

千亿大数据 即席踪迹分析

核心技术原理与实现详解



个人介绍



母延年 十年行业经验 曾任职 新浪、酷六、阿里与腾讯

11年：支付宝黄金策的海狗

12年：阿里开源项目

MDRLL（多维分析）与 JStorm（流计算）

14年：腾讯的Hermes(每天千亿总量万亿)

15年：技术创业，潜心研究技术。

什么是即席踪迹分析，那些场景适合即席踪迹分析

现有大数据技术在即席踪迹分析上存在的问题

千亿规模的即席踪迹分析关键技术实现

什么是千亿数据即席踪迹分析？



千亿

数据量在每天千亿条，总量在万亿规模的实时数据。

即席

首先含有当场，即兴的意思。没有事先准备，想查哪里就查哪里。其次要求很快的响应，不能等待太久，即查即所得。

踪迹

了解数据里的风吹草动，行踪查询，轨迹分析。一般指的是时间、位置、操作等有关。

即席踪迹分析是做什么的

通过在大数据中快速的查询比对。发现目标的行踪轨迹。

常用于破获案件、截获情报、舆论定位、排查故障、资金流向追溯、通话记录分析等。

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

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



**由于公安场景比较敏感，暂不对外公布
本PPT仅保留技术实现.....**

这些场景带来的挑战！



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

转换为文本

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



目前业界现有主流方案存在的问题

Hive , Spark SQL , SQL on Hadoop : 纯粹的暴力扫描

HBase , KV型NoSQL数据库 : 只能局部计算 , 不灵活

Kylin : 本质上是预计算 , 只能看特定的维度、粒度。



“千亿即席踪迹分析” 的性能比对

“千亿即席踪迹分析”的必备特性



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

- 想查什么立即就查，不需要预计算。
- 保存原始数据，任意维度组合均可见。

超快排序

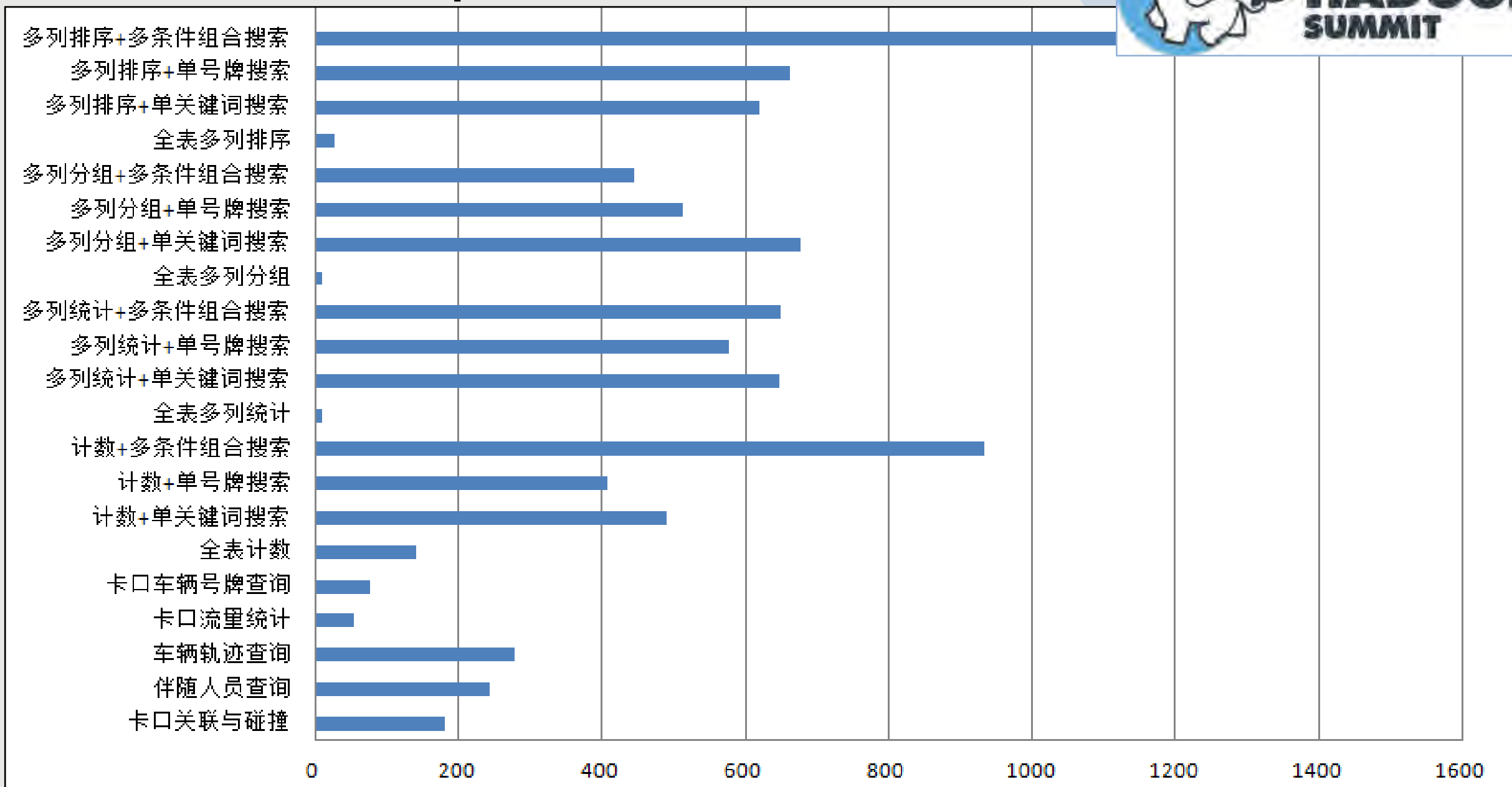
- 百亿数据，2台24core机器，秒级时间排序。

新方案的排序性能对比



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	1110

检索过滤性能相对于原生spark提升倍数



机动车稽查



测试环境	
数据条数	200亿条
数据大小	1000G
CPU	2*6核
内存	64GB
机器台数	2台

场景测试	
行车轨迹查询/重点车辆分析	0.43秒
同行车辆分析	1.56秒
区域碰撞分析	1.23秒
昼伏夜出、落脚点分析	1.5秒
陌生车辆分析	7.9秒
嫌疑车牌模糊搜索与定位	1.6秒

