ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
INFORMATION TECHNOLOGY AND TELECOMMUNICATIONS

УДК 004.85

DOI: 10.21822/2073-6185-2026-53-2-110-123 Обзорная статья/ Review article

Сравнительный анализ инструментов обработки больших данных на платформе Hadoop

С.А. Резниченко, А.Н. Алексеева
Финансовый университет при Правительстве Российской Федерации,
125167, г. Москва, Ленинградский пр-т, 49/2, Россия

Резюме. Цель. В статье представлен всесторонний сравнительный обзор ведущих инструментов, используемых для обработки больших данных в экосистеме Apache Hadoop, с целью обеспечения организаций критериями выбора оптимального решения для различных типов рабочих нагрузок. Актуальность исследования обусловлена тем, что современный бизнес требует анализа данных всех форм и размеров, включая структурированные и неструктурированные, чтобы лучше понимать свою продукцию, клиентов и рынки. Рост объемов, скорости и разнообразия данных (Big Data) привел к необходимости разработки специализированных платформ для их обработки. Проблема нестабильности Apache Hive является ключевым архитектурным вызовом, поскольку Hive, является наиболее ресурсоемким в выполнении идентичных задач со Spark, так как начинает обработку входных данных без их предварительного анализа. **Метод.** Проведен обзор источников, охватывающих широкий спектр инструментов и архитектурных решений для обработки больших данных в экосистеме Apache Hadoop и за ее пределами. Предложенная методология сочетает сравнительный анализ существующих подходов и комплексную методику выбора оптимальных инструментов и архитектурных решений для обработки больших данных. **Результат.** Проанализированные характеристики Cloudera Impala и Apache Hive позволяют сделать вывод, что данные SQL-инструменты для аналитической обработки данных в экосистеме Hadoop дополняют друг друга. **Вывод.** Полностью заменять Apache Spark на Flink преждевременно из-за потенциальной потребности в дополнительной инфраструктуре, но Flink является жизнеспособным решением для задач, требующих минимального времени отклика.

Ключевые слова: информационная безопасность, большие данные, Apache Hadoop, Apache Hive (Hive), Cloudera Impala (Impala), Apache Flink (Flink), Spark Structured Streaming, Kafka Streams, OLAP, ETL, OLTP, MPP.

Для цитирования: С.А. Резниченко, А.Н. Алексеева. Сравнительный анализ инструментов обработки больших данных на платформе Hadoop. Вестник Дагестанского государственного технического университета. Технические науки. 2026;53(2):110-123. DOI:10.21822/2073-6185-2026-53-2-110-123

Comparative analysis of big data processing tools on the Hadoop platform

S.A. Reznichenko , A.N. Alexeeva
Financial University under the Government of the Russian Federation,
49, Leningradsky Ave., Moscow 125167, Russia

Abstract. Objective. This article provides a comprehensive comparative review of the leading tools used for processing big data in the Apache Hadoop ecosystem, with the aim of providing organizations with criteria for selecting the optimal solution for different types of workloads. The relevance of this research is driven by the fact that modern businesses require the analysis of data of all shapes and sizes, including structured and unstructured data, in order to gain a better understanding of their products, customers, and markets. The increasing volume, velocity,

and variety of data (Big Data) has led to the need for specialized platforms to handle it. The Apache Hive instability issue is a key architectural challenge, as Hive is the most resource-intensive when performing identical tasks with Spark, as it starts processing input data without pre-analyzing it. **Method.** A review of sources covering a wide range of tools and architectural solutions for processing big data in the Apache Hadoop ecosystem and beyond. The review included specialized scientific articles, technical guides, experimental research results, and industry analysis reports. The proposed methodology combines a comparative analysis of existing approaches and a comprehensive methodology for choosing optimal tools and architectural solutions for processing big data. **Result.** The analyzed characteristics of Cloudera Impala and Apache Hive allow us to conclude that these SQL tools for analytical data processing, in the Hadoop ecosystem, complement each other. **Conclusion.** It is premature to completely replace Apache Spark with Flink due to the potential need for additional infrastructure, but Flink is a viable solution for tasks that require minimal response time.

Keywords: information security, big data, Apache Hadoop, Apache Hive (Hive), Cloudera Impala (Impala), Apache Flink (Flink), Spark Structured Streaming, Kafka Streams, OLAP, ETL, OLTP, MPP.

For citation: S.A. Reznichenko, A.N.Alexeeva. Comparative analysis of big data processing tools on the Hadoop platform. Herald of Daghestan State Technical University. Technical Sciences. 2026;53(2):110-123. (In Russ) DOI:10.21822/2073-6185-2026-53-2-110-123

Введение. Современный бизнес требует анализ данных всех форм и размеров, включая структурированные и неструктурированные, чтобы лучше понимать свою продукцию, клиентов и рынки. Рост объемов, скорости и разнообразия данных (Big Data) привел к необходимости разработки специализированных платформ для их обработки.

Постановка задачи. Цель исследования состоит в разработке и обосновании комплексной методики выбора оптимальных инструментов и архитектурных решений для обработки Больших Данных в экосистеме Apache Hadoop (SQL-движков и систем потоковой обработки), основываясь на критериях временной эффективности (задержка и пропускная способность) и функциональной целесообразности (отказоустойчивость, сложность запросов и режим обработки).

Для достижения поставленной цели в работе решаются следующие задачи: анализ архитектурных основ и контекста Big Data, сравнительный анализ SQL-движков (сравнительный анализ архитектур, производительности и сценариев использования Apache Hive, сравнительный анализ передовых систем потоковой обработки (Stream Processing Engines, SPEs, формулировка практических рекомендации по выбору оптимального SQL-движка (Impala, Hive, Spark SQL) и потокового движка (Flink, Structured Streaming, Kafka Streams) в зависимости от требований к задержке и типу рабочего процесса).

В анализируемых архитектурах часто присутствуют оба инструмента: Apache Hive и Apache Spark. Оба инструмента нацелены на работу с большими данными и могут выполнять параллельные запросы. Однако существует проблема нестабильности работы инструмента Hive. Hive уступает Spark в части потребления ресурсов, являясь наиболее ресурсоемким в выполнении одних и тех же задач со Spark. Это происходит потому, что Hive начинает обработку входных данных без их предварительного анализа, потребляя тем самым больше ресурсов на задачу.

Методы исследования. Для решения этой архитектурной проблемы предлагаются следующие альтернативные пути:

1. Использовать решение объединения продуктов Spark+Python+Scala.
2. Переход от Hive на Impala. Hive и Impala - два мощных инструмента анализа данных, широко используемых в экосистеме Hadoop.

