

Hadoop在网盘和在线备份的应用与挑战

微博：@卢亿雷

Carbonite China(凯备份)技术总监兼高级架构师

johnlya@163.com

2012-12-1

- 互联网存储应用的特点
- 网盘与在线备份的特点
- 分布式存储平台简介
- 总体实现方案
- 分布式数据库分析
- 分布式数据库特点
- 分布式文件系统分析
- 小结

互联网存储应用的特点

- 高可靠：数据多份存储
- 高可用：7X24小时服务
- 高扩展：透明升级扩容
- 高性能：高并发，低延迟
- 高安全：严格IDC管理制度和分布式密钥管理
- 高性价比：成本价格可控制
- 易监控维护：响应及时

网盘与在线备份的特点

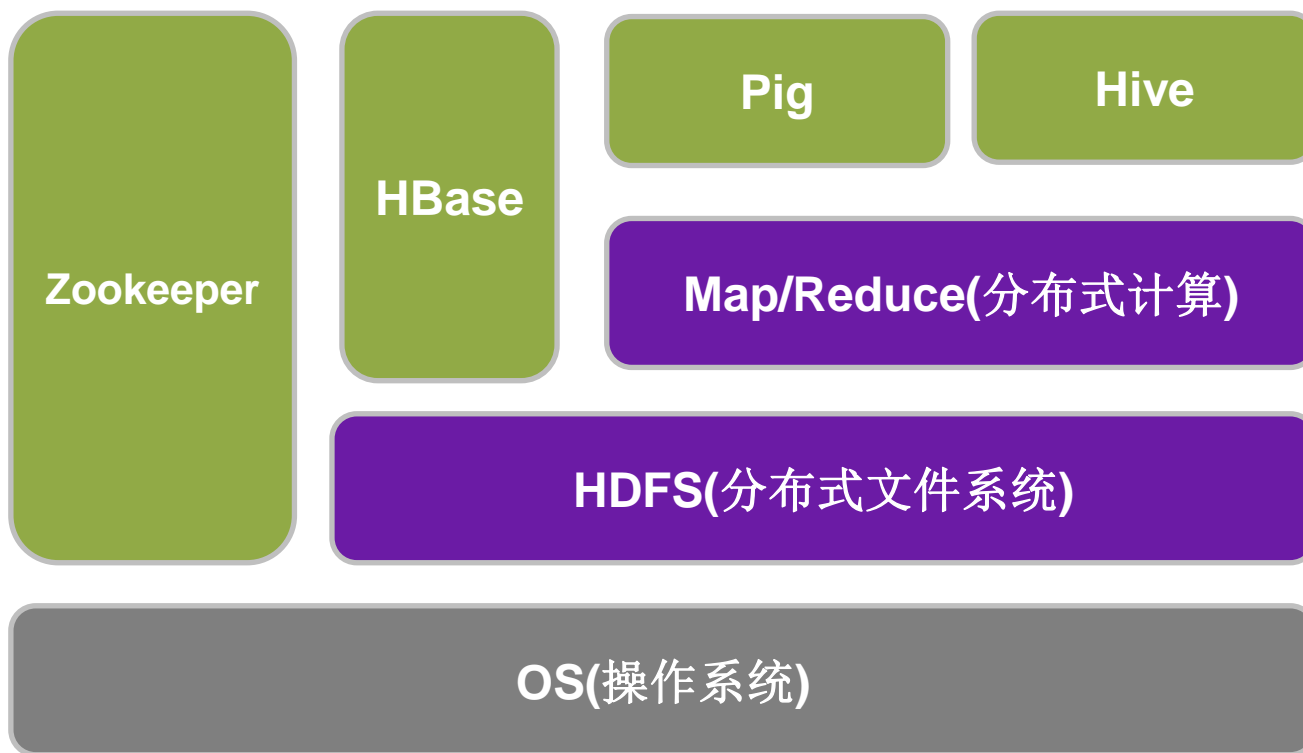
➤ 网盘与在线备份有很多相同点

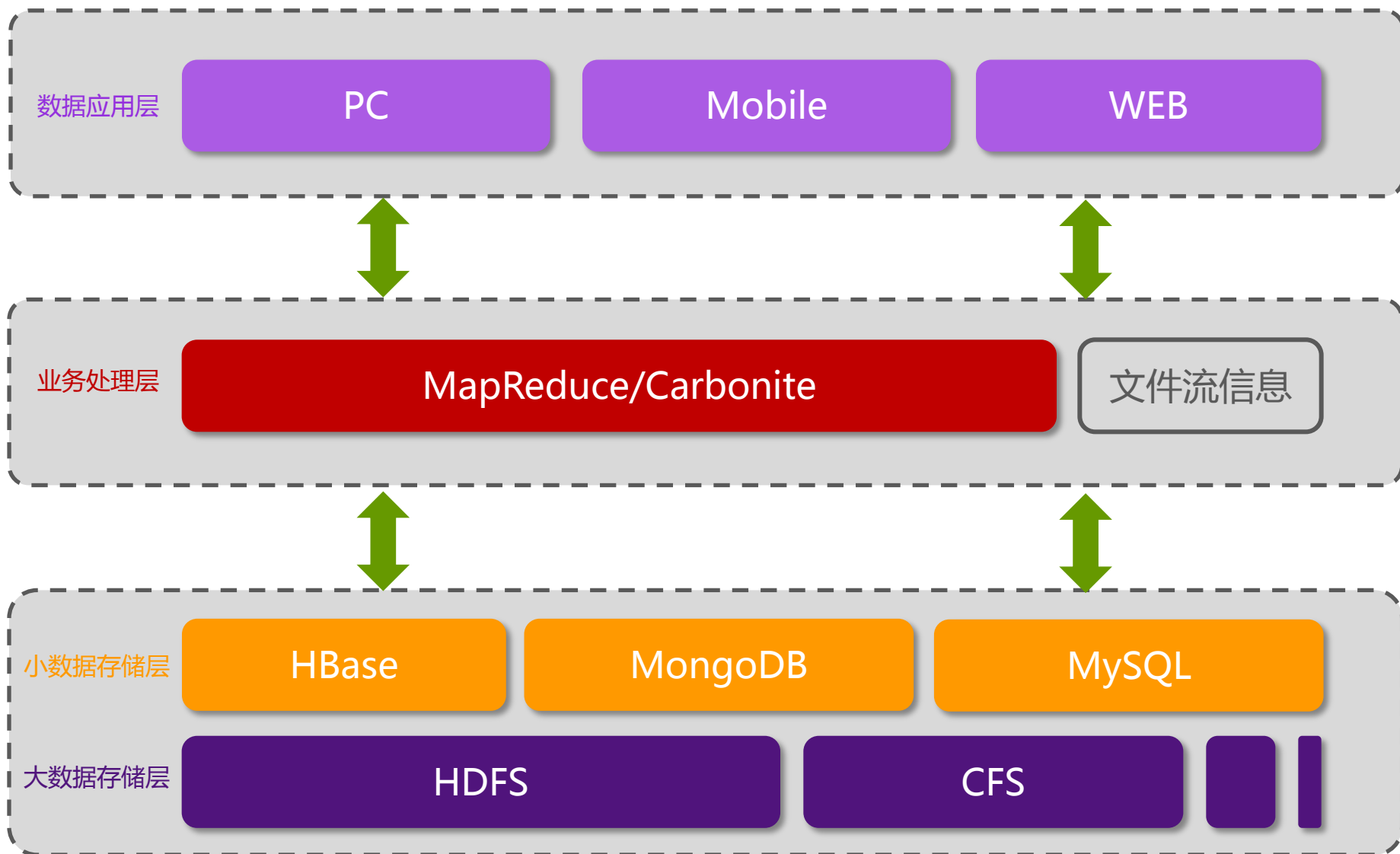
- 后台架构类似
- 访问方式相似（上传及下载）
- 都是云存储服务
- 目标都是解决数据存储问题等

