УДК 004.6

ПЕРСПЕКТИВЫ И ТЕНДЕНЦИИ РАЗВИТИЯ КОНЦЕПЦИИ БОЛЬШИХ ДАННЫХ В СОВРЕМЕННОЙ IT ИНДУСТРИИ

С.Г. СУРТО
(Полоцкий государственный университет)

Рассмотрены определение термина «большие данные», характерные особенности этой технологии и области применения. Изучены способы работы с большими данными, проанализированы существующие способы реализации концепции больших данных, на основании чего были выявлены потенциальные области для дальнейших исследований.

Ключевые слова: *большие данные, анализ, статистика, структурированные данные, структурированные данные.*

Введение. В учреждениях здравоохранения пациенты не имеют произвольного доступа к своей истории болезни. Такие данные могут быть получены только при посещении врача. С помощью мобильных устройств люди могут создавать свои собственные базы данных. Приложения помогают следить за частотой пульса, гаджеты – отслеживать физическую активность и сохранять все накопленные данные. Пользователи могут извлекать эти данные и рассчитывать на профессиональную помощь специалистов в их обработке (интерпретации). В этой связи революция в области больших данных важна в плане оценки перспективности и прибыльности тех или иных случаев. Например, юристы смогут оперативнее находить лучшие аргументы в делах и рассчитывать стоимость услуг исходя из большого объема статистических данных. Используя искусственный интеллект, можно эффективно работать с большими и сложными массивами данных. Они не только улучшают обучение, этот инструмент позволяет руководству образовательных учреждений лучше понимать студентов и их действия. Активность студентов может отслеживаться и сохраняться в виде баз данных, анализ которых может подсказать преподавателю на чем заострять внимание на занятиях. Эта сфера имеет, пожалуй, наибольший потенциал для эффективного использования больших данных. В розничных инвестициях опытные люди больше не полагаются на брокеров, они принимают решения на основе доверенных компаний, обрабатывающих данные.

Современные тенденции в развитии концепции больших данных можно сформулировать следующим образом.

1. Быстрое увеличение объемов данных. Рост объемов данных актуализировал проблему оперативной аналитики полученной информации с целью извлечения полезных сведений. Организации должны перерабатывать терабайты сырых данных во что-то полезное.
2. Широкое использование Hadoop. Все больше организаций используют Hadoop и другие хранилища данных. С его помощью организации могут обрабатывать большие объемы данных, используя продвинутую аналитику для нахождения ценной информации и выработки выгодных решений.
3. Прогнозирующая аналитика. Точное предсказание будущего поведения и событий для увеличения прибыли.
4. Больший фокус на анализ данных на основе облачных технологий. Перенос анализа данных в облако ускоряет принятие современных технологий по обработке данных. Уменьшается стоимость обслуживания и оперирования в связи с тем, что данные обрабатываются в облаке.
5. Виртуализация данных. Этот процесс позволяет увидеть скрытое в огромных объемах данных, манипулировать информацией независимо от того, как данные отформатированы или где расположены.
6. Конвергенция IoT (Internet of Things), облака, больших данных и кибербезопасности. Конвергенция технологий управления данными, таких как качество данных, подготовка данных, аналитика, интеграция данных и др., привела к повышению роли кибербезопасности, поскольку все больше используются умные устройства, обмен данными и машинное обучение.

1. Сущность концепции больших данных

Понимание термина «большие данные». Большие данные (англ. big data, [ˈbɪg ˈdeɪtə]) – обозначение структурированных и неструктурированных данных огромных объемов и значительного многообразия, эффективно обрабатываемых горизонтально масштабируемыми (англ. scale-out) программными инструментами, появившимися в конце 2000-х годов и альтернативных традиционным системам управления базами данных и решениям класса Business Intelligence [12].

Под Big Data можно понимать именно методы обработки данных, позволяющие обрабатывать информацию параллельно, распределяя ее между различными вычислительными ресурсами, а не объем самих данных. Эти методы можно применить как к огромным массивам данных, так и к небольшим.

Несколько примеров того, что может быть источником данных, для которых необходимы методы работы с большими данными:

– логи поведения пользователей в интернете;
– GPS-сигналы от автомобилей для транспортной компании;
– данные, снимаемые с датчиков в большом адронном коллайдере;
– информация о транзакциях всех клиентов банка.

Принципы работы с большими данными. Исходя из определения Big Data, можно сформулировать основные принципы работы с такими данными:

1. **Горизонтальная масштабируемость.** Поскольку объемы данных постоянно растут (особенно при способе хранения, характерном для больших данных), то логично, что системы для хранения должны легко расширяться простым добавлением новых единиц хранения (например, компьютеров) в кластер.