Кроме того, в Apache Spark добавили оптимизации. Это позволило сделать бакетирование применимым к большему количеству сценариев и упростить миграцию с Hive

на Spark SQL. Однако бакетирование в Spark SQL требует сортировки по времени чтения, что может снижать производительность выполнения запроса, а запись данных в бакетированную таблицу может генерировать множество небольших файлов, которые неэффективно обрабатываются HDFS.

Архитектура Big Data, Apache Hadoop, имеет платформу для хранения и работы с большими объемами данных. Impala и Hive также используют YARN в качестве кластерного менеджера ресурсов. Для обеспечения безопасности, включая авторизацию, учет и контроль, в экосистеме Hadoop используется Ranger. Плагины Ranger доступны для таких сервисов, как Hive, Impala, Spark3 и YARN. Читая обзор архитектуры Impala, можно понять, что она поддерживает аутентификацию по актуальным отраслевым стандартам, включая Kerberos или LDAP.

Основная цель сравнительных статей основных систем потоковой обработки данных: Apache Flink, Apache Kafka Streams и Apache Spark Structured Streaming для понимания фундаментальных концепций потоковой обработки, включая управление состоянием (stateful processing), контрольные точки (checkpointing), временную семантику (event time) и обработку обратного давления.

В аналитике Больших Данных существуют жесткие требования к Актуальности (Timeliness) данных, что напрямую связано с выбором технологического инструмента и его архитектурой. Требование актуальности (Timeliness) критично для финансовых транзакций, мониторинга показателей в реальном времени и обнаружения мошенничества. Apache Flink разработан для низкой задержки и real-time processing. Flink последовательно достигает ультранизкой задержки — менее 100 миллисекунд для сложных stateful операций. Apache Spark Structured Streaming использует микропакетный подход (micro-batching), что вносит более высокую задержку. Низкое качество данных (например, низкая актуальность) может привести к неправильным выводам и ошибочным решениям.

Таким образом, если регулятивные требования сосредоточены на обеспечении безопасности (авторизация через Kerberos/LDAP) и отказоустойчивости (избыточность HDFS), то архитектурные требования, вытекающие из самой природы Больших Данных, требуют обязательной оценки временной эффективности (Latency, Throughput), что достигается выбором между Flink (для low-latency) и Impala (для OLAP).

Объектом исследования выбран Apache Hadoop - это платформа с открытым исходным кодом. Ее ключевое преимущество в распределении вычислений между большим количеством узлов. Hadoop стал первым фреймворком, появившимся для обработки больших наборов данных с использованием программной модели MapReduce. [14] Компоненты фреймворка: Hadoop Distributed File System (HDFS); MapReduce; Yet Another Resource Negotiator (YARN).

Hadoop обеспечивает масштабируемость до тысяч узлов, экономию (использование дешевых серверов), отказоустойчивость (благодаря распределенной архитектуре и автоматической репликации) и гибкость (работа со структурированными и неструктурированными данными). Hadoop эффективен для пакетной обработки, когда время выполнения не критично: Spark Core; Spark SQL; Spark Streaming.

При разработке методологии выбора критериев эффективности был использован системный подход. Исследование носило прикладной характер с опорой на комбинацию нескольких методов: сравнительный анализ Hadoop и Spark; детальный сравнительный анализ SQL-движков (HIVE, IMPALA, SPARK SQL); анализ потоковой обработки: FLINK, SPARK STREAMING и KAFKA STREAMS. Необходимость сравнительного анализа Apache Hadoop и Apache Spark обусловлена их архитектурной эволюцией и различиями в подходах к обработке Больших Данных. Важно определить их оптимальные роли в единой гибридной архитектуре: Hadoop предоставляет отказоустойчивое хранение, а Spark высокоскоростные вычисления. Анализ позволит организациям определить оптимальные роли: использовать ли Hadoop как основное файловое хранилище (HDFS), а Spark

как высокоскоростной вычислительный движок. В табл. 1 приведен анализ ключевых характеристик платформ по следующим аспектам: скорость, место хранения промежуточных результатов, программная модель, поддержка потоковой обработки, наличие встроенной библиотеки машинного обучения и оптимизация.

Таблица 1. Сравнение основных характеристик платформ Apache Hadoop (MapReduce) и Apache Spark

Table 1. Comparison of main characteristics of Apache Hadoop (MapReduce) and Apache Spark platforms

	Характеристика Characteristic	**Apache Hadoop (MapReduce)**	**Apache Spark**
1.	Скорость Speed	Не отличается высокой скоростью Not very fast	До 100 раз быстрее Hadoop MapReduce [17] Up to 100 times faster
2.	Промежуточные результаты Intermediate Results	Записывает на диск Writes to disk	Хранит в памяти Stores in memory
3.	Программная модель Program Model	Двухэтапная map и reduce Two-stage map and reduce	Более гибкая, использует RDD и DataFrame API More flexible
4.	Потоковая обработка Stream Processing	Ограниченная поддержка Limited support	Полная поддержка через Spark Streaming Full support
5.	Машинное обучение Machine Learning	Нет встроенной библиотеки No built-in library	Встроенная библиотека MLlib Built-in library
6.	Оптимизация Optimization	Сложно реализовывать многоэтапные алгоритмы Difficult to implement multi-stage algorithms	Использует DAG для оОптимизации Uses DAG for optimization

В результате проведенного анализа установлено, что Apache Spark является эволюционным шагом по отношению к Hadoop MapReduce, превосходя его в скорости до 100 раз. MapReduce, записывая данные на диск, уменьшает скорость выполнение задачи. В тоже время Spark более универсален, имеет полную поддержку потоковой обработки и встроенную библиотеку MLlib для машинного обучения, в то время как Hadoop имеет в этом отношении ограниченную поддержку.

Обсуждение результатов. Сравнительный анализ SQL-ДВИЖКОВ (HIVE, IMPALA, SPARK SQL). Apache Hive и Cloudera Impala имеют ряд существенных сходств, несмотря на различия в архитектуре [1]. Hive и Impala предназначены для аналитической обработки больших данных. Оба инструмента позволяют выполнять запросы и проводить анализ массивов информации, хранящихся в распределённых файловых системах, таких как HDFS, HBase, а также в облачных хранилищах, например Amazon S3.

Hive и Impala являются проектами с открытым исходным кодом, распространяемыми под лицензией Apache Software Foundation (ASF). Hive была разработана компанией Facebook в 2010 году и позже передана в управление ASF, тогда как Impala изначально создавалась компанией Cloudera и интегрирована в экосистему Apache Hadoop.

