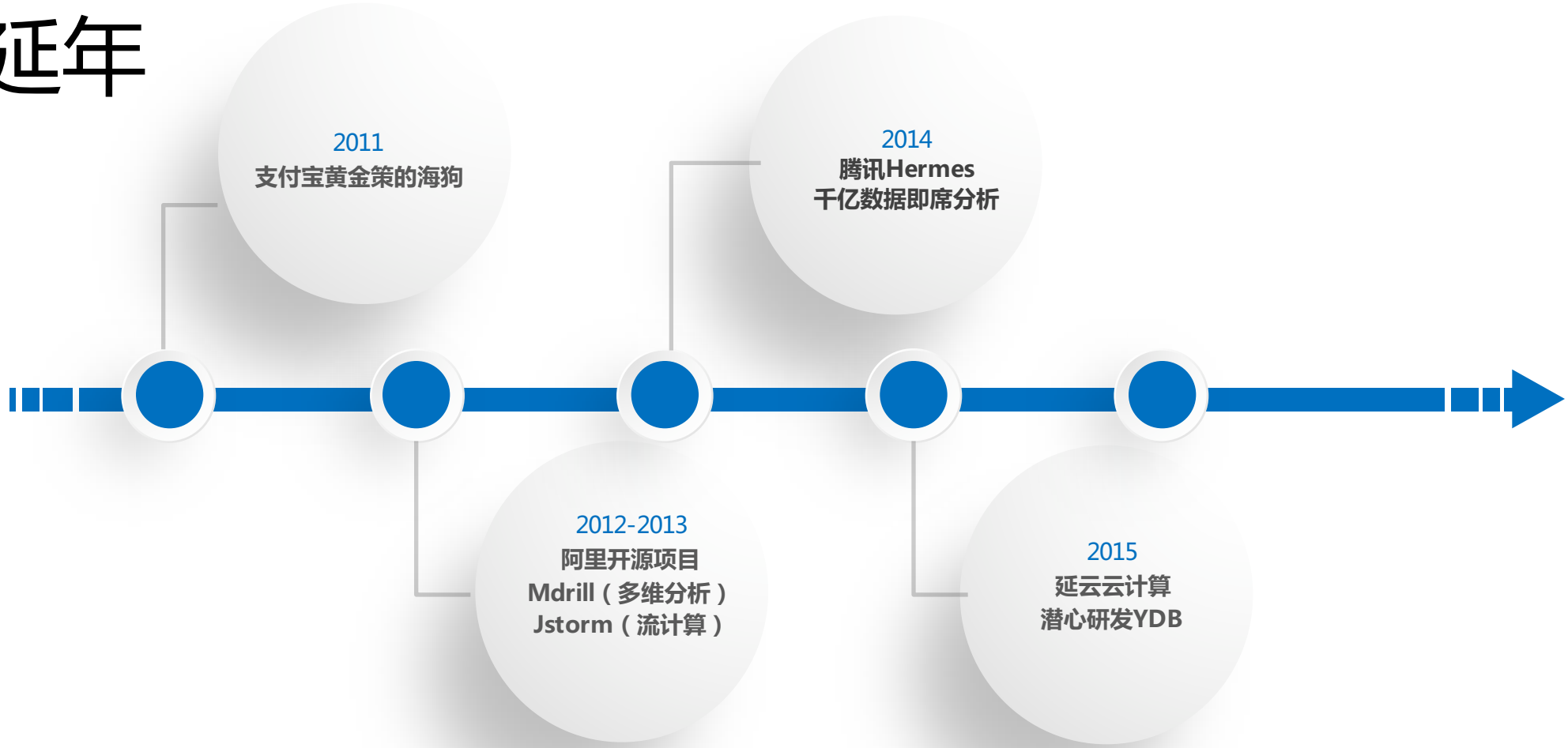
延云云计算

千亿数据
探索式即席分析

核心技术原理与实现详解

母延年


2011
支付宝黄金策的海狗
2012-2013
阿里开源项目
Mdrill（多维分析）
Jstorm（流计算）
2014
腾讯Hermes
千亿数据即席分析
2015
延云云计算
潜心研发YDB

