ПРОГРАММНЫЕ ПЛАТФОРМЫ ДЛЯ РАБОТЫ С БОЛЬШИМИ ДАННЫМИ (BigData). СРАВНИТЕЛЬНЫЙ АНАЛИЗ. Часть 1

А.А. ЕГОРОВ (Журнал “Автоматизация и IT в энергетике”)

АВТОМАТИЗАЦИЯ & IT в энергетике
научно-производственный журнал

В статье рассмотрены толкования термина **BigData** различными специалистами. Приведено определение термина “Большие данные” (BigData) в соответствии с национальным стандартом РФ ГОСТ Р ИСО/МЭК ТО 20547-1-2021 “Эталонная архитектура больших данных, Часть 1. Структура и процесс применения”. Рассмотрено обоснование аналитиками емкой формулы Big Data, а именно, что в определении проекта Big Data должны фигурировать семь важных характеристик, “7 V”: Volume, Velocity, Variety, Veracity, Variability, Visualization, Value. Исходя из определения Big Data, сформулированы основные принципы работы с этими данными. В этой части статьи рассмотрены лучшие инструменты для анализа больших данных: классический метод **MapReduce**, **Apache Hadoop**, **DataRobot**, **Alteryx**, **H2O AI Hybrid Cloud, D3.js**, **Project Jupyter**. Перечислены их основные ключевые характеристики, на которые следует обращать внимание при выборе конкретной платформы для анализа данных пользователей.

Ключевые слова: *большие данные; **BigData**; ГОСТ Р ИСО/МЭК ТО 20547-1-2021; эталонная архитектура больших данных; характеристики “7 V”; **MapReduce**; **Apache Hadoo**; **DataRobot**; **Altery**; **H2O AI Hybrid Cloud**; **D3.js**; **Project Jupyter**.*

Экономический эффект систем ИИ на наше общество к 2030 году составит 15 триллионов долларов. Ни один сектор экономики не останется в стороне.

PwC, report “Sizing the prize”
Source: *https://preview.thenewsmarket.com/Previews/PWC/DocumentAssets/476830.pdf*
PwC report “Sizing the prize”

Термин **Big Data** появился сравнительно недавно. Google Trends показывает начало активного роста употребления словосочетания начиная с 2011 года [1].

При этом уже сейчас термин не использует только ленивый. Особенно часто не по делу термин используют маркетологи. Так что же такое **Big Data** на самом деле?

Сначала следует определиться с самим понятием **Big Data**.

В литературе можно встретить большое количество самых разных определений и толкований этого термина. Рассмотрим некоторые из них.

- **Big Data** – это когда данных больше, чем 100 Гб (500 Гб, 1 ТБ, кому как нравится).
- **Big Data** – это такие данные, которые невозможно обрабатывать в среде Excel.
- **Big Data** – это такие данные, которые невозможно обработать на одном персональном компьютере.
- **Big Data** – это вообще любые данные.

В **Wikipedia** [2] дается следующее определение: *Большие данные* (англ. *big data*) – серия подходов, инструментов и методов обработки структурированных и неструктурированных данных огромных объёмов и значительного многообразия для получения воспринимаемых человеком результатов, эффективных в условиях непрерывного прироста, распределения по многочисленным узлам вычислительной сети, сформировавшихся в конце 2000-х годов, альтернативных традиционным системам управления базами данных и решениям класса **Business Intelligence**.

Business intelligence (BI) – обозначение компьютерных методов и инструментов для организаций, обеспечивающих перевод транзакционной деловой информации в человекочитаемую форму, а также средства для массовой работы с такой обработанной информацией. Цель **BI** – интерпретировать большое количество данных, заостряя внимание лишь на ключевых факторах эффективности, моделируя исход различных вариантов действий, отслеживая результаты принятия решений.

В статье [3] рассматривается история разработки национального стандарта РФ ГОСТ Р ИСО/МЭК ТО 20547-1-2021 “Эталонная архитектура больших данных, Часть 1. Структура и процесс применения”. Отмечается, что цифровая трансформация промышленности сфокусировала внимание на необходимости разработки компьютерных систем и программного обеспечения для работы с большими и разнообразными наборами данных. В стандарте введены термины и определения в области архитектуры больших данных. Рассмотрены мотивы и цели стандартизации больших данных. Приведены основные понятия структуры и представления эталонной архитектуры больших данных. Рассмотрен процесс применения эталонной архитектуры больших данных. Определены функциональные компоненты для реализации деятельности по проектированию систем больших данных.

Этот стандарт идентичен техническому отчёту **ИСО/МЭК ТО 20547–1:2020** “Эталонная архитектура больших данных. Часть 1. Структура и прикладные процессы” (ISO/IEC TR 20547–1:2020 “Information technology – Big data reference architecture – Part 1: Framework and application process”, IDT).

Для целей стандарта используются следующие термины и определения. ГОСТ Р ИСО/МЭК ТО 20547-1-2021 2 ISO (ИСО) и IEC (МЭК) поддерживают терминологические базы данных для использования в стандартизации по следующим адресам:

- Онлайн-библиотека стандартов ISO (ИСО): доступна по адресу *https://www.iso.org/obp*
- Международный электротехнический словарь МЭК (IEC Electropedia): доступен по адресу *http://www.electropedia.org/*

Большие данные (big data): большие массивы данных, отличающиеся главным образом такими характеристиками, как объем, разнообразие, скорость обработки и/или вариативность, которые требуют использования технологии масштабирования для эффективного хранения, обработки, управления и анализа.

Рис. 1. Сложный мир больших данных [4]

Примечание: *Термин “большие данные” широко применяется в различных значениях, например, в качестве названия технологии масштабирования, используемой для обработки больших массивов данных [Международный стандарт ИСО/МЭК (ISO/IEC) 20546:2019, 3.1.2].*

Таким образом, под термином **Big Data** будем понимать не какой-то конкретный объём данных и даже не сами данные, а методы их обработки, которые позволяют обрабатывать информацию с помощью распределенных технологий вычислений. Эти методы можно применить как к огромным массивам данных (таким как содержание всех страниц в интернете), так и к сравнительно маленьким.

Итак, большие данные – это информационные активы, которые характеризуются большим объёмом, высокой скоростью и/или многообразием, а также требуют экономически эффективных инновационных форм обработки информации, что приводит к усиленному пониманию, улучшению принятия решений и автоматизации процессов (рис. 1).

Следует отметить, что ключевое замечание – слово “большие” в больших данных относится не только к объёму данных. Вы не просто получаете много данных, они поступают в реальном времени очень быстро и в различных комплексных форматах представления, а ещё, что очень важно, – из большого многообразия различных источников.

Аналитики придумали емкую формулу **Big Data** – они считают, что в определении проекта **Big Data** должны фигурировать семь важ-

ных характеристик, **“7 V”: Volume; Velocity; Variety; Veracity; Variability; Visualization; Value.** То есть объем, скорость, разнообразие, достоверность, изменчивость, визуализация, ценность. При этом каждая “V” важна для понимания общей картины [5].

Как правило меньше всего вопросов вызывают первые три “V”, а именно **Volume, Velocity, Variety.** Действительно, кто будет спорить, что **Big Data** – это прежде всего объем, **Volume**? Объем данных растет по экспоненте: например, самолеты ежегодно генерируют 2,5 млрд ТБ данных с различных датчиков, установленных в авиационных двигателях. При этом данные постоянно обновляются, генерируются новые данные, и скорость обновления (**Velocity** – вторая “V”) также важна для того, чтобы считать их “большими”. Например, каждую минуту в мире выполняется почти 2,5 миллиона запросов к поисковой системе Google. Задача проектов **Big Data** заключается в том, чтобы справиться с огромной скоростью, с которой данные формируются и анализировать их в режиме реального времени.

Третья “V” – **Variety**, разнообразие. Это означает, что проекты **Big Data** должны включать данные в самых различных форматах представления. К ним можно отнести структурированные и неструктурированные данные, текстовые, графические, данные корпоративной почты или соцсетей, вплоть до видеоинформации. Каждый из этих типов данных требует различных типов анализа и подходящих инструментов для обработки. Социальные медиа могут помочь владельцам бренда проанализировать настроения клиентов, а сенсорные данные предоставят информацию о том, как чаще всего используется продукт, чтобы применить эти знания для его улучшения.

Еще недавно трех “V” было вполне достаточно. Но все на свете изменяется, в том числе и подходы к определению. Поэтому аналитики присовокупили еще четыре “V”, чтобы более полно отразить понятие BigData. Поэтому в определение были добавлены **Veracity, Variability, Visualization, Value.** Рассмотрим каждое из этих понятий.