Impala использует тот же язык запросов, что и Hive - HiveQL, а также совместно применяет одни и те же метаданные, ODBC-драйверы и пользовательские интерфейсы. Это обеспечивает высокую степень совместимости между системами и упрощает переход пользователей с одного инструмента на другой. Интеграция с YARN: оба решения используют YARN в качестве диспетчера ресурсов кластера. Это позволяет эффективно распределять вычислительные мощности, управлять нагрузкой и обеспечивать стабильное выполнение задач в распределённой среде Hadoop. Apache Hive - это инфраструктура хранилища данных, построенная поверх Hadoop.

Hive работает в режиме пакетной обработки (OLTP), обрабатывая данные пакетами. Hive не походит для OLTP-задач в реальном времени. Он идеально подходит для длительных ETL-задач и задач анализа больших наборов данных. Hive был построен для обработки совершенства (sophistication) и очень эффективен при выполнении сложных запросов, возможно, требующих тяжелых преобразований и/или множественных join.

LLAP (Live Long And Process): Механизм LLAP является механизмом выполнения под управлением Hive, который поддерживает длительные процессы, используя одни и те же ресурсы для кэширования и обработки. Hive написан на Java. Он генерирует выражения запросов во время компиляции (compile time), что приводит к проблеме «холодного старта» при первом запуске приложения. Hive поддерживает сложные типы данных.

Cloudera Impala - это открытый MPP-движок (Massively Parallel Processing), разработанный для высокоскоростной аналитической обработки данных в экосистеме Hadoop. Проект был впервые представлен в октябре 2012 года, а его общедоступная версия вышла в мае 2013 года.

Impala обеспечивает выполнение SQL-запросов в режиме реального времени, что делает её мощным инструментом для интерактивной аналитики (OLAP). Благодаря оптимизированной архитектуре и отсутствию необходимости в использовании промежуточных систем вроде MapReduce, Impala демонстрирует высокую производительность и минимальную задержку. Это особенно ценно при проведении ad-hoc-анализа и оперативного исследования больших массивов данных.

Оптимальной областью применения Impala являются витрины данных (data marts), где критически важны быстрый доступ к информации и удобство работы с ней для конечных аналитиков и бизнес-пользователей. Impala была разработана прежде всего для скорости. Impala значительно превосходит Hive в скорости, работая в 6-69 раз быстрее с простыми SQL-запросами. Impala была создана с приоритетом на высокую производительность. По сравнению с Hive, она демонстрирует значительно более высокую скорость выполнения запросов - в зависимости от нагрузки. Основой её производительности является архитектура Massively Parallel Processing (MPP), которая обеспечивает эффективное распределённое выполнение запросов за счёт параллельной обработки на всех узлах кластера и использования оптимизированного плана выполнения в виде многоуровневого дерева операций [16]. Основные технические особенности Impala:

1. Impala написан на C++, что обусловливает более эффективное использование памяти. Impala также использует элементы Java.
2. Impala поддерживает in-memory data processing, обращаясь к данным, хранящимся на узлах Hadoop, без их перемещения или преобразования.
3. Impala передает промежуточные результаты потоком (streams) между исполнителями, жертвуя масштабируемостью ради скорости.
4. Impala выполняет генерацию кода во время выполнения (runtime code generation) для «больших циклов».
5. Impala не обладает встроенной отказоустойчивостью на уровне выполнения запросов — при сбое одного из узлов процесс обработки прерывается, и запрос необходимо запускать заново.

Кроме того, система не поддерживает сложные типы данных, что ограничивает ее гибкость при работе с вложенными или полуструктурированными данными. Из-за реализации на языке C/C++ Impala может испытывать проблемы с совместимостью форматов файлов и пользовательских функций, особенно если они разработаны на Java и изначально ориентированы на экосистему JVM, используемую в других компонентах Hadoop. Impala представляет собой движок распределенной базы данных с массовой параллельной обработкой (MPP). Решение на основе Impala состоит из Клиентов (Hue, JDBC и Impala Client), Hive Metastore, процессов Impala (работающих на нодах DataNode) и HBase/HDFS (хранилища данных). Компоненты сервиса Impala в ADH включают:

1. Impala Daemon (impalad): Impala Daemon (impalad) основной исполняющий компонент системы, отвечающий за приём, планирование и выполнение запросов от клиентов. При поступлении запроса один из экземпляров impalad выступает в роли координатора: он анализирует запрос, формирует план его выполнения и разбивает на параллельные подзадачи, которые затем распределяются между другими узлами кластера, где также запущены процессы impalad. Каждый демон включает три ключевых модуля:

Query planner - генерирует оптимизированный план выполнения запроса;

Query coordinator - управляет распределением подзапросов и сбором результатов;

Query executor - непосредственно выполняет свою часть запроса на локальном узле.

2. Impala Statestore (statestored) - служба отслеживания состояния кластера. Она непрерывно проверяет доступность всех узлов с запущенными демонами impalad. Если один из узлов становится недоступным, StateStore оповещает остальные узлы, чтобы они исключили его из обработки запросов. Это предотвращает отправку задач на неработающие ноды и обеспечивает стабильность выполнения. В кластере требуется только один экземпляр StateStore.

3. Impala Catalog Service (catalogd) - централизованная служба, отвечающая за синхронизацию метаданных. При выполнении DDL-операций (например, создание или изменение таблиц) через Impala, Catalog Service автоматически рассылает обновлённую информацию всем демонам impalad в кластере. Это гарантирует, что все узлы оперируют актуальной схемой данных без необходимости ручной синхронизации [16]. Impala использует HDFS в качестве основного хранилища данных, полагаясь на встроенную репликацию файловой системы для обеспечения отказоустойчивости и сохранности информации.

Метаданные о структуре таблиц хранятся в централизованной службе Metastore, реализованной на базе СУБД, такой как MySQL или PostgreSQL. Эта база метаданных разделяется с Apache Hive, что обеспечивает сквозную совместимость между двумя системами и позволяет свободно обмениваться таблицами и схемами в рамках единой экосистемы Hadoop. Благодаря этому Impala способна читать и обрабатывать таблицы, созданные или загруженные через Hive, при условии, что используются типы данных, поддерживаемые Impala. Это обеспечивает тесную интеграцию между двумя системами и позволяет легко обмениваться данными в рамках одной экосистемы. В табл. 2 проведена дифференциация SQL-движков по критериям производительности, отказоустойчивости и соответствия задачам.

