Технологии обработки больших данных: Вводная лекция

А.А. Сухобоков

Оглавление

1. Технологии обработки больших данных

Технологии обработки больших данных – это группа технологий и методов эффективной обработки динамически растущих объемов данных (структурированных и неструктурированных) в распределенных информационных системах.

Появившаяся в последние годы необходимость формирования и изучения такой дисциплины обусловлена целой группой факторов, характерных для существующих информационных систем:

- экспоненциальный рост объёмов хранимой и обрабатываемой информации;
- ухудшение качества поступающей и обрабатываемой информации, увеличение её несоответствия реальной обстановке и имеющимся ситуациям;
- произвольным образом осуществляемая систематизация информационных массивов;
- фрагментация процессов обработки информации;
- динамичное включение в процессы обработки информации ранее не учитываемых новых сущностей и факторов;
- практическое отсутствие гетерогенности в информационных системах.

Распространение методов обработки больших данных привело к появлению нескольких новых профессий на рынке труда IT-специалистов:

- Data Scientist – специалист по разработке моделей Machine Learning;
- Data Engineer – специалист по организации обработки больших данных;
- Machine Learning Engineer – специалист по встраиванию моделей Machine Learning в конкретные приложения, скорингу моделей, актуализации моделей посредством их переобучения на новых данных.

При этом конкретную задачу часто решает не один человек, а проектная команда. Основные требования, к этой команде представлены на рисунке 1.

Наиболее известной среди перечисленных профессий является Data Scientist. Необходимая математическая подготовка частично даётся в рамках обучения на специальности ИУ5, какие-то разделы могут быть получены в ходе дальнейшего обучения в аспирантуре, но всё равно многого не хватает, надо самостоятельно учиться и искать другие возможности для образования.

Знания о предметной области вы должны получать, работая в реальной экономике и обучаясь на других специальностях.

Рис. 1. Требования к специалистам в области обработки больших данных

Начальные знания по технологиям обработки больших данных будут изучаться в курсе «Технологии обработки больших данных». Чтобы реально работать по специальности Data Engineer необходима дополнительная подготовка.

Начальные знания по специальности Machine Learning Engineer будут изучаться в курсе «Инструменты бизнес-аналитики». Чтобы реально работать по этой специальности необходима дополнительная подготовка.

2. Понятие Big Data

В соответствии с современными представлениями Big Data представляет собой данные большого объёма, для которых характерны пять V – пять характеристик, начинающиеся на V: Volume - объём, Variety - разнообразие, Velocity – скорость, Veracity – достоверность и Value – ценность [1].

Volume – объём. Объём данных считается большим, когда возникают затруднения при обработке этого объёма средствами традиционных СУБД. Самые большие структурированные базы данных имеют объём несколько сотен ТВ (1 ТВ = 10^{12} байт). При возникновении концепции Big Data 1 РВ (1*10^{15}байт) считался таким объёмом, с которым обычные реляционные СУБД уже не справляются. С развитием процессорных технологий и технологий СУБД эта цифра растёт, однако рост не происходит быстро из-за отсутствия качественных изменений, обусловленных технологическими инновациями.

Внутренней причиной перехода к новым технологиям обработки данных является необходимость распараллелить обработку, распределить её на большое число независимых процессоров, каждый из которых обрабатывает свой фрагмент данных.

Современные дисковые массивы, если их ставить рядами в дата-центре, конечно, могут вместить десятки и сотни петабайтов данных, размещаемых на платформах Big Data и при этом они займут в несколько раз меньше площади, чем серверные стойки. Однако при

использовании данных тысячами или десятками тысяч параллельно работающих серверов узким местом становится пропускная способность каналов доступа к этим данным. Так называемое бутылочное горлышко интерфейса не в состоянии обеспечить пропускную способность, необходимую для обслуживания большого числа параллельно работающих процессоров.

Распределение данных по обрабатывающим их серверам позволяет снять это ограничение. Распараллеливание – наиболее естественный способ преодоления сложностей вызванных большим объёмом данных. Основное преимущество платформ Big Data по сравнению с традиционными клиент-серверными СУБД – это способность распараллелить процессы обработки данных таким образом, когда каждый узел обрабатывает распложенные в его памяти фрагменты общего массива данных. Именно эта идея и дала толчок появлению инструментов Big Data.

Variety – разнообразие. Данные такого объёма очень редко бывают однородными. В подавляющем большинстве случаев общий массив данных включает как структурированные, так и неструктурированные данные. Под неструктурированными данными имеются в виду изображения, фотоснимки, аудио-треки, фильмы и видео-ролики, данные социальных сетей. Пропорции структурированных и неструктурированных данных в разных массивах могут быть самыми разными, например от 1:9 до 9:1.

Velocity – скорость. Скорость трактуется не только как скорость прироста, но и как скорость обновления ранее полученных значений, что неизбежно влечёт за собой необходимость высокоскоростной обработки и получения результатов. В пределе – в реальном времени.

Veracity – достоверность. В условиях работы с большими объемами данных особое значение приобретает отделение достоверных данных от информационного «шума» и мусора, отсеивание этого шума и мусора.

Value – ценность. Именно ценность информации предопределяет целесообразность её обработки. Собираемые данные должны давать ответы на предварительно сформулированные и вновь появляющиеся вопросы. Эффекты, получаемые в результате сбора и обработки данных, должны оправдывать затраты на эти операции. Собираемые данные должны приносить пользу.

Перечисленный перечень ключевых характеристик Big Data появился не сразу. Сначала были сформулированы первые три – Volume, Variety и Velocity. Потом по одной последовательно добавились Veracity и Value.

В последние годы в публикуемых материалах часто высказываются идеи, что для больших данных и процессов их обработки характерны не 5, а большее число V, как, например, на рисунке 2. На этом рисунке к ранее приведенному списку добавились **Versatility** для обозначения динамической природы и непостоянства больших данных и **Visibility**, требующая обозреваемости и ясности общей картины данных.

.

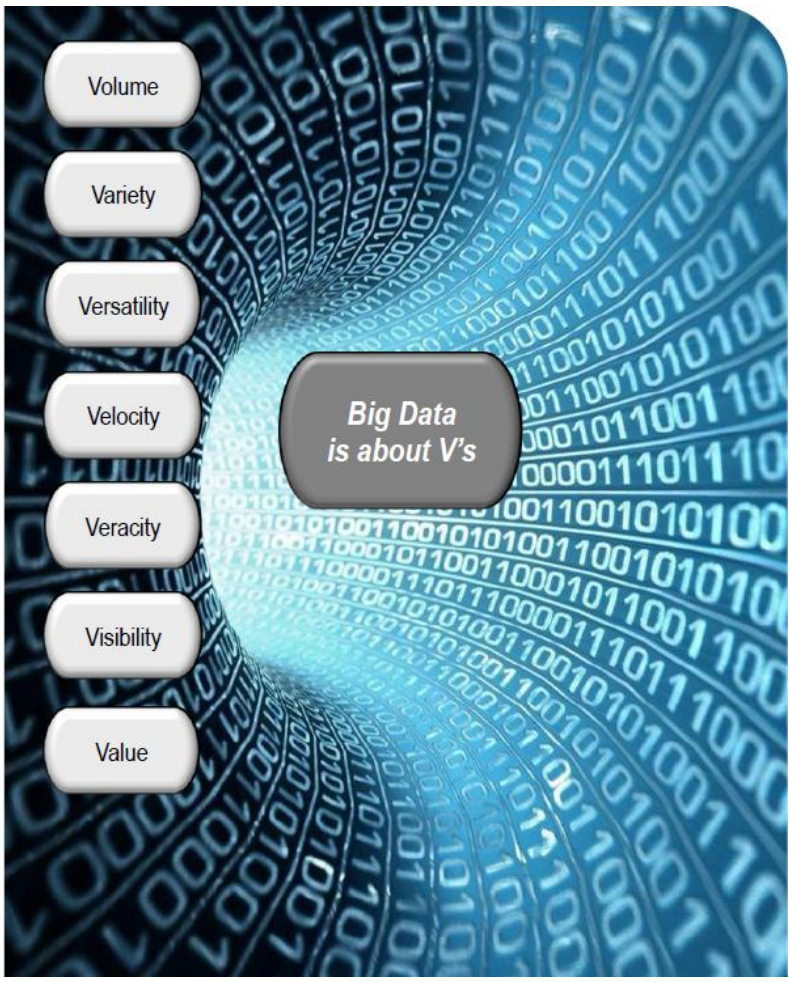


Рис. 2. Иллюстрация семи V, характерных для больших данных

Это не единственное представление 7V, в [2] и [3] приведено ещё одно, показанное на рисунке 3.

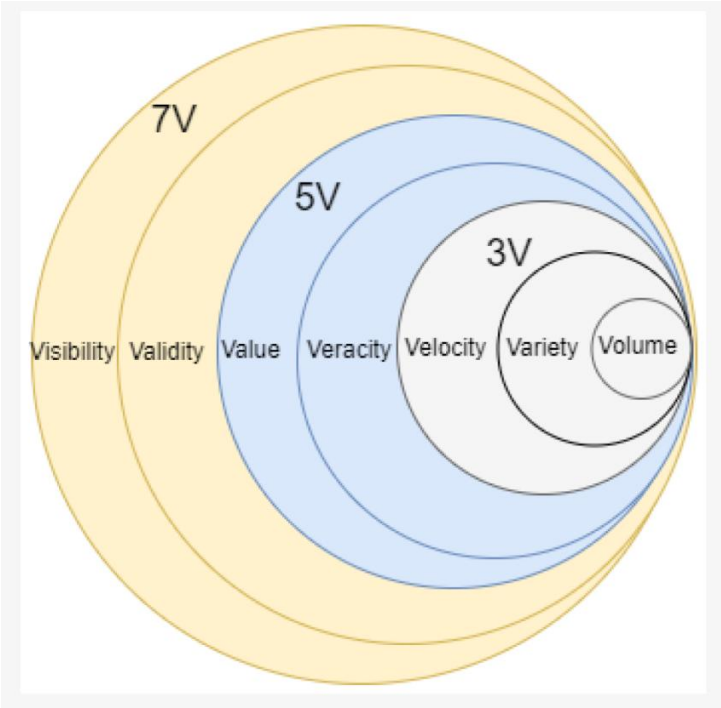


Рис. 3. Ещё одна иллюстрация семи V, характерных для больших данных

В этом случае вместо **Versatility** мы видим **Validity**. На первый взгляд этот аспект можно было бы отнести к **Veracity**. Однако в действительности кроме достоверности важно учитывать обоснованность данных, имеют ли они смысл, который им приписывается.

В [4] представлена модель для оценки проектов в области больших данных на основе восьми характеристик V. Эта модель показана на рисунке 4.

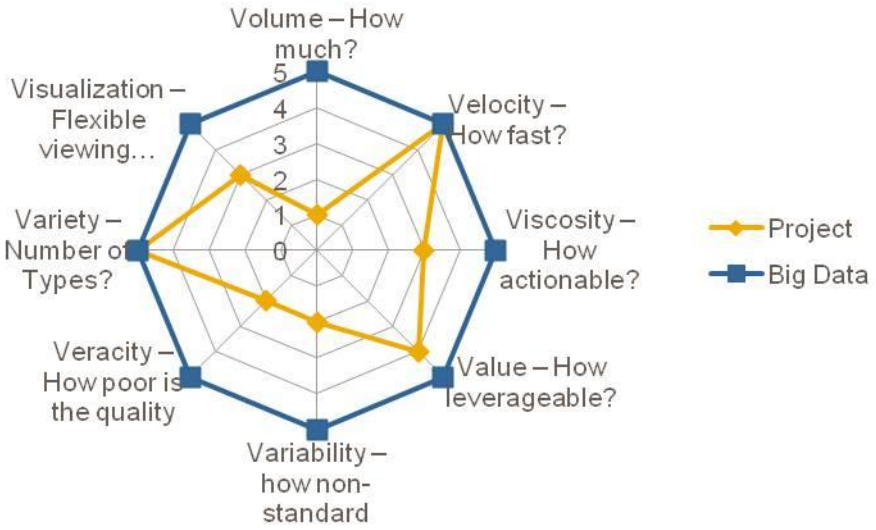


Рис. 4. Оценка проекта в области больших данных с помощью восьми характеристик V

В описанном подходе характеристика **Versatility** заменена сходной по смыслу характеристикой **Variability** и добавлена характеристика **Viscosity** (вязкость; липкость; тягучесть; клейкость; вязкотекучесть) для оценки того, насколько быстро данные могут быть обработаны. **Validity** отсутствует, то есть всего их уже можно насчитать 9.

В [5] представлено 10 характеристик V, в том числе новая характеристика – **Virality**, используемая для того, чтобы показать, что они имеют тенденцию быстро распространяться.

1. **Volume**: огромный объем разработанных данных, с которыми традиционные методы обработки и хранения больше не справляются;
2. **Velocity**: высокая скорость, с которой данные обрабатываются после сбора;
3. **Variety**: характер данных, которые могут принимать различные формы, такие как частично структурированные, неструктурированные или структурированные данные;
4. **Variability**: постоянно меняющиеся значения данных, что может повлиять на их однородность и согласованность;
5. **Veracity**: характеристики качества данных, которые перед использованием должны быть предварительно обработаны, поскольку они часто неоднозначны и ошибочны;
6. **Viscosity**: характеризует интервалы времени между событием и его появлением в аналитике, большие интервалы характерны при работе с большими наборами данных;
7. **Virality**: распределённость данных в сети и высокая скорость обмена информацией;
8. **Validity**: качество и пригодность данных для предполагаемого использования;
9. **Value**: важность знаний и идей, получаемых из сложных, завуалированных и разнообразных данных;
10. **Visibility**: полное визуальное представление данных, позволяющее выявлять скрытые закономерности и корреляции.

Существуют публикации, в которых представлено большее число V – 11, 14, 56 [6]. Однако там же рассматриваются отдельные частные свойства, в большинстве своём относящиеся к используемым методам обработки данных, машинного обучения и искусственного интеллекта. Не будем углубляться. Всё хорошо в меру.

3. Задачи Big Data

Приведём типичные области, в которых возникают данные, которые можно охарактеризовать как Big Data, и необходимо применять соответствующие инструменты. В значительной мере это те области, которые рассматривались в первой лекции как вызовы, для преодоления которых необходимо развивать следующие поколения корпоративных систем управления:

- **Финансовые услуги.** По оценке консалтинговой компании Capgemini, число безналичных транзакций в мире в 2021 году составило около 1 триллиона, а в 2026 году, как ожидается, достигнет 2,1 трлн. [7]. С учетом объёмов данных каждой транзакции и сроков их хранения для архивирования таких финансовых данных предпочтительно использовать инструменты Big Data. Помимо организации хранения больших объёмов данных они позволяют проводить сегментацию пользователей, подключать данные пользователей из социальных сетей и на основе выявленных корреляций формировать для пользователей индивидуальные предложения, лучше учитывающие их потребности. Кроме того, применение инструментов Big Data позволяет выявлять мошенников и предотвращать потери.
- **Производственные процессы в промышленности.** Пример показан на рисунке 5.

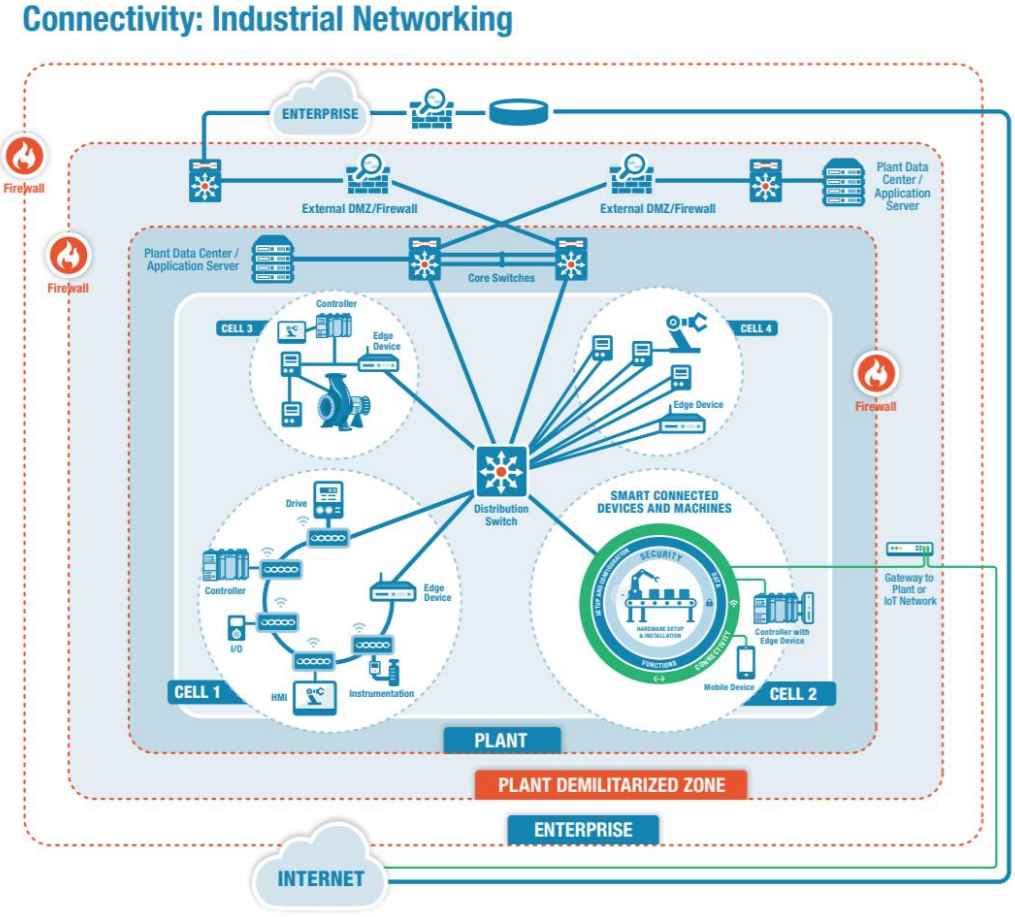


Рис. 5. Пример внутренней сети IoT компании

Данные поступают с сенсоров и используются для управления технологическими процессами, контроля объёма и качества выпускаемой продукции. Компании не раскрывают используемые современные архитектуры и показатели внутрикорпоративных сетей IoT. Однако уже в 2010-х годах трафик внутренних индустриальных сетей превзошёл по объёму трафик открытого интернет. Сейчас трафик внутреннего индустриального интернет отдельных больших компаний может превосходить трафик всемирного открытого интернет. Использование этих решений позволяет дополнительно ежегодно получать огромную прибыль.

- **Здравоохранение.** The Institute for Health Technology Transformation показал, что человеческое тело представляет собой неиссякаемый источник больших данных. Объём архивов изображений в медицине ежегодно возрастает на 20–40%:
  - объём данных одного снимка трёхмерной рентгеновской компьютерной томографии (3D CT Scan) составляет примерно 1GB;
  - объём данных одного снимка трёхмерной магнитно-резонансной компьютерной томографии (3D MRI) составляет примерно 150MB;
  - объём данных одного рентгеновского снимка составляет примерно 30MB;
  - объём данных одной маммограммы составляет примерно 120MB.

  Также наблюдается отчётливая тенденция быстрого роста числа носимых (wearable) устройств, которые находятся на теле пациентов и снимают информацию в реальном времени. Типичный примеры: устройства для ЭКГ по Холтеру, фитнес-трэкеры для записи показателей во время тренировок или Apple Watch. Число таких устройств в мире уже превысило 500 миллионов [8].
- **Эксплуатация и обслуживание сложного оборудования.** Например, внутренние системы одного современного самолёта ежедневно порождают 1 TB данных. Основное назначение собираемых данных – контроль состояния оборудования, планирование технического обслуживания и ремонтов для поддержания необходимого уровня его надёжности.
- **Сельское хозяйство.** Использование методов и инструментов Big Data для анализа цепочек ДНК отдельных растений и животных позволит радикально сократить время выведения новых сортов и пород с заданными свойствами, ранее занимавшее 10 лет и более. Выращивание сельхоз продукции на полях, контролируемых дронами с компьютерным зрением и оснащённых большим числом датчиков, в реальном времени передающих информацию об уровне влажности, освещённости, температуре, наличии питательных веществ в почве и пр. позволит управлять процессами выращивания урожая. Ожидается, что сочетание высокоэффективных сортов, оптимальной ирригации, правильной дозировки пестицидов, гербицидов и удобрений позволит к 2050 году повысить урожайность зерновых до 250%. Объёмы данных, которые придётся при этом собирать и обрабатывать, характерны для Big Data.
- **Сложные логистические процессы.** Данные о размещении каждой упаковки товара на складах, данные об отгрузках и поступлении товаров имеют объёмы, измеряемые терабайтами, и в большинстве случаев могут быть обработаны SCM-системами, которые релевантны масштабам цепочки поставок. Необходимость использования инструментов Big Data в логистических сетях крупных компаний, военных и правительственных

организаций возникла после перехода к современным технологиям, реализующим сбор и обработку данных с меток RFID, установленных на каждой транспортной упаковке, а также сбор, хранение и обработка данных геолокации о каждом транспортном средстве.

- **Информационные технологии.** В 2022году через Facebook было отправлено 5 триллионов сообщений и 200 миллионов видео отправляется через Facebook каждый день. Однако это не самый популярный мессенджер, у WhatsUp показатели намного выше. У мессенджеров, специализирующихся на передаче фото и видео, число отправленных сообщений и просмотров меньше, но суммарный трафик больше. Понятно, что такие потоки сообщений с высокой скоростью порождают новые записи в журналах соответствующих баз данных. В 2014 году Facebook опубликовал информацию, что компания использует для хранения данных платформу на основе HDFS и Hive объёмом 300 PB. В 2024 году Facebook сообщил, что каждый день у них добавляется 4 PB данных. 300 PB в таком режиме хватит на 75 дней, ну или на полгода, если они имели в виду несжатые данные. Очень сложно представить существующий объём хранимой базы, который явно должен измеряться многими ZB.
- **Розничная торговля.** Обработка всей совокупности данных об истории продаж, объёмах запасов, ценах, а также других дополнительных данных, например, о постоянных клиентах, имеющих дисконтные карты, о конкурентах и т.д. позволяет понять факторы, влияющие на объёмы продаж, сформировать конкурентные цены и проводить эффективные маркетинговые компании.
- **Телекоммуникации.** В мире уже произведено больше мобильных телефонов, чем имеется людей на земле. Число находящихся в использовании смартфонов в мире в 2023 году составляет почти 7 миллиардов. Накопление данных об оказанных клиентам услугах (звонках, SMS, передаче информации), а также последующая аналитическая обработка этих данных позволяет идентифицировать поведение пользователей и более точно определять их потребности. На основе этого можно оптимизировать инфраструктуру и сокращать затраты на развитие сети, с меньшими затратами и более полно удовлетворять потребности клиентов. В случае требований гос. органов протоколировать и хранить определенное время все голосовые разговоры, SMS и трафик, телекоммуникационные компании не смогут решить эти задачи без применения инструментов Big Data.
- **Коммунальное хозяйство (электро-, водо-, тепло-, газоснабжение).** В каждом крупном городе в жилом секторе и на предприятиях установлены миллионы счётчиков, с которых регулярно собираются показания, счётчики подлежат учёту, периодически проводится их поверка и замена. Использование «умных» счётчиков, позволяющих регулярно регистрировать и передавать данные по сети, в сочетании с последующей обработкой собираемых данных позволяет улучшить качество обслуживания. В сервисный центр сразу поступают данные об отсутствии подачи электроэнергии, воды и т.п. Кроме того, расширение объёма передаваемых данных, например добавление сведений о поддерживаемой температуре, позволяет сэкономить 10% потребляемых ресурсов за счёт отказа от поставки излишнего тепла. Применение гибких тарифов позволяет влиять на поведение клиентов и сгладить пиковые нагрузки. Обработка накопленных

данных за период позволяет учесть потребности клиентов и оптимизировать инфраструктуру.

- **Муниципальное управление.** Сбор и обработка данных об автомобильном трафике и загруженности магистралей позволяют гражданам оптимизировать маршруты перемещения, экономя время и автомобильное топливо. Использование этого же подхода для оценки использования общественного транспорта позволяет сократить затраты на него и улучшить качество обслуживания. Очень востребованной является регистрация всех заездов автомобилей на парковки и выездов оттуда. Это позволяет водителям узнавать с помощью мобильного телефона наличие свободных мест на парковках и сокращает время поиска свободного места. Такой глобальный сервис уже имеется, и область его действия распространяется на 45 городов мира.
- **Образование.** Применение инструментов Big Data позволяет сформировать и поддерживать индивидуальную модель для каждого обучаемого, в которой будут отражены его индивидуальные характеристики и предпочтения, сведения об уже изученных темах и предметах, отзывы и рекомендации, данные преподавателями и менторами. Соответствующий сервис может быть одновременно использован миллионами пользователей в режиме online обучения, но в то же время предусматривать возможность расширения моделей, обучаемых за счёт сведений поступающих из различных учреждений offline обучения (университетов, колледжей, курсов) [9].

Обобщая возможные применения инструментов Big Data, перечислим типичные задачи, решаемые с их помощью:

- Аналитика по клиентам / объектам;
- Операционная и поведенческая аналитика;
- Построение хранилищ данных, экономически эффективных с точки зрения затрат на единицу объёма хранимых данных;
- Борьба с мошенничеством и контроль соблюдения норм.

4. Системы хранения данных и знаний

В ходе развития вычислительной техники сформировалось несколько классов систем, в которые можно загрузить данные или знания, хранить их там и модифицировать, а также получать их целиком или частично по запросам пользователей или приложений. При этом получаемые результаты в соответствии с запросами могут быть преобразованы и перекомпонованы, но системы хранения не решают каких-либо прикладных задач, а только хранят и обрабатывают информацию по запросам пользователей и/или приложений. Перечень основных типов таких систем:

- Файловые системы, которые появились в 1950-х, первоначально были предназначены для того, чтобы представить перечень хранимых файлов и спрятать от пользователя технические подробности взаимодействия с физическими устройствами. В ходе 70 лет дальнейшего развития были созданы: многослойные (multi-tier) файловые системы, сетевые файловые системы, облачные файловые системы, распределённые файловые системы, Blockchain based Peer-to-Peer file system и другие их разновидности.

- В 1960-х появились и до сих пор широко используются системы управления базами данных (СУБД). По мере развития этих систем появляются всё новые их типы, отвечающие изменяющимся требованиям. На момент написания статьи на сайте db-engines.com приведена информация о 397 СУБД, а число их типов составляет около полутора десятков [10]. Имеется большое число описаний эволюции систем управления систем управления базами данных, например [11–13].
- Создаваемые с 1960-х годов автоматизированные библиотечные информационные системы, поддерживающие каталоги и/или работающие с электронными изданиями [14, 15], а также автоматизированные информационные системы архивов, поддерживающие каталоги и/или работающие с электронными копиями документов, можно рассматривать как специализированные файловые системы.
- С 1970-х годов развивается теория хранилищ данных. Широко распространятся эти системы начали в 1990-х. И сейчас они являются информационной основой многочисленных эксплуатируемых аналитических приложений. Современные системы для построения хранилищ данных помимо обычных offline хранилищ позволяют создавать различные их виды: онлайн, субъектно-ориентированные, неизменяемые, итоговые, реального времени, масштабов предприятия, интегрированные и другие их варианты [16–18].
- С середины 1990-х развиваются идеи создания и использования систем управления мастер-данными. Активное внедрение этих систем началось в середине 2000-х [19]. В 2010-х однодоменные MDM-системы, такие как системы управления мастер-данными о клиентах или системы управления мастер-данными о продуктах развились в мультидоменные MDM-системы [20]. Далее появились контекстуальные MDM-системы, и аналитические MDM-системы [21]. В конце 2010-х появились MDM-системы, работающие в интересах множества предприятий, и мультивекторные MDM-системы [22].
- В 2000-х была разработана система для хранения и обработки больших данных Apache Hadoop. Её открытый исходный код породил появление многочисленных дистрибутивов Hadoop, на основе которых стали развиваться платформы для работы с большими данными [23, 24] и Data Lake, как способ организации данных [25, 26]. В составе платформ для работы с большими данными можно выделить: платформы, сформированные на основе дистрибутивов Hadoop, интегрированные платформы и облачные платформы [27].
- В 2009 г. был реализована первая платформа блокчейн, которая использовалась при выполнении операций с криптовалютой Биткоин [28]. Поскольку в ней использовался открытый программный код, в 2010-х годах появилось много разных версий таких платформ, используемых для операций с криптовалютами [29]. Во второй половине 2010-х технологии блокчейн начали использоваться в корпоративной среде и при работе с гражданами для регистрации прав, финансовых транзакций, операций в цепочках поставок и для других целей [30, 31].
- Системы работы с графами знаний [32], появившиеся во второй половине 2010-х и используемые для семантического поиска информации, включая RDF [33], векторные [34], If-Then [35]. Системы этого класса относятся к более широкой категории систем управления базами знаний, которые включают в себя различные типы систем, каждый из которых ориентирован на конкретный способ представления знаний.

- Системы управления озёрами знаний.

Исходя из функциональности, все перечисленные классы систем можно объединить в одну большую категорию “системы хранения данных и знаний”. Анализ перечисленных классов систем позволяет выделить следующие основные способы организации данных и знаний:

- файловые системы;
- базы данных;
- хранилища данных;
- блокчейн;
- озёра данных;
- базы знаний;
- озёра знаний.

Далее мы увидим, что каждый из перечисленных способов организации имеет множество разновидностей и, в том числе, каждый из них может быть реализован таким образом, что будет использоваться множество (не обязательно большое) параллельно работающих серверов для хранения и обработки данных или знаний. Практически это означает, что независимо от объёма хранимых данных, в каждом таком случае мы будем иметь дело с системой обработки больших данных.

5. Файлы и методы доступа как одни из первичных элементов организации данных

Появление развитие файлов и методов доступа характеризуется следующими основными вехами:

- Данные на голом оборудовании. В 1950-х годах необходимые для работы программ массивы данных часто хранились на бумажных носителях в виде колод перфокарт и рулонов перфолент. При запуске программы требующиеся перфокарты и перфоленты оператор вставлял в устройства ввода, и они считывались программой. Первые устройства для магнитной записи данных – это магнитные барабаны. Первые опыты по их использованию проводились в 1946 и 1947 годах [36]. Магнитная лента впервые была использована для записи данных на компьютере Univac I в 1951 году [37]. Первый магнитный диск создан в 1956 году [38]. Однако при отсутствии методов доступа и файловой системы использование этих устройств в качестве носителей данных мало чем отличалось от использования устройств, работающих с данными на бумажных носителях – программа подключалась к устройству и считывала или записывала массив данных.
- Методы доступа, впервые появившиеся в семействе IBM360 [39], позволили с помощью API обеспечить независимость от конкретных типов устройств и стандартизовать работу с буферами в оперативной памяти. Наиболее распространёнными стали последовательный, прямой, и индексно-последовательный методы доступа. При этом понятие метода доступа у IBM шире, чем интерфейс взаимодействия с устройствами хранения данных, они включают в методы доступа программные средства, реализующие способы взаимодействия с телекоммуникационным оборудованием.

- Методы доступа для работы с виртуальной памятью IBM370, обеспечивающие работу с устройствами хранения данных [40, 41], появились в 1972 году. Набор методов доступа был обновлён в связи с переходом на использование виртуальной памяти при выполнении всех вычислений.
- Методы доступа для поддержки СУБД, разработанные для работы с записями базы данных IBM IMS, представляют собой дополнительные методы доступа, работающие с записями базы данных внутри физических записей, сохраняемых СУБД на диске [42, 43]. При этом доступ к самим физическим записям на диске осуществляется с помощью стандартных методов доступа операционной системы.
- Методы доступа для работы с большими двоичными объектами возникли при реализации парадигмы обработки и хранения объектов. Впервые возможности работы с объектами на уровне файловой системы появились на мини-компьютерах. Позднее, в 1989 году IBM расширила свой набор методов доступа, был добавлен OAM (Object access method) – метод доступа к объектам, предназначенный для работы с наборами данных, не имеющими записей. Во входном потоке поступают большие двоичные объекты, например изображения [44].

