Анализ и диагностика утечек памяти в запросах и расширениях PostgreSQL

Исполнитель — Тагаев Т. В., студент группы ПИ-22
Руководитель — Целебровский О. Б., младший разработчик ПО PostgresPro

Постановка проблемы

- Расширения выполняются в адресном пространстве серверных процессов PostgreSQL;
- Ошибки управления памятью в них критичны из-за отсутствия изоляции;
- Аварийное завершение backend-процесса может инициировать цикл восстановления кластера.

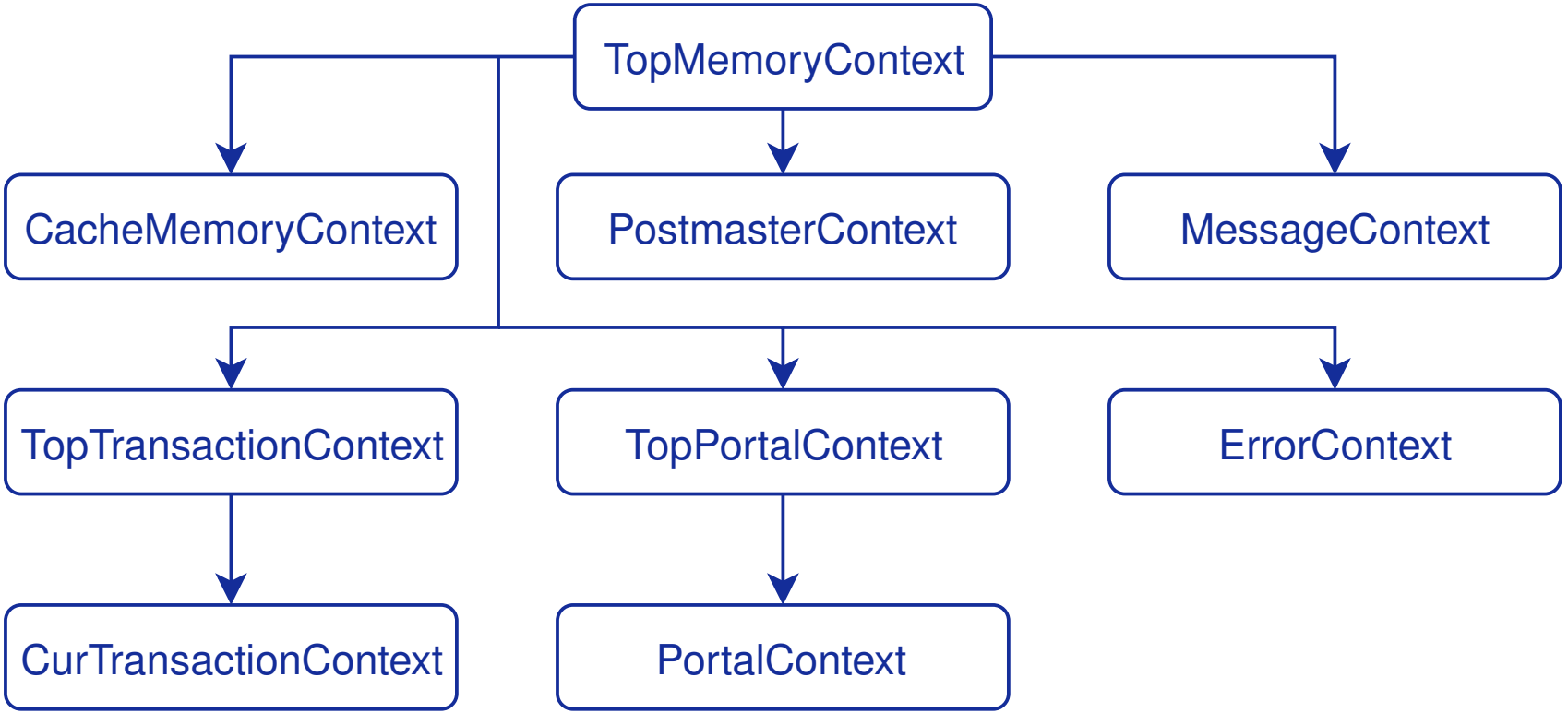
Цель и задачи работы

Цель: разработать C-расширение для обнаружения и диагностики логических утечек памяти в PostgreSQL.

Задачи:

- Анализ ограничений существующих инструментов профилирования;
- Проектирование алгоритма снятия снимков памяти;
- Разработка конвейера для профилирования клиентских запросов;
- Разработка архитектуры межпроцессного взаимодействия для анализа изолированных фоновых процессов;
- Экспериментальная оценка накладных расходов разработанного решения.

Локальная память процесса



Разделяемая (общая) память

- Статическая (*Shmem*)
  - Выделяется при запуске сервера
  - Фиксируется до перезапуска сервера
- Динамическая (*DSM*)
  - Создаётся во время работы сервера
  - Уничтожается после завершения задач

Ограничения внешних инструментов анализа

Статические анализаторы:

- Анализируют исходный C-код;
- Не способны смоделировать динамическое изменение дерева контекстов в runtime;
- Не учитывают специфику жизненного цикла структур PostgreSQL.

