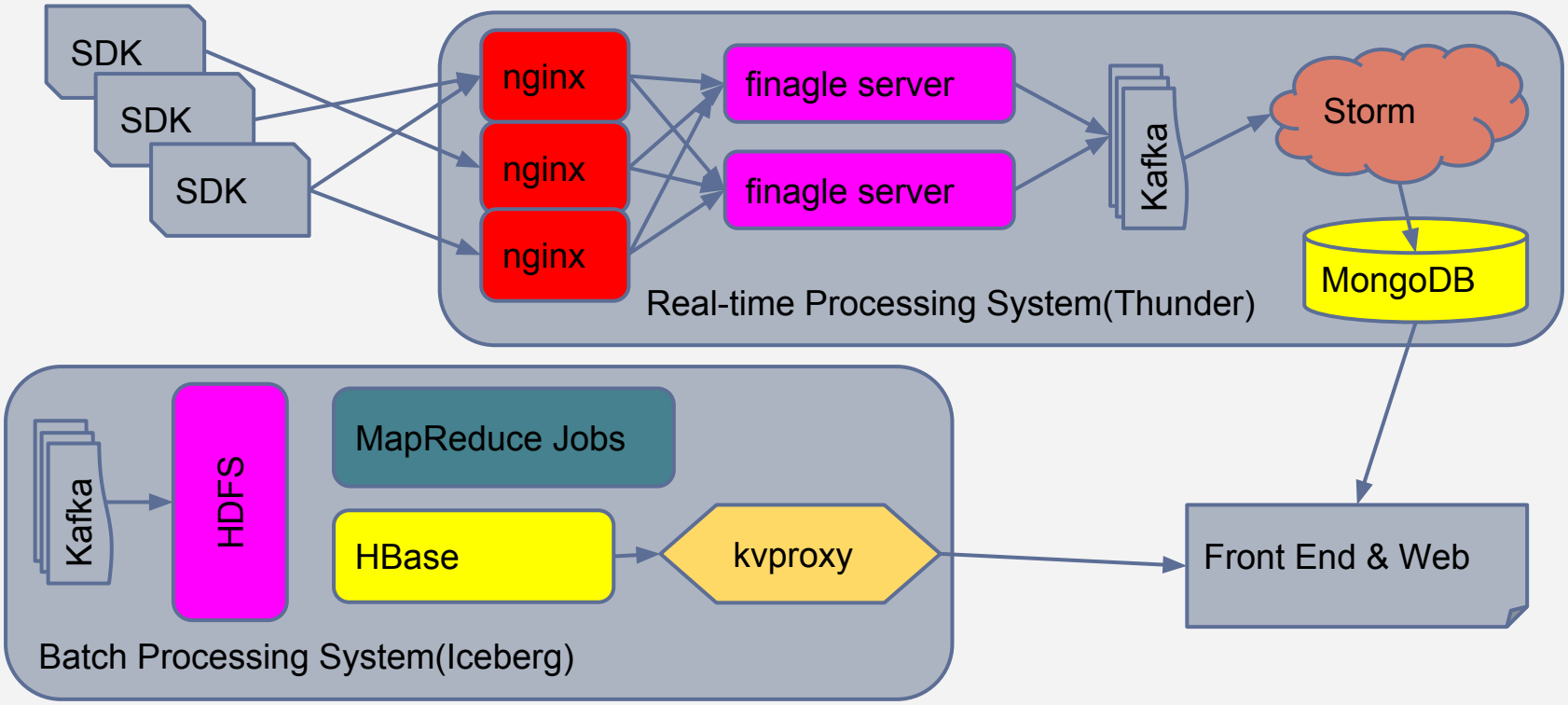
如何在一天之内收集3.6亿移动设备的数据

章炎@友盟

Overview


SDK
SDK
SDK
nginx
nginx
nginx
finagle server
finagle server
Kafka
Storm
MongoDB
Real-time Processing System(Thunder)
Kafka
HDFS
MapReduce Jobs
HBase
kvproxy
Front End & Web
Batch Processing System(Iceberg)

Outline

- SDK
- 实时计算
- 离线计算
- 主要问题

SDK

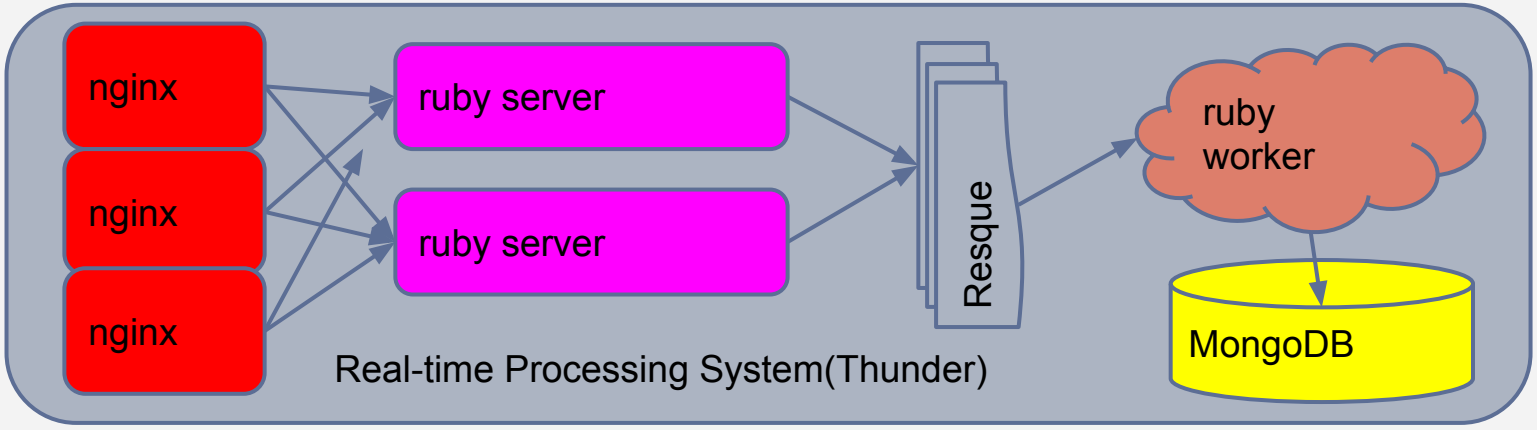
- 数据传输格式
  - JSON + gzip
  - 切换到thrift(增加schema, 减小数据包大小)
  - 校验(for spam)和去重(for unstable network)
- 发送模式
  - 启动时发送, 关闭时发送, 实时发送
  - 避免DDOS(推送使得大量设备同时对前端请求, 前段没有办法及时响应致使客户端超时, 可是客户端又使用固定间隔时长重试)

