УДК 004.651.4

Студент В.П. Мойсеенко
ГВУЗ «Национальный горный университет»

HBASE. УНИВЕРСАЛЬНОЕ ХРАНИЛИЩЕ ДАННЫХ

В данных тезисах содержатся сведенья о системе хранения данных Apache HBase, представлена модель данных и различные подходы к проектированию таблиц со сравнительным анализом производительности этих подходов.

Компоненты HBase

HBase – нереляционная, распределенная, горизонтально-масштабируемая база данных. HBase является частью проекта Apache Hadoop, который поддерживается Apache Software Foundation под лицензией The Apache Software License, Version 2.0.

Работа HBase основана на использовании двух независимых систем: Apache HDFS и Apache Zookeeper.

Apache HDFS(Hadoop Distributed File System) – модуль, реализующий распределенную файловую систему. Данный модуль является частью Apache Hadoop – системы распределенного решения трудоемких задач. Главными задачами HDFS является контроль и поддержание файловой системы, которая обеспечивает хранение данных на большом числе физических машин(серверов), предоставление возможности быстрого доступа к требуемым данным, а также репликация данных. К вторичным задачам HDFS можно отнести мониторинг и представление мета-данных файловой системы пользователю, обеспечение возможности быстрого, горячего подключения/замены жестких дисков и серверов. HDFS выступает основной, необходимой для функционирования HBase.

Apache Zookeeper – система контроля серверов для распределенных систем. Основной задачей Apache Zookeeper является контроль серверов-хранилищ данных HBase. Apache Zookeeper выступает шлюзом для подключении при работе с хранилищем HBase. Все подключения к HBase обслуживаются и контролируются Apache Zookeeper.

Apache HBase состоит из двух типов компонентов:

1. Модуль “Region Server” – основной компонент хранения данных, который представляет собой логический сервер, выполняющий работу по получению, записи, хранению и изменению определенного массива данных. Region Server является единицей масштабирования системы. Число Region-серверов равно числу машин, отвечающих за хранение данных в кластере. Каждый Region Server хранит определенную часть массива данных.

2. Модуль “Master Server” – компонент контролирующий Region-сервера. Master Server обрабатывает запросы к БД путем делегации к определенным Region серверам и агрегации результатов выборок. Master Server, являясь управляющим компонентом в горизонтально-масштабируемой системе, располагается на одном из серверов кластера.

<u>Модель данных</u>

Модель данных HBase основана на использовании отношения «ключ-значение», несколько отношений объединены в структуру типа «карта». Логически данные разделены на таблицы. Каждая таблица имеет четыре уровня хранения данных: строка, семейство, столбец, версия. Модель данных HBase представлена на рисунке 1.

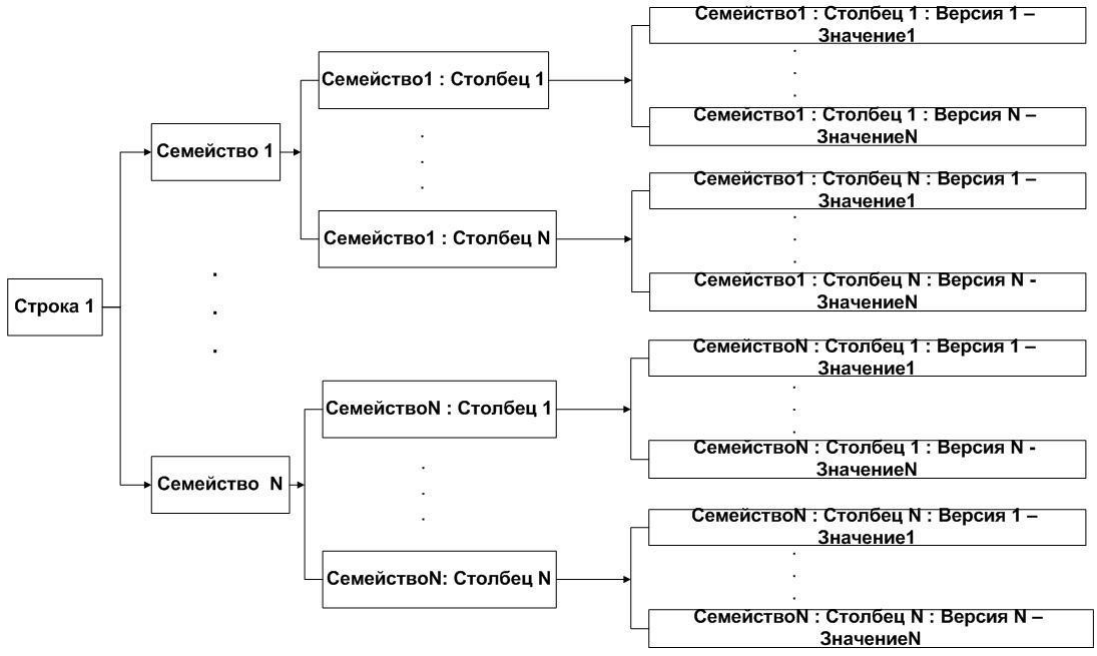

Рис.1 Модель данных HBase

Данные в HBase могут быть двух типов: массив байт и метка времени(timestamp).