目录
Contents

1 探索性即席分析简述

探索性

不管什么先查一下，看到数据后，再激发下一步的想法，直到分析出问题所在。

即席

当场，立即查询。
能很快的响应，不能等待太久。

目录
Contents

2 哪些行业需要探索性即席分析

公安

技侦：通话记录分析，同行同住，尾随人识别，连环案件等

网监：海量信息搜索，关键词统计，相似度匹配等

交通

机动车缉查布控：车牌模糊查询，同行车辆，昼伏夜出，陌生车辆等

交通运输：车辆行驶轨迹，特种车辆监控，道路养护，流量监控等

电信

通话数据统计与分析：通话质量，套餐推荐，用户习惯分析等

通信设备保养与维护：故障定位，故障预警，负载评估等

探索性即席分析

金融

流水日志分析：日志快速定位，明细查询，问题追溯，投诉处理等

行情监测分析：指标监控，多维分析，监管合规等

电商

用户画像，趋势分析，精准营销，推荐系统等

日志定位监测，用户行为分析，探索性数据分析等

物流

订单轨迹，订单状态，物流车辆状态，服务质量评估等

第二章
面临的挑战以及现有技术存在的问题

1. 探索性即席分析带来的挑战！
2. 探索性即席分析的技术特性！
3. 业界现有主流方案存在的问题！

1 探索性即席分析带来的挑战！

IOT-物联网，传感器，摄像头
视频识别
行车：位置，速度，车牌，颜色，车内人数
行人：性别，衣服颜色、人群密度，行走速度

UGC:用户产生内容

上网发帖，发博客
语音通话，照片分享

Transactions：事物日志

刷卡消费，买火车票
酒店入住，网吧上网
手机信令位置定位

技术挑战

超大规模：
数据总量达到数十亿到数万亿条
日均产生数千万到千亿条数据

超多维度：
字段数达到数百个，数千个，
甚至数十几万个

无法预计算：
每种组合都算好的话可能达数年。

即席查询：
即查即所见、任意多维组合分析

2 探索性即席分析的技术特性

实时导入

- 数据产生后约1~2分钟，系统内可查
- 每天千亿增量，总量可达万亿

多维分析

- 任意维度组合统计分析，任意维度过滤筛选
- 像百度那样快速的搜索与响应

即席查询

- 任意维度组合统计分析，任意维度过滤筛选
- 像百度那样快速的搜索与响应

超快排序

- 百亿数据，2台24core机器，秒级时间排序。

3 探索性即席分析的功能要求

4 业界现有主流方案存在的问题

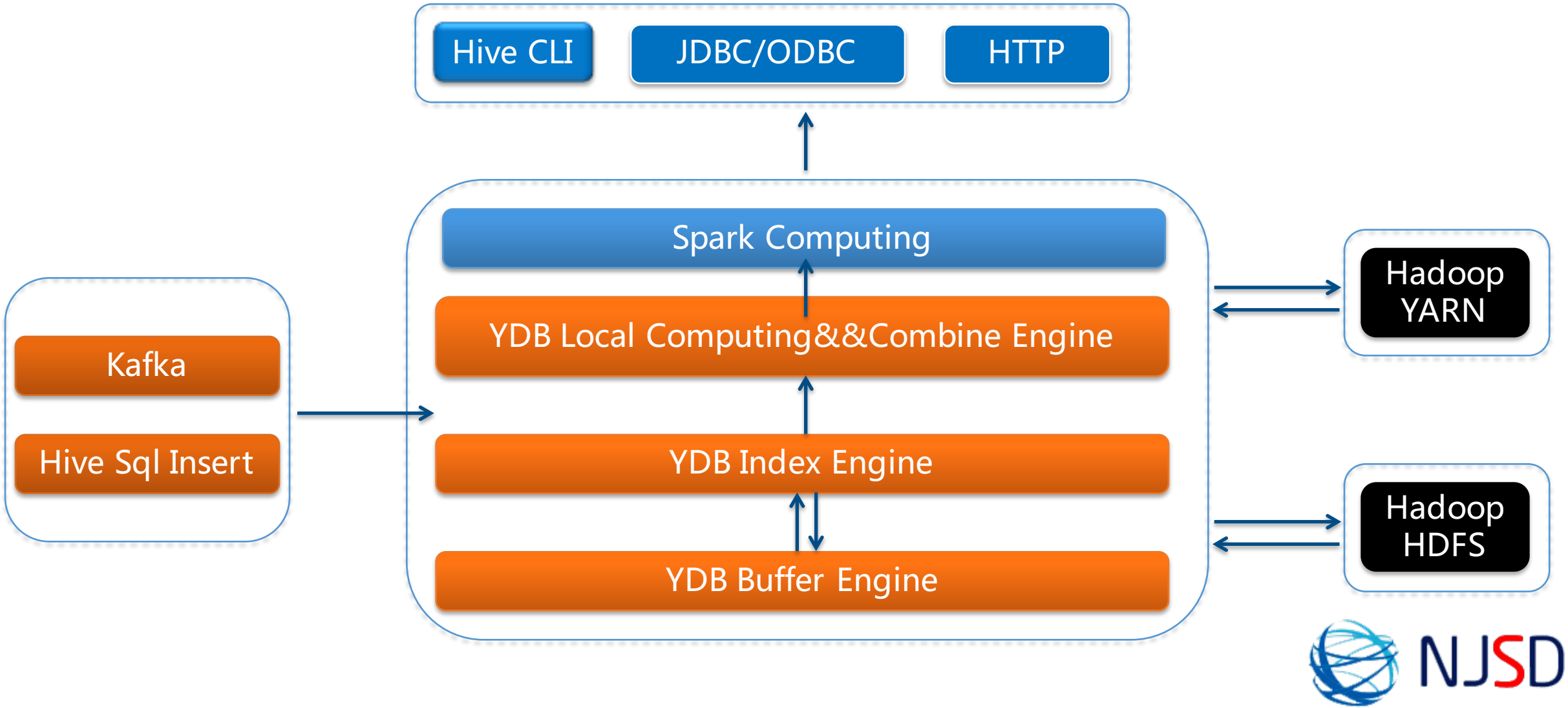
01	02	03
Hive，Spark SQL，SQL on Hadoop	HBase，KV型NoSQL数据库	Kylin
纯粹的暴力扫描	**只能局部计算，不灵活**	**本质上是预计算，只能看特定的维度、粒度**