Тенденции, которые можно проследить, анализируя приведенный перечень инноваций, состоят в том, что от пользователей (в том числе от программистов) постепенно прячется специфика работы конкретных устройств, сохраняя видимой только логическую организацию и операции работы с данными. Это позволило обеспечить однообразие при работе с разными устройствами.

6. Эволюция файловых систем

Основным свойством файловой системы является возможность хранения и обработки совокупностей файлов. Наряду с платформами обработки больших данных файловые системы являются одним из важнейших способов организации данных, используемых при обработке больших данных. В частности, это подтверждается тем, что самые большие объёмы данных в настоящее время обрабатываются файловыми системами. Ключевые инновации, связанные с появлением и развитием файловых систем:

- Первая файловая система, имеющая классический механизм иерархически организованных папок, была разработана и представлена на конференции в 1958 году [45]. Доступ к каждому файлу и каждой папке осуществляется по иерархическому имени. Вскоре иерархические файловые системы стали общим решением, которое позволяет структурировать пространство имён и организовывать файловую структуру необходимой сложности на внешнем устройстве памяти или в некоторой его области.
- Применение расширений файлов как видимого пользователям элемента метаданных файла, определяющего его тип. Расширения (типы) файлов начали использоваться ещё в середине 1960-х в пользовательской операционной системе CMS для виртуальных машин, работающих под управлением CP-40 и CP-67 на IBM S/360-40 и IBM S/360-67 соответственно [46]. Впоследствии CMS была перенесена в состав VM/370 [47] и последующую линейку этих систем вплоть до используемой в настоящее время z/VM. Однако массовое распространение расширения файлов получили после того, как в 1974

году они появились в CP/M – первой операционной системе для микропроцессоров Z80, которая впоследствии перевоплотилась в CP/M-86 для Intel 8086, IBM PC-DOS и MS-DOS [48].

- Первые виртуальные диски и ленты, отображаемые на реальные физические устройства такой же или большей ёмкости, тоже появились в CP/CMS [49].
- Использование в рамках одной файловой системы нескольких внешних устройств произошло естественным путём за счёт подключения нескольких устройств к одному компьютеру.
- Создание виртуального дорогого быстродействующего устройства с большой памятью из нескольких более простых потребовало дополнительных аппаратных и программных решений. В 1974 году IBM анонсировала ленточную библиотеку IBM 3850 Mass Storage System [50]. С магнитными дисками это произошло позже. Предложение в 1987 году использовать в одной файловой системе несколько однотипных дисков [51] дало начало рынку RAID-систем, которые могут быть как аппаратными, так и программными.
- Использование для хранения файлов иерархии подключенных к одному компьютеру внешних устройств с различными характеристиками и автоматическое перемещение файлов по иерархии для сокращения затрат на хранение при обеспечении приемлемого времени доступа [52]. Первые экспериментальные разработки таких систем появились ещё в 1960-х годах [53]. Однако началом широкого применения файловых систем с иерархическим управлением памятью можно считать появление в 1989 году системы DFSMS для мэйнфреймов IBM, в которой компонент иерархического управления сначала назывался DFHSM [54], а позднее был переименован в DFSMShsm. После этого средства иерархического управления памятью появились в других системах для больших дата-центров [55], а затем настолько распространились, что присутствуют даже в Windows [56, 57] и Mac OS [58]. Вместо перемещения файлов в зависимости от частоты обращения к ним были разработаны значительно более изощрённые методы [59–61].
- Сетевая файловая система предоставляют возможности доступа к файлам другим компьютерам (и их пользователям), подключенным к данному компьютеру (серверу) по сети. Это не только средства для работы с файлами на внешних носителях, но ещё и протокол обмена файлами. Исторически, первые такие протоколы для персональных компьютеров были разработаны в 1983 году в IBM [62] и в Novell, Inc. [63]. Первый стал основой для сетей Microsoft Windows, а второй – Novell Netware.
- Распределённая файловая система – это дальнейшее развитие сетевой: в сети не один, а несколько территориально рассредоточенных серверов, которые поддерживают общее пространство имён файловой системы [64]. Пользователь может подключиться к сети в любом месте, где она доступна. Наличие нескольких (или множества) серверов позволяет делать backup и хранить несколько копий файлов, но рабочей активной является только одна, остальные служат для восстановления в случае сбоя сервера с основной версией файла. Впоследствии термин стал зонтичным и в класс Распределённые файловые системы вошло ещё несколько типов систем, рассматриваемых далее [65].
- Параллельная файловая система обеспечивают чтение информации из разных копий файла, расположенных на разных серверах и дисках сетевой файловой системы [66]. Для обеспечения высокой производительности выполняется максимально возможное

распараллеливание на всех уровнях и используется изощрённый механизм блокировки. Распараллеливается не только чтение данных из файлов, но и чтение метаданных оглавления. Так же как в предыдущем случае, в класс Параллельные файловые системы вошло ещё несколько типов систем, рассматриваемых далее [67].

- Shared-disk file system позволяет устранить задержки и очереди, возникающие в сетевых и распределённых файловых системах на уровне серверов [68]. Клиентские компьютеры по сети подключаются непосредственно к SAN и получают прямой доступ к дискам на уровне блоков. Они полностью независимы, сбой или проблемы с производительностью на одном из них не влияет на другие. Именно на клиентских компьютерах осуществляется управление доступом и преобразование операций уровня файлов, используемых приложениями, в операции уровня блоков, используемые SAN. В системе присутствует механизм блокировок, предотвращающий повреждение и непреднамеренную потерю данных, когда несколько клиентов пытаются одновременно изменять одни и те же файлы. Дублирование и восстановление данных в случае сбоев может быть реализовано как на уровне управлениния пулами дисков, так и у клиентов.
- Cluster-based file system является по определению распределённой, но только в пределах кластера серверов, и параллельной, поскольку должна обеспечивать параллелную работу отдельных серверов кластера. Часто в таких системах имеется Master node, который управляет работой сотен Data node [65]. Возможны дублирующие Master node для сохранения работоспособности в случае сбоя основного сервера и вспомогательные сервера, выполняющие часть функций управления, что помогает разгрузить основной Master node и увеличить размер кластера до нескольких тысяч серверов. Хранимые файлы разбиваются на чанки, чанки распрделяются между разными Data node [65]. Чанки состоят из фиксированного числа блоков. Восстановление в случае сбоев обеспечивается либо хранением нескольких копий одного чанка, либо использованием кодов с возможностью коррекции ошибок.
- Multi-tier cluster-based file system – это файловая систем для использования на кластере, в котором каждый сервер использует иерархию устройств памяти [69].
- Peer-to-Peer file system – это система без централизованного управления. Конечные пользователи делятся ресурсами посредством прямого обмена между компьютерами. Информация распределяется между узлами-участниками, а не концентрируется на нескольких серверах. Это всем знакомые торренты. Системы P2P обладают способностью к самоорганизации, поскольку топология сети P2P может часто меняться по мере того, как узлы входят в систему или выходят из нее. Система P2P значительно отличается от традиционных распределенных систем, огромное количество узлов, участвующих в архитектуре P2P, вносит свой вклад в ее преимущество, такое как адаптивность, масштабируемость и надежность в случае отказов компонентов [70].
- Облачная файловая система – это не какой-то единый продукт. Каждый крупный облачный провайдер, такие, как Amazon, Azure, Google, предлагает целый комплекс сервисов: Block Storage в виде виртуальнх дисков, Object Storage, File Storage, Archive Storage, Hybrid Storage и Bulk Data Transport [71]. Помимо этого, существуют независимые Cloud-backed File Systems, которые могут поддерживать единое пространство

имён, управление доступом и блокировками, работая одновременно с несколькими сервисами хранения данных от разных вендоров облачных услуг [72].

- Fog file system, согласно [73] используется на уровне облака, так и на edge-devices. Эти продукты на момент написания статьи находятся на самом острие развития научных иисследований и разработок промышленности. Пока что ещё нет обзорных статей, можно приводить только примеры таких систем. Один из примеров показан в [74].
- Mist file system работает только на edge-devices. Пример такой системы описан в [75].
- Файловая система с облачным иерархическим управлением памятью, – это файловая система с иерархическим управлением памятью, в которой один или больше слоёв – облачные. Например, два облачных слоя в [76]. Может также присутствовать несколько слоёв реальных устройств.
- Федеративная файловая система оббеспечивает взаимодействие пользователя с несколькими локальными файловыми системами и обеспечивает доступ к расположенным в них файлам. На начальных этапах, когда это делалось для небольшого количества отдельно функционирующих файловых систем, создавалось и в каждой из локальных систем поддерживалось общее виртуальное оглавление, включающее оглавления всех локальных файловых систем, участвующих в федерации [77]. Позднее, когда число взаимодействующих файловых систем (кластеров или дата-центров) стало достигать 150 и более, кроме компонентов, работающих на каждой локальной файловой системе, дополнительно стали разрабатываться и использоваться stand alone системы, чтобы осуществлять поиск файла и редирект в нужную локальную файловую систему [78].
- In-Memory файловые системы сначала были разработаны как средства поддержки и работы с виртуальными дисками в постоянной памяти бездисковых компьютеров [79] и в виртуальной памяти [80. Позднее появилась In-Memory система для памяти NVM [81]. В 2016 году была предложена a file system для виртуальной памяти, которая полностью использует аппаратное обеспечение отображения памяти для доступа к файлу [82].
- Blockchain based Peer-to-Peer file system используют многие решения, разработанные для платформ блокчейн, в частности: идентификацию узлов сети по хеш-коду, сетевые протоколы обмена данными, Merkle DAG для хранения данных, которые идентифицируют данные и ссылки в каждом объекте данных с помощью техники мультихеширования, методы поощрения участников, механизмы консенсуса. Это повышает масштабируемость и конфиденциальность файловых систем [83]. Доступ к файлам осуществляется по хэшу их содержимого. Распределённые файловые системы на базе блокчейн, используются для хранения очень большого количества файлов. Примером такой системы является развиваемая с 2014 года система IPFS. Идентификатор файла в таких системах может включать: код варианта идентификатора, код хеш-функции, код кодировки данных, формат контента и непосредственно данные хеша [84]. Естественно, что конечным пользователям сложно работать с такими идентификаторами файлов, поэтому для них создаются приложения. Типичное приложение содержит распределённую таблицу, в которой сопоставляются друг с другом понятные пользователю идентификаторы на естественном языке и содержащие данные хеширования идентификаторы в файловой системе.

- Семантические файловые системы. Первоначальная идея этих систем была, что это системы хранения, которые обеспечивают гибкий ассоциативный доступ к содержимому системы путем автоматического извлечения атрибутов из файлов с помощью преобразователей, специфичных для типа файла. Ассоциативный доступ обеспечивается консервативным расширением существующих протоколов древовидной файловой системы и протоколами, которые разработаны специально для доступа на основе содержимого [85]. Однако в реализованных проектах это системы, в которых семантика файлов определяется с помощью присвоения тэгов и по тэгам осуществляется поиск необходимых файлов. В большинстве случаев тэги присваиваются вручную. При большом разнообразии тем хранимых файлов и использовании файловой системы группой пользователей, число тэгов быстро разрастается, появляется необходимость использовать и поддерживать отдельную большую таблицу тэгов.

Файловые системы разрабатывались и разрабатываются сейчас не только как самостоятельные программные системы, поддерживающие определённый способ организации и обработки данных, но и как системы, встроенные в другие способы и системы организации данных (СУБД, интегрированные или облачные платформы), а также встроенные в приложения. Архитектура и возможности таких встроенных файловых систем могут быть опубликованы, например, как в случае CFS (Cassandra File System) – встроенной файловой системы СУБД Cassandra [86, 87], или раскрываться только во внутренних документах разработчика. Встроенные файловые системы часто присутствуют как компонент в следующих типах приложений:

- Document Image Processing System [88].
- Транзакционные On-Premise и облачные системы управления документами и записями (Document Management System, Record Management System) [89].
- Библиотечные информационные системы, поддерживающие каталоги и/или работающие с электронными изданиями [14, 15].
- Специализированные и универсальные тиражируемые системы для хранения мультимедийного контента и соответствующих метаданных, в том числе системы управления контентом масштаба предприятия (Enterprise Content Management – ECM).
- Цифровые фильмофонды, в том числе используемые для доступа из сети.
- GIS, имеющие хранилища картографической информации.
- Wiki-системы.
- Git-системы.
- Архивы документов на торрент-клиентах.

Приведенный перечень, скорее всего, – не полный, существуют и другие типы приложений, где имеются встроенные файловые системы. Например, весь интернет можно рассматривать как глобальную распределённую файловую систему с централизованным управлением.

Рассмотренный перечень инноваций в ходе развития файловых систем позволяет выделить следующие присущие их развитию тенденции:

- Первоначальные решения впоследствии были распространены на разные виды памяти: оперативную, виртуальную, облачную, ...

- Переход от одного объекта к множеству объектов: первоначально использовали одно внешнее устройство, потом – несколько, первоначально работали на одном сервере, потом – на нескольких и далее – на кластере серверов.
- Последовательное увеличение сложности файловых систем как способа организации данных и поддерживающего программного обеспечения в связи с общим развитием информационных технологий и средств вычислительной техники.
- Быстрое распространение диалогового доступа к оглавлению файловой системы.
- Постепенное всё более широкое распространение механизмов виртуализации.
- Постепенное всё более широкое применение механизмов распараллеливания вычислений.
- Последовательное увеличение мобильности: сначала доступ к данным в точке их расположения, потом в любой точке университетского или корпоративного кампуса, потом в любой точке мира, где есть интернет.
- Постепенно расширяющееся комбинирование различных способов организации данных. Например: файловая система поверх облачной платформы Big Data, файловая система поверх цепочки блокчейн.
- Последовательное увеличение числа пользователей одной файловой системы вплоть до 100 миллионов и более для Blockchain based Peer-to-Peer file system.
- Постепенное повышение уровня интеллектуальности файловых систем, в частности применение методов Policy Learning и Reinforcement Learning при управлении перемещением блоков между уровнями иерархии системы хранения.
- Постепенное расширение диверсификации: сначала разделение на системы с файловым и блочным хранением, далее двух инструментов уже недостаточно, облачные провайдеры предлагают шесть разных сервисов, удовлетворяющих различные потребности пользователей, а кроме этого, есть ещё Cloud-backed File Systems, Fog file systems, Mist file systems, …
- Возникновение федеративности, когда одна файловая система взаимодействует с другими файловыми системами для обеспечения доступа к расположенным в них файлам.
- Глобализация – распространение одной файловой системы сначала на несколько компьютеров в одной локальной сети, потом на сервера в разных ЦОДах, далее на разные страны, континенты, на весь земной шар, как в случае Peer-to-Peer file system.

7. Возникновение платформ обработки больших данных

Предпосылки для появления платформ обработки больших данных сложились в результате развития трёх направлений вычислительной техники – распределённых файловых систем, систем организации распределённых вычислений и NoSQL СУБД, которые поддерживают высокую скорость обработки данных за счёт параллельного использования нескольких серверов. Состояние во всех этих трёх областях в данном разделе будет нас интересовать на момент начала 2000-х годов, когда был создан Hadoop. Пока мы не ушли далеко от файловых систем рассмотрим более подробно ситуацию в это время в области файловых систем.

7.1. Сетевые и распределённые файловые системы для хранения больших объёмов данных

7.1.1. Сетевые файловые системы

Впервые распределение файлов по сети компьютеров было реализовано компанией DEC в 1976 г. для сети DECnet из миникомпьютеров PDP-11. Первые действительно массово применяемые сетевые файловые системы были созданы в середине 80-х годов: Network File System (NFS) использовалась в Linux/Unix, а CIFS (Common Internet File System) использовалась Microsoft Windows [90].

NFS была разработана в 1984 г. Sun Microsystems. Она позволяла распределить файлы по различным узлам сети и работать с ними, обращаясь к ним через сеть. NFS имела много версий, она начала широко использоваться, начиная с версии NFSv2, появившейся в марте 1989 г. NFSv3 вышла в июне 1995 года. NFSv4 была выпущена в декабре 2000 г., в ней были улучшены безопасность и производительность. В 2010 г. была одобрена версия NFSv4.1. Важным нововведением версии 4.1 является спецификация pNFS – Parallel NFS, механизма параллельного доступа NFS-клиента к данным множества распределенных NFS-серверов. Наличие такого механизма в стандарте сетевой файловой системы поможет строить распределённые облачные хранилища и информационные системы. NFS – это распределённая файловая система с большой историей, и она продолжает развиваться.

CIFS (сокр. от англ. Common Internet File System, Единая Файловая Система Интернета) — сетевой протокол прикладного уровня для удалённого доступа к файлам, принтерам и другим сетевым ресурсам, а также для межпроцессного взаимодействия. Является первой версией протокола SMB (Server Message Block). CIFS была разработана компаниями IBM, Microsoft, Intel и 3Com в 1980-х годах. Вторая версия (SMB 2.0) была создана Microsoft и появилась в Windows Vista. В настоящее время SMB связан главным образом с операционными системами Microsoft Windows, где используется для реализации «Сети Microsoft Windows» и «Совместного использования файлов и принтеров» (англ. File and Printer Sharing).

NFS и CIFS соответствуют стандарту POSIX. Т.е. приложения, используя NFS или CIFS, могут работать с распределённой файловой системой, как будто они работают с локальной файловой системой. Это означает, что при внедрении нового приложения или выполнении существующего не возникает необходимости разным способом готовить данные в локальной файловой системе и в распределённой файловой системе.

При использовании NFS или CIFS широко применяются NAS (Network Attached Storage) для улучшения производительности и скорости доступа к данным. NAS имеют свою уникальную файловую систему и выполняют поступающие по сети запросы на доступ к файлам через NAS gateway, поддерживающий протоколы NFS или CIFS. Общая схема NAS показана на рисунке 6.

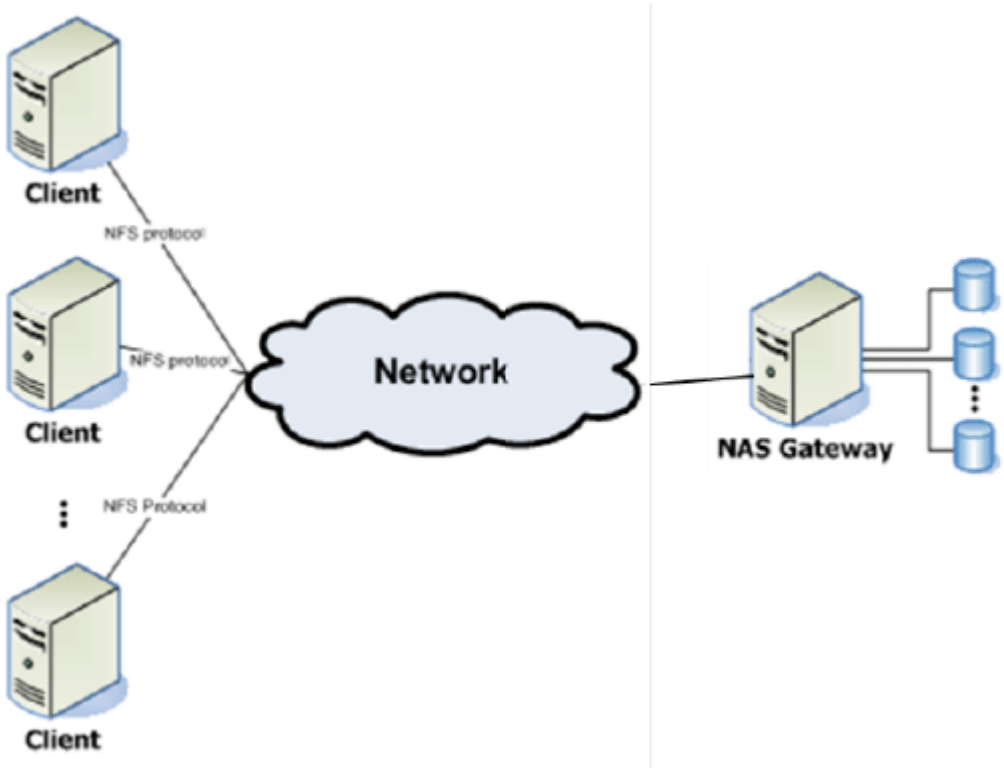


Рис. 6. Общая схема NAS

7.1.2. Распределённые файловые системы

В ходе эволюции файловых систем произошёл переход от сетевых систем, когда файл целиком располагается на одном узле сети, к распределённым, когда разные блоки одного файла располагаются на разных узлах сети. Распределённые файловые системы на кластерах начали обсуждаться и проектироваться ещё в середине 90-х годов XX века после того, как стало понятно, что NAS-системы в состоянии обеспечить хранение большого объёма данных, но не в состоянии обеспечить пропускную способность, необходимую для параллельной работы большого числа процессоров, выполняющих обработку данных (независимо от того, кто управляет их работой - независимые пользователи или распределённая программа). С этого момента до настоящего времени спроектированы и разработаны десятки таких систем: Vesta, Galley, PVFS, Swift, GPFS, StorageTank, LegionFS, Google File System, Federated Array of Bricks (FAB), pNFS, Lustre, Panasas file system, zFS, Sorrento, Kybos, Ceph, Intel's Distributed Application Object Storage (DAOS), RADOS, Sirocco, Ursa Minor, SOS – этот список взят из разделов, посвящённых обзору близких работ всего двух публикаций, посвящённых созданию распределённых файловых систем [91, 92]. Обращение к Википедии даст дополнительно полтора – два десятка наименований [93]. В начале 2000-х годов распределённые файловые системы стали рассматриваться как основной инструментарий для хранения данных объёмом несколько петабайтов.

Перечисленные выше системы разрабатывались как университетскими командами и стартапами, так и гигантами отрасли, такими как Intel и IBM. Они имеют разный уровень готовности. В то время как одни из них представляют собой пилотные проекты, разработанные для иллюстрации некоторых новых концепций, другие уже являются зрелыми продуктами, прошедшими проверку и доводку во многих внедрениях. С ростом зрелости рынка в последние

годы появились публикации, в которых приводятся результаты сравнительного тестирования нескольких распределённых файловых систем [94, 95].

7.2. Система распределённых вычислений MPI

В 80-х годах в силу своей относительной дешевизны и высокой удельной производительности на единицу вложенных средств получили распространение кластерные суперкомпьютеры, имеющие архитектуру MPP (Massively Parallel Processing – массовая параллельность). В MPP-системе оперативная память – распределенная, т. е. каждый вычислительный модуль имеет прямой доступ лишь к своей – локальной памяти. Модуль представляет собой один процессор или несколько процессоров с общей памятью. К нелокальным данным запросы производятся через коммуникационную среду. Главным достоинством MPP-суперкомпьютера является масштабируемость – количество процессоров может доходить до сотен тысяч. Недостатки системы вызваны той же причиной, что и ее единственное достоинство. Распределение памяти по вычислительным модулям значительно усложняет распараллеливание программ, и ведет к значительной потере эффективности использования большого количества процессоров в случае, когда метод плохо распараллеливается [96].

Для решения проблем, возникающих при разработке программ для MPP-систем, в 1980-х годах разными коллективами разработчиков велась разработка систем передачи сообщений. В качестве примеров можно указать: p4, PICL, PARMACS, PVM, TCGMSG, Zipcode, Express и др. [97]. Возникла необходимость координировать и стандартизировать этот процесс, поэтому в рамках конференции Supercomputing’92 состоялось совещание, на котором было принято решение о разработке стандарта программного интерфейса обмена сообщениями. Процесс стандартизации был поддержан индустрией, поскольку она заинтересована в средстве разработки эффективных программ для высокопроизводительных вычислительных систем. В разработке стандарта в разное время участвовали компании: Convex, Cray, IBM, Intel, Meiko, nCUBE, NEC, Thinking Machines [97].

Версия MPI 1 была принята в 1994 году.

Версия MPI 2 была принята в 1998 году, первая реализация появилась в 2002 году.

Версия MPI 2.1 была принята 4 сентября 2008 года.

Версия MPI 2.2 была принята 4 сентября 2009 года.

Версия MPI 3.0 была принята 21 сентября 2012 года.

Версия MPI 3.1 была принята 4 июня 2015 года.

Версия MPI 3.3 была принята 30 июня 2020 года.

Версия MPI 4.0 была принята 9 июня 2021 года.

Версия MPI 4.1.6 была принята 30 сентября 2023 года

В настоящее действующей является версия MPI 5.0.5, принятая 22 июля 2024 года [98].

MPI расшифровывается как "Message passing interface" ("Интерфейс передачи сообщений"). MPI предоставляет программисту единый механизм взаимодействия процессов внутри параллельно исполняемой задачи независимо от машинной архитектуры (однопроцессорные, многопроцессорные с общей или раздельной памятью), взаимного расположения процессов (на одном физическом процессоре или на разных) и API операционной системы. Программа, использующая MPI, легко отлаживается и переносится на другие платформы, часто для этого достаточно простой перекомпиляции исходного текста программы [99].

Узлы MPP-суперкомпьютера, работающего под управлением MPI, относительно самостоятельны, так же как узлы кластера Big Data. Главное отличие состоит в том, что они не имеют собственной дисковой памяти. Сильная сторона MPI состоит в том, что этот интерфейс предъявляет очень низкие требования к аппаратной части каждого узла кластера. Все, что нужно этому интерфейсу, – чтобы процессоры или ядра совместно использовали одну сеть, пригодную для передачи сообщений между любыми двумя процессами. Это позволяет MPI работать на широком спектре оборудования [100]. В принципе, его можно развернуть на сети из нескольких локальных компьютеров. Ещё одним преимуществом MPI является то, что на нём реализовано большое число параллельных численных алгоритмов. Для выполнения какого-нибудь численного расчёта можно найти готовую программу в одной из многочисленных библиотек параллельных методов и минимизировать затраты ресурсов на программирование.

Тем не менее, хотя MPI и обеспечивает высокий уровень распределённой обработки данных, в полной мере этот механизм не является инструментом обработки больших данных, поскольку в узлах MPI нет хранения данных. Основная область применения этого инструмента-суперкомпьютерные кластеры, на которых выполняются сложные расчёты.

7.3. Базы данных NoSQL

Эволюция систем управления базами данных сейчас содержит 4 периода и три революции, которые имели место при переходе от одного периода к другому [101].

На первом этапе для организации хранения данных использовались плоские файлы и различные методы доступа (последовательный, прямой, индексно-последовательный (ISAM), библиотечный и др.). Данные хранились на магнитных лентах, магнитных барабанах, магнитных дисках.

Задачи первой революции состояли в том, чтобы обеспечить независимость приложений от организации физического хранения данных (структуры файлов и их размещения, организации доступа). В результате, на втором этапе развития появились СУБД первого поколения, которые работали на больших мэйнфреймах. Использовались две основные модели организации данных – сетевая и иерархическая. Сетевая модель была формализована в предложениях рабочей группы по языкам систем обработки данных (DBTG CODASYL) [102], а иерархическая модель развивалась компанией IBM для СУБД IMS. Помимо этих двух доминирующих способов описания данных существовали различные вариации сетевой и иерархической моделей, используемые в конкретных СУБД. Основным недостатком обоих подходов была необходимость указывать путь (последовательность уровней иерархии или ссылок) для доступа к нужным данным. Именно поэтому иерархическую и сетевую модель называют навигационными.

Вторая революция началась, когда Эдгар Кодд предложил в 1970 г. реляционную модель данных. Основная идея состояла в том, чтобы нормализовать таблицы и использовать операции теории множеств для получения необходимых данных. Разработка первых экспериментальные СУБД, основанных на реляционной модели, началась только в середине 70-х годов, а первые промышленные системы появились только в начале 80-х. Более 10 лет мощная теория не поддерживалась доступными для практического применения системами. С тех пор СУБД на основе реляционной модели стали общепринятыми. Практика развития методов и языков программирования внесла некоторые расширения в используемую модель:

- в качестве отдельных элементов записи стало можно хранить сложные объекты;
- появились In-memory системы и системы с поколоночным хранением таблиц.

Однако на основную суть реляционных СУБД это не повлияло. В начале 2000-х годов стали очевидны пределы, в которые упираются традиционные реляционные базы данных:

- какими бы мощные сервера мы ни использовали, поддержать базу данных объёмом более 200–300 Тб – очень сложно;
- реляционные СУБД в силу используемых механизмов не способны обрабатывать большие потоки данных, не хватает скорости.

Это положило начало четвёртому этапу развития СУБД – появились NoSQL системы, преодолевающие указанные ограничения:

- **Key-Value СУБД**, идея которых восходит к индексно последовательному методу доступа. Очень простая модель данных позволяет распараллелить большой поток данных (например, поступающий из интернета) между несколькими серверами и обеспечить необходимую скорость обработки потока.
- **Документарные СУБД**, идеи которых берут своё начало от библиотечных систем, в которых для каждой книги хранится небольшой набор ключевых слов, по которым идёт поиск. Также и в документарных СУБД для каждого документа имеется фиксированное количество ключевых признаков, значения которых задаются с помощью JSON.
- **Суперстолбцовые СУБД,** модель данных которых представляет собой таблицы с очень большим числом столбцов, но в каждой строке хранятся только непустые значения. Хороший пример для демонстрации этой модели – интернет-магазин компьютерных комплектующих, в котором для каждого товара (памяти, мониторов, дисков и пр.) хранятся только те характеристики, которые описывают этот товар.
- **Табличные СУБД,** хранящие в памяти полноразмерные таблицы, но имеющие ограниченные возможности по работе со строками и столбцами этих таблиц.
- **Seach Engines**, предназначенные для поиска документов в сети по значениям и комбинациям содержащихся в них слов (пример – поиск Yandex или Google). СУБД этого типа строят и хранят очень большой индекс, но сами документы, с которыми они работают распределены по сети.
- **Графовые СУБД,** предназначенные для работы с большими графами, часто не умещающимися на одном сервере. Доступ к данным в этих системах – навигационный. Данные привязаны к узлам и рёбрам графа. Перемещение между узлами графа, выполняется по имеющимся рёбрам. Графовые СУБД – это новая реинкарнация идей, развивавшихся DBTG CODASYL.

NoSQL СУБД за счёт распараллеливания обработки и упрощения методов обработки позволили обрабатывать намного большие объёмы данных, однако при этом часто возникают нарушения целостности данных, и чтобы её поддержать применяются специальные механизмы.

Общей теории NoSQL СУБД пока нет, хотя системы применяются уже более 20 лет. Этим этап сильно отличается от предыдущего.

7.4. Hadoop

Одной из важнейших вех в развитии инструментов Big Data является создание Hadoop в начале 2000-х годов [103]. Хотя и до этого были работы по созданию распределённых файловых систем, именно в Hadoop распределённая файловая система HDFS (Hadoop Distributed File System) была объединена с фреймворком MapReduce. В результате появился инструмент обработки данных, в котором была очень чётко представлена основная идея решений Big Data – данные распределены между узлами кластера, и каждый узел обрабатывает расположенные на нём блоки данных. Этот инструмент стало возможно использовать для решения самых разных задач по сбору и обработке больших данных.

7.4.1. История разработки Hadoop

Hadoop создал Дуг Каттинг – создатель Apache Lucene, широко используемой библиотеки текстового поиска. Hadoop произошёл от Apache Nutch – системы поиска с открытым кодом, которая сама по себе являлась частью проекта Lucene. Построение поисковой системы с нуля было амбициозной целью – не только из-за сложности написания программного обеспечения для обхода и индексирования веб-сайтов, но и из-за сложности ведения проекта, содержащего огромное количество часто изменяющихся компонентов, без специально выделенной группы управления проектом. Кроме того, задача была весьма дорогостоящей: по оценкам Майка Кафареллы и Дуга Каттинга, оборудование системы, ведущей индекс для миллиарда страниц, стоило полмиллиона долларов, а ежемесячные затраты на его содержание составляли $30 000 [104]. Тем не менее они считали, что цель того стоит, так как результатом будет открытие и исключительная демократизация алгоритмов поисковых систем.