Veracity – Достоверность: безусловно, эта характеристика является крайне важной, поскольку любой анализ будет совершенно бесполезен, если данные окажутся недостоверными. Более того, крайне важно заранее удостовериться, что с данными все в порядке, ведь их неточность может привести к неправильным решениям и даже к аварийным ситуациям.

Variability – Изменчивость: новое веяние в сфере **Big Data**. Здесь речь идет о том, что значение одних и тех же данных может существенно различаться в зависимости от контекста, например, одни и те же слова в Твиттере могут иметь различные значения и отражать различные настроения. Следовательно, необходимо учитывать все нюансы! Для того чтобы выполнить правильный анализ настроений, алгоритмы должны быть в состоянии понять контекст и быть в состоянии расшифровать точное значение слова именно в этом контексте.

Visualization – Визуализация: это необходимая часть анализа, поскольку именно визуализация делает большие данные доступными для восприятия человеком. Визуализация больших объемов сложных разнотипных данных гораздо более эффективна и понятна для человека, чем электронные таблицы и отчеты, полные чисел и формул. Конечно, визуализация в рамках **Big Data** не означает построение обычных графиков или круговых диаграмм: возможно, будут построены сложные графики, которые будут включать в себя множество взаимосвязанных переменных данных, однако они все равно останутся понятными и воспринимаемыми человеком.

Value – Ценность: здесь говорится о том, чтобы извлечь максимум пользы из результатов анализа больших данных. Важно то, как будут использоваться эти данные и какую практическую пользу они принесут, например, для принятия своевременных и правильных решений.

Многие специалисты отмечают, что и этих семи “V” недостаточно для понимания сути **Big Data**. Это связано с тем, что все эти семь характеристик должны быть применены к сложной задаче, как правило, со многими переменными, нетривиальными условиями и т.п.

В настоящее время в рамках **Prognoz Platform** развиваются все необходимые для работы с большими данными технологии: поддержка **Hadoop**, интеграция с программно-аппаратными комплексами, интеграция с решениями SAP [Prognoz Platform – Википедия (*wikipedia.org*)].

Prognoz Platform – платформа бизнес-аналитики для создания информационных систем и применения в качестве самостоятельного решения. Позволяет разрабатывать приложения (в том числе мобильные) для оперативного анализа данных, а также моделирования бизнес-процессов и прогнозирования.

Исходя из определения **Big Data**, можно сформулировать следующие основные принципы работы с такими данными:

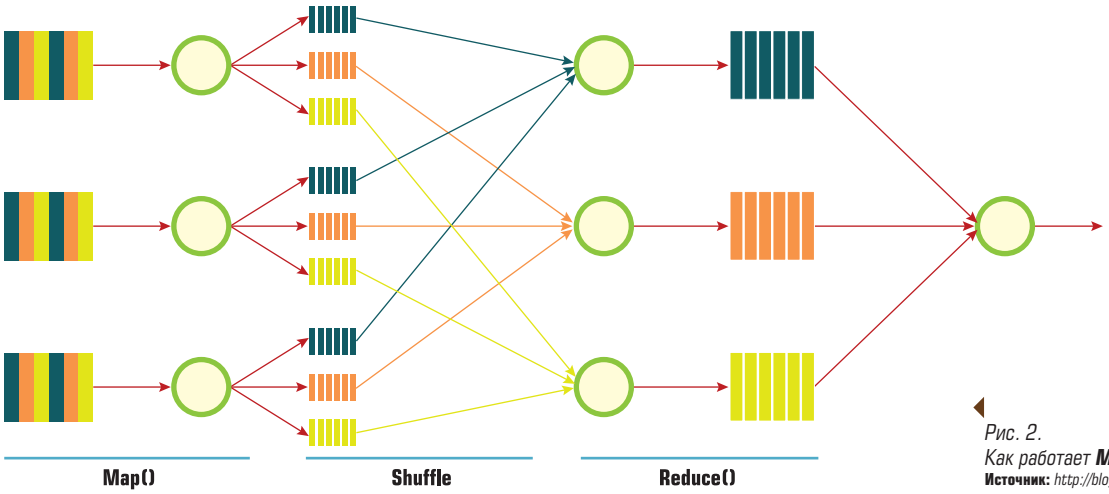

Рис. 2.
Как работает ***MapReduce***
Источник: *http://blog.sqlauthority.com*

1. **Горизонтальная масштабируемость.** Поскольку данных может быть сколь угодно много – любая система, которая подразумевает обработку больших данных, должна быть расширяемой. В 2 раза вырос объём данных – в 2 раза увеличили объем железа (аппаратно-программных средств) в кластере и всё продолжило работать.
2. **Отказоустойчивость.** Принцип горизонтальной масштабируемости подразумевает, что вычислительных машин в кластере может быть много. Например, **Hadoop**-кластер Yahoo имеет более 42 000 вычислителей. Это означает, что часть этих вычислителей будет гарантированно выходить из строя. Методы работы с большими данными должны учитывать возможность таких сбоев и переживать их без каких-либо значимых последствий.
3. **Локальность данных.** В больших распределённых системах данные распределены по большому количеству вычислителей. Если данные физически находятся на одном сервере, а обрабатываются на другом – расходы на передачу данных могут превысить расходы на саму обработку. Поэтому одним из важнейших принципов проектирования BigData-решений является принцип локальности данных – по возможности обрабатываем данные на той же машине, на которой их храним.

Все современные средства работы с большими данными так или иначе следуют этим трём принципам. Для того, чтобы им следовать – необходимо придумывать какие-то методы, способы и парадигмы разработки средств разработки данных.

Рассмотрим лучшие инструменты для анализа больших данных [1, 6]. Вы когда-нибудь задумывались, какой ключевой процесс стоит за такими достижениями в области технологий, как машинное обучение и дополненная реальность? Ответ – наука о данных. Наука о данных – это область знаний, объединяющая статистику, информатику, анализ данных и различные смежные методы для понимания сложных закономерностей, скрытых в структурированных или неструктурированных данных. В ней используются теории и методы, взятые из многих различных областей в контексте информатики, вычислительной техники и знаний о предметной области. Сегодня в Интернете можно найти множество современных инструментов для извлечения знаний из различных типов данных. Однако не все из них стоит пробовать. В статье [6] собраны некоторые из лучших инструментов науки о данных, которые могут быть использованы исследователями и бизнес-аналитиками для получения ценных знаний. Отметим, что в этом списке представлены только инструменты Data Science, а не языки программирования или скрипты для реализации Data Science.

Классический метод MapReduce

Про **MapReduce** в Интернете представлено много информации [7].

MapReduce представляет собой модель распределенной обработки данных, предложенная компанией Google для обработки больших объёмов данных на компьютерных кластерах.

MapReduce был разработан компанией Google в качестве модели программирования для обработки больших наборов данных с помощью параллельного распределенного алгоритма в кластере. Хотя **MapReduce** изначально была запатентованной технологией Google, в последнее время это был довольно обобщенный термин.

MapReduce (рис. 2) содержит процедуры **Map()** и **Reduce()**. Процедура **Map()** выполня-

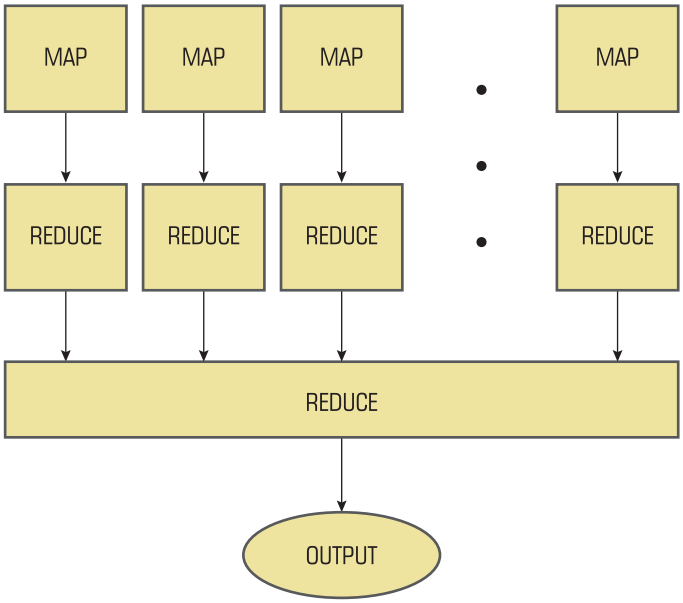

*Рис. 3. Блок-схема архитектура **MapReduce***

ет операцию фильтрации и сортировки производительности для данных, где в качестве процедуры **Reduce()** выполняется операция обобщения данных. Эта модель основана на модифицированных концепциях функций отображения и сокращения, обычно доступных в функциональном программировании. Библиотека, в которой находятся процедуры **Map()** и **Reduce()**, написана на многих разных языках. Самой популярной бесплатной реализацией **MapReduce** является **Apache Hadoop**.