Строки, семейства, столбцы, значения хранятся как массивы байт, в то время как версии указываются с помощью меток времени. При вставке уже имеющихся строк, столбцов, версий их содержимое будет переписано новыми данными. HBase хранит строки и столбцы в литеральном порядке. Весь массив данных распределен между Region-серверами.

Операции с данными

HBase поддерживает четыре типа операций: put, delete, get, scan.

Put – вставка данных.

Синтаксис: *put 'table1','row1','family:column1',('version'),'value'.*

Примечание: При дублировании значений строки, столбца, версии данные будут заменены.

Delete – удаление данных.

Синтаксис: *delete 'table1','row1',(family:column1),('version').*

Примечание: В случае использования команды в виде *delete 'table1','row1'* будут удалены данные всей строки.

Get – операция выборки одной строки.

Синтаксис: *get 'table1','row1'*.

Примечание: Основной синтаксис может быть расширен путем указания столбца и количества версий для выборки:

get 'table1','row1',{COLUMN=>'family:column1', VERSIONS => 4}

По умолчанию выбирается одна версия столбца.

Scan – операция выборки набора строк.

Синтаксис: *scan 'table1'.*

Примечание: В таком виде данная команда используется редко, чаще всего она расширяется указанием начальной и конечной строки выборки, предельным количеством строк в выборке, количеством версий:

scan

'table1',{STARTROW=>'row1',STOPROW=>'row10',VERSIONS=>2,LIMIT =>15}

Значение STOPROW не входит в выборку.

Проектирование БД. Row/Column oriented

Существует два принципа построения таблиц в HBase. Первый принцип основан на использовании строки, как уникального идентификатора, второй - на использовании в качестве уникального идентификатора строки и столбца.

Рассмотрим оба подхода на простом примере. Предположим нам необходимо хранить сообщения пользователя в блоге. Сообщение имеет следующие поля: идентификатор сообщения, уникальный идентификатор пользователя, текст сообщения, дата написания.

При использовании строки, как уникального идентификатора вид данных в HBase будет следующим:

<userId - messageId> - <data:> - <version1> - <text>

<userId - messageId> - <data:> - <version2> - <date_write>.

При использовании стороки и столбца в качестве уникального идентификатора, данные в HBase будут выглядеть следующим образом:

<userId> - <data:messageId> - <version1> - <text>

<userId> - <data:messageId> - <version2> - <date_write>.

Оба подхода имеют как преимущества, так и недостатки.

Первый подход позволяет легко выбирать сообщения определенного пользователя с помощью запроса:

scan 'blogs',{STARTROW=>'userId',STOPROW=>'userId+1'}.

При выполнении команды в выборку попадут все сообщения пользователя userId, так как HBase хранит все данные в отсортированном виде. Также просто строится запрос на получение определенного сообщения пользователя путем команды get:

get 'blogs','userId-messageId',{COLUMN=>'data:',VERSIONS=>2}.

Однако существует несколько недостатков данного подхода:

1. Низкая производительность scan-запросов;

2. Быстро растущее количество строк в таблице;

3. Проблемы с литеральной сортировкой чисел.

Второй подход к хранению данных позволяет получать легкий доступ к одному сообщению пользователя, но при этом затрудняет постраничную выборку сообщений для отображения. Построение всех запросов происходит при использовании команды get, что существенно ускоряет выборки, так как доступ к данным идет напрямую, по ключу. Намного уменьшается количество строк в таблице, а значит, снижается нагрузка на механизм поиска HBase. Число строк равняется числу пользователей, имеющих блог, тогда как, в первом случае число строк существенно больше и равняется числу сообщений во всей системе блогов.

Для сравнения производительности двух подходов был создан тест. Тест состоит из трех этапов: вставка, выборка диапазона(Range Selection), выборка одного сообщения(Single Selection).

Вставка(Put): 10 потоков одновременно вставляют 1 000 000 сообщений от 1000 пользователей. Выборка диапазона(Range Selection): 10 потоков выбирают по 10 сообщений каждого пользователя. Выборка одного сообщения(Single Selection): 10 потоков выбирают все сообщения по одному.

HBase был развернут на системе с такими параметрами: Intel Core i5, 16 Gb RAM, HDD 250 Gb + 3*500 Gb. Результаты теста представлены в таблице 1.

Табл. 1 Результаты сравнения Row Oriented и Column Oriented подходов

Действие	Row Oriented		Column Oriented	
	среднее время	максимальное время	среднее время	максимальное время
Put	8,5 ms	10 ms	9,5	12 ms
Range Selection	2136 ms	3125 ms	27 ms	57 ms
Single Selection	100 ms	189 ms	110 ms	201 ms

Из полученных результатов можно сделать вывод о предпочтительном использовании Column Oriented подхода для решения задачи хранения сообщений в блогах пользователей.

Перечень литературы:

1. Lars George. HBase: The Definitive Guide. O'Reilly Media. 2011. 554 c.

2. Apache HBase Reference Guide, 2011 (http://hbase.apache.org/book/book.html).