Проект Nutch был запущен в 2002 году. Работоспособный обходчик и поисковая система появились очень быстро. Однако разработчики поняли, что их архитектура не будет масштабироваться на миллиарды страниц. Помощь пришла в 2003 году, когда была опубликована статья с описанием архитектуры GFS (Google File System) – распределённой файловой системы, которая использовалась в реальных проектах Google [105]. Система GFS или какая-то подобная могла бы решить проблемы хранения больших файлов, генерируемых в процессе обхода и индексирования. В частности, система GFS сэкономила бы время на выполнение таких административных задач, как управление узлами хранения данных. В 2004 году разработчики взялись за написание такой системы с открытым кодом – NDFS (Nutch Distributed Filesystem).

В 2004 году была опубликована статья [106], в которой компания Google представила технологию MapReduce. В начале 2005 года у разработчиков Nutch появилась работоспособная реализация MapReduce на базе Nutch, а к середине года все основные алгоритмы Nutch были адаптированы для использования MapReduce и NDFS.

Возможности применения NDFS и реализации MapReduce в Nutch выходили далеко за рамки поиска, и в феврале 2006 года был образован независимый подпроект Lucene, получивший наименование Hadoop. Примерно в то же время Дуг Каттинг поступил на работу в компанию Yahoo!, которая предоставила группу и ресурсы для превращения Hadoop в систему, которая может обрабатывать данные в масштабах всего интернета. Результаты были продемонстрированы в феврале 2008 г., когда компания Yahoo! объявила, что используемый ею поисковый индекс был сгенерирован 10 000-ядерным кластером Hadoop [107].

В 2008 г. Hadoop стал одним из ведущих проектов Apache, что доказало его успех и наличие широкого, разнообразного активного сообщества. К этому времени технология Hadoop использовалась не только в Yahoo!, но и во многих других компаниях – например, в Last.fm, Facebook и New York Times.

7.4.2. Принципы работы

В используемом кластере выделяется главный узел, который организует процесс, и узлы данных, на которых располагаются фрагменты данных, и выполняется их обработка. Изначально предполагается, что узлы кластера – это низко-надёжные компьютеры эконом класса (commodity servers).

Основная идея HDFS состоит в том, чтобы разделить массив больших данных на блоки и распределить эти блоки между узлами данных вычислительного кластера. Все блоки имеют одинаковый размер. Поскольку узлы кластера не обладают высокой надёжностью, каждый блок размещается на нескольких узлах в соответствии с предварительно установленным коэффициентом репликации. В случае выхода одного из узлов из строя, все располагавшиеся на нём блоки копируются на другие узлы, чтобы поддерживать заданное число копий. Это обеспечивает устойчивость к отказам. HDFS поддерживает традиционное иерархическое пространство имён: главным является корневой каталог, каталоги могут быть вложены друг в друга, в одном каталоге могут располагаться файлы и другие каталоги. Обновление существующих файлов не поддерживается, необходимо записывать изменившийся файл как новый.

MapReduce предназначен для того, чтобы организовать параллельный процесс решения задачи на кластере. Процесс выполнения строится из двух фаз: фазы отображения Map и фазы свёртки Reduce. Функции Map и Reduce определяются разработчиком в зависимости от решаемой задачи. Функция Map выполняет первичную обработку данных, лежащих в HDFS. Она запускается на узлах данных, где лежат блоки файла с исходными данными. Результаты работы функции Map передаются функции Reduce, которая объединяет результаты, полученные независимо выполнявшимися функциями Map. В связи с тем, что на первой фазе может выполняться большое число функций Map, для свёртки результатов может быть параллельно запущено много функций Reduce. Однако, в конечном счёте, все их результаты подаются на вход одной функции Reduce, которая формирует окончательный результат. Надёжность работы обеспечивается повторным выполнением задач. В случае если какая-то из задач не выполнилась из-сбоя и не выдала результат, она повторно запускается на другом узле.

7.4.3. Современное состояние

В 2013 году появилась версия Hadoop 2.0 [108]. Два самых важных нововведения, которые были реализованы в этой версии:

- В составе Hadoop появился менеджер YARN, отвечающий за управление ресурсами кластера и планирование заданий. MapReduce реализован поверх YARN, как один из вариантов обработки данных. С помощью YARN можно реализовать и другие схемы обработки, например, представленные графом сложной структуры, в узлах которого будут выполняться определённые функции. YARN обеспечивает возможность параллельного выполнения нескольких различных задач в рамках кластера и их изоляцию.

- Часть функций по мониторингу заданий была снята с центрального узла, распределяющего ресурсы, и перенесена на новый тип узлов ApplicationMaster. Распределение ресурсов реализовано более эффективно. За счёт этого повысилась производительность Hadoop на 10-15% при одной и той же конфигурации кластера. Ещё одним важным эффектом стал рост реального числа узлов, которые могут эффективно работать в кластере с 4-х до 10 тысяч.

Последним стабильным релизом версии 2 является 2.10.2, выпущенный в мае 2022 года.

В декабре 2017 года появился первый после бета-версии релиз 3.0.0. За счёт применения кодирования, исправляющего ошибки в версии 3.0 устранено дублирование данных на нескольких узлах кластера. Это позволяет в несколько раз увеличить эффективность использования дисковой памяти. Главная цель перехода к версии 3 – дальнейшее усовершенствование архитектуры и обеспечение поддержки кластеров, включающих несколько десятков тысяч серверов. В настоящее время параллельно развивается стабильный релиз 3.3.6 от июня 2023 года и в марте 2024 года выпущен релиз 3.4. При переходе от релиза к релизу исправляется несколько сотен багов. Последний релиз ещё широко не тестировался.

Когда Дуг Каттинг искал файловую систему для создаваемого с нуля проекта Nutch, который, как часть, включал в себя Hadoop, он взял за основу идеи GFS (Google File System) и реализовал систему с открытым кодом NDFS (Nutch Distributed File Systems), которая в последствие стала называться HDFS. Это было хорошим решением с практической точки зрения – система была достаточно простой в реализации, идеально сочеталась с MapReduce и обеспечивала надёжное хранение файлов большого объёма, которые не могли поместиться в памяти одного узла. Однако, дальнейшее развитие Hadoop и теоретические исследования показали, что не всё идеально. HDFS:

- спроектирована в расчёте только на однопользовательский режим;
- имеет архитектуру Master-Slave, при которой все метаданные размещаются в головном узле, и поэтому система имеет ограничения по масштабированию;
- не полностью POSIX-совместима;
- хорошо справляется с большими файлами, но сильно теряет производительность при работе с большим числом мелких файлов.

Поэтому достаточно быстро появились проприетарные и Open Source дистрибутивы, в которых предлагались варианты усовершенствования или замены HDFS в составе Hadoop другой файловой системой. Этот процесс продолжается по настоящее время, в частности продолжается развитие проекта Ozone [109]. Это масштабируемое, распределенное хранилище с резервированием, называемое HDDS (Hadoop Distributed Data Store). Предназначено для хранения объектов Hadoop. Помимо масштабирования до миллиардов объектов разного размера, Ozone может эффективно работать в контейнерных средах, таких как Kubernetes и YARN.

7.5. Экосистема Hadoop

Широкий спектр задач, решаемых Hadoop, привёл к созданию большого числа дополнительных программных продуктов, которые органично расширяют возможности первоначальной системы. Быстрому появлению множества новых продуктов способствовал открытый характер лицензии Apache Hadoop. Нередко новые программные продукты начинали свое

развитие в коммерческих компаниях, таких как: Twitter, Facebook, Amazon, но потом были переданы в сообщество под свободными лицензиями. К настоящему времени Hadoop является центром большой экосистемы. Число продуктов, входящих в экосистему, составляет несколько сотен. Существуют разные схемы классификации этих продуктов, различающиеся как по составу классов продуктов, так и по составу классифицируемых продуктов [110–112]. Ниже предлагается классификация входящих в экосистему продуктов, в которой сделана попытка закрыть белые пятна, присутствующие в каждом из перечисленных источников [113]. В таблице 1 перечислены все выделенные классы, указывается назначение каждого класса, а также представлены примеры программных продуктов, отнесённых к этому классу. Таблица не претендует на сколько-нибудь полный список продуктов.

Таблица 1. Состав экосистемы Hadoop

Класс продуктов	Назначение	Примеры продуктов
Распределенные файловые системы	Экосистема Hadoop сконфигурирована таким образом, что может использоваться целый спектр различных распределенных файловых систем. В большинстве прикладных задач применяется классическая HDFS, но некоторые вендоры развивают свои дополнительные решения в этой области.	HDFS, FTP File system, Amazon S3, Windows Azure Storage Blobs (WASB), IBM General Parallel File System, Parascale file system, Appistry CloudIQ Storage, Lustre, Ceph, Intel's Distributed Application Object Storage (DAOS)
NoSQL СУБД	Наибольшее распространение в экосистеме Hadoop получили NoSQL СУБД, которые весьма удобны для обработки и хранения разнообразной структурированной и плохо структурированной информации. Все базы данных, задействованные в экосистеме, рассчитаны на большие массивы данных и реализуют повышенные требования по отказоустойчивости. Как правило, эти БД используют одну из следующих моделей данных: ключ-значение, документоориентированная, потокоориентированная, графоориентированная.	Apache HBase, Apache HCatalog, Hypertable, Apache Acuumulo, Apache Cassandra, Druid, Neo4j, InfoGrid
NewSQL Базы данных	Представляют собой новое поколение классических реляционных баз данных, но с улучшенной поддержкой горизонтального масштабирования и поддержкой шардирования.	MemSQL, VoltDB
Интерпретаторы SQL запросов	Реализуют интерпретацию SQL-like языка запроса, позволяя решать задачи выборки данных простым и знакомым индустрии способом. Каждая реализация отличается полнотой реализации SQL и производительностью.	Apache Hive, Apache Drill, Apache Spark SQL, Apache Phoenix, Cloudera Impala, HAWQ for Pivotal HD, Presto, Oracle Big Data SQL, IBM BigSQL Hadapt
Интерпретаторы других языков запросов	Реализуют специализированные языки запросов, которые могут обладать преимуществами перед SQL на определенных классах задачах.	Apache Pig, Clydesdale, Perldoop
Системы текстового поиска и ндексирования текстов	Выполняют построение индексов, глобальный и корпоративный текстовый поиск	Apache Lucene, Apache Nutch, Apache Solr, ElasticSearch, Katta, HFSS (Forensic Indexed Search System), DTPS (Distributed Text Processing System)
Обработчики геопространственных данных	Обработка векторных геопространственных данных, представленных в виде шейп-файлов	GS-Hadoop

Обработчики данных широкого спектра применения	Обеспечивают непосредственно способ обработки поступающих в экосистему данных. Обработчики отличаются типом решаемых прикладных задач, стеком применяемых технологий и способом реализации прикладных задач.	Apache MapReduce, Apache Tez, Apache Hama
Интегрированные платформы	Объединяют несколько модулей, реализующих различные методы обработки данных	Apache Spark, Apache Flink
Системы для работы с большими графами	Выполняют операции с графами, которые не умещаются на одном компьютере	Apache Giraph, GraphLab, GBASE, Spark GraphX
Системы потоковой обработки данных	Обрабатывают потоки данных в режиме реального времени	Apache Storm, Apache Hadoop Streaming, Apache Samza
Системы управления задачами кластера	Планирование и выполнение различных Hadoop и прочих задач. Управление очередью и временем исполнения.	Apache Oozie, Apache Fair Scheduler, Apache CapacityScheduler
Менеджеры кластера	Распределяют ресурсы кластера в процессе решения задач	Apache YARN, Apache Mesos
Библиотеки безопасности и управления доступом	Реализуют пользовательские политики доступа к различным компонентам экосистемы, а также синхронизацию с пользовательскими службами различных операционных систем.	Apache Knox, Apache Ranger,
Сервисы аналитики и визуализации данных	Занимают центральное место в экосистеме Hadoop, как продукты, формирующие добавочную ценность обработанной информации. Реализуют непосредственную визуализацию данных, удобную для конечного пользователя.	Datameer, Pentaho, Pivotal HD, RapidMiner Radoop, SPARQLcity
Библиотеки машинного обучения	Решение задач кластеризации, фильтрации, категоризации данных.	Apache Mahout, Spark MLBase
Библиотеки сериализации данных	Хранение данных в удобном для прикладных приложений виде, а также упрощение обмена сложно структурированной информацией.	Apache Avro, Apache Thrift
Системы развёртывания	Установка дополнительных сервисов кластера. Управление конфигурациями кластера. Пуск, останов, реконфигурация Hadoop, сервисов всего кластера.	Apache Ambari Cloudera Director
Сервисы сбора, модификации, перемещения данных, а также сервисы обмена сообщениями	Решают задачу опроса различных источников данных (получение логов с различных серверов, граббинг сайтов и т.д.), преобразования к общему формату, сохранение данных в используемой среде хранения.	Apache Flume, Apache Sqoop, Apache Kafka Apache NiFi
Дистрибутивы Hadoop	Программные продукты различных производителей, включающие преднастроенные библиотеки и сервисы экосистемы Hadoop, как свободные, так и проприетарные.	Hortonworks HDP, Cloudera CDH,
Системы координации	Реализуют хранение конфигурации, именования, блокировки, и устранения “гонки” за ресурс для узлов кластера.	Apache ZooKeeper
Системы мониторинга и аудита	Мониторинг кластера при помощи досок объявлений (dashboards), ведение метрик, уведомление о событиях кластера (отсутствие места на диске, падение узла и т.д.)	Apache Ambari, Apache Knox, Apache Ranger, Apache ZooKeeper Apache Chukwa
Библиотеки тестирования	Тестирование различных модулей и компонентов экосистемы, тестовые наборы данных	Apache Bigtop
Прочие библиотеки	Библиотеки, упрощающие разработку прикладных приложений	Cascading Development Framework

Как видно из представленной таблицы, экосистема Hadoop представляет разработчику набор средств, который позволяет построить программно-аппаратное решение для широкого

спектра задач с практически неограниченными объемами данных, которые нужно обработать или проанализировать. Многообразие представленных на рынке решений отображает колоссальные темпы развития данной экосистемы, а также всей совокупности решений, связанных с обработкой больших данных.

Однако всё многообразие имеющихся пакетов плохо совместимо между собой, новые версии выпускаются по независимым графикам. Для поддержки эксплуатируемого в компании комплекса из нескольких десятков пакетов необходима большая команда дата-инженеров.

8. Первое поколение платформ Big Data: платформы на основе дистрибутивов Hadoop

Для сокращения затрат на развёртывание и интеграцию технологических платформ ряд компаний начали выпускать перечисленные далее единые дистрибутивы, содержащие Hadoop и 2–3 десятка работающих на нём пакетов. При установке можно было из коробки получить отлаженный комплекс, на котором можно было развивать и эксплуатировать прикладные решения. Этих платформ сейчас становится всё меньше, но у оставшихся до сих пор появляются новые версии. Постепенно они вытесняются с рынка.

8.1. Hortonworks Data Platform

Hortonworks – американская компания [114], которая была основана в 2011 году двадцатью четырьмя инженерами Yahoo!. Компания утверждала, что у её сотрудников больше всего опыта в работе с Hadoop. Hortonworks Data Platform представлял собой полностью открытый дистрибутив Hadoop масштаба предприятия [115]. В дистрибутив вносились все основные нововведения, появляющиеся в Apache Hadoop, и в то же время обеспечивается совместимость более чем с сотней других активно развивающихся проектов Big Data. SAP поставлял дистрибутив Hadoop от Hortonworks под маркой SAP HD.

В 2013 году Hortonworks объявила о доступности продукта Hortonworks Data Platform 1.3 (HDP) для Windows. В тот момент HDP 1.3 для Windows с открытым исходным кодом представлял собой единственный дистрибутив на базе Apache Hadoop, сертифицированный для работы под управлением Windows Server 2008 R2 и Windows Server 2012. Он позволял создавать и развертывать аналитические приложения в операционных системах Windows на основе Hadoop.

Помимо HDP Hortonworks поставлял Hortonworks DataFlow (HDF) – разработанную с использованием Apache NiFi интегрированную платформу для сбора данных в режиме реального времени из множества источников и управления потоками данных [116].

В октябре 2018 года Hortonworks и Cloudera объявили о своем слиянии через слияние всех акций равных компаний. После слияния продукты Apache компании Hortonworks превратились в Cloudera Data Platform.

8.2. Cloudera Enterprise Data Hub

Cloudera, Inc. – американская компания, основанная в 2008 году, основная цель ее создания – это коммерциализация проекта Hadoop. Компания выпустила коммерческую версию Hadoop. Дистрибутив Hadoop от Cloudera первоначально создавался при участии разработчика

Hadoop Дуга Каттинга. Компания также оказывает услуги по облачной обработке данных при помощи технологии Hadoop [117].

Сегодняшний флагманский продукт Cloudera Enterprise Data Hub – это комплексная платформа управления данными, включающая основные отдельно поставляемые продукты: Data Science & Engineering, Operational DB, Analytic DB и Cloudera Essentials:

- Cloudera Analytic DB – инструменты бизнес-аналитики от Cloudera, основанные на базовой платформе Cloudera Essentials;
- Cloudera Operational DB – это высоко масштабируемые технологии NoSQL от Cloudera для приложений обработки данных в реальном времени, построенные на базовой платформе Cloudera Essentials;
- Cloudera Data Science and Engineering – технологии Cloudera, которые обеспечивают эффективную и высоко масштабируемую обработку данных, анализ данных и машинное обучение на основе базовой платформы Cloudera Essentials;
- Cloudera Essentials – основная платформа управления данными Cloudera для быстрой, простой и безопасной высоко масштабируемой обработки данных, которая включает в себя средства масштаба предприятия для управления кластером, развёртывания Hadoop, его администрирования и оптимизации производительности (Cloudera Manager), а также дистрибутив платформы с открытым исходным кодом (CDH);
- CDH (Дистрибутив Cloudera, включающий Apache Hadoop) – это дистрибутив платформы Cloudera с 100% открытым исходным кодом, в который входит Apache Hadoop, Apache Spark, Apache Impala, Apache Kudu, Apache HBase и многое другое.

В состав CDH входит несколько продуктов, которые на первых этапах развивались компанией Cloudera, а потом были переданы под лицензию Apache, в том числе:

- Apache Impala – массово-параллельный механизм интерактивного выполнения запросов на языке SQL к данным, хранимым в HDFS, Apache HBase и Apache Kudu. В отличие от Hive, обеспечивающего трансляцию SQL-запросов в задания MapReduce, выполняемые в пакетном режиме, Impala выполняет запросы в распределённой среде интерактивно, распределяя запрос по узлам обработки на основе собственного механизма, не прибегая к MapReduce;
- Apache Kudu – табличная NoSQL СУБД, обеспечивающая высокую производительность и масштабируемость.

В дополнение к своему дистрибутиву CDH с открытым исходным кодом Cloudera предлагает следующие продукты:

- Cloudera Director - инструмент, распространяемый бесплатно, который позволяет легко развертывать облачные кластеры Cloudera по запросу у нескольких облачных провайдеров;
- Cloudera Data Science Workbench – инструмент анализа данных для безопасного сотрудничества и надстройка для разработки моделей в Cloudera Data Science and Engineering и Cloudera Enterprise Data Hub;
- Cloudera Navigator – выполняет критически важные функции управления данными для платформы Cloudera, предлагая такие возможности, как дата-майнинг, аудит,

наследование, управление метаданными, шифрование, управление ключами шифрования и соблюдение политик, чтобы помочь удовлетворить нормативные требования;

- Cloudera Navigator Optimizer – инструмент SaaS, помогающий идентифицировать, переносить и настраивать традиционные рабочие нагрузки баз данных на платформу Cloudera, а также анализировать и настраивать рабочие нагрузки, выполняемые на платформе Cloudera;
- Cloudera Manager – инструмент администрирования для быстрого, простого и безопасного развертывания, мониторинга, оповещения и управления платформой Cloudera;
- Altus Data Engineering – облачное предложение от Cloudera для инжиниринга данных.

Cloudera тесно сотрудничает с Intel. Компания Intel владеет акциями Cloudera и несколько раз инвестировал в нее деньги, увеличивая свою долю в этой компании. В марте 2017 г Cloudera стала публичной компанией, выведя свои акции на биржу. В июне 2019 г. компания заключила партнёрское соглашение с IBM. Cloudera осталась единственным независимым игроком на рынке дистрибутивов Hadoop играет одну из ведущих ролей среди компаний, развивающих экосистему Apache Hadoop.

8.3. MapR Convergent Data Platform (MapRCDP)

MapRCDP разрабатывалась и поддерживалась компанией MapR Technologies со штабквартирой в Сан-Хосе, штат Калифорния. Converged Data Platform включает собственный дистрибутив Hadoop, распределенную файловую систему, систему управления NoSQL базами данных, набор инструментов для потоковой обработки данных и другое сопутствующее программное обеспечение [118]. Важнейшей особенностью дистрибутива Hadoop от MapR является файловая система MapR (MapR-FS), заменяющая файловую систему HDFS Apache Hadoop.

MapR-FS – это полноценная файловая система, обеспечивающая высокую скорость обмена данными, что необходимо для работы в реальном времени, а также имеющая дополнительные интерфейсы для NFS и POSIX. Это позволяет клиентам различных файловых систем использовать MapR-FS. Ещё одним отличием от HDFS является возможность физически перезаписывать данные в одно и то же место, обеспечивая при этом, так же как HDFS, наличие нескольких копий данных для защиты от сбоев. Это позволяет избежать многократного увеличения требований к памяти при перемещении и дублировании данных.

Так же как HDFS, MapR-FS позволяет объединять несколько физических дисков в один том хранилища. Это даёт возможность осуществлять геораспределение данных и комбинирование различных типов дисков в единую систему, в том числе, SSD и вращающихся дисков. В результате можно организовать хранение часто используемых данных во флэш-памяти, а хранение архивных исторических данных – на более дешевых, вращающихся носителях.

Дистрибутив MapR используется во многих организациях в сфере финансов, розничной торговли, медиа, здравоохранения, производства, телекоммуникаций, в правительственных организациях и в компаниях, которые возглавляют рейтинги Fortune 100, а также в крупнейших ИТ-компаниях, среди которых Amazon, Cisco, Google, Microsoft, SAP.

В 2019 году MapR Technologies была поглощена корпорацией Hewlett Packard Enterprise (HPE), включая продукт MapRCDP и другую интеллектуальную собственность в областях искусственного интеллекта, машинного обучения, управления данными и аналитики.

8.4. Pivotal HD

Pivotal HD – дистрибутив Hadoop и смежных приложений был создан и развивался компанией Pivotal Software, сформированной в 2012 году путём выделения из EMC Corporation и VMware, большая часть акций которой также принадлежала EMC. Название было получено от компании Pivotal Labs LLC, ранее приобретенной EMC. Кроме Pivotal HD компания поставляла различные категории продуктов [119]:

- Cloud Foundry – инструментарий для разработки облачных приложений;
- Greenplum – SQL СУБД для использования на кластерах;
- RabbitMQ – система управления очередями сообщений;
- Spring Framework – инструментарий для разработки Java-приложений, предназначенный для замены или используемый в качестве дополнения к Enterprise JavaBeans (EJB).

Отличительной чертой Pivotal HD является его интеграция с Greenplum и приспособленность для разработки аналитических приложений.

14 августа 2019 г. VMware объявила о дискуссии по слиянию с Pivotal, а окончательное соглашение о приобретении Pivotal Software было подписано 22 августа 2019 г. Слияние завершено 30 декабря 2019 г. VMware включила продукты Pivotal в пакет приложений Tanzu, а в январе 2021 года консалтинговая группа Pivotal Labs сменила бренд на VMware Tanzu Labs.

9. Второе поколение платформ Big Data: интегрированные платформы

9.1. Возникновение и функциональность интегрированных платформ Big Data

В отличие от рассмотренных в предыдущем разделе платформ, которые объединяют несколько разных взаимосвязанных продуктов, функционирующих на одном дистрибутиве Hadoop, на рубеже нулевых и десятых годов возник новый класс программных продуктов Integrated Big Data platform, куда входят программные продукты, ориентированные на обработку больших данных и включающие несколько разнородных инструментов. Термин (Integrated) Big Data platform появился в печати спустя 1–2 года после появления первых продуктов этого класса [120], [121]. Строгое и прямое определение этого термина до сих пор отсутствовало; но поставщики, консультанты и аналитики рынка интуитивно понимают это и широко используют термин [122], [123], [124] для обозначения программных продуктов, предназначенных для работы с большими данными и включающих несколько инструментов различных типов. Некоторые из этих продуктов рассматриваются в обзоре Gartner [125].

Интегрированные платформы появились как результат предшествующего развития большого числа пакетов в рамках экосистемы Hadoop. Платформы устраняли недостатки HDFS, обеспечивали более эффективное использование оперативной памяти, а также объединяли разнородные приложения, обеспечив интеграцию между ними.

Как таковая, интегрированная платформа представляет собой единый пакет, в котором совмещено несколько возможностей – СУБД, работа с потоками данных, несколько инструментов анализа данных, комплексная обработка графов. Наиболее популярная платформа с открытым кодом – Spark, на ней работает примерно половина аналитиков в мире. Пример мощной дорогой лицензируемой платформы – SAP HANA (High-Performance Analytic Appliance). Она ориентирована на работу в оперативной памяти и в ней помимо типовых возможностей есть ещё текстовый процессор и процессор координатных данных. Приложения, построенные на основе платформы, используют только те возможности, которые им нужны. Как правило, в компании на одной платформе могут создаваться и эксплуатироваться десятки приложений.

Специфические бизнес-приложения, такие как различные ERP-, CRM- и SCM-системы, или системы моделирования аэро- и гидродинамики, САПР, системы моделирования электрических сетей, системы моделирования нефтегазоносных пластов и др., не включаются в состав интегрированных платформ, а развиваются как слой приложений над ними. Пример взаимодействия слоя приложений и интегрированной платформы Big Data приведен на рисунке 7.

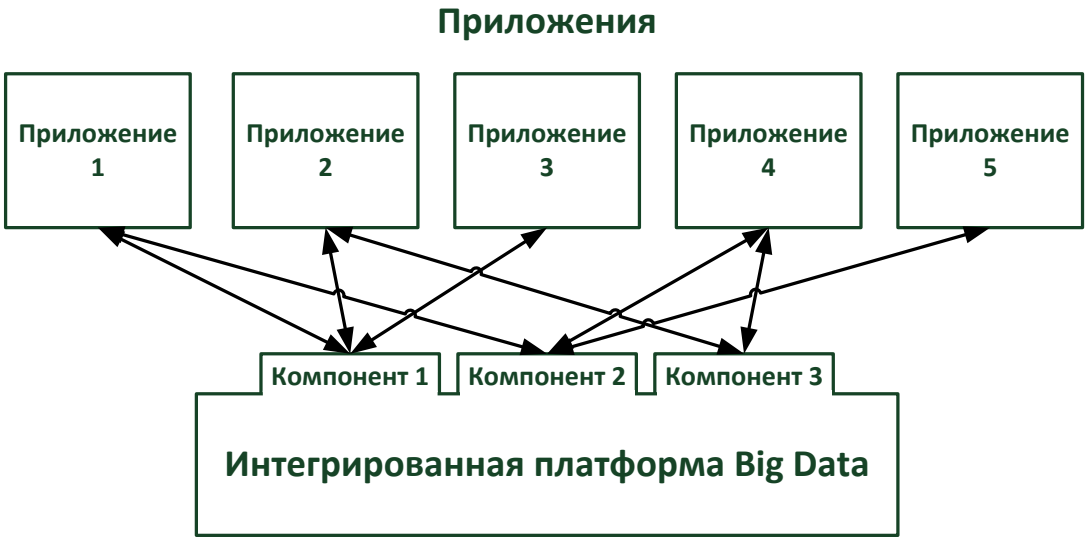


Рис. 7. Пример взаимодействия слоя приложений и интегрированной платформы Big Data

Отдельные компоненты интегрированных платформ могут уступать по функциональности аналогичным независимым приложениям. Например, GraphLab наиболее эффективно оптимизирует обмен данными между узлами кластера при выполнении сложных алгоритмов над большими графами, распределёнными на много узлов кластера [126], а Apache Mahout "Samsara" имеет внутренний язык для проектирования алгоритмов Machine Learning [127]. Тем не менее, интегрированные платформы выигрывают у независимых пакетов по массовости применения. В большинстве случаев интеграция из коробки с другими компонентами и большая распространённость продукта перевешивают наличие специфической функциональности, для использования которой потребуются дополнительные усилия по интеграции.

В период интенсивного развития платформ второго поколения это были продукты, на развитии которых были сосредоточены самые большие усилия сообщества разработчиков. Spark, как наиболее зрелый и раньше всех начавший развиваться Open Source проект этого класса

являлся самым активным Open Source проектом в области Big Data [128, 129]. Для него ежегодно выполнялось самое большое число разработок. Такая же ситуация наблюдается и для проприетарных платформ. Хотя компании разработчики не предоставляют соответствующих данных, это косвенно подтверждается высокой частотой обновления продуктов и маркетинговой активностью вокруг SAP HANA, Teradata Vantage, Databricks Lakehouse Platform и др.

Несмотря на т, что на смену платформам второго поколения пришли платформы третьего и на горизонте уже просматриваются очертания платформ четвёртого поколения, развитие интегрированных платформ в их традиционном виде не прекратилось. Они стали важнейшими сервисами в рамках облачных платформ

В следующих пунктах данного раздела будет более рассмотрens примеры интегрированных платформ Big Data,

9.2. Apache Spark

Spark – проект зародился в лаборатории Калифорнийского университета в Беркли (UC Berkeley) примерно в 2009 г. [130, 131]. Основатели проекта – известные ученые из области баз данных, и по философии своей Spark в каком-то роде ответ на MapReduce. Несмотря на то, что проект перешёл под «крышу» Apache, идеологи и основные разработчики остались теми же.

Spark не предлагает свою собственную среду для хранения данных, а функционирует поверх других (HDFS, HBase, JDBC, Cassandra, …). Очень часто ставится поверх Hadoop, из которого использует только HDFS.

Основным представлением данных в Spark на первом этапе стал устойчивый распределенный набор данных (Resilient Distributed Dataset, RDD) – абстракция для распределенной памяти. Такой набор данных представляет из себя неизменяемую коллекцию, которая может быть преобразована в другую коллекцию с помощью групповых операций, например отображения (map). Spark поддерживает историю всех преобразований и для каждого набора данных знает, как он был создан. Таким образом, он может пересоздать его в случае необходимости, что делает такой набор данных устойчивым к сбоям. Также это позволяет вычислениям выполняться только тогда, когда потребуются результаты для выдачи. Такой «ленивый» режим работы позволяет комбинировать вычисления и выполнять их партиями, а также вообще не выполнять то, что так и не потребовалось для конечного результата. Устойчивые распределенные наборы данных образуют направленный ациклический граф, пример которого представлен на рисунке 8.

