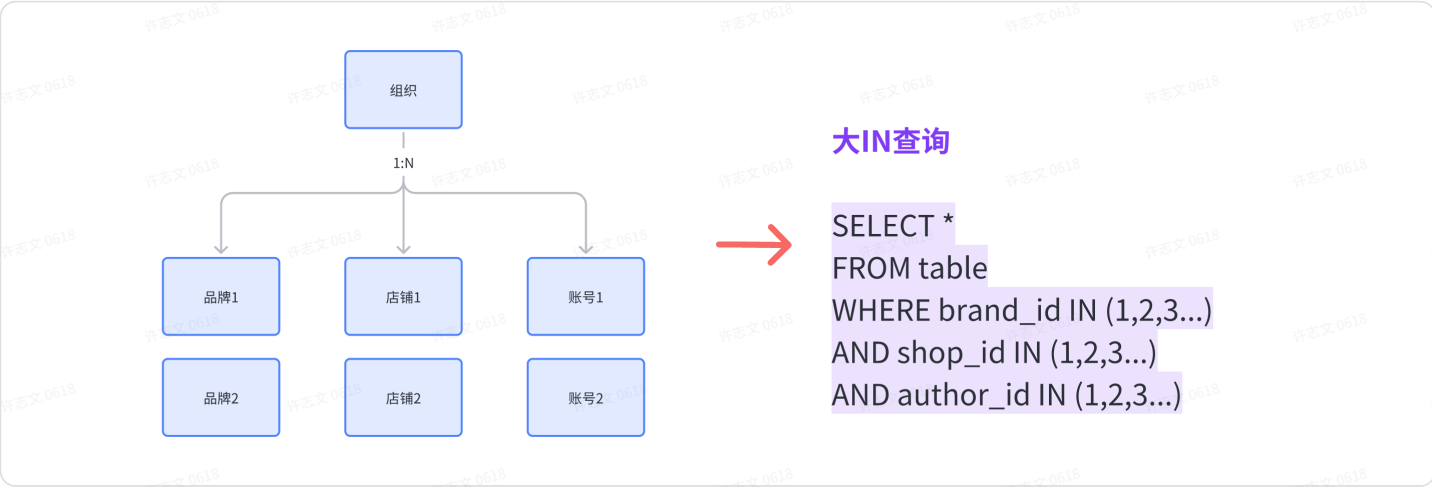


罗盘策略实时接口HSAP方案改造

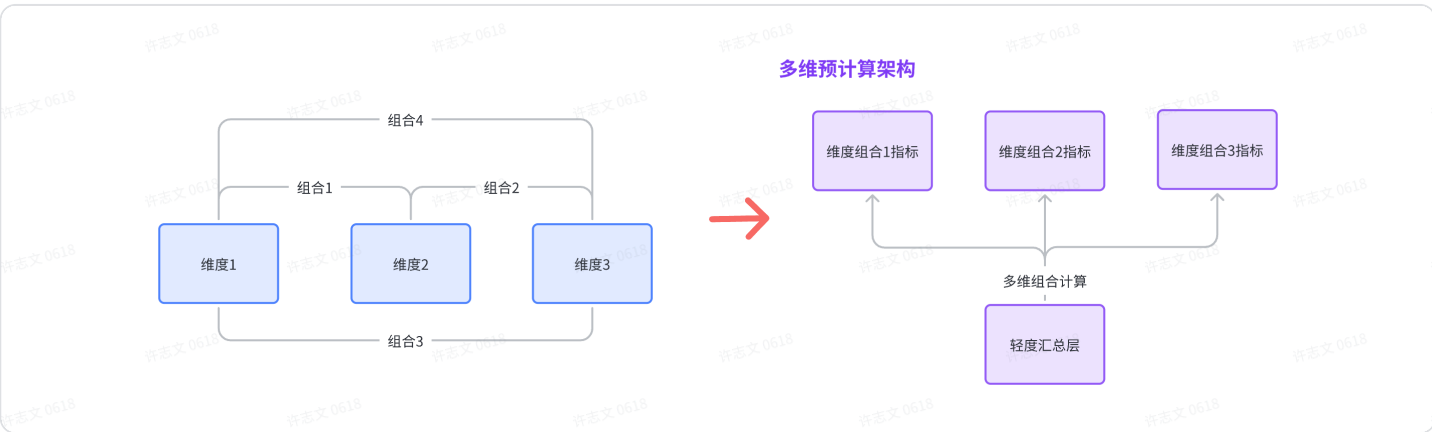
1. 背景

罗盘策略版是服务于品牌集团的生意增长引擎，目前策略实时接口存在性能差，接口报错率高等问题，从业务向存储的角度看，目前主要存在以下两个问题：

1. 企业组织账号体系的核心是子品牌、商家和达人的资源绑定 [罗盘·策略相关账号体系](#)。从实时数据角度看，基于基础数据灵活使用的建设思路，没有引入组(织)概念，而完全依赖于绑定关系做查询过滤来获取品牌所属指标。然而组织所绑定的资源数量可能非常多，达到数百个，产生了大IN查询而导致底层存储NDB/Abase+数据服务OS的综合查询性能无法满足。



