Архитектура PostgreSQL

Процессы и память

При подключении к серверу клиент соединяется с процессом postmaster. В задачи этого процесса входит порождение других процессов и присмотр за ними. Таким образом, postmaster порождает серверный процесс и дальше клиент работает уже с ним. На каждое соединение создается по серверному процессу, поэтому при большом числе соединений следует использовать пул (нпример, с помощью расширения pgbouncer).

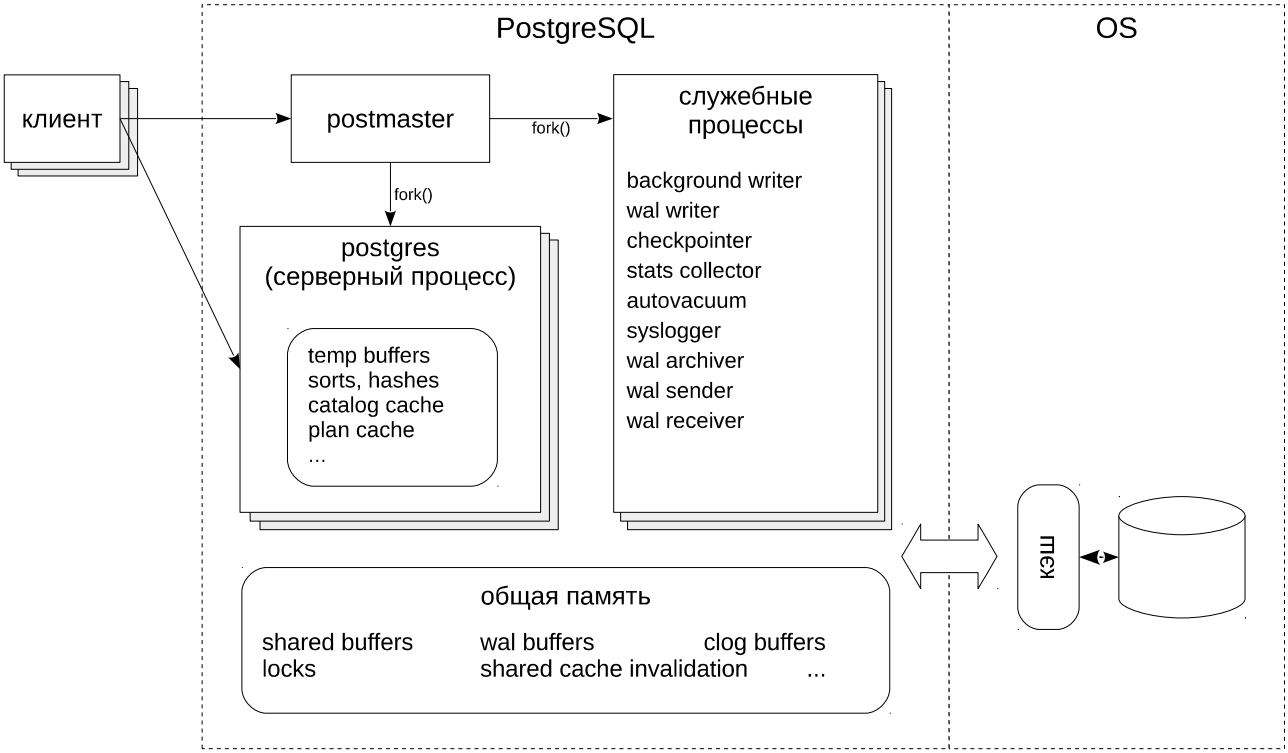
Postmaster также порождает ряд служебных процессов, про основные из которых будет сказано ниже в этом разделе. Дерево процессов можно увидеть с помощью команды ps fax.

У экземпляра СУБД имеется общая для всех серверных процессов память.

Большую ее часть занимает буферный кэш (shared buffers), необходимый для ускорения работы с данными на диске. Обращение к дискам происходит через операционную систему (которая тоже кэширует данные в оперативной памяти). PostgreSQL полностью полагается на операционную систему и сам не управляет устройствами. В частности, он считает, что вызов fsync() гарантирует попадание данных из памяти на диск.

