Spark的大数据应用实践

卢亿雷 From AdMaster

johnlya@163.com

@johnlya

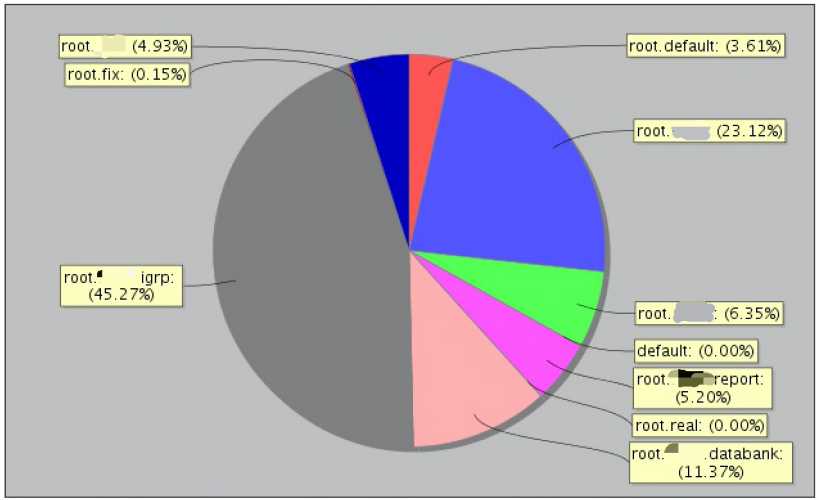
目录

我们是谁

- AdMaster : Make a Happy world with Data
- 使用的大数据技术:
  - Hadoop
  - HBase
  - MapReduce
  - Pig
  - Hive
  - Spark
  - Storm
  - ElasticSearch
  - Pinot

Why On Yarn

-
- MR/Spark/Storm 计算方式众多, On yarn 方便统一协调
- 服务器众多，方便资源统一控制
- 部门众多，方便资源统计和成本核算



Why On Yarn

- Yarn 越来越成熟了
- 流式处理的输出，批量处理的输入/输出，基本就是 HDFS
- Yarn + Fair Scheduler 我们自己在持续优化调度
  - 某些场景下, 集群闲, 但是不调度
  - 经常会“空转”

坑：Yarn

- Cdh 2.6.0-5.5.1　子队列不生效的 bug：对于子队列名转换的处理，解析和调用不一致
- mapreduce.fileoutputcommitter.algorithm.version　2
  - 对作业日志进行合并清理，拖慢运行速度
  - 2.6 中新增参数，默认为 1，即之前的做法．设置为 2，速度可以提升约 30%
- Pig 对 HistoryServer 的依赖

坑 : Yarn

- “container 内存限制” 机制各种两难
  - 设置太小或不设置，作业有可能因为内存不足而失败
  - 设置太大，资源浪费
  - 不启用，机器 OOM

坑：Spark on Yarn

- Executor的内存没达到上限前被kill
  - 调高spark.yarn.executor.memoryOverhead，默认384，根据实际需求调高
- 当有较多MapReduce job，scheduler调度压力增大的时 候，Spark job会被kill掉
  - 升级hadoop集群到 2.6 以上版本
- Executor OOM
  - 增加job的并行度
  - 大数据集切分成更小的数据
  - 调整spark.storage.memoryFraction和spark.executor.memory
  - 设置spark.cleaner.ttl清理元数据

ElasticSearch & Spark

- Spark and ES
  - ES 简单维度下钻/汇聚/搜索
  - Spark 复杂业务处理
- Spark on ES
  - https://www.elastic.co/products/hadoop
  - Write : saveToEs
  - Read : val RDD = sc.esRDD(“radio/artists”) // 后面是 index

ElasticSearch比较

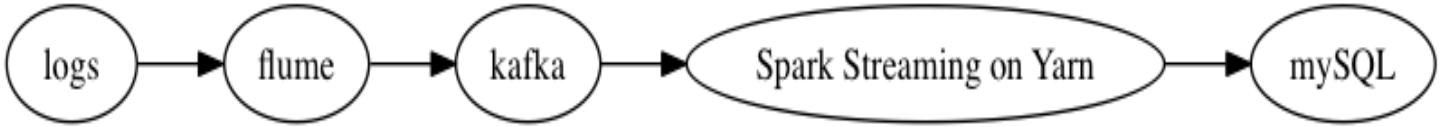
- 测试条件
  - 记录条数分为100亿以内和1000亿条
  - 服务器数量为70台，配置为:CPU 12核，内存96G，硬盘48T
  - 测试语句：select count(*) from test where age > 25 and gender > 0 and os > "500" and sc in ("0001009","0002036","0016030",”…") or bs>585 and group by age,gender,os,bs
  - 总共200列：动态列为3列（多值列），普通列为11列