2. 相同资源间、不同资源间的生意协同频繁，看数场景以灵活多变的ad-hoc查询分析为主。从实时数据角度看，基于数据查询时效性的保障，需要进行多维组合的指标预计算。然而随着业务查询诉求的灵活变化，预计算指标越来越多，但用户实际查询的指标可能只占到其中的很少一部分（并不是所有用户都会频繁使用统一功能），造成了资源浪费问题，也给数仓运维增加了额外的成本。



从电商实时数仓的角度看，电商实时数仓目前大多使用分层加工(ODS->DWD->DWS->(APP)->ADS)的方式。在对外场景,使用Abase提供点查, ByteNDB/ByteSQL(即将下线)/ES提供列表&排序查

询；对内场景亦大量使用Abase/ByteNDB, 少量使用ClickHouse；另外在近实时、实时诊断、稳定性看板等场景逐渐应用数据湖ByteLake。

虽然对各个场景使用了不同的存储，但是总体来说，也存在以下3个问题：

- **数据冗余存储**: 同一份数据常常需要冗余的写入不同的存储, 以支持高QPS的点查和中低QPS的列表&排序类查询
- **存储存在局限**: 存储在承担自身职责时,也存在局限, 比如ByteNDB因dbatman中间件带来的语法限制、对写入能力的支持有限, ES集群负载高、无法支持高QPS的查询等
- **计算链路压力大**: 大多数数据都在计算层进行了预聚合, 计算层加工成本高, 任务数量多, 人力和资源都消耗较大