Кроме буферного кэша в общей памяти находится информация о блокировках и многое другое; через нее же серверные процессы общаются друг с другом.

У каждого серверного процесса есть своя локальная память. В ней находится кэш каталога (часто используемая информация о базе данных), планы запросов, рабочее пространство для выполнения запросов и другое.



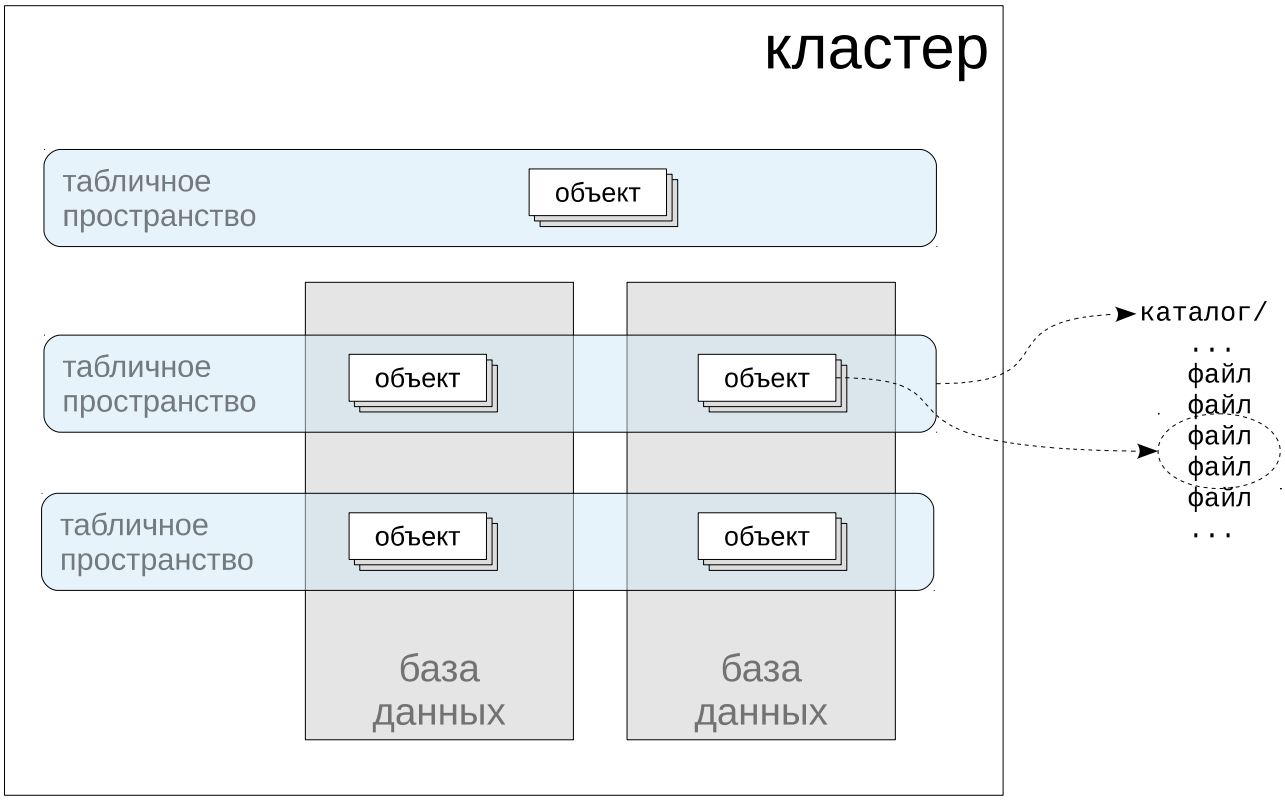
Организация данных

Экземпляр СУБД работает с несколькими базами данных. Эти базы данных называются кластером. Это не должно вводить в заблуждение — термин «кластер» имеет здесь не тот смысл, который в него обычно вкладывается: это не несколько экземпляров, работающих над одними данными, а именно несколько баз данных, обрабатываемых лжин экземпляром.

