Apache Doris 在国产化环境中的应用探索

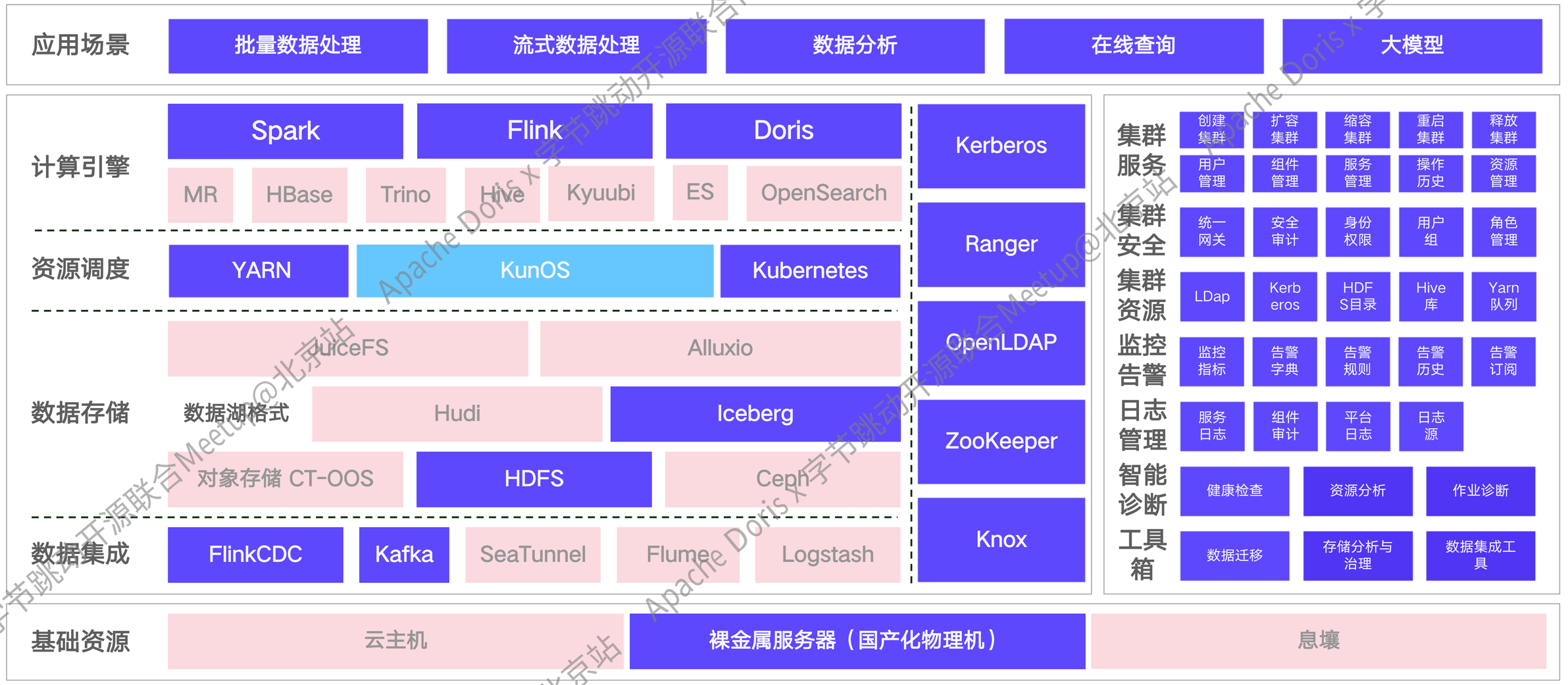
刘重阳

天翼云 研发专家

目录

天翼云-大数据产品

翼MR核心包括大数据组件和翼MR Manager管理平台两部分，在国产化基础设施（CPU、OS）上，围绕采集、存储、调度、计算等四层核心业务构建国产化大数据平台。



天翼云-公有云产品

数据湖	DataLake：数据湖分析提供了数据湖分析场景所需的服务和相关湖格式，包括 Hadoop、Hive、Spark、Trino、Flink以及Hudi、Iceberg等。在数据湖场景下，基于一湖多架构，支持离线湖、实时湖和湖仓分析等多种计算模式，满足不同的业务需求。通过采集程序，将用户的埋点数据可以基于离线或者近实时模式写入到数据湖存储中，利用 Hive 和Spark等计算引擎对原始数据进行清洗和加工，提取业务所需的指标。
数据分析	OLAP：数据分析提供了高性能、实时的分析型数据库Doris服务。采用MPP架构、预聚合技术、向量化执行引擎和劣势存储，提供极致查询性能。支持多种数据模型，通过数据建模、视图、物化视图、Rollup构建实时数仓。支持海量数据实时写入、更新、加工和分析，能够满足企业对多维报表、实时大屏、即席分析、联邦查询、人工智能和替换传统数仓数据集市场景等需求。
云搜索	AnalyticSearch：云搜索为结构化/非结构化数据提供低成本、高性能及可靠性的检索、分析服务能力。可以实现快速地对大量文本数据进行全文搜索，支持多种语言和高级搜索功能，如模糊匹配、自动完成和同义词搜索。 可以轻松完成对各类文本、数字、日期、IP地理数据等数据的高性能读写，从而快速搭建商品检索、企业文档检索、订单搜索、APP搜索等各种检索服务。 ELK：通过将Elasticsearch、Logstash和Kibana三个工具配合使用，完成了对日志的收集、传输、存储、分析和可视化展示，是一个功能强大的日志集中处理解决方案。
数据服务	DataServing：数据服务提供了HBase和HDFS等服务。通过选择HBase，将其数据存储到HDFS的湖上，实现处理大规模数据集、实时数据访问、支持自定义数据模型等能力，满足电商场景下存储网站的交易信息、物流信息、游览信息等及对实时数据处理的需求。
实时数据流	DataFlow：实时数据流提供流式计算、消息队列等服务，主要用于实时数据ETL和日志采集分析等场景。通过采集程序将业务数据、日志和埋点数据等投递到消息队列服务中，并且利用实时计算引擎功能将数据写入不同的分析系统中，以提供实时分析、点查调用和BI报表分析等操作。

天翼云 - Doris应用场景

实时数据链路

- 近实时数据写入
  - Flink-Doris-Connector微批高效写入
  - Routineload便捷接入Kafka数据
  - GroupCommit提升JDBC写入性能
- 高并发查询
  - 行列混存
  - 短路径查询
  - 行级缓存
- 报表对接
  - 多表物化视图
  - 聚合模型
  - JDBC协议，快速适配BI引擎

大数据体系

- 数据导入
  - HDFS Load、StreamLoad等多种导入方式
  - 通过Connector高效对接Flink，Spark提供更多数据接入方式
  - 多种数据模型，适应多类型业务场景
- 数据查询
  - 高并发点差。利用行列混存和短路径优化，增加点查询并发能力，单节点可达上万QPS。
  - 大宽表查询。通过向量化计算引擎和合适的索引列可提升大宽表的查询性能。
  - 复杂多表Join查询。CBO查询优化器进行RuntimeFilter动态减少数据计算量，配合大规模并行计算能力，提升数据关联查询。