2. 目标



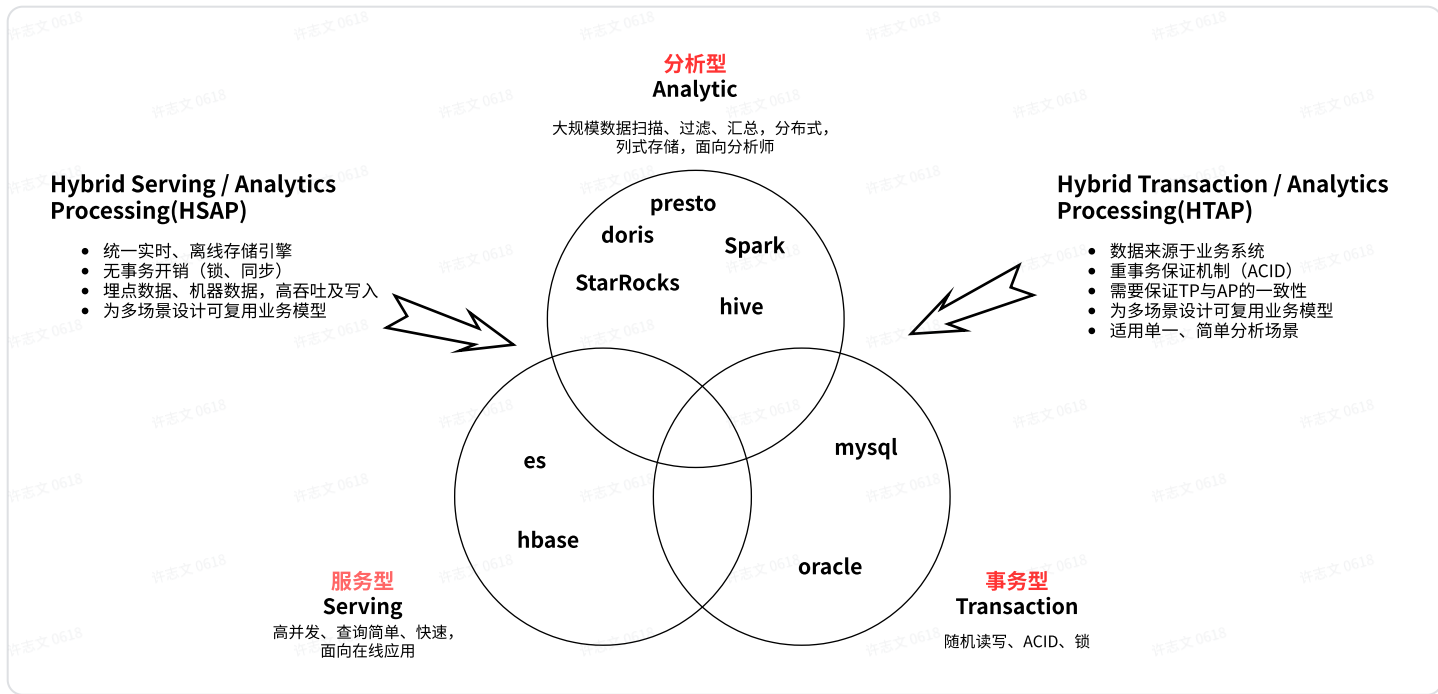
对于罗盘实时业务或者实时数仓角度看，希望存在某种存储，能够具有：

1. 高qps查询以及复杂分析能力
2. 具有高吞吐写入和更新能力
3. 数据在写入存储时的预计算能力，记录物化视图，提高查询效率

在罗盘**常见实时场景**中实现：

4. 大IN查询性能从5s->1s
5. API接口失败率从2%->0%。各API失败率参考 [策略实时数据表](#)
6. 简化多维预计算架构，以商品监控模块为例，作业数从19->3个。

3. 关于存储类型

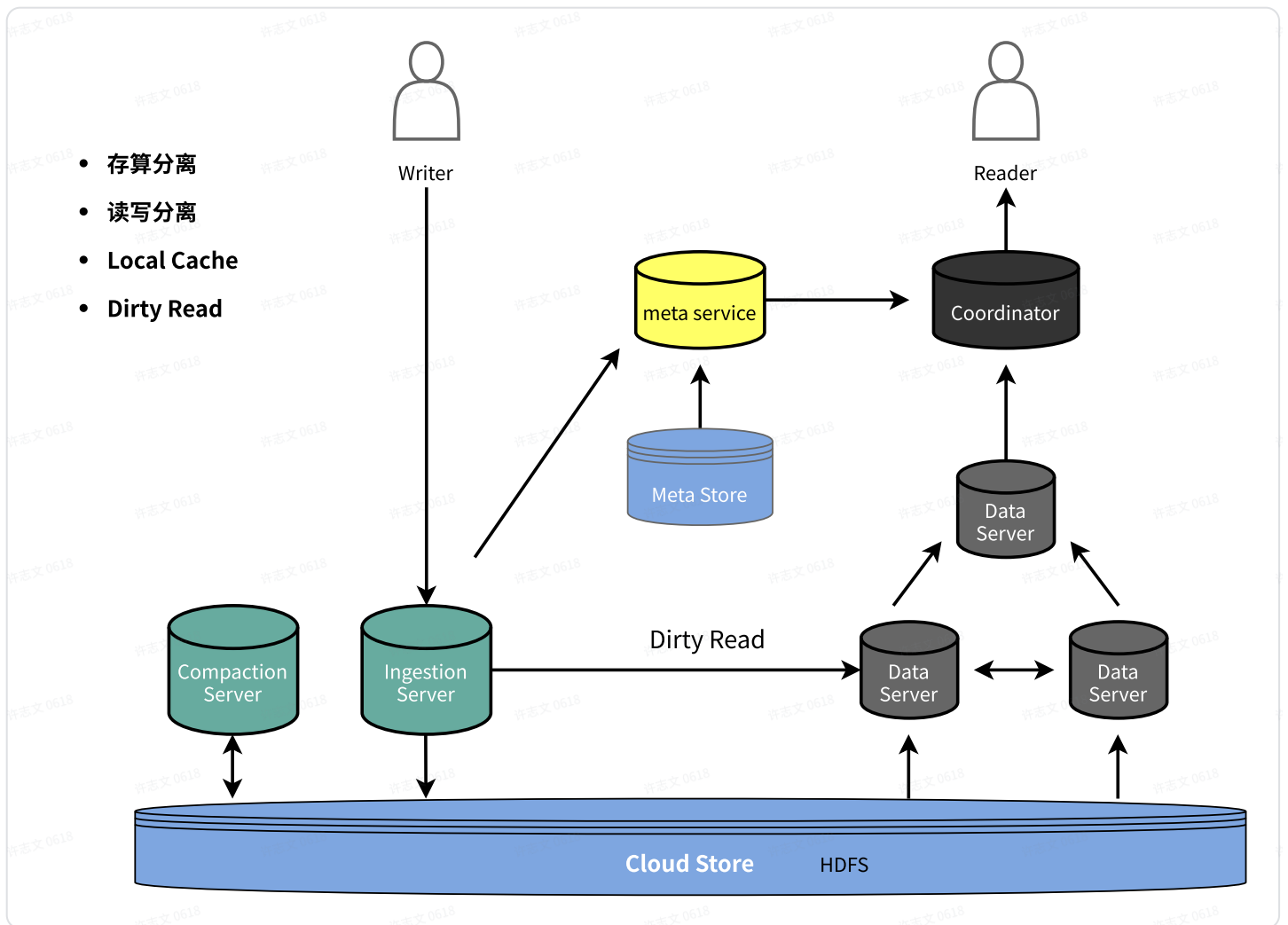


为什么需要HSAP/HTAP? [HTAP/HSAP 演进的背景和思考](#)

