Тема 2. Архитектура и технологии хранения больших данных

Лекция 2.1. Принципы распределённого хранения данных

Хранение петабайт данных на одном сервере технически невозможно: ни один диск не вмещает такой объём, ни одна машина не обеспечивает достаточной пропускной способности. Решение — распределённые файловые системы, в которых данные разбиваются на блоки и хранятся на множестве узлов кластера. Подход обеспечивает линейное масштабирование: чтобы удвоить ёмкость или скорость, достаточно добавить новые узлы.

— Репликация: каждый блок данных хранится на нескольких узлах (стандарт — 3 копии). При выходе узла из строя данные доступны с другого. Система автоматически создаёт новую реплику для восстановления заданного уровня избыточности.

— Шардирование (Sharding): данные делятся на части (шарды) и распределяются по узлам. Каждый узел обрабатывает только свою часть, что обеспечивает параллелизм.

— Локальность данных (Data Locality): вычисления перемещаются к данным, а не наоборот. Если код и данные находятся на одном узле, сетевой трафик минимален. Принцип «перемещай код, а не данные» критически важен при объёмах в петабайты.

— Теорема CAP: распределённая система одновременно может гарантировать лишь два из трёх свойств — Consistency (согласованность), Availability (доступность), Partition tolerance (устойчивость к сетевому разделению). Выбор компромисса определяет архитектуру системы.

Лекция 2.2. HDFS — Hadoop Distributed File System

HDFS создана в 2006 году командой Apache Hadoop как открытая реализация Google File System (статья Google, 2003). Сегодня HDFS — де-факто стандарт хранения данных в корпоративных Big Data-системах. Система оптимизирована для последовательного чтения больших файлов, а не для случайного доступа к небольшим записям.

Архитектура: NameNode и DataNode

HDFS использует архитектуру ведущий-ведомый (master-slave). NameNode — главный узел, хранящий исключительно метаданные: дерево файловой системы, список блоков каждого файла и их расположение на DataNode. NameNode не хранит сами данные. DataNode — рабочие узлы, хранящие блоки данных. Каждые несколько секунд DataNode отправляют NameNode сигнал heartbeat, подтверждая работоспособность. При пропуске сигналов NameNode считает узел недоступным и инициирует создание новых реплик его блоков.

Поскольку NameNode является единой точкой отказа, в продуктивных системах используется режим High Availability: два NameNode (Active и Standby) синхронизируются через JournalNode. При отказе Active автоматически происходит переключение на Standby.

Запись и чтение файлов

При записи клиент разбивает файл на блоки (по умолчанию 128 МБ), запрашивает у NameNode список DataNode для размещения каждого блока, затем передаёт данные по цепочке репликации: DataNode1 получает блок и передаёт его DataNode2, тот — DataNode3. Это конвейерная репликация, которая эффективнее широковещательной передачи. При чтении клиент получает от NameNode карту расположения блоков и читает каждый с ближайшего (предпочтительно — того же узла, где выполняется код) DataNode.

— Размер блока 128 МБ выбран не случайно: он снижает нагрузку на NameNode (меньше записей в метаданных) и уменьшает накладные расходы на сетевые соединения при чтении больших файлов.

— Rack Awareness: HDFS учитывает физическое расположение серверов. Две реплики хранятся на узлах одной стойки (низкая задержка записи), одна — в другой стойке (защита от отказа стойки целиком).

— HDFS Federation: несколько NameNode управляют разными пространствами имён, разделяя нагрузку и устраняя ограничение на количество файлов.

Лекция 2.3. Data Warehouse, Data Lake и Data Lakehouse

Data Warehouse

Data Warehouse (DWH) — централизованное хранилище структурированных, интегрированных и исторических данных, предназначенное для аналитических запросов. Данные поступают через ETL-конвейер: извлекаются из операционных систем (ERP, CRM, POS), трансформируются в единую модель и загружаются в хранилище. Схема данных определяется заранее (schema-on-write). Для аналитики оптимальны звёздообразные (star) и снежинкообразные (snowflake) схемы. Представители: Oracle Exadata, Teradata, Amazon Redshift, Google BigQuery, Snowflake.

Data Lake

Data Lake — хранилище сырых данных в исходном формате без предварительной трансформации. Схема применяется при чтении (schema-on-read): разные аналитики могут интерпретировать одни и те же данные по-разному в зависимости от задачи. Реализуется на базе HDFS или объектных хранилищ (Amazon S3, Azure Data Lake Storage Gen2, Google Cloud Storage). Стоимость хранения в несколько раз ниже, чем в

DWH. Основной риск — «болото данных» (Data Swamp): без каталога метаданных и политик управления озеро превращается в неструктурированную свалку, из которой невозможно извлечь нужные данные.

Data Lakehouse

Data Lakehouse — архитектурный паттерн, объединяющий дешевизну и гибкость Data Lake с надёжностью и аналитическими возможностями DWH. Форматы Delta Lake (Databricks), Apache Iceberg и Apache Hudi добавляют поверх Parquet-файлов на S3/HDFS транзакции ACID, версионирование данных, откат к предыдущим версиям и обновление/удаление отдельных строк. Это устраняет главные ограничения Data Lake без необходимости дублировать данные в отдельное хранилище.