小型数仓

- 数据存储
  - 行列混存。列式存储便于压缩，行式存储便于读取多字段。行列结合满足多场景。
  - 智能索引。多种类型索引可以加速查询，倒排索引的引入丰富了日志场景应用
- 数据生态
  - 数据导入。多种数据导入方式。
  - 数据共享。Mysql协议数据共享
- 易用性
  - 兼容标准SQL。
  - 轻松扩展集群。

国产化平台适配

大数据平台 翼 MapReduce 完成 CtyunOS+（鲲鹏、海光、Intel）、麒麟+（鲲鹏、Intel）、欧拉+飞腾国产化适配。

实践案例-基于鲲鹏+CTyunOS（1/4）

HISILICON

鲲鹏920

ctyunOS

ARM

HYGON 中科海光

海光

ctyunOS

X86

ctyunOS

鲲鹏

毕昇编译器

毕昇JDK

采用毕昇编译器，编译速度提升 30%

ctyunOS 集成鲲鹏处理器软件优化

实践案例-基于鲲鹏+CTyunOS（2/4）

- 以 TPC-H Q5 为例

```sql
--Q5
select
    n_name,
    sum(l_extendedprice * (1 - l_discount)) as revenue
from
    customer,
    orders,
    lineitem,
    supplier,
    nation,
    region
where
    c_custkey = o_custkey
    and l_orderkey = o_orderkey
    and l_suppkey = s_suppkey
    and c_nationkey = s_nationkey
    and s_nationkey = n_nationkey
    and n_regionkey = r_regionkey
    and r_name = 'ASIA'
    and o_orderdate >= date '1994-01-01'
    and o_orderdate < date '1994-01-01' + interval '1' year
group by
    n_name
order by
    revenue desc;
```

实践案例-基于鲲鹏+CTyunOS（3/4）

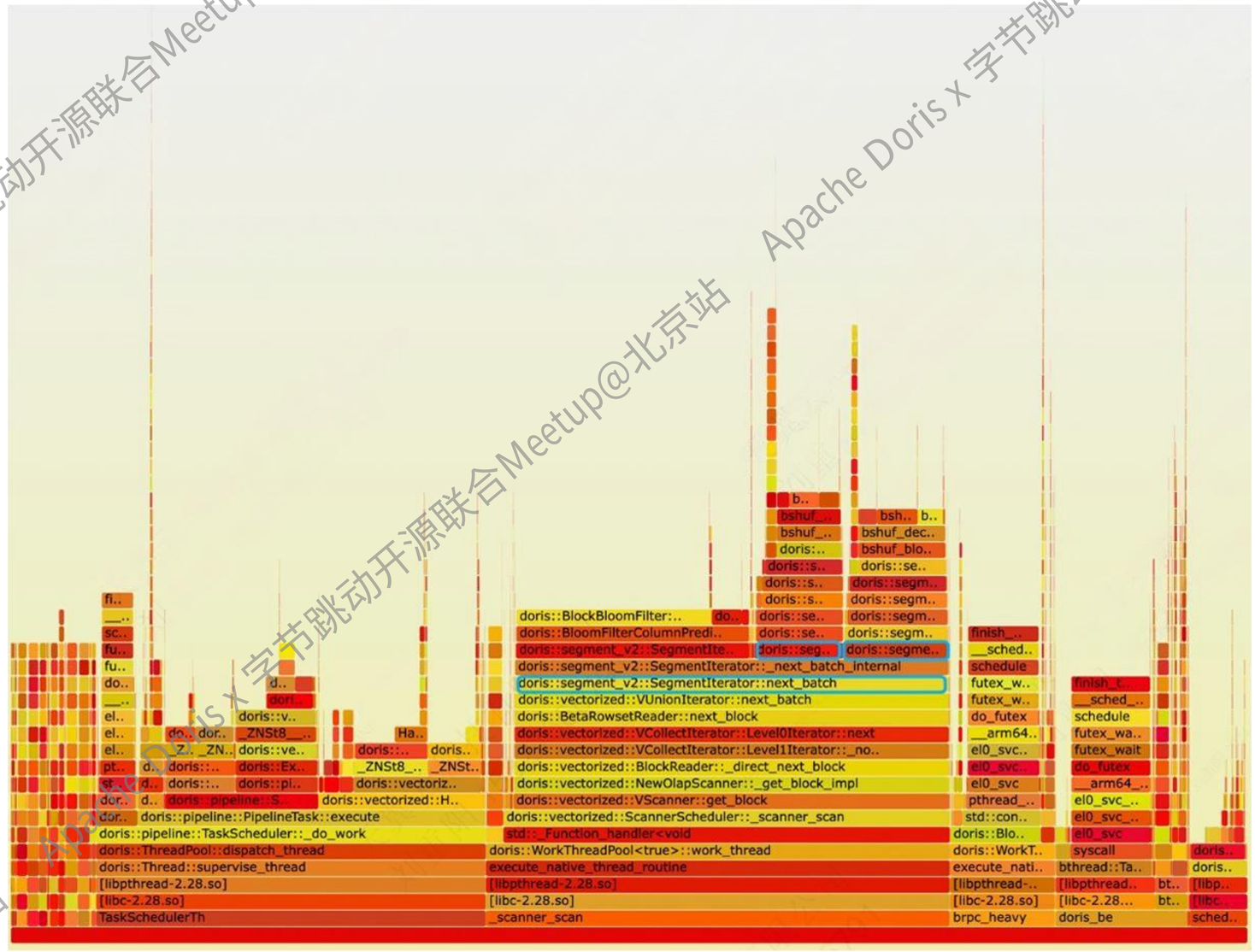
- 以 TPC-H Q5 为例

◆CRC

- Doris 中查询引擎的数据校验性能主要就是 CRC32 性能
- Arm 分支的 CRC 正常源码里会调用 cpu 的 crc32cx 汇编指令进行高效计算。

