УДК 658.5.011

АРХИТЕКТУРА БОЛЬШИХ ДАННЫХ

В.В. Малиновская
Магистрант БГУИР,
проектный менеджер в игровой мобильной индустрии

В.Ф. Алексеев
Доцент кафедры проектирования информационных компьютерных систем, кандидат технических наук, доцент

Кафедра проектирования информационно-компьютерных систем факультета компьютерного проектирования Белорусского государственного университета информатики и радиоэлектроники, Республика Беларусь
Белорусский государственный университет информатики и радиоэлектроники, Республика Беларусь
ООО «СейГеймс», Республика Беларусь
E-mail: viktoria.malinovskaya7@gmail.com

В.В. Малиновская

Окончила Белорусский государственный университет информатики и радиоэлектроники. Магистрант кафедры проектирования информационных компьютерных систем БГУИР. Работает в SayGames в должности проектного менеджера. Проводит исследования организации процессов управления на предприятии.

В.Ф. Алексеев

Окончил Минский радиотехнический институт. Область научных интересов связана с разработкой методов и алгоритмов построения информационно-компьютерных систем, исследованием проблем тепловой нестационарности полупроводниковых структур, изучением проблем обеспечения электромагнитной совместимости радиоэлектронных средств, организацией учебного и научно-исследовательского процессов в техническом университете.

Аннотация. В статье обобщаются основные принципы обработки больших объемов данных, а также объясняется, как архитектура для обработки больших данных позволяет принимать, обрабатывать и анализировать данные, которые являются слишком объемными или слишком сложными для традиционных систем баз данных.

Ключевые слова: Big Data, Database, Data Processing, Architecture, Data Access.

Введение.

За последние 20 лет ИТ-специалисты научились обрабатывать постоянно растущий объема данных с помощью технологий, таких как реляционные базы данных *OLTP* (*Online Transaction Processing*), хранилище данных, *ETL* (*Extraction, Transformation and Loading*) и *OLAP* (*Online Analytical Processing*), отчеты по большим данным, а теперь и облачные и *IoT*.

Все эти технологии были реализованы благодаря быстрому росту вычислительной мощности, в частности, совершенствование процессоров, памяти, места хранения и скорости сети.

В конвейере данных, данные обычно проходят два этапа: обработку и обращение к ним. Для любого типа данных, когда они поступают в организацию, они либо не являются чистыми, либо не имеют формата, который может быть сообщен или проанализирован

непосредственно конечными бизнес-пользователями внутри или за пределами организации. Поэтому сначала требуется обработка данных, которая обычно включает очистку, стандартизацию, преобразование и агрегирование данных. Обработанные данные затем представляются на уровне доступа к данным – готовым для отчетности и используются для аналитики во всех аспектах. Обработка данных включает в себя подготовку данных, интеграцию данных или *ETL*, среди них *ETL*, вероятно, самое популярное название.

Обработка данных и доступ к ним преследуют разные цели и поэтому достигаются с помощью различных технологий. Data Processing for big data подчеркивает «масштабирование» с самого начала, что означает, что при увеличении объема данных время обработки по-прежнему должно соответствовать ожиданиям с учетом имеющихся аппаратных средств. Общее время обработки данных может варьироваться от минут и часов до нескольких дней в зависимости от объема данных и сложности логики обработки [1–8].

С другой стороны, доступ к данным подчеркивает «быстрое» время отклика порядка секунды. На высоком уровне масштабируемость обработки данных достигается главным образом параллельной обработкой, в то время как быстрый доступ к данным достигается оптимизацией структуры данных, а также увеличенных объемов памяти, доступной на серверах.

Обработка данных.

Для очистки, стандартизации и преобразования данных из различных источников обработка данных должна касаться каждой записи в последующих данных. Как только запись будет очищена и завершена, работа будет выполнена. Это принципиально отличается от доступа к данным – последний приводит к повторяющемуся извлечению и доступу к одной и той же информации с разными пользователями и/или приложениями.

