MHBD (Методы хранения больших данных)

Экзаменационные вопросы 2025

1. Что такое большие данные. Характеристики больших данных (3V, 4V)

以下是关于**大数据（Big Data）**及其特性的俄中对照回答：

1. 什么是大数据 (Что такое большие данные)

Большие данные (Big Data) относится к наборам данных, размер которых выходит за пределы возможностей программных средств типичной базы данных собирать, хранить, обрабатывать и анализировать данные. **大数据 (Big Data)** 是指其规模超出了典型数据库软件工具在采集、存储、管理和分析方面能力的资料群。

- **История термина (术语历史):** Термин был предложен Клиффордом Линчем, редактором журнала *Nature*, в сентябре 2008 года по аналогии с метафорами «большая нефть» или «большая руда». 该术语由《自然》杂志编辑克利福德·林奇 (Clifford Lynch) 于2008年9月提出，类比了“大石油”或“大矿石”等商业隐喻。

---

2. 大数据的特性 (Характеристики больших данных)

通常使用 **V模型** 来描述大数据的特性。最初的经典模型是 **3V**，随后演变为 **4V**（甚至更多）。 Для описания характеристик больших данных обычно используется **V-модель**. Изначально классической была модель **3V**, которая затем расширилась до **4V** (и более).

3V 模型 (Модель 3V):

1. **Объём (Volume) / 数据量:**
   - Огромное количество накопленных данных (от терабайтов до зеттабайтов). Например, в Сбербанке к 2020 году было накоплено более 120 петабайт данных.
   - 海量的累积数据（从太字节到泽字节）。例如，到2020年，俄罗斯联邦储蓄银行已积累了超过120PB的数据。

2. **Разнообразие (Variety) / 多样性:**
    - Способность обработки разнообразных типов данных: **структурированные** (таблицы в СУБД), **слабоструктурированные** (XML, JSON) и **неструктурированные** (тексты, аудио, видео, данные с сенсоров).
    - 能够处理各种类型的数据：**结构化数据**（数据库表格）、**半结构化数据**（XML, JSON）以及**非结构化数据**（文本、音频、视频、传感器数据）。
3. **Скорость (Velocity) / 速度:**
    - Высокая скорость поступления и изменения данных, требующая реакции в реальном или близком к реальному времени (например, потоковая обработка GPS-данных).
    - 数据输入和变化的极快速度，需要实时或准实时地做出反应（例如，GPS数据的流处理）。

4V 模型 (Модель 4V - добавляет следующий пункт):

4. **Достоверность (Veracity) / 真实性:**
    - Вопрос «чистоты» и точности данных. При работе с большими объемами информация может быть неполной, содержать шумы или аномалии, поэтому важна верификация для верного анализа.
    - 关于数据的“洁净度”和准确性的问题。在处理海量数据时，信息可能是不完整的、包含噪声或异常值的，因此为了进行正确分析，数据的验证至关重要。

*注：在某些模型中还会加入第五个特性——**Стоимость (Value / 价值)**，指从数据中获取商业利益或发现新机会的能力。*

2. Сетевые и распределенные файловые системы

以下是关于**网络和分布式文件系统（Сетевые и распределенные файловые системы）**及其核心技术（特别是 HDFS）的俄中对照详细解析：

1. 分布式文件系统概述 (Обзор распределенных файловых систем)

分布式文件系统允许数据存储在物理上分散的多个节点上。 Распределенные файловые системы позволяют хранить данные на нескольких физически разделенных узлах.

- **核心架构 (Архитектура):** 通常采用**无共享架构 (Shared Nothing Architecture, SNA)**，其中每个节点拥有独立的内存、磁盘阵列和输入输出设备。在这种架构中，每个节点都是自给自足的，不与其他节点共享硬件资源。 Обычно используется архитектура **Shared Nothing (SNA)**, в которой отдельные узлы имеют собственную память, дисковые массивы и устройства ввода/вывода. Каждый узел в такой архитектуре самодостаточен и ничего не делится с другими узлами сети.

- **典型示例 (Примеры):** 常见的系统包括 **AFS** (Andrew File System)、**GFS** (Google File System) 和 **HDFS** (Hadoop Distributed File System)。 Типичные примеры включают **AFS** (Andrew File System), **GFS** (Google File System) и **HDFS** (Hadoop Distributed File System).

---

2. HDFS 的设计与特性 (Проектирование и характеристики HDFS)

HDFS 是大数据生态系统中最核心的分布式文件系统。 **HDFS** является основной распределенной файловой системой в экосистеме больших данных.

- **设计目标 (Цели разработки):** 专门设计用于处理**超大文件**，并支持**流式数据访问模式**。 Разработан для обработки **очень больших файлов** с использованием **потоковых шаблонов доступа** к данным.
- **存储单位——块 (Блоки данных):** 数据被切分为固定大小的**块 (Blocks)** 存储。默认大小通常为 **64MB**，但生产环境建议使用 **128MB**。这种设计便于计算数据量，并允许单个文件的大小超过任何单块磁盘的容量。 Данные делятся на **блоки** фиксированного размера. По умолчанию размер составляет **64 МБ**, но рекомендуется **128 МБ**. Это позволяет легко подсчитать объем данных, а размер файла может превышать емкость любого отдельного диска в сети.
- **副本机制与容错 (Репликация и отказоустойчивость):** 为了确保存储的**高可用性**，数据块会进行**副本复制 (Replication)**。默认情况下会保存 **3 个副本**。 Для обеспечения **высокой доступности** блоки данных **реплицируются**. По умолчанию создается **3 реплики**.
- **副本放置策略 (Стратегия размещения реплик):** 为了实现容错，副本通常会分发到**不同的服务器**甚至**不同的机架 (Rack)** 上。这样即使某个服务器或机架发生故障，也不会导致数据丢失。 Для отказоустойчивости реплики блоков копируются на **разные серверы** и даже в **разные стойки**. Отказ одного или нескольких серверов не ведет к потере данных.

---

3. 系统组件与扩展性 (Компоненты системы и масштабируемость)

- **管理节点 (NameNode):** 集群中的主节点（Master），负责管理元数据和各区域服务器（Region Servers）之间的分布。 **NameNode** — главный узел в кластере, управляющий метаданными и распределением между серверами регионов.
- **数据节点 (DataNode):** 实际存储数据块的节点。 **DataNode** — узлы, на которых физически хранятся блоки данных.
- **水平扩展 (Горизонтальное масштабирование):** 增加新服务器时不需要重新配置整个文件系统。**负载均衡器 (Balancer)** 会自动将现有数据块从繁忙节点迁移到新节点。 Добавление нового сервера не требует перенастройки файловой системы. **Балансировщик** нагрузки автоматически копирует часть существующих блоков со старых серверов на новый.

---

4. 交互与命令 (Взаимодействие и команды)

访问 HDFS 通常使用 **URI 格式**： `hdfs://authority/path` 。 Доступ к HDFS осуществляется через **URI формат**: `hdfs://authority/path` .

