НЕКОТОРЫЕ АЛГОРИТМЫ ПОТОКОВОЙ ОБРАБОТКИ ДАННЫХ

С.С. Божко[1]
Магистрант кафедры Информатики, разработчик в Adform, Республика Беларусь

И.И. Пилецкий[2]
Научный руководитель совместной лаборатории БГУИР-IBA и АЦКТ IBM, главный архитектор отделения по программному обеспечению IBA IT Park, кандидат физико-математических наук,
Республика Беларусь

К.Ю. Слисенко[3]
Старший разработчик в компании JazzTeam, магистр технических наук,
Республика Беларусь

[1]*Белорусский государственный университет информатики и радиоэлектроники, svt.bozhko@gmail.com*
[2]*Белорусский государственный университет информатики и радиоэлектроники, ianmenski@gmail.com*
[3]*Белорусский государственный университет информатики и радиоэлектроники, kslisenko@gmail.com*

This paper will describe some algorithmic techniques developed for handling large amounts of data that is often available in limited ways. Topics that will be covered include data stream algorithms that allow real-time processing of data, scale extraordinarily well, and are simple to implement, random algorithms and probabilistic data structures. They are algorithmically efficient and can provide shockingly good practical results.

Существуют различные определения больших данных, в которых используются такие понятия, как объем, скорость, достоверность и неструктурированность данных. Термин «большие данные» также применяется, когда традиционные решения, например реляционные базы данных, становятся слишком медленными, слишком маломощными либо слишком дорогими в сопровождении и использовании. Поэтому существует проблема обработки больших объемов данных, которая традиционными средствами не может быть решена. Более того, в настоящее время многие задачи бизнеса требуют обработки сверхбольших объемов данных в реальном времени. Существующая инфраструктура Hadoop в полной мере не удовлетворяет потребности бизнеса в силу своей нацеленности на отложенное выполнение задач.

Lambda-архитектура [1] помогает решить данную проблему. Она сочетает в себе два похода для обработки поступающих данных: потоковый (stream) в режиме реального времени и серийный пакетный режим (batch).

Следует отметить, что обработка и анализ большого количества данных в хранилищах – это сложная задача, но обработка бесконечного потока данных – более простая. Офлайн-обработка подразумевает единовременный расчет всей выборки как одного целого. Онлайн-алгоритмы приспособлены для того, чтобы обрабатывать данные по мере их получения: порция данных в единицу времени, когда есть лимитированное количество памяти и вычислительной мощности.

В задачах промышленного характера зачастую достаточно знать приблизительное значение с заранее установленной погрешностью. Среди них: подсчет примерного количества уникальных элементов в большой или бесконечной (потоковой) выборке данных; оптимизация запросов с использованием эффективных, но неточных фильтров; определение частотного распределения уникальных элементов в очень большом массиве данных. Задачи, которые требуют подсчета максимально, минимального и среднего значений, в потоковой обработке решаются тривиально, поэтому не будут рассматриваться.

Основные требования, которые предъявляются к потоковой системе: масштабирование без необходимости дополнительного администрирования; получение результатов в реальном времени, в отличие от Hadoop; по возможности простота технического решения.

Для подсчета статистики в потоковых системах широко применимы вероятностные структуры данных. Но ценой за скорость и масштабирование является точность результирующих данных. Применяемые подходы позволяют получить ответ, точность которого с высокой вероятностью близка к реальному.

Так, фильтры Блума [2] по своей сути – это структура данных set, которая может помочь ответить на вопрос о принадлежности какого-то элемента данному множеству. Особенно полезно ее применять, когда доступ к некоторым данным дорогостоящий. Жертвуя детерминизмом, структура данных показывает высокую эффективность по памяти: обычно в несколько раз эффективнее, чем эквивалентная хеш-таблица. Отрицательной характеристикой данной структуры является возможность ложноположительных ответов (элемента в множестве нет, но структура данных сообщает, что он есть). Ярким примером использования является Google BigTable. Фильтры Блума применяются для уменьшения числа обращений к жесткому диску при проверке на существование заданной строки или столбца в таблице базы данных [3]. Также важной особенностью является тот факт, что данный алгоритм легко поддается параллелизации и распределению на вычислительные мощности кластера. Для генерации k хеш-значений можно посчитать два хеш-значения с разными значениями ключа внутреннего состояния (seed), а затем линейно интерполировать k значений между этими хеш-значениями.

