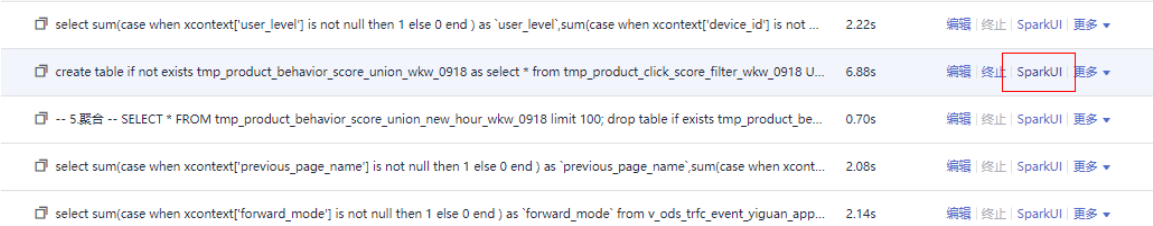
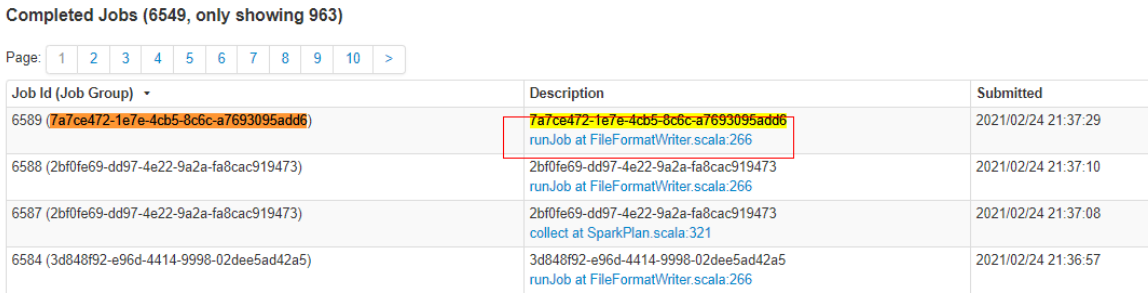
通过 SparkUI 分析