- **命令行操作 (Командная строка):** 用户可以使用类似本地操作系统的命令，例如： Пользователи могут использовать команды, аналогичные локальной ОС, например:
  - 列出文件: `hadoop fs -ls`
  - 上传文件: `hadoop fs -copyFromLocal`
  - 查看内容: `hadoop fs -cat`
  - 设置副本数: `hadoop fs -setrep`

3. Распределенные базы данных

以下是关于**分布式数据库（Распределенные базы данных）**的俄中对照详细解析：

1. 分布式架构 (Распределенная архитектура)

分布式数据库通常基于**无共享架构 (Shared Nothing Architecture, SNA)** 构建。 Распределенные базы данных часто строятся на базе архитектуры **Shared Nothing (SNA)**.

- **架构定义 (Определение архитектуры):** 在这种分布式计算架构中，每个节点都是**自给自足**的。 每个节点拥有独立的内存、磁盘阵列和输入输出设备，不与其他节点共享硬件资源。 В этой архитектуре распределенных вычислений каждый узел **самодостаточен**. Отдельные узлы имеют собственную память, дисковые массивы и устройства ввода/вывода, не разделяя их с другими узлами.
- **优势 (Преимущества):** 这种架构具有良好的**水平扩展性 (Горизонтальное масштабирование)**，正变得越来越流行。 Такая архитектура хорошо **масштабируется** и становится все более популярной.

---

2. 分布式数据库实例 (Примеры распределенных БД)

资料中重点介绍了两类主流的分布式 NoSQL 数据库：

HBase (分布式列存储数据库):

- **数据模型 (Модель данных):** 是一种**稀疏的、分布式的、多维有序映射**。 它专门为高吞吐量设计，支持每秒**数百万次请求**。 Это **разреженное, распределенное, многомерное отсортированное отображение**. Она обеспечивает высокую пропускную способность — **миллионы запросов в секунду**.

- **存储规模 (Масштаб хранения):** 最大数据容量可达 **PB (拍字节)** 级，而传统关系型数据库（РСУБД）通常仅支持 TB 级。 Максимальный размер данных достигает **Пб (петабайтов)**, в то время как традиционные РСУБД обычно работают с Тб.
- **底层支持 (Основа):** HBase 运行在 **HDFS**（分布式文件系统）之上，物理数据以 HFile 格式存储。 HBase работает поверх **HDFS**, данные физически хранятся в формате HFile.

MongoDB (面向文档的数据库):

- **特性 (Характеристики):** 这是一种**开源、高性能、可扩展**的分布式数据库。 Это **масштабируемая высокопроизводительная** база данных с открытым исходным кодом.
- **存储方式 (Способ хранения):** 它是**面向文档 (Документоориентированная)** 的，不需要像关系型数据库那样预先定义严格的表结构。 Она ориентирована на **документы**, что исключает необходимость строгого описания схемы таблиц.

---

3. 分布式组件与管理 (Компоненты и управление)

分布式数据库通过特定的组件来维持运行和数据一致性：

- **主节点 (Master/HMaster):** 负责管理集群中的**区域服务器 (Region Servers)**，并协调数据的恢复和负载均衡。 **Мастер сервер (Master)** управляет распределением регионов по серверам и координирует процессы восстановления или балансировки.
- **协调服务 (ZooKeeper):** 用于管理配置和实现服务的**同步**，监控所有区域服务器的状态。 **ZooKeeper** используется для управления конфигурациями и **синхронизации** сервисов, а также мониторинга доступности узлов.
- **容错机制 (Отказоустойчивость):** 块数据会在多个服务器上进行**副本复制 (Репликация)**。 即使一个或多个服务器发生故障，也不会导致数据丢失。 Блоки данных **реплицируются** на несколько серверов. Отказ одного или нескольких серверов не ведет к потере данных.

4. Распределенная файловая система Hadoop

以下是关于 **Hadoop 分布式文件系统 (HDFS)** 的俄中对照详细解析：

1. 定义与设计目标 (Определение и цели проектирования)

Hadoop 分布式文件系统 (HDFS) 是大数据生态系统中最核心的分布式文件系统。它专门设计用于在通用硬件集群上存储**超大文件**，并支持**流式数据访问模式**。 **Распределенная файловая система Hadoop (HDFS)** является основной распределенной файловой системой в экосистеме больших данных. Она разработана для обработки **очень больших файлов** с использованием **потоковых шаблонов доступа** к данным。

---

2. 数据存储机制：块 (Механизм хранения: Блоки)

- 数据分块 (Деление на блоки): HDFS 将文件切分为固定大小的块 (Blocks) 进行存储。 HDFS делит файлы на **блоки** фиксированного размера для хранения。
- 块大小 (Размер блока): 默认块大小通常为 **64MB**，但在实际应用中建议使用 **128MB**。相比之下，普通操作系统（如 UNIX）的块大小仅为 4KB。 Размер блока по умолчанию составляет **64 МБ**, но рекомендуется **128 МБ**. Для сравнения, размер блока в обычных ОС (например, UNIX) составляет 4 КБ。
- 优势 (Преимущества): 固定大小的块便于计算存储容量，且允许单个文件的大小超过任何单块物理磁盘的容量。 Фиксированный размер блоков позволяет легко подсчитать объем данных, а размер файла может превышать емкость любого отдельного диска в сети。

---

3. 架构组件 (Архитектурные компоненты)

HDFS 采用主从架构（Master/Slave）： HDFS использует архитектуру Master/Slave:

- **NameNode (管理节点):** 集群中的主节点，负责管理文件系统的元数据及数据块在各服务器上的分布。 **NameNode** — главный узел в кластере, управляющий метаданными и распределением блоков по серверам。
- **DataNode (数据节点):** 实际存储数据块的服务器节点。 **DataNode** — узлы, на которых физически хранятся блоки данных。

---

4. 容错与可靠性 (Отказоустойчивость и надежность)

- 副本机制 (Репликация): 为了确保存储的高可用性，数据块会进行副本复制，默认情况下保存 **3 个副本**。 Для обеспечения **высокой доступности** блоки данных реплицируются. По умолчанию создается **3 реплики**。
- 容错策略 (Стратегия отказоустойчивости): 为了防止硬件故障，副本通常会分发到不同的服务器，甚至其中一个副本会放置在不同的机架 (Rack) 上。这样即使某个服务器或整个机架发生故障，也不会导致数据丢失。 Для защиты от сбоев реплики копируются на **разные серверы**, и как минимум одна реплика размещается на **другой стойке**. Отказ одного или нескольких серверов не ведет к потере данных。

---

5. 扩展性与维护 (Масштабируемость и обслуживание)

- 水平扩展 (Горизонтальное масштабирование): 增加或删除服务器时不需要重新配置整个文件系统。 Добавление или удаление серверов **не требует перенастройки** файловой системы。
- 负载均衡 (Балансировка): 系统内置的负载均衡器 (Balancer) 会自动将现有数据块从负载较高的服务器迁移到新加入的服务器上。 **Балансировщик** нагрузки автоматически копирует часть существующих блоков с загруженных серверов на новый сервер。