Архитектура MapReduce

Архитектура построена с использованием двух функций: **map** и **reduce**. Каждый вычислительный блок отображает входные данные и выполняет начальное уменьшение. Наконец, некоторое централизованное устройство выполняет окончательное уменьшение и возвращает вывод. Блок-схема архитектуры представлена на рис. 3.

Эта архитектура имеет два важных преимущества:

1. **Архитектура масштабируема**, поскольку, если у пользователя есть больше данных, единственное, что ему нужно сделать, это добавить больше единиц обработки. И что очень важно изменение кода не требуется!
2. **Архитектура универсальна**, поскольку эта архитектура поддерживает широкий спектр задач, пользователь может изменить функционал процедур map и reduce так, как ему необходимо.

Важно отметить, что в большинстве случаев данные пользователя будут очень большими и статичными. Это означает, что разбиение на фрагменты каждый раз неэффективно и фактически излишне. Таким образом, в большинстве приложений в реальной жизни пользователь будет хранить свои данные в виде кусков (или фрагментов) с самого начала. Затем он сможет выполнять различные вычисления, используя технику **MapReduce**.

Типичная структура **MapReduce** содержит петабайты данных и тысячи узлов. Вот основное объяснение процедур **MapReduce**, в которых используется этот массивный набор серверов.

Процедура Map( )

В этой инфраструктуре всегда есть главный узел, который принимает входные данные. Сразу после получения входных данных главный узел делит их на более мелкие под-входы или подзадачи. Эти подзадачи распределяются по рабочим узлам. Рабочий узел позже обрабатывает их и выполняет необходимый анализ. Как только рабочий узел завершает процесс с этой подзадачей, он возвращает ее обратно на главный узел.

Процедура сокращения( )

Все рабочие узлы возвращают ответ на подзадачу, назначенную им главному узлу. Главный узел собирает ответ и еще раз агрегирует его в виде ответа на исходную большую проблему, которая была назначена главному узлу.

Эта структура выполняет описанную выше процедуру **сопоставления()** и **сокращения()** параллельно и независимо друг от друга. Все процедуры **Map()** могут выполняться параллельно друг другу, и как только каждый рабочий узел выполнит свою задачу, они могут отправить ее обратно в мастер-код для компиляции с одним ответом. Эта конкретная процедура может быть очень эффективной, когда она реализуется на очень большом объеме данных (*больших данных*).

Структура **MapReduce** состоит из пяти различных этапов:

- Подготовка ввода карты().
- Выполнение кода Карты, Предоставленного Пользователем().

- Перетасуйте вывод карты, чтобы уменьшить процессор.
- Выполнение Предоставленного Пользователем Кода Сокращения.
- Производство конечной продукции.

Вот поток данных структуры **MapReduce**:

- Устройство для считывания ввода.
- Функция карты.
- Функция разбиения на разделы.
- Функция Сравнения.
- Уменьшите Функцию.
- Устройство Записи выходных данных.
- Одно Утверждение.

Для справки: *карта – это пользовательская функция, которая принимает серию пар ключ-значение и обрабатывает каждую из них, чтобы сгенерировать ноль или более пар ключ-значение.*

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

1. **Стадия Map.** На этой стадии данные предобрабатываются при помощи функции map(), которую определяет пользователь. Работа этой стадии заключается в предобработке и фильтрации данных. Работа очень похожа на операцию map в функциональных языках программирования – пользовательская функция применяется к каждой входной записи.

   Функция map() примененная к одной входной записи и выдаёт множество пар ключ-значение. Множество – т.е. может выдать только одну запись, может не выдать ничего, а может выдать несколько пар ключ-значение. Что будет находиться 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.** Проходит незаметно для пользователя. В 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 задаётся пользователем и вычисляет финальный результат для отдельной "корзины". Множество всех значений, возвращённых функцией reduce(), является финальным результатом **MapReduce**-задачи.

Приведем несколько дополнительных фактов про **MapReduce**:

1. Все запуски функции map работают независимо и могут работать параллельно, в том числе на разных машинах кластера.
2. Все запуски функции reduce работают независимо и могут работать параллельно, в том числе на разных машинах кластера.
3. Shuffle внутри себя представляет параллельную сортировку, поэтому также может работать на разных машинах кластера.

Пункты 1-3 позволяют выполнить принцип горизонтальной масштабируемости.

4. Функция map, как правило, применяется на той же машине, на которой хранятся данные – это позволяет снизить передачу данных по сети (принцип локальности данных).
5. **MapReduce** – это всегда полное сканирование данных, никаких индексов нет. Это означает, что **MapReduce** плохо применим, когда ответ требуется очень быстро.

Рассмотрим примеры задач, эффективно решаемых при помощи **MapReduce**.

Начнём с классической задачи – **Word Count.** Задача формулируется следующим образом: имеется большой объем документов. Задача – для каждого слова, хотя бы один раз встречающегося в этом объеме документов, посчитать суммарное число раз, которое оно встретилось в этом объеме документов.

Решение:

Раз имеем большой объем документов – пусть один документ будет одной входной записью для **MapRreduce**-задачи. В **MapReduce** можно только задавать пользовательские функции, что и сделаем (будем использовать **python-like** псевдокод). Фрагмент программы представлен ниже:

```
def map(doc):
    for word in doc:
        yield word, 1

def reduce(word, values):
    yield word, sum(values)
```

Функция **map** превращает входной документ в набор пар (слово, 1), **shuffle** прозрачно для нас превращает это в пары (слово, 1,1,1,1,1,1), **reduce** суммирует эти единички, возвращая финальный ответ для слова.

Пример – обработка логов рекламной системы.

Второй пример взят из реальной практики **Data-Centric Alliance**.

Задача: имеется **csv-лог** рекламной системы вида:

```
<user_id>,<country>,<city>,<campaign_id>,
<creative_id>,<payment></p>

11111,RU,Moscow,2,4,0.3
22222,RU,Voronezh,2,3,0.2
13413,UA,Kiev,4,11,0.7
...
```

Примечание: *CSV от англ. Comma-Separated Values – значения, разделённые запятыми — текстовый формат, предназначенный для представления табличных данных.*

Необходимо рассчитать среднюю стоимость показа рекламы по городам России.

Фрагмент программы представлен ниже:

```
def map(record):
    user_id, country, city, campaign_id, creative_id,
    payment = record.split(",")
    payment = float(payment)
        if country == "RU":
            yield city, payment

def reduce(city, payments):
    yield city, sum(payments)/len(payments)
```

Функция map проверяет, нужна ли нам данная запись – и, если нужна, оставляет только нужную информацию (город и размер платежа). Функция reduce вычисляет финальный ответ по городу, имея список всех платежей в этом городе.

Таким образом можно сделать следующие выводы.

Выше были рассмотрены несколько вводных моментов про большие данные:

- Что такое **Big Data** и откуда они берутся.
- Каким основным принципам следуют все средства и парадигмы работы с большими данными.
- Рассмотрена парадигма **MapReduce** и разобрано несколько задач, в которых она может быть применена.

Вызвавшая в свое время фурор модель распределенных вычислений **MapReduce** уже в настоящее время успела морально устареть – производительности, обеспечиваемой данной моделью, для аналитики BigData уже недостаточно.

Модель **MapReduce** открыла немыслимую прежде возможность надежной обработки больших массивов данных на кластерах, собранных из доступных серверов. Впрочем, при всей сенсационной новизне не стоит упускать из виду то обстоятельство, что методы ускорения, использующие распараллеливание вычислений в многопроцессорной среде и базирующиеся на разделении процесса счета с последующей сборкой результатов, далеко не новы. Еще полвека назад в 1963 году **Мелвин Конвей** предложил модель **Fork/Join**, которая, в отличие от **MapReduce**, лучше приспособлена не к кластерам, а к многоядерным процессорам. Об этой работе, как и о работе **Дага Ли**, вспоминали редко, тем не менее к ним пришлось вернуться, причем в тот момент, когда у компании Google возникла необходимость обрабатывать большие массивы на кластерах, собранных из собственных серверов. Эта компания умеет скрывать свои секреты, и широкая общественность узнала о существовании **MapReduce** лишь из вышедшей в 2004 году статьи **Джефри Дина** и **Санжая Чемавата** "**MapReduce**: упрощенная обработка данных на больших кластерах" (**MapReduce**: Simplified Data Processing on Large Clusters). Позже появилось описание файловой системы Google File System (GFS), а год спустя **Даг Каттинг** и **Майкл Кафарелла** по образу и подобию этой модели и файловой системы создали свою открытую версию, назвав ее **Hadoop**.