2. **Отказоустойчивость.** Объем хранимых данных теоретически не ограничен, поэтому количество компьютеров, участвующих в распределенной сети, обеспечивающей хранение и доступ к данным, может быть огромным (пример: Hadoop-кластер Yahoo имеет более 42 000 машин). Это неизбежно ведет к периодическим отказам оборудования. Методы работы с большими данными должны учитывать эту особенность и обеспечивать бесперебойный доступ к данным в таких условиях.

3. **Локальность данных.** В распределенных системах данные распределены по большому количеству источников. В случае нахождения данных на одном сервере, а обработки – на другом, время их передачи может превысит время самого процесса обработки. При этом одним из лучших вариантов решения является обработка данных на той же системе, на которой они хранятся.

Сообщество разработчиков открытого кода создало за последние годы немало технологий для больших данных. Среди них можно выделить пять категорий.

1. **Системы пакетных вычислений.** Обладают высокой производительностью, но и большой задержкой. Могут выполнять вычисления практически любого объема, однако это может занять много времени (час, день). Пример такой системы – Hadoop, состоящий из двух проектов: Hadoop distributed file system (HDFS) и Hadoop MapReduce – горизонтально масштабируемый каркас вычислений, интегрируемый с HDFS.

2. **Каркасы сериализации.** Представляют инструменты и библиотеки, позволяющие переносить объекты данных между системами, работающими на различных языках программирования. Иначе говоря, объект сериализуется из исходных кодов на одном языке, а потом десериализуется в объект на другом языке. Примеры каркасов сериализации – Apache Thrift, Protocol Buffers, Avro.

3. **Базы данных типа NoSQL с произвольным доступом.** За последние годы было создано довольно много распределенных нереляционных баз данных (NoSQL): Cassandra, HBase, MongoDB, Riak, CouchDB и др.

4. **Система обмена сообщениями и постановки их в очередь.** Позволяют обмениваться сообщениями между процессами, при этом процесс обмена является отказоустойчивым и асинхронным. Например, Apache Kafka.

5. **Система вычислений в реальном времени.** Отличительные особенности таких систем – это высокая производительность, малая задержка, способность обрабатывать поток данных. Не могут выполнять операции, на которые способны системы пакетной обработки, но способны очень быстро обрабатывать сообщения.

Хранение необработанных данных. Информационная система отвечает на вопросы об информации, приобретенной в прошлом, при этом заранее неизвестно на какие вопросы придется отвечать. В связи с этим большое значение приобретает вопрос сохранения необработанных (сырых) данных. Сохраняя данные в как можно более необработанном виде, можно максимально увеличить возможность получать новые представления данных. При этом чем более необработанными оказываются данные, тем больше места они занимают при хранении [5]. Технологии больших данных специально предназначены для работы с петабайтами и даже экзабайтами данных, позволяя сохранять данные распределенным, масштабируемым способом.

Неизменяемость данных. Еще одним важным аспектом является неизменяемость данных. Это свойство нехарактерно для реляционных баз данных, но в случае больших данных ситуация иная. Неизменяемость данных дает как минимум два огромных, в условиях большого объема информации, преимущества:

1. **Устойчивость к отказам, связанным с человеческим фактором.** В случае с изменяемыми данными неверное обновление данных ведет к их потере, при использовании концепции неизменяемых данных это не ведет к потерям, т.к. все предыдущие записи (версии) этих данных будут сохранены.

2. **Простота.** В моделях изменяемости данных подразумевается, что данные индексируются для извлечения и обновления конкретных записей. В моделях же неизменяемости данных есть только возможность дополнять массив новыми данными. Для этого не требуется индексация. Пример, как может выглядеть модель для больших данных предложен в таблице, где представлены сведения о месте жительства людей, идентифицируемых по User_id

Таблица. – Данные места жительства

User_id	Location	Timestamp
1	Витебск, РБ	29.03.2003 08:24:15
2	Полоцк, РБ	15.04.2003 09:08:23
3	Брест, РБ	05.07.2004 15:21:11
1	Минск, РБ	14.01.2005 17:35:21

Человек идентифицируемый по User_id = 1 в 2003 году жил в Витебске, а в 2005 переехал в Минск, при этом запись об этом просто добавилась в таблицу с новым timestamp (таблица). Такая схема хранения данных позволяет хранить несколько записей для одного человека, сохраняя всю историю изменений.

