



腾讯社交广告  
Tencent Social Ads

The Power to  
Connect Businesses and People  
赋能商业 | 始终于人

# 腾讯广告数据系统

李锐





## 关于我

11年加入腾讯

关注分布式存储和计算系统

现负责腾讯广告数据系统



# 腾讯广告简介



国内 **流量最大** (日均约500亿PV, 百亿展现)

**场景最丰富** (微信, QQ, 视频, 新闻等几十种产品)

**覆盖人群最广泛** (超过10亿受众)

的互联网广告平台

微信广告

广告, 也可以是生活的一部分

QQ广告

最懂年轻人的营销平台

腾讯视频广告

国内移动视频第一入口

腾讯新闻、快报广告

国内移动咨询top入口

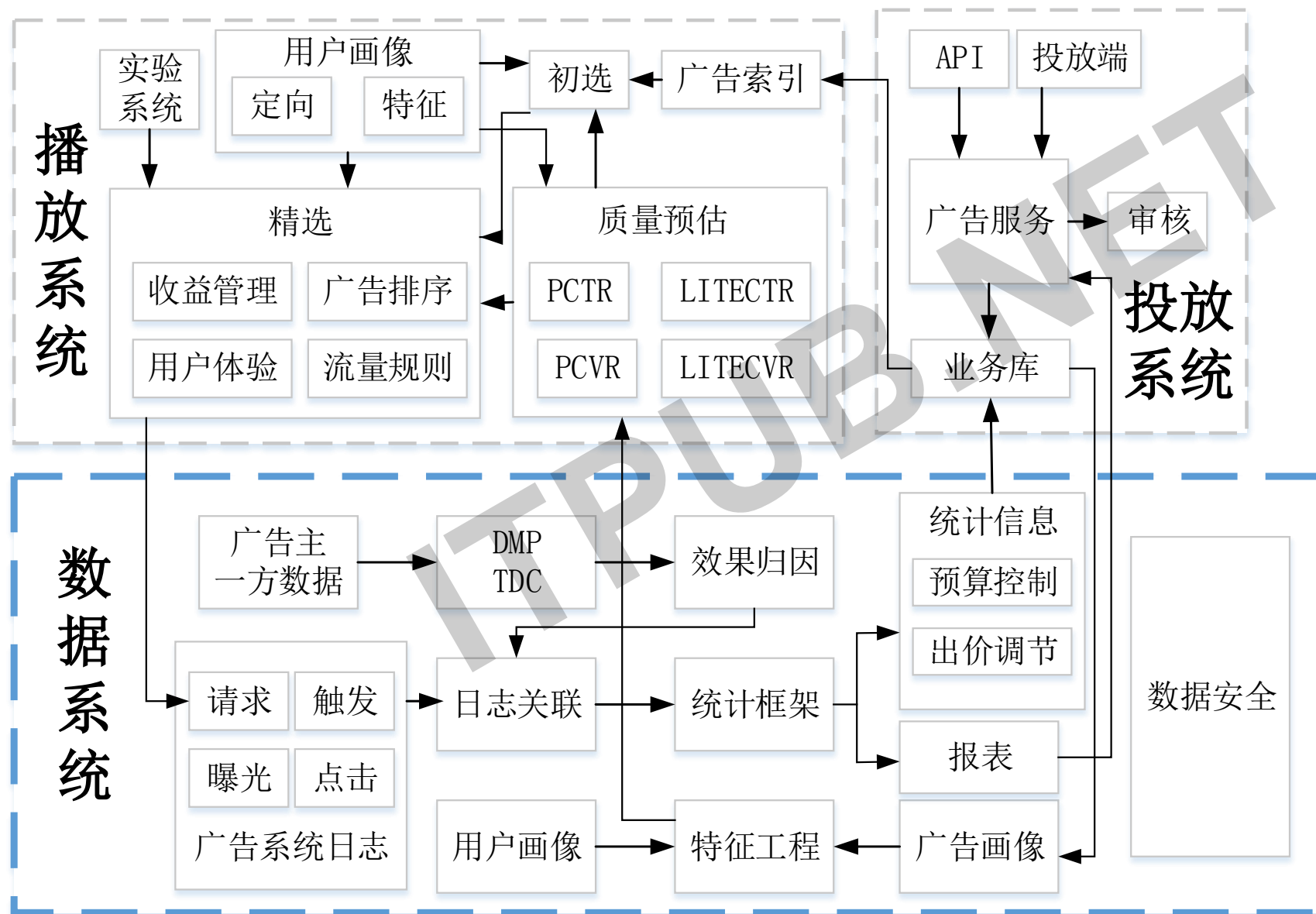
联盟广告

海量优质APP  
月活跃用户5亿

应用宝广告

用户覆盖率第一的安卓应用商店

# 广告系统架构



数据带来价值:

- 模型训练
- 策略机制
- 报表和BI分析
- DMP数据管理



1. 数据分析

2. 特征工程

3. 数据安全



1. 数据分析

2. 特征工程

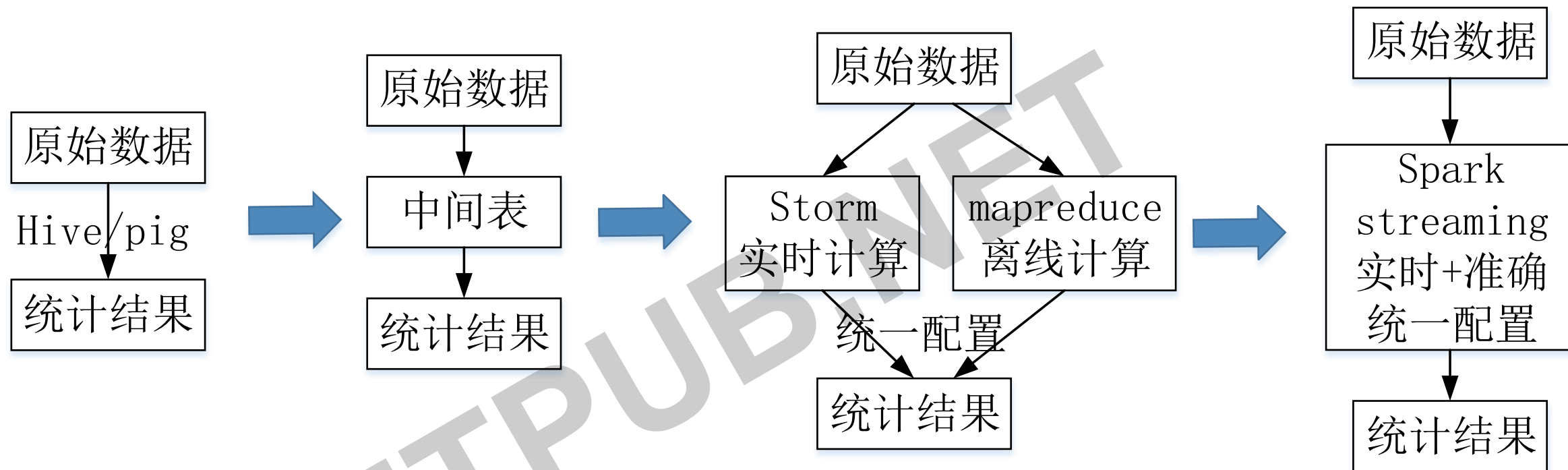
3. 数据安全

# 数据分析任务的分类



分类	预先聚合计算	在线实时计算 OLAP
优点	查询速度快	灵活性高
缺点	灵活性不够	查询时计算量大
应用场景	报表 Pushmail 统计特征 预算控制 出价调节	DMP BI多维透视分析

# SEAGULL - 基于spark的聚合计算平台(1)



## • 业务特点

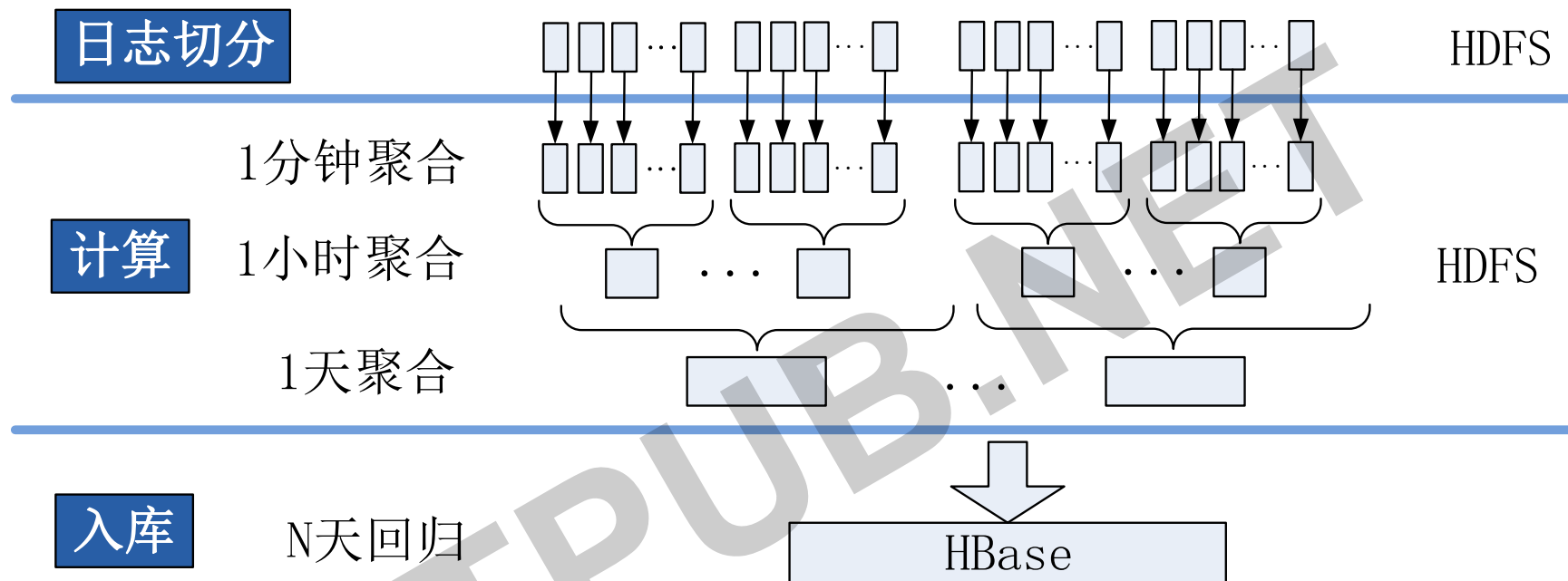
- 一份输入，多份输出
- 数据量大，每天100多T
- 实时性，准确性

## • 实现方案

- 统一数据流
- 配置化计算逻辑
- 一套代码解决按key的聚合问题



# SEAGULL - 基于spark的聚合计算平台 (2)

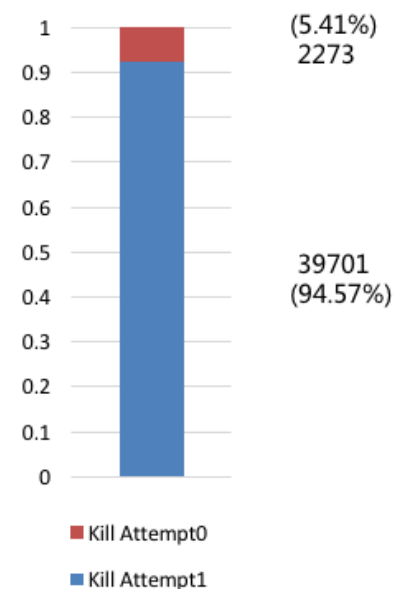
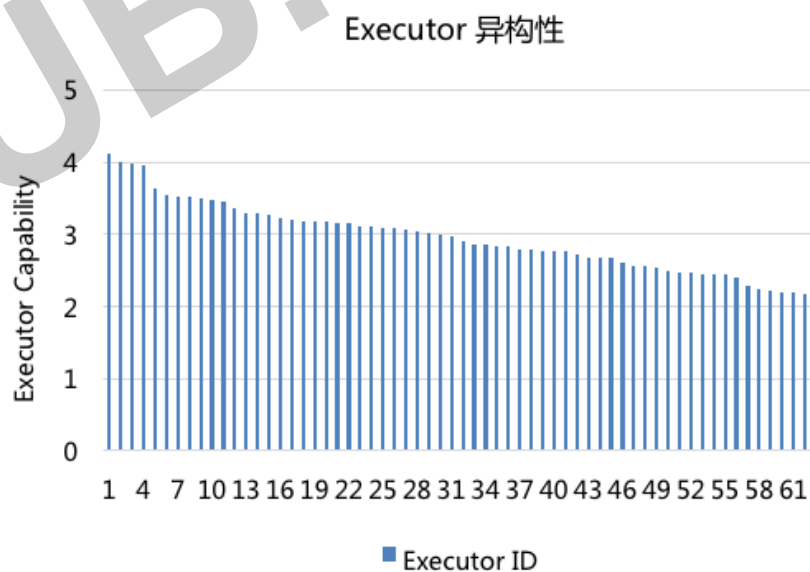
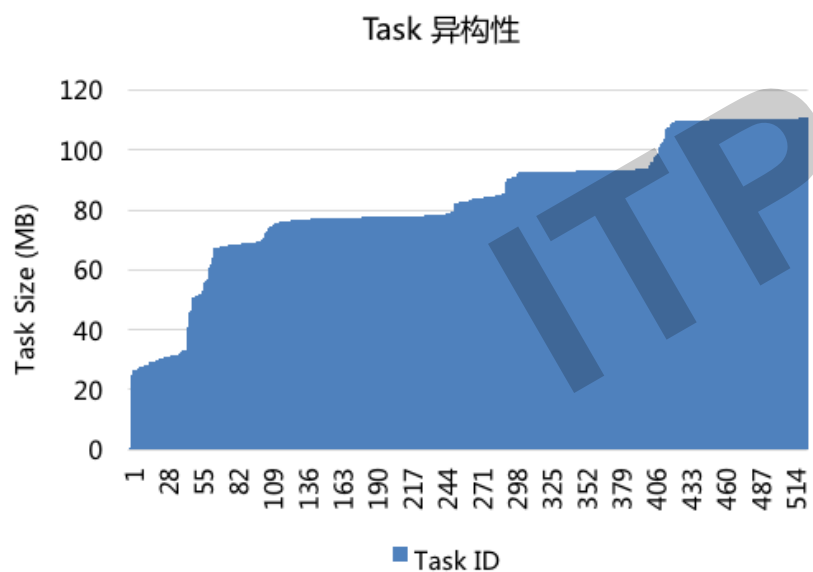


- 准确:
  - 输入输出都是hdfs
  - 目录结构做checkpoint
- 实时:
  - 在spark中执行分钟级别的mapreduce
  - 常驻进程，分层调度
- 优化:
  - spark内存做cache，命中率96%
  - 先计算diff再写hbase，写入量减少93%
  - 分钟级调度，时间维度上资源均匀消耗
  - 不采用spark window，避免输出任务batch之间依赖，在集群抖动时快速恢复
  - 利用hbase version，避免新数据被老数据覆盖

# 调度优化(1)



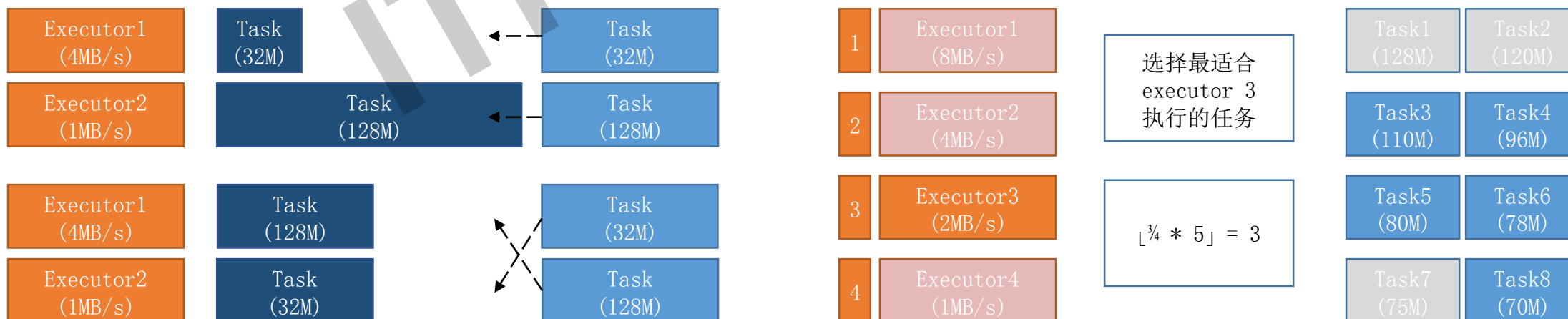
- 推测执行，避免struggle
- 会有很多的推测备份任务被杀掉



# 调度优化(2)



- 优化思路
  - 利用 streaming 周期任务的特点，动态评估每个 executor 的吞吐量
  - 能力强的节点分配大任务，能力弱的节点分配小任务
- 实现
  - a) RDD 及 NewHadoopRDD: 获取每个 task 对应的 HDFS 文件的大小并上报
  - b) 在每个 batch 结束时，分析 executor 处理任务的情况，动态更新到 driver 端的记录里
  - c) 按照 executor 的吞吐量排序在任务列表里选择合适的任务

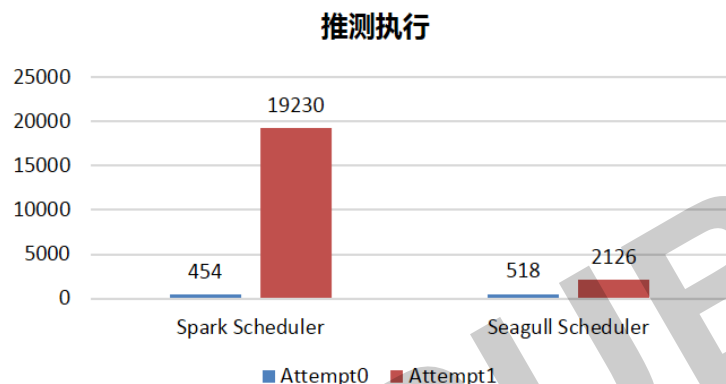


# 调度优化(3)

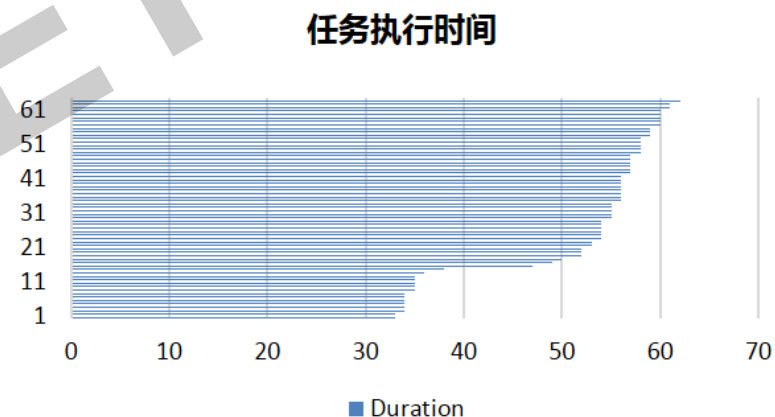


- 性能提升

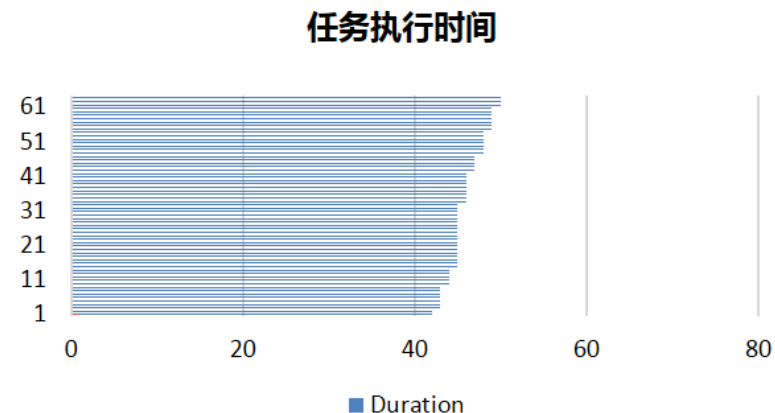
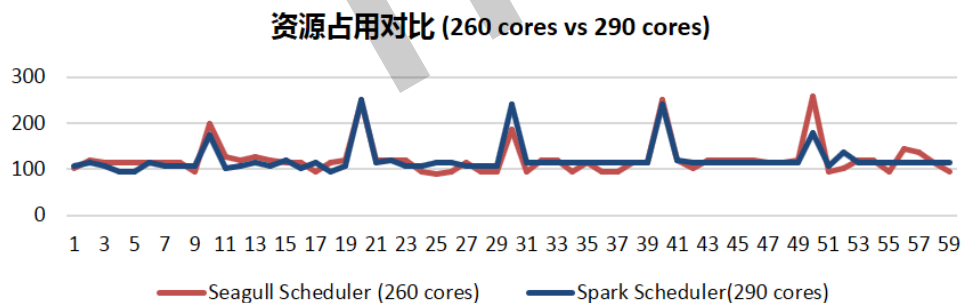
- a) 推测任务减少 86.57%



- b) 计算延迟降低 20.96%



- c) 少用 1/10 资源





预先聚合计算

在线实时计算

# 要解决的问题 - SQL举例



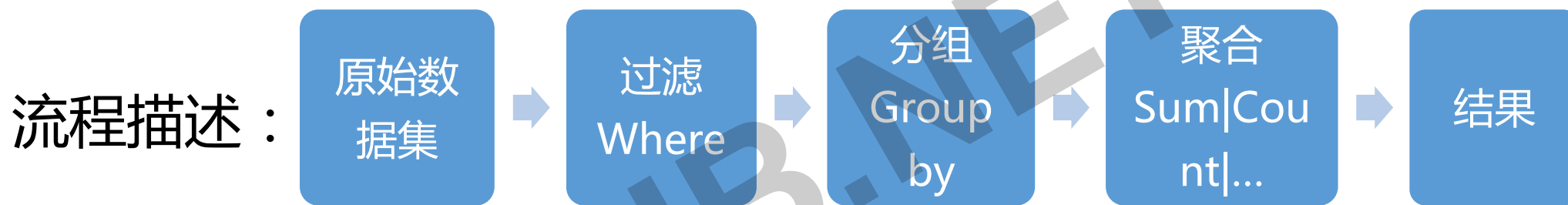
- 从万亿条数据中选择符合条件的数据，计算聚合结果

- 查询用户年龄性别的分布

```
SELECT age, gender, COUNT(*) FROM log WHERE advertiser_id=123 group by age, gender;
```

- 查询不同曝光次数的用户的占比、点击率、消耗等

```
SELECT exposure_num, COUNT(*) as user_num,  
       SUM(sum_click) / SUM(exposure_num) as click_rate, SUM(sum_cost) AS total_cost  
FROM  
  (SELECT user_id, COUNT(*) AS exposure_num,  
         SUM(click_count) AS sum_click, SUM(cost) AS sum_cost  
   FROM log  
   GROUP BY user_id) temp_table  
GROUP BY exposure_num;
```



- 从万亿条数据中选择符合条件的数据，计算聚合结果
- 为了提高查询速度，就需要预先将数据整理成适合查询的格式

# 两种数据模型



1. 平铺结构: 宽表, 其中每个cell可以是term list, 可以有正排, 倒排
2. 嵌套结构: proto buffer, 正排存储类似于dremel/parquet, 没法倒排

	Column 0	Column 1	Column 2	...
Doc 0	term list	term list	term list	term list
Doc 1	term list	term list	term list	term list
...	term list	term list	term list	term list

```
Message pageview {  
  optional UserProfile user;  
  repeated Position pos {  
    repeated Impression imps {  
      optional Advertisement ad;  
      repeated Click clicks {  
        optional BillInfo bill;  
      }  
    }  
  }  
}
```



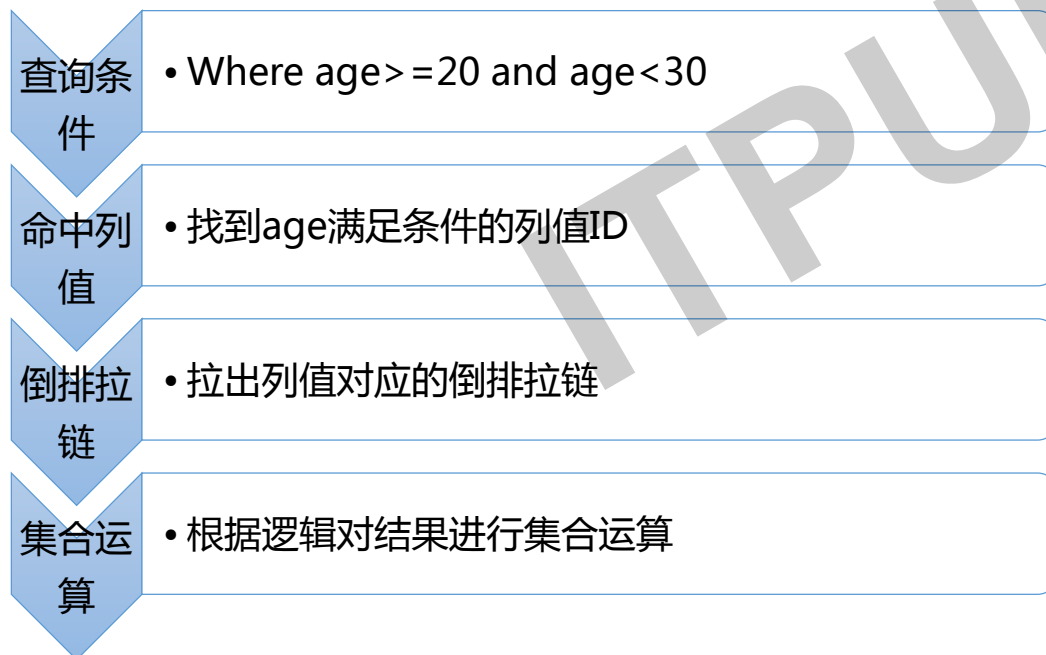
# 索引文件设计(1)



索引数据结构：

全局信息	列值字典	倒排数据	正排数据
	每一列(Column)的值 编码为列值(Term) ID	列值ID对应的文档ID 列表	列式存储的压缩数据

检索查询流程：



```
SELECT gender, COUNT(*)
FROM log
WHERE age >= 20 and age < 30
GROUP BY gender;
```

# 索引文件设计(2)



## • 倒排

- 文档ID在文件内编码，减少值域范围
- 稀疏数据保存array，稠密数据保存bitmap
- 采用roaring bitmap进行自适应
- 实现高效的bitmap与array之间交并差的计算

## • 正排

- 减少磁盘IO访问-列存储
- 最大程度节省磁盘空间-编码压缩
- 通过采样进行比较，自动确定最优编码方案

对于不同的数据特征，适用于不同的编码算法

定长类型编码：定长数组编码、列值个数索引编码、定长列表编码、变长列表编码、run length encoding

String类型编码：单string长度索引编码、多string长度索引编码、单string列表编码、多string列表编码

简单字典编码：适用列值重复较多的string类型

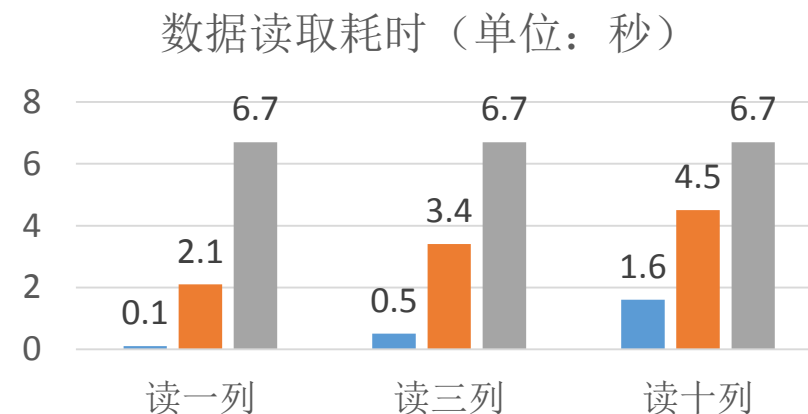
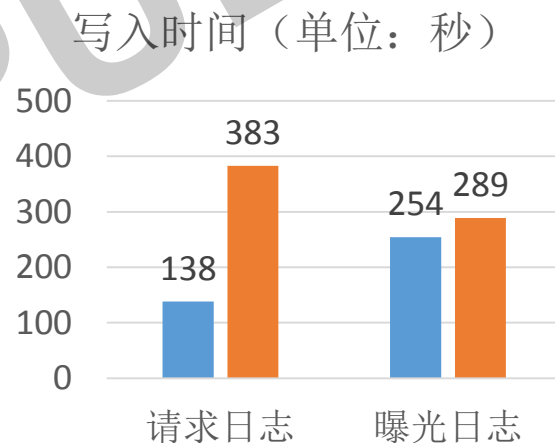
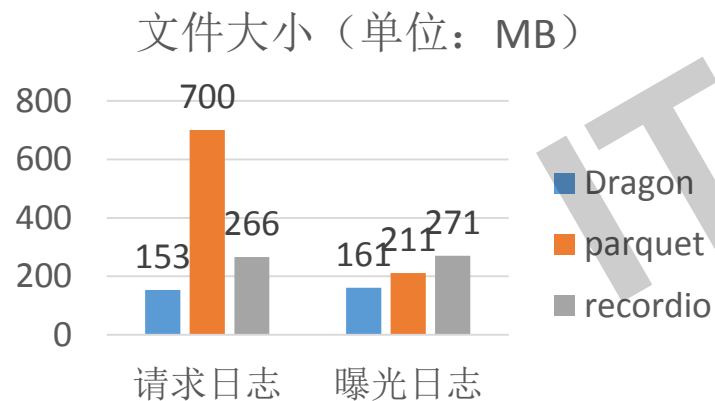
Huffman字典编码：列值分布不均匀情况下对简单字典编码的优化

二进制String编码：较特殊的二进制数据类型

# 嵌套结构列式存储格式dragon(1)



- Pivot需要支持直接查原始日志proto buffer格式
- 嵌套结构，无法建倒排
- 针对稀疏的proto buffer数据进行优化
- 谷歌 dremel vs 开源parquet vs 自研 dragon

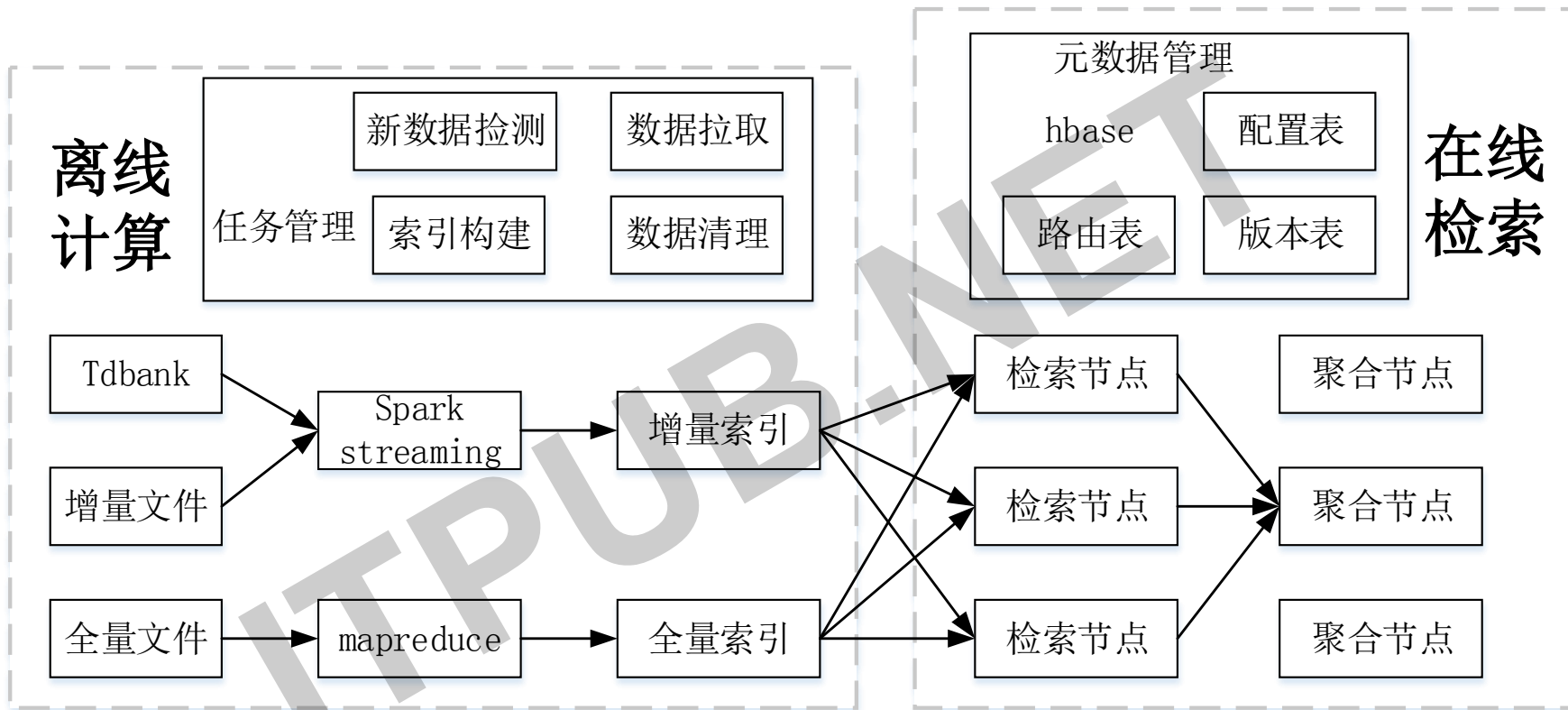


# 嵌套结构列式存储格式dragon (2)



- 存储：同列数据连续存储，压缩效率高
- 存储：不保存空节点的rlevel和dlevel
- 写：暂时为空的节点cache rlevel dlevel
- 写：Flush cache的时候只flush到下层节点
- 读：读取数据只需读取需要的列，节省磁盘IO
- 读：热点hash map优化为查表
- 读：重复计算前置
- 读：自适应分级缓存，栈上小内存分配

# PIVOT-实时多维透视分析引擎



## • 业务特点

- 万亿条数据
- 秒级响应

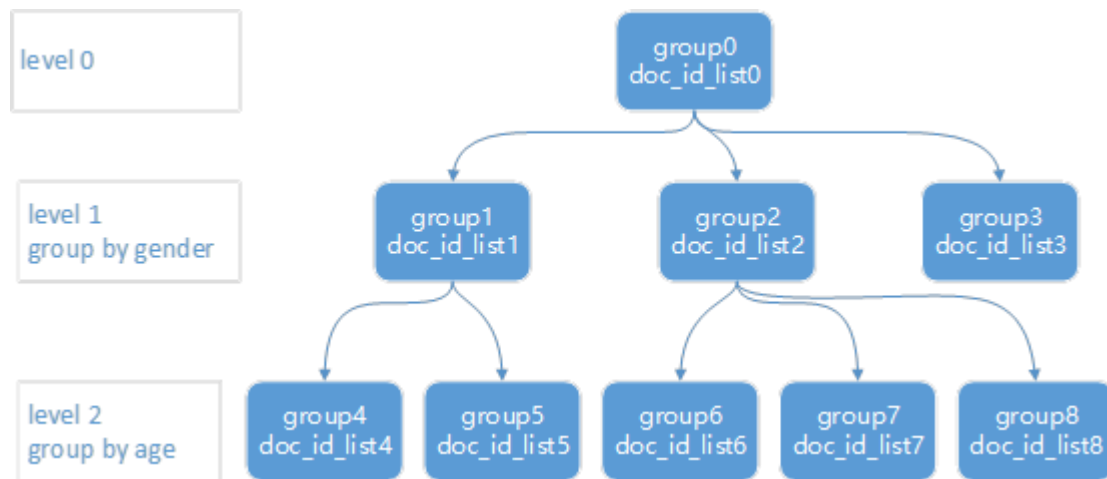
## • 建索引

- 列存储
- 利用SSD进行检索，而不是内存
- 利用HADOOP集群构建索引

# 遍历正排数据实现分组



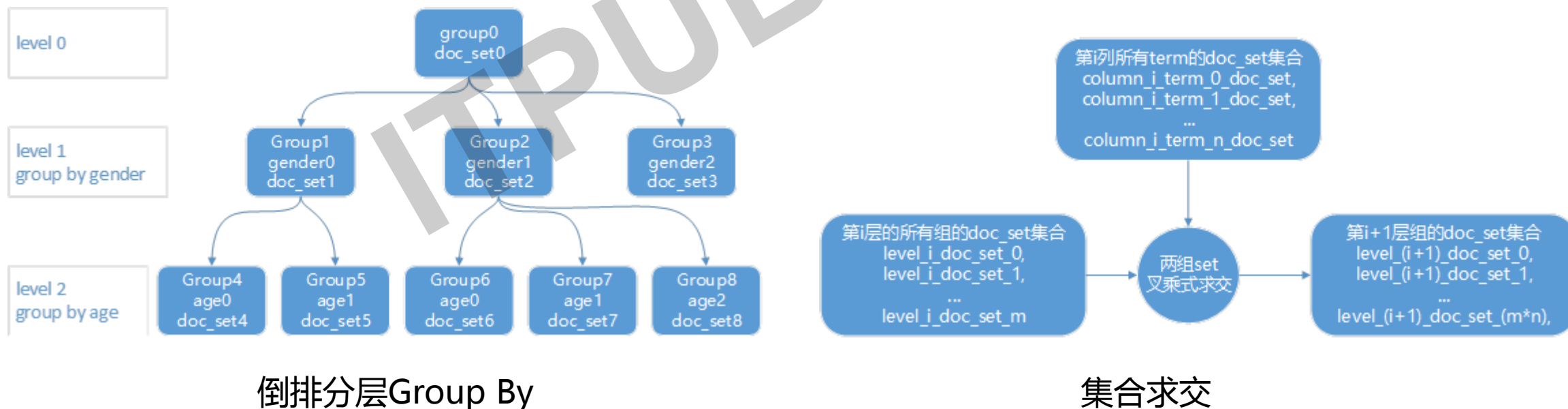
- Group by age, gender, city
- 如果直接排序：
  - 每次比较age\_gender\_city，而前缀基本一致，浪费计算量
  - Map太大，log(N)的插入操作也很耗时
- 按照列的顺序分层计算
  - 先group by age，然后在每个age里面group by gender，以此类推
  - 每次查找和插入都只操作一个很小的map





# 基于倒排数据实现分组

- 基于正排数据，计算量跟命中的doc数量成正比
- 基于倒排的roaring bitmap，进行集合求交计算量有上限，在命中doc很多时会更快
- 仅适用于group分组数量比较小的情况，否则需要求交的次数太多





1. 数据分析

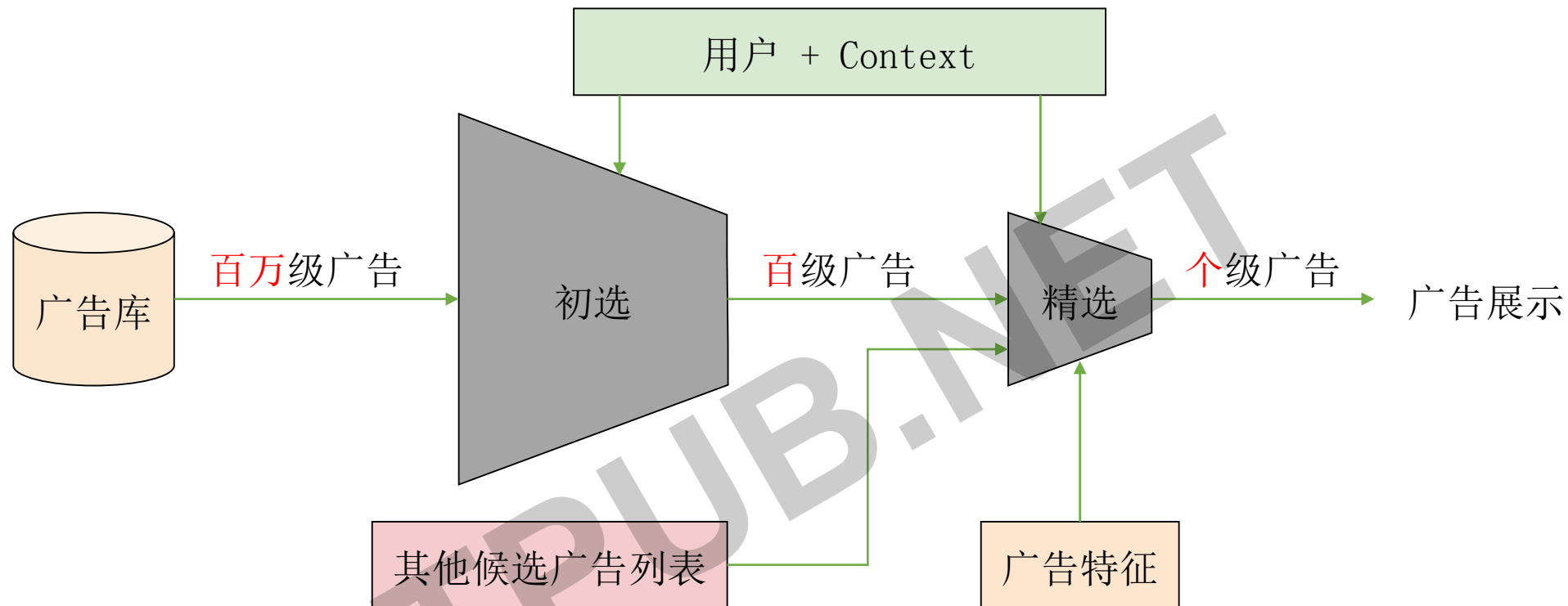
2. 特征工程

3. 数据安全





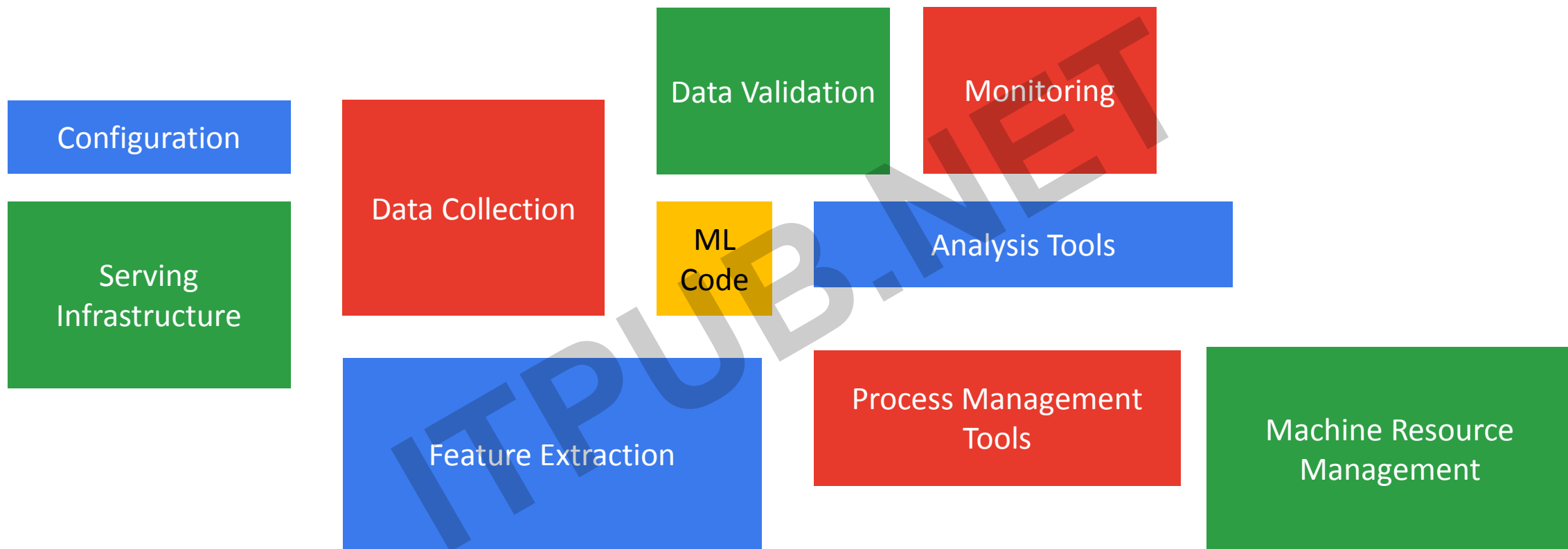
# 广告业务多模型优化



初选模型：相似人群扩展，自动定向，动态创意，商品推荐，litectr，litecvr，...

精选模型：pctr，pcvr，负反馈，...

# 模型优化不仅仅只是模型训练



# 需求1: 从业务角度如何提升特征工作迭代效率？

## ❖ 模型方特征工作迭代内容

- 已经有哪些特征？(有哪些用户特征、统计特征、广告特征等等)
- 特征数据质量如何？(数据分布是否符合预期？是否存在脏数据？)
- 有没有其他模型尝试过？(离线调研和在线调研效果如何)
- 快速地尝试新的特征（走通特征生产、调研和在线应用流程）

## ❖ 模型方如何看待特征工作？

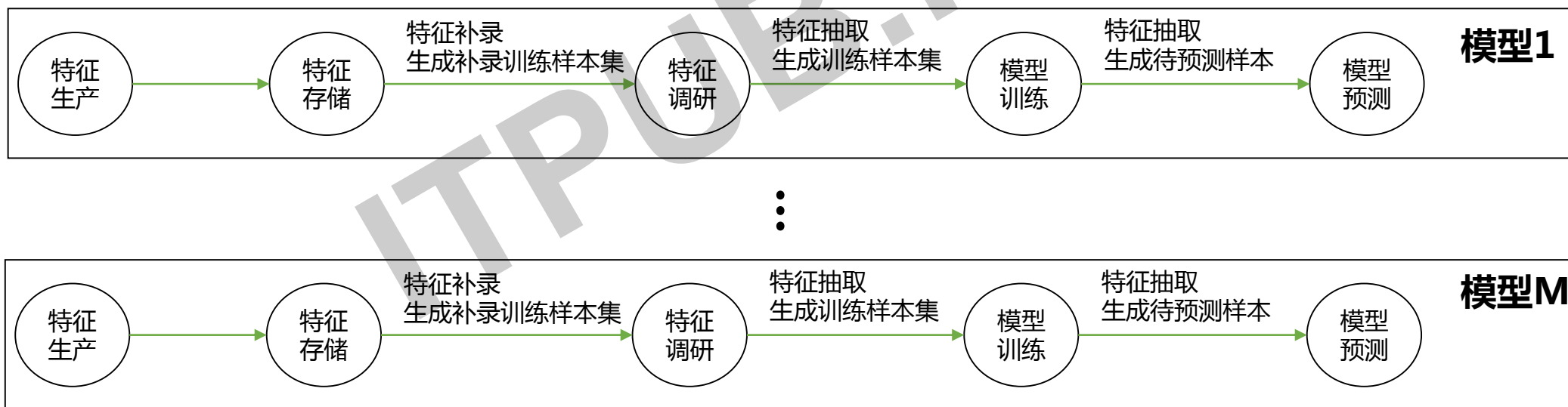
- 只关注如何使用特征，即简单配置要什么特征就可以完成特征调研、上线等工作
- 不关注特征数据如何流通、存储、访问计算
- 计算逻辑（如特征Join）能否尽量少写代码，简单配置复用公共组件



# 需求2: 从系统角度如何提高特征工作资源利用率？

## ❖ 多模型优化在特征工作上高度重叠

- ❑ 都需要进行特征生产、存储、特征补录并进行离线调研、离线特征抽取生成训练样本集、在线特征抽取生成待预测样本
- ❑ 存储和计算资源上能否共享使用并优化，提升资源利用率？



# 需求3： 如何避免训练和预测的偏差

## ❖ 训练和预测的偏差

Ø 指的是训练阶段和预测阶段的效果差异（如线下AUC和在线AUC相差很大）

## ❖ 引起训练和预测的偏差的可能原因

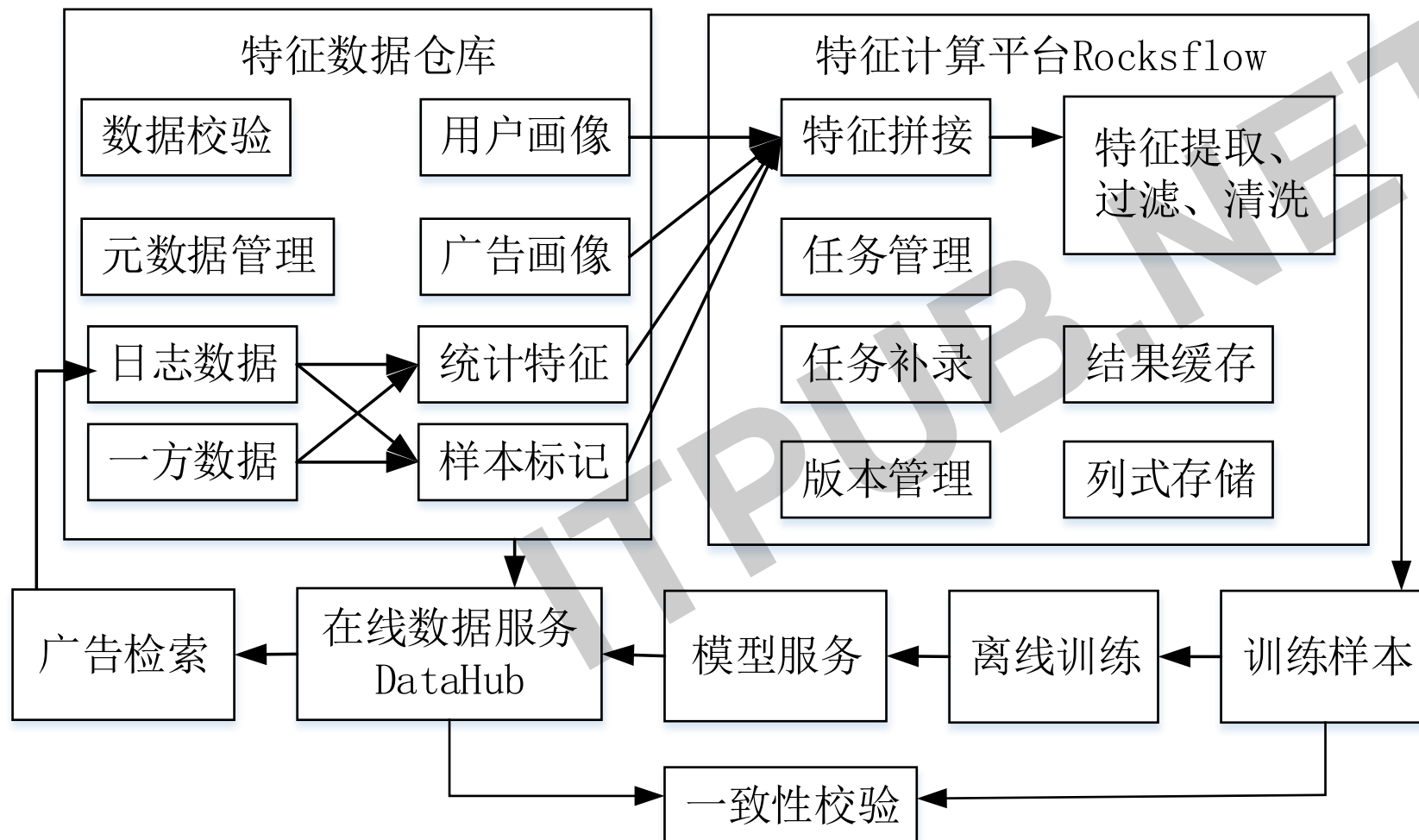
Ø 训练和预测数据处理逻辑（如特征抽取等）是否存在不同

Ø 由于训练和预测时间存在差异，数据上是否有变化

# 特征工程平台



1. 数据分析
2. 特征工程
3. 数据安全



- 模型多
- 数据量大
- 节省机器成本
- 提升特征调研效率
- 共享特征数据
- 避免训练和预测的偏差

# 前文统计特征生产框架-CF2

## ❖ CF2 – Context Feature Computation Framework

Ø 支持按照配置的<Interval, Window>进行统计的通用系统框架

➤ 使用者只需要配置如下参数即可完成统计

1. 源数据输入，如HDFS路径
2. Interval和Window配置信息
3. 聚合逻辑，类似于Spark UDAF函数
4. 统计数据输出，如HDFS路径

### 特征举例 (用户为主体)

每小时统计最近3小时、每天统计最近14天的曝光量、点击量

每天统计用户在App激活场景下最近14天的点击量、转化量

每天统计用户在H5下单场景下最近14天的点击量、转化量

每小时统计用户最近7天的历史转化广告ID列表、商品ID列表

### 特征举例( 非用户为主体)

每天统计每个商品ID最近7天或者14天的曝光数和点击数

每小时统计每个广告ID最近14天的曝光量、点击量

# CF2存储HBase 表结构设计

interval_table_1_MIN																
rowkey	cf:d								cf:f							
	T <sub>1m</sub>	...	T <sub>30m</sub>	...	T <sub>60m</sub>	...	W <sub>30m</sub>	W <sub>60m</sub>	T <sub>1m</sub>	...	T <sub>30m</sub>	...	T <sub>60m</sub>	...	W <sub>30m</sub>	W <sub>60m</sub>
k1																

interval_table_1_HOUR																
rowkey	cf:d								cf:f							
	T <sub>1h</sub>	T <sub>2h</sub>	...	T <sub>24h</sub>	...	W <sub>6h</sub>	W <sub>24h</sub>	W <sub>72h</sub>	T <sub>1h</sub>	T <sub>2h</sub>	...	T <sub>24h</sub>	...	W <sub>6h</sub>	W <sub>24h</sub>	W <sub>72h</sub>
k1																

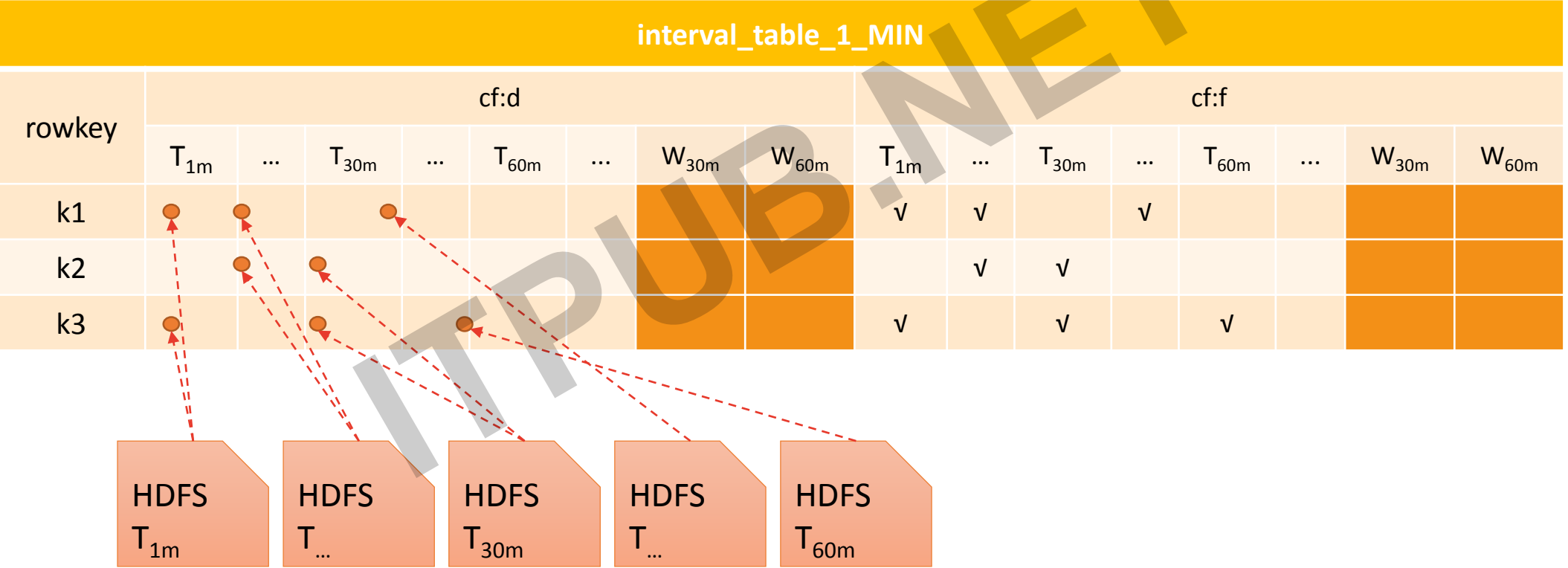
interval_table_1_DAY										
rowkey	cf:d					cf:f				
	T <sub>1d</sub>	T <sub>2d</sub>	...	...	W <sub>14d</sub>	T <sub>1d</sub>	T <sub>2d</sub>	...	...	W <sub>14d</sub>
k1										



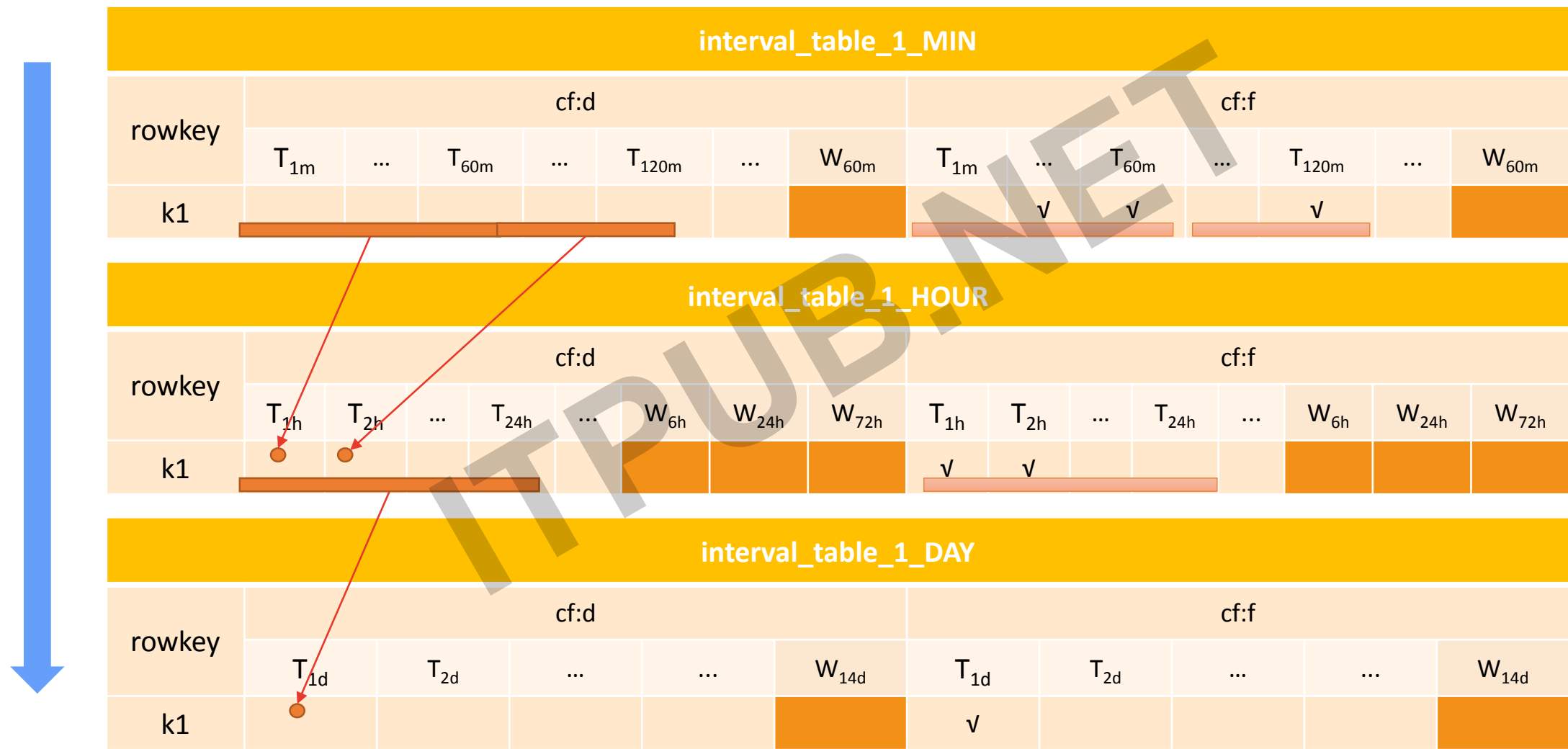
# CF2计算之读取输入数据

## ❖ 读取输入数据

Ø 将数据按最细时间粒度写入对应的Interval Table



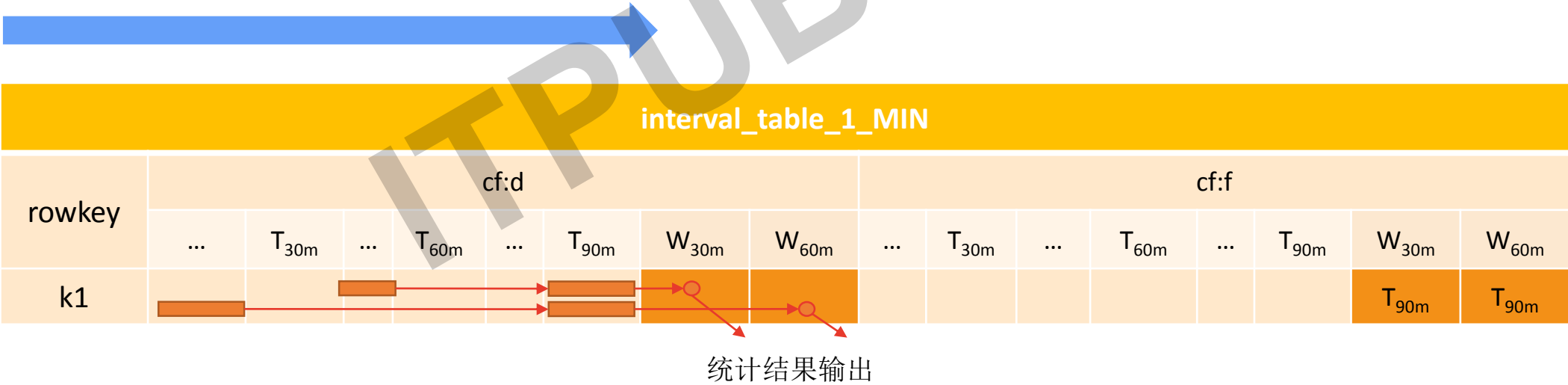
# CF2计算之Interval统计聚合



# CF2计算之Window统计聚合

❖ 滑动窗口进行计算

$$W_t = W_{t-1} + T_t - T_{t-w}$$



# 广告特征生产框架-TerraStore

## ❖ 广告、创意素材

- Ø 广告数据分层级，Advertiser -> Campaign -> AdGroup -> Ad -> AdCreative
- Ø AdGroup层级里存储有扣费类型、计费类型、优化目标类型、定向等数据
- Ø AdCreative层级有广告标题、描述、落地页、图片素材等数据
- Ø 特征应用: 从广告标题描述提取分类信息、爬取落地页提取核心关键词、从图片素材上提取美感特征等等

## ❖ 广告特征生产目标

- 存储所有广告（包括当前在线和非在线的广告）的特征数据
- 广告、创意素材变更，快速地在模型上生效
- Ø 记录版本变更，避免Training-Serving Skew问题

# TerraStore系统实现

## ❖ 实现原理

- 使用HBase Observer Coprocessor和MVCC生成流水表，确保数据和流水数据一致性
- Manager模块读取各个HBase Region里的流水数据
- Manager将变更前后的数据有选择地发送给特征逻辑处理方
- 特征生产完写回多版本表，并通过多版本流水数据实时发布上线

b创意URL变更

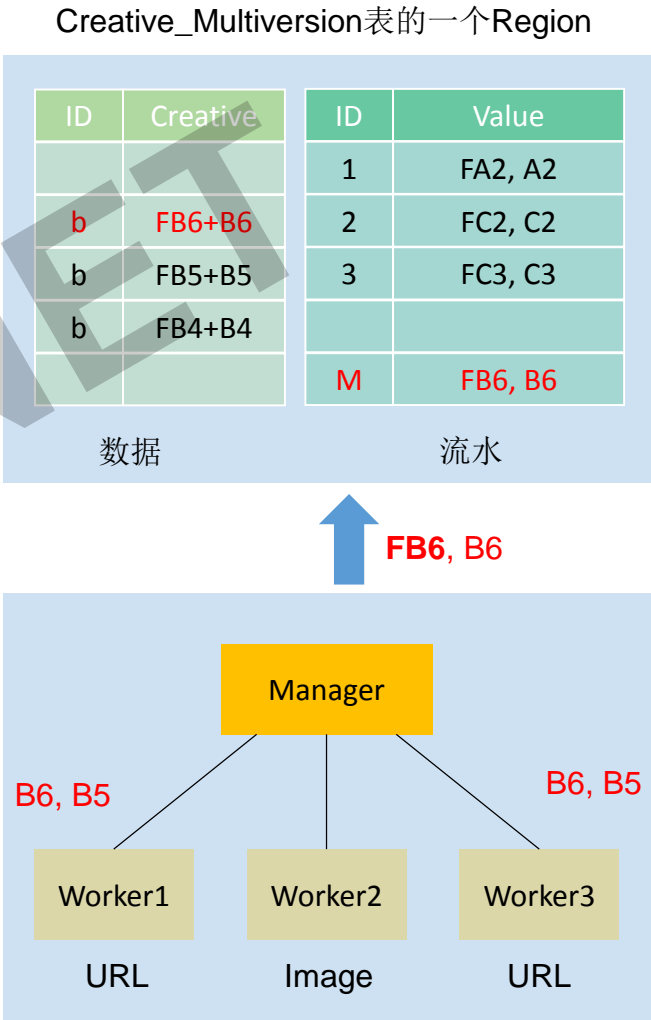
b, B6

创意变更数据流

数据		流水	
ID	Creative	ID	Value
a	A8	1	A2, A1
b	B6	2	C2, C1
c	C5	3	C3, C2
		K	B6, B5

Creative表的一个Region

B6, B5



# 特征仓库存储系统

表T版本V8的数据

	F1	F2	F3	...	FM
U1	V8	V8	V8	...	V8
U2	V8	V8	V8	...	V8
...	...	...	...	...	...
UK	V8	V8	V8	...	V8

HBase存储，多版本(N个)  
用在实时的训练样本集生成

	F1	F2	F3	...	FM		
U1	V6	V6	V6	...	V6		
U2	V6	F1	F2	F3	...	FM	
...	U1	V7	V7	V7	...	V7	
UK	U2	V7	F1	F2	F3	...	FM
	...	U1	V8	V8	V8	...	V8
	UK	U2	V8	V8	V8	...	V8
		...	...	...	...	...	...
		UK	V8	V8	V8	...	V8

在线kv存储，最新版本  
用在在线预测

	F1	F2	F3	...	FM
U1	V8	V8	V8	...	V8
U2	V8	V8	V8	...	V8
...	...	...	...	...	...
UK	V8	V8	V8	...	V8

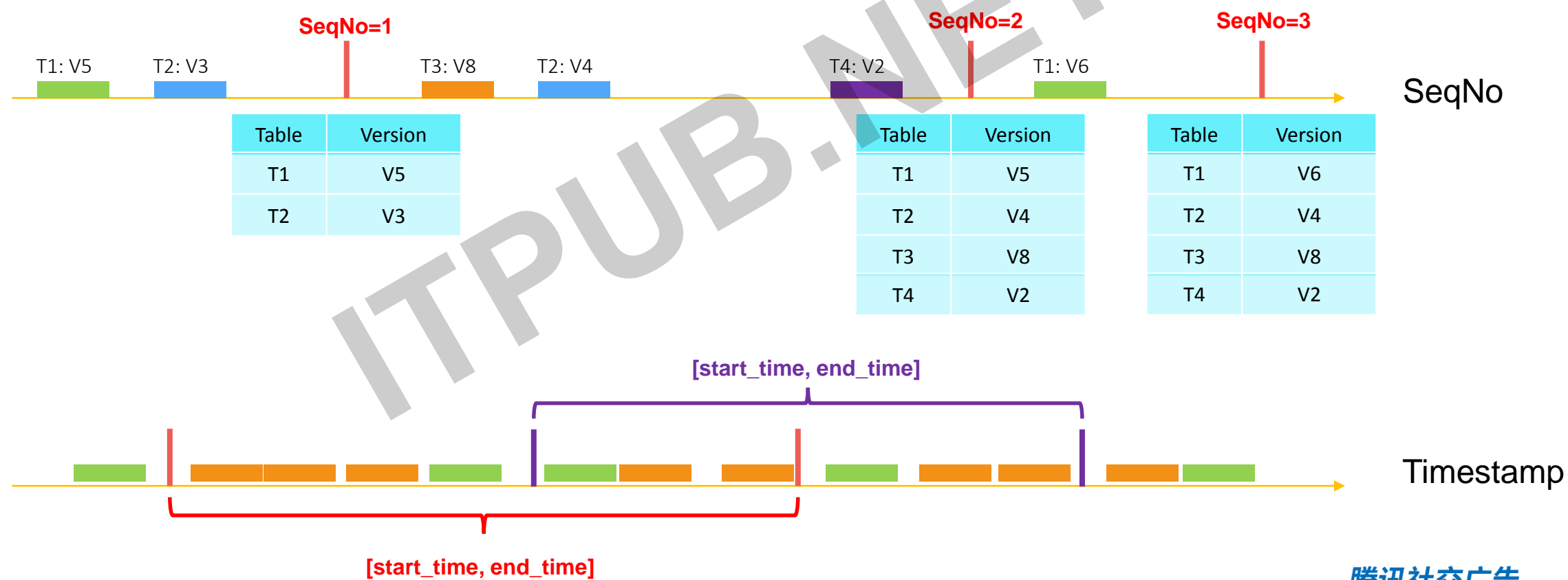
Dragon列式存储，多版本  
用在特征补录和数据质量验证分析

	F1	F2	F3	...	FM	
U1	V1	V1	V1	...	V1	版本V1
...	...	...	...	...	...	
U1	V2	V2	V2	...	V2	
...	...	...	...	...	...	版本V2
U1	V8	V8	V8	...	V8	
...	...	...	...	...	...	
UK	V8	V8	V8	...	V8	版本V8
...	...	...	...	...	...	
UK	V8	V8	V8	...	V8	

# 特征仓库存储系统之多版本说明

❖ 解决特征数据Serving阶段和Training阶段的一致率，避免Training-Serving Skew

- 表记录批量更新，Serving阶段记录SeqNo，Training阶段检索对应版本
- 时间序列数据更新，Serving阶段记录[start\_time, end\_time]，Training阶段检索指定时间区间的数据



# 特征计算之调研环境下特征补录

## ❖ 特征补录

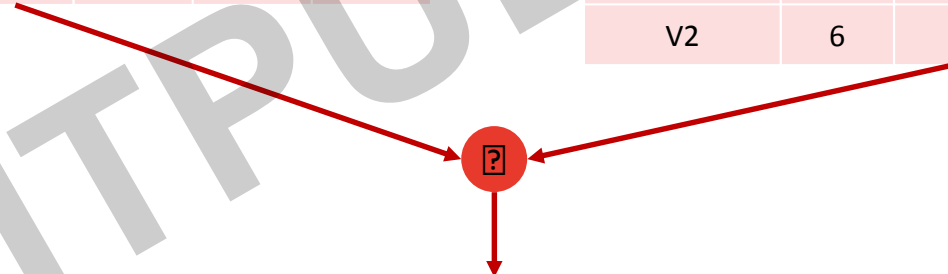
- Ø 新加了一个特征集，验证下模型加入这个新特征的效果
- Ø 每个生产环境下跑的模型都会存储之前的训练样本集
- Ø 将之前的训练样本集+新的特征集，再生成新的训练样本集。需要保证不要出现时间泄漏问题

timestamp	ID	F1	F2	...	Fm
V1	2				
V1	5				
V2	5				

训练样本集

timestamp	ID	Fr	Fs	...	Fz
V1	2				
V1	8				
V2	5				
V2	6				

待补录特征集



timestamp	ID	F1	F2	...	Fm	Fr	...	Fz
V1	2							
V1	5							
V2	5							

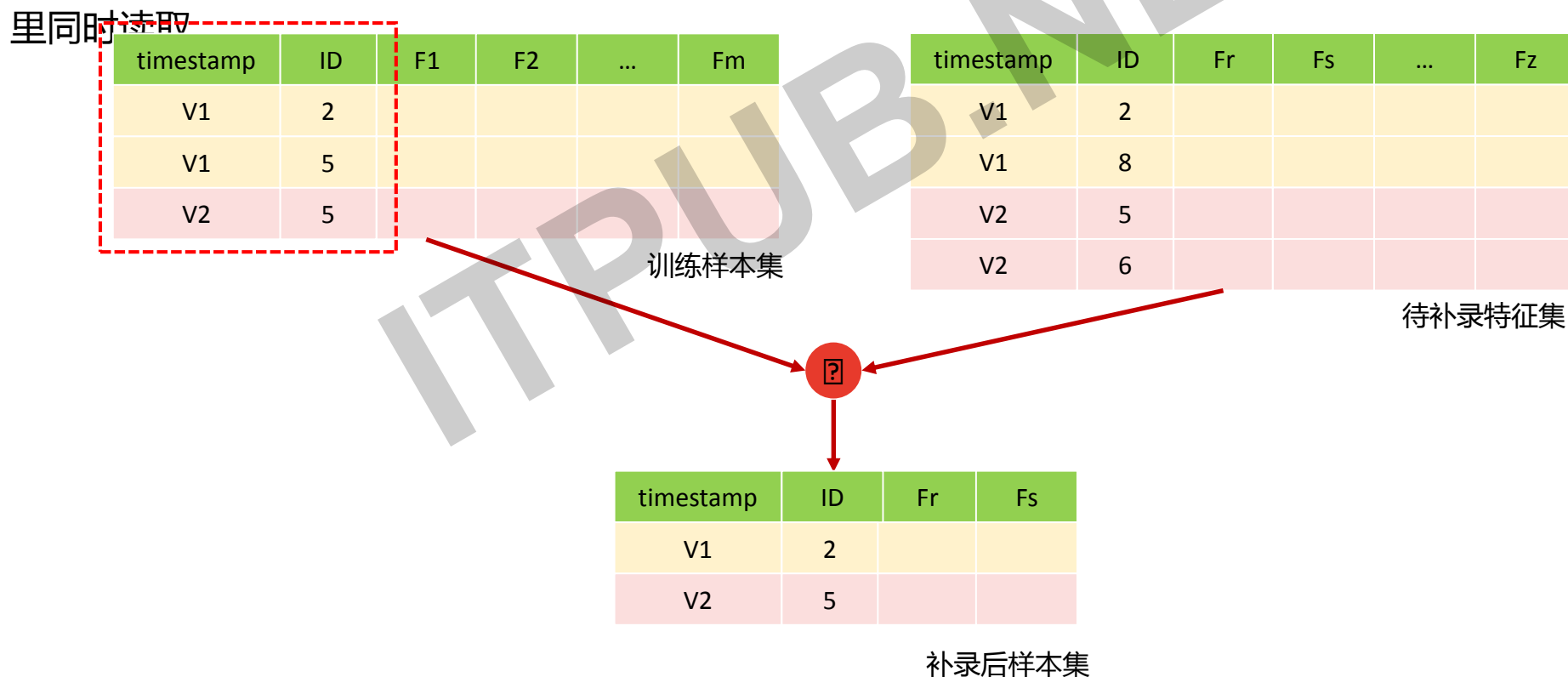
补录后样本集



# 特征补录优化

## ❖ 主要优化措施

- ❑ 训练数据集和特征数据集，都使用列式存储，减少读取数据流(如待补录特征集里只使用Fr和Fs两列)
- ❑ 训练数据集只需要读取<timestamp, id>两列，进行broadcast join提升补录效率
- ❑ 将待补录的样本集单独生成列式文件，之前的训练样本集不变。读取的时候使用MultiInputFormat从两个列式文件里同时读取





1. 数据分析

2. 特征工程

3. 数据安全



# 数据安全治理



## 系统提供的 安全机制

安全传输协议

脱敏发布

隐私集合求交

数据水印

认证

授权

鉴权

审计

差分隐私

## 系统自身安全

权限最小化原则

敏感数据生命周期管理

数据分级管理

数据脱敏

加密存储

密钥管理服务

机器访问控制

敏感端口访问控制

内部认证管理

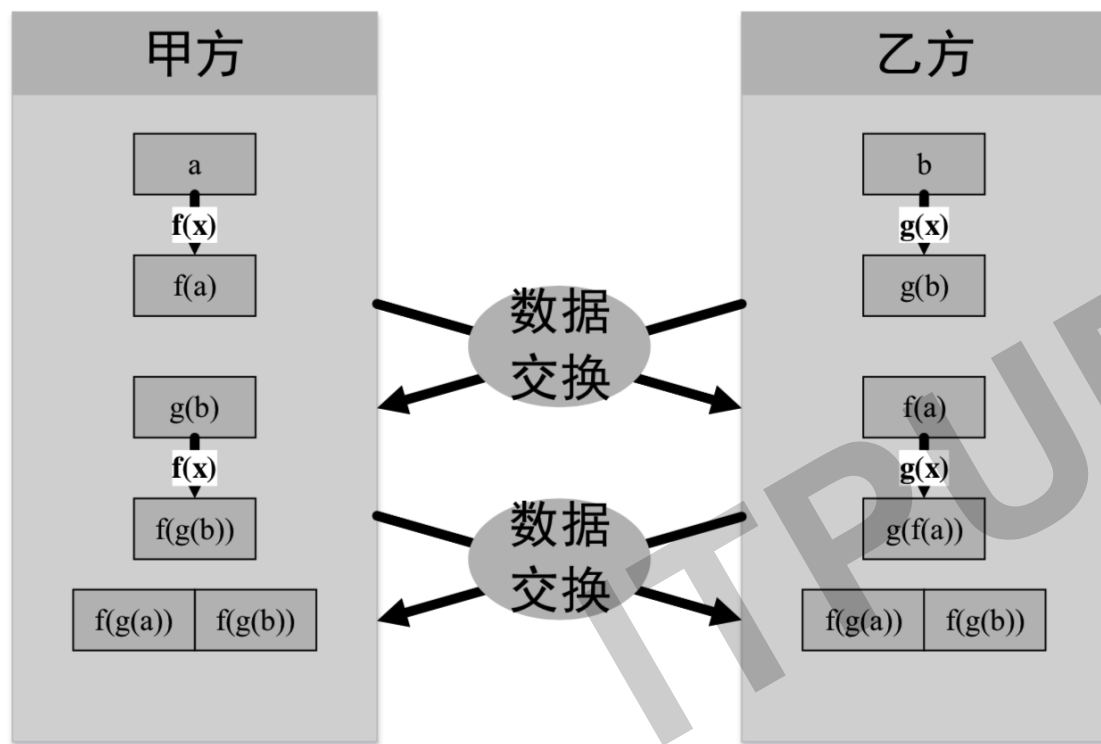
备份与恢复

内网隔离策略

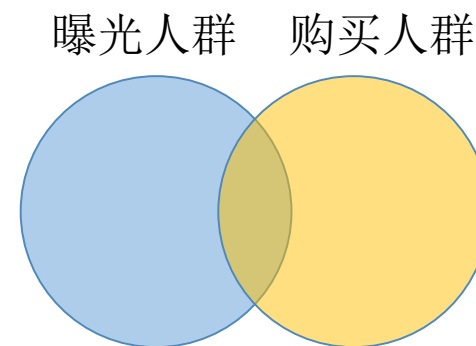
IP白名单

审计与合规

# 隐私集合求交



- 问题：在双方数据保密的情况下计算部分结果
- 解决方案：利用两个可交换的加密函数  $f(g(x))=g(f(x))$
- 应用场景：
  - 曝光归因
  - 第三方监测
  - 广告效果衡量



# 差分隐私



- 提供聚合查询结果时，客户可能通过构造相差很小的数据集，通过差分的方式获取单条数据
- 给结果附加少量扰动，防止差分攻击
- 限制访问的数据集，限制访问频率，控制差分预算



1. 数据分析

2. 特征工程

3. 数据安全



腾讯社交广告  
Tencent Social Ads

The Power to  
Connect Businesses and People  
赋能商业 | 始终于人

# 谢谢！