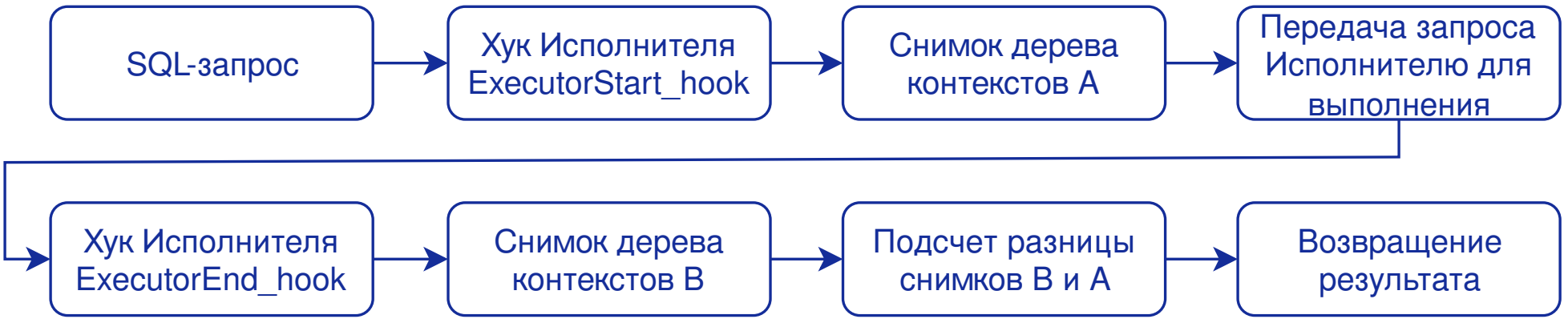
Динамические профилировщики:

- Наблюдают аллокации на уровне системных вызовов ОС;
- Не связывают выделенную память с конкретным SQL-запросом или сессией;
- Не соотносят физические аллокации с логической структурой *MemoryContext*.

Встроенные инструменты СУБД и их ограничения

- *pg_backend_memory_contexts*
  - предоставляет только моментальный снимок дерева контекстов памяти;
  - отражает состояние только текущего backend-процесса.
- *pg_log_backend_memory_contexts*
  - выводит данные снимка в лог-файл сервера, а не в интерфейс SQL-сессии;
  - не предназначен для анализа или агрегации данных.

Конвейер профилирования клиентских сессий



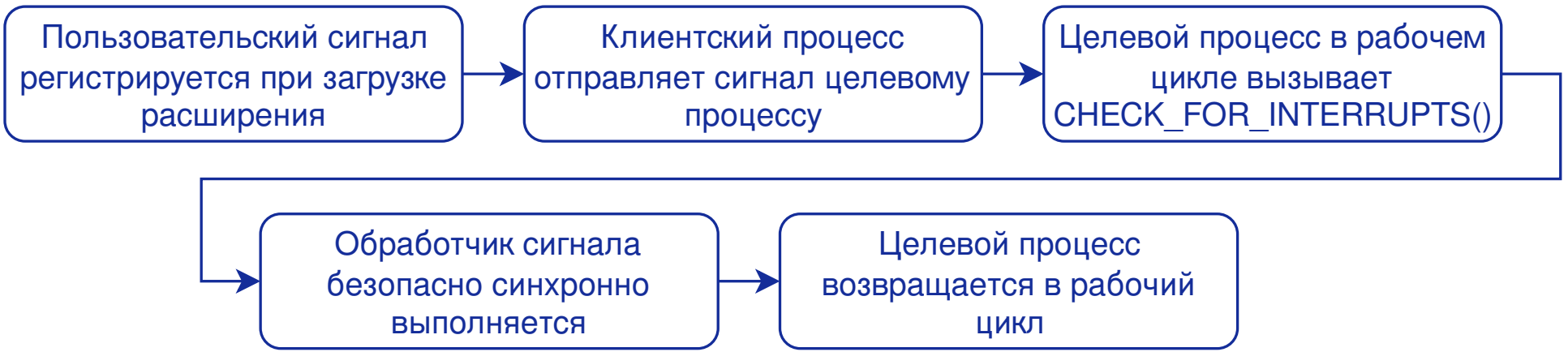
Фильтрация системного шума

- Целевой запрос запускается «вхолостую» перед профилированием для наполнения системных кэшей;
- Профилируемый запрос исполняется внутри субтранзакции с последующим откатом, что изолирует эффект выполнения и очищает краткоживущие транзакционные контексты.

