# Обработка больших данных

Традиционные подходы к обработке данных часто не справляются с современными масштабами. Базы данных, которые отлично обрабатывают 10 ГБ данных, начинают терять производительность при 100 ГБ, а при масштабах в терабайтах требуются совершенно другие архитектурные подходы. Для облачных SaaS-систем понимание обработки больших данных является необходимым условием для создания масштабируемых и производительных приложений.

# Понимание больших данных: "три V"

В отрасли в основном принято определять большие данные с помощью трех фундаментальных характеристик, часто называемых тремя V. Эта определение оказывается полезным при оценке того, действительно ли проблема требует решений для больших данных.

### Объем

Относится к размеру данных — терабайтам, петабайтам или даже эксабайтам. Когда сервисы не справляются с обработкой или данные не влезают в память, а вертикальные подходы к масштабированию исчерпывают себя.

### Скорость

Это скорость, с которой данные поступают и требуют обработки. Часто требуется достаточно быстрый анализ информации, чтобы она оставалась полезной. Например, системы обнаружения мошенничества должны анализировать транзакции за миллисекунды.

### Разнообразие

Касается неоднородности источников и форматов данных. Современные приложения SaaS редко работают исключительно со структурированными данными в таблицах. Логи JSON, данные CSV, двоичные файлы, временные метрики, неструктурированный текст и изображения или видеоконтент — все это требует разных стратегий обработки.

Некоторые специалисты добавляют четвертое V — Veracity (достоверность) — относящееся к качеству и надежности данных, хотя это представляет собой универсальную проблему данных, а не что-то уникальное для больших данных. В больших масштабах даже небольшие проблемы с качеством данных становятся огромными проблемами.

# Основные алгоритмы и парадигмы обработки

Обработка больших данных опирается на элегантные алгоритмы, позволяющие решать казалось бы неразрешимые задачи. Эти проверенные на практике шаблоны лежат в основе всего — от результатов поиска Google до рекомендаций Netflix.

# MapReduce: основа обработки больших данных

MapReduce остается концептуальной основой для распределенной обработки данных в больших масштабах, даже несмотря на то, что реализации вышли за рамки его первоначальной формы. Хотя часто его упрощают до Map и Reduce, на самом деле этот алгоритм включает в себя несколько критически важных этапов, которые делают возможной распределенную обработку.

01

### Этап Map

Данные преобразуются и обрабатываются параллельно на тысячах компьютеров. Каждый маппер обрабатывает части входных данных независимо, выводя пары ключ-значение в качестве промежуточного результата. Полная изоляция исключает необходимость в коммуникации между мапперами.

