тематической модели радиометрической цепочки в алгоритме пирометра.

3. Вследствие постоянного изменения условий измерения для описания величин, входящих в состав модели, целесообразно использовать методы теории вероятностей и статистической теории.
4. Результаты моделирования показывают, что использование коррекции на основе априорной информации позволяет снизить погрешность измерения температуры в 5 раз и более.

СПИСОК ЛИТЕРАТУРЫ

1. Госсорг, Ж. Инфракрасная термография. Основы, техника, применение [Текст] / Ж. Госсорг/ – М.: Мир, 1988. – 416 с.
2. Ионов, Б.П. Спектрально-статистический подход к бесконтактному измерению температуры [Текст] / Б.П. Ионов, А.Б. Ионов // Датчики и системы. – 2009. – №2. – С. 9-12.
3. Магунов, А.Н. Спектральная пирометрия [Текст] / А.Н. Магунов // Приборы и техника эксперимента. – 2009. – № 4. – С. 5-28.
4. Свет, Д.Я. Оптические методы измерения истинных температур [Текст] / Д.Я. Свет. – М.: Наука, 1982. – 296 с.
5. Вольф, У. Справочник по инфракрасной технике [Текст] / У. Вольф, Г. Цисис. – Т1. Физика ИК-излучения – М.: Мир, 1995 г., 606 с.
6. Левин, Б.Р. Теоретические основы статистической радиотехники [Текст] / Б.Р. Левин. – М.: Радио и связь, 1989. – 656 с.
7. Слынько, Ю.В. Модель расчета переноса излучения на основе открытой базы данных HITRAN [Текст] / Ю.В. Слынько // Вопросы радиоэлектроники. – 2007. – №4. – С. 5-11.

Старший преподаватель, к.т.н. ***А.Б. Ионов*** *– antionov@mail.ru; с.н.с., к.т.н. Б.П. Ионов; аспирант* ***А.И. Мирная*** *– aly_mir@mail.ru; магистрант* ***Е.В. Плоткин*** *– ega-vp@mail.ru – Омский государственный технический университет, кафедра «Радиотехнические устройства и системы диагностики» (3812)65-25-98.*

УДК 004.42

ИССЛЕДОВАНИЕ СХЕМ ХРАНЕНИЯ ИНФОРМАЦИИ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ С УЧЕТОМ ОСНОВНЫХ ЗАКОНОМЕРНОСТЕЙ ДОСТУПА К ДАННЫМ

Харламов А.И., Сучкова Л.И., Бочкарева Е.В.

В статье описан подход к построению и оценке оптимальности различных вариантов структуры данных для высоконагруженных систем, основанный на имитационном моделировании. Предложенный метод позволяет выбрать способ хранения и распределения данных между узлами системы, характеристики оборудования и хранилищ данных, а также используемое программное обеспечение. При этом учитывается специфика выполняемых запросов и способы оптимизации структуры системы путем партиционирования, денормализации и агрегации.

Ключевые слова: оптимизация, запрос данных, высокая нагрузка, имитационное моделирование, Big Data, распределенная система, структура данных, план выполнения, Apache HBase, партицирование, процессор правил.

Введение

Организация обработки данных в высоконагруженных распределенных системах является важной практической задачей. В настоящее время к названному классу можно отнести биржевые системы, системы электронной коммерции, рекламные сети и т.п.

Основные проблемы технической реализации систем описанного класса связаны с невозможностью использования общепринятых решений и недостаточной производительностью отдельной аппаратной системы. Кроме того, входные данные для таких систем часто являются машинно - генерируемыми, что приводит к необходимости обработки и хранения 50 000 и более событий в секунду и терабайтов информации в день. Будем называть такие системы **системами Big Data**.

Для решения задач такого масштаба традиционные методы, такие как OLTP базы данных или построение OLAP витрин данных (data mart), экономически невыгодны в связи с длительностью цикла подготовки данных и стоимостью программно-аппаратного обеспечения [1]. В связи с этим для их решения обычно используют массивно-параллельные решения, в которых каждый узел кластера хранит и обрабатывает только часть общего массива данных [2]. Выбор оптимальной схемы хранения данных при этом является нетривиальной задачей.