Когда объем данных мал, сложность обработки данных является менее сложной, чем в сравнении с доступом к данным, и поэтому обычно происходит внутри той же базы данных, где находятся окончательные данные. По мере роста объема данных было установлено, что обработка данных должна осуществляться вне баз данных, с тем чтобы обойти все накладные расходы и ограничения, вызванные системой баз данных, которая явно непредназначена для обработки больших данных. Это было тогда, когда *ETL* и затем *Hadoop* начали играть критическую роль в эпохе хранения данных и больших данных соответственно.

Проблема обработки больших данных заключается в том, что объем обрабатываемых данных всегда находится на уровне того, что может хранить жесткий диск, но намного больше объема вычислительной памяти, доступной в данный момент времени.

Основным способом эффективной обработки данных является разделение данных на более мелкие части и их параллельная обработка.

Другими словами, масштабируемость достигается за счет того, что сначала обеспечивается возможность параллельной обработки так, что при увеличении объема данных количество параллельных процессов будет увеличиваться, в то время как каждый процесс продолжает обрабатывать такое же количество данных, как и ранее; во-вторых, добавляя больше серверов с большим количеством процессоров, памяти и дисков по мере увеличения числа параллельных процессов [8].

Параллельная обработка больших данных была впервые реализована методом разбиения данных в системах баз данных и *ETL*-инструментах. После логического разделения набора данных каждый раздел может обрабатываться параллельно. *Hadoop HDFS* (высокораспределенные файловые системы) адаптирует тот же принцип наиболее масштабируемым путем. *HDFS* разбивает данные на блоки с постоянным размером. Затем блоки распределяются по различным узлам сервера и записываются хранилищем метаданных в так называемом узле *Names*. Когда начинается процесс передачи данных, количество процессов определяется количеством блоков данных и доступных ресурсов

(например, процессоров и памяти) на каждом узле сервера. Это означает, что *HDFS* обеспечивает массовую параллельную обработку при наличии достаточного количества процессоров и памяти от нескольких серверов.

В настоящее время *Spark* стал одним из самых популярных сервисов для масштабной обработки данных в памяти.

В пространстве больших данных объем обрабатываемых больших данных всегда намного больше объема доступной памяти. Прежде всего, *Spark* использует общий объем памяти в распределенной среде с несколькими узлами данных. Однако объем памяти все еще недостаточен и может быть дорогостоящим, если какая-либо организация попытается вписать большие данные в кластер *Spark*. Авторами рассмотрено, для какого типа обработки подходит *Spark*.

Обработка данных всегда начинается с считывания данных с диска в память, а в конце записи результатов на диски. Если каждую запись необходимо обработать только один раз перед записью на диск, что характерно для обычной пакетной обработки, *Spark* не даст преимуществ по сравнению с *Hadoop*.

С другой стороны, *Spark* может хранить данные в памяти для нескольких этапов преобразования данных, в то время как *Hadoop* не может. Это означает, что *Spark* предлагает преимущества при итеративной обработке одной и той же части данных несколько раз, что именно то, что необходимо в аналитике и машинном обучении.

Другой актуальной темой в области обработки данных является обработка потока. Это дает большие преимущества в снижении скорости обработки, поскольку в данный момент времени требуется обрабатывать только небольшой объем данных при каждом поступлении данных.

Однако это не так универсально, как пакетная обработка в двух аспектах: первый заключается в том, что входные данные должны поступать в «потоковом» режиме, а второй заключается в том, что определенная логика обработки, которая требует агрегирования по периодам времени, все еще должна обрабатываться в пакетном режиме после этого.

Наконец, облачные решения дают возможность масштабировать распределенную систему обработки более динамично на основе объема данных, а следовательно, и количества параллельных процессов. Этого трудно достичь на предприятии, поскольку новые серверы должны планироваться, закладываться в бюджет и закупаться. Если емкость не планируется должным образом, обработка больших данных может быть либо ограничена количеством оборудования, либо дополнительная покупка приводит к растрате ресурсов без использования. Обработка в облаке получает большое преимущество эластичности инфраструктуры, что может дать больше гарантий достижения наилучшего баланса более экономичным способом.

Доступ к данным.

По сравнению с обработкой данных доступ к данным имеет очень разные характеристики, включая:

1. Зависит от того, как приложения или пользователи должны извлекать данные.
2. Паттерны поиска данных должны быть поняты.
3. Объем данных должен быть целевым и должен содержать часть доступных данных.