Таблица 2. Сводная таблица сравнения характеристик SQL-движков Apache Hive, Cloudera Impala и Apache Spark SQL

Table 2. Summary table comparing the characteristics of the Apache Hive, Cloudera Impala, and Apache Spark SQL engines

№	Характеристика Characteristic	Apache Hive	Cloudera Impala	Apache Spark SQL
1.	Режим обработки Processing Mode	Пакетная (OLTP/ETL)	Интерактивная (OLAP)	Пакетная/Потоковая
2.	Пропускная способность Throughput	Самая низкая (но выше Impala с LLAP)	Самая высокая	Выше Hive, но в 7 раз ниже Impala
3.	Отказоустойчивость Fault Tolerance	Да (восстановление в середине запроса)	Нет (запрос начинается сначала)	Да (восстановление в середине запроса)
4.	Язык разработки Development Language	Java	C++ (C/C++ и Java)	Scala, Java, Python, R
5.	Промежуточные результаты Intermediate Results	Дисковая материализация	In-memory стриминг	In-memory кэширование
6.	Сложные типы данных Complex Data Types	Да	Нет	Поддержка
7.	Основной Use Case Primary Use Case	EDW, ETL, сложные запросы	BI Interactive, Ad-hoc	Data Engineering, Spark pipelines

В результате проведенного анализа подтверждается, что Impala является движком Massively Parallel Processing (MPP), оптимизированным прежде всего для скорости, благодаря in-memory стримингу промежуточных результатов, что делает его идеальным для интерактивной аналитики (OLAP). Однако Impala не является отказоустойчивой платформой. Напротив, Hive (OLTP/ETL) демонстрирует отказоустойчивость и способен восстановиться после сбоев в середине запроса, что делает его предпочтительным для длительных ETL-задач. Spark SQL занимает промежуточное положение, обладая отказоустойчивостью и универсальностью, но его пропускная способность в 7 раз ниже Impala.

С ростом потребности в обработке больших данных в режиме реального времени возрастает интерес к современным потоковым платформам. На сегодняшний день Apache Flink, Structured Streaming (в рамках Apache Spark) и Kafka Streams выделяются как ключевые фреймворки, определяющие развитие области потоковой обработки.

Центральные принципы эффективной потоковой обработки строятся вокруг фундаментальных концепций:

- Stream Processing (Потоковая обработка): характеризуется обработкой неограниченных потоков данных (unbounded data streams), которые не имеют известного конца.
- Stateful Processing (Обработка с состоянием): необходима для выполнения операций, которые должны помнить информацию о прошлых событиях, таких как агрегации, оконные вычисления или обнаружение паттернов.
- Event Time Processing: позволяет корректно обрабатывать события на основе фактического времени их возникновения, даже если они поступают поздно или не по порядку.
- Watermarks (Водяные знаки): механизм, используемый для отслеживания прогресса времени в неограниченном потоке и позволяющий процессорам решить, когда запускать оконные вычисления.
- Checkpointing: механизм для периодического сохранения снимка состояния приложения и его позиции в источниках. Flink гарантирует exactly-once processing semantics (строго однократная доставка сообщений). [2]
- Учёт временных меток (время) - способность корректно обрабатывать события не только по времени их поступления, но и по времени их возникновения (event time), включая задержанные и неупорядоченные данные.
- Управление состоянием - поддержка промежуточных данных между событиями, что необходимо для агрегаций, окон и сложной логики обработки.
- Гарантии доставки и надёжность - обеспечение корректного поведения системы при сбоях, включая гарантии exactly-once, at-least-once или at-most-once, в зависимости от требований приложения.Потоковая обработка: время, состояние и надежность.

Три ведущих платформы различаются в фундаментальном подходе к обработке данных.

1. Apache Flink: Flink, является распределенным движком для stateful вычислений над неограниченными и ограниченными потоками данных. Flink имеет истинную модель стриминга (true streaming model), обрабатывая каждое событие, как только оно прибывает. Flink обеспечивает обработку данных в реальном времени с задержкой около 1 миллисекунды и последовательно достигает 100-миллисекундной задержки для сложных операций. Flink, будучи разработанным для streaming-first использования, предлагает низкую задержку.

2. Spark Structured Streaming (SS): Structured Streaming — это масштабируемый и отказоустойчивый движок потоковой обработки, построенный на движке Spark SQL. Structured Streaming использует микропакетный подход (micro-batch approach), обрабатывая данные короткими, частыми пакетами, что лишь аппроксимирует потоковое поведение. Этот подход приводит к более высокой задержке по сравнению с Flink

(в диапазоне секунд). Однако SS превосходит Flink в ситуациях, требующих непревзойденной пропускной способности для крупномасштабной обработки данных.

3. Kafka Streams (KS): Kafka Streams (добавленная в релизе Kafka 0.10.0.0) является легковесной библиотекой потоковой обработки для создания приложений и микросервисов, где входные и выходные данные всегда хранятся в топиках Kafka. Kafka Streams имеет низкий барьер для входа, поскольку позволяет быстро писать и запускать небольшие приложения на одной машине. KS предлагает самую простую операционную модель.

В табл. 3 систематизированы ключевые механизмы потоковой обработки (Event Time, Checkpointing, Stateful Processing), проведен анализ компромиссов между Apache Flink и Spark Structured Streaming, а также определена ниша Kafka Streams на основе метрик задержки и пропускной способности.

Таблица 3. Сравнение потоковых движков (Apache Flink, Kafka Streams, Structured Streaming): дизайн, производительность и отказоустойчивость

Table 3. Comparison of streaming engines (Apache Flink, Kafka Streams, Structured Streaming): design, performance, and fault tolerance

№ пп	Характеристика Characteristic	Apache Flink	Kafka Streams	Structured Streaming
1.	Модель обработки Processing Model	Чистый стриминг Pure streaming	Чистый стриминг Pure streaming	Микропакетная обработка Microbatch processing
2.	Пропускная способность Throughput	Высокая High	Высокая High	Непревзойденная Unrivaled
3.	Event Time	A (Комплексная поддержка watermarks)	B (Специальная обработка suppress)	C (Базовая поддержка)
4.	Обработка поздних данных Processing late data	A (Комплексная, включая side outputs)	B (Поздние данные отбрасываются)	C (Ограниченный контроль, данные отбрасываются)
5.	Fault Tolerance	A (Асинхронные инкрементальные снимки RocksDB)	B (Changelog topic, долгое восстановление)	B (Snapshot checkpointing, риск больших checkpoints)
6.	Backpressure	A (Продвинутая обработка с динамическим регулированием)	B (Pull-based модель Kafka Streams, риск накопления данных в Kafka)	B (Автоматическая настройка, требует тюнинга микропакетов)
7.	Интерактивные запросы Interactive queries	A (Queryable state, по всему кластеру)	A (Interactive queries, для stateful microservices)	C (Ограниченная поддержка)

Управление ресурсами и масштабируемость: Flink является Лучшим в классе по Динамическому управлению ресурсами. Flink обладает эффективным управлением памятью с тонким контролем над ресурсами. Flink поддерживает динамическое выделение ресурсов и автоматически регулирует выделение памяти рабочим процессам.

