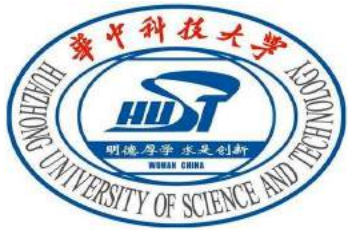
第三章 大数据存储与管理



肖江

Mail：jiangxiao@hust.edu.cn

Office: 东五楼 222 室

3.1 引言

目录

3.1 引言

- 数据非结构化的特征明显
- 需依靠分布式文件系统、分布式数据库、NoSQL数据库、云数据库等技术来实现
- 将学习基本原理、基本结构、访问接口、运行机制等内容

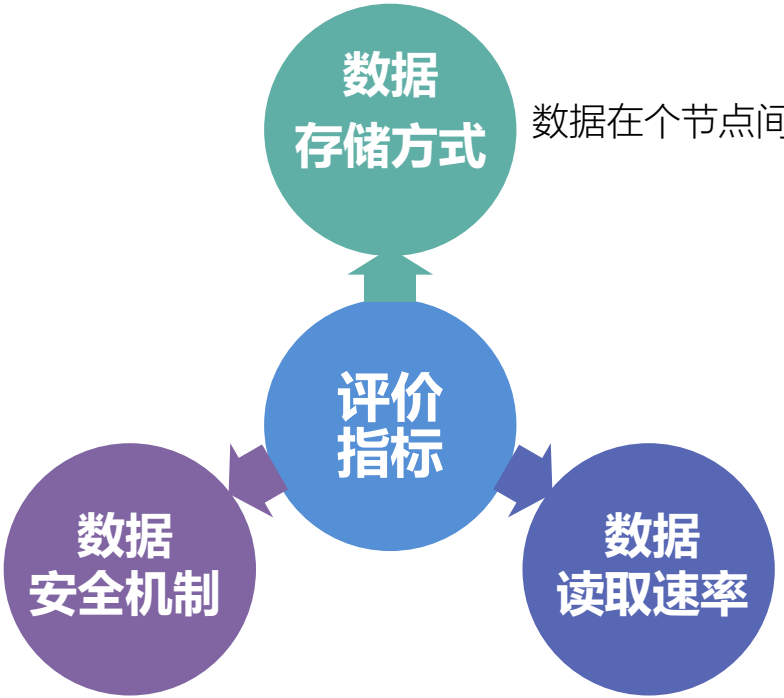
3.2 分布式文件系统

3.2.1 分布式文件系统基本概念

- **分布式文件系统(Distributed File System, DFS)指文件系统管理的物理存储资源不仅存储在本地节点上，还可以通过网络连接存储在非本地节点上。**
  - 分布式文件系统可以有效解决备份、安全、可扩展等数据存储和管理的难题：将固定于某个节点的文件系统，扩展到多个节点，众多的节点组成一个文件系统网络。

3.2 分布式文件系统

3.2.1 分布式文件系统基本概念



3.2 分布式文件系统

3.2.1 分布式文件系统基本概念

3.2 分布式文件系统

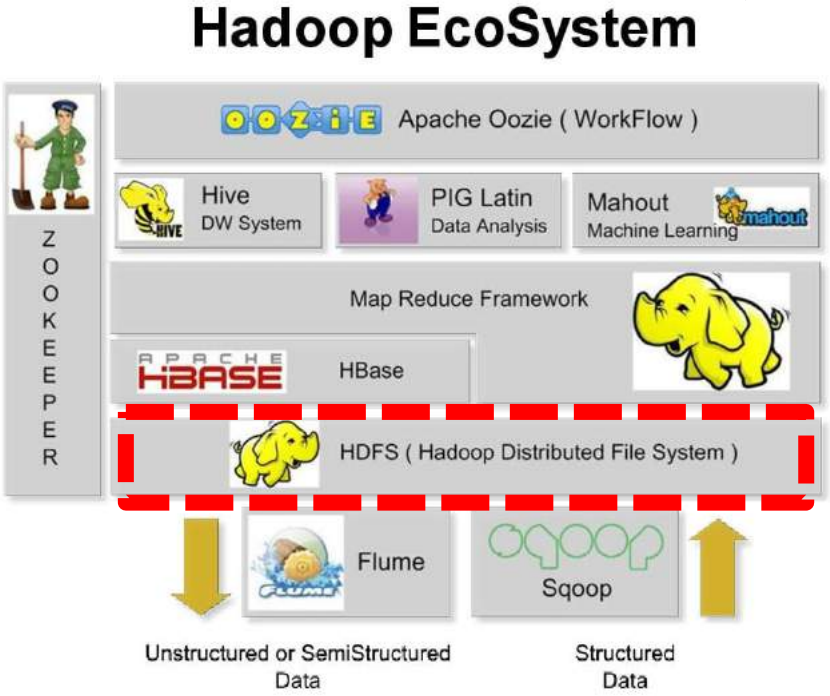
3.2.1 分布式文件系统基本概念

- 分布式文件系统的关键技术包括

统一名字空间	锁管理机制	副本管理机制
数据存取方式	安全机制	可扩展性

3.2 分布式文件系统

3.2.2 HDFS简介



3.2 分布式文件系统

3.2.2 HDFS简介

- **HDFS特性**

一次写入多次读取

(Write-Once-Read-Many)

- 降低并发性控制要求
- 支持高吞吐量访问

将处理逻辑

放置到数据附近

- 移动计算比移动数据更划算

3.2 分布式文件系统

3.2.2 HDFS简介

- **HDFS的主要设计目标是：**
  - 通过自动维护多副本和在故障发生时自动重新部署处理逻辑，实现可靠性
  - 通过 MapReduce 流进行数据访问
  - 采用简单可靠的聚合模型
  - 处理逻辑接近数据，而不是数据接近处理逻辑
  - 跨异构普通硬件和操作系统的可移植性
  - 系统规模可伸缩性好
  - 通过跨多个普通个人计算机集群分布数据和处理，以节约成本
  - 通过分布数据和逻辑到数据所在的多个节点上进行并行处理来提高效率

3.2 分布式文件系统

3.2.2 HDFS简介

- HDFS提供一个原生 Java应用程序编程接口（API）和一个针对这个Java API的原生C语言封装器。另外，可以使用Web浏览器来浏览 HDFS 文件。

应用程序	说明
FileSystem (FS) shell	命令行接口，类似于常见的 Linux和UNIX shells（bash、csh，等），支持与 HDFS 数据进行交互。
DFSAdmin	可用于管理一个 HDFS 集群的命令集。
fsck	Hadoop 命令/应用程序的一个子命令。可使用 fsck 命令来检查文件的不一致（比如缺失块），但不能使用 fsck 命令更正这些不一致。
Name node 和 Data node	这些节点拥有内置 web 服务器，允许管理员检查集群的当前状态。

