# 前言

这部分文档主要是在进行大数据开发过程中，由于主要目的是想实现对于文件可以随机写、随机读取实现，但是Hadoop本身并不支持这样的操作，只是支持append操作HDFS applications need a write-once-read-many access model for files. A file once created, written, and closed need not be changed except for **appends and truncates**. Appending the content to the end of the files is supported **but cannot be updated at arbitrary point**. This assumption **simplifies data coherency issues** and **enables high throughput data access**. A MapReduce application or a web crawler application fits perfectly with this model.

其实HBase/LevelDB等系统，背后都应用了Log-structured merge tree。HBase的写操作有两个步骤：1) 写入HLog(防止宕机丢失数据), 2) 写入对应的Memstore(内存)。当Memstore写满时，其中的数据被flush到HDFS，形成一个新的数据文件(HFile)。简单的讲，HBase的update/delete操作并不影响先前写入的字段，update只是对同一个key增加了一条记录，而delete只是将某个key标记为删除。由于HBase的数据文件是按key/column/timestamp有序的，所以读取时只要按照某种策略，可能只需要读取少数的数据文件(HFile)就能读取到对应key的最新状态。而系统提供了major comaction操作，该操作用于将对应列族的所有HFile合并为一个HFile，并删除TTL过期的数据，多余的版本，以及删除的数据。总之，LSM在某种程度上将随机写转化为顺序写，这也是HBase写性能及其优秀的原因。

## HBase Compaction

参考

<http://blog.cloudera.com/blog/2013/12/what-are-hbase-compactions/>

Compactions choose some files from a single store in a region and combine them.

# 第一章 简介

## 1.1 海量数据的黎明

### 1.1.1 列式存储数据库

简单来说列式存储数据库就是将传统的行式数据库中的数据以列为关键字来进行存储。如图1-1所示：

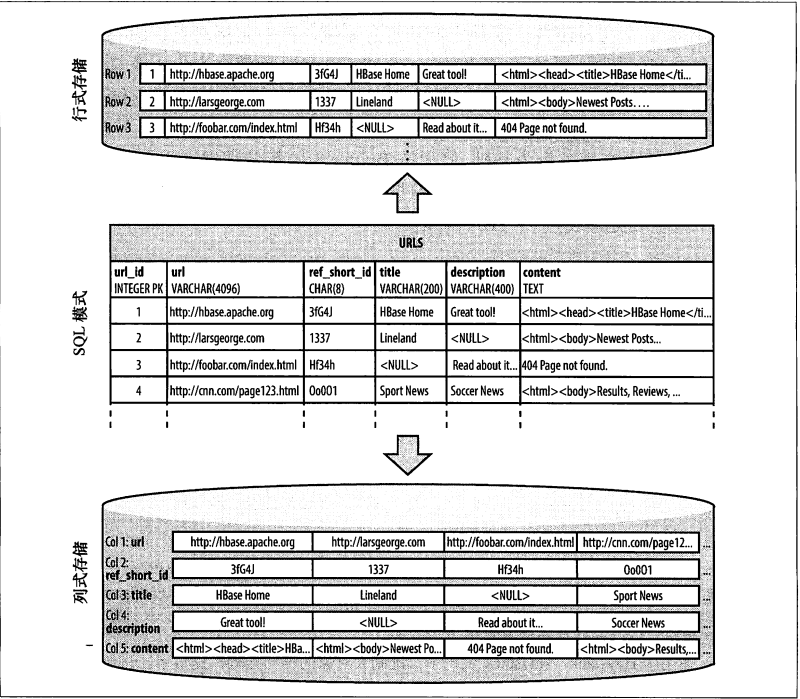


图 1-1 列式存储结构与行式存储结构

通过以上的介绍，通过表1-1介绍俩者存储的优缺点对比

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

优点：

1. 对于特定的查询，不是所有的值都是必须的，因此可以通过只检索对应的列来找到特定的数据，从而避免遍历整张表来降低I/O。
2. 可以明显的看出，对于相同的列来说，数据一般是

# 第二章 安装

# 第三章 客户端API：基础知识

# 第四章 客户端API：高级特性

# 第五章 客户端API：管理功能

# 第六章 可用客户端

# 第七章 与Mapreduce集成

# 第八章 架构

# 第九章 高级用法

# 第十章 集群监控

# 第十一章 性能优化

# 第十二章 集群管理