完备的功能-复杂的SQL查询咱必须要支持

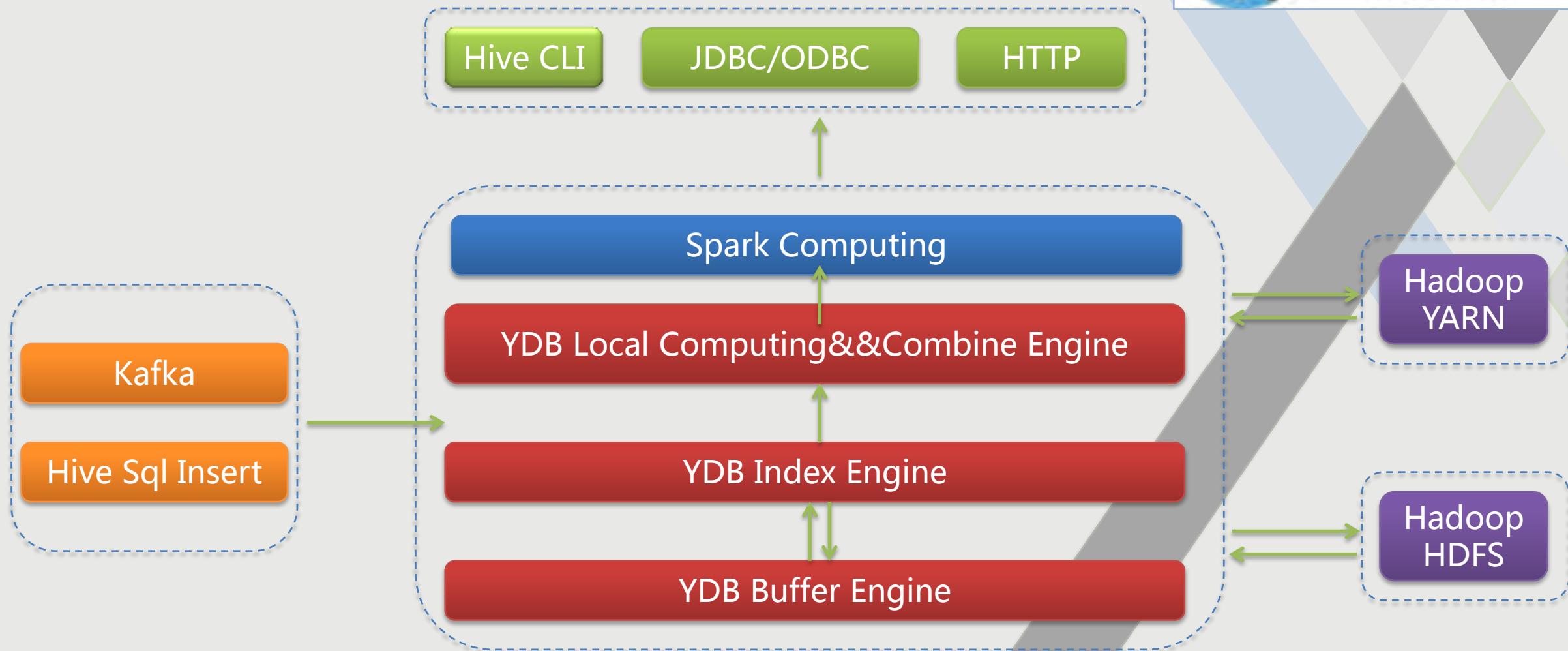


功能	概述
检索过滤	等值匹配,支持 in操作,>,<,>=,<= and与or的嵌套组合
统计分析排序	单/多列group by,max,min,sum,avg,count,distinct,order by
复杂SQL	自定义udf,udaf,udft,SQL多层嵌套,union,多表关联join
模糊查询	全文检索,临近搜索,相似文本(文章)搜索, like。
数据类型	string,int,long,float,double,一列多值,地理位置,行存储,列存储
中文分词	内嵌二元分词,IK词库分词,也可自定义或拓展第三方分词。 YDB自带的ngram多元分词也更适合数字,邮箱,车牌,符号的匹配。



