1. HBase数据库介绍

1.1. HBase产生背景

自1970年以来，关系数据库用于数据存储和维护有关问题的解决方案。大数据的出现后，好多公司实现处理大数据并从中受益，并开始选择像Hadoop的解决方案。Hadoop使用分布式文件系统，用于存储大数据，并使用MapReduce来处理。Hadoop擅长于存储各种格式的庞大的数据，任意的格式甚至非结构化的处理。

Hadoop的特点：

> 对于任意格式的庞大数据集，Hadoop可以做到安全存储
> 但是对于需要在庞大数据集做针对于单条记录的增删改查是做不到的。

Hadoop的限制：

> Hadoop只能执行批量处理，并且只以顺序方式访问数据。这意味着必须搜索整个数据集，即使是最简单的搜索工作。当处理结果在另一个庞大的数据集，也是按顺序处理一个巨大的数据集。在这一点上，一个新的解决方案，需要访问数据中的任何点（随机访问）单元。

Hive的特点：

对于存储在HDFS上的结构化的数据，如果增加一些描述这些数据的元数据信息，那么我们可以把存储在HDFS上的数据抽象成一张二维表格，使用Hive进行各种Insert/Select操作。但是Hive还是天生不支持对于单条记录的增删改查，也不是设计用来做单条记录的增删改查的。

Hadoop随机存取数据库

应用程序，如HBase，Cassandra，CouchDB，Dynamo 和 MongoDB 都是一些存储大量数据和以随机方式访问数据的数据库。

总结：

（1）海量数据量存储成为瓶颈，单台机器无法负载大量数据
（2）单台机器IO读写请求成为海量数据存储时候高并发大规模请求的瓶颈
（3）随着数据规模越来越大，大量业务场景开始考虑数据存储横向水平扩展，使得存储服务可以增加/删除，而目前的关系型数据库更专注于一台机器

1.2. HBase简介

HBase 是 Hadoop Database 的简称，Hadoop Database，是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，利用 HBase 技术可在廉价 PC Server 上搭建起大规模结构化存储集群。HBase 项目是由 Powerset 公司的 Chad Walters 和 Jim Kelleman 在2006年末发起，根据 Google 的 Chang 等人发表的论文"**Bigtable：A Distributed Storage System for Strctured Data**"来设计的。

2007年10月发布了第一个版本。
2010年5月，HBase 从 Hadoop 子项目升级成 Apache 顶级项目。

1.3. HBase官网简介

官网：http://hbase.apache.org/

```
1、Apache HBase™ is the Hadoop database, a distributed, scalable, big data
store.

2、Use Apache HBase™ when you need random, realtime read/write access to your
Big Data.

3、This project's goal is the hosting of very large tables -- billions of rows X
millions of columns -- atop clusters of commodity hardware.

4、Apache HBase is an open-source, distributed, versioned, non-relational
database modeled after Google's Bigtable: A Distributed Storage System for
Structured Data by Chang et al. Just as Bigtable leverages the distributed data
storage provided by the Google File System, Apache HBase provides Bigtable-like
capabilities on top of Hadoop and HDFS.
```

HBase 是 BigTable 的开源（源码使用Java编写）版本。是 Apache Hadoop 的数据库，是建立在 HDFS 之上，被设计用来提供高可靠性、高性能、列存储、可伸缩、多版本的 NoSQL 的分布式数据存储系统，实现对大型数据的实时、随机的读写访问。

```
HBase依赖于HDFS做底层的数据存储，BigTable依赖Google GFS做数据存储
HBase依赖于MapReduce做数据计算，BigTable依赖Google MapReduce做数据计算
HBase依赖于ZooKeeper做服务协调，BigTable依赖Google Chubby做服务协调
```

与 Hadoop 一样，HBase 目标主要依靠横向扩展，通过不断增加廉价的商用服务器，来增加计算和存储能力。所以，HBase 是一个通过大量廉价机器解决海量数据的高速存储和读取的分布式数据库解决方案

1.4. HBase特点总结

以下五点是HBase这个NoSQL数据库的要点：

```
1、高并发，以扩展，解决海量数据集的随机实时增删改查
2、HBase本质依然是Key-Value数据库，查询数据功能很简单，不支持join等复杂操作(可通过Hive支持来实现多表join等复杂操作)
3、不支持复杂的事务，只支持行级事务
4、HBase中支持的数据类型：byte[]（底层所有数据的存储都是字节数组）
5、主要用来存储结构化和半结构化的松散数据。
```

