УДК 004.94

РАЗРАБОТКА СИСТЕМЫ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ДЛЯ ИССЛЕДОВАНИЯ ВАРИАНТОВ АРХИТЕКТУРЫ РАСПРЕДЕЛЕННОЙ ВЫСОКОНАГРУЖЕННОЙ BIGDATA-СИСТЕМЫ

А.И. Харламов, Л.И. Сучкова, Е.В. Бочкарева
Алтайский государственный технический университет им. И.И. Ползунова
г. Барнаул

Целью исследования является разработка метода генерации различных вариантов хранения данных в распределенных высоконагруженных системах, построенных с использованием NoSQL-подхода к хранению данных, и исследование их производительности при выполнении заданных запросов. Для реализации предложенного подхода разработано программное обеспечение, позволяющее осуществлять имитационное моделирование работы высоконагруженной распределенной вычислительной системы для различных вариантов ее структуры, при выполнении из «главного» множества, т.е. запросов, которые выполняются чаще всего и скорость выполнения которых влияет на производительность системы в целом. При этом учитываются варианты логической структуры данных, физические характеристики оборудования и опыт конечного пользователя.

Ключевые слова: запрос данных, высокая нагрузка, имитационное моделирование, Big Data, распределенная система, структура данных, партицирование, процессор правил.

Распределенные высоконагруженные системы (Big Data системы) должны обеспечивать высокую скорость отклика, возможность работы с большими и постоянно растущими объемами машинно-генерируемых данных (до терабайтов информации в сутки) [1]. Важной особенностью Big Data систем является то, что количество типов запросов известно заранее, и разработчик может самостоятельно указать все особенности оптимального выполнения каждого из них. Вместо планировщиков выполнения запросов востребованы средства мониторинга разрабатываемой системы – цель их работы состоит в нахождении «узких» мест и выработке рекомендаций по оптимизации работы системы. Эту особенность важно учитывать при проектировании архитектуры системы, что на сегодняшний день делается исключительно на основе опыта разработчиков программного обеспечения. Ошибки на этапе проектирования системы чреваты потерей скорости работы, производительности системы, вероятной потерей данных из-за отказа оборудования и денежными потерями.

Таким образом, важным этапом проектирования системы Big Data является поиск оптимальной структуры, что подразумевает выбор способа хранения и распределения данных между ее узлами, подбор оборудования и хранилищ данных. При этом необходимо учесть специфику выполняемых запросов, а также способы оптимизации структуры системы путем партиционирования, денормализации и агрегации.

Разработанная имитационная система позволяет генерировать варианты архитектуры системы Big Data и исследовать их работоспособность при выполнении запросов. При этом предусмотрена возможность задавать варианты логической организации таблиц базы данных, выбирать типы индексов для каждой таблицы и настраивать семейства столбов (column families), учитывать варианты распределения данных по физическим устройствам, задавать характеристики этих устройств. А также реализовать моделирование репликации данных и механизмов разбиения и слияния регионов данных.

На вход имитационной систем подается реляционная модель R базы данных и множество «ключевых» запросов Q. В процессе работы имитационной системы структура базы данных претерпевает ряд изменений, в ходе которых переходит к нереляционной форме представления данных NR. При удачном выборе структуры данных, их распределения по машинам кластера, правильном выборе характеристик оборудования это позволяет ускорить

выполнение запросов и увеличить производительность в целом.

При переходе от реляционного табличного описания данных к описанию в нереляционной форме важно учесть многовариантность трансформации: для одной логической структуры данных может существовать достаточно большое (если вообще ограниченное) количество вариантов HBase представления. Каждый из этих вариантов будет обладать разными характеристиками быстродействия и производительности.

Варианты нереляционной структуры данных могут варьироваться по следующим параметрам:

– варианты логической организации таблиц базы данных;

– выбор полей и типов индексов для каждой таблицы;

– варианты денормализации данных.

Схематично опишем архитектуру и принципы работы построенной имитационной системы. Так как имитационная система представляет собой сложную и объемную структуру, для простоты ее рассмотрения, представим систему в виде взаимосвязанных блоков (рисунок 1):

Рисунок 1 – Архитектура имитационной системы

На первом этапе работы имитационной системы осуществляется переход от реляционного представления базы данных к NoSQL представлению. Это сложное преобразование осуществляется через серию последовательных переходов от описания реляционной структуры базы данных в виде *.sql файла, через синтаксический его разбор и построения дерева к триплетам вида субъект-предикат-объект. Парсер для грамматики, описывающей реляционную базу данных и совокупность запросов к ней, реализован с помощью генератора парсеров ANTLR, для чего составлена соответствующая грамматика [2-3].