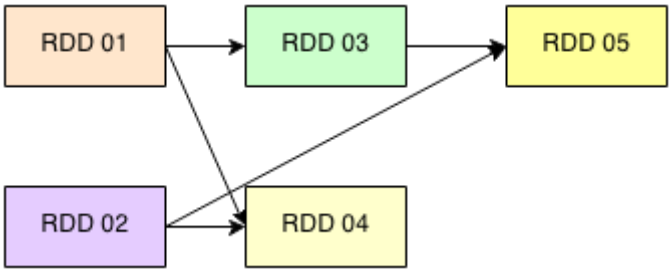


Рис. 8. Пример схемы порождения RDD

Spark позволяет пользователям указывать, как хранить данные. Можно специфицировать что будет храниться в памяти, а что будет кэшироваться. Возможность хранения данных в памяти предоставляет большие преимущества для части вычислительных задач и является одной из сильных сторон платформы Spark. Hadoop MapReduce, например, не предоставляет такой опции. Если нужно сохранить промежуточные результаты в Hadoop MapReduce, то их нужно записывать в файловую систему, что может быть очень затратным. Для задач, которые требуют значительного использования промежуточных данных, Spark работает многократно (вплоть до ста раз) быстрее Hadoop MapReduce. Spark предоставляет пользователям осуществлять гибкую настройку использования памяти, что даёт возможность оптимально использовать имеющиеся ресурсы. Spark также позволяет контролировать распределение данных по узлам. Можно указать по какому полю группировать данные. Spark старается оптимизировать распределение вычислений и производить их рядом с данными.

Контроль за размещением данных в памяти помогает не только более эффективно проводить вычисления с повторным использованием данных, но и открывает возможность интерактивной работы с данными. Можно корректировать и уточнять свои запросы и при этом не нужно начинать все вычисления с нуля. Быстрая интерактивная работа с данными открывает широкие возможности для бизнес-аналитики и дата-майнинга в оперативном режиме.

Позднее в Spark появились и другие представления данных кроме RDD:

RDD (Spark1.0) —> Dataframe (Spark1.3) —> Dataset (Spark1.6)

Их соответствие показано на рисунке 9.

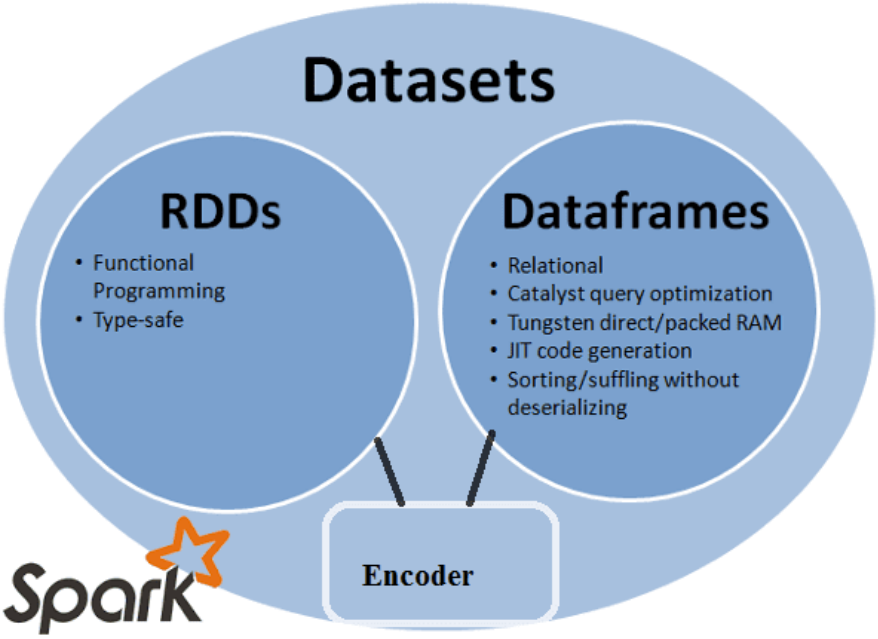


Рис. 9. Формы представления данных в Spark

- RDD – это распределенная коллекция данных, расположенных по нескольким узлам кластера, набор объектов Java или Scala, представляющих данные. RDD работает со структурированными и с неструктурированные данными. Также, как DataFrame и

DataSet, RDD не выводит схему загруженных данных и требует от пользователя ее указания.

- DataFrame – это распределенная коллекция данных в виде именованных столбцов, аналогично таблице в реляционной базе данных. DataFrame работает только со структурированными и полуструктурированными данными, организуя информацию по столбцам, как в реляционных таблицах. Это позволяет Spark управлять схемой данных.
- DataSet – это расширение API DataFrame, обеспечивающее функциональность объектно-ориентированного RDD-API (строгая типизация, лямбда-функции), производительность оптимизатора запросов Catalyst и механизм хранения вне кучи. DataSet эффективно обрабатывает структурированные и неструктурированные данные, представляя их в виде строки JVM-объектов или коллекции. Для представления табличных форм используется кодировщик (encoder).

Spark предоставляет интерфейсы для работы с RDD, Dataframe и Dataset на Java, Python, Scala и R. Со всеми объектами можно взаимодействовать с помощью вызова методов. Методы могут возвращать новые объекты или осуществлять операции, например, подсчет количества элементов. Изначально Spark был написан на Scala, впоследствии добавлена существенная часть кода на Java для предоставления возможности написания программ непосредственно на Java.

Пользователь контролирует как хранятся данные и может, например, указать приоритет сохранения в памяти каждого массива данных. Это позволяет использовать всю имеющуюся память для самых часто используемых данных, а остальное закешировать. Обеспечивая эффективное использование оперативной памяти и одновременно работу с внешними носителями данных Spark, по сути, реализует преимущества In-Memory систем, связанные с высокой скоростью обработки данных. И одновременно, оставляя малоиспользуемые данные во внешней памяти, он не имеет ограничений In-Memory систем, связанных с объёмом оперативной памяти.

Spark состоит из ядра и нескольких расширений:

- Spark SQL, цель которого предоставить SQL подобную функциональность на базе Spark. Пользователь может получить быстрые ответы на свои вопросы, составив нужные запросы. Также можно встраивать Spark SQL запросы в код приложений. Spark SQL интегрирован с Hive, JDBC и ODBC.
- Spark Streaming для обработки потоков. Предоставляет разработчикам высокоуровневое API на Java и Scala для того, чтобы легко писать устойчивые к сбоям и быстрые приложения для обработки потоков. Интеграция с другими проектами Spark позволяет решать проблемы, требующие комбинации различных подходов.
- Spark GraphX для осуществления вычисления с графами. Утверждается, что он имеет производительность наравне со специализированными программными продуктами.
- Spark MLBase для машинного обучения. Это, по сути, замещение Apache Mahaout, только намного серьезнее. Помимо эффективного параллельного машинного обучения (не только средствами RDD, но и дополнительными примитивами) SparkML еще намного качественнее работает с локальными данными, используя пакет нативной линейной алгебры Breeze, написанный на Фортране.

Как и Hadoop, Spark это не просто отдельный проект, а целая экосистема основанных на нем разнообразных проектов

9.3. SAP HANA

В 2011 году компания SAP вывела на рынок In-Memory платформу SAP HANA. В составе HANA объединены объектно-графическое, построчное и постолбцовое хранилища данных, а также несколько серверов приложений и их окружение [132, 133], в том числе:

- сервер выполнения SQL-запросов;
- планировщик и оптимизатор вычислений;
- сервер обработки текстовых данных;
- сервер работы с графами;
- сервер предиктивной аналитики;
- средства сжатия данных;
- серверные компоненты (среда исполнения) для языков программирования;
- библиотеки встроенных бизнес-функций;

и т.д.

Общая архитектура SAP HANA была представлена в [134] в виде, показанном на рисунке 10.

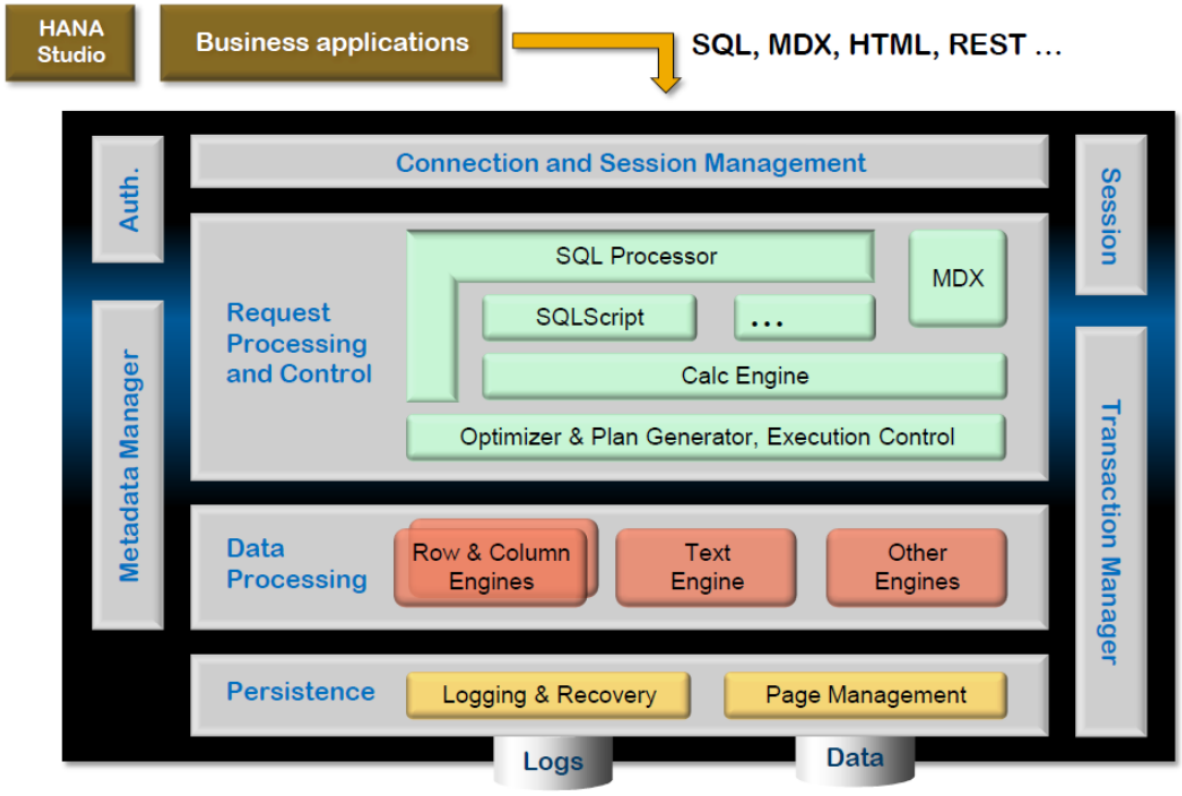


Рис. 10. Общая архитектура SAP HANA

Ключевой особенностью SAP HANA является выполнение всех операций в оперативной памяти. Дисковая память в основном используется для протоколирования всех операций и поддержки актуальных резервных копий данных. В результате переноса всех процессов в оперативную память обеспечивается высокая скорость обработки данных. Объединение хранилищ данных и серверов приложений в рамках единой платформы позволяет перейти к

двухуровневой архитектуре приложений вместо традиционной трёхуровневой (клиент – сервер приложений – сервер баз данных). Это также позволяет повысить скорость работы приложений за счёт устранения узкого бутылочного горлышка, которым являлся интерфейс между сервером приложений и сервером баз данных.

Высокая степень параллелизма обработки табличных данных при выполнении SQL-запросов обеспечивается в SAP HANA техническими решениями по распределению процессов обработки таблиц данных, их отдельных столбцов и фрагментов столбцов между разными серверами, процессорами и процессорными ядрами аппаратной среды [135]. Эти решения показаны на рисунках 11 и 12.

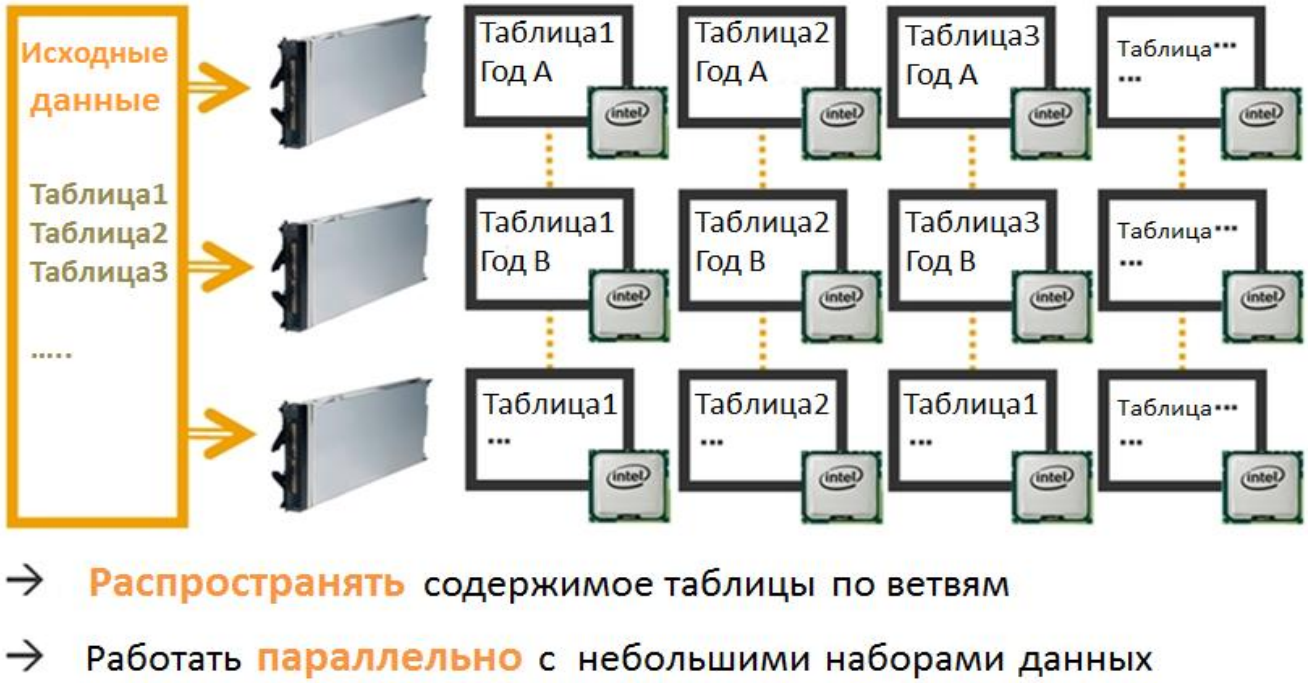


Рис. 11. Распределение обработки таблиц между серверами и процессорами по группам строк

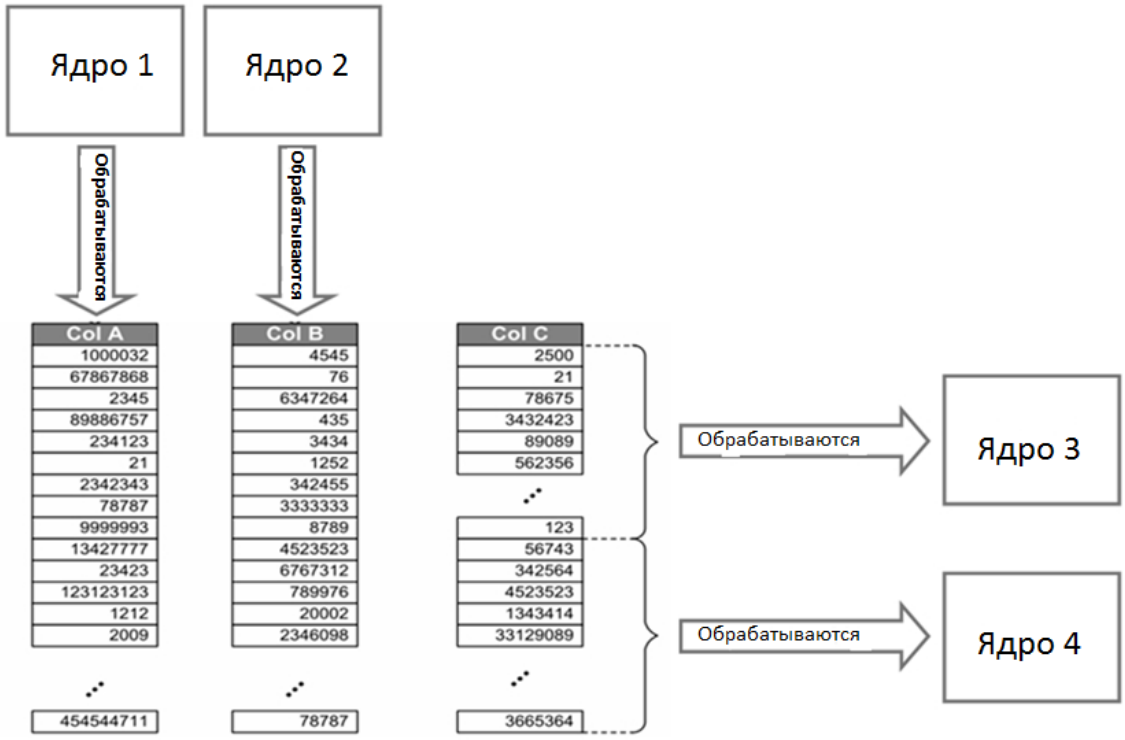


Рис. 12. Распределение обработки столбцов одной таблицы между процессорными ядрами

Максимальные аппаратные конфигурации кластеров, которые используются для работы SAP HANA, по состоянию на 2023 год насчитывали более 28 серверов, содержащих до 8 процессоров и общую память 22 TB памяти NVMe [136, 137]. Каждый серверный процессор Intel сейчас может содержать до 24 ядер. В результате, суммарно, перспективные максимальные конфигурации SAP HANA могут иметь более 5 000 процессорных ядер и петабайты памяти NVMe. В связи с высокой стоимостью таких конфигураций на практике в подавляющем большинстве случаев применяются существенно менее мощные аппаратные комплексы. Чтобы иметь возможность обрабатывать большие объёмы данных в последних версиях платформы добавлена возможность прозрачного вытеснения редко используемых данных в дисковую память и подкачки их оттуда в случае необходимости.

Распределение обработки данных между отдельными процессорами и процессорными ядрами и превышение условного ограничения в 1 PB обрабатываемых данных однозначно свидетельствуют о том, что платформа SAP HANA – это инструмент Big Data. Выполнение всех процессов в оперативной памяти делает SAP HANA идеальным средством обработки данных в реальном времени.

В качестве одного из продуктов, входящих в экосистему SAP, можно представить SAP Sybase Event Stream Processor, обобщённая архитектура которого представлена на рисунке 13 [138].

Рис. 13. Обобщённая архитектура SAP Sybase Event Stream Processing

В 2015 году появилась функционально похожая на SAP HANA платформа SAP Vora на базе Spark. Первоначально она называлась SAP HANA Vora, но впоследствии стала позиционироваться как самостоятельный продукт. Позже в экосистеме SAP HANA появился SAP Data Hub, который работает на базе SAP Cloud Platform и является, скорее, интеграционным решением.

9.4. Переход в интегрированные платформы

Случается, что интегрированными платформами становятся программные продукты, которые первоначально таковыми не являлись. Они как бы вырастают до платформ. Например, калифорнийская компания с российскими корнями GridGain Systems в 2016 году продала лицензии на свой продукт GridGain In-Memory Data Fabric Сбербанку РФ. В профессиональной среде распространился слух, что это конкурент SAP HANA. Однако продукт GridGain, в тот момент хотя и осуществлял обработку больших массивов данных в оперативной памяти, СУБД не являлся [139]. Это ПО логически и архитектурно находилось в слое над БД и под приложением. Цель продукта была - обеспечить более высокую производительность и масштабируемость приложений в сравнении с системами, основанными на дисковом хранении данных. По сути, речь шла о своеобразном кэше в ОЗУ, куда помещаются все активно используемые данные из самых разных источников, включая реляционные, NoSQL, Hadoop, потоковую информацию и т.д.

Исходный код Ignite был открыт компанией GridGain Systems в конце 2014 года и в том же году принят в программу Apache Incubator. В сентябре 2015 г. проект Ignite стал полноправным проектом Apache Software Foundation. Бесплатно предоставляется только код базовой версии Ignite, за дополнительные функции GridGain In-Memory Data Fabric, средства управления и техподдержку заказчику необходимо платить.

Основная особенность технологии Ignite, которая отличала её первые версии от обычного кэширования, состояла в том, что этот кэш, процессоры для его обработки и сопутствующие

инструменты реализованы не на отдельном сервере, а на кластере из серверов стандартной архитектуры, и он обладает большой вертикальной и горизонтальной масштабируемостью.

На рисунке 14 представлена архитектура корпоративной системы с использованием первых версий GridGain In-Memory Data Fabric. На представленной архитектуре IGFS – это Ignite File System, а по сути, GridGain In-Memory Data Fabric.

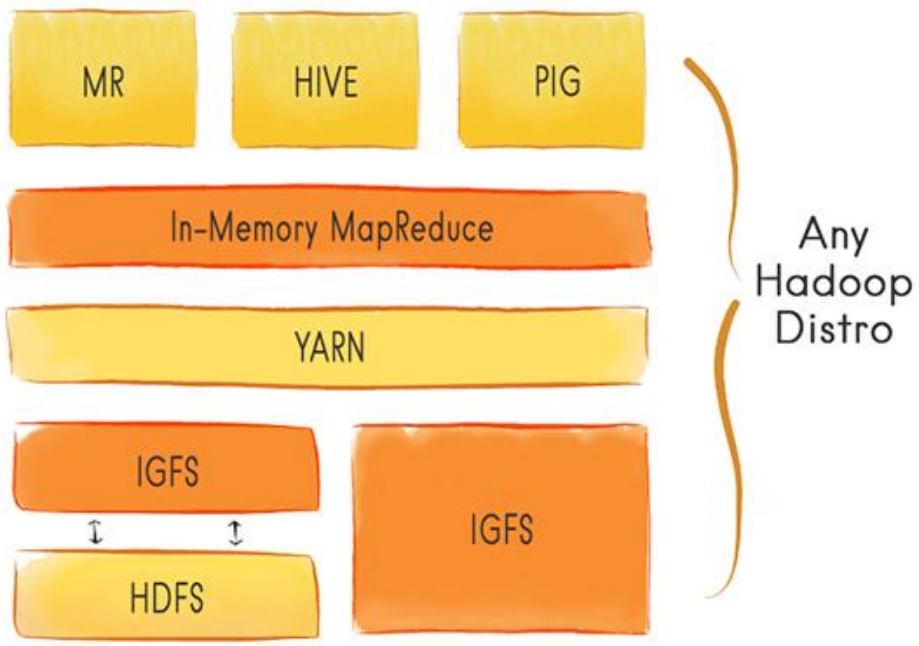


Рис. 14. Корпоративная архитектура с использованием мпевыхх версий Apache Ignitec

При последующем развитии в Apache Ignite были добавлены возможности работы с SQL-запросами к распределённо хранимым кэшируемым данным, средства Machine Learning (как свои собственные модули, так и интеграция с Google TensorFlow) и средства выполнения SQL-запросов над потоками данных. Продукт постепенно дорос до платформы, позволяющей решать разнородные задачи в интересах широкого круга приложений.

9.5. Псевдоплатформы

Часто интегрированными платформами называют программные продукты, которые таковыми не являются. Например, проектом и продуктом, о котором шли разговоры, что он может стать платформой, является Apache Beam. Первый релиз появился в 2016 году в момент острой конкуренции между Spark и Flink. Spark Streaming в это время работал только с использованием microbatch'ей и сильно уступал настоящим системам потоковой обработки данных. Flink очень хорошо обрабатывал потоковые данные, но у него было существенно меньше возможностей и что-то не ладилось с пакетной обработкой. В этот момент появился Apache Beam, реализующая унифицированную модель программирования с открытым исходным кодом для определения и выполнения конвейеров обработки данных, включая ETL, пакетную и потоковую обработку. Однако появившимся надеждам не суждено было сбыться, Apache Beam не вырос в платформу, а так и остался фреймворком, позволяющим на основе единого подхода реализовать потоковую и пакетную обработку данных.

9.6. Вторичные интегрированные платформы

В отличие от SAP HANA, ориентированной на In-Memory обработку данных, SAP Vora на реализованная на базе Apache Spark, может хранить и обрабатывать сотни петабайтов данных. В SAP Vora перенесено большинство case-средств и упрощающих разработку интерфейсов из SAP HANA, но в основе лежит ядро от Apache Spark. Архитектура SAP Vora представлена на рисунке 15 [140].

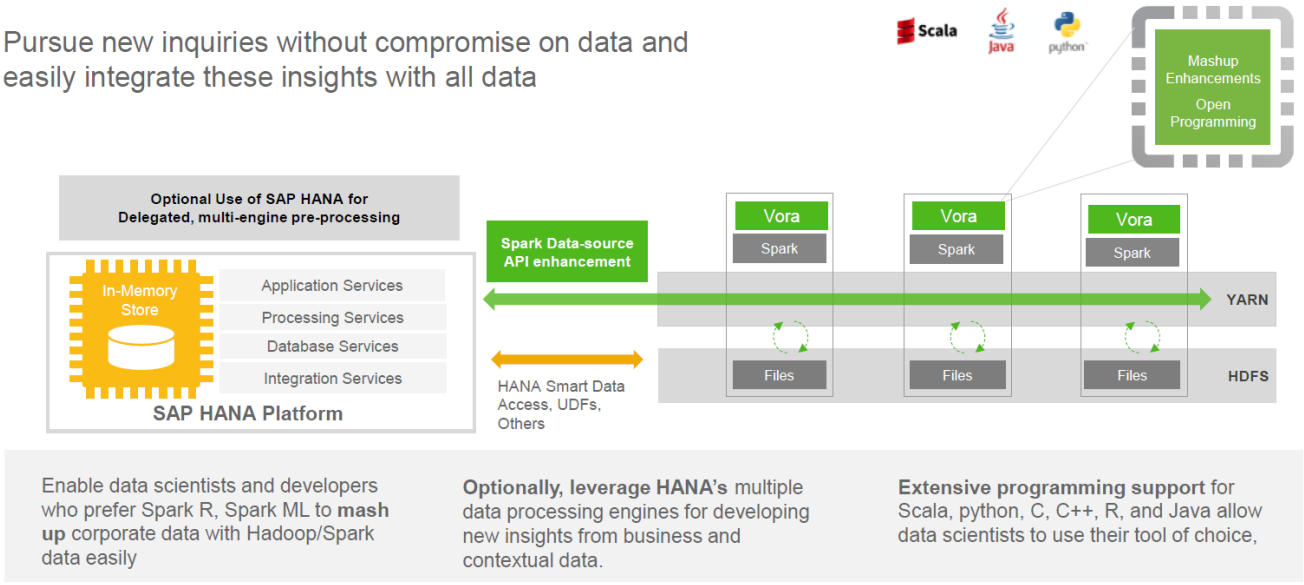


Рис. 15. Обобщённая архитектура SAP Vora

По сути, SAP Vora является вторичной интегрированной платформой, построенной с использованием механизмов первичной платформы Apache Spark.

9.7. Функциональность интегрированных платформ

Функциональность некоторых популярных интегрированных платформ по состоянию на февраль 2021 года представлена в таблице 2.

Таблица 2. Функциональность интегрированных платформ Big Data

Components \ Integrated Platforms	SAP HANA	Teradata Vantage	Apache Spark	Apache Flink	Apache Ignite	Apache Apex
SQL DBMS	+	+	+	+	+	
NoSQL DBMS		+				
Graph DBMS	+					
Object Storage	+	+				
Graph Engine	+		+	+		
Search Engine	+					
Crawler	+					
Master Data Management Tools	+					
Streaming Engine	+		+	+		+
Event Processing Tools				+		
Streaming Analytics	+				+	
Machine Learning Library	+	+	+	+	+	
Machine Learning Automation Tools	+	+				

ETL Tools	+					
R Engine	+	+	+			
Spatial Engine	+	+				
Business Analytics	+	+				
Social Networks Analytics	+	+				
Rapid Application Development Tools	+					+
Business Rules Engine	+					
Web Server for Data	+					
Data Modeling Tools	+	+				
Integration with Mobile Devices	+					
IoT Tools	+					
Security Tools	+	+	+	+	+	+
Natural Language Services	+	+				

Показанная в таблице существенно меньшая функциональность платформ Open Source во многом является обманчивой. Просто в разработчики проприетарных платформ заинтересованы сразу продемонстрировать всем максимально широкую функциональность. В то же время вокруг платформ Open Source существуют большие экосистемы из множества независимых разработчиков. В экосистемы входят независимые компании со своими продуктами. Их очень сложно свести в какие-то общие таблицы, но зачастую в таких экосистемах можно найти намного больше продуктов разной функциональности, чем поставляет один проприетарный вендор.

10. Третье поколение платформ Big Data: облачные платформы

Общая функциональность облачных платформ: облачные платформы позволяют выделить пользователям через интернет любое необходимое количество серверов и развернуть на них нужные приложения. Кроме того, меню таких платформ, как правило включает больше сотни разных сервисов. Среди них есть платформенные – различные СУБД, инструменты для анализа данных и развёртывания блокчейн, в том числе интегрированные платформы второго поколения, а также различные приложения. Наиболее известные облачные платформы: Amazon EMC, Microsoft Azure, Google Cloud Platform, SAP Hana Enterprise Cloud.

Большинство проприетарных интегрированных платформ предоставляют провайдеры облачных услуг. В число предоставляемых ими сервисов всегда входят интегрированные платформы второго поколения из семейства Apache. Несколько позже к числу вендоров облачных платформ подключились поставщики комплексов тяжелых приложений – SAP и Oracle. Они предлагают в составе облачных сервисов свои платформы второго поколения (SAP HANA Enterprise Cloud). Одновременно с платформами второго поколения все поставщики облачных платформ они предлагают (в их составе или отдельно) дополнительные возможности для разработки приложений с использованием облачных ресурсов.

В качестве иллюстрации многочисленности и разнообразия сервисов, предлагаемых в составе облачных платформ, в следующем разделе рассмотрим одну из них подробнее.

10.1. Платформа IBM Bluemix

Компания IBM объединила все свои технологические решения по работе с большими данными в составе облачной платформы IBM Bluemix [141]. Платформа является инструментом

для разработки веб- и мобильных приложений для работы с Big Data. Обеспечивается работа с данными в Hadoop, в SQL и NoSQL базах данных IBM, а также в базах данных других производителей, ориентированных на работу с большими данными, такими как MongoLab, ElephantSQL, Redis Cloud и др. В состав платформы входят:

- средства хранения и обработки данных: Hadoop, средства построения хранилищ данных, средства управления данными, средства управления контентом, средства поточной обработки данных, анализа данных социальных сетей;
- развитые средства аналитики для принятия решений, средства построения отчётов, средства контентной аналитики, аналитики по данным геолокации, средства предиктивной аналитики и датамайнинга;
- средства разработки мобильных приложений (SDK для iOS и Android, плюс набор облачных сервисов);
- средства управления разработкой и развертыванием ПО (DevOps - планирование и контроль процесса разработки и развертывания приложений);
- средства построения гибридных облаков за счёт управления API и интеграции облачных приложений;
- средства информационной безопасности (AppScan, выявление уязвимостей, анализ кода);
- средства разработки приложений для интернета вещей (сервисы взаимодействия приложений с IoT-устройствами).

Одной из ключевых особенностей платформы Bluemix является тесная интеграция с сервисами IBM Watson для взаимодействия с пользователями и обработки данных на естественном языке.

На базе платформы Bluemix IBM ведёт разработку и поставляет готовые решения для работы с большими данными:

- крос-функциональные для организации продаж, маркетинга, обработки финансовых данных, управления рисками, управления операциями, управления ИТ, обеспечения защиты от мошенничества, обработки данных персонала;
- отраслевые для различных отраслей: медицины, страхования и т.д.

Поддерживаются следующие языки программирования для разработки приложений: Java, Java Script, Go, PHP, Python, Ruby и ряд других языков. Программные решения развернуты в облаке IBM, готовы к работе и доступны через открытые интерфейсы. Основные компоненты IBM Bluemix представлены в таблице 3.