---

6. 交互与命令 (Взаимодействие и команды)

访问 HDFS 通常使用 **URI 格式**：`hdfs://authority/path`。用户可以通过命令行调用与 **POSIX** 兼容的命令来管理文件： Доступ к HDFS осуществляется через **URI формат**: `hdfs://authority/path`. Пользователи могут использовать **POSIX-подобные** команды для управления файлами:

- **列出文件 (Список файлов):** `hadoop fs -ls`
- **上传文件 (Загрузка файла):** `hadoop fs -copyFromLocal` 或 `hadoop fs -put`
- **查看内容 (Просмотр содержимого):** `hadoop fs -cat`
- **设置副本数 (Установка репликации):** `hadoop fs -setrep`

5. В чем отличие систем хранения больших данных (масштабируемость, отказоустойчивость)

大数据存储系统（如HDFS和HBase）与传统关系型数据库（РСУБД）在**可扩展性**和**容错性**方面有显著区别：

1. 可扩展性 (Масштабируемость)

- **架构基础 (Основа архитектуры):** 大数据系统基于**无共享架构 (Shared Nothing Architecture, SNA)**。在这种架构中，每个节点拥有独立的内存、磁盘阵列和输入/输出设备，各节点互不干扰，这为系统提供了极佳的扩展基础。 Системы больших данных основаны на **архитектуре без общих ресурсов (Shared Nothing Architecture, SNA)**. В такой архитектуре каждый узел имеет собственную память, дисковые массивы и устройства ввода/вывода, что обеспечивает отличную базу для расширения.
- **扩展方式 (Способ масштабирования):** 大数据系统采用**水平扩展 (Горизонтальная масштабируемость)**。增加新服务器时**无需重新配置**整个系统。相比之下，传统关系型数据库通常处理 TB（太字节）级数据，而 HBase 等系统可扩展至 **PB（拍字节）** 级别。 Системы больших данных используют **горизонтальную масштабируемость**. Добавление нового сервера **не требует перенастройки** всей системы. В то время как традиционные РСУБД работают с терабайтами (Тб), системы вроде HBase масштабируются до **петабайтов (Пб)**.
- **负载均衡 (Балансировка нагрузки):** 系统内置的**均衡器 (Balancer)** 会自动将现有数据块从负载较高的节点迁移到新加入的节点。 Встроенный **балансировщик** нагрузки автоматически копирует часть существующих блоков с загруженных серверов на новый сервер.

2. 容错性 (Отказоустойчивость)

- **副本机制 (Механизм репликации):** 大数据文件系统（如 HDFS）通过**数据块副本复制**确保存储的可靠性，默认通常保存 **3 个副本**。 Отказоустойчивость в системах типа HDFS обеспечивается за счет

репликации блоков данных (по умолчанию создается **3 реплики**).

- **机架感知策略 (Стратегия размещения реплик):** 为了防止机架级故障，系统会将其中一个副本放置在不同机架的服务器上。这样即使某个服务器或整个机架发生故障，也不会导致数据丢失。 Для защиты от сбоев одна реплика блока обязательно копируется на сервер из **другой стойки**. Таким образом, отказ одного или нескольких серверов (или всей стойки) не ведет к потере данных.
- **自动监控与恢复 (Автоматическое восстановление):** 系统的主节点（如 HBase 的 HMaster）会持续监控区域服务器的状态。一旦检测到节点失效，系统会自动重新恢复数据副本的数量，确保系统的高可用性。 Главный узел системы (например, HMaster в HBase) постоянно отслеживает состояние серверов регионов. В случае сбоя система **автоматически восстанавливает количество копий блоков**, обеспечивая высокую доступность данных.

6. Зачем нужны NoSQL базы данных

根据资料，**NoSQL 数据库（NoSQL базы данных）** 的出现主要是为了解决传统关系型数据库（РСУБД）在大数据场景下的局限性。其核心需求包括：

1. 海量数据与高吞吐量 (Огромные объемы и высокая пропускная способность)

- **规模差异：** 传统数据库（РСУБД）通常处理 **TB（太字节）** 级数据，而 NoSQL 系统（如 HBase）设计用于处理 **PB（拍字节）** 级数据。
- **吞吐能力：** 传统数据库每秒处理数千次请求，而 NoSQL 数据库每秒可处理数百万次请求。 **Масштабируемость:** Традиционные РСУБД обычно работают с Тб данных, в то время как NoSQL системы (например, HBase) рассчитаны на Пб (петабайты). NoSQL обеспечивает пропускную способность в миллионы запросов в секунду против тысяч в РСУБД.

2. 灵活的数据模型 (Гибкая модель данных)

- **无固定模式 (Schemaless)：** 像 MongoDB 这样的数据库在更改数据结构时无需重新创建数据库模式。这允许存储半结构化和非结构化数据。
- **多样化存储：** NoSQL 提供了面向文档（MongoDB）、稀疏列存储（HBase）和图数据库（Neo4j）等多种模型，以适应不同的业务逻辑。 **Гибкость:** В таких БД, как MongoDB, не нужно пересоздавать схему при изменении структуры данных. NoSQL предлагает различные модели: документоориентированные, разреженные отображения и графовые базы.

3. 高水平扩展性与容错性 (Горизонтальное масштабирование и отказоустойчивость)

- **无共享架构 (Shared Nothing Architecture, SNA)：** NoSQL 节点通常是自给自足的，不与其他节点共享资源，这使得系统易于水平扩展。
- **自动分片与副本：** 系统支持自动分片 (Sharding) 以支持扩展性，并通过数据副本 (Replication) 确保高可用性和容错性。 **Архитектура SNA:** Узлы в NoSQL независимы и самодостаточны, что

позволяет легко масштабировать систему горизонтально. Поддерживается автоматическое шардирование и репликация для высокой доступности.

4. 性能权衡 (Компромисс в производительности)

- 放弃严格一致性： 为了换取极致的性能和扩展能力，NoSQL 系统通常选择**放弃严格的 ACID（原子性、一致性、隔离性、持久性）支持**（例如 HBase 仅支持单行事务）。
- 降低开销： 传统数据库的复杂索引和优化在重负载下会导致速度剧烈下降，而 NoSQL 通过更简单的查询语言（如 get/put/scan）降低了这些开销。 **Отказ от ACID:** Для достижения высокой производительности NoSQL системы часто отказываются от строгой консистентности (ACID поддерживается, например, только на уровне одной строки). Это минимизирует накладные расходы, которые в РСУБД приводят к падению скорости под нагрузкой.

7. Виды NoSQL баз данных

根据提供的资料，**NoSQL 数据库（NoSQL базы данных）** 主要分为以下几种类型：

1. 宽列存储 / 稀疏列家族存储 (Разреженные столбцовые хранилища / Column-family stores)

- 典型代表 **(Пример): HBase**,。
- **特性 (Характеристики):** 这是一种**稀疏的、分布式的、多维有序映射**,。它专门为高吞吐量设计，每秒可处理**数百万次请求**，且最大数据容量可达 **PB 级**。
- **HBase** — это **разреженное, распределенное, многомерное отсортированное отображение**. Она обеспечивает высокую пропускную способность (миллионы запросов в секунду) и масштабируется до петабайтов.