◆Bitshuffle

- bshuf_untrans_bit_elem_NEON 函数是一个向量化函数，实现比较冗长。在 Arm 架构下，这个函数的性能不是最优的。



实践案例-基于鲲鹏+CTyunOS（4/4）

◆其他优化

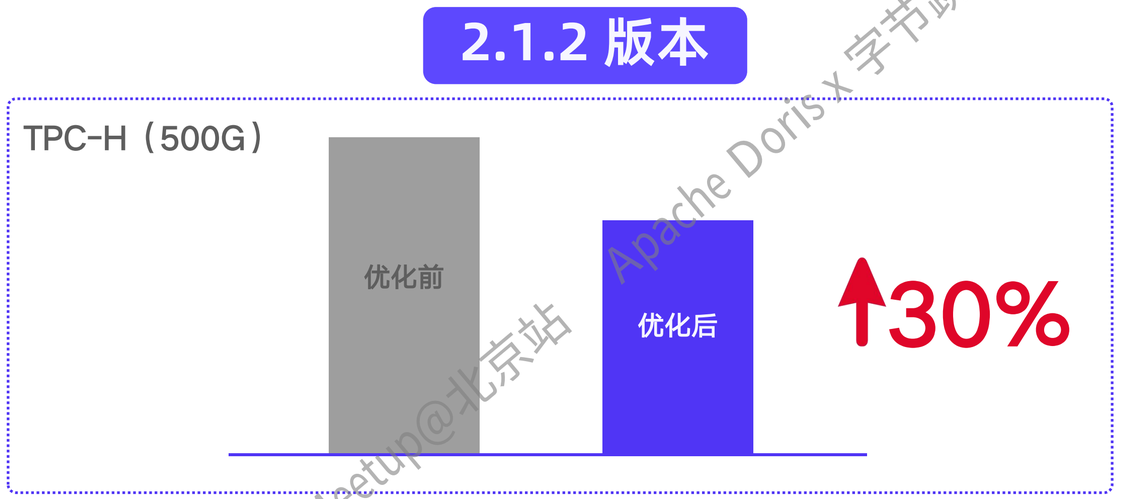
➢KAE

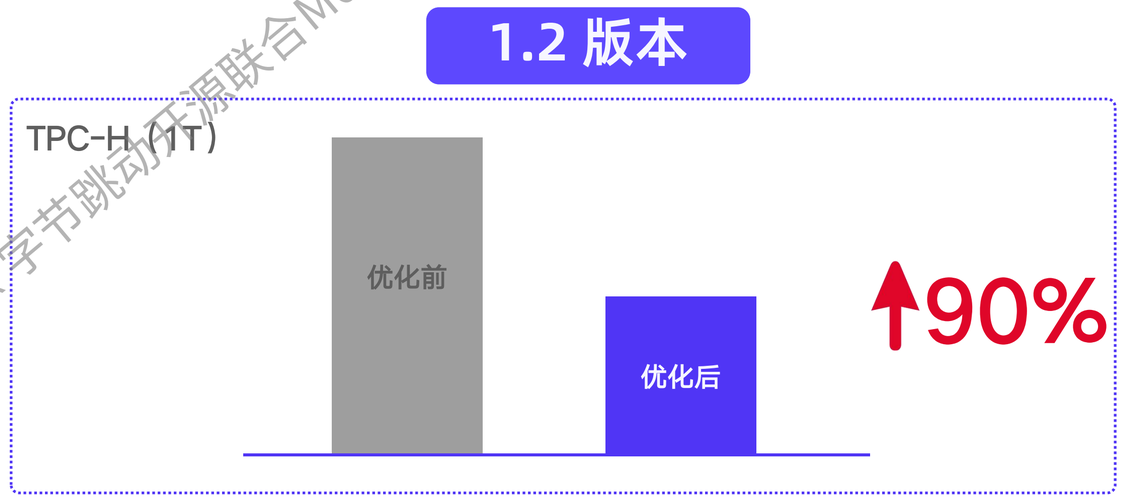
➢BIOS

整体优化后

优化版本												
第1次				第2次运行				第3次				第4次			
Time Unit: ms				Time Unit: ms				Time Unit: ms				Time Unit: ms			
q1	16216	2008	1305	q1	1462	2699	1854	q1	1447	1387	1445	q1	5855	1355	1351
q2	1586	518	477	q2	483	388	390	q2	508	376	381	q2	385	364	392
q3	2955	2031	1598	q3	2656	1658	1760	q3	2061	3551	2559	q3	2908	3109	2110
q4	556	495	477	q4	489	460	457	q4	462	455	454	q4	470	471	459
q5	1489	1458	1457	q5	1579	1584	1480	q5	1674	1639	1564	q5	1707	1695	1582
q6	151	128	137	q6	123	131	127	q6	120	117	115	q6	125	115	114
q7	978	866	879	q7	938	920	904	q7	898	900	855	q7	916	885	886
q8	809	784	780	q8	794	794	797	q8	777	782	782	q8	752	752	759
q9	5061	5082	5861	q9	6631	6263	6756	q9	6531	5986	5962	q9	6239	4822	4198
q10	1094	1025	1068	q10	1098	1038	961	q10	1203	995	1047	q10	995	998	976
q11	327	308	335	q11	290	289	293	q11	283	284	286	q11	280	284	286
q12	221	198	196	q12	187	183	193	q12	425	187	181	q12	178	178	182
q13	3490	2662	4291	q13	3998	4003	3991	q13	2962	3727	3118	q13	2641	2654	2742
q14	1635	1030	477	q14	797	319	231	q14	332	289	254	q14	398	237	232
q15	379	374	388	q15	349	351	333	q15	347	340	338	q15	335	390	361
q16	465	429	438	q16	428	431	437	q16	438	433	419	q16	428	430	434
q17	469	437	451	q17	444	438	447	q17	453	434	436	q17	445	441	429
q18	2287	2302	2300	q18	2490	2256	2180	q18	2092	2134	2397	q18	2150	2147	2078
q19	691	689	691	q19	1257	683	681	q19	706	701	690	q19	705	689	687
q20	3213	4470	2471	q20	2726	3032	3106	q20	2775	3158	3546	q20	3348	3303	2795
q21	1496	1298	1327	q21	1709	1321	1290	q21	1477	1329	1298	q21	1417	1283	1286
q22	1161	964	785	q22	1187	740	784	q22	870	708	753	q22	750	756	672
Total cold run time: 46729 ms				Total cold run time: 32418 ms				Total cold run time: 28841 ms				Total cold run time: 33325 ms			
Total hot run time: 25635 ms				Total hot run time: 28311 ms				Total hot run time: 28065 ms				Total hot run time: 24874 ms			
Finish tpch queries.				Finish tpch queries.				Finish tpch queries.				Finish tpch queries.			

