Студенческий журнал по математике и её приложениям, 2025, Том 4, № 1. С. 143–147.

Архитектура и инженерия платформ обработки больших данных: современные паттерны и технологии

Крюкова А. Д.
1472578@bsuedu.ru

Аннотация. Статья посвящена анализу архитектуры и инженерии платформ обработки больших данных, с акцентом на современные паттерны и технологии, используемые для решения задач штабируемости, аналитики и безопасности. В условиях стремительного роста объемов данных, эффективная обработка и управление информацией становятся ключевыми аспектами для бизнеса и науки. Рассматриваются основные архитектурные подходы, такие как пакетная и потоковая обработка данных, а также гибридные решения, позволяющие решать задачи в реальном времени и с высокой нагрузкой. В статье детально изучены популярные технологии, включая Hadoop, Apache Spark, NoSQL базы данных, а также системы для потоковой обработки данных, такие как Apache Kafka и Flink. Особое внимание уделяется вопросам безопасности, отказоустойчивости и масштабируемости платформ. Статья также поднимает текущие тренды в области больших данных, включая использование искусственного интеллекта и машинного обучения для аналитики и обработки информации. Обсуждаются перспективы развития технологий и их влияние на будущее обработки больших данных.

Ключевые слова: архитектура больших данных, инженерия платформ, обработка данных, масштабируемость, аналитика данных, потоковая обработка, пакетная обработка, Hadoop, Apache Spark, NoSQL базы данных, Apache Kafka, отказоустойчивость, безопасность данных, искусственный интеллект, машинное обучение, данные в реальном времени, облачные технологии, гибридные архитектуры, Data Lakes, Data Warehouses, ETL процессы

Для цитирования: Крюкова А. Д. 2025. Архитектура и инженерия платформ обработки больших данных: современные паттерны и технологии. *Студенческий журнал по математике и её приложениям*, 4(1): 143–147.

---

1. Введение. В последние десятилетия объемы данных, генерируемых во всех сферах жизни, значительно возросли. От социальных сетей и онлайн-торговли до здравоохранения и финансов — данные стали важнейшим активом для принятия решений, разработки инноваций и оптимизации бизнес-процессов. Однако для эффективного использования этого потока информации необходимы мощные и масштабируемые платформы, которые способны не только хранить, но и обрабатывать огромные массивы данных в реальном времени. Архитектура и инженерия платформ обработки больших данных (Big Data) играют ключевую роль в создании таких систем. Эти платформы требуют не только хранения данных, но и быстрой обработки, аналитики, а также обеспечения безопасности и отказоустойчивости. Современные технологии и паттерны архитектуры должны решать задачи масштабируемости, интеграции данных, управления потоком информации и обработки данных в различных форматах, включая структурированные, полуструктурированные и неструктурированные данные.

2. Архитектурные подходы в платформам обработки больших данных.

Определение 1.1. *Платформы обработки больших данных — это комплексные технологические решения, предназначенные для сбора, хранения, обработки и анализа больших объемов данных, которые могут быть как структурированными, так и неструктурированными [3].*

Такие платформы позволяют организациям работать с данными, которые слишком велики или сложны для традиционных систем управления базами данных (СУБД), таких как реляционные базы данных. В условиях стремительного роста объемов данных и их разнообразия, платформы обработки больших данных становятся важнейшими инструментами для бизнеса, научных исследований, финансовых организаций и других отраслей [4]. Основные компоненты платформ обработки больших данных включают:

- Хранение данных Платформы обеспечивают хранение больших объемов информации в распределенных системах. Примеры технологий хранения: Распределенные файловые системы (например, Hadoop Distributed File System — HDFS), которые позволяют эффективно управлять большими объемами данных на множестве серверов. Облачные хранилища (например, Amazon S3, Google Cloud Storage), которые предоставляют высокую масштабируемость и доступность данных. Data Lakes и Data Warehouses — решения для хранения как сырых, так и обработанных данных [6, 8].

- Обработка данных Платформы обработки данных позволяют выполнять различные операции: от простой агрегации и фильтрации до сложной аналитики и машинного обучения. Существуют два основных типа обработки: Пакетная обработка (Batch Processing) — обработка данных порциями, часто с использованием таких технологий, как Apache Hadoop или Apache Spark. Потоковая обработка (Stream Processing) — обработка данных в реальном времени, например, с помощью Apache Kafka, Apache Flink или Apache Storm [1].

