W11.a Big data

大数据（Big Data）

大数据的定义

- 大数据是指数据集规模极大，传统数据处理技术难以高效处理的数据。
- 大数据的

  3Vs

  特征：

  1. **Volume（数据量）**：数据规模大，如TB（太字节）、PB（拍字节）、EB（艾字节）。
  2. **Velocity（速度）**：数据生成和处理速度快，如实时数据流。
  3. **Variety（多样性）**：数据形式多样，包括非关系型数据（如嵌套关系、文档数据、XML、图数据、多媒体数据等）。

扩展的大数据特征

- **Veracity（准确性）**：数据的质量和可靠性。
- **Value（价值）**：通过机器学习、数据挖掘等技术，从数据中提取有价值的知识或商业价值。

传统关系型数据库的限制

- 传统的关系型数据库管理系统（RDBMS）使用SQL，但无法高效处理大数据。

数据库模型（Database Models）

常见的数据库模型包括：

1. 文件系统（File System）。
2. 层次模型（Hierarchical Model, IMS）。
3. 网络模型（Network Model, IDMS）。
4. 关系模型（Relational Model）。
5. 嵌套关系模型（Nested Relational Model）。
6. 实体-关系模型（Entity-Relationship Approach）。
7. 面向对象的数据模型（Object-Oriented Data Model）。
8. 半结构化数据模型（Semi-structured Data Model, XML）。
9. RDF和链接数据（RDF and Linked Data）。
10. 其他。

RDBMS中的问题与性能限制

冗余和更新异常

- 关系模型通过规范化消除冗余，减少更新异常。
- 但冗余不一定带来更新异常，以下为例子：
  - **例1**：供应关系（supply）中包含供应商名称和零件名称的冗余，但因这些字段不更改，不会产生更新异常。
  - **例2**：销售交易中存储冗余数据（如商品名称和价格）可以提高分析性能，无需频繁连接（join）操作。

RDBMS的其他限制

1. 效率问题：
   - 连接（join）操作代价高，性能低。
2. ACID特性：
   - 为确保事务的一致性，ACID（原子性、一致性、隔离性、持久性）需要高昂的开销。
3. 多值属性和复合属性处理不便：
   - 例如，嵌套关系需要拆分成多张表，查询效率低。

NoSQL数据库

NoSQL的特点

1. **灵活的模式（Flexible Schema）**：无需固定模式，避免复杂的对象关系映射（ORM）。
2. **高扩展性（Scalability）**：可以通过水平扩展实现大规模数据处理。
3. **放松一致性（Relaxed Consistency）**：牺牲一致性以提高性能和可用性。
4. **高性能**：适合快速设置和操作。

BASE模型

- BASE（Basically Available, Soft state, Eventually consistent）：
  - **基本可用（Basically Available）**：系统始终可用。
  - **软状态（Soft State）**：状态可能是暂时不一致的。
  - **最终一致性（Eventually Consistent）**：确保最终所有节点数据一致。

NoSQL数据库的分类

1. **键值存储（Key-Value Stores）**：
   - 数据以键值对（Key-Value Pair）形式存储。
   - 适用于简单存取需求，如Amazon Dynamo、Redis。
2. **宽列存储（Wide-Column Stores）**：
   - 数据按列族（Column Family）存储，适合稀疏数据。
   - 例如Google BigTable、Cassandra。
3. **文档存储（Document Stores）**：

- 数据以文档形式存储，如JSON、BSON。
- 例如MongoDB、CouchDB。

4. **图数据库（Graph Database）**：
   - 适合存储大量节点及其关系（如社交网络）。
   - 例如Neo4j、Google Knowledge Graph。

SQL与NoSQL的对比

特点	SQL	NoSQL
模式（Schema）	固定模式，模式演化难	无需固定模式，支持半结构化数据
数据类型	扁平关系型，定长字段	树/图结构，支持嵌套和多值属性
数据持久性	数据存储于磁盘，访问速度较慢	以内存为主，访问速度快
查询语言	使用标准SQL语言，声明式	需要编程，如MapReduce、JSON API等
事务支持	支持ACID，强调一致性	使用BASE，强调性能和可用性
分布式处理	有限支持	原生支持分布式和并行处理

大数据存储系统

分布式文件系统（Distributed File Systems）

- 特点：
  - 提供单一文件系统视图。
  - 支持大规模数据存储（如10PB）。
  - 数据通过复制应对硬件故障。
- 示例：
  - Google文件系统（GFS）。
  - Hadoop分布式文件系统（HDFS）。

分片（Sharding）

- 数据基于分片键（Shard Key）水平分割到多个数据库。
- 优点：易扩展。
- 缺点：应用需管理查询路由，跨数据库查询复杂。

MapReduce编程模型

简介

- 一种可靠的、大规模并行计算框架。
- 核心逻辑由用户提供（Map和Reduce函数），系统负责并行化和协调。

工作流程

1. **Map阶段**：对输入数据进行映射，生成中间键值对。
2. **Shuffle阶段**：根据键值对分组。
3. **Reduce阶段**：对分组数据进行归约，输出最终结果。

