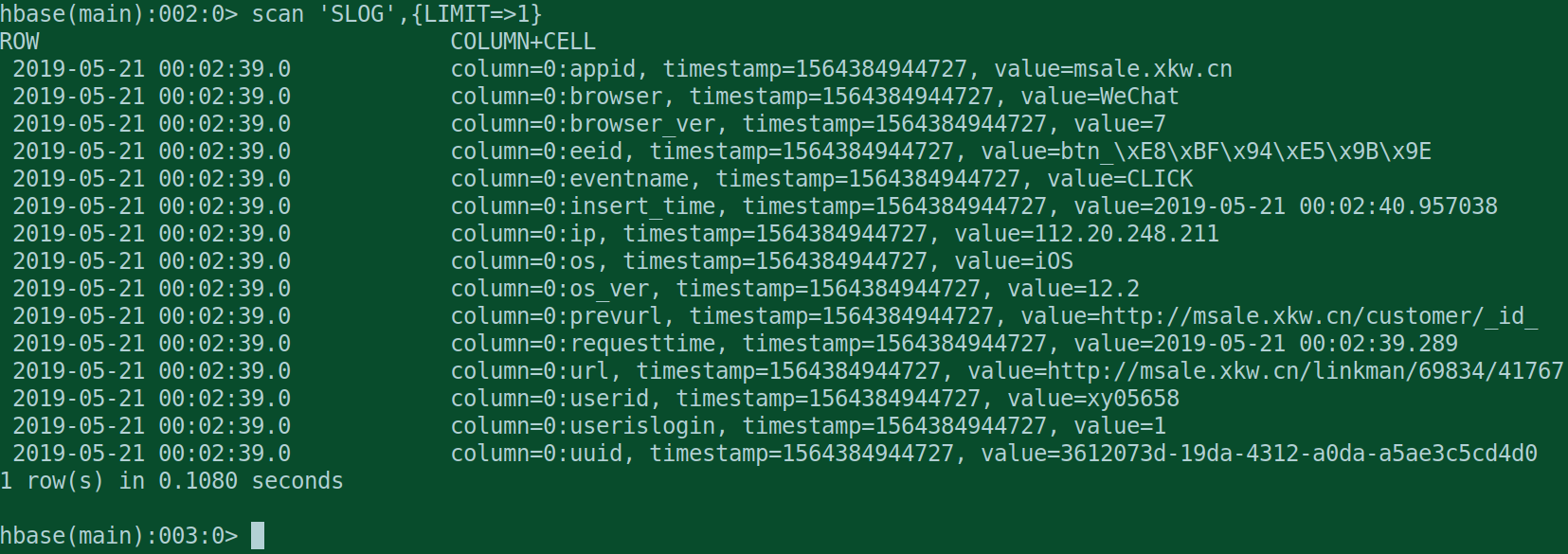
Hadoop组件介绍

Hadoop是为了应对大量非结构化数据而产生的架构，因扩展性好，后来开逐步出现了基于hadoop或类似于hadoop架构可处理结构化数据的组件。Hadoop只有部分组件有事务功能，而且也只能保持在小规模范围内，所以hadoop很难用于oltp系统；同等硬件配置下hadoop组件的性能只会小于等于数据库，hadoop的主要优势在于大规模集群的非事务性处理上。Hadoop的早期版本注重大量磁盘非实时数据处理，2G内存就可运行，磁盘个数越多，跑得越快；现在的版本组件分支较多，大多以内存处理为主，各个组件都建议服务器配置大内存。

组件介绍：

1. hdfs 数据存储模块，提供类似文件目录的功能，对于大于128M(默认，可改)的文件自动切分成128M大小存储，对于小于128M的文件直接存储。对于文件的读取自动多线程操作，每个线程操作一个数据块(128M大小的文件)，可设置备份数量，可手动或自动均衡数据分布。存储块的大小可根据数据量和集群内磁盘数量来估算，如果所有大表都使用分区存储，则存储块大小影响不大。
2. hbase使用专用语法(非sql)，数据存储于内存，同时会写入磁盘或hdfs，非结构化存储，key-value形式，只能使用主键进行查询，主键自动索引，查询速度很快，数据有版本信息，数据可标记为删除状态。主要使用命令行或java接口进行操作。数据的存储如下图