➤ 在线备份相对网盘（同步盘）的特殊性

- 是否保持用户的原始路径
- 是否有设备的区分
- 是否可以备份企业级数据库
- 是否密钥用户可以自己保存
- 元数据的存储方式不一样

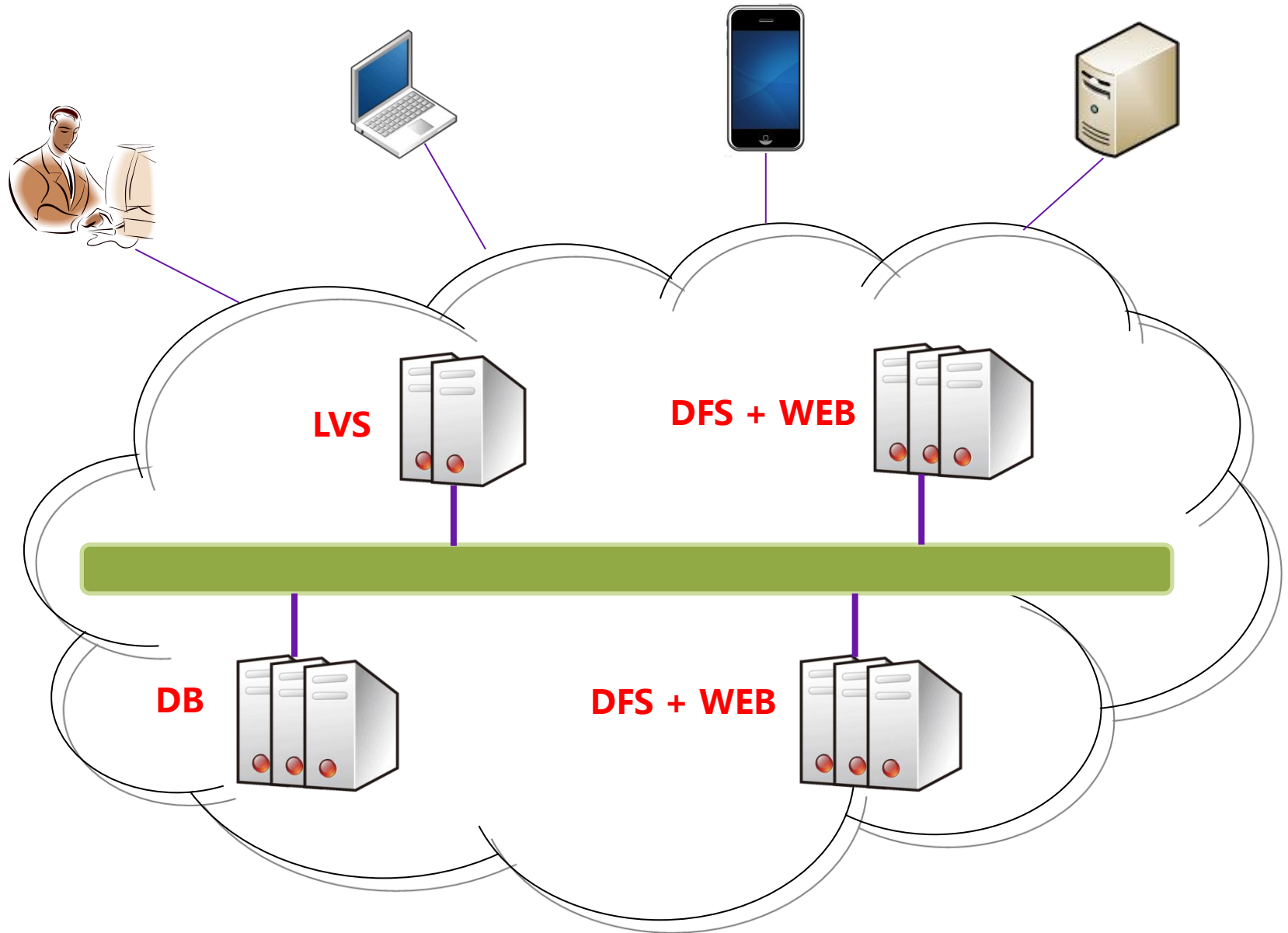
分布式存储平台简介





- 总共有1000亿个文件
- 每天增加近1个亿新文件
- 总共服务器台数约有1万台
- 总存储量约为200P

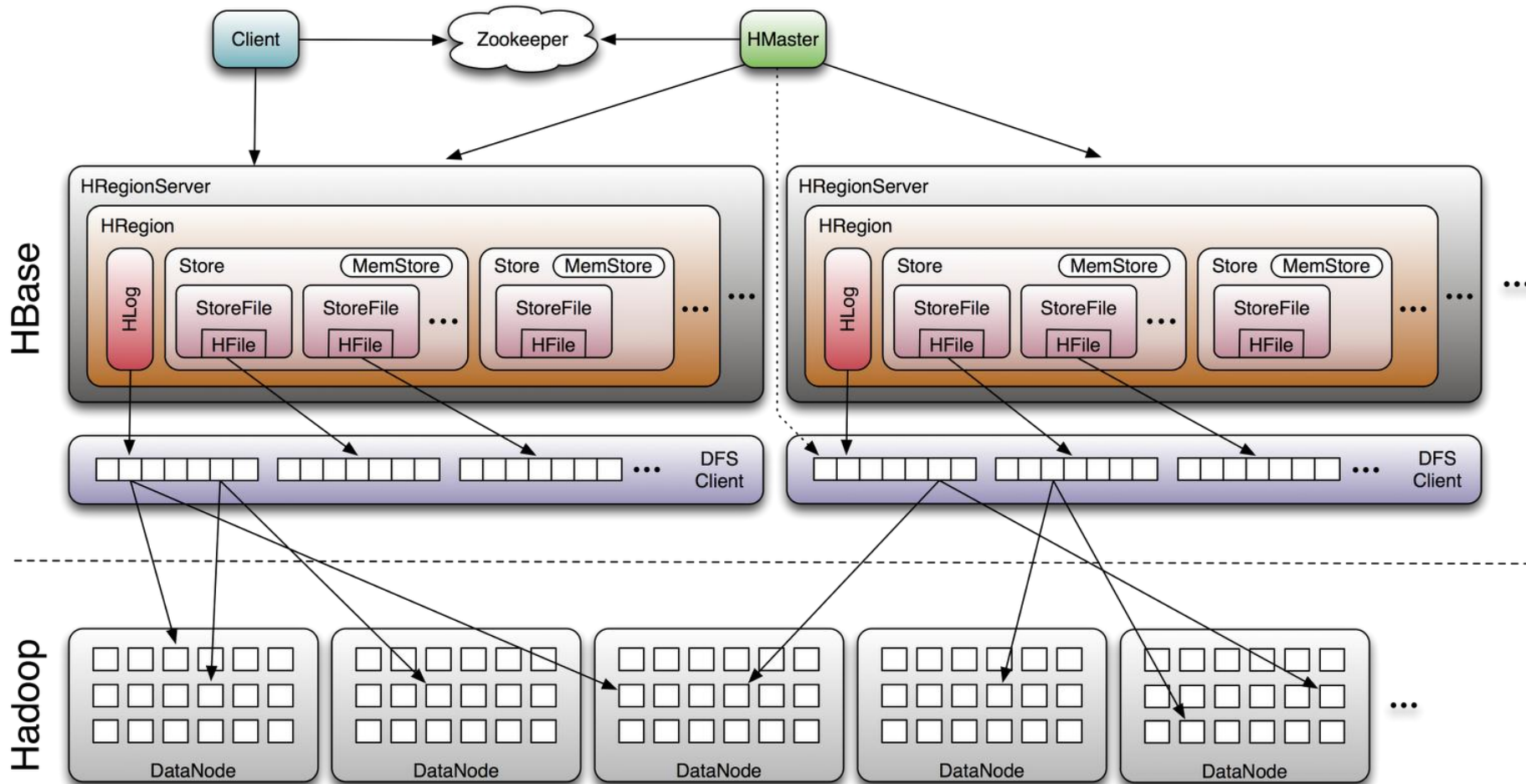
总体实现方案

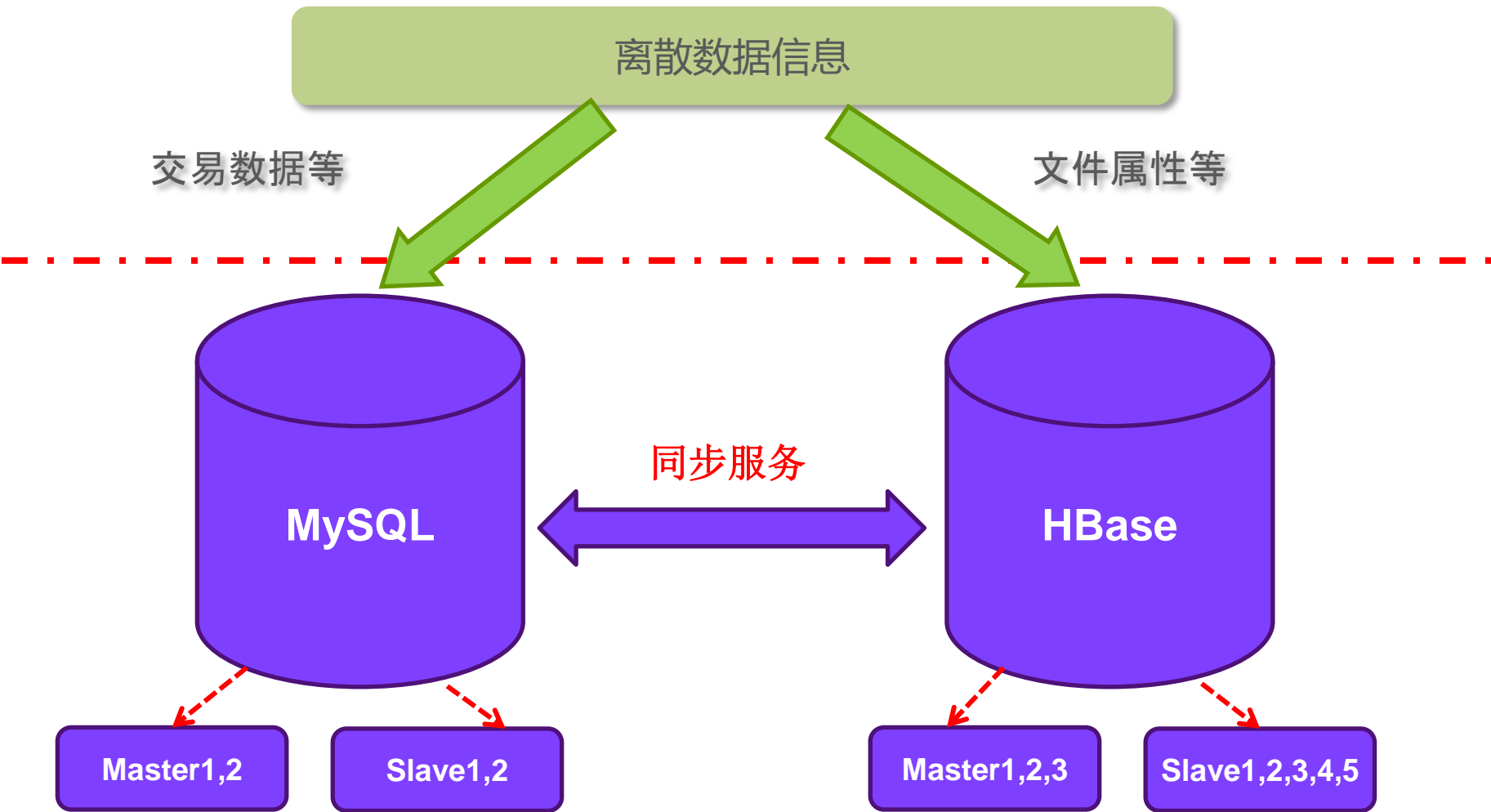


- 分布式数据库：关系数据库与非关系型数据库 (NoSql) 集群解决结构化数据的海量存储和高效访问
 - MySQL
 - HBase
- 分布式文件系统：以文件为存储单位的非结构化数据存储
 - HDFS
 - MongoDB
- 前端应用负载均衡
 - LVS
 - Ngnix/Jetty

分布式数据库分析

HBase和Mysql





分布式数据库特点

➤ 成熟度比较高，目前已有多个著名公司在使用

- Taobao
- Facebook
- Adobe
- Twitter
- Yahoo!