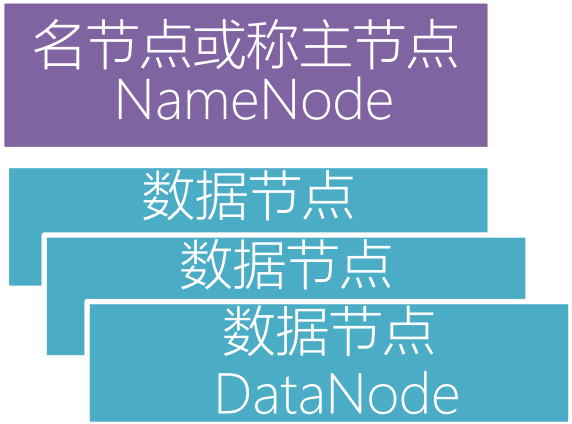
3.2 分布式文件系统

3.2.2 HDFS简介

- ❖HDFS的体系结构
- ❖存储原理
- ❖数据读写方法

3.2 分布式文件系统

3.2.3 HDFS体系结构

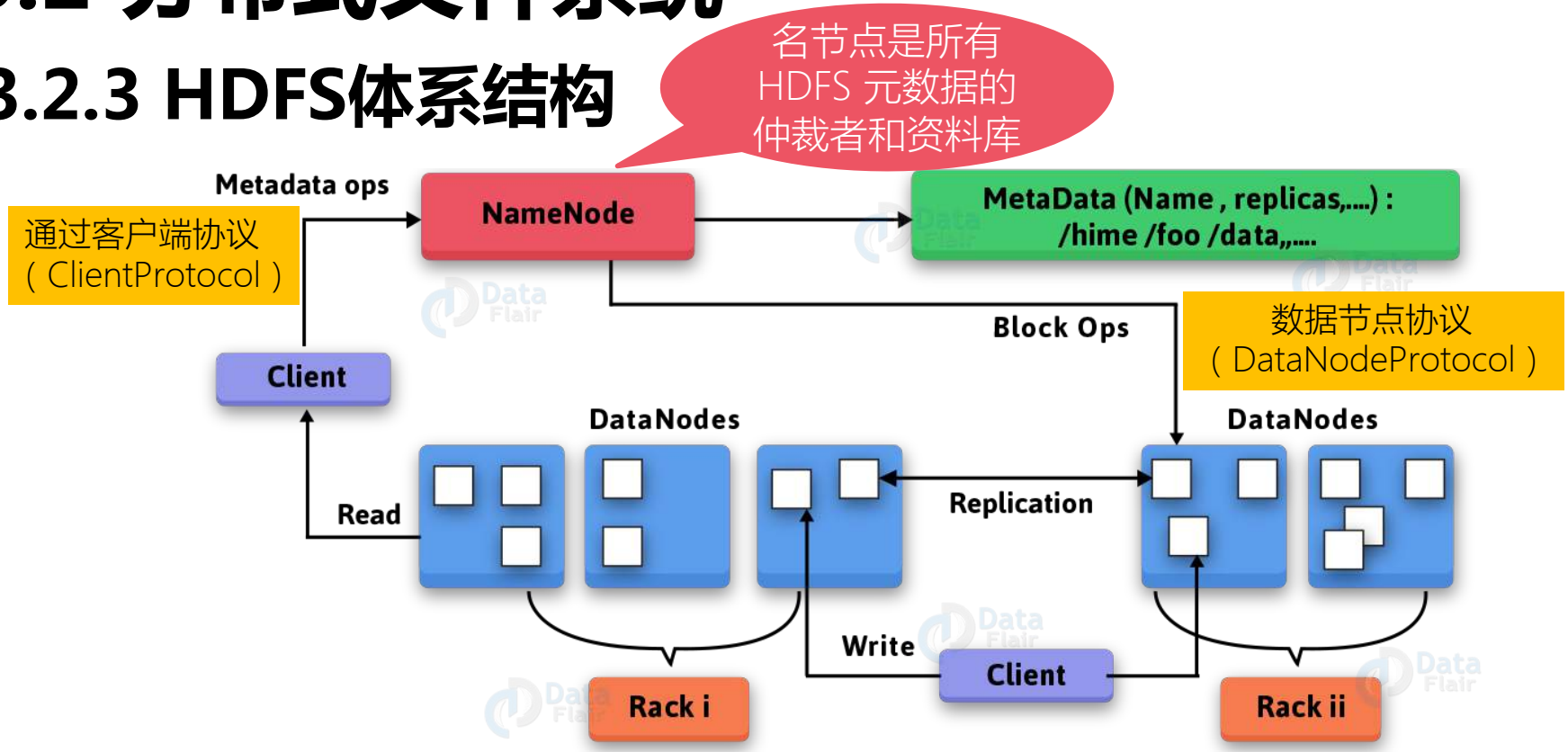


- 通信基于TCP/IP

设计目标：高可靠、高容错、高吞吐量、可扩展

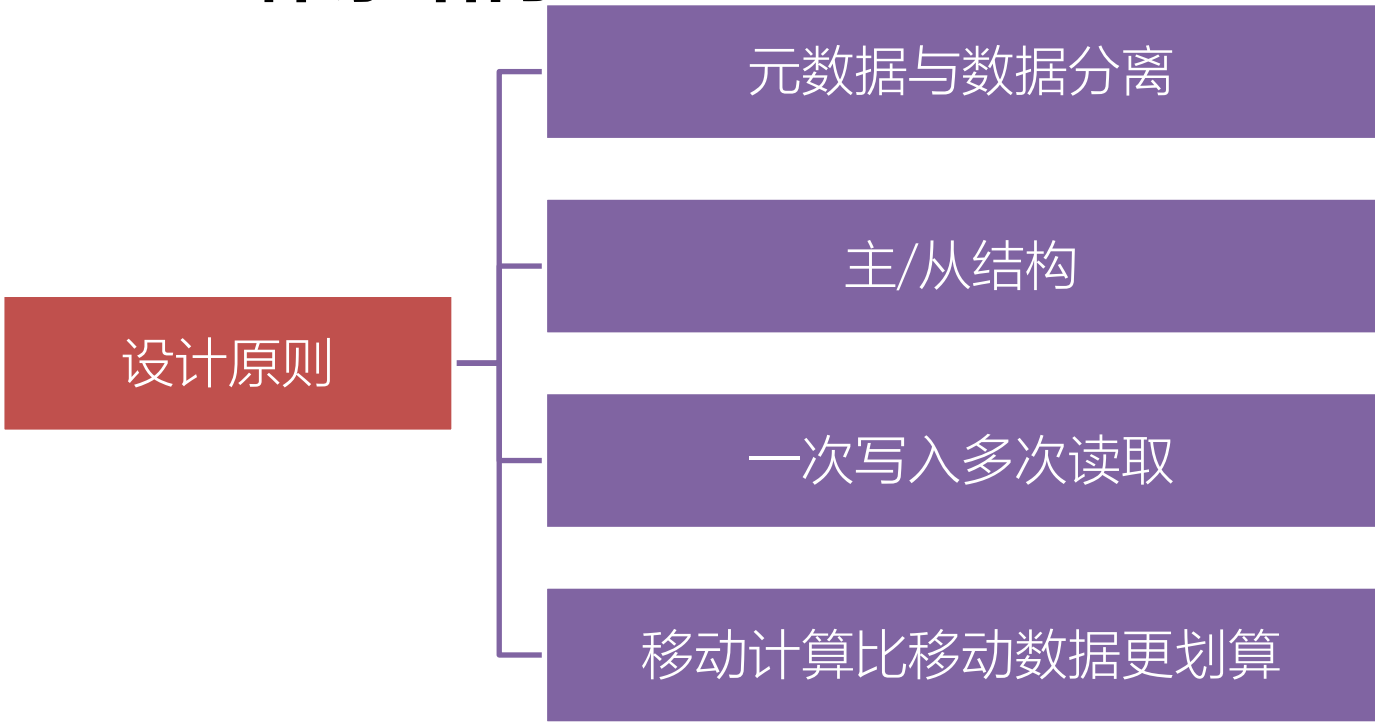
3.2 分布式文件系统

3.2.3 HDFS体系结构



3.2 分布式文件系统

3.2.3 HDFS体系结构



3.2 分布式文件系统

3.2.4 HDFS存储原理

- HDFS以文件分块的形式实现对大文件、超大文件**安全**、**可靠**、**快速访问**的**分布式**存储。

  ❖分块原理？

  ❖如何管理分块文件？

  ❖错误检测及恢复机制如何实现？

3.2 分布式文件系统

3.2.4 HDFS存储原理

名节点和数据节点：名节点负责执行文件系统的操作；数据节点负责来自客户端的文件读写。

文件系统命名空间：名节点维护文件系统命名空间。

数据复制：存储文件副本

文件系统元数据的持久存储

多副本的流式复制

心跳检测和重新复制

3.2 分布式文件系统 HDFS

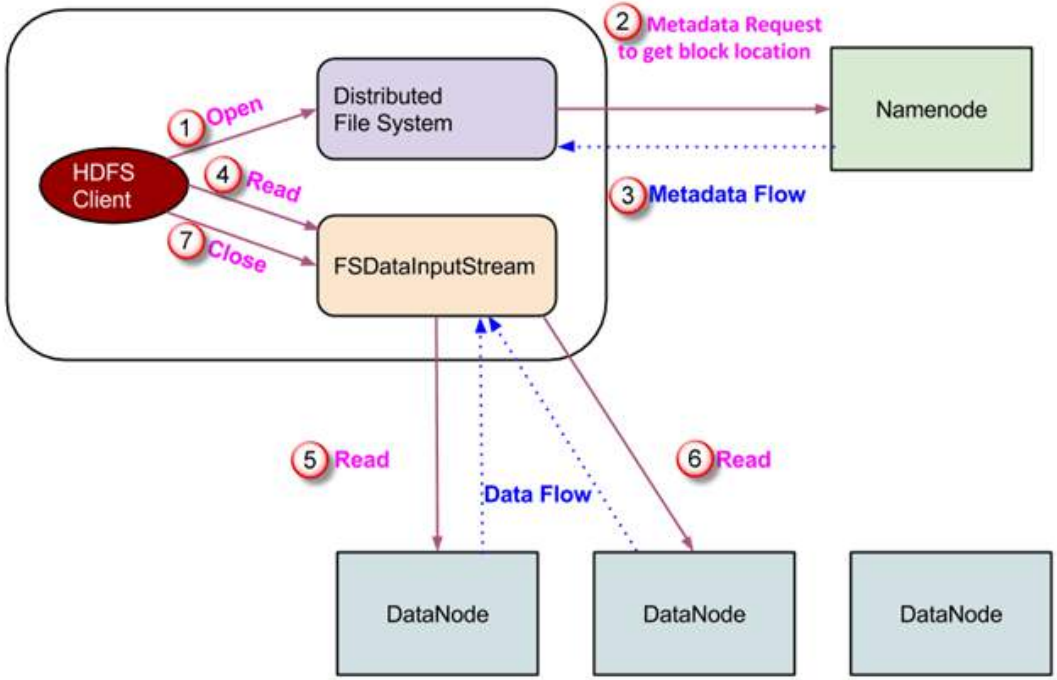
3.2.5 HDFS数据读写

- HDFS集群主要由管理文件系统元数据（Metadata）的名节点和存储实际数据的数据节点构成；
- HDFS数据文件被分成大小固定的块 (默认64MB)，作为独立的单元存储；
- 读/写操作运行在块级；
- HDFS操作上是数据复制的概念，其中在数据块的多个副本被创建，分布在整个集群的多个节点，以便在节点故障的情况下实现数据的高可用性。

3.2 分布式文件系统 HDFS

3.2.5 HDFS数据读写

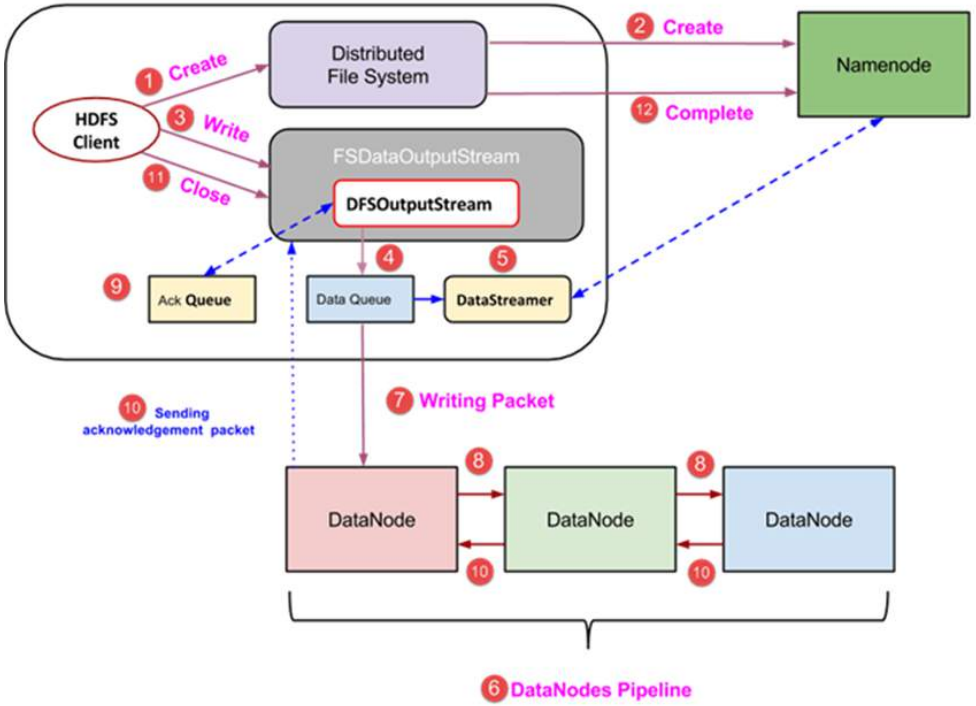
- HDFS数据读



3.2 分布式文件系统 HDFS

3.2.5 HDFS数据读写

- HDFS数据写



目录

内容回顾-3.2 分布式文件系统

3.2.1 分布式文件系统基本概念

- **分布式文件系统(Distributed File System, DFS)指文件系统管理的物理存储资源不仅存储在本地节点上，还可以通过网络连接存储在非本地节点上。**
  - 分布式文件系统可以有效解决备份、安全、可扩展等数据存储和管理的难题：将固定于某个节点的文件系统，扩展到多个节点，众多的节点组成一个文件系统网络。

内容回顾-3.2 分布式文件系统

3.2.1 分布式文件系统基本概念

- 分布式文件系统的关键技术包括

统一名字空间	锁管理机制	副本管理机制
数据存取方式	安全机制	可扩展性

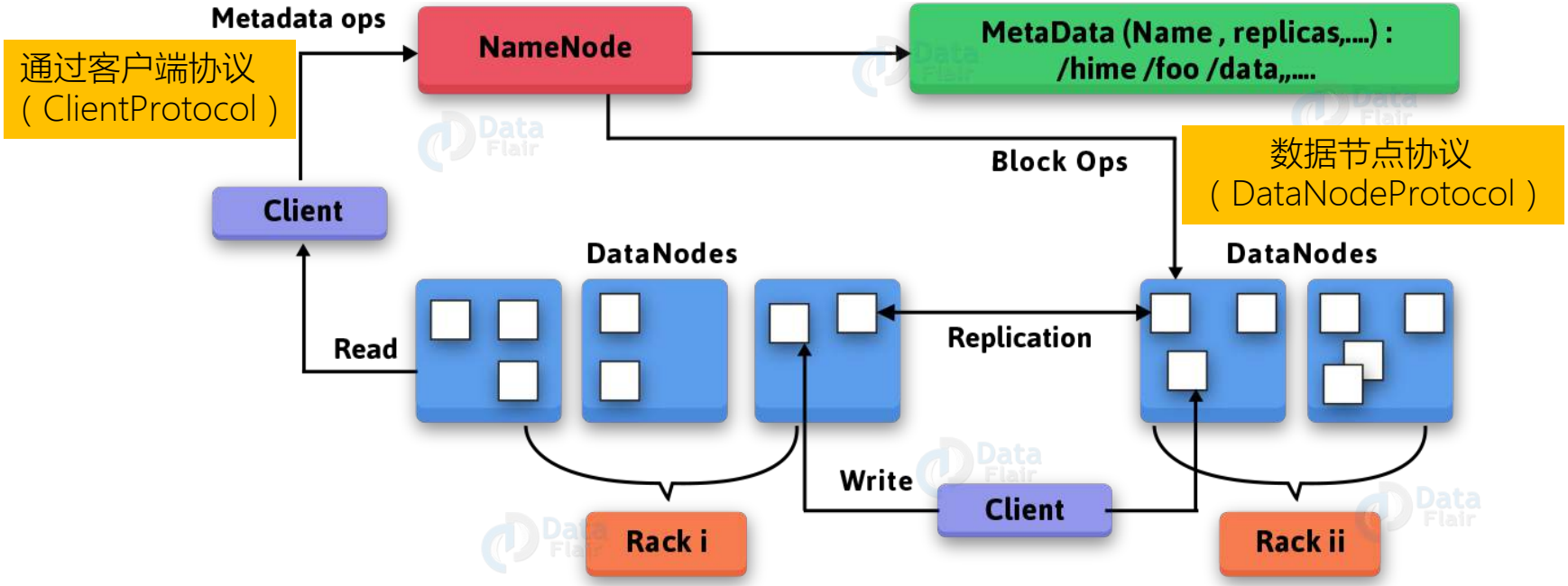
内容回顾-3.2 分布式文件系统

3.2.2 HDFS简介

- **HDFS的主要设计目标是：**
  - 通过自动维护多副本和在故障发生时自动重新部署处理逻辑，实现可靠性
  - 通过 MapReduce 流进行数据访问
  - 采用简单可靠的聚合模型
  - 处理逻辑接近数据，而不是数据接近处理逻辑
  - 跨异构普通硬件和操作系统的可移植性
  - 系统规模可伸缩性好
  - 通过跨多个普通个人计算机集群分布数据和处理，以节约成本
  - 通过分布数据和逻辑到数据所在的多个节点上进行并行处理来提高效率