- Аналитика и извлечение данных На этих платформах реализуются различные инструменты и технологии для анализа данных и извлечения полезной информации: Инструменты для аналитики (например, Apache Drill, Presto, Google BigQuery) позволяют выполнять запросы по огромным объемам данных. Машинное обучение и искусственный интеллект — использование алгоритмов для обработки и анализа данных (например, TensorFlow, PyTorch, Scikit-learn) в рамках платформ больших данных [5].

- Масштабируемость и отказоустойчивость Платформы обработки больших данных проектируются с учетом высокой нагрузки и необходимости в горизонтальном масштабировании. Это означает добавление новых серверов или узлов в систему без значительных изменений в архитектуре. Также критически важным является обеспечение отказоустойчивости, что достигается за счет дублирования данных и процессов (например, репликация в Hadoop или использование распределенных баз данных) [4, 2].

- Интеграция и обработка данных Платформы обработки данных часто включают средства для интеграции и обработки данных из разных источников. Примером является использование ETL (Extract, Transform, Load) процессов, когда данные извлекаются из разных источников, преобразуются и загружаются в хранилище для последующего анализа [10].

Примеры популярных платформ:

- Apache Hadoop — одна из первых платформ для распределенной обработки и хранения больших данных, включающая HDFS для хранения и MapReduce для обработки [14].

- Apache Spark — более современная альтернатива Hadoop, позволяющая выполнять обработку данных в памяти (in-memory), что значительно ускоряет анализ.

- Google BigQuery — облачная платформа для аналитики больших данных, ориентированная на работу с SQL-запросами [9].

- Amazon Redshift — облачная платформа для аналитики и обработки данных с высокой производительностью.

Задачи, решаемые платформами:

- Обработка больших объемов данных (террабайты и петабайты).

- Обработка различных типов данных: структурированных, полуструктурированных и неструктурированных.

- Интеграция данных из различных источников (например, баз данных, сенсоров, интернет-источников) [7].

- Аналитика в реальном времени для принятия быстрых решений.

- Хранение данных для долгосрочного использования и архивации.

В итоге, платформы обработки больших данных служат основой для работы с информацией, которая в современном мире значительно увеличивает свою ценность и может быть использована для принятия более обоснованных решений, разработки новых продуктов и услуг, а также для улучшения бизнес-процессов [11, 12].

3. Паттерны и подходы к обработке данных. Основные архитектурные паттерны для обработки больших данных фокусируются на том, как эффективно хранить, обрабатывать и анализировать данные, которые по объему и сложности превышают возможности традиционных систем. Рассмотрим несколько ключевых архитектурных паттернов, которые применяются в платформах обработки больших данных (Таблица 1).

Эти архитектурные паттерны можно комбинировать в зависимости от потребностей бизнеса и задач обработки данных. Выбор подходящей архитектуры зависит от множества факторов, включая тип данных, требуемую скорость обработки, стоимость решения и сложность системы.

В зависимости от специфики бизнеса и характера данных, платформы хранения данных могут использовать различные подходы и технологии. Распределенные файловые системы и облачные хранилища подходят для работы с большими объемами данных, Data Lakes предоставляют гибкость для хранения сырых данных, а Data Warehouses оптимизированы для аналитики. Важно выбрать правильную технологию в зависимости от конкретных задач и требуемой производительности [16].

Архитектурный паттерн	**Описание**	**Примеры технологий**	**Преимущества**	**Недостатки**
Пакетная обработка (Batch Processing)	Модель, при которой данные накапливаются и обрабатываются пакетами через определённые промежутки времени.	Apache Hadoop (MapReduce), Apache Spark	Подходит для обработки больших объёмов данных; хорошо масштабируется.	Не подходит для обработки в реальном времени; возможна задержка в принятии решений.
Потоковая обработка (Stream Processing)	Обработка данных в режиме реального времени по мере их поступления.	Apache Kafka, Apache Flink, Apache Storm	Позволяет мгновенно реагировать на события; актуально для online-систем.	Высокие требования к инфраструктуре; неэффективно для анализа исторических данных.
Гибридная архитектура (Hybrid Architecture)	Комбинирует пакетную и потоковую обработку данных в одной системе.	Apache Kafka + Spark Streaming, Apache Flink + Hadoop	Гибкость, возможность обрабатывать разные типы задач; эффективное использование ресурсов.	Сложность в реализации и поддержке; может требовать больше вычислительных ресурсов.
Микросервисная архитектура для обработки данных	Состоит из независимых компонентов (микросервисов), каждый из которых выполняет отдельные функции обработки данных.	Docker, Kubernetes, Apache Kafka	Высокая масштабируемость; независимая разработка и развертывание; отказоустойчивость.	Сложность в управлении и синхронизации множества сервисов.
Архитектура Data Lake	Хранение больших объёмов сырых данных из различных источников.	Hadoop HDFS, Amazon S3, Google Cloud Storage, Apache Spark	Гибкость и масштабируемость; удобство хранения неструктурированных данных.	Сложность в управлении и организации данных; требует мощную инфраструктуру.
Архитектура Data Warehouse	Централизованное хранилище для структурированных и очищенных данных, ориентированное на аналитику.	Amazon Redshift, Google BigQuery, Snowflake, Apache Hive	Удобство анализа; высокая производительность запросов; чистые данные.	Ограниченная гибкость; высокие затраты на хранение и поддержку.