第三章

千亿规模的即席踪迹分析关键技术实现

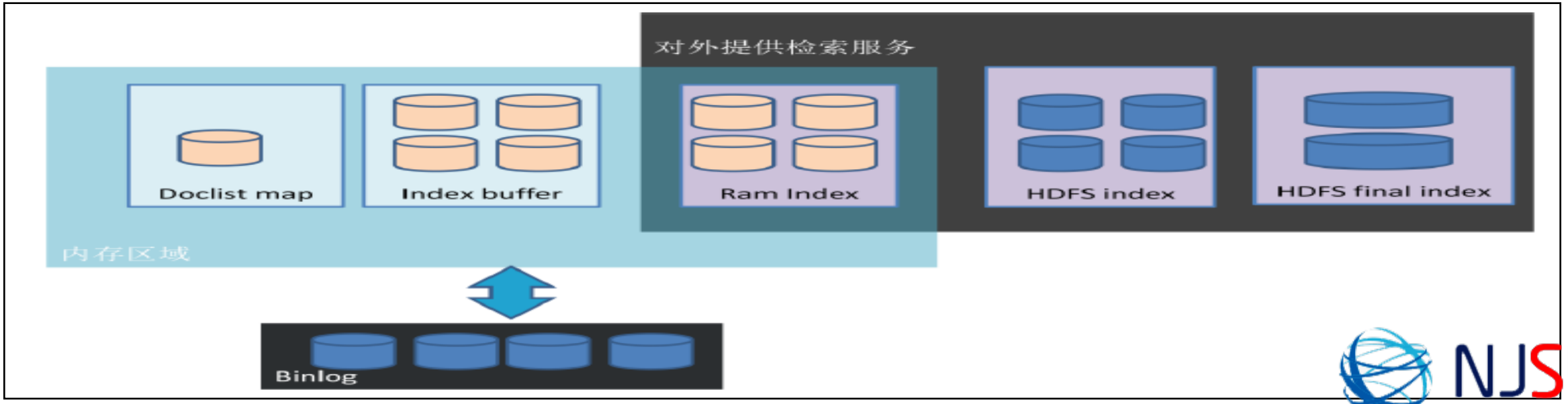
1. 架构描述
2. 分布式实时索引
3. 技术原理
4. Spark Bug修改源码

1 架构描述


Hive CLI
JDBC/ODBC
HTTP
Spark Computing
YDB Local Computing&&Combine Engine
YDB Index Engine
YDB Buffer Engine
Kafka
Hive Sql Insert
Hadoop YARN
Hadoop HDFS

2 我们在hdfs之上的分布式实时索引

```
[root@ydbslave01 ~]# hadoop fs -du -h  /data/ycloud/ydb/ydbpath/index/ydb_trade_demo
29.9 G   /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index/20151011/part-00000
33.6 G   /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index/20151011/part-00001
30.4 G   /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index/20151011/part-00002
32.3 G   /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index/20151011/part-00003
28.2 G   /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index/20151011/part-00004
31.3 G   /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index/20151011/part-00005
30.4 G   /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index/20151011/part-00006
31.6 G   /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index/20151011/part-00007
```



3 利用大索引技术跳过不需要的行

汉语拼音音节索引

1. 每一音节后举一字做例，可按例字读音去查同音的字。
2. 数字指本字典正文页码。

大数据就好比是一本新华字典

大多时候不需要一页一页的翻

- 等值匹配：

如 qq="165162897"

- 支持 in 操作，

如：indexnum in (1,2,3)

- >,<,>=,<=,区间查询的写法

clickcount >="10" and clickcount <="11"