内容回顾-3.2 分布式文件系统

3.2.3 HDFS体系结构



内容回顾-3.2 分布式文件系统

3.2.4 HDFS存储原理

名节点和数据节点：名节点负责执行文件系统的操作；数据节点负责来自客户端的文件读写。

文件系统命名空间：名节点维护文件系统命名空间。

数据复制：存储文件副本

文件系统元数据的持久存储

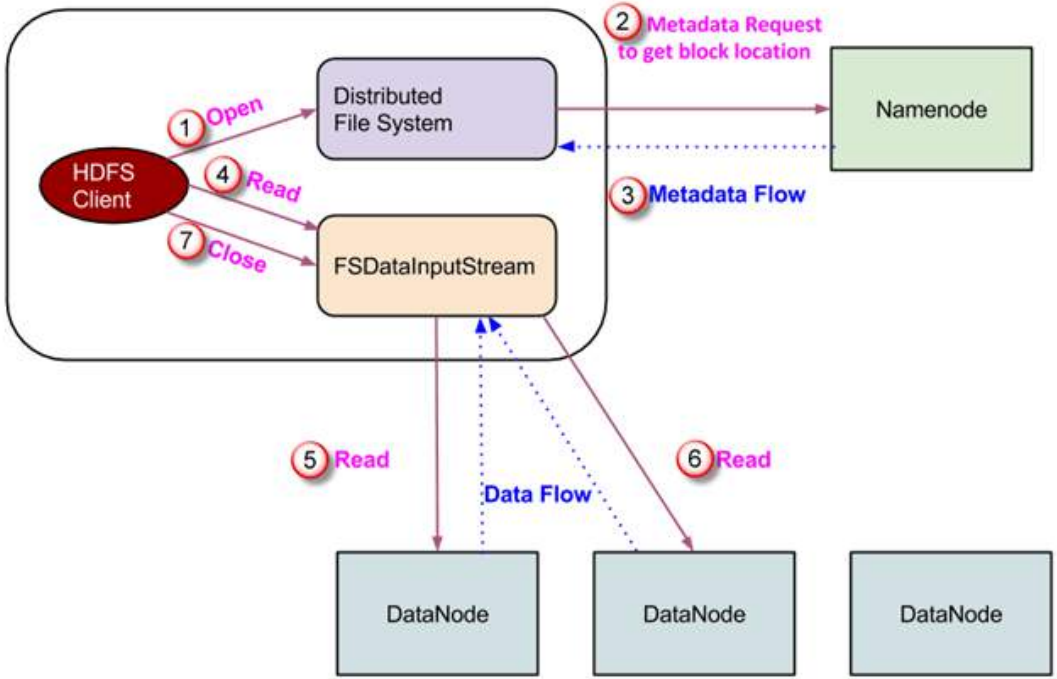
多副本的流式复制

心跳检测和重新复制

内容回顾-3.2 分布式文件系统

3.2.5 HDFS数据读写

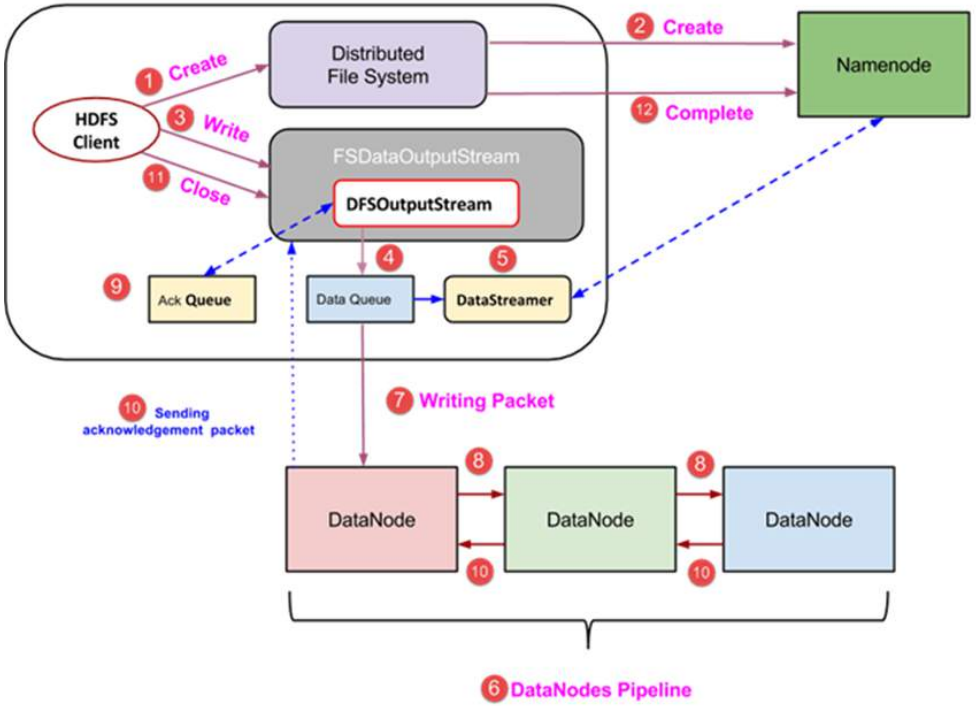
- HDFS数据读



内容回顾-3.2 分布式文件系统

3.2.5 HDFS数据读写

- HDFS数据写

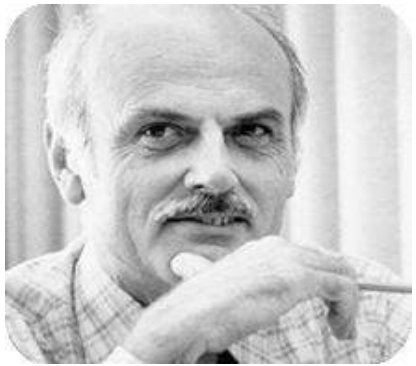


目录

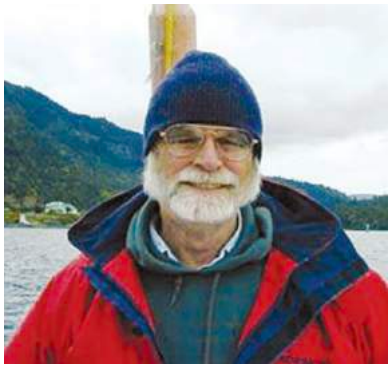
内容回顾

1.6.1 大数据生命周期

查尔斯·威廉·巴赫曼
Charles William Bachman
网状数据库之父
1973年图灵奖



埃德加·弗兰克·科德
Edgar Frank Codd
关系型数据库之父
1981年图灵奖



詹姆斯·尼古拉·格雷
James Nicholas "Jim" Gray
SQL之父，事务处理
1998年图灵奖

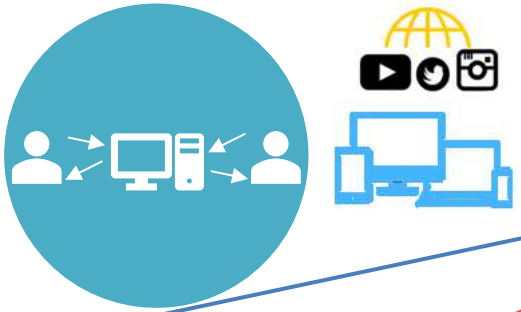
迈克尔·斯通布雷克
Michael Stonebraker
现代数据库概念和实践
2014年图灵奖

Web 3.0: The Internet of Value

廉价集群结构以及具有高容错能力、易于水平扩展的分布式数据处理技术

Web 1.0 [Push]

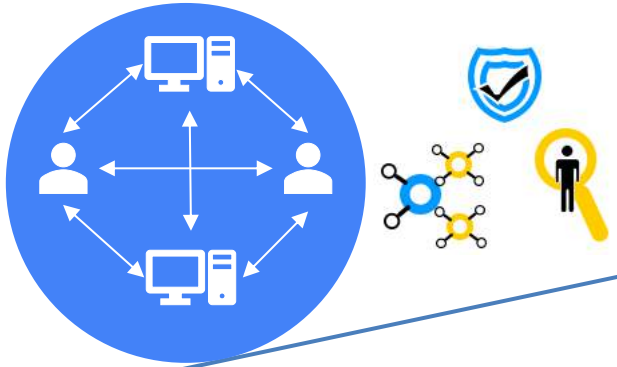
- Single PC-based
- Static (Read-only)



Web 2.0 [Share]

- Centralized Servers
- Read-Write Interactive Web

Google BigTable



Web 3.0 [Interoperable]

- Decentralized Nature
- Full Freedom of Interaction
- Read-Write Intelligent Web

3.3 分布式数据库HBase

- 概述
- HBase数据模型
- HBase实现模块
- HBase工作原理
- 与Hive的应用场景差异

HBase的原型是
谷歌的分布式存储系统BigTable

HBase是一个高可靠、高性能、面向列、可伸缩分布式数据库

https://hbase.apache.org

3.3 分布式数据库HBase

- 概述

66.3. What Is The Difference Between HBase and Hadoop/HDFS?

HDFS is a distributed file system that is well suited for the storage of large files. Its documentation states that it is not, however, a general purpose file system, and does not provide fast individual record lookups in files. HBase, on the other hand, is built on top of HDFS and provides fast record lookups (and updates) for large tables. This can sometimes be a point of conceptual confusion. HBase internally puts your data in indexed "StoreFiles" that exist on HDFS for high-speed lookups. See the Data Model and the rest of this chapter for more information on how HBase achieves its goals.

http://hbase.apache.org/book.html#arch.overview.hbasehdfs

3.3 分布式数据库HBase

- 概述

HBase vs just HDFS

	Plain HDFS/MR	HBase
Write pattern	Append-only	Random write, bulk incremental
Read pattern	Full table scan, partition table scan	Random read, small range scan, or table scan
Hive (SQL) performance	Very good	4-5x slower
Structured storage	Do-it-yourself / TSV / SequenceFile / Avro / ?	Sparse column-family data model
Max data size	30+ PB	~1PB

If you have neither random write nor random read, stick to HDFS!

cloudera

3.3 分布式数据库HBase

- 概述

F.3. HBase Papers

BigTable by Google (2006).

HBase and HDFS Locality by Lars George (2010).

No Relation: The Mixed Blessings of Non-Relational Databases by Ian Varley (2009).

F.4. HBase Sites

Cloudera's HBase Blog has a lot of links to useful HBase information.

- CAP Confusion is a relevant entry for background information on distributed storage systems.

