揭秘京东大数据平台

京东大数据平台从无到有，从集中式到分布式，从Oracle数据仓库到JDW2.0，在演变过程中，一直在思索两个问题：

1. 如何建设电商特有的复杂业务的数据仓库？
2. 如何在保障安全的情况下降低使用数据的成本？

Part1:京东大数据分析及创新应用

目录 CONTENTS

京东大数据

商业思维面临的挑战

京东开放日 JD.COM 京东

中国电子商务市场依然在快速增长，“红利”时代还在延续

- **电商的人员构成（传统+互联网）**
- **电商的组织架构（业务主导 VS 数据支持）**
- **电商的商业目标（GMV VS Data-Driven）**

商业智能面临的挑战

京东技术 开放日 JD.COM 京东

传统BI

- 结构化数据
- 关系型数据库
- 数据规模一般TB级
- 集中式，数据向计算靠近
- 批处理为主
- 离线计算
- 报表展示
- 统计分析
- 使用算法
- 看数据

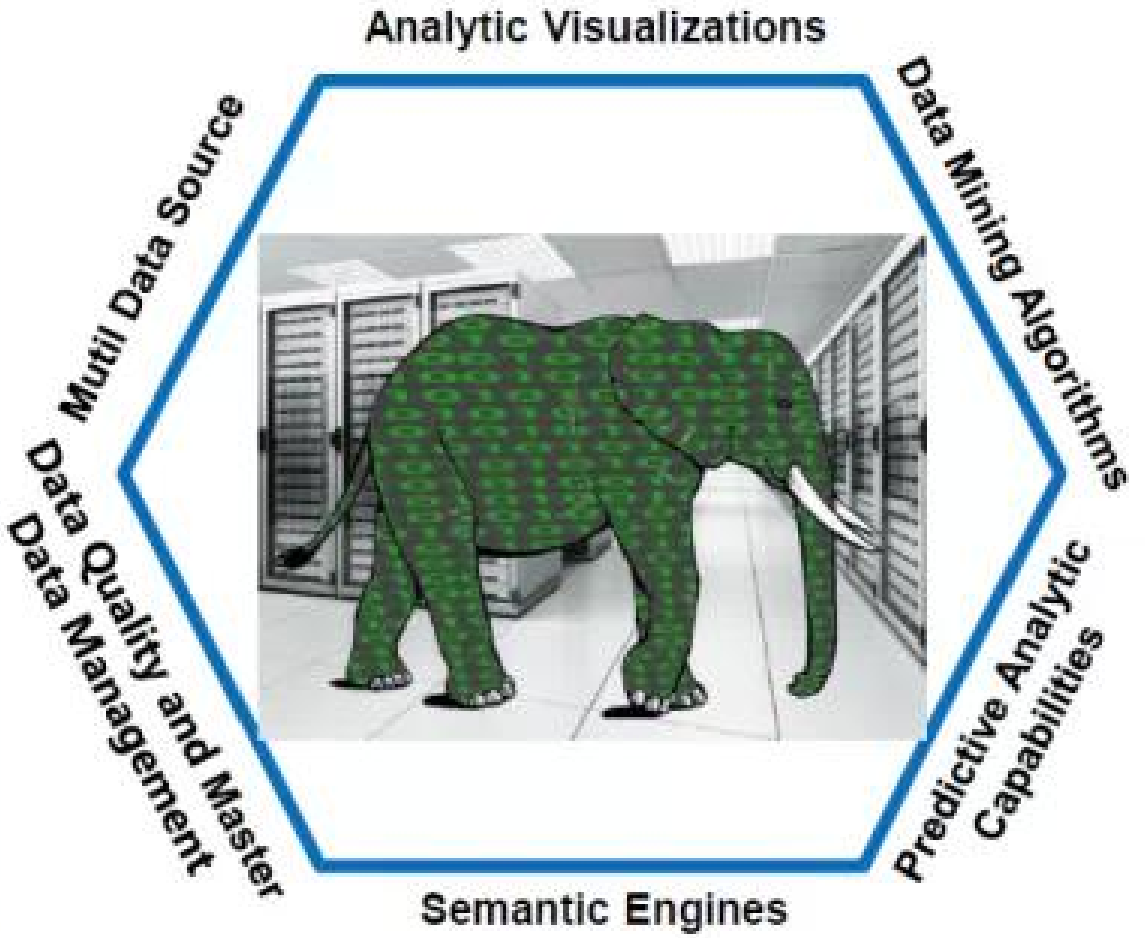
大数据时代的BI

- 非结构化数据+结构化数据
- 集群
- 数据规模从数十TB到PB级
- 分布式，计算向数据靠近
- 支持流式计算
- 实时分析+离线计算
- 智能决策
- 自动化分析
- 深度应用算法
- 解读数据