02

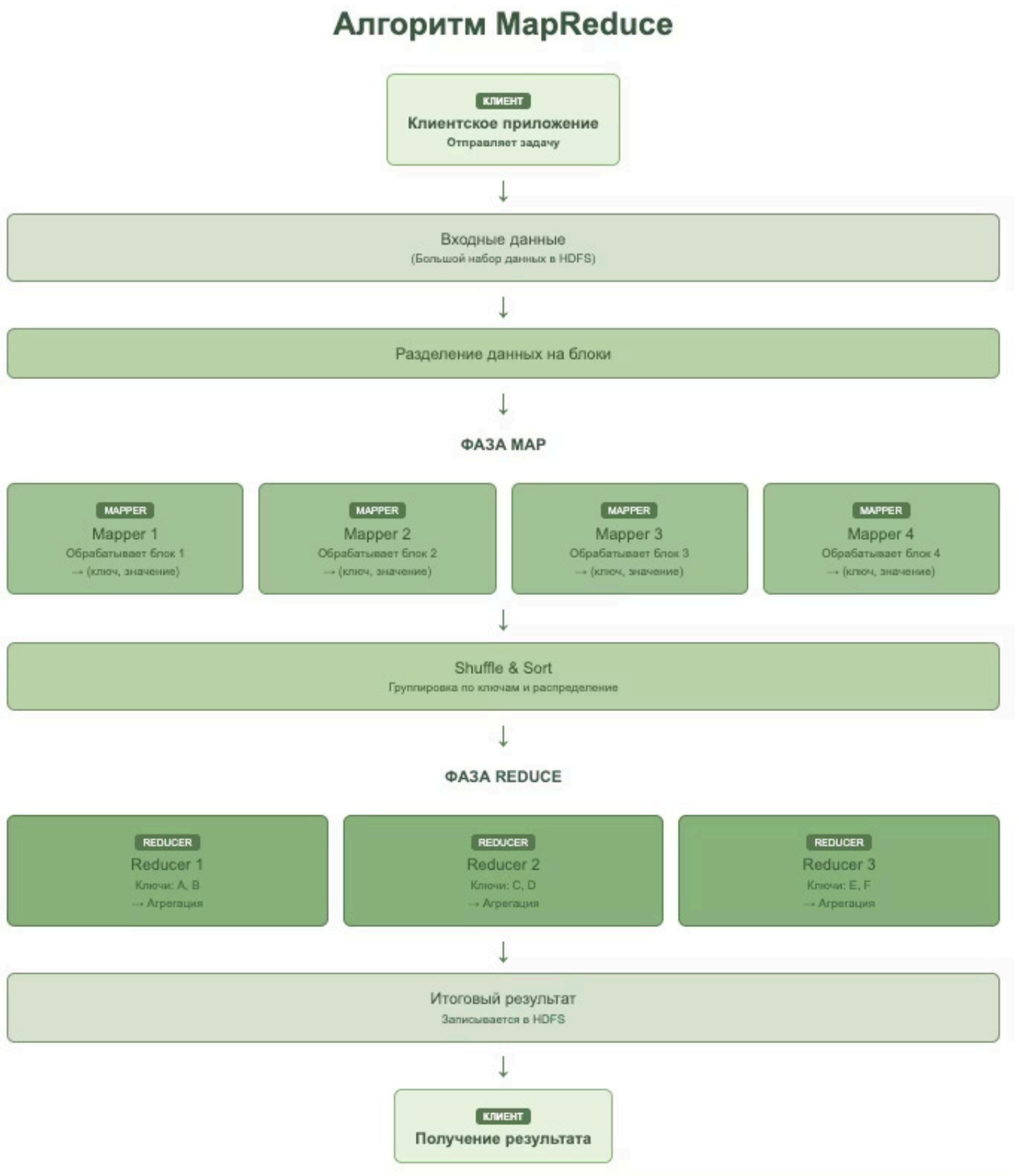
### Этап Shuffle and Sort

Представляет собой основной механизм, хотя его часто упускают из виду в обсуждениях. На этом этапе данные перераспределяются по кластерам, чтобы все значения для заданных ключей попадали в один и тот же редуктор. Фреймворк обрабатывает разбиение, сортировку и фактическую передачу по сети. Эта фаза перемешивания обычно представляет собой самую дорогостоящую операцию в заданиях MapReduce.

03

### Фаза редукции

Агрегирует промежуточные результаты. Каждый редуктор обрабатывает данные, полученные мапперами и собранные вместе на этапе Shuffle & Sort, производя конечный результат. Фреймворк гарантирует, что каждый редуктор видит ключи в отсортированном порядке, что позволяет эффективно обрабатывать сгруппированные данные.




Что делает MapReduce уникально подходящим для больших данных, так это тщательная оркестрация в сочетании со встроенной отказоустойчивостью. Hadoop, например, автоматически обрабатывает распределение данных, сбои узлов и восстановление. Если мапперы или редукторы выходят из строя, фреймворк повторно запускает задачи на других узлах.

Типичный случай использования — анализ данных сеансов пользователей по миллиардам событий для расчета средней продолжительности сеанса по странам. На этапе Map из каждого сеанса извлекаются страна и продолжительность, на этапе Shuffle все сеансы группируются по странам, а на этапе Reduce вычисляются средние значения. То, что на отдельных машинах заняло бы несколько дней, выполняется за несколько минут на кластерах.

Этот алгоритм часто путают с scatter-gather и fork-join, которые также являются мощными инструментами для параллельной обработки. Но они не были разработаны специально для задач больших данных.

- Scatter-gather подходит, когда у нас есть несколько машин, и данные полностью помещаются в память каждой. Тогда запрос посылается на каждую машину, которая обрабатывает свою часть и возвращает результат. Хороший пример - поиск Elasticsearch по фрагментам.
- Fork-join это алгоритм рекурсивного разложения задач на одной машине с несколькими ядрами. Пример - одноименный фреймворк из стандартной библиотеки в Java.

Но когда речь идет о больших данных, которые не помещаются в памяти на стандартном оборудовании, нам нужен именно MapReduce.

# Экосистема Hadoop: платформа-пионер

Hadoop проложил путь для обработки больших данных. HDFS (Hadoop Distributed File System) обеспечил надежное распределенное хранилище, а MapReduce — инфраструктуру для обработки. Экосистема быстро расширилась за счет таких инструментов, как Hive для запросов типа SQL, Pig для сценариев потока данных и HBase для хранения NoSQL.