- Trend Micro
- 广告分析公司

➤ 行强一致性

- 同一行数据的读写只在一台RS (RegionServer) 机器上进行
- 同一行的列的写入是原子操作

➤ 水平自动伸缩

- Region的自动分裂(生产系统需要看具体情况)
- Master的自动均衡
- 增加RegionServer机器即增加读写吞吐量及处理能力
- 增加DataNode机器可增加容量

- 任意增加列
- 高性能随机写
- 支持Thrift框架

➤ 合理设计RowKey 和 Pre-Sharding

- 尽量避免只操作少数几台机器；
- 根据数据量、RegionServer个数合理Pre-Sharding。

➤ 充分利用Filter功能

- SingleColumnValueFilter
- **SubstringComparator**
- BinaryPrefixComparator
- FamilyFilter
- QualifierFilter
- ColumnPrefixFilter
- ColumnPaginationFilter

➤ 可根据应用需求重写某些方法

- **SubstringComparator**

@Override

```
public int compareTo(byte[] value) {  
    String laststr = Bytes.toString(value).toLowerCase();  
    return laststr.contains(substr) ? 0 : 1;  
}
```

➤ 考虑容量开启压缩

- 目前主要是lzo方式

➤ 提高随机读性能

- 前端增加一个分布式缓存Redis系统

➤ 系统参数优化

- GC策略：`-XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:CMSInitiatingOccupancyFraction=70`
- 读写策略优化

➤ 系统参数优化

- 读优化：

- `hbase.regionserver.handler.count`
- `hbase.regionserver.global.memstore.upperLimit/lowerLimit`
- `hbase.hregion.memstore.block.multiplier`
- `hbase.hstore.blockingStoreFiles`
- `hbase.hregion.max.filesize`