Таблица 1: Ключевые архитектурные паттерны.

4. Пример реализации кода на базе Apache Spark для обработки больших данных.
Приведем пример простого кода, который демонстрирует использование распределённого хранилища данных и аналитической платформы на базе Apache Spark для обработки больших данных.

Допустим, у нас есть набор логов или данных о транзакциях в формате CSV, которые нужно обработать с использованием Apache Spark. Этот код будет читать данные из CSV файла, очищать данные и выполнять базовую аналитику, такую как подсчет количества уникальных пользователей. Пример реализации кода на базе Apache Spark для обработки больших данных Пример кода на Python с использованием PySpark:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col
spark = SparkSession.builder \
.appName("BigDataProcessingExample") \
.getOrCreate()
df = spark.read.option("header", "true").csv("path_to_your_data.csv")
df.show(5)
df = df.withColumn("amount", col("amount").cast("float"))
df_clean = df.dropna(subset=["user_id", "amount"])
unique_users_count = df_clean.select("user_id").distinct().count()

print(f"Количество уникальных пользователей: {unique_users_count}")
agg_data = df_clean.groupBy("user_id").agg({"amount": "sum"})
agg_data.show(10)
agg_data.write.option("header", "true").csv("output_path.csv")
spark.stop()
```

Пояснения:

- Инициализация Spark. Мы начинаем с создания сессии Spark, которая необходима для работы с данными в Spark. SparkSession.builder.appName() позволяет задать название приложения.
- Загрузка данных. Данные загружаются из CSV файла с помощью метода .read.csv(). Мы указываем опцию header=True, чтобы Spark знал, что первая строка является заголовком.
- Предобработка данных. Мы очищаем данные, например, приводим столбец amount к типу float, чтобы проводить математические операции, а затем удаляем строки с пропущенными значениями в столбцах userID и amount.
- Аналитика. Мы выполняем подсчёт количества уникальных пользователей с помощью .distinct().count(), а также агрегацию по сумме транзакций для каждого пользователя с помощью .groupBy().agg() [15].
- Сохранение данных. После обработки данных мы сохраняем результат в новый CSV файл с помощью метода .write.csv().
- Завершение работы. В конце важно завершить работу сессии Spark, вызвав spark.stop().

Преимущества и возможности:

- Этот код можно запустить в распределённой среде, чтобы обрабатывать данные на тысячах узлов в кластере.
- Возможность адаптации кода для обработки разных типов данных, например, с использованием потоковой обработки или работы с большими данными в различных форматах (Parquet, JSON, Avro и т. д.).
- В Spark можно интегрировать более сложные аналитические запросы или модели машинного обучения, если необходимо проводить более глубокий анализ [13].

Этот пример является только основой для обработки больших данных и может быть значительно расширен для решения более сложных задач, таких как интеграция с Kafka для потоковой обработки данных или использование более сложных алгоритмов машинного обучения в MLlib.

5. Заключение. Современные платформы для обработки больших данных и соответствующие архитектурные паттерны играют критически важную роль в успешной работе организаций в условиях стремительного роста объемов информации. Архитектуры, такие как пакетная обработка, потоковая обработка и гибридные решения, обеспечивают разнообразие подходов к решению задач, требующих

работы с большими объемами данных. Каждое из решений имеет свои уникальные преимущества и ограничения, которые должны учитываться при выборе подходящей архитектуры.

Кроме того, современные технологии хранения данных, такие как Data Lakes, Data Warehouses и облачные хранилища, позволяют обеспечить гибкость, масштабируемость и доступность данных для аналитики и последующей обработки. Инструменты, такие как Apache Spark, Hadoop, Apache Kafka, и облачные сервисы (например, Amazon S3 и Google BigQuery), становятся основой для эффективной обработки и анализа данных.