Таблица 3. Состав сервисов IBM Bluemix

Группа	Список сервисов
Watson (когнитивная аналитика): 23. Толкование понятий 24. Идентификация языка 25. Машинный перевод 26. Экспертные системы 27. Выявление отношений 28. Распознавание визуальных образов	Concept Expansion, Concept Insights, Language Identification, Machine Translation, Personality Insights, Question and Answer, Relationship Extraction, Speech To Text, Text to Speech, Tradeoff Analytics, Visual Recognition, Cognitive Commerce™, Cognitive Graph, Cognitive Insights™

Mobile (мобильные приложения): 29. Аналитиз работы приложения 30. Безопасность мобильных приложений 31. Мобильные данные 32. Контроль качества приложений 33. Отслеживание присутствия пользователя 34. Push-уведомления	Advanced Mobile Access, Mobile Application Security, Mobile Data, Mobile Quality Assurance, Presence Insights, Push, Push iOS 8, Twilio
Data Management (управление данными): 35. Реляционные данные (SQL) 36. NoSQL данные 37. Облачные хранилища 38. Гео-данные	Cloudant NoSQL DB, DataWorks, Object Storage, SQL Database, ClearDB MySQL Database, ElephantSQL, MongoLab, Redis Cloud
DevOps (разработка и выпуск ПО): 39. Авто-масштабирование ресурсов 40. Автоматизация развертывания ПО 41. Мониторинг и аналитика 42. Планирование процесса выпуска и контроль 43. Контроль производительности работы приложения	Auto-Scaling, Delivery Pipeline, Monitoring and Analytics, Track & Plan, BlazeMeter, Load Impact, New Relic
Business Analytics (бизнес-аналитика и предиктивное моделирование): 44. Встроенные отчеты (Cognos) 45. Предиктивное моделирование (SPSS)	Embeddable Reporting, Predictive Modeling, Cupenya Insights
Big Data (аналитика больших данных): 46. Аналитика Hadoop (BigInsights) 47. Аналитика гео-данных 48. Анализ данных Twitter	BigInsights for Hadoop, dashDB, Geospatial Analytics, IBM Analytics for Hadoop, Insights for Twitter, Time Series Database
Integration (интеграция приложений и систем): 49. Управление API 50. Интеграция облачных приложений 51. Шлюзы безопасности	API Management, Cloud Integration, Secure Gateway
Security (информационная безопасность): 52. Аутентификация пользователей 53. Безопасность приложений 54. Выявление уязвимостей 55. Динамический анализ кода 56. Анализ кода мобильных приложений 57. Статический анализ кода	Application Security Manager, AppScan Dynamic Analyzer, AppScan Mobile Analyzer, Mobile Analyzer for iOS, Single Sign On, Static Analyzer, aPersona Adaptive Security Manager
Web and Application (Web-приложения): 58. Работа с бизнес-правилами 59. Кэш данных 60. Брокер сообщений (MQ) 61. Кэш сессий 62. Управление бизнес-процессами (BPEL) 63. Балансировка нагрузки 64. Гео-кодирование объектов	Business Rules, Data Cache, MQ Light, Session Cache, Workflow, Workload Scheduler, Geocoding, Memcached Cloud, Reverse Geocoding, Travel Boundary Service, Validate Address
Internet of Things (интернет вещей): 65. Взаимодействие приложения с IoT-устройствами	Internet of Things, flowthings.io

11. Data Lake – основной способ организации больших данных

11.1. Понятие Data Lake

Термин Data Lake в основном характерен для организации данных на платформах Big Data. Если проектирование баз данных и хранилищ данных ведётся в ходе проектирования и разработки приложений, когда исходя из запросов, которые будут поступать, определяются все ключевые поля, атрибуты, иерархии, таблицы, многомерные кубы и т.д. В больших данных всё происходит иначе. Сначала появляется платформа Big Data и туда начинают складывать какие-то данные большого объёма. Очень быстро там появляются всё новые и новые наборы данных, хранения которых раньше никто и не предполагал. Примерно такую картину вы видите на своём жёстком диске: нашли что-то интересное в интернет, создали папку, положили туда. Постепенно количество папок нарастает, появляется большая вложенность, много файлов, к которым давно не обращались и т.д. В крупных компаниях, где работает много аналитиков, финансистов, менеджеров, на их дисках, файловых серверах и других ресурсах хранится большое множество различных табличных данных, текстовых документов, презентаций, фото и видео файлов и т.д. Всё это можно сложить в один Data Lake.

Принципиальное отличие Data Lake от более ранних способов организации данных состоит в том, что там содержатся данные для ответа на будущие запросы, содержание и структура которых пока не известны [142]. Это обусловлено тем, что в озере хранятся данные не только в исходной, полученной из источников форме, но часто и в промежуточных формах, порождаемых в процессе очистки, верификации и согласования данных, а также в окончательной форме, предназначенной для использования. Кроме того, в озере данных помимо структурированных данных могут содержаться неструктурированные данные – изображения, видео, записи сигналов и пр.

Термин Data Lake был предложен в октябре 2010 года Джеймсом Диксоном – основателем и бывшим техническим директором Pentaho. Диксон утверждал, что хранилища данных и Data Marts имеют несколько проблем: от ограничений по размеру до узких параметров исследования. Современный обзор архитектуры Data Lake представлен в [143].

В блогах и публикациях встречается много шуток по поводу термина Data Lake. Как только не называют: болото данных, трясина данных, лужа данных. «У нас очень маленькое озеро данных, не озеро, а лужа данных.» «Из-за отсутствия стратегии развития и из-за того, что никто не был назначен ответственным за развитие, наше озеро данных превратилось в болото данных.» Очень много аналогий с водной тематикой – переплыть, утонуть и т.д.: «Spark позволяет быстрее плыть по озеру данных.», «Мы отказались от перехода в публичное облако, объём данных вырос, и сильно увеличилось время ожидания ответа. В итоге наш ИТ-директор утонул в озере данных, его уволили.» Тем не менее, термин завоевал признание и широко используется. Пример шутливой графики, обыгрывающей «водную» природу Data Lake, представлен на рисунке 16.

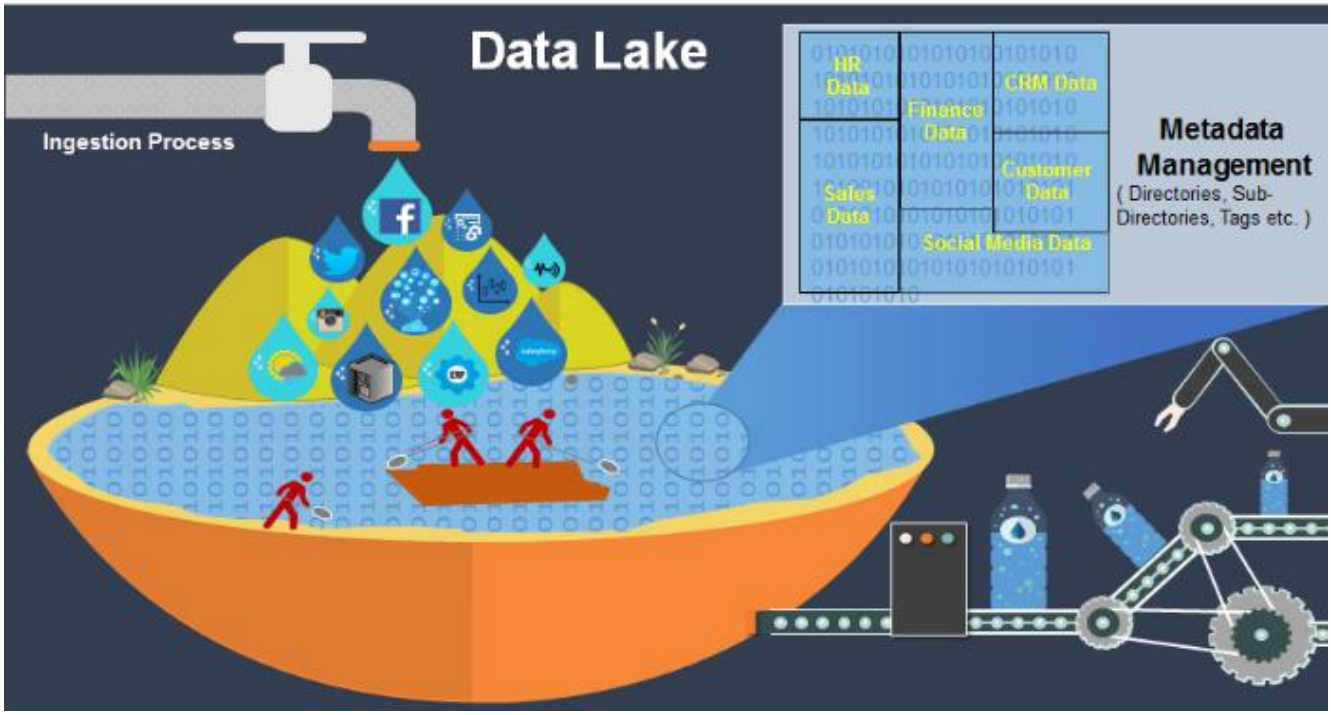
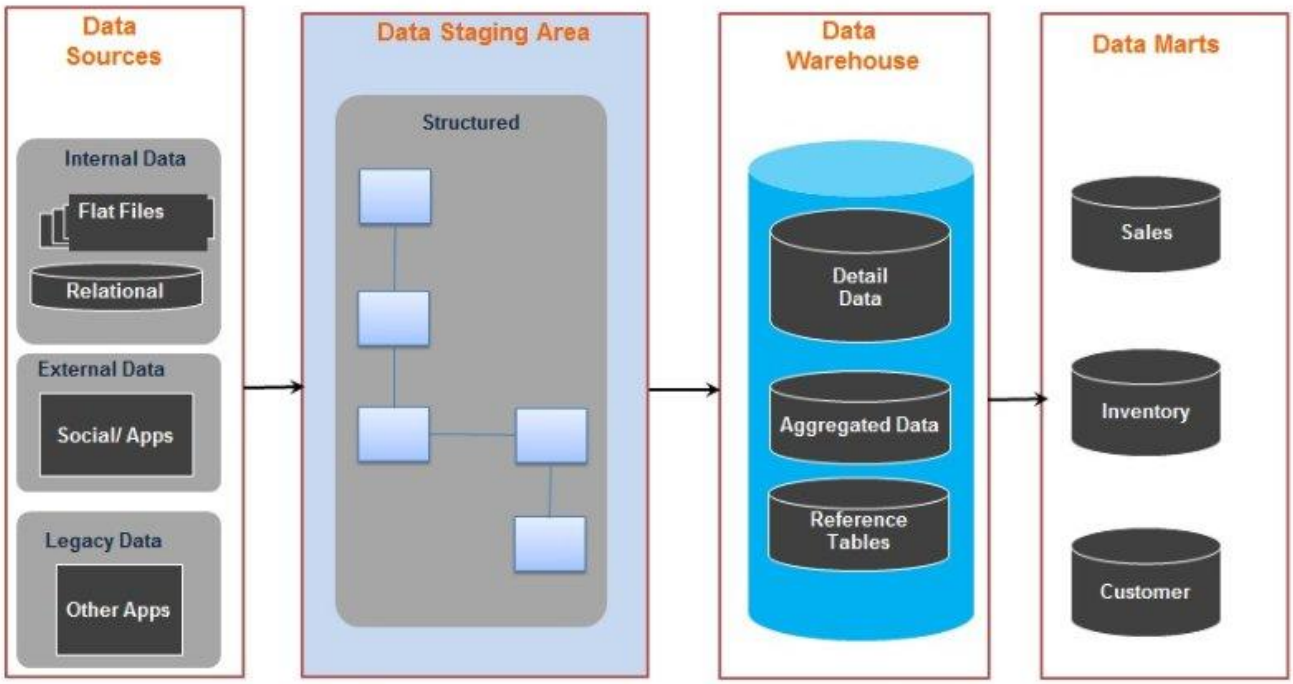


Рис. 16. Пример шутливой графики, обыгрывающей «водную» природу Data Lake

Поскольку основной платформой Big Data долгое время является Hadoop с его различными дистрибутивами и версиями, то организация Data Lake строится исходя из возможностей, имеющихся в HDFS – плоское оглавление файлов, каждый из которых имеет уникальный идентификатор и набор тегов метаданных [103]. Плоское оглавление с набором тегов метаданных позволяет полностью смоделировать любую иерархическую структуру и даже более сложные ситуации, когда один набор данных одновременно входит в несколько папок.

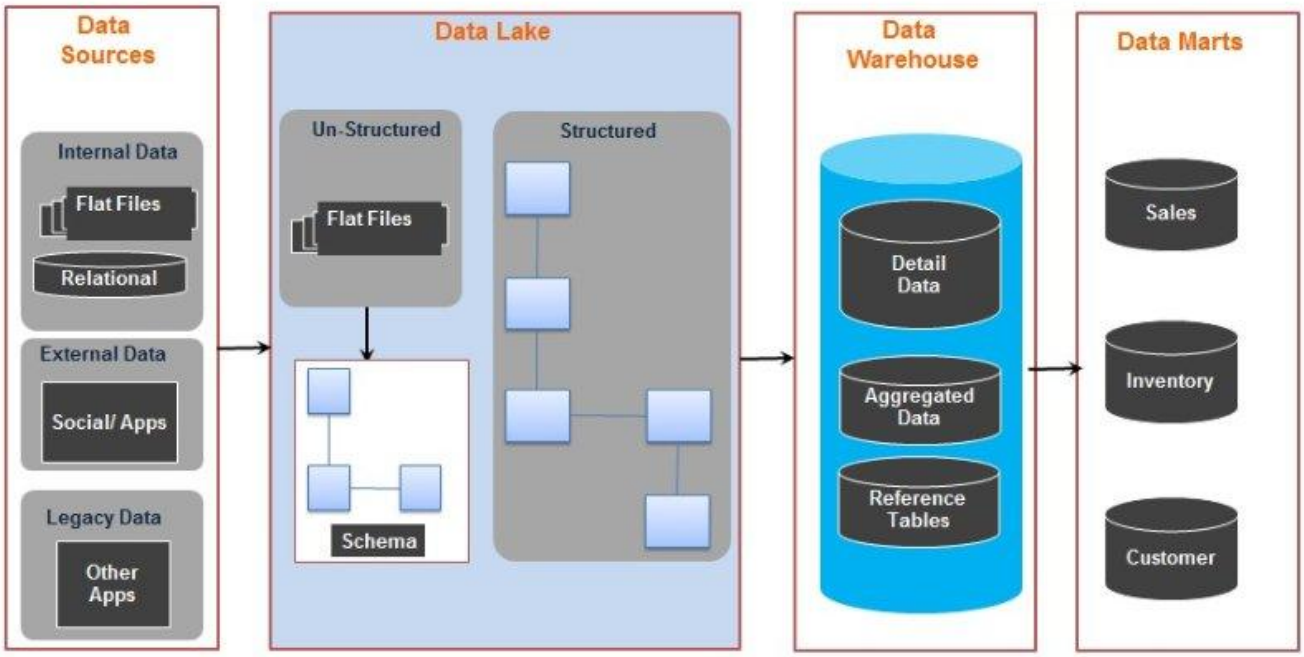
По мере перехода к использованию платформ Big Data концепция Data Lake постепенно становится одним из основных способов организации данных при построении аналитических бизнес-приложений [144], её использование становится оправданным [145]. Использование Data Lake позволяет упорядочить архивы и систематизировать безумное количество таблиц Excel, формируемых и хранимых во многих компаниях. Но процесс создания Data Lake нельзя начинать, не имея соответствующего плана и стратегии его развития [146], при этом должны использоваться подходы, наработанные в ходе предыдущего развития ИТ [147].

В продолжение этого подхода в [148] предлагается сопоставление концепций Data Lake и хранилища данных Data Warehouse, представленных на рисунках 17 и 18. При этом автор не противопоставляет эти концепции и не разносит их по времени, когда одна приходит на смену другой. Наоборот, он утверждает, что в компании, если всё правильно спроектировано, они должны использоваться одновременно, быть синхронизированы и дополнять друг друга, увеличивая общую эффективность.



Data Staging Area	
Schema	• A data model should be created and ready for storing data, this approach is referred to as **schema on write**. • Downside to this approach is that careful design can be **time consuming** which means lot of **planning** has to happen upfront.
Storage	• **Staged data doesn't live forever** rather data is flushed after moving to Data warehouse / Data mart. • Reason being storage back in days was **every expensive** and organizations could not afford to save unwanted data.
Processing	• **Processing massive Data sets** and support high frequency data is challenging. • It can process up to medium data volumes (up to multiple Terabytes) at a **very high cost.**
Scale Up	• Typically a staging area is built on RDBMS like Teradata, IBM and Oracle, which scales up vertically and has a **ceiling limit.** • Setting up, maintaining and scaling up has always been an **expensive proposition.**

Рис. 17. Структура Data Warehouse и функционирование Data Staging Area



Data Lake	
Schema	• **Bring all data in** and then create schema based on your need which is referred to as **schema on read**. • This approach brings a lot of **flexibility** and **offers extreme agility**, but may require some tweaking after schema is created.
Storage	• **No throw away data;** unlike staging area, data mostly resides forever as storage has become **very economical.**
Processing	• It **can process massive Data sets** from Disparate Sources and support high frequency data. It can process up to very large data volumes (up to Zettabytes) at a very low cost.
Scale Up	• Can be scaled up horizontally for storage and compute with ease using **commodity hardware**.

Рис. 18. Структура и функционирование Data Lake

11.2. Архитектура Data Lake

Организации долгое время стремились к созданию единой модели данных, которая может представлять каждый объект в разных ракурсах. Это было проблемой по следующим причинам:

- сущность может иметь несколько представлений по всему предприятию. Следовательно, для сущности может не существовать единой и полной модели;
- различные корпоративные приложения могут обрабатывать объекты на основе конкретных бизнес-целей, которые могут соответствовать или не соответствовать ожидаемым корпоративным процессам;

- разные приложения могут иметь разные шаблоны доступа и структуры хранения для каждого объекта.

Эти проблемы давно вызывают беспокойство и чувство неудовлетворённости на предприятиях, ограничивая стандартизацию бизнес-процессов, определение услуг и их состав. Внедрение Data Lake означает неявное достижение единой модели данных без значительного реального воздействия на приложения, которые способны решать очень специфические бизнес-задачи. Data Lake может представлять объект в полном объеме на основе информации, полученной из различных систем, которые владеют этими данными. Благодаря тому, что сущности представляются с гораздо более совершенными и полными деталями, Data Lake предоставляет предприятию множество дополнительных возможностей для обработки и управления данными.

По своей архитектуре классические озёра данных подразделяются на пондовые (прудовые) и зонные. Пондовая архитектура [149] разделяет принятые данные по их статусу и использованию. В частности, принятые данные сначала сохраняются в пруду необработанных данных, затем преобразуются и перемещаются в соответствии со своим назначением в один из следующих прудов: пруд аналоговых данных, пруд данных приложений или пруд текстовых данных. Напротив, зонная архитектура [150] разделяет жизненный цикл каждого набора данных на различные этапы. Например, могут быть отдельные зоны для загрузки данных и проверки качества данных, хранения необработанных данных, хранения очищенных и проверенных данных, обнаружения и изучения данных или использования данных для бизнес-/исследовательского анализа.

При классификации озёр данных в [151] они подразделяются на:

- озёра с файловым хранением данных;
- озёра на основе единой базы данных;
- озёра, использующие для хранения данных комбинацию разных систем хранения, например, Hadoop, графовую и реляционную СУБД.

В [152] обосновано, что построение таких мультипарадигмальных озёр обусловлено необходимостью обеспечить поддержку разных моделей данных. В [153] приведены примеры таких решений. Особенно большое разнообразие одновременно используемых систем хранения актуально для озёр данных масштаба предприятия, поскольку в больших компаниях, как правило, используется много разных приложений.

Рассмотренная в предыдущем разделе архитектура Data Lake возникла на начальном этапе формирования этой концепции. Архитектура Data Lake продолжает развиваться. На рисунке 19 показан более поздний вариант архитектуры Data Lake.

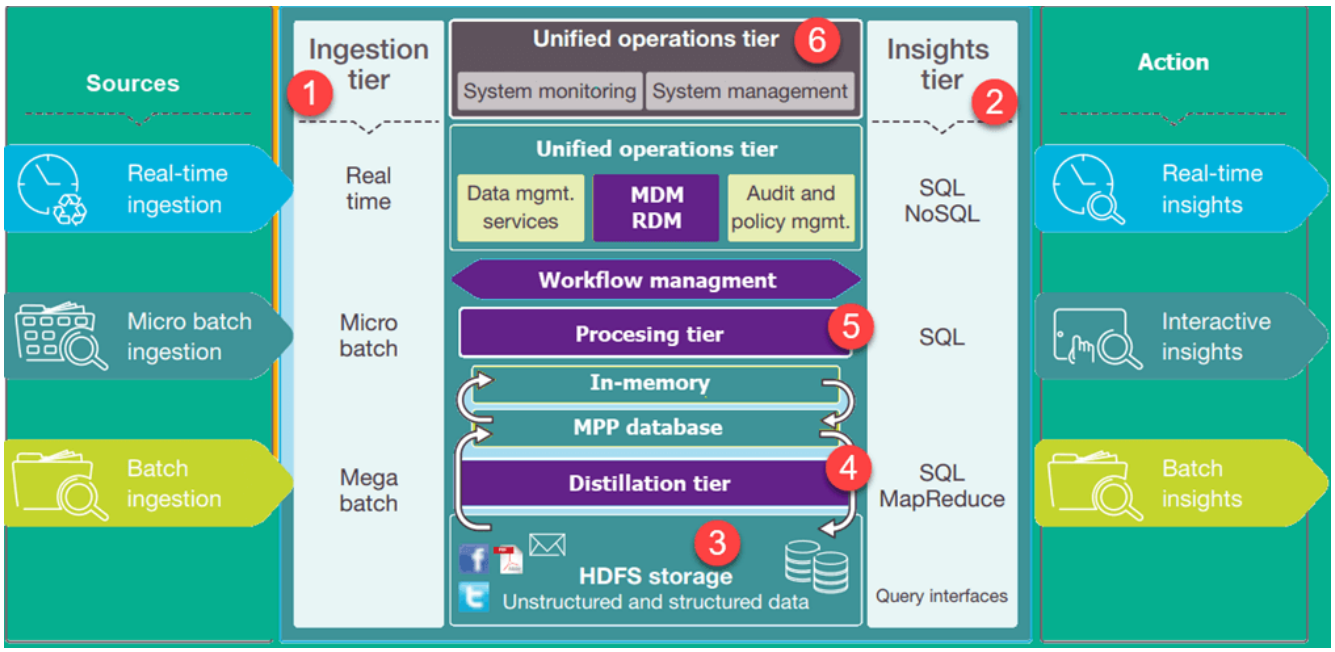


Рис. 19. Архитектура Data Lake

Нижние уровни представляют данные, которые в основном находятся в состоянии покоя, тогда как верхние уровни показывают данные транзакций в реальном времени. Эти данные проходят через систему без задержки или с небольшой задержкой. Краткое представление основных уровней представленной архитектуры.

- Уровень поглощения (Ingestion Tier): Элементы на левой стороне отображают источники данных. Данные могут быть загружены в озеро данных в пакетном режиме или в режиме реального времени.
- Уровень аналитики (Insights Tier): элементы справа представляют исследовательскую сторону, где используются аналитические данные системы. Для анализа данных могут быть использованы SQL- и NoSQL-запросы или даже Excel.
- HDFS – это экономичное решение для структурированных и неструктурированных данных при локальном хранении. Это зона сбора для всех данных, которые находятся в состоянии покоя в системе. в случае облачного хранения вместо HDFS могут использоваться Amazon S3, Microsoft Azure Blob Storage, Google Cloud Storage
- Уровень дистилляции (Distillation tier) берет данные из системы хранения и преобразует их в структурированные данные для более легкого анализа.
- На уровне обработки (Processing tier) выполняются аналитические алгоритмы и пользовательские запросы, Выполнение осуществляется в интерактивном пакетном режиме или режиме реального времени. Результатом являются структурированные данные, более удобные для анализа.
- Уровень унифицированных операций (Unified operations tier) – это управление и мониторинг системы. Он включает в себя аудит и управление навыками, управление данными, управление рабочим процессом.

11.3. Архитектуры систем, являющихся производными от Data Lake

Для того чтобы объединить преимущества озера данных и хранилища данных и устранить их недостатки с целью ускорения эффективного извлечения данных была предложена архитектура Data Lakehouse [154, 155] и разработаны решения на её основе. В Data Lakehouse обеспечивается как хранение как структурированных, так и неструктурированных данных. Для поступающих сырых данных обеспечивается возможность их очистки и последующего их хранения с возможностями, характерными для баз и хранилищ данных. Фактически, Data Lakehouse представляет собой объединение Data Lake и Data Warehouse.

Несмотря на наличие многочисленных определений понятия Data Space [156], начавших появляться с 2006 года, Data Space можно считать развитием архитектуры Data Lake для множества одновременно работающих пользователей с дата-контрактами, развитыми средствами определения полномочий доступа к данным и защиты данных от несанкционированного доступа [157]. В своей эволюции Data Space прошли 3 этапа: Closed Data Spaces, Open Data Spaces, Federations of Data Spaces [158]. Функциональные возможности Data Space: распределение данных, управление данными, аналитика данных, обработка данных и защита данных [159]. В некоторых платформах Data Space используется механизм блокчейн. Чтобы обеспечить контролируемое распределение данных и обмен данными между организациями.

Для объединения широкого спектра возможностей по обработке данных в пределах компании и интеграции различных инструментов была предложена концепция Data Fabric [160–162], как дальнейшее развитие Data Lake и Data Lakehouse. В рамках Data Fabric были объединены: Data Lake, Data Warehouse, разнообразные СУБД, использующие разные модели данных, средства потоковой обработки данных, средства Master Data Management (MDM), каталог метаданных, средства машинного обучения и средства визуализации данных. Объединение такого широкого спектра инструментов на единой интегрированной платформе позволяет с помощью Data Fabric выполнять все необходимые операции по работе с данными.

Ещё одним развитием концепций Data Lake and Data Warehouse является Data Mesh [163, 161, 162]. Она ориентирована на большие предприятия, в которых есть независимые территориально распределённые подразделения, самостоятельно развивающие и поддерживающие отдельные Data Lakehouse или Data Fabric для разных независимых предметных областей (domains). Data Mesh представляет собой федеративную структуру, позволяющую организовать взаимодействие между этими независимыми Data Lakehouse или Data Fabric и получение интегрированных данных потребителями.

12. Четвёртое поколение платформ Big Data: туманные платформы

Платформы для туманной обработки данных пока только появляются в своих ранних версиях. Это сейчас одно из горячих направлений исследований и конкуренции на мировом рынке. Их суть состоит в том, чтобы различные многочисленные относительно большие и совсем маленькие компьютеры, встроенные в оборудование, умные здания, автомобили, самолёты и другие активы для управления и обеспечения интернета вещей, а также пользовательские ноутбуки, смартфоны и другие носимые устройства могли образовывать временные сети для распределённой обработки данных без взаимодействия с центральными облаками. Совокупная вычислительная мощность и объём памяти этих периферийных устройств часто уже в

десятки раз превышает возможности корпоративного облачного ядра. Вся трудоёмкая обработка данных, включая моделирование для принятия решений и прогнозирование может делаться на местах, а облачные корпоративные приложения будут получать только итоговые результаты.

К 2020 г. завершена разработка ERP III, сейчас идет массовая эксплуатация этих систем, и одновременно с 2018–2019 гг. идет разработка концепции, опытная разработка и внедрение отдельных приложений следующего четвертого поколения корпоративных систем (в терминологии Gartner, EBC – Enterprise Business Capabilities) [164–166]. ERP 4-го поколения характеризуется широким использованием технологий Интернета вещей, туманных вычислений, дымчатых вычислений и блокчейна. Подключение большого количества Edge-компьютеров и мобильных устройств увеличивает объем данных, которые необходимо хранить, до эксабайт [167]. Такие объемы, естественно, исключают возможность дублирования всех данных для использования в аналитике.

Чтобы исключить фрагментацию данных и их избыточное дублирование необходим уровень, на котором осуществляется управление всеми данными, где бы они ни находились. Старшей по уровню структурой в организации управления данными сейчас является озеро данных, поэтому естественно эти функции возложить на него, расширив его роль.

Результаты поиска по публикациям позволяют предположить, что первым продуктом, в котором была сделана попытка распределить хранимые данные между центральным облаком и Edge-компьютерами, была Nebula [168], ранние рабочие версии которой появились ещё 2011 г. [169], позднее появились продукты Microsoft и Amazon Web Services [170]. При этом необходимо иметь в виду, что сложившееся распределение вычислительных мощностей на 3 слоя – Cloud, Fog и Mist (Edge) по мере распространения и развития решений IoT подвергается эрозии – появляются системы с пятью и шесть слоями [171], и нет универсального подхода, на каком слое заканчивать управление данными, т.е. где должна проходить береговая линия озера данных.

Ситуация с Edge-компьютерами во многом переносится на ситуацию с мобильными устройствами, поскольку нет чёткой границы между мобильными устройствами и Edge-компьютерами – Edge-компьютеры, установленные на транспортных средствах будут мобильными, а среди мобильных устройств возможно несколько слоёв, например телефон или планшет могут синхронизировать данные с ноутбуком через интерфейс M2M. И поэтому точно также для разных реализаций может варьироваться решение о том, какие устройства принадлежат озеру данных, а какие – нет.

Есть ещё ряд проблем, на которых положительно скажется объединение всех данных предприятия в единое озеро данных. Во-первых, это мультиоблачная обработка данных [172]. В случае объединения всех данных предприятия в единое озеро будет проведена чёткая граница, показывающая, что принадлежащие предприятию данные на разных облаках входят в состав озера, а другие используемые в процессе работы данные на этих облаках – это внешние данные. Во-вторых, будут объединены в единое озеро все данные предприятия, расположенные на разных географически распределённых дата-центрах [173]. Это позволит при необходимости проводить обмен данными между ними средствами системы управления озером данных.

Всё перечисленное показывает большие преимущества в случае, когда озеро становится единой структурой и способом организации хранения всех данных предприятия. В такой ситуации для построения озера данных целесообразно использовать единую универсальную модель данных, которая обеспечит обработку всех данных предприятия как в режиме OLAP, так и OLTP.

13. Архитектурные решения с использованием инструментов Big Data

13.1. Использование инструментов Big Data в интернет-компаниях

Центральное звено хранения и обработки данных во всех крупных интернет-компаниях реализуется с помощью инструментов Big Data. Потребности этих компаний, собственно, и были первой причиной создания этих инструментов. Так HDFS разрабатывалась как открытая альтернатива проприетарной GFS (Google File System) [103, 174]. Детальная архитектура созданных и используемых корпоративных систем, различающаяся из-за разнообразия решаемых задач, редко обсуждается в полном объёме, тем не менее, в блогах иногда представлены обобщённые схемы или обсуждаются отдельные технические подробности. Так в [175] и [176] показаны два различающихся варианта общей архитектура корпоративной системы Facebook по состоянию на 2012 г. Вариант из [175] приведен на рисунке 20.