Главное следствие неизменяемости данных состоит в том, что каждый фрагмент данных оказывается истинным всегда. Однажды став истинным, фрагмент должен оставаться таковым навсегда. Без этого свойства неизменяемость данных не имеет смысла, в таблице показано как фрагмент данных сделать истинным вечно, обозначив его отметкой времени.

В общем массив данных постоянно растет в результате ввода новых неизменяемых данных, но имеются особые случаи, когда данные приходится удалять.

– **Сборка «мусора».** Удаляются фрагменты данных, имеющих малую ценность. С помощью сборки мусора можно контролировать рост главного массива данных. Например, можно реализовать правило, предусматривающее сохранение только одного места жительства на каждого человека в год.

– **Нормативы.** Государственные нормативные документы, которые требуют очистки содержимого баз данных при определенных условиях.

Несмотря на то, что данные вечно истинны, можно предпочесть вообще «забыть» об информации, потому что это придется сделать или потому что она не представляет достаточной ценности по сравнению с затратами на ее хранение.

Парадигма MapReduce. MapReduce – это модель распределенной обработки данных, предложенная компанией Google для обработки больших объемов данных на компьютерных кластерах [5].

MapReduce предполагает, что данные организованы в виде определенных записей. Обработка данных происходит в 3 стадии:

1. **Стадия Map.** На этой стадии данные предобрабатываются при помощи функции **map()**, определенной пользователем. Работа этой стадии заключается в предобработке и фильтрации данных. Функция **map()**, примененная к одной входной записи, выдает одно или множество пар ключ–значение. Что будет находится в ключе и в значении – решать пользователю, но ключ – очень важная вещь, так как данные с одним ключом в будущем попадут в один экземпляр функции **reduce**.

2. **Стадия Shuffle.** На этой стадии вывод функции **map** «разбирается по корзинам» – каждая корзина соответствует одному ключу вывода стадии **map**. В дальнейшем эти корзины послужат входом для **reduce**.

3. **Стадия Reduce.** Каждая «корзина» со значениями, сформированная на стадии **shuffle**, попадает на вход функции **reduce()**. Функция **reduce** задается пользователем и вычисляет финальный результат для отдельной «корзины». Множество всех значений, возвращенных функцией **reduce()**, является финальным результатом MapReduce-задачи.

Несколько дополнительных фактов про MapReduce:

1) Все потоки вычисления **map** и **reduce** работают независимо и могут работать параллельно, в том числе на разных машинах кластера, т.е. горизонтально масштабируемы.

2) **Shuffle** представляет из себя параллельную сортировку, поэтому также может работать на разных машинах кластера, обеспечивая горизонтальное масштабирование.

3) Функция **map**, как правило, применяется на той же машине, на которой хранятся данные, – это позволяет снизить задержки, связанные с передачей данных по сети (принцип локальности данных).

4) **MapReduce** – это всегда полное сканирование данных, никаких индексов нет. Это означает, что MapReduce неэффективен, когда ответ требуется очень быстро, с другой стороны, для задач, требующих сканирования полного массива данных, производительность максимальная.

В качестве примера задачи, эффективно решаемой при помощи MapReduce, можно привести классическую задачу Word count (подсчет слов).

К примеру, нужно подсчитать количество повторений слов в предложении «*Cats could not speak, could not at all.»*

Стадия **map** на выходе даст такой результат в виде пар ключ–значение:

{Cats: 1, could: 1, not: 1, speak: 1, could: 1, not: 1, at: 1, all: 1}.

Далее, **Shuffle**, получая исходные данные с выхода **map**, преобразует их в следующий вид:

{Cats: (1), could:(1, 1), not: (1,1), speak:(1), at:(1), all:(1)}.

После этой стадии убираются дубликаторы ключей, размер данных существенно уменьшается.

На последней стадии **Reduce** получают удобно подготовленные данные и остается подсчитать количество элементов в поле «значение» для каждого ключа, в результате получится

{Cats: (1), could:(2), not: (2), speak:(1), at:(1), all:(1)}. Что и будет решением задачи.

2. Реализация концепции

Hadoop. Изначально Hadoop был, в первую очередь, инструментом для хранения данных и запуска MapReduce-задач, сейчас же Hadoop представляет собой большой стек технологий, так или иначе связанных с обработкой больших данных (не только при помощи MapReduce) [3].