ElasticSearch比较

1000亿	单次	并发5个	并发10个
ElasticSearch	19005ms	21005ms	27736ms
Pinot	19019ms	failed	failed

Spark 案例 1 : 实时监控

- 130,000 event / s
- Delay < 30s（全国各地多机房)



实时监控

- 为什么我做不到那么快?是机器不够多吗?
  - Kafka 收数据的速度
    - 机器 cpu 的性能
    - 网卡的性能（ 网卡独占一个 cpu)
    - 数据的分散程度（ 人工打散 ）
  - Spark 收数据的速度,取决于 kafka 的 partitions 和 replica 量，这个要多
  - Spark 计算的速度
    - 算法效率（ 越简单越好 ）
    - Excuter 数（ 正比于机器数)
    - 单个 excuter gc 的次数

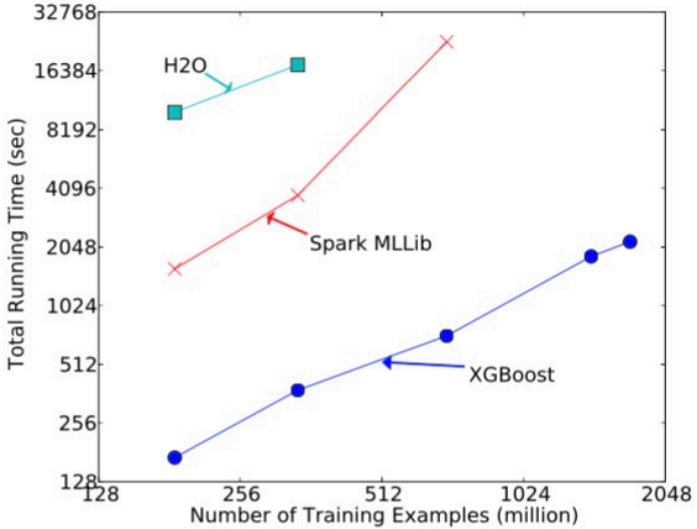
实时监控

- 大数据就是”一堆开源软件的堆砌”?
  - AdMaster 自己开发的 flume source，支持多种数据类型的收集
  - AdMaster 自己开发的 flume 消息加密算法，安全高效
  - AdMaster 自己定义的 flume to kafka 算法，使用数据足够均匀
  - AdMaster 自己设计的并发入库算法，足以支撑 13w+/s 的吞吐量
  - ….

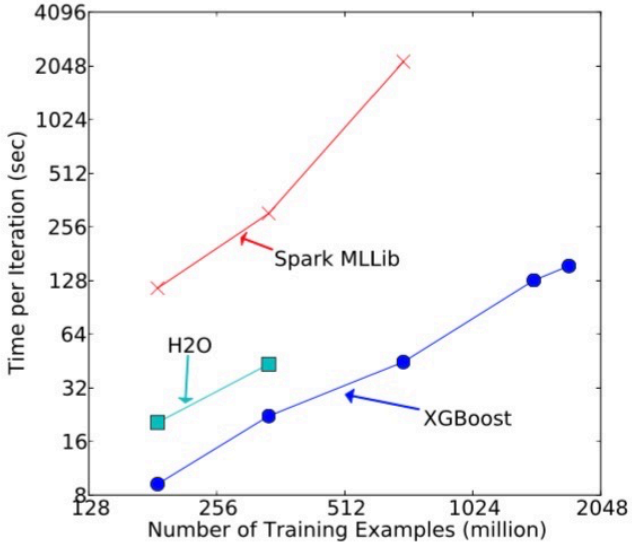
Spark 案例 2 : 跨屏分析

- 移动互联网的发展, cookie 已经不足以标识一个”互联网人”
- 有登录 id (如 qq, taobao) 的用自身内部 id 来标识
- 没有登录 id 的怎么办?
- Ip/imei/idfa/其它加密唯一信息 + 机器学习