Сегодня система **Apache Hadoop** [7] ушла далеко вперед от первого варианта, теперь она состоит из библиотек и утилит, распределенной файловой системы **Hadoop Distributed File System (HDFS)**, платформы для управления ресурсами **Hadoop YARN**, масштабируемой программной модели **Hadoop MapReduce** и еще множества компонентов. В 2014 году начался переход на версию **Hadoop 2.0**, имеющую целый ряд отличий, среди которых две новые, более современные модели параллельных вычислений **Apache Tez** и **Apache Spark**.

Создание **Apache Tez** и **Apache Spark** было вызвано требованиями поддержки получающих широкое распространение аналитических приложений (машинное обучение, обработка текстов на естественных языках, распознавание речи, методы извлечения информации из данных). Итерационная природа подобных приложений противоречит организации потока работ в пакете – реализация итераций средствами традиционного **Hadoop MapReduce** оказывается чрезвычайно длительной. Сам переход с **MapReduce** на **Apache Tez** и **Apache Spark** напоминает случившийся в семидесятые годы переход от пакетного режима взаимодействия с компьютером к интерактивному.

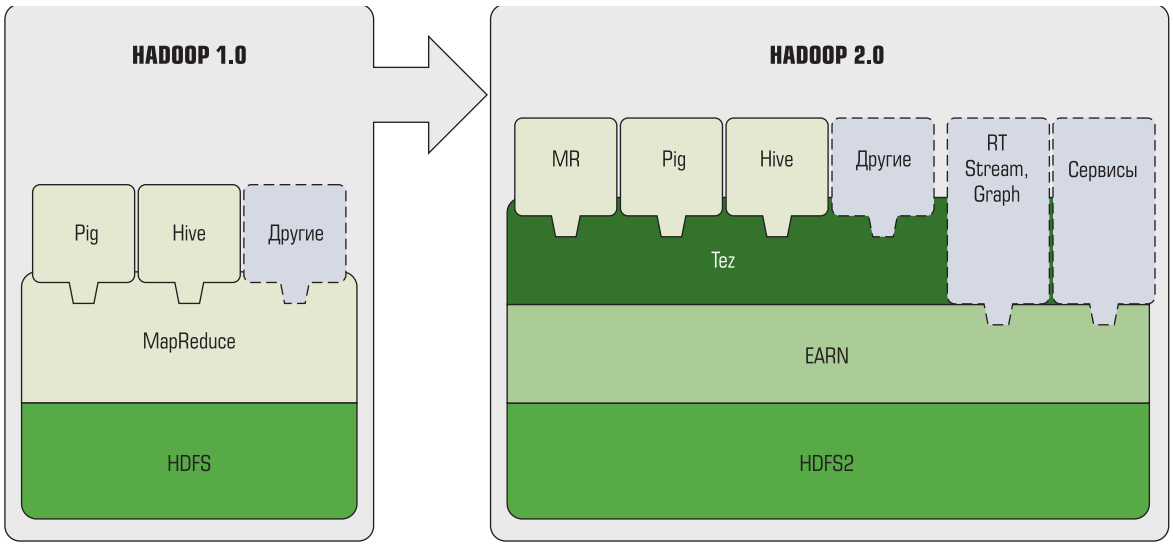

Рис. 4. ***Apache Tez*** *в среде* ***Hadoop 2***

Применительно к **Apache Tez** и **Apache Spark** используют понятие “реальное время”, однако речь не идет о строгом инженерном понимании реального времени – мера латентности в данном случае определяется готовностью человека к ожиданию результатов. И на этот раз, как раньше, история началась с публикации в 2010 году группой сотрудников Google статьи о системе **Dremel** (*Dremel: Interactive Analysis of WebScale Datasets*).

Dremel представляет собой интерактивную систему, специально созданную для обработки запросов, адресованных к сгруппированным (*nested*) данным, доступным только для чтения. Комбинация многоуровневых исполнительных деревьев (*multi-level execution trees*) с поколоночным размещением данных позволяет обрабатывать миллиарды табличных строк в секунду. Система масштабируется на тысячи центральных процессоров, петабайты данных и тысячи пользователей. Сообщество программирующих в открытых кодах ответило своей версией, названной **OpenDremel**, вошедшей в проект **Apache Drill**, превосходящий **Dremel** по функциональности.

Причина ограниченности **MapReduce** при работе с аналитическими приложениями, в машинном обучении и т.п. кроется в использовании *ациклической модели потоков данных* (*Directed Acyclic Graph, DAG*), не предполагающей какой-либо возможности вложения циклов на всем протяжении процесса от **Map** до **Reduce**, что противоречит сути подобных приложений. Сейчас, чтобы приспособить модель **MapReduce** для целей анализа, приходится многократно повторять основной цикл, а это приводит к заметному снижению производительности. **Apache Tez** и **Apache Spark** позволяют в десятки раз повысить скорость обработки: в первой это происходит за счет усовершенствования процедуры выполнения графовой модели, а во второй – благодаря переходу на принципиально иную алгоритмическую основу.

Tez переводится с хинди как “*скорость*”. Система является типичным образцом продукта, созданного в Индии, – в нем реализовано более эффективное, чем **MapReduce**, решение, но это не революция. **Spark** же (от английского “*искра*”) родом из Калифорнийского университета Беркли, в нем использована качественно новая математика, и не исключено, что **Spark** – это именно та искра, из которой может разгореться пламя нового поколения решений по распределенной обработке данных, и далеко не случайно, что после выхода **Spark** из тени к этому решению сразу было привлечено большое внимание.

Apache Tez выполняет **DAG** как набор задач, интерпретируя обработку данных как граф потоков, где вершины – процессы, а ребра – перемещения данных между процессами. В вершинах размещается аналитическая логика, а ребра отражают процесс передачи данных и связи между их источниками и потребителями. В **Tez** единый этап **Reduce** разделяется на отдельные, связанные между собой стадии **Reduce**. Такое деление стало возможным с появлением конструкции **YARN** (рис. 4), поддерживающей связь с файловой системой **HDFS2** и кластером.

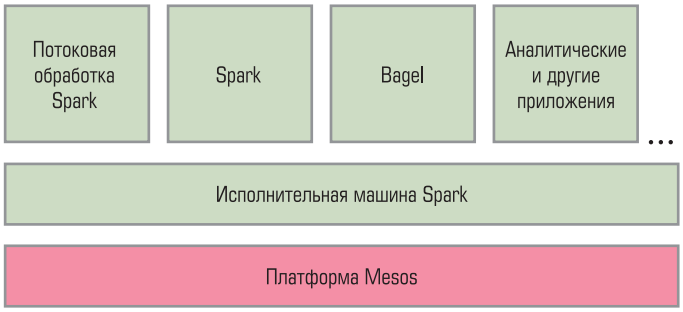

*Рис. 5. Среда **Spark***

Более высокая производительность алгоритмов **Tez** по сравнению с **Hadoop MapReduce** обеспечивается двумя факторами. Во-первых, динамическим распознаванием графа, отражающим тот факт, что обычно распределенные данные оказываются динамичными по своей природе и многое из того, что с ними нужно делать, определяется уже в процессе исполнения. Динамика находится в противоречии с пакетным режимом, где все следует определить заранее, а **Tez** включает в себя модули модификации вершин графа, позволяющие найти лучшее решение. Во-вторых, **Tez** использует оптимальное управление ресурсами. Эта функция реализуется им совместно с YARN, где есть элемент динамики – потребность в ресурсах меняется, и на разных фазах решения задачи они могут быть разными. Система **Tez** не предназначена для конечных пользователей и ориентирована на разработчиков приложений, поддерживая хранилище данных **Apache Hive** и аналитическую платформу с языком высокого уровня **Apache Pig**.

