§7 БАЗЫ ДАННЫХ

Лучинин З. С.

СТРУКТУРА ДАННЫХ ДЛЯ ДОКУМЕНТО-ОРИЕНТИРОВАННЫХ БАЗ ДАННЫХ

Аннотация. *В статье приводится подход, который позволяет уменьшить нагрузку при запросах к нереляционной СУБД, благодаря использованию алгоритмов деревовидных структур хранения данных. Производительность операций по обработке данных различна в зависимости от используемых структур данных. Исследование древовидных структур, таких как B + деревья, объединения деревьев в журнальную структуру или фрактальные деревья показало, что в алгоритмах с их использованием операции с данными осуществляются быстрее, чем в MySQL. В работе рассмотрен алгоритм LSM дерева в применении к документо-ориентированным базам данных. Описана работа алгоритма при выполнении основных операций (создание, чтение, редактирование и удаление) с данными. В основу предложенного алгоритма работы с индексом ставят B деревья или B+ деревья. Недостатками данных структур являются трудоемкость балансировки дерева при добавлении нового значения в индекс и ресурсоемкость, так как индекс хранится в оперативной памяти. Дерево слияния со структурой журнала (Log-Structured Merge-Trees, LSM) является структурой данных, обеспечивающей низкую стоимость операции индексирования и высокую скорость добавления и удаления данных Алгоритм на основе LSM-дерева можно использовать при горизонтальном масштабировании. Каждый узел образует отсортированную последовательность данных по ключу. Диапазон ключей каждого сервера хранится на мастер-сервере, что позволяет без дополнительных запросов обратиться к серверу, на котором хранятся запрашиваемые данные. Таким образом, достигается увеличение скорости поиска данных и балансировка нагрузки по доступным серверам.*

Ключевые слова: *базы данных, документо-ориентированные базы данных, структура данных, B + деревья, LSM деревья, нереляционные системы, поиск данных, обработка данных, производительность операций, древовидные структуры.*

1. Введение

Увеличение мощности вычислительных машин и увеличение объема обрабатываемой информации требует от программистов пересмотра алгоритмов обработки данных и структур хранения для увеличения производительности. Как следствие возникает проблема эффективного доступа и обработки данных.

Одним из актуальных на сегодняшний день подходов для обработки и хранения большого объема данных, вызывающих наибольший интерес у разработчиков программного обеспечения, являются нереляционные системы управления базами данных (MongoDB, Amazon SimpleDB, CouchDB). Структуры данных в нереляционных СУБД могут быть представлены в виде хеш-таблиц, деревьев, графов, что позволяет хранить данные в подходящей для программного обеспечения структуре. Для связи требований к скорости обработки данных и гибкой модели данных необходим алгоритм, позволяющий совместить в себе выдвигаемые требования. Каждый узел LSM-дерева образует отсортированную последовательность данных по ключу. Диапазон ключей каждого сервера хранится на мастер-сервере, что позволяет без дополнительных запросов обратиться к серверу, на котором хранятся запрашиваемые данные.

DOI: 10.7256/2305-6061.2013.3.10772

2. Описание алгоритма

Увеличение скорости обработки данных в базе достигается с помощью индексов. Реляционные системы управления базами данных в основу алгоритма работы с индексом ставят B деревья или B+ деревья [1]. Недостатками данных структур являются трудоемкость балансировки дерева при добавлении нового значения в индекс и ресурсоемкость, так как индекс хранится в оперативной памяти. Дерево слияния со структурой журнала (Log-Structured Merge-Trees, LSM) является структурой данных [2], обеспечивающей низкую стоимость операции индексирования и высокую скорость добавления и удаления данных, как при больших, так и при малых объемах.

LSM-дерево состоит как минимум из двух одинаковых по структуре деревьев, а именно C_0 и C_1. Один из компонентов, в данном случае C_0, находится в оперативной памяти, а второй компонент C_1 — на накопителе жесткого диска. Каждая новая операция добавления данных генерирует в журнале закрепления запись с определенным ключом. Журнал закрепления — один на все пространства имен и представляет последовательность операций модификации данных, хранящуюся на жестком диске. После добавления в журнал ключ записи добавляется в дерево C_0. Данный объект ограничен объемом выделенной оперативной памяти и служит, как первичное хранилище ключа записи, так как скорость записи и обработка данных в оперативной памяти намного выше, чем на накопителе жестких магнитных дисков.