HBase中的表的特点

```
1、大：一个表可以有上十亿行，上百万列
2、面向列：列可以灵活指定，面向列(族)的存储和权限控制，列(簇)独立检索。
3、稀疏：对于为空(null)的列，并不占用存储空间，因此，表可以设计的非常稀疏。
4、无严格模式：每行都有一个可排序的主键和任意多的列，列可以根据需要动态的增加，同一张表中不同的行可以有截然不同的列
```

HBase特点总结：

```
1、海量存储
HBase适合存储PB级别的海量数据，在PB级别的数据以及采用廉价PC存储的情况下，能在几十到百毫秒内返回数据。这与HBase的极易扩展性息息相关。正式因为HBase良好的扩展性，才为海量数据的存储提供了便利。

2、列式存储
这里的列式存储其实说的是列族存储，Hbase是根据列族来存储数据的。列族下面可以有非常多的列，列族在创建表的时候就必须指定。

3、极易扩展
HBase的扩展性主要体现在两个方面，一个是基于上层处理能力（RegionServer）的扩展，一个是基于存储的扩展（HDFS）。通过横向添加RegionSever的机器，进行水平扩展，提升HBase上层的处理能力，提升HBase服务更多Region的能力。
备注：RegionServer的作用是管理region、承接业务的访问，这个后面会详细的介绍通过横向添加DataNode的机器，进行存储层扩容，提升HBase的数据存储能力和提升后端存储的读写能力。

4、高并发
由于目前大部分使用HBase的架构，都是采用的廉价PC，因此单个IO的延迟其实并不小，一般在几十到上百ms之间。这里说的高并发，主要是在并发的情况下，HBase的单个IO延迟下降并不多。能获得高并发、低延迟的服务。

5、稀疏
稀疏主要是针对HBase列的灵活性，在列族中，你可以指定任意多的列，在列数据为空的情况下，是不会占用存储空间的。
```

1.5. HBase核心物理概念

1.5.1. Namespace（表命名空间）

表命名空间不是强制的，当想把多个表分到一个组去统一管理的时候才会用到表命名空间。这个概念之前没提到，因为初学者一般用不到，当数据库中没有那么多表的时候也用不到这个概念。

1.5.2. Table

表结构逻辑视图：

rowkey	列簇 Column Family1			列簇 Column Family2		
	column	timestamp	value	column	timestamp	value
rk01	favor	ts4	pingpong	name	ts3	wangbaoqiang
rk02	name	ts7	xuzheng			
	age	ts5	20			
rk03	age	ts8	18	course	ts5	english
		ts7	19		ts4	math
	name		huangbo		ts3	algrithm
	telephone	ts9	13422556677		ts2	bigdata
		ts5	13422556688	friend	ts6	xuzheng
		ts3	13422556699			

关于图的标注：

```
箭头：表示索引顺序
红色数据：key
黄色数据：value
```

上图总结：

```
索引数据的流程：table ---> rowkey ---> column family ---> column ---> timestamp
行健（rowkey）：HBase中的每张表，都会按照rowkey全局排序
列簇（Column Family）：包含一组列，列在插入数据之前指定，列簇就必须在建表的时候指定
列（Column）：一个列簇中会包含多个列，并且可以不同
时间戳（TimeStamp）：每个列的值都可以保存多个版本，使用时间戳来表示，并且按照时间戳由近到远排序
```

理解MySQL帮助理解HBase：

```
1、RDBMS完全可以抽象成是一张二维表格，表由行和列组成。由行和列确定一个唯一的值
2、HBase本质是key-value数据库，key是行健rowkey，value是所有真实key-value的集合
3、HBase也可以抽象成为一张四维表格，四维分别由行健RowKey，列簇Column Family，列Column和时间戳Timestamp组成。
4、其中，一张HBase的所有列划分为若干个列簇（Column Family）
```

1.5.3. 行键（RowKey）

与NoSQL数据库们一样，RowKey是用来检索记录的主键。访问HBase Table中的行，只有三种方式：

```
1、通过单个row key访问
2、通过row key的range
3、全表扫描
```