整体耗时从48秒优化到28秒。

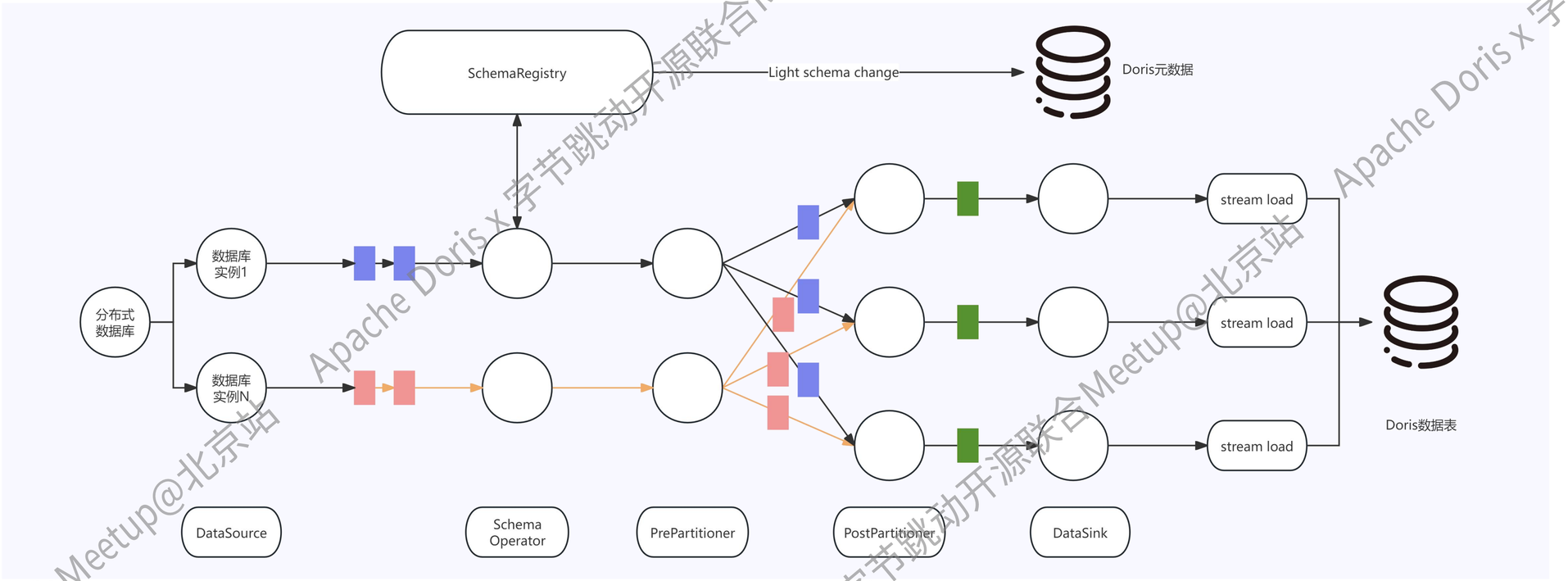




目录

实践问题1：FlinkCDC 多库多表同步(1/3)

●FlinkCDC 3.0 数据流



存在问题：

1、源表数据量比较大，对于 Doris 节点 IO 等资源消耗较大。

2、当 sink 端压力大时，CDC任务出现 Label already used 异常。

3、并行度设置源端和目标端无法分开设置，不够灵活。

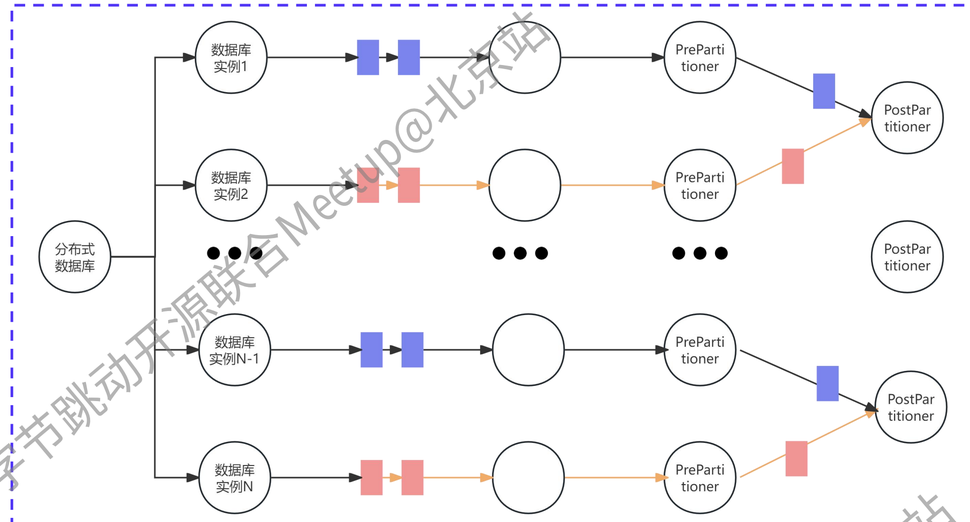
实践问题1：FlinkCDC 多库多表同步(2/3)

●Flink CDC 按分区字段进行 shuffle，降低 BE 资源消耗

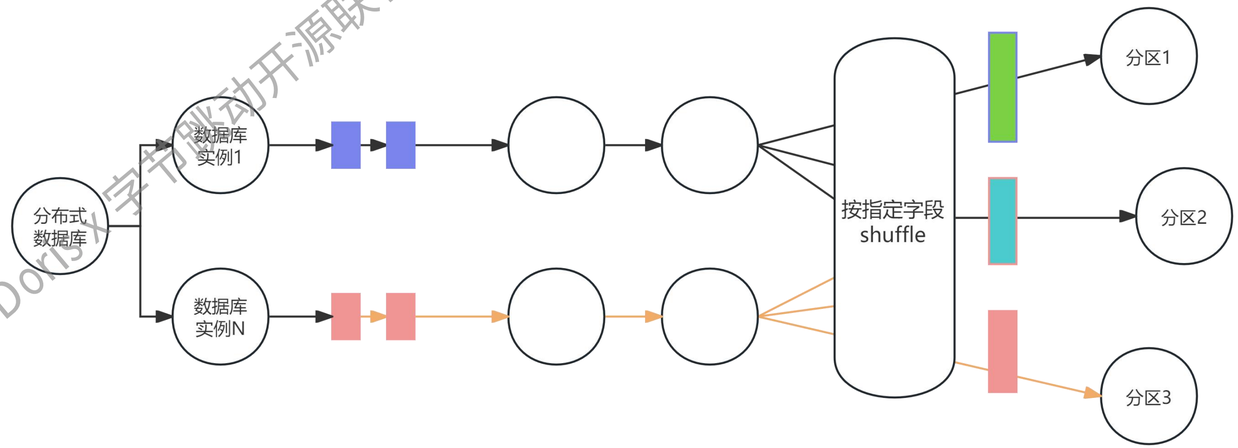
◆优化目标是降低 StreamLoad 数据分散程度。
◆方案核心是对数据进行分区，提前进行数据分组。
◆方案一：人为进行数据库实例分组，每个实例分组启动一个 CDC 数据流导入到固定分区。
◆方案二：CDC 过程中进行 shuffle，根据分区字段进行 shuffle，streamload 提交到指定分区。

本地文件导入测试

表模型	MOW分区表	MOW非分区表	MOR非分区表
数据导入文件1	14.35	16.87	13.34
数据导入文件2	12.53	14.15	12.69
数据导入文件3	10.20	12.86	13.39
数据导入文件4	10.28	11.46	8.82
数据查询	0.11	0.13	1.5
内存	28-23=5G	16G	7G
CPU	26%	40%	17%



优化方案1



优化方案2

实践问题1：FlinkCDC 多库多表同步(3/3)