大数据分析/挖掘的方向

京东技术 开放日 JD.COM 京东

- ✓ 可视化分析
- ✓ 数据挖掘算法
- ✓ 预测性分析能力
- ✓ 语义引擎
- ✓ 数据质量和数据管理
- ✓ 数据来源多样化

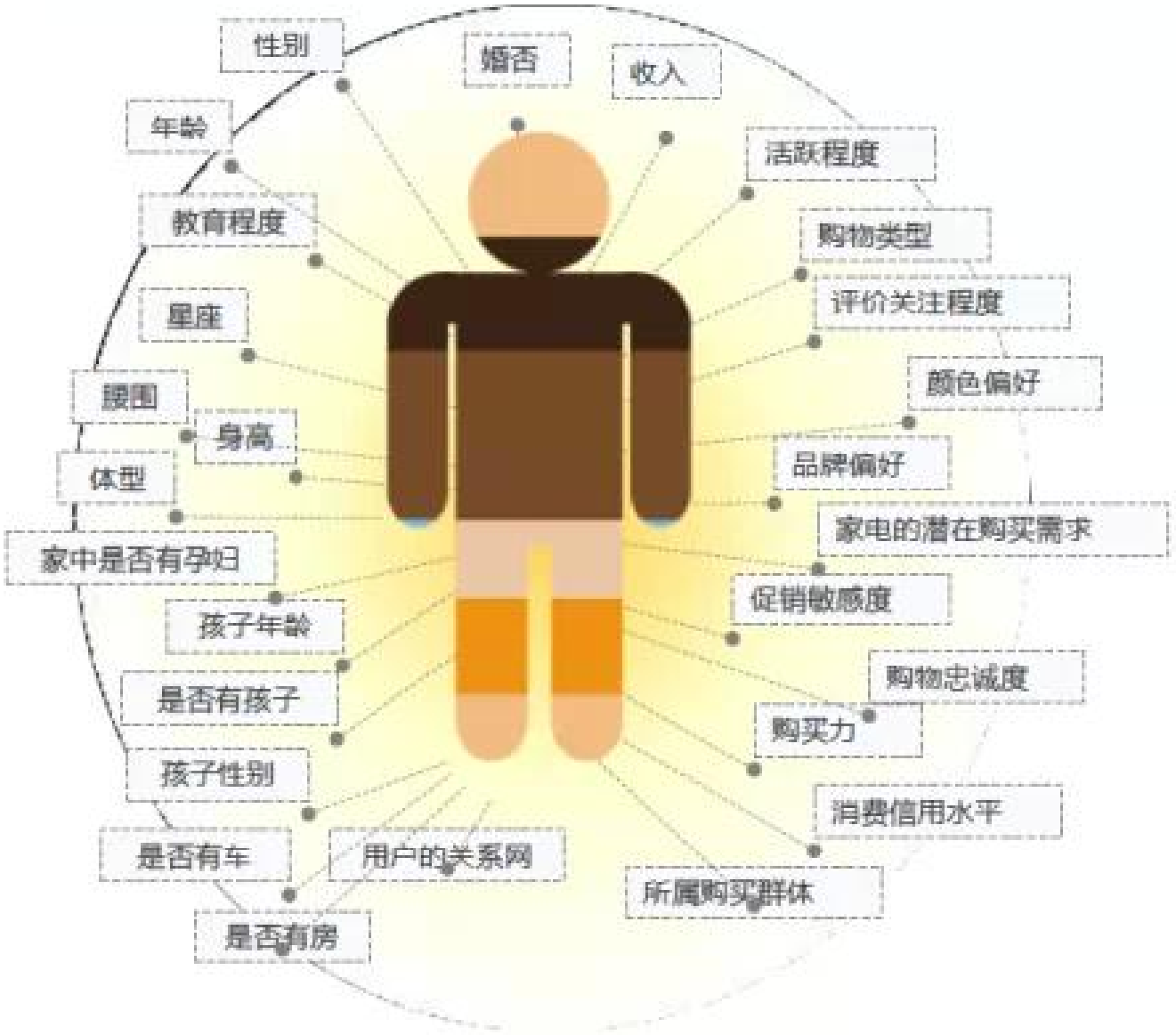


算法实践

京东技术开放日 JD.COM 京东

用户画像

京东技术 开放日 JD.COM 京东



用户分群

京东技术 开放日 JD.COM 京东

用户画像应用

京东技术 开放日 JD.COM 京东

用户画像技术

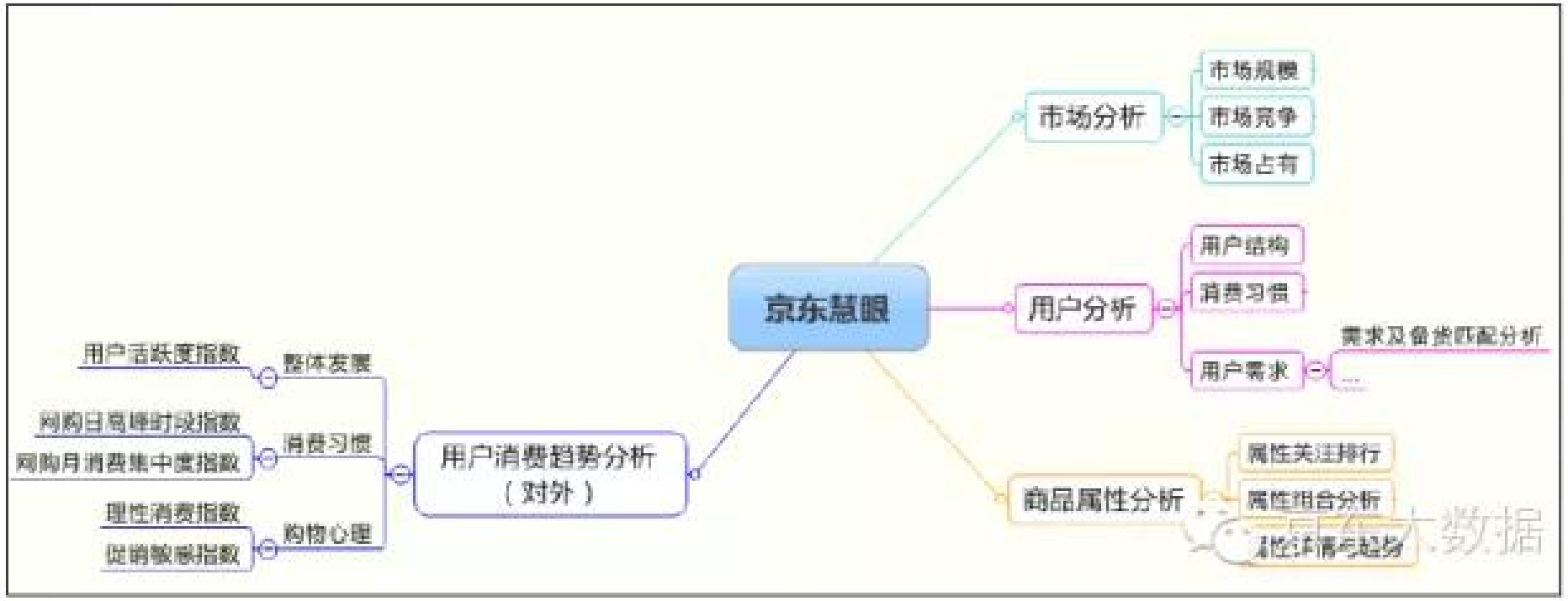
用户画像	基本属性	购买能力	行为特征	兴趣爱好	心理特征	社交网络
行为建模	文本挖掘	自然语言处理	机器学习	预测算法	聚类算法	
数据收集	网络日志数据	用户行为数据	网站交易数据			

京东慧眼：C2B智能决策

京东技术 开放日 JD.COM 京东

基于电商大数据，打造C2B智能决策系统：京东慧眼

- 市场分析：从规模、竞争程度及各品牌占有率对全品类进行市场分析
- 用户分析：从人群结构、购物习惯、需求分布，对全品类进行用户分析
- 商品属性分析：从商品扩展属性的消费关注排行、属性组合，属性趋势与详情，对全品类进行商品分析
- 消费趋势分析：从消费者的“整体发展”，“消费习惯”，“购物心理”等角度，对外发布对政府、行业有价值的消费趋势指数



JDPhone的实践

京东技术开放日 JD.COM 京东

JDPhone计划

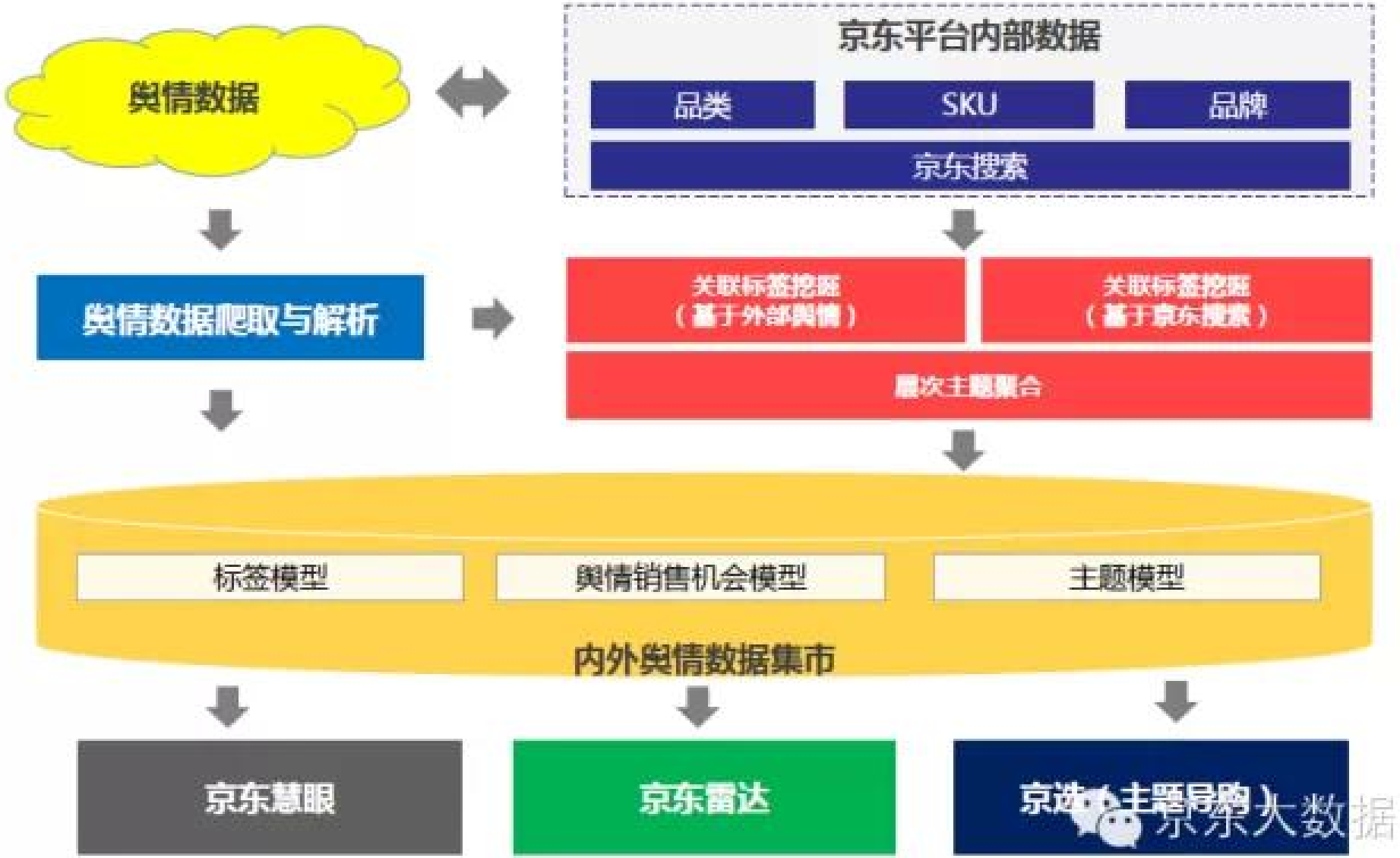
- JDPhone手机销售火爆，“**第一夫人**”**的选择**
- 基于浏览、搜索、评论等**亿级**数据进行用户需求建模与挖掘
- **15个**手机品类重点硬件规格的深度分析
- **200+**手机品牌进行潜力分析，合作品牌进行深度评估分析
- **5个**深度硬件生产配置方案
- 利用**用户画像**技术进行目标用户的结构及趋势分析