Для работы со структурой базы данных выбрана модель Resource Description Framework (RDF) [4], в которой объекты и связи между ними представляются в виде триплетов: «объект, отношение, субъект». Основа RDF – это представление данных в виде утверждений-триад субъект-предикат-объект, описывающих направленную связь от субъекта к объекту. В отличие от реляционной модели, имеющей жесткую структуру, модель RDF достаточно гибкая – каждый субъект может содержать свои собственные предикаты и объекты.

Первым шагом осуществляется преобразование логической структуры проектируемой БД в набор триплетов: «субъект – предикат – объект». Например, логическая структура базы данных (Рисунок), описанная на языке SQL:

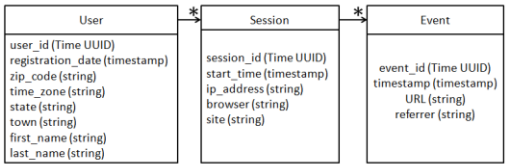

Рисунок 2 – Логическая модель базы данных

может быть приведена к следующему виду:

User, is_a, table
User, has_name,”user”
User, has_attribute, user_id
User_id, is_a, attribute
User_id, is_a, primary key
User_id, has_type, Time UUID
…

Данное преобразование осуществляется при помощи RuleEngine (процессора правил) с некоторым пользовательским набором правил (RuleSet). Выбранный метод организации этого преобразования позволяет добиться универсальности имитационной системы: меняя набор правил RuleSet, можно работать с базой, описанной на любом формальном языке [5].

На основании полученного набора триплетов генерируется множество вариантов логической структуры данных в нереляционном (NoSQL) представлении. Одним из возможных способов отображения логической структуры данных на физическую является представление данных в виде разложения их на множество строк и столбцов. Каждая строка и каждый столбец адресуются уникальным ключом (rowkey и columnkey соответственно).

Данные в NoSQL структуре данных хранятся в виде: (Table, RowKey, Family, Column, Timestamp) → Value, где левая часть образует ключ, а правая – значение. Значение не играет важной роли при моделировании работы распределенной высоконагруженной системы, а ключ напрямую определяет скорость доступа к данным. Его выбор определяется несколькими составляющими, часть которых (RowKey, ColumnFamily, Column) вариативная. Именно от способа выбора ключевых полей зависит, как распределятся данные по кластеру и, как следствие, как быстро можно будет получить к ним доступ при выполнении запросов.

Ключ строки может быть выбран разными способами, а именно это может быть:

– одно поле из таблицы (уникальное поле или первичный ключ).

– конкатенация или комбинирование нескольких полей из таблиц.

– хэш-функция от поля(-ей).

Поля, не вошедшие в ключ строки, должны быть распределены между семейства колонок (column families). Это делается в целях облегчения управления и манипуляции данными, а также ускорения выполнения запросов. Так как все данные каждой column family хранятся в фиксированном наборе файлов, то выбор распределения столбцов напрямую влияет на скорость выполнения запросов. Выбор способа разбиения данных зависимости опрееляется множеством основных запросов Q. Таким образом, поля, которые встречаются вместе в одном запросе целесообразно выделить в одну column family.

Генерация NoSQL-структуры может быть произведена программным способом (полный перебор вариантов ключа и способов выбора column family) либо пользователем. В имитационной системе производится генерация нереляционных структур на основе выбора пользователя (предусмотрена возможность задания способов группировки и вариантов выбора ключа).

На следующем шаге для каждого варианта NoSQL структуры данных может быть сгенерировано множество вариантов физической организации хранения информации. Распределение данных по машинам кластера можно производить несколькими способами [6]:

По первичным ключам и их производным (хеш-кодам):

– partition key = hash function (дата) mod N,

где partition key – номер машины, где будут размещены данные;

hash function – некоторая хэш-функция;

N – число машин в вычислительном кластере;

– атрибутам данных (например, данные 2001-2003 годов находятся на машине номер m, данные 2004-2005 годов – на машине n и т.д);

– композитным ключам;

– таблицам;

– столбцам и записям таблиц;

В имитационной системе моделируется шардирование с применением хеш-функции от ключа записи. Используется MD5-хеш от ключа строки [7].