В существующих работах авторами затрагиваются некоторые аспекты выбора оптимальной структуры распределенной базы данных, в частности партиционирование [3-6]. Однако приемы оптимизации в реальных приложениях часто включают денормализацию и построение агрегатов, не рассмотренные в проанализированных разработках. Денормализация при этом предполагает приведение структуры данных в состояние, не соответствующее критериям нормализации, например, путем сокращения количества таблиц за счет их объединения или ввода дополнительного поля (расчетного значения, длинного поля и т.п.). Построение агрегатов же позволяет учесть методы группировки данных. Эти приемы позволят повысить скорость выполнения запросов и, как следствие, производительность системы Big Data в целом.

Целью работы, таким образом, является поиск для системы Big Data оптимальной структуры, что подразумевает выбор способа хранения и распределения данных между ее узлами, подбор оборудования и хранилищ данных, а также построение соответствующего программного обеспечения. При этом необходимо учесть специфику выполняемых запросов, а также способы оптимизации структуры системы путем партиционирования, денормализации и агрегации.

Описание предлагаемого метода

Для решения данной задачи предложено использовать подход, основанный на имитационном моделировании. Опишем структуру данных в виде ER-диаграмм логической модели данных, а процессы загрузки и выборки данных - в виде множества главных запросов Q. Обычно к данному множеству относятся наиболее часто выполняемые запросы, быстрая обработка которых критично отражается на производительности системы в целом. Варьируя способы отображения логической структуры данных на физическое расположение и структуру данных v, будем оценивать оптимальность архитектуры как функцию времени выполнения главного множества запросов F(v,Q).

Для оценки полученных вариантов физической структуры данных введем функцию стоимости, которую можно приближенно описать в виде соотношения:

$$J(v, Q) = F(v,Q) + \lambda * C(v) \rightarrow \min, \quad (1)$$

где $v \in V$ – множество вариантов физической структуры данных; Q – множество главных запросов; F – задаваемая исследователем функция стоимости; λ - коэффициент регуляризации, C(v) – штрафная функция. Минимизация идет по времени выполнения запросов множества Q. Регуляризационный компонент λ*C(v) позволяет избежать подгонки модели (overfitting). В качестве штрафной функции C(v) возьмем сумму количества получившихся в ходе денормализации таблиц, индексов и агрегатов.

Для исследования времени выполнения запросов из Q строится множество возможных планов выполнения каждого из запросов. Далее на основе данных, полученных при имитационном моделировании, вычисляется время обработки запроса в соответствии с соотношением:

$$F(v, Q) = \mathrm{Sum}(q| q \in Q, t(q, v) * Pq), \quad (2)$$

где Pq - вероятность запроса q из Q, t(q, v) - время выполнения запроса согласно наилучшему плану на варианте структуры v. Так же функция стоимости может включать и другие характеристики аппаратных ресурсов: пропускную способность сети и дисковых систем и т.п.

Применение предлагаемого метода для решения реальной прикладной задачи

Для систем обработки больших объемов данных характерной является задача исследования поведения пользователя, подразумевающая, в частности, поиск ответов на следующие вопросы: какие действия выполнял пользователь; какова средняя продолжительность сессии его работы на ресурсе; какие операции были выполнены в заданный промежуток времени.

Для решения описанного класса задач традиционно используются нереляционные

базы данных с дизайном на основе Google Big Table [7]. В том числе **Apache HBase** – нереляционная, распределённая база данных с открытым исходным кодом на Java, которая разрабатывается в рамках проекта Apache Hadoop (Apache Software Foundation) и обеспечивает отказоустойчивый способ хранения больших объёмов разреженных данных [8]. Кроме того, Apache HBase не использует сложных стратегий управления хранением данных, например, нечетких контрольных точек и упреждающего чтения, что позволяет использовать более простой аппарат для оценки стоимости выполнения запроса.

Для описания этой задачи построим ER-модель данных (рисунок 1):

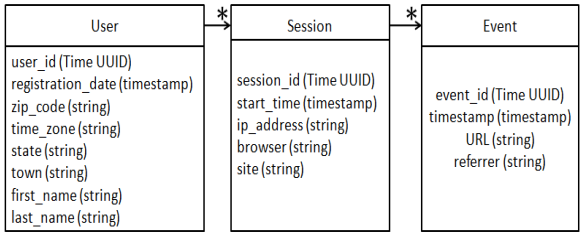

Рисунок 1 - Логическая модель базы данных

В качества источника исходной информации также выступает следующее множество запросов:

1. SELECT E.* FROM Session S
   INNER JOIN Event ON S.session_id = E.session_id
   WHERE S.user_id=*USER* (все действия пользователя *USER*);
2. SELECT * FROM Session S
   INNER JOIN Event E ON S.session_id = E.session_id
   WHERE S.user_id=*USER*
   AND E.timestamp BETWEEN [*TIME1, TIME2*]
   (действия пользователя *USER* в заданном промежутке времени);
3. SELECT S.session_id, S.start_time,
   COUNT(E.id), MAX(E.timestamp)
   FROM Session S
   INNER JOIN Event E ON S.session_id = E.session_id
   WHERE S.user_id=*USER*
   GROUP BY S.session_id, S.start_time
   (средняя продолжительность времени пребывания в системе пользователя *USER* – в секундах и событиях);
4. SELECT SITE_SECTION(E.url),
   TIME_SPENT(E.timestamp)
   FROM Session S
   INNER JOIN Event E ON S.session_id = E.session_id
   WHERE S.user_id=*USER*
   GROUP BY SITE_SECTION(E.url)
   ORDER BY E.timestamp
   (количество времени, проведенного пользователем *USER* в разных разделах первого уровня).

На примере этих запросов можно проиллюстрировать влияние выбора способа разбиения данных (ключа разбиения, **partition key**) по машинам и дискам кластера на время выполнения запроса и, как следствие, на производительность системы в целом. Например, в случае выбора session_id ключом разбиения таблицы Event выполнение запроса вида

SELECT * FROM Event
WHERE timestamp BETWEEN [jan 01; feb 01],

потребует просмотра данных всех машин. Напротив, если выбрать в качестве ключа разбиения поле timestamp, то запрос будет выполнятся только на машинах, которые потенциально могут содержать эти данные. С другой стороны, если необходимо выполнить поиск всех действий конкретного пользователя, то запрос выполнится быстрее при использовании для разбиения поля user_id или его хеш-кода.

Часто в качестве основания для разбиения данных используют составной ключ (**composite key)**, который образуется путем комбинирования или конкатенации различных полей базы данных. Например, для приведенной выше базы возможен следующий вариант:

partition key = user_id+session_id.

Использование ключей такого вида для разбиения данных между физическими хранилищами позволит существенно ускорить выполнение ряда запросов.

Рассмотрим алгоритм работы имитационной системы, схематичное изображение которого приведено на рисунке 2.

Рисунок 2 - Укрупненный алгоритм работы имитационной системы

Для удобства работы опишем логическую модель данных, приведенную выше, при помощи **Resource Description Framework** (triplets, relational form) [9]. В этой нотации объекты и связи между ними представляются в виде триплетов: «объект, отношение, субъект». Например структура данных, приведенная выше в табличной форме, может быть описана следующим образом:

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

Проанализируем главные запросы множества Q с целью выявления полей-кандидатов для индексирования и агрегирования. Таким образом, на подготовительном этапе (рисунок 2, блок 0) на вход системы поступает описание логической структуры данных в виде триплетов, которое включает информацию о таблицах, связях между ними, полях и их типах, а также множество главных запросов Q. Эти данные являются исходными для этапа порождения множества возможных вариантов структуры данных (рисунок 2, блок 1).

Далее, используя набор эвристических правил, переведем исходное описание во множество вариантов представления логической структуры данных в терминах HBase. Этот переход осуществляется при помощи так называемого **rule engine** - компонента исполнения бизнес-правил [10]. Иногда сервера исполнения бизнес-правил дополнительно предоставляют средства для редактирования, классификации, управления, проверки целостности (согласованности) и тестирования правил, ориентированные на специалиста в предметной области, но не программиста, а также репозитории для хранения правил. Использование механизма бизнес-правил позволяет легко модифицировать логику работы системы, не изменяя при этом ее внутреннюю структуру и алгоритмы работы. Одним из способов для описания правил может служить язык запросов SPARQL [11].

Набор правил для перехода от логической структуры к представлению данных в виде терминах Hbase (rule set1) является базой знаний для системы. При этом важно учесть многовариантность трансформации: для одной логической структуры данных может существовать достаточно большое (если вообще ограниченное) количество вариантов HBase представления. Использование знаний о главных запросах позволяет уменьшить количество возможных вариантов HBase представления, так как бессмысленно индексировать поля или создавать агрегаты, не входящие в запросы из множества Q. Также, исходя из этой информации, можно определить **column families** – наборы ячеек строки, который содержит столбцы связанных по смыслу данных, использующихся вместе в запросах, и хранится совместно в одном файле (рисунок 3).