4. 字节HSAP系统Krypton

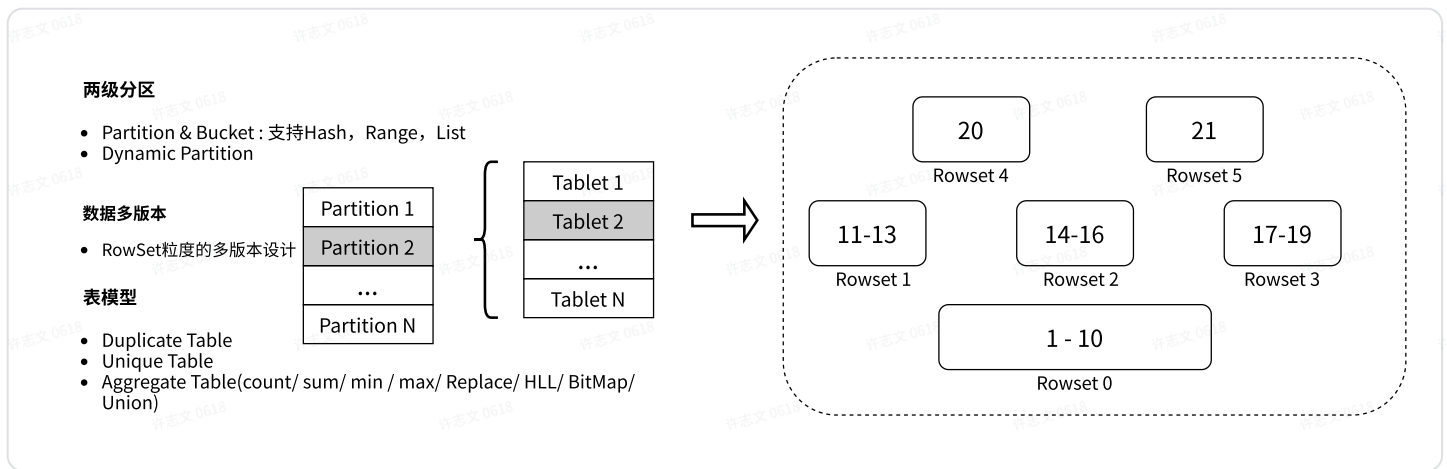
[Krypton介绍](#)

4.1 架构



- 存算分离：数据都放在Cloud Store上（HDFS），Meta Store放在分布式kv系统中。
- 读写分离：整个架构分读链路（图右边）和写链路（图左边），两个链路通过meta service进行同步。写链路也做了拆分，将计算密集型的任务（Compaction Server）专门拆分了出来，这样方便做该任务集群扩容。
- Local Cache：读链路的数据服务器中会有Local Cache，该组件是为了解决直接读Cloud Store的时延问题。Local Cache可以支持多种介质（DRAM, SSD等），极端情况下也可以把数据全部load到本地，这样相当于后续查询都走Cache。
- Dirty Read：架构本身存在一个问题，数据因为通过Ingestion Server写入，写入后并不会每条数据都Flush到Cloud Store中，而是数据会堆积到一个Buffer中，堆积到一定量后才会Flush。但是在这个Flush过程前，这部分数据（buffer中）都是不可见的。因此对于一些数据可见性要求高的场景，Data Server可以直接读取Ingestion Server中的数据（内存中数据）。

4.2 数据模型



整体数据模型参考doris。

- 两级分区：与doris结构类似。最外层是Partition，每个Partition下可以再分区成 Tablet(Bucket)。每个分区可以支持不同的partition策略。此外也可以做成Partition级别的TTL (Dynamic Partition) [动态分区](#)。
- 数据多版本：每个Rowset中都会有一个最低和最高的版本号，每次对Tablet中数据进行变更时会生成一个新版本。这些Rowset里最新的版本是这个Tablet的commit version(类似mysql的mvcc)。commit version之下的版本可见，之上版本不可见（实现Dirty Read）。
- 表模型：与doris基本相同。 [表设计和数据模型](#)