## Преимущества Hadoop

- Обработка больших данных на стандартном оборудовании
- Надежное распределенное хранилище
- Богатая экосистема инструментов
- Открытый исходный код

## Проблемы эксплуатации

- Значительная сложность эксплуатации
- Необходимость глубоких знаний
- Пакетный характер MapReduce
- Сложность отладки

Хотя Hadoop произвел революцию в обработке больших данных, он сопровождался значительной сложностью эксплуатации. Для работы кластеров Hadoop требовались глубокие знания, а пакетный характер MapReduce означал необходимость ждать завершения заданий в течение нескольких часов. К общим проблемам относились сложность отладки неудачных заданий, борьба с размерами кучи JVM и оптимизация локальности данных.

# Эволюция за пределами MapReduce

Хотя MapReduce был пионером в области распределенной обработки данных, его ограничения, в частности подход, требующий большого объема дискового пространства, и обработка только пакетами, привели к появлению нескольких эволюционных потомков, которые основываются на основных концепциях и устраняют недостатки.

### Apache Spark

Представляет собой де-факто стандарт для обработки больших данных на данном этапе. Он заменил жесткую двухэтапную модель MapReduce более гибким механизмом выполнения DAG (Directed Acyclic Graph). RDD и DataFrames Spark по-прежнему используют операции map и reduce, но могут связывать несколько преобразований без записи промежуточных результатов на диск. Такая обработка в памяти обеспечивает 10-100-кратное повышение производительности для итеративных алгоритмов, таких как машинное обучение.

### Apache Tez

Apache Tez — это фреймворк для распределённых вычислений, оптимизированный для повышения производительности обработки больших объёмов данных в Hadoop-экосистеме. Основное отличие от MapReduce заключается в динамической оптимизации запросов и уменьшении количества промежуточных операций ввода-вывода, что позволяет быстрее обрабатывать запросы, особенно в сложных графовых структурах зависимостей, тогда как Spark фокусируется больше на обработке данных в памяти и интерактивных запросах.

### Apache Flink и Storm

Применили MapReduce на потоковой обработке, применив аналогичные концепции распределенной обработки к неограниченным потокам данных. Они сохраняют концепции распределенного преобразования и агрегирования, но работают с непрерывными потоками данных, а не со статическими наборами данных.

### Движки SQL-on-Hadoop

Такие как Presto, Impala и Drill, использовали другой подход, реализовав распределенные движки SQL-запросов, которые полностью обходят MapReduce, но при этом продолжают работать с теми же распределенными файловыми системами. Они используют технологии параллельных баз данных, но применяют их к масштабам и гибкости систем больших данных.

Даже облачные сервисы следуют принципам MapReduce. AWS Athena, Google BigQuery и Snowflake используют различные варианты распределенной обработки в стиле map-reduce, хотя они абстрагируют сложность, скрытую за интерфейсами SQL.

# Apache Spark: революционное решение

Spark устранил многие недостатки Hadoop и быстро стал де-факто стандартом для обработки больших данных. Его ключевой инновацией стала абстракция Resilient Distributed Dataset (RDD), позволяющая выполнять обработку в памяти, которая часто в 10–100 раз быстрее, чем дисковая MapReduce.

Что делает Spark особенно привлекательным для систем SaaS, так это его унифицированный API для пакетной обработки, SQL-запросов, потоковой передачи данных, машинного обучения и обработки графов. Вместо того чтобы соединять несколько инструментов, он предоставляет единую платформу. API DataFrame и Dataset предоставляют абстракции более высокого уровня, которые оптимизируются автоматически, указывая, что нужно вычислить, а не как это сделать.

### Обработка в памяти

RDD позволяют выполнять обработку в памяти, обеспечивая 10-100-кратное повышение производительности для итеративных алгоритмов

### Унифицированный API

Единая платформа для пакетной обработки, SQL-запросов, потоковой передачи данных, машинного обучения и обработки графов

### Автоматическая оптимизация

Оптимизатор Catalyst применяет оптимизации на основе правил и затрат, создавая планы выполнения лучше настроенного вручную кода

Оптимизатор Catalyst в Spark применяет к запросам оптимизации на основе правил и затрат, часто создавая планы выполнения, которые намного лучше, чем настроенный вручную код. Функция адаптивного выполнения запросов (AQE) в Spark 3.0+ динамически корректирует планы на основе статистики выполнения, обрабатывая перекосы данных и другие реальные проблемы.

# Потоковая обработка: революция в режиме реального времени

В то время как MapReduce произвела революцию в пакетной обработке, современные большие данные все чаще требуют аналитики в режиме реального времени. Потоковая обработка обрабатывает данные по мере их поступления, продолжая вычисления без ожидания накопления пакетов.

Этот сдвиг парадигмы потребовал новых алгоритмов и концепций. Стратегии окон (tumbling, sliding, session windows) позволяют агрегировать бесконечные потоки в управляемые блоки.