HBase Wiki has a page with a number of presentations.

F.5. HBase Books

HBase: The Definitive Guide by Lars George.

F.6. Hadoop Books

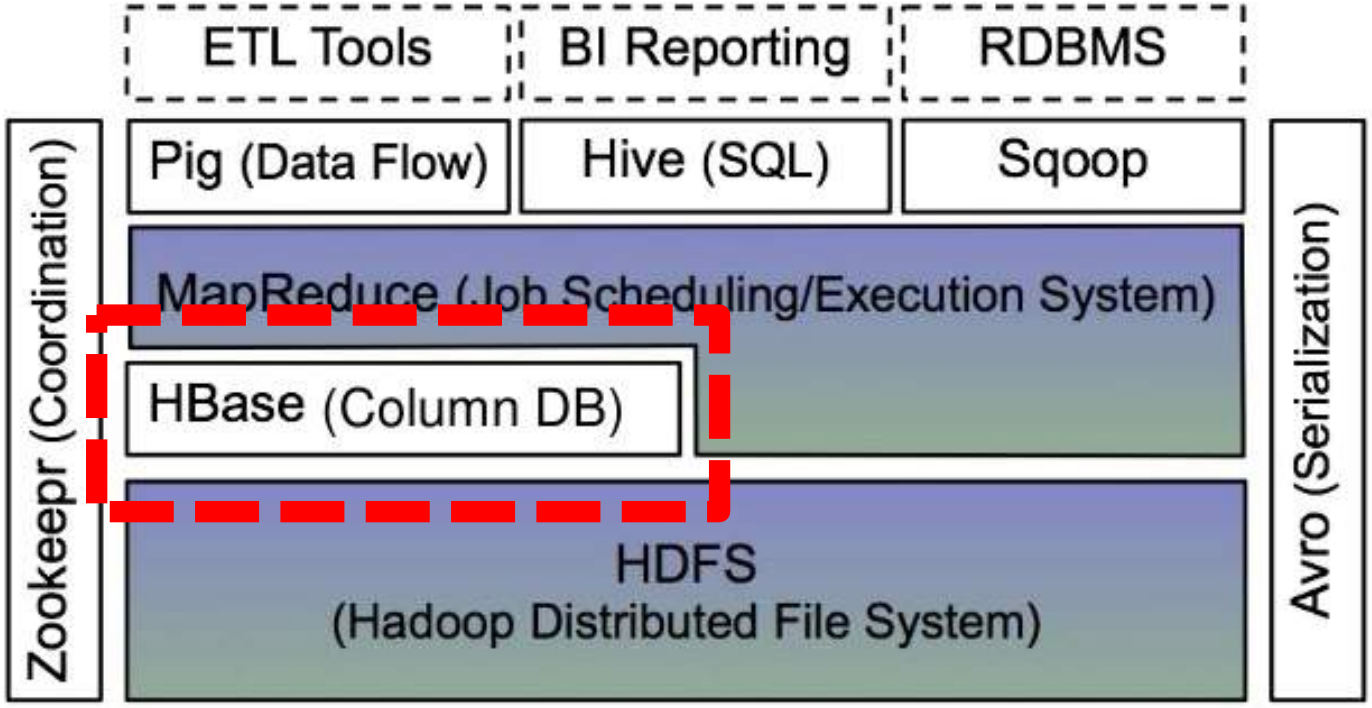
Hadoop: The Definitive Guide by Tom White.

http://abloz.com/hbase/book.html#other.info.papers

3.3 分布式数据库HBase

3.3.1 概述

主要用来存储非结构化和半结构化的数据



3.3 分布式数据库HBase

3.3.2 HBase数据模型

- HBase本质上是一个稀疏、多维度、排序的数据映射表；
- 行键，列族，列限定符和时间戳等

Row Keys	Column Family 1		Column Family 2	
	C1	C2	C3	C4
R1	V3	V6		V2
R2			V2	
R3		t1 V3 t2 V1	V6	
R4	V5			V9

HFile for CF1

R1:CF1:C1::V3
R1:CF1:C2::V6
R3:CF1:C2:t1:V3
R3:CF1:C2:t2:V1
R4:CF1:C1::V5

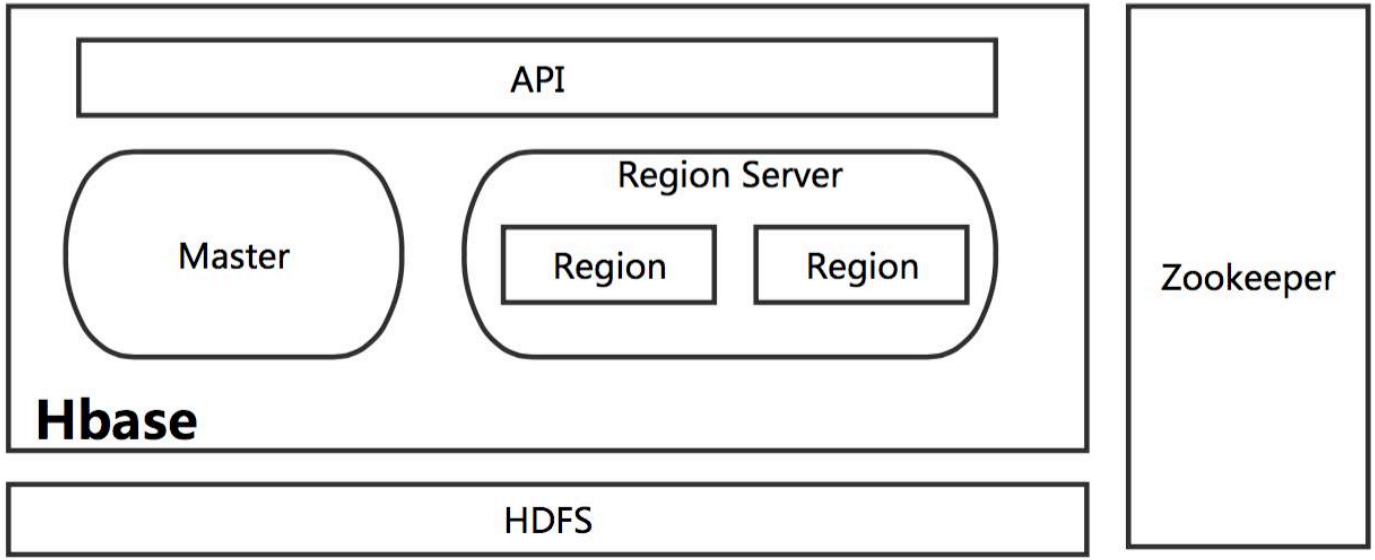
HFile for CF2

R1:CF2:C4::V2
R2:CF2:C3::V2
R3:CF2:C3::V6
R4:CF2:C4::V9

3.3 分布式数据库HBase

3.3.3 HBase实现模块

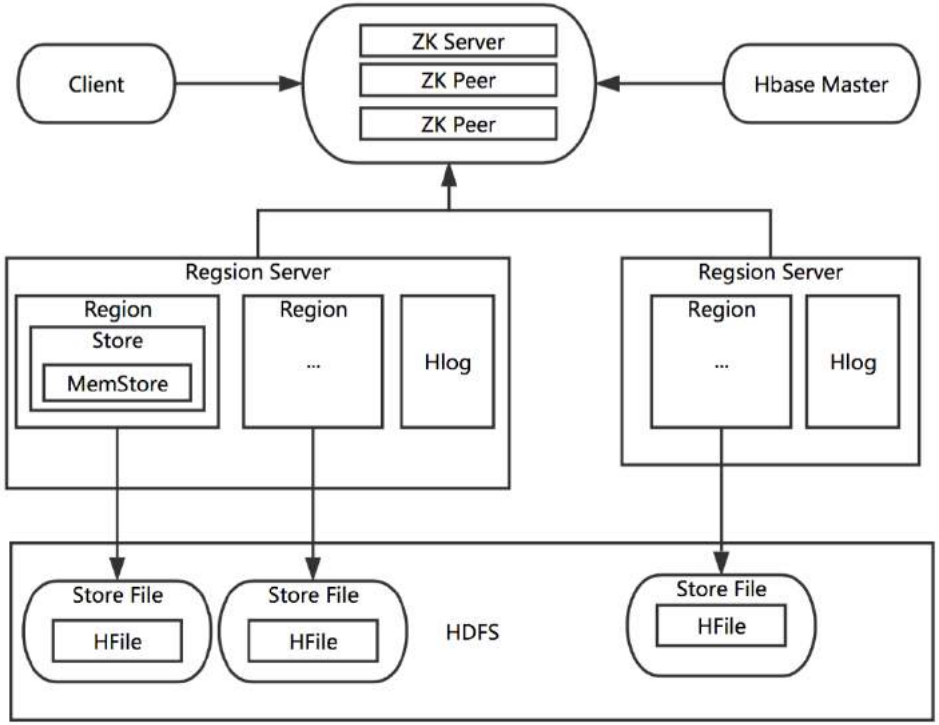
- 库函数、主 (Master) 服务器、分区 (Region) 服务器



3.3 分布式数据库HBase

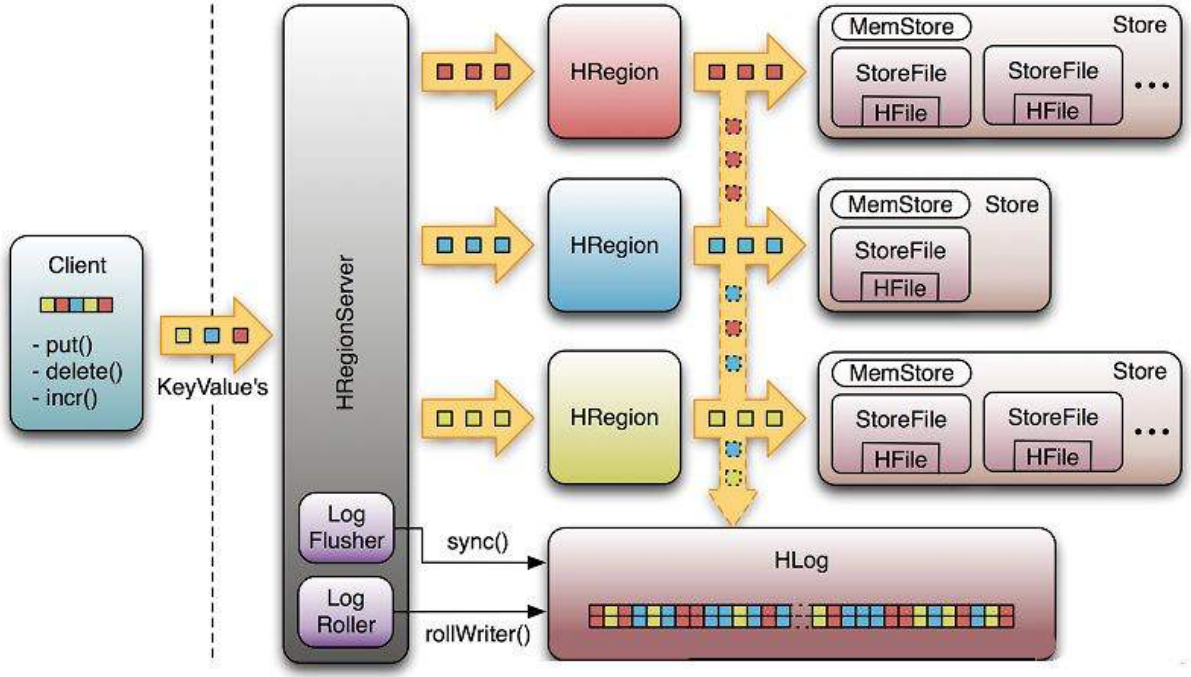
3.3.4 HBase工作原理

- Client：使用HBase的RPC机制与主服务器和分区服务器通信
- Zookeeper：集群管理
- Master：管理用户对表的增删改查
- Region：维护位于本机的所有分区，响应读写请求