Лекция 2.4. NoSQL базы данных: типы и принципы

NoSQL-системы появились в конце 2000-х годов как ответ на три ограничения реляционных СУБД при работе с большими данными: сложность горизонтального масштабирования, жёсткая схема, плохая поддержка неструктурированных данных. Термин «Not Only SQL» подчёркивает, что NoSQL дополняет, а не заменяет реляционные базы данных.

Документоориентированные базы данных

Единица хранения — документ (JSON или BSON), содержащий все данные об объекте включая вложенные структуры. Нет фиксированной схемы: документы в одной коллекции могут иметь разные поля. Это удобно при частом изменении структуры данных или когда она заранее неизвестна. MongoDB, CouchDB, Amazon DocumentDB. Применение: каталоги товаров, профили пользователей, системы управления контентом.

Хранилища ключ-значение

Простейшая модель: значение (строка, список, хэш, число) хранится по уникальному ключу. Операции get и set выполняются за константное время. Redis хранит данные в памяти, обеспечивая субмиллисекундные задержки. DynamoDB — управляемое хранилище от AWS с гарантиями доступности. Применение: кэширование сессий, счётчики просмотров, корзины покупок, очереди задач.

Столбцовые базы данных (Wide-Column)

Данные организованы в семейства столбцов (column families). В отличие от реляционных таблиц, строки могут иметь разный набор столбцов. Запросы по конкретным столбцам выполняются без чтения остальных данных строки. Apache Cassandra, HBase, ScyllaDB. Применение: временные ряды, данные IoT, аналитика

кликов.

Графовые базы данных

Данные представлены в виде графа: узлы (сущности) и рёбра (связи между ними). Обход графа (traversal) — поиск связанных узлов — выполняется на порядки быстрее, чем эквивалентные многоуровневые JOIN-запросы в SQL. Neo4j, Amazon Neptune, JanusGraph. Применение: социальные графы, системы рекомендаций, выявление мошенничества через анализ связей.

Лекция 2.5. MongoDB: архитектура и возможности

MongoDB — наиболее распространённая документоориентированная СУБД, используется в стеках MEAN и MERN. Документы хранятся в формате BSON (Binary JSON), поддерживающем типы данных JSON плюс Date, ObjectId, Binary, Decimal128 и другие. Иерархия: Database — Collection — Document (аналог Database — Table — Row в реляционных системах).

— Гибкая схема позволяет добавлять поля без изменения структуры всей коллекции. Начиная с версии 3.6 поддерживается Schema Validation для наложения ограничений.

— Агрегационный конвейер (Aggregation Pipeline) — мощный механизм обработки: $match (фильтрация), $group (группировка), $sort (сортировка), $lookup (аналог JOIN), $unwind (раскрытие массивов), $project (проекция полей).

— Шардирование: коллекция распределяется по нескольким шардам по значению ключа шардирования. Mongos — маршрутизатор, направляющий запросы к нужному шарду.

— Репликация: Replica Set из трёх узлов (один Primary, два Secondary). Запись принимает Primary; при его отказе Secondary проводят выборы и избирают нового Primary за несколько секунд.

— Индексы: B-tree (основной), составные, по тексту (text), геопространственные (2dsphere), TTL (автоматическое удаление документов через заданное время).

Лекция 2.6. Apache Cassandra и HBase: столбцовые хранилища для Big Data

Apache Cassandra

Cassandra разработана в Facebook в 2008 году для хранения входящих сообщений пользователей — задачи с высочайшими требованиями к доступности и скорости записи. Архитектура — peer-to-peer кольцо (ring topology): все узлы равноправны, нет выделенного мастера, нет единой точки отказа. Данные распределяются по кольцу на основе consistent hashing.

Tunable Consistency позволяет для каждого запроса выбрать уровень согласованности: ONE (ответ от одного узла, минимальная задержка), QUORUM (большинство узлов, баланс), ALL (все узлы, максимальная согласованность). Это позволяет гибко управлять компромиссом между CAP-свойствами в зависимости от требований конкретной операции. CQL (Cassandra Query Language) похож на SQL, но не поддерживает JOIN и произвольные GROUP BY — запросы должны быть спроектированы под конкретные паттерны доступа.

Apache HBase

HBase — распределённая столбцовая СУБД, работающая поверх HDFS. Вдохновлена статьёй Google о Bigtable (2006). В отличие от HDFS, HBase обеспечивает случайный доступ к отдельным строкам за миллисекунды. Модель данных: Table — Row Key — Column Family — Column Qualifier — версионированное значение (каждой ячейке может соответствовать несколько версий с временными метками). Region Server хранит и обслуживает несколько регионов (диапазонов строк). Используется совместно с Apache Phoenix — SQL-слоем поверх HBase, позволяющим выполнять стандартные SQL-запросы.