Хранение данных на диске организовано с помощью табличных пространств. Табличное пространство указывает расположение данных (каталог на файловой системе). Оно может использоваться несколькими базами данных. Например, можно организовать одно табличное пространство на быстрых дисках для активно использующихся данных и другое — на медленных дисках для архивных данных.

Объекты (таблицы и индексы) хранятся в файлах; каждый объект занимает один или несколько (2-3) файлов внутри каталога табличного пространства. Кроме того, файлы разбиваются на части по 1 ГБ. Необходимо учитывать влияние потенциально большого количества файлов на используемую файловую систему.

http://www.postgresql.org/docs/9.4/static/storage-file-layout.html



Буферный кэш

Буферный кэш используется для сглаживания скорости работы памяти и дисков. Он состоит из массива буферов, которые содержат страницы (блоки) данных и дополнительную информацию об этих страницах.

Размер страницы обычно составляет 8 КБ, хотя может устанавливаться при сборке.

Буферный кэш, как и другие структуры в памяти, защищен блокировками от одновременного доступа. Блокировки организованы достаточно эффективно, чтобы не создавать большой конкуренции.

Любая страница, с которой работает СУБД, попадает в кэш. Часто используемые страницы остаются в кэше надолго; редко используемые — вытесняются и заменяются другими страницами.

Буфер, содержащий измененную страницу, называется «грязным». Процесс Background Writer постепенно записывает их на диск в фоновом режиме — это позволяет снизить нагрузку на диски и увеличить производительность. Если Background Writer не успевает записать вытесняемый серверным процессом грязный буфер, то процесс записывает его сам. С точки зрения производительности этого лучше не допускать.

Транзакции

Транзакции представляют собой последовательности операций, которые должны удовлетворять требованиям ACID: атомарность (atomicity), согласованность (consistency), изоляция (isolation) и долговечность (durability).

- Атомарность означает, что при фиксации выполняются все операции, составляющие транзакцию, при откате — не выполняется ни одна.
- Согласованность — поддержание целостности данных. Транзакция начинает работу в согласованном (целостном) состоянии и по окончании своей работы также оставляет данные согласованными.
- Под изоляцией понимается, что на результат работы транзакции не оказывают влияния другие одновременно с ней выполняющиеся транзакции. По стандарту изоляция может иметь несколько различных уровней, в различной степени защищающих транзакции от внешних воздействий.

Эти три свойства достаточно тесно связаны друг с другом. Одним из основных механизмов, обеспечивающих их эффективную реализацию, является многоверсионность.

- Долговечность подразумевает возможность восстановить базу данных после сбоя в согласованном состоянии.

Долговечность обеспечивается журналом предварительной записи.

Оба этих механизма рассматриваются ниже.

Журнал упреждающей записи

В журал упреждающей записи (Write Ahead Log, WAL) записывается информация, достаточная для повторного выполнения всех действия с базой данных.

Записи этого журнала обязаны попасть на диск раньше, чем изменения в соответствующей странице. Тогда при восстановлении можно прочитать страницу с диска, посмотреть в ней номер последней записи WAL, и применить к странице все записи WAL, которые еще не были применены.

Записью журнала занимается процесс WAL Writer. Можно сохранять записи журнала непосредственно при фиксации изменений. В таком случае гарантируется, что зафиксированные данные не пропадут при сбое. Второй вариант — сохранять записи асинхронно, что более эффективно. В этом случае некоторые зафиксированные данные могут пропасть, но согласованность все равно гарантируется.

Журнал состоит из нескольких файлов (обычно по 16 МБ), которые циклически перезаписываются. Старые файлы могут сохраняться и архивироваться процессом WAL Archiver.

Поскольку страница может долго не попадать на диск, возникает необходимость хранить неопределенно много журнальных записей. Чтобы избежать этого, периодически происходит так называемая контрольная точка, при которой все грязные буферы принудительно сбрасываются на диск. Этим занимается процесс Checkpointer.

http://www.postgresql.org/docs/9.4/static/wal.html

Многоверсионность

Идея многоверсионности (Muliversion Concurrency Control, MVCC) состоит в том, чтобы разделить два уровня представления данных.