●Flink-Connector 按状态进行重试，解决 Label 重复问题

Flink-Connector 会使用 HttpClient 提交 StreamLoad 请求，该方法自带重试且不可管控，查看源码未禁用重试功能，异常情况会触发。

```
public HttpClientBuilder getHttpClientBuilderForBatch() {
    return HttpClients.custom()
            .setRedirectStrategy(
                    new DefaultRedirectStrategy() {
                        @Override
                        protected boolean isRedirectable(String method) {
                            return true;
                        }
                    })
            .setDefaultRequestConfig(requestConfig);
}
```

Flink-Connector 本身会使用返回的 load 状态进行重试，但是缺少了一些异常如 connect timeout 捕获，这种需要根据 label 探查状态重试提交。

```
/** enum of LoadStatus. */
public class LoadStatus {
    public static final String SUCCESS = "Success";   madong, 2022/3/8 9:45 · [refactor]doris
    public static final String PUBLISH_TIMEOUT = "Publish Timeout";
    public static final String LABEL_ALREADY_EXIST = "Label Already Exists";
    public static final String FAIL = "Fail";
}
```

实践问题2：查询解析优化(1/2)

●查询解析优化，解决无法过滤分桶键数据问题

```
CREATE TABLE `test1` (
  `col1` BIGINT NULL,
  `col2` VARCHAR(32) NOT NULL,
  `col3` VARCHAR(32) NOT NULL,
  。。。。。。
  `colN` BIGINT NULL,
  。。。。。。
) ENGINE=OLAP
DUPLICATE KEY(`col1`, `col2`)
DISTRIBUTED BY HASH(`col1`,`col2`) BUCKETS 50
PROPERTIES (
"replication_allocation" = "tag.location.default: 1",
);
```

```
select colN from test1 where
col3=123 and col2=123 and
col1 in('1');
```

```
Explain String(Nereids Planner)

PLAN FRAGMENT 0
  OUTPUT EXPRS:
    [redacted]
  PARTITION: UNPARTITIONED

  HAS_COLO_PLAN_NODE: false

  VRESULT SINK
     MYSQL_PROTOCAL

  1:VEXCHANGE
     offset: 0
     distribute expr lists:

PLAN FRAGMENT 1

  PARTITION: HASH_PARTITIONED: [redacted]

  HAS_COLO_PLAN_NODE: false

  STREAM DATA SINK
    EXCHANGE ID: 01
    UNPARTITIONED

  0:VOlapScanNode(125)
     TABLE: testdb.[redacted]                                    PREAGGREGATION: ON
     PREDICATES: (CAST([redacted][#1] AS DOUBLE) = 9322192288) AND (CAST([redacted][#2] AS DOUBLE) = 1567710) AND (CAST([redacted][#0] AS TEXT) = '1')
     partitions=1/1 ([redacted]), tablets=50/50, tabletList=46056,46058,46060 ...
     cardinality=500000000, avgRowSize=0.0, numNodes=1
     pushAggOp=NONE
     final projections: [redacted]
     final project output tuple id: 1
```

实践问题2：查询解析优化(2/2)

●查询解析优化，解决无法过滤分桶键数据问题

◆建表以 varchar 作为分桶键

查询条件	新优化器	旧优化器
=123	触发类型转换 Double	触发类型转换 Double
="123"	不触发类型转换，结果正常	不触发类型转换，结果正常
in(123)	不触发类型转换，结果正常	不触发类型转换，结果正常
in("123")	不触发类型转换，结果正常	不触发类型转换，结果正常

◆建表以 bigint 作为分桶键

查询条件	新优化器	旧优化器
=123	不触发类型转换，结果正常	不触发类型转换，结果正常
="123"	不触发类型转换，结果正常	不触发类型转换，结果正常
in(123)	不触发类型转换，结果正常	不触发类型转换，结果正常
in("123")	触发类型转换 TEXT	不触发类型转换，结果正常

在业务查询时最好能一一进行数据类型对应，避免查询性能损失。同时查询优化器还有优化空间，比如查询计划与收集到的数据类型映射，转换时按表字段进行转换。

```
Explain String(Nereids Planner)
PLAN FRAGMENT 0
  OUTPUT EXPRS:
  PARTITION: UNPARTITIONED

  HAS_COLO_PLAN_NODE: false

  VRESULT SINK
     MYSQL_PROTOCAL

  1:VEXCHANGE
     offset: 0
     distribute expr lists:
     tuple ids: 1N

PLAN FRAGMENT 1

  PARTITION: HASH_PARTITIONED:

  HAS_COLO_PLAN_NODE: false

  STREAM DATA SINK
     EXCHANGE ID: 01
     UNPARTITIONED

  0:VOlapScanNode(125)
     TABLE: testdb.
     PREDICATES: (
     partitions=1/1 (
     cardinality=500000000, avgRowSize=0.0, numNodes=1
     pushAggOp=NONE
     final projections:
     final project output tuple id: 1
     tuple ids: 0

Tuples:
TupleDescriptor{id=0, tbl=sum_first_20240721}
  SlotDescriptor{id=0, col=
  SlotDescriptor{id=1, col=
  SlotDescriptor{id=2, col=
  SlotDescriptor{id=22, col=

TupleDescriptor{id=1, tbl=
  SlotDescriptor{id=26, col=
```

实践问题3：Decimal 类型作为分桶键(1/2)