4.3 计算引擎优点

- 完全兼容mysql语法
- 多种Join Reorder算法支持
- Materialized view support：支持强一致实时的mv生成

4.4 适用场景

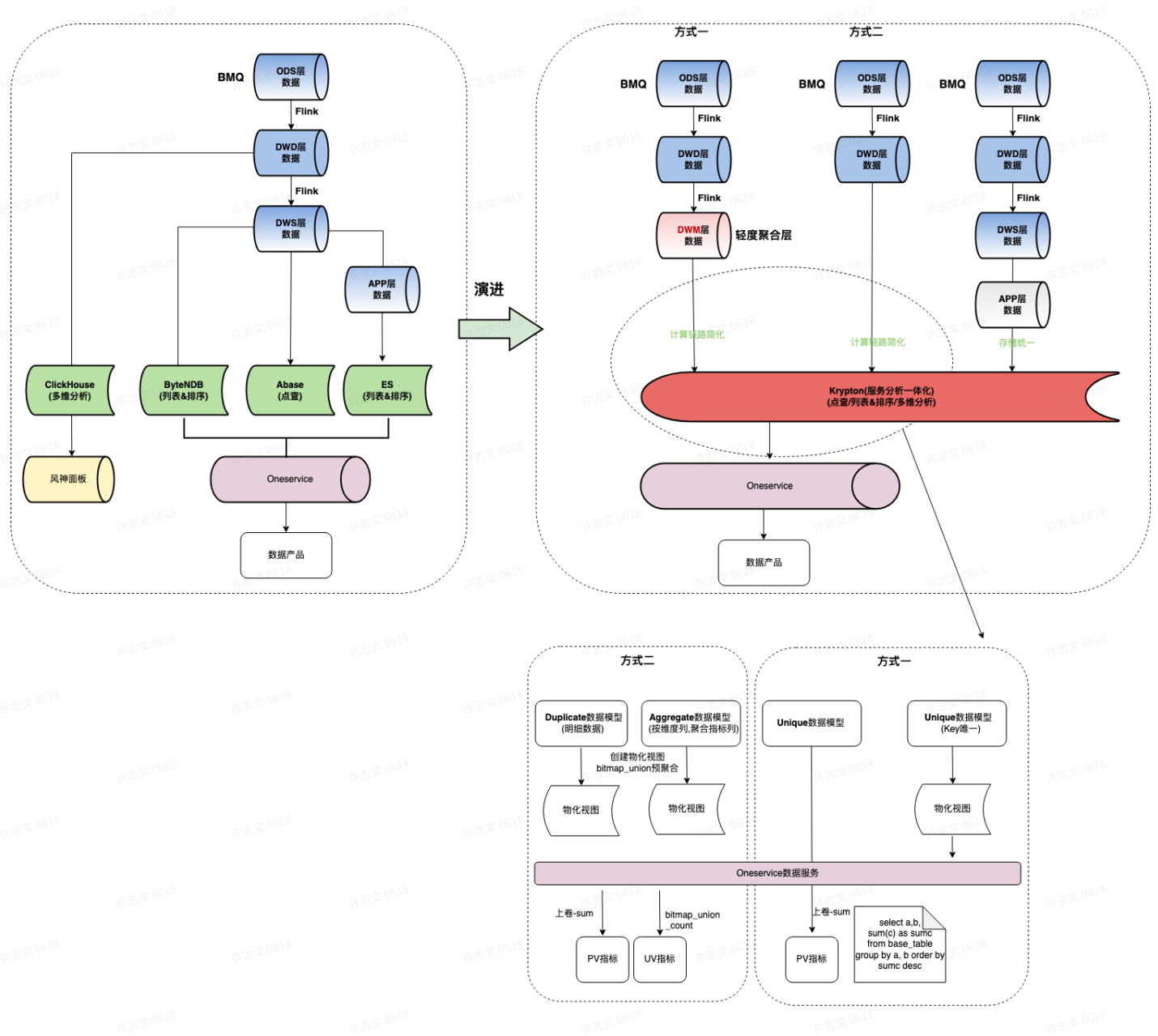
- 高 QPS点查场景。大约10w以内qps点查（先看看1w以内）。数据在写入存储时预计算，空间换时间，提高效率
- 对外Olap多维分析场景（大数据量级(亿级~百亿级)，qps>=100，数据更新时效性为秒级）。在对外场景能应用OLAP多维分析能力, 在满足性能要求的前提下, 提高数据的复用性。在需要基于同一数据源出多种维度组合的聚合指标时，减少计算层任务的数量，并降低数据加工的成本。例如罗盘单商品卡流量-商品列表(商品X天维度)，商品详情页-流量来源列表(商品X渠道X天维度)等这些取多维数据的场景比较多。
- 列表与排序场景；千qps的排序、聚合, 对标ByteNDB的索引查询和依赖ES的列表&排序类查询。
 - 相比于NDB，有更快的写入能力（测试结果，HSAP 3台机器写入avg=71w/s, 而ByteNDB单实例上限约为1w/s）；其次，由于语法限制，NDB的查询必须带上分片键等，在聚合查询时可能需要带多个分片键字段，使用可能不便利。hsap不受影响。
 - 相比于es，查询qps更高。es查询的复杂度越高，查询速度越慢。