- 对于带有范围的过滤筛选，使用下面的方式能提升查询效率

indexnum like "({0 TO 11}) "　　不包含边界值

indexnum like "([10 TO 11]) "　　包含边界值

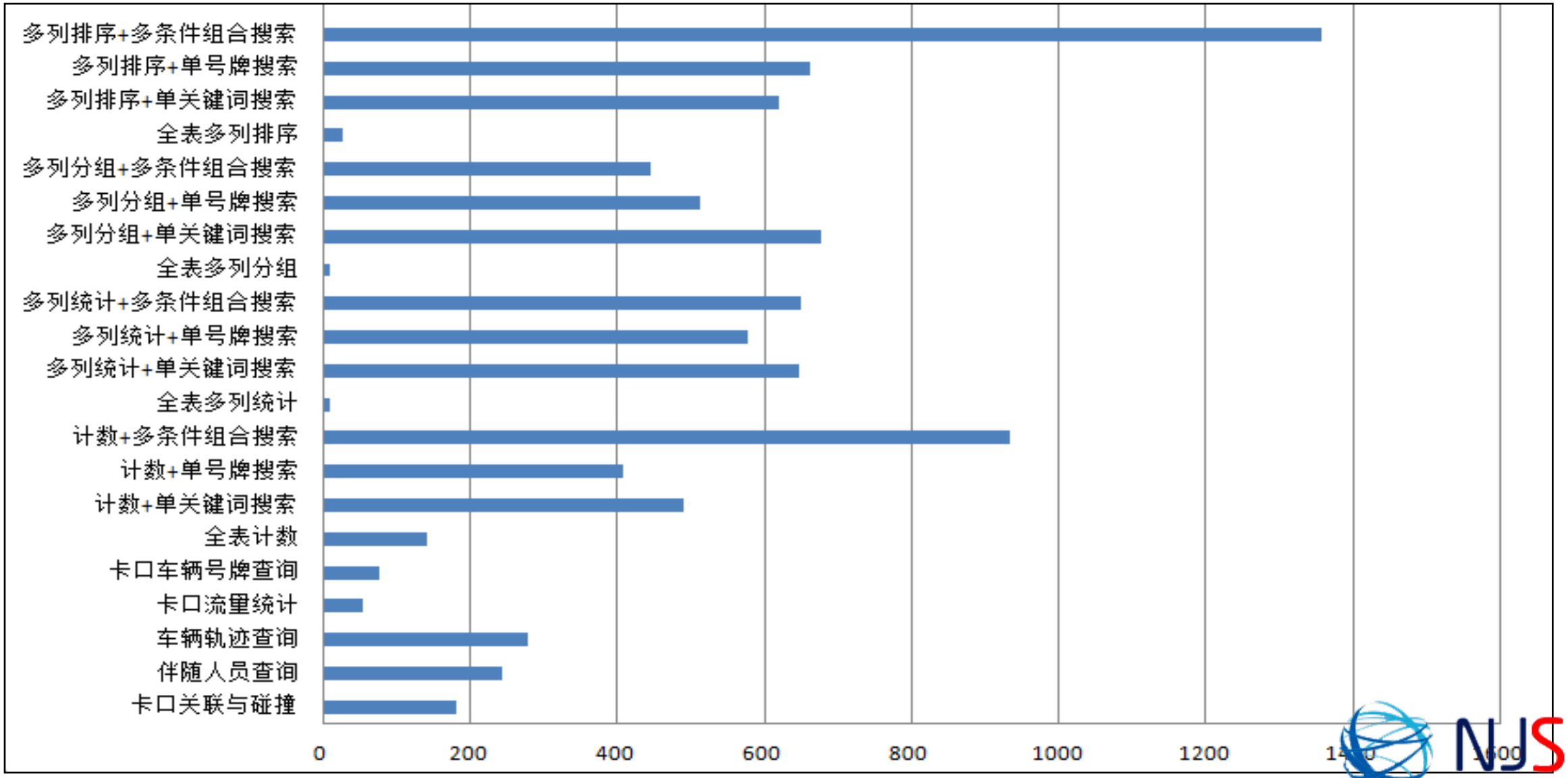
- 不等于的写法

label<>"l_14" and label<>"l_15"

- 过滤条件可以进行 and 与 or 的组合

indexnum="1" or indexnum="2" or (clickcount >="5" and clickcount <="

