1. 函数式语言的特点：如果比作数学函数，输入的是自变量，输出因变量，好处是减少了代码量，而且使程序更加灵活易读
2. Kafka高低阶API的使用：
	1. KafkaUtils.createDstream：

构造函数为KafkaUtils.createDstream(ssc, [zk], [consumer group id], [per-topic,partitions] ) 使用了receivers来接收数据，利用的是Kafka高层次的消费者api，对于所有的receivers接收到的数据将会保存在Spark executors中，然后通过Spark Streaming启动job来处理这些数据，默认会丢失，可启用WAL日志，该日志存储在HDFS上

创建一个receiver来对kafka进行定时拉取数据，ssc的rdd分区和kafka的topic分区不是一个概念，故如果增加特定主体分区数仅仅是增加一个receiver中消费topic的线程数，并不增加spark的并行处理数据数量,对于不同的group和topic可以使用多个receivers创建不同的DStream ,如果启用了WAL，需要设置存储级别，即KafkaUtils.createStream(….,StorageLevel.MEMORY\_AND\_DISK\_SER)

* 1. KafkaUtils.createDirectStream：

区别Receiver接收数据，这种方式定期地从kafka的topic+partition中查询最新的偏移量，再根据偏移量范围在每个batch里面处理数据，使用的是kafka的简单消费者api

优点:

A、 简化并行，不需要多个kafka输入流，该方法将会创建和kafka分区一样的rdd个数，而且会从kafka并行读取。

B、高效，这种方式并不需要WAL，WAL模式需要对数据复制两次，第一次是被kafka复制，另一次是写到wal中

C、恰好一次语义(Exactly-once-semantics)，传统的读取kafka数据是通过kafka高层次api把偏移量写入zookeeper中，存在数据丢失的可能性是zookeeper中和ssc的偏移量不一致。EOS通过实现kafka低层次api，偏移量仅仅被ssc保存在checkpoint中，消除了zk和ssc偏移量不一致的问题。缺点是无法使用基于zookeeper的kafka监控工具

* 1. 高级消费者使用代码：



1. 3.1配置：

LINUX 配置静态ip、更改hosts name ip地址、配置公钥

hadoop 集群配置（7个配置文件）

1. core-site.xml(namenode和临时存放目录)

2. hadoop-env.sh (配置环境环境变量和java\_home)

3. hdfs-site.xml(配置副本数目replication和secondaryNameNode)

4. yarn-env.sh (配置java\_home)

5. yarn-site.xml(配置recourseManger和获取数据的方式和日志的聚集)

6. map-env.sh(配置java\_home)

7. mapred-site.xml(指定MR运行在yarn上，配置历史服务器)

slaves 配置集群下每个节点的域名

时间同步，指定其中一台机器为时间来源、群发脚本

3.2高可用配置



3.3 MR运行原理

<https://blog.csdn.net/qq_36864672/article/details/78561375>

<https://www.cnblogs.com/arachis/p/Hadoop_MR.html>

4 HIVE篇

1. 慎用count(distinct),group by，两者比较：<https://blog.csdn.net/xiaoshunzi111/article/details/68484426>；
2. Orderby也会使用一个reduce，所以也可能导致OOM，效率变慢
3. 同时对于key值空值多的，可以采用key值随机数的 方式，减少数据倾斜。对于链接条件字段，类型不同的尽量先转换为相同的类型。在join时，用小表驱动大表。

2.只是依附于MapReduce的一款数据分析工具

|  |  |  |
| --- | --- | --- |
|  | Hive | Rdbms |
| 查询语言 | HQL | SQL |
| 数据存储 | HDFS | Raw Device  |
| 执行器 | MapReduce | Executor |
| 数据插入 | 支持批量导入/单条插入 | 支持单条或者多条导入插入 |
| 数据更新 | 不支持 | 支持 |
| 处理数据规模 | 大 | 小 |
| 执行延迟 | 高 | 低 |
| 分区 | 支持 | 支持 |
| 索引 | 0.8之后加入索引 | 支持 |
| 扩展性 | 高 | 有限 |

总结：hive具有sql数据块的外表，但是应用场景不同，hive只适合处理海量数据 分析，对实时性要求不高的。

3.sort by 是局部排序，order by是全局排序，cluster by是分桶加排序，他的排序默认使用的是倒序，并且当distrbute by和sort by使用相同字段时，并且排序方式指定倒叙时，这个等于其他两者同时的效果，distrbute by是分桶。

4.默认使用内嵌的derby数据库；可以修改为使用本地mysql数据库存储；还可以设置为远程mysql数据库存储模式；使用内嵌的derby数据库时，只支持一个客户端同时访问，一般用于测试。所以修改为mysql数据库。

5.内部表会将hdfs上的/input/data目录下的数据转移到/input/table\_data目录下。删除test表后，会将test表的数据和元数据信息全部删除。