3.3 分布式数据库HBase

3.3.4 HBase工作原理



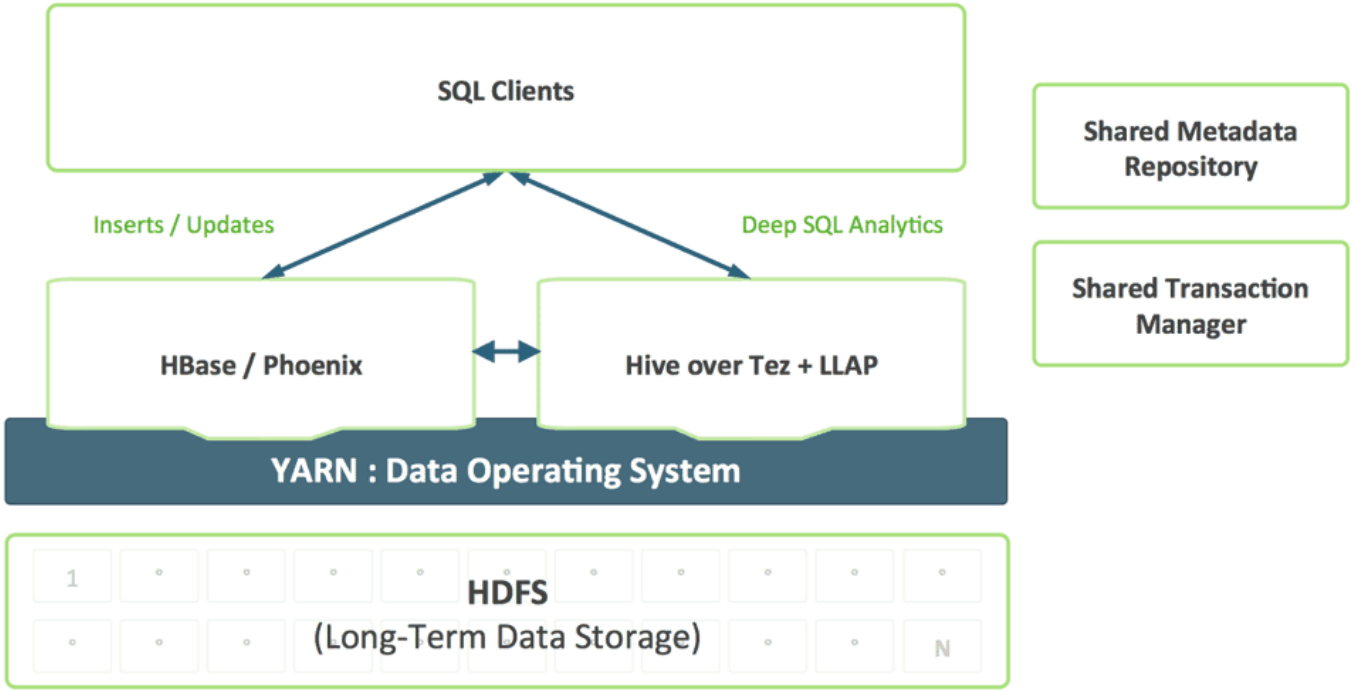
3.3 分布式数据库HBase

3.3.5 与Hive的应用场景差异

- Hive是基于Hadoop的数据仓库工具，可用于对一段时间内的数据进行分析查询；
- Hive适合对一段时间内的数据进行分析查询：不追求实时性；而HBase适合大数据的实时查询；
- HBase不直接支持SQL的语句查询，需要和Apache Phoenix搭配使用才支持SQL；而Hive直接支持

3.3 分布式数据库HBase

3.3.5 与Hive的应用场景差异



3.4 非关系型数据库

- 概述
- 典型NoSQL数据库
- NoSQL相关理论
- NoSQL与关系数据库对比
- NoSQL的新发展

3.4 非关系型数据库

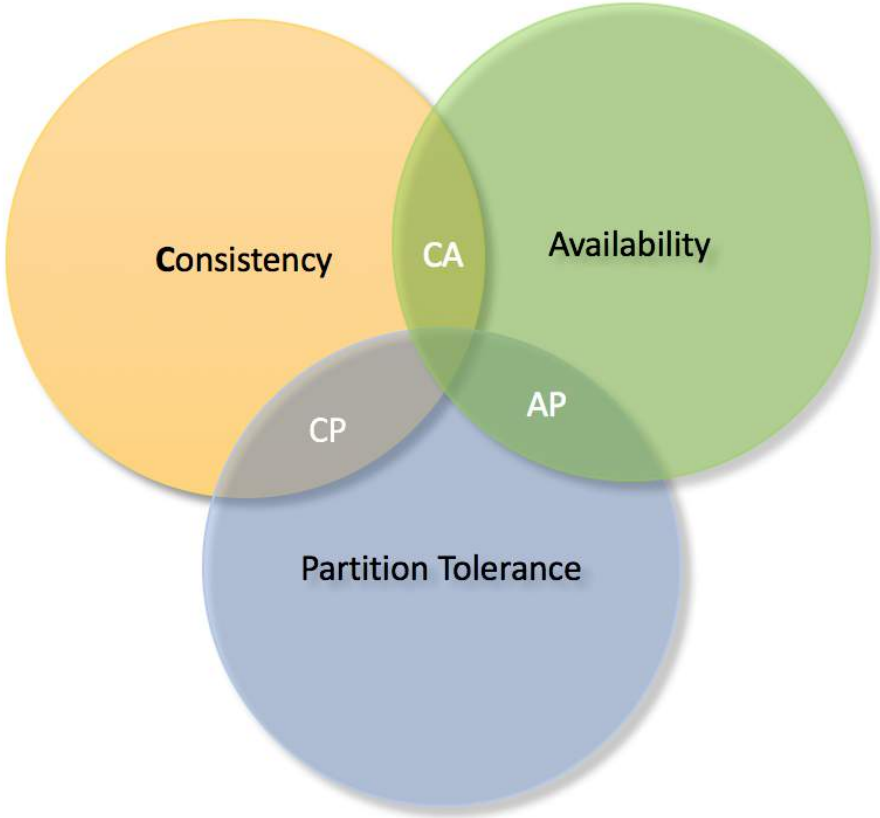
3.4.1 概述

- NoSQL是一种不同于关系数据库的数据库管理系统方案是对非关系型数据库的统称；
- 数据模型采用键/值，列族，文档等非关系模型；
- 非关系型数据库没有固定的表结构，通常也不存在连接操作，也没有严格遵守ACID约束；
- 灵活的可扩展性、灵活的数据模型、与云计算紧密融合

3.4 非关系型数据库

3.4.2 NoSQL相关理论

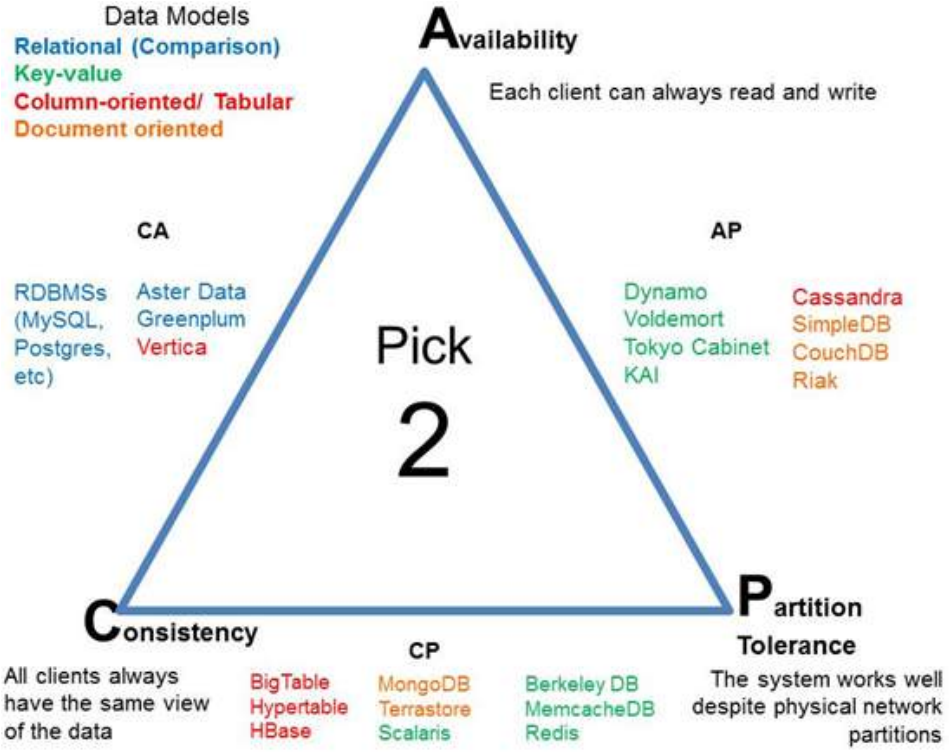
- **CAP理论**
  - C (Consistency)：一致性
  - A (Availability)：可用性
  - P (Tolerance of Network Partition)：分区容忍性



3.4 非关系型数据库

3.4.2 NoSQL相关理论

- **CAP理论**
  - C (Consistency)：一致性
  - A (Availability)：可用性
  - P (Tolerance of Network Partition)：分区容忍性



3.4 非关系型数据库

3.4.2 NoSQL相关理论

- **BASE理论**
  - BA (Basic Availability)：基本可用性，允许分区失败；
  - S (Soft state): 状态允许有短时间不同步，异步；
  - E (Eventual consistency)：最终一致性，数据最终是一致的，但不是实时一致；
  - 保证ACID中的A (原子性)和D (持久性)，适当牺牲C (一致性)和I (隔离性)

3.4 非关系型数据库

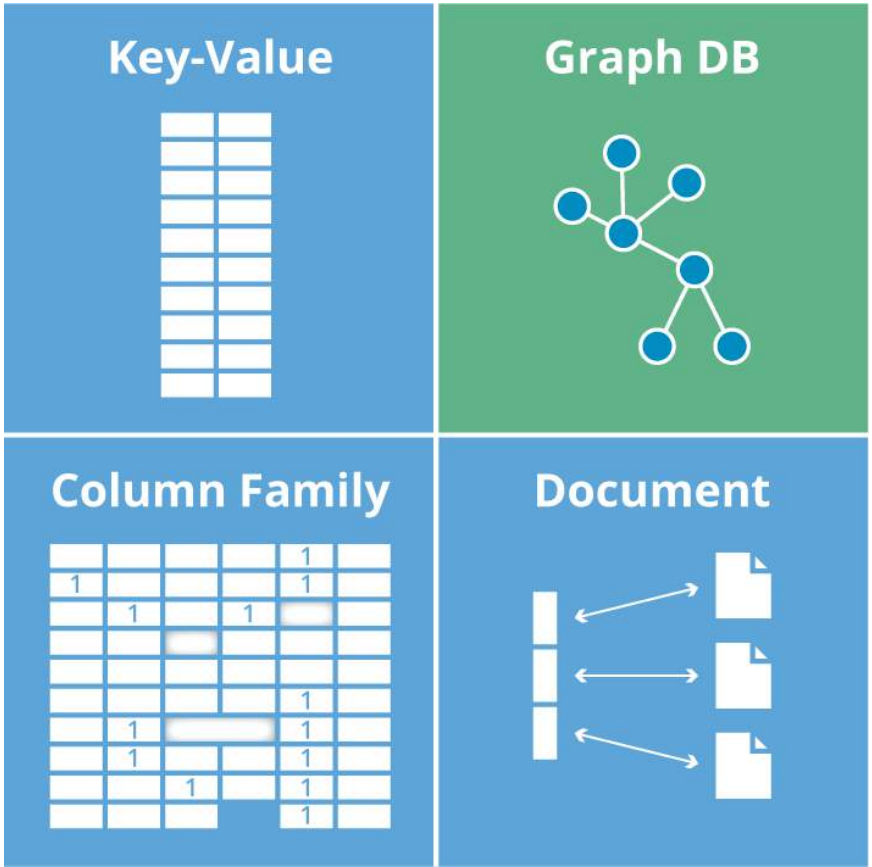
NoSQL与关系数据库对比

- 关系数据库
  - 优势：完善的关系数据理论、支持事务ACID、借助索引实现高效的查询、技术成熟
  - 劣势：可扩展性较差、无法较好的支持海量数据存储、事务的机制影响了系统整体的性能
- NoSQL
  - 优势：支持大规模数据存储、灵活的数据模型、强大横向扩展能力
  - 劣势：缺乏数据理论基础、复杂查询性能不高、不能实现事务强一致性、很难实现数据完整性、技术尚不成熟