Однако несмотря на значительные преимущества этих решений, существуют и вызовы, связанные с управлением данными, их безопасностью и оптимизацией стоимости хранения. Необходимость в комплексных и масштабируемых системах требует от компаний постоянного совершенствования их архитектуры и инструментов для работы с данными, что может быть дорогостоящим и технологически сложным процессом.

Список литературы

1. Алпатов А. Н. 2025. Современные архитектуры и унификация обработки больших данных. Интеллектуальный потенциал России, № 2: 3–16.
2. Архитектурный паттерн для обработки больших данных: Lambda [Электронный ресурс]. - Режим доступа: https://habr.com/ru/companies/otus/articles/766672/ (дата обращения 10.06.2025)
3. Исаченко Ю. В., Степанов В. А. 2024. Архитектура вычислений. Обработка больших объемов данных : методические рекомендации. Витебск. Витебский государственный университет имени П. М. Машерова, Каф. прикладного и системного программирования: 44 с.
4. Клеменков П.А., Кузнецов С.Д. 2012. Большие данные: современные подходы к хранению и обработке. Труды ИСП320 РАН: 143–156.
5. Матвеева П. Р. 2018. Сравнение лямбда и традиционной архитектур. Форум молодых ученых, № 1 (17): 734–740.
6. Новиков Б.А., Графеева Н.Г. 2014. Big data: Новые задачи и современные подходы. КИО, №4: 10–18.
7. Осипов Д. 2022. Технологии проектирования баз данных. Litres, 2022.
8. Понин Ф.Н. 2024. Методология проектирования и создания баз данных для современного программного обеспечения. Universum: технические науки, №1 (118). URL: https://cyberleninka.ru/article/n/metodologiya-proektirovaniya-i-sozdaniya-baz-dannyh-dlya-sovremennogo-programmnogo-obespecheniya (дата обращения: 10.06.2025).
9. Пролетарский А. В., Березкин Д. В. 2018. Методы ситуационного анализа и графической визуализации потоков больших данных. Москва. Вестник МГТУ им. Н.Э. Баумана. Серия «Приборостроение», №2 (119): 98–123.
10. Радченко И.А., Николаев И.Н. 2018. Технологии и инфраструктура Big Data. Современные технологии в науке и образовании: 53 с.
11. Сокольников А. М. 2014. Сравнительный анализ подходов к разработке архитектуры и систем управления базами данных для высоконагруженных WEB-сервисов. Кибернетика и программирование, №. 4: 1–13.
12. AWS vs Azure vs Google Cloud [Электронный ресурс]. – Режим доступа: https://cloudfresh.com/ru/cloud-blog/aws-vs-azure-vs-google-cloud/?utm_source=chatgpt.com (дата обращения 10.06.2025)
13. Data Age 2025 [Электронный ресурс]. – Режим доступа: https://www.seagate.com/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf (дата обращения 10.06.2025)
14. Gartner [Электронный ресурс]. – Режим доступа: https://www.gartner.com/en (дата обращения 10.06.2025)
15. Hadoop vs. Spark: What's the difference? [Электронный ресурс]. - Режим доступа: https://www.ibm.com/think/insights/hadoop-vs-spark?utm_source=chatgpt.com (дата обращения 10.06.2025)
16. Kafka vs. Spark vs. Hadoop [Электронный ресурс]. – Режим доступа: https://www.logicmonitor.com/blog/kafka-vs-spark-vs-hadoop?utm_source=chatgpt.com (дата обращения 10.06.2025)
17. Seagate Reports [Электронный ресурс]. – Режим доступа: https://www.seagate.com/ (дата обращения 10.06.2025)

Поступила в редакцию 15.06.2025

СВЕДЕНИЯ ОБ АВТОРЕ

Крюкова Анжелика Дмитриевна – магистрант 1-го года обучения, Белгородский государственный национальный исследовательский университет

РУКОВОДИТЕЛЬ

Чернова Ольга Викторовна – кандидат физико-математических наук, доцент, доцент кафедры прикладной математики и компьютерного моделирования, Белгородский государственный национальный исследовательский университет

Chernova_Olga@bsuedu.ru

К содержанию