2. 面向文档的数据库 (Документоориентированные базы данных / Document stores)

- 典型代表 **(Пример): MongoDB**,。
- **特性 (Характеристики):** 这是一种高性能、可扩展的开源数据库。它以**文档（JSON 或 BSON 格式）**为存储单元，不需要在修改数据结构时重新创建数据库模式（即 **无模式/Schemaless**）。
- **MongoDB** — масштабируемая высокопроизводительная база данных, ориентированная на **документы** (JSON, BSON). В ней **не нужно пересоздавать схему** при изменении структуры данных.

3. 图数据库 (Графовые базы данных / Graph databases)

- 典型代表 **(Пример): Neo4j**。
- **特性 (Характеристики):** 用于存储实体（**节点**）及其之间的**关系**。这种格式在处理复杂结构的数据时比传统关系型数据库有更好的优化性能。它使用专门的声明式查询语言 **Cypher**，该语言基于模式匹配。

- **Neo4j** используется для хранения информации о сущностях (**узлах**) и **отношениях** между ними. Для запросов используется декларативный язык **Cypher**, основанный на сопоставлении шаблонов.

4. 总结对比 (Итоговое сравнение)

资料中提到，NoSQL 数据库通常采用**无共享架构 (Shared Nothing Architecture, SNA)**，这使得它们在**水平扩展性**和**容错性**方面优于传统关系型数据库。虽然它们可能不支持完整的 **ACID** 事务（例如 HBase 仅支持单行 ACID），但其在大数据处理中的性能优势非常明显。

8. Модель данных HBase

根据资料，**HBase 的数据模型**及其核心特性可以通过以下俄中对照进行解析：

1. 核心定义 (Основное определение)

HBase 的数据模型被定义为一种**稀疏的、分布式的、多维有序映射**。 Модель данных HBase определяется как **разреженное, распределенное, многомерное отсортированное отображение**。

---

2. 模型要素 (Элементы модели)

与传统关系型数据库（РСУБД）的平铺表结构不同，HBase 采用嵌套和版本化的结构： В отличие от плоских таблиц в РСУБД, HBase использует вложенную и версионную структуру:

- **行键 (Row Key / Ключ строки):**
  - 这是数据的主键。HBase 物理上**按照行键的字典顺序**对数据进行排序。
  - Это основной идентификатор. HBase физически **сортирует данные по ключу строки** в лексикографическом порядке,。
- **列族 (Column Families / Семейства столбцов):**
  - 在创建表时必须预先定义。它将相关的列组合在一起（例如资料示例中的 ‘ФИО’ 和 ‘Работа’）。
  - Определяются при создании таблицы. Они объединяют связанные столбцы (например, ‘ФИО’ и ‘Работа’ в примерах из источников)。
- **列修饰符 (Column Qualifiers / Столбцы):**
  - 列族下的具体字段（如 ‘Имя’, ‘Должность’）。列是动态的，不需要预先创建模式。
  - Конкретные поля внутри семейств (например, ‘Имя’, ‘Должность’). Столбцы динамичны и не требуют строгого описания схемы。
- **时间戳 (Timestamp / Метки времени):**

- HBase 支持**多版本并发控制**。每个单元格（Cell）可以根据时间戳存储同一数据的多个版本（例如某个员工在不同年份的职位变化）。
- HBase поддерживает **многоверсионность**. Каждая ячейка может хранить несколько версий данных с привязкой к метке времени (например, изменение должности сотрудника по годам)。

---

3. HBase 与关系型数据库 (РСУБД) 的对比

(Сравнение HBase и РСУБД)

特性 (Характеристика)	HBase (大数据系统)	传统关系型数据库 (РСУБД)
数据模型 (Модель данных)	多维有序映射 (Многомерное отображение)	关系模型 (Реляционная)
事务 (Транзакции)	仅支持单行 ACID (ACID только для одной строки)	完整 ACID 支持 (Полная поддержка ACID)
查询语言 (Язык запросов)	get / put / scan 操作	SQL 语言
索引 (Индексы)	仅限行键或使用辅助表 (Только по Row Key)	支持多字段索引 (Множество индексов)

---

4. 物理存储逻辑 (Логика физического хранения)

- **存储单位:** 数据物理上存储在 **HDFS**（Hadoop 分布式文件系统）中，格式为 **HFile**。 **Хранение:** Данные физически хранятся на **HDFS** в специальном формате **HFile**。
- **写入流程:** 数据首先写入预写日志（**WAL**），然后进入内存缓冲区（**MemStore**），最后才刷新到磁盘。 **Процесс записи:** Данные сначала записываются в лог (**WAL**), затем в буфер оперативной памяти (**MemStore**) и только потом переносятся на диск。

9. Модель распределенных вычислений MapReduce

MapReduce 是一种由 **Hadoop** 采用的**分布式计算范式**。该模型旨在处理海量数据，其核心操作基于**键值对 (key/value)** 和**列表 (lists)**。

MapReduce — это **парадигма распределенных вычислений**, используемая в технологии **Hadoop**. Эта модель предназначена для обработки больших объемов данных, а ее логика основана на операциях с **парами ключ/значение** и **списками (lists)**.

---

1. 计算阶段 (Этапы вычислений)

该模型主要分为两个核心阶段，以及一个中间处理过程： Модель делится на два основных этапа и промежуточный процесс:

- **Map 阶段 (Этап Map):** 接收输入对 `<k1, v1>` 并产生中间键值对列表 `list(<k2, v2>)` 。此阶段通常用于数据的过滤、转换或投影（如 SQL 中的 `SELECT` 或 `WHERE` ）,。 **Этап Map:** Принимает входную пару `<k1, v1>` и создает список промежуточных пар `list(<k2, v2>)` . Этот этап обычно используется для **фильтрации, преобразования или проекции** данных (аналоги `SELECT` или `WHERE` в SQL),。
- **Shuffle 阶段 (Этап Shuffle):** 这是一个中间过程，系统将具有相同键 **(key)** 的所有值收集并分组在一起，形成 `<k2, list(v2)>` 的形式提供给 Reducer。 **Этап Shuffle:** Это промежуточный процесс, при котором система собирает и группирует все значения с **одинаковым ключом (key)** в формате `<k2, list(v2)>` для передачи в Reducer。
- **Reduce 阶段 (Этап Reduce):** 接收键及对应的列表 `<k2, list(v2)>` ，并产出最终结果列表 `list(<k3, v3>)` 。资料强调，**Reduce 任务仅在所有 Map 任务全部完成后才会启动。** **Этап Reduce:** Принимает ключ и соответствующий ему список `<k2, list(v2)>` и выдает итоговый список результатов `list(<k3, v3>)` . Важно, что **задача Reduce запускается только после завершения всех задач Map**。