示例：单词计数

- 输入：带有文本的记录。
- Map函数输出：`("word", 1)`。
- Reduce函数输出：`("word", count)`。

Hadoop与MapReduce

- Hadoop是开源的MapReduce实现，支持分布式文件系统（如HDFS）。
- 提供Java编程接口，支持多种输入/输出格式（如CSV、JSON）。

W11.b Blockchain based storage

区块链

- **基础定义**
  区块链（Blockchain）是一种用于存储数据的数据库替代方案。
- **核心应用**
  - 区块链的主要应用是创建**去中心化的数字账本**（Decentralised Digital Ledger）。
  - 该账本由多个节点协作维护，每笔交易都经过数字签名以确保真实性。
- **特性**
  - **不可篡改性**
    一旦数据被添加，任何单一参与者无法删除或修改，而不会被其他参与者发现。
  - **去信任化**
    无需完全信任任何一个参与方，即可提供安全的数据存储和处理基础。

区块链分类

公有链（Public Blockchain）

- 特点
  - 任何人都可以下载必要的软件并创建一个区块链节点。
  - 不假设节点之间的信任。

许可链（Permissioned Blockchain）

- 特点
  - 节点需要获得权限授予方的许可。
  - 某种程度上放宽了对“去信任”和“自治”的要求。

区块链的结构

- **链式结构**
  区块链是由多个区块组成的链表，每个区块包含：
  - 指向前一个区块的指针；
  - 前一个区块的哈希值；
  - 第一个区块称为创世区块（Genesis Block）。
- **抗篡改性**
  - 若想篡改一个区块的内容，需要重新计算该区块及其后续所有区块的哈希值。
  - 数据的分布式存储（Replication）进一步增强了抗篡改性。
- **节点类型**
  - **全节点（Full Node）**
    维护完整的区块链副本并参与共识过程。
  - **轻节点（Light Node）**
    提交更新但不参与共识过程。

区块链的特性

- **去中心化（Decentralisation）**
  无需中央权威，依靠多数共识。
- **抗篡改性（Tamper Resistance）**
  修改区块链内容几乎不可能。
- **不可否认性（Irrefutability）**
  用户无法否认提交的交易。
- **匿名性（Anonymity）**
  用户身份（公钥）与现实世界实体无直接关联。

加密哈希函数（Cryptographic Hash Functions）

- **定义**
  哈希函数 h 必须满足以下性质：
  - **抗碰撞性（Collision Resistant）**
    不可能找到两个不同的输入 x 和 y，使得 h(x) = h(y)。
  - **不可逆性（Irreversibility）**
    给定 h(x)，几乎无法反推出 x。
- **用途**
  - 使用公钥加密消息；
  - 使用私钥对消息签名。

区块链交易模型

比特币（Bitcoin）

- **交易结构**
  交易包括：
  - 输入交易（Input Transactions）：用于指定花费哪些资金；
  - 输出（Outputs）：指定接收者和金额；
  - 数字签名：用户签名以验证交易真实性。
- **脚本语言**
  比特币使用简单的脚本语言支持更复杂的交易。

以太坊（Ethereum）

- **账户余额**
  以太坊维护账户余额，通过交易进行修改。
- **图灵完备语言**
  支持复杂的编程逻辑。

共识机制（Consensus）

- **作用**
  所有节点必须对区块链的新增内容达成一致。
- **分类**
  - **工作量证明（Proof of Work, PoW）**
    节点通过解决复杂的计算难题获得添加区块的权利。
  - **权益证明（Proof of Stake, PoS）**
    基于节点持有的加密货币数量决定添加区块的概率。
  - **拜占庭共识（Byzantine Consensus）**
    在许可链中，通过信息交换和投票实现共识。

工作量证明（Proof of Work, PoW）

- **原理**
  节点需要找到一个随机数 n，使得 `h(n || B) < 某个指定值`。
- **分叉（Forks）**
  当多个节点几乎同时添加新块时，可能出现分叉。

权益证明（Proof of Stake, PoS）

- **原理**
  持币越多的节点被选中的概率越高。

拜占庭共识（Byzantine Consensus）

- **拜占庭故障（Byzantine Failure）**
  节点可能以多种不可信的方式运行。
- **容错能力**
  最多允许 `(n-1)/3` 个节点失效。

数据管理

- 高效查询
  - 通过**未消费交易索引**提高查询效率。
  - 使用**默克尔树（Merkle Tree）**存储数据验证信息。

默克尔树（Merkle Tree）

- **定义**
  一种树形数据结构，允许区块链节点仅存储根哈希值即可验证数据。
- **验证过程**
  通过比较计算得到的根哈希值和存储的根哈希值验证数据完整性。

智能合约（Smart Contracts）

- **定义**
  智能合约是存储在区块链上的程序，满足特定条件时自动执行。
- 应用
  - 自动化执行协议；
  - 创建新型加密货币或代币；
  - 跨链交易。

以太坊的“Gas”概念

- Gas费用参数
  - **Gas价格**：用户支付给矿工的费用；
  - **交易Gas限制**：每笔交易允许消耗的最大Gas量；
  - **区块Gas限制**：区块中所有交易Gas的总和上限。

新兴应用

区块链技术的应用场景包括但不限于：

- 学术证明分发；
- 会计与审计；
- 资产管理；
- 电子政务；
- 外汇交易；
- 医疗保健；
- 供应链管理；
- 物联网（IoT）；
- 票务销售与转售；
- 贸易融资等。