大数据基础与应用
BIG DATA FOUNDATION AND APPLICATION

大数据开源平台与工具

北京理工大学计算机学院　王一拙

2019年1月

内容提要

- 数据采集与清洗
- 数据存储与管理
- 数据处理与分析
- 资源管理与调度

2. 数据存储与管理

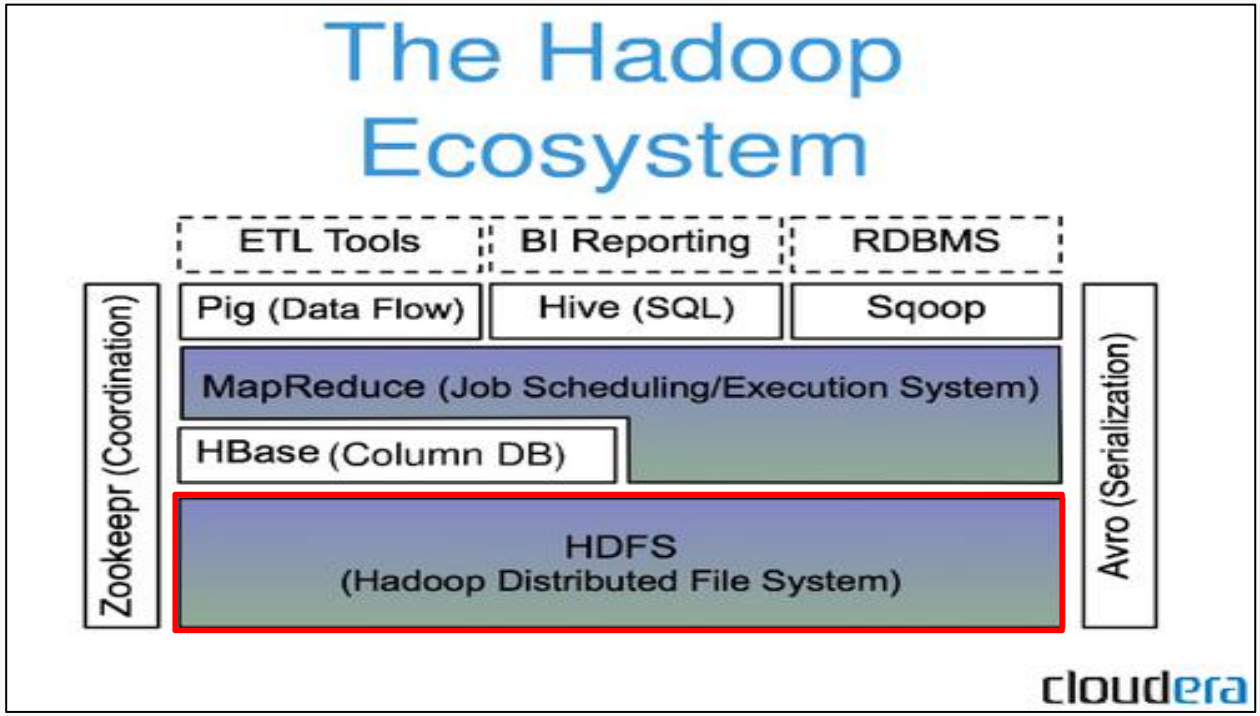
- 分布式文件系统HDFS
- 分布式数据库Hbase
- 数据仓库Hive
- NoSQL数据库技术

HDFS

- HDFS概述
- HDFS架构
- HDFS读写过程
- HDFS操作

HDFS概述

- Hadoop Distributed File System
  - 基于Google发布的GFS设计开发
  - 运行在通用硬件上的分布式文件系统
  - Hadoop应用程序的主要存储系统



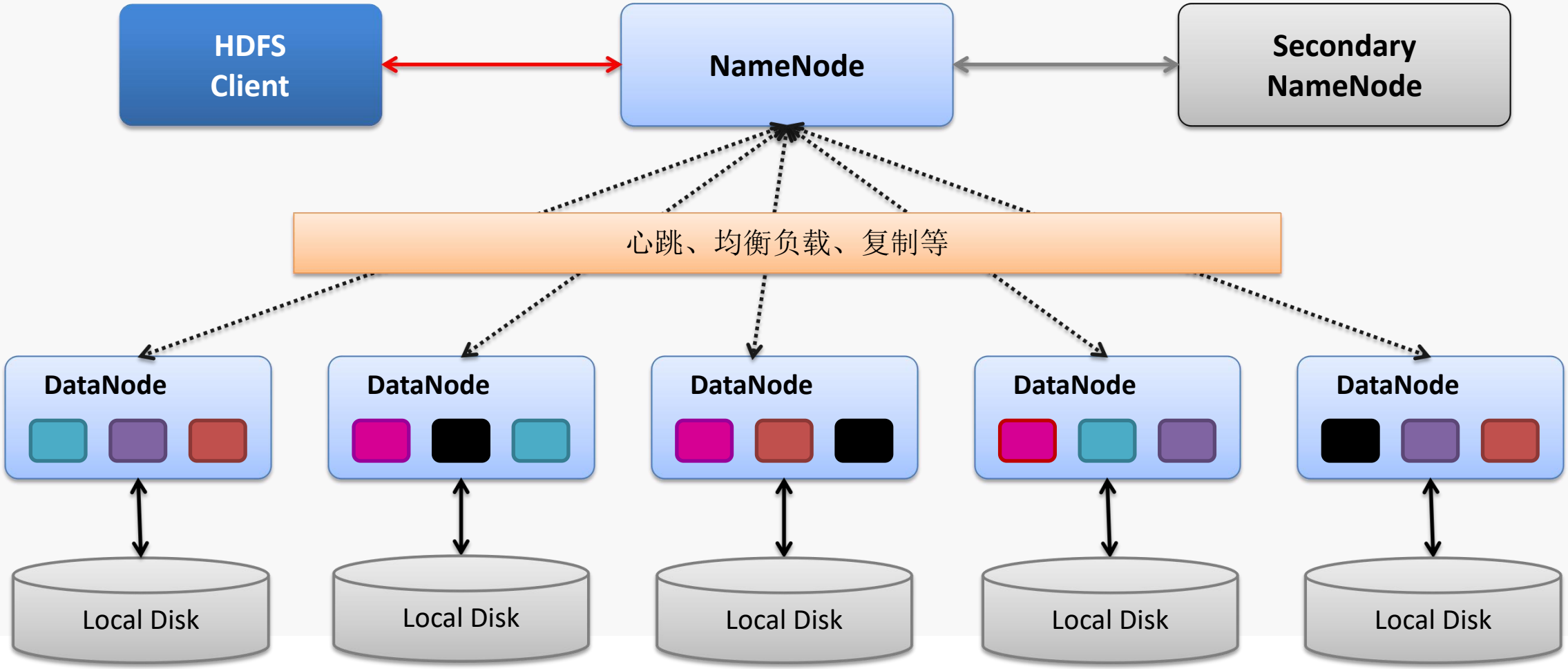
HDFS特性

- 大文件存储
  - 支持存储TB-PB级别的数据
- 容错能力
  - 硬件故障是常态，而不是异常
- 异构和可扩展性
  - 能够部署在不同的硬件设备上，能够水平扩展
- 流式数据访问
  - 适合批量处理，而不是用户交互式访问
  - 重点是在数据吞吐量，而不是数据访问的反应时间
- 简单的一致性模型
  - 一次写入多次读取
- 数据局域性
  - 移动计算要比移动数据更划算