用户的生活圈

京东技术 开放日 JD.COM 京东

京东技术开放日 JD.COM 京东

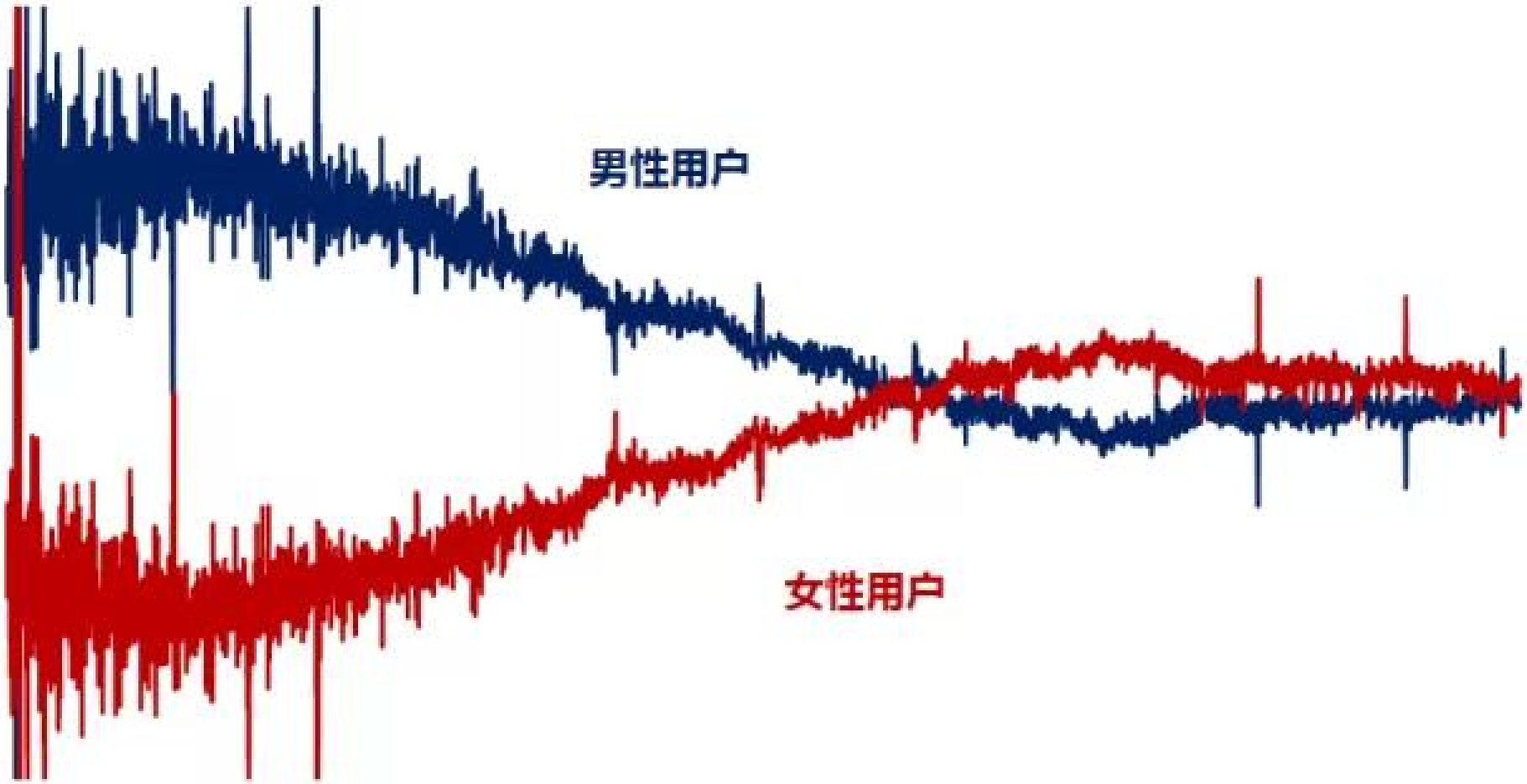
舆情深度挖掘


舆情数据
京东平台内部数据
品类
SKU
品牌
京东搜索
舆情数据爬取与解析
关联标签挖掘
（基于外部舆情）
关联标签挖掘
（基于京东搜索）
层次主题聚合
标签模型
舆情销售机会模型
主题模型
内外舆情数据集市
京东慧眼
京东雷达
京选（主题导购）