В отличие от этого, Structured Streaming не поддерживает динамическое выделение ресурсов, как Spark batch jobs или Flink. Structured Streaming масштабируется путем добавления/удаления узлов, подобно Spark batch. Kafka Streams - это легковесная библиотека, которая не имеет встроенного динамического выделения ресурсов; его масштабируемость ограничена партициями Kafka.

Отказоустойчивость: Flink использует механизм Checkpointing, который обеспечивает надежность и гарантирует, что состояние и прогресс потока не будут потеряны в случае сбоя. Flink выполняет асинхронные, распределенные снимки состояния backends (например, RocksDB) и хранит их в надежном хранилище (HDFS, S3). Kafka Streams достигает восстановления, записывая журналы изменений (changelogs) состояния в специальные топики Kafka. Structured Streaming поддерживает асинхронное checkpointing, записывая состояние и метаданные в HDFS или облачные блобы.

В результате проведенного анализа архитектур потоковой обработки демонстрируется, что Apache Flink является лучшим в классе (A) по дизайну движка, обеспечивая чистый стриминг, что критично для низкой задержки (latency) и комплексной обработки состояния (Stateful Processing). Flink демонстрирует превосходство в управлении временной семантикой (Event Time) и Fault Tolerance, гарантируя exactly-once processing semantics.

В то же время Structured Streaming (SS), используя микропакетную обработку, является Лучшим в классе по пропускной способности (Throughput), но имеет базовую поддержку (C) для временной семантики. Kafka Streams (KS) также использует чистый стриминг, но его производительность и отказоустойчивость часто ограничены зависимостью от Kafka-топиков. Выбор платформы для стриминговой обработки данных должен опираться на требования бизнеса, существующую технологическую среду и уровень компетенций инженерной команды.

Представим расширенное сравнение трех ключевых платформ по основным направлениям: экосистема, производительность и операционная сложность, а также типичные сценарии применения.

1. Экосистема и поддержка разработки.

Apache Spark Structured Streaming (SS) занимает лидирующие позиции (A) в категории интеграции с экосистемой и доступности инструментов разработки. Он предлагает глубокую и нативную поддержку Lakehouse-форматов (Delta Lake, Apache Hudi, Iceberg, Paimon), что делает его особенно привлекательным для аналитических платформ, ориентированных на единое хранилище данных. Apache Spark Structured Streaming поддерживается всеми основными облачными провайдерами (AWS, GCP, Azure) и коммерческими дистрибутивами, такими как Databricks. Благодаря зрелой операционной модели и мультиюзиковой поддержке (Java, Scala, Python) SS обеспечивает низкий порог входа для команд с разным стеком.

Apache Flink, хотя и предлагает аналогичную языковую поддержку (Java, Scala, Python), требует значительно большего опыта для эффективной разработки и эксплуатации (B). Его экосистема быстро развивается, особенно в контексте современных lakehouse-архитектур и систем event processing, однако порог входа остается относительно высоким из-за сложной модели состояний и управления ресурсами.

Kafka Streams (KS) заметно уступает конкурентам в плане гибкости (C). Поддержка ограничена Java и Scala, что сужает аудиторию разработчиков. Кроме того, интеграция с файловыми форматами и data catalog-системами минимальна (C), поскольку KS фокусируется на задачах трансформации потоков в рамках Kafka-экосистемы. Тем не менее, именно эта узкая направленность делает его идеальным для микросервисов, плотно связанных с Kafka.

2. Производительность и операционная сложность. Производительность стримингового движка во многом определяется его архитектурой:

Apache Flink реализует истинную модель потоковой обработки (A), где каждое событие обрабатывается сразу после поступления. Благодаря этому достигаются экстремально низкие задержки менее 100 мс даже для сложных stateful-операций. Его механизм асинхронного и инкрементального чекпоинтинга обеспечивает максимальную надежность и устойчивость при сбоях. Поддержка fine-grained state management и гибкое распределение ресурсов делают Flink предпочтительным выбором для высоконагруженных real-time систем.

Spark Structured Streaming использует модель микробатчей, что приводит к задержкам на уровне секунд. Несмотря на это, Spark предлагает рекордную масштабируемость по пропускной способности (A), устойчиво работая с терабайтными потоками данных. Простота эксплуатации, а также встроенная поддержка lakehouse-инфраструктуры позволяют снижать операционные затраты и ускоряют внедрение.

Kafka Streams опирается на нативную streaming-модель Kafka (B), однако масштабирование ограничено количеством партиций топика (C). Это означает, что система масштабируется «горизонтально» лишь до тех пор, пока это позволяет структура данных в Kafka. Тем не менее, KS обеспечивает минимальную операционную нагрузку, приложение разворачивается как обычный микросервис, не требует кластеров и упрощает CI/CD.

3. Use Cases и выбор инструмента. При выборе стриминговой платформы важно сопоставить ее способности с бизнес-сценарием:

Apache Flink представляет лучший выбор для задач, требующих ультранизкой задержки, точной обработки состояния и сложной логической семантики: детекция мошенничества, антифрод, real-time scoring моделей ML, обработка телеметрии IoT, event-driven контроль производственных процессов.

Spark Structured Streaming оптимален для крупномасштабной аналитики, ETL/ELT в реальном времени, построения data lakehouse-архитектуры, обработки логов, CDC-конвейеров и гибридных сценариев batch + streaming.

Kafka Streams — эффективен в легковесных микросервисах, ориентированных на Kafka: агрегации событий в рамках одного сервиса, потоковая нормализация данных, enrichment-операции на уровне edge-микросервисов, потоковый роутинг.

Выбор между Flink, Spark Structured Streaming и Kafka Streams - это стратегический баланс:

Flink - максимальная скорость, точность и управление состоянием; Spark SS - универсальность, простота эксплуатации и тесная интеграция с lakehouse-экосистемами; Kafka Streams - минимальная сложность и идеальная поддержка Kafka-native микросервисов.

Каждая из платформ оптимальна в своем домене, и правильный выбор позволяет существенно повысить надежность, производительность и управляемость потоковых систем (табл. 4).