С учетом вышеизложенных принципов за последние два десятилетия было несколько вех, которые отражают, как получить доступ к постоянно растущему объему данных при одновременном возврате запрошенных данных в течение нескольких секунд:

Хранение данных. Требуется избегать табличных соединений, которые могут быть очень дорогими, когда объем данных большой. Здесь появляется понятие «таблица фактов», в котором все столбцы объединяются без принципов нормализации базы данных, как в реляционной базе данных [3].

Место хранения. Каждый столбец хранится и индексируется, и поэтому доступ к нему

осуществляется отдельно. Это дает более быстрое время отклика, чем доступ к обычным реляционным базам данных на основе строк, когда строка содержит много столбцов, тогда как запросы извлекают только несколько столбцов одновременно.

База данных NoSQL. Устраняет соединения и реляционную структуру и адаптируется к быстрому извлечению данных более специфическим образом.

База данных в памяти. Обеспечивает высокую производительность за счет хранения всей базы данных или всей таблицы в памяти.

На рисунке 1 приведены некоторые популярные примеры каждого типа базы данных.

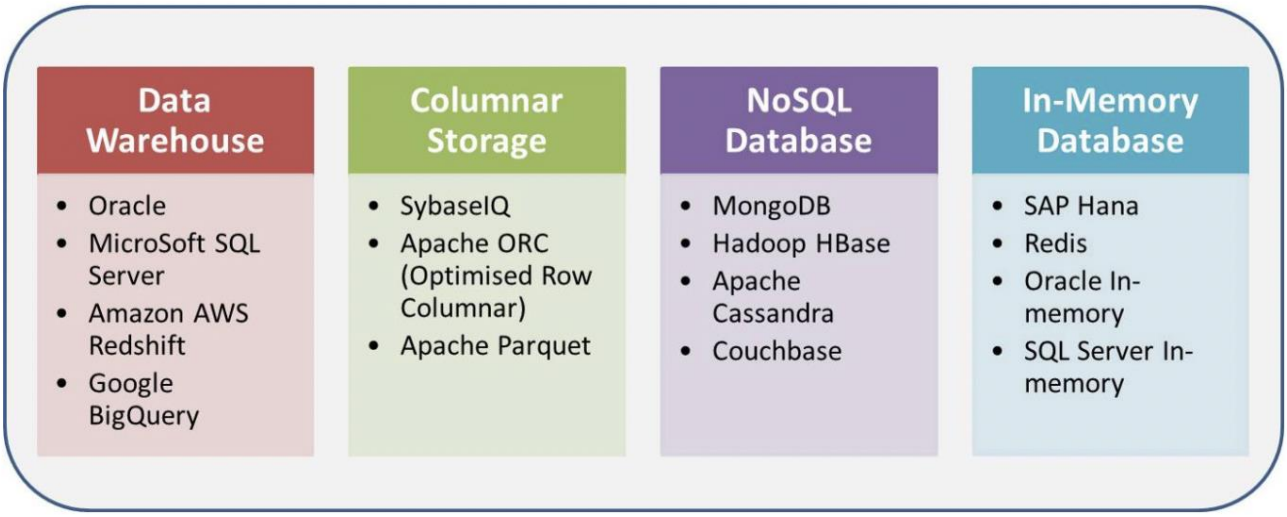

Рисунок 1. Примеры типов базы данных

База данных может объединять более одной технологии. Например, *Redis* является базой данных *NoSQL*, а также in-memory. Кроме того, для извлечения данных из хранилищ данных и *Columnar Storages* используются параллельные процессы. Поскольку существует множество вариантов различных типов баз данных в зависимости от содержания данных, структуры данных и узоров поиска пользователями и/или приложениями, доступ к данным является областью, в которой организация должна быстро и постоянно развиваться. Иногда могут использоваться различные типы баз данных или инструментов одновременно для разных целей.

Заключение.

Большое различие между обработкой данных и доступом к данным заключается в том, что доступ к данным в конечном итоге исходит от потребностей клиентов и бизнеса, а выбор правильной технологии стимулирует будущие разработки новых продуктов и расширяет возможности пользователей.