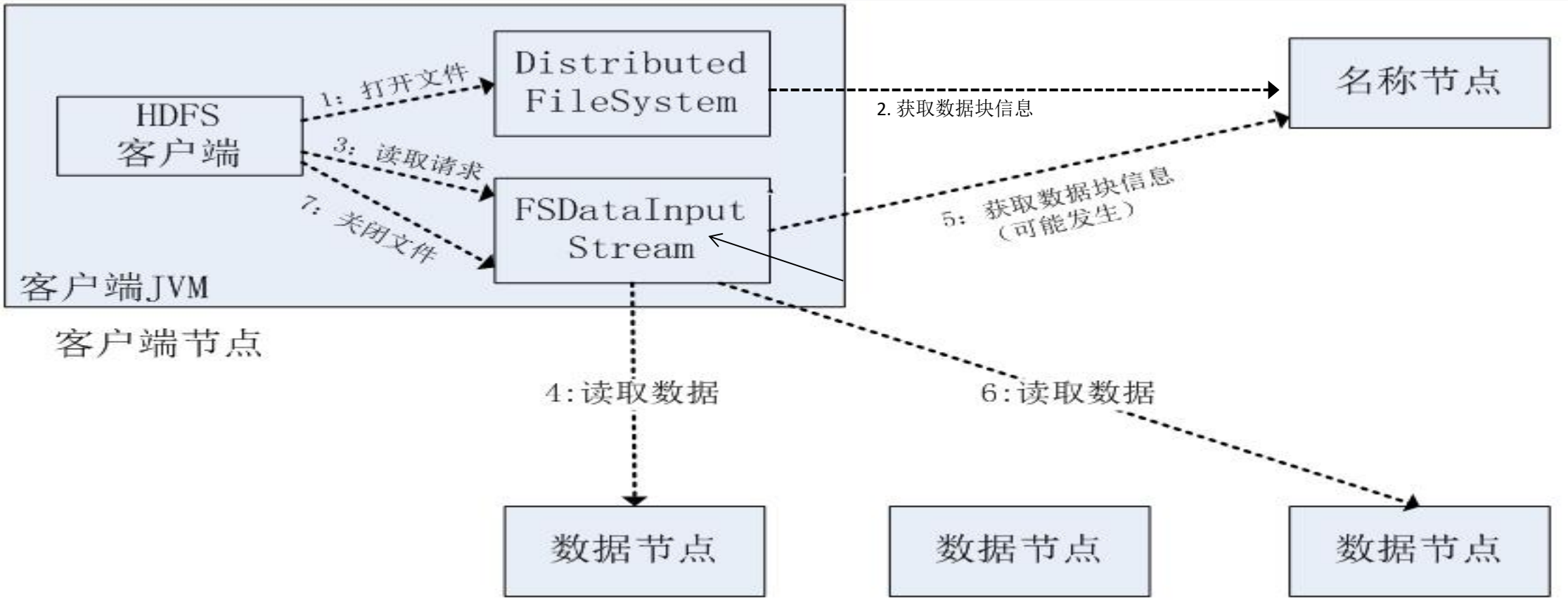
HDFS不适用的场景

- 低时间延迟数据访问的应用，例如几十毫秒范围
- 大量小文件的存储
- 多用户写入，任意修改文件

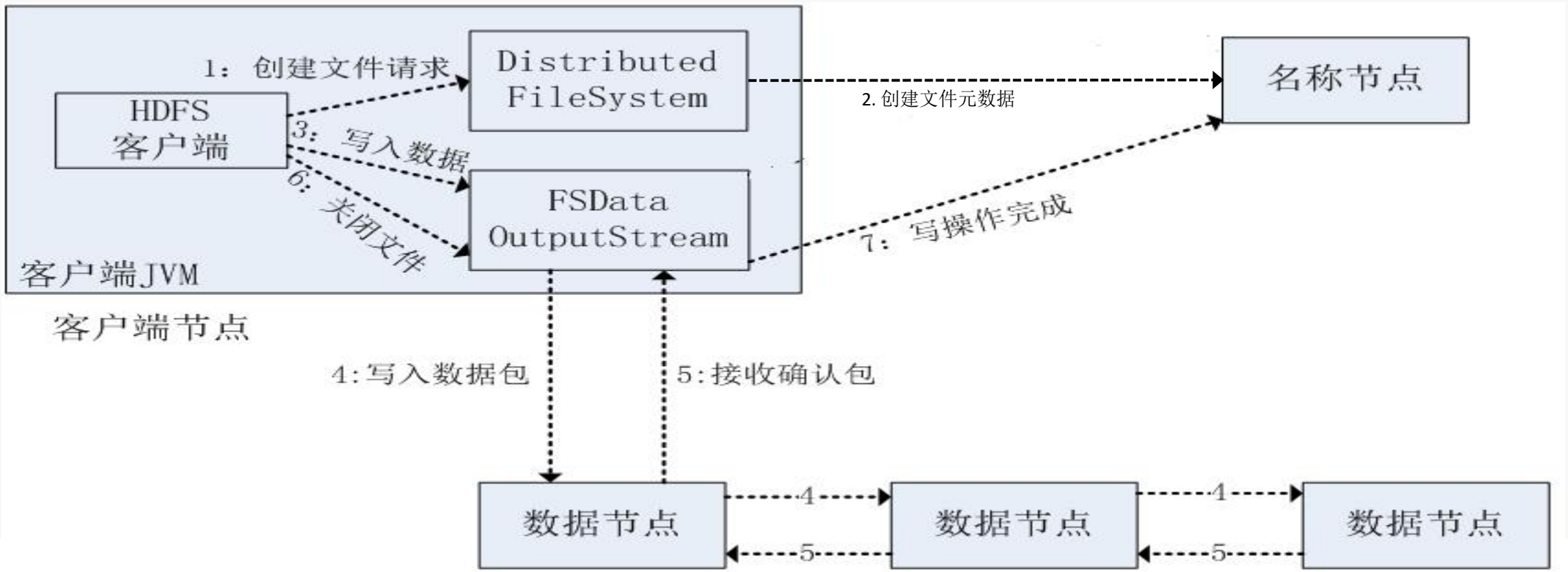
HDFS 架构



读数据的过程



写数据的过程



HDFS Configuration

HDFS Defaults

- Block Size - 64 MB
- Replication Factor - 3
- Web UI Port - 50070

HDFS conf file – /etc/hadoop/conf/hdfs-site.xml

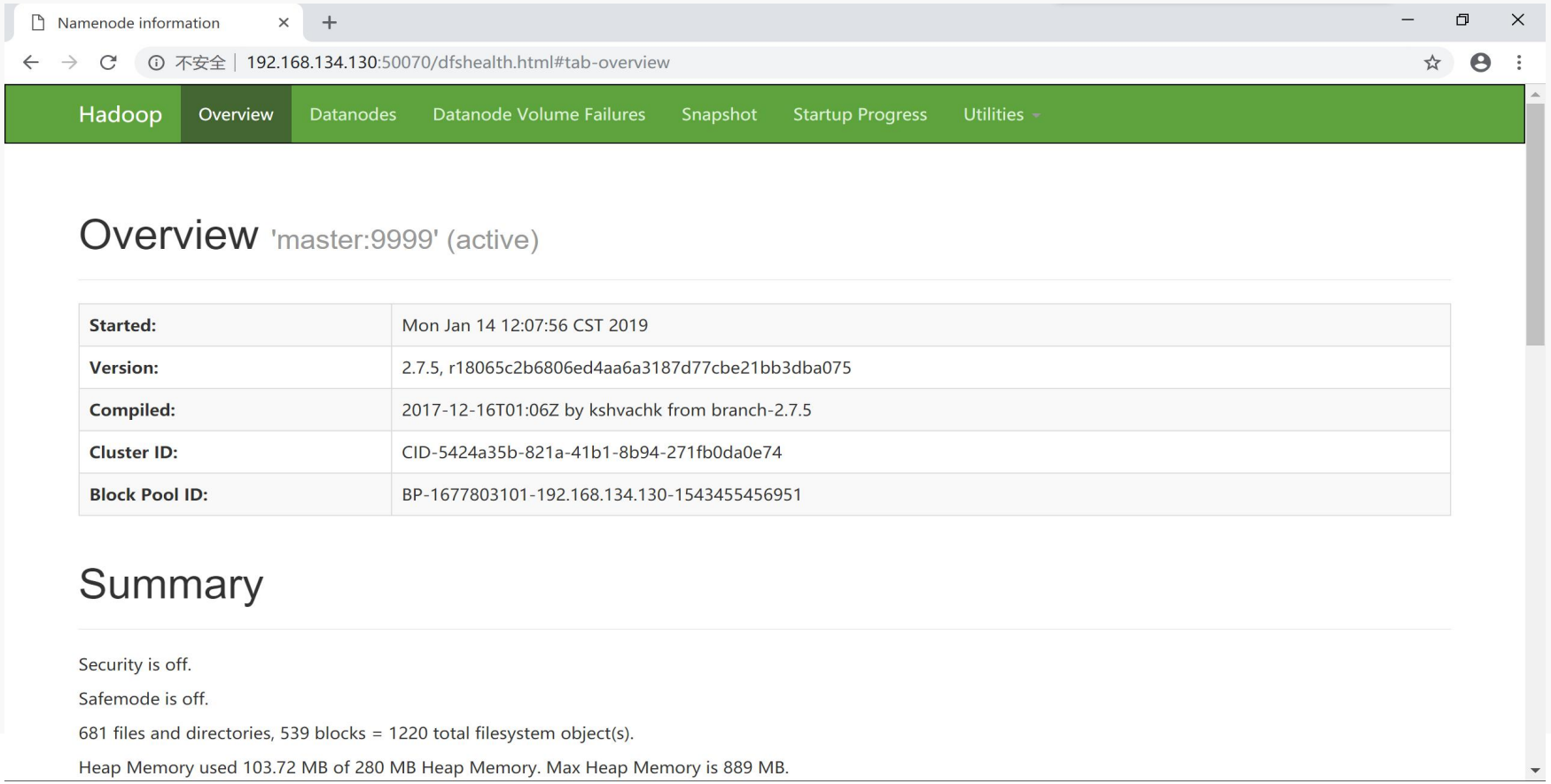
```
<property>
   <name>dfs.blocksize</name>
   <value>268435456</value>
</property>

<property>
   <name>dfs.replication</name>
   <value>3</value>
</property>

<property>
   <name>dfs.namenode.name.dir</name>
   <value>/home/hadoop-frank/bigdata/dfs/name</value>
</property>

<property>
   <name>dfs.datanode.data.dir</name>
   <value>/home/hadoop-frank/bigdata/dfs/data</value>
</property>
```

HDFS支持接口

接口类别	接口举例	接口说明
JAVA	mkdirs(Path f)	通过该接口可在HDFS上创建文件夹，其中f为文件夹的完整路径。
	create(Path f)	通过该接口可在HDFS上创建文件，其中f为文件的完整路径。
HTTP	curl -L --negotiate -u : "http://<HOST>:<PORT>/webhdfs/v1/<PATH>?op=OPEN"	打开并读取HDFS文件内容。
	curl -i -X DELETE --negotiate -u : "http://<host>:<port>/webhdfs/v1/<path>?op=DELETE"	删除指定的文件。
SHELL	hdfs dfs [COMMAND [COMMAND_OPTIONS]]	在HDFS文件系统上运行filesystem命令。
