**Kudu:**

Kudu是用于存储结构化(structured)的表(Table)。表有预定义的带类型的列(Columns)，每张表有一个主键(primary key)。主键带有唯一性(uniqueness)限制，可作为索引用来支持快速的random access。

Kudu的表是由很多数据子集构成的，表被水平拆分成多个Tablets. Kudu用以每个tablet为一个单元来实现数据的durability。Tablet有多个副本，同时在多个节点上进行持久化。

Kudu有两种类型的组件，Master Server和Tablet Server。Master负责管理元数据。这些元数据包括talbet的基本信息，位置信息。Master还作为负载均衡服务器，监听Tablet Server的健康状态。对于副本数过低的Tablet，Master会在起replication任务来提高其副本数。Master的所有信息都在内存中cache，因此速度非常快。每次查询都在百毫秒级别。Kudu支持多个Master，不过只有一个active Master，其余只是作为灾备，不提供服务。

Tablet Server上存了10~100个Tablets，每个Tablet有3(或5)个副本存放在不同的Tablet Server上，每个Tablet同时只有一个leader副本，这个副本对用户提供修改操作，然后将修改结果同步给follower。Follower只提供读服务，不提供修改服务。副本之间使用raft协议来实现High Availability，当leader所在的节点发生故障时，followers会重新选举leader。根据官方的数据，其MTTR约为5秒，对client端几乎没有影响。Raft协议的另一个作用是实现Consistency。Client对leader的修改操作，需要同步到N/2+1个节点上，该操作才算成功。

**列式存储**

　　持久化的列式存储存储，与HBase完全不同，而是使用了类似Parquet的方式，同一个列在磁盘上是作为一个连续的块进行存放的

HBase支持snappy存储，然而因为它的LSM的数据存储方式，使得它很难对数据进行特殊编码，这也是Kudu声称具有很快的scan速度的一个很重要的原因。不过，因为列式编码后的数据很难再进行修改，因此当这写数据写入磁盘后，是不可变的，这部分数据称之为base数据。Kudu用MVCC(多版本并发控制)来实现数据的删改功能。更新、删除操作需要记录到特殊的数据结构里，保存在内存中的DeltaMemStore或磁盘上的DeltaFIle里面。DeltaMemStore是B-Tree实现的，因此速度快，而且可修改。磁盘上的DeltaFIle是二进制的列式的块，和base数据一样都是不可修改的。因此当数据频繁删改的时候，磁盘上会有大量的DeltaFiles文件，Kudu借鉴了Hbase的方式，会定期对这些文件进行合并。

### **impala（[官网](http://www.cloudera.com/documentation/archive/impala/2-x/2-1-x/topics/impala_insert.html?scroll=insert" \t "https://www.cnblogs.com/skyEva/p/_blank)）**

实时交互SQL大数据查询工具

它提供SQL语义，能查询存储在Hadoop的HDFS和HBase中的PB级大数据。

Impala的最大特点也是最大卖点就是它的快速。

Impala完全抛弃了Hive+MapReduce这个不太适合做SQL查询的缓慢范式，shuffle太慢

通过使用与商用并行关系数据库中类似的分布式查询引擎，可以直接从HDFS或HBase中用SELECT、JOIN和统计函数查询数据，从而大大降低了延迟

Impala与Hive都是构建在Hadoop之上的数据查询工具各有不同的侧重适应面

但从客户端使用来看Impala与Hive有很多的共同之处，如数据表元数据、ODBC/JDBC驱动、SQL语法、灵活的文件格式、存储资源池、用户界面（Hue Beeswax）等。

Hive适合于长时间的批处理查询分析，而Impala适合于实时交互式SQL查询