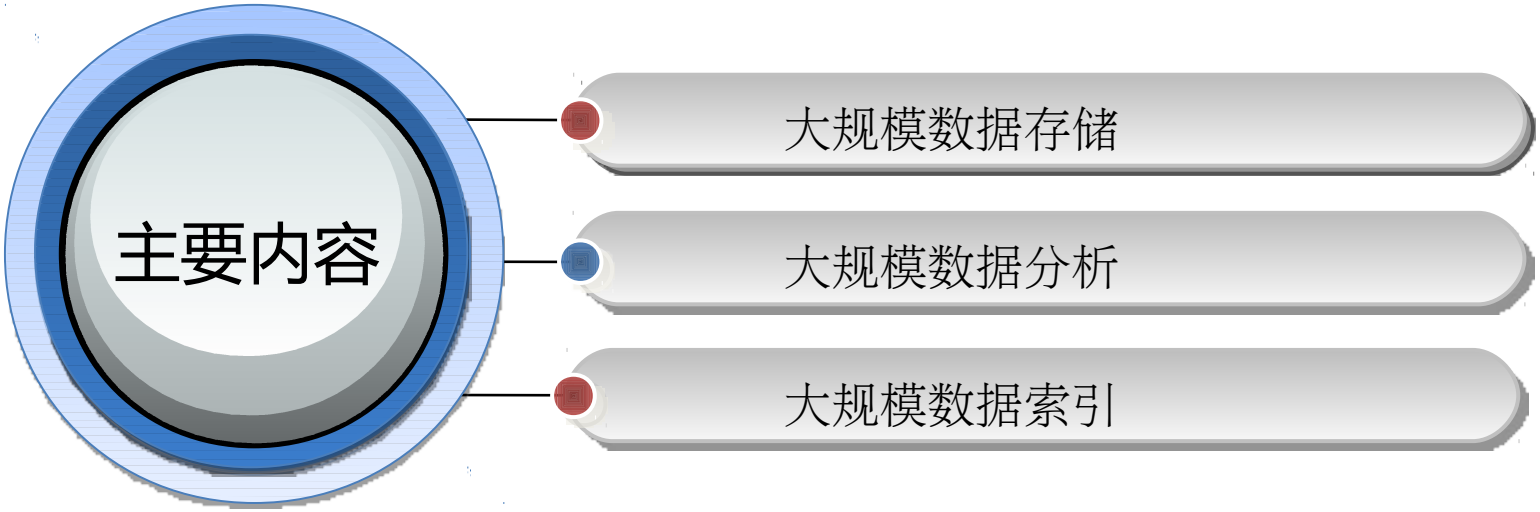
大规模数据处理

Baidu 百度



大规模数据存储

Lustre：

设计前提： 硬件是不容易坏的。

特　　点： 随机访问性能好，没有容错，规模低于 1PB ，节点失效后部分数据不能访问。

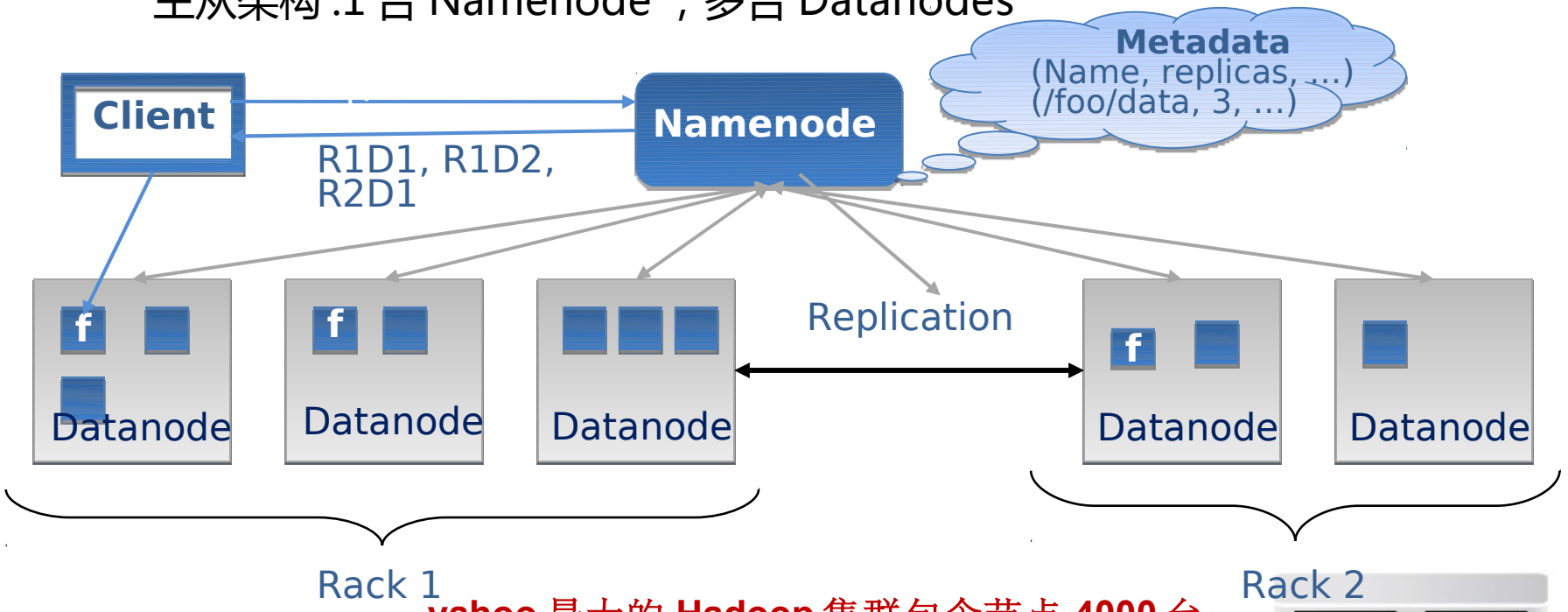
HDFS：

设计前提： 硬件是容易坏的，坏了也可以自动恢复。

特　　点： 容错，不需要人工干预，节点失效后系统任然可持续提供服务，规模可以扩展到 EB 。

系统结构

主从架构 :1 台 Namenode ，多台 Datanodes



yahoo 最大的 Hadoop 集群包含节点 4000 台；所有 Hadoop 集群节点总共一万台

HDFS 优势

- ✓ 支持海量存储；
- ✓ 全局命名空间；
- ✓ 数据高可用性；
- ✓ 服务高可靠性；
- ✓ 系统扩展性好；
- ✓ 数据安全性；
- ✓ 易用性（ vfs 兼容层）；
- ✓ 支持 MapReduce 编程框架；
- ✓ 支持 Hbase 、 Hypertable 等分布式索引系统。

HDFS 不足

- ✓ 随机读性能较差；
- ✓ 只支持单一追加（已满足应用需要）；
- ✓ 文件写入不立即可读，不支持“ tail –f”；
- ✓ 不支持 sync 、 mmap 和软硬链接操作；
- ✓ Namenode 是单点（双机备份策略基本解决问题）；
- ✓ 大量小文件会面临 Namenode 内存不足等问题；

百度应用实践 - 问题

- ✓ 存储超过 20PB 数据
- ✓ 每日新增数据超过 10TB
- ✓ NameNode 瓶颈问题（容量和性能）
- ✓ 数据安全性
- ✓ 每周近百块故障硬盘

百度应用实践 - 对策

- ✓ 2000+ NODES
- ✓ NODES ： 2*4 core ， 12*1 TB disk
- ✓ 分布式 NameNode
- ✓ 访问权限控制
- ✓ 故障硬盘自动发现并淘汰