“千亿即席踪迹分析”的技术实现

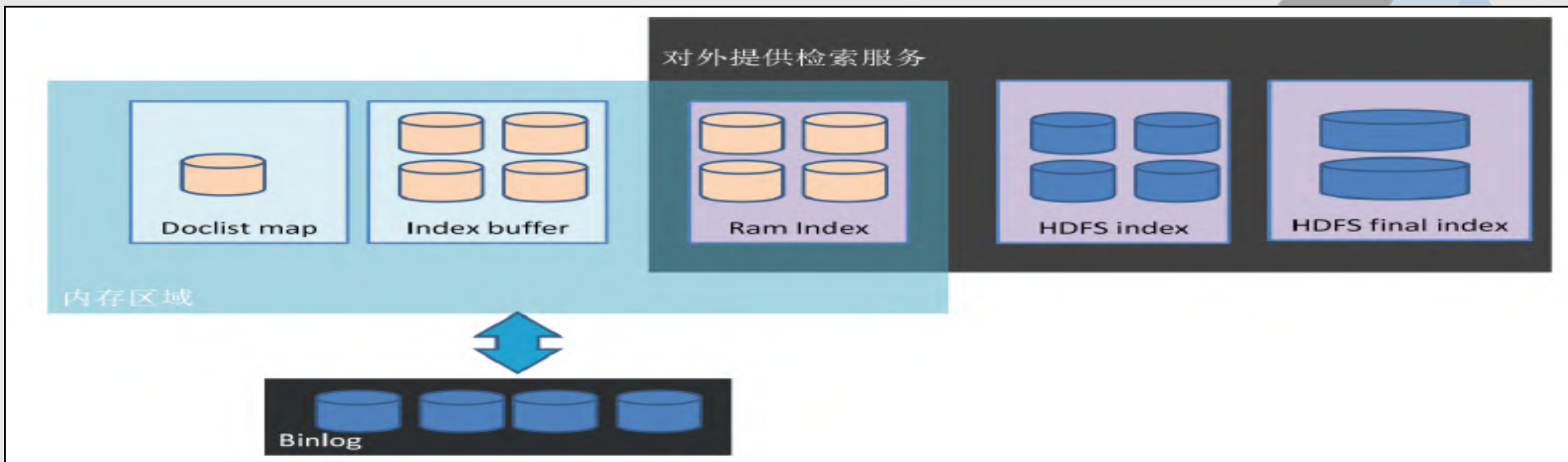
架构描述



我们在hdfs之上的分布式实时索引



```
52204401450  /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index
Droot@ydbslave01:~$ hdfs -du -h /data/ycloud/ydb/ydbpath
29,9 G /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index/20151011/part-00000
37,6 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
29,2 G /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index/20151011/part-00004
31,5 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
29,8 G /data/ycloud/ydb/ydbpath/index/ydb_trade_demo/index/20151011/part-00008
```

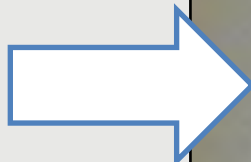


利用大索引技术跳过不需要的行

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 <="10")



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。

a	啊	1	cai	猜	41	ci	词	73
ai	哀	2	can	餐	42	cong	聪	75
an	安	3	cang	仓	43	cou	凑	76
ang	肮	5	cao	操	43	cu	粗	76
ao	熬	5	ce	策	44	cuan	撺	77
			cen	岑	45	cui	崔	78
			ceng	层	45	cun	村	79
			cha	插	45	cuo	搓	80
			chai	拆	48			
ba	八	7	chan	搀	48			
bai	白	10	chang	昌	51	da	搭	81
ban	班	12	chao	超	53	dai	呆	83
bang	帮	14	che	车	54	dan	丹	86
bao	包	15	chen	尘	55	dang	当	88
bei	杯	18	cheng	称	57	dao	刀	90
ben	奔	20	chi	吃	59	de	德	92
beng	崩	22	chong	充	62	dei	得	94
bi	逼	23	chou	抽	64	den	抻	94
bian	边	27	chu	初	65	deng	登	94
biao	标	30	chua	欸	68	di	低	95
bie	别	32	chuai	揣	68	dia	哆	99
bin	宾	32	chuan	川	69	dian	颠	99
bing	兵	33	chuang	窗	70	diao	刁	102
bo	玻	35	chui	吹	71	die	爹	103
bu	不	38	chun	春	71	ding	丁	104
ca	擦	40	chuo	戳	72	diu	丢	106