4 性能对比-千亿数据秒级检索与过滤



5 采用blockSort实现2台机器百亿数据秒级排序

冒泡排序？

快速排序？

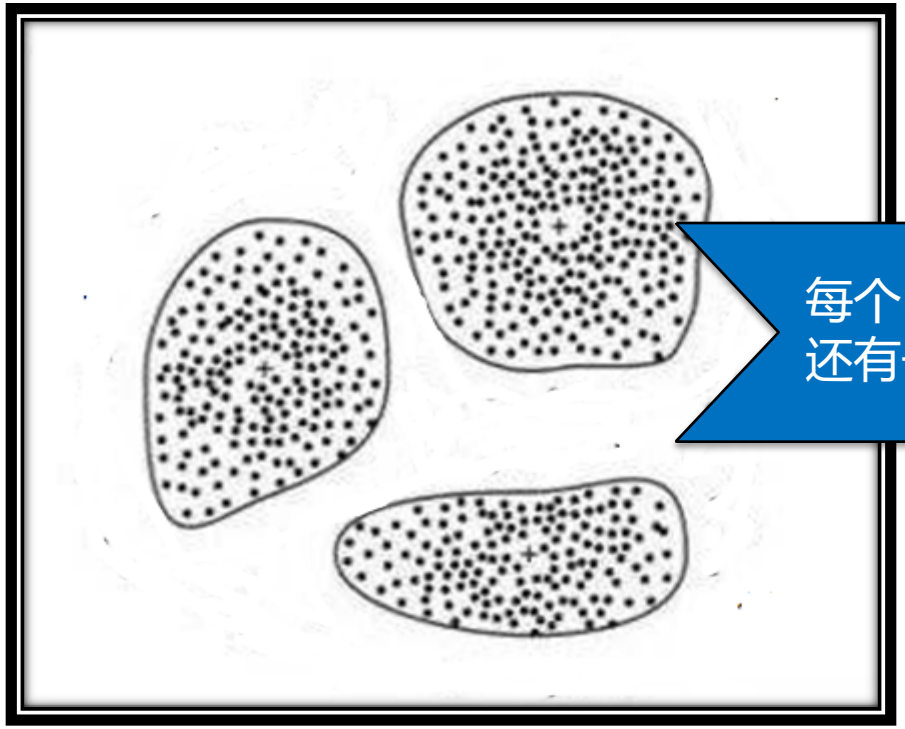
插入排序？

希尔排序？

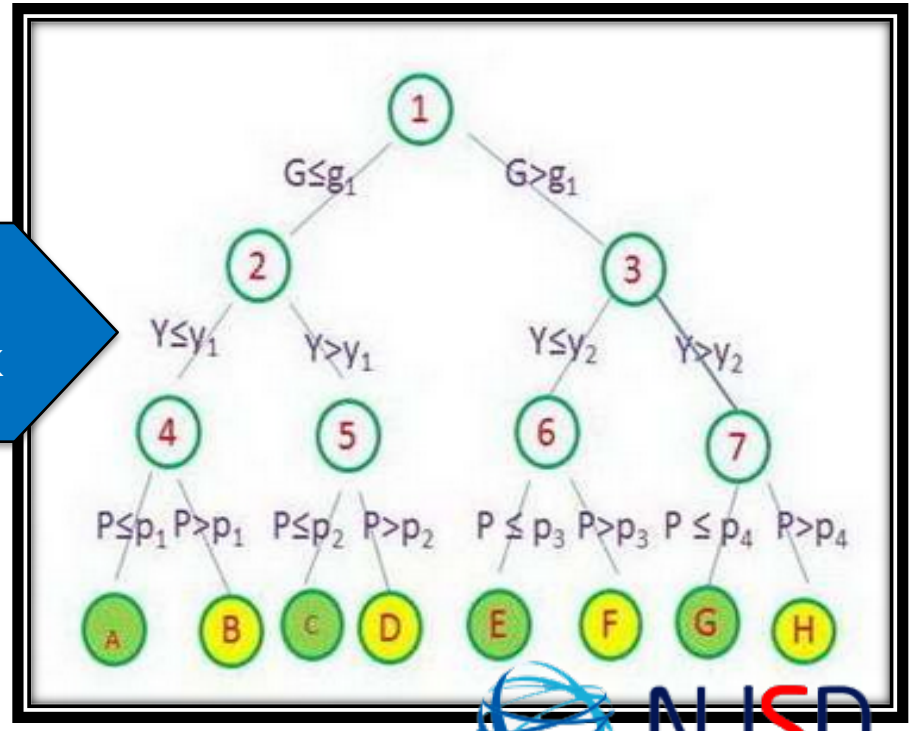
哈！两台机器，就算CPU够快，磁盘也转不了那么快！

将数据按照大小预先划分好，如划分成 大、中、小三个块(block)。

如果想找最大的数据，那么只需要在最大的那个块里去找就可以了。



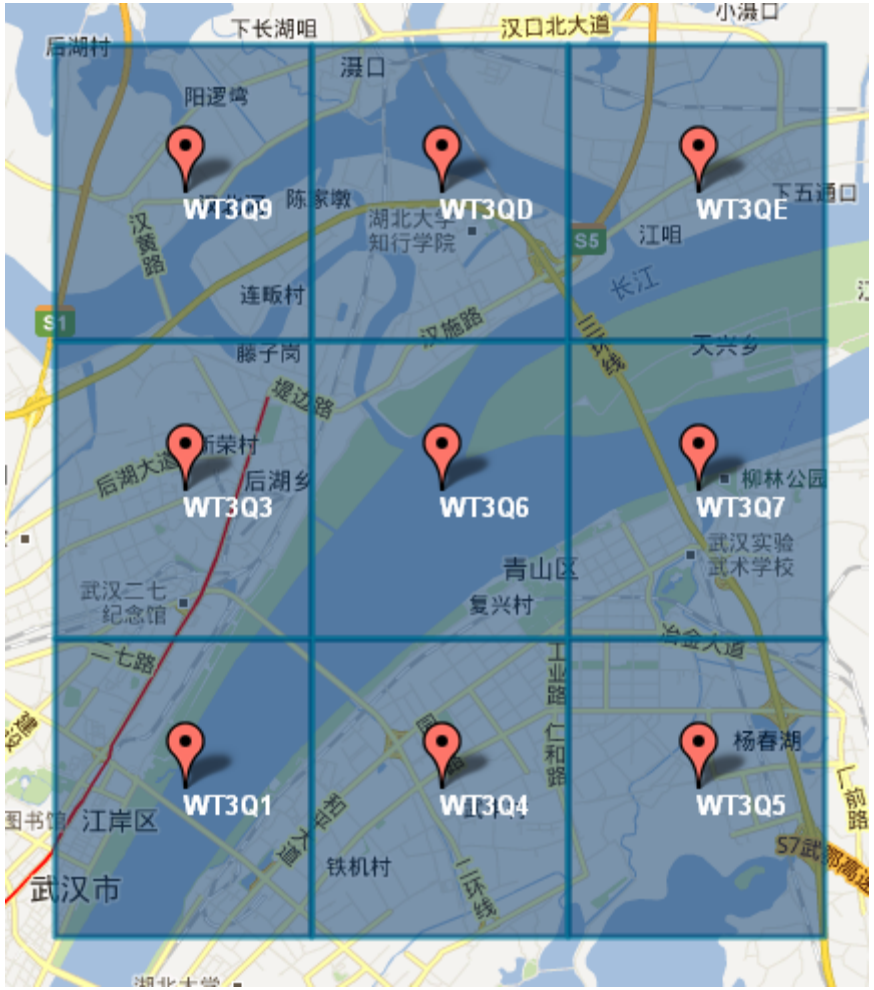
每个block内还有子block



6 性能对比-百亿数据秒级排序

amtint 列筛选	筛选后条数	排序方式	YDB BlockSort	Spark
无筛选	100 亿	降序	3.3	1118
		升序	3.6	1085
100 TO 900	80 亿	降序	1.5	1093
		升序	1.3	1070
100 TO 600	50 亿	降序	1.53	1104
		升序	1.38	867
100 TO 200	10 亿	降序	7.00	1115
