1. 每天的离线数据要处理多长时间

8小时左右

1. 京东：
2. Spark Executor具体配置

spark.executor.extraJavaOptions -verbose:gc -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=70 -XX:MaxHeapFreeRatio=70 -XX:+CMSClassUnloadingEnabled -XX:OnOutOfMemoryError='kill -9 %p'

spark.hadoop.yarn.timeline-service.enabled false

spark.executor.memory 10240M

spark.executor.cores 4、

memory和core的数据基本是yarn的nodeManager 在节点上申请的资源的1倍或1/2,1/3，这样能充分利用计算机的资源。

1. 调优之前与调优之后性能的详细对比（例如调整map个数，map个数之前多少、之后多少，有什么提升）

这里举个例子。比如我们有几百个文件，会有几百个map出现，读取之后进行join操作，会非常的慢。这个时候我们可以进行coalesce操作，比如240个map，我们合成60个map，也就是窄依赖。这样再shuffle，过程产生的文件数会大大减少。提高join的时间性能。

1. 一个项目，从拿到需求到得到报表统计结果，经过哪些实际的工作流程

* 先与产品讨论，看报表的各个数据从哪些埋点中取
* 将取得逻辑过程设计好，与产品确定后开始开发
* 开发出报表sql脚本，并且跑几天的历史数据，观察结果
* 将报表放入调度任务中，第二天给产品看结果。
* 周期性将表结果导出或是导入后台数据库，生成可视化报表

1. 工作中用到过哪些shell脚本，shell脚本的具体功能

* 我们的数仓的脚本都是shell的。
* Flume的统一部署用shell
* 一些spark任务，用shell执行，并且监控结果

功能就是传入时间，或是地址等其他参数，给脚本执行。

1. 数仓拿到数据之后怎么根据数据特征进行表格的设计，具体怎么维度建模

表格是指二维表吧。这个如果说数据特征，看是行为分析类型的，还是对象分析类型的。前面用kimball建模，适合数据量大的事实表，后面用ER建模，适合聚合分析。

1. 对于具体的业务要如何设计表格，如何设计日志采集系统

看是行为分析类型的，还是对象分析类型的。前面用kimball建模，后面用ER建模

Flume基本能满足所有的数据源全量采集（jdbc，文件，kafka）。文件的增量采集也试用，而数据库的增量采集需要监控日志（如监控Mysql的binlog）。所以我们设计日志采集系统是

* 找出适合该数据源的source，如flume，或是canal
* 设计中间的缓冲层，kafka或是flume的filechannel
* 设计最后的落地层，根据实时性，是redis,Mysql，hbase还是hdfs等

注：日志采集系统的后面部分看实时性，比如elasticsearch,redis（有缓存，实时性高，可覆盖）,hdfs（原始数据结构可读性高，但文件流写入，实时低，可以覆盖），HBASE（实时高，可以覆盖），Mongodb（实时高，不可覆盖）。

1. 常用的报表工具

只知道一个kibana，和sqlpad，这个不太懂，ETL用Kettle

1. Spark任务使用什么进行提交，javavEE界面还是脚本

Shell 脚本。

1. 数据仓库每天跑多少张表，全跑还是跑一部分，大概什么时候跑？

每天跑一百多张表，0：20开始跑。（有时候数据会有延时）

## 20181125

一个情景处理：

数据流向：数据源->Flume->Kafka->HDFS->Hive(只做离线处理)

总共7台机器，每天几亿条数据

面临问题：

数据统计主要用HiveSQL，没有数据倾斜，小文件已经做了合并处理，开启的JVM重用，而且IO没有阻塞，内存用了不到50%。但是还是跑的非常慢，而且数据量洪峰过来时，整个集群都会宕掉。基于这种情况有没有优化方案

1. 内存利用率不够。这个一般是yarn的2个配置造成的，单个任务可以申请的最大内存大小，和hadoop单个节点可用内存大小。调节这两个参数能提高系统内存的利用率。
2. 集群宕掉。

Flume宕机：

* oom这种情况，可以靠调高jvm的堆内存大小。
* 采用filechannel而不是memory channel
* 部署多个flume同时处理

Kafka宕机：

* 如果只做离线处理，可以不需要kafka了。多个flume直接写在硬盘上，使用file roll sink切分文件，并且压缩。再直接上传到hdfs。大大提高效率
* 某个节点上的broker宕掉，查看leader是否过多。所有的producer都跟它建立连接。此时要考虑重新分配leader，

Hadoop宕机

* MR造成系统宕机。此时要控制yarn同时运行的任务数，和每个任务申请的最大内存
* 写入文件过量造成namenode的宕机。那么调高kafka的存储大小，控制从kafka到HDFS的写入速度。高峰期的时候用kafka进行缓存，高峰期过去数据同步会自动跟上。