RowKey行键可以是任意字符串(最大长度是64KB，实际应用中长度一般为10-100bytes)，最好是16。在HBase内部，RowKey保存为字节数组。HBase会对表中的数据按照rowkey排序(字典顺序)

存储时，数据按照RowKey的字典序(byte order)排序存储。设计Key时，要充分排序存储这个特性，将经常一起读取的行存储放到一起。(位置相关性)

1.5.4. 列簇（Column Family）

HBase表中的每个列，都归属与某个列簇。列簇是表的Schema的一部分(而列不是)，必须在创建表的时候指定。指定好了列簇就不能更改。列簇可以增加或者删除，删除的时候会删除这个列簇中的所有数据

列名都以列簇作为前缀。例如 `courses:history`， `courses:math` 都属于courses这个列簇，访问控制、磁盘和内存的使用统计等都是在列簇层面进行的。

列簇越多，在取一行数据时所要参与IO、搜寻的文件就越多，所以，如果没有必要，不要设置太多的列簇，官网推荐是小于等于3（最好就一个列簇）

1.5.5. 时间戳（TimeStamp）

HBase 中通过 RowKey 和 Column 确定的为一个存储单元称为 Cell。每个 Cell 都保存着同一份数据的多个版本。版本通过时间戳来索引。时间戳的类型是64位整型。时间戳可以由HBase (在数据写入时自动)赋值，此时时间戳是精确到毫秒的当前系统时间。时间戳也可以由客户显式赋值。如果应用程序要避免数据版本冲突，就必须自己生成具有唯一性的时间戳。每个Cell中，不同版本的数据按照时间倒序排序，即最新的数据排在最前面。HBase在查询的时候，默认返回最新版本/最近的数据。如果需要读取旧版本的数据，可以指定时间戳

为了避免数据存在过多版本造成的的管理(包括存储和索引)负担，HBase提供了两种数据版本回收方式：

```
保存数据的最后n个版本
保存最近一段时间内的版本（设置数据的生命周期TTL）
```

用户可以针对每个列簇进行设置。

你既可以把它称为是时间戳，也可以称为是版本号，因为它是用来标定同一个列中多个单元格的 版本号的。当你不指定版本号的时候，系统会自动采用当前的时间戳来作为版本号；而当你手动定义了一个数字来当作版本号的 时候，这个Timestamp只有版本号的意义了。

1.5.6. 单元格（Cell）

由 `{RowKey, Column(=<Column Family> + <Qualifier>), Version}` 唯一确定的单元，Cell中的数据是没有类型的，全部是字节码形式存储。

一个列中可以存储多个版本的数据。而每个版 本就称为一个单元格（Cell），所以在HBase中的单元格跟传统关系型数据库的单元格概念不一样。HBase中的数据细粒度比传统数据结构更细一级，同一个位置的数据还细分成多个版本。Timestamp（时间戳/版本号）：你既可以把它称为是时间戳，也可以称为是版本号，因为它是用来标定同一个列中多个单元格的版本号的。

1.6. HBase应用场景

1、半结构化或非结构化数据

对于数据结构字段不够确定或杂乱无章很难按一个概念去进行抽取的数据适合用HBase。而且HBase是面向列的，HBase支持动态增加字段

2、记录非常稀疏

RDBMS的行有多少列是固定的，为null的列浪费了存储空间。而HBase为null的Column是不会被存储的，这样既节省了空间又提高了读性能。

3、多版本数据

对于需要存储变动历史记录的数据，使用HBase就再合适不过了。HBase根据Row key和Column key定位到的Value可以有任意数量的版本值。

4、超大数据量的随机、实时读写

当数据量越来越大，RDBMS数据库撑不住了，就出现了读写分离策略，通过一个Master专门负责写操作，多个Slave负责读操作，服务器成本倍增。随着压力增加，Master撑不住了，这时就要分库了，把关联不大的数据分开部署，一些join查询不能用了，需要借助中间层。随着数据量的进一步增加，一个表的记录越来越大，查询就变得很慢，于是又得搞分表，比如按ID取模分成多个表以减少单个表的记录数。经历过这些事的人都知道过程是多么的折腾。采用HBase就简单了，只需要加机器即可，HBase会自动水平切分扩展，跟Hadoop的无缝集成保障了其数据可靠性（HDFS）和海量数据分析的高性能（MapReduce）。

