Tales of your misdeeds are told from Ireland to Cathay.

云计算技术

从爱尔兰到契丹

O'RLY?

Meisha

VM 与 DOCKER

vm
- 虚拟机在本质上就是在模拟一台真实的计算机设备，包括硬件和独立的操作系统。虚拟机一旦被开启，预分配给他的所有资源全部被占用。
- 虚拟机的 guest os 是一个完整的操作系统内核。hypervisor 层可以简单理解为一个硬件虚拟化平台，在 host os 是以内核态驱动存在的。
- 虚拟机实现资源隔离的方法是利用独立的 os，并利用 hypervisor 虚拟 cpu、内存和 IO 设备等。比如 hypervisor 虚拟 cpu,创建一个数据结构，存储模拟 cpu 的全部寄存器；模拟内存，创建 shadow page table，实现虚拟内存到虚拟物理内存到物理内存的翻译；模拟 IO，以软件的方式实现并返回，比如写磁盘，hypervisor 把要写入的文件写入 host os 中的一个文件上，这个文件就模拟虚拟机的磁盘。

docker
- docker 是 Linux 容器的一种封装，提供简单易用的 Linux 容器使用接口。
- Linux 容器不是模拟一个完整的操作系统，而是对进程进行隔离，相当于在正常进程外面套了一个保护层，对于容器里面的进程来说，它接触到的各资源都是隔离的，从而实现和底层系统的隔离。
- docker 最大的用处是将应用程序从基础设施中隔离开，并将基础设施作为一个管理平台。
- dockER engine 可以堪称对 Linux 的 namespace，cgroup、镜像管理文件系统操作的封装。
- Docker 并没有和虚拟机一样用一个完全独立的 os 实现环境隔离，它利用的是目前 linux 内核本身支持的容器方式实现资源和环境隔离。
- docker 利用 namespace 实现系统环境的隔离，利用 cgroup 实现资源限制，利用镜像实现根目录隔离。
- 一个可读写的统一文件系统加上隔离进程应用和进程，容器里对文件系统的读写都作用于读写层。

对比
- docker 占用资源少，启动块
- vm 较为臃肿，启动慢
- docker 容器与内核交互，几乎没有性能损耗，hypervisor 性能较差
- docker 快速分发，vm 镜像分发没有体系化
- docker 是进程级别的隔离，vm 系统级别的隔离

ZOOKEEPER

Zookeeper 简介
- Apache Zookeeper 提供分布式协调服务，包含一个简单的原语集。分布式应用程序可以基于它实现服务同步，配置维护和命名服务等。

为什么 ZK 集群的数目一般是奇数个

- Paxos 核心思想，多数 server 写成功则任务数据写成功。3 台 server2 个成功，4 个 server3 个成功，5 个 server3 个成功。
- 选举，偶数有可能选不出
- 过半数有效，3 挂 1，4 挂 1，容灾能力一样。

三角色
- leader。负责投票的发起和决议，更新系统状态，自身也有投票权。
- follower，接收客户端请求并返回结果，参与投票。
- observer，接收客户端连接，将写请求转发给 leader，不能参与投票。同步 leader 状态，扩展系统，提高读取速度。

znode
- 保存在内存中的数据节点，以类似目录的树形命名空间进行管理。
- 临时节点。生命周期与客户端会话绑定，会话生成，实效清除，不能有字节点
- 序列节点，父节点负责维护字节点创建的先后顺序。如果撞见的事序列节点，父节点自动为这个极点分配数值后缀，追加到节点名中。

watcher
- 监视器，监控节点的数据变化、子目录变化。变化发通知
-

写操作
- leader 接收写请求，来自 client 或者 roller 和 observer 转发
- leader 发起请求
- follower 回 ack,leader 收集
- 过半数以上，向所有 f 和 o 发送 commit

实现高可用
- 持久性 znode,/regitry
- rmi 注册，生成临时且有序的 znode，存放 rmi 地址
- 消费者监听/registry 的 Nodechildchange，一旦变化，更新 rmi 地址

实现分布式锁
- 在/lock 节点下创建临时有序节点
- 获取/lock 节点的所有字节点，自己是不是最小的。
- 是则获取锁
- 不是则向比自己小的节点注册 watcher，监听该节点是否存在，进入等待
- 当获取锁的任务完成，删除临时节点。
- 下一个节点收到通知，更新字节点 list，发现自己是最小的了，获得锁。

zookeeper 工作原理
- 每个 server 在内存中存储一份数据，定期存回磁盘
- zk 启动时选举 leader,负责数据更新
- 当且精当大多数 server 在内存中修改成功，zk 数据才算更新成功

HBASE

HBase 简介
- HBase 是一个高可靠性、高性能、面向列、可伸缩、实时读写的分布式数据库
- 利用 Hadoop HDFS 作为其文件存储系统。
- 利用 mapreduce 处理海量数据
- 利用 zookeeper 作为分布式协同服务
- 主要用来存储非结构化、半结构化的松散数据(列式存储、noSQL)

HBase 数据模型
- rowKey，相当于主键，唯一确定一行数据。按照字典顺序升序。RowKey 只能存储 64KB 的字节数据。
- Column Family、Qualifier。列族属于表模式的一部分，在定义表的时候给出。每个列都属于一个列族，列可以按需、动态添加。列名以列族为前缀，每个列族可以有多个列。同一个列族在同一个 store 里。
- timestamp。同一份数据有多个版本，根据唯一的时间戳区分。不同版本的数据按照时间戳倒序排列。
- cell，由<rowKey, column(=<cf>+<qualifier>),version>唯一确定。有版本，内容是未解析的字节数组，没有数据类型。
- region。Hbase 自动把表划分为若干个 region，每个 region 存储表中某段连续数据。region 增大，会被切分。

HBase 体系结构
- HLog，write ahead log。记录写入数据的归属信息
- zookeeper，监控 region server 的上下线信息，通知 Master；保证 Master 可用；存储 region 的寻址地址，存储 HBase 的表模式信息和表的元数据。
- Master。为 region server 分配 region，负责 region server 的负载平衡，发现失效的 regionserver，重新为其上的 region 分配 server。管理用户对表的增删改查操作。
- region server，维护 region，处理 region 的读写请求。负责 split 过大的 region。

memstore 和 storefile
- 一个 region 由多个 store 组成，一个 store 对应一个 cf
- store 包含位于内存中的 memstore、位于磁盘的 store file
- 写操作先写入 memstore，当 memstore 中的数据达到一个阀值，region server 会自动执行 flush 操作，持久化 memstore 到 store file。每次 flush 形成单独的一个 store file。
- 读数据时，在 memstore 和 store file 里找。
- 当 store file 数量阀值，合并，形成更大的 storefile。
- 当一个 region 的所有 storefile 的大小和数量超过阀值，region server 切割。master 分配到相应的 region server，实现负载平衡。

region&store
- region 是 HBase 中分布式存储和负载平衡的最小单位。
- region 由一到多个 store 组成，每个 store 对应一个 column family
- 每个 store 包含一个 memstore,0 或多个 store file
- 每个 store file 都以 HFile 格式保存在 HDFS 上

病人表
- row key。病人+timestamp
- column f.病人信息
- qualifier，心跳，提问
- cell ,值

book & user
- userId data:fname rating:bookid rating：bookid

树形数据
- rwo key，root id
- cf,P 父节点，C 字节点
- qualifier，父节点或者字节点 id
- cell 等于节点类型

继承映射，商品表
- row key 商品类型+ID
- CF，产品信息
- qualifier，产品属性
- cell，属性值

自连接关系
- row key。关系类型作为分隔符，follows，followedBy

hdfs 优点
- 适合大数据处理
- 高容错性。数据保存多个副本，副本丢失后自动恢复。
- 可构建在廉价机器上。机器故障是常态，需要通过多副本提高可靠性、容错和

恢复机制。
- 适合并行分布计算，移动计算代码，而非数据。数据位置暴露给计算框架。
hdfs 缺点
- 不适合低延迟数据访问。HDFS 寻道时间长。
- 不适合小文件存取。1. 小文件元数据代价高，nn 的内存压力很大。2. 大量小文件读取性能差,需要从一个 datanode 跳到另一个 datanode。3. 小文件写性能差。写数据流的时间远远小于 RPC 交互、建立 socket、磁盘寻道等时间。
- 不适合并发写入、文件随机读写。不能保证并发写入的数据一致性。不支持随机写文件。一次读入，多次读取。
hdfs 适用场景
- 提供高可靠、高扩展性、高吞吐量的应用程序数据访问服务。
- 高容错性，多个副本。
- 适合大数据
- 适合批处理、大文件流式读取，不适合大量小文件存储。
- 不适合低时间延迟的访问，因为文件读写时经过的通信次数较多。
- 不适合并发写，不支持随机写。
- 一次写入，多次读取，不支持修改。
为什么 hdfs 不适用于小文件
- 大量小文件的元数据空间开销太大了，namenode 的内存压力太大
- hdfs 的设计是为了流式读取大文件开发的，访问大量小文件需要不断从一个 datanode 跳到另一个 datanode，严重影响性能。
- 写入一个文件，经过的通信次数比较多；如果对于一个大文件，它写数据流的时间远远大于 RPC 通信，socket 建立连接，及其磁盘寻址等时间，因此大文件比较适合；如果对于小文件则相反，其他方面所消耗时间比写数据流的时间消耗得多。

HDFS 小文件存储方案
针对小文件问题，HDFS 自身也有考虑这种场景，利用 SquenceFile、HAR、CombinedFile 等技术将多个小文件归档成一个大文件进行处理。

写数据的过程中，pipline 上的一个 Datanode 失效了，HDFS 该如何处理
如果在写入过程中发生数据节点错误，请执行以下步骤：1）关闭管道； 2）已发送到 pipline 但还未收到确认的数据包写回数据队列，确保数据包不丢失。； 3）正常 DN 上已保存的 block ID 升级，这样故障的 DN 会在恢复正常后删除这个 Block 数据，实效节点从 pipline 删除 4）将其余的块写入其中，保留两个普通的 datanode； 5）名称节点找到另一个数据节点以创建块的副本。

HDFS 如何保证数据的完整性
HDFS 计算所有写入数据的校验和，并在读取数据时验证校验和。
管线中写入数据的最后一个 datanode 负责验证写入数据的校验和。如果检测到错误，则抛出 ChecksumException，并且客户端接收该异常并执行特定的处理，例如重试该操作。
客户端从数据节点读取数据时，还将验证校验和。验证每个数据块后，将通知 datanode 更新校验和日志。

冫每个数据节点均允许后台线程中的 DataBlockScanner 定期验证此数据节点上存储的所有数据块的校验和。

HDFS 高可靠性
- 冗余副本。
- 相同 block 的不同副本分布在不同机架。也可以提高带宽利用率。
- 心跳机制。DN 定时向 NN 汇报当前状态。DN 实效后及时恢复。
- 安全模式。NN 启动时先安全模式，收集 DN 报告，检测到副本数不足的数据块进行复制。
- 校验和。每个数据块写入时有校验和，验证数据完整性。

block
- 文件被切分为固定大小的数据块，随机存储到不同节点。默认 128MB，大小不足 128，单独存成一个 block。
- 每个 block 默认有 3 个副本，分布在不同机架的 datanode 中。

namenode
- 主要功能：接收客户端的读写请求
- 保存元数据（ownership&permission， 文件包含哪些数据块， 数据块副本保存在哪些 datanode 上）。block 的位置信息保存在 datanode，启动时由 datanode 自动上传（不在 fsimage）
- metadata 在启动后加载到内存
- metadata 存储--磁盘文件「fsimage」，block 的位置信息不存。
- editslog，对 metadata 的操作日志。

secondary namenode
- 帮助 NN 合并 fsimage 文件和 editlog 文件
- NN 启动时读取 fsimage，恢复到某个时间的检查点，然后读取 editlog 进行重建
- editlog 太大，NN 启动时间过长。由 SN 辅助 NN 完成 editlog，保持 fsimage 一个比较新的状态。

datanode
- 存储数据块
- 启动 DN 时，向 NN 汇报保存的数据块信息。
- 发送心跳了解，包括 DN 的负载信息。
- NN10 分钟没收到 DN 的心疼，认为 DN 丢失，复制其数据块到其他 DN。

block 副本存放策略
- 多个副本
- 新的块副本放在低于平均硬盘使用率的 DataNode 上
- 限制每个 dataNode 的最近创建块次数。
- 块副本分布在多个机柜上。
- 目的：提高可靠性可用性，提高带宽利用率

一、 虚拟化和 Docker

各自怎么虚拟化资源，提高系统资源，区别联系？

虚拟机（Virtual Machine）指通过软件模拟的具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统。
简单来说：我们通过虚拟机可以把一台硬件划分为多个机器，每个机器都能独立运行。
容器技术也能实现在一台硬件机器上虚拟出多个容器，每个容器中都可以放置一个互相隔离的服务器。所以，我们知道，容器其实也采用的是虚拟的方式进行，只不过，同等条件下，容器技术会比虚拟机技术效率要高，但是容器中没有包含独立的操作系统，虚拟机中包含独立的操作系统，所以每台虚拟机中的功能会比每个容器的功能要多些，正因为虚拟机带有很多东西，所以占的系统资源要大，效率要低些。

https://www.jianshu.com/p/9134ac70c776

属于 Infrastructure Service

1.2 云计算（Cloud Computing）——大数据的计算

（不重要：什么是云计算？hadoop 和云计算关系？）

二、 HDFS

属于 Platform Service，作为接口方便向下使用资源

Hadoop：（1）分布式文件系统 HDFS：提供高可靠、高扩展性、高吞吐率的分布式数据存储服务。（2）并行、分布式计算框架 MapREduce：具有易使用、高容错性、高扩展性等优点。

HDFS 的特点：

优点：（1）高容错性：数据自动保存多个副本（默认三个）。部分副本丢失后，自动恢复。

（2）适合批处理：移动计算代码而非数据；数据位置暴露给计算框架——MapReduce

（3）适合大数据处理

（4）存储空间大

缺点：（1）低延迟数据访问：HDFS 上寻道时间超过读取时间

（2）不适合小文件存取

（3）并发写入、文件随机读写

适合场景是什么（例百度云）？为什么适用？

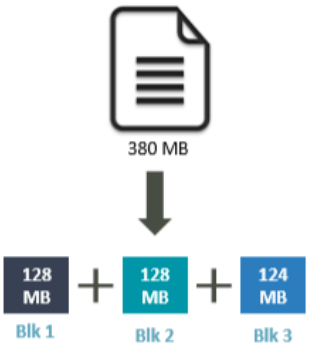
百度网盘里的文档、图片等小文件如何存储？百度网盘里的文件可以修改吗？

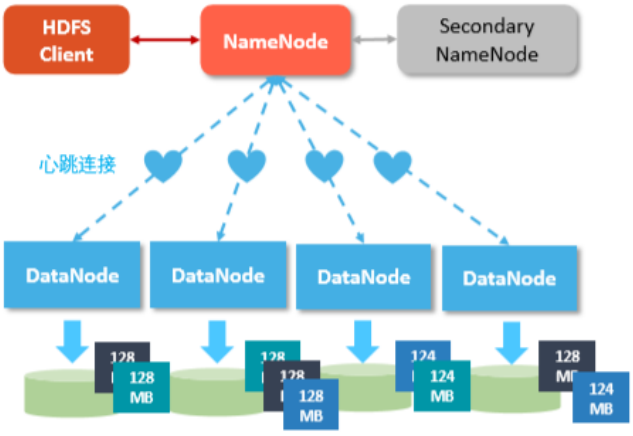
百度网盘文件不能修改（不能覆盖），一次读写顺序读写，不支持随机读写，希望批量写（传同名文件直接改名存储宁愿浪费空间）

DataNode、NameNode、SecondaryNameNode

一个文件存储方式：按大小分成若干个 Block，随机存储到不同节点（DataNode）；

默认情况下，每个 Block 都有三个副本（可配置）





NameNode （NN）

- NN 主要功能：接受客户端的读写请求
- NN 保存元数据（Metadata），包括：
  - 文件ownership和permissions；
  - 文件包含哪些数据块（File -> Blocks）；

  } 保存在NameNode上
  - 数据块的各个副本保存在哪各DataNode上（Block -> Replicate）

保存在DataNode上
启动时，由DataNode主动上传

- NN 的元数据（Metadata）在启动后加载到内存
  - Metadata 存储——磁盘文件『fsimage』
  - Block的位置信息不会保存到『fsimage』
  - 记录对Metadata的操作日志——磁盘文件『editsLog』

Secondary NameNode （SNN）

- 不是 NN 的备份（但可以备份），其主要工作是帮助 NN 合并fsimage文件和editsLog文件，减少 NN 启动时间。

- SNN 执行合并时机
  - 根据配置文件设置的时间间隔 fs.checkpoint.period （默认 3600秒）
  - 根据配置文件设置的editsLog大小，fs.checkpoint.size 规定了edisLog文件的最大默认值 64MB。

为什么要合并FsImage和 edits Log ？
合并过程为什么由SNN来做？NN 为什么不做？
NN 突然掉电或宕机，重新启动后，会不会丢失数据？
如果一年不执行合并操作，会有什么问题？
NN 硬盘彻底损坏，怎么恢复数据？会丢数据哪些数据？

DataNode （DN）

- 存储数据块（Block）
- 启动 DN 线程时，向 NN 汇报保存的数据块信息。
- 通过向 NN 发送心跳保持与其联系（3秒一次），包括 DN 的负载信息。
- 如果 NN 10分钟没有收到 DN 的心跳，则任务该 DN 已经丢失，并复制其上保存的数据块到其他 DN。

DN 最少几台？

三、 MapReduce 运行过程

WordCount 程序

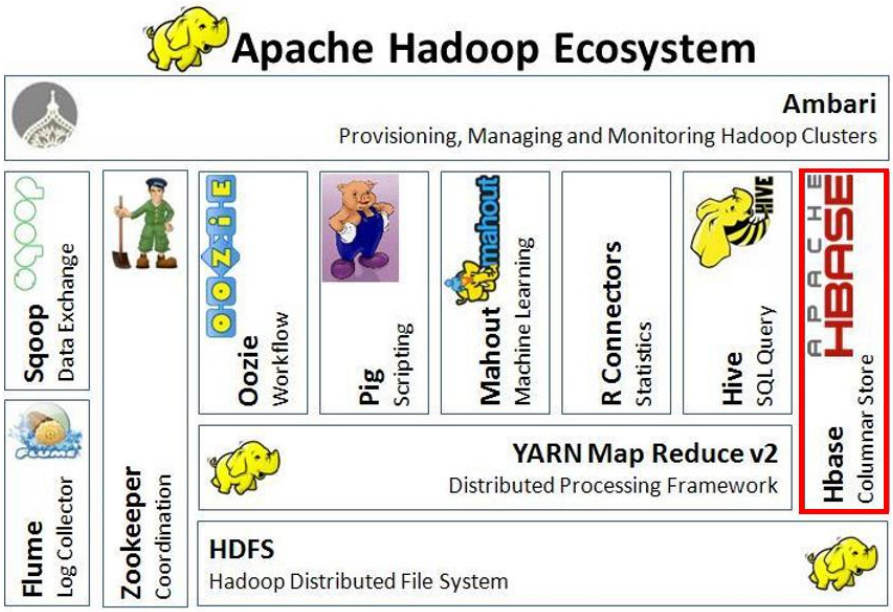
Mapper、Reducer、sluffer 各阶段的过程及每个部分的输出

（不用写 MapReduce 代码）

四、 Hbase（hadoop 数据库的角色）

Hadoop生态系统简介

计算机科学与技术学院 School of Comp. Sci. & Tech.



- **Apache HBase**——Hadoop Database, 是一个**高可靠性**、**高性能**、**面向列**、**可伸缩**、实时读写的分布式数据库。
- 利用 Hadoop **HDFS** 作为其文件存储系统。
- 利用 Hadoop **MapReduce** 来处理HBase中的海量数据。
- 利用 **ZooKeeper** 作为其分布式协同服务。
- 主要用来存储**非结构化**、**半结构化的松散数据**（列式存储、NoSQL 数据库）

Hbase 体系架构：Hmaster、HregionServer.....

Hbase 数据结构

考试方法：（1）理论；（2）根据具体查询需求设计 Hbase 表格

五、 ZooKeeper

选举服务器（Master、Leader）

Zookeeper 实现 Paxos 算法，数据一致性

工作原理，优缺点

（1）为什么集群数目一般是奇数个?

（2）Zookeeper 的角色：Leader、Learner、Observer

三个服务角色及作用怎么存数据?

Znode 类型：临时节点；序列节点（各自作用）

监视器 Watcher

如何构造实现分布式锁、高可用服务

六、Hive

- Hive：数据仓库。
  - 不是一种交互式；存历史数据；引入冗余，不要求遵守范式；
  - 面向分析任务，不是事务型任务。
- Hive：解释器、编译器、优化器等。（SQL → MapReduce）
- Hive：运行时、元数据存储在关系型数据库（MySQL）里。
- Hive：擅长非实时、离线、对响应及时性要求不高的海量数据批量计算，统计分析。

MapReduce 如何实现自连接?

（两列表父母 id，子女 id）