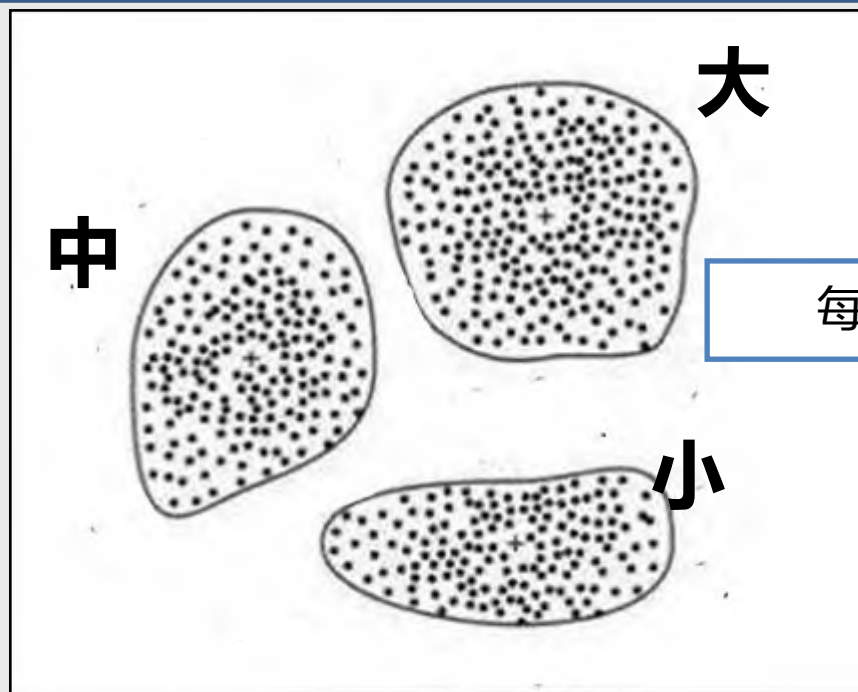
采用blockSort实现2台机器百亿数据秒级排序

采用冒泡排序、快速排序、插入排序还是希尔排序？

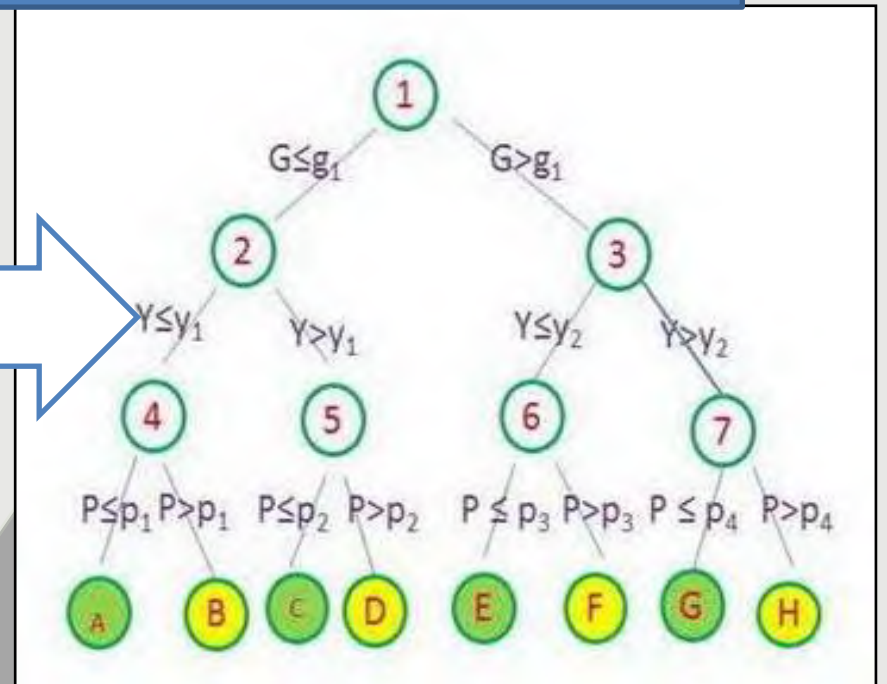
别闹了！就两台机器，就算您的CPU能算的那么快，您的磁盘也转不了那么快！

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



采用标签技术的按单元格存储-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

优点

1. 重复值仅存储一份，可以减少存储空间占用。
2. 标签值采用定长存储，可随机读取。
3. 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。
4. 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。
5. 标签比原始值占的内存少。

缺点

1. 如果数据重复值很低，存储空间相反比原始数据大。
2. 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。

因为标签的存在-我们的数据按列定长存储



按行存储

Row 1	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row 2	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row 3	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row 4	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row 5	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10