5、查询简单

不涉及到复杂的Join查询，基于RowKey的简单查询或者RowKey的范围查询

1.7. HBase的访问接口

1、Native Java API，最常规和高效的访问方式，适合Hadoop MapReduce Job并行批处理HBase表数据

2、HBase Shell，HBase的命令行工具，最简单的接口，适合HBase管理使用

3、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

4、REST Gateway，支持REST 风格的Http API访问HBase，解除了语言限制

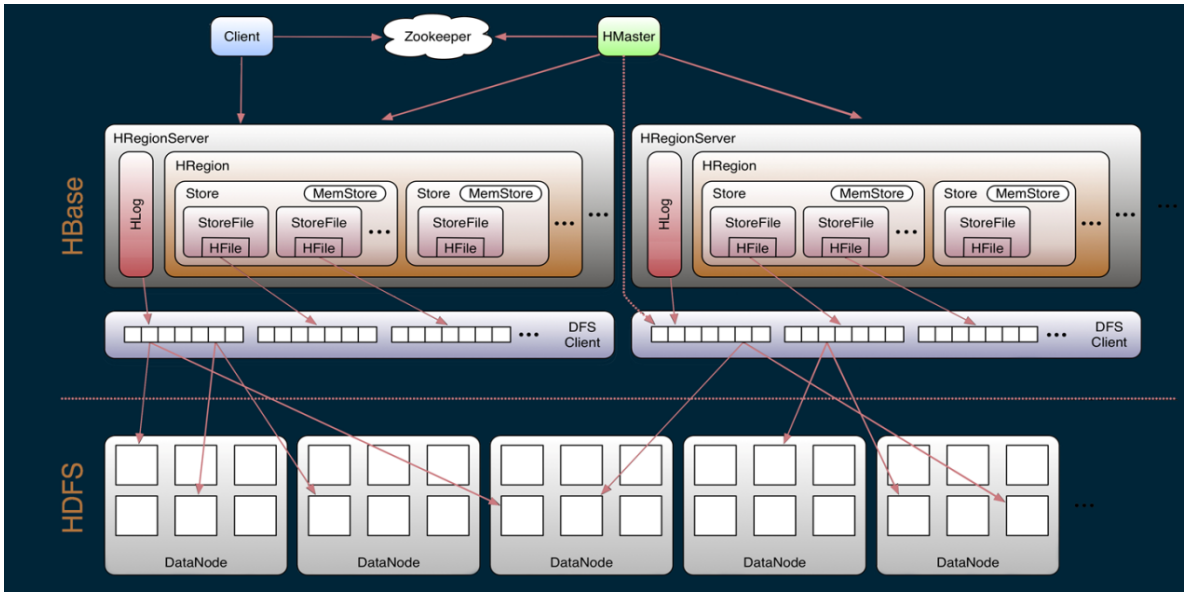
5、Pig，可以使用Pig Latin流式编程语言来操作HBase中的数据，和Hive类似，本质最终也是编译成MapReduce Job来处理HBase表数据，适合做数据统计

