# аналитическая записка по spark

# из экосистемы hadoop

1. **Краткий обзор**

Spark – это платформа с открытым исходным кодом, ориентированная на интерактивные запросы, машинное обучение и рабочие нагрузки в реальном времени. У нее нет собственной системы хранения, но она выполняет аналитику в других системах хранения данных, таких как HDFS, или в иных популярных хранилищах, таких как Amazon Redshift, Amazon S3, Couchbase, Cassandra и других.

Spark использует идею локальности данных, однако выносит большинство вычислений в память вместо диска. Ключевым понятием в Spark-е является RDD (resilient distributed dataset) — указатель на ленивую распределённую коллекцию данных. Благодаря RDD Apache Spark запоминает, как извлекает определенную информацию из хранилища, и может восстановить данные в случае сбоя основного хранилища. Большинство операций над RDD не приводит к каким-либо вычислениям, а только создаёт очередную обёртку, обещая выполнить операции только тогда, когда они понадобятся.

1. **Общий обзор для IT специалиста**

Hadoop MapReduce – это модель программирования для обработки больших наборов данных с помощью параллельного распределенного алгоритма. Разработчики могут писать многократно распараллеленные операторы, не беспокоясь о распределении работы и отказоустойчивости. Однако проблема MapReduce заключается в последовательном многоэтапном процессе выполнения задания. На каждом шаге MapReduce считывает данные из кластера, выполняет операции и записывает результаты обратно в HDFS. Поскольку каждый шаг требует чтения и записи на диск, задания MapReduce выполняются медленнее из-за задержки операции ввода-вывода на диск.

Spark был создан для устранения ограничений MapReduce за счет обработки в памяти, сокращения количества шагов в задании и повторного использования данных в нескольких параллельных операциях. При использовании Spark требуется всего один шаг для считывания данных в память, выполнения операций и обратной записи результатов, что значительно ускоряет выполнение. Spark также повторно использует данные, применяя кэш в памяти для значительного ускорения алгоритмов машинного обучения, которые многократно вызывают функцию в одном и том же наборе данных. Повторное использование данных осуществляется путем создания DataFrames, абстракции по сравнению с устойчивым распределенным набором данных (RDD), представляющим собой набор объектов, кэшируемых в памяти и повторно используемых в нескольких операциях Spark. Это значительно снижает задержку, благодаря чему Spark в несколько раз быстрее MapReduce, особенно при машинном обучении и интерактивной аналитике.

Платформа Spark включает в себя:

*Spark Core*

Spark Core является основой платформы – движком. Он отвечает за управление памятью, устранение неисправностей, планирование, распределение и мониторинг заданий, а также за взаимодействие с системами хранения данных. Spark Core доступен через интерфейс прикладного программирования (API), созданный для Java, Scala, Python и R. Эти API скрывают сложность распределенной обработки за простыми высокоуровневыми операторами.

*MLlib -* Машинное обучение

Spark включает в себя MLlib – библиотеку алгоритмов, которые подходят для машинного обучения любых масштабов. Специалисты по обработке данных могут обучать модели машинного обучения на R или Python на любом источнике данных Hadoop, сохранять их с помощью MLlib, а также импортировать их в конвейер на основе Java либо Scala. Spark был разработан для быстрых интерактивных вычислений, производимых в памяти, что позволяет быстро выполнять операции машинного обучения. Алгоритмы включают возможность классификации, регрессии, кластеризации, совместной фильтрации и анализа закономерностей.

*Spark Streaming -* Режим реального времени

Spark Streaming – это решение, работающее в режиме реального времени, которое использует функцию быстрого планирования Spark Core для анализа потоковой передачи. Это решение принимает данные в виде мини-пакетов и дает возможность анализировать их с помощью того же прикладного кода, написанного для пакетной аналитики. Это повышает производительность разработчиков, поскольку они могут использовать один и тот же код для пакетной обработки и для приложений потоковой передачи в реальном времени. Spark Streaming поддерживает данные из приложений Twitter, Kafka, Flume, HDFS и ZeroMQ, а также многие другие данные из экосистемы Spark Packages.

*Spark SQL -* Интерактивные запросы

Spark SQL – это распределенный механизм запросов, который обрабатывает интерактивные запросы с низкой задержкой почти в 100 раз быстрее, чем MapReduce. Он включает экономичный оптимизатор, столбцовое хранилище и генерацию кода для быстрых запросов с возможностью масштабирования до тысяч узлов. Для запросов к данным бизнес-аналитики могут использовать стандартный SQL или язык запросов Hive. Разработчики могут использовать API, доступные на языках Scala, Java, Python и R. Они поддерживают различные встроенные источники данных, включая JDBC, ODBC, JSON, HDFS, Hive, ORC и Parquet. Другие популярные хранилища – Amazon Redshift, Amazon S3, Couchbase, Cassandra, MongoDB, Salesforce.com, Elasticsearch и многие другие – можно найти в экосистеме Spark Packages.