通过 DLI 作业管理打开 UI 界面

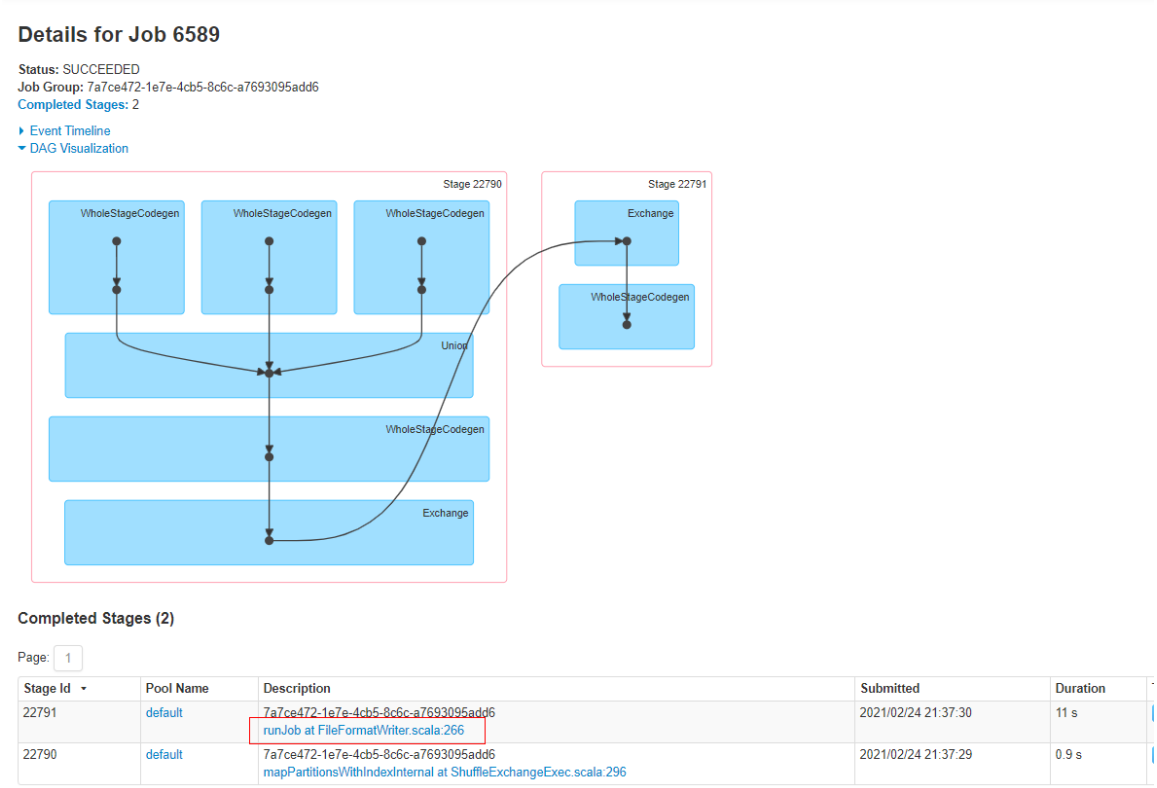


UI 界面 jobs 页签

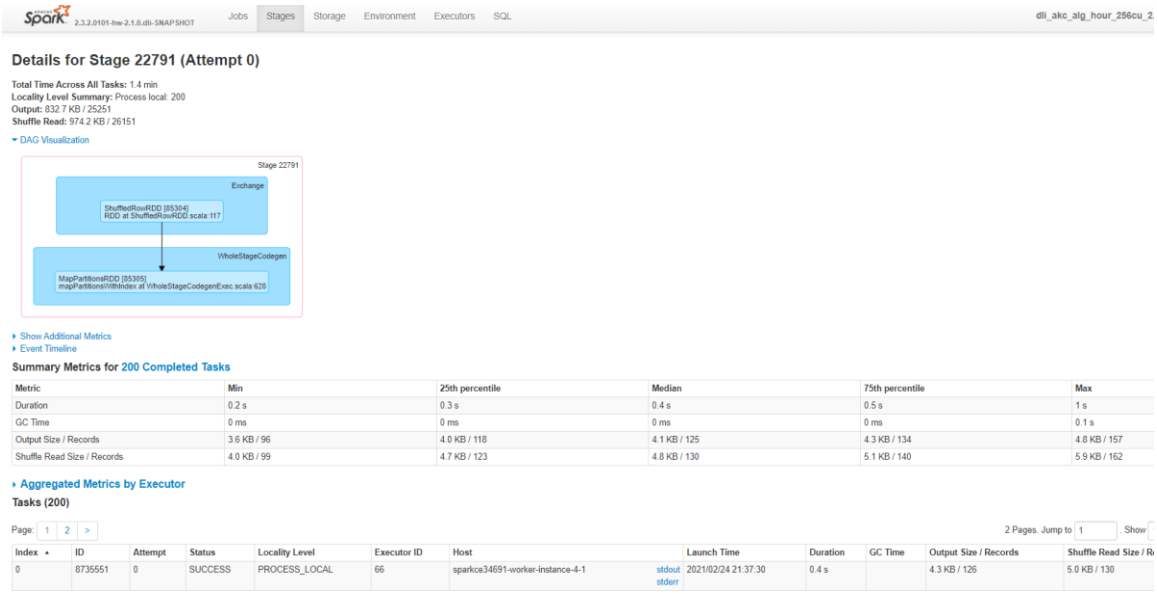
通过 DLI 作业 ID 搜索作业



点击链接可以进入 job 执行页面

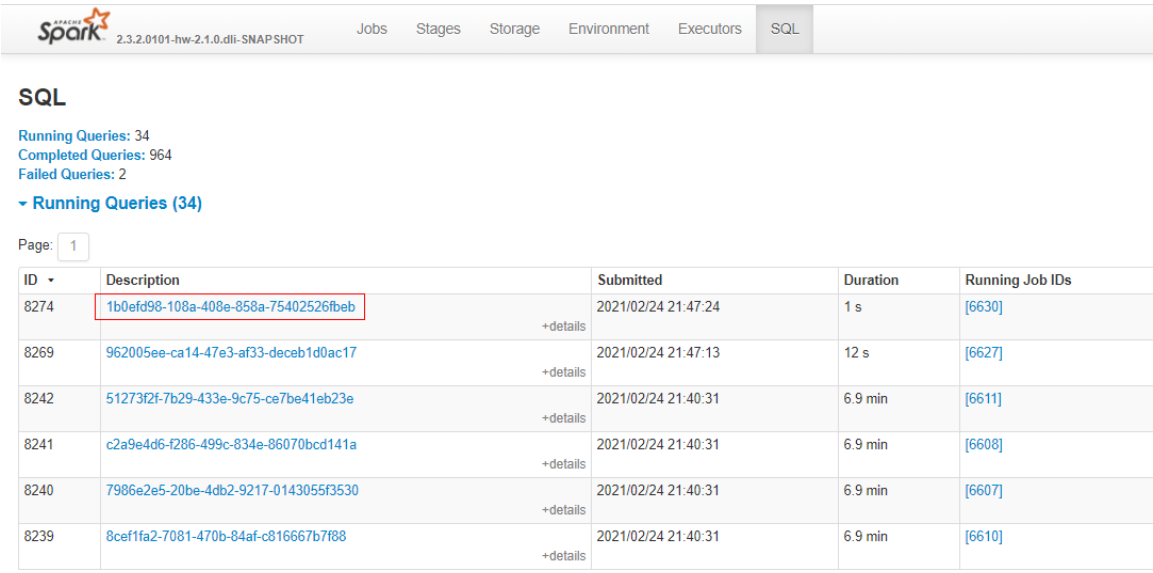


点击链接进入 stage 执行页面

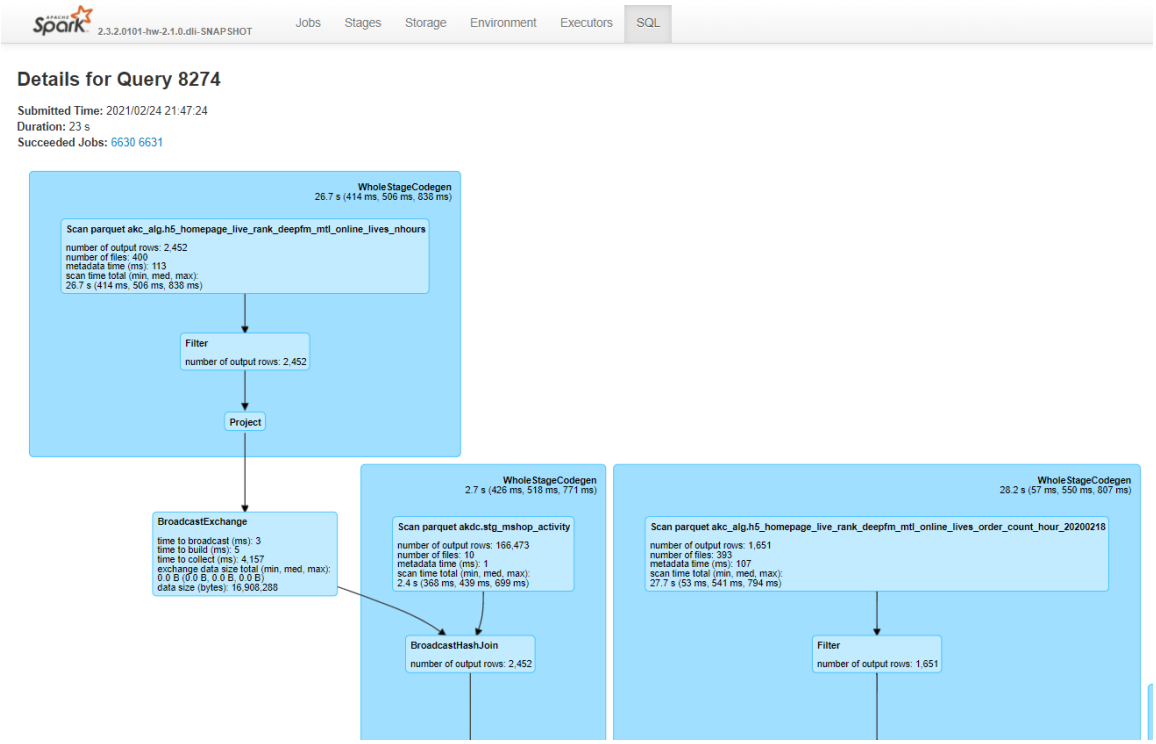


UI 界面 SQL 页签

通过 DLI 作业 ID 搜索作业



点击链接查看 SQL 执行的详细 DAG 图



SQL 性能分析

SQL1

作业 id: f499f768-cc9b-44d8-94a9-9439e6fad595

```sql
create table if not exists tmp_day_hot_cheap_goods_white_wide as
SELECT distinct
-- 商品
a1.product_id
-- ,a1.product_event_hour
,to_char(GETDATE(),"yyyy-mm-dd hh:00:00") as product_event_hour
        -- 库存
-- ,ROUND(a6.daigou_fee,1) AS daigou_fee
-- ,ROUND(a6.sale_price,1) AS sale_price
-- ,ROUND(a6.tag_price,1) AS tag_price
-- ,ROUND(a6.supply_price,1) AS supply_price

-- 用户
,a2.user

-- 场景
,a2.spm
,a2.dt
,a2.server_ts
,to_char(a2.server_ts,"yyyy-mm-dd hh:00:00") as event_hour
,a2.action_type


-- 品牌
-- ,a2.brand_id
-- ,a4.brand_index
-- ,a5.brand_level
-- ,a5.brand_score


-- 类目
-- ,a2.ctgry_three_id
-- ,cate3.cate3_index as ctgry_three_index

-- 活动
,a2.live_id
-- ,a4.online_product_count



,a2.record_num

FROM
tmp_day_cheap_goods a1
left JOIN
(
    SELECT
    o1.dt,
```

```
    MIN(case when to_char(o1.client_ts,"yyyymmdd") = o1.dt THEN o1.clie
nt_ts ELSE o1.server_ts END) as server_ts,
    o1.spm,
    s1.new_product_id as product,
    cast(o1.user_id as string) as user,
    s3.external_code as live_id ,
    o1.action_type,
    count(1) as record_num

    FROM akcbi.ods_ubt_event o1
    JOIN akdc.stg_mshop_product s2  on REGEXP_EXTRACT(o1.properties, '"
name":"productNo","value":"([0-9]+a*)","type")', 1)  = cast(s2.no as s
tring)
    JOIN akdc.dim_product_info  s1 on s1.product_id = s2.external_produ
ct_code
    JOIN akdc.stg_mshop_activity s3 on REGEXP_EXTRACT(o1.properties, '"
name":"activityNo","value":"([0-9]+a*)","type")', 1) = cast(s3.no as s
tring)
    WHERE o1.dt = to_char(getdate(),"yyyymmdd")
    and datediff1(GETDATE(), server_ts,"hh") <= 4
    -- and o1.spm in (   '30.38.11.12.21'
    --                     '30.30.20.21',
    --                     '30.17.20.21',
    --                     '30.28.52.21','30.28.828.48')
    GROUP BY o1.dt,o1.spm,s1.new_product_id,cast(o1.user_id as string),
s3.external_code,o1.action_type
) a2
ON cast(a1.product_id as string) = cast(a2.product as string)
-- LEFT JOIN akdc.dim_product_info a3 ON a2.product_id = a3.product_id
-- left join akdc.stg_mshop_activity a4 on a2.live_id=a4.external_code
```

Summary Metrics for 200 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	7 s	10 s	11 s	13 s	3.3 h
GC Time	51 ms	0.1 s	0.2 s	0.2 s	1.9 min
Shuffle Read Size / Records	41.4 MB / 262688	56.4 MB / 330150	63.9 MB / 366475	71.5 MB / 402443	15.7 GB / 114208300
Shuffle Write Size / Records	33.2 MB / 170545	46.5 MB / 239554	52.9 MB / 275998	59.9 MB / 311582	86.0 MB / 447235
Shuffle spill (memory)	0.0 B	0.0 B	0.0 B	0.0 B	94.8 GB
Shuffle spill (disk)	0.0 B	0.0 B		0.0 B	15.6 GB

Total shuffle bytes and records read (includes both data read locally and data read from remote executors).

Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Shuffle Read Size / Records	Shuffle Write Size / Records	Shuffle Spill (Memory)	Shuffle Spill (Disk)	Blacklisted	Logs
186	sparkce34046-worker-instance-26-1:22757	3.3 h	2	0	0	2	15.8 GB / 114569547	122.1 MB / 624123	94.8 GB	15.6 GB	false	stdout stderr
401	sparkce34046-worker-instance-27-1:22899	1.8 min	9	0	0	9	565.3 MB / 3236193	468.4 MB / 2421231	0.0 B	0.0 B	false	stdout stderr
298	sparkce34046-worker-instance-4-1:22898	1.4 min	8	0	0	8	493.9 MB / 2838887	409.1 MB / 2116220	0.0 B	0.0 B	false	stdout stderr

SQL 慢的原因：数据倾斜

SQL2

作业 id: 5854d14b-41da-4daf-b49f-b23f41691122

```
create table if not exists h5_capsule_products_rec_svd_channel_hourly_u
ser_link_product_recommend as
select
concat("recsys:productid:h5_capsule_pic_bargain_als:","",cast(user_id a
s string)) as buyer_id
,concat(SPLIT_PART(SPLIT_PART(recommendations,',',1),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',2),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',3),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',4),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',5),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',6),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',7),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',8),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',9),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',10),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',11),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',12),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',13),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',14),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',15),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',16),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',17),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',18),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',19),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',20),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',21),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',22),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',23),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',24),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',25),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',26),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',27),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',28),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',29),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',30),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',31),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',32),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',33),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',34),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',35),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',36),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',37),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',38),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',39),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',40),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',41),':',1),','
```

```
        ,SPLIT_PART(SPLIT_PART(recommendations,',',42),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',43),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',44),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',45),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',46),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',47),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',48),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',49),':',1),','
        ,SPLIT_PART(SPLIT_PART(recommendations,',',50),':',1)
```

……

```
    ) as recommend_product_list
from h5_capsule_products_rec_svd_channel_hourly_user_topk_products
```

split 计算慢，需要修改 SQL

单个 task 执行 40 分钟

Page: 1 2 3 > 3 Pages. Jump to 1 . Show 100 items in a page. Go

Index	ID	Attempt	Status	Locality Level	Executor ID	Host		Launch Time	Duration	GC Time	Input Size / Records	Output Size / Records	Errors
0	3801966	0	SUCCESS	PROCESS_LOCAL	47	sparkce34046-worker-instance-28-1	stdout stderr	2021/01/31 13:03:00	39 min	33 s	128.4 MB / 54534	39.2 MB / 54534	
1	3801967	0	SUCCESS	PROCESS_LOCAL	10	sparkce34046-worker-instance-17-1	stdout stderr	2021/01/31 13:03:00	39 min	44 s	128.4 MB / 54597	40.9 MB / 54597	
2	3801968	0	SUCCESS	PROCESS_LOCAL	124	sparkce34046-worker-instance-6-1	stdout stderr	2021/01/31 13:03:00	40 min	38 s	128.4 MB / 54548	39.3 MB / 54548	
3	3801969	0	SUCCESS	PROCESS_LOCAL	21	sparkce34046-worker-instance-12-1	stdout stderr	2021/01/31 13:03:00	39 min	38 s	128.3 MB / 54222	39.3 MB / 54222	
4	3801970	0	SUCCESS	PROCESS_LOCAL	6	sparkce34046-worker-instance-30-1	stdout stderr	2021/01/31 13:03:01	40 min	43 s	128.4 MB / 55591	38.2 MB / 55591	
5	3801971	0	SUCCESS	PROCESS_LOCAL	61	sparkce34046-worker-instance-24-1	stdout stderr	2021/01/31 13:03:01	38 min	36 s	128.5 MB / 54532	40.9 MB / 54532	

Page: 1 2 3 4 5 6 7 8 9 > 9 Pages. Jump to 1 . Show 100 items in a page. Go

Index	ID	Attempt	Status	Locality Level	Executor ID	Host		Launch Time	Duration	GC Time	Input Size / Records	Output Size / Records	Errors
615	2772152	0	SUCCESS	PROCESS_LOCAL	8	sparkce34253-worker-instance-9-1	stdout stderr	2021/01/31 17:41:22	25 min	2.7 min	52.8 MB / 22558	16.2 MB / 22558	
758	2772295	0	SUCCESS	PROCESS_LOCAL	81	sparkce34253-worker-instance-16-1	stdout stderr	2021/01/31 17:41:22	25 min	3.7 min	51.8 MB / 22486	15.6 MB / 22486	
562	2772099	0	SUCCESS	PROCESS_LOCAL	30	sparkce34253-worker-instance-16-1	stdout stderr	2021/01/31 17:41:22	25 min	3.1 min	53.0 MB / 22599	16.1 MB / 22599	
613	2772150	0	SUCCESS	PROCESS_LOCAL	143	sparkce34253-worker-instance-2-1	stdout stderr	2021/01/31 17:41:22	25 min	2.5 min	52.8 MB / 22499	16.0 MB / 22499	

