【最佳实践】实时计算Flink在游戏行业的实时数仓建设实践

行业背景

- 行业现状：
  - 随着互联网和移动互联网的相互促进与融合，以及PC终端和各类移动终端在智能化和便携性上的趋同，游戏产品跨平台运行于各类终端的需求逐步显现，特别是互联网页面游戏中的社交类游戏等产品跨平台运行于各类移动终端已经出现，随着版权价值意识的增强，游戏开发商和运营商在取得版权后，加强了对文化内容的开发利用，力图以多形式多媒介的产品实现版权价值的最大化。
- 大数据在游戏行业中的作用：
  - 根据游戏数据分析游戏产品趋势，实现精准营销
  - 根据玩家付费和活跃度等进行玩家画像，针对不同的玩家设计不同的商业化活动方案，提升付费玩家的体验，提升游戏消费额

业务场景

某游戏公司开发了个游戏APP，该公司在APP中会发布一些游戏场景、游戏角色、装备、精美皮肤等内容，玩家在线娱乐，产生充值购买等行为。
业务的构建涉及到几个端：

1. APP：应用程序，玩家访问入口，玩家主要进行如下操作：
   1. 注册账号
   2. 在线娱乐
   3. 游戏充值
2. 后台系统：对玩家行为数据进行分析，提供给运营/运维人员，用于辅助公司决策。
   1. 实时归档日志：用于OLAP查询或离线数据分析
   2. 实时KPI统计：统计不同时间段的游戏点击量，作为确定游戏活动开启、版本升级、服务器维护等操作时间的依据；根据游戏收益金额，制定更合理的商业化活动方式
   3. 实时统计TopN游戏：辅助公司对游戏APP开发资源、运营资源的分配决策

技术架构

架构解析：

数据采集：该场景中，数仓的数据来源有两部分：用户操作日志采集至日志服务（SLS），用户的购买充值等信息则通过RDS Binlog日志同步至DataHub。

实时数仓架构：该场景中，整个实时数仓的聚合统计，全部通过Flink完成，Flink实时读取SLS和DataHub的数据进行处理，并与维表进行关联查询等操作，最终实时统计的结果输入到下游数据库ODPS和RDS中。

业务指标

- 日志归档
- KPI统计
  - 游戏UV
  - 新增角色累积收益总额
  - 游戏评论次数
- 热门游戏TOP3

说明：该案例中仅包含以上场景及指标，在实际的应用场景下还包括游戏账号异地登录、玩家画像等其他指标。

业务代码

场景一：日志归档

> 本场景将用户点击游戏APP产生的日志，实时同步至ODPS进行日志归档，并提取日志产生的时间（按天、小时维度）等信息，用于运营人员进行离线分析。

输入表

```
CREATE TABLE game_log_source (
  log_t BIGINT,
  app_id VARCHAR ,
  app_ver VARCHAR,
  body VARCHAR,
  param1 VARCHAR,
  param2 VARCHAR,
  param3 VARCHAR,
  param4 VARCHAR,
  param5 VARCHAR,
  device_id VARCHAR,
  lcmid BIGINT
) WITH (
    type= 'sls',
  ...
);
```

输出表

```
CREATE TABLE game_log (
  log_t bigint,
  app_ver VARCHAR,
  device_id VARCHAR,
  mbga_uid bigint,
  param1 VARCHAR,
  param2 VARCHAR,
  param3 VARCHAR,
  param4 VARCHAR,
  param5 VARCHAR,
  `user_id` VARCHAR,
  a_typ VARCHAR,
  `zone` VARCHAR,
  `ahour` bigint,
  `dt` bigint
) with (
  type = 'odps',
  ...
);
```