4.5 和doris对比

	Doris	Kryptor
适用场景	<p>报表分析</p> <ul style="list-style-type: none">实时看板（Dashboards）面向企业内部分析师和管理者的报表面向用户或者客户的高并发报表分析（Customer Facing Analytics）。 <p>即席查询（Ad-hoc Query）。</p> <p>统一数仓构建。</p> <p>数据湖联邦查询等</p>	<p>基于Doris一些分析场景，对于fix（例如网络日志分析对历史数据）对实时数据进行实时监控和分析）构，利用node group技术在一份隔离</p>
索引类型	<ul style="list-style-type: none">前缀索引（Short key index）Ordinal 索引Zone MapBloom filter（主动添加）Bitmap（主动添加）	<p>多种索引支持：</p> <ul style="list-style-type: none">PK indexSkip indexBitmap indexBloom filter index <p>.....</p>
架构	<p>Doris只设FE(Frontend)、BE(Backend)两种角色、两个进程的主从架构。计算和存储一体。</p>	<p>存算分离和读写分离。doris在做如高频写入场景，短时间内会产生Compaction不及时，就会造成大写入速度。Krypton将Compaction以直接横向扩展，降低高频写入</p>
存储方式	<p>列存储，减少I/O消耗，提高查询性能</p>	<p>行列混合存储，配合多种行存和大规模数据查询，以及高效的点</p>
数据模型	<ul style="list-style-type: none">Aggregate keyUnique key（是Aggregate的一种特殊形式）Duplicate key <p>此外这三种还可以上卷（Rollup）成物化视图（粗粒度聚合数据，单独存储，依赖base表，目前物化视图可以单独创建，不依赖Rollup）</p>	<p>与doris基本一致。</p> <p>支持物化视图。</p> <p>都有MVCC（物化视图和明细表的Krypton支持脏读</p>

5. 罗盘接入Krypton实际效果

5.1 罗盘Krypton接入方式



5.2 接入实际效果

5.1.1 多店巡检实时模块

多店巡检实时模块5.16号完成改造并上线，以下指标数据来自稳定性看板，接入hsap系统还需要长时间持续观察。



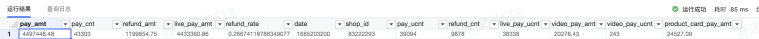
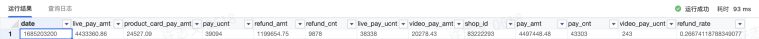
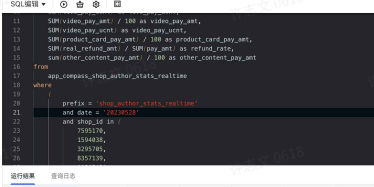


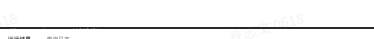
https://data.bytedance.net/aeolus#/dashboard/522081?appId=317523&fromsubpush=true&lang=zh_CN&sheetId=584700

1. 接口错误详情:

	时间	全部请求数	错误请求数
InspectionOverviewTotal	5.22 - 5.28	323,467	




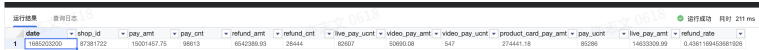
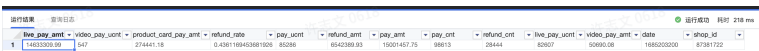
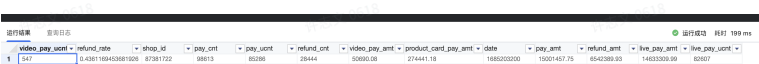
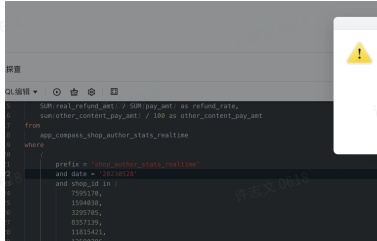
	5.8 - 5.14	278,385	1.
	5.1 - 5.7	226,454	1
	4.24- 4.30	219,492	1
	4.17 - 4.23	287,736	1
InspectionAccountDetail	5.22 - 5.28	8,781	
	5.8 - 5.14	11,169	.
	5.1 - 5.7	11,891	
	4.24- 4.30	10,220	
	4.17 - 4.23	13,742	