		升序	1.11	1131
100 TO 110	1 亿	降序	2.1	1160
		升序	3.44	1114
100 TO 101	0.1 亿	降序	10.67	1089
		升序	7.0	110

7 位置感知搜索-百亿数据2台机器1秒圈定数据



8 采用标签技术-提升统计分析性能

原始值
张三
王五
李四
张三
李四
张三
王五
李四
赵六
张三
王五
张三

给原始值排重后由小到大加标签

标签值	字典值
0	张三
1	李四
2	王五
3	赵六

1 优点

- **重复值仅存储一份**，可以减少存储空间占用。
- 标签值采用**定长存储，可随机读取。**
- Group by分组计算的时候，使用标签代替原始值，**数值型计算**速度比字符串的计算速度快很多。
- 标签值的大小原始值的大小是对应的，故**排序**的时候也仅读取标签进行排序。
- 标签比原始值占的内存少。

2 缺点

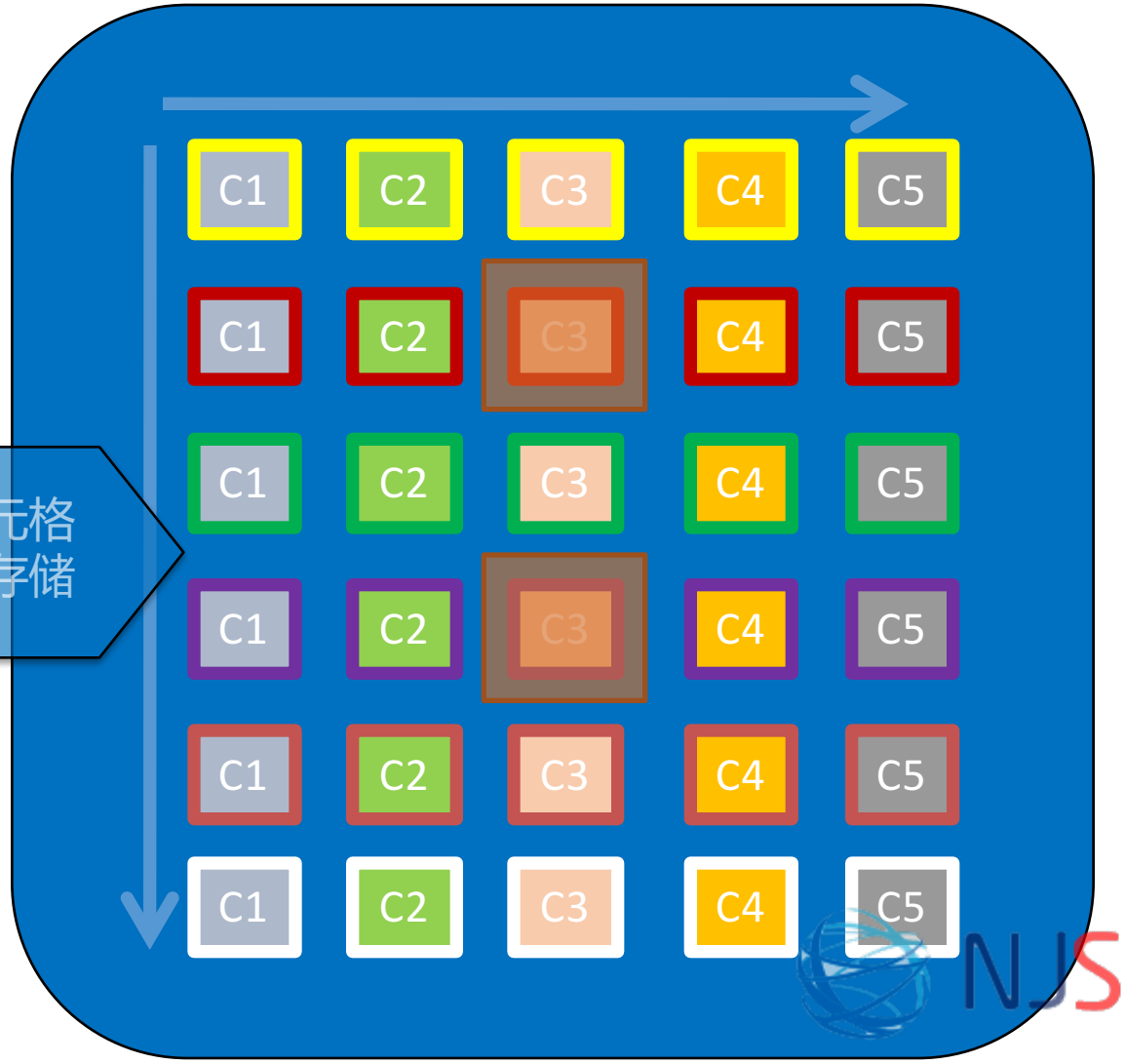
- 如果数据重复值很低，存储空间相反比原始数据大。
- 如果重复值很低，且查询逻辑需要大量的根据标签值获取原始值的操作的时候，性能比原始值慢。