修改方式：

```
select
  concat(
    "recsys:liveid:h5_searchnull_a:",
    "",
    cast(user_id as string)
  ) as buyer_id_a,
  concat(
    "recsys:liveid:h5_searchnull_b:",
    "",
```

```
    cast(user_id as string)

  ) as buyer_id_b,

 REGEXP_REPLACE(recommendations, ':(\\d+\\.?\\d*)', '') as recommend_li
ve_list

from

  h5_searchnull_live_recommend_footprint_hourly_user_near_live_topk
```

每一行记录做一次字符串格式化。

SQL3

问题：执行时间超过 4 小时

DLI 作业 id: ed7c81f0-2cae-46e9-83b5-97aedb6d4c21

DAYU 作业：

根因：大表小表执行 SortMergerJoin，执行速度慢

解决：spark.sql.autoBroadcastJoinThreshold=262144000

```
create table if not exists h5_livepage_product_ranking_online_users_exp
ose_products_30days as
select t1.server_date,t1.buyer_id,count(distinct t4.external_product_co
de) as expose_product_count_30days,count(t4.external_product_code) as e
xpose_product_record_count_30days
from akc_alg.h5_livepage_product_ranking_online_buyers_ndays t1
left join akdc.stg_mshop_member_member t2 on t1.buyer_id=t2.user_id
left join akdc.ods_ubt_event t3 on t2.open_id=t3.user_id
left join akdc.stg_mshop_product t4 on REGEXP_EXTRACT(t3.properties, '"
name":"productNo","value":"(.*?)(","type")', 1)=cast(t4.no as string)
where datediff1(to_date1(t1.server_date, 'yyyymmdd'),to_date1(t3.dt, 'y
yyymmdd'), 'dd')<31 and datediff1(to_date1(t1.server_date, 'yyyymmdd'),
to_date1(t3.dt, 'yyyymmdd'),
'dd')>0 and t3.dt>to_char(dateadd(getdate(), -31, 'dd'),'yyyymmdd')
and replace(t3.spm,'.0','') in ('30.17.20.21')
and t3.action_type='expose'
and REGEXP_EXTRACT(t3.properties, '"name":"productNo","value":"(.*?)(",
"type")',
1)<>'' and t2.open_id is not
null and t4.external_product_code is not
null group by t1.server_date,t1.buyer_id
```

SQL4

问题：执行时间超过 4 小时

DLI 作业 ID：49507b3d-ac45-4870-979f-c6cd8c08e63d

DAYU 作业：

根因：大表小表执行 SortMergerJoin，执行速度慢

解决：spark.sql.autoBroadcastJoinThreshold=262144000

spark.sql.statistics.fallBackToHdfs=true

```
create table if not exists h5_livepage_product_rank_for_list_online_use
rs_expose_products_30days as
select t1.server_date,t1.buyer_id,count(distinct t4.external_product_co
de) as expose_product_count_30days,count(t4.external_product_code) as e
xpose_product_record_count_30days
from akc_alg.h5_livepage_product_rank_for_list_online_buyers_ndays t1
left join akdc.stg_mshop_member_member t2 on t1.buyer_id=t2.user_id
left join akdc.ods_ubt_event t3 on t2.open_id=t3.user_id
left join akdc.stg_mshop_product t4 on REGEXP_EXTRACT(t3.properties, '"
name":"productNo","value":"(.*?)(","type")', 1)=cast(t4.no as string)
where datediff1(to_date1(t1.server_date, 'yyyymmdd'),to_date1(t3.dt, 'y
yyymmdd'), 'dd')<31 and datediff1(to_date1(t1.server_date, 'yyyymmdd'),
to_date1(t3.dt, 'yyyymmdd'), 'dd')>0
and t3.dt>to_char(dateadd(getdate(), -31, 'dd'),'yyyymmdd')
and replace(t3.spm,'.0','') in ('30.17.20.21')
and t3.action_type='expose'
and REGEXP_EXTRACT(t3.properties, '"name":"productNo","value":"(.*?)(",
"type")', 1)<>''
and t2.open_id is not null
and t4.external_product_code is not null
group by t1.server_date,t1.buyer_id
```

SQL5

问题：运行失败，executor 内存超出 20GB, sortmerge join 慢

DLI 作业 ID：83b13cf9-302e-4427-8cdf-16b8d388b7f2

DAYU 作业：app*store*healthy*degree*offline*score*for*leading*flow*20210121*v1

```
create table if not exists app_store_healthy_degree_offline_have_uv_in_
1month as
select t3.user_id
,count(distinct t1.buyer_id) as uv_count
from akdc.edw_trfc_traffic_detail_di t1
```

```
left join akdc.dim_seller_shop_h5_info t2 on t1.shop_id=t2.h5_shop_id
left join akdc.dim_user_info_akucun_app t3 on t2.user_code=t3.user_code
where datediff1(to_date1(to_char(dateadd(getdate(), -7, 'dd'), 'yyyymmd
d'), 'yyyymmdd'),to_date1(t1.dt, 'yyyymmdd'), 'dd')>0
and datediff1(to_date1(to_char(dateadd(getdate(), -7, 'dd'), 'yyyymmdd
'), 'yyyymmdd'),to_date1(t1.dt, 'yyyymmdd'),
-- and is_success=1
'dd')<=31 group by
t3.user_id
```

参数：

```
-- 无统计信息时，从 hdfs 拿参数
spark.sql.statistics.fallBackToHdfs=true;

-- 大需要调大，减少 task 数量
spark.sql.files.maxPartitionBytes=1024*1024*1024;
-- 调大 broadcast 参数
spark.sql.autoBroadcastJoinThreshold=200*1024*1024;
```

SQL6

作业 ID:25245e98-39dc-4dc0-ac0f-a552a75844ec

JOB ID：h5*invitegift*recommend*total*20210121*v1*
*h5*invitegift*recommend*coupon*post*1*20210121*v1

```
create table if not exists h5_invitegift_recommend_user_top_cluster_sty
le_no_1 as
select t1.buyer_id,t1.cluster_index,t1.predict_score,t1.predict_score_r
ank,t3.style_no_code
,t2.factors_1*t3.factors_1+t2.factors_2*t3.factors_2+t2.factors_3*t3.fa
ctors_3+t2.factors_4*t3.factors_4+t2.factors_5*t3.factors_5+t2.factors_
6*t3.factors_6+t2.factors_7*t3.factors_7+t2.factors_8*t3.factors_8+t2.f
actors_9*t3.factors_9+t2.factors_10*t3.factors_10+t2.factors_11*t3.fact
ors_11+t2.factors_12*t3.factors_12+t2.factors_13*t3.factors_13+t2.facto
rs_14*t3.factors_14+t2.factors_15*t3.factors_15+t2.factors_16*t3.factor
s_16+t2.factors_17*t3.factors_17+t2.factors_18*t3.factors_18+t2.factors
_19*t3.factors_19+t2.factors_20*t3.factors_20+t2.factors_21*t3.factors_
21+t2.factors_22*t3.factors_22+t2.factors_23*t3.factors_23+t2.factors_2
4*t3.factors_24+t2.factors_25*t3.factors_25+t2.factors_26*t3.factors_26
+t2.factors_27*t3.factors_27+t2.factors_28*t3.factors_28+t2.factors_29*
t3.factors_29+t2.factors_30*t3.factors_30+t2.factors_31*t3.factors_31+t
2.factors_32*t3.factors_32+t2.factors_33*t3.factors_33+t2.factors_34*t
3.factors_34+t2.factors_35*t3.factors_35+t2.factors_36*t3.factors_36+t
2.factors_37*t3.factors_37+t2.factors_38*t3.factors_38+t2.factors_39*t
```

```
3.factors_39+t2.factors_40*t3.factors_40+t2.factors_41*t3.factors_41+t
2.factors_42*t3.factors_42+t2.factors_43*t3.factors_43+t2.factors_44*t
3.factors_44+t2.factors_45*t3.factors_45+t2.factors_46*t3.factors_46+t
2.factors_47*t3.factors_47+t2.factors_48*t3.factors_48+t2.factors_49*t
3.factors_49+t2.factors_50*t3.factors_50+t2.factors_51*t3.factors_51+t
2.factors_52*t3.factors_52+t2.factors_53*t3.factors_53+t2.factors_54*t
3.factors_54+t2.factors_55*t3.factors_55+t2.factors_56*t3.factors_56+t
2.factors_57*t3.factors_57+t2.factors_58*t3.factors_58+t2.factors_59*t
3.factors_59+t2.factors_60*t3.factors_60+t2.factors_61*t3.factors_61+t
2.factors_62*t3.factors_62+t2.factors_63*t3.factors_63+t2.factors_64*t
3.factors_64+t2.factors_65*t3.factors_65+t2.factors_66*t3.factors_66+t
2.factors_67*t3.factors_67+t2.factors_68*t3.factors_68+t2.factors_69*t
3.factors_69+t2.factors_70*t3.factors_70+t2.factors_71*t3.factors_71+t
2.factors_72*t3.factors_72+t2.factors_73*t3.factors_73+t2.factors_74*t
3.factors_74+t2.factors_75*t3.factors_75+t2.factors_76*t3.factors_76+t
2.factors_77*t3.factors_77+t2.factors_78*t3.factors_78+t2.factors_79*t
3.factors_79+t2.factors_80*t3.factors_80+t2.factors_81*t3.factors_81+t
2.factors_82*t3.factors_82+t2.factors_83*t3.factors_83+t2.factors_84*t
3.factors_84+t2.factors_85*t3.factors_85+t2.factors_86*t3.factors_86+t
2.factors_87*t3.factors_87+t2.factors_88*t3.factors_88+t2.factors_89*t
3.factors_89+t2.factors_90*t3.factors_90+t2.factors_91*t3.factors_91+t
2.factors_92*t3.factors_92+t2.factors_93*t3.factors_93+t2.factors_94*t
3.factors_94+t2.factors_95*t3.factors_95+t2.factors_96*t3.factors_96+t
2.factors_97*t3.factors_97+t2.factors_98*t3.factors_98+t2.factors_99*t
3.factors_99+t2.factors_100*t3.factors_100 as product_predict_score
from h5_invitegift_recommend_user_cluster_distance_1_cluster_top3 t1
left join h5_invitegift_recommend_user_factors_res t2 on t1.buyer_id=t
2.buyer_id
left join (select * from (
select t1.*
,rank() over ( partition by t1.cluster_index order by t1.distance) as d
istance_tocenter_rank
from h5_invitegift_recommend_idxTable_1 t1 ) where distance_tocenter_ra
nk<=2000) t3 on t1.cluster_index=t3.cluster_index
```

