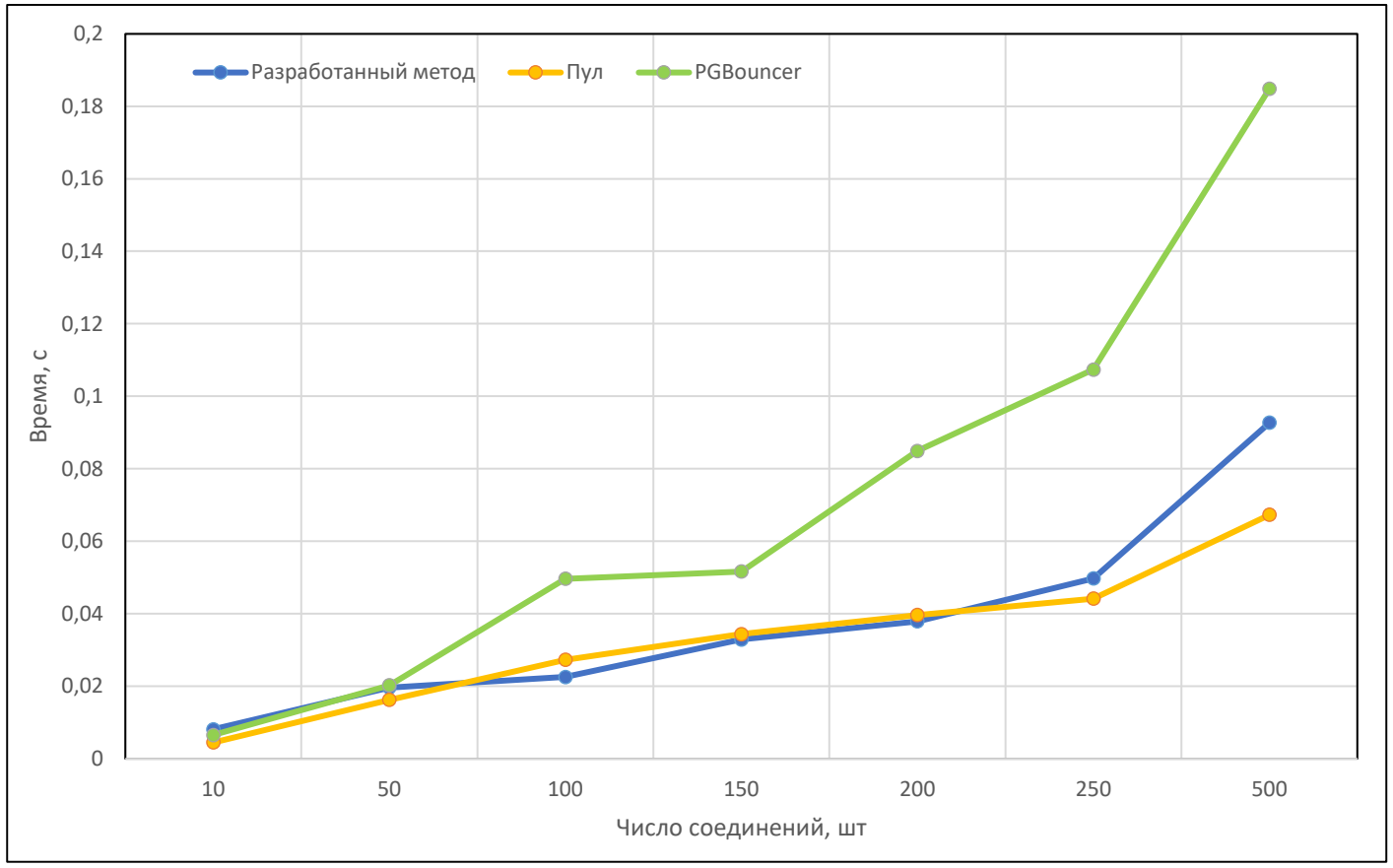
Зависимость времени работы различных реализаций создания соединения от количества соединений с БД



Для каждого опыта учитывалось время:

- создания соединения;
- выполнения запроса;
- очистки результата выполнения;
- закрытия соединения.

Исследование зависимости времени работы реализованного метода и пулов соединений от количества соединений



Сравнение времени работы:

- пула, использующего библиотеку libpq;
- пула, реализованного в качестве внешней службы (PGBouncer);
- разработанного метода.

Исследование требуемых ресурсов для выполнения простого запроса к БД

Простой запрос — выборка всех данных посредством одного оператора ‘SELECT’ и одного оператора ‘FROM’.

SELECT * FROM table100;
Создание 10 соединений.

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 разработанный метод;
- Выполнен сравнительный анализ времени работы метода и различных реализаций выполнения запросов.

Дальнейшее развитие

- Реализация пользовательского вывода информации об ошибках в случае конкатенации нескольких запросов в одну команду.
- Разработка метода управления ресурсным пулом в случае потери потоком соединения с базой данных.