*GraphX -* Обработка графов

Spark GraphX – это распределенная среда обработки графов на основе Spark. GraphX предоставляет ETL, исследовательский анализ и итеративные вычисления на графах, позволяющие пользователям интерактивно создавать и преобразовывать графовую структуру данных в любом масштабе. Эта среда поставляется с очень гибким API и набором распределенных алгоритмов Graph.

1. **Плюсы и минусы Spark**

*Преимущества:*

*Скорость*

Spark может обеспечивать быстрое выполнение аналитических запросов к данным любого объема благодаря кэшированию в памяти и оптимизированному исполнению запросов.

*Удобство для разработчиков*

Apache Spark по умолчанию поддерживает Java, Scala, R и Python, что позволяет выбирать язык для написания приложений. Эти API упрощают работу разработчиков, поскольку скрывают сложность распределенной обработки за простыми высокоуровневыми операторами, что значительно сокращает объем требуемого кода.

*Поддержка нескольких рабочих нагрузок*

Apache Spark поддерживает несколько рабочих нагрузок, включая интерактивные запросы, аналитику в реальном времени, машинное обучение и обработку графов. Одно приложение может легко комбинировать несколько рабочих нагрузок.

*Spark UI: Удобный интегрированный веб-интерфейс пользователя*

Apache Spark выделяется на фоне других платформ с открытым исходным кодом благодаря интегрированному пользовательскому веб-интерфейсу, известному как Spark UI. Этот инструмент позволяет специалистам следить за состоянием своих кластеров Spark и потреблением ресурсов.

*Возможности расширенной аналитики*

Расширенные аналитические возможности Spark - еще одно его значительное преимущество. Он включает в себя обширный набор библиотек, в том числе Spark SQL для обработки структурированных данных, MLlib для машинного обучения, GraphX для обработки графов и Spark Streaming. Эти встроенные библиотеки позволяют пользователям выполнять сложные задачи - от аналитики в реальном времени до сложного машинного обучения и вычислений на графах.

*Подробная документация*

Одним из существенных преимуществ Apache Spark является подробная и исчерпывающая документация. Это обширное руководство охватывает все аспекты архитектуры, API и библиотек Spark. Оно окажется бесценным для разработчиков с разным уровнем опыта.

Недостатки Spark.

*Высокое потребление памяти и увеличение стоимости оборудования*

По своей сути Apache Spark является ресурсоемкой системой, особенно в отношении памяти. Он в значительной степени использует оперативную память для вычислений в памяти, что позволяет ему обеспечивать высокую скорость обработки данных. Однако эта особенность также может привести к высокому потреблению памяти, что влечет за собой увеличение эксплуатационных расходов, ограничения по масштабируемости и проблемы для приложений, работающих с большими объемами данных.

Большая зависимость от оперативной памяти также означает, что для Spark может потребоваться более дорогое оборудование, чем для решений на базе дисков, таких как Hadoop MapReduce. Необходимость в больших объемах оперативной памяти и более быстрых процессорах может увеличить первоначальные затраты на создание среды Spark, особенно для задач обработки больших объемов данных.

*Ограниченная поддержка обработки в реальном времени*

Хотя Apache Spark часто рекламируют за возможности обработки данных в режиме, близком к реальному времени, он не способен обрабатывать данные в реальном времени. Это связано с тем, что Spark Streaming опирается на микропакеты, которые обрабатывают небольшие группы событий, собранных за заранее определенный интервал времени. Такие инструменты, как Apache Flink или Apache Storm, могут лучше подойти для приложений, требующих реальной потоковой аналитики.

*Тяжелая кривая обучения*

Хотя Apache Spark обладает неоспоримо мощными возможностями, один из аспектов, который часто всплывает среди пользователей, - это трудная кривая обучения. Многие пользователи на таких платформах, как Reddit и Quora, отмечают, что, хотя основы Spark, такие как DataFrames и наборы данных, относительно просты, по-настоящему понять и использовать Spark может быть сложнее.

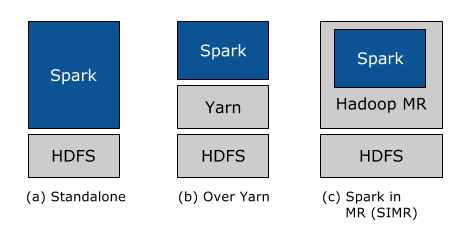
Такие ключевые понятия, как распределенное хранение, обработка в памяти, стоимость обмена данными, форматирование столбцов и преимущества хранения таблиц, в числе прочего, составляют основу возможностей Spark. Эти принципы, несмотря на их важность, могут представлять значительную трудность для новичков в области обработки данных.

Тем не менее, это не является непреодолимой проблемой. Многие пользователи также утверждают, что изучение Apache Spark может быть более легким при наличии правильного руководства, четкого понимания сценариев использования и практического опыта. Более того, если у вас уже есть базовые знания Python, изучение Spark с помощью PySpark может оказаться более простым путем, чем постижение тонкостей таких систем, как Hadoop.