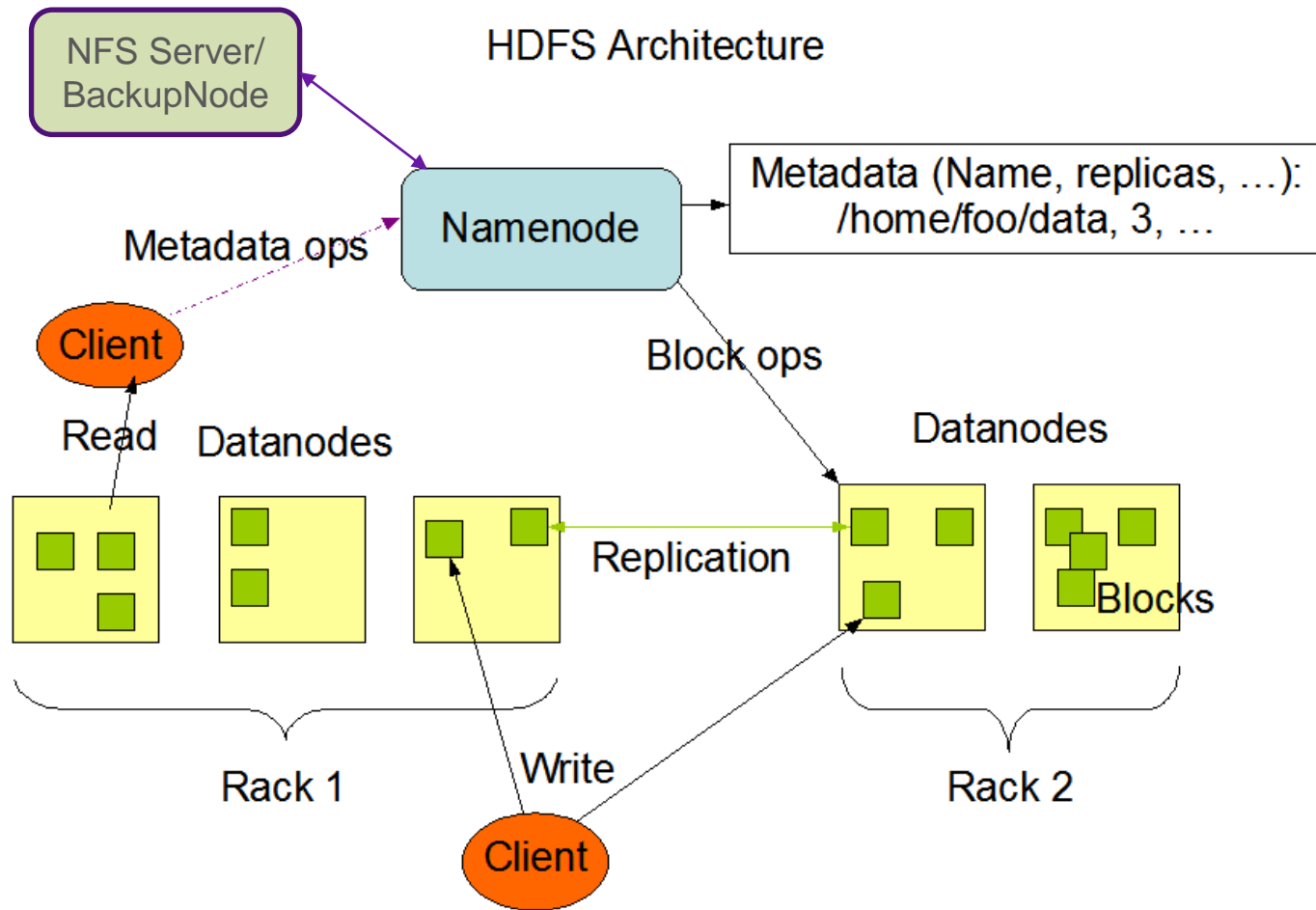
- 写优化：

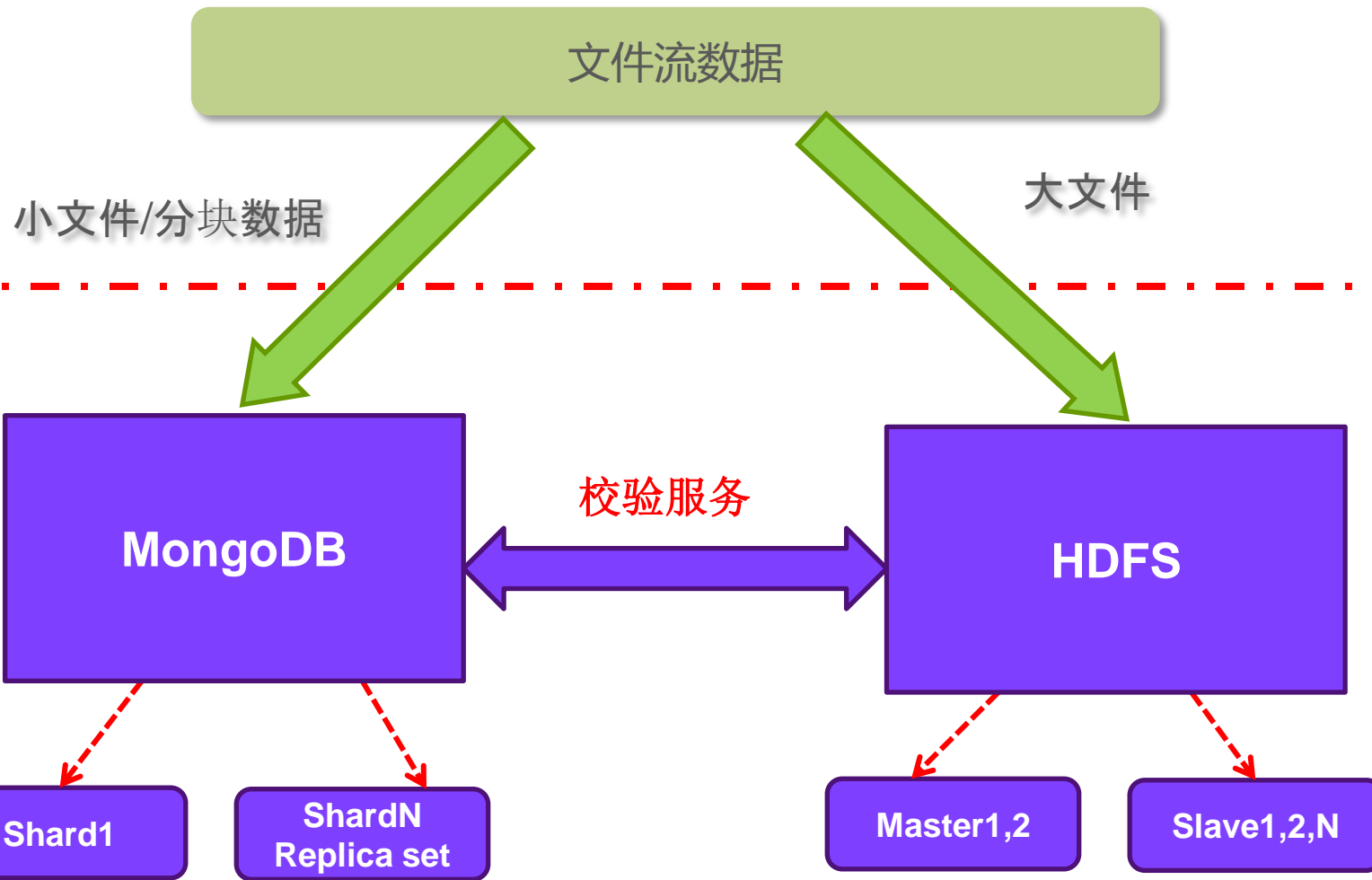
- Bloomfilter
- in-memory
- Blockcache
- `hfile.block.cache.size`