Пример профилирования клиентской сессии

```
postgres=# SELECT context_name, parent_name, level, allocated_before, allocated_after, delta_bytes
postgres-# FROM memleak_analyzer.analyze_query('SELECT leak_example.trigger_context_leak();');
      context_name       |   parent_name    | level | allocated_before | allocated_after | delta_bytes
-------------------------+------------------+-------+------------------+-----------------+-------------
 TopMemoryContext        | -                |     0 |          2196224 |         3244864 |     1048640
 dynahash                | TopMemoryContext |     1 |           171360 |          171360 |           0
 RowDescriptionContext   | TopMemoryContext |     1 |             1280 |            1280 |           0
 MessageContext          | TopMemoryContext |     1 |            21184 |           21184 |           0
 PgStat Shared Ref Hash  | TopMemoryContext |     1 |             8576 |            8576 |           0
 PgStat Shared Ref       | TopMemoryContext |     1 |             2640 |            2640 |           0
 PgStat Pending          | TopMemoryContext |     1 |             2536 |            2536 |           0
 TopTransactionContext   | TopMemoryContext |     1 |             1296 |            1296 |           0
 TransactionAbortContext | TopMemoryContext |     1 |              240 |             240 |           0
 TopPortalContext        | TopMemoryContext |     1 |              512 |             512 |           0
 CacheMemoryContext      | TopMemoryContext |     1 |           477288 |          477288 |           0
 WAL record construction | TopMemoryContext |     1 |            43392 |           43392 |           0
 MdSmgr                  | TopMemoryContext |     1 |              264 |             264 |           0
 GUCMemoryContext        | TopMemoryContext |     1 |            15512 |           15512 |           0
 ErrorContext            | TopMemoryContext |     1 |              240 |             240 |           0
(15 rows)
```

Внедрение механизма пользовательских прерываний



Конвейер профилирования фоновых процессов

Пример профилирования фонового процесса

В SQL-функцию передаются 2 аргумента: PID целевого процесса (*468657*) и время наблюдения за ним в секундах (*5*).

```
postgres=# SELECT context_name, parent_name, level, allocated_before, allocated_after, delta_bytes
postgres-# FROM memleak_analyzer.analyze_bgw(468657, 5);
      context_name       |   parent_name    | level | allocated_before | allocated_after | delta_bytes
-------------------------+------------------+-------+------------------+-----------------+-------------
 TopMemoryContext        | -                |     0 |           425704 |          425704 |           0
 dynahash                | TopMemoryContext |     1 |           157152 |          157152 |           0
 PgStat Shared Ref Hash  | TopMemoryContext |     1 |             8576 |            8576 |           0
 PgStat Shared Ref       | TopMemoryContext |     1 |             1640 |            1640 |           0
 PgStat Pending          | TopMemoryContext |     1 |             4168 |            4168 |           0
 TopTransactionContext   | TopMemoryContext |     1 |              240 |             240 |           0
 TransactionAbortContext | TopMemoryContext |     1 |              240 |             240 |           0
 TopPortalContext        | TopMemoryContext |     1 |              240 |             240 |           0
 CacheMemoryContext      | TopMemoryContext |     1 |           673152 |          673152 |           0
 WAL record construction | TopMemoryContext |     1 |            43392 |           43392 |           0
 MdSmgr                  | TopMemoryContext |     1 |              576 |             576 |           0
 GUCMemoryContext        | TopMemoryContext |     1 |            15056 |           15056 |           0
 ErrorContext            | TopMemoryContext |     1 |              240 |             240 |           0
(13 rows)
```

Конфигурационные параметры

Имя параметра	Тип	Назначение
rollback_mode	bool	Откат/коммит субтранзакций
enable_warmup	bool	Предварительное наполнение кэшей СУБД
max_content_level_displayed	int	Ограничение глубины обхода дерева
merge_contexts	bool	Агрегация дублирующихся контекстов
show_positive_deltas_only	bool	Фильтрация вывода (отображение узлов только с дельтой > 0)

Тестирование производительности

TPS — количество транзакций в секунду (Transactions Per Second), определяющих пропускную способность сервера.

Конфигурация	Средняя пропускная способность, TPS	Относительное падение TPS, %
Без расширения	23556.2	0
Пассивный режим	23576.2	-0.08
Активный режим (без оптимизаций)	5031.6	78.64
Активный режим (с оптимизацией глубины)	7916.6	66.39
Активный режим (с агрегацией контекстов)	6870.8	70.83

Заключение

Результаты работы:

- Выявлены недостатки существующих методов профилирования;
- Разработано инструментальное C-расширение *memleak_analyzer*;
- В ходе разработки адаптирован патч сообщества для безопасного аудита фоновых процессов;
- Проведено нагрузочное тестирование, в ходе которого были определены накладные расходы профилирования.

Анализ и диагностика утечек памяти в запросах и расширениях PostgreSQL

Исполнитель — Тагаев Т. В., студент группы ПИ-22
Руководитель — Целебровский О. Б., младший разработчик ПО PostgresPro