大规模数据分析

MPI ：

设计前提： 输入数据一般不会多于 10TB ，计算很密集，计算相关性很强，硬件不容易坏。

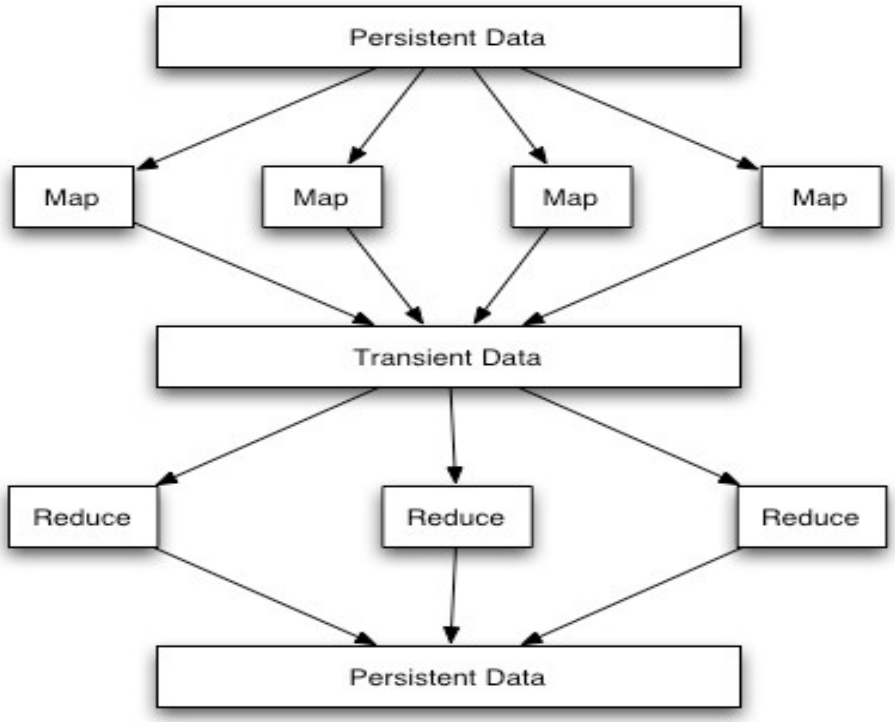
特　　点： 适用于数据相关性强，迭代次数多的计算，不适合处理过大规模数据，节点数不超过百台，节点失效会影响全局。