Основными (core) компонентами Hadoop являются:

1) Hadoop Distributed File System (HDFS) – распределенная файловая система, позволяющая хранить информацию практически неограниченного объема;
2) Hadoop YARN – фреймворк для управления ресурсами кластера и менеджмента задач, в том числе включает фреймворк MapReduce;
3) Hadoop common.

Существует большое количество проектов, непосредственно связанных с Hadoop, но не входящих в Hadoop core:

1) Hive – инструмент для SQL-like запросов над большими данными (превращает SQL-запросы в серию MapReduce-задач);
2) Pig – язык программирования для анализа данных на высоком уровне. Одна строчка кода на этом языке может превратиться в последовательность MapReduce-задач;
3) Hbase – колоночная база данных, реализующая парадигму BigTable;
4) Cassandra – высокопроизводительная распределенная key-value база данных;
5) ZooKeeper – сервис для распределенного хранения конфигурации и синхронизации изменений этой конфигурации;
6) Mahout – библиотека и движок машинного обучения на больших данных.

Отдельно стоит отметить проект Apache Spark, представляющий собой движок для распределенной обработки данных. Apache Spark обычно использует компоненты Hadoop, такие как HDFS и YARN, для своей работы, при этом сам проект в последнее время стал популярнее, чем Hadoop.

3. Методы исследований

Семантические структуры и причинные модели больших данных для принятия решений. В настоящее время проблема обработки больших данных относится к числу наиболее актуальных в области информационных технологий, и она же порождает наиболее трудные проблемы алгоритмического характера, связанные с обеспечением точности, устойчивости и вычислительной эффективности процессов их обработки. Эти проблемы обусловлены тем, что большинство традиционных методов интеллектуального анализа данных напрямую не могут быть применены для анализа больших данных либо вследствие вычислительной неустойчивости, либо вследствие вычислительной сложности. Не менее трудные проблемы обусловлены гетерогенным характером больших данных: они могут содержать атрибуты разных типов и неструктурированные данные, например, тексты на естественном языке [11].

Одно из важных требований к методам обработки больших данных – это семантически ясная интерпретация результатов. В современных моделях знаний это обеспечивается средствами онтологии, однако ее построение для больших данных также является проблемой: ввиду огромного количества потенциальных понятий ручная разработка онтологии становится непомерно трудоемкой, а потому требует максимальной автоматизации, а в ряде классов приложений – полной автоматизации [11].

Одним из решений является разработка алгоритмов обучения и принятия решений в задачах классификации на основе семантических и причинных моделей больших данных. Применяются алгоритмы поиска множества причинных зависимостей в больших данных. Такие алгоритмы используют семантическую модель данных.

Анализ экспериментальных результатов показал, что с точки зрения поиска причинных связей в данных наиболее перспективными являются коэффициент регрессии, мера Клозгена, убеждение и фактор уверенности [11]. К наилучшим из них относятся две меры, а именно: коэффициент регрессии

$$R(A,B) = \frac{(p_{AB} - p_A \cdot p_B)}{p_A \cdot (1 - p_A)} \quad (1)$$

и мера Клозгена

$$K(A,B) = \sqrt{p_B} \cdot (p_{BA} - p_B)\,. \quad (2)$$

В формулах (1) и (2) A и B – это атрибуты данных булевого типа, интерпретируемые как случайные события, представленные выборкой больших данных; p – выборочные вероятности атрибутов (случайных событий), указанных в нижнем индексе p [11].

Программный алгоритм оптимизации размещения данных при многоуровневом хранении на основе эвристического подхода. Широко распространенным подходом к решению оптимизационных задач, которое не может быть найдено за полиномиальное время, является использование эвристических алгоритмов. К их основным преимуществам относятся высокая скорость выполнения и допустимый уровень качества найденного решения [10].

Алгоритм Greedy относится к семейству «жадных», основной особенностью которого является решение задачи выбором наиболее важных элементов с точки зрения определенного в алгоритме критерия. Он использует понятия задач и их интегрального показателя важности для реализации приоритетных особенностей в ходе его выполнения. В начале работы алгоритма вычисляется значение стоимости, ассоциируемое для всех задач на основе необходимых для ее выполнения файлов. Стоимость задачи складывается из значений стоимости всех файлов, необходимых для выполнения данной задачи. Стоимость файла рассчитывается как отношение суммы важности задач, к которым он относится, к числу файлов задачи [10]:

$$f_{icost} = \sum_{k=0}^{l} \frac{c_k}{T_{ksum}},$$

где c_k – важность k задачи;

l – задачи, к которым относится файл i;

T_{ksum} – сумма всех файлов в задаче k .