На нижнем уровне имеем дело со страницами и физическим хранением данных в них. На этом уровне при изменении строки таблицы хранятся несколько версий этой строки, как старые, так и текущая актуальная.

При удалении строки она не удаляется физически из страницы, а помечается номером удалившей его транзакции.

Новая строка помечается номером создавшей его транзакции.

Обновление реализуется как удаление старой строки и вставка новой.

Таким образом, в каждой версии строки хранится информация о начале и конце ее действия.

На верхнем уровне имеем так называемые снимки данных. С помощью информации о начальном и конечном номере транзакции они отбирают версии строк, дающие согласованную картину на определенный момент времени.

Транзакции работаю только со снимками данных. Они не имеют представления о физическом хранении и видят только те строки, которые предоставлены снимком. За счет этого достигается изоляция — каждая транзакция видит свою картину данных и не мешает другим, одновременно с ней работающим транзакциям.

Преимущество многоверсионности перед реализациями, основанными только на блокировках, состоит в существенно большей эффективности, так как гарантируется, что блокируется только одновременное изменение одних и тех же строк. Но читатели никогда не блокируют писателей, а писатели — читателей.

Старые версии строк, которые не видны ни одной из активных транзакций, должны быть физически удалены, чтобы освободить занимаемое ими место. Этим занимается процесс Autovacuum Launcher, запускающий для выполнения работы процессы Autovacuum Worker. Также очистку можно запустить вручную командой VACUUM.

http://www.postgresql.org/docs/9.4/static/mvcc.html

Обработка запросов

Серверный процесс, порожденный Postmaster, ожидает от подключенного к нему клиента запросы, выполняет их и возвращает результаты. При этом текст запроса проходит несколько этапов обработки.

Анализатор (parser) выполняет первоначальный синтаксический и семантический разбор текста запроса. Для этого он использует системный каталог, в котором хранится информация обо всех объектах базы данных.

Результатом работы анализатора является дерево разбора.

Далее запрос **переписывается (rewriter)** с помощью системы правил. Правила — открытый механизм, с помощью которого можно изменить исходный запрос. В частности, на этом этапе происходит замена представлений на текст запроса.

Результатом является измененное дерево разбора.

Планировщик (planner) выбирает для запроса лучший (в смысле минимизация стоимости выполнения) план. Стоимость вычисляется на основе статистики, которую собирает процесс Stats Collector, а также Vacuum. План определяет, в каком порядке будут соединяться таблицы, какие методы доступа (полное сканирование таблиц, использование индексов) и

соединений (вложенные циклы, соединение слиянием или с помощью хэширования) будут использоваться.

Ключевое требование для корректной работы оптимизатора — наличие аккуратной и актуальной статистики.

Затем управление передается **исполнителю (executor).** Запрос выполняется в соответствии с планом. В случае, если это был оператор select, клиенту возвращается набор строк, удовлетворяющий сформулированным условиям.

Расширяемость

PostgreSQL спроектирован с расчетом на расширяемость.

В его стандартный состав, помимо SQL и PL/pgSQL, входят три языка программирования (PL/Perl, PL/Python, PL/Tcl), и еще несколько доступно в качестве расширений. Кроме того, есть возможность добавлять новые языки.

http://www.postgresql.org/docs/9.4/static/plhandler.html

Поддерживается множество различных типов данных, включая как стандартные (числа, строки, даты), так и логический тип, перечисления, массивы, типы для работы с геометрическими объектами и др.

http://www.postgresql.org/docs/9.4/static/datatype.html

Если их не достаточно, можно реализовать свой тип и определить для него новые операции. Чтобы индексировать поля с новым типом, можно реализовать свой тип индекса.

http://www.postgresql.org/docs/9.4/static/indexam.html

Для доступа к данным вне СУБД можно написать обертку сторонних данных (foreign data wrapper, FDW).

http://www.postgresql.org/docs/9.4/static/fdwhandler.html