	hdfs fsck <path> [COMMAND_OPTIONS]	运行HDFS文件系统检查工具。

HDFS的Web界面

在配置好Hadoop集群之后，可以通过浏览器登录“http://[*NameNodeIP*]:50070”访问HDFS文件系统



HDFS – Shell 命令

User Commands

hdfs dfs – runs filesystem commands on the HDFS

hdfs fsck – runs a HDFS filesystem checking command

Administration Commands

hdfs dfsadmin – runs HDFS administration commands

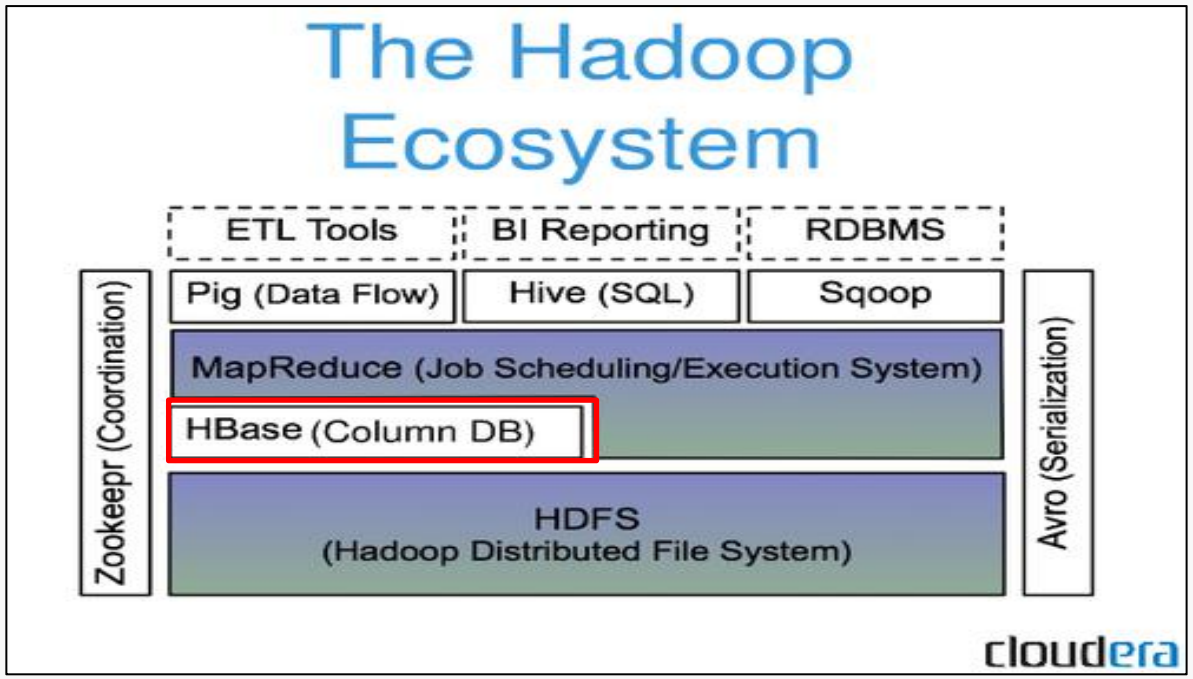
```
[hadoop-frank@master ~]$ hadoop fs
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALM
        [-chown [-R] [OWNER][:[GROUP]] PATH..
        [-copyFromLocal [-f] [-p] [-l] <local
        [-copyToLocal [-p] [-ignoreCrc] [-crc
        [-count [-q] [-h] <path> ...]
        [-cp [-f] [-p | -p[topax]] <src> ...
        [-createSnapshot <snapshotDir> [<snap
        [-deleteSnapshot <snapshotDir> <snaps
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] <path> ...]
        [-expunge]
        [-find <path> ... <expression> ...]
        [-get [-p] [-ignoreCrc] [-crc] <src>
        [-getfacl [-R] <path>]
        [-getfattr [-R] {-n name | -d} [-e er
        [-getmerge [-nl] <src> <localdst>]
        [-help [cmd ...]]
        [-ls [-d] [-h] [-R] [<path> ...]]
        [-mkdir [-p] <path> ...]
```