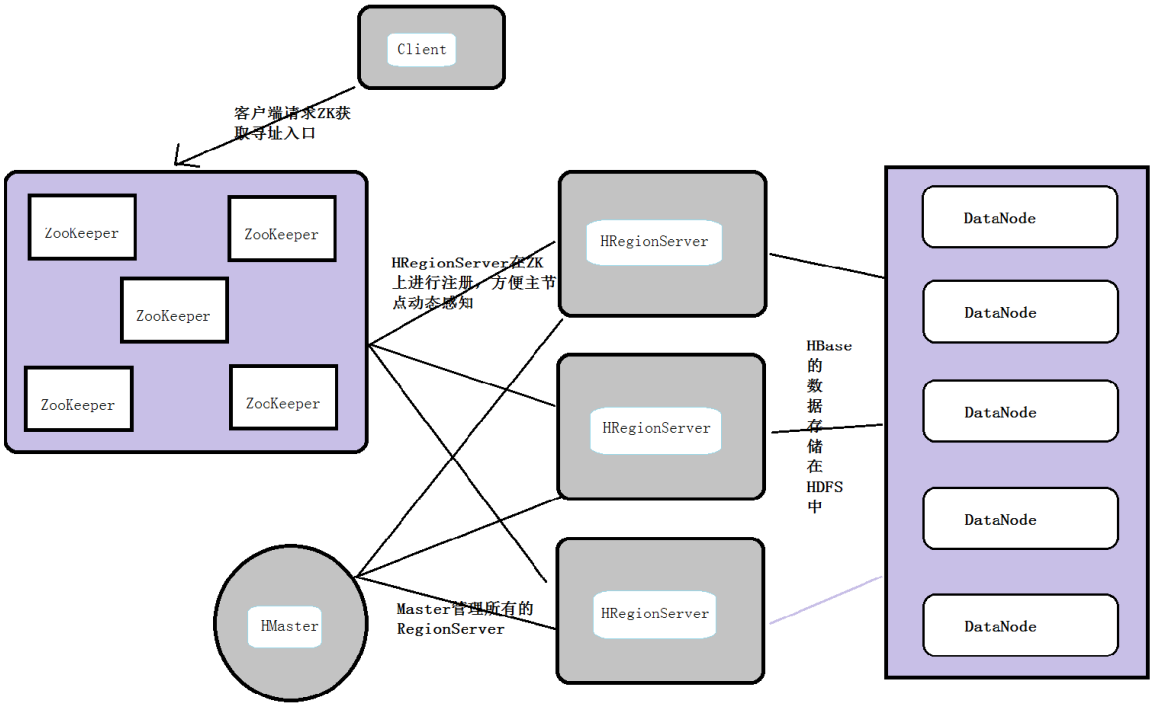
6、Hive，使用Hive整合HBae，能实现针对单条记录的增删改查，也能假设一个SQL客户端，针对HBaes中的数据进行查询分析。

2. HBase集群结构

2.1. 系统架构

HBase也是一个类似于HDFS的主从架构体系，依托于zookeeper实现高可用





2.2. HBase核心角色

HMaster

HBase的主节点，负责整个集群的状态感知、负载分配、负责用户表的元数据(schema)管理（可以配置多个用来实现HA），HMaster负载压力相对于HDFS的NameNode会小很多。HBase的HMaster其实就算是宕机一段时间也可以正常对外提供服务的(要搞清楚为什么)。

RegionServer：

HBase中真正负责管理Region的服务器，也就是负责为客户端进行表数据读写的服务器。每一台RegionServer会管理很多的Region，一个RegionServer上面管理的所有的region不属于同一张表。负责Region的Split，负责和底层的HDFS的存储交互，负责StoreFile的Compact。

ZooKeeper

整个HBase中的主从节点协调，元数据的入口，主节点之间的选举，集群节点之间的上下线感知……都是通过ZooKeeper来实现

HDFS

用来存储HBase的系统文件，或者表的Region文件

Client

Client包含了访问HBase的接口，另外Client还维护了对应的Cache来加速HBase的访问，比如Cache的.META.元数据的信息。

Region

是HBase将一个表中的所有数据按照RowKey的不同范围进行切割的逻辑单元，每个Region负责一定范围数据的读写访问。Region由RegionServer负责管理。HBase中的Region的概念就和HDFS中的数据块的概念差不多，Region是HBase表切分出来的一个分片。数据块是HDFS中的一个大文件切分出来的一个分片。

3. HBase和Hive的比较

3.1. 相同点

1、HBase和Hive都是架构在Hadoop之上，用HDFS做底层的数据存储，用MapReduce做数据计算

3.2. 不同点

1、Hive是建立在Hadoop之上为了降低MapReduce编程复杂度的ETL工具。
HBase是为了弥补Hadoop对实时操作的缺陷

2、Hive表是纯逻辑表，因为Hive的本身并不能做数据存储和计算，而是完全依赖Hadoop
HBase是物理表，提供了一张超大的内存Hash表来存储索引，方便查询

3、Hive是数据仓库工具，需要全表扫描，就用Hive，因为Hive是文件存储
HBase是数据库，需要索引访问，则用HBase，因为HBase是面向列的NoSQL数据库

4、Hive表中存入数据（文件）时不做校验，属于读模式存储系统
HBase表插入数据时，会和RDBMS一样做Schema校验，所以属于写模式存储系统

5、Hive不支持单行记录操作，数据处理依靠MapReduce，操作延时高

HBase支持单行记录的CRUD，并且是实时处理，效率比Hive高得多

4. HBase集群搭建

详见 安装文档 和 视频