С другой стороны, обработка данных является основным активом компании, а обработка в масштабе и обеспечение хорошего качества данных является необходимым фактором, позволяющим компании наращивать объем своих данных.

Многие компании испытывают “слежку” за своей системой обработки данных, когда объем данных растет, и восстановление платформы обработки данных с нуля обходится дорого.

Принцип параллельной обработки данных и масштабируемости должен быть тщательно продуман и разработан с самого начала.

Обработка данных также осуществляется параллельно с управлением данными и интеграцией данных – все три необходимы для успеха любой организации, занимающейся интенсивным использованием данных. Кроме того, каждая организация в настоящее время сталкивается с множеством вариантов решений для больших данных как от сообществ с

открытым исходным кодом, так и от сторонних поставщиков. Четкое понимание различий между обработкой данных и доступом к данным позволяет лидерам ИТ и бизнеса не только создавать надежную архитектуру данных, но и принимать правильные решения для ее расширения и модернизации на постоянном темпе.

Список литературы

[1] Фрэнкс Билл. Революция в аналитике. Как в эпоху Big Data улучшить ваш бизнес с помощью операционной аналитики / Билл Фрэнкс. – М.: Альпина Паблишер, 2016. — 430 с.

[2] Вайгенд Андреас. BIG DATA. Вся технология в одной книге / Андреас Вайгенд. – Москва: Эксмо, 2018. — 384 с.

[3] Алексеев, В.Ф. Информационная поддержка управления инновационной деятельностью предприятия / В.Ф. Алексеев, Д.В. Лихачевский, В.В. Хорошко // BIG DATA and Advanced Analytics = BIG DATA и анализ высокого уровня: сб. материалов VI Междунар. науч.-практ. конф., Минск, 20-21 мая 2020 года: в 3 ч. Ч. 3 / редкол.: В.А. Богуш [и др.]. – Минск: Бестпринт, 2020. – С. 412 – 417.

[4] Алексеев, В.Ф. Разработка онлайн платформы оценки и финансирования инновационных проектов / В.Ф. Алексеев, Д.В. Лихачевский, Г.А. Пискун // BIG DATA Advanced Analytics: collection of materials of the fourth international scientific and practical conference, Minsk, Belarus, May 3 – 4, 2018 / editorial board: M. Batura [etc.]. – Minsk, BSUIR, 2018. – P. 398 – 404.

[5] Алексеев, В.Ф. Взаимосвязь операционного и логистического менеджмента в структуре маркетинговой и производственной стратегии фирмы / В.Ф. Алексеев // Экономическое развитие общества: инновации, информатизация, системный подход. Международная научно-практическая конференция. Тезисы докладов. − Минск: Изд-во «ПАРАДОКС», 2008. – С. 310–312.

[6] Алексеев, В.Ф. Анализ системы маркетинга на предприятии и её совершенствование с использованием Internet-технологий / В.Ф. Алексеев [и др.] // Современные информационные компьютерные технологии: Сб. науч. ст. в 2ч. Ч.1 − Гродно: ГрГУ, 2008. – С. 118–122.

[7] Tadviser.ru [Электронный ресурс]. – Режим доступа: https://www.tadviser.ru

[8] Microsoft [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com

[9] Молодой учёный [Электронный ресурс]. – Режим доступа: https://moluch.ru

BIG DATA ARCHITECTURE

V.V. Malinovskaya
BSUIR Master,
Project Manager
SayGames

V.F. Alekseev
Associate Professor, Department of Information Computer Systems Design,
Candidate of Technical sciences,
Associate Professor

Department of Information and Computer Systems Design
Faculty of Computer Engineering
Belarusian State University of computer science and Radio Electronics, Republic of Belarus
EPAM Systems, Republic of Belarus
E-mail: viktoria.malinovskaya7@gmail.com

Abstract. The paper summarizes the basic principles of processing large amounts of data, and explains how the architecture for processing big data allows you to receive, process and analyze data that is too large or too complex for traditional database systems.

Keywords: Big data, Database, Data Processing, Architecture, Data Access