存在数据倾斜，开启 AE join。

SQL7

h5*homepage*liver*rank*deepfm*total*20210117*v1*
*h5*homepage*live*rank*deepfm*MTL*online*buyers*click*20210117_v1

7055a1ac-6c6a-43aa-b6d0-945c12291323

```
SELECT
a1.buyer_id as user_id
,a1.server_date as user_event_date
,a2.dt
,a2.server_ts
,to_char(a2.server_ts,"yyyymmdd") as click_date
```

```
,provincemap.province_index as click_province_index
,citymap.city_index as click_city_index
,ctgry1.product_ctgry_one_index as ctgry_one_index
,ctgry2.product_ctgry_two_index as ctgry_two_index
,ctgry3.product_ctgry_three_index as ctgry_three_index
,a2.spm
,a2.product_id
,a4.brand_index
,a5.brand_level as brand_level
,a6.daigou_fee
,a6.sale_price
FROM
h5_homepage_live_rank_deepfm_MTL_online_buyers_ndays a1
JOIN
h5_homepage_live_rank_deepfm_MTL_online_user_click_list_20200221_prepar
e a2
ON a1.buyer_id = a2.user_id
AND datediff1(to_date1(a1.server_date,"yyyymmdd"), to_date1(to_char(a2.
server_ts,"yyyymmdd"),"yyyymmdd"),"dd") >
0 AND datediff1(to_date1(a1.server_date,"yyyymmdd"), to_date1(to_char(a
2.server_ts,"yyyymmdd"),"yyyymmdd"),"dd")
<=30 LEFT JOIN akdc.dim_product_info a3 ON a2.product_id = a3.product_i
d
LEFT JOIN akc_alg_dev.h5_homepage_live_rank_deepfm_MTL_ctg_mapping_fixe
d_with_model_brand_name a4 ON a3.brand_name = a4.brand_name
LEFT JOIN (SELECT max(brand_level) as brand_level,brand_name FROM akdc.
dim_brand_info GROUP BY brand_name) a5 ON a5.brand_name = a3.brand_name
LEFT JOIN akc_alg_dev.h5_homepage_live_rank_deepfm_MTL_ctg_mapping_fixe
d_with_model_product_ctgry_one ctgry1 ON a3.ctgry_one_name = ctgry1.ctg
ry_one_name
LEFT JOIN akc_alg_dev.h5_homepage_live_rank_deepfm_MTL_ctg_mapping_fixe
d_with_model_product_ctgry_two ctgry2 ON a3.ctgry_two_name = ctgry2.ctg
ry_two_name
LEFT JOIN akc_alg_dev.h5_homepage_live_rank_deepfm_MTL_ctg_mapping_fixe
d_with_model_product_ctgry_three ctgry3 ON a3.ctgry_three_name = ctgry
3.ctgry_three_name
LEFT JOIN akc_alg_dev.h5_homepage_live_rank_deepfm_MTL_ctg_mapping_fixe
d_with_model_user_province provincemap ON a2.click_province = provincem
ap.province
LEFT JOIN akc_alg_dev.h5_homepage_live_rank_deepfm_MTL_ctg_mapping_fixe
d_with_model_user_city citymap ON a2.click_city = citymap.city
LEFT
JOIN
(SELECT  cast(product_id as string) as product_id,AVG(daigou_fee) as da
igou_fee,AVG(sale_price) as
sale_price
FROM  akdc.stg_mer_inventory_mer_inventory
GROUP BY product_id)
a6 ON a3.new_product_id = a6.product_id
```

SQL8

h5*search*rank*job syj*online*h5*search*rank*user_click

efc19d07-8ae3-43fa-8b9b-92e46aa82dcd

```
--[notSupport]dli not support this grammar
--set odps.stage.mapper.split.size= 50;
CREATE TABLE syj_online_h5_search_rank_user_click_agg_20201103 as
SELECT
agg_func.user_id,
agg_func.user_event_date,
day1_user_productpageview_expose,
day2_user_productpageview_expose,
day7_user_productpageview_expose,
day15_user_productpageview_expose,
day30_user_productpageview_expose,
day2_user_click_count,
day7_user_click_count,
day7_user_click_brandA_count,
day30_user_click_brandA_count,
day7_user_click_brandB_count,
day30_user_click_brandB_count,
day7_user_click_brandC_count,
day30_user_click_brandC_count,
day7_user_click_brandD_count,
day30_user_click_brandD_count,
day7_user_click_brandS_count,
day30_user_click_brandS_count,
day7_user_click_brand0_count,
day30_user_click_brand0_count,
day30_user_click_count,
day30_user_click_daigou_max,
day30_user_click_daigou_min,
day2_user_click_daigou_avg,
day7_user_click_daigou_avg,
day30_user_click_daigou_avg,
day2_user_click_sale_max,
day7_user_click_sale_max,
day30_user_click_sale_max,
day2_user_click_sale_min,
day7_user_click_sale_min,
day30_user_click_sale_min,
day2_user_click_sale_avg,
day7_user_click_sale_avg,
day30_user_click_sale_avg,

day2_user_click_brand_count,
day7_user_click_brand_count,
day30_user_click_brand_count,
```

```
day2_user_click_ctgry3_count,
day7_user_click_ctgry3_count,
day30_user_click_ctgry3_count,

day2_user_click_sale_percentile02,
day7_user_click_sale_percentile02,
day30_user_click_sale_percentile02,
day2_user_click_sale_percentile08,
day7_user_click_sale_percentile08,
day30_user_click_sale_percentile08
FROM

((SELECT
user_id ,user_event_date,
sum( CASE WHEN datediffx<=1 AND a1.spm = '30.28.0.0.0' THEN 1 END) as d
ay1_user_productpageview_expose,
sum( CASE WHEN datediffx<=2 AND a1.spm = '30.28.0.0.0' THEN 1 END) as d
ay2_user_productpageview_expose,
sum( CASE WHEN datediffx<=7 AND a1.spm = '30.28.0.0.0' THEN 1 END) as d
ay7_user_productpageview_expose,
sum( CASE WHEN datediffx<=15 AND a1.spm = '30.28.0.0.0' THEN 1 END) as
day15_user_productpageview_expose,
sum( CASE WHEN datediffx<=30 AND a1.spm = '30.28.0.0.0' THEN 1 END) as
day30_user_productpageview_expose,
sum( CASE WHEN datediffx<=2 THEN 1 END) as day2_user_click_count,
sum( CASE WHEN datediffx<=7 THEN 1 END) as day7_user_click_count,
sum( CASE WHEN datediffx<=7 AND brand_level = "A" THEN 1 END) as day7_u
ser_click_brandA_count,
sum( CASE WHEN brand_level = "A" THEN 1 END) as day30_user_click_brandA
_count,
sum( CASE WHEN datediffx<=7 AND brand_level = "B" THEN 1 END) as day7_u
ser_click_brandB_count,
sum( CASE WHEN brand_level = "B" THEN 1 END) as day30_user_click_brandB
_count,
sum( CASE WHEN datediffx<=7 AND brand_level = "C" THEN 1 END) as day7_u
ser_click_brandC_count,
sum( CASE WHEN brand_level = "C" THEN 1 END) as day30_user_click_brandC
_count,
sum( CASE WHEN datediffx<=7 AND brand_level = "D" THEN 1 END) as day7_u
ser_click_brandD_count,
sum( CASE WHEN brand_level = "D" THEN 1 END) as day30_user_click_brandD
_count,
sum( CASE WHEN datediffx<=7 AND brand_level = "S" THEN 1 END) as day7_u
ser_click_brandS_count,
sum( CASE WHEN brand_level = "S" THEN 1 END) as day30_user_click_brandS
_count,
sum( CASE WHEN datediffx<=7 AND brand_level = "0" THEN 1 END) as day7_u
ser_click_brand0_count,
sum( CASE WHEN brand_level = "0" THEN 1 END) as day30_user_click_brand0
_count,
```

```
COUNT(1) as day30_user_click_count,
MAX(CAST(daigou_fee as DOUBLE)) as day30_user_click_daigou_max,
MIN(CAST(daigou_fee as DOUBLE)) as day30_user_click_daigou_min,
AVG(CASE WHEN datediffx<=2 THEN CAST(daigou_fee as DOUBLE) END) as day2
_user_click_daigou_avg,
AVG(CASE WHEN datediffx<=7 THEN CAST(daigou_fee as DOUBLE) END) as day7
_user_click_daigou_avg,
AVG(CAST(daigou_fee as DOUBLE)) as day30_user_click_daigou_avg,
MAX(CASE WHEN datediffx<=2 THEN CAST(sale_price as DOUBLE) END) as day2
_user_click_sale_max,
MAX(CASE WHEN datediffx<=7 THEN CAST(sale_price as DOUBLE) END) as day7
_user_click_sale_max,
MAX(CAST(sale_price as DOUBLE)) as day30_user_click_sale_max,
MIN(CASE WHEN datediffx<=2 THEN CAST(sale_price as DOUBLE) END) as day2
_user_click_sale_min,
MIN(CASE WHEN datediffx<=7 THEN CAST(sale_price as DOUBLE) END) as day7
_user_click_sale_min,
MIN(CAST(sale_price as DOUBLE)) as day30_user_click_sale_min,
AVG(CASE WHEN datediffx<=2 THEN CAST(sale_price as DOUBLE) END) as day2
_user_click_sale_avg,
AVG(CASE WHEN datediffx<=7 THEN CAST(sale_price as DOUBLE) END) as day7
_user_click_sale_avg,
AVG(CAST(sale_price as DOUBLE)) as day30_user_click_sale_avg
FROM
(SELECT *, datediff1(to_date1(user_event_date,"yyyy-mm-dd"), to_date1(c
lick_date,"yyyy-mm-dd"),"dd") as datediffx
FROM syj_online_h5_search_rank_user_click_list_20201103) a1 GROUP BY us
er_id,user_event_date) agg_func
join
(SELECT
user_id, user_event_date,
COUNT(DISTINCT CASE WHEN datediffx<=2 THEN brand_index END) as day2_use
r_click_brand_count,
COUNT(DISTINCT CASE WHEN datediffx<=7 THEN brand_index END) as day7_use
r_click_brand_count,
COUNT(DISTINCT brand_index) as day30_user_click_brand_count,
COUNT(DISTINCT CASE WHEN datediffx<=2 THEN ctgry_three_index END) as da
y2_user_click_ctgry3_count,
COUNT(DISTINCT CASE WHEN datediffx<=7 THEN ctgry_three_index END) as da
y7_user_click_ctgry3_count,
COUNT(DISTINCT ctgry_three_index) as day30_user_click_ctgry3_count
FROM
(SELECT *, datediff1(to_date1(user_event_date,"yyyy-mm-dd"), to_date1(c
lick_date,"yyyy-mm-dd"),"dd") as datediffx
FROM syj_online_h5_search_rank_user_click_list_20201103) GROUP BY user_
id,user_event_date) agg_dist_func
join
(SELECT
user_id, user_event_date,
percentile(CASE WHEN datediffx<=2 THEN CAST(sale_price as DOUBLE) END,
```

```
0.2) as day2_user_click_sale_percentile02,
percentile(CASE WHEN datediffx<=7 THEN CAST(sale_price as DOUBLE) END,
0.2) as day7_user_click_sale_percentile02,
percentile(CAST(sale_price as DOUBLE),0.2) as day30_user_click_sale_per
centile02,
percentile(CASE WHEN datediffx<=2 THEN CAST(sale_price as DOUBLE) END,
0.8) as day2_user_click_sale_percentile08,
percentile(CASE WHEN datediffx<=7 THEN CAST(sale_price as DOUBLE) END,
0.8) as day7_user_click_sale_percentile08,
percentile(CAST(sale_price as DOUBLE),0.8) as day30_user_click_sale_per
centile08
FROM
(SELECT *, datediff1(to_date1(user_event_date,"yyyy-mm-dd"), to_date1(c
lick_date,"yyyy-mm-dd"),"dd") as datediffx
FROM syj_online_h5_search_rank_user_click_list_20201103) GROUP BY user_
id,user_event_date) percentile_func
on agg_func.user_id = agg_dist_func.user_id and
agg_func.user_id = percentile_func.user_id and
agg_func.user_event_date = agg_dist_func.user_event_date and
agg_func.user_event_date = percentile_func.user_event_date)
```