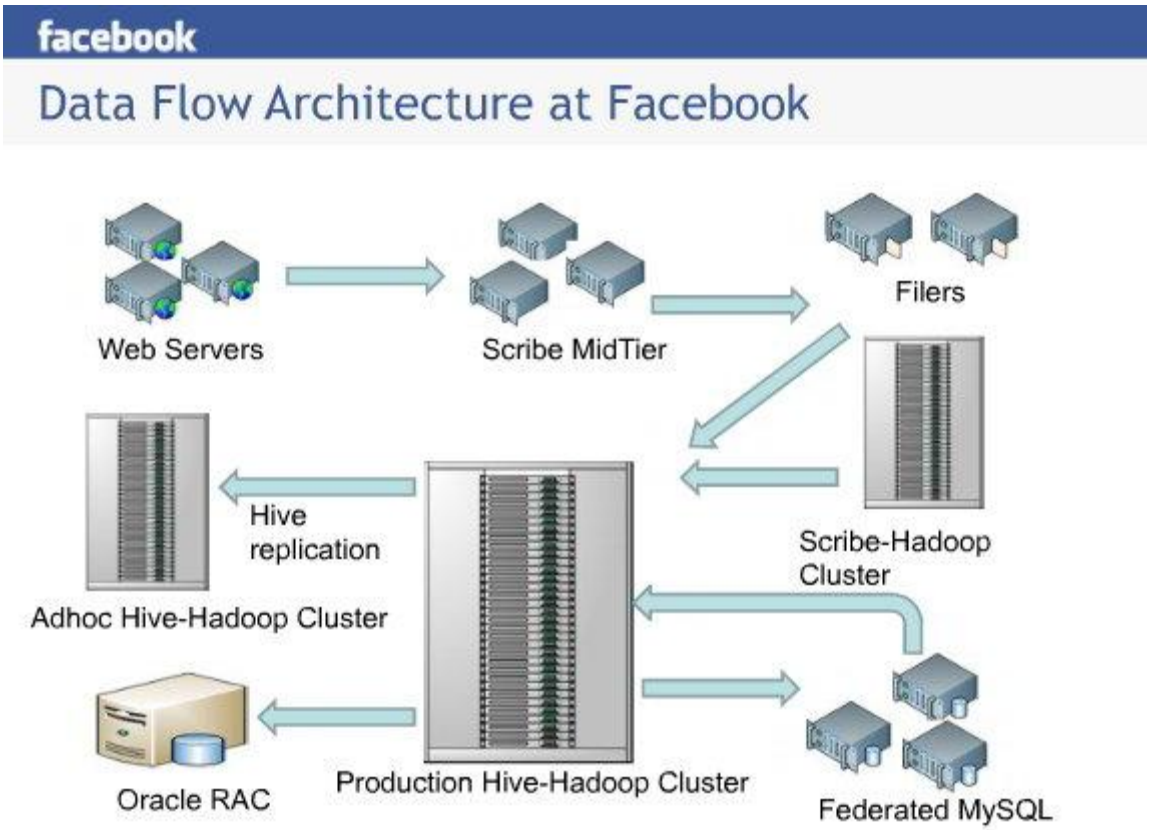


Рис. 20. Корпоративная архитектура, реализованная в Facebook

В отличие от Facebook для Yahoo не опубликована общая архитектура корпоративной системы за исключением эскиза [177], который представлен на рисунке 21, но зато опубликован ряд блогов и интервью [178–180], из которых можно понять, что в корпоративной системе

используются такие продукты как Apache Hadoop, Apache Pig, Apache Oozie, Apache HBase, Apache Hive, HCatalog (сервер метаданных Hive), Apache Storm, YARN, Apache Falcon, Apache Spark, Apache Tez, Apache ZooKeeper, Tableau, MicroStrategy.

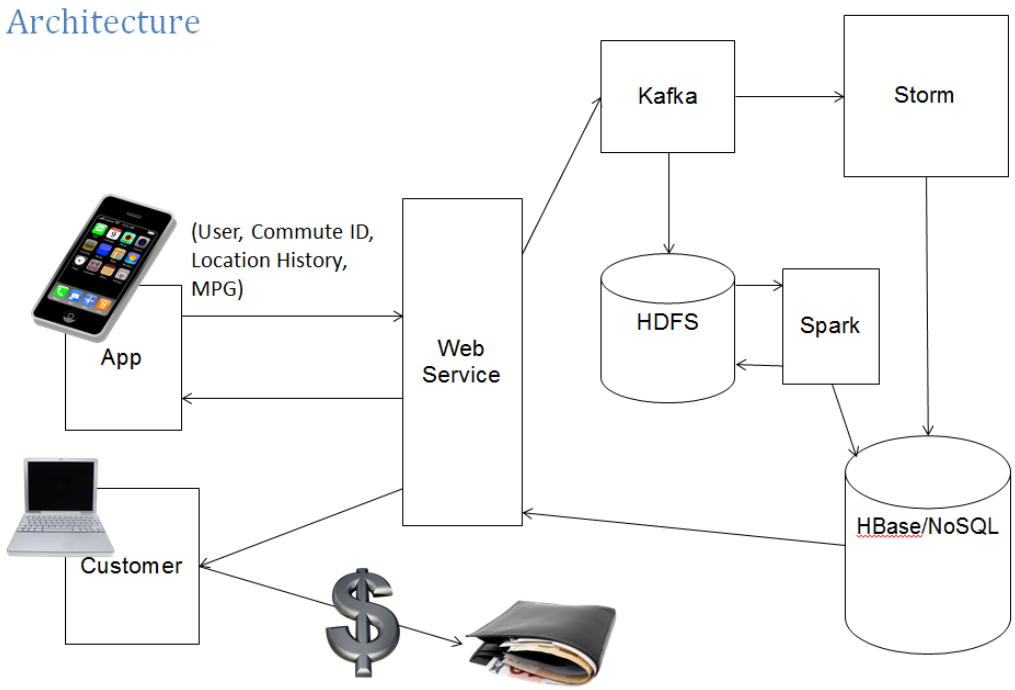


Рис. 21. Эскиз корпоративной архитектуры, реализованной в Yahoo

Представленные варианты архитектуры и различные комментарии позволяют сделать выводы, что основу корпоративной системы крупной интернет-компании представляют кластеры Hadoop, причём используется один основной кластер и один или несколько дополнительных, решающих различные задачи. Поверх Hadoop используется SQL-подобная СУБД Hive и ряд дополнительных инструментов Big Data, среди которых можно выделить средства управления мастер-данными и средства бизнес-аналитики.

Такие же тенденции можно увидеть в корпоративной архитектуре крупных российских интернет-компаний. Подтверждением этого является описание опыта работы с Hadoop в Mail.Ru [181].

13.2. Лямбда-архитектура для BI-проектов

Платформы Big Data пока ещё не стали центральным звеном корпоративной архитектуры, тем не менее, они широко используются в различных приложениях Business Intelligence, которые отдельными островками возникают в корпоративной архитектуре приложений. Для таких приложений, работающих в режиме реального времени, Натан Марц предложил специальную архитектуру, которую назвал лямбда-архитектурой [182, 183]. Термин получил признание и вошёл в широкое употребление.

Цель введения и использования лямбда-архитектуры – обеспечить создание систем, полностью устойчивых к сбоям оборудования и человеческим ошибкам, способных работать при разных уровнях нагрузки и в разных вариантах использования, одним из основных требований, к которым является малое время задержки при получении и обновлении данных.

Создаваемые на основе этой архитектуры системы должны обеспечить линейное масштабирование в связи с ростом нагрузки и будут лучше масштабироваться по нагрузке, чем по расширению типов обрабатываемых запросов.

Высокоуровневое представление лямбда архитектуры представлено на рисунке 22.

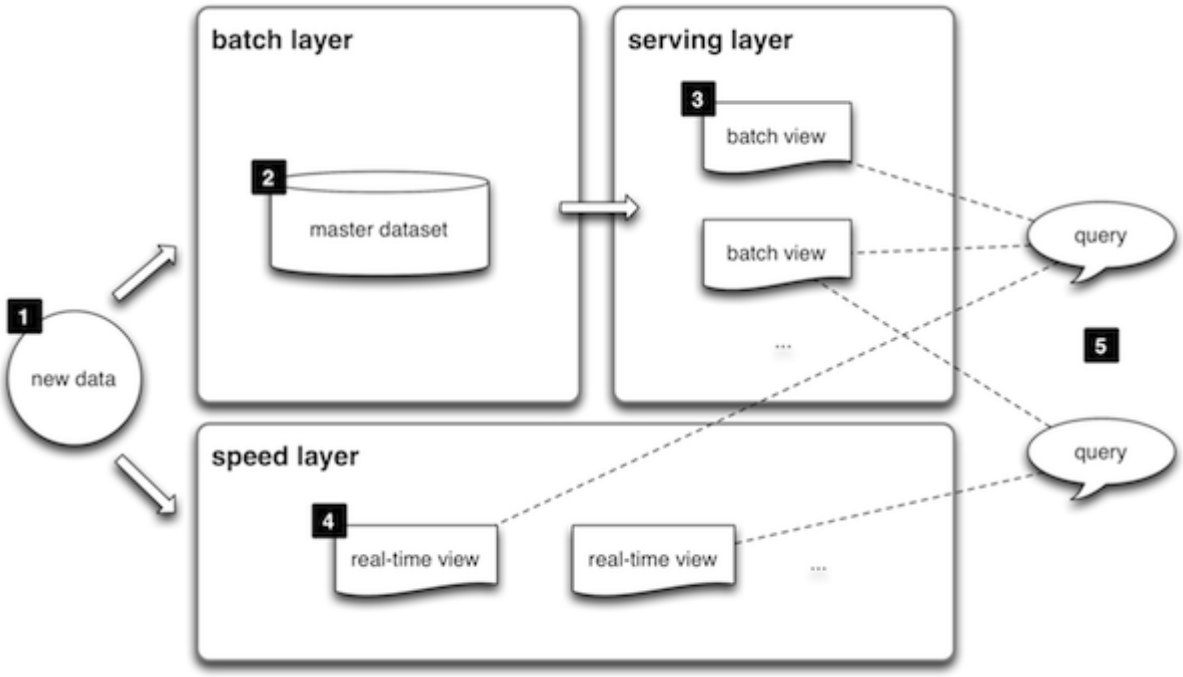


Рис. 22. Обобщённое представление лямбда-архитектуры

На представленной схеме:

1. Все поступающие входные данные направляются на обработку двумя способами – в пакетную обработку и в оперативную on-line обработку.
2. Подсистема пакетной обработки имеет две функции: а) управление основным набором данных (master dataset), в который не вносятся корректировки, а только добавляется новая информация; б) предварительное вычисление представлений данных в результате пакетной обработки.
3. Служебная подсистема индексирует представления данных, полученные в ходе пакетной обработки так, чтобы они могли на лету использоваться в запросах, требующих малого времени ответа.
4. Подсистема оперативной on-line обработки компенсирует высокие задержки времени служебной подсистемы при обработке обновлений и работает только с недавно поступившими данными.
5. Результирующая информация по любому входящему запросу формируется в результате объединения данных пакетной обработки и данных оперативной on-line обработки в реальном времени.

Системы, построенные на основе лямбда-архитектуры помимо высокой устойчивости к ошибкам и сбоям позволяют преодолеть основной недостаток приложений на базе Hadoop – большое время реакции, характерное для пакетной обработки данных.

13.3. Подход и рекомендовавшаяся архитектура SAP при переходе к SAP HANA в составе корпоративных системных ландшафтов

При проектировании корпоративных системных ландшафтов архитекторы компании SAP, лидирующей на мировом рынке корпоративных приложений, до появления SAP Vora считали целесообразным рассматривать возможность применения Hadoop при следующих условиях [184, 112]:

- необходимо обрабатывать данные объёмом в петабайты или даже в перспективе экзабайты, в любом случае их объём намного больше 100 ТВ и превосходит возможности традиционных реляционных СУБД и SAP HANA;
- не требуется быстрого получения результатов или обработки данных в реальном времени;
- не предъявляется стандартных для транзакционной обработки данных требований обязательного выполнения транзакции или отката в исходное состояние.

Применение Hadoop в указанных случаях значительно увеличит сроки обработки данных, она будет занимать часы или даже дни, однако удельные затраты на единицу объёма данных (MB, GB) значительно сократятся.

Все случаи, когда целесообразно применять Hadoop, были классифицированы, и для каждого из них компания SAP предложила шаблон типовой архитектуры [184, 185]. Объединённое представление первых четырёх шаблонов показано на рисунке 23.

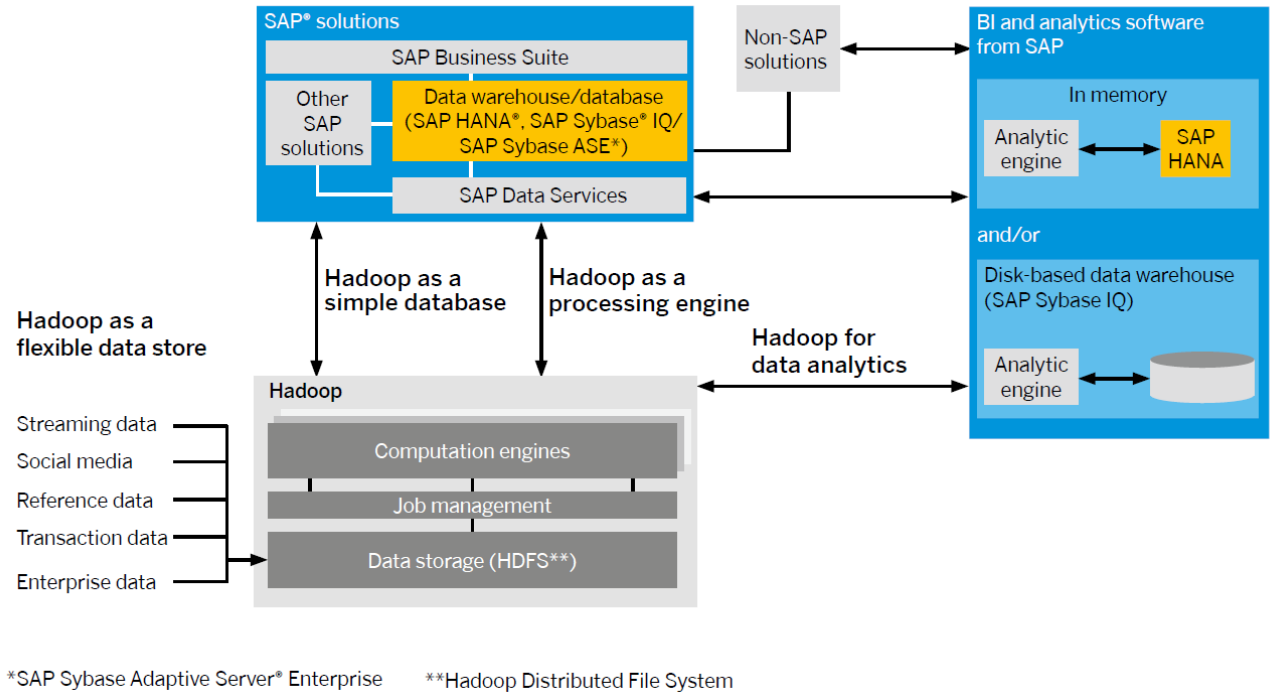


Рис. 23. Обобщённое представление четырёх архитектурных шаблонов применения Hadoop в корпоративных системах

Перечень всех разработанных шаблонов:

- Hadoop как гибкое хранилище данных;
- Hadoop как простая база данных;

- Hadoop как средство обработки данных;
- Hadoop для аналитики данных (простая аналитика);
- Hadoop для аналитики данных (двухфазная аналитика);
- Hadoop для аналитики данных (федеративные запросы / виртуализация данных);
- Использование Hadoop в масштабах предприятия.

Шаблон для двухфазной аналитики представлен на рисунке 24.

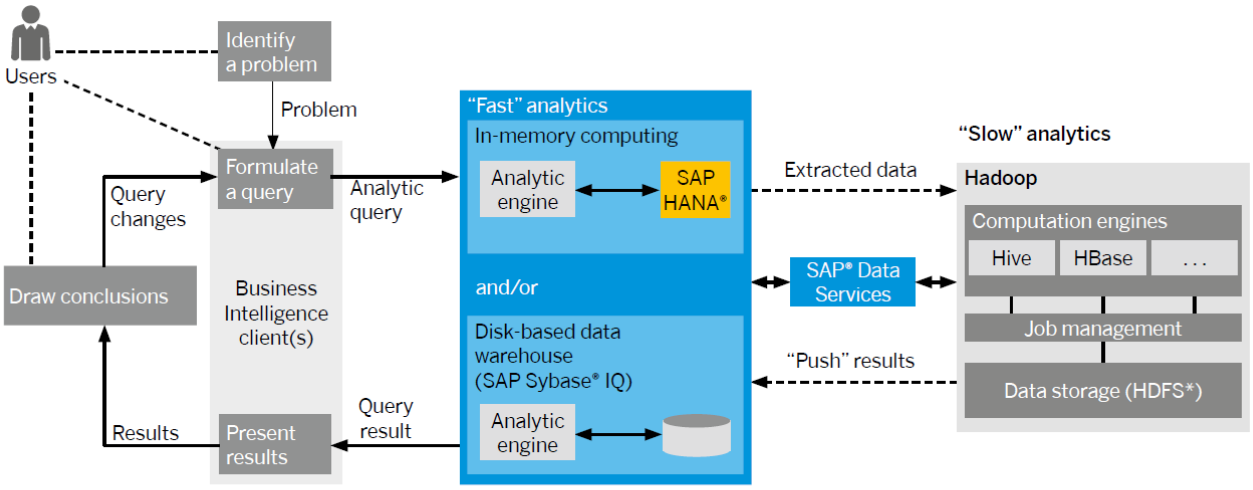


Рис. 24. Архитектурный шаблон применения Hadoop в корпоративных системах для двухфазной аналитики

Шаблон для аналитики федеративных запросов представлен на рисунке 25.

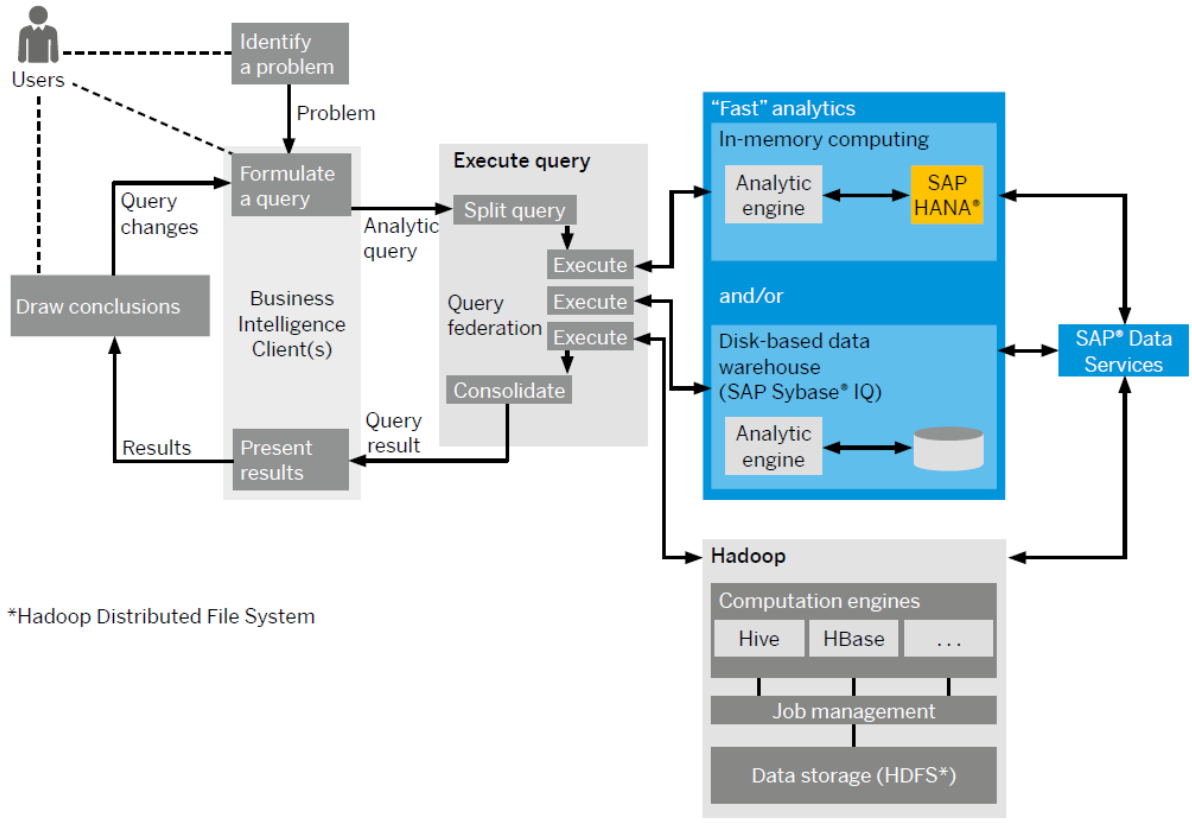


Рис. 25. Архитектурный шаблон применения Hadoop в корпоративных системах для аналитики федеративных запросов

Помимо архитектурных шаблонов компания SAP предложила также референсную архитектуру, показывающую, как Hadoop может быть встроен в ландшафт корпоративных приложений SAP [184, 186]. Эта архитектура представлена на рисунке 26.

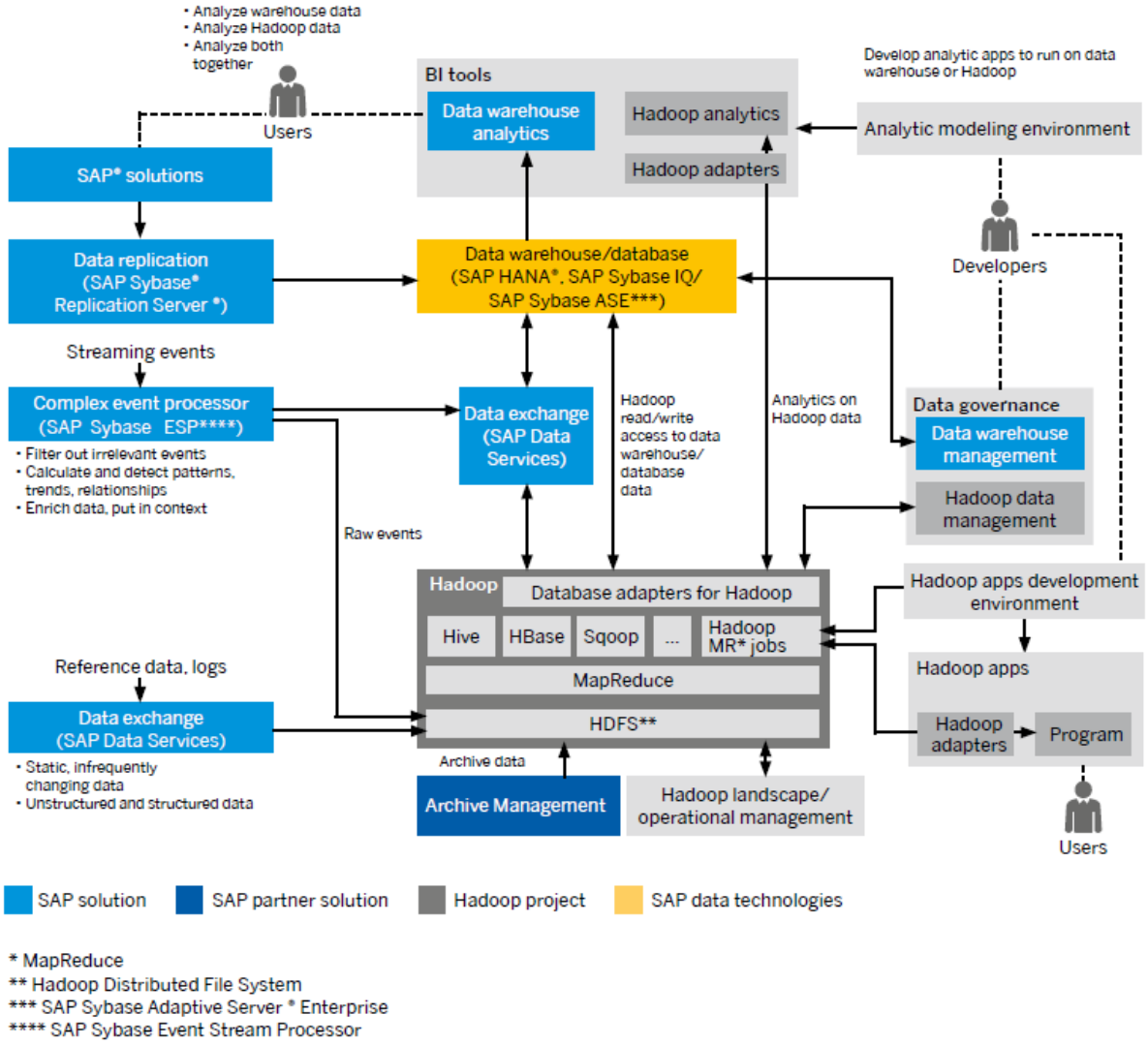


Рис. 26. Референсная архитектура с использованием Hadoop в системных ландшафтах на основе приложений SAP

В рамках референсной архитектуры выделены компоненты технологии обработки данных, источники данных, аналитические приложения и бизнес-приложения. Интеграция Hadoop с аналитическими и бизнес-приложениями всегда реализуется через хранилище данных или базу данных (SAP HANA, SAP IQ, SAP ASE) с использованием специализированных компонентов обмена данными и управления качеством данных.

Для иллюстрации разработанной архитектуры компания SAP показала четыре примера её применения (без глубокой детализации, только на верхнем уровне):

- использование Hadoop для упреждающего обслуживания оборудования;
- использование Hadoop для выработки рекомендаций в реальном времени по розничным покупкам;
- использование Hadoop для выявления проблем идентификации у оператора телекоммуникационной сети.
- миграция в Hadoop хранилища данных объёмом более 1 PB.

Дополнительно, вместе с архитектурой компания SAP сформулировала общие принципы использования Hadoop в составе корпоративных системных ландшафтов и типовую

последовательность действий по развёртыванию Hadoop в составе таких ландшафтов [184, 186]. Взаимодействие бизнес-приложений с Hadoop через SAP HANA SAP рассматривает как одно из перспективных направлений развития корпоративных систем. В SAP HANA, начиная с версии SPS 09, помимо возможностей обмена данными с Hadoop появилась возможность прямого вызова MapReduce из SAP HANA и обратного получения результатов [187].

Роль основного интегрирующего звена для работы с Big Data, которую SAP возлагает на SAP HANA, нашла подтверждение при создании комплекса для фиксации рекорда в книге рекордов Гиннесса. 5 марта 2014 года SAP построила связку SAP HANA–SAP IQ и продемонстрировала работу в режиме реального времени с базой данных объёмом 12,1 PB [188]. SAP HANA в этой связке обеспечивала взаимодействие с клиентами и играла роль гигантского кэша над СУБД, использующей жёсткие диски. 50% данных были структурированными и 50% - неструктурированными. Этим тестом SAP продемонстрировал возможность расширить область применения SAP HANA и других своих технологий до 12 PB без использования Hadoop или других интегрированных платформ Big Data. Для обработки данных существенно больших объёмов может использоваться SAP Vora.

14. Новые направления, возникающие в результате применения и дальнейшего развития инструментария Big Data, в научных дисциплинах, использующих моделирование

14.1. Переход к использованию кластеров Big Data создаёт новые возможности

Основным, бросающимся в глаза, эффектом от развития инструментов для работы с Big Data и их последующего применения является возможность решения множества новых, прежде не решавшихся задач в самых разных областях деятельности, что показано в разделе 3 данной части лекции. Однако есть и другой менее видимый эффект от распространения инструментов Big Data, который пока только формируется, – внутри дисциплин (областей знаний), связанных с моделированием окружающей действительности (технических, естественных или социально-экономических систем) и / или решением практических задач на базе использования построенных моделей, возникают новые направления. Эти новые направления позволяют решать свойственные каждой дисциплине задачи для систем, содержащих значительно большее количество отдельных элементов и связей между ними. В зависимости от сложности отдельных элементов новые направления могут решать задачи для систем с миллионами элементов, каждый из которых имеет сотни или тысячи параметров или для систем, содержащих миллиарды элементов, описываемых одним-двумя параметрами.

Первой такой частной дисциплиной, в которой возникло новое направление в связи с появлением инструментов Big Data, является теория алгоритмов сортировки и поиска данных, детально изложенная в [189]. В ходе создания распределённых файловых систем и ряда других программных продуктов Big Data вырабатывались новые подходы и методы теории сортировки и поиска, позволяющие работать с большими объёмами данных. Однако инструменты Big Data начинают проникать и в другие дисциплины, каждая из которых обладает свойственным ей специфическим аппаратом моделирования и решения задач. В этом разделе мы более детально рассмотрим указанную тенденцию на примере следующих областей науки:

- численные методы,
- теория имитационного моделирования,
- теория управления, приложения которой можно разделить на модели и методы управления в технических системах, модели и методы управления в биологических системах, модели и методы управления в социальных и экономических системах.

Обсуждаемая тенденция появления новых направлений на основе применения инструментов Big Data не ограничивается перечисленными дисциплинами. Можно с уверенностью говорить о факте появления и начале развитии такого направления в теории графов, и в математической статистике, а также о прогнозе появления таких направлений в исследовании операций, в теории игр и в других дисциплинах. В теории графов развитие нового направления связано со спецификой графов социальных сетей, а в математической статистике – со средствами предиктивной аналитики на больших данных. Однако, анализ проявлений этой тенденции в перечисленных областях выходит за рамки данной лекции, как из-за ограниченности её объёма, так и из-за того, что дополнительно потребуется рассмотреть особенности наблюдаемой тенденции в дисциплинах, являющихся разделами математики.

Фактически, формирующиеся новые направления возникают вследствие того, что появляется возможность решать проблему, привлекая десятки и сотни тысяч самостоятельных узлов обработки данных. Работу этих узлов нужно организовать, для этого нужно использовать новые подходы, новые алгоритмы, схемы распределения работ и консолидации полученных результатов.

Может показаться, что эти задачи не являются чем-то новым, – уже долгое время развивается направление суперкомпьютерных вычислений, в рамках которого решаются похожие задачи [190]. Кластеры Big Data и самые мощные суперкомпьютеры с распределённой памятью, тоже представляющие собой кластеры, на первый взгляд, очень похожи. В обоих случаях используется архитектура Master–Slave. Тем не менее, имеется много различий:

- в суперкомпьютерных кластерах используются значительно более высокоскоростные каналы для обмена данными между узлами;
- ветви программы, параллельно выполняющиеся в разных узлах суперкомпьютерного кластера, обмениваются между собой сообщениями или данными. В отличие от этого в кластере Hadoop возможности обмена данными очень ограничены. Map-задачи при запуске берут исходные данные из блоков HDFS, располагающихся на данном узле, а результат направляют на вход одной из Reduce-задач;
- при создании суперкомпьютеров используются более надёжные компоненты, чем commodity-сервера в кластерах Hadoop. Поэтому в алгоритмах для кластеров Hadoop обязательно предусматривается ситуация, что в части узлов могут возникнуть сбои, и решаемые ими задачи надо решить повторно.

Предельно экономный подход, реализуемый в кластерах Hadoop, позволил снизить стоимость параллельной обработки данных примерно в 10 раз по сравнению с суперкомпьютерами. Это дало возможность массово применять инструменты Big Data в корпоративных системах и выйти далеко за пределы традиционных областей применения суперкомпьютеров:

- ядерная физика,
- моделирование климата,

- генная инженерия,
- проектирование интегральных схем,
- анализ загрязнения окружающей среды,
- создание лекарственных препаратов и новых материалов,
- проектирование эффективных форм с учётом гидро- и аэродинамики.

Использование инструментов Big Data в составе корпоративных систем с их большим числом параллельно обращающихся пользователей и режимом работы 24/7 сразу же заставило предъявлять к ним значительно более высокие требования по надёжности, доступности, скорости работы по сравнению с программами для суперкомпьютерных расчётов, используемыми, в основном, в научно-исследовательской и проектно-конструкторской деятельности. Эти более высокие требования, а также радикально отличающиеся от суперкомпьютерных систем принципы построения процессов обработки данных делают необходимым появление внутри каждой научной дисциплины нового направления, объединяющего разрабатываемые решения по использованию создаваемых инструментов в рамках экосистемы Hadoop.

Термин Big Data объединил под единый зонтик все теоретические результаты, технические решения, алгоритмы и инструменты для работы с большими объёмами данных, развиваемые в рамках экосистемы Hadoop. Эти решения алгоритмы и инструменты содержат элементы нового раздела теории алгоритмов сортировки и поиска. По аналогии, можно ввести похожие названия для новых направлений в других науках для того, чтобы отделить все наработки и инструменты от похожих по назначению аналогов в рамках суперкомпьютерных вычислений:

- Big Calculation - для нового направления в численных методах, связанного с вычислениями на кластерах Hadoop;
- Big Simulation - для нового направления в имитационном моделировании на базе кластеров Hadoop;
- Big Management - для нового направления в управлении социальными и экономическими системами на базе кластеров Hadoop;
- Big Optimal Control - для нового направления в теории оптимального управления на базе кластеров Hadoop.