9 比列存储更精细的按照单元格标签定长存储

按行存储

按列存储

按单元格
定长存储



10 采用两段式查询与延迟读取技术 减少不必要的IO消耗

```
select phone,name, lon,lat,logtime,speed,mac,content
from spark_txt order by logtime desc limit 10;
```

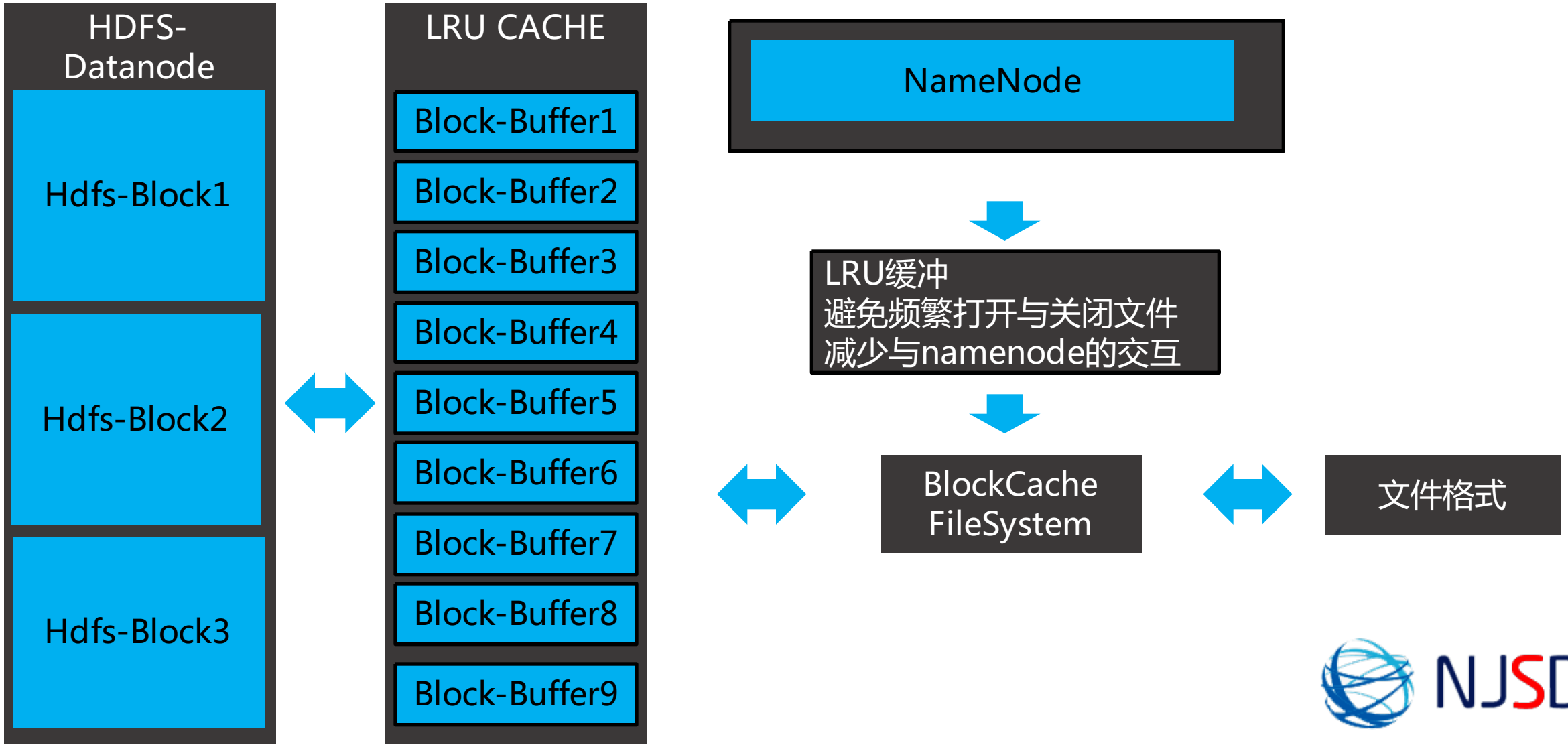
第一次查询

只读需要排序的列，以及行ID,其他列并不读取
不会获取数据的真实值,仅仅读取数据标签

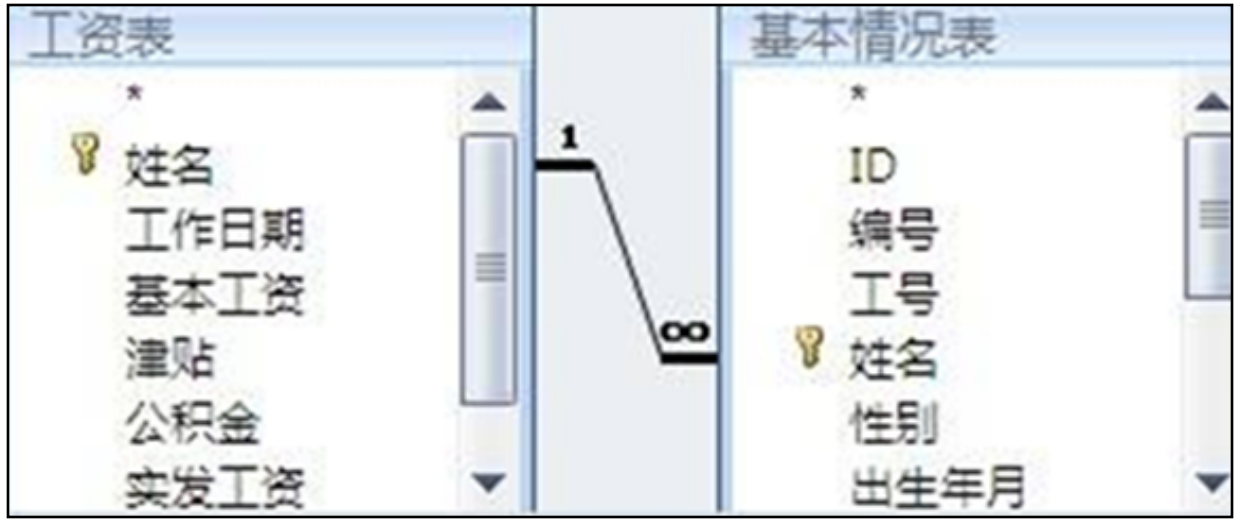
第二次查询

将剩余过滤后的10行的其他列的值返查回来
延迟读取可以节省磁盘IO与跳跃次数。

11 持久化的进程+process local+LRU Cache



12 巧用Spark的brocast广播 提升join性能




```
select a.info from a
left join b on (a.id=b.id)
where b.usernick='张三'
```

问　题

- A 表需要全表扫描么？是不是利用上索引更好？
- 索引的in操作，如何获取到B表的筛选结果？

思　路

- 借助Broadcast的p2p特性，将小表数据广播到每个节点。
- 利用索引让小表关联更高效。

13 高纬值列group by，特定场景优化

13401007670	5次
17099831107	3次
13812458374	9次
18303337163	3次

```
select userId,
    count(*) from table
group by userId
    limit 100
```

问　题

- phone重复值很少，group by性能太低。