SQL9

作业 : h5*livepage*product*rank*for*list*total*20210117*v1

h5*livepage*product*rank*for*list*online*buyers*ctr*30days*20210117_v1

job id: e33dccf6-1c62-42da-ac48-4b6b3dab72de

```
create table if not exists h5_livepage_product_rank_for_list_online_use
rs_expose_products_30days as
select t1.server_date,t1.buyer_id,count(distinct t4.external_product_co
de) as expose_product_count_30days,count(t4.external_product_code) as e
xpose_product_record_count_30days
from akc_alg.h5_livepage_product_rank_for_list_online_buyers_ndays t1
left join akdc.stg_mshop_member_member t2 on t1.buyer_id=t2.user_id
left join akdc.ods_ubt_event t3 on t2.open_id=t3.user_id
left join akdc.stg_mshop_product t4 on REGEXP_EXTRACT(t3.properties, '"
name":"productNo","value":"(.*?)(","type")', 1)=cast(t4.no as string)
where datediff1(to_date1(t1.server_date, 'yyyymmdd'),to_date1(t3.dt, 'y
yyymmdd'), 'dd')<31 and datediff1(to_date1(t1.server_date, 'yyyymmdd'),
to_date1(t3.dt, 'yyyymmdd'), 'dd')>0
and t3.dt>to_char(dateadd(getdate(), -31, 'dd'),'yyyymmdd')
and replace(t3.spm,'.0','') in ('30.17.20.21')
and t3.action_type='expose'
and REGEXP_EXTRACT(t3.properties, '"name":"productNo","value":"(.*?)(",
"type")', 1)<>''
and t2.open_id is not null
and t4.external_product_code is not null
group by t1.server_date,t1.buyer_id;
```

修改参数

```
spark.sql.statistics.fallBackToHdfs=true;
spark.sql.adaptive.join.enabled=true;
spark.sql.adaptive.enabled=true;
spark.sql.autoBroadcastJoinThreshold=262144000;
spark.sql.shuffle.partitions=2000;
spark.sql.adaptive.minNumPostShufflePartitions=500;
spark.sql.adaptive.maxNumPostShufflePartitions=2000;
spark.sql.files.maxPartitionBytes=2147483648;
```

SQL10

作业： h5*homepage*live*rank*deepfm*total*20210117_v1

h5*homepage*live*rank*deepfm*MTL*online*buyers*click_prepare

id：4f2ab675-8492-4252-bedf-dc2219aec813

```
CREATE TABLE h5_homepage_live_rank_deepfm_MTL_online_user_click_list_20
200221_prepare as
SELECT
cast(s1.user_id as string) as user_id,o1.dt,
s2.external_product_code as product_id,s3.external_code as live_id ,o1.
spm,
MIN(case when to_char(o1.client_ts,"yyyymmdd") = o1.dt THEN o1.client_t
s ELSE o1.server_ts END) as server_ts,
max(o1.city) as click_city,
max(o1.province) as click_province
FROM akdc.ods_ubt_event o1
JOIN akdc.stg_mshop_member_member s1 on o1.user_id=s1.open_id
JOIN akdc.stg_mshop_product s2 on REGEXP_EXTRACT(o1.properties, '"name
":"productNo","value":"([0-9]+a*)","type")', 1) = cast(s2.no as strin
g)
JOIN akdc.stg_mshop_activity s3 on REGEXP_EXTRACT(o1.properties, '"name
":"activityNo","value":"([0-9]+a*)","type")', 1) = cast(s3.no as strin
g)
WHERE replace(o1.spm,'.0','') in ('30.28','30.30.20.21','30.17.20.21','
30.38.11.12','30.38.11.12.21','30.38.11.12.13','30.38.11.12.34')
AND o1.dt >= to_char(dateadd(getdate(), -35, 'dd'), 'yyyymmdd')
AND o1.action_type in ("click","pageview")
GROUP BY cast(s1.user_id as string),o1.dt,s2.external_product_code,s3.e
xternal_code,o1.spm
```

SQL11

开启 AE 失败

```sql
SELECT distinct

-- 用户
a1.user_id
,a1.user_event_hour

-- 场景
,a2.spm
,a2.dt
,a2.server_ts
,to_char(a2.server_ts,"yyyy-mm-dd hh:00:00") as event_hour
,a2.action_type

-- 商品
,a3.new_product_id as product_id
        -- 库存
    ,ROUND(a6.daigou_fee,1) AS daigou_fee
    -- 平台销售价格
    ,ROUND(a6.sale_price,1) AS sale_price
    -- 吊牌价
    ,ROUND(a6.tag_price,1) AS tag_price
    ,ROUND(a6.tag_price-a6.sale_price,1) as diff_price
    -- 供货价
    ,ROUND(a6.supply_price,1) AS supply_price

-- 品牌
,a3.brand_id
-- ,a4.brand_index
-- ,a5.brand_level
-- ,a5.brand_score
,case when a5.brand_level is not null then a5.brand_level else -1 end a
s brand_level
,case when a5.brand_score is not null then a5.brand_score else -1 end a
s brand_score

-- 类目
,a3.ctgry_three_id
-- ,cate3.cate3_index as ctgry_three_index

-- 活动
,a2.live_id
,a4.online_product_count



,a2.record_num

FROM
```

