- 存储系统
  - 存储系统分类
    - 传统存储系统
      - 块存储
        - 针对磁盘的区块，调用系统linux kernal的驱动操作
        - 典型如数据库，磁盘阵列，nas等
        - 性能往往因为随机访问数据块的次数而出现降低
      - 文件系统存储
        - 通过文件系统进行文件操作，写入数据到磁盘
        - 典型如ext，ntfs，nfs，samba文件系统，以及nas，san等
        - 文件系统存储无法应对海量的文件，因为文件数目太多，导致文件系统缓存，Inode存储无法使用
      - 对象存储
        - 数据作为对象存入存储系统时，会返回一个对象存储元数据，用于定位数据位置
        - 对象里的数据可以是结构的，也可以是非结构的
        - 数据存储在存储系统时，都在同一层，不会像文件系统一般分层
    - 分布式存储系统
  - 存储系统模型
    - 常规三大模型介绍
      - 块模型
        - hash模型
          - 实际举例 tokyocabinet
          - 数据结构
          - 功能优点 最快的增删改查性能
          - 功能缺陷 不支持条件查询，全局扫描，范围查询
          - mysql
        - 实际举例
          - mysql
          - hadoop
        - 数据结构（mysql）
          - B树
            - 结构特性 树的每个结点中存储数据
            - 块特性 B+树的叶子分裂会造成块的随机访问，严重影响性能
          - B+树
            - 块特性 B+树的叶子分裂会造成块的随机访问，严重影响性能
            - 结构特性
              - 树的最底层叶子结点存储数据
              - 树的最底层叶子结点，构成一个双向链表
        - 功能优点
          - 条件查询
          - 范围查询
          - 全局扫描
          - 支持增删改查
          - 以上以mysql为例，hadoop当我没说
        - 功能缺陷 针对条件查询需要额外的存储空间
      - 对象模型 实际举例 ceph
  - 存储系统理念
    - 分布式系统CAP
      - CAP介绍
        - C-Consistency： 一致性
          - 特性说明 任何一个读操作总是能读取到之前完成的写操作结果，也就是在分布式环境中，多点的数据是一致的；
          - 实现原则
            - 强一致性 当更新完存储系统的内容后，多客户端访问时，无论是那个客户端读出的值都是一致的
            - 弱一致性 当更新完存储系统的内容后，多客户端访问时，系统不保证后续读到的数据都是最新的，这之间存在一个窗口时间，只有满足一定条件后，才保证数据的一致性
            - 最终一致性 这是弱一致性的一种特殊形式；存储系统保证如果对象没有新的更新，最终所有访问都将返回最后更新的值。
        - A-Availability：可用性 特性说明 每一个操作总是能够在确定的时间内返回，也就是系统随时都是可用的
        - P-Tolerance of network Partition：分区容灾处理性 特性说明 在出现网络分区（比如断网）的情况下，分离的系统也能正常运行；
      - 实现形式 分布式系统，一般无法同时满足CAP，实现会造成性能上的损失，因此一般只存在CP（并不代表不实现A的功能）和AP（不代表不实现C的功能）
    - 关系型数据库 ACID
      - A 原子性(Atomicity)
      - C 一致性(Consistency)
      - I 隔离性(Isolation)
      - D 持久性(Durability)
  - 常见难点
    - 海量小文件存储 关键问题
      - 元数据存储问题
        - 假设一个元数据需要100字节，则一百亿条数据，但是元数据信息就有1T，算上其它结构类信息将会更大。以hadoop的namenode为例，是无法准备这么大内存的机器
        - 常见于存在元数据中心的的存储系统，典型如Hadoop
      - 文件系统管理问题
        - 在读取文件时，先读取文件名，以获得Inode结点号，根据Inode结点号获取data存储信息，最后定位具体数据。由于文件太多，导致Inode信息无法缓存，甚至文件缓存也难以使用
        - 典型如glusterfs，ceph