2. oneservice测试：365个店铺和354个账号

	Krypton	Abase
表	ads_shop_author_metrics_hour_v3	app_compass_shop_autho
查询sql	<div>1 select event_date as date, MAX(shop_ic</div>	<div>1 select date, MAX</div>
查询结果 (各查询4次)	<div>1. 125ms</div> <div></div> <div>2. 103ms</div> <div></div> <div>3. 85ms</div> <div></div> <div>4. 93ms</div> <div></div>	<div>1. 871ms</div> <div></div> <div>2. 1236ms</div> <div></div> <div>3. 1245ms</div> <div></div> <div>4. 1155ms</div> <div></div>

查询耗时	平均在101ms	平均在1127ms
结论	<div>1. 两张表得出数据结果一致，证明Krypton数据准确性</div> <div>2. Krypton平均用时是Abase表的十分之一，查询耗时降低了91.1%</div> <div>证明Krypton能够适用于策略实时模块</div> <div>可能原因：</div> <div>abase在对两个key做大in（shop in and author in）查询时，会将两个key做差乘，差乘结果重子，就会产生365*354个key，在计算层做预聚合，导致查询性能变慢；</div> <div>而krypton 会通过物化视图预计算后存储成特殊表，在查询时，直接查询物化视图，降低查询时</div>	

其他组织测试，本次测试Krypton在原附身情况下超时的组织：1210个店铺和1732个账号

	Krypton	Abase
表	ads_shop_author_metrics_hour_v3	app_compass_shop_autho
查询sql	 Krypton.sql	 Abase.sql
查询结果 (各查询4次)	<div>1. 231ms</div>  <div>2. 211ms</div>  <div>3. 218ms</div>  <div>4. 199ms</div> 	 <div>多次测试，全部超时</div>
查询耗时	平均在215ms	全部超时



多次测试，全部超时

3. Argos监控指标



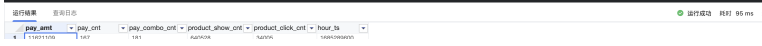

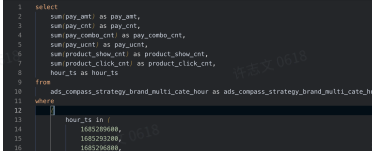

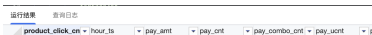
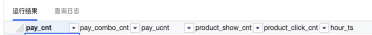


5.1.2 商品监控实时模块

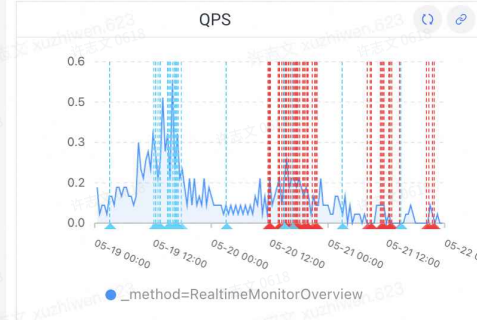
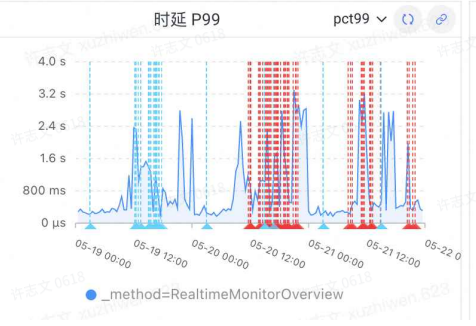
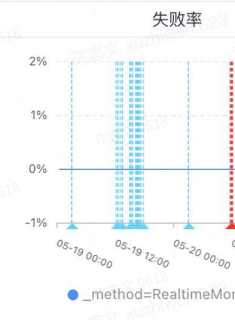
商品监控模块5.26号上线，目前正在持续观察中