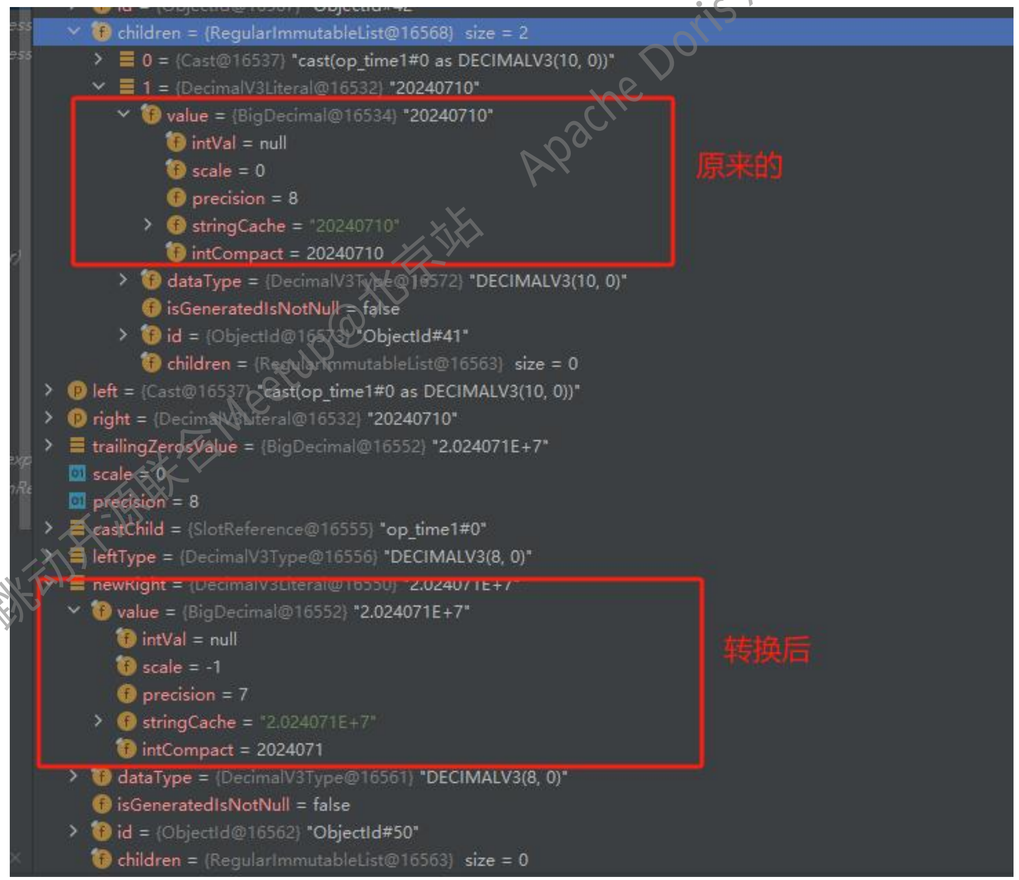
●Decimal 作为分桶键，无法查询结果问题

- 根本原因：写数据的 bucket 路由和读数据时 tablet 的过滤输入数据不一样导致的。
- 写数据时：使用的是 decimal 的原始数据来计算
- 读数据时：使用 decimal 的 intvalue 来计算

修复方式

1.修改 FE 计算 hash 时取原始值计算

2.通过修改输入的 decimal 值的精度来解决，把 scala 为负的变为正，这样 decimal 值的整数部分和原始值一样，等同于也是使用原始值计算。



实践问题3：Decimal 类型作为分桶键(2/2)

●Decimal 作为分桶键，无法查询结果问题

目前跟社区保持一致采用第二种方式修复。

[fix](Nereids) tablet prune wrong when decimal value scale is nagtive by morrySnow · Pull Request #37889 · apache/doris · GitHub

DECIMAL(8,0), DECIMAL(16,0)

before fix

```
mysql> select * from ctyun_test_db.check1 where op_time1=20240710 and user_id1=110000000000000000;
Empty set (0.06 sec)

mysql> select * from ctyun_test_db.check1 where op_time1=20240711 and user_id1=110000000000000001;
+----------+--------------------+
| op_time1 | user_id1           |
+----------+--------------------+
| 20240711 | 110000000000000001 |
+----------+--------------------+
```

after fix

```
mysql> select * from ctyun_test_db.check1 where op_time1=20240710 and user_id1=110000000000000000;
+----------+--------------------+
| op_time1 | user_id1           |
+----------+--------------------+
| 20240710 | 110000000000000000 |
+----------+--------------------+
1 row in set (0.04 sec)

mysql>
ERROR:
No query specified

mysql> select * from ctyun_test_db.check1 where op_time1=20240711 and user_id1=110000000000000001;
+----------+--------------------+
| op_time1 | user_id1           |
+----------+--------------------+
| 20240711 | 110000000000000001 |
+----------+--------------------+
1 row in set (0.04 sec)
```

DECIMAL(7,1), DECIMAL(15,1)

before fix

```
mysql>  select * from ctyun_test_db.check21 where op_time1=407000.1 and user_id1=11000000000000.1;
+----------+------------------+
| op_time1 | user_id1         |
+----------+------------------+
| 407000.1 | 11000000000000.1 |
+----------+------------------+
1 row in set (0.02 sec)

mysql>  select * from ctyun_test_db.check21 where op_time1=407000.0 and user_id1=1100000000000.0;
+----------+----------------+
| op_time1 | user_id1       |
+----------+----------------+
| 407000.0 | 1100000000000.0 |
+----------+----------------+
1 row in set (0.02 sec)

mysql> select * from ctyun_test_db.check21 where op_time1=407000 and user_id1=1100000000000;
Empty set (0.02 sec)
```

after fix

```
mysql> select * from ctyun_test_db.check21 where op_time1=407000.0 and user_id1=1100000000000.0;
+----------+----------------+
| op_time1 | user_id1       |
+----------+----------------+
| 407000.0 | 1100000000000.0 |
+----------+----------------+
1 row in set (0.04 sec)

mysql> select * from ctyun_test_db.check21 where op_time1=407000.1 and user_id1=11000000000000.1;
+----------+------------------+
| op_time1 | user_id1         |
+----------+------------------+
| 407000.1 | 11000000000000.1 |
+----------+------------------+
1 row in set (0.04 sec)

mysql> select * from ctyun_test_db.check21 where op_time1=407000  and user_id1=1100000000000;
+----------+----------------+
| op_time1 | user_id1       |
+----------+----------------+
| 407000.0 | 1100000000000.0 |
+----------+----------------+
1 row in set (0.03 sec)
```

实践问题4：实时数据流临界点Compaction(1/1)