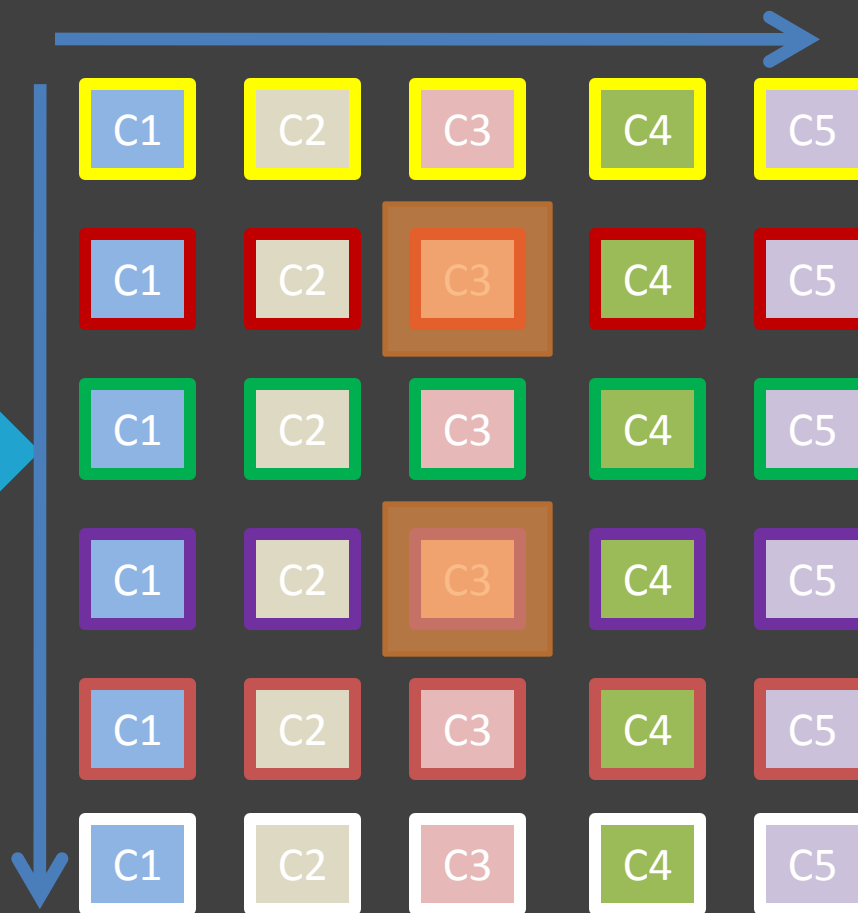
按列存储

Row 1	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row 2	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row 3	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row 4	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row 5	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row 6	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row 7	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row 8	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