业务代码

```
INSERT INTO game_log
SELECT
  log_t,
  app_ver,
  device_id,
  lcmid as mbga_uid,
  param1,
  param2,
  param3,
  param4,
  param5,
  SPLIT_INDEX (JSON_VALUE (body, '$.a_usr'), '@', 1) AS user_id,
  JSON_VALUE (body, '$.a_typ') AS a_typ,
  concat ('', SPLIT_INDEX (JSON_VALUE (body, '$.a_usr'), '@', 0)) AS `zone`,
  cast (from_unixtime (log_t, 'yyyyMMddHH') as bigint) AS `ahour`,
  cast (from_unixtime (log_t, 'yyyyMMdd') as bigint) AS `dt`
FROM
  game_log_source;
```

场景二：KPI统计

游戏UV

> 本场景统计每天每小时的游戏UV。
>
> 以server_date_day和game_id分组，然后与维表进行join扩展玩家信息，使用类似count(distinct user_id)

> filter (where reg_hour=0)的方法求得00：00—00：59时间段的游戏UV，从而统计每天每小时的游戏UV。

输入表

```
CREATE TABLE agent_login (
  user_id                       VARCHAR,
    user_name                     VARCHAR,
    gender                        VARCHAR,
    birth                         VARCHAR,
    age                           VARCHAR,
    game_id                       VARCHAR,
    game_name                     VARCHAR,
    channel_id                    VARCHAR,
    game_channel_id               VARCHAR,
  os_type                       VARCHAR,
  server_date_day               VARCHAR,
  reg_date                      VARCHAR,
    reg_hour                      BIGINT,
    ad_id                         VARCHAR,
    reg_via                       VARCHAR,
    dt                            VARCHAR
)WITH (
  type='datahub',
  ...
);
```

维度表

```
CREATE TABLE advertising (
  id                                  INT,
  ad_name                             VARCHAR,
  game_id                             INT,
  game_name                           VARCHAR,
  media_id                            INT,
  media_account_id                    INT,
  package_id                          INT,
  ad_resource_id                      INT,
  ad_media_params                     VARCHAR,
  admin_id                            INT,
  create_time                         TIMESTAMP,
  PRIMARY KEY (package_id,ad_media_params),
  PERIOD FOR SYSTEM_TIME
)WITH (
  type='rds',
  ...
);
```

输出表

```
 CREATE TABLE hour_uv(
`date`                          VARCHAR,
 ad_game_id                     VARCHAR,
 channel_id                     VARCHAR,
 package_id                     VARCHAR,
 ad_media_params                VARCHAR,
 hour_active_nuv_0              BIGINT,
 hour_active_nuv_1              BIGINT,
 hour_active_nuv_2              BIGINT,
 hour_active_nuv_3              BIGINT,
 hour_active_nuv_4              BIGINT,
 hour_active_nuv_5              BIGINT,
 hour_active_nuv_6              BIGINT,
 hour_active_nuv_7              BIGINT,
 hour_active_nuv_8              BIGINT,
 hour_active_nuv_9              BIGINT,
 hour_active_nuv_10             BIGINT,
 hour_active_nuv_11             BIGINT,
 hour_active_nuv_12             BIGINT,
 hour_active_nuv_13             BIGINT,
 hour_active_nuv_14             BIGINT,
 hour_active_nuv_15             BIGINT,
 hour_active_nuv_16             BIGINT,
 hour_active_nuv_17             BIGINT,
 hour_active_nuv_18             BIGINT,
 hour_active_nuv_19             BIGINT,
 hour_active_nuv_20             BIGINT,
 hour_active_nuv_21             BIGINT,
 hour_active_nuv_22             BIGINT,
 hour_active_nuv_23             BIGINT,
 create_time                    VARCHAR,
 via                            VARCHAR,
 media_id                       BIGINT,
 media_account_id               BIGINT,
 ad_resource_id                 BIGINT,
 game_id                        BIGINT,
 admin_id                       BIGINT,
 ad_id                          BIGINT,
 os_type                        VARCHAR
 )WITH (
   type='rds',
   ...
 );
```