外部表把hdfs上/input/edata/下的数据转到/user/hive/warehouse/et下，删除这个外部表后，/user/hive/warehouse/et下的数据不会删除，但是/input/edata/下的数据在上一步load后已经没有了！数据的位置发生了变化！

6.Antlr定义SQL的语法规则，完成SQL词法，语法解析，将SQL转化为抽象语法树AST Tree；遍历AST Tree，抽象出查询的基本组成单元QueryBlock；遍历QueryBlock，翻译为执行操作树OperatorTree；逻辑层优化器进行OperatorTree变换，合并不必要的ReduceSinkOperator，减少shuffle数据量；遍历OperatorTree，翻译为MapReduce任务；物理层优化器进行MapReduce任务的变换，生成最终的执行计划

7.select a.key,a.value from a where a.key not exists (select b.key from b);

8.load data local inpath ’/opt/module/data/test.txt’ into table test partition(l\_date=’2016-10-10’);

9.编写代码继承udf类，编写evaluate方法，打成jar包，上传到服务器，进入hive客户端执行:hive>add jar [jar路径]，创建临时函数：create temporary function 自定义名称AS‘自定义UDF的全类名’，执行HQL语句套用函数，销毁函数hive>drop temporary function 自定义函数名称。

10.Sdf

11.TextFile和sequencefile的存储格式都是基于行存储的，ORCFile和Parquet都是基于列存储的。

12.小表在前，小表驱动大表，在 0.7 版本后。也能够用配置来自己主 动优化：set hive.auto.convert.join=true。

13.可以在reduce阶段join，或者在map阶段进行join。

14.不是，当使用select\*时，默认不走mR的job，可以设置hive.fetch.task.conversion=more，简单查询就不走MR任务，设置为minimal，任何查询都会走。

15.UDF用户自定义函数，一进一出，需要继承UDF类，UDAF用户自定义聚合函数，一斤多出，需要继承UDAF类，UDTF用户自定义表函数，需要继承GenericUDTF，一进多出，比如炸裂函数lateral view explore()；

16.桶时更细粒度的划分，针对表中某一列进行分桶，hive采用对列值进行哈希，然后初一桶的个数，求余方式巨鼎记录存放在哪个桶中，好处时查询效率更高，并且可以对大量数据进行抽样查询。

5 HBase篇

 5.1刷写过程：

 [*https://blog.csdn.net/joeyon1985/article/details/71511891*](https://blog.csdn.net/joeyon1985/article/details/71511891)

 <https://www.jianshu.com/p/e2bbf23f1ba2>

 5.2 rowKey设计

<https://www.cnblogs.com/cxzdy/p/5118456.html>

5.3使用场景

spark on HBase

<https://yq.aliyun.com/articles/220721?utm_content=m_31885>

6 spark篇

 6.1 spark运行流程：

 client模式driver在提交任务节点，applicationmaster在集群任意节点

cluster模式：driver和applicationMaster在集群中同一节点

yarn-cluster模式：

（1）启动

submit提交任务的时候，client向resourcemanager请求启动applicationmaster，rm随机选一个节点启动driver，并启动applicationmaster(ExecutorLauncher)，

（2）申请资源

applicationmaster启动后的第一件事就是向rm请求资源，rm根据提交参数和集群配置封装container容器和用于启动executor的nm,

（3）启动executor

am在对应nm上启动executor,启动后的executor向am反向注册，然后driver将应用进行job划分，每个job分为多个stage,以stage为单位，创建一批task，分发给各个executor，在executor上执行task，当属于一个stage的所有task执行完毕后，driver发送下一个stage的tasks,当所有的stage执行完毕后，am销毁executor,并向resourcemanager申请注销自己，resourcemanage，rm将am注销，整个任务执行结束

<https://blog.csdn.net/do_what_you_can_do/article/details/53128480>

6.2 spark算子



6.3 shuffle

<https://blog.csdn.net/u010697988/article/details/70173104>

6.4 sparkSQL

<https://www.cnblogs.com/hadoop-dev/p/6742677.html>

6.5 sparkStreaming

<https://www.cnblogs.com/ChouYarn/p/6388070.html>

<https://blog.csdn.net/lwb314/article/details/79408201>

6.6 structuredStreaming

<https://blog.csdn.net/dongyunlon/article/details/79037366>

7 zookeeper篇

 <https://www.cnblogs.com/felixzh/p/5869212.html>

8 flume篇

 <https://www.cnblogs.com/zhangyinhua/p/7803486.html>

 <https://www.jianshu.com/p/e8f44da97379>

9 sqoop篇

 <https://blog.csdn.net/jiangsanfeng1111/article/details/53332635>

10 linux篇

 

名词解释：



数据仓库，cute上卷下钻

<https://www.cnblogs.com/muchen/p/5318808.html>

其他知识：

大数据架构：

<https://www.cnblogs.com/DarrenChan/articles/7567344.html>