.....	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Row n	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10

age 1 age 2 age 3 age 4 age 5 age 6 age 7 age 8 age 9 age 10

既按行 又按照列

按列定长存储



采用两段式查询与延迟读取技术 减少不必要的IO消耗



SQL场景：按照日志时间排序

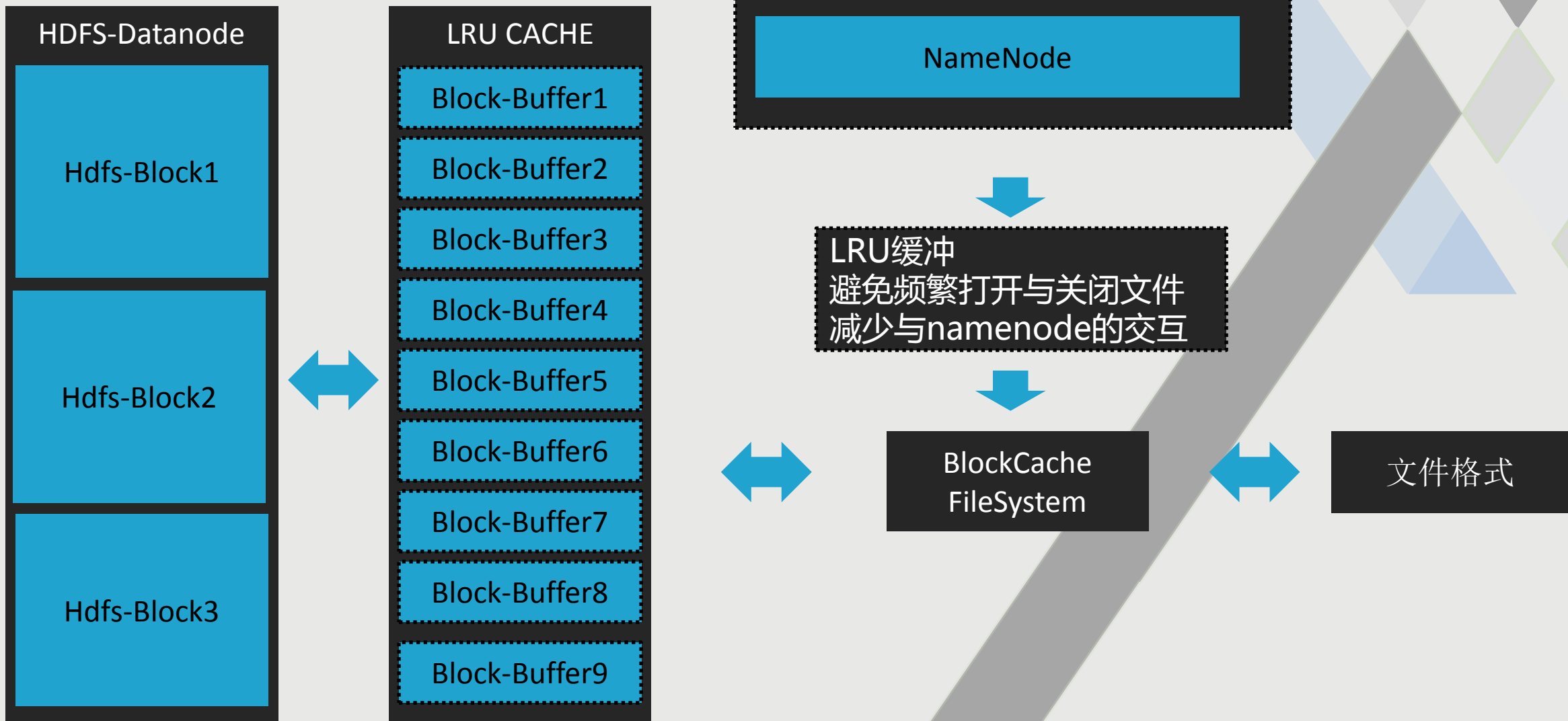
```
select
  phone,name, lon,lat,logtime,speed,mac,content
from spark_txt
  order by logtime desc
limit 10;
```