业务代码

```
INSERT INTO hour_uv
select
server_date_day as server_date,
o.game_id,
o.channel_id,
```

```
o.game_channel_id,
o.ad_id,
count(distinct user_id) filter (where reg_hour=0) as hour_active_nuv_0,
count(distinct user_id) filter (where reg_hour=1) as hour_active_nuv_1,
count(distinct user_id) filter (where reg_hour=2) as hour_active_nuv_2,
count(distinct user_id) filter (where reg_hour=3) as hour_active_nuv_3,
count(distinct user_id) filter (where reg_hour=4) as hour_active_nuv_4,
count(distinct user_id) filter (where reg_hour=5) as hour_active_nuv_5,
count(distinct user_id) filter (where reg_hour=6) as hour_active_nuv_6,
count(distinct user_id) filter (where reg_hour=7) as hour_active_nuv_7,
count(distinct user_id) filter (where reg_hour=8) as hour_active_nuv_8,
count(distinct user_id) filter (where reg_hour=9) as hour_active_nuv_9,
count(distinct user_id) filter (where reg_hour=10) as hour_active_nuv_10,
count(distinct user_id) filter (where reg_hour=11) as hour_active_nuv_11,
count(distinct user_id) filter (where reg_hour=12) as hour_active_nuv_12,
count(distinct user_id) filter (where reg_hour=13) as hour_active_nuv_13,
count(distinct user_id) filter (where reg_hour=14) as hour_active_nuv_14,
count(distinct user_id) filter (where reg_hour=15) as hour_active_nuv_15,
count(distinct user_id) filter (where reg_hour=16) as hour_active_nuv_16,
count(distinct user_id) filter (where reg_hour=17) as hour_active_nuv_17,
count(distinct user_id) filter (where reg_hour=18) as hour_active_nuv_18,
count(distinct user_id) filter (where reg_hour=19) as hour_active_nuv_19,
count(distinct user_id) filter (where reg_hour=20) as hour_active_nuv_20,
count(distinct user_id) filter (where reg_hour=21) as hour_active_nuv_21,
count(distinct user_id) filter (where reg_hour=22) as hour_active_nuv_22,
count(distinct user_id) filter (where reg_hour=23) as hour_active_nuv_23,
dt,
reg_via,
cast(min(ad.media_id) as bigint),
cast(min(ad.media_account_id) as bigint),
cast(min(ad.ad_resource_id) as bigint),
cast(min(ad.game_id) as bigint),
cast(min(ad.admin_id) as bigint),
cast(min(ad.id) as bigint),
COALESCE((case when o.os_type = 'h5' then 'android' else o.os_type end),'android'
from agent_login AS o
LEFT JOIN advertising FOR SYSTEM_TIME AS OF PROCTIME() AS ad
on (o.ad_id=ad.ad_media_params and o.game_channel_id=ad.package_id)
where server_date_day=reg_date
group by server_date_day,o.game_id,o.channel_id,o.game_channel_id,o.ad_id,dt,reg_
```

新增角色累积收益总额

输入表

> 本场景统计新增游戏角色在不同时间段内（新增日、新增日和次日、新增日至新增第3日）产生的收益总额。
> 以reg_date和game_id分组，与维表join扩展玩家信息，通过TopN进行去重，然后比如使用sum(money) filter (where reg_date>=server_date_day-14)，得到新增角色后15天内的收益金额。从而得到新增游戏角色后不同时间段的收益总额。

```
CREATE TABLE `order` (
    `server`                        VARCHAR,
  os_type                         VARCHAR,
  create_time                     VARCHAR,
  update_time                     VARCHAR,
  money                           DOUBLE,
  user_id                         VARCHAR,
  id                              VARCHAR,
  channel_id                      VARCHAR,
  order_sn                        VARCHAR,
  status                          VARCHAR,
  game_id                         VARCHAR,
  game_channel_id                 VARCHAR,
  first_order_date                VARCHAR,
  server_date_day                 VARCHAR,  --角色登录时间
  reg_date                        VARCHAR,  --角色发布时间
  ad_id                           VARCHAR,
  via                             VARCHAR,
  reg_via                         VARCHAR,
  server_ts                       VARCHAR,
  game_name                       VARCHAR,
  package_name                    VARCHAR,
  dt                              VARCHAR
) WITH (
  type = 'datahub',
  ...
);
```