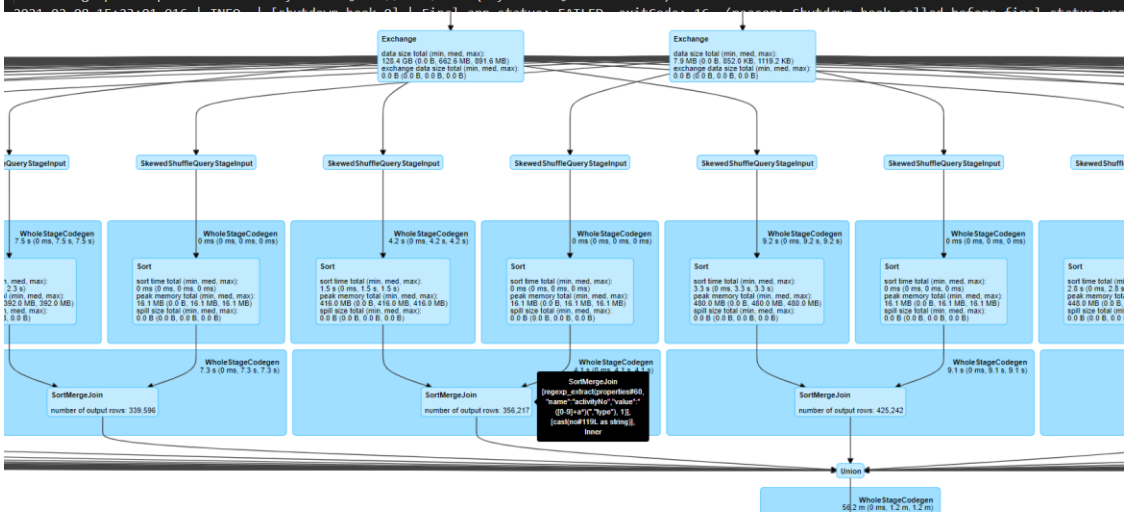
```
tmp_online_pro_rank_user_event_hour_20201026 a1
JOIN
(
    SELECT
    o1.dt,
    MIN(case when to_char(o1.client_ts,"yyyymmdd") = o1.dt THEN o1.clie
nt_ts ELSE o1.server_ts END) as server_ts,
    o1.spm,
    cast(s1.user_id as string) as user_id,
    s2.external_product_code as product_id,
    s3.external_code as live_id,
    o1.action_type,
    count(1) as record_num

    FROM akcbi.ods_ubt_event o1
    JOIN akdc.stg_mshop_member_member s1 on o1.user_id=s1.open_id
    JOIN akdc.stg_mshop_product s2  on REGEXP_EXTRACT(o1.properties, '"
name":"productNo","value":"([0-9]+a*)(","type")', 1)  = cast(s2.no as s
tring)
    JOIN akdc.stg_mshop_activity s3 on REGEXP_EXTRACT(o1.properties, '"
name":"activityNo","value":"([0-9]+a*)(","type")', 1) = cast(s3.no as s
tring)
    WHERE o1.spm in (
                        -- 搜索列表        活动列表          商详列表
                        -- 点击
                        '30.30.20.21',   '30.17.20.21',   '30.28.52.21
',
                        -- 转发
                        '30.30.20.21.19','30.17.20.21.19','30.28.52.21.
19'
                        )
    -- AND o1.dt >= (SELECT TO_CHAR(dateadd(sample_begin_time, -1, 'dd
'),'yyyymmdd') FROM tmp_online_pro_rank_sliding_window_20201026)
    -- and o1.dt <= (SELECT to_char(sample_end_time,"yyyymmdd") FROM tm
p_online_pro_rank_sliding_window_20201026)
    AND o1.dt >= to_char(dateadd(getdate(), -1, 'dd'),"yyyymmdd")
    and o1.dt <= to_char(dateadd(getdate(), 0, 'dd'),"yyyymmdd")

    GROUP BY o1.dt,o1.spm,cast(s1.user_id as string),s2.external_produc
t_code,s3.external_code,o1.action_type
) a2
ON a1.user_id = a2.user_id
AND datediff1(a1.user_event_hour, to_char(a2.server_ts,"yyyy-mm-dd hh:0
0:00"),"hh") > 0
AND datediff1(a1.user_event_hour, to_char(a2.server_ts,"yyyy-mm-dd hh:0
0:00"),"hh") <=
24 LEFT JOIN akdc.dim_product_info a3 ON a2.product_id = a3.product_id
left join akdc.stg_mshop_activity a4 on a2.live_id=a4.external_code
LEFT JOIN akdc.dim_brand_info a5 ON a5.brand_id = a3.brand_id
```

```sql
LEFT JOIN
 (   SELECT
     cast(product_id as string) as product_id,
    AVG(daigou_fee) as daigou_fee,
    AVG(sale_price) as sale_price,
    AVG(tag_price) as tag_price,
    AVG(supply_price) as supply_price
    FROM akdc.stg_mer_inventory_mer_inventory GROUP BY product_id
) a6 ON a3.new_product_id = a6.product_id;
```

```
2021-02-08 15:23:01,905 | ERROR | [spark-listener-group-eventLog] | Uncaught exception in thread Thread[spark-listener-group-eventLog,5,main] | org
java.lang.OutOfMemoryError: Java heap space
    at java.lang.AbstractStringBuilder.<init>(AbstractStringBuilder.java:68)
    at java.lang.StringBuilder.<init>(StringBuilder.java:101)
    at com.fasterxml.jackson.core.util.TextBuffer.contentsAsString(TextBuffer.java:416)
    at com.fasterxml.jackson.core.io.SegmentedStringWriter.getAndClear(SegmentedStringWriter.java:83)
    at com.fasterxml.jackson.databind.ObjectMapper.writeValueAsString(ObjectMapper.java:3410)
    at org.apache.spark.util.JsonProtocol$.sparkEventToJson(JsonProtocol.scala:103)
    at org.apache.spark.scheduler.EventLoggingListener.logEvent(EventLoggingListener.scala:151)
    at org.apache.spark.scheduler.EventLoggingListener.onOtherEvent(EventLoggingListener.scala:296)
    at org.apache.spark.scheduler.SparkListenerBus$class.doPostEvent(SparkListenerBus.scala:76)
    at org.apache.spark.scheduler.AsyncEventQueue.doPostEvent(AsyncEventQueue.scala:37)
    at org.apache.spark.scheduler.AsyncEventQueue.doPostEvent(AsyncEventQueue.scala:37)
    at org.apache.spark.util.ListenerBus$class.postToAll(ListenerBus.scala:91)
    at org.apache.spark.scheduler.AsyncEventQueue.org$apache$spark$scheduler$AsyncEventQueue$$super$postToAll(AsyncEventQueue.scala:92)
    at org.apache.spark.scheduler.AsyncEventQueue$$anonfun$org$apache$spark$scheduler$AsyncEventQueue$$dispatch$1.apply$mcJ$sp(AsyncEventQueue.scal
    at org.apache.spark.scheduler.AsyncEventQueue$$anonfun$org$apache$spark$scheduler$AsyncEventQueue$$dispatch$1.apply(AsyncEventQueue.scala:87)
    at org.apache.spark.scheduler.AsyncEventQueue$$anonfun$org$apache$spark$scheduler$AsyncEventQueue$$dispatch$1.apply(AsyncEventQueue.scala:87)
    at scala.util.DynamicVariable.withValue(DynamicVariable.scala:58)
    at org.apache.spark.scheduler.AsyncEventQueue.org$apache$spark$scheduler$AsyncEventQueue$$dispatch(AsyncEventQueue.scala:87)
    at org.apache.spark.scheduler.AsyncEventQueue$$anon$1$$anonfun$run$1.apply$mcV$sp(AsyncEventQueue.scala:83)
    at org.apache.spark.util.Utils$.tryOrStopSparkContext(Utils.scala:1447)
    at org.apache.spark.scheduler.AsyncEventQueue$$anon$1.run(AsyncEventQueue.scala:82)
```



配置

spark.sql.enableToString=false

spark.sql.adaptive.enableToString=false

SQL12

```sql
create table if not exists h5_capsule_products_rec_svd_channel_us
er_top_cluster_style_no_1 as
```

```
select
  a.buyer_id,
  a.cluster_index,
  a.predict_score,
  a.predict_score_rank,
  a.style_no_code,
  a.product_predict_score,
  ROW_NUMBER() over (
    partition by a.buyer_id,
    a.predict_score_rank
    order by
      a.product_predict_score desc
  ) as product_score_rank
from
  (
    select
      t1.buyer_id,
      t1.cluster_index,
      t1.predict_score,
      t1.predict_score_rank,
      t3.style_no_code,
      t2.factors_1 * t3.factors_1 + t2.factors_2 * t3.factors_2 +
 t2.factors_3 * t3.factors_3 + t2.factors_4 * t3.factors_4 + t2.f
actors_5 * t3.factors_5 + t2.factors_6 * t3.factors_6 + t2.factor
s_7 * t3.factors_7 + t2.factors_8 * t3.factors_8 + t2.factors_9 *
 t3.factors_9 + t2.factors_10 * t3.factors_10 + t2.factors_11 * t
3.factors_11 + t2.factors_12 * t3.factors_12 + t2.factors_13 * t3
.factors_13 + t2.factors_14 * t3.factors_14 + t2.factors_15 * t3.
factors_15 + t2.factors_16 * t3.factors_16 + t2.factors_17 * t3.f
actors_17 + t2.factors_18 * t3.factors_18 + t2.factors_19 * t3.fa
```

```
ctors_19 + t2.factors_20 * t3.factors_20 + t2.factors_21 * t3.fac
tors_21 + t2.factors_22 * t3.factors_22 + t2.factors_23 * t3.fact
ors_23 + t2.factors_24 * t3.factors_24 + t2.factors_25 * t3.facto
rs_25 + t2.factors_26 * t3.factors_26 + t2.factors_27 * t3.factor
s_27 + t2.factors_28 * t3.factors_28 + t2.factors_29 * t3.factors
_29 + t2.factors_30 * t3.factors_30 + t2.factors_31 * t3.factors_
31 + t2.factors_32 * t3.factors_32 + t2.factors_33 * t3.factors_3
3 + t2.factors_34 * t3.factors_34 + t2.factors_35 * t3.factors_35
 + t2.factors_36 * t3.factors_36 + t2.factors_37 * t3.factors_37
+ t2.factors_38 * t3.factors_38 + t2.factors_39 * t3.factors_39 +
 t2.factors_40 * t3.factors_40 + t2.factors_41 * t3.factors_41 +
t2.factors_42 * t3.factors_42 + t2.factors_43 * t3.factors_43 + t
2.factors_44 * t3.factors_44 + t2.factors_45 * t3.factors_45 + t2
.factors_46 * t3.factors_46 + t2.factors_47 * t3.factors_47 + t2.
factors_48 * t3.factors_48 + t2.factors_49 * t3.factors_49 + t2.f
actors_50 * t3.factors_50 + t2.factors_51 * t3.factors_51 + t2.fa
ctors_52 * t3.factors_52 + t2.factors_53 * t3.factors_53 + t2.fac
tors_54 * t3.factors_54 + t2.factors_55 * t3.factors_55 + t2.fact
ors_56 * t3.factors_56 + t2.factors_57 * t3.factors_57 + t2.facto
rs_58 * t3.factors_58 + t2.factors_59 * t3.factors_59 + t2.factor
s_60 * t3.factors_60 + t2.factors_61 * t3.factors_61 + t2.factors
_62 * t3.factors_62 + t2.factors_63 * t3.factors_63 + t2.factors_
64 * t3.factors_64 + t2.factors_65 * t3.factors_65 + t2.factors_6
6 * t3.factors_66 + t2.factors_67 * t3.factors_67 + t2.factors_68
 * t3.factors_68 + t2.factors_69 * t3.factors_69 + t2.factors_70
* t3.factors_70 + t2.factors_71 * t3.factors_71 + t2.factors_72 *
 t3.factors_72 + t2.factors_73 * t3.factors_73 + t2.factors_74 *
t3.factors_74 + t2.factors_75 * t3.factors_75 + t2.factors_76 * t
3.factors_76 + t2.factors_77 * t3.factors_77 + t2.factors_78 * t3
.factors_78 + t2.factors_79 * t3.factors_79 + t2.factors_80 * t3.
factors_80 + t2.factors_81 * t3.factors_81 + t2.factors_82 * t3.f
actors_82 + t2.factors_83 * t3.factors_83 + t2.factors_84 * t3.fa
ctors_84 + t2.factors_85 * t3.factors_85 + t2.factors_86 * t3.fac
tors_86 + t2.factors_87 * t3.factors_87 + t2.factors_88 * t3.fact
ors_88 + t2.factors_89 * t3.factors_89 + t2.factors_90 * t3.facto
rs_90 + t2.factors_91 * t3.factors_91 + t2.factors_92 * t3.factor
s_92 + t2.factors_93 * t3.factors_93 + t2.factors_94 * t3.factors
_94 + t2.factors_95 * t3.factors_95 + t2.factors_96 * t3.factors_
96 + t2.factors_97 * t3.factors_97 + t2.factors_98 * t3.factors_9
8 + t2.factors_99 * t3.factors_99 + t2.factors_100 * t3.factors_1
00 as product_predict_score

    from

      h5_capsule_products_rec_svd_channel_user_cluster_distance_1
 t1
```

```
      left join h5_capsule_products_rec_svd_channel_user_svd_factors_1 t2 on t1.buyer_id = t2.buyer_id

      left join h5_capsule_products_rec_svd_channel_idxTable_1 t3 on t1.cluster_index = t3.cluster_index

    where

      t1.predict_score_rank <= 20

  ) a
```