第一次查询，只读需要排序的列，以及行ID,第一次查询不会获取数据的真实值,仅仅读取数据标签

第二次查询，才将剩余过滤后的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。

持久化的进程+process local+LRU Cache



基于spark，细粒度的原子调度，一个比job更细粒度的调度



现有spark任务调度的缺点

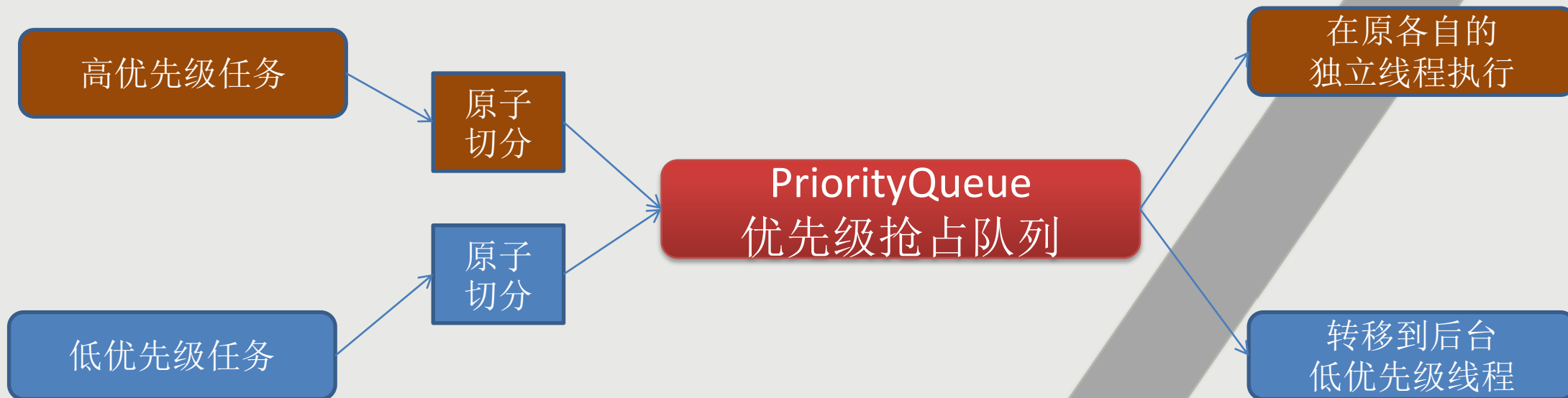
- 1.只有fifo与fair 没有容量调度，一个任务可以占满全部的资源。
- 2.调度粒度为job级别，资源腾出时间为分钟级别，是等待还是kill掉。

原子调度：

- 1让用户能够精细化的管控资源，任务切分粒度更细。
- 2.优先级较高的任务可以毫秒的时间抢占优先级低的任务资源

原子任务切分

将一个job切分成更细的小的原子粒度，如每个64m的task任务按照8k切分成一万个原子单元





```
select * from A where  
userid in (select userid from B where action= 'TO DO STH')
```

问题：

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

思路：

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

基于spark , 但我们修改了spark的一些影响稳定的BUG



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

**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。**

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

谢谢大家!