Одна из самых серьезных проблем Больших Данных состоит в том, что к этой еще только формирующейся парадигме многие подходят со старыми мерками без понимания целостности картины. Среди немногочисленных мест, где предпринимаются попытки разобраться с неизбежным в таких случаях клубком противоречий, есть небольшая исследовательская лаборатория **AMPLab** (*Algorithms, Machines, People*), в ней с 2008 года ведется комплекс исследований, приведший к проекту **Apache Spark**, который представляет собой работающую в памяти программную конструкцию, способную составить конкуренцию программной модели **MapReduce**. Автором данной новации стал аспирант Беркли **Матей Захариа**, выходец из Румынии, тема диссертации которого – “Архитектура для быстрой и универсальной обработки данных на больших кластерах” (An Architecture for Fast and General Data Processing on Large Clusters). Одним из двух руководителей Захариа является **Скотт Шенкер** – он известен своими работами в области программно-конфигурируемых сетей (SDN) и входил в число создателей компании Nicira, купленной VMware в 2012 году.

Spark можно рассматривать как дальнейшее развитие идеи **DAG** с возможностью циклической обработки потоков данных, размещенных в памяти. Отход от пакетного режима поддержан новым, не имеющим аналогов способом представления данных, получившим название **Resilient Distributed Dataset (*RDD*)**. Организация данных в виде **RDD** позволяет распределять и перераспределять данные на различных стадиях параллельных вычислений, открывая путь к итерационным процессам, невозможным в случае **DAG**. Вот почему **RDD** оказывается эффективным с точки зрения как скорости обработки, так и обеспечения высокой надежности, требуемой при распределенной работе на кластерах.

Работа **Spark** (рис. 5) поддерживается платформой **Mesos**, управляющей кластерами, а реализация функционала **RDD** воплощена в управляющей машине **Spark RDD**, поверх которых выполняется потоковая обработка, машина для обработки графов **Bagel**, **Shark** (аналог Hive SQL, но работающий на два порядка быстрее), а также аналитические приложения, системы машинного обучения и др. В отличие от **Tez**, решение **Spark** рассчитано не только на профессиональных разработчиков – оно может стать инструментом для специалистов по данным (*data scientist*), которым предоставляется современный язык **Scala**, иногда называемый преемником **Java**. Возможно также использование **Python** с интерпретатором **IPython**.

На данный момент имеется не так много аналитических систем, работающих на **Tez** и **Spark**, но даже уже известные результаты позволяют отнести эти модели, и особенно последнюю, к числу суперзвезд **Open Source**, способных определить будущее индустрии Больших Данных [9].

Программный продукт DataRobot [10]

Этот программный продукт позволяет визуально и интерактивно исследовать, объединять и формировать различные наборы данных в данные, готовые к машинному обучению

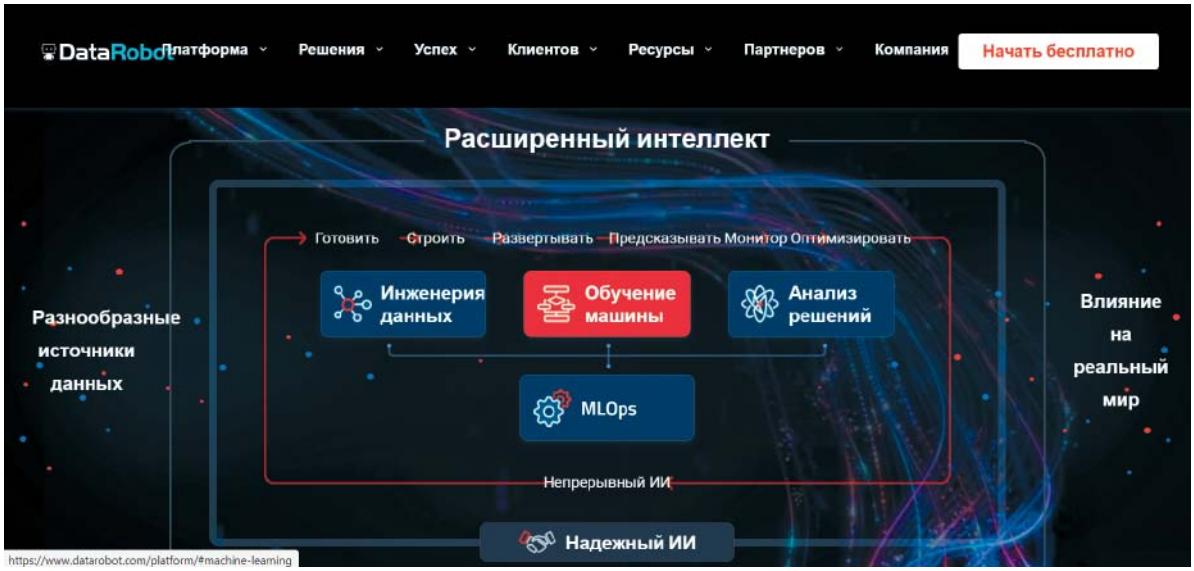

*Рис. 6. Платформа **DataRobot***

и приложениям искусственного интеллекта в масштабе предприятия. **DataRobot** автоматизирует многие рутинные задачи, выполняемые специалистами по обработке и анализу данных, представляя бизнес-аналитикам, стремящимся включить машинное обучение в свои текущие рабочие процессы (рис. 6).

Являясь неотъемлемой частью платформы, **DataRobot Data Prep** позволяет как начинающим, так и опытным пользователям быстро и интерактивно исследовать, профилировать, очищать, обогащать и формировать разнообразные данные в активы ИИ, готовые к разработке и развертыванию модели машинного обучения.

С **DataRobot Data Prep** процесс подготовки данных значительно ускоряется благодаря интуитивно понятному и визуальному интерфейсу “укажи и щелкни”. Кроме того, встроенные алгоритмы предоставляют пользователям помощь ИИ во всех действиях по подготовке данных для автоматического профилирования, кластеризации и очистки данных, ускоряя время, необходимое для подготовки и формирования данных.

DataRobot предоставляет первую и наиболее полную в отрасли комплексную платформу **AI Cloud** для ускорения окупаемого с помощью ИИ от необработанных данных до ROI. **DataRobot Data Prep** тесно интегрирован в платформу **DataRobot AI Cloud**, обеспечивая самостоятельную подготовку данных не только для квалифицированных специалистов по обработке и анализу данных, но и для бизнес-аналитиков и гражданских специалистов по обработке и анализу данных.

Плюсы: идеально подходит для масштабирования возможностей машинного обучения. Содержит массивную библиотеку моделей с открытым исходным кодом и собственной разработки. Решает самые сложные проблемы в области науки о данных. Предоставляет полностью объяснимый (**ИИ**) с помощью удобных для человека визуальных представлений.

Минусы: довольно дорого по сравнению с другими инструментами. Цена продукта зависит от размера и сложности проекта.

Доступна бесплатная пробная версия. **DataRobot** интегрирован с сотнями новейших алгоритмов машинного обучения, что обеспечивает полную прозрачность и контроль над процессом построения и развертывания моделей.

Начнем с того, что с помощью этой системы можно выбрать наиболее подходящую модель для развертывания из множества возможных. Используя искусственный интеллект можно быстро запустить любую модель в производство, независимо от того, нужны ли вам пакетные развертывания, прогнозы в реальном времени или скоринг на **Hadoop**. Возможно, вам потребуется добавить несколько строк кода для настройки процесса. Помимо того, что **DataRobot** уделяет особое внимание таким методам, как трансфертное обучение и машинное обучение, он также включает функции, обеспечивающие ценность для бизнеса, такие как кривые прибыли, прогнозы на основе дан-