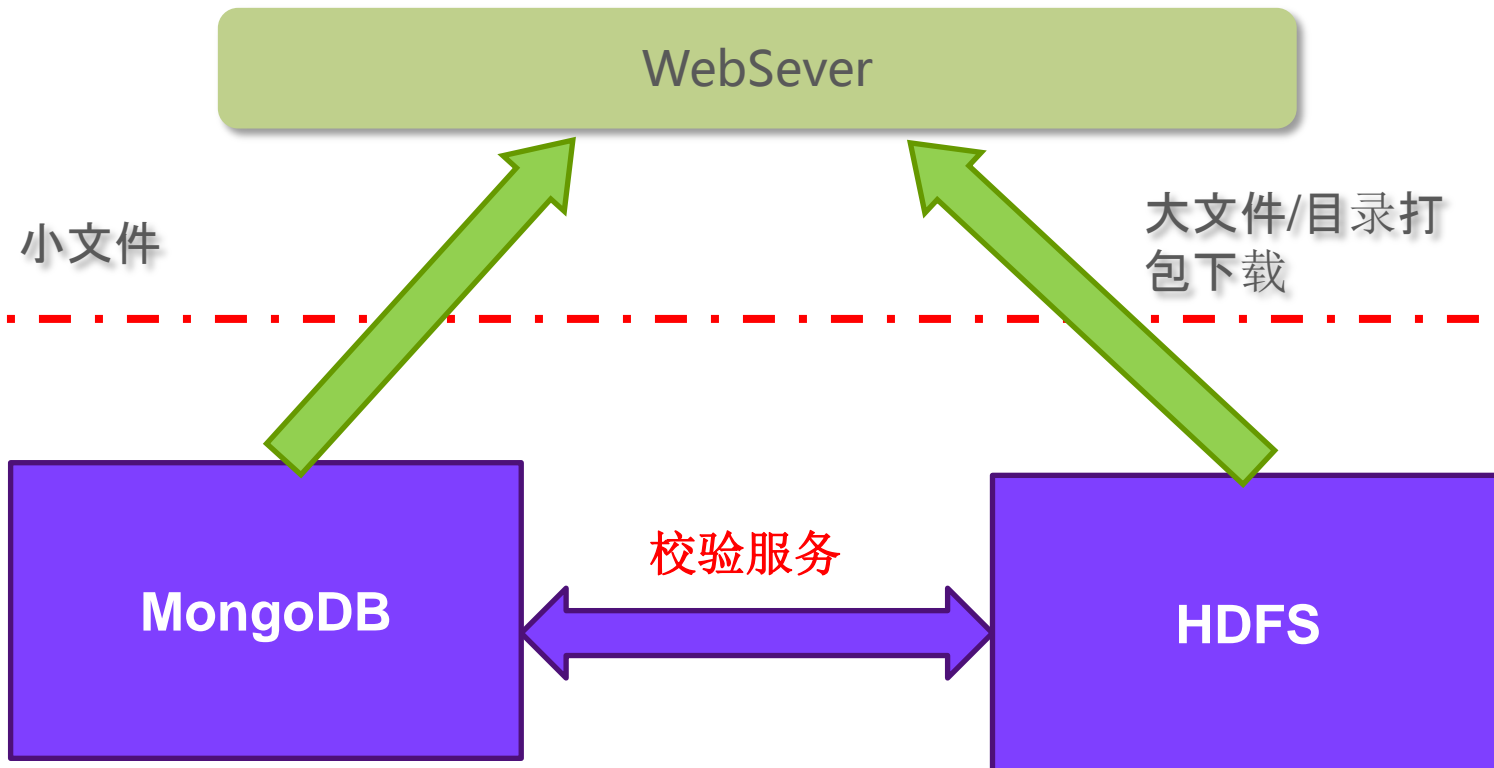
分布式文件系统分析

HDFS和MongoDB

- 大文件数据
 - HDFS
- 小文件数据及分块数据
 - MongoDB
- 文件“垃圾”数据回收







Hadoop Archive , Sequence file, CombineFileInputFormat

- 尽量创建索引
- 限定返回结果条数
- Filter只返回需要的数据
- 优化主键，尽量自己控制主键ID
- UUID主键使用BinaryData数据类型存储
- 注意文件大小不超过16M

➤ 系统参数优化

- GC策略

➤ 带宽策略优化

- 区分带内与带外心跳
- NameNode的备份不影响正常带宽使用

➤ 同步锁机制尽量少用

- 所有文件IO操作的地方尽可能不要加同步锁
- 大锁尽可能拆成小锁

➤ 文件副本数设置

- 根据应用的访问频率设置不同份数

➤ NameNode采取人工切换模式

➤ NameNode启动失败分析方法

- 查看日志，确定问题位置
- 备份fsimage，edits等
- 使用OfflineImageViewer方法查看（可能不行）
- 编译相应Hadoop版本
- 在相关读取文件信息的地方适当加一些try/catch
- 多次查找分析元数据信息
- Replay重新生成fsimage

➤ 明析需求

- 取舍需求
- 抓住本质
- 动态追踪

➤ 选择方法

- 成熟技术
- 均衡复杂度
- 设计模型
- 多次迭代
- 定期产出

➤ 高效运营

- 优化配置
- 智能监控
- 分级管理

THANKS

www.carbonite.cn