MapReduce ：

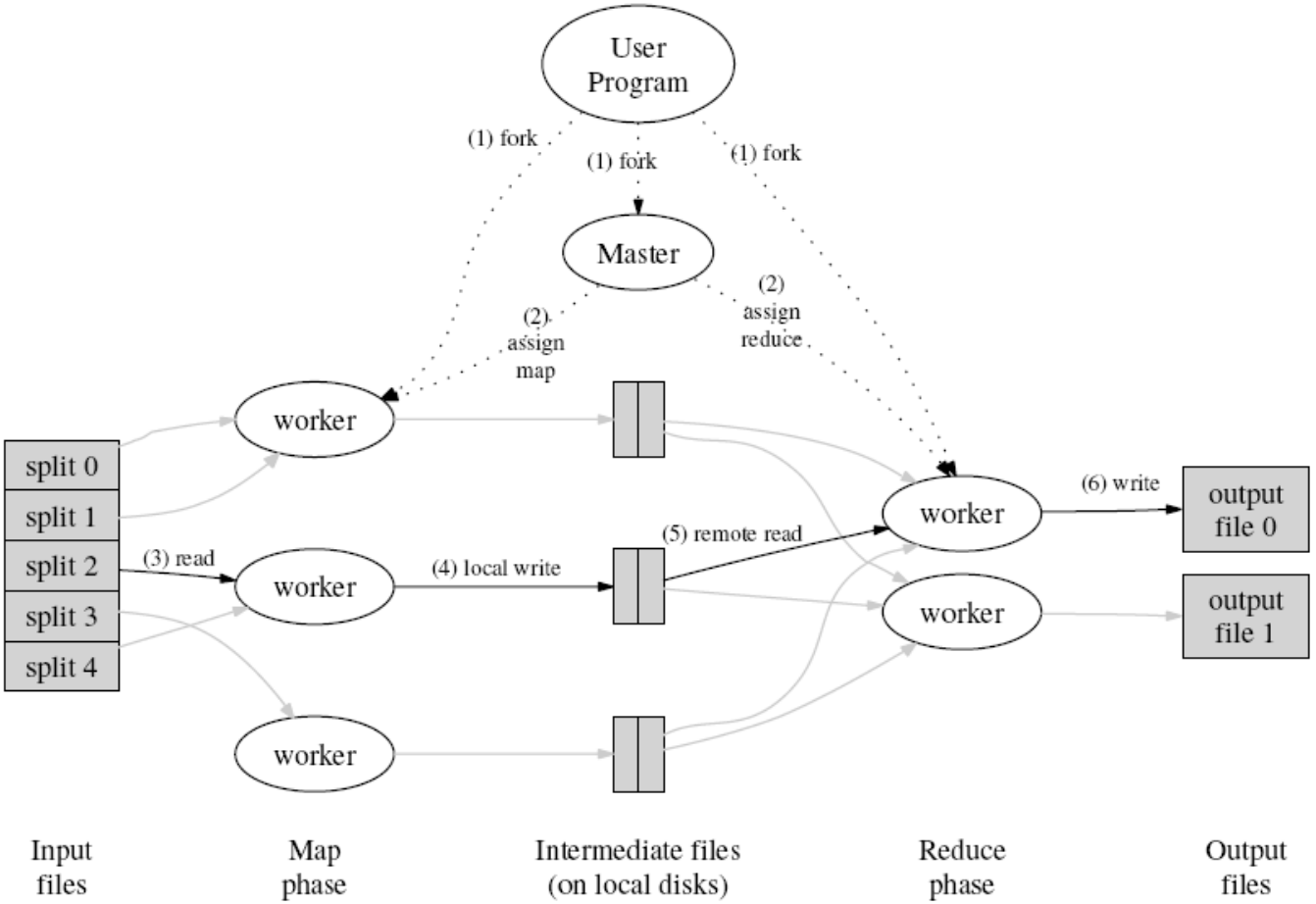
设计前提： 输入数据会超过 100TB ，数据全局相关性弱，硬件是容易坏的。

特　　点： 适用于大规模数据处理，节点规模可以达到数千台，节点失效对系统无影响。

MapReduce 概念模型



MapReduce 实现模型



MapReduce-Hadoop 实现

- ✓ Master-JobTracker
  - 作业与任务调度
  - 负责将中间文件信息通知给 reducer 所在的 worker
  - Master 周期性检查 Worker 的存活
- ✓ Worker-TaskTracker
  - TaskTracker 空闲，向 Master 要任务
  - 执行 mapper 或者 reducer 任务
- ✓ 框架所做的处理
  - 作业任务调度
  - 错误处理，失败任务自动重算
  - 防止慢作业，任务的预测执行

百度应用实践 - 问题

- ✓ 每天处理 1PB 以上数据
- ✓ 每天提交 10000+JOBs
- ✓ 多用户共享机群
- ✓ 实时 JOB 和优先级问题
- ✓ JobTracker 压力
- ✓ JAVA 语言效率
- ✓ Hadoop map-reduce 效率
- ✓ 复杂机器学习算法应用

百度应用实践 - 对策

- ✓ 可伸缩的计算资源调度系统
- ✓ 计算资源和 IO 资源的平衡
- ✓ 提高硬盘吞吐降低 IOPS
- ✓ 计算层重构（**Hadoop C++ 扩展**）
- ✓ Shuffle 和 Sort 重构（ **Hadoop C++ 扩展** ）
- ✓ MapReduce 与 MPI 配合使用

大规模数据索引

Mysql ：

设计前提：数据规模不超过 100GB ，数据相关性比较强，不考虑服务器失效。

特　　点：能提供复杂的 SQL 语义和事务处理，数据规模不能动态扩展，服务器死了，服务就会受影响。

HBase ：

设计前提：数据规模可能超过 PB ，数据相关性比较弱，必须实现分布式容错。

特　　点：语义比较简单，事务支持有限，数据规模能动态扩展，节点失效，自动冗余。

Hbase

			Access Group：首末班时间		Access Group：经过中关村时间			default
Range Server	Range	Row	CF:首班车时间	CF:末班车时间	CF:中关村西	CF:中关村东	CF:中关村北	CF:是否空调车
Range Server i	(0.. 113]	113	6:00 (2008.6.1 06:00 起执行)	22:00	6:15			否
Range Server ii	(113.. 407]	386	5:50	21:30				否
		407	5:30	21:00				否
	(407.. 753]	696	6:00	21:00			6:20	否
		740	内外环 5:30 (2008.7.1 06:00 起执行)	外环 21:00 内环 21:30	内环 6:20	外环 5:50		否
		753	6:00	21:00				否
Range Server i	(753.. 983]	944	6:00	21:00	6:30			否
		983	6:00	20:30	6:25	6:40		是

百度应用实践 - 问题和对策

- ✓ 随机访问效率偏低
- ✓ 节点故障时超时时间长
- ✓ API 易用性问题
- ✓ 与 HDFS 耦合时的稳定性问题

总结：正在重点解决的

- ✓ HDFS namenode 的分布式改进
- ✓ HDFS datanode 的读写异步化
- ✓ MapReduce 的 jobtracker 的分布式改进
- ✓ MapReduce 的新的作业和任务调度器
- ✓ MapReduce 的 hadoop c++ 扩展框架

总结：原则

- ✓ 大规模数据处理要求系统容错性好
- ✓ 规模可以通过机器数量扩展
- ✓ 为了满足容错性和扩展性，放弃兼容性
- ✓ 成熟的系统同时使用传统的方案和新方案

问题解答