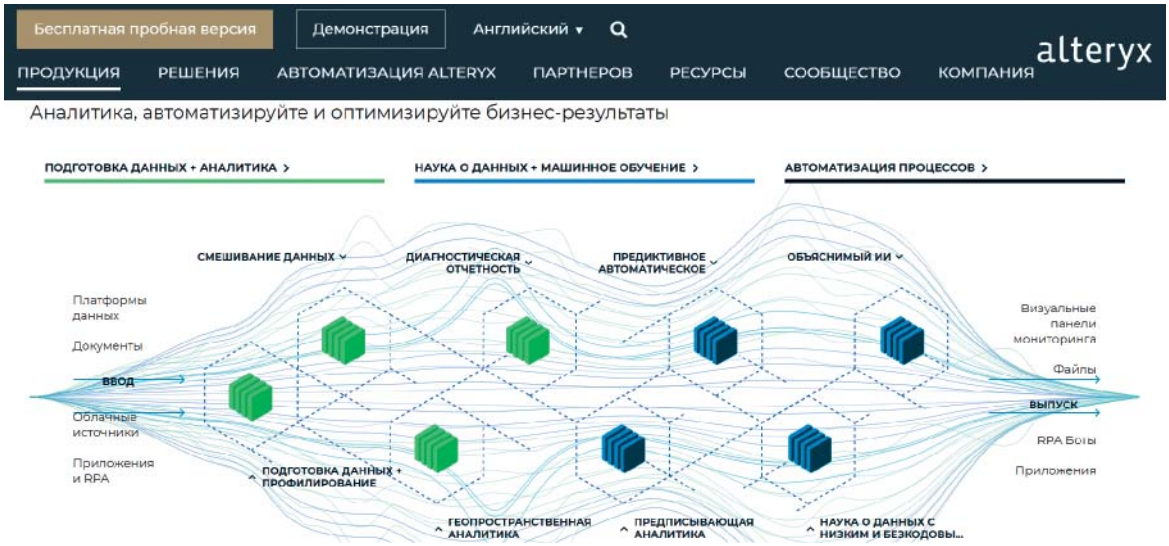

Рис. 7. Аналитика, автоматизируйте и оптимизируйте бизнес-результаты с ***Alteryx***

ных и развертывание в один клик с управлением. Платформа может быть использована для решения широкого спектра задач науки о данных, начиная от прогнозирования продаж миллионов товаров и заканчивая работой со сложными геномными данными.

Программный продукт Alteryx [11]

Alteryx – это компания по разработке компьютерного программного обеспечения со штаб-квартирой в Ирвине, штат Калифорния, но связана с компаниями группы по всему миру. Сквозная платформа компании объединяет аналитический опыт, позволяя организациям преодолевать барьеры данных. Платформа **Alteryx** обеспечивает аналитическую гибкость, необходимую бизнес-аналитикам, ученым по обработке и анализу данных и ИТ-отделу для обнаружения, подготовки, анализа и ввода в эксплуатацию аналитических моделей с помощью совместной и управляемой платформы. Компания предоставляет клиентам доступ к определенным синдицированным данным, которые они объединяют со своими собственными данными, чтобы лучше понять свой бизнес или своих клиентов и принимать лучшие решения.

Простой интерфейс главного продукта **Alteryx**, одноименного средства для обработки данных, позволяет всем, от недавних выпускников до почетных директоров, превращать сырые числа в диаграммы и графики. Возможности **Alteryx** намного шире, чем у Excel. Введите несколько чисел, выберите нужную операцию – например, “очистка данных” или “линейная регрессия” – и готово.

Применение **Alteryx** можно найти в любой отрасли. Coca-Cola пользуется ей, чтобы помочь ресторанам определить, сколько напитка заказывать, авиалинии – чтобы хеджировать цену самолетного топлива, банки – чтобы моделировать деривативы. Анализ данных – это “навык, который необходим всем нам, если мы хотим выжить в новую эпоху, – говорит Стокер, основатель компании. – Более важный, чем умение сводить баланс” (рис. 7).

Плюсы: интуитивно понятный интерфейс. Готовые к использованию шаблоны прогностического моделирования. Визуализация сложных запросов. Подготовка, смешивание и анализ данных с помощью перетаскивания данных. Интегрированный OCR и текстовый анализ.

Минусы: продукт дорогой. Функции вспомогательного моделирования требуют дополнительной лицензии.

Продукт продается по цене от $ 2300 в год на одного пользователя. Доступна 30-дневная бесплатная пробная версия.

Alteryx объединяет аналитику, машинное обучение, науку о данных и автоматизацию процессов в единую сквозную платформу. Она принимает данные с сотен платформ (включая Oracle, Amazon и Salesforce), позволяя специалистам тратить больше времени на анализ и меньше на поиск. Специалисты могут исследовать данные, создавая, получая доступ и выбирая функции с помощью визуального интерфейса программирования – **Analytic**

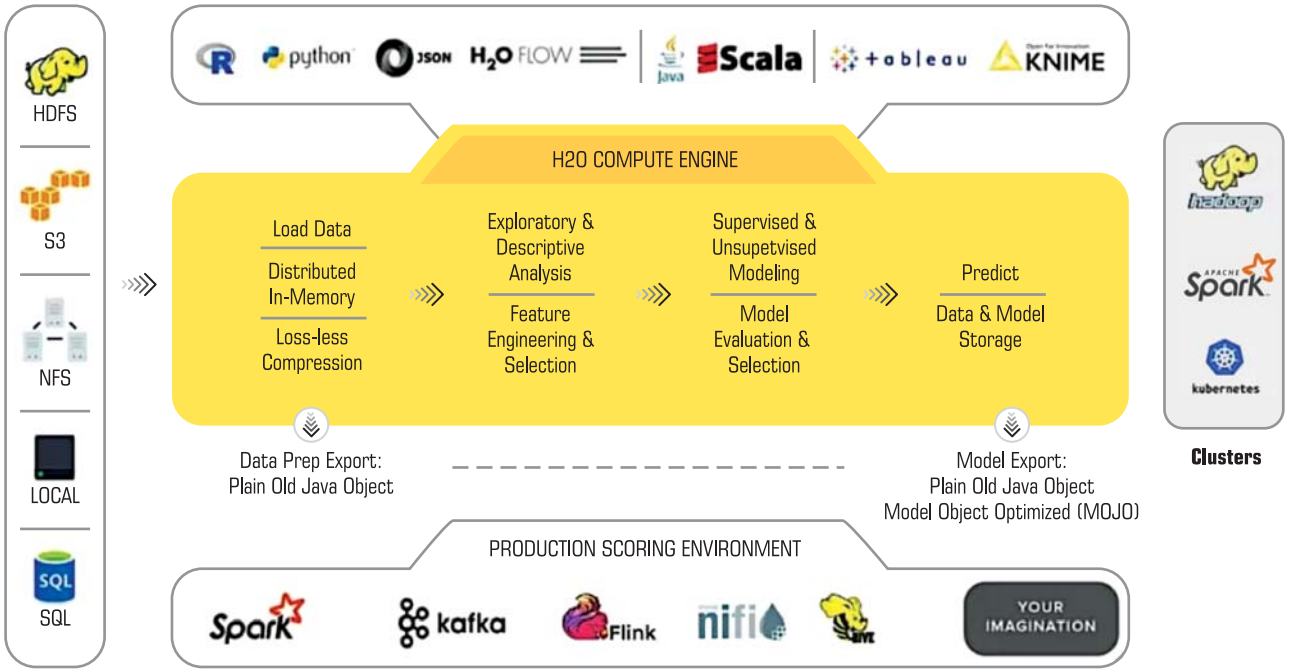

Рис. 8. Программный продукт ***H2O AI Hybrid Cloud***

Process Automation. Она позволяет вносить детальные изменения в отдельные аналитические блоки, используя готовые варианты конфигурации или добавляя собственный код на **Python** или **R** в аналитический рабочий процесс.

Alteryx позволяет быстро создавать прототипы моделей машинного обучения и конвейеров с помощью автоматизированных блоков обучения моделей. Она помогает легко визуализировать данные на протяжении всего пути решения задач и моделирования. Как? Автоматически создает таблицы, графики и отчеты на любом этапе анализируемого процесса. Платформа предназначена для компаний любого размера. Если у вас средний бизнес, она поможет вам найти новые идеи и добиться высокоэффективных результатов.

Программный продукт H2O AI Hybrid Cloud [12]

H2O AI Hybrid Cloud предлагает комплексную платформу, которая демократизирует искусственный интеллект, предоставляя каждому сотруднику, клиенту и гражданину сложную технологию ИИ и простые в использовании приложения ИИ (рис. 8).

H2O AI Hybrid Cloud – это сквозная платформа, которая позволяет организациям быстро создавать модели и приложения ИИ мирового класса. Нет никаких ограничений для пользователей, что позволяет всем в организации внедрять инновации и использовать ИИ для принятия лучших решений, оптимизации операций, снижения рисков и персонализации клиентского опыта.

Производители сталкиваются с растущим давлением, чтобы снизить затраты, продолжая поставлять высококачественную продукцию и услуги. Ведущие производители используют проактивный подход к оптимизации операций. Суть этого подхода – в постоянной оптимизации процессов на основе актуальных данных, что повышает вероятность успеха. В проактивном управлении используют известное “правило Парето”: 20 % скрытых угроз способны вызвать 80 % критических ситуаций, наносящих вред компании.