Таблица 4. Сравнение потоковых движков: экосистема, поддержка разработки и операционная сложность

Table 4. Comparison of streaming engines: ecosystem, development support, and operational complexity

№ пп	Характеристика Characteristic	Apache Flink	Kafka Streams	Structured Streaming
1.	Языковая поддержка Language support	A (Java, Scala, Python)	C (Нативная поддержка только Java, Scala)	A (PySpark, Scala, Java)
2.	Streaming SQL	A (Первоклассная поддержка, динамические таблицы)	C (Ограничено Java DSL)	B (Ограниченная возможность запуска Spark SQL)
3.	Lakehouse Support	A (Hudi, Delta, Iceberg, Paimon)	C (Нет прямых sinks или sources)	A (Hudi, Delta, Iceberg, Paimon)
4.	Catalog Support	B (Hive, Glue, Polaris)	C (Ограниченная нативная поддержка)	A (HMS, Glue, Unity Catalog, Polaris)
5.	Операционная простота Operational simplicity	B (Крутая кривая обучения)	C (Сложно, плюс управление Kafka-топиками)	A (Простая модель)
6.	Мониторинг Monitoring	A (Built-in Flink UI, Prometheus, Grafana)	B (Limited built-in metrics, полагается на Confluent Monitoring)	A (Native Spark UI, Datadog, CloudWatch)

В результате проведенного исследования экосистемной поддержки и операционной сложности установлено, что Structured Streaming является лучшим в классе (A) по операционной простоте и поддержке каталогов данных (Catalog Support), так как его модель проще для команд, уже использующих Spark.

Однако Flink (A) и Structured Streaming (A) демонстрируют равную первоклассную поддержку (A) форматов Lakehouse (Hudi, Delta, Iceberg, Paimon). Kafka Streams (C) имеет самые серьезные ограничения, включая нативную поддержку только Java/Scala и ограниченную интеграцию с каталогами данных. Use Cases и адаптация предусматривает:

1. Capital One: Компания перешла с Spark на Flink (с AWS KDA) для realtime processing в финансовом секторе. Flink был выбран за его realtime capabilities, эффективное управление памятью и SQL-based трансформации.
2. Lyft: запустил инициативу «Realtime Machine Learning with Streaming», используя Apache Flink для бесшовной интеграции потоковых данных в ML-платформу LyftLearn. Flink был центральным элементом для вычисления realtime features, realtime learning и принятия event-driven decisions.
3. Uber: преимущественно использует Apache Spark для критических бизнес-функций, работая на более чем 10,000 узлах и потребляя более 95% вычислительных ресурсов кластера аналитики для обработки сотен петабайт данных ежедневно.

В настоящее время существуют проблемы Flink на уровне реализации, такие как высокое потребление CPU и чувствительность к сетевым сбоям, хотя эти недостатки уменьшаются с каждой новой апробацией. Apache Flink зарекомендовал себя как лучшее решение для компаний, ориентированных на непрерывную обработку потоковых данных. Его способность эффективно управлять сложной логистикой обработки с высокой пропускной способностью, минимальной задержкой и поддержкой продвинутых операций с состоянием (stateful processing) утвердила Flink в качестве стандарта де-факто в области потоковой аналитики. Дополнительным преимуществом платформы является широкая поддержка языков программирования , включая Java, Python и SQL, что делает её доступной для разработчиков с разным уровнем подготовки и расширяет сферу применения в реальных проектах [7].

Spark Structured Streaming (SS) сталкивается с ограничениями из-за своей микропакетной архитектуры. SS также менее эффективен для stateful операций и имеет более высокую стоимость ресурсов по сравнению с Flink, который создан специально для потоковой обработки. Таким образом, Apache Flink предоставляет более универсальное и эффективное решение, хотя Spark Structured Streaming сохраняет свое место для пакетно-ориентированных аналитических рабочих нагрузок.

В экосистеме Hadoop для обеспечения безопасности кластера используется Ranger. Ranger является комплексным ресурсом для обеспечения безопасности кластера Hadoop, охватывающим авторизацию, учет и защиту данных. Плагины Ranger доступны для таких сервисов, как HBase, HDFS, Hive, Impala, Ozone, Solr, Spark3, Trino и YARN. Impala поддерживает аутентификацию Kerberos или LDAP. YARN используется как кластерный менеджер ресурсов для Hive и Impala, управляя ресурсами кластера Hadoop.

Вывод. Исходя из результатов исследования, можно сформулировать следующие рекомендации.

Рекомендации по выбору движка.

1. Для интерактивной аналитики (OLAP), Ad-hoc запросов и BI дашбордов: Impala является оптимальным выбором. Impala обеспечивает самую низкую задержку и идеально подходит для бизнес-интерактивных рабочих нагрузок.
2. Для корпоративных хранилищ данных (EDW) и долговременных ETL-задач: Hive LLAP - наилучший выбор. Hive был построен для сложности (sophistication) и его отказоустойчивость критически важна для долго выполняющихся запросов.
3. Для Data Engineering и построения Spark Pipelines: Spark SQL - отличный выбор для долго выполняющихся запросов и аналитических целей.

Impala оптимальна для задач, требующих быстрых интерактивных запросов, тогда как Hive лучше подходит для выполнения сложных пакетных операций. Вместе они охватывают широкий спектр аналитических сценариев от оперативной аналитики до глубокой обработки больших данных.

Критерии выбора SQL-движка:

- Если требуется интерактивный и быстрый доступ к данным с низкой задержкой (например, для BI), следует выбрать Impala.
- Если требуются сложные, длительные ETL-процессы с высокой отказоустойчивостью, предпочтителен Hive.
- Если необходима гибкость и возможность переключения между программированием и SQL, Spark SQL - отличный выбор.

Критерии выбора потокового движка:

- Для ультранизкой задержки и stateful workloads (fraud detection, IoT) рекомендуется Flink.
- Для высокой пропускной способности и глубокой интеграции с Lakehouse - Spark Structured Streaming.
- Для легковесных, Kafkanative микросервисов — Kafka Streams.

Полностью заменять Apache Spark на Flink преждевременно из-за потенциальной потребности в дополнительной инфраструктуре, но Flink является жизнеспособным решением для задач, требующих минимального времени отклика.

В экосистеме Hadoop для обеспечения безопасности кластера используется Ranger. Ranger является комплексным ресурсом для обеспечения безопасности кластера Hadoop, охватывающим авторизацию, учет и защиту данных. Плагины Ranger доступны для таких сервисов, как HBase, HDFS, Hive, Impala, Ozone, Solr, Spark3, Trino и YARN. Impala поддерживает аутентификацию Kerberos или LDAP.

YARN используется как кластерный менеджер ресурсов для Hive и Impala, управляя ресурсами кластера Hadoop. Для интерактивной аналитики (OLAP), Ad-hoc запросов и BI дашбордов: Impala является оптимальным выбором. Impala обеспечивает самую низкую задержку и идеально подходит для бизнес-интерактивных рабочих нагрузок. Для корпоративных хранилищ данных (EDW) и долговременных ETL-задач: Hive LLAP — наилучший выбор. Hive был построен для сложности (sophistication) и его отказоустойчивость критически важна для долго выполняющихся запросов. Для Data Engineering и построения Spark Pipelines: Spark SQL - отличный выбор для долго выполняющихся запросов и аналитических целей.