Затем производится сортировка всех задач по их стоимости. Файлы внутри каждой задачи также сортируются по стоимости. Таким образом получается список файлов, которые отсортированы по двум критериям: принадлежности к задачам по их стоимости и собственному значению стоимости. Алгоритм производит обработку списка файлов в очередности, полученной при помощи двухуровневой сортировки. За счет этого в первую очередь будут обрабатываться самые весомые для функции оценки оптимальности файлы в самых важных задачах.

Данный алгоритм позволяет достаточно быстро выработать решение допустимого качества, т.е. с положительным приростом к производительности. Однако вследствие сложной специфики задачи и односторонней эвристической логики алгоритма результат поиска решений близок к оптимальному только в достаточно простых случаях и далек от оптимального в сложных задачах.

Заключение. По результатам анализа развития больших данных можно сделать вывод о некоторых трудностях. Системы хранения для больших данных оптимизированы для хранения неизменяемых данных и плохо подходят для процедур случайной выборки. Алгоритмы пакетной обработки неоптимально работают с анализом потока новых данных. Кластеризация баз данных, как правило, требует больших усилий в настройке поскольку кластерная организация выполняет сложную задачу по обеспечению надежности хранения и доступности данных в условиях возможных сбоев отдельных узлов. Все эти узкие места являются потенциальной темой для исследований.

ЛИТЕРАТУРА

1. Schmarzo, B. Big Data: Understanding How Data Powers Big Business / B. Schmarzo. – M. : Wiley, 2013. – 240 с.
2. Frank, J.O. Big Data Analytics: Turning Big Data into Big Money (Wiley and SAS Business Series) / J.O. Frank – M. : Гостехиздат, 2012. – 176 с.

3. Prajapati, V. Big Data Analytics with R and Hadoop / V. Prajapati. – M., 2013. – 238 c.
4. Mayer-Schonberger, V. Big Data: Revolution That Will Transform How We Live, Work and Think / V. Mayer-Schonberger, K. Cukier. – Canada : Eamon Dolan/Houghton Mifflin Harcourt, 2013. – 242 p.
5. Dean, J. MapReduce: Simplfied Data Processing on Large Clusters / J. Dean, S. Ghemawat // In Sixth Symposium on Operating System Design and Implementation (OSDI'04), San-Francisco, CA, December, 2004.
6. Lammel, R. Google's MapReduce Programming Model – Revisited / R. Lammel // Science of Computer Programming. – Amsterdam, 2018. – 30 p.
7. Ghemawat, S. The Google file system / Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung // SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles, Bolton Landing, NY, USA, October 19–22, 2003. – Bolton Landing, 2003. – P. 29–43.
8. Zikopoulos, P. Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data / P. Zikopoulos, C. Eaton. – New York : McGraw-Hill Osborne Media, 2012. – 166 p.
9. Gorodetsky, V. Data-driven Semantic Concept Analysis for User Profile Learning in 3G Recommender Systems / V. Gorodetsky, O. Tushkanova // 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT) : Conference. – Singapore, 2015. – P. 92–97.
10. Hybrid Evolutionary Workflow Scheduling Algorithm for Dynamic Heterogeneous Distributed Computational Environment / D. Nasonov [et al.] // International Joint Conference SOCO'14-CISIS'14-ICEUTE'14, Bilbao, Spain, June 25th-27th, 2014. – C. 83–92.
11. Тушканова, О.Н. Сравнительный анализ численных мер оценки ассоциативных и причинных связей в больших данных / О.Н. Тушканова // Перспективные системы и задачи управления : материалы 10-й Всероссийской научно-практической конференции, Домбай, 6–10 апр. 2015 г., / ЮФУ. – Домбай, 2015. – Т. 2. – С. 54–65.
12. Atre, S. Business Intelligence Roadmap: The Complete Project Lifecycle for Decision-Support Applications / Shaku Atre, Larissa T. Moss. – Boston : Addison-Wesley Professional, 2003. – 576 p.

Поступила 24.02.2019

PROSPECTS AND TENDENCIES OF DEVELOPMENT OF THE BIG DATA CONCEPT IN MODERN IT INDUSTRY

S. SURTO

Big data in 2019 continue to gain popularity. The structured or raw, big data represent huge information massifs which at due processing can be a source of a bigger information. Volumes of the stored data in different spheres and also growth rates of speed of accumulation say that the relevance of methods of storage and information processing of large volumes will only grow. The traditional industries have also not avoided influence of the growing volume of data which need to be processed.

Keywords: *big data, analysis, statistic, structured data, unstructured data.*