Компании, которые широко используют ИИ, пожинают плоды повышения эффективности, сокращения времени простоя при одновременном повышении удовлетворенности клиентов. Эти компании используют ИИ для ряда сценариев, включая прогностическое обслуживание, проектирование прогностических процессов, оптимизацию цепочки поставок и многое другое.

H2O.ai – лидер в области искусственного интеллекта и автоматизации с открытым исходным кодом и автоматизацией, расширяет возможности ведущих производственных

Рис. 9. ***H2O.ai*** *лидер в области искусственного интеллекта и автоматизации с открытым исходным кодом*

компаний по предоставлению решений ИИ, которые меняют отрасль (рис. 9).

Плюсы: Распределенное машинное обучение в памяти. Простота развертывания больших моделей. Автоматизация рабочего процесса машинного обучения. Работает на существующей инфраструктуре больших данных.

Минусы: Ограниченные возможности обработки данных. Отсутствие документации.

Цена программного продукта зависит от размера и сложности проекта. Доступна 14-дневная бесплатная пробная версия.

H2O.ai представляет собой инструмент машинного обучения с открытым исходным кодом и распределенной памятью, обладающий линейной масштабируемостью. Этот инструмент поддерживает почти все популярные статистические алгоритмы и алгоритмы машинного обучения, включая обобщенные линейные модели, глубокое обучение и машины с градиентным усилением. Инструмент принимает данные непосредственно из **Spark**, **Azure**, **Spark**, **HDFS** и различных других источников в свое распределенное хранилище ключевых значений **in-memory**.

Для построения моделей можно использовать либо язык программирования **R/Python**, либо **H2O Flow** (*графический блокнот*), который не требует кодирования. **H2O AutoML** упрощает обучение и оценку моделей машинного обучения. Это помогает автоматизировать задачи науки о данных (такие, как выбор алгоритма, итеративное моделирование, настройка гиперпараметров, генерация признаков и оценка моделей) и больше сосредоточиться на важных проблемах. Платформа чрезвычайно популярна в сообществах **Python** и **R** и используется более чем 18 000 организаций.

Программный продукт D3.js [13]

D3.js (*сокращение от Data-Driven Documents*) – это библиотека **JavaScript** для манипулирования документами на основе данных (рис. 10). **D3** помогает оживить данные с помощью средств **HTML**, **SVG** и **CSS**. Акцент **D3** на веб-стандартах дает вам все возможности современных браузеров, не привязывая себя к проприетарной структуре, сочетая мощные компоненты визуализации и управляемый данными подход к манипулированию **DOM**.

Продукт **D3** позволяет привязать произвольные данные к модели **DOM**, а затем применить к документу управляемые данными преобразования. Например, **D3** можно использовать для создания HTML-таблицы из массива чисел. Или используйте те же данные для создания интерактивной линейчатой диаграммы SVG с плавными переходами и взаимодействием.

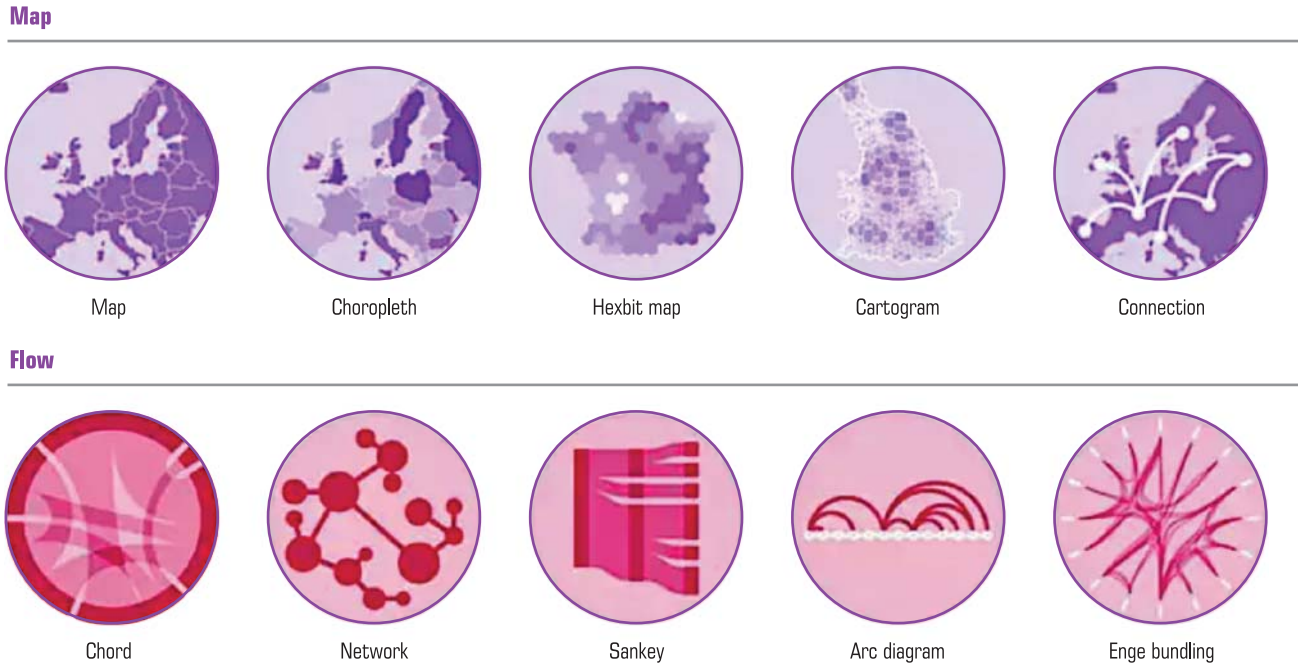

*Рис. 10. Библиотека **JavaScript** для манипулирования документами на основе данных*

D3 не является монолитной структурой, которая стремится обеспечить все мыслимые функции. Вместо этого **D3** решает суть проблемы: эффективное манипулирование документами на основе данных (рис. 11). Это позволяет избежать проприетарного представления и обеспечивает исключительную гибкость, предоставляя все возможности веб-стандартов, таких как HTML, SVG и CSS. С минимальными накладными расходами **D3** чрезвычайно быстр, поддерживая большие наборы данных и динамическое поведение для взаимодействия и анимации. Функциональный стиль **D3** позволяет повторно использовать код через разнообразную коллекцию официальных и разработанных сообществом модулей.

Плюсы: Легкий и быстрый. Дает полный контроль над визуализацией данных. Работает с такими веб-стандартами, как SVG и HTML. Множество встроенных многократно используемых функций и фабрик функций.

Минусы: Документация может быть немного улучшена.

Продукт распространяется бесплатно.

Библиотека **JavaScript** для создания динамических, интерактивных визуализаций данных в веб-браузерах использует выбороч-