维度A表

```
CREATE TABLE advertising (
     id                                 INT,
  channel_id                          INT,
  game_id                             INT,
  game_name                           VARCHAR,
  media_id                            INT,
  media_account_id                    INT,
  package_id                          INT,
  package_name                        VARCHAR,
  ad_resource_id                      INT,
  ad_media_params                     VARCHAR,
  `type`                              TINYINT,
  status                              TINYINT,
  admin_id                            INT,
  create_time                         TIMESTAMP,
  update_time                         TIMESTAMP,
    PRIMARY KEY (package_id,ad_media_params),
    PERIOD FOR SYSTEM_TIME
) WITH (
  type= 'rds',
  ...
);
```

维度B表

```
CREATE TABLE advertising_divided (
    id                          INT,
  ad_id                       INT,
  media_id                    INT,
  media_account_id            INT,
  ad_resource_id              INT,
  game_id                     INT,
  package_id                  INT,
  ad_media_params             VARCHAR,
  ratio                       decimal(10,2),
  divide_date                 VARCHAR,
  create_time                 TIMESTAMP,
  update_time                 TIMESTAMP,
    PRIMARY KEY (package_id,ad_media_params,divide_date),
    PERIOD FOR SYSTEM_TIME
) WITH (
  type= 'rds',
  ...
);
```

输出表

```
CREATE TABLE total_revenue (
    `date`                        VARCHAR,
  ad_game_id                  VARCHAR,
  channel_id                  VARCHAR,
  package_id                  VARCHAR,
  ad_media_params             VARCHAR,
  pay_people_yet              BIGINT,
  pay_amount_yet              DECIMAL,
  pay_amount_1                DECIMAL,
  pay_amount_2                DECIMAL,
  pay_amount_3                DECIMAL,
  split_share_rate            decimal(10,2),
  create_time                 VARCHAR,
  via                         VARCHAR,
  media_id                    BIGINT,
  media_account_id            BIGINT,
  ad_resource_id              BIGINT,
  game_id                     BIGINT,
  admin_id                    BIGINT,
  ad_id                       BIGINT,
  os_type                     VARCHAR,
    PRIMARY KEY (`date`,ad_game_id,channel_id,package_id,ad_media_params,create_t
) WITH (
  type= 'rds',
  ...
);
```

业务代码

```
INSERT INTO total_revenue
select reg_date,o.game_id,o.channel_id,o.game_channel_id,o.ad_id,
count(distinct user_id) filter (where server_date_day=first_order_date) as pay_pe
cast(sum(money) as decimal),
--某日新增的角色中，每个角色在接下来1天内（新增日）为游戏带来的收入
cast(sum(money) filter (where reg_date>=server_date_day) as decimal)as pay_amount
--某日新增的角色中，每个角色在接下来2天内（新增日和之后的2日）为游戏带来的收入
cast(sum(money) filter (where reg_date>=DATE_SUB(server_date_day,1)) as decimal)
--某日新增的角色中，每个角色在接下来3天内（新增日和之后的3日）为游戏带来的收入
cast(sum(money) filter (where reg_date>=DATE_SUB(server_date_day,2)) as decimal)
cast(max(COALESCE(ra.ratio,0)) as decimal(10,2)),dt,reg_via,cast(min(ad.media_id)
from
(
  SELECT *,
    ROW_NUMBER() OVER (PARTITION BY order_sn ORDER BY server_date_day ASC) as rowN
  FROM `order`
) AS o
LEFT JOIN advