分布式爬虫

云峰

www.weibo.com@fengyuncrawl

2011.12.10

提纲

- 全网爬虫和垂直爬虫
- 分布式框架
- 海量数据存储
- SSD作缓存和虚拟内存
- Q&A

通用爬虫

初始url（种子）

下载页面

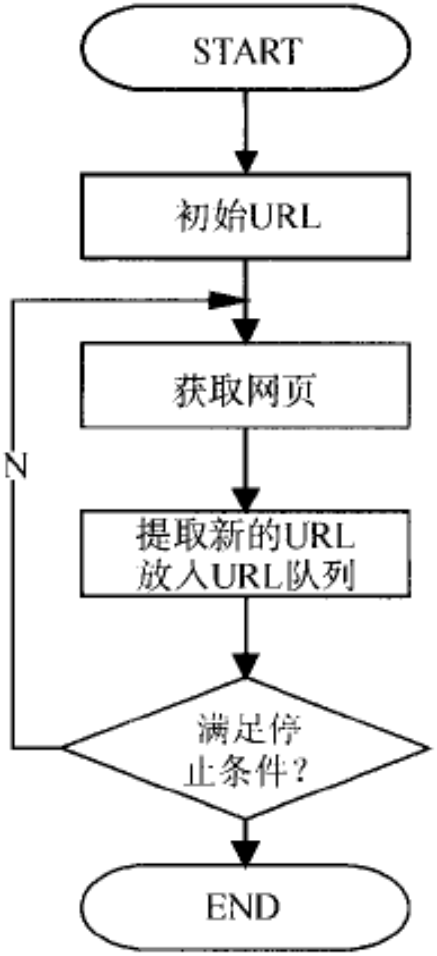
解析

将抽取的内容保存

缺点：
1. 噪音多

2. 信息密度低，无联系

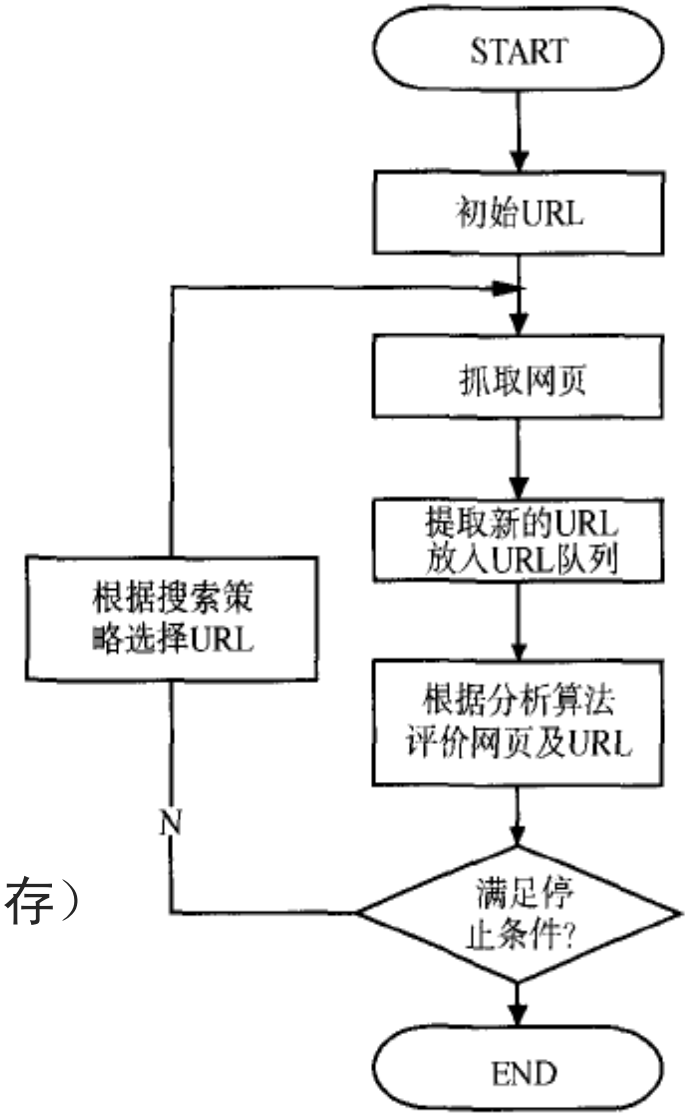
3. 结构简单化



深度爬虫

增加
1. link分析，网页权重计算

2. URL权重队列



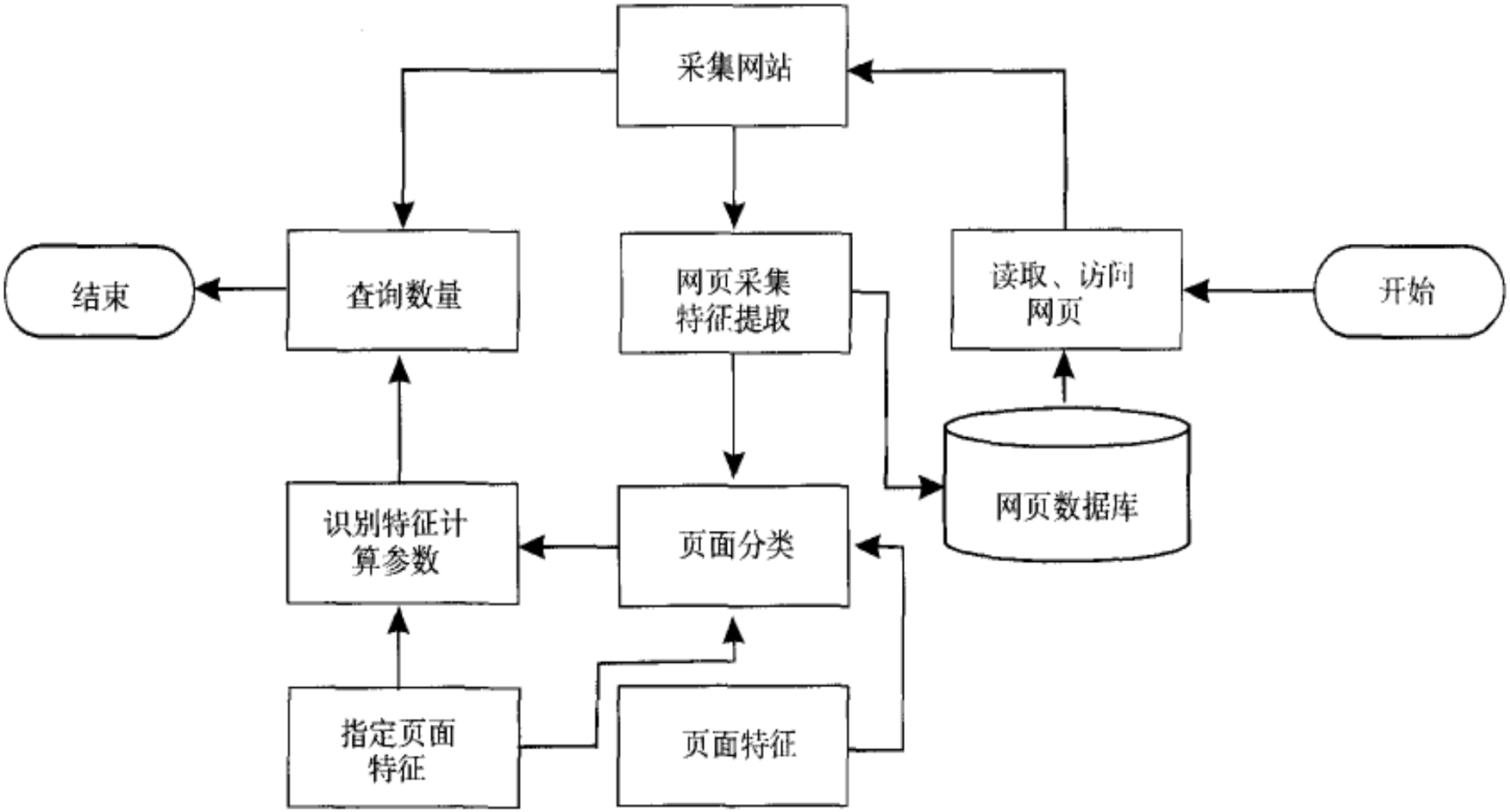
四个困难：
1. 高效算法去重（bloom filter）

2. 表单验证（校验码）（OCR）

3. 正文抽取（vips，dom特征计算和缓存）

4. Ajax如何解析(webkit)

深度爬虫架构


采集网站
结束
查询数量
网页采集
特征提取
读取、访问
网页
开始
网页数据库
识别特征计
算参数
页面分类
指定页面
特征
页面特征

垂直爬虫

四大微博采集

新浪api受限，一次只能采集500粉丝，历史微博数据只能采集最新的200条
前期被封ip，账号。用代理ip采集（代理算法）目前这个方法也快撑不住了
如何实时采集微博（通过马甲），新浪有“2亿”用户，如何能采集完，僵尸粉怎么过滤？
登入的两种方式：
1.使用自动登录的方式获得网页内容。
2.使用cookie（人工申请）的方式获得网页内容。

采集是个持久战
近期在想新方法采集

实时微博搜索

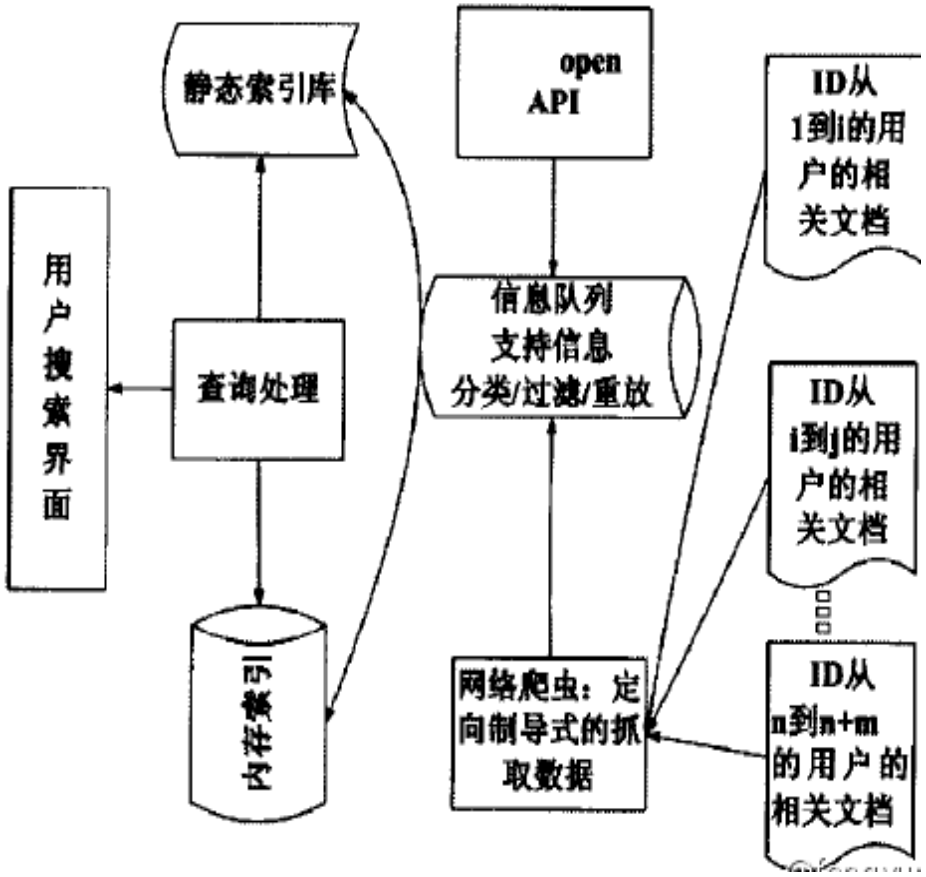
1.社交网络的用户更新的状态信息，发表微博，评论，转发等都依附于用户存在，通过用户ID不同，把网络分为不同区域。

2.信息队列负责信息的过滤，分类，排序。

3.内存索引周期更新实时性很高的数据。

4.静态索引将内存索引的信息同步入磁盘，毕竟内存容量有限。

5.用户界面提供数据的接口进行搜索与查看



疑问

1. 系统越来越庞大，分散节点越来越多

2. 不同模块，不同输入和输出，配置文件杂乱无章

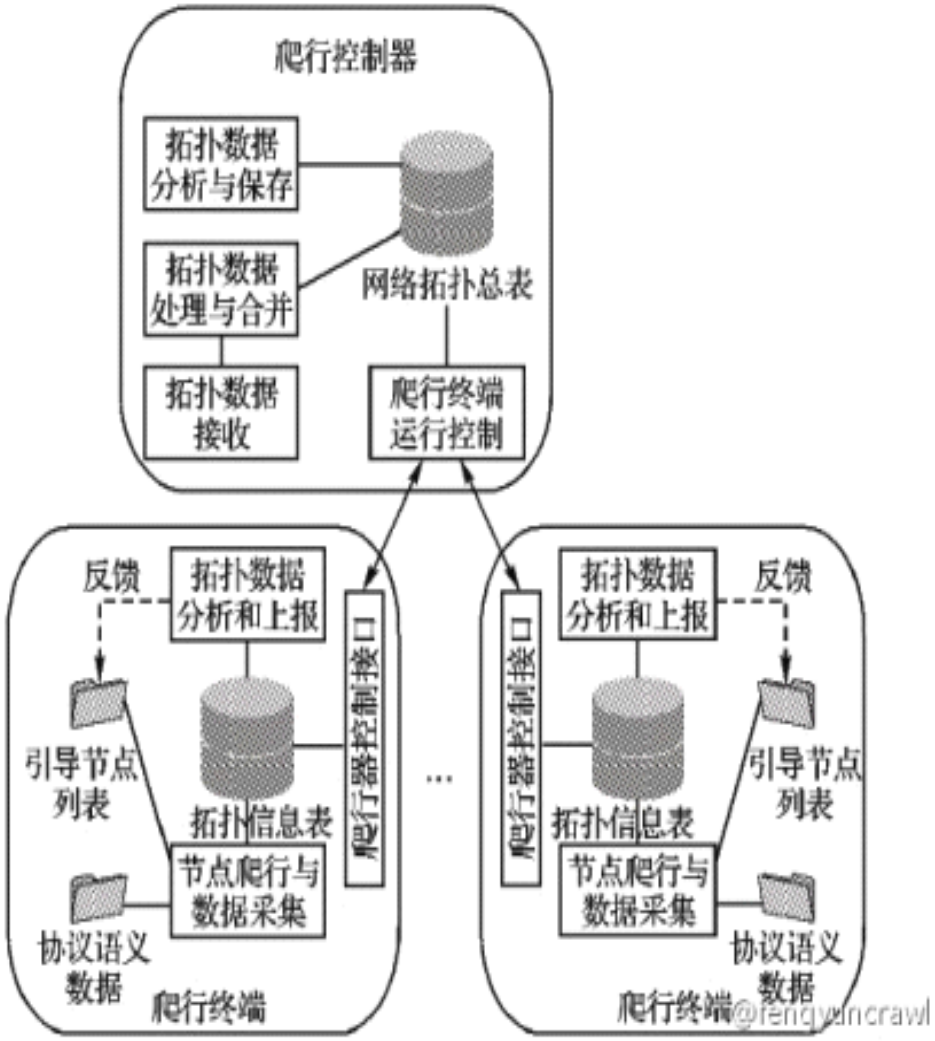
3. 重复工作很多

分布式爬虫(主从架构)

1. 主从结构，爬行控制器和终端

2. 控制器控制(master)全部爬行器同步和终止命令

3. 终端(slave)负责信息的采集，将拓扑信息反馈控制器



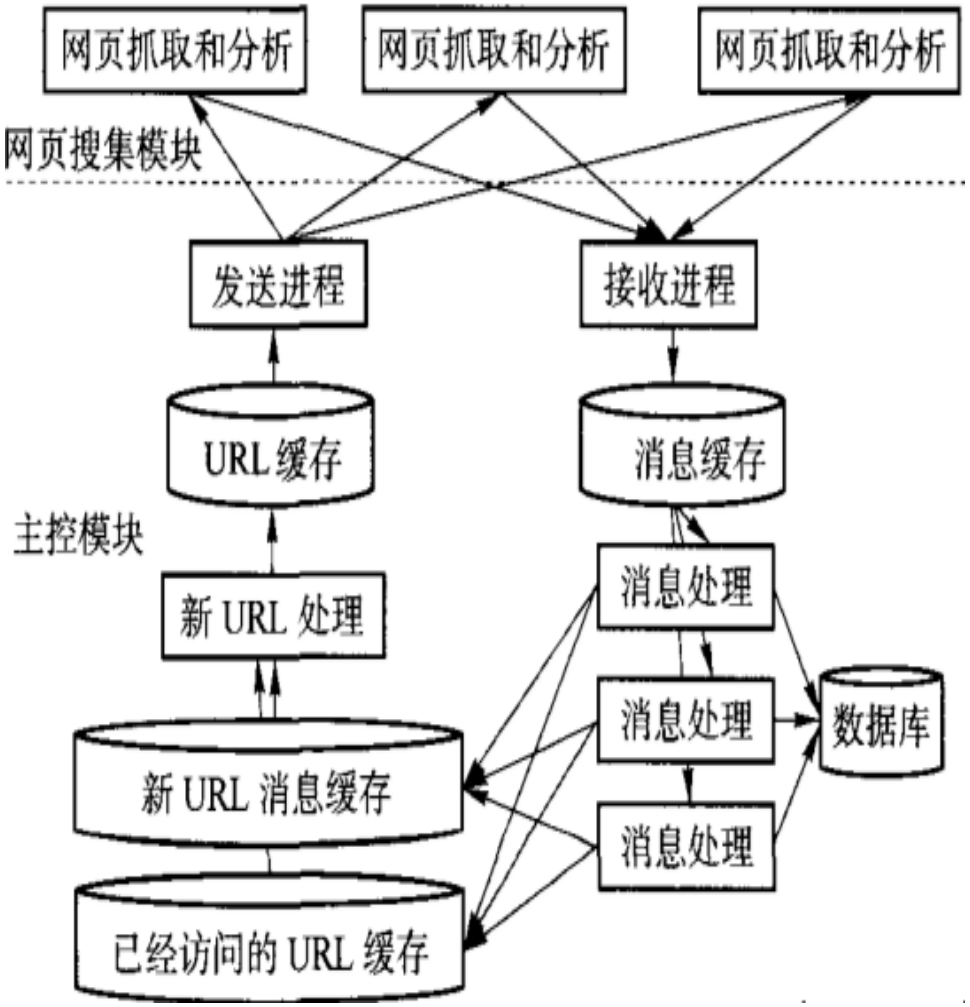
中间件

1. 主控和搜集之间使用中间件链接

2. 高性能，可扩展

3. 中间件提供api可以让系统人员尽量将注意力放应用层上，避免底层通信接口的编写繁琐工作

缺点：各抓取进程之间是独立的，master会成为系统的性能瓶颈



疑问

海量数据如何存储

海量数据存储

1. memcached+redis+mysql，分级存储

2. Hadoop分布式存储

3. mongodb集群

4. fastdb，voltdb实时内存分布式数据库+海量存储

分级存储

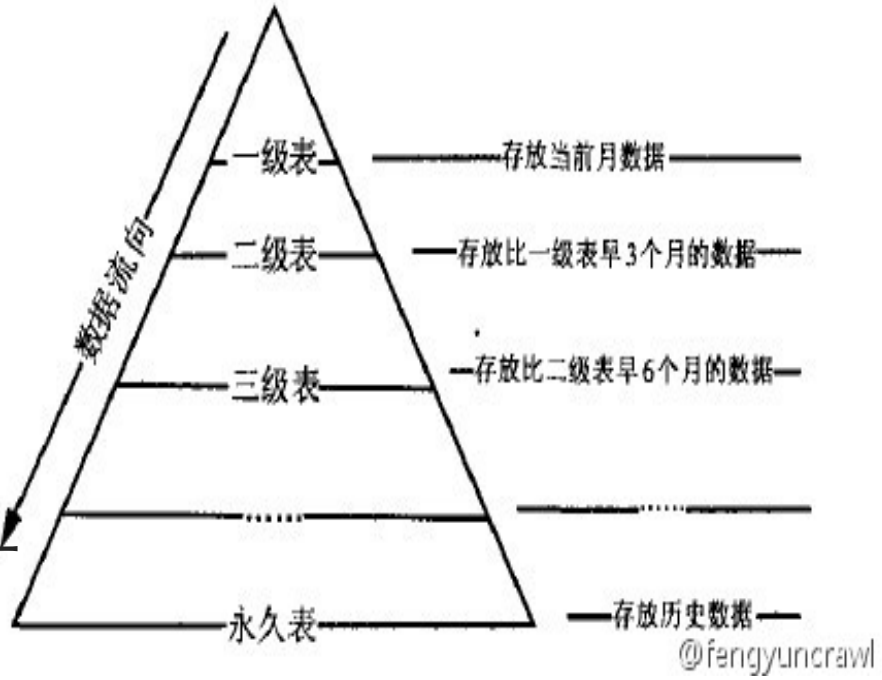
1. 根据时间维度划分数据表（冷热）

2. 存储同时也是缓存（一级，二级...）

举例：

memcached+redis+mysql

缺点：一致性不能保证，只能实现最终一致性



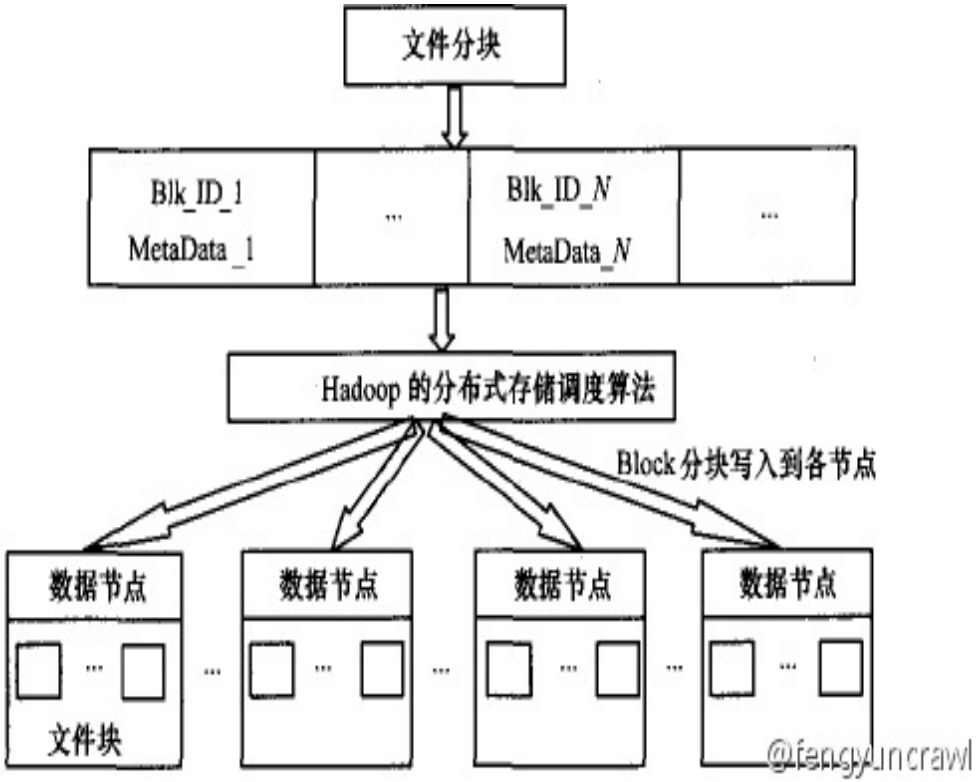
Hadoop分布式存储（大文件）

1. 适合大文件分块存储，每块64块，冗余存储

2. 分块持久化2种方法：

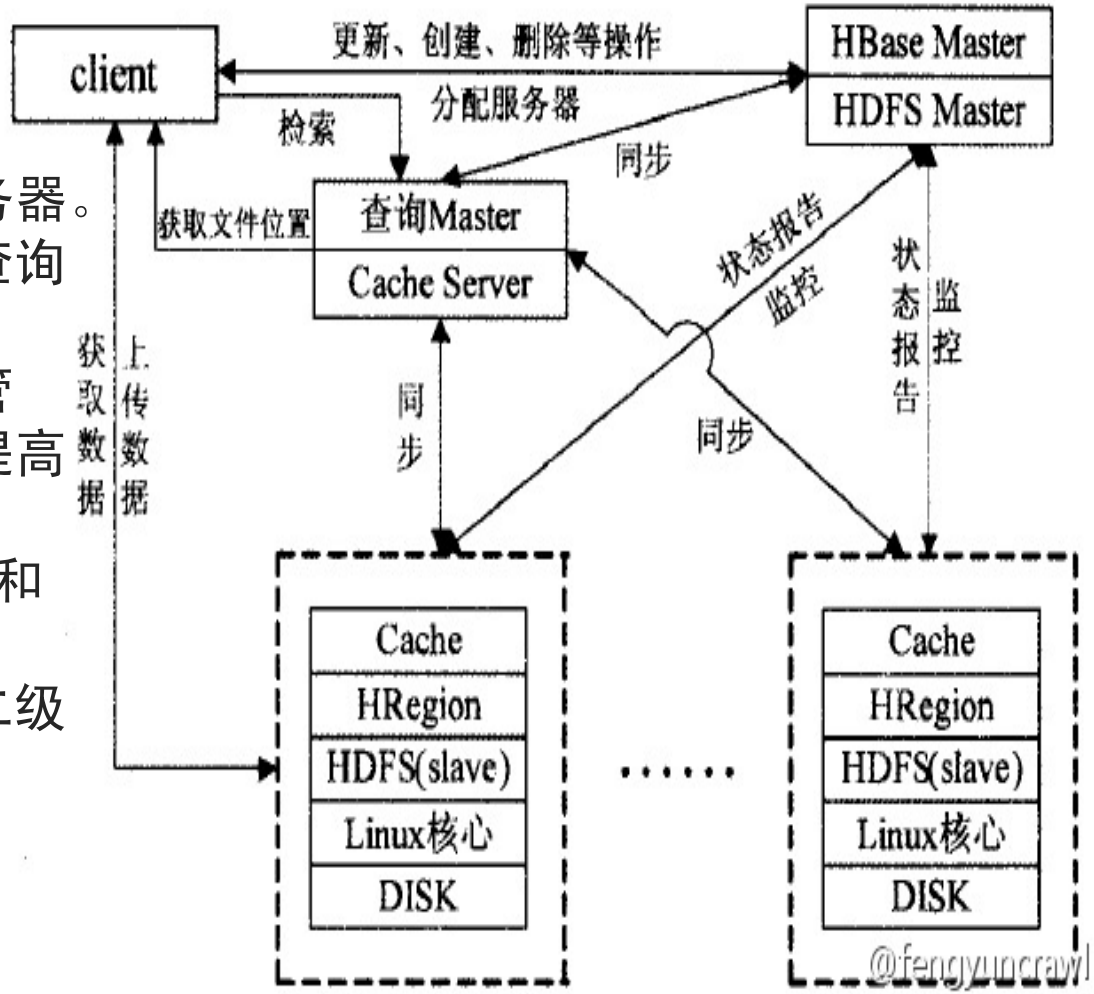
每个分块单独存储一个文件，用文件名做索引键号。

将属于同一大文件的分块存储为一个文件，利用调度，将文件分散存储在分布式节点。



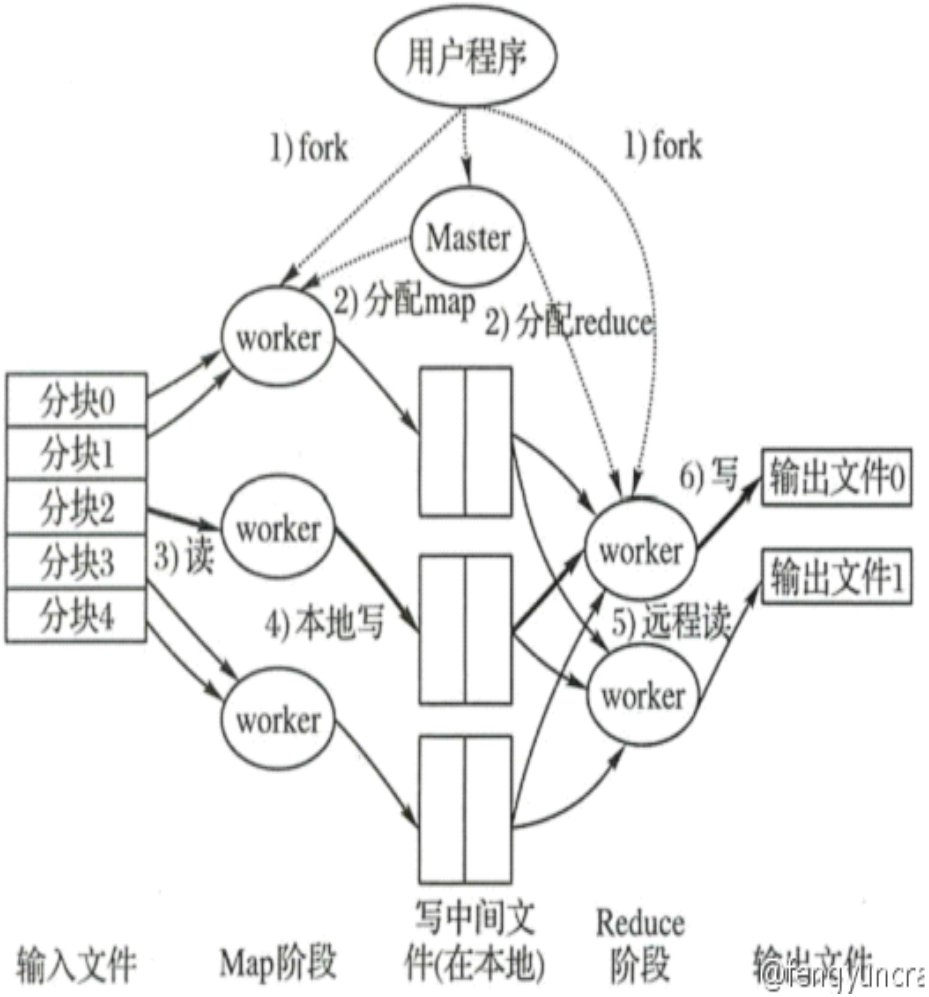
Hadoop分布式存储（大小文件混合）

1. 双master，主服务器和查询服务器。
2. 主服务器发送修改信息，同步查询服务器。
3. 查询负责文件和数据块的映射管理，提高文件的读取速率，进而提高系统对小文件的处理速度
4. 结构化和非结构化分别用Hbase和HDFS存储
5. 为了数据快速读取，采用分层二级分布式缓存memcached



MapReduce

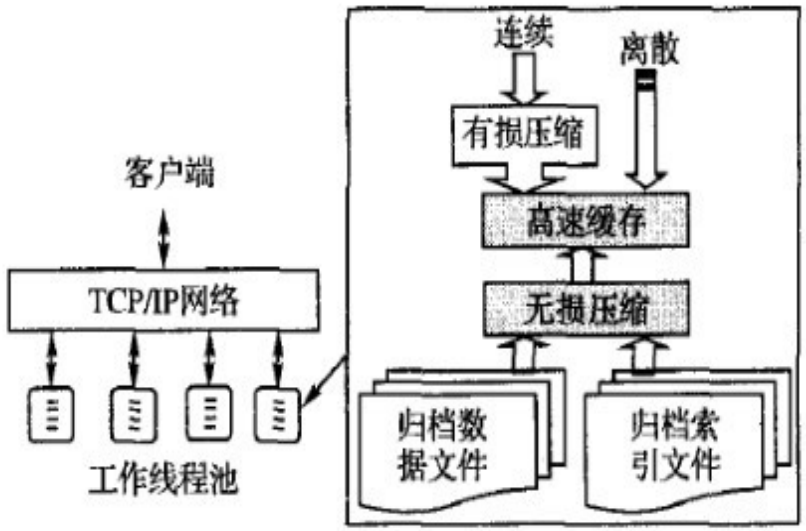
1. Map-Reduce五步骤：输入文件，将文件分配给多个worker并行地执行，写中间文件（本地写），多个Reduce workers同时运行，输出最终结果
2. 本地写中间文件减少网络带宽与时间耗费。执行Reduce时，从Master获得中间文件位置信息，Reduce使用远程过程调用，从中间文件所在节点读取所需数据。



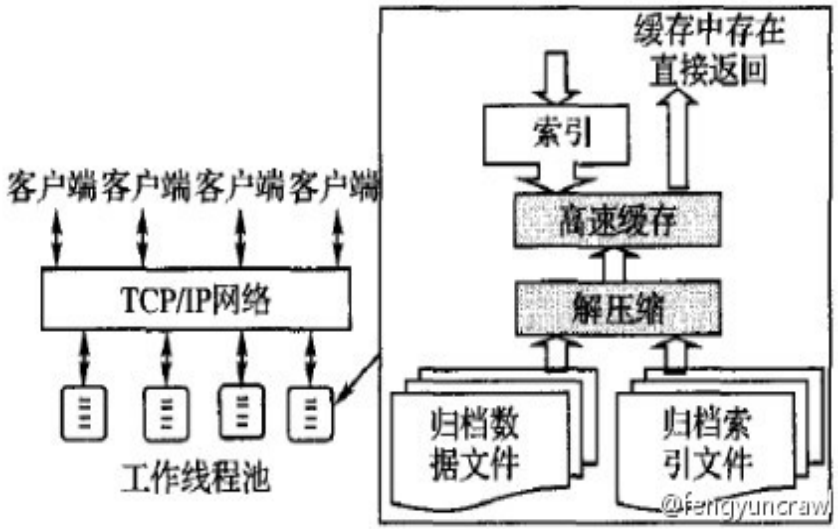
MR的缺点：
1. 很多问题难以抽象成Map和Reduce操作；
2. 实现迭代算法时效率低；
3. 执行多集的交运算时效率不高。

海量实时数据存储

实时海量数据库将数据分为两类：离散和连续数据。采用不同的数据处理方法。离散数据指变化不连续数据，采用无损压缩处理（Huffman），而连续数据指连续变化，采用有损压缩（带宽压缩），对历史数据进行有损压缩，压缩处理后再经过无损压缩最后存储在磁盘上。



1.为了高速处理，采用线程池技术实现数据写入请求，通过使用多个多核cpu并行处理来自一个tcp链接上的多个请求报文。
2.同样读取也采用线程池并行处理，采用LRU缓存算法将常用数据保持在高速缓存中。
3.采用分块内存映射文件方法，多个工作线程并发地将数据文件需要的部分进行映射

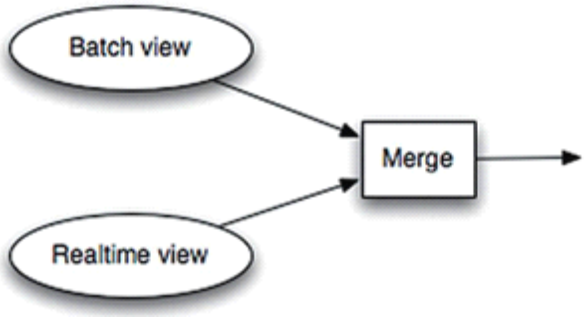


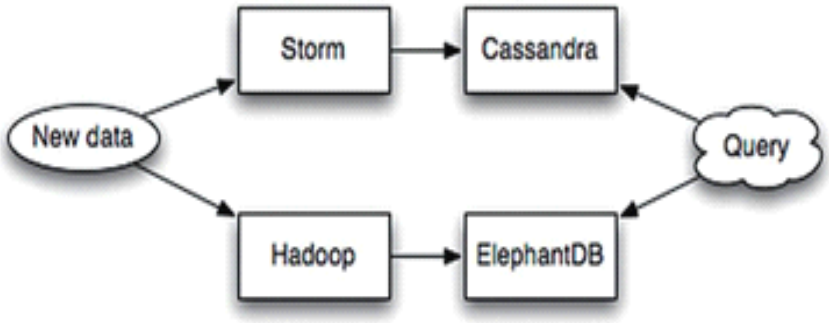
“打败”CAP原理

Twitter首席工程师发表了一篇论文
http://www.programmer.com.cn/9260/

数据CRUD-->CR

持久化层分为实时+海量批处理



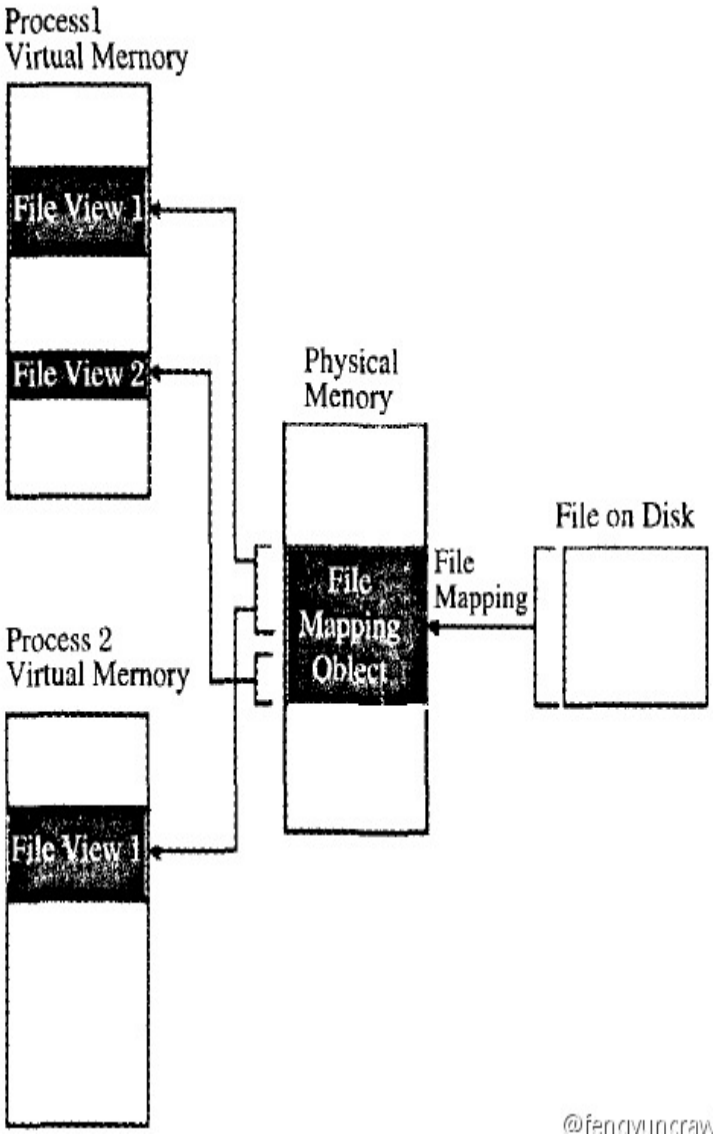


fastdb实时内存数据库

1.使用内存映射文件技术作为内存数据库文件由磁盘至内存的加载和访问

2.如果数据库的大小大于系统中实际物理内存时，操作系统会swap

3.当用户访问不在内存中，操作系统产生缺页，将需要的数据从磁盘调度到内存，并将近期不用数据调回磁盘。

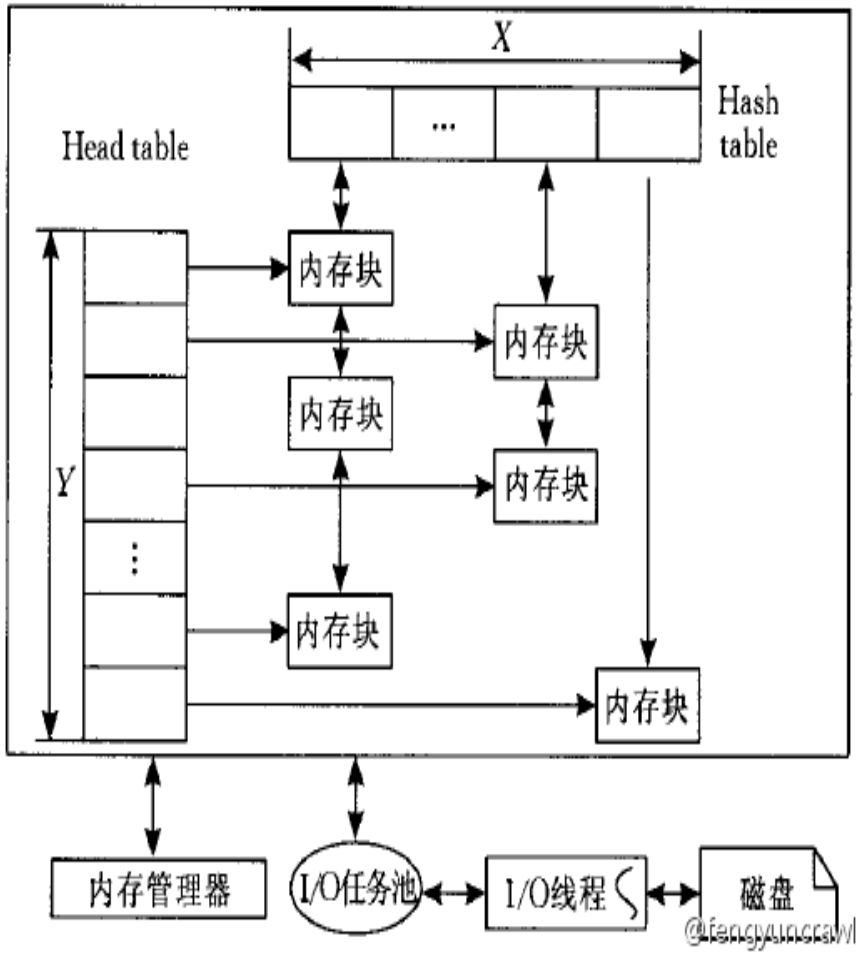


磁盘缓冲管理

1.用户线程的磁盘块读写操作请求，将首先检查该数据块在磁盘缓冲中是否存在，如果已存在则直接返回给用户线程

2.如果不在则磁盘缓冲区向I/O任务池写入一个对应的数据块读请求，在I/O线程返回有效磁盘块前，用户线程等待请求完成。

3.总之尽可能将重要数据放内存，减少对磁盘访问



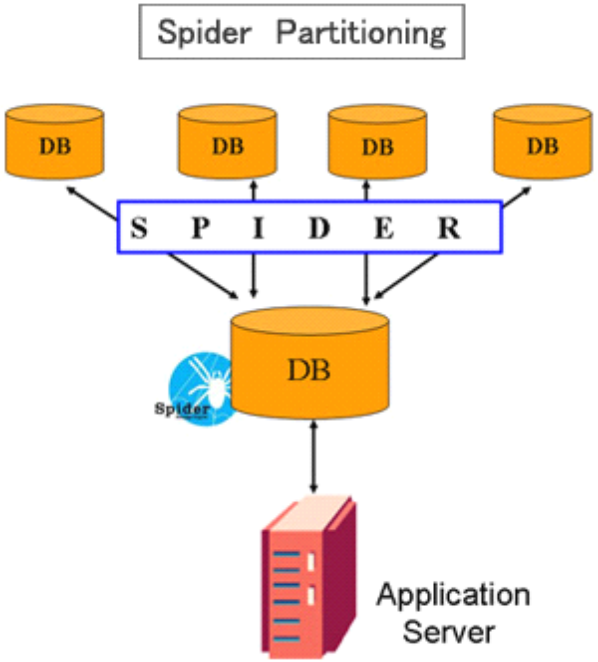
疑问

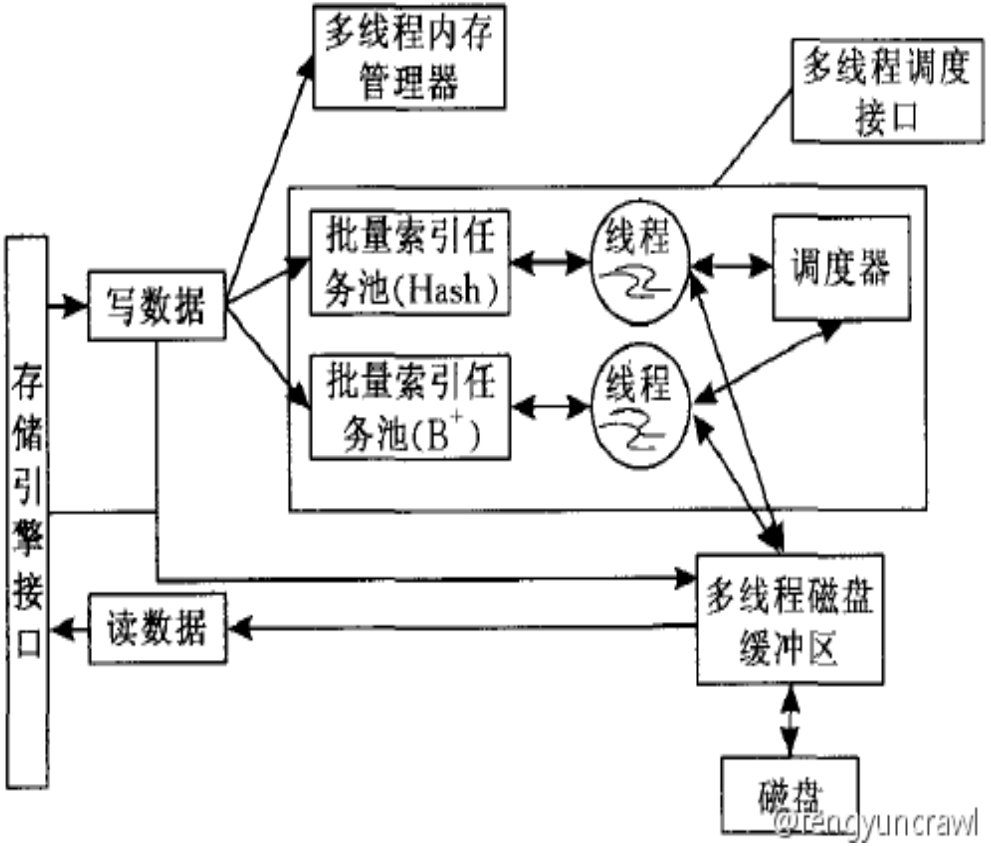
mysql storage engine（myisam，innodb）关系型数据库
保证数据一致性
能否将mysql改成并行分布式挑战Hadoop？

mysql并行分布式

mysql+spider+Distributed
kv store

http://spiderformysql.com/product.html

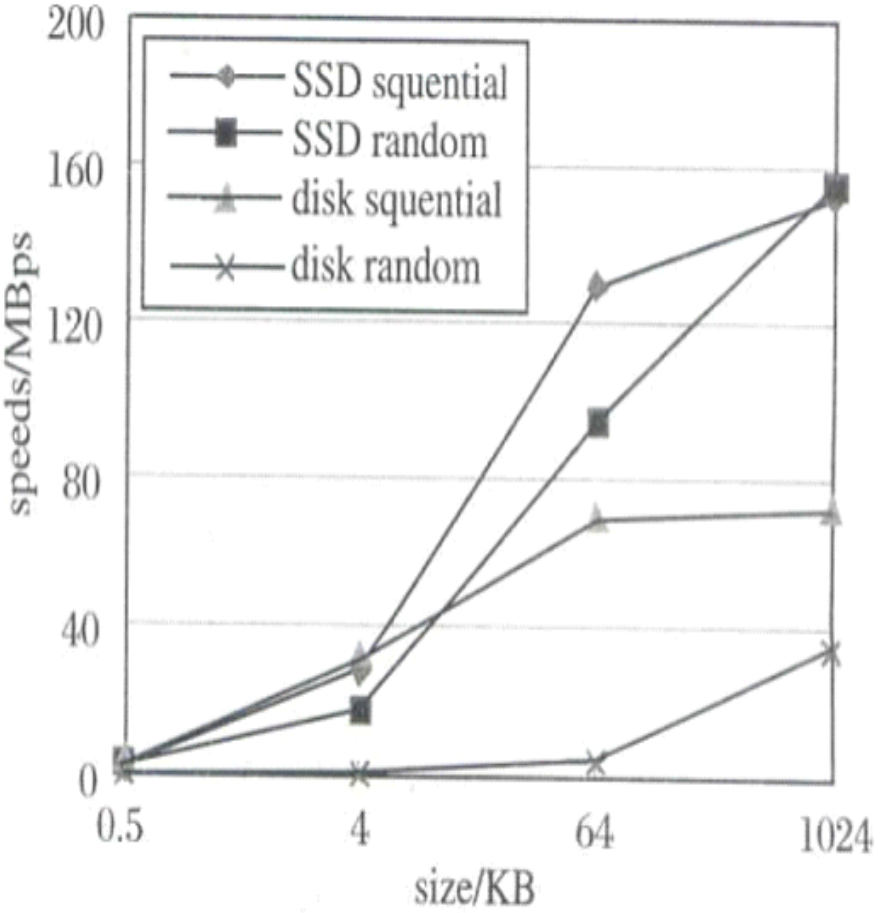




SSD vs 磁盘（读取）

磁盘为West Digital WD5000AAKS,所用SSD为ADATA 32G。测试闪存在不同块大小下的顺序读取速度和随机读取速度,随着块大小的增加无论固态硬盘还是磁盘读取速度都显著提高。固态硬盘的顺序读取速度高于磁盘。随机存取性能固态硬盘优势比较明显,要高于磁盘一至二个数量级

这是由于磁盘采用机械设备寻址缓慢。在相同块大小情况下固态硬盘顺序与随机读取速度相差不多,这是SSD的一大特点。

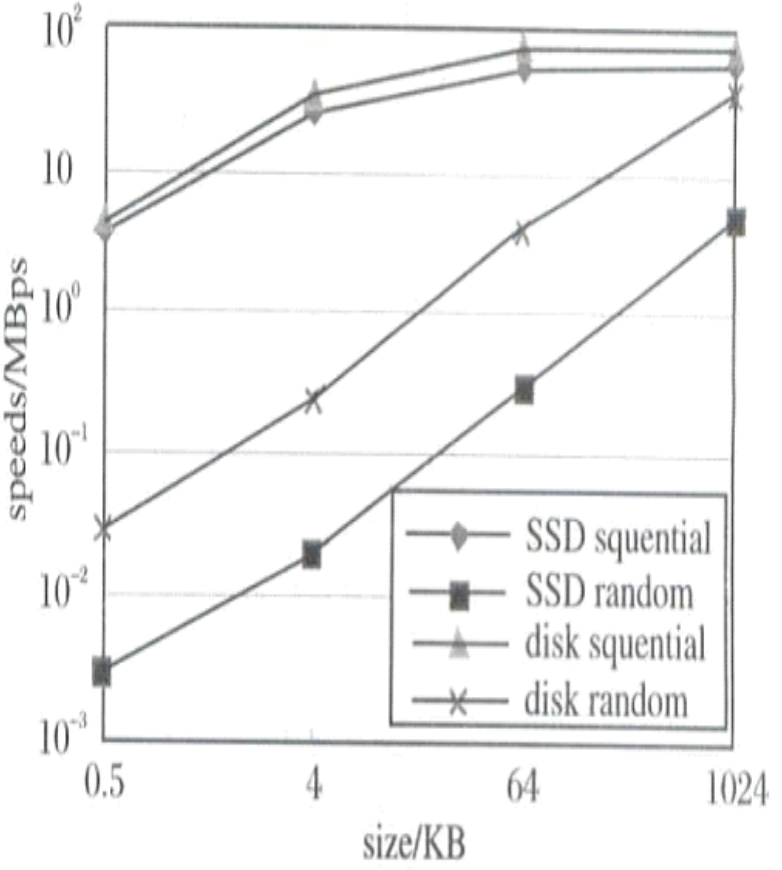


固态硬盘与磁盘的顺序与随机读取速度对比

SSD vs 磁盘（写入）

在不同块大小下固态硬盘顺序写入与随机写入的区别，由于写入测试需要事先清空所有数据,而操作系统装在磁盘上,所以无法对磁盘写入进行测试,不过磁盘读取与写入速度相近。固态硬盘的顺序写入速度要远远快于随机写入,但两者都要慢于相应的读取操作。这是由于随机写入需要更多的擦除操作造成

随机写入速度甚至要慢于磁盘相应速度一个数量级。所以在设计固态硬盘的策略时,要尽量避免随机写入操作,有时甚至不惜以增加读取操作为代价。写入速度与读取速度之间的明显差异是固态硬盘与磁盘的一个不同之处。

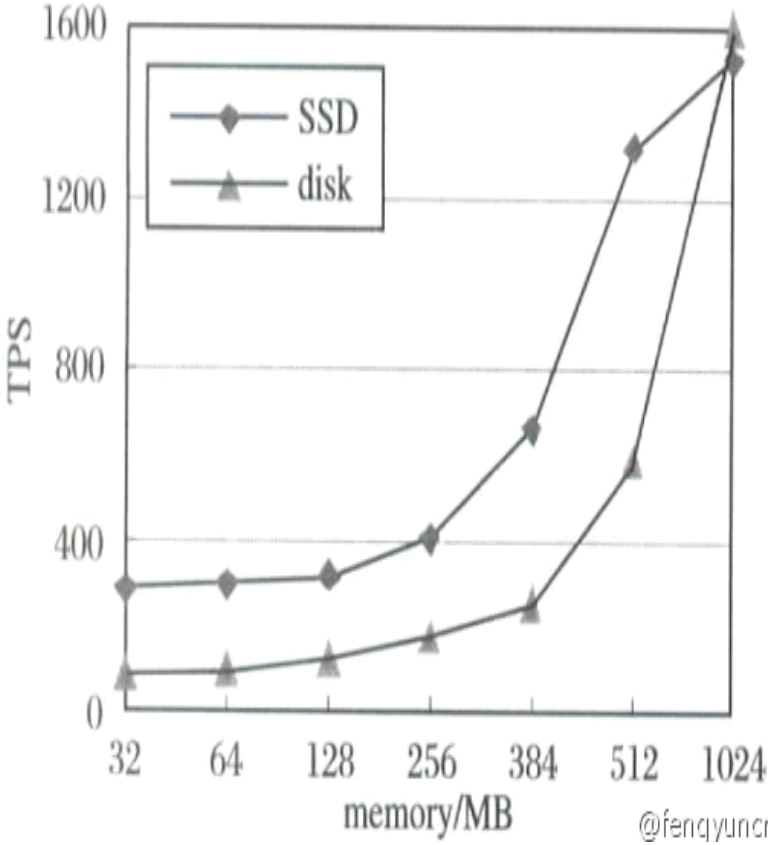


固态硬盘的顺序与随机写入速度对比

SSD vs 磁盘（测试性能）

1.随着分配的内存增加,磁盘与SSD的每秒处理的事务数量(TPS)均有所增加,但是SSD要高于磁盘,特别是当分配内存较小时,SSD比磁盘快3倍左右。

2.小粒度的写入操作如数据库中日志的记录,SSD性能甚至要低于磁盘。所以,要提高ssd的性能,改进随机写入操作是关键。



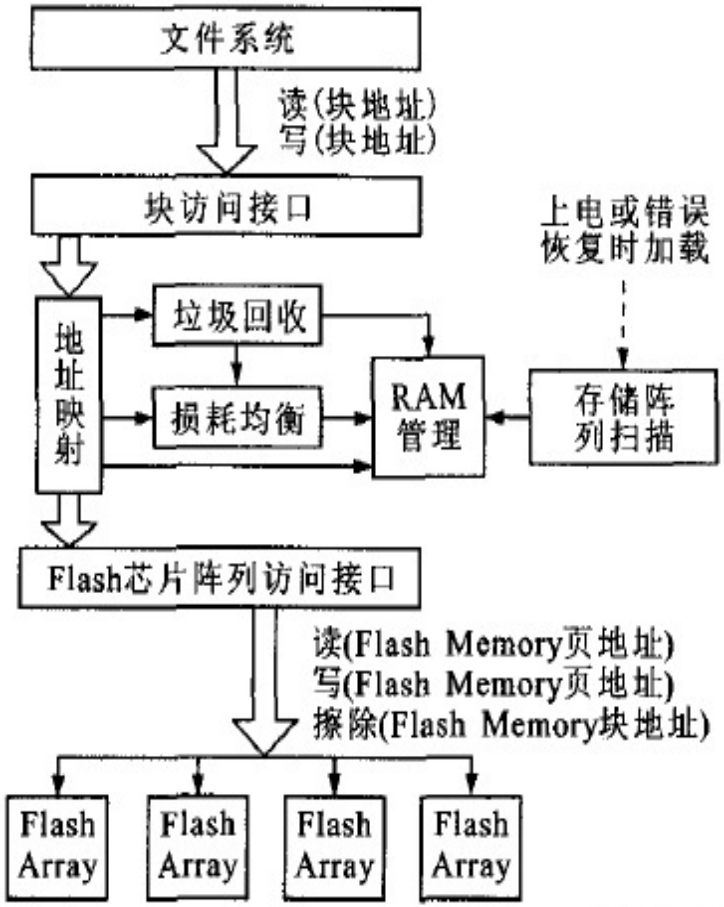
SSD固态硬盘

页地址映射

1.RAM上存储页映射表

2.Flash上存储各单元的数据信息。

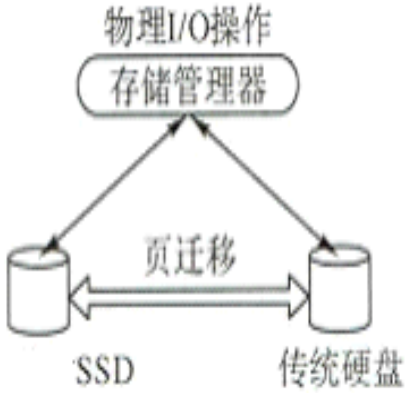
3.SSD非定点更新和写之前擦除整个块，导致写操作和磁盘不同，当FTL层接到写命令后，直接将数据写到空闲块中，之后写命令通过逻辑地址查找地址映射表找到物理地址，最后写入这一页。垃圾回收将块中的有效数据迁移擦除放入空闲区

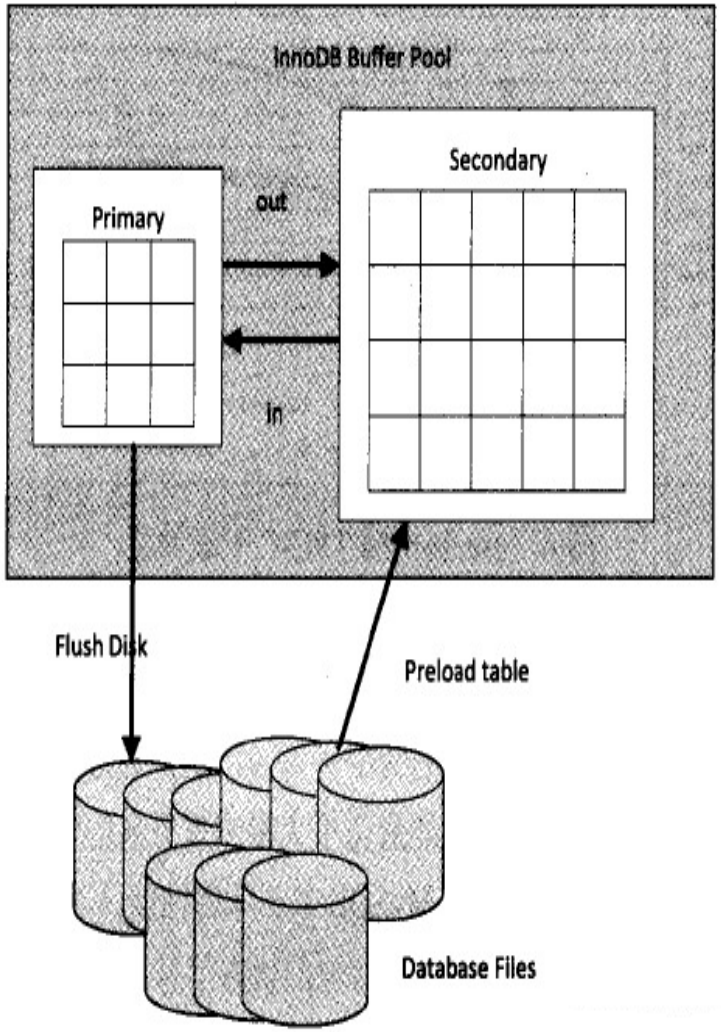


SSD+Innodb（磁盘）

1.应用一，顺序I/O放传统磁盘，随机I/O放SSD，随机I/O文件包括(*.ibd)文件，ibdata文件，顺序I/O包括redo log ，binary， slow query logs，error logs。

2.应用二，SSD作为InnoDB的二级缓存,配置参数innodb_secondary_buffer_pool。当缓存页移出InnoDB的缓冲池时，InnoDB会将其移动到SSD上



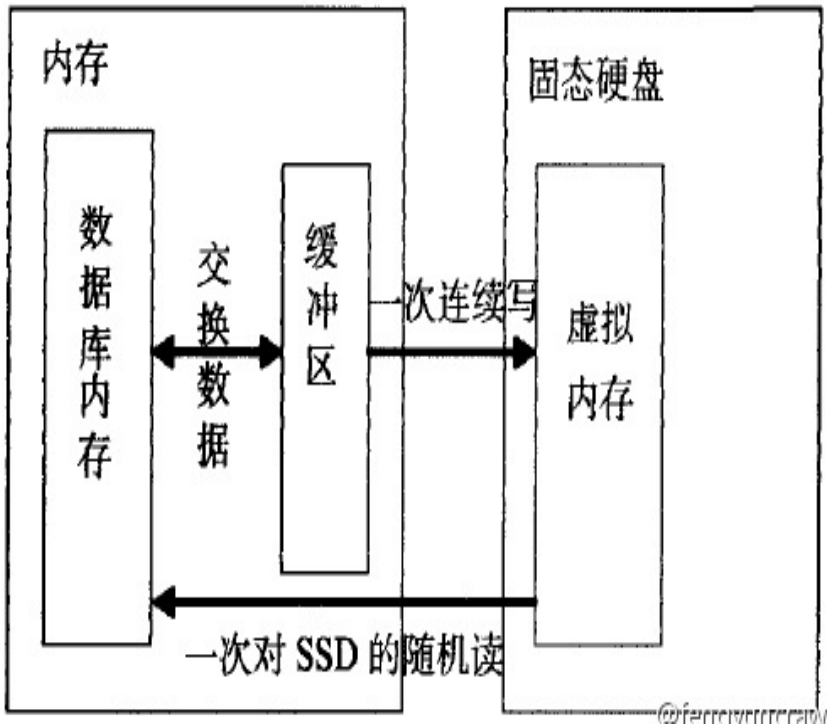


SSD作为虚拟内存

Redis+SSD存储

用SSD为块单位作为写缓冲区，将数据库大量对虚拟内存随机写作为主存缓存起来，写满时，将缓冲数据一同写入虚拟内存中，这样将大量随机写转化为一次连续写，利用SSD高速的随机读同时，将多个随机写数据缓存，合并，转化为一个连续的写入操作，提高数据库对虚拟内存写操作的速度

缺点是虽然向SSD按块append写，而且追加方式不存在数据块的擦除和重写，但是虚拟内存会爆增。导致系统经常swap，和内存交换数据频繁，系统性能下降



SSD文件系统

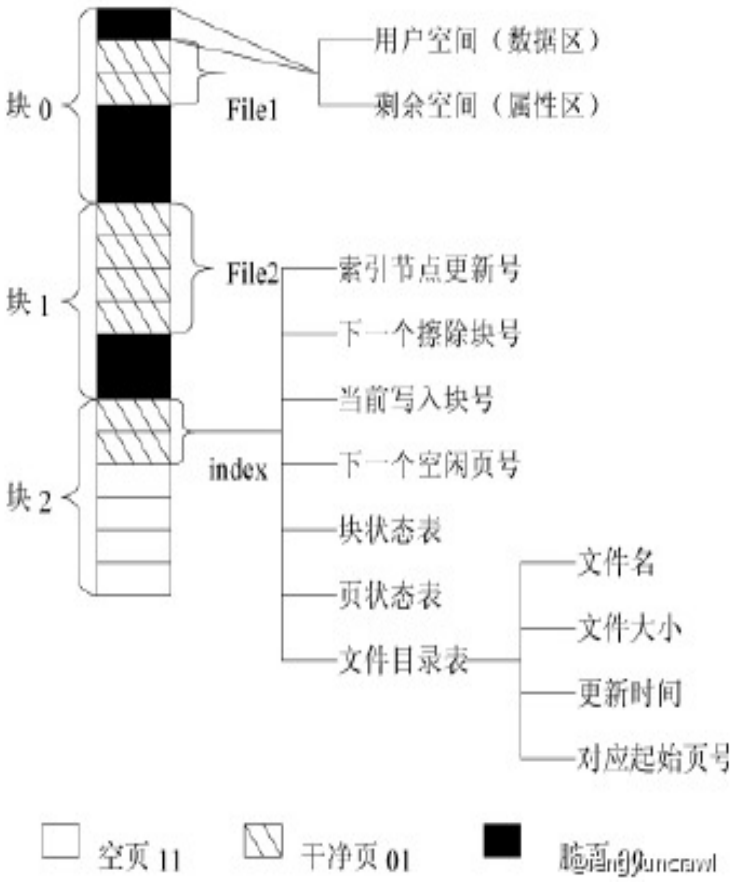
1.存储分为块+页。

2.每页由用户空间和剩余空间。

3.页的用户空间存文件数据，剩余空间放页状态(空，好，脏)和页类别（是否有索引）

4.索引存索引更新号，下一个擦除块，空闲页号，块状态表(好，坏)，页状态表，文件目录表。

5.文件建立内存映射，文件append写入，将擦除块脏页数据写到空页再垃圾回收



结束

谢谢观赏！