Summary Metrics for 387 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	0 ms	53 ms	62 ms	0.2 s	3.4 h
GC Time	0 ms	0 ms	0 ms	14 ms	2.9 min
Shuffle Read Size / Records	0.0 B / 0	0.0 B / 0	0.0 B / 0	0.0 B / 0	1557.7 MB / 2246254
Shuffle Write Size / Records	0.0 B / 0	0.0 B / 0	0.0 B / 0	0.0 B / 0	16.4 GB / 1242562560
Shuffle spill (memory)	0.0 B	0.0 B	0.0 B	0.0 B	73.9 GB
Shuffle spill (disk)	0.0 B	0.0 B	0.0 B	0.0 B	17.8 GB

Aggregated Metrics by Executor

Tasks (400)

Page: 1 2 3 4 > 4 Pages. Jump to 1 . Show 100 items in a page. Go

Index	ID	Attempt	Status	Locality Level	Executor ID	Host	Launch Time	Duration	GC Time	Shuffle Read Size / Records	Write Time	Shuffle Write Size / Records	Shuffle Spill (Memory)	Shuffle Spill (Disk)	Errors
218	11068422	0	SUCCESS	PROCESS_LOCAL	76	sparkce34046-worker-instance-7-1 stdout stderr	2021/01/31 16:39:35	3.4 h	2.5 min	978.2 MB / 1715289	44 s	16.4 GB / 1148405655	69.3 GB	17.1 GB	
64	10505322	0	SUCCESS	PROCESS_LOCAL	22	sparkce34046-worker-instance-5-1 stdout stderr	2021/01/31 16:18:35	2.4 h	1.8 min	715.2 MB / 1122840	34 s	12.3 GB / 871178569	52.7 GB	13.1 GB	
204	11062471	0	SUCCESS	PROCESS_LOCAL	143	sparkce34046-worker-instance-13-1 stdout stderr	2021/01/31 16:39:21	2.2 h	1.6 min	240.2 MB / 309561	37 s	11.8 GB / 829437174	49.4 GB	11.8 GB	

队列名称	执行引擎	创建时间	作业类型	状态	执行语句	运行时长	操作
sql2_1024	spark	2021/01/31 16:17:5...	INSERT	已成功	create table if not exists h5_capsule_products_re...	11h 3min 2...	编辑 | 终止 | SparkUI | 更多

队列名称	sql2_1024	作业ID	ed79f241-ff88-42b1-97b5-d92382346e5c
创建时间	2021/01/31 16:17:53 GMT+08:00	作业类型	INSERT
作业状态	已成功	执行语句	create table if not exists h5_capsule_products_rec_svd_channel_user_to...
运行时长	11h 3min 22.33s	配置项	spark.sql.parser.quotedRegexColumnNames:true; spark.sql.shuffle.parti...
已扫描数据	20.25 GB	执行用户	zhaoxiaoliang
结果条数	38146002643	扫描数据条数	2177434882
错误记录条数	0		

```
  spark.sql.adaptive.enabled:true; spark.sql.adaptive.skewedJoin.enabled:true; spark.sql.parser.quotedRegexColumnNames:true; spark.sql.adaptive.skewedPartitionMaxSplits:10;
```

调优参数优化后：

队列名称	执行引擎	创建时间	作业类型	状态	执行语句	运行时长	操作
dli_akc_sql_256	spark	2021/01/31 20:47:2...	QUERY	已成功	select a.buyer_id, a.cluster_index, a.predict_scor...	3h 1min 51....	编辑 | 终止 | SparkUI | 更多

队列名称	dli_akc_sql_256	作业ID	8148a52d-0e76-405c-9be4-11f749c90f50
创建时间	2021/01/31 20:47:29 GMT+08:00	作业类型	QUERY
作业状态	已成功	执行语句	select a.buyer_id, a.cluster_index, a.predict_score, a.predict_score_rank, ...
运行时长	3h 1min 51.04s	配置项	spark.sql.adaptive.enabled:true; spark.sql.adaptive.skewedJoin.enabled:...
结果条数	38041708643 （导出结果）	已扫描数据	20.26 GB
执行用户	zhouxuelin	结果状态	数据已缓存 （查看结果）

Spark 参数配置

spark.sql.autoBroadcastJoinThreshold

broadCastJoin 小表的最大值，默认值为 10MB，单位 byte。如果值为-1，表示不开启 broadCastJoin。当表没有统计信息时，表的大小默认为 Long.Max， 因此当表没有统计信息时，优化器默认不做 BroadcastJoin。因为 Driver 是允许多个 SQL 作业并行运行的，如果多个 SQL 有 BroadCastJoin，或者 autoBroadcastJoinThreshold 值比较大时，比较容易导致 Driver 内存溢出，因此需要谨慎使用 BroadCastJoin。

spark.sql.statistics.fallBackToHdfs

默认为 false。 设置为 true，逻辑优化使用的表的统计信息(主要是 sizeInBytes)不从元数据中获取， 而是从文件系统中计算所有文件大小而得到。在生产环境中，元数据中经常没有表的统计信息，Catalyst 优化器无法做一些基于统计信息的优化，比如**Join Reorder、BroadCastJoin** ，因此可以开启此参数，从文件系统中获取表的统计信息，以便做一些逻辑优化。
对于 SQL 里面查询 hive 外表，同时执行时间比较长的配置改参数，改参数会导致生成执行计划时间变长。

spark.sql.files.maxPartitionBytes

DataSource 表每个 partition 读取最大字节数，默认 128MB，通过这个参数可以调节 map 阶段的 task 数量。

spark.sql.shuffle.partitions

默认是 200，对数据膨胀情况，可以调大

1 Pages. Jump to 1 . Show 1000 items in a

ation	GC Time	Shuffle Read Size / Records	Write Time	Shuffle Write Size / Records	Shuffle Spill (Memory)	Shuffle Spill (D
s	4.0 min	844.3 MB / 1204808		0.0 B / 2001731584	120.2 GB	29.5 G
s	3.6 min	807.8 MB / 1242811		0.0 B / 1855979520	111.5 GB	27.4 G
h	35 s	516.2 MB / 675557	13 s	6.4 GB / 443409300	26.4 GB	6.4 GB
h	2.5 min	1466.9 MB / 2011578	1.3 min	23.4 GB / 1628722080	97.9 GB	24.2 G

spark.sql.adaptive.enabled

开启 AE 查询模式，支持版本:`DLI spark-2.3.2`

spark.sql.adaptive.skewedJoin.enabled

开启 join 倾斜优化

spark.sql.adaptive.skewedPartitionMaxSplits

处理 join 倾斜分区的最大任务数量，默认为 5。最好不要调整该参数，某些场景下会导致执行计划太复杂，影响 driver。

spark.sql.adaptive.join.enabled

默认 false,开启后，在运行时可根据统计信息将 sortMergeJoin 转为 BroadCastHashJoin.

spark.sql.enableToString

在开启 AE 情况下，配置为 false，减少 driver 内存占用，建议所以开启 AE 都配置为 false

spark.sql.adaptive.enableToString

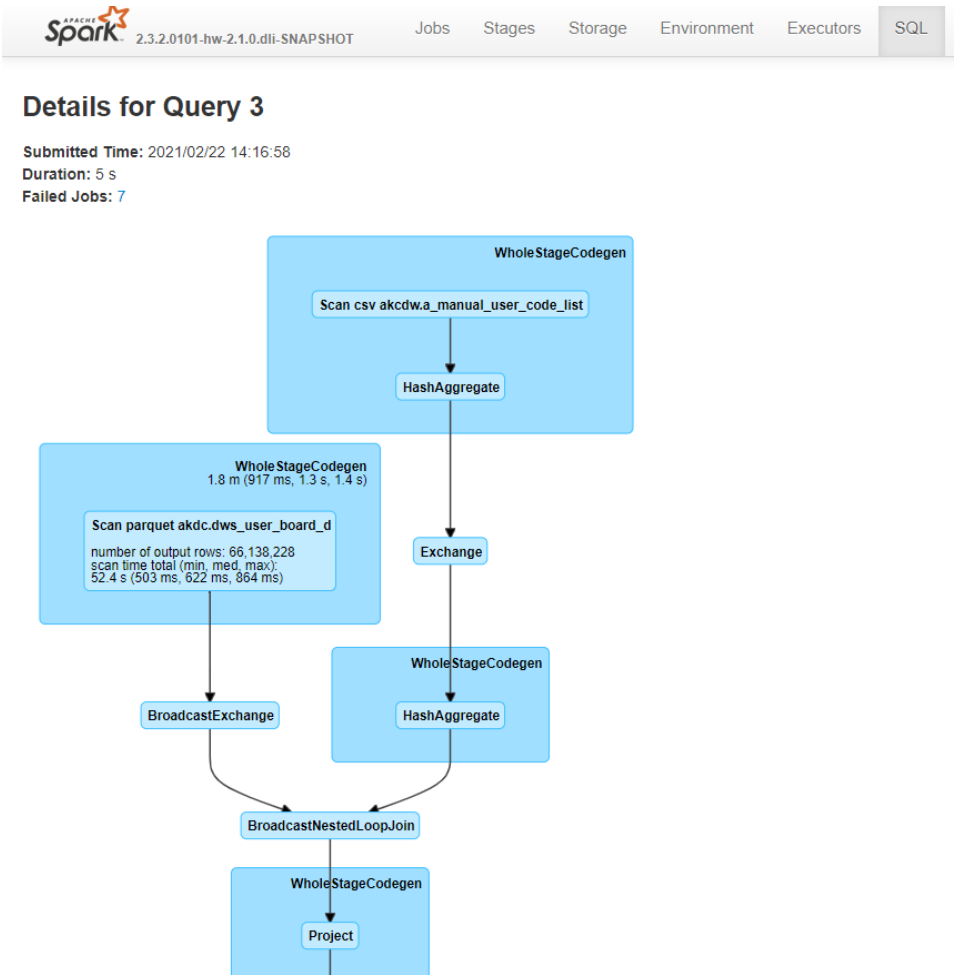
在开启 AE 情况下，配置为 false，减少 driver 内存占用，建议所以开启 AE 都配置为 false