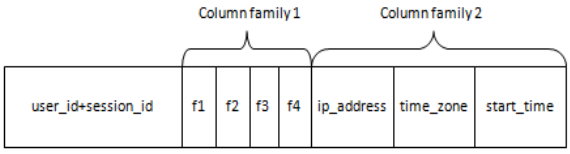

Рисунок 3 - Пример выбора column families

Применение правил из rule set 1 приводит к лавинообразному появлению различных вариантов HBase структуры данных, так как она может варьироваться по следующим параметрам:

- варианты логической организации таблиц базы данных;
- выбор полей и типов индексов для каждой таблицы;
- варианты денормализации данных (например, за счет хранения вычисляемых и агрегированных значений, часто

фигурирующих в запросах, с целью уменьшения времени выполнения последних).

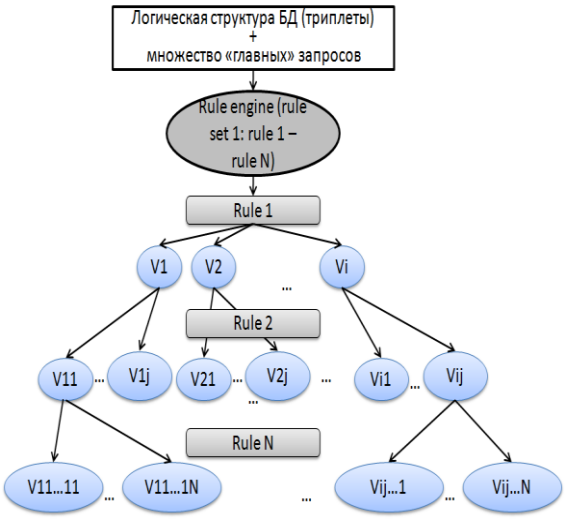

Рисунок 4 - Поиск вариантов структуры данных в терминах HBase на основе rule set 1.

На следующем шаге для каждого варианта HBase структуры данных генерируется множество вариантов физической организации данных (рисунок 2, блок 3). При этом варьируются следующие параметры:

- возможные варианты распределения данных по физическим устройствам. При этом разбиение можно проводить по
  - первичным ключам и их производным (хеш-кодам). Например, partition key = hash function (дата) mod N,
    где partition key – номер машины, где будут размещены данные, hash function – некоторая хэш-функция, а N – число машин в вычислительном кластере;
  - атрибутам данных: Например, данные 2001-2003 годов находятся на машине номер k, данные 2004-2005 годов – на машине l и т.д.;
  - композитным ключам
  - таблицам;
  - столбцам и записям таблиц;
- репликация таблиц
- разбиение столбцов по семействам (column families).
- Варианты разбиения данных между машинами и дисками строятся имитационной системой при помощи rule engine с набором правил rule set 2, с правилами вида:
- Обработка триплета «user, is_a, table», подразумевающая необходимость создания физической таблицы user. То есть, правило вида «IF X is_a table, THEN create hbase_table X».
- Добавление ячейки в таблицу: «IF X is_a table AND X has_attribute Y? THEN create column Y in table X».

Таким образом, на выходе порождающего слоя системы (рисунок 2, блок 3) имеем множество вариантов физической структуры данных (множество V). Далее для каждого варианта структуры v из множества V необходимо построить план выполнения каждого запроса q из множества Q и оценить стоимость его выполнения t(v,q). Выполнение запроса при этом условно можно условно разбить на следующие стадии:

1. разбор запроса в абстрактное синтаксическое дерево;
2. определение набора операций реляционной алгебры: projection, join, filtering, groupping, sorting;
3. определение последовательности выполнения алгоритмов, реализующих операции реляционной алгебры (hash join, nested loop join, merge join и т.д.);
4. собственно выполнение.

Для оценки времени выполнения запроса для конкретной пары “вариант структуры данных, вариант плана выполнения запроса” t(v,q) достаточно знаний о наборе, порядке выполнения и оценке времени работы алгоритмов, реализующих операции реляционной алгебры, необходимые для обработки запроса. А поскольку HBase используется на доступных серийных серверах, то дисковые системы имеют достаточно простую структуру, поведение которой достаточно просто моделировать.

Следует учитывать, что в ходе исследования возможны изменения множества главных запросов Q и множества порождающих правил из rule set 1 и rule set 2. После внесения изменений в описание модели системы Big Data генерация вариантов ее структуры и исследование характеристик каждого из них с использованием имитационной системы повторяется с новыми входными данными. То есть процесс является итеративным.

