* 什么是分布式系统？
  + 用多台机器解决一台机器上不能解决的问题。e.g. 存储不够， QPS太大
  + 两派： Google用便宜机器， Sun用超级计算机（Sun已经不复存在）
* 谷歌三剑客： Distributed File System， MapReduce， BigTable
* 以Google File System为例 讲解 Distributed File System
* Takeaways: 掌握 Master Slave Pattern / How to check and handle system failure and error / How to design Distributed File System
* GFS（高富帅） vs HDFS（Hadoop File System）:
  + GFS： 不开源
  + HDFS: Yahoo / Alitaba Open Source of GFS (离职google科学家重写了GFS)
* 用4S方法来分析Design GFS (分布式文件系统（存储file）)：
  + Scenario: 哪些场景
    - 最基本功能 （读 / 写）
    - 用户读取一个文件，用户写一个文件？支持多大的文件？(越大越好： > 1000T 1PetaByte -- 全部文件 )
    - 多台机器存储这些文件。支持多少台机器 （2007 年 Google 10万台机器 1
    - 1 hundred thousand）
  + Service:
    - 分布式系统 是偏 后端 ， 和 前端/ web service 不太一样， 这个是后端，提供给 web service 的。 Client + Server ? Is it enough? No. 需要多台机器， 一台机器存不下
    - 多台机器之间 沟通的方式：
      * 第一种结构： Peer-to-peer: 所有Server平分， 三台机器是等价的。 如果机器1 存储数据1， 机器2存书局2， 机器之间需要沟通，同步。备份机制，每台机器保证数据是一致的 (同步是一个问题)。data 不是 partition的逻辑关系，而是 备份的关系，需要 eventual persistence
        + 问题：没法很好同步数据,因为需要沟通，将数据2从机器2传到机器1
        + 优点：如果一台机器挂了，另外的机器能保证部分数据依然能获取。能通过任意一台连接
      * 第二种结构：Master / Slave 结构 。 老大哥master管理三台Server机器。数据是采用partition 的方式存储，某一块数据只存储在 某一台机器上
        + 优点：存储方式比较简单，数据**不需要同步**，master来管理。每台Server机器存储不同的数据。master来分配，知道哪个data 存储在哪里
        + 缺点：single point of failure. 如果master挂了，整个系统就跪了
    - GFS 的 master / slave 和 DB 的 Master / Slave 的区别：DB 中master存储数据， DBslave用来backup。一个挂了另一个上。但是GFS的系统里，master是管理者的角色，不存储数据，slave用来存储数据
    - Peer 2 Peer example: BitComet 下载的电影是 别人分享给你的, Cassandra, 不是课程重点）
    - Master / Slave： 大部分都是Master / Slave 结构：master是管理者，设计比较简单，坏处是， master挂了就挂了。 大部分系统使用的是master / SLAVE， 挂了 重启一下服务就是，挂的概率在 0.1%。 解决方案： 使用一个hot Backup Server来随时备份Master
    - master <-> hot backup （很多系统甚至都不用 backup，master挂了重启就好）
      * slave1
      * slave2
      * slave3
  + Storage：
    - 大文件存在哪儿：
      * 内存？数据库？文件系统？ 数据库存储的是**结构性**数据，文件不是很适合存储在数据库中。 内存有限制，并且是volatile。所以最后会存在**文件系统**。
    - 怎么样存在文件系统：
      * Interview Question: How to save a regular file in a machine?
        + Separate **Metadata** with **File content**. 因为文件内容可能只访问1次，但是（文件名\大小\时间\Last Visited\file location）经常会被访问 （搜索文件，打开文件夹。。。）
        + 文件内容File content 连续存储 vs. 分开存储。

连续存储： windows -- 需要进行碎片化整理。Fragmentation。

好处：读的比较快.找到一块，就找到了所有的

坏处：如果某一天删掉了一个file （dengchao.MP4），删掉那一块的地址空间可能没法被后续使用 （比如后续存的file特别大），这样会产生很多碎片）

**分开存储**： Linux 的存储方式。每个文件拆成一块Block(1KB - 1024 Byte)。把一个大的电影拆开，分别一个小块一个小块放。

好处：碎片化整理的需求减少。把小块放在空隙中。 Question:怎么决定block的大小？如果**block**太大：最后一块浪费会比较多。如果block太小，metadata中存储的index会占用太多空间。 分开存储另一个优点：方便检查错误,如果某一块坏掉，直接 替换这一块就好。当前1KB大小是经验之谈

缺点：读的慢一点

* + - 进阶：How to save a large file ：100T size，可以认为是很多文件合成一块。HDFS经常会 merge file，将几个文件合成到一个) in a machine？
      * 如果可以拆分成 100 \* 1000 \* 1000 \* 1000 block。问题：索引文件非常大。
        + solution: Google 提出了chunk的概念 可以认为是 big block： 1 chunk = 64 M = 64 000 blocks。

优点：缩小metadata的存储量。

tradeoff: 小文件会浪费存储量。 为什么浪费的空间不那么重要：64M的size是这样决定的： 因为，在GFS中大的文件比较多，所以并没有很多的浪费

chunk 其实就是 大 block 的概念。文本信息 4KB，MP3 x MB， image 200KB。如果存储的都是比较大的文件，那么浪费的也就没这么多。tradeoff 问题。

Google 一开始chunk 下面还有block的概念，后来觉得太麻烦，取消了 block的概念

* + - How to save an extra-large files in several machine ? (10 Peta byte file size in total ， 很难找到那么大块的硬盘， 10T 的硬盘还是买的到的，一定要用多台机器来实现)。总共存储量 10Peta Byte
      * 如何维护多台机器？Master / Slave 来管理多台机器 One master + many chunk servers
      * 如何存储：Master存储File 的 metadata（邓超.MP4的大小，时间，位置（哪个chunk server 以及具体的哪个 chunk））） ， 对应的chunk信息存在slave上面，叫做chunk servers。Master中存储 chunk在哪个机器的哪个offset上面。问题： master被累死， metadata 太多，
        + 小优化： 每个chunk在chunk server的偏移量offset存储在对应的chunk server上，这样可以减轻master的负担，（master只是说 chunk 01 在 chunk server 02 上， 然后去chunk 02 上再存储一个 map 表明 chunk 01 具体 在 哪个 offset）。同时，减少了master/slave的沟通: 如果每个chunk在chunk server的offset变化了，master不需要知道。有效减少了 master / slave 之间的偏移量
      * Master 的metadata：
        + 存在内存还是硬盘？
        + 存储10P文件的metadata需要多少容量：

1chunk = 64MB 需要 一个指针 4Byte， size int 4B， 名字 ， text， 差不多 64Byte （4Byte 一个property大概64Bmetadata信息量）。所以基本上 是1MB 对应 1B 的metadata， 10P 也就是 10^15 B, 减掉6个0，就是 10^9 也就是 10GB

10P需要10G的metadata信息量，10G完全可以放在内存中。同时在硬盘中进行备份。 Serving的时候其实是从内存中访问，访问内存比访问 硬盘快一个量级 1K 倍。

* + - How to Write a file? 如何写到文件系统：直接写还是碎尸万段写？碎尸万段写：
      * 一次写和多次写的区别？
        + 一次传输如果中间出错 ，需要重新传输整个文件。如果多次传输，则只需要重传出错的部分。迅雷的 断点续传 就是这样的功能。多次写的话，每次的大小是多少？ chunk的size
        + 是在client端切，还是server的master端来切？ Client 切，并不是一下子把胖子写进去。首先是client 要存到 GFS中，如果让master来分。一份份分开，传给master，按照chunk大小，写到GFS中。如果是 576M，就切成9份，然后传到GFS中。
      * 每个chunk 如何写入server? 直接写入chunk 还是 通过 master写
        + Master Server看看哪个Chunk Server不干活（idle），），然后把chunk server id传给client，告诉client chunk server id 1比较闲，传给它吧。然后client写入chunk server 1. 然后chunkServer告知master写入完成。

chunk server 是否可以不告诉 master 传完了？

不可以，网络不稳定，可能会丢包，出错，写失败是有可能的， 要告诉master 写 成功了

* + - * Client到底指什么？
        + 指的是Google Drive / Google docs的web server 或 Database。并不是 browser user interface。client的概念是相对的。

Database 和 Web Server 都是 GFS的 client

* + - How to modify a file??
      * 要修改Dengchao.mp4为sunli.MP4怎么办？
      * 三个思路：
        + 找到 那个 chunk，然后直接写硬盘修改（错）