Таким образом, может быть достаточно большое количество вариантов физической реализации для каждого из вариантов NoSQL архитектуры системы.

Основной единицей масштабируемости и балансировки нагрузки является регион (region). Регионы - это непрерывные диапазоны строк, хранящиеся вместе. Они могут динамически расщепляться (split) системой, в том случае если они становятся слишком большими по объему. При добавлении данных в регион, система следит за тем, чтобы он не превысил заданный размер. Если достигнут предел, регион будет разделен по среднему ключу (middlekey) на два. Один из новых регионов останется на прежнем сервере, а второй может быть перенесен на другой, менее загруженный. На этом же этапе осуществляется моделирование реплицирования данных региона. Каждый регион обслуживается одним регион-сервером (region server), и каждый из этих серверов может обрабатывать несколько регионов в одно и то же время.

При добавлении информация сохраняется в памяти (memstore). Как только объем данных в memstore достигает максимально установленного размера, они записываются (операция flush) на диск. В то время пока информация пишется из памяти на диск, система может продолжать производить операции чтения и записи без блокировки. Это достигается разделением memstore, а именно, пустая часть памяти принимает обновления, а заполненная в то же время конвертируется в файл.

Информация, находящаяся в memstore, уже отсортирована по ключам. Таким образом, информация просто переписывается на диск без сортировки или какой-либо другой обработки. При чтении и выполнении запросов происходит слияние (merge) данных, хранящихся в memstores, и данных на диске.

По причине необходимости создания все большего и большего количества файлов во время перезаписи из памяти на диск, использует механизм объединения файлов в более крупные (compaction), используя слияние. Существует два типа слияния: minor compactions и major compactions. Первое уменьшает число хранимых файлов с помощью перезаписи многочисленных маленьких файлов в несколько крупных. Major compaction перезаписывает все файлы одного семейства колонок для региона в один новый файл. В спроектированной имитационной системе имитируется первый тип слияния.

Существует два основных компонента: один основной сервер (master server) и множество регион-серверов (region servers). Регион-сервера – хранилища данных. Основной сервер ответственен за назначение регионов регион-серверам и за распределение данных по регион-серверам. Также master-сервер отвечает за балансировку нагрузки между регион-серверами. Регион-сервера ответственны за все операции чтения и записи для всех регионов, которые они обслуживают.

При записи данных на регион-сервер запросы от клиента на запись поступают на мастер-сервер, который хранит информацию о регионах. Далее мастер сервер передает данные на нужный регион, т.е. на регион-сервер, который обслуживает данный регион.

Если рассмотреть блок распределения данных по кластеру как «черный ящик», то на вход данному блоку подается структура и конфигурация кластера, а на выходе поступает кластер, заполненный сгенерированными данными.

Данные для наполнения имитационной системы генерируются порциями: задается некоторый объем данных, которые необходимо распределить по системе. Каждая запись в имитационной системе состоит из ключа и значения. В качестве значения для удобства моделирования храним объем обрабатываемых данных. В процессе наполнения имитационной системы тестовыми данными заданного объема генерируется пара KeyValue, которая затем отправляется на MasterServer для распределения ее в нужный регион.

После моделирования распределения данных по физическим устройствам хранения можно осуществить тестирование выбранного варианта архитектуры. В качестве параметра быстродействия системы выбрано время выполнения запросов из «главного» множества, при этом оценивается время обработки запроса в чистом виде, т.е. без накладных расходов добавление данных, отказ оборудования и т.п. Время выполнения запроса на выборку данных для одной пары Key-Value можно определить формулой:

T = TRead(RowKey) + K1*TRead(Value) + K2*(TParse(RowKey) + TCMP(RowKey)) + K3*(TParse(Value) + TCMP(Value)) + K4*TTransfer(RowKey) + K5*TTransfer(Value),

где TRead(RowKey) – время чтения RowKey;

TRead(Value) – время чтения Value;

TParse(RowKey) – время выделения поля из RowKey для сравнения;

TCMP(RowKey) – время сравнения поля из RowKey;

TParse(Value) – время выделения поля из Value для сравнения;

TCMP(Value) – время сравнения поля из Value;

TTransfer(RowKey) – время передачи RowKey;

TTransfer(Value) – время передачи Value;

K1 = if(Если все поля из запроса находятся в RowKey):0:1;

K2 = if(Если в RowKey нет полей в блоке WHERE):0:1;

K3= if(Если в Value нет полей в блоке WHERE):0:1;

K4 = if(Если Key Value не подходит под критерии запроса):0:1;

K5 = if(Если K4=1 и в запросе есть поля из Value):1:0.