Современные платформы потоковой передачи, такие как Apache Kafka, Pulsar и AWS Kinesis позволяют решать такие проблемы. Это не просто очереди сообщений — это распределенные журналы, которые позволяют осуществлять событийный сорсинг, захват изменений данных и потоковую обработку.

Kafka Streams и Apache Flink представляют собой современный уровень потоковой обработки. Они обрабатывают сложные окна, семантику «точно один раз» и операции с сохранением состояния, сохраняя при этом высокую пропускную способность. Для систем SaaS это позволяет создавать функции реального времени, которые были бы невозможны несколько лет назад.

# Решения облачных провайдеров: управление сложностью

Крупные облачные провайдеры осознали, что большинство организаций хотят получить возможности больших данных без операционных затрат. Их управляемые услуги упрощают многие сложные процессы, обеспечивая при этом надежность корпоративного уровня.

Три основных облачных провайдера предлагают комплексные решения для работы с большими данными, каждый со своими преимуществами:

**AWS** предоставляет наиболее полный набор инструментов. EMR (Elastic MapReduce) управляет кластерами Hadoop и Spark с автоматическим масштабированием. Athena революционизирует аналитику бессерверными SQL-запросами к данным S3, Kinesis обеспечивает потоковую обработку с автоматическим масштабированием и режимом On-Demand без планирования мощностей, а Glue в сочетании с Athena позволяет создавать озера данных с минимальной операционной нагрузкой.

**Google Cloud Platform** выделяется возможностями для машинного обучения. BigQuery обрабатывает петабайты данных за секунды без управления инфраструктурой, а Dataflow (управляемый Apache Beam) обеспечивает пакетную и потоковую обработку. Dataproc предлагает управляемые Spark и Hadoop с отличительной особенностью — посуточной оплатой и быстрым запуском кластеров.

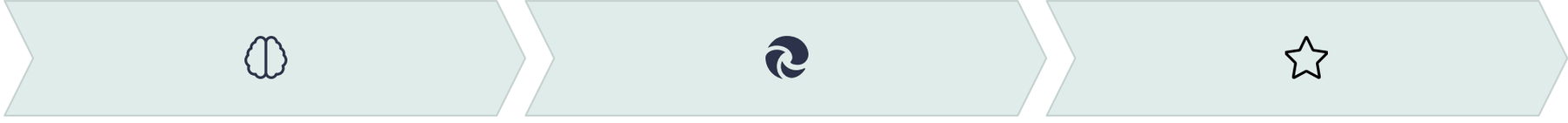
**Azure** фокусируется на корпоративной интеграции. Synapse Analytics объединяет большие данные и хранилища через единый интерфейс для запросов к озерам данных, хранилищам и пулам Spark. HDInsight предоставляет управляемые платформы с корпоративными функциями — интеграцией Active Directory и сквозным шифрованием, что привлекательно для организаций с жесткими требованиями к соответствию нормативам.

# Практические соображения и извлеченные уроки

После многих лет создания и эксплуатации систем больших данных вырисовываются определенные закономерности. Важно начинать с простого — инструменты для больших данных не нужны, пока большие данные не появятся в реальности. Команды часто развертывают кластеры Spark для гигабайтов данных, с которыми PostgreSQL может легко справиться. Для небольших объемов использование Spark или Hadoop будет медленнее, чем обработка всего в памяти. Многие компании используют гибридные подходы: до определенных пороговых значений используют обработку в памяти, а при превышении этих значений прибегают к инструментам для больших данных.

# Заключение

Будущее обработки больших данных в системах SaaS выглядит многообещающим. Машинное обучение становится неотъемлемой частью платформ данных, а такие функции, как Spark MLlib и BigQuery ML, делают передовые аналитические инструменты доступными для большего числа команд. Машинное обучение в реальном времени на потоковых данных открывает новые возможности для персонализации и автоматизации.



### Машинное обучение

Становится неотъемлемой частью платформ данных, делая передовые аналитические инструменты доступными для большего числа команд

### Edge вычисления

Приближают обработку к источникам данных, сокращая задержки и затраты на пропускную способность

### Бессерверная обработка

Устраняет последние остатки управления инфраструктурой для обработки больших данных

Edge вычисления приближают обработку к источникам данных, сокращая задержки и затраты на пропускную способность. Бессерверная обработка больших данных устраняет последние остатки управления инфраструктурой. Такие технологии, как Apache Arrow, стандартизируют форматы данных в памяти, улучшая совместимость и производительность.

По мере роста масштабов и сложности систем SaaS возможности обработки больших данных будут становиться все более важными для обеспечения конкурентного преимущества. Команды, которые освоят эти технологии, понимая не только как их использовать, но и когда и почему, будут иметь наилучшие возможности для создания приложений нового поколения, основанных на данных.