●实时导入日分区后，查询性能问题

●问题背景：实时导入的聚合表在查询前一天数据的时候时间消耗比较大约15s。

●问题出现：准备一张一模一样的临时表，Insert into 日数据进入，查询时间2s

●问题原因：读时合并问题影响。

●解决方案：

1、每天进行前一天数据的手动合并。

2、自动触发compaction，进行合并。

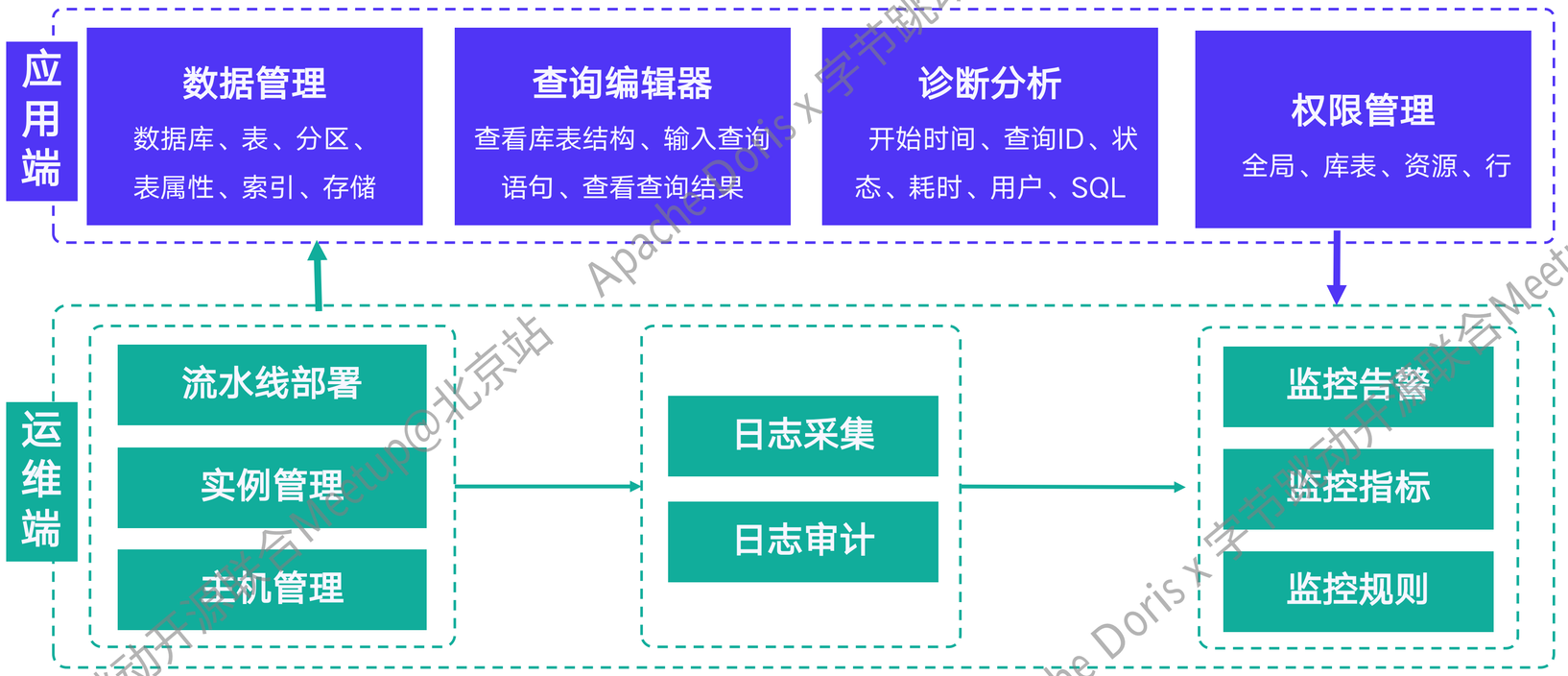

```
{
    "cumulative policy type": "size_based",
    "cumulative point": 17099,
    "last cumulative failure time": "2024-06-19 10:09:03.780",
    "last base failure time": "1970-01-01 08:00:00.000",
    "last full failure time": null,
    "last cumulative success time": "2024-06-19 10:09:03.780",
    "last base success time": "2024-06-21 11:34:32.355",
    "last full success time": "1970-01-01 08:00:00.000",
    "rowsets": [
        "[0-17098] 3 DATA NONOVERLAPPING 02000000005b7c42e145826cecd05619d6e47f3bd25490bb 655.00 MB",
        "[17099-20254] 1 DATA NONOVERLAPPING 0200000000ba235c984d64e8dcc4cf9aa446e3e61c5a729b 86.06 MB",
        "[20255-20962] 1 DATA NONOVERLAPPING 0200000000ba9aba984d64e8dcc4cf9aa446e3e61c5a729b 19.50 MB",
        "[20963-21251] 1 DATA NONOVERLAPPING 0200000000bacc60984d64e8dcc4cf9aa446e3e61c5a729b 8.08 MB",
        "[21252-21261] 1 DATA NONOVERLAPPING 0200000000bacec4984d64e8dcc4cf9aa446e3e61c5a729b 1.23 MB",
        "[21262-21271] 1 DATA NONOVERLAPPING 0200000000bad0f7984d64e8dcc4cf9aa446e3e61c5a729b 425.53 KB",
        "[21272-21291] 0 DATA NONOVERLAPPING 0200000000bad5be984d64e8dcc4cf9aa446e3e61c5a729b 0"
    ],
    "missing_rowsets": [],
    "stale_rowsets": [
        "[0-13411] 3 0200000000b5cf33984d64e8dcc4cf9aa446e3e61c5a729b 528.31 MB",
        "[13412-17098] 1 0200000000b80f64984d64e8dcc4cf9aa446e3e61c5a729b 128.05 MB"
    ],
    "stale version path": [
        {
            "path id": "1",
            "last create time": "2024-06-10 18:57:04 +0800",
            "path list": "1 -> [0-13411] -> [13412-17098]"
        }
    ]
}
```

```
{
    "cumulative policy type": "size_based",
    "cumulative point": 3,
    "last cumulative failure time": "1970-01-01 08:00:00.000",
    "last base failure time": "2024-06-12 10:15:46.462",
    "last full failure time": null,
    "last cumulative success time": "2024-06-12 09:58:49.186",
    "last base success time": "1970-01-01 08:00:00.000",
    "last full success time": "1970-01-01 08:00:00.000",
    "rowsets": [
        "[0-1] 0 DATA OVERLAP_UNKNOWN 0200000000d256df984d64e8dcc4cf9aa446e3e61c5a729b 0",
        "[2-2] 26 DATA NONOVERLAPPING 0200000000d260d8984d64e8dcc4cf9aa446e3e61c5a729b 847.24 MB"
    ],
    "missing_rowsets": [],
    "stale_rowsets": [],
    "stale version path": []
}
```

目录

集群运维与使用

通过界面化简单操作，实现集群服务的部署与管理、日志采集审计、监控告警等运维能力，满足集群运维的基本需求，同时提供元数据管理、查询编辑器、诊断分析、权限管理等能力满足数据分析人员的需求。



1.基于流水线编排技术实现自动运维部署
2.界面配置进行数据库管理、导入作业管理、权限、资源监控等集群管理能力
3.数据分析能力
4.集群日志采集、用户操作日志采集
5.集群、实例、作业级细粒度的监控告警

集群运维与使用

- 通过流水线进行监控部署与配置、检测集群健康状态
- 通过集群服务管理获取主机和集群信息
- 通过监控服务端、告警服务端、监控可视化系统进行监控采集、告警推送、大屏展示
- 通过用户中心，第一时间调用消息通知中心进行告警消息下发

