HBase

**一、什么是HBase**

  HBase是一种分布式的、面向列的开源Hadoop数据库(Hadoop Database)，经常被描述为一种稀疏的，分布式的，持久化的，多维有序映射，它基于行键、列键和时间戳建立索引，是一个可以随机访问的存储和检索数据的平台。HBase不限制存储的数据的种类，允许动态的、灵活的数据模型，不用SQL语言，也不强调数据之间的关系。HBase被设计成在一个服务器集群上运行，可以相应地横向扩展。

***行存储与列存储***

  目前大数据存储有两种方案可供选择：行存储和列存储。业界对两种存储方案有很多争持，集中焦点是:谁能够更有效地处理海量数据，且兼顾安全、可靠、完整性。从目前发展情况看，关系数据库已经不适应这种巨大的存储量和计算要求，基本是淘汰出局。列式存储是相对于传统关系型数据库的行式存储来说的。

  （1）行存储的写入是一次完成。如果这种写入建立在操作系统的文件系统上，可以保证写入过程的成功或者失败，数据的完整性因此可以确定。列存储由于需要把一行记录拆分成单列保存，写入次数明显比行存储多，再加上磁头需要在盘片上移动和定位花费的时间，实际时间消耗会更大。所以，行存储在写入上占有很大的优势。

  （2）数据读取时，行存储通常将一行数据完全读出，如果只需要其中几列数据的情况，就会存在冗余列，出于缩短处理时间的考量，消除冗余列的过程通常是在内存中进行的。

  （3）列存储每次读取的数据是集合的一段或者全部，不存在冗余性问题。

  （4）由于列存储的每一列数据类型是同质的，不存在二义性问题。比如说某列数据类型为整型，那么它的数据集合一定是整型数据。这种情况使数据解析变得十分容易。相比之下，行存储则要复杂得多，因为在一行记录中保存了多种类型的数据，数据解析需要在多种数据类型之间频繁转换，这个操作很消耗CPU，增加了解析的时间。所以，列存储的解析过程更有利于分析大数据。

**二、HBase的特点**

**优点**

（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非常难按一个概念去进行抽取的数据适合用HBase，因为HBase支持动态添加列。

（2）记录很稀疏： RDBMS的行有多少列是固定的。为null的列浪费了存储空间。HBase为null的Column不会被存储，这样既节省了空间又提高了读性能。

（3）多版本号数据： 依据Row key和Column key定位到的Value能够有随意数量的版本号值，因此对于须要存储变动历史记录的数据，用HBase是很方便的。比方某个用户的Address变更，用户的Address变更记录也许也是具有研究意义的。

（4）仅要求最终一致性： 对于数据存储事务的要求不像金融行业和财务系统这么高，只要保证最终一致性就行。（比如HBase+elasticsearch时，可能出现数据不一致）

（5）业务场景简单： 不需要太多的关系型数据库特性，列入交叉列，交叉表，事务，连接等。

**缺点**

（1）单一RowKey固有的局限性决定了它不可能有效地支持多条件查询。

（2）不适合于大范围扫描查询。

（3）不直接支持 SQL 的语句查询。

**三、HBase架构**

  在分布式的生产环境中，HBase 需要运行在 HDFS 之上，以 HDFS 作为其基础的存储设施。HBase 上层提供了访问的数据的 Java API 层，供应用访问存储在 HBase 的数据。在 HBase 的集群中主要由 Master 和 Region Server 组成，以及 Zookeeper。

（1）**Master**

    HBase Master用于协调多个Region Server，侦测各个RegionServer之间的状态，并平衡RegionServer之间的负载。HBaseMaster还有一个职责就是负责分配Region给RegionServer。HBase允许多个Master节点共存，但是这需要Zookeeper的帮助。不过当多个Master节点共存时，只有一个Master是提供服务的，其他的Master节点处于待命的状态。当正在工作的Master节点宕机时，其他的Master则会接管HBase的集群。

（2）**Region Server**

    对于一个RegionServer而言，其包括了多个Region。RegionServer的作用只是管理表格，以及实现读写操作。Client直接连接RegionServer，并通信获取HBase中的数据。对于Region而言，则是真实存放HBase数据的地方，也就说Region是HBase可用性和分布式的基本单位。如果当一个表格很大，并由多个CF组成时，那么表的数据将存放在多个Region之间，并且在每个Region中会关联多个存储的单元（Store）。