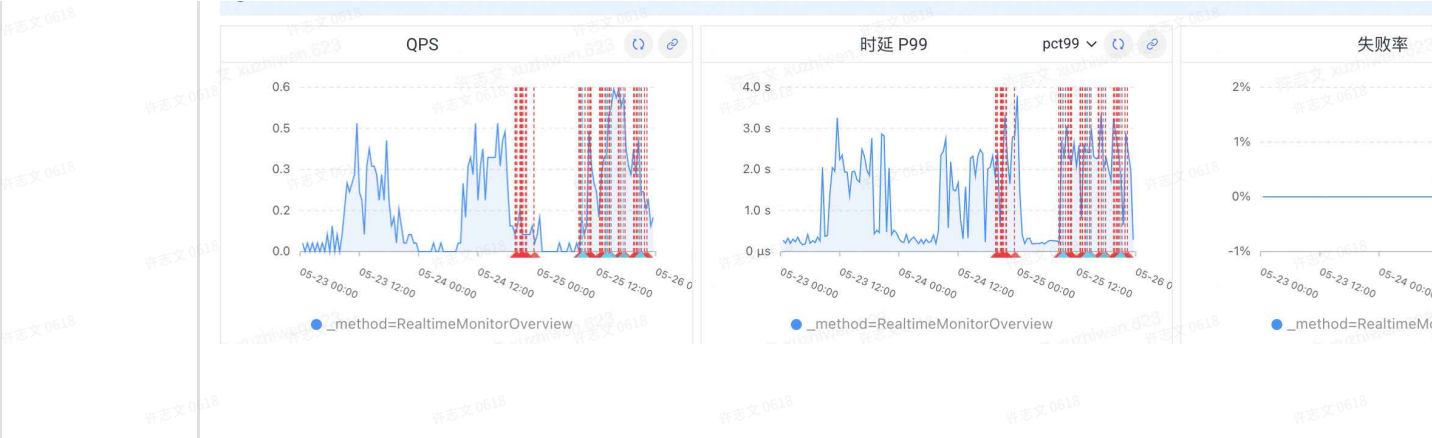
接口为strategy_product/realtime_monitor/overview?shop_group_type=1

	Krypton	Mysql
表		ads_compass_strategy_bra

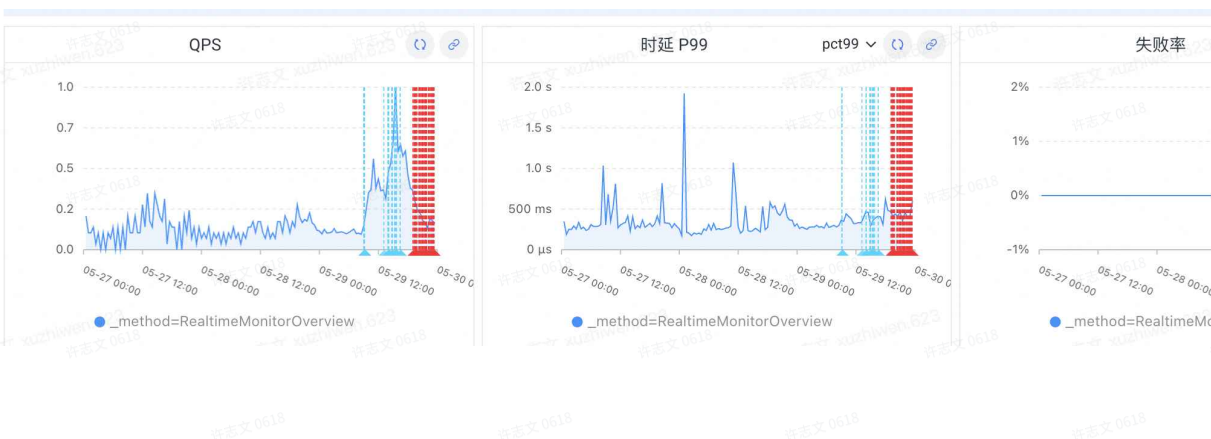
	compass_strategy_brand_product_spu_content_multi_cate_hour	
查询sql	<div><div></div>美的Krypton.sql</div>	<div><div></div>美的Mysql.sql</div>
查询结果 (各查询4次)	<div><div>1. 80ms</div><div></div><div>2. 86ms</div><div></div><div>3. 95ms</div><div></div><div>4. 92ms</div><div></div></div>	<div><div>1. 超时</div><div></div><div>2. 4140ms</div><div></div><div>3. 4207ms</div><div></div><div>4. 4189ms</div><div></div></div>
查询耗时	平均在88ms	偶现超时，平均4s多
结论	目前接入Krypton后，美的商品实时监控接口没有出现超时报错情况，查询耗时降低了98% 可以持续观察，长期看看稳定性问题	

Argos监控指标

5.19 - 5.22	<div><div></div><div></div><div></div></div>		
5.23 - 5.26			



5.27 - 5.30



5.3 接入Krypton问题记录

虽然Krypton在上述实验中能够适用于策略实时产品，但是还是需要一段时间观察其稳定性

国 电商 & Krypton Issues