SQL 调整

NOT IN

```
select
 a.month,
 a.register_month,
 datediff1('2021-02-22 00:00:00', a.register_time2, "dd") shichang,
 a.user_id,
 a.user_code,
 a.user_level,
 sum(a.gmv) gmv,
 sum(a.gmv_mshop_distributor) gmv_distributor,
 sum(a.gmv - a.gmv_mshop_distributor) gmv_seller,
 sum(a.new_trans_buyer_cnt) new_trans_buyer_cnt,
 count(
  distinct case
   when a.month = a.register_month then a.user_id
  end
 ) is_newb,
 count(
  distinct case
   when a.month = a.register_month
   and a.gmv > 0 then a.user_id
  end
```

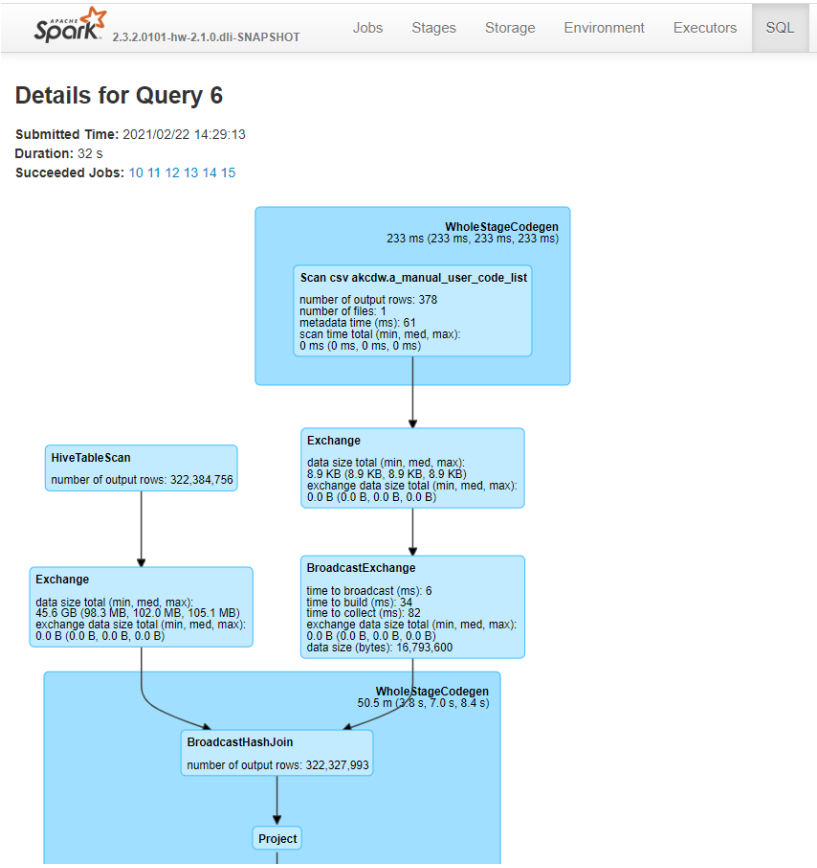
```
  ) is_deal_newb
from
  (
    select
      *,
      substr(dt, 1, 6) month,
      replace(substr(register_time, 1, 7), '-', '') register_month,
      to_date1(register_time, "yyyy-mm-dd") register_time2
    from
      akdc.dws_user_board_d
    where
      dt between '20200901'
      and '20210131'
      and user_code not in (
        select
          distinct user_code
        from
          akcdw.a_manual_user_code_list
      )
  ) a
group by
  a.month,
  a.register_month,
  datediff1('2021-02-22 00:00:00', a.register_time2, "dd"),
  a.user_id,
  a.user_code,
  a.user_level
```




```
select
  a.month,
  a.register_month,
  a.user_id,
  a.user_code,
  a.user_level,
  sum(a.gmv) gmv,
  sum(a.gmv_mshop_distributor) gmv_distributor,
  sum(a.gmv - a.gmv_mshop_distributor) gmv_seller,
  sum(a.new_trans_buyer_cnt) new_trans_buyer_cnt,
  count(
```

```
    distinct case
      when a.month = a.register_month then a.user_id
    end
  ) is_newb,
  count(
    distinct case
      when a.month = a.register_month
      and a.gmv > 0 then a.user_id
    end
  ) is_deal_newb
from
  (
    select
      *,
      substr(dt, 1, 6) month,
      replace(substr(register_time, 1, 7), '-', '') register_month,
      datediff1('2021-02-22 00:00:00', register_time, "dd") shichang
    from
      akdc.dws_user_board_d uc
    where
      dt >= '20200901'
      and dt <= '20210131'
      and not exists (
        select
          *
        from
          akcdw.a_manual_user_code_list amu
        where
```

```
        uc.user_code = amu.user_code
    )
  ) a
group by
  a.month,
  a.register_month,
  a.user_id,
  a.user_code,
  a.user_level
```



相似的计算逻辑重复执行

```
h5_livepage_product_rank_for_list_total_20210117_v1
```

h5_livepage_product_rank_for_list_online_buyers_ctr_14days

```
create table if not exists h5_livepage_product_rank_for_list_onli
ne_users_expose_products_14days as

select

  t1.server_date,

  t1.buyer_id,

  count(distinct t4.external_product_code) as expose_product_coun
t_14days,

  count(t4.external_product_code) as expose_product_record_count_
14days

from

  h5_livepage_product_rank_for_list_online_buyers_ndays t1

  left join akdc.stg_mshop_member_member t2 on t1.buyer_id = t2.u
ser_id

  left join akdc.ods_ubt_event t3 on t2.open_id = t3.user_id

  left join akdc.stg_mshop_product t4 on REGEXP_EXTRACT(

    t3.properties,

    '"name":"productNo","value":"(.*?)(","type")',

    1

  ) = cast(t4.no as string)

where

  datediff1(

    to_date1(t1.server_date, 'yyyymmdd'),

    to_date1(t3.dt, 'yyyymmdd'),

    'dd'

  ) < 15

  and datediff1(
```

```
    to_date1(t1.server_date, 'yyyymmdd'),
    to_date1(t3.dt, 'yyyymmdd'),
    'dd'
  ) > 0
  and t3.dt > to_char(dateadd(getdate(), -15, 'dd'), 'yyyymmdd')
  and replace(t3.spm, '.0', '') in ('30.17.20.21')
  and t3.action_type = 'expose'
  and REGEXP_EXTRACT(
    t3.properties,
    '"name":"productNo","value":"(.*?)(","type")',
    1
  ) <> ''
  and t2.open_id is not null
  and t4.external_product_code is not null
group by
  t1.server_date,
  t1.buyer_id
```

h5_livepage_product_rank_total_20210117_v1

h5_livepage_product_ranking_online_buyers_ctr_14days

```
create table if not exists h5_livepage_product_ranking_online_use
rs_expose_products_14days as
select
  t1.server_date,
  t1.buyer_id,
```

```
  count(distinct t4.external_product_code) as expose_product_coun
t_14days,

  count(t4.external_product_code) as expose_product_record_count_
14days

from

  h5_livepage_product_ranking_online_buyers_ndays t1

  left join akdc.stg_mshop_member_member t2 on t1.buyer_id = t2.u
ser_id

  left join akdc.ods_ubt_event t3 on t2.open_id = t3.user_id

  left join akdc.stg_mshop_product t4 on REGEXP_EXTRACT(

    t3.properties,

    '"name":"productNo","value":"(.*?)(","type")',

    1

  ) = cast(t4.no as string)

where

  datediff1(

    to_date1(t1.server_date, 'yyyymmdd'),

    to_date1(t3.dt, 'yyyymmdd'),

    'dd'

  ) < 15

  and datediff1(

    to_date1(t1.server_date, 'yyyymmdd'),

    to_date1(t3.dt, 'yyyymmdd'),

    'dd'

  ) > 0

  and t3.dt > to_char(dateadd(getdate(), -15, 'dd'), 'yyyymmdd')

  and replace(t3.spm, '.0', '') in ('30.17.20.21')

  and t3.action_type = 'expose'
```

```
  and REGEXP_EXTRACT(
    t3.properties,
    '"name":"productNo","value":"(.*?)(","type")',
    1
  ) <> ''
  and t2.open_id is not null
  and t4.external_product_code is not null
group by
  t1.server_date,
  t1.buyer_id
```

两个 SQL 的逻辑非常类似，在多个作业中重复执行，可以考虑从数据上面优化，抽取一些中间表。

每天计算 30 天数据的，也可以考虑业务上进行优化

```
create table if not exists
h5_livepage_product_ranking_online_users_expose_products_30days as
select
 t1.server_date,
 t1.buyer_id,
 count(distinct t4.external_product_code) as expose_product_count_30days,
 count(t4.external_product_code) as expose_product_record_count_30days
from
 akc_alg.h5_livepage_product_ranking_online_buyers_ndays t1
 left join akdc.stg_mshop_member_member t2 on t1.buyer_id = t2.user_id
 left join akdc.ods_ubt_event t3 on t2.open_id = t3.user_id
 left join akdc.stg_mshop_product t4 on REGEXP_EXTRACT(
  t3.properties,
```

```
    '"name":"productNo","value":"(.*?)(","type")',
    1
  ) = cast(t4.no as string)
where
  datediff1(
    to_date1(t1.server_date, 'yyyymmdd'),
    to_date1(t3.dt, 'yyyymmdd'),
    'dd'
  ) < 31
  and datediff1(
    to_date1(t1.server_date, 'yyyymmdd'),
    to_date1(t3.dt, 'yyyymmdd'),
    'dd'
  ) > 0
  and t3.dt > to_char(dateadd(getdate(), -31, 'dd'), 'yyyymmdd')
  and replace(t3.spm, '.0', '') in ('30.17.20.21')
  and t3.action_type = 'expose'
  and REGEXP_EXTRACT(
    t3.properties,
    '"name":"productNo","value":"(.*?)(","type")',
    1
  ) <> ''
  and t2.open_id is not null
  and t4.external_product_code is not null
group by
  t1.server_date,
  t1.buyer_id
```