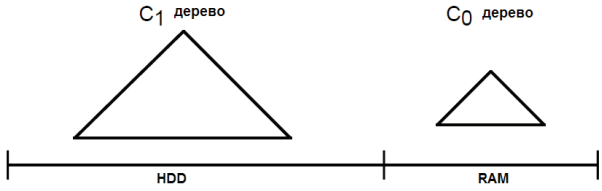

Рисунок 1. *Схематическое изображение LSM дерева, состоящего из 2 компонентов*

При исчерпании объектом C_0 выделенных объемов оперативной памяти происходит перенос данных на жесткий диск. При переносе данных происходит слияние сегментов деревьев. Структура дерева хранимого на жестком диске сопоставима со структурой хранения B-дерева, но оптимизирована для последовательного доступа к диску. Для данной структуры данных свойственно хранение родительского узла перед дочерними узлами в одной последовательности, в виде смежных многостраничных блоков на жестком диске для эффективного использования.

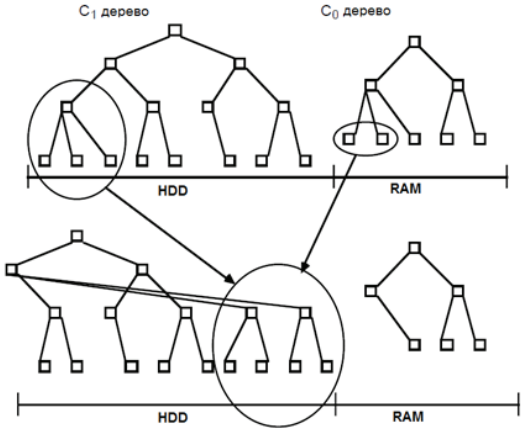

Рисунок 2. *Схема переноса данных из оперативной памяти на жесткий диск*

DOI: 10.7256/2305-6061.2013.3.10772

Новые объединенные данные записываются на чистые блоки жесткого диска, таким образом, старые блоки не перезаписываются и будут доступны для восстановления в течение определенного промежутка времени в случае аварии. Предложенная схема позволяет добиться скорости записи данных ограниченной лишь скоростью записи жесткого диска. Журнал закрепления так же позволяет восстановить данные в случае аварийной остановки узла. Рассмотренная схема записи реализована в технологии хранения данных на жестком диске SSTable.

Операция поиска осуществляется с помощью индекса LSM-дерева, что гарантирует быстрый поиск данных. В первую очередь проверяются данные в оперативной памяти, а именно в C_0 дереве, а после в C_1 дереве. Поиск в таком порядке связан с наличием несохраненных на жесткий диск данных в оперативной памяти компьютера.

3. Заключение

Алгоритм на основе LSM-дерева можно использовать при горизонтальном масштабировании. Каждый узел образует отсортированную последовательность данных по ключу. Диапазон ключей каждого сервера хранится на мастер-сервере, что позволяет без дополнительных запросов обратиться к серверу, на котором хранятся запрашиваемые данные. Таким образом достигается увеличение скорости поиска данных и балансировка нагрузки по доступным серверам.

Библиография

1. Jeremy Cole B+Tree index structures in InnoDB. Ссылка на ресурс в Интернете: http://blog.jcole.us/2013/01/10/btree-index-structures-in-innodb/
2. Patrick O'Neil The Log-Structured Merge-Tree (LSM-Tree). Ссылка на ресурс в Интернете: http://goo.gl/2OcRQ

References

1. Jeremy Cole B+Tree index structures in InnoDB. Ssylka na resurs v Internete: http://blog.jcole.us/2013/01/10/btree-index-structures-in-innodb/
2. Patrick O'Neil The Log-Structured Merge-Tree (LSM-Tree). Ssylka na resurs v Internete: http://goo.gl/2OcRQ

DOI: 10.7256/2305-6061.2013.3.10772