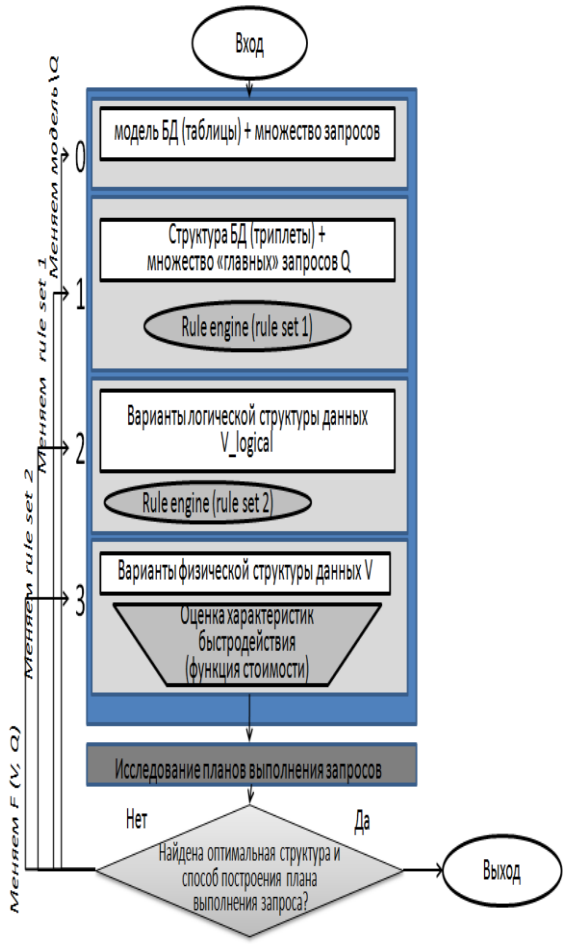

Рисунок 5 - Итеративный процесс поиска оптимальной структуры данных.

Вывод

Таким образом, после работы имитационной системы исследователь получает множество 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 функции стоимости (1) позволит выбрать наиболее оптимальный вариант или скорректировать описание модели и перейти на следующую итерацию цикла моделирования.

СПИСОК ЛИТЕРАТУРЫ

1. Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. Методы и модели анализа данных: OLAP и Data Mining – СПб: БХВ-Петербург, 2004 – 336 с.
2. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. – М.: Вильямс, 2003 – 1440 с.
3. Mearian L. Data growth remains IT's biggest challenge, Gartner says: [Электронный документ]. – (http://www.computerworld.com/s/ article/ 9194283/ Data_growth_remains_IT_s_biggest_challenge_Gartner_says). Проверено 19.11.2011.
4. S.Papadomanolakis, A.Ailamaki. AutoPart: Automated schema design for large scientific databases using data partitioning. In Proceedings of the 16th International Conference on and Statistical Database Management, 2004.
5. Stratos Papadomanolakis, Anastassia Ailamaki, Workload-Driven Schema Design for Large Scientific Databases, Bulletin of the Technical Committee on Data Engineering, p. 21, vol. 27(4), (2004).
6. Graefe G. Query evaluation techniques for large databases. [Электронный документ] // Journal ACM Computing Surveys (CSUR). – 1993. - Vol.25 Iss.2. – (http://dx.doi.org/10.1145/152610.152611 Query evaluation techniques for large databases). Проверено 15.12.2011.
7. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber, Bigtable: A Distributed Storage System for Structured Data [Электронный документ] (http://research.google.com/archive/bigtable.html) Проверено 23.07.2012.
8. The Apache Software Foundation, Apache HBase [Электронный документ] (http://hbase.apache.org/) Проверено 11.16.2012.
9. Joshua Tauberer Quick Intro to RDF [Электронный документ] (http://rdfabout.com/quickintro.xpd) Проверено 06.05.2012.
10. Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, Foundations of Semantic Web Technologies - NY: Taylor&Francis Group, 2010 - 427 с.
11. SPARQL Query Language for RDF [Электронный документ] (http://www.w3.org/TR/rdf-sparql-query/) Проверено 06.05.2012.

*Аспирант **Харламов А.И.**, к.т.н., профессор **Сучкова Л.И.**, инженер **Бочкарева Е.В.** - Алтайский Государственный Технический Университет им. И.И. Ползунова (г. Барнаул).*