将硬盘上的 2 改成 10^10 （4 Byte int存不下了， 要改成 8Byte long）；

* + - * + 或者读出来，写进去（错）；

读出来修改可能存不下，写回还是不行

* + - * + 第三种，写在另一部分（！正确）
      * 其实没有修改操作，直接删掉原 chunk，作废了。别的数据可以 ，然后重写。在新的chunk重写，然后change index. One time to write, many time to read. GFS不支持modify，直接重写。
      * 修改次数 <<< 读取次数。 如果同时读和写，需要锁，所以GFS为了支持快速写和快速读，所以往新的地方去写。
    - How to read a file?
      * 一次读整个文件，还是多次读chunks
        + 读多次， 问老大master，知道 mp4 视频在哪里
        + Client 问 master，get file 对应的chunk servers
        + Client 从 chunk server 读取
    - Master任务：
      * 存储每个文件的metadata，维护每个文件到底在哪里
      * read时帮助找到 对应的 chunk server
      * write 时，分配空闲的 chunk server 给 client 去 写
      * 存储Map (File name + cunk index + chunkserver）
    - 为什么不在Write的时候，直接 让 master 去 写，让master 写到 chunk server 上？
      * Master 会有bottleneck，这样做并不是特别好
  + Scale:
    - 系统如何优化与维护： **GFS精髓，**解决了多台机器 协同工作的问题
    - 单 master 够不够？ 工业界90%系统都采用单master. Simple is best. Facebook发了一篇关于双 master的paper: Apache Hadoop Goes realtime at Facebook. Master 太多 同步很麻烦
    - How to identify whether a chunk on the disk is broken?
      * Checksum校验码： MD5，SHA1， SHA256, SHA512 -- 存在 chunk server里 （最简单的checksum : xor 异或 ）。每次 把 checksum 写在 末尾。 如果某一天 读 这个 数据的 时候，发现 checksum 和 存 着的 checksum 不同，就说明 这个 chunk is broken。看一下wiki checksum 里面的 介绍
      * 通过计算checksum来决定这个chunk server是否坏掉。
      * 什么时候把checksum写进chunk？ 写完每个chunk后在末尾加入checksum。如果checksum 本身坏了，也说明 整个 chunk坏了
      * 1 checksum = 4bytes， 1P / 64M \* 4B = 62.5 MB checksum size可以接受
      * 每次写完chunk之后在末尾 加入这个 checksum
      * Master的内存也存储这个checksum
      * 电影如果 看了很久，数据会发生跳帧，就是数据发生了错误
      * 什么时候检查checksum？
        + 每隔一个月检查一下 ： periodically
        + 读数据的时候
    - How to avoid data loss when a chunkServer is down/fail?
      * Replica: 双保险 -- 备份。备几份比较好，存三份。两份比较近（美国）
        + 一共三份；一份和 original 相对比较近，用于快速恢复，一份放在中国（如果主机房发生火灾，另一份能恢复）。
      * 选chunk server有什么策略？
        + 最近写入比较少 （LRU）谁不干活 就 把备份放在那里
        + 硬盘存储比较小
    - How to recover when a chunk is broken 如何修复?
      * 找妈妈。 一个chunk存在了Chunk server CS3， CS5， CS4. 如果CS3的chunk坏了，找master，知道兄弟在哪儿，然后通过CS5或CS4来恢复。然后靠兄弟去还原备份
    - How to find whether a ChunkServer is down? 机器蓝屏了
      * 如何发现蓝屏了： **Heartbeat**. Master定时接收chunk server的ping。 单向： 每隔5-10分钟chunk server向master发送心跳。master 没有必要去主动问。master只需要等就好了。如果过了一天都没人，知道挂了！如果有人偷了海底光缆，master就知道挂了！
    - Read/Write Scale:
      * 之前只写1个chunk server，现在client要往三个replica都写好。 How to solve client bottleneck （client 每份都要写三遍）? 选择chunk server 队长。 由chunk server队长 （CS1）写给CS2 和 CS3.服务器之间的写会快很多，减少了client的写入量。 当写完成时，chunkServer1高速client写完成。 怎么选队长？ 找距离client最近的或者最闲的 。master 每次 分配三台机器，并 分配 队长是谁。
      * 如果写的过程中，一个chunk server挂了-- 怎么办：chunk server 队长没有收到cs3的确认。cs队长主动承认错误， Retry。 这时候chunk server队长发送给client 写失败。 client再次从master reuquest chunk server，master重新分配chunk server （这时就不会分配给原 机器了）
      * 如果cs3坏了，怎么办？
        + 修当前机器 还是？
        + 把当前 机器 decomission，从 云备胎 里拿一个新的机器，用cs3数据备份到cs4上（或许需要cs3的备份），然后用cs4替换cs3。3号有另外两台机器做备份，然后4号可以从备份去拿
    - 看google原题： www.jiuzhang.com/qa/627

GFS 的好处： Master / Slave Pattern； How to handle failure; How to use GFS.

* 看 GFS 论文：
* takeaway：
  + Master / Slave PATTERN
  + How to handle failure

GFS 实现好后，对外界来说就 变成了一个Service：

* Read
* Write

Google Drive 的client-server结构：

* browser hit web server
* web server （文件的整理，查询，搜索） 会把 多个 文件 汇集成 一个 大chunk，然后存储到 GFS上。
* 普通的google drive 不是 直接存在 GFS 上，而是 暴露其他接口
* GFS 向 web server 提供的 接口只有 read / write。