---

2. 优化机制 (Механизм оптимизации)

- **Combiner (合并器):** 为了减少网络传输的数据量，可以使用 **Combiner** 作为一个附加步骤。它在同一个节点上执行局部聚合操作（类似于“小型的 Reducer”），从而提高整体计算效率。
- **Combiner (Комбайнер):** Для сокращения объема передаваемых по сети данных может использоваться **Комбайнер** как дополнительный шаг. Он выполняет операции локальной агрегации данных **в рамках одного узла**, что повышает общую эффективность вычислений。

---

3. 应用示例 (Примеры применения)

- **词频统计 (Word Count):** Map 提取单词并标记为 `(word, 1)` ，Reduce 统计每个单词的总数,。 **Подсчет слов (Word Count):** Map извлекает слова и помечает их как `(word, 1)` , Reduce суммирует общее количество для каждого слова,。
- **矩阵运算 (Матричные вычисления):** MapReduce 也可用于实现复杂的数学运算，如矩阵与向量的乘法。 **Матричные вычисления:** MapReduce также может быть использован для реализации сложных математических операций, таких как **произведение матрицы на вектор**。

10. Технология Spark

根据提供的资料，关于**Spark 技术（Технология Spark）**的信息总结如下：

1. Spark 在生态系统中的定位 (Позиционирование Spark в экосистеме)

- **Spark 技术**是资料中列出的构成 Hadoop 大数据处理生态系统的核心组件之一。 **Технология Spark** является одним из основных компонентов, составляющих экосистему обработки больших данных Hadoop, согласно списку в источниках。
- 它与 **MapReduce**、**YARN**、**Hive** 和 **Pig** 等技术共同协作，旨在分布式环境下处理大规模数据集。 Она работает совместно с такими технологиями, как **MapReduce**, **YARN**, **Hive** и **Pig**, для обработки масштабных наборов данных в распределенной среде。

2. 应用场景 (Сценарии применения)

- 根据系统处理流程图，此类技术通常被用于支持**实时决策**（Real-time decision making）的任务需求。 Согласно схемам процессов обработки, подобные технологии обычно используются для поддержки задач **принятия решений в реальном времени** (Real-time decision making)。
- 在现代计算模型中，它被定位在处理**流式数据**（Streaming data）的关键环节。 В современных вычислительных моделях она занимает ключевое место в звене обработки **потоковых данных** (Streaming data)。

---

3. 局限性与补充说明 (Ограничения и дополнительные пояснения)

- **资料局限性：** 资料中仅在技术清单和架构图中提及了 Spark，并未提供关于其编程模型、核心抽象（如 RDD）或其具体运行机制的详细描述,。 **Ограничения источников:** Spark упоминается в источниках только в списках технологий и на схемах архитектуры; детальное описание его программной модели, основных абстракций (таких как RDD) или конкретных механизмов работы в материалах отсутствует,。
- **非资料来源信息：** 通常认为 **Spark** 的核心优势在于其**内存计算**（In-memory computing）能力，这使得它在执行迭代式计算时比传统的 MapReduce 效率更高。**由于这些信息并非源自提供的资料，建议您独立核实相关细节。** **Информация не из источников:** Обычно считается, что главным преимуществом **Spark** являются его возможности **вычислений в оперативной памяти** (In-memory computing), что делает его более эффективным по сравнению с традиционным MapReduce при выполнении итеративных вычислений. **Так как эта информация не содержится в предоставленных источниках, рекомендую проверить данные детали самостоятельно.**

11. Языки программирования высокого уровня над Hadoop

在 Hadoop 生态系统中，为了简化大规模数据集的处理，开发了多种**高级编程语言**。根据资料（特别是《1_Введение.pdf》），主要包括以下内容：

1. 核心高级语言 (Основные языки высокого уровня)

- **Pig:** 专门设计用于 Hadoop 之上的高级查询语言，允许开发者通过类似于脚本的逻辑来处理并行计算任务，而无需直接编写底层的 MapReduce 程序。 **Pig** — это высокоуровневый язык запросов над Hadoop, который позволяет разработчикам описывать логику обработки данных через скрипты, не прибегая к прямому написанию низкоуровневого кода MapReduce。
- **Hive:** 另一种核心的高级语言工具，它提供了一种类 SQL 的查询方式（HiveQL），使得熟悉关系型数据库的人员能够轻松地在 Hadoop 分布式文件系统上执行查询和数据分析。 **Hive** — еще один ключевой инструмент высокого уровня, который предоставляет SQL-подобный интерфейс (HiveQL), позволяя специалистам по базам данных выполнять запросы и анализ данных непосредственно в распределенной среде Hadoop。

2. 相关处理技术 (Связанные технологии обработки)

除了上述语言，资料中还列出了其他关键的计算与分析框架： Помимо указанных языков, в источниках упоминаются и другие важные вычислительные среды:

- **Spark:** 它是大数据处理生态系统中的重要组成部分，常用于**高级分析 (Advanced analytics)**、事件处理以及支持**实时决策**的流式数据处理。 **Spark** является важной частью экосистемы обработки больших данных, используемой для **продвинутой аналитики**, обработки событий и работы с потоковыми данными в задачах **принятия решений в реальном времени**。
- **MapReduce:** 虽然它是 Hadoop 的底层计算模型（基于键值对 `<key, value>`），但 Pig 和 Hive 等高级语言最终都会被转化为 MapReduce 任务或类似的分布式任务来执行。 **MapReduce** — хотя это базовая вычислительная парадигма Hadoop (основанная на парах `<key, value>`), высокоуровневые языки, такие как Pig и Hive, в конечном итоге транслируются в задачи MapReduce или аналогичные распределенные процессы для выполнения。

3. 应用场景 (Сценарии применения)

这些语言和技术共同协作，支持从**非结构化数据 (Unstructured data)** 中进行提取、转换和聚合，以便进行后续的深度分析或存入对象关系型数据仓库中。 Эти языки и технологии работают совместно, обеспечивая извлечение, трансформацию и агрегирование из **неструктурированных данных**, что необходимо для последующего глубокого анализа или загрузки в объектно-реляционные хранилища。

12. Что такое Hadoop. Основные компоненты Hadoop (HDFS, MapReduce)

Hadoop 是一个用于分布式存储和处理海量数据（尤其是非结构化数据）的技术框架。它基于**无共享架构 (Shared Nothing Architecture, SNA)**，在这种架构中，每个节点都拥有独立的资源且互不干扰，从而实现了卓越的水平扩展能力。

Hadoop — это технологический стек для распределенного хранения и обработки огромных объемов данных (особенно неструктурированных). Он основан на архитектуре **Shared Nothing (SNA)**, где каждый узел автономен, что обеспечивает отличную горизонтальную масштабируемость.