3.4 非关系型数据库

3.4.3 典型NoSQL数据库

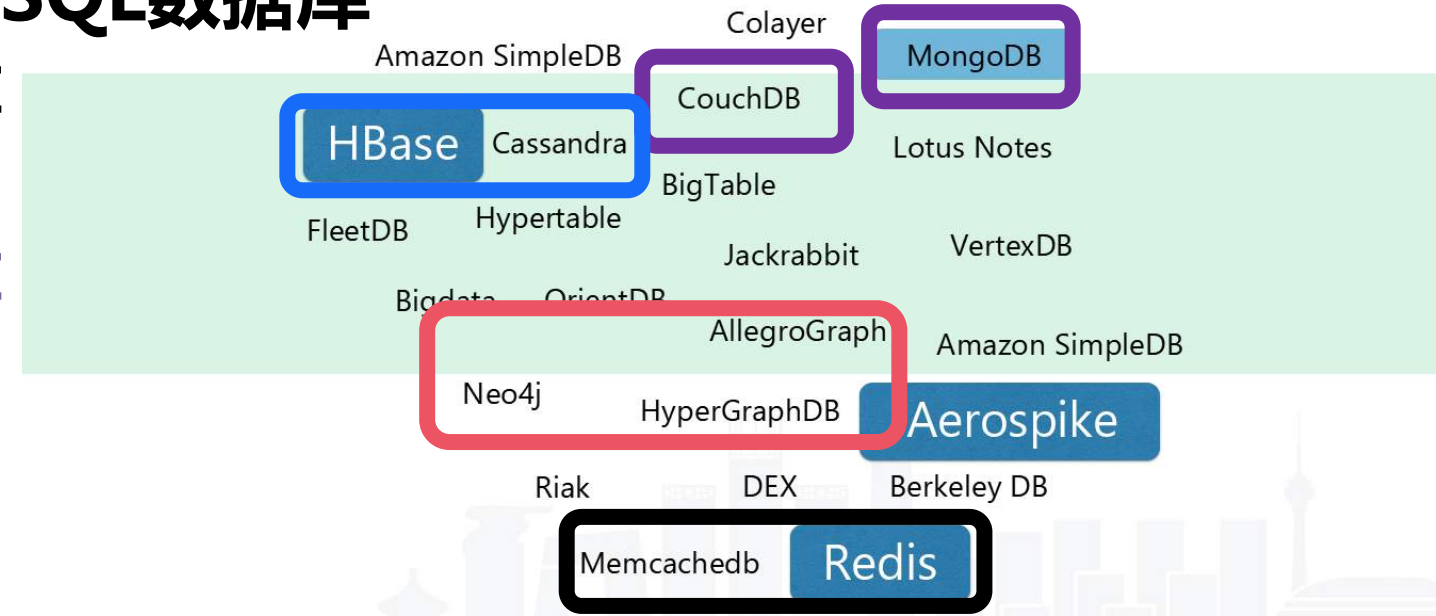
- 键值数据库
- 列数据库
- 文档数据库
- 图数据库



3.4 非关系型数据库

3.4.3 典型NoSQL数据库

- 键值数据库
- 列数据库
- 文档数据库
- 图数据库



3.4 非关系型数据库

3.4.4 NoSQL与关系数据库对比

- **Pros**
  - Transactions
  - Indexes
  - Joins
  - Strong consistency
- **Cons**
  - Limited scalability
  - Schema
  - Weak consistency with datacenter failure

- **Pros**
  - Scalable (w/ caveats)
  - Transactions (limited)
  - Indexes (limited)
  - Joins
  - Tunable consistency
- **Cons**
  - Schema
  - App-level sharding
  - Weak consistency with datacenter failure

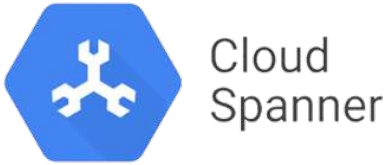
- **Pros**
  - Scalable
  - Unstructured data
  - Tunable consistency
- **Cons**
  - No transactions
  - No indexes
  - No joins
  - No availability or weak consistency with datacenter failure

- **Pros**
  - Scalable
  - Transactions
  - Indexes
  - Joins
  - Availability and strong consistency with datacenter failure
  - Tunable consistency
- **Cons**
  - Semi-relational schema
  - Higher write latencies

3.4 非关系型数据库

3.4.5 NoSQL的新发展

- 从非关系型数据库到NewSQL
- NewSQL是对各种新型可扩展、高性能数据库的简称
- NewSQL不仅具有非关系型数据库对海量数据的存储管理能力，还保持了传统数据库的ACID和SQL等特性



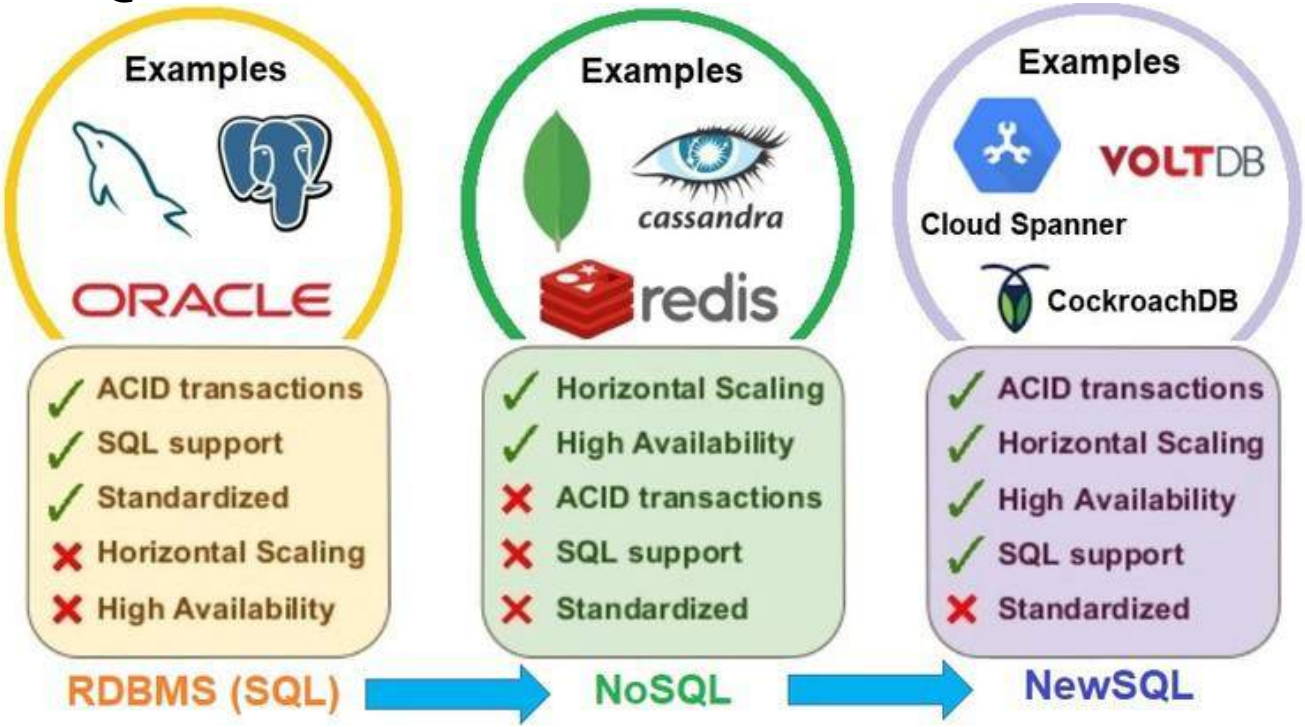
3.4 非关系型数据库

3.4.5 NoSQL的新发展

2014 年图灵奖得主迈克尔•斯通布雷克 (Michael Stonebraker) 对 NewSQL 的描述是：“关系数据库界面，支持事务处理 ACID 特性及并行控制，单一结点高性能，高可扩展结构。”

3.4 非关系型数据库

3.4.5 NoSQL的新发展



课外延展阅读


第一阶段
ORACLE DATABASE
Microsoft SQL Server
IBM DB2
PostgreSQL
关系数据库
（单机）
无所不能
第二阶段
NoSQL
HDFS
HBASE
cassandra
mongoDB
先做减法
提高性能
再做加法
拓展功能
MapReduce
Spark
Pig
TEZ
分布式计算框架
第三阶段
HIVE on TEZ
Google
Spanner
Spark SQL
NewSQL
融合回归

课外延展阅读

- Google 的“三驾马车”
  - 为大规模数据处理优化的 GFS 文件系统
  - 高容错的并行计算引擎 MapReduce
  - NoSQL 数据库鼻祖 BigTable

3.5 云数据库

- 概述
- 典型的云数据库
- 云数据库产品

3.5 云数据库

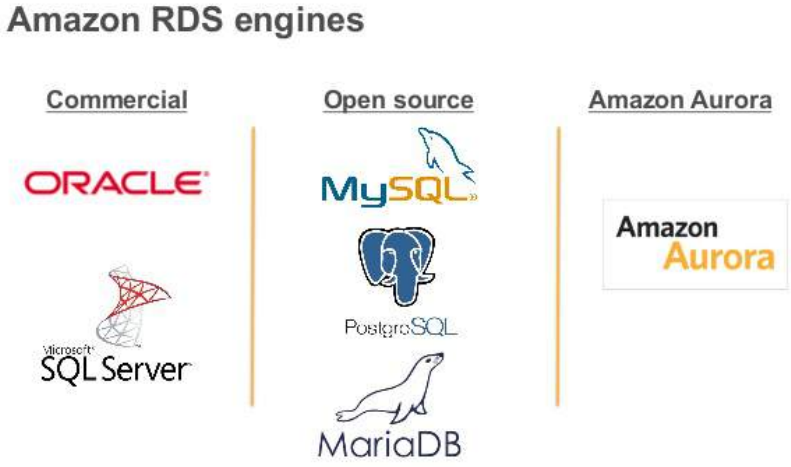
3.5.1 概述

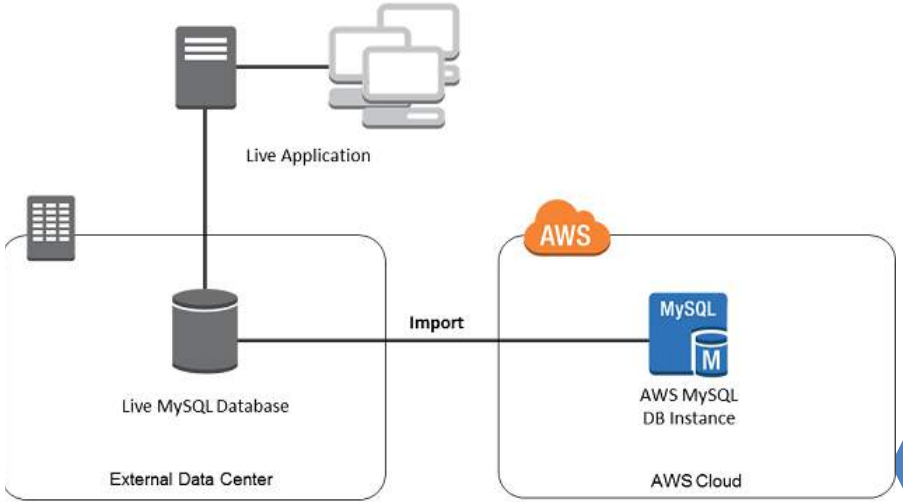
- 被优化或部署到虚拟环境中的，本身以云计算服务的方式对外提供服务的数据库。
- 云数据库具有多项优势：
  - 部署快捷
  - 可靠性高
  - 成本低