电商用户群体趋向男女均衡

京东技术开放日 JD.COM 京东

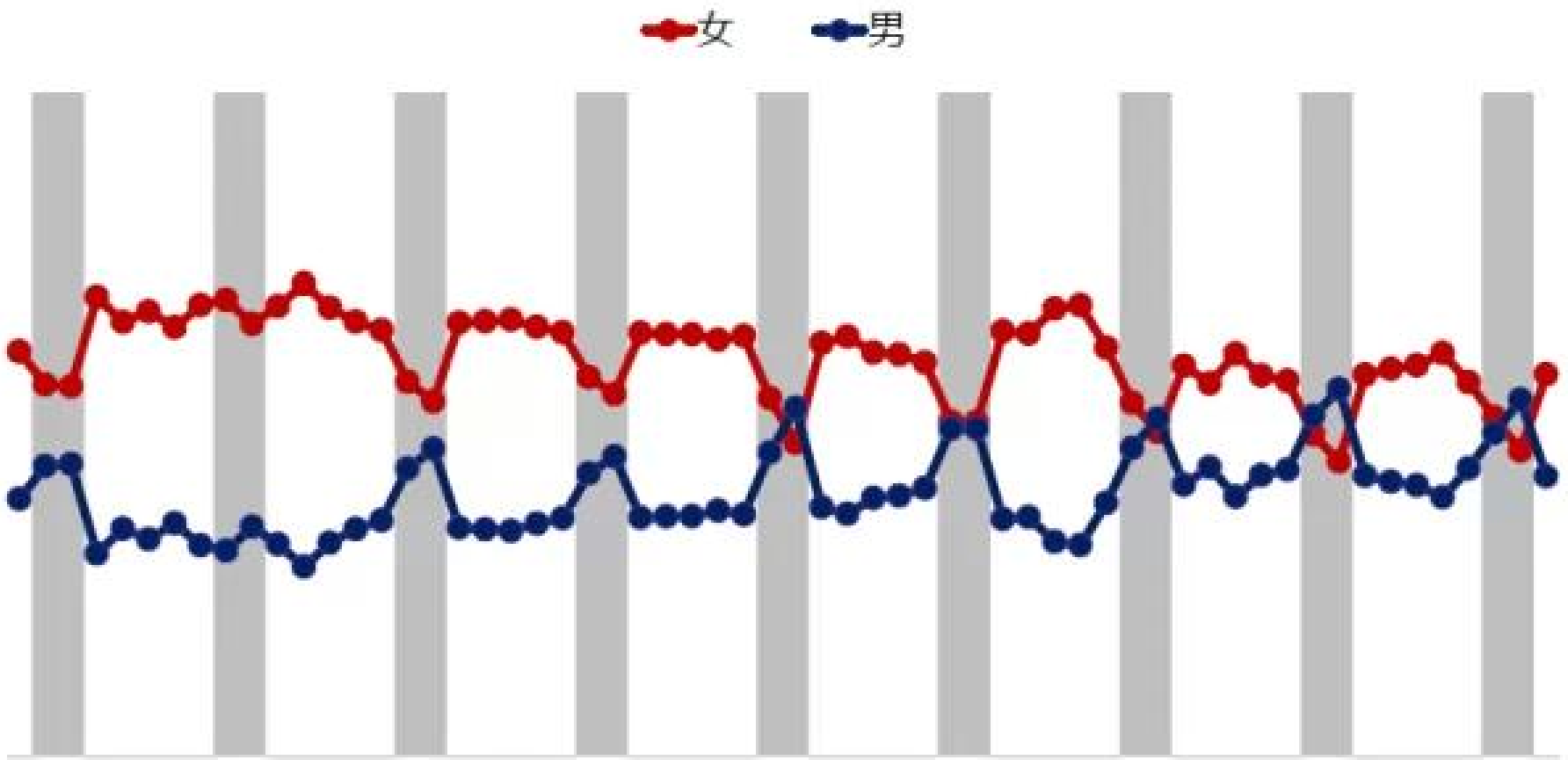
- ✓ B2C网上购物人群的性别比例正在悄然变化，正趋向男女平衡



线上线下购物将互补发展

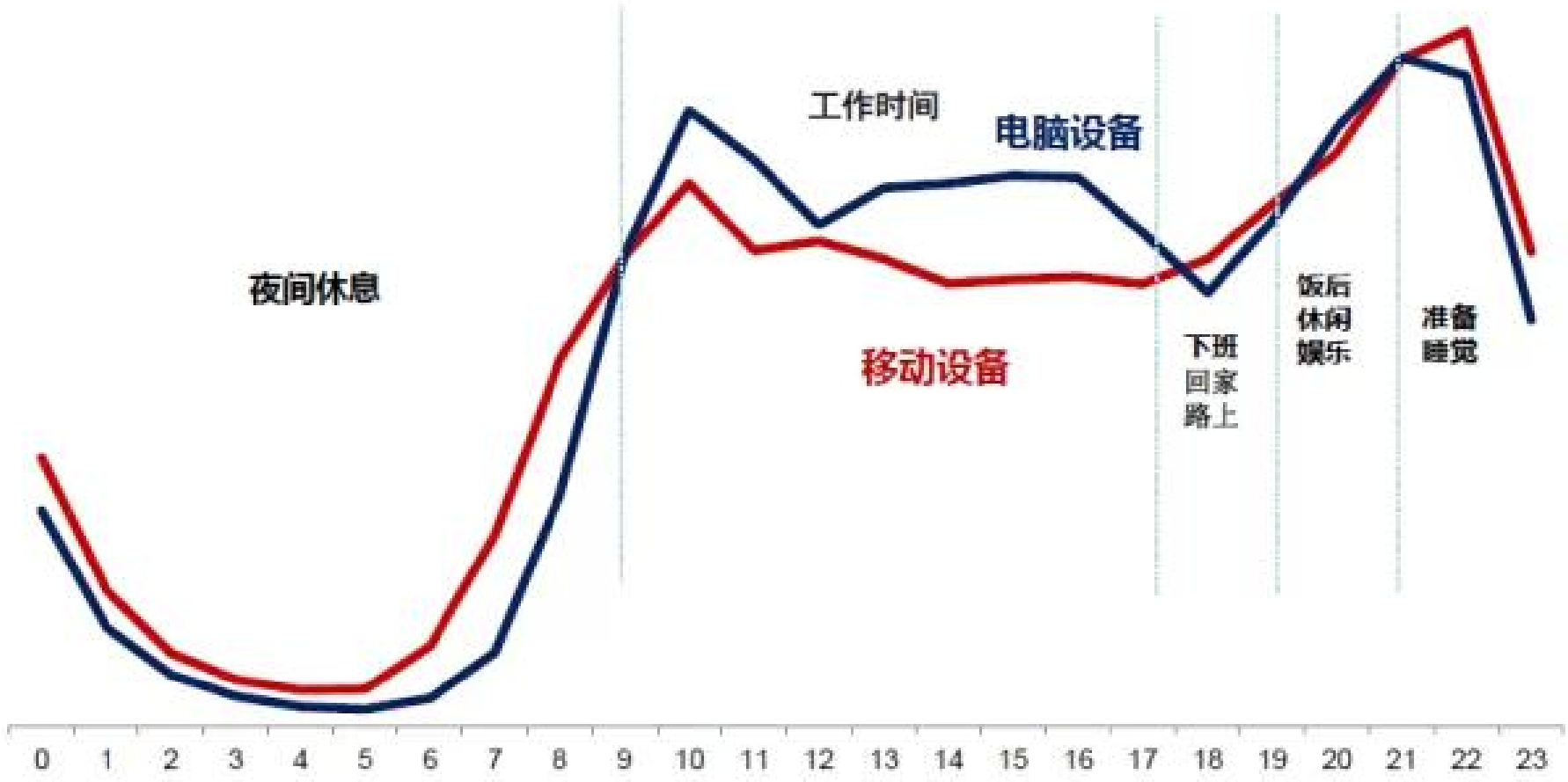
京东技术 开放日 JD.COM 京东

✓ 网购女性用户更喜欢选择在上班期间购物，而节假日还延续逛街习惯



移动购物成为电商未来趋势

京东技术 开放日 JD.COM 京东



Part 2 京东大数据基础架构和实践

目录

- 大数据
- 京东大数据平台
- 我们技术突破
- JDW&Jmart
- JDMP数据挖掘平台
- 展望

京东大数据

少说些漂亮话,多做些日常平凡的事情

京东大数据平台

京东技术开放日 JD.COM 京东

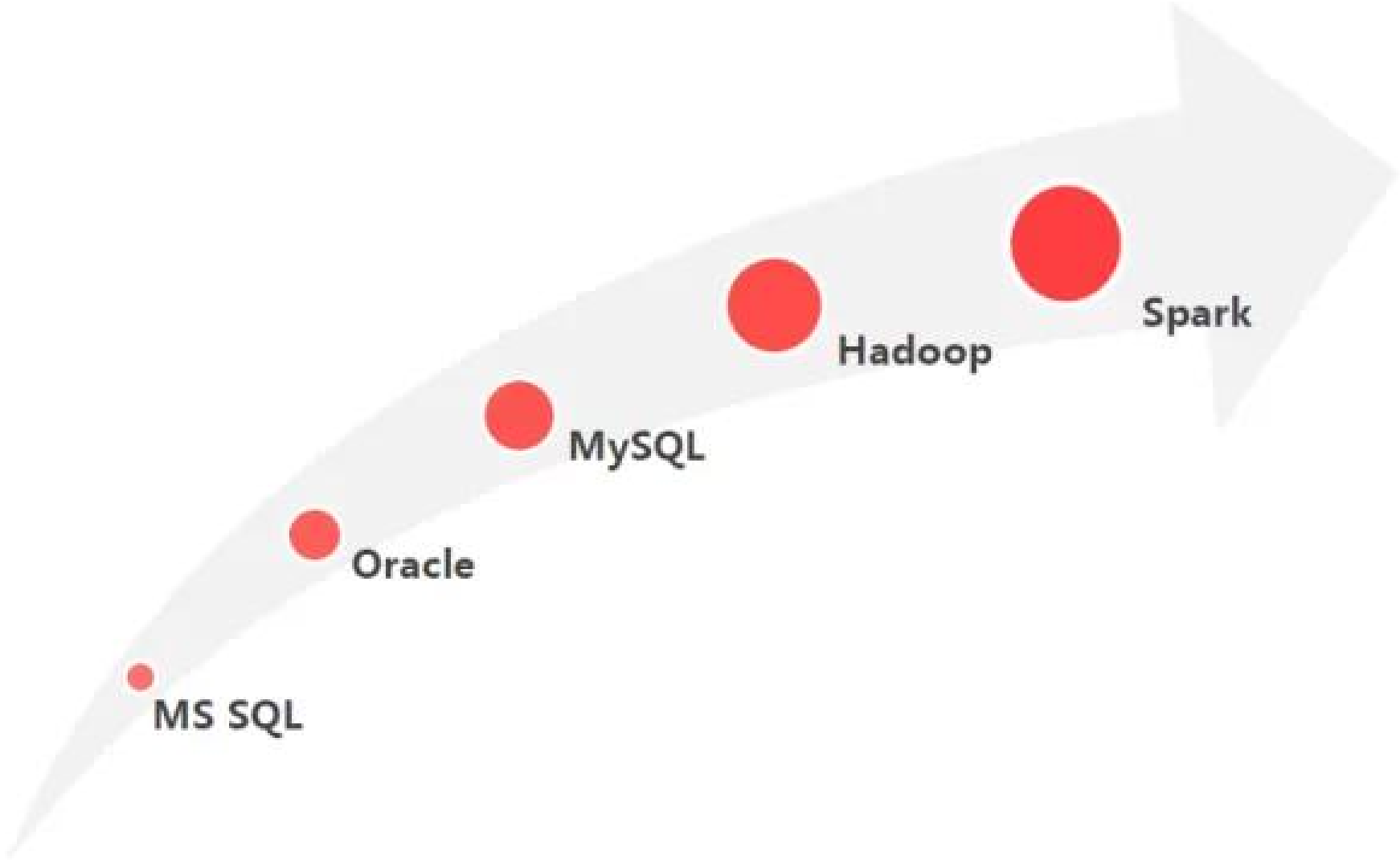
我们的技术突破

京东技术开放日 JD.COM 京东

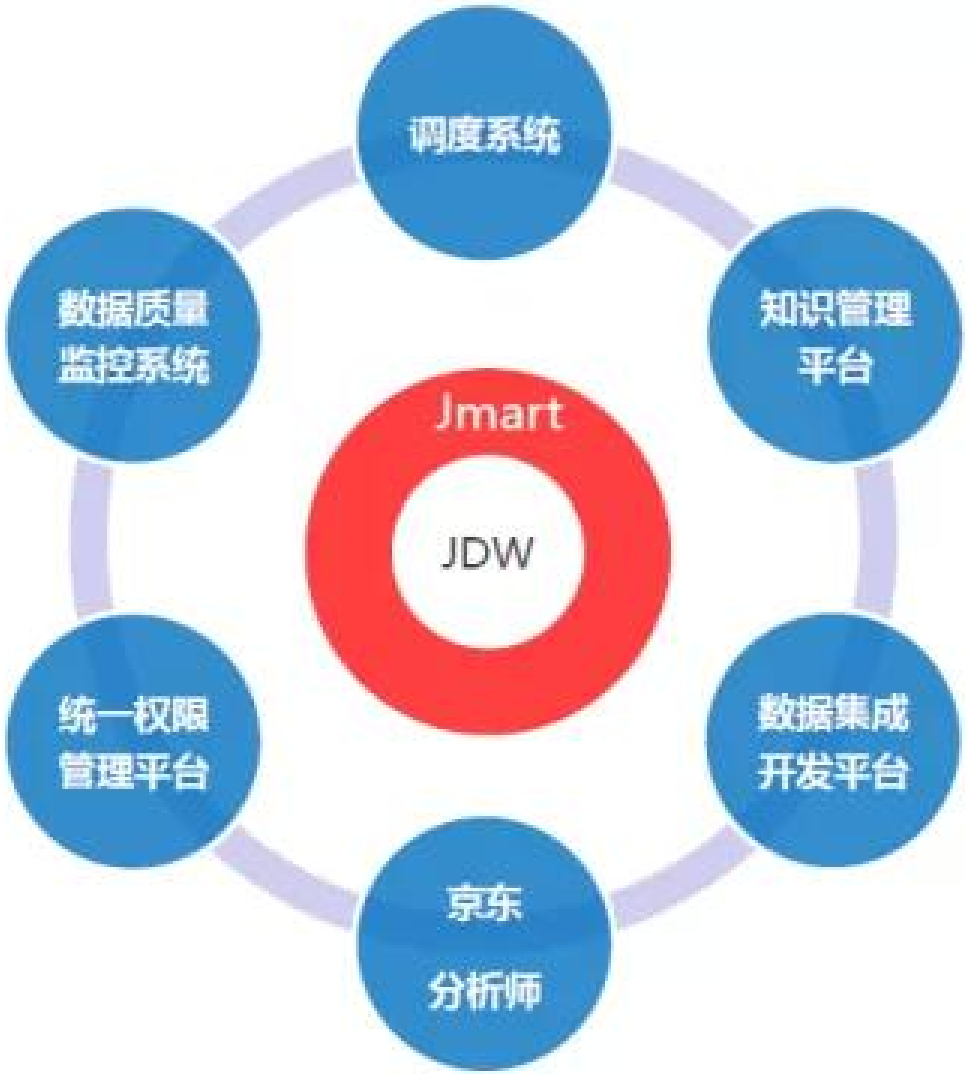
- **分布式系统技术突破**
  - 稳定性、性能、HA、故障恢复、多集群、运维和管理
- **多用户共用平台**
  - 数据安全、隐私保护
- **数据任务运行监控**
  - 每日数万个数据任务、核心任务及时性
- **挖掘数据价值**
  - 数据量大、迭代效率
- **数据实时化**
  - 关系型数据、AD HOC、实时计算
- **离线、实时平台合并**
  - Hadoop、Spark、Storm

JDW发展历程

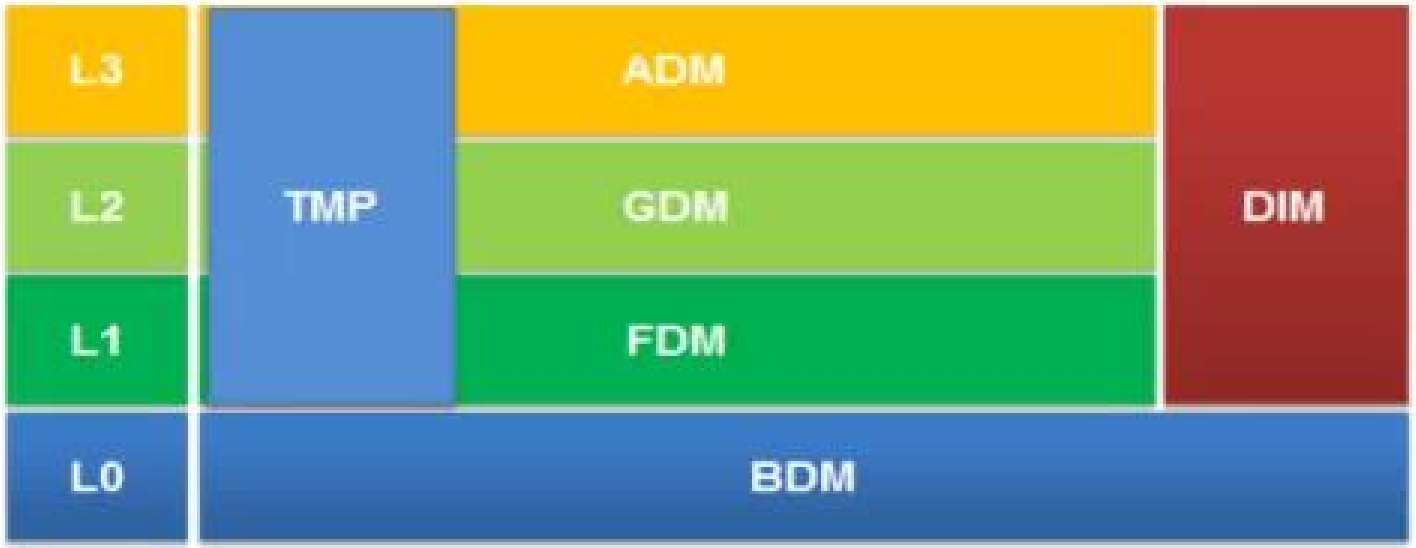
京东技术开放日 JD.COM 京东



JDW架构-概述



EDW的核心数据架构分为四层：**缓冲数据层、基础数据层、通用数据层、聚合数据层**，其次是**临时层**和**维度层**。其示意图如下：



Buffering Data Model	缓冲数据层
Fundamental Data Model	基础数据层
General Data Model	通用数据层
Aggregative Data Model	聚合数据层
DIM	维度数据库
TMP	计算中间库/临时数据库

JDW FDM 存储方案优化

京东技术 开放日 JD.COM 京东

在线交易系统、商品中心、用户中心等出于效率的考虑，不会长期保存大量历史数据，而JDW作为企业数据分析及挖掘的基础设施，天生具有保存历史数据的职责，非但如此，如何快速、高效的获取历史上任意一天的快照数据也成为设计历史数据存放方式时的重要考量。通过比较，记录数据的**生命周期**；能**快速还原任意天的历史快照**，极大的**节省了存储**

- **快照的还原**

```
SELECT * FROM t_chain
WHERE P_DATE>=start_date
  AND  P_DATE<end_date
```

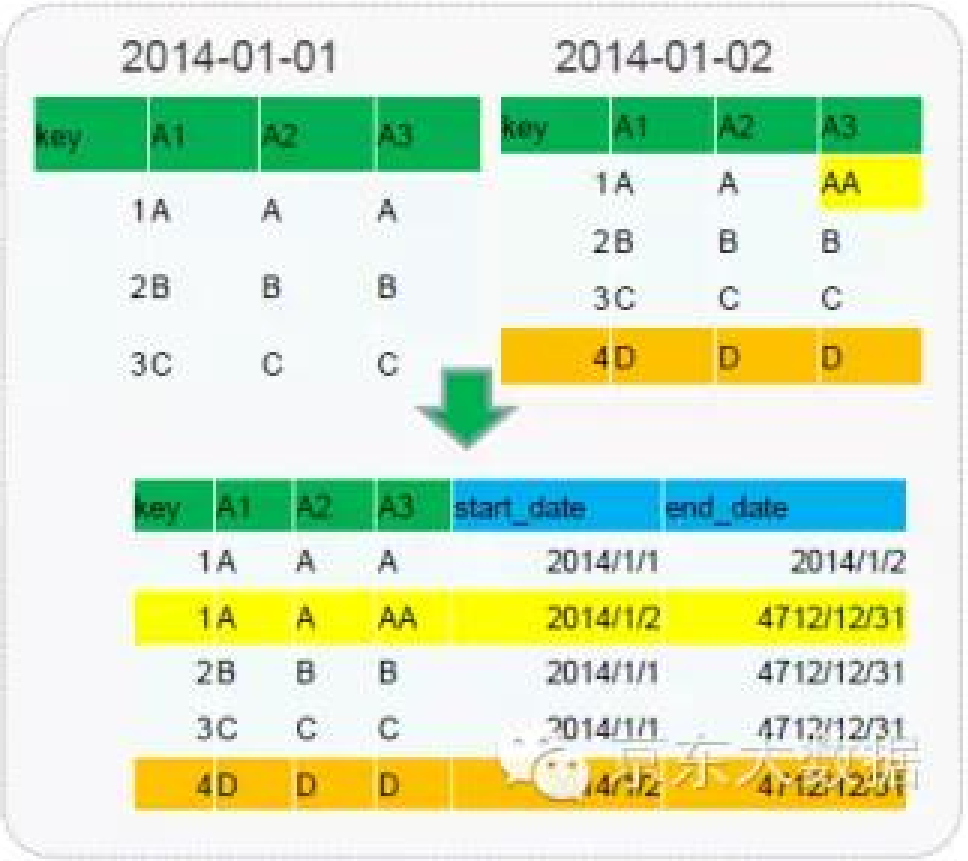
- **空间节省率**

$$Ratio = 1 - \frac{base + \frac{1+N}{2} * C + M * N}{base * N + (1+N) * C/2}$$

Base：基数（十亿级）　　N：天数

C：每日增量（百万级）

M：每日变化量（百万级）



JDW FDM 存储方案优化

京东技术开放日 JD.COM 京东

实际的拉链表的设计更加精巧，充分考虑了京东业务的各种情况。

譬如订单表，考虑了其结转特点，充分利用了数据分拣技术，

如下是订单表的**多分区**设计，

```
dp=HISTORY/dt=2014-06-22/end_date=4712-12-31
.....
dp=HISTORY/dt=2014-06-23/end_date=4712-12-31
dp=HISTORY/dt=2014-06-24/end_date=4712-12-31
.....
dp=EXPIRED/dt=2013-10-11/end_date=2013-10-11
dp=EXPIRED/dt=2013-10-12/end_date=2013-10-12
dp=EXPIRED/dt=2013-10-13/end_date=2013-10-13
dp=EXPIRED/dt=2013-10-14/end_date=2013-10-14
.....
dp=ACTIVE/dt=4712-12-31/end_date=4712-12-31
```

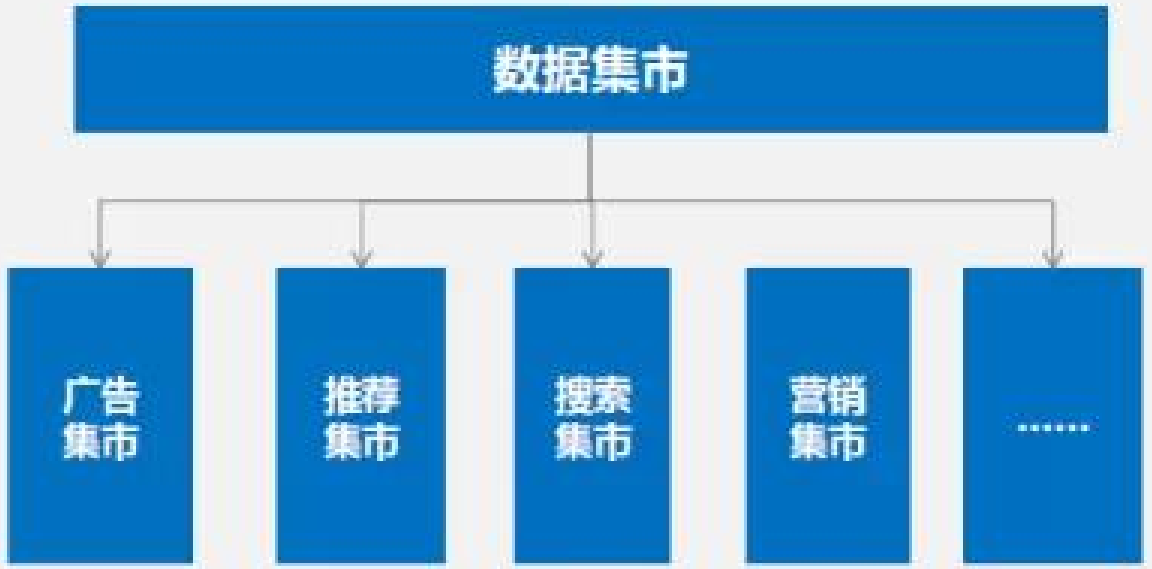
之前，采取快照累积的方式，订单表加工完成甚至到下午1点，

利用该方法后，一般在凌晨3左右点完成。

而且在后续的查询上，能充分利用分区裁剪特性，能迅速的返回数据

Jmart概述

京东数据集市是基于JDW构建的面向条线的数据生产环境，为各条线提供数据应用服务，包含广告、推荐、搜索、财务、营销、运营、BDA、移动、拍拍等数十个部门，上千用户提供数据服务。



JDMP-大数据挖掘中的“痛点

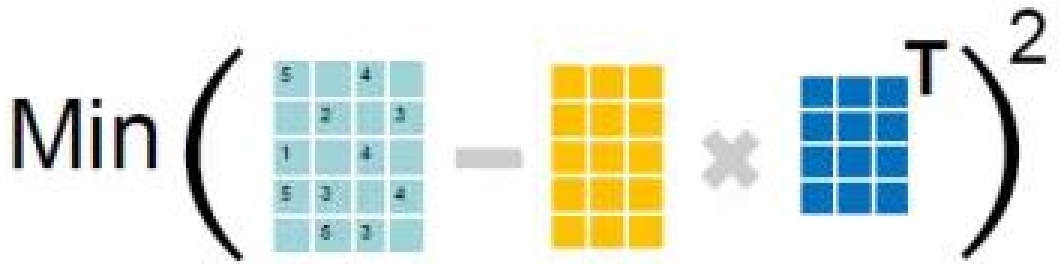
京东技术开放日 JD.COM 京东

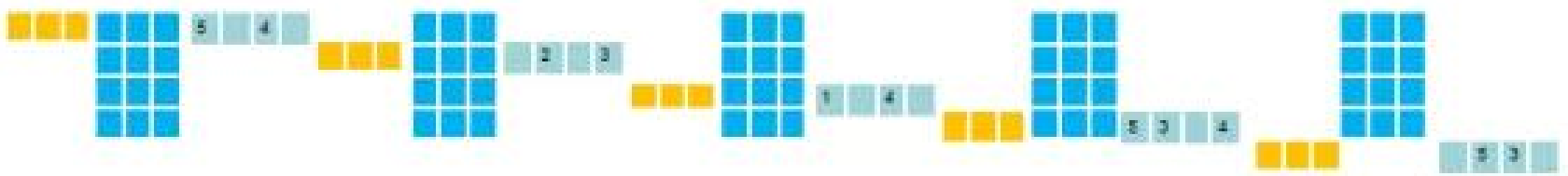
JDMP-支持的算法

京东技术开放日 JD.COM 京东

推荐	ALS-MF， FP-Growth，Item/User-CF，RBM
分类	LR，NB，SVM，gbdt，soft-max
回归	linear，ridge，lasso
聚类	k-means
主题模型	LDA，PLSA

JDMP-分布式ALS





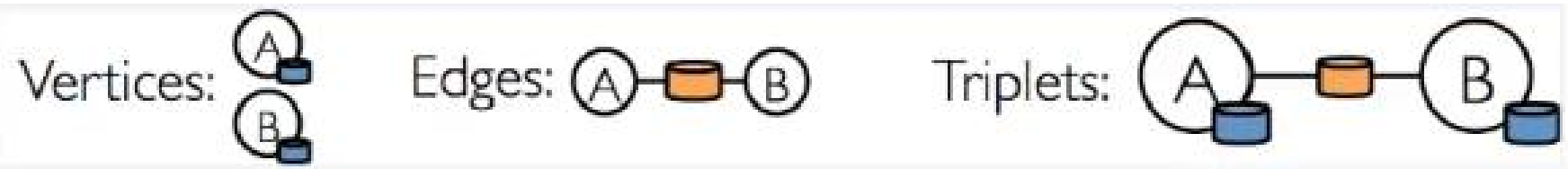
JDMP-Graphx图计算

京东技术 开放日 JD.COM 京东

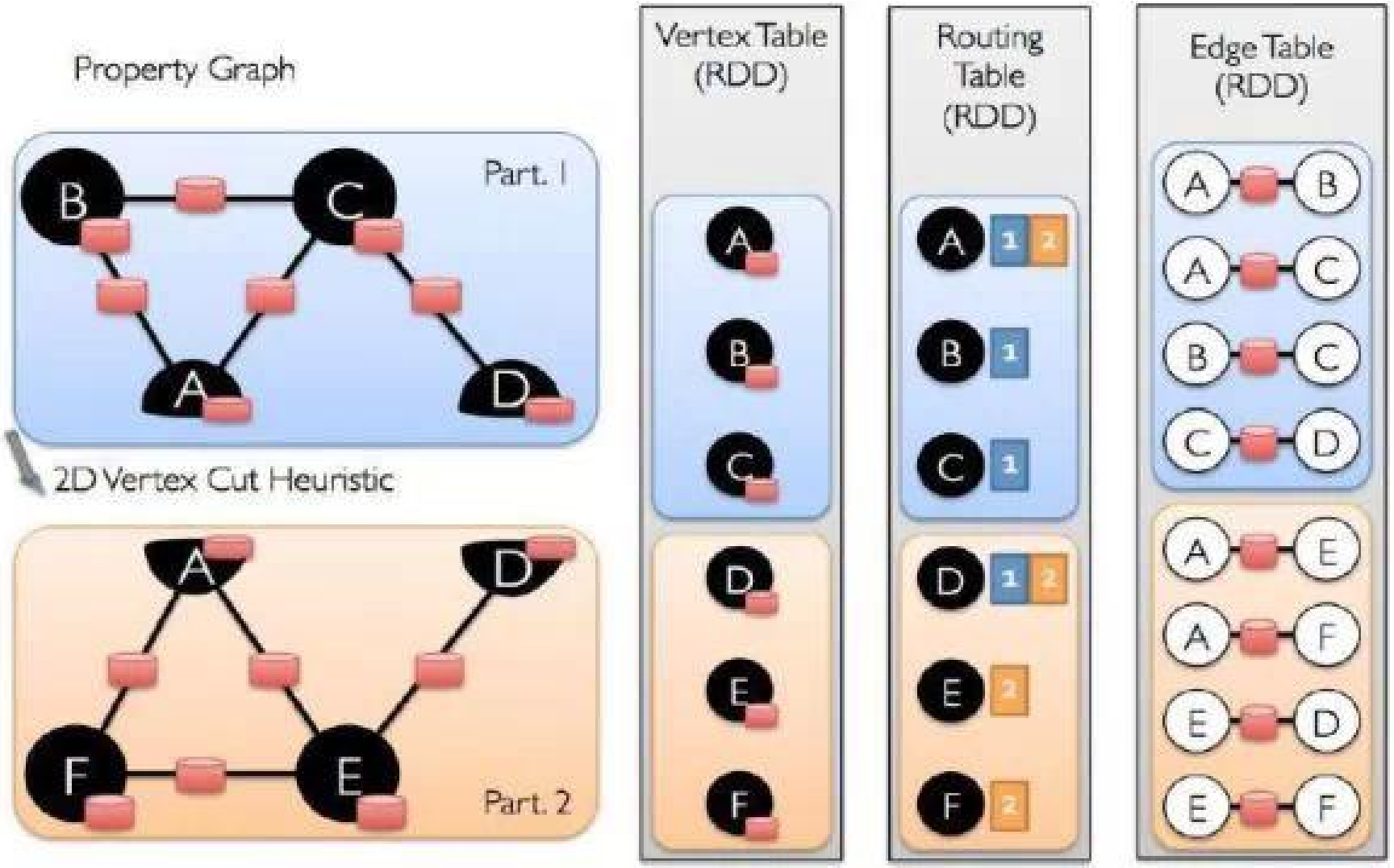
Vertices： 由顶点、顶点属性构成

Edges ： 由顶点、边属性构成

Triplets ： 由顶点、顶点属性和边属性构成



JDMP-Graphx图计算



Part 3：京东数据仓库海量数据交换工具-Plumber开发实践

目 录 CONTENTS

京东大数据

技术突破

京东技术 开放日 JD.COM 京东

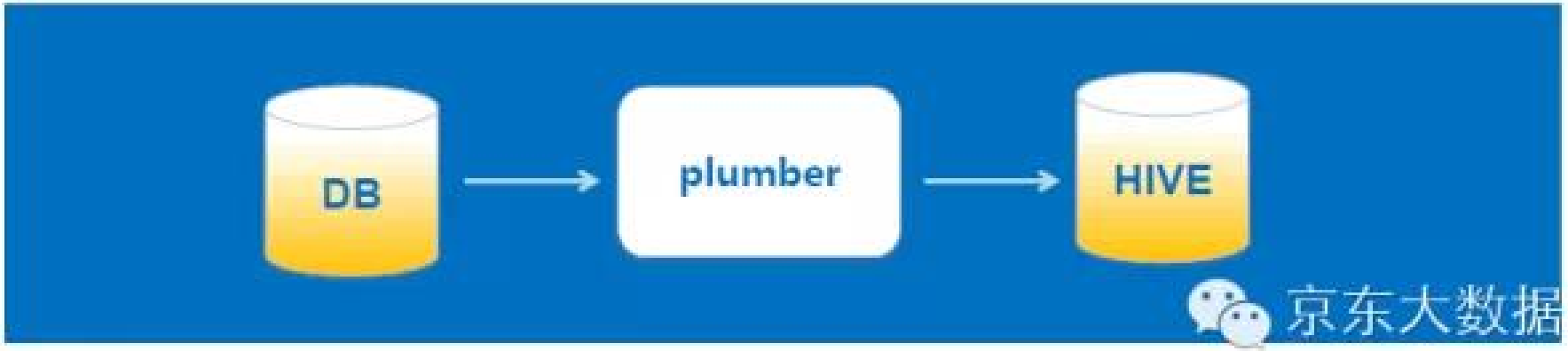
- **海量**
  - 每日进出上TB数据
  - 每天数千数据传输任务
- **异构**
  - 结构化：mysql, sqlserver, oracle, hive
  - 非结构化: mongodb, hbase, log
- **场景复杂**
  - mysql分库分表
  - 全国各地仓库数据抽取

流程优化

京东技术 开放日 JD.COM 京东

三次传输　三次落地　一次清洗

一次传输　无落地　实时清洗



技术特点及实现方案

京东技术 开放日 JD.COM 京东

- 读写分离插件化
- 多线程并行执行
- 配置化和实时统计信息
- 定制化开发全国仓库抽取

读写插件化

Reader： mysql, sqlserver, oracle, mongodb, hive, log

Writer ： hive, mysql, oracle, hbase

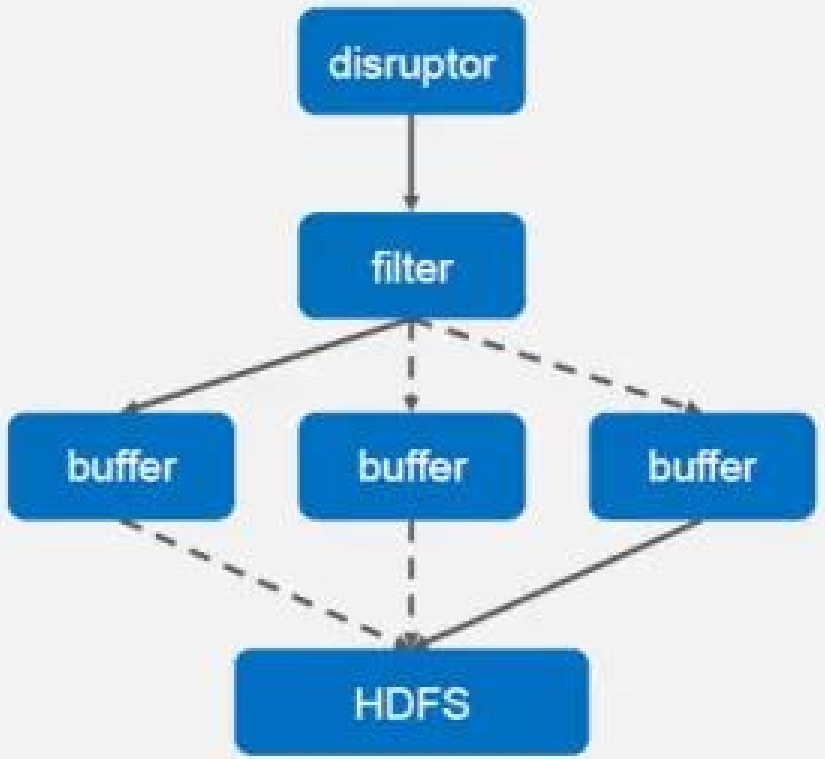
插件实现

Reader

- RDBMS – JDBC
- NOSQL – API
- LOG – http断点续传

Writer

- RDBMS – JDBC
- Hive – write hdfs & add partitions



hdfs w.. r

多线程并行执行

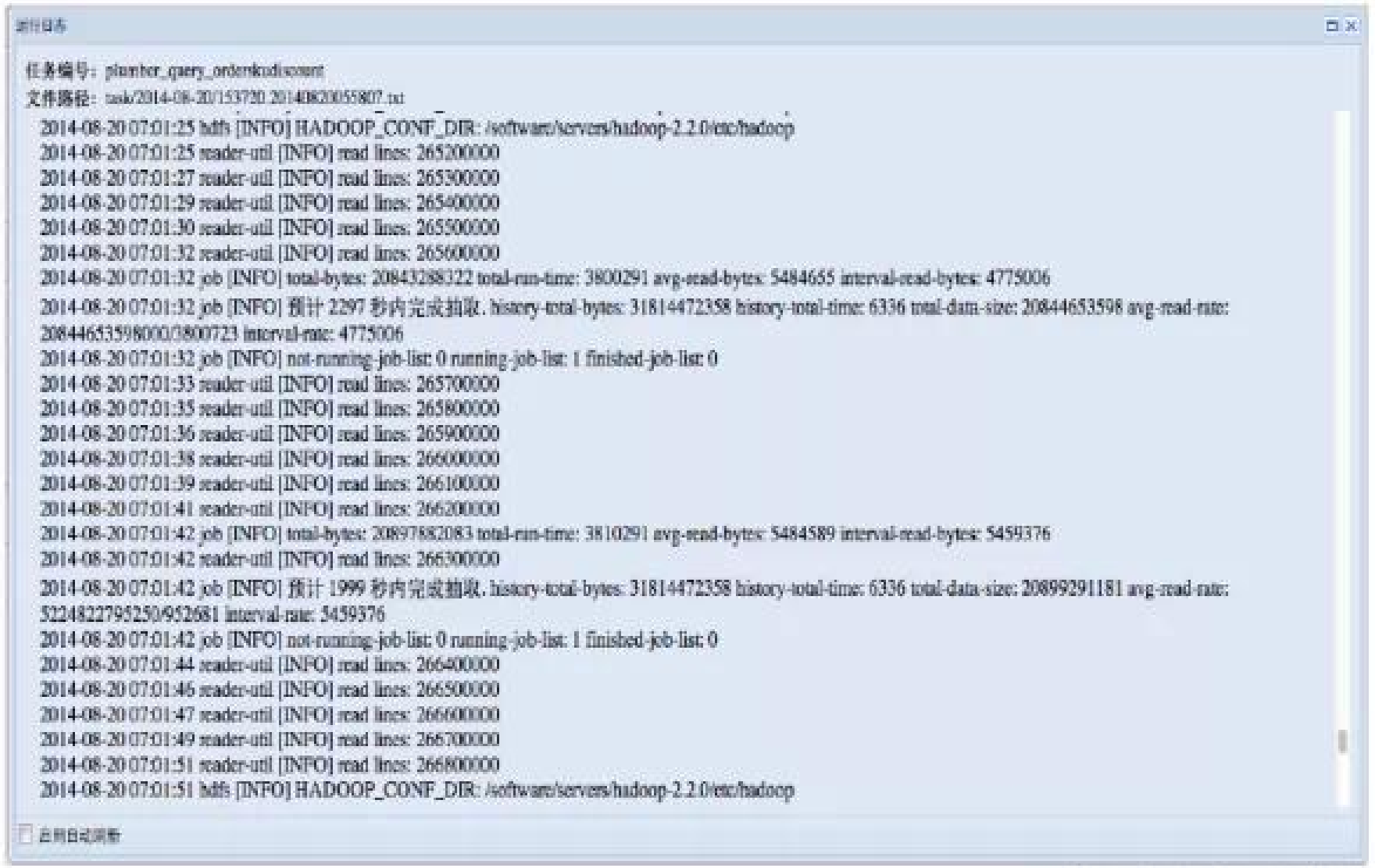
并行执行任务

- 分库分表，库名表名sql拆分

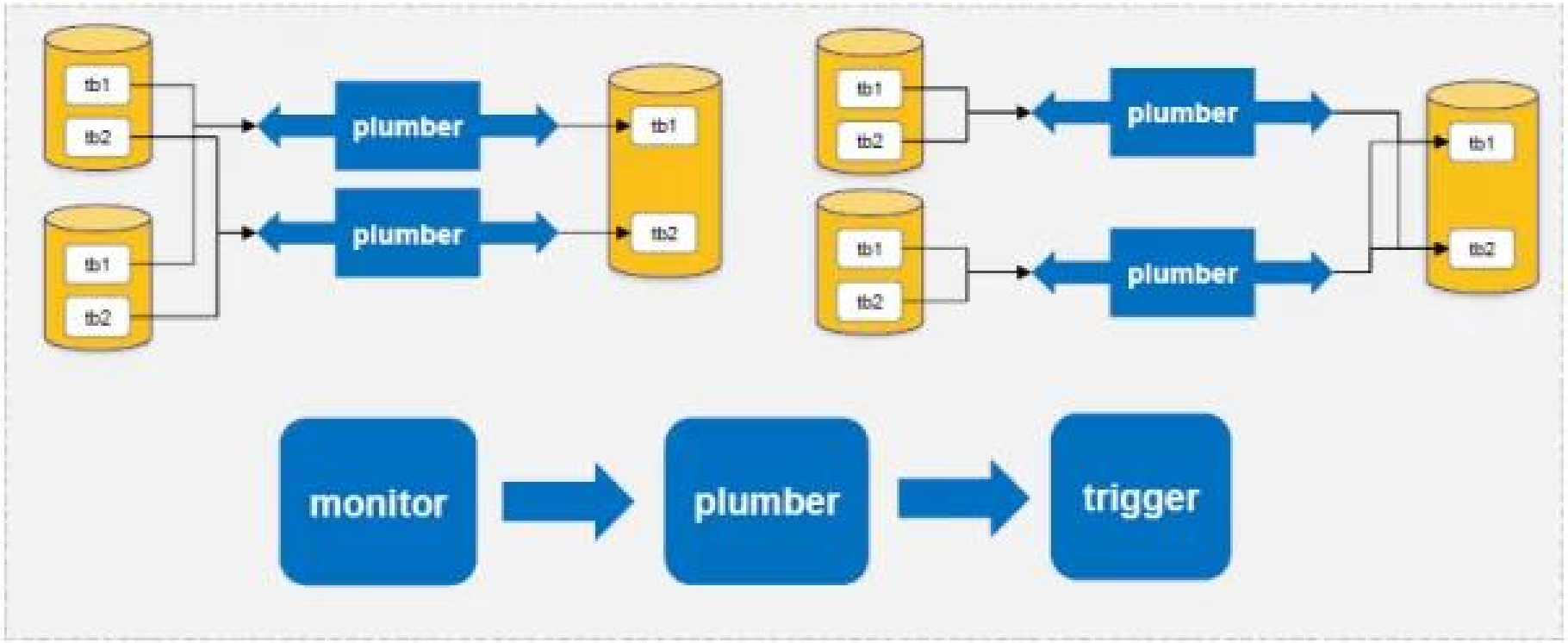
资源有效利用

- 根据系统资源增加线程数

调度模板和实时监控



定制化全国仓库数据抽取



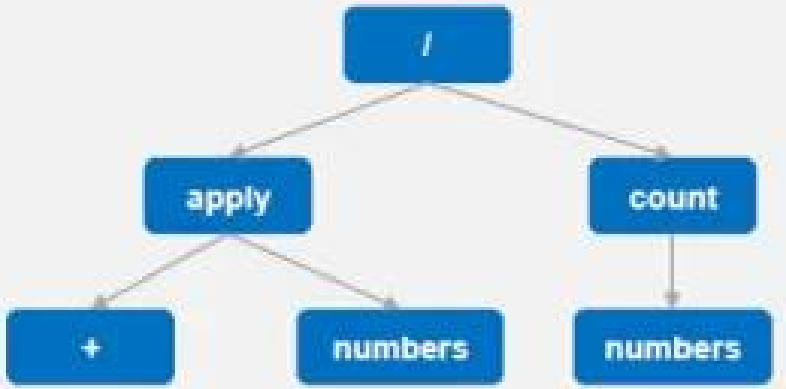
- 仓库分处全国各地，网络情况不确定性大
- 各地仓库下班时间不一，可抽取时间点不一
- 个别仓库宕机不能影响第二天全国仓库报表生成时间点

为什么是clojure

Clojure

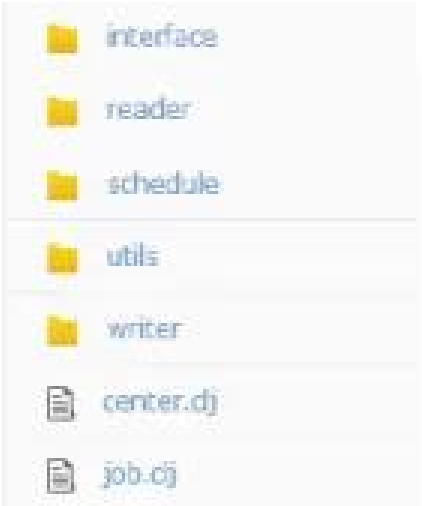
- 纯函数式
- 代码即数据
- 代码即AST
- JVM上的Lisp

```
(defn average
  [numbers]
  (/ (apply + numbers) (count numbers)))
```




```
User=> (average [ 60 80 100 400])
160
```

代码结构




```
localhost:/tmp/plumber/src/plumber/interface phoenix$wc -l *
     113 dispatch.clj
     112 dispatch_spider.clj
      25 hbase.clj
      31 hive.clj
      35 http.clj
      34 interface.clj
      34 load.clj
     111 local.clj
      35 mongo.clj
      33 rdb.clj
     563 total
```

```
localhost:/tmp/plumber/src/plumber/reader phoenix$wc -l *
      14 db_reader.clj
      29 hdfs_reader.clj
      38 hive_reader.clj
      18 http_reader.clj
      92 mongo_reader.clj
       9 mysql_reader.clj
      10 oracle_reader.clj
      56 reader.clj
      36 spider_reader.clj
      10 sqlserver_reader.clj
     312 total
```

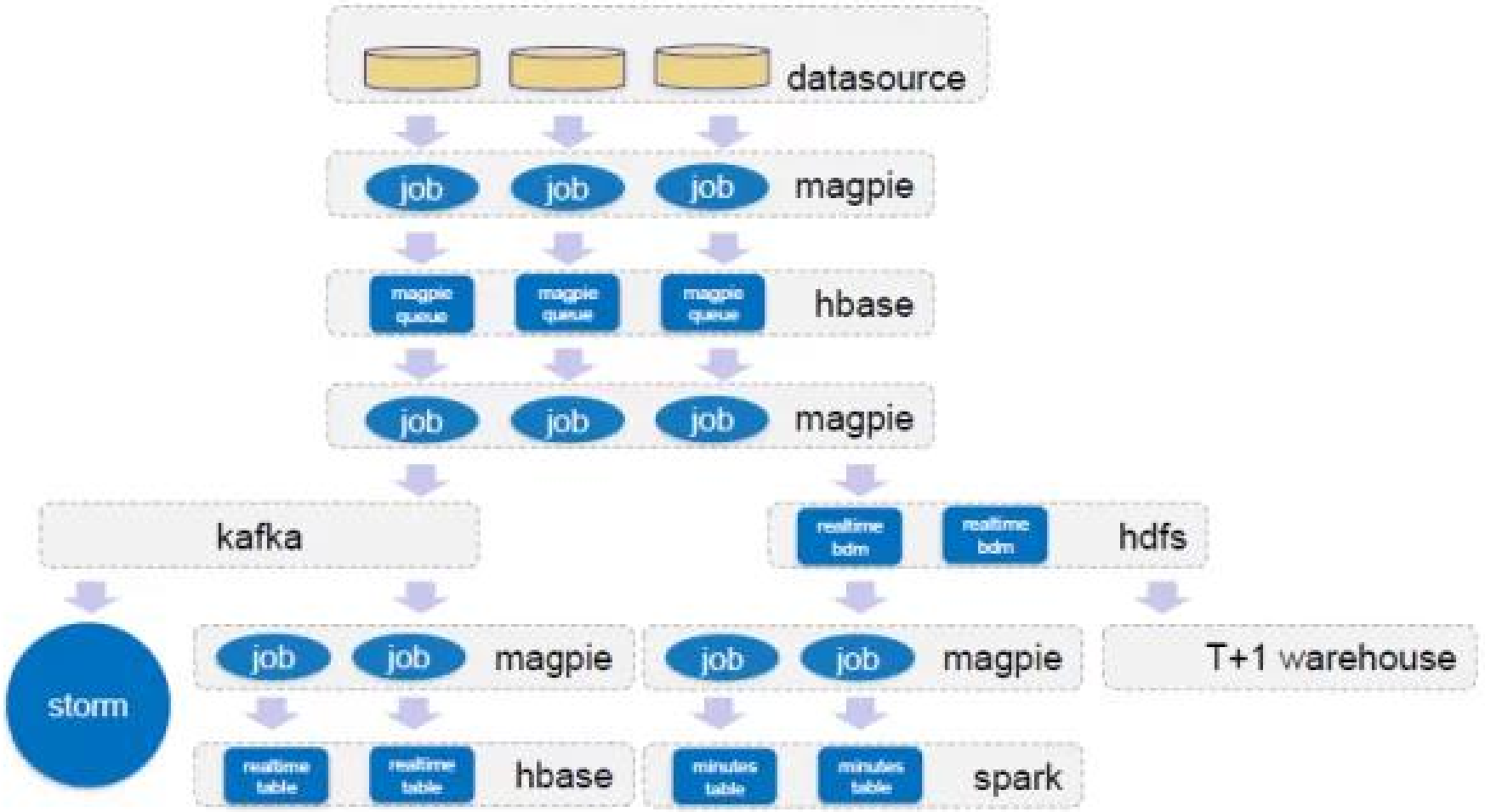
```
localhost:/tmp/plumber/src/plumber/writer phoenix$wc -l *
     135 hbase_writer.clj
     141 hbase_writer_dispatch.clj
     156 hdfs_writer.clj
     126 local_infile.clj
     124 local_infile.clj.back
      68 localfs_writer.clj
     101 mysql_load_writer.clj
     102 rdbms_writer.clj
     147 spider_writer.clj
     108 writer.clj
    1208 total
```

多线程编程


```clojure
(defn- sub-job
  "[illegible]"
  [sub-job-conf]
  (let [reader-config (atom (:reader sub-job-conf))
        writer-config (:writer sub-job-conf)
        reader-result (ref {:finished false :success false })
        writer-result (ref {:finished false :success false})
        monitor-interval 10000
        total-bytes (atom 0) ;;[illegible]
        last-read-bytes (atom 0)
        avg-read-rate (atom 0) ;; [illegible]
        interval-read-rate (atom 0);;[illegible] interval [illegible]
        start-time (System/currentTimeMillis)
        last-static-time (atom start-time)
        static-fn (fn [total-read-bytes] ;;[illegible] writer [illegible]
                    (let [now (System/currentTimeMillis)
                          _ (reset! total-bytes total-read-bytes)
                          interval (- now @last-static-time)]
                      (if (>= interval monitor-interval)
                        (let [interval-read-bytes (- total-read-bytes @last-read-bytes)
                              total-run-time (- now start-time)
                              _ (reset! last-read-bytes total-read-bytes)
                              _ (reset! last-static-time now)
                              _ (reset! avg-read-rate (long (/ total-read-bytes (/ total-run-time 1000))))
                              _ (reset! interval-read-rate (long (/ interval-read-bytes (/ interval 1000))))]
                          (log/info "total-bytes:" @total-bytes "total-run-time:" total-run-time "avg-read-bytes:" @avg-read-rate "interval-read-bytes:" @interval-read-rate)))))]
    (reify Job
      (init [this] (log/debug "init sub-job" @reader-config) )
      (start [this]
        (let [exec (java.util.concurrent.Executors/newCachedThreadPool)
              [disruptor publisher writer] (d-util/wire-up-disruptor exec writer-config static-fn)
              reader-future (future (r/reader @reader-config publisher))
              writer-future (future (writer))
              listener (fn [] (do (while (not (future-done? writer-future))
                                   (try
                                     (Thread/sleep 1000)
                                     (catch Exception e (log/error e))))
                                 (d-util/shutdown-disruptor disruptor exec)
                                 (dosync (ref-set reader-result @reader-future)
                                         (ref-set writer-result (first @writer-future)))
                                 ))]
          (listener))) ;;[illegible] reader-result & writer-result [illegible]

      (stop [this] (reset! reader-config (assoc @reader-config :force-stop true)))
      (finished? [this] (:finished @writer-result))
      (success? [this] (and (:success @reader-result) (:success @writer-result)))
      (monitor [this] {:job-id (:id @reader-config) :job-start-time start-time :static-time @last-static-time :total-bytes @total-bytes :avg-read-rate @avg-read-rate :interval-read-rate @interval-read-rate})
      ;; when call finish,please check success? is true
      (result [this] (if (success? this){:reader @reader-result :writer (merge @writer-result (monitor this))} nil))
      (info [this] sub-job-conf))))
```

Clojure in JD



magpie: realtime task scheduling system for realtime data warehouse