Spark 案例 2：跨屏分析



(a) End-to-end time cost include data loading



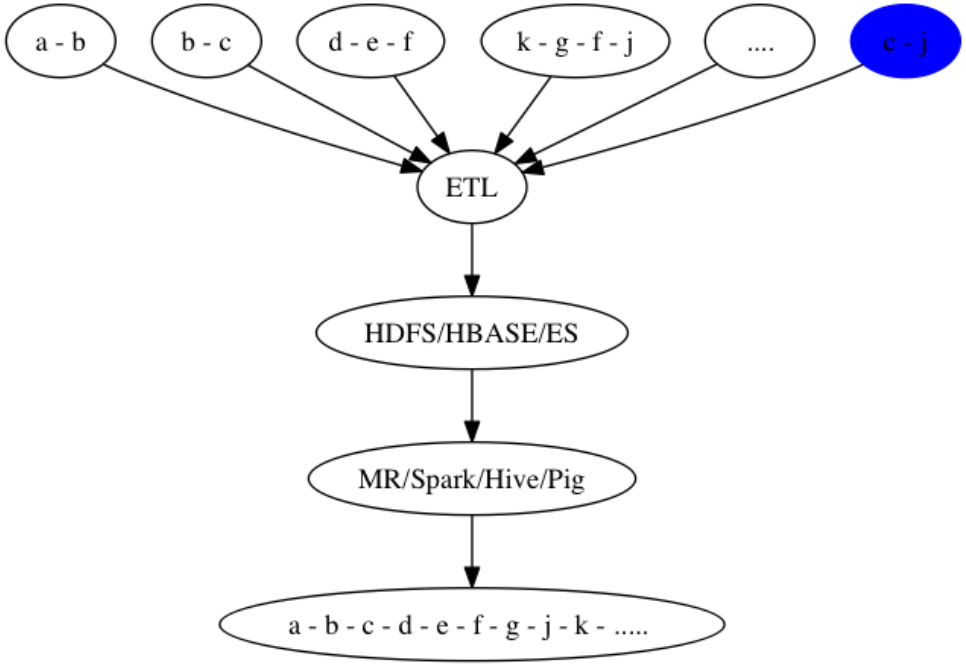
(b) Per iteration cost exclude data loading

引自：XGBoost: A Scalable Tree Boosting System

问题分析

- 每天请求数超过 100 亿
- 每天增长超过 5TB 级数据
- 每天对几千亿条记录进行上 1000 种维度的计算
- 客户有流式、实时、离线需求
- 数据源有广告曝光、点击数据，有微博、微信、新闻、论坛数据，还有CRM、渠道等第三方数据

看起来像这样


a - b
b - c
d - e - f
k - g - f - j
....
c - j
ETL
HDFS/HBASE/ES
MR/Spark/Hive/Pig
a - b - c - d - e - f - g - j - k - .....

理解

- 从关系型数据库的解决来理解，就是多表多维度　join
- 难点在于
  - 数据本来不在一起，需要各种 ETL 来放到一起
  - 数据量太大，原有的存储方式行不通
  - 数据质量不好保证，清洗至关重要
  - 大数据算法基本都需要按业务修正
  - 上图中蓝色的数据源，现实中有可能找不到或成本太高，导致 join 变得不可能