---

1. HDFS (Hadoop Distributed File System)

- **定义 (Определение):** Hadoop 的分布式文件系统，专为在大规模集群上处理超大文件而设计。 Распределенная файловая система Hadoop, специально разработанная для обработки очень больших файлов на массовых кластерах.
- **数据块机制 (Блочная структура):** 文件被切分为固定大小的**数据块**（默认为 **64MB**，推荐为 **128MB**）进行存储。这使得单个文件的大小可以超过集群中任何单个磁盘的容量。 Файлы делятся на **блоки** фиксированного размера (по умолчанию **64 Мб**, рекомендуется **128 Мб**). Это позволяет хранить файлы, размер которых превышает емкость любого отдельного диска в системе.
- **容错与副本 (Отказоустойчивость и репликация):** 系统会自动进行**数据块副本复制**（默认通常为 **3 个副本**）。为了防止机架故障，其中一个副本会被放置在**不同机架**的服务器上。 Система автоматически выполняет **репликацию блоков** (по умолчанию **3 реплики**). Для защиты от сбоев стоек одна из копий обязательно размещается на сервере в **другой стойке**.

2. MapReduce

- **计算范式 (Парадигма вычислений):** 这是一个分布式计算模型，其逻辑核心是处理**键值对 (key/value)**和**列表 (lists)**。 Это модель распределенных вычислений, логика которой основана на работе с парами **ключ/значение** и **списками**.
- **主要阶段 (Основные этапы):**
  - **Map 阶段:** 接收输入对 `<k1, v1>` 并产出中间键值对列表 `list(<k2, v2>)`。 **Этап Map:** принимает входную пару `<k1, v1>` и выдает список промежуточных пар `list(<k2, v2>)`.
  - **Shuffle 阶段:** 系统自动将具有相同键的所有值收集并分发给对应的 Reducer。 **Этап Shuffle:** система автоматически собирает все значения с одинаковым ключом и передает их соответствующему Reducer.
  - **Reduce 阶段:** 接收键和对应的值列表 `<k2, list(v2)>`，并产出最终的聚合结果 `list(<k3, v3>)`。 **Этап Reduce:** принимает ключ и список его значений `<k2, list(v2)>`, формируя итоговый результат `list(<k3, v3>)`。

3. YARN

- **资源管理 (Управление ресурсами):** 资料将其列为 Hadoop 的核心组件之一，负责管理集群中的计算资源并进行任务调度。 В источниках YARN указан как один из основных компонентов Hadoop, отвечающий за управление вычислительными ресурсами кластера и планирование задач.

13. Узлы Hadoop кластера (DataNode, NameNode, TaskTracker, JobTracker)

在 Hadoop 集群架构中，节点根据其承担的角色（存储或计算）进行划分。根据资料和分布式系统通用原理，各节点的功能如下：

1. NameNode (管理节点)

- **NameNode** 是 HDFS（Hadoop 分布式文件系统）的主节点。
- 它负责管理文件系统的**命名空间**，并存储关于文件块如何分布在集群服务器上的**位置信息**。
- **NameNode** — это главный узел HDFS, который управляет пространством имен и хранит информацию о расположении блоков данных на серверах кластера。

2. DataNode (数据节点)

- 这些节点是**物理存储数据**的地方，数据以块（Block）的形式存在（默认通常为 128MB），。
- 为了确保容错性，数据块会在多个 **DataNode** 之间进行副本复制。
- **DataNode** — это узлы, на которых **данные физически хранятся** в виде блоков. Для обеспечения отказоустойчивости блоки реплицируются между разными DataNode,。

3. JobTracker 与 TaskTracker (计算节点 - 传统模型)

注：提供的资料主要提及了 MapReduce 和 YARN 资源管理。以下关于 JobTracker 和 TaskTracker 的具体功能描述属于大数据领域的通用补充信息，建议独立核实：

- **JobTracker:** 负责接收用户提交的计算作业，并将任务分发给集群中的 TaskTracker。它还负责监控任务进度。 **JobTracker** отвечает за прием заданий от пользователей, распределение задач между TaskTracker-ами и мониторинг их выполнения。
- **TaskTracker:** 运行在 DataNode 上的代理程序，负责执行由 JobTracker 分配的具体 **Map** 或 **Reduce** 任务。 **TaskTracker** — это агент на узлах хранения, который выполняет конкретные операции **Map** или **Reduce**, назначенные JobTracker-ом。

4. 架构演进与 Master 角色 (Эволюция архитектуры)

- 在现代 Hadoop 生态系统中，**YARN** 承担了资源管理职能。
- 类似于 HBase 的 **Master** 节点，这类主节点持续通过**心跳信号 (Heartbeat)** 监控从节点（如 Region Server）的状态，并在检测到故障时自动启动恢复程序。
- В современных системах за управление ресурсами отвечает **YARN**. Главный узел (например, Master в HBase) постоянно проверяет доступность серверов через **сигналы heartbeat** и запускает восстановление при сбоях。

14. Отказоустойчивость, что если упадет DataNode, NameNode, TaskTracker, JobTracker

在 Hadoop 架构中，系统的容错性主要通过副本机制和主从监控来实现。以下是针对不同节点失效时的处理机制：

В архитектуре Hadoop отказоустойчивость системы обеспечивается в основном за счет **механизма репликации** и **мониторинга «главный-подчиненный»**. Ниже описаны механизмы обработки сбоев для различных узлов:

1. DataNode 失效 (Отказ DataNode)

- **数据安全 (Сохранность данных):** 当 DataNode 掉线时，数据不会丢失，因为 HDFS 默认会将每个数据块复制 **3 个副本**。 При отказе **DataNode** данные не теряются, так как HDFS по умолчанию реплицирует каждый блок данных в **3 экземплярах**。
- **自动恢复 (Автоматическое восстановление):** 系统的主节点会通过**心跳信号 (Heartbeat)** 发现失效节点，并自动在其他健康节点上**重新恢复数据块的副本数量**。 Главный узел системы обнаруживает отказавший узел через **сигналы сердцебиения (heartbeat)** и автоматически **восстанавливает количество копий блоков** на других исправных узлах。
- **机架感知 (Стойкоустойчивость):** 为了防止整个机架掉电，系统会确保至少有一个副本存储在**不同机架**的服务器上。 Для защиты от отказа всей стойки система гарантирует, что как минимум одна реплика хранится на сервере в **другой стойке**。

2. NameNode 或 JobTracker (Master) 失效 (Отказ NameNode или JobTracker)

- **备用机制 (Резервирование):** 资料提到，为了防止主节点（Master）单点故障，系统设有**备用主服务器 (Backup Master servers)**。 В источниках указано, что для предотвращения единой точки отказа главного узла (Master) используются **резервные Master-серверы**。
- **状态接管 (Перехват управления):** 备用服务器持续同步活动服务器的状态信息。一旦 **ZooKeeper** 检测到活动主节点失效并发出信号，其中一个备用服务器会立即接管并变为活动状态。 Резервные серверы хранят информацию о состоянии активного узла. Как только **ZooKeeper** обнаруживает сбой и посылает сигнал, один из резервных серверов становится активным。