Для потоковой обработки данных Apache Flink демонстрирует превосходство в сценариях, требующих ультранизкой задержки, последовательно достигая результатов менее 100 миллисекунд для сложных операций, тогда как Spark Structured Streaming предлагает непревзойденную пропускную способность для крупномасштабной аналитики, а Kafka Streams – самую простую операционную модель для легковесных Kafka-центричных микросервисов.

При выборе SQL-движка для интерактивной аналитики (OLAP) рекомендуется использовать Cloudera Impala, который обеспечивает низкую задержку и может работать в 6-69 раз быстрее Hive с простыми запросами, оставляя Hive LLAP лучшим выбором для длительных ETL-задач с критической отказоустойчивостью.

Библиографический список:

1. Шейк А.С. Достижения В Обработке Потоков В Реальном Времени: Сравнительное Исследование Apache Flink, Spark Streaming И Kafka Streams //Международный Журнал Компьютерной Инженерии И Технологий (IJCET). – 2024. – Т. 15. – №. 6. – Дата обращения: 10.11.2025.
2. Лакшмипати С. Apache Flink vs Apache Kafka Streams vs Apache Spark Structured Streaming — Сравнение движков обработки потоков [Электронный ресурс] // Onehouse Blog. – Рекомендации по кибербезопасности AA23-059A. – 2025. – Режим доступа: https://www.onehouse.ai/blog/apache-spark-structured-streaming-vs-apache-flink-vs-apache-kafka-streams-comparing-stream-processing-engines?, свободный. – Дата обращения: 10.11.2025.

3. Кевалрамани С. Impala против Hive против Spark SQL: Выбор SQL-движка для совместной работы в хранилище данных Cloudera [Электронный ресурс] / Пер. с англ. английский. mongohtotech // 29 января 2020. Режим доступа: https://habr.com/ru/articles/486124/. – Дата обращения: 10.11.2025.
4. ProjectPro. Impala vs Hive: разница между Sql и компонентами Hadoop [Электронный ресурс] // Блог ProjectPro. Обновлено 28 октября 2024 г. Режим доступа: https://www.projectpro.io/article/impala-vs-hive-difference-between-sql-on-hadoop-components/180. – Дата обращения: 11.11.2025.
5. Мааян Г. Д. Spark vs. Flink: ключевые отличия и как выбрать [Электронный ресурс] // Dataversity. 8 мая 2023 г. Режим доступа: https://www.dataversity.net/articles/spark-vs-flink-key-differences-and-how-to-choose/?. – Дата обращения: 11.11.2025.
6. Потоковая обработка с помощью Apache Flink: Руководство для начинающих 2025 [Электронный ресурс] // Ververica. [Электронный ресурс]. Режим доступа: https://www.ververica.com/stream-processing-with-apache-flink-beginners-guide. – Дата обращения: 11.11.2025.
7. Венер К. Основные тенденции в области потоковой передачи данных с помощью Apache Kafka и Flink в 2025 году [Электронный ресурс] // Блог Кая Венера. 2 декабря 2024 г. Режим доступа: https://www.kai-waehner.de/blog/2024/12/02/top-trends-for-data-streaming-with-apache-kafka-and-flink-in-2025/?. – Дата обращения: 11.11.2025.
8. Ашырова Ю. Управление данными и их анализ в BIG DATA // Символ науки. 2025. № 1-1-2. URL: https://cyberleninka.ru/article/n/data-management-and-analysis-in-big-data. - Дата обращения: 12.11.2025.
9. Белов В.А., Никульчев Е.В. Экспериментальная оценка временной эффективности обработки больших данных в заданных форматах хранения // International Journal of Open Information Technologies. 2021. № 9. URL: https://cyberleninka.ru/article/n/eksperimentalnaya-otsenka-vremennoy-effektivnosti-obrabotki-bolshih-dannyh-v-zadannyh-formatah-hraneniya. – Дата обращения: 10.11.2025.
10. Алтыев Аганазар, Бекдурдыев Гурбанназар, Атаев Аллаберди, Тачев Юсуп Компьютерные технологии для обработки больших данных // Наука и мировоззрение. 2025. № 41. URL: https://cyberleninka.ru/article/n/kompyuternye-tehnologii-dlya-obrabotki-bolshih-dannyh. – Дата обращения: 12.11.2025.
11. Смайлов Н.К. Методы и технологии оценки и управления качеством данных // Экономика и социум. 2025. № 8 (135). URL: https://cyberleninka.ru/article/n/metody-i-tehnologii-otsenki-i-upravleniya-kachestvom-dannyh. – Дата обращения: 12.11.2025.
12. Упаева П.В. Оптимизация ETL-процессов: обзор отечественного рынка // Вестник науки. 2024. № 6 (75). URL: https://cyberleninka.ru/article/n/optimizatsiya-etl-protsessov-obzor-otechestvennogo-rynka. – Дата обращения: 12.11.2025.
13. С.Г. Ермаков, М.М. Халил, А.Д. Хомоненко, В.А. Гончаренко, В.А. Ходаковский, Р. Абу Хасан Переход от хранилища данных университета к озеру: модели и методы обработки больших данных // ИВД. 2024. № 12 (120). URL: https://cyberleninka.ru/article/n/perehod-ot-hranilischa-dannyh-universiteta-k-ozeru-modeli-i-metody-obrabotki-bolshih-dannyh. – Дата обращения: 12.11.2025.
14. Almeida, A., Brás, S., Sargento, S. et al. Time series big data: a survey on data stream frameworks, analysis and algorithms. J Big Data 10, 83. URL:https://doi.org/10.1186/s40537-023-00760-1. – Дата обращения: 12.11.2025.
15. Овездурдыева Ирина Курбангельдыевна, Мырадов Максат Тачмухаммедович. Технологии работы с большими данными “big data”: сбор, хранение и обработка больших данных // Наука и мировоззрение. 2024. № 29. URL: https://cyberleninka.ru/article/n/tehnologii-raboty-s-bolshimi-dannymi-big-data-sbor-hranenie-i-obrabotka-bolshih-dannyh. – Дата обращения: 12.11.2025.
16. Кузина Е. Обзор архитектуры Impala ADH Arenadata Docs [Электронный ресурс]. – Режим доступа: https://docs.arenadata.io/ru/ADH/current/concept/impala/impala.html. – Дата обращения: 12.11.2025.
17. Егорцев В. Работа с большими данными: введение в Apache Hadoop и Spark [Электронный ресурс] // Tproger. [Б.г.]. Режим доступа: https://tproger.ru/articles/rabota-s-bolwimi-dannymi--vvedenie-v-apache-hadoop-i-spark. – Дата обращения: 12.11.2025.