HBase

- 概述
- 数据模型
- 体系架构
- 操作

Hbase概述

- 开源Apache项目
- 分布式的非关系数据库系统
- 基于HDFS的列式数据存储
- 源于Google BigTable技术
- 用Java实现
- 一致性和分区容错
- 支持大数据集(TB to PB)
- 对HDFS低延迟的随机读写
- 广泛的应用领域
  - Facebook, Twitter, Adobe, Mozilla, Yahoo!, Trend Micro……

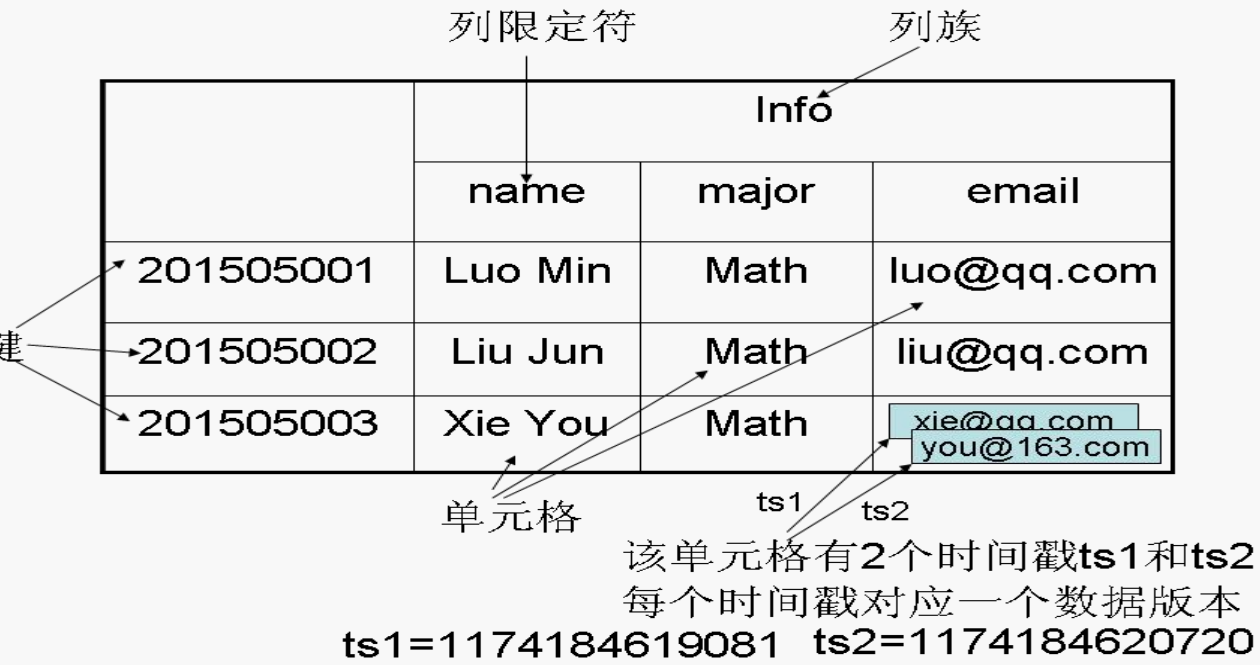


HBase与RDBMS对比

	HBase	RDBMS
数据类型	只有字符串	丰富的数据类型
硬件环境	和Hadoop一样，可部署在大量廉价的PC Server上	昂贵的企业集群系统
数据操作	简单的增删改查	各种各样的函数，表连接
错误容忍	单个节点的错误基本不影响整体性能	高可靠性，宕机恢复成本高
存储模式	基于列存储	基于表格结构和行存储
数据保护	更新后旧版本仍然会保留	替换
可伸缩性	轻易的进行增加节点，可扩展性高	需要中间层

HBase数据模型

- 表：HBase采用表来组织数据，表由行和列组成，列划分为若干个列族
- 行：每个表由若干行组成，每个行由行键（row key）来标识
- 列族：一个HBase表被分组成许多“列族”（Column Family）的集合
- 列限定符：列族里的数据通过列限定符（Qualifier）来定位
- 单元格：通过行、列族和列限定符确定一个“单元 行键格”（cell）
- 时间戳：每个单元格都保存着同一份数据的多个版本，这些版本采用时间戳进行索引



HBase逻辑视图

RowKey	TimeStamp	ColumnFamily ：“contents”	ColumnFamily ：“anchor”	ColumnFamily：“mime”
“cn.edu.tsinghua.www”	t9	<html>a1</html>	anchor: pku.edu.cn=” PKU”	
	t7		anchor: ruc.edu.cn=” RUC”	
	t5	<html>c3</html>		mime:type=” text/html”
	t4	<html>b2</html>		
	t3	<html>d4</html>		

HBase的物理模型

- 列族contents物理视图

RowKey	TimeStamp	ColumnFamily ： “contents”
“cn.edu.tsinghua.www”	t9	<html>a1</html>
	t5	<html>c3</html>
	t4	<html>b2</html>
	t3	<html>d4</html>