SDK(cont.)

- 模块化设计
  - 网络传输，设备数据
  - 加密解密，encoding
- 设备标识变迁
  - Android: imei + mac
  - iOS: udid, open udid, mac, idfa, idfv
  - 不同SDK版本能够获取设备信息不同
  - umid(umeng id) service 来统一设备标识
    - finagle + mysql(SSD) 40k/s

实时计算

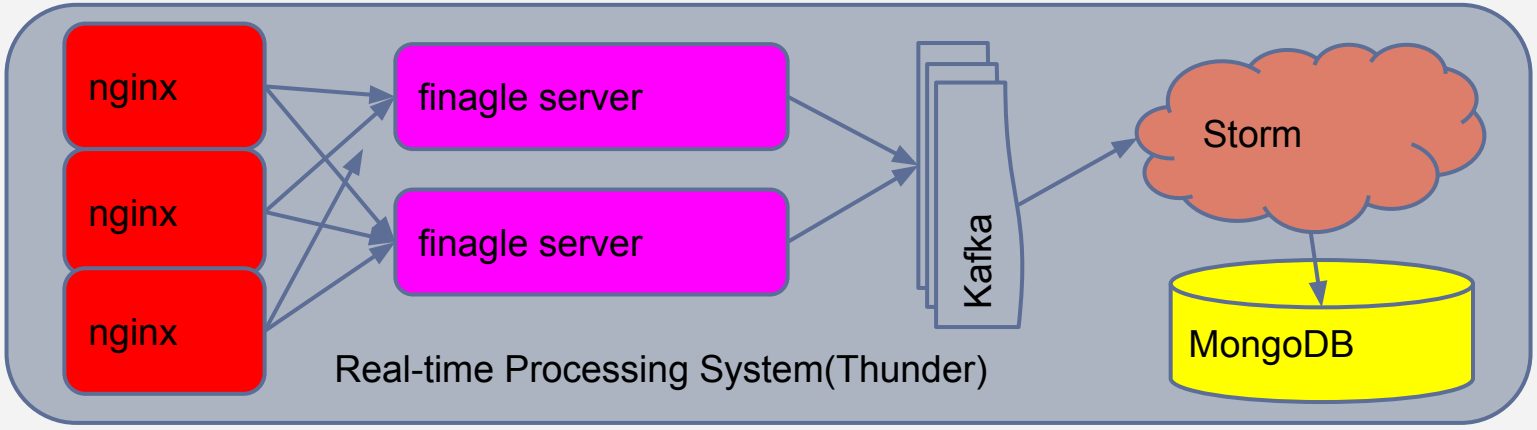
- 2010. Resque + Ruby worker 2k/s



- 简单直接，没有考虑性能情况
- MongoDB还没有出现性能和扩展性问题
- 没有在数据接入端做清洗和规范

实时计算(cont.)

- 2012. Kafka + Storm(batch store) 5k/s



- JVM replaces Ruby for better performance
- 在数据接入端做清洗和规范(计算umid, 检查mac是否合法, 校验和去重等)

实时计算(cont.)

- MongoDB
  - NO auto-sharding
  - write performance not good because of global lock(table lock is still not good enough)
  - SSD for boosting write performance

离线计算

- 初期误将离线系统当做在在线系统设计
  - 8台机器的微型集群
  - 过于依赖HBase, 中间结果存储于HBase
  - 原始日志存储elephant-bird(Protocol Buffer+ lzo)
- 减小对HBase依赖，中间结果存储于HDFS, HBase只存储最终结果
- 使用bulk import 来将结果载入HBase

离线计算(cont.)

- 重新设计HBase row-key避免单个region读写压力过大
- 定制化的查询中间层kvproxy
- 改进Hadoop提高集群利用率
  - 动态优先选择磁盘空间充裕机器执行任务(HDFS使用在80%以上时就需要考虑扩容)
  - map/reduce槽位混用,任务基于内存计算所需槽位(离线计算CPU使用少但是内存占用多)
  - lzma压缩支持(elephant-bird-lzma, Protocol Buffer + lzma) 用于存储冷原始日志

主要问题

- 大数据量的挑战 Go big or Go home
- 和设备标识做持续斗争
- 构建一个正确稳定的数据分析系统,统一实时计算和离线计算 (lambda arch) summingbird or spark.
- 密切关注开源社区
- Lambda Architecture
- Big Data: principles and best practices of scalable realtime data systems 推荐