При моделировании выполнения запросов и расчете времени их выполнения, помимо структуры хранения, учитываются физические характеристики юнитов в кластере распределенной высоконагруженной вычислительной системы.

Результаты имитационного моделирования работы распределенной высоконагруженной системы отображаются графически (рисунки 3,4).

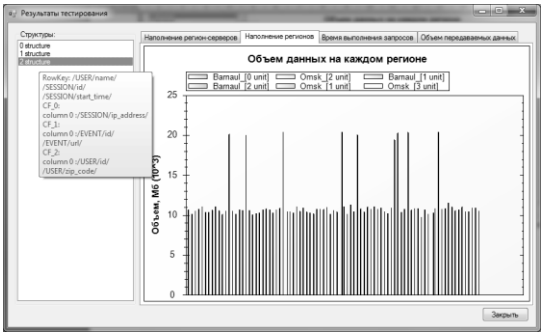

Рисунок 3 – Пример графика наполнения регионов

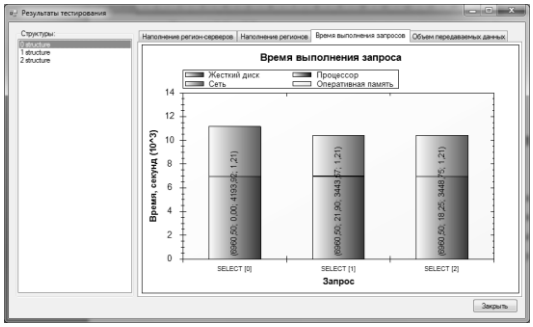

Рисунок 4 – Пример графика времени выполнения запроса

В ходе дальнейших исследований предполагается использовать для моделирования выполнения процессов добавления данных с учетом дублирования и репликации данных одновременно с процессом выполнения запросов по выборке данных аппарат теории массового обслуживания. При этом предлагается использовать в качестве обслуживающих устройств регион-сервера, а в качестве событий различных типов – запросы на запись и выборку данных.

Вывод

Таким образом, в ходе работы имитационной системы исследователь получает возможность оценить характеристики работоспособности различных вариантов структуры данных системы Big Data, построенных на основе часто выполняющихся запросов Q и учитывающих способы хранения и распределения данных между узлами системы, характеристики оборудования и программного обеспечения, а также способы оптимизации системы путем партиционирования, денормализации и агрегации. Наличие же для каждого из найденных вариантов структуры данных оценки его эффективности в виде оценки времени выполнения запросов из «главного» множества Q позволит выбрать наиболее оптимальный вариант или скорректировать описание модели и перейти на следующую итерацию цикла моделирования.

СПИСОК ЛИТЕРАТУРЫ

1. Viktor Mayer-Schonberger, Kenneth Cukier. Big Data: A Revolution That Will Transform How We Live, Work, and Think – NY.: Eamon Dolan/Houghton Mifflin Harcourt 2013, 256 с.

2. ANTLR [Электронный ресурс] - 2013. – Режим доступа: http://www.antlr.org/

3. Parr T. The Definitive ANTLR Reference. Building Domain-Specific Language [Текст] / T. Parr - Raleigh: The Pragmatic Bookshelf, 2007. – 369 с.

4. Головков В. RDF — инструмент для неструктурированных данных [Текст] / В. Головков, А. Портнов, В. Чернов // Открытые системы. – 2012. - №9. – с. 46-49.

5. Malcolm Chisholm. How to Build a Business Rules Engine, 1st Edition - Morgan Kaufmann, 2003, 483 с.

6. Hrishikesh Karambelkar. Scaling Big Data with Hadoop and Solr – Packt 2013, 144 с.

7. Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger: MD5 considered harmful today - Creating a rogue CA certificate: [Электронный документ]. – (http://www.win.tue.nl/hashclash/rogue-ca/). Проверено 10.10.2013.

Харламов Алексей Иванович – аспирант, тел.: (3852) 290786; Сучкова Лариса Иннокентьевна – к.т.н., профессор; Бочкарёва Екатерина Владимировна – к.т.н., инженер, e-mail: 22bev@mail.ru.