HBase的物理模型

- 列族anchor物理视图

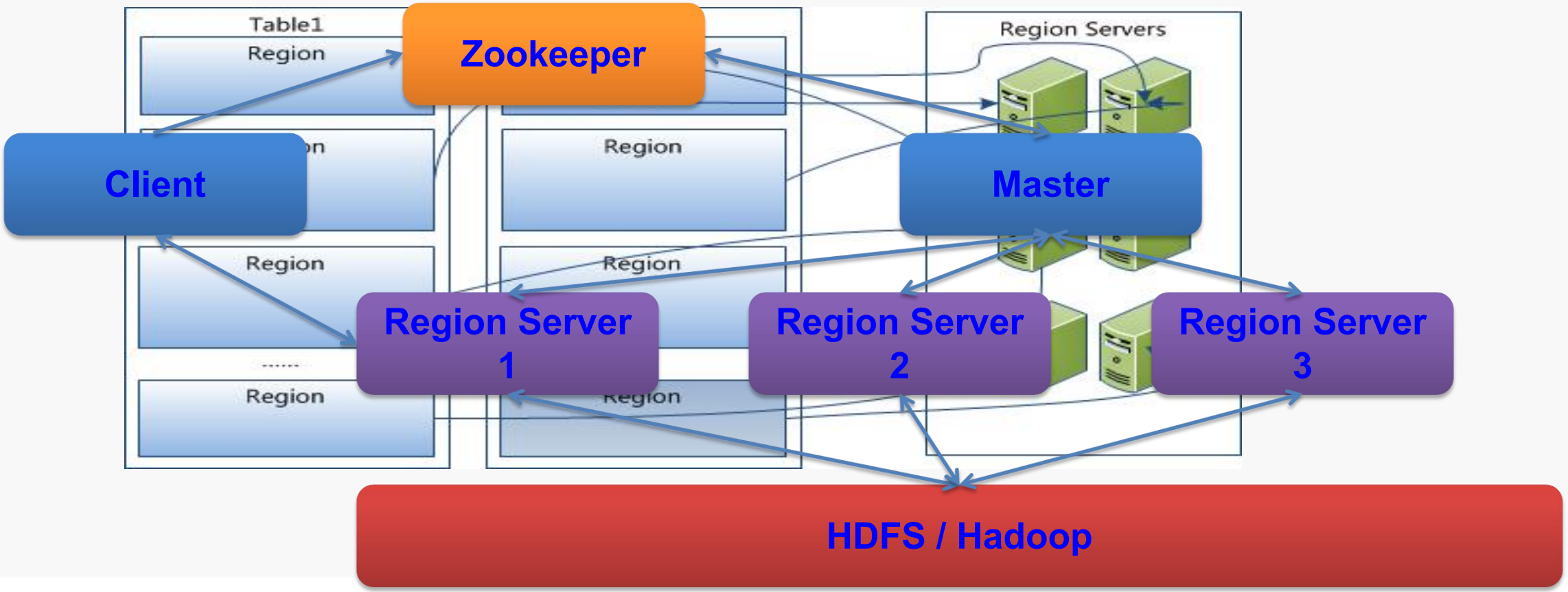
RowKey	TimeStamp	ColumnFamily ：“anchor”
“cn.edu.tsinghua.www”	t9	anchor: pku.edu.cn=” PKU”
	t7	anchor: ruc.edu.cn=” RUC”

HBase的物理模型

- 列族mime物理视图

RowKey	TimeStamp	ColumnFamily ：“**mime**”
“cn.edu.tsinghua.www”	t5	mime:type=” text/html”

HBase Architecture



HBase访问接口

类型	特点	场合
Native Java API	最常规和高效的访问方式	适合Hadoop MapReduce作业并行批处理HBase表数据
HBase Shell	HBase的命令行工具，最简单的接口	适合HBase管理使用
Thrift Gateway	利用Thrift序列化技术，支持C++、PHP、Python等多种语言	适合其他异构系统在线访问HBase表数据
REST Gateway	解除了语言限制	支持REST风格的Http API访问HBase
Pig	使用Pig Latin流式编程语言来处理HBase中的数据	适合做数据统计
Hive	简单	当需要以类似SQL语言方式来访问HBase的时候

Hbase – Shell (commands)

Command	Description
list	Shows list of tables
create 'users', 'info'	Creates users table with a single column family name info.
put 'users', 'row1', 'info:fn', 'John'	Inserts data into users table and column family info.
get 'users', 'row1'	Retrieve a row for a given row key
scan 'users'	Iterate through table users
disable 'users' **drop 'users'**	Delete a table (requires disabling table)

CRUD explained

CREATE	=	PUT
READ	=	GET
UPDATE	=	PUT
DELETE	=	DELETE

进入**Hbase**的安装目录，启动**Hbase**
bin/start-hbase.sh
打开**shell**命令行模式
bin/hbase shell
关闭**Hbase**
bin/stop-hbase.sh

Hbase – Java API (examples)

Command	Description
Get	Get get = new Get(String.valueOf(uid).getBytes()); Result[] results = table.get(gets);

Put	Put p = new Put(Bytes.toBytes(""+user.getUid()));
	p.add(Bytes.toBytes("info"), Bytes.toBytes("fn"), Bytes.toBytes(user.getFirstName()));
	p.add(Bytes.toBytes("info"), Bytes.toBytes("ln"), Bytes.toBytes(user.getLastName()));
	table.put(p);

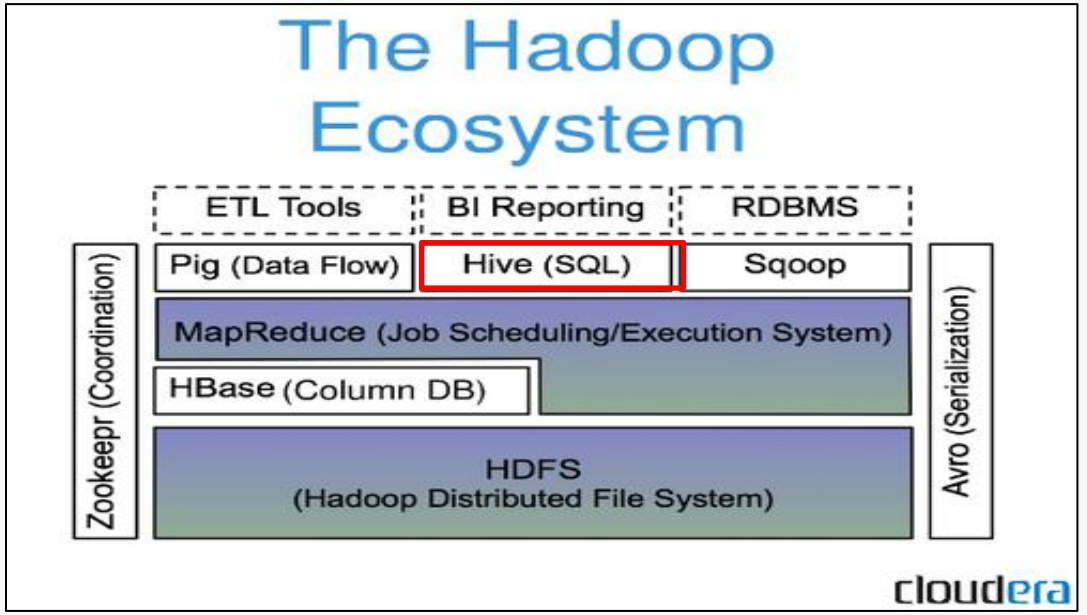
Delete (column, column family)	Delete d = new Delete(Bytes.toBytes(“”+user.getUid())); d.deleteColumn(Bytes.toBytes("info"), Bytes.toBytes("fn"), Bytes.toBytes(user.getFirstName()), **timestapmp1**);
Batch Operations	List of Get, Put or Delete operations
Scan	Iterate over a table. Prefer Range / Filtered scan. Expensive operation.