3. TaskTracker 失效 (Отказ TaskTracker)

- **任务重试 (Повторное выполнение):** 虽然资料中未详细展开 TaskTracker 的具体术语，但提到分布式计算模型（如 MapReduce）会监控计算节点的可用性。 Хотя в материалах детально не описывается термин TaskTracker, указано, что модели распределенных вычислений (такие как MapReduce) отслеживают доступность вычислительных узлов。
- **心跳监控 (Мониторинг через Heartbeat):** 如果计算节点停止发送心跳信号，Master 节点会认为该任务失败，并将其重新分配给集群中的其他节点执行。 Если вычислительный узел перестает

посылает **heartbeat-сигналы**, Master-узел считает задачу проваленной и переназначает ее другому узлу кластера。

总结 (Итог)

大数据存储系统通过**无共享架构 (Shared Nothing Architecture)** 确保了节点间的独立性，使得任何单点故障都能通过冗余副本和自动化的主节点切换来解决。 Системы хранения больших данных на базе **архитектуры без общих ресурсов (Shared Nothing Architecture)** обеспечивают независимость узлов, что позволяет устранять любые сбои через избыточное копирование и автоматическое переключение главных узлов。

15. Основные проблемы Hadoop 1. Основные отличия Hadoop 2 от Hadoop 1

根据提供的资料及大数据系统的通用演进逻辑，**Hadoop 1 的主要问题**以及 **Hadoop 2 与其核心区别**总结如下：

1. Hadoop 1 的主要问题 (Основные проблемы Hadoop 1)

- **单点故障 (Единая точка отказа):** 在早期架构中，**NameNode** 是唯一的管理节点，如果它发生故障，整个集群将不可用。 **Единая точка отказа (SPOF):** В ранней архитектуре **NameNode** был единственным управляющим узлом. Его выход из строя приводил к остановке всего кластера。
- **可扩展性瓶颈 (Проблемы масштабируемости): JobTracker** 同时负责资源管理和任务监控，当集群达到数千个节点时，负载过重成为性能瓶颈。 **Узкое место в масштабируемости: JobTracker** одновременно отвечал за управление ресурсами и мониторинг задач. При росте кластера до тысяч узлов он становился критическим узлом, замедляющим работу。
- **功能局限性 (Ограниченность функционала):** Hadoop 1 几乎只支持 **MapReduce** 计算模型，难以运行流处理或实时分析任务。 **Ограниченность моделей вычислений:** Hadoop 1 был практически «заточен» только под модель **MapReduce**, что затрудняло запуск потоковой обработки или real-time аналитики。

---

2. Hadoop 2 的核心区别与改进 (Основные отличия Hadoop 2 от Hadoop 1)

- **引入 YARN 资源管理 (Внедрение YARN):**
  - Hadoop 2 将资源管理从计算逻辑中分离出来，引入了 **YARN**（Yet Another Resource Negotiator）。
  - 这使得 Hadoop 集群不仅能运行 MapReduce，还能同时运行 **Spark**、**Hive** 和 **Pig** 等多种计算引擎。

- **Внедрение YARN:** Hadoop 2 отделил управление ресурсами от логики вычислений. Это позволило запускать на одном кластере не только MapReduce, но и другие движки, такие как **Spark, Hive и Pig**。

- **高可用性 (Высокая доступность - HA):**
  - 资料提到，为了防止主节点故障，系统引入了**备用主服务器 (Backup Master servers)**。
  - 通过 **ZooKeeper** 进行监控，一旦主节点失效，备用节点可以自动接管，消除了单点故障。
  - **High Availability (HA):** Для предотвращения сбоев были внедрены **резервные Master-серверы**. Система использует **ZooKeeper** для мониторинга; если основной узел падает, резервный перехватывает управление, устраняя SPOF。
- **更灵活的架构 (Гибкая архитектура):**
  - 基于**无共享架构 (Shared Nothing Architecture)**，节点间的独立性增强，使得横向扩展（Horizontal scaling）更加高效。
  - **Архитектура SNA:** Использование архитектуры **Shared Nothing** обеспечивает автономность узлов, что делает горизонтальное масштабирование более эффективным。

总结对比 **(Краткое сравнение)**

特性 (Характеристика)	Hadoop 1	Hadoop 2
资源管理 (Ресурсы)	JobTracker (单负载/Один узел)	**YARN** (分布式管理/ Распределенное управление)
高可用性 (Доступность)	无 (单点故障/SPOF)	**HA** (备用主节点/Backup Masters + ZooKeeper)
支持模型 (Движки)	仅 MapReduce (Только MR)	**多引擎** (Spark, MapReduce, etc.)

16. Информационная безопасность больших данных. Конфиденциальность, Целостность, Доступность

在大数据环境下，**信息安全（Информационная безопасность）** 是确保系统可靠性的核心。根据提供的资料，大数据的安全保障可以从信息安全的三要素——**机密性、完整性、可用性**进行分析：

1. 机密性 (Конфиденциальность)

机密性确保敏感数据不被未经授权的用户访问。 **Конфиденциальность** гарантирует, что доступ к чувствительным данным имеют только авторизованные пользователи.

- **数据敏感性 (Чувствительность данных):** 大数据涵盖了社会网络数据、国防数据（**Данные министерства обороны**）、医疗观察（**Медицинские наблюдения**）和电子商务数据等。这些领域对隐私保护有极高要求。
- **访问控制机制 (Механизмы контроля доступа):** 以 HBase 为例，系统通过身份验证（**Аутентификация**）和授权（**Авторизация**）来确保只有合法用户能操作特定数据。

2. 完整性 (Целостность)

完整性确保数据在存储或处理过程中不被非法篡改或损坏。 **Целостность** обеспечивает защиту данных от несанкционированного изменения или повреждения в процессе хранения и обработки.

- **数据真实性 (Veracity / Достоверность):** 作为大数据“5V”特性之一，真实性要求必须准备好面对并非所有数据都是百分之百可靠的情况，系统需确保数据“洁净度”以满足正确分析的需求。
- **事务支持 (Поддержка транзакций):** HBase 支持单行级别的 **ACID** 特性，保证了在对单行数据进行并发读写操作时的原子性和一致性。
- **反欺诈应用 (Противодействие мошенничеству):** 利用大数据进行反欺诈检测（**Выявление мошенничества**），通过分类算法（有监督学习）和异常检测（无监督学习）来识别并拦截虚假交易，从而维护金融数据的完整性。

3. 可用性 (Доступность)

可用性确保授权用户在需要时能够及时、可靠地访问数据和服务。 **Доступность** гарантирует, что авторизованные пользователи имеют своевременный и надежный доступ к данным и сервисам по мере необходимости.