3.5 云数据库

3.5.2 典型的云数据库

- Amazon RDS
  - 在底层数据库上构建了中间层，中间层进行简单的转发请求，底层连接各种数据库。

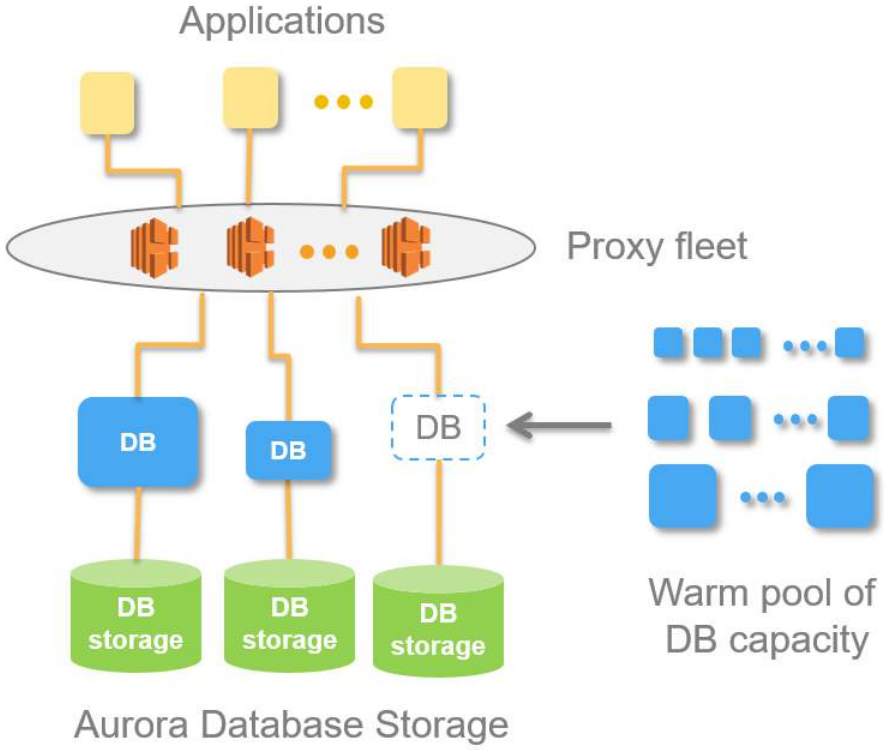




3.5 云数据库

3.5.2 典型的云数据库

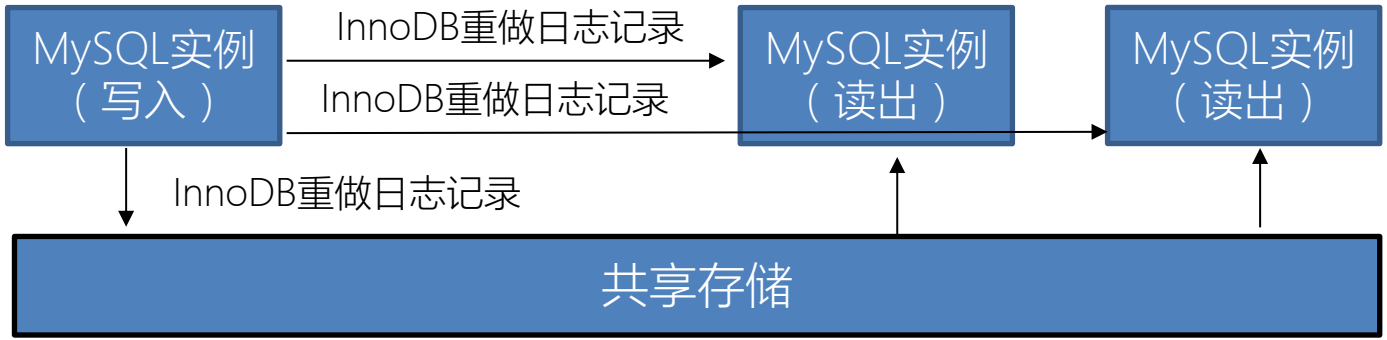
- Amazon Aurora
  - 在MySQL前端实现了分布式共享存储层，可将负载均摊到前端的各个MySQL实例，因此可以实现兼容性。
  - 对于大数据量、复杂查询的支持比较弱。



3.5 云数据库

3.5.2 典型的云数据库

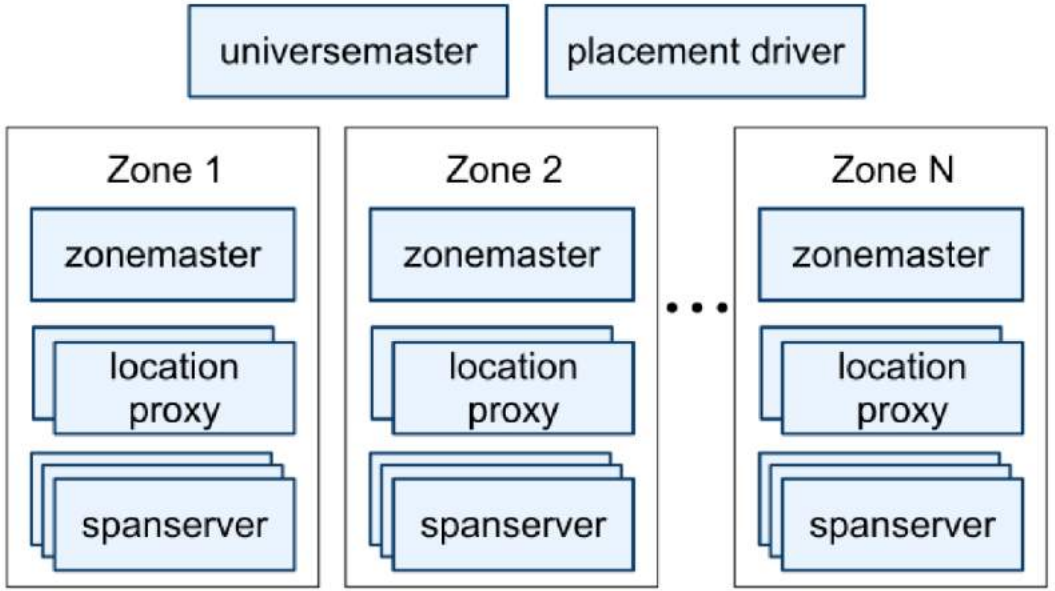
- Google Cloud Big Table
  - 是一种{Key: 二维表格结构}，和HBase兼容
  - 是一种主/从的层次结构，因此具有强一致性



3.5 云数据库

3.5.2 典型的云数据库

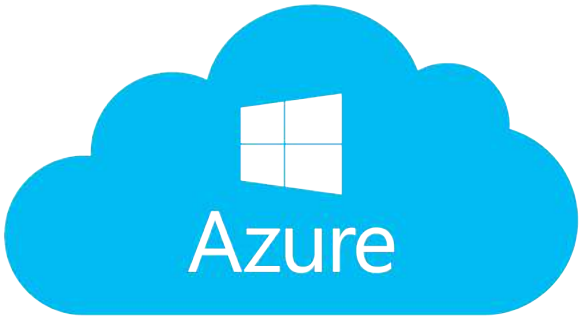
- Google Spanner
  - 一种主从的架构
  - 历史数据的非阻塞读取
  - 事务都是无锁只读的
  - 原子策略更改保证安全性
  - 支持外部一致的分布式事务（Paxos算法实现）



3.5 云数据库

3.5.3 云数据库产品

- 服务提供商通过云技术推出可在公有云托管数据库的方法，实现DBaaS (数据库即服务)



3.6 大数据的SQL查询引擎

- 概述
- Phoenix
- Hive
- Apache Drill和Presto
- Cloudera Impala

3.6 大数据的SQL查询引擎

3.6.1 概述

- SQL是一种结构化查询语言，可设计成为大数据的访问和查询接口，服务于非关系数据库。
- 大数据SQL查询引擎用来处理大规模数据，运行在分布式系统的上层。

3.6 大数据的SQL查询引擎

3.6.2 Phoenix

- Phoenix是构建在HBase上的一个SQL层，可以使用JDBC APIs来创建表、插入数据和对HBase数据进行查询。
- Phoenix引擎工作方式
  - 创建关联视图
  - 创建关联表
- Phoenix性能
  - 简单查询：毫秒级
  - 百万级别的行数：秒级

JDBC(java database connectivity)驱动程序是对JDBC规范完整的实现，它的存在在JAVA程序与数据库系统之间建立了一条通信的渠道。

3.6 大数据的SQL查询引擎

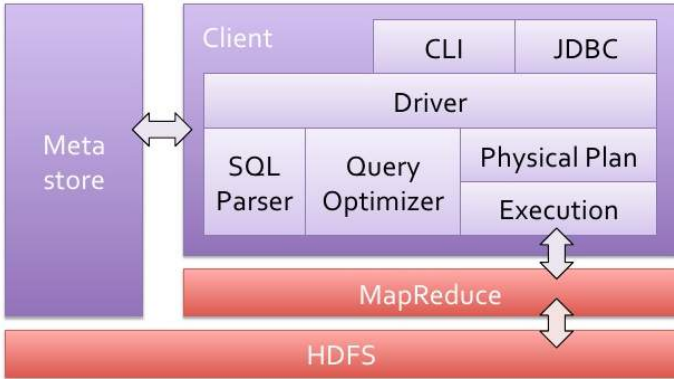
3.6.2 Phoenix

- Phoenix特性：
  - 嵌入式的JDBC驱动
  - 可通过多部行键或者键/值单元对列进行建模
  - 完善的查询支持
  - DDL支持, DML支持
  - 版本化的模式仓库
  - 通过客户端的批处理实现的有限事务支持
  - 符合ANSI SQL标准