（3）**Zookeeper**

   为HBase Master的HA解决方案（High Availability，简称 HA，高可用性，指提供在本地系统单个组件故障情况下，能继续访问应用的能力，无论这个故障是业务流程、物理设施、IT软/硬件的故障）。也就是说，是Zookeeper保证了至少有一个HBase Master 处于运行状态。并且Zookeeper负责Region和Region Server的注册。其实Zookeeper发展到目前为止，已经成为了分布式大数据框架中容错性的标准框架。不光是HBase，几乎所有的分布式大数据相关的开源框架，都依赖于Zookeeper实现HA。

**四、HBase的数据模型**

  表在行的方向上分割为多个Region；Region是Hbase中分布式存储和负载均衡的最小单元，不同Region分布到不同RegionServer上。Region按大小分割的，随着数据增多，Region不断增大，当增大到一个阀值的时候，Region就会分成两个新的Region；Region虽然是分布式存储的最小单元，但并不是存储的最小单元。每个Region包含着多个Store对象。每个Store包含一个MemStore或若干StoreFile，StoreFile包含一个或多个HFile。MemStore存放在内存中，StoreFile存储在HDFS上。

   -ROOT-和.META.是HBase的两张内置表，从存储结构和操作方法的角度来说，它们和其他HBase的表没有任何区别，你可以认为这就是两张普通的表，对于普通表的操作对它们都适用。它们与众不同的地方是HBase用它们来存贮一个重要的系统信息——Region的分布情况以及每个Region的详细信息。

   HBase的所有Region元数据被存储在.META.表中，随着Region的增多，.META.表中的数据也会增大，并分裂成多个新的Region。为了定位.META.表中各个Region的位置，把.META.表中所有Region的元数据保存在-ROOT-表中，最后由Zookeeper记录-ROOT-表的位置信息。所有客户端访问用户数据前，需要首先访问Zookeeper获得-ROOT-的位置，然后访问-ROOT-表获得.META.表的位置，最后根据.META.表中的信息确定用户数据存放的位置。

   -ROOT-表永远不会被分割，它只有一个Region，这样可以保证最多只需要三次跳转就可以定位任意一个Region。为了加快访问速度，.META.表的所有Region全部保存在内存中。客户端会将查询过的位置信息缓存起来，且缓存不会主动失效。如果客户端根据缓存信息还访问不到数据，则询问相关.META.表的Region服务器，试图获取数据的位置，如果还是失败，则询问-ROOT-表相关的.META.表在哪里。最后，如果前面的信息全部失效，则通过ZooKeeper重新定位Region的信息。

**五、HBase的高可用（HA）**

***Write-Ahead-Log（WAL）保障数据高可用***

   HBase中的HLog机制是WAL的一种实现，而WAL（一般翻译为预写日志）是事务机制中常见的一致性的实现方式。每个RegionServer中都会有一个HLog的实例，RegionServer会将更新操作（如 Put，Delete）先记录到 WAL（也就是HLog）中，然后将其写入到Store的MemStore，最终MemStore会将数据写入到持久化的HFile中。这样就保证了HBase的写的可靠性。如果没有 WAL，当RegionServer宕掉的时候，MemStore 还没有写入到HFile，或者StoreFile还没有保存，数据就会丢失。不用担心HFile本身会不会丢失，这是由 HDFS 来保证的。在HDFS中的数据默认会有3份。因此这里并不考虑 HFile 本身的可靠性。

   HFile由很多个数据块（Block）组成，并且有一个固定的结尾块。其中的数据块是由一个Header和多个Key-Value的键值对组成。在结尾的数据块中包含了数据相关的索引信息，系统也是通过结尾的索引信息找到HFile中的数据。

***组件高可用***

 （1）Master容错：Zookeeper重新选择一个新的Master。如果无Master过程中，数据读取仍照常进行，但是，region切分、负载均衡等无法进行；

 （2）RegionServer容错：定时向Zookeeper汇报心跳，如果一旦时间内未出现心跳，Master将该RegionServer上的Region重新分配到其他RegionServer上，失效服务器上“预写”日志由主服务器进行分割并派送给新的RegionServer；

 （3）Zookeeper容错：Zookeeper是一个可靠地服务，一般配置3或5个Zookeeper实例。