1. kudu使用专用语法(非sql)，数据存储于磁盘，不基于hdfs，也是分布式存储，但功能没有hdfs强大，数据为结构化列式存储，可更改或删除，只有主键有索引，主键可以是联合主键，但不能包含数字类型（是bug，可建立，使用中出错）。数据存储时可以选择压缩，压缩后占用空间小但压缩和解压都会占用较多cpu。
2. hive主要对用户提供sql，并有专门的资料库用于存储数据结构；数据存储于hadoop的hdfs或kudu，也可接入hbase的数据，hive在使用数据时会将数据转换成结构化后再使用；计算时基于tez/mr/spark三种；基于hdfs的数据是行式文本存储。
   1. hive本身的特性是无索引，数据存储在hdfs上时可以可压缩存储，减少空间使用量，但压缩和解压都需要较多的cpu计算，实际速度并不快。基于kudu或hbase的数据由hive负责转换，实际的存储及管理仍由kudu或hbase完成。
   2. hive基于mr的计算所需内存小，一般不会发生内存溢出导致计算失败。但速度慢、且单任务运行，其他任务等待。对于内存不多的机器推荐使用此方式。
   3. hive基于spark的计算需要较大内存，可多任务同时进行，速度快。但容易内存溢出、连接不断开会一直占用spark资源。
   4. hive基于hbase的计算如果只是以id查询单条记录，会很快，如果使用到其他字段，则会读取所有hbase的数据，如果数据量较小且全部存储在内存中时，可以使用此种方式。基于hbase的表只能插入和读取数据，不支持update和delete语句，想修改数据按id插入即可覆盖之前的数据。Delete语句只能在hbase的shell或api中执行。
   5. hive基于kudu的计算使用时只读取使用到的列，相对于使用hdfs会快一些。可以支持update，必须有主键，按主键插入会自动覆盖同主键记录，不会报错，数据可以压缩。已知的bug有主键中有数字类型的列有时会报错。
   6. hive可以创建列存储表，使用parquet组件即可，列存储数据可以进行压缩，压缩后数据所占空间减少，cpu使用增加，在磁盘读取io性能不高的情况下可以选择此种方式。建表时不需要指定主键，只支持insert，不支持update和delete。
3. mr是hadoop自带计算组件，hadoop1代和2代中间过程都会将临时数据写入hdfs，计算相对较慢，hadoop3代进行了较大改进，类似于spark和tez，减少中间过程，同时会使用内存缓存数据，性能有较大提高。Mr的计算过程是java编写，提供接口，重写非常方便，对于处理非结构化数据用处很大。
4. tez和spark都是重写了mr，中间不再将数据写入磁盘，而是存在内存，同时减少中间步骤。Hadoop3代是最近刚出的，tez和spark都只是针对1代和2代的应用。
5. phoenix是基于hbase的应用，因hbase本身不能使用sql，又没有索引，所以phoenix提供了使用中将hbase的数据转换成结构化数据，并可以使用sql对数据进行处理的功能；同时提供建索引功能，当对表A建立索引时，phoenix会建一个新的hbase表A1，使用表A的所有要使用索引的列连在一起作为主键，这样在查询时就可以使用主键的范围扫描了；但phoenix技术不够成熟，在少量写入，大量查询情况下性能较好，大量写入时性能较差且会发生崩溃。
6. impala使用hive的资料库，可读取hive的表结构并进行操作；计算功能是使用C++重写了mr的计算，数据由hive进行转换，转换后由impala计算；中间过程会缓存数据到内存中，性能较好，内存占用较大，官方推荐每台服务器128G内存。官方说impala不会缓存数据，对于语句第二次执行会快是因为底层数据源或服务器缓存了数据，不是impala缓存的。Impala中只能建基于hive(hdfs)的表，无法直接建立基于hbase或kudu的表，需要在hive中执行，hive中建好后，impala可以使用。
7. greenplum不是hadoop组件，是基于postgresql的数据库，支持行存储，也支持列存储，但列存储只能增加数据，不能delete或update，但同一个库中即可以有行存储表也可以有列存储表。相对kudu来说，因为是强数据类型，对数据的有效性管理更好一些，在事务方面也更好，但对于需要修改的大表，处理起来就相当麻烦了。性能比kudu低一些。
8. sqoop是导数据工具，可以在数据库与hadoop集群间传输数据，速度相对较快。
9. datax是阿里开发的导数据工具，比sqoop支持更多类型的数据库或格式，但速度稍慢。
10. druid 有两个，一个是阿里开发的数据源管理工具，相当于jdbc，同时可以监控mysql、pg等数据库的连接数与服务器性能。一个是apache的预处理计算框架，对一个表设定维度，在导入数据的同时进行count、sum、avg等计算，同时会把数据排序并切分成多个块来存储；数据可以存储在磁盘或hdfs上。对于带where条件或sum、count等查询速度较快，对于其他计算查询速度一般。
11. kylin 也是类似于druid的预计算框架，主要由中国人参与开发，kylin需要设置任务，在任务中设置维度和计算值，定时进行处理，处理后生成cube，在查询时对cube进行查询，已经生成的数据在查询时速度很快。

数据表存储的选择：

1. hive impala 都是不支持存储过程的，如果要实现类似存储过程的功能，需要把sql写在文本中，再进行调用。
2. kudu和phoenix完善程度相对较低，不建议过多使用。
3. 数据量不大，可完全存储在内存中，且频繁修改的小表使用hbase。如果不经常修改的可以全量同步存储在hive中。
4. 需要经常修改且有时效性要求或必须要使用update和delete语句的小表可以使用kudu。
5. 无法分区，以主键为主要查询条件的大表也可以使用hbase。比如a\_res\_soft\_info。
6. 全量同步表、大数据量表、字段长度较长的表、可分区的表都尽量使用hive。
7. 所有的前端dml操作都使用impala。比如用hive向hbase表中插入一条数据，由于需要启动mr，一般需要7秒以上的时间，而使用impala可以在毫秒级别完成。