3.6 大数据的SQL查询引擎

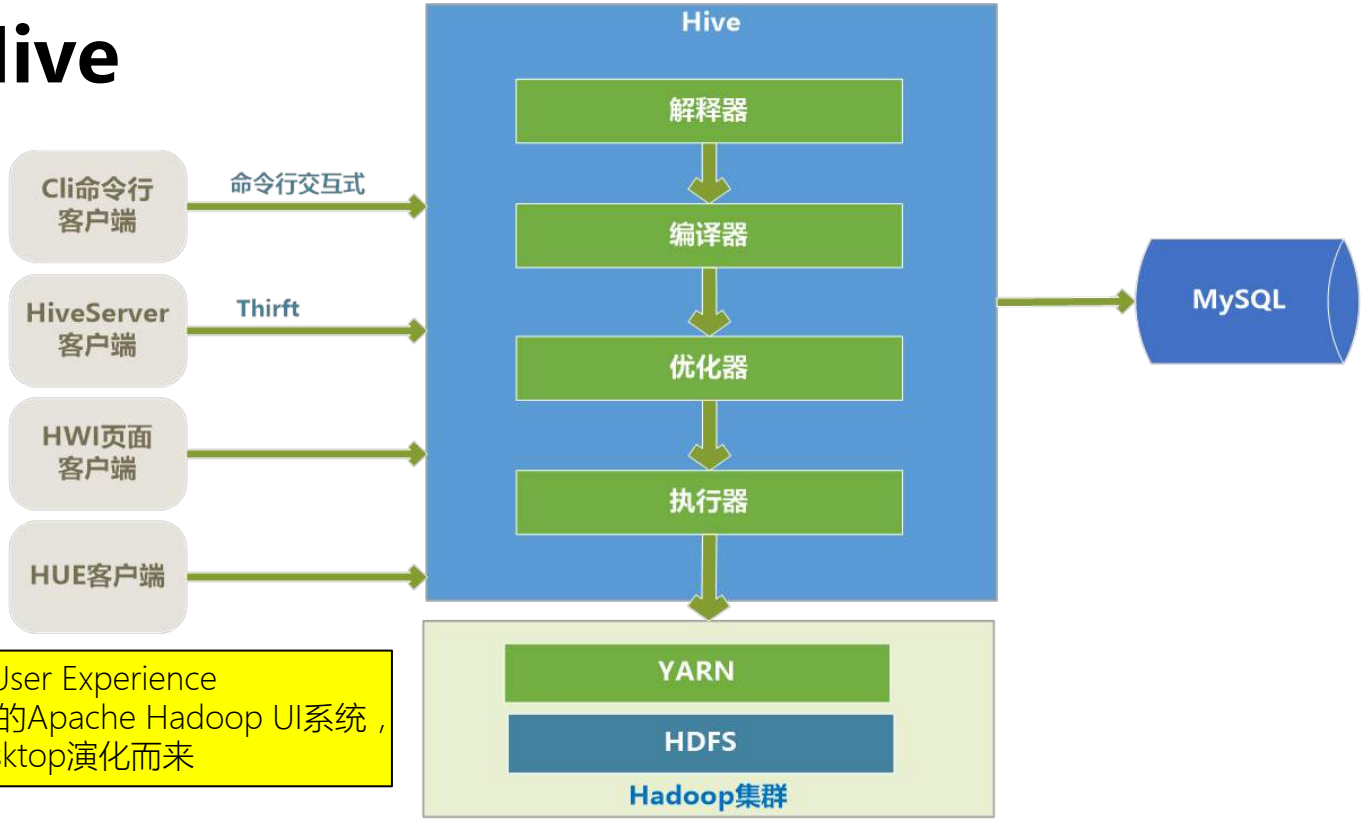
3.6.3 Hive

- Facebook 的工程师在2007年介绍Hive，并在2008年将代码捐献给Apache 软件基金会。
- 2010年9月，Hive 成为Apache 顶级项目。
- Apache Hive 是Hadoop 生态系统中的第一个SQL 框架，用来处理结构化数据的数据仓库基础工具。



3.6 大数据的SQL查询引擎

3.6.3 Hive

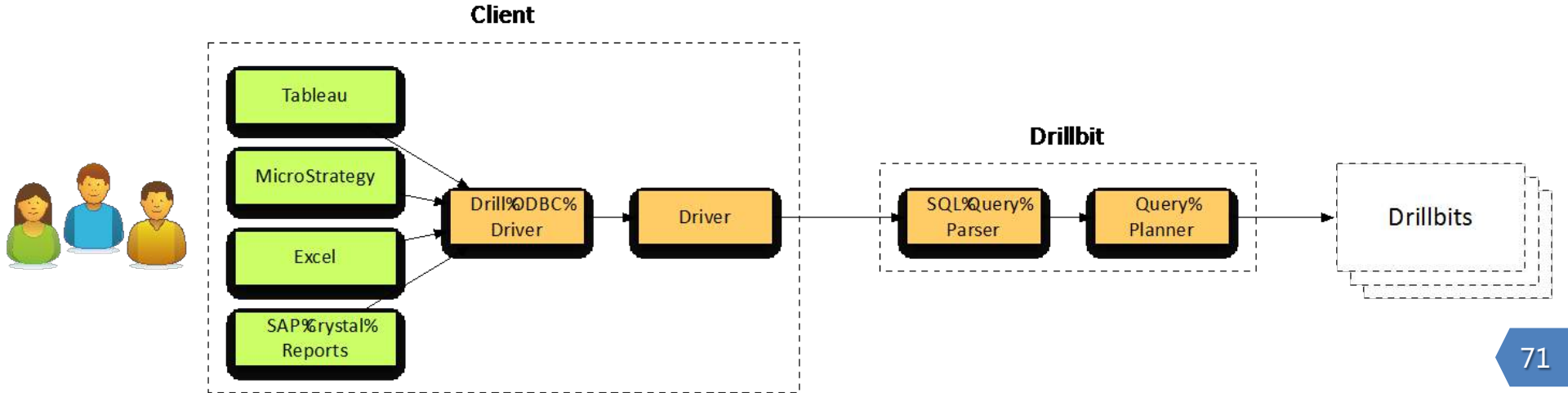


HUE=Hadoop User Experience
Hue是一个开源的Apache Hadoop UI系统，由Cloudera Desktop演化而来

3.6 大数据的SQL查询引擎

3.6.4 Apache Drill和Presto

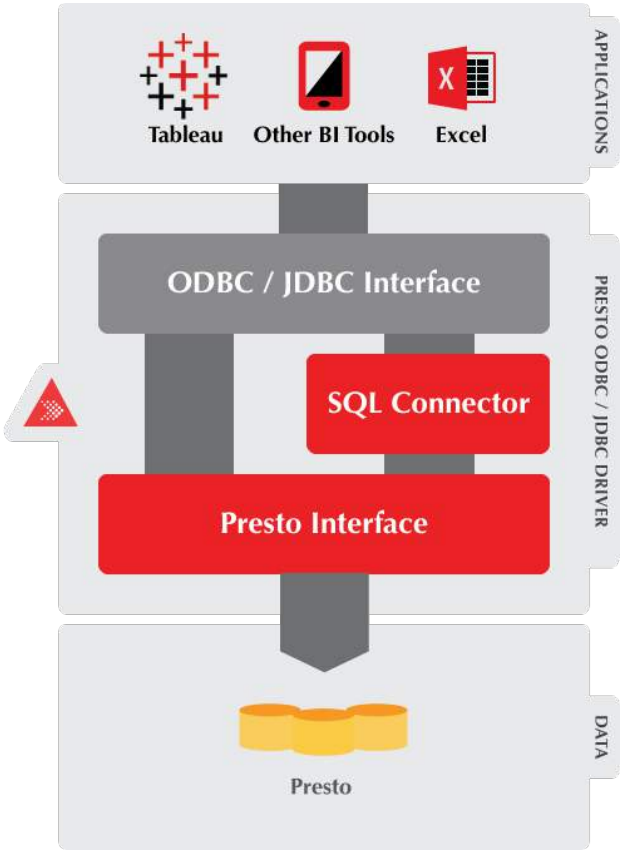
- Apache Drill是一个低延迟的分布式海量数据交互式查询引擎，支持本地文件、HDFS、Hive、HBase等后端存储，支持Parquet、JSON、CSV、TSV、PSV等数据格式。



3.6 大数据的SQL查询引擎

3.6.4 Apache Drill和Presto

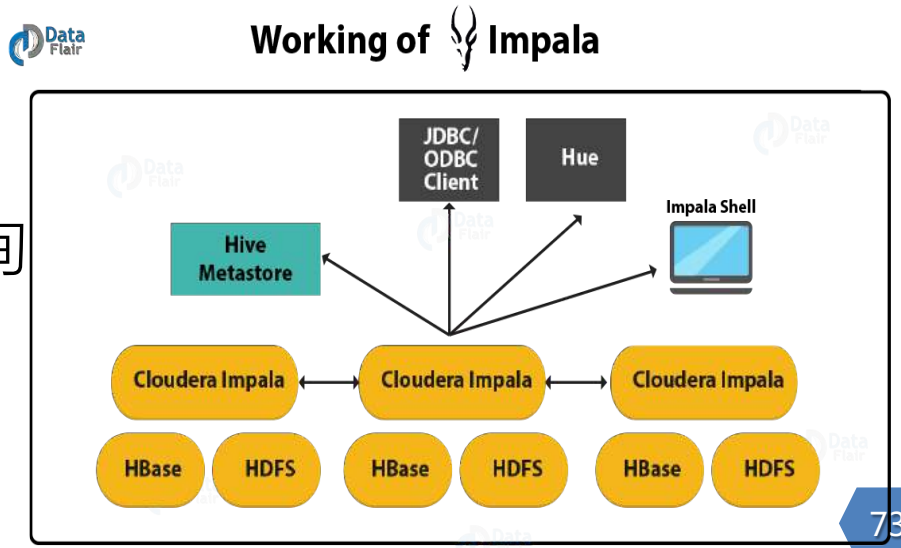
- Presto是一个用于查询分布在一个或者多个不同数据源的分布式SQL查询引擎。
  - 每个用户最多能同时运行5个查询
  - big查询同时只能运行一个
  - 最多能同时运行10个pipeline来源的查询
  - 最多能同时运行100个非big查询



3.6 大数据的SQL查询引擎

3.6.5 Cloudera Impala

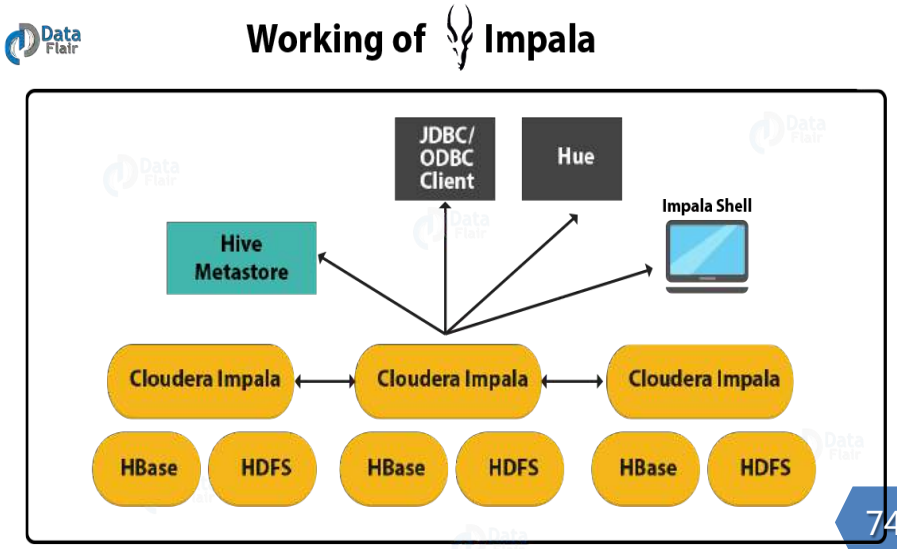
- Cloudera Impala提供对Hadoop文件格式的高性能、低延迟SQL查询，是一个可以提供直接查询互动SQL的分布式查询引擎。
  - 提供SQL界面
  - 在Hadoop上实现交互查询
  - 在集群环境中进行分布式查询
  - 避免modeling和ETL开销



3.6 大数据的SQL查询引擎

3.6.5 Cloudera Impala

- 分布式查询引擎（由Query Planner、Query Coordinator和Query Exec Engine三部分组成）
- 优势：
  - 查询速度快
  - 灵活性高
  - 易整合
  - 可伸缩性



3.7 小结

- 首先详细介绍了分布式文件系统的基本概念、常见的分布式文件系统、分布式文件系统的关键技术。
- 其次，详细介绍了 HBase 数据库的基本概念和运行机理。
- 然后，详细介绍了非关系型数据库的概念及其相关基础理论。
- 再然后，在介绍云数据库基本概念的基础上，侧重描述了云数据库的结构及典型产品。
- 最后，介绍了常见的支持大数据 SQL 查询的引擎。

大数据之思维的变革

- 一旦完成了数据本身的目的之后，数据就没有用处了吗？
- 在飞机降落之后，票价数据就没用了吗？
- 一个检索命令完成之后，检索数据就没用了吗？
- 用户完成购物后，订单数据就没用了吗？