Задача частотного анализа элементов множества: например, нахождение наиболее активных пользователей, получение числа уникальных пользователей,

сгруппированных по ip, дате, сегменту. Существует множество подходов к решению этой задачи. Наиболее интересными являются алгоритмы Space Saver [4] и разновидности LogLog [5].

Алгоритм Space Saver широко используется в параллельно-потоковой обработке данных. В памяти хранится k кортежей: элемент и соответствующее ему количество упоминаний. Когда на вход подается элемент, происходит проверка, есть ли уже в памяти данный элемент. Если есть, то увеличивается счетчик, иначе элемент с наименьшей частотой заменяется на входной, при этом снова происходит инкремент количества упоминаний. Данный алгоритм детерминирован, использует O(k) памяти, константное время обновления, а степень ошибки зависит от распределения данных.

В алгоритме LogLog используются не сами данные, а их хеш-значения. Поэтому очень важно выбрать подходящую хеш-функцию. Она должна быть детерминированной: для одинаковых входных значений выходные параметры должны быть неизменными; и при этом давать равномерное распределение. Для этих целей есть готовые библиотеки для разных языков программирования. В бинарном представлении хеш-значения максимально длинная цепочка нулей говорит о том, насколько редкому значению принадлежит данное хеш-значение. Исходя из этого можно сделать вывод о том, как много уникальных элементов в множестве. Для увеличения точности обычно используют несколько хеш-функций, а затем берут среднее арифметическое их значений. При этом подсчет хеш-функций может быть легко распараллелен. Улучшения данного алгоритма: SuperLogLog, HyperLogLog (HLL), HyperLogLog++. В HyperLogLog вместо среднего арифметического рассчитывается среднее гармоническое. В совокупности с другими улучшениями это дает более высокую точность результатов. Часто есть необходимость в отслеживании динамики количества уникальных пользователей. Для этого можно, конечно, использовать несколько HyperLogLog, а потом их сравнивать, но существует более приемлемый подход: HyperLogLog с плавающим окном.

Рассмотренные алгоритмы потоковой обработки данных могут быть применены для разработки приложений как на платформе open-source, так и на платформе enterprise, потому как они не привязаны к какому-то конкретному технологическому продукту. В настоящее время некоторые алгоритмы нашли широкое применение в сфере рекомендательных систем для онлайн-рекламы. Например: «In the online advertising company I previously worked for, we used them to track thousands of metrics across billions of data points comfortably in memory on a single machine» [6]; «For the past two years, we've been using HLL at AK to do just that: count the number of unique users in a stream of ad impressions» [7].

Литература

1. Lambda Architecture. [Электронный ресурс]. – Режим доступа: – http://lambda-architecture.net/. – Дата доступа: 16.05.2015.

2. Paulo Sergio, Almeida Carlos Baquero, Nuno Preguica. Scalable Bloom Filters. [Электронный ресурс]. – Режим доступа: – http://gsd.di.uminho.pt/members/cbm/ps/dbloom.pdf. – Дата доступа: 16.05.2015.

3. 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 ресурс]. – Режим доступа: – http://research.google.com/archive/bigtable.html. Дата доступа: 16.05.2015.

4. Ahmed Metwally, Divyakant Agrawal Amr El Abbadi. Efficient Computation of Frequent and Top-k Elements in Data Streams. [Электронный ресурс]. – Режим доступа: – https://icmi.cs.ucsb.edu/research/tech_reports/reports/2005-23.pdf. Дата доступа: 16.05.2015.

5. Marianne Durand, Philippe Flajolet. Loglog counting of large cardinalities. [Электронный ресурс]. – Режим доступа: – http://www.ic.unicamp.br/~celio/peer2peer/math/bitmap-algorithms/durand03loglog.pdf. Дата доступа: 16.05.2015.

6. PWLSF#13 – Armon Dadgar on Bloom Filters and HyperLogLog. [Электронный ресурс]. – Режим доступа: – https://www.youtube.com/watch?v=T3Bt9Tn6P5c. Дата доступа: 16.05.2015.

7. Neustar Research. HyperLogLog Engineering. [Электронный ресурс]. – Режим доступа: – http://research.neustar.biz/tag/hyperloglog/. Дата доступа: 16.05.2015.