Hive

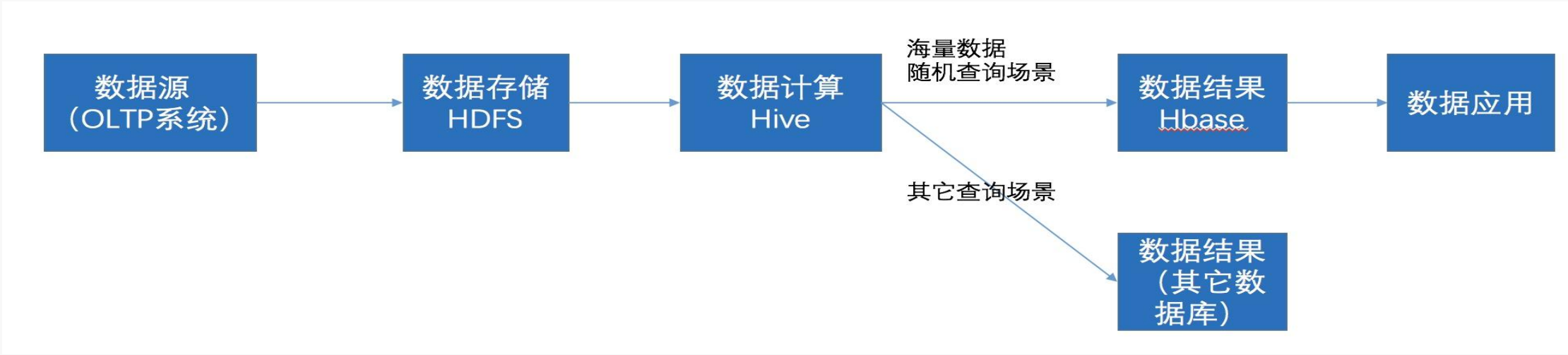
- 概述
- 数据模型和架构
- 操作

HIVE概述

- Hive是基于Hadoop的数据仓库软件
- 由Facebook开发
- 把SQL转换成MapReduce执行
- 提供类SQL的查询语言HiveQL（HQL）
- Hive shell是主要的操作方式



Hive和HBase协作关系



Hive的应用场景

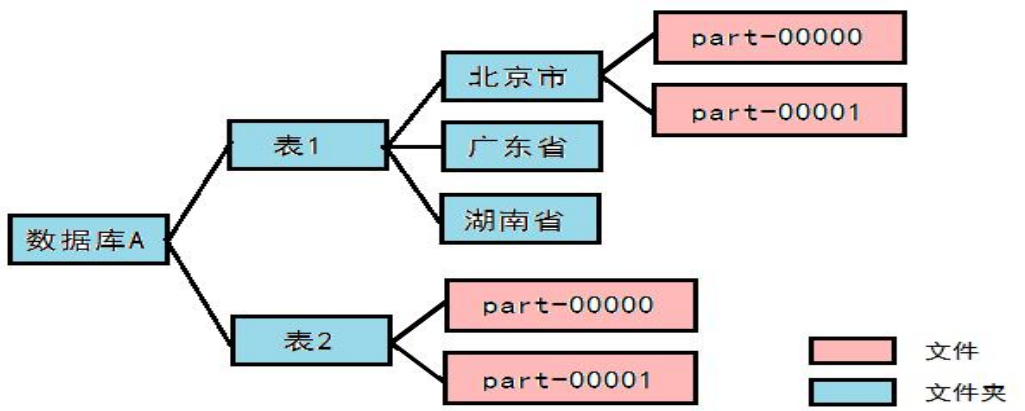
数据挖掘	➢用户行为分析 ➢兴趣分区 ➢区域展示
非实时分析	➢日志分析 ➢文本分析
数据汇总	➢每天/每周用户点击数 ➢流量统计
数据仓库	➢数据抽取 ➢数据转换 ➢数据加载

HIVE 数据模型

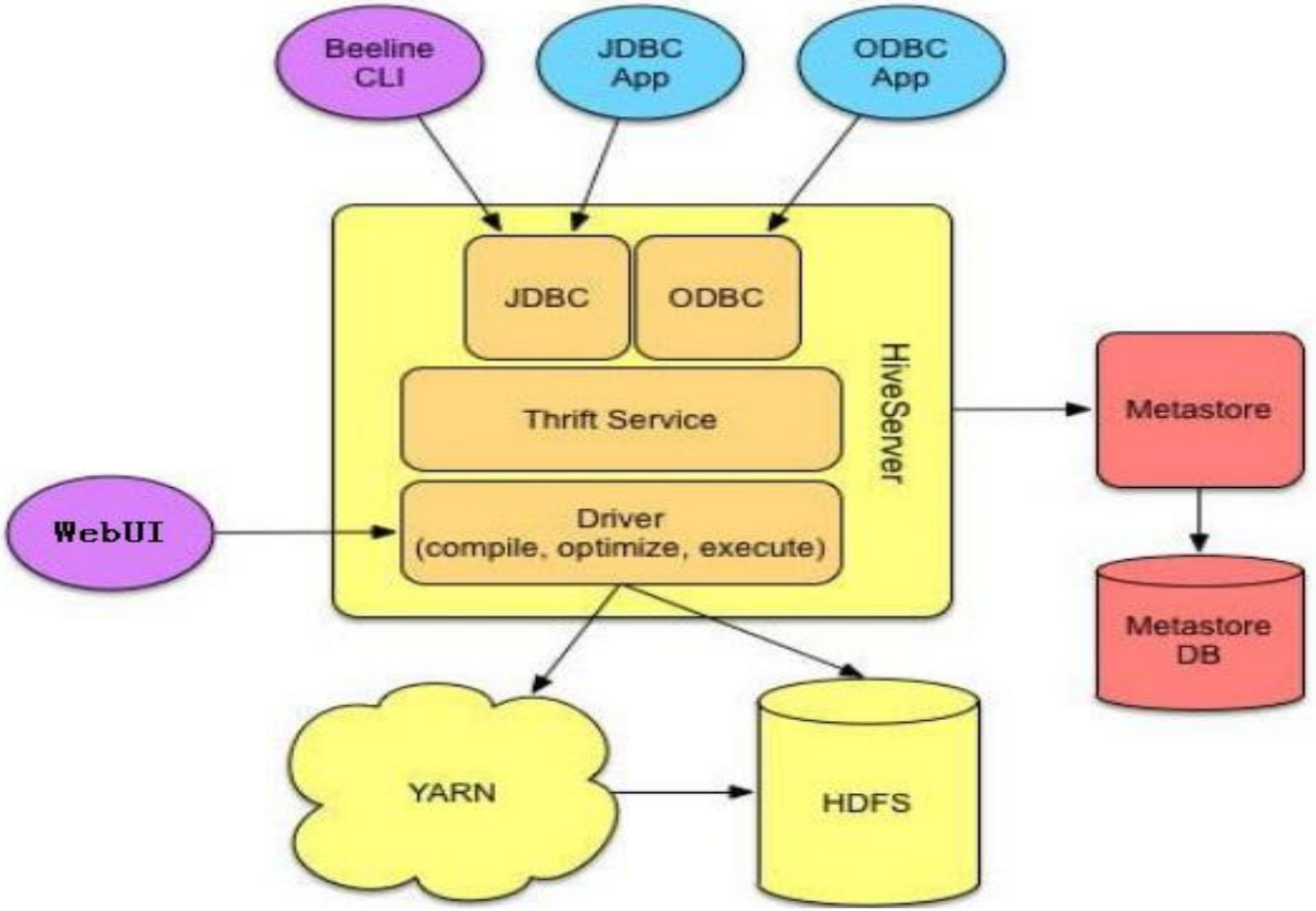
- 表（Tables）：所有数据存储在HDFS中的一个目录
  - Primitives: numeric, boolean, string and timestamps
  - Complex: Arrays, maps and structs