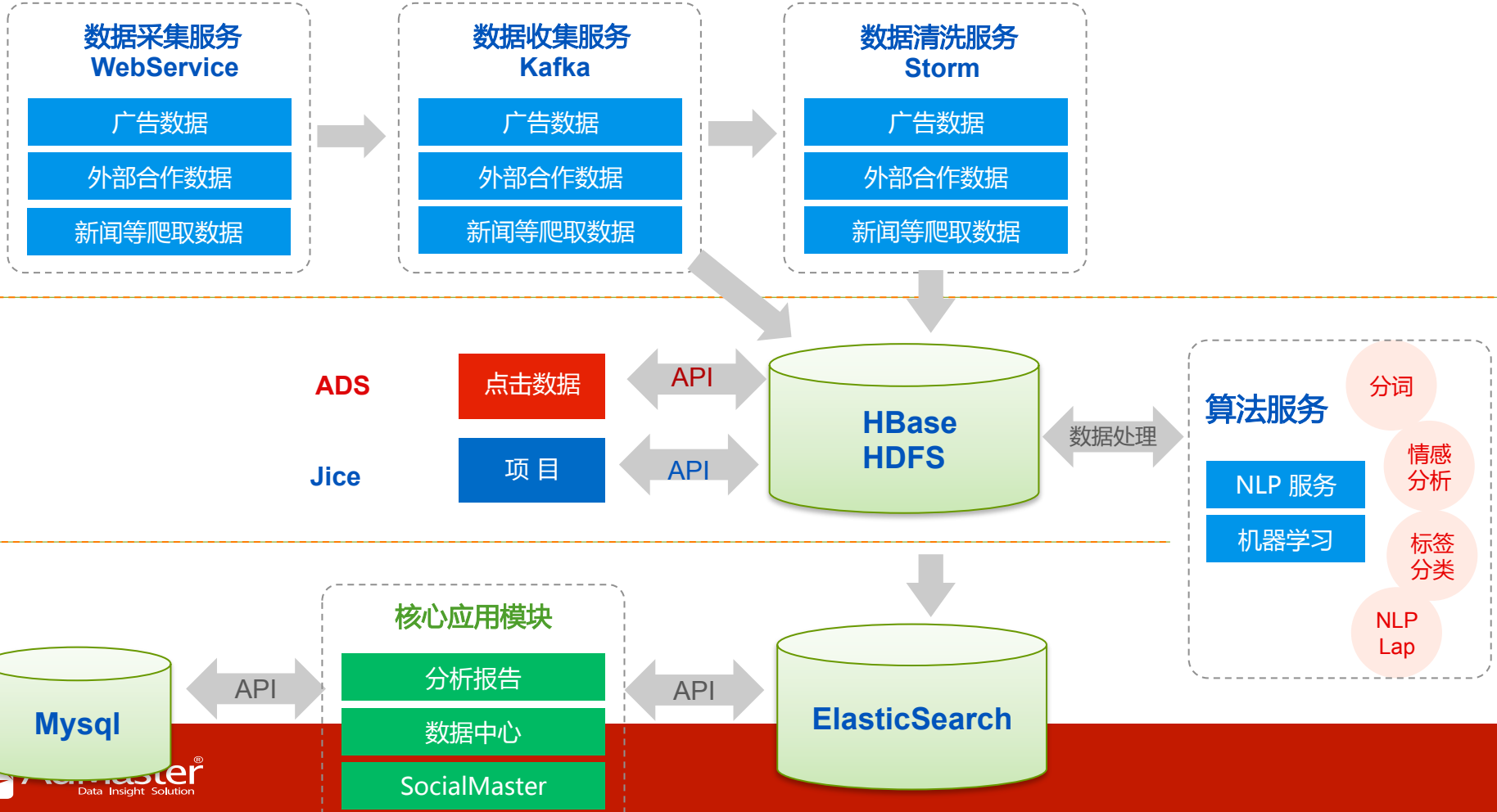
Storm or Spark streaming

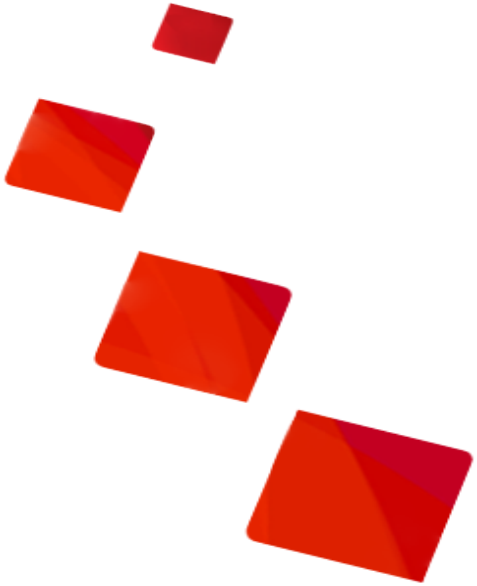
	Storm	Spark	需求
语言	Clojure/java	Scala/java	Java/scala
开发量	多	少	希望少而快
调试难度	简单	复杂	希望容易
实时性	非常快	streaming 一般快，伪实时	足够快就好

所以:

- 实时, Storm
- 快速实现/足够快就行 demo Spark
- 偏爱 java，其实还有 jStrom

数据流分析案例


数据采集服务
WebService
广告数据
外部合作数据
新闻等爬取数据
数据收集服务
Kafka
广告数据
外部合作数据
新闻等爬取数据
数据清洗服务
Storm
广告数据
外部合作数据
新闻等爬取数据
ADS
点击数据
Jice
项 目
API
API
HBase
HDFS
数据处理
算法服务
分词
情感分析
NLP 服务
机器学习
标签分类
NLP Lap
核心应用模块
分析报告
数据中心
SocialMaster
API
API
Mysql
ElasticSearch



Q & A

Email:johnlya@163.com