指标查询

1. 支持查询角色实例级、主机级的监控指标
2. 支持指标结果的绘图操作，让用户更直观获取监控项变化

指标字典

1. 展示主机和大数据组件的监控指标
2. 支持新增、查看、编辑和删除监控指标

告警规则

1. 支持新增、查看、编辑和删除告警规则
2. 支持同步告警规则到监控服务端

告警历史

1. 支持按照集群级、角色实例级、主机级，查询历史告警内容

告警订阅

1. 新增、编辑订阅告警信息，并支持微信、邮件、短信三种方式发送告警消息

集群运维与治理：

●Tablets

➢分桶数合理性检测。小数据大分桶和大数据小分桶都会影响性能。

➢数据倾斜检测。检查最大tablet和最小tablet差值百分比，超过阈值认为存在数据倾斜。

```
mysql> show data;
+-----------------------------------------+------------+--------------+------------+
| TableName                               | Size       | ReplicaCount | RemoteSize |
+-----------------------------------------+------------+--------------+------------+
| bas[redacted]_ctyuntest                 | 0.000      | 96           | 0.000      |
| es_[redacted]                           | 0.000      | 30           | 0.000      |
| es_[redacted]                           | 3.046 GB   | 30           | 0.000      |
| es_[redacted]                           | 3.054 GB   | 30           | 0.000      |
| lab[redacted]                           | 0.000      | 2            | 0.000      |
| lxp_[redacted]                          | 50.308 MB  | 30           | 0.000      |
| mob[redacted]ctyun_test                 | 0.000      | 2520         | 0.000      |
| mob[redacted]ctyun_test1                | 0.000      | 2520         | 0.000      |
| net_[redacted]                          | 40.116 GB  | 7200         | 0.000      |
| ord[redacted]                           | 0.000      | 30           | 0.000      |
| pt_[redacted]                           | 188.643 KB | 6            | 0.000      |
| wh_[redacted]                           | 7.959 GB   | 60           | 0.000      |
| wh_[redacted]                           | 0.000      | 360          | 0.000      |
| Total                                   | 54.225 GB  | 12914        | 0.000      |
| Quota                                   | 1024.000 TB| 1073741824   |            |
| Left                                    | 1023.947 TB| 1073728910   |            |
| Transaction Quota                       | 1000       | 1000         |            |
+-----------------------------------------+------------+--------------+------------+
```

●compaction与小文件

➢存储检测。制定重点更新表检测策略，进行单副本采样检测，计算数据文件是否存在未合并状态。

```
[redacted] 18336643]# ll
total 892M
-rw-r--r-- 1 doris doris 250M Jun 10 20:02 020000000092bbc98
-rw-r--r-- 1 doris doris 262M Jun 10 20:03 020000000092bbc98
-rw-r--r-- 1 doris doris 261M Jun 10 20:03 020000000092bbc98
-rw-r--r-- 1 doris doris  17M Jun 10 20:03 020000000092bbc98
-rw-r--r-- 1 doris doris  87M Jun 10 22:57 020000000094865a8
-rw-r--r-- 1 doris doris  17M Jun 10 23:41 020000000094f0a48
-rw-r--r-- 1 doris doris 1.3M Jun 10 23:42 020000000094f37c8
```

集群易用性：

数据导入

- HDFS LOAD。页面配置化导入，进行字段配置，认证信息直接采用预制模板。
- 各类数据导入记录页面化，直接查看相关导入记录。

数据查询

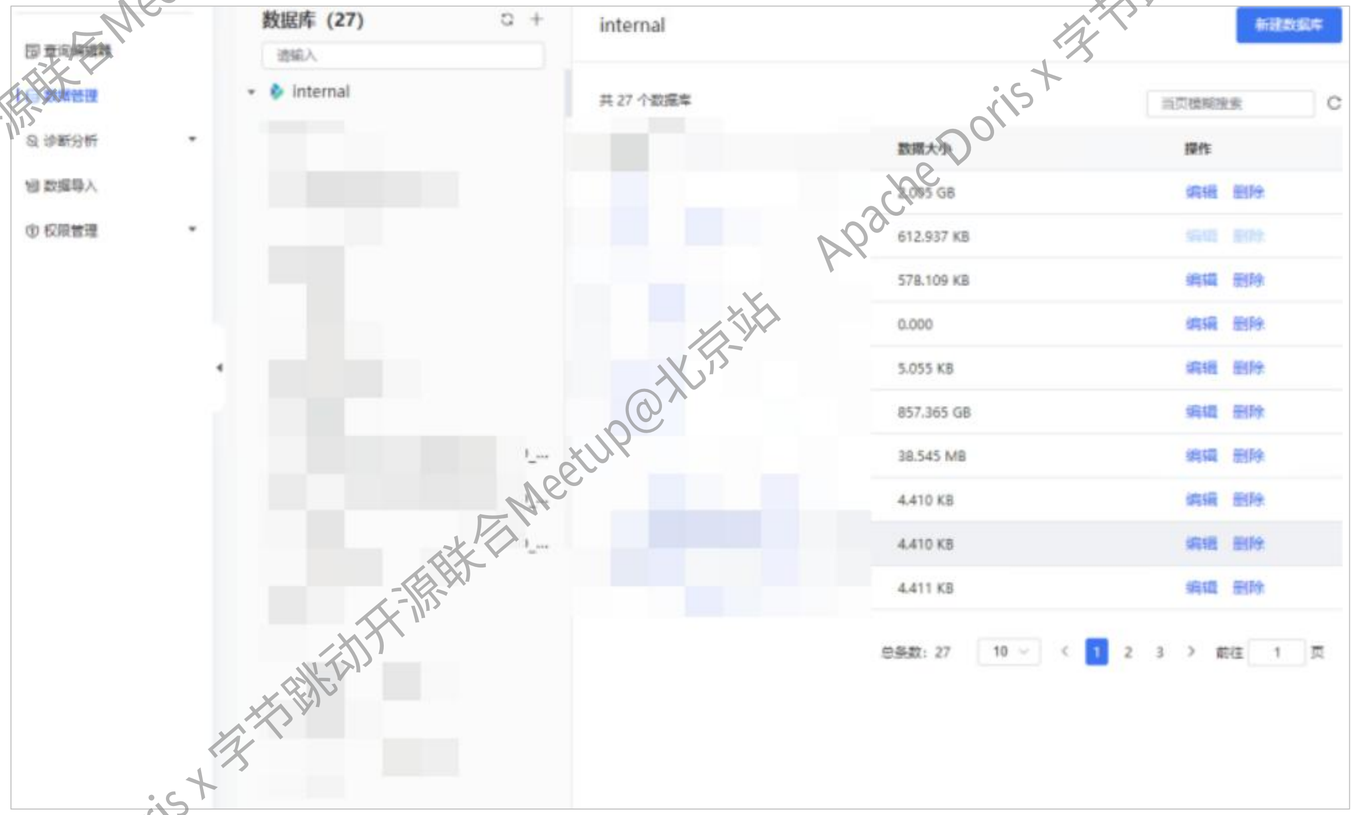
- 数据查询资源消耗可视化。目前有相应系统表，考虑工具接入。
- 数据查询资源组设置可视化和查询展示。

操作提醒

- 熔断规则可视化配置和查询。
- 常见报错与解决方案提示。

数据权限

- 多IP白名单权限统一设置管理。
- Ranger权限统一管理。



目录

未来规划

1、FlinkConnector 优化
2、Ranger 认证体系完善
3、国产化适配优化
4、参与社区开发测试任务

加大社区贡献

1、线上生产修复 SOP
2、监控告警体系完善
3、加强故障分析诊断
4、集群治理能力完善

智能运维

探索存算分离在电信体系内的应用场景和实践

存算分离

Thanks !