- Partitions:表可以按照某个字段的值划分分区
  - 对表格进行分区处理(Partition)，便于局部数据的查询操作，如按时间分区、按地域分区等，将具有相同性质的数据存储到同一磁盘块上，从而加快查询效率。

- Buckets: 分区数据可以进一步分成桶
  1. 进一步提升查询效率
  2. 使数据抽样更有效



Hive的架构



MetaStore : 提供元数据服务。

Driver : 管理HQL执行的生命周期，贯穿Hive任务整个执行期间。

Compiler : 编译HQL并将其转化为一系列的Map/Reduce任务。

Optimizer : 优化器，优化HQL生成的执行计划和MapReduce任务。

Executor : 执行Map/Reduce任务。

ThriftServer : 提供thrift接口，作为JDBC和ODBC的服务端，将Hive和其他应用程序集成起来。

Clients :为用户访问提供命令行接口Beeline和JDBC/ODBC接口。

Hive SQL介绍

DDL-数据定义语言

- 建表（库或视图），修改表、分区、列，删除表等等

DML-数据管理语言

- 数据导入
- 数据导出

DQL-数据查询语言

- 一般查询
- Group by, Order by, Cluster by, Join, Union
- 子查询

函数（内置函数/UDF/UDAF/UDTF)

Hive基本操作 – 数据库操作

```
（1）创建数据库
create database db_hive;
create database if not exists db_hive;
create database if not exists db_hive location '/db_hive.db';

（2）查看数据库
show databases;
show databases like 'db_hiv*';

（3）使用数据库
use db_hive;

（4）查看数据库字段格式
desc database db_hive;

（5）删除数据库
drop database db_hive;
drop database if exists db_hive;
```

Hive基本操作 – DDL之创建表

```
CREATE TABLE IF NOT EXISTS example.employee
(Id INT COMMENT ‘Employee id',
Department STRING,
Salary FLOAT  COMMENT 'Employee salary',
MapData Map<STRING,ARRAY<STRING>>,
ArrayData ARRAY<INT>,
StructData STRUCT<col1:STRING,col2:STRING>)
PARTITIONED BY (century STRING, year STRING)
CLUSTERED BY (Department) into 10 buckets
ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’
COLLECTION ITEMS TERMINATED BY '@'
MAP KEYS TERMINATED BY '$'
STORED AS TEXTFILE;
```

数据格式示例：1,Sales,6000.0,m1$a,1@2@3@4,c1@c2

查询结果：1 Sales 6000.0 {"m1":["a"]} [1,2,3,4] {"col1":"c1","col2":"c2"} 21 2012

Hive基本操作-DML

```
--从本地文件加载数据到Hive表
LOAD DATA LOCAL INPATH 'employee.txt' OVERWRITE INTO TABLE example.employee
partition (century='21',year='2012');

--从另一个表加载数据到Hive表
INSERT INTO TABLE company.person PARTITION(century= '21',year='2010')
SELECT id, name, age, birthday FROM company.person_tmp WHERE century= '21' AND
year='2010';

--导出数据到HDFS
EXPORT TABLE company.person TO '/department';

--从HDFS导入数据
IMPROT TABLE company.person FROM '/department‘;
```

Hive基本操作-DQL

```
--Group by having
 SELECT Department, avg(Salary) AS avg_sal FROM example.employee GROUP BY Department
   HAVING avg_sal >8000.0;

--Union ALL & sub-Query
 SELECT users.id, actions.date FROM ( SELECT uid, date FROM action_video WHERE date =
   '2018-06-03' UNION ALL SELECT uid, date FROM action_comment
  ) actions
 JOIN users ON (users.id = actions.uid)
 Limit 100;
```

NoSQL数据库技术

NoSQL简介

概念演变

Not only SQL

最初表示“反SQL”运动
用新型的非关系数据库取代关系数据库

现在表示关系和非关系型数据库各有优缺点
彼此都无法互相取代

谁在使用NoSQL数据库?

NoSQL数据库的优点

- 灵活的数据模型，能处理非结构化/半结构化的大数据
- 很好的容灾能力，数据能够被划分和备份：
  - 故障节点很容易被替换
  - 无单点故障问题
- 优良的可扩展性，容易通过添加新节点来简单的扩展
- 低廉的成本，大多数是开源且容易实现
- 读写性能高
  - 高并发的写操作性能
  - 快速key-value访问