*Проблема маленьких файлов*

Apache Spark сталкивается с трудностями при работе с большим количеством маленьких файлов. Большее количество файлов в рабочей нагрузке означает больше метаданных для разбора и больше задач для планирования, что может значительно замедлить обработку.

1. **Примерный набор экосистемы для эффективного использования Spark**



Автономное развертывание (Standalone deployment): При автономном развертывании можно статически выделить ресурсы на всех или на части машин в кластере Hadoop и запустить Spark бок о бок с Hadoop MR. Пользователь может запускать произвольные задания Spark на своих данных HDFS. Благодаря своей простоте этот вариант развертывания является предпочтительным для многих пользователей Hadoop 1.x.

Развертывание Hadoop Yarn (Over YARN): Пользователи Hadoop, которые уже развернули или планируют развернуть Hadoop Yarn, могут просто запускать Spark на YARN без предварительной установки или административного доступа. Это позволяет пользователям легко интегрировать Spark в свой стек Hadoop и использовать всю мощь Spark, а также других компонентов, работающих поверх Spark.

Spark In MapReduce (SIMR): Для пользователей Hadoop, которые еще не используют YARN, еще одним вариантом, помимо автономного развертывания, является использование SIMR для запуска заданий Spark внутри MapReduce. С SIMR пользователи могут начать экспериментировать со Spark и использовать его оболочку уже через пару минут после загрузки! Это значительно снижает барьер развертывания и позволяет практически всем играть со Spark.

Spark взаимодействует не только с Hadoop, но и с другими популярными технологиями работы с большими данными.

*Apache Hive*: Благодаря Shark, Spark позволяет пользователям Apache Hive выполнять свои немодифицированные запросы гораздо быстрее. Hive - это популярное решение для хранения данных, работающее поверх Hadoop, а Shark - это система, которая позволяет фреймворку Hive работать поверх Spark, а не Hadoop. В результате Shark может ускорить запросы Hive в 100 раз, если входные данные помещаются в память, и в 10 раз, если входные данные хранятся на диске.

*AWS EC2*: Пользователи могут легко запускать Spark (и Shark) поверх Amazon EC2, используя скрипты, поставляемые вместе со Spark, или размещенные версии Spark и Shark на Amazon's Elastic MapReduce.

*Apache Mesos*: Spark работает поверх Mesos, системы управления кластерами, которая обеспечивает эффективную изоляцию ресурсов в распределенных приложениях, включая MPI и Hadoop. Mesos обеспечивает мелкозернистое разделение, что позволяет заданию Spark динамически использовать незадействованные ресурсы кластера во время его выполнения. Это приводит к значительному повышению производительности, особенно для долго выполняющихся заданий Spark.

1. **Описание целей использования Spark**

Она используется во всех типах сценариев использования больших данных для выявления закономерностей и предоставления аналитических данных в режиме реального времени. Примеры использования приведены ниже.

*Финансовые сервисы*

Spark используют в банковской сфере для прогнозирования оттока клиентов и предоставления рекомендаций новых финансовых продуктов. В банковских инвестициях Spark используют для анализа цен на акции с целью прогнозирования будущих тенденций.

*Здравоохранение*

Spark используют сотрудники организаций комплексного ухода за пациентами для предоставления данных о каждом взаимодействии с пациентом своим медицинским работникам первичного звена. Spark также можно использовать для прогнозирования и предоставления рекомендаций для лечения пациентов.

*Обрабатывающая промышленность*

Spark используют для предотвращения простоев подключенного к Интернету оборудования, рекомендуя, когда проводить профилактическое обслуживание.

*Розничная торговля*

Spark используется для привлечения и удержания клиентов с помощью персонализированных услуг и предложений.

Зависимость от внешних систем хранения данных: Слабость?

Отношения Apache Spark с внешними системами хранения данных - это отличительный аспект, который может быть как преимуществом, так и недостатком.

С одной стороны, хотя Spark не имеет встроенной системы хранения данных, его архитектура рассчитана на взаимодействие с различными решениями для хранения данных, такими как HDFS, Amazon S3, Apache Cassandra или Apache HBase. Это позволяет пользователям выбирать систему, которая лучше всего соответствует их конкретным потребностям и среде, что повышает гибкость Spark.

1. **Список литературы**

Что такое Apache Spark? URL: <https://aws.amazon.com/ru/what-is/apache-spark/>

В чем разница между Hadoop и Spark? URL: <https://aws.amazon.com/ru/compare/the-difference-between-hadoop-vs-spark/>

Apache Spark and Hadoop: Working Together. URL: <https://www.databricks.com/blog/2014/01/21/spark-and-hadoop.html>

The Good and the Bad of Apache Spark Big Data Processing URL: <https://www.altexsoft.com/blog/apache-spark-pros-cons/>