References:

1. Shaik A.S. Advancements In Real-Time Stream Processing: A Comparative Study Of Apache Flink, Spark Streaming, And Kafka Streams. *International Journal Of Computer Engineering And Technology (IJCET)*. 2024;15(6). Accessed: 10.11.2025.
2. Lakshmipathy S. Apache Flink vs Apache Kafka Streams vs Apache Spark Structured Streaming — Comparing Stream Processing Engines [Электронный ресурс]. Onehouse Blog. – Cybersecurity Advisory AA23-059A. – 2025. – Режим доступа: https://www.onehouse.ai/blog/apache-spark-structured-streaming-vs-apache-flink-vs-apache-kafka-streams-comparing-stream-processing-engines?, Accessed: 10.11.2025.
3. Kewalramani S. Impala vs Hive vs Spark SQL: Выбор правильного SQL движка для правильной работы в Cloudera Data Warehouse [Электронный ресурс] / Пер. с англ. mongohtotech // Хабр. 29 янв. 2020. Режим доступа: https://habr.com/ru/articles/486124/ Accessed: 10.11.2025.

4. ProjectPro. Impala vs Hive: Difference between Sql on Hadoop components [Электронный ресурс] // ProjectPro Blog. Обновлено 28 окт. 2024. Режим доступа: https://www.projectpro.io/article/impala-vs-hive-difference-between-sql-on-hadoop-components/180. Accessed: 11.11.2025.
5. Maayan G.D. Spark vs. Flink: Key Differences and How to Choose [Электронный ресурс] // Dataversity. 8 мая 2023. Режим доступа: https://www.dataversity.net/articles/spark-vs-flink-key-differences-and-how-to-choose/?. Accessed: 11.11.2025.
6. Stream Processing with Apache Flink: Beginner's Guide 2025 [Электронный ресурс] // Ververica. [Б.г.]. Режим доступа: https://www.ververica.com/stream-processing-with-apache-flink-beginners-guide. Accessed: 11.11.2025.
7. Waehner K. Top Trends for Data Streaming with Apache Kafka and Flink in 2025 [Электронный ресурс] // Kai Waehner Blog. 2 дек. 2024. Режим доступа: https://www.kai-waehner.de/blog/2024/12/02/top-trends-for-data-streaming-with-apache-kafka-and-flink-in-2025/?. Accessed: 11.11.2025.
8. Ashyrova Y. Data management and analysis in BIG DATA. *Symbol of Science.* 2025. №1-1-2. URL: https://cyberleninka.ru/article/n/data-management-and-analysis-in-big-data. Accessed: 12.11.2025.
9. Belov V.A., Nikulchev E.V. Experimental Evaluation of the Time Efficiency of Processing Large Data in Specified Storage Formats. *International Journal of Open Information Technologies*. 2021; 9. URL: https://cyberleninka.ru/article/n/eksperimentalnaya-otsenka-vremennoy-effektivnosti-obrabotki-bolshih-dannyh-v-zadannyh-formatah-hraneniya. – Accessed: 10.11.2025. (In Russ)
10. Altiev Aganazar, Bekdurdyev Gurbannazar, Ataev Allaberdi, Tachev Yusup Computer Technologies for Processing Big Data. *Science and Worldview*. 2025; 41. URL: https://cyberleninka.ru/article/n/kompyuternye-tehnologii-dlya-obrabotki-bolshih-dannyh. – Accessed: 12.11.2025.
11. Smilov N.K. Methods and Technologies of Data Quality Assessment and Management. *Economics and Sociology.* 2025;8(135). https://cyberleninka.ru/article/n/metody-i-tehnologii-otsenki-i-upravleniya-kachestvom-dannyh. – Accessed: 12.11.2025.
12. Upaeva P.V. Optimization of ETL Processes: A Review of the Domestic Market. *Bulletin of Science*. 2024; 6 (75). URL: https://cyberleninka.ru/article/n/optimizatsiya-etl-protsessov-obzor-otechestvennogo-rynka. – Accessed: 11/12/2025.
13. S.G. Ermakov, M.M. Khalil, A.D. Khomonenko, V.A. Goncharenko, V.A. Khodakovsky, R. Abu Hassan The transition from the university data warehouse to the lake: models and methods of big data processing. IVD. 2024; 2(120). URL: https://cyberleninka.ru/article/n/perehod-ot-hranilischa-dannyh-universiteta-k-ozeru-modeli-i-metody-obrabotki-bolshih-dannyh. – Date of appeal: 12.11.2025.
14. Almeida, A., Brás, S., Sargento, S. et al. Time series big data: a survey on data stream frameworks, analysis and algorithms. J Big Data 10, 83. URL:https://doi.org/10.1186/s40537-023-00760-1. – Date of access: 12.11.2025.
15. Ovezdurdyeva Irina Kurbangeldievna, Myradov Maksat Tachmukhammedovich Technologies of working with big data “big data”: collection, storage and processing of big data // Science and worldview. 2024. No. 29. https://cyberleninka.ru/article/n/tehnologii-raboty-s-bolshimi-dannymi-big-data-sbor-hranenie-i-obrabotka-bolshih-dannyh. – Accessed: 12.11.2025.
16. Kuzina E. Overview of Impala Architecture ADH Arenadata Docs [Electronic resource]. – Access mode: https://docs.arenadata.io/ru/ADH/current/concept/impala/impala.html. – Date of access: 12.11.2025. (In Russ)
17. Egorzev V. Working with Big Data: Introduction to Apache Hadoop and Spark [Electronic resource] // Tproger. [B.g.]. Access mode: https://tproger.ru/articles/rabota-s-bolwimi-dannymi--vvedenie-v-apache-hadoop-i-spark. – Accessed: 12.11.2025. (In Russ)

Сведения об авторах:

Алексеева Анна Николаевна, студент, кафедра информационной безопасности; 241266@edu.fa.ru, ORCID 0009-0004-1417-7779

Резниченко Сергей Анатольевич, кандидат технических наук, доцент, кафедра информационной безопасности; rsa_5@bk.ru, ORCID 0000-0002-1539-0457

Information about authors:

Anna N. Alexeeva, Student, Department of Information Security; 241266@edu.fa.ru, ORCID 0009-0004-1417-7779

Sergey A. Reznichenko, Cand.Sci. (Eng.), Assoc. Prof., Department of Information Security; rsa_5@bk.ru, ORCID 0000-0002-1539-0457

Конфликт интересов/Conflict of interest.

Авторы заявляют об отсутствии конфликта интересов/The authors declare no conflict of interest.

Поступила в редакцию/Received 10.12.2025**.**

Одобрена после рецензирования/Reviced 29.01.2026.

Принята в печать/Accepted for publication 24.04.2026.