- shuffle性能太差，耗费内存很多。

思　路

- 利用索引只计算需要返回的组。
- 倒排索引就是按分组存储的，没必要二次分组。
- 搜索提示对于同一个组内的数据，只需读一条。
- distinct的优化

14 索引中的索引（待开发）

1. 任意数据类型的高性能极速排序
2. 跨多列索引用于多列极速分组排序
3. 跨越多列的多值列索引，多列的过滤

索引中的索引

- 索引的索引提升特定场景的性能-tlong
- 解决多值列协同检索，嵌套检索的新思路

影响稳定性的Spark Bug源码修改

- 二十一、 spark 内存泄露
  - 1. 高并发情况下的内存泄露的具体表现
  - 2. 高并发下AsynchronousListenerBus引起的WEB UI的内存泄露
  - 3. AsynchronousListenerBus本身引起的内存泄露
  - 4. 高并发下的Cleaner的内存泄露
  - 5. 线程池与threadlocal引起的内存泄露
  - 6. 文件泄露
  - 7. deleteONExit内存泄露
  - 8. JDO内存泄露
  - 9. listerner内存泄露
- 二十二、 spark源码调优
  - 1. SessionState 的创建目录 占用较多的时间
  - 2. HiveConf的初始化过程占用太多时间
  - 3. 广播broadcast传递的hadoop configuration序列化很耗时
  - 4. 对spark广播数据broadcast的Cleaner的改进
- 第十三章YDB常见问题FAQ

这些地方修正后
基于spark 300并发 上亿次的查询没问题。
感兴趣的可以下载《 YDB编程指南》，了解并修复这些问题。

谢谢大家！

1820150327

17099831107

www.ycloud.net.cn