Естественный вопрос, который может возникнуть: почему нельзя подвести все перечисленные направления под единый зонтичный термин Big Data? Дело в том, что для решения многих задач может потребоваться именно большой объём вычислений или большой объём моделирования, при этом объём используемых данных будет существенно меньше условного 1 PB, и данные не будут иметь некоторых других характеристик, свойственных Big Data согласно разделу 2 данной лекции. Иначе говоря, данные могут быть обработаны в обычной СУБД, но для реализации алгоритмов потребуется кластер Hadoop.

Ещё одним теоретическим обобщением, которое не позволяет отобразить характерные особенности процессов моделирования и обработки данных с использованием различных моделей, является подход 5W, описываемый во многих источниках (например, [191]). Согласно этому подходу, все инструменты, создаваемые в рамках экосистемы Hadoop и даже за её пределами, относят к средствам аналитики, отвечающим на один или несколько из следующих пяти вопросов:

- What is happening? – Что происходит?
- Why did it happen? – Что случилось?
- What could happen? – Что может случиться?
- What action should I take? – Какие действия я должен сделать?
- What did I learn, what's best? – Чему я научился, что является лучшим?

В этом случае инструменты, построенные на основе совершенно разных методов моделирования, математических и инженерных подходов, объединяются в одну группу. При таком объединении и обобщении нивелируется специфика используемых методов моделирования и тормозится развитие новых востребованных разделов отдельных дисциплин.

14.2. Big Calculation

Большинство алгоритмов численных методов изначально разрабатывались в последовательной парадигме. Мы рассмотрим только два примера, чтобы показать применение нового подхода, который возникает под влиянием использования инструментов Big Data. Первый пример – это симплекс метод для решения задач линейного программирования, который можно обобщённо представить следующим образом:

- Поиск одной из вершин выпуклого многогранника, представляющего собой область допустимых решений.
- Последующее перемещение по рёбрам этого многогранника от одной вершины к другой до тех пор, пока не будет найдена вершина, в которой целевая функция принимает максимальное значение.

Второй пример – метаэвристика поиска с чередующимися окрестностями для решения задач непрерывной и дискретной оптимизации VNS [192], которую в общем виде можно описать так:

- Определение последовательности размеров окрестностей и начальной точки.
- Циклический поиск локального оптимума, начиная с первого заданного размера окрестности и заданной начальной точки. Если в результате локального поиска на очередном шаге будет найдено новое лучшее значение оптимума, использовать на следующем шаге найденную точку оптимума в качестве начальной, в противном случае перейти к поиску на следующем размере окрестностей.

Появление вычислительных кластеров повлекло за собой разработку новых параллельных численных методов. Так компания SAP к 2004 г. испытывала потребность в переходе на параллельные алгоритмы решения задач целочисленного линейного программирования из-за имеющей место тенденции роста размерности при оптимизации цепочек поставок [193]. В компании проводились исследования по реализации параллельных алгоритмов с помощью декомпозиции исходных матриц на блоки и параллельного решения задачи оптимизации отдельных блоков на разных узлах кластера. Результаты проведенных исследований показали, что при увеличении числа параллельно обрабатываемых блоков основной матрицы более 25–35 дальнейшего увеличения скорости решения задачи не происходит.

Для повышения эффективности поиска глобального оптимума с помощью чередующихся окрестностей также было разработано несколько вариантов параллельного алгоритма VNS (PVNS). Наиболее эффективный из них заключался в наращивании числа решений,

выбираемых в текущей окрестности, и параллельном выполнении локального поиска для каждого из них. Этот подход, как и в предыдущем примере, тоже предполагает использование нескольких или, как максимум, нескольких десятков параллельно работающих узлов кластера.

Инструменты Big Data в своём современном состоянии ориентированы на совершенно другие характеристики кластеров. Число параллельно работающих узлов может составлять десятки и сотни тысяч. Для эффективного использования таких ресурсов требуются принципиально иные алгоритмы. Например, та же задача линейного программирования может решаться следующим образом: параллельно будут найдены все вершины выпуклого многогранника и вычислены значения целевой функции в каждой из них, а потом из этих значений будет выбрано максимальное. При больших размерностях задачи каждый узел кластера будет последовательно решать несколько таких независимых подзадач. Однако для любой задачи конкретной размерности всегда будет существовать некоторое количество узлов в кластере, начиная с которого дальнейшее увеличение их числа при решении задачи описанным методом будет уменьшать общее время решения по сравнению с методами распараллеливания, ориентированными на несколько десятков узлов. Общий объём выполненных вычислений при использовании предложенного подхода существенно увеличится, однако общее время решения может значительно сократиться.

Точно также подход максимального распараллеливания может быть применён для быстрого решения задачи поиска вместо применения метаэвристики VNS, – область допустимых решений может быть покрыта сеткой начальных точек, общее число которых может в несколько раз превысить число узлов в кластере. Из каждой начальной точки параллельно будет выполнен локальный поиск, а потом сопоставлены полученные результаты.

По мере расширения использования инструментов Big Data будут появляться всё новые и новые численные методы, ориентированные на возможности больших кластеров.

14.3. Big Simulation

Системы имитационного моделирования прошли длинный путь развития, начиная с 70-х годов XX века разработаны десятки систем [194]. Системы имитационного моделирования, работающие на отдельном компьютере или сервере, позволяют моделировать поведение максимум нескольких десятков тысяч объектов. Для преодоления этого ограничения был разработан новый подход к построению имитационных моделей – агентное моделирование и инструменты для построения агентно-ориентированных моделей с использованием грид-систем [195]. В рамках агентного моделирования имитационная модель представляет собой децентрализованное сообщество независимо действующих агентов. К настоящему моменту реализованы десятки инструментов агентного моделирования на базе суперкомпьютеров, с помощью которых можно строить имитационные модели, включающие сотни миллионов и миллиарды объектов. Это позволяет решать задачи, в которых необходимо моделировать большое число объектов, например:

- прогнозирование развития социально-экономических систем (стран, регионов, городов);
- моделирование миграционных процессов;
- имитация и оптимизация пешеходного движения;

- моделирование транспортных перевозок и транспортных систем;
- прогнозирование экологического состояния окружающей среды;
- моделирование работы систем сотовой связи

и др.

Появление в составе корпоративных систем кластеров Hadoop, которые могут содержать сведения о сотнях миллионов участников социальной сети, или о десятках миллионов активов (зданий, сооружений, единиц оборудования) естественным образом выдвигает вопрос об использовании этих данных для построения имитационных моделей. Например, имитационной модели, которая будет позволять прогнозировать надёжность работы оборудования в зависимости от использования различных стратегий технического обслуживания и ремонта. В качестве технических систем с большим числом элементов (десятки миллионов) могут выступать региональная или мультинациональная электрическая сеть, сеть трубопроводов, сеть железных дорог и т.д. Другой задачей, где необходимо использовать имитационную модель на базе кластера Hadoop, является прогноз развития социальной сети или прогноз поведения её участников при возникновении определённых обстоятельств.

Использование суперкомпьютеров не нашло широкого применения для решения этих задач. Во многом, на наш взгляд, это обусловлено высокими затратами, возникающими при этом. Затраты в случае использования кластера Hadoop будут на порядок меньше, и сам кластер доступнее – часто он уже есть в корпоративном периметре или арендуется в общедоступном облаке. До появления Hadoop 2.0 и Spark сложно было ожидать реализации систем имитационного моделирования. Жёсткий двухстадийный однонаправленный процесс обработки данных MapReduce не соответствовал характеру многостадийной циклической обработки данных при имитационном моделировании с обязательным обменом данными между взаимосвязанными элементами модели после каждого шага. Появление YARN и Spark радикально изменило эту ситуацию. Стало возможным конструировать процессы обработки данных с любым числом стадий и реализовывать сложные схемы обмена данными между задачами, выполнявшимися на разных узлах.

Для упрощения разработки на базе YARN приложений, в которых необходимо обеспечить многостадийный процесс обработки данных и обмен информацией между узлами, была разработана платформа Apache Hama. Эта платформа реализует модель программирования BSP (Bulk Synchronous Parallelism). Согласно этой модели, весь процесс вычислений состоит из последовательности супершагов [196]. Каждый супершаг выполняется параллельно каждым узлом, участвующим в BSP-вычислениях. Супершаг содержит три стадии: локальные вычисления, обмен информацией и синхронизационный барьер. Каждый узел имеет локальную память, которая доступна только этому узлу в течение всех супершагов. Кроме того, во время локальных вычислений на данном супершаге каждый узел имеет доступ к сообщениям, посланным другими узлами во время предыдущего супершага. Он также может послать сообщения другим узлам во время стадии информационного обмена, чтобы они были прочитаны ими во время следующего супершага. Синхронизационный барьер позволяет синхронизировать работу всех узлов, чтобы обеспечить получение ими всех посланных им сообщений до начала следующего супершага. Предусматривается также обработка возможных сбоев. Каждый узел может использовать точки восстановления, чтобы эпизодически сохранять изменившуюся

часть памяти в распределённую файловую систему. Это позволяет восстановить последнее запомненное состояние в случае сбоя.

В связи с созданием необходимого инструментария для разработки систем имитационного моделирования на базе кластеров Hadoop в ближайшем будущем можно ожидать появления сначала отдельных имитационных моделей, а потом систем имитационного моделирования класса Big Simulation. Новые большие модели могут создаваться как посредством разработки на языках программирования, так и с использованием на каждом узле платформы Big Data отдельного экземпляра какой-нибудь из существующих систем имитационного моделирования и описания соответствующего компонента модели на соответствующем языке моделирования. В последнем случае необходима только разработка программного обеспечения, которое объединяет в единую модель системы имитационного моделирования, работающие в отдельных узлах платформы Big Data.

Примеры моделей с миллионами или десятками миллионов параллельно моделируемых объектов, которые могут быть построены с использованием платформ Big Data:

- Имитационная модель работы городского транспорта и пешеходного движения в крупном городе на базе данных сотовой связи о перемещениях всех жителей крупного города за определённый период. Наличие такой модели позволит оптимизировать маршруты городского транспорта и интервалы движения транспортных средств по маршрутам.
- Имитационная модель пассажирских перевозок в пределах региона, страны, всего мира в целом на базе данные о купленных отдельными гражданами авиационных или железнодорожных билетах за определённый период. Наличие такой модели позволит оптимизировать способы, маршруты и расписание пассажирских перевозок.
- Имитационная модель социальной сети на базе данных социальной сети о связях и активности всех участников. Наличие такой модели позволит получать оценки скорости распространения информации по социальным сетям, эффективно планировать рекламные кампании и другие маркетинговые мероприятия в социальных сетях.
- Имитационная модель сети сотовой связи на базе данных сети сотовой связи о всех операциях, выполненных отдельными гражданами Наличие такой модели позволит оптимизировать затраты на развитие сети сотовой связи.
- Имитационная модель парка оборудования крупной компании на базе данных компании об отказах и ремонтах всего парка оборудования. Наличие такой модели позволит оптимизировать затраты на ремонт существующего и приобретение нового оборудования.

Приведенный перечень моделей, в которых необходимо обеспечить работу миллионов или десятков миллионов независимых процессов, является далеко неполным. Помимо пассажирских перевозок, можно также моделировать грузовые. Помимо сетей сотовой связи можно моделировать работу торговых сетей. И т.д. Всё это свидетельствует о перспективности имитационных моделей на основе платформ Big Data и скором появлении таких моделей.

Big Management

Под термином Management понимается управление в социальных и экономических системах. В настоящее время в автоматизированных системах управления используется всего два

способа сведения множества частных показателей к укрупнённым показателям, отражающим соответствие принятой стратегии развития: использование системы сбалансированных показателей (Balanced Scorecard) и метод управления портфелями [197]. Система сбалансированных показателей строится как иерархическая система, основанием которой является множество показателей деятельности сотрудников низовых звеньев, а на верхнем уровне они консолидируются в небольшое число показателей, контролируемых топ-менеджментом и характеризующих деятельность компании в целом. Число уровней иерархии в системе сбалансированных показателей соответствует числу уровней иерархии в системе управления компанией. Система управления портфелями позволяет классифицировать и объединить в ограниченное число портфелей набор относительно однородных объектов или действий, которыми необходимо управлять. При этом число элементов в каждом портфеле может быть достаточно большим.

Использование программных инструментов для поддержки двух указанных способов менеджмента позволяет ставить чёткие количественно выраженные цели, контролировать их своевременное достижение или выявлять причины, почему они не достигнуты. В случае появления отклонений по результатам анализа их причин вносятся изменения в бизнес-процессы и/или корректировки в систему показателей.

Почему возникает потребность в инструментах Big Management?

1. Мы уже упоминали в предыдущем разделе, что крупная компания может владеть или использовать десятки миллионов активов (зданий, сооружений, единиц оборудования). Все активы нуждаются в профилактике (техническом осмотре, контроле состояния), ремонте, модернизации. Это всё работы или проекты, для которых необходимо выделение бюджета, специалистов, зачастую, временный вывод из эксплуатации и т.д. Если с каждым активом необходимо провести работы хотя бы один раз в квартал, мы сразу получим около 100 миллионов отдельных мелких или крупных проектов. До последнего времени системы управления проектами могли уверенно поддерживать одновременное планирование и учёт работы по нескольким десяткам тысяч проектов. После перевода SAP PPM (Project Portfolio Management) на платформу SAP HANA SAP заявил, что система сможет поддерживать неограниченное число проектов, однако сообщений об опыте внедрения для поддержки миллионов одновременно выполняемых проектов, не говоря уже о десятках или сотнях миллионов, пока не появлялось.

2. Системы сбалансированных показателей успешно функционируют в крупных компаниях, численность сотрудников в которых может составлять десятки тысяч человек. В масштабах целой страны, или такого объединения, как Евросоюз, общая численность управленцев может составлять несколько миллионов. Потребность в организации эффективной работы таких больших аппаратов управления очень велика, однако пока нет примеров внедрения системы сбалансированных показателей для таких больших структур.

14.4. Big Optimal Control

Основная задача теории оптимального управления – найти последовательность управляющих воздействий, которые обеспечат переход системы из имеющегося начального состояния в некоторое заданное конечное и при этом будет достигаться максимум или минимум заданного критерия. Математическая модель, используемая для описания задачи, включает в себя: начальную точку, параметры управления, описание поведения системы, оптимизируемый

критерий, существующие ограничения на ресурсы. Поведение детерминированных систем описываются дифференциальными уравнениями, дифференциальными уравнениями в частных производных и конечными автоматами. Вероятностные системы описываются стохастическими дифференциальными уравнениями и марковскими процессами.

Проблемы, связанные с решением практических задач оптимального управления, привели к появлению отдельных групп численных методов для решения задач оптимального управления и специальных программных комплексов [198]. В случаях, когда формальное описание задачи не может быть сформулировано из-за его сложности, но может быть построена имитационная модель системы, оптимальное решение может быть найдено методами прямой оптимизации, работающими поверх имитационной модели.

На начальном этапе теория оптимального управления создавалась для оптимизации управления техническими системами. Примеры, приведенные в предыдущих разделах, показывают, что существует много технических систем с миллионами параметров управления (сеть газопроводов страны, региона, электрическая сеть аналогичных масштабов и т.д.). Для упрощения управления такими сложными объектами соответствующие системы управления строятся в виде многоуровневых иерархических систем. В большинстве случаев нижние уровни полностью управляются автоматически, на их долю приходятся рутинные операции и предотвращение аварийных ситуаций. На верхних уровнях таких больших систем из-за высокой сложности и недостаточной проработанности систем управления автоматическое управление в большинстве случаев подменяется автоматизированным - в контур управления включают людей-операторов.

По мере развития теория оптимального управления вышла за пределы чисто технических систем, и на стыке техники и технологий с естественнонаучными и экономическими дисциплинами появились и продолжают возникать всё новые и новые задачи большой размерности. Приведём по одному примеру:

- (на стыке с науками о земле): на давно разрабатываемом нефтяном месторождении пробурено несколько тысяч скважин, получены данные о характеристиках проницаемости пород в точках бурения, имеются данные сейсморазведки и данные истории добычи. По этим данным должна быть построена гидродинамическая модель месторождения, а затем решена задача оптимального управления, которая может быть поставлена в одном из двух вариантов:
  - в условиях заданного ограничения на бюджет найти совокупность геолого-технических мероприятий, которые позволят максимально повысить нефтеотдачу;
  - для заданного уровня нефтеотдачи найти совокупность геолого-технических мероприятий, для выполнения которых потребуется минимальный бюджет.
- (на стыке с сельским хозяйством): в рамках примера по использованию больших данных для увеличения производительности сельскохозяйственного производства, приведенного в разделе 3 данной лекции, должна решаться задача минимизации затрат на достижение заданного объёма урожая. Управляющими воздействиями в данном случае будут являться агротехнические мероприятия.
- (на стыке с экономикой): в рамках примера 1, приведенного в разделе 7.3 данной лекции, по планированию технического обслуживания и ремонта сложного технического

комплекса, должна решаться задача минимизации затрат на достижение заданного уровня надёжности его работы. Управляющими воздействиями в данном случае будут являться работы по техническому обслуживанию и ремонту.

Для решения больших задач оптимального управления на базе низкобюджетных кластеров потребуется развивать всю совокупность решений и методов, перечисленных в предыдущих разделах 11.2, 11.3, 11.4: новые численные методы, приложения на основе модели программирования BSP, большие имитационные модели, методы построения иерархических систем управления и методы управления портфелями однородных процессов или объектов.

Список литературы

1. NIST Big Data Interoperability Framework: Volume 1. Definitions. Final Version 1. September 2015. DOI 10.6028/NIST.SP.1500-1.
2. Gandomi A., Haider M. Beyond the hype: Big data concepts, methods, and analytics // International journal of information management. 2015. V. 35. № 2. P. 137–144. DOI 10.1016/j.ijinfomgt.2014.10.007.
3. Jin S.J., Abdullah A.H., Mokhtar M., Abdul Kohar U.H. The Potential of Big Data Application in Mathematics Education in Malaysia // Sustainability. 2022. V. 14. № 21. Статья 13725. DOI 10.3390/su142113725.
4. 8V Spider - Big Data Assessment Model // IT Strategy and Architecture. November 27, 2012. URL: http://infrastructurearchitecture.blogspot.ru/2012/11/8v-spider-big-data-assessment-model.html (дата обращения 20.08.2016).
5. Moro Visconti R., Morea D. Big data for the sustainability of healthcare project financing // Sustainability. 2019. V. 11. № 13. Статья 3748. DOI 10.3390/su11133748.
6. Hussein A. A. How many old and new big data v's characteristics, processing technology, and applications (bd1) // International Journal of Application or Innovation in Engineering & Management. 2020. V. 9. №. 9. P. 15–27. URL: https://www.researchgate.net/profile/Abou-El-Ela-Hussien/publication/344646513_International_Journal_of_Application_or_Innovation_in_Engineering_Management_IJAIEM/links/5f86bacf458515b7cf7f78b1/International-Journal-of-Application-or-Innovation-in-Engineering-Management-IJAIEM.pdf (дата обращения 17.10.2022).
7. World Payments Report 2023 // Capgemini Research Institute. URL: https://www.capgemini.com/insights/research-library/world-payments-report/ (дата обращения 17.10.2022).
8. Wearables unit shipments worldwide from 2014 to 2028 // Statista. URL: https://www.statista.com/statistics/437871/wearables-worldwide-shipments/ (дата обращения 17.10.2022).
9. van Rijmenam M. Big Data Will Revolutionize Education. // DATAFLOO. April 29, 2013. URL: https://datafloq.com/read/big-data-will-revolutionize-learning/206 (дата обращения 17.10.2022)
10. DB-engines. URL: https://db-engines.com/en/ (дата обращения 17.10.2022)
11. Królikowski Z., Morzy T. Database Systems: from File Systems to Modern Database // Handbook on Data Management in Information Systems. Blaiewicz J., Kubiak W., Morzy T., Rusinkiewicz M. (Eds). Springer, 2003. P. 18–48.
12. Harrison G. Next Generation Databases: NoSQL, NewSQL, and Big Data. Berkeley, CA, USA: Apress, 2015.
13. Liu L., Özsu T.M. (Eds.) Encyclopedia of Database Systems. Second Edition. Springer, 2018.
14. Brown-Syed C. Parents of Invention:The Development of Library Automation Systems in the Late 20th Century. Libraries Unlimited, 2011

15. Kilgour F.G. History of Library Computerization. URL: https://www.researchgate.net/publication/234581818_History_of_Library_Computerization (дата обращения 17.10.2022).
16. Kimball R., Ross M. The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. Third Edition. John Wiley & Sons, Inc., 2013.
17. Vaisman A., Zimányi E. Data Warehouse Systems: Design and Implementation. Springer, 2014
18. Data Warehouse. URL: http://www.tech-faq.com/data-warehouse.html (дата обращения 17.10.2022).
19. Loshin D. Master Data Management. Morgan Kaufmann OMG Press, 2009.
20. Allen M., Cervo D. Multi-Domain Master Data Management: Advanced MDM and Data Governance in Practice. Morgan Kaufmann, 2015.
21. The Forrester Wave™: Master Data Management, Q1 2016: The 12 MDM Providers That Matter Most And How They Stack Up. Goetz M. 16 March 2016.
22. Magic Quadrant for Master Data Management Solutions. Gartner ID G00346112. By Moran M., O'Kane B., Walker S., Parker S., Dayley A. 12 December 2018.
23. Gupta A.P. The characteristics of Big Data platforms and their importance for data exploration. URL: https://towardsdatascience.com/the-characteristics-of-big-data-platforms-and-their-importance-for-data-exploration-650ec45cec14 (дата обращения 17.10.2022).
24. Backaitis V. Thinking about big data platforms in the cloud. URL: https://digitizingpolaris.com/can-yesterdays-hadoop-vendors-beat-the-megaclouds-as-data-platforms-for-machine-learning-ff3411faa656 (дата обращения 17.10.2022).
25. John T., Misra P. Data Lake for Enterprises: Leveraging Lambda Architecture for building Enterprise Data Lake. Packt Publishing, 2017.
26. Gupta S., Giri V. Practical Enterprise Data Lake Insights: Handle Data-Driven Challenges in an Enterprise Big Data Lake. Berkeley, CA, USA: Apress, 2018.
27. Sukhobokov A.A., Baklikov V., Lakhvich D.S., Sukhobokov A.V., Tikhonov I.V. An Approach to Improvement of Master's and PhD Studies in Data Processing and Management Systems // Handbook of Research on Engineering Education in a Global Context . IGI Global, 2019. P. 138–153.
28. Gates M. Blockchain: Ultimate guide to understanding blockchain, bitcoin, cryptocurrencies, smart contracts and the future of money. CreateSpace Independent Publishing Platform, 2017.
29. Cryptocurrencies: growing in number but falling in value. URL: https://graphics.reuters.com/CRYPTO-CURRENCIES-CONFLICTS/0100818S2BW/index.html (дата обращения 17.10.2022).
30. Morabito V. Business Innovation Through Blockchain: The B³ Perspective. Springer, 2017.
31. Shen C., Pena-Mora F. Blockchain for Cities – A Systematic Literature Review // IEEE Access. 12 November 2018. V. 6. P. 76787–76819. DOI 10.1109/ACCESS.2018.2880744.
32. Ehrlinger L., Wöß W. Towards a Definition of Knowledge Graphs // Joint Proceedings of the Posters and Demos Track of 12th Int. Conf. on Semantic Systems - SEMANTiCS2016 and 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS16). Leipzig, Germany. V. 1695. URL: http://ceur-ws.org/Vol-1695/paper4.pdf (дата обращения 17.10.2022).
33. Resource Description Framework (RDF). URL: https://www.w3.org/RDF/ (дата обращения 17.10.2022).
34. Mittal S., Joshi A., Finin T. Thinking Fast, Thinking Slow! Combining Knowledge Graphs and Vector Spaces // arXiv:1708.03310v2 [cs.AI]. 21 Aug 2017.
35. Sap M., LeBras R., Allaway E., Bhagavatula C., Lourie N., Rashkin H., Roof B., Smith N., Choi Y. ATOMIC: An Atlas of Machine Commonsense for If-Then Reasoning // arXiv:1811.00146v3 [cs.CL]. 7 Feb 2019.
36. Daniel, E.D., Mee, C.D., Clark, M.H. (Eds.) Magnetic Recording: The First 100 Years. New York, New York: IEEE Press, 1999.

37. Welsh, H.F., Lukoff, H. The Uniservo - Tape Reader and Recorder // Proceedings of the Joint AZEE-IRE-ACM Computer Conference (Review of Input and Output Equipment used in Computer Systems). New York, Dec. 10–12, 1952. P. 47–53. DOI 10.1109/AFIPS.1952.28.
38. IBM 350 disk storage unit // IBM. URL: https://www.ibm.com/ibm/history/exhibits/storage/storage_350.html (дата обращения 17.10.2022).
39. IBM System/360 Operating System. Concepts and Facilities. File Number: 8360-36. Form C28-6535-0 // IBM. URL: http://www.bitsavers.org/pdf/ibm/360/os/R01-08/C28-6535-0_OS360_Concepts_and_Facilities_1965.pdf (дата обращения 17.10.2022).
40. OS/VS2 Planning and Use Guide. VS2 Release 1. GC28-0600-2 // IBM. URL: http://bitsavers.informatik.uni-stuttgart.de/pdf/ibm/370/OS_VS2/Release_1_1972/GC28-0600-2_OS_VS2_Planning_and_Use_Guide_Rel_1_Sep72.pdf (дата обращения 17.10.2022).
41. OS/VS Virtual Storage Access Method (VSAM) Planning Guide. GC26-3799-0 // IBM. URL: http://bitsavers.informatik.uni-stuttgart.de/pdf/ibm/370/OS_VS2/Release_1_1972/GC26-3799-0_OS_VS_Virtual_Storage_Access_Method_VSAM_Planning_Guide_Jul72.pdf (дата обращения 17.10.2022).
42. Information Management System Virtual Storage (IMS/VS). General Information Manual. GH20-1260-0 // IBM. URL: http://bitsavers.informatik.uni-stuttgart.de/pdf/ibm/370/IMS_VS/GH20-1260-0_IMS_VS_General_Information_Manual_Jan73.pdf (дата обращения 17.10.2022).
43. Types of IMS databases. IBM IMS Documentation // IBM. URL: https://www.ibm.com/support/knowledgecenter/SSEPH2_15.1.0/com.ibm.ims15.doc.dag/ims_dbtypes.htm (дата обращения 17.10.2022).
44. Harding W.B., Clark C.M., Gallo C.L., Tang H. Object storage hierarchy management // IBM Systems Journal. V. 29. № 3. P. 384–397. DOI 10.1147/sj.293.0384.
45. Barnard G.A.III, Fein L. Organization and retrieval of records generated in a large-scale engineering project // Eastern joint computer conference “Modern computers: objectives, designs, applications”. AIEE-ACM-IRE '58 (Eastern). December 03–05, 1958. P. 59–63. DOI 10.1145/1458043.1458058
46. Varian M. VM and the VM Community: Past, Present, and Future // Office of Computing and Information Technology, Princeton University, Princeton, NJ, 1997. URL: https://www.cs.tufts.edu/~nr/cs257/archive/melinda-varian/neuvm.pdf (дата обращения 17.10.2022).
47. IBM Virtual Machine Facility /370: System Programmer's Guide. File No. S370-37, Order No. GC20-1807-3. Release 2 PLC 13 // IBM. URL: http://www.bitsavers.org/pdf/ibm/370/VM/370/Release_2/GC20-1807-3_VM370_System_Programmers_Guide_Rel_2_Jan75.pdf (дата обращения 17.10.2022).
48. Edwards B. What Was CP/M, and Why Did It Lose to MS-DOS? // Howtogeek. Mar 22, 2021. URL: https://www.howtogeek.com/718124/what-was-cpm-and-why-did-it-lose-to-ms-dos/ (дата обращения 17.10.2022).
49. Creasy R.J. The origin of the VM/370 time-sharing system // IBM Journal of Research & Development. September 1981. V. 25. № 5. P. 483–490. URL: http://pages.cs.wisc.edu/~stjones/proj/vm_reading/ibmrd2505M.pdf (дата обращения 17.10.2022).
50. Introduction to the IBM 3850 Mass Storage System (MSS). GA32-0028-2. File No. S370-07. Third Edition (July 1975) // IBM. URL: http://bitsavers.org/pdf/ibm/3850/GA32-0028-7_Introduction_to_the_IBM_3850_Mass_Storage_System_Jul75.pdf (дата обращения 17.10.2022).
51. Patterson D.A., Gibson G., Katz R.H. A Case for Redundant Arrays of Inexpensive Disks (RAID) // Proceedings of the 1988 ACM SIGMOD international conference on Management of data. June 1988. P. 109–116. DOI 10.1145/50202.50214.

52. Hierarchical Storage Management // The Fractal Structure of Data Reference. Advances in Database Systems, V. 22. Springer, Boston, MA. DOI 10.1007/0-306-47034-9_8.
53. CSIRO Computing History // Csiropedia. URL: https://csiropedia.csiro.au/csiro-computing-history/ (дата обращения 17.10.2022).
54. Data Facility Hierarchical Storage Manager. Version 2. Release 5.0 Planning Guide. Order Number GC35-0109-2. Program Number 5665–329 // IBM. URL: http://bitsavers.org/pdf/ibm/370/MVS_XA/DFHSM/GC35-0109-2_Data_Facility_Heirarchical_Storage_Manager_Version_2_Release_5.0_Planning_Guide_Jul1989.pdf (дата обращения 17.10.2022).
55. Wodrow T.S. Hierarchical storage management system evaluation // Third NASA Goddard Conference on Mass Storage Systems and Technologies. NASA, 1994. P. 187–216. https://ntrs.nasa.gov/api/citations/19940029285/downloads/19940029285.pdf (дата обращения 17.10.2022).
56. Hierarchical Storage Management. Windows Server Brain // Server Brain. URL: https://www.serverbrain.org/administration-practice-2003/hierarchical-storage-management-1.html (дата обращения 17.10.2022).
57. HSM for Windows client overview. IBM Spectrum Protect HSM for Windows // IBM. URL: https://www.ibm.com/docs/en/sphfw/8.1.11?topic=hsm-windows-client-overview (дата обращения 17.10.2022).
58. Shimpi A.L. Understanding Apple's Fusion Drive // AnandTech. October 24, 2012. URL: https://www.anandtech.com/show/6406/understanding-apples-fusion-drive (дата обращения 17.10.2022).
59. Beigi M., Devarakonda M., Jain R., Kaplan M., Pease D., Rubas J., Sharma U., Verma A. Policy-based information lifecycle management in a large-scale file system // Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05). DOI 10.1109/POLICY.2005.26.
60. Schroder A., Fritzsche R., Schmidt S., Mitschick A., Meißner K. A Semantic Extension of a Hierarchical Storage Management System for Small and Medium-sized Enterprises // Proceedings of the 1st International Workshop on Semantic Digital Archives (SDA 2011). P. 23–36. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c313476a041ee84eb81e5eaa1adcc46cdbd389e5#page=25 (дата обращения 17.10.2022).
61. Lakshmi T.G., Sedamkar R.R., Patil H. Reinforcement Learning Approach for Data Migration in Hierarchical Storage Systems // International Journal of Enhanced Research in Management & Computer Applications. Nov.–Dec. 2013 V. 2. № 9. P. 30–35. http://www.erpublications.com/uploaded_files/download/download_21_12_2013_05_23_40.pdf (дата обращения 17.10.2022).
62. Tridgell A. Myths About Samba // Samba. URL: https://www.samba.org/samba/docs/myths_about_samba.html (дата обращения 17.10.2022).
63. White Jr.R.B. Hardware Independence // Surfing the High Tech Wave: A history of Novell Inc. from 1980 to 1990. Chapter 04. Chapter section. Cyreenik Says, 2010. URL: http://whiteworld.com/cyreenikland/books/surfing/surf-04b.htm (дата обращения 17.10.2022).
64. Satyanarayanan M., Howard J.H., Nichols D.A., Sidebotham R.N., Spector A.Z., West M.J. The ITC distributed file system: Principles and design // ACM SIGOPS Operating Systems Review. Dec. 1-4, 1985. V. 19. № 5. P. 35–50. DOI 10.1145/323627.323633.
65. Thanh T.D., Mohan S., Choi E., Kim S., Kim P. A Taxonomy and Survey on Distributed File Systems // 2008 Fourth International Conference on Networked Computing and Advanced Information Management. DOI 10.1109/NCM.2008.162.
66. Hellwagner H. Design considerations for scalable parallel file systems // The Computer Journal. 1993. V. 36. № 8. P. 741–755. DOI 10.1093/comjnl/36.8.741.