*Рис. 11. **D3** Документы, управляемые данными*

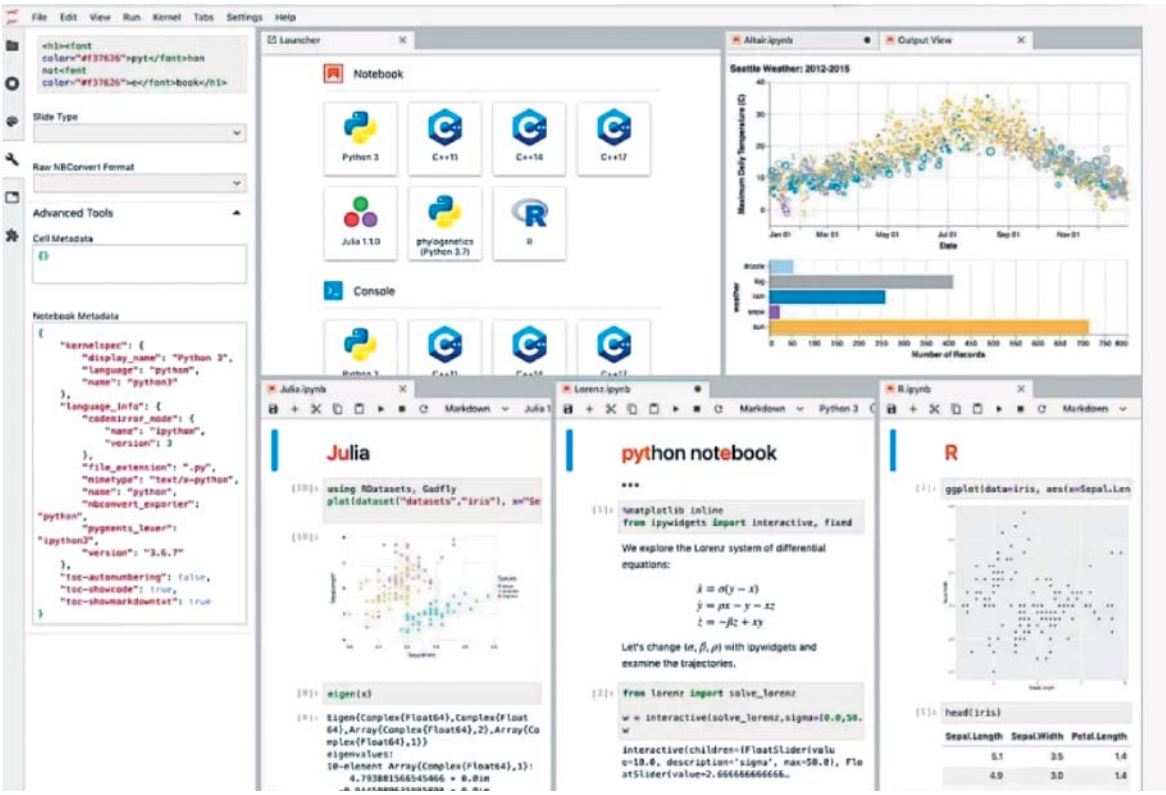

Рис. 12. ***Project Jupyter*** – это некоммерческий проект с открытым исходным кодом

ные предварительно разработанные функции для создания объектов SVG, их настройки или добавления к ним динамических эффектов. К этим SVG-объектам можно присоединять большие наборы данных для создания текстовых/графических диаграмм и графиков. **D3** не имеет стандартного формата визуализации. Она позволяет создавать что угодно – от круговых диаграмм и графиков до HTML-таблиц и геопространственных карт. Данные могут быть в различных форматах, таких как CSV или JSON. Вы можете даже написать код **JavaScript** для чтения других форматов данных или повторно использовать код с помощью широкой коллекции официальных и разработанных сообществом модулей.

Программный продукт Project Jupyter [14]

Project Jupyter – это некоммерческий проект с открытым исходным кодом, родившийся из проекта **IPython** в 2014 году, когда он развивался для поддержки интерактивной науки о данных и научных вычислений на всех языках программирования. **Jupyter** всегда будет на 100 % программным обеспечением с открытым исходным кодом, бесплатным для всех и выпущенным на либеральных условиях модифицированной лицензии BSD (рис. 12).

Для справки: ***BSD (англ. Berkeley Software Distribution)*** *– система распространения программного обеспечения в исходных кодах, созданная для обмена опытом между учебными заведениями. Особенностью пакетов ПО BSD была специальная лицензия BSD, которую кратко можно охарактеризовать так: весь исходный код – собственность BSD, все правки – собственность их авторов.*

Jupyter разрабатывается открыто на **GitHub**, благодаря консенсусу сообщества **Jupyter**.

Для справки: ***GitHub*** *– крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc (ранее Logical Awesome). Сервис бесплатен для проектов с открытым исходным кодом и (с 2019 года) небольших частных проектов, предоставляя им все возможности (включая SSL), а для крупных корпоративных проектов предлагаются различные платные тарифные планы.*

Все онлайн- и личные взаимодействия, и коммуникации, непосредственно связанные с проектом, охватываются Кодексом поведения **Jupyter**. Настоящий Кодекс поведения устанавливает ожидания, позволяющие разнообразному сообществу пользователей и участников участвовать в проекте с уважением и безопасностью.

*Рис. 13. Программный продукт **Jupyter***

Проект **Jupyter** существует для разработки программного обеспечения с открытым исходным кодом, открытых стандартов и сервисов для интерактивных вычислений на десятках языков программирования.

JupyterLab – это интерактивная веб-среда разработки для записных книжек, кода и данных **Jupyter**. **JupyterLab** является гибким: настройте и упорядочите пользовательский интерфейс для поддержки широкого спектра рабочих процессов в области обработки и анализа данных, научных вычислений и машинного обучения. **JupyterLab** является расширяемым и модульным: пишите плагины, которые добавляют новые компоненты и интегрируются с существующими.

Jupyter Notebook – это веб-приложение с открытым исходным кодом, которое позволяет создавать и совместно использовать документы, содержащие живой код, уравнения, визуализации и повествовательный текст. Области применения: очистка и преобразование данных, численное моделирование, статистическое моделирование, визуализация данных, машинное обучение и многое другое.

Jupyter поддерживает более 40-ка языков программирования, включая **Python**, **R**, **Julia** и **Scala**. Записными книжками можно поделиться с другими пользователями с помощью электронной почты, **Dropbox**, **GitHub** и **Jupyter Notebook Viewer**.

Код может создавать многофункциональные интерактивные выходные данные: HTML, изображения, видео, LaTeX и пользовательские типы MIME.

Можно использовать инструменты для работы с большими данными, такие как **Apache Spark**, от **Python**, **R** и **Scala** и исследовать те же данные с помощью панд, **scikit-learn**, **ggplot2**, **TensorFlow** (рис. 13).

Плюсы: Легкий и простой в использовании. Отличная поддержка математических библиотек **Python**. Предопределенные модели визуализации. Легко редактировать и отслеживать потоки данных. Автоматически создает контрольные точки.

Минусы: Сложность работы с несколькими ядрами. Ограниченные возможности сотрудничества.

Продукт распространяется бесплатно.

Проект **Jupyter** – это коллекция интерактивных веб-инструментов с открытым исходным кодом, которые ученые, изучающие данные, могут использовать для объединения программного кода, результатов вычислений, мультимедийных ресурсов и пояснительного текста в одном документе. Хотя **Jupyter** существует уже несколько десятилетий, его популярность резко возросла за последние пару лет. **Jupyter** предлагает различные продукты для разработки программного обеспечения с открытым исходным кодом, открытых стандартов и услуг для интерактивных вычислений.

Jupyter Notebook позволяет создавать и обмениваться документами, содержащими живые уравнения, код, визуализации и повествовательный текст.

Jupyter Kernels обрабатывает множество запросов, таких как выполнение и проверка кода, и предоставляет ответ.

JupyterLab предоставляет строительные блоки (терминал, файловый браузер, текстовый редактор, расширенные выходные данные и т.д.) в интуитивно понятном пользовательском интерфейсе.

JupyterHub поддерживает множество пользователей, порождая, управляя и проксируя (перенаправляя трафик) несколько отдельных серверов **Jupyter Notebook**. Можно использовать эти инструменты (бесплатно) для проведения численного моделирования, очистки данных, статистического моделирования, визуализации данных и многого другого прямо из браузера.

Список литературы

1. *Электронный ресурс: https://habr.com/ru/post/267361/*
2. *Электронный ресурс: https://ru.wikipedia.org/wiki/Большие_данные*
3. *Егоров А.А.* Обзор национального стандарта РФ ГОСТ Р ИСО/МЭК ТО 20547-1-2021 “Эталонная архитектура больших данных, Часть 1. Структура и процесс применения”, журнал “Автоматизация и IT в нефтегазовой области”. 2021 г., № 3.
4. *Электронный ресурс: https://medium.com/nuances-of-programming/apache-spark-%D0%B3%D0%B0%D0%B9%D0%B4-%D0%B4%D0%BB%D1%8F-%D0%BD%D0%BE%D0%B2%D0%B8%D1%87%D0%BA%D0%BE%D0%B2-959145ef6167*
5. *Электронный ресурс: https://www.fsight.ru/blog/formula-big-data-sem-v-neordinarnaja-zadacha-2/*
6. *Электронный ресурс: https://new-science.ru/13-luchshih-instrumentov-dlya-analiza-dannyh/*
7. *Электронный ресурс: https://neurohive.io/ru/tutorial/mapreduce/*
8. *Электронный ресурс: https://hadoop.apache.org/*
9. *Электронный ресурс: https://www.computerworld.ru/author/4760976*
10. *Электронный ресурс: https://www.datarobot.com/*
11. *Электронный ресурс: https://www.alteryx.com/*
12. *Электронный ресурс: https://www.h2o.ai/hybrid-cloud/*
13. *Электронный ресурс: https://d3js.org/*
14. *Электронный ресурс: https://jupyter.org/*

Продолжение следует.

***Егоров Александр Александрович** – канд. техн. наук, профессор АВН РФ, главный редактор журнала “Автоматизация и IT в энергетике”.*