- **容错与副本机制 (Отказоустойчивость и репликация):** 资料提到，大数据系统（如 HBase）通过数据副本（**Репликация данных**）来实现容错性。如果某个存储节点发生故障，系统可以从其他节点读取备份。
- **自动恢复与监控 (Автоматическое восстановление и мониторинг):**
  - **Master** 节点通过 **ZooKeeper** 和心跳信号（**Heartbeat**）持续监控区域服务器（Region Servers）的状态。
  - 一旦检测到故障，主服务器会立即启动自动恢复流程并重新分配数据任务。
- **高吞吐性能 (Высокая пропускная способность):** HBase 设计用于处理每秒数百万次请求，确保在海量数据环境下依然能提供高水平的数据服务响应。

17. Информационная безопасность больших данных. Аутентификация, авторизация и аудит

在大数据生态系统中，信息安全是确保海量敏感数据（如国防、医疗和金融数据）可靠性的核心要素。根据提供的资料，大数据安全主要通过以下机制实现：

В экосистеме больших данных информационная безопасность является ключевым элементом обеспечения надежности огромных массивов чувствительных данных (таких как оборонные, медицинские и финансовые). Согласно предоставленным источникам, безопасность больших данных реализуется через следующие механизмы:

1. 身份验证与授权 (Аутентификация и авторизация)

- 核心安全机制： 资料明确指出，**HBase** 等大数据系统通过身份验证（**Authentication**）和授权（**Authorization**）来保障安全性，这与传统的关系型数据库（РСУБД）类似。 **Основные механизмы безопасности:** В источниках прямо указывается, что системы больших данных, такие как **HBase**, обеспечивают безопасность через **аутентификацию** и **авторизацию**, аналогично традиционным РСУБД。
- 访问控制： 只有经过身份验证的用户才能获得对特定数据资源（如 HBase 中的列族或表）的访问权限。 **Контроль доступа:** Только аутентифицированные пользователи могут получить права доступа к конкретным информационным ресурсам (например, к семействам столбцов или таблицам в HBase)。

2. 审计与操作监控 (Аудит и мониторинг операций)

- 日志记录： 在 HBase 架构中，所有数据写入操作首先记录在预写日志（**WAL / Write-Ahead Log**）中。虽然 WAL 主要用于故障恢复，但它也记录了系统内发生的所有数据变更，是实施安全审计的基础。 **Протоколирование:** В архитектуре HBase все операции записи данных сначала фиксируются в **логе упреждающей записи (WAL / Write-Ahead Log)**. Хотя WAL в основном используется для восстановления после сбоев, он также фиксирует все изменения данных в системе, являясь основой для проведения **аудита безопасности**。
- 状态监控： 系统的主节点（Master）通过 **ZooKeeper** 和心跳信号（**Heartbeat**）持续监控集群内各服务器的状态，确保操作的合规性与系统的可用性。 **Мониторинг состояния:** Главный узел системы (Master) через **ZooKeeper** и **сигналы heartbeat** постоянно отслеживает состояние всех серверов в кластере, обеспечивая соблюдение регламентов и доступность системы。

3. 大数据安全的必要性 (Необходимость безопасности Big Data)

- 数据敏感性： 大数据涵盖了国防数据、医疗观察记录、社会网络以及电子商务数据，这些信息的泄露将带来严重后果。 **Чувствительность данных:** Большие данные включают в себя **данные министерства обороны**, **записи медицинских наблюдений**, данные социальных сетей и электронной коммерции; утечка этой информации может иметь серьезные последствия。
- 真实性保障： 作为“5V”特性之一，真实性（**Veracity**）要求系统具备验证数据“洁净度”的能力，以防范虚假信息带来的决策风险。 **Обеспечение достоверности:** Как одна из характеристик «5V», **достоверность (Veracity)** требует, чтобы система обладала способностью проверять «чистоту» данных для предотвращения рисков принятия решений на основе ложной информации。
- 反欺诈应用： 大数据技术被广泛用于金融领域的反欺诈（**Anti-fraud**），通过分类算法和异常检测来保护交易的完整性。 **Применение в антифроде:** Технологии больших данных широко

используются для **противодействия мошенничеству (Anti-fraud)** в финансовой сфере, защищая целостность транзакций с помощью алгоритмов классификации и поиска аномалий。

18. Безопасность в Hadoop

在 **Hadoop** 生态系统中，信息安全是确保处理大规模敏感数据（如国防、医疗和金融数据）可靠性的核心。根据资料，Hadoop 及其相关组件（如 HBase）的安全性主要体现在以下几个维度：

В экосистеме **Hadoop** информационная безопасность является ключевым фактором обеспечения надежности при обработке масштабных чувствительных данных (таких как оборонные, медицинские и финансовые). Согласно источникам, безопасность Hadoop и его компонентов (например, HBase) реализуется в нескольких измерениях:

---

1. 身份验证与授权 (Аутентификация и авторизация)

- 核心机制： 资料明确指出，大数据系统（如 HBase）通过**身份验证（Authentication）**和**授权（Authorization）**来保障安全，其模式与传统的关系型数据库类似。
- **Основные механизмы:** В источниках прямо указано, что системы больших данных (такие как HBase) обеспечивают безопасность через **аутентификацию** и **авторизацию**, аналогично традиционным реляционным СУБД.

2. 高可用性与容错性 (Высокая доступность и отказоустойчивость)

- 数据可用性： **HDFS** 通过副本复制机制（默认为 3 个副本）确保数据不会因单个或多个服务器故障而丢失,。
- **Доступность данных: HDFS** гарантирует сохранность данных при отказе одного или нескольких серверов за счет **механизма репликации** (по умолчанию 3 реплики),.
- 集群协调： 系统利用 **ZooKeeper** 进行服务同步和配置管理，防止单点故障。**Master** 节点持续监控所有服务器的状态，并在检测到故障时自动启动恢复流程。
- **Координация кластера:** Система использует **ZooKeeper** для синхронизации сервисов и управления конфигурациями, предотвращая единые точки отказа. **Master-узел** постоянно мониторит состояние всех серверов и автоматически запускает процессы восстановления при сбоях.

3. 数据完整性与恢复 (Целостность и восстановление данных)

- 预防机制： 资料提到了预写日志 **(WAL)** 技术，所有数据变更都会先记录在日志中，以确保系统在发生故障后能够恢复数据完整性。
- **Механизмы защиты:** В источниках упоминается технология **лога упреждающей записи (WAL)**; все изменения данных сначала фиксируются в логе, что позволяет восстановить целостность данных после сбоя.

- 备用方案： 为了进一步增强系统的稳定性，Hadoop 环境通常配置有**备用主服务器 (Backup Master servers)**，它们同步活动服务器的状态信息以便在必要时接管。
- **Резервное копирование:** Для дополнительного повышения стабильности в среде Hadoop часто настраиваются **резервные Master-серверы**, которые хранят информацию о состоянии активного узла для перехвата управления в случае необходимости.

4. 应用安全：反欺诈 (Прикладная безопасность: Антифрод)

- 大数据技术被广泛应用于**反欺诈检测**，通过对海量交易数据进行实时分析和分类算法处理，识别并阻止虚假或恶意行为,。
- Технологии Big Data широко применяются для **выявления мошенничества**; путем анализа огромных массивов транзакций в реальном времени и использования алгоритмов классификации система идентифицирует и блокирует подозрительные действия,。