67. Ross R., Carns P., Metheny D. (2009) Parallel File Systems. // Chan Y., Talburt J., Talley T. (Eds.) Data Engineering. International Series in Operations Research & Management Science. 2009. Vol. 132. Springer, Boston, MA. DOI 10.1007/978-1-4419-0176-7_8.
68. Soltis R.S., Erickson G.M., Preslan K.W., O'Keefe M.T., Ruwart T.M. The Global File System: A File System for Shared Disk Storage // University of Minnesota. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=50b198b87dc4e5ae103521135fa410f594573d3f (дата обращения 17.10.2022).
69. Herodotou H. Towards a distributed multi-tier file system for cluster computing // 2016 IEEE 32nd International Conference on Data Engineering Workshops (ICDEW). DOI 10.1109/ICDEW.2016.7495633.
70. Hasan R., Anwar Z., Yurcik W., Brumbaugh L., Campbell R. A survey of peer-to-peer storage techniques for distributed file systems // International Conference on Information Technology: Coding and Computing (ITCC'05). Vol. II. DOI 10.1109/ITCC.2005.42.
71. Khan I., Dewangan B., Meena A., Birthare M. Study of Various Cloud Service Providers: A Comparative Analysis // 5th International Conference on "Next Generation Computing Technologies" (NGCT 2019). URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3672950 (дата обращения 17.10.2022).
72. Shalima N.S., Shamna H.R. An Analysis of Cloud Backed File System and Backup Algorithms // International Journal for Research in Applied Science & Engineering Technology. V. 8. № VI. June 2020. P. 1034–1041. DOI 10.22214/ijraset.2020.6168.
73. Beregi R., Pedone G., Mezgár I. A novel fluid architecture for cyberphysical production systems // International Journal of Computer Integrated Manufacturing. 2019. V. 32. № 4–5: Smart Cyber-Physical System Applications in Production and Logistics. P. 340–351. DOI 10.1080/0951192X.2019.1571239.
74. Scotece D., Paiker N.R., Foschini L., Bellavista P., Ding X., Borcea C. MEFS: Mobile Edge File System for Edge-Assisted Mobile Apps // 2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM). DOI 10.1109/WoWMoM.2019.8792987.
75. Copstein R.N., Dotti F.L. Distributed File System for an Edge-Based Environment // arXiv:2006.09182v1 [cs.DC]. 16 Jun 2020.
76. Xia Q., Liang W., Xu Z., Zhou B. Online Algorithms for Location-Aware Task Offloading in Two-Tiered Mobile Cloud Environments // 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing. DOI 10.1109/UCC.2014.19.
77. Gopalakrishnan S., Arumugam A., Iftode L. Federated File Systems for Clusters with Remote Memory Communication // USENIX FAST Conference. 2002. URL: https://www.usenix.org/legacy/publications/library/proceedings/fast02/wips/gopalakrishnan.pdf (дата обращения 17.10.2022).
78. Ahn S.U., Yoon H.J., Park S.O. Storage Federations Using Xrootd // International Journal of Multimedia and Ubiquitous Engineering. 2015. V. 10. № 11. P. 285–292. DOI 10.14257/ijmue.2015.10.11.27.
79. Gait J. Phoenix: A Safe In-Memory File System // Communications of the ACM. V. 33, № 1. Jan. 1990. P. 81–86. DOI 10.1145/76372.76378.
80. Snyder P. tmpfs: A Virtual Memory File System // Autumn 1990 European Unix Systems User Group (EUUG) Conference. Oct. 1990. P. 241–248. URL: http://www.sunhelp.org/history/pdf/tmpfs.pdf (дата обращения 17.10.2022).
81. Kim H., Ahn J., Ryu S., Choi J., Han H. In-memory file system for non-volatile memory // RACS '13: Proceedings of the 2013 Research in Adaptive and Convergent Systems. Oct. 2013. P. 479–484. DOI 10.1145/2513228.2513325.
82. Sha E.H.-M., Chen X., Zhuge Q., Shi L., Jiang W. A new design of in-memory file system based on file virtual address framework // IEEE Transactions on Computers. 2016. V. 65. № 10. P. 2959–2972. DOI 10.1109/TC.2016.2516019.

83. Huang H., Lin J., Zheng B., Zheng Z., Bian J. When Blockchain Meets Distributed File Systems: An Overview, Challenges, and Open Issues // IEEE Access. 2020. V. 8. P. 50574–50586. DOI 10.1109/ACCESS.2020.2979881.
84. IPFS Docs. Content Identifiers (CIDs) // IPFS. URL: https://docs.ipfs.tech/concepts/content-addressing/#what-is-a-cid (дата обращения 17.10.2022).
85. Gifford D.K., Jouvelot P., Sheldon M.A., O'Toole Jr.J.W. Semantic file systems // ACM SIGOPS operating systems review. V. 25. № 5. P. 16–25. DOI 10.1145/121133.121138.
86. Lakshman A., Malik P. Cassandra: A Decentralized Structured Storage System // ACM SIGOPS Operating Systems Review. 2010. V. 44. № 2. P. 35–40. DOI 10.1145/1773912.1773922.
87. Luciani J. Cassandra File System Design // DataStax. February 11, 2012. URL: https://www.datastax.com/blog/cassandra-file-system-design (дата обращения 17.10.2022).
88. Green W.B. Introduction to electronic document management systems. California, San Diego: Academic Press, Inc., 1993.
89. Adam A. Implementing Electronic Document and Record Management Systems. Florida, Boca Raton: Taylor & Francis Group, 2008.
90. Won J.S. Overview and Recommendations for Distributed File Systems // DZone. URL: https://dzone.com/articles/overview-and-recommendations (дата обращения 17.10.2022)
91. Weil S.A., Brandt S.A., Miller E.L, Long D.D.E., Maltzahn C. Ceph: A Scalable, High-Performance Distributed File System // OSDI ’06: 7th USENIX Symposium on Operating Systems Design and Implementation. P. 307–320. URL: http://static.usenix.org/event/osdi06/tech/full_papers/weil/weil.pdf (дата обращения 17.10.2022).
92. Brim M.J., Dillow D.A, Oral S., Settlemyer B.W., Wang F. Asynchronous Object Storage with QoS for Scientific and Commercial Big Data // 8th Parallel Data Storage Workshop. November 18, 2013. Denver, CO. URL: http://www.pdsw.org/pdsw13/papers/p7-pdsw13-brim.pdf (дата обращения 17.10.2022).
93. List of file systems // Wikipedia. URL: https://en.wikipedia.org/wiki/List_of_file_systems#Distributed_file_systems (дата обращения 17.10.2022).
94. Depardon B., Le Mahec G., Seguin C. Analysis of Six Distributed File Systems. Research Report <hal-00789086> // HAL 2013. 44 p. URL: https://hal.inria.fr/hal-00789086/PDF/a_survey_of_dfs.pdf (дата обращения 17.10.2022).
95. Donvito G., Marzulli G., Diacono D. Testing of several distributed file-systems (HDFS, Ceph and GlusterFS) for supporting the HEP experiments analysis // 20th International Conference on Computing in High Energy and Nuclear Physics (CHEP2013). IOP Publishing. Journal of Physics. 2014. Conference Series 513. Статья 042014. URL: http://iopscience.iop.org/1742-6596/513/4/042014/pdf/1742-6596_513_4_042014.pdf (дата обращения 17.10.2022).
96. Рахматуллин Д.Я. Введение в MPI // ИМВЦ УНЦ РАН. Уфа. URL: http://matem.anrb.ru/sites/default/files/publications/2006._rahmatullin_d.ya_._vvedenie_v_mpi._ufa.pdf (дата обращения 17.10.2022).
97. Немнюгин С.А. Введение в программирование на кластерах. Лекция 2: Программирование с использованием Intel MPI. Введение. URL: http://www.intuit.ru/studies/courses/4448/984/lecture/14935 (дата обращения 17.10.2022).
98. MPI Forum. URL: http://mpi-forum.org/ (дата обращения 17.10.2022).
99. Евсеев И. MPI для начинающих // ИВВиБД (ЦСТ). URL: https://norman.jiht.ru/student/morozov/MPI_for_beginners.htm (дата обращения 17.10.2022).
100. Интерфейс MPI. URL: http://pro-spo.ru/parallel/2151--mpi (дата обращения 17.10.2022).
101. Harrison G. Next Generation Databases: NoSQL, NewSQL, and Big Data. Berkeley, CA, USA: Apress, 2015. 256 p.
102. Олле Т.В. Предложения КОДАСИЛ по управлению базами данных. М.: Финансы и статистика, 1981. 286 с.

103. Уайт Т. Hadoop. Подробное руководство. СПб.: Питер, 2013. 672 с.: ил. (Серия «Бестселлеры O’Reilly»).
104. Cafarella M., Cutting D. Building Nutch: Open Source Search: A case study in writing an open source search engine // ACM Queue. 2004. V. 2. № 2. P. 54–61. DOI 10.1145/988392.988408.
105. Ghemawat S., Gobioff H., Leung S.-T. The Google File System // Google. October 2003. URL: http://static.googleusercontent.com/media/research.google.com/en/us/archive/gfs-sosp2003.pdf (дата обращения 17.10.2022).
106. Dean J., Chemawat S. MapReduce: Simplified Data Processing on Large Clusters // Communications of the ACM. 2008. V. 51. № 1. P. 107–113. DOI 10.1145/1327452.1327492.
107. Yahoo! Launches World’s Hadoop Production Application // Yahoo!, February 19, 2008. URL: https://lucene-group-group.iteye.com/group/topic/4244 (дата обращения 17.10.2022).
108. Емельянов И. Как обновление Hadoop 2.0 сделало «большие данные» доступнее для бизнеса // Interface.ru. URL: https://www.interface.ru/home.asp?artId=35027 (дата обращения 17.10.2022).
109. Apache Ozone™ // Apache.org. URL: https://ozone.apache.org/ (дата обращения 20.11.2024).
110. Kerzner M., Maniyam S. Chapter 12. Big Data Ecosystem // hadoop illuminated, 2014. URL: http://hadoopilluminated.com/hadoop_illuminated/Bigdata_Ecosystem.html (дата обращения 20.11.2024).
111. Powlas T. Big Data in an SAP Landscape – ASUG Webcast Part 1. // SAP Community Network. 29.12.2013. URL: http://scn.sap.com/community/business-intelligence/blog/2013/12/29/big-data-in-an-sap-landscape-asug-webcast-part-1 (дата обращения 20.11.2024).
112. The Hadoop Ecosystem Table // GitHub, URL: http://hadoopecosystemtable.github.io/ (дата обращения 20.11.2024).
113. Сухобоков А.А., Лахвич Д.С. Влияние инструментария Big Data на развитие научных дисциплин, связанных с моделированием // Наука и Образование. МГТУ им. Н.Э. Баумана. 2015. № 03. С. 207–240.
114. Hortonworks // wikipedia.org URL: https://en.wikipedia.org/wiki/Hortonworks (дата обращения 20.11.2024).
115. Hortonworks // bigdataschool.ru. URL: https://bigdataschool.ru/wiki/hortonworks (дата обращения 20.11.2024).
116. Cloudera DataFlow, formerly Hortonworks DataFlow // cloudera.com. URL: https://www.cloudera.com/downloads/cdf.html (дата обращения 20.11.2024).
117. Cloudera Products // cloudera.com. URL: http://www.cloudera.com/products.html (дата обращения 20.11.2024).
118. MapR // bigdataschool.ru. URL: https://bigdataschool.ru/wiki/mapr (дата обращения 20.11.2024).
119. VMware Tanzu Platform // vmware.com. URL: https://www.vmware.com/products/app-platform/tanzu (дата обращения 20.11.2024).
120. Sukhobokov A.A., Baklikov V., Lakhvich D.S., Sukhobokov A.V., Tikhonov I.V. An Approach to Improvement of Master's and PhD Studies in Data Processing and Management Systems. // Handbook of Research on Engineering Education in a Global Context. P. 138–153. IGI Global. 2019.
121. Yuan L., Yanmei L., Minjing Z. Technical architecture of integrated big data platform // 2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2022. P. 1551–1556.
122. Oracle: Big Data for the Enterprise // Oracle.com. October 2011. URL: https://www.oracle.com/technetwork/database/bi-datawarehousing/wp-big-data-with-oracle-521209.pdf (дата обращения 20.11.2024).
123. Big Data // Teradata.ru. URL: http://teradata.ru/products-and-services/integrated-big-data-platform (дата обращения 20.11.2024).
124. IBM Big Data Platform // Tdwi.org. URL: https://tdwi.org/-/media/692A428D271F4D648BF6732EF0120EC0.PDF (дата обращения 20.11.2024).

125. Magic Quadrant for Data Management Solutions for analytics // Gartner.com. February 20, 2017. ID: G00302535. Analysts: R. Edjlali, A. M. Ronthal, R. Greenwald, M. A. Beyer, D. Feinberg.
126. Low Y., Gonzalez J.E., Kyrola A., Bickson D., Guestrin C.E., Hellerstein J. Graphlab: A new framework for parallel machine learning // arXiv, 9 Aug 2014. arXiv preprint arXiv:1408.2041.
127. Lyubimov D., Palumbo A. Apache Mahout: Beyond MapReduce. Distributed Algorithm Design. CreateSpace Independent Publishing Platform. 232 p.
128. Zakharia M. Spark's Role in the Big Data Ecosystem // Spark Summit. San Francisco, June 30, 2014. URL: https://www.youtube.com/watch?v=e-Ys-2uVxM0 (дата обращения 20.11.2024).
129. Zakharia M. Introduction to Spark // Databrics. Databrics Intern Event. August 19, 2015. URL: http://www.slideshare.net/databricks/introduction-to-spark-intern-event-presentation (дата обращения 20.11.2024).
130. Apache Spark // Databricks.com. URL: https://www.databricks.com/glossary/what-is-apache-spark (дата обращения 20.11.2024).
131. Карау Х., Конвински Э., Венделл П., Захария М. Изучаем Spark: молниеносный анализ данных. М.: ДМК Пресс, 2015. 304 с.
132. SAP HANA® Database for Next-Generation Business Applications and Real-Time Analytics // SAP AG. 10.10.2013. URL: https://www.domain-group.com/wp-content/uploads/2012/09/SAP-HANA-for-Next-Generation-Business-Applications-and-Real-Time-Analytics.pdf (дата обращения 20.11.2024).
133. SAP HANA Platform 2.0 SPS 08 // SAP Help Portal. URL: https://help.sap.com/docs/SAP_HANA_PLATFORM?locale=en-US (дата обращения 20.11.2024).
134. Färber F., May N., Lehner W., Große P., Müller I., Rauhe H., Dees J. The SAP HANA Database – An Architecture Overview // IEEE Data Engineering Bulletin. 2012. V. 35. № 1. P. 28–33.
135. HA100 SAP HANA Introduction // SAP AG. Course Version: 99. Participant Handbook, Material Number: 50119238. 329 p.
136. Sauer K.-P. SAP HANA in Data Centers Update // SAP SE, April 2016. URL: http://www.sapevents.edgesuite.net/atsapsummit2016/2016/pdfs/20160427_SAP_HANA_in_DataCenters_HPSauer.pdf (дата обращения 20.11.2024).
137. IBM FlashSystem max configuration with SAP HANA // IBM. 2020. URL: https://www.ibm.com/support/pages/system/files/inline-files/IBM_FlashSystem_max_config_SAP%20HANA.pdf (дата обращения 20.11.2024).
138. Rapid Development of SAP Sybase Event Stream Processor Applications // SAP AG. TechEd 2013. Amsterdam. October 22-25. Session RDP279.
139. Свинарёв С. Зачем Сбербанку GridGain In-Memory Data Fabric? // PC Week/RE. 18.01.2016. URL: http://www.pcweek.ru/infrastructure/blog/infrastructure/8168.php (дата обращения 20.11.2024).
140. SAP Vora // Help.SAP.com. URL: https://help.sap.com/docs/SAP_VORA?locale=en-US (дата обращения 20.11.2024).
141. Big Data Overview: презентация IBM на семинаре Big Data & Analytics Day, Москва, офис IBM, 27 января 2015 года
142. Data lake definition // TechTarget. URL: http://searchaws.techtarget.com/definition/data-lake (дата обращения 20.11.2024).
143. Azzabi S., Alfughi Z., Ouda A. Data Lakes: A Survey of Concepts and Architectures // Computers. 2024. V. 13. № 7. Article 183. URL: https://www.mdpi.com/2073-431X/13/7/183 (дата обращения 20.11.2024).
144. Alam A. Why Hadoop is used for Big Data Analysis? // Medium.com. Jun 25, 2023. URL: https://medium.com/@azfaralam/why-hadoop-is-used-for-big-data-analysis-4e1907fa4db5 (дата обращения 20.11.2024).
145. Hechler E., Weihrauch M., Wu Y. Evolution of Data Architecture // Data Fabric and Data Mesh Approaches with AI. Berkeley, P. 3–15. Berkeley, CA, USA: Apress, 2023. DOI 10.1007/978-

1-4842-9253-2_1. URL: https://link.springer.com/chapter/10.1007/978-1-4842-9253-2_1#citeas (дата обращения 20.11.2024).
146. Gupta S., Giri V. Practical Enterprise Data Lake Insights: Handle Data-Driven Challenges in an Enterprise Big Data Lake. Berkeley, CA, USA: Apress, 2018.
147. Farmer D. 5 principles of a well-designed data architecture // TechTarget. 20 Jul 2021. URL: https://www.techtarget.com/searchdatamanagement/tip/5-principles-of-a-well-designed-data-architecture (дата обращения 20.11.2024).
148. Donepudi K. "Data Lake", do you need one? // Linkedin.com. 20.11.2016. URL: https://www.linkedin.com/pulse/why-data-lake-kiran-donepudi (дата обращения 20.11.2024).
149. Inmon B. Data Lake Architecture: Designing the Data Lake and avoiding the garbage dump. Denville, NJ, United States: Technics Publications, LLC. 2016. 166 p.
150. LaPlante A., Sharma B. Architecting data lakes: data management architectures for advanced business use cases. Sebastopol, USA: O'Reilly Media, Inc. 2018. 50 p.
151. Hai R., Koutras C., Quix C., Jarke M. Data lakes: A survey of functions and systems // IEEE Transactions on Knowledge and Data Engineering. 2023. V. 35. № 12. P. 12571–12590. DOI 10.1109/TKDE.2023.3270101.
152. Sawadogo P., Darmont J. On data lake architectures and metadata management // Journal of Intelligent Information Systems. 2021. V. 56. № 1. P. 97–120. DOI 10.1007/s10844-020-00608-7.
153. Oukhouya L., El haddadi A., Er-raha B., Asri H., Sbai A. Designing Hybrid Storage Architectures with RDBMS and NoSQL Systems: A Survey // International Conference on Advanced Intelligent Systems for Sustainable Development. Cham, Switzerland: Springer Nature Switzerland, 2023. V. 637. P. 332–343. DOI 10.1007/978-3-031-26384-2_29.
154. Armbrust M., Ghodsi A., Xin R., Zaharia M. Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics // Proceedings of CIDR. 2021. URL: https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf (дата обращения 24.08.2024).
155. Harby A.A., Zulkernine F. From data warehouse to lakehouse: A comparative review // 2022 IEEE International Conference on Big Data (Big Data): IEEE, 2022. P. 389–395. DOI 10.1109/BigData55660.2022.10020719.
156. Curry E., Scerri S., Tuikka T. Data Spaces: Design, Deployment and Future Directions // Data Spaces: Design, Deployment and Future Directions. Cham, Switzerland: Springer International Publishing, 2022. P. 1–17. DOI 10.1007/978-3-030-98636-0_1.
157. Jarke M., Quix C. On warehouses, lakes, and spaces: the changing role of conceptual modeling for data integration // Conceptual Modeling Perspectives. Cham, Switzerland: Springer International Publishing, 2017. P. 231–245. DOI 10.1007/978-3-319-67271-7_16.
158. Otto B. The evolution of data spaces // Designing data spaces: The ecosystem approach to competitive advantage. Cham, Switzerland: Springer International Publishing, 2022. P. 3–15. DOI 10.1007/978-3-030-93975-5_1.
159. Anjomshoaa A., Ghodsi A., Xin R., Zaharia M. Data platforms for data spaces // Data Spaces: Design, Deployment and Future Directions. Cham, Switzerland: Springer International Publishing, 2022. P. 43–64. DOI 10.1007/978-3-030-98636-0_3.
160. Sharma V., Balusamy B., Thomas J.J., Atlas L.G. Data Fabric Architectures: Web-Driven Applications. Berlin, Boston: Walter de Gruyter GmbH & Co KG, 2023. 200 p. DOI 10.1515/9783111000886.
161. Hechler E., Weihrauch M., Wu Y. Data Fabric and Data Mesh Approaches with AI: A Guide to AI-based Data Cataloging, Governance, Integration, Orchestration, and Consumption. Berkeley, CA, USA: Apress, 2023. 427 p. DOI 10.1007/978-1-4842-9253-2.
162. Serra J. Deciphering Data Architectures: Choosing Between a Modern Data Warehouse, Data Fabric, Data Lakehouse and Data Mesh. Sebastopol, USA: O'Reilly Media, Inc., 2024. 275 p.
163. Dehghani Z., Fowler M. Data Mesh: Delivering Data-driven Value at Scale. Sebastopol, USA: O'Reilly Media, Inc., 2022. 345 p.

164. Сухобоков А.А. Перспективы развития систем обработки информации и управления в следующем десятилетии (2020–2030 гг.) // Информационно-измерительные и управляющие системы. 2016. № 12. С. 69–78.
165. Sukhobokov A.A. Business analytics and AGI in corporate management systems // Procedia computer science. 2018. V. 145. P. 533–544. DOI 10.1016/j.procs.2018.11.118.
166. Gartner IT Market Clock for ERP 2020 – Preparing for the 4th Generation of EBC // Gartner. 2020, 28 Feb. ID: G00465788. Analysts: Nguyen D., Schenck P.
167. SAP SE IOT100. Internet of Things Fundamentals // SAP SE. 2017. Course Version: 10. Material Number: 50139413.
168. Jonathan A., Ryden M., Oh K., Chandra A., Weissman J. Nebula: Distributed edge cloud for data intensive computing // IEEE Transactions on Parallel and Distributed Systems. 2017. V. 28. № 11. P. 3229–3242. DOI 10.1109/TPDS.2017.2717883.
169. Ryden M., Oh K., Chandra A., Weissman J. Nebula: Distributed edge cloud for data intensive computing // 2014 IEEE International Conference on Cloud Engineering P. 57–66. DOI 10.1109/IC2E.2014.34.
170. Ferrer A.J., Marquès J.M., Jorba J. Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing // ACM Computing Surveys. 2019. V. 51. № 6. Статья 111. P. 1–36. DOI 10.1145/3243929.
171. Naha R.K., Garg S., Chan A. (2018). Fog computing architecture: Survey and challenges. arXiv:1811.09047v1 [cs.DC]. 22 Nov 2018.
172. Fu Y., Qiu X., Wang J. F2MC: Enhancing data storage services with fog-toMultiCloud hybrid computing // 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC). DOI 10.1109/IPCCC47392.2019.8958748.
173. Bergui M., Najah S., Nikolov N.S. A survey on bandwidth-aware geo-distributed frameworks for big-data analytics // Journal of Big Data. 2021. V. 8. № 1. P. 1–26. DOI 10.1186/s40537-021-00427-9.
174. Google File System (GFS) vs. Hadoop Distributed File System (HDFS) // geeksforgeeks.org. 03.10.2024. URL: https://www.geeksforgeeks.org/google-file-system-gfs-vs-hadoop-distributed-file-system-hdfs/ (дата обращения 20.11.2024).
175. Lovett M. Apache Hadoop: The Open Source Elephant of Big Data // Trenton Systems. 06.07.2012. URL: https://www.trentonsystems.com/news/apache-hadoop-the-open-source-elephant-of-big-data (дата обращения 03.02.2015)
176. Popescu A., Bacalu A. Life of Data at Facebook // nosql.mypopescu.com. 03.09.2012. URL: http://nosql.mypopescu.com/post/30815314471/life-of-data-at-facebook (дата обращения 20.11.2024).
177. Evans B., Graves T. Yahoo talks about Spark vs. Storm // youtube.com. 19.09.2014. URL: https://www.youtube.com/watch?v=uJ5rdAPHE1w (дата обращения 20.11.2024).
178. Singh S. Apache HBase at Yahoo! – Multi-tenancy at the Helm Again // slideshare.net. 07.06.2015. URL: https://www.slideshare.net/slideshow/hbasecon-2013-multitenant-apache-hbase-at-yahoo/50238617 (дата обращения 20.11.2024).
179. Zicari R.V. Hadoop at Yahoo. Interview with Mithun Radhakrishnan // ODBMS Industry Watch. 21.09.2014. URL: http://www.odbms.org/blog/2014/09/interview-mithun-radhakrishnan/ (дата обращения 20.11.2024).
180. Henschen D. Yahoo Talks Apache Storm: Real-Time Appeal // informationweek.com. 22.10.2014. URL: https://www.informationweek.com/machine-learning-ai/yahoo-talks-apache-storm-real-time-appeal (дата обращения 20.11.2024).
181. Лапань М. Практическое слоноводство // Презентация на семинаре Hadoop Kitchen в Mail.Ru Group. 27 сентября 2014 г., URL: http://www.youtube.com/watch?v=QAejaeTvj_M начало с 1:20:40 (дата обращения 20.11.2024).

182. Kiran M., Murphy P., Monga I., Dugan J., Baveja S.S. Lambda architecture for cost-effective batch and speed big data processing // 2015 IEEE international conference on Big Data (Big Data) P. 2785–2792. IEEE, 2015. DOI 10.1109/BigData.2015.7364082.
183. Marz N., Warren J. Big Data. Principles and best practices of scalable realtime data systems. — NY:Manning Publications Co., 2015. 328 p.
184. Burdett D., Tripathi R. CIO Guide. How to Use Hadoop with Your SAP® Software Landscape // SAP AG. Февраль 2013 г., URL: http://hortonworks.com/wp-content/uploads/2013/09/CIO.Guide_.How_.to_.Use_.Hadoop.with_.Your_.SAP_.Software.Landscape.pdf (дата обращения 20.11.2024).
185. Powlas T. Using HANA and Hadoop, Key Scenarios - Part 2 ASUG Big Data Webcast // SAP Community Network. 29.12.2013. URL: http://scn.sap.com/community/business-intelligence/blog/2013/12/29/using-hana-and-hadoop-key-scenarios (дата обращения 20.11.2024).
186. Powlas T. Fitting Hadoop in an SAP Software Landscape – Part 3 ASUG Webcast // SAP Community Network. 29.12.2013. URL: http://scn.sap.com/community/business-intelligence/blog/2013/12/29/fitting-hadoop-in-an-sap-software-landscape-part-3-asug-webcast (дата обращения 20.11.2024).
187. SAP HANA Hadoop Integration. 2.0 SPS07 // Help.SAP.com. URL: https://help.sap.com/docs/SAP_HANA_SPARK_CONTROLLER/c618c36d79e34077a680bac61affc8b7/286287b48712457e8b58b6e05f193a7c.html?locale=en-US&q=MapReduce (дата обращения 20.11.2024).
188. World’s Largest Data Warehouse record -12.1 PB from SAP // Pat Research. URL: https://www.predictiveanalyticstoday.com/worlds-largest-data-warehouse-record-12-1-pb-from-sap/ (дата обращения 20.11.2024).
189. Кнут Д.Э.: Искусство Программирования. Том 3. Сортировка и Поиск. М.: Вильямс, 2012. 824 с.
190. Воеводин В.В., Воеводин Вл.В., Параллельные вычисления. СПб.: БХВ-Петербург, 2002, 608 с.
191. Analytics and Big Data and ROI…Oh My! // BlueSpire. 16 сентября 2014 г. URL: http://www.slideshare.net/bluespiremarketing/analytics-and-big-data-and-roioh-my-trendlab-webinar (дата обращения 20.11.2024).
192. Кочетов Ю.А., Младенович Н., Хансен П., Локальный поиск с чередующимися окрестностями // Дискретный анализ и исследование операций. 2003. Т. 10. № 1. С. 11–43. URL: http://www.mathnet.ru/links/e09eed4f2e8ea54ad33b6efc32c376a2/da161.pdf (дата обращения 20.11.2024).
193. Ralphs T., Shinano Y., Berthold T., Koch T. Parallel solvers for mixed integer linear optimization // Hamadi Y., Sais L. (Eds.) Handbook of Parallel Constraint Reasoning. P. 283–336. Springer, Cham, 2018. DOI 10.1007/978-3-319-63516-3_8.
194. Смирнов А.Б. Имитационное моделирование в экономике. Модуль 1. Общие вопросы имитационного моделирования. Рабочий учебник. М.: РосНОУ, 2009. 52 с., URL: http://cis.rosnou.ru/UniversysDWNL/Library/D9CA647C-7F45-4069-9623-7018580F554B/D9CA647C-7F45-4069-9623-7018580F554B.pdf (дата обращения 20.11.2024).
195. Макаров В.Л., Бахтизин А.Р., Васенин В.А., Роганов В.А., Трифонов И.А. Средства суперкомпьютерных систем для работы с агент-ориентированными моделями // Программная инженерия. 2011. № 3. С. 2–14, URL: http://iipo.tu-bryansk.ru/fileadmin/user_upload/_content_admins/Journal_PrI/ZHurnal_Programmnaja_Inzhenerija_2011_No_3.pdf (дата обращения 20.11.2024).
196. Fegaras L. Supporting Bulk Synchronous Parallelism in Map-Reduce Queries // University of Texas at Arlington. 2012. URL: http://lambda.uta.edu/mrql-bsp.pdf (дата обращения 20.11.2024).

197. Сухобоков А.А. Исследование и разработка моделей и архитектуры средств контроллинга для межрегиональных предприятий в составе систем класса ERP II: дисс. на соискание уч. ст. канд. техн. наук, М.: МГТУ им. Баумана, 2009, 196 с., Науч. библ. дисс. и автореф. disserCat. URL: http://www.dissercat.com/content/issledovanie-i-razrabotka-modelei-i-arkhitektury-sredstv-kontrollinga-dlya-mezhregionalnykh-#ixzz3O5emDlst (дата обращения 20.11.2024).
198. Маджара Т.И. Интеллектуальная система для решения задач оптимального управления с вычислительными особенностями: дисс. на соискание уч. ст. канд. техн. наук, Владивосток: Институт автоматики и процессов управления ДВО РАН, 2011, 149 с., Науч. библ. дисс. и автореф. disserCat, URL: http://www.dissercat.com/content/intellektualnaya-sistema-dlya-resheniya-zadach-optimalnogo-upravleniya-s-vychislitelnymi-oso#ixzz3O5bV01s9 (дата обращения 20.11.2024).