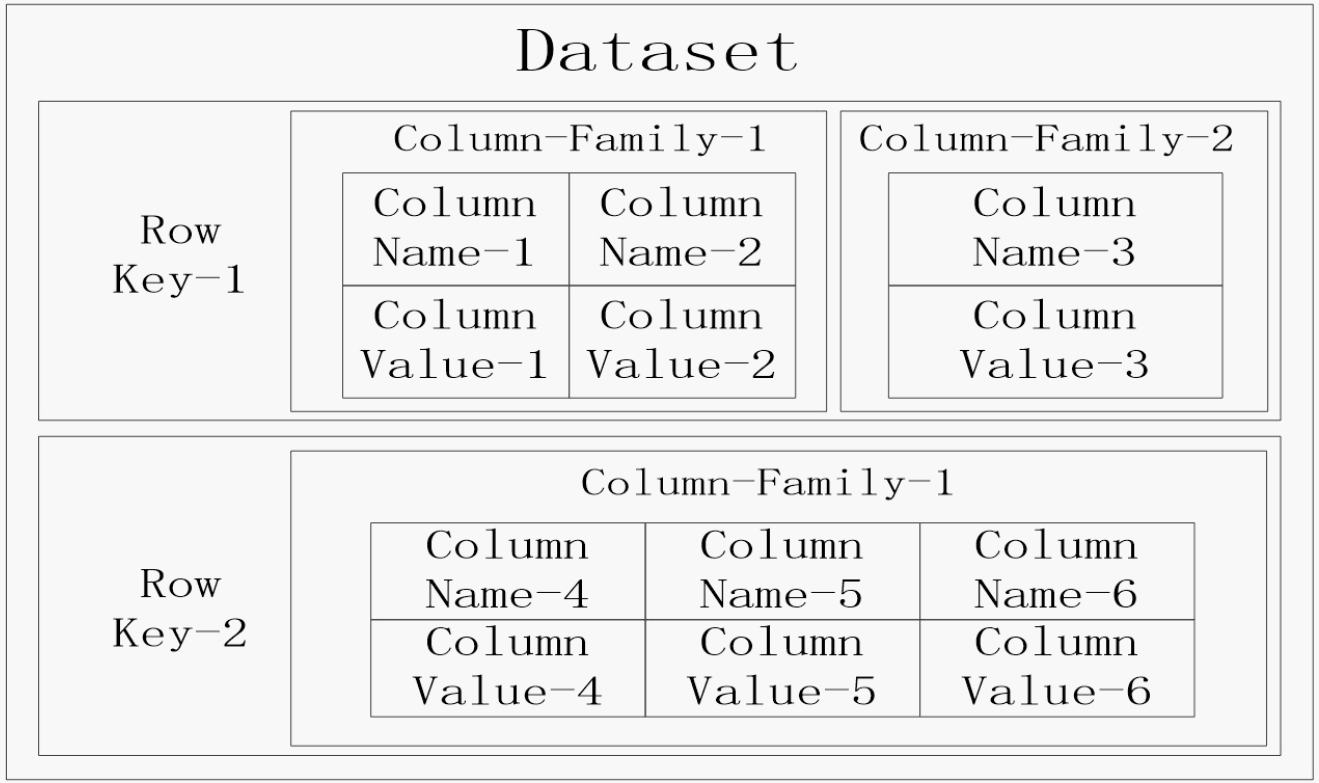
NoSQL数据库的缺点

- 不支持许多关系数据库的特性
  - 没有join，group by等操作
  - 没有跨分区的外键约束
- 不支持SQL这样的标准查询语
- 事务支持较弱或者不支持
- 没有关系数据库那么成熟，不容易和已有的支持SQL的应用融合

NoSQL的四大类型

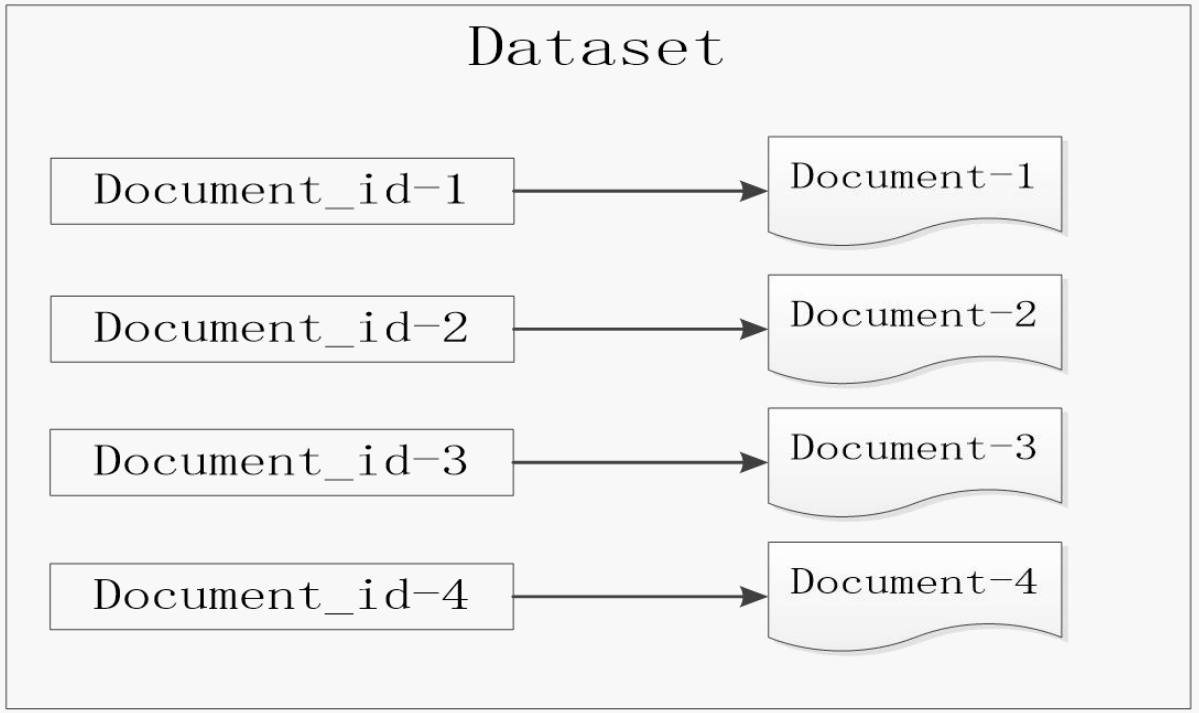
Key_1	Value_1
Key_2	Value_2
Key_3	Value_1
Key_4	Value_3
Key_5	Value_2
Key_6	Value_1
Key_7	Value_4
Key_8	Value_3

键值数据库

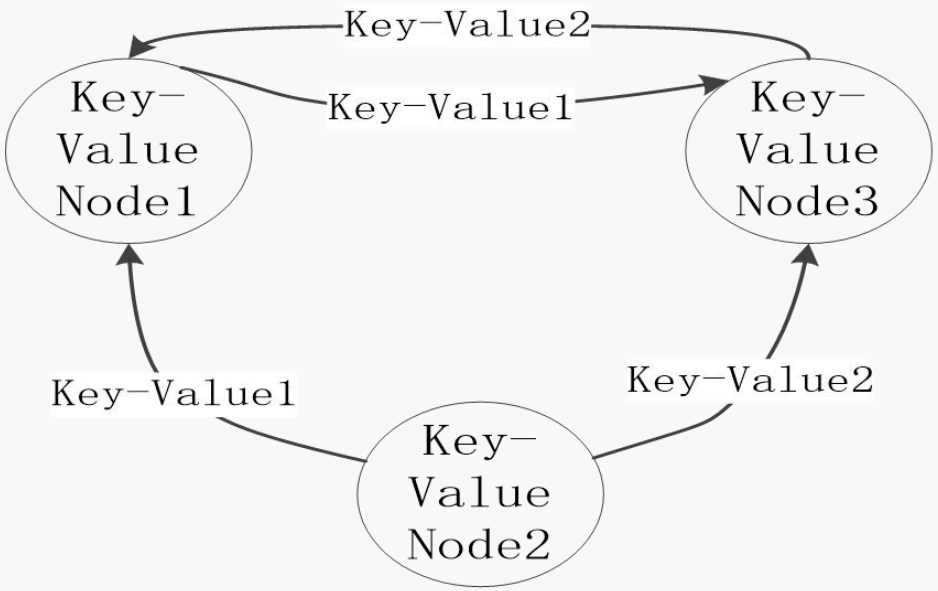


列族数据库

NoSQL的四大类型



文档数据库



图形数据库

NoSQL的四大类型

文档数据库	图数据库
 Couchbase MarkLogic mongoDB	 InfiniteGraph The Distributed Graph Database
键值数据库	**列族数据库**
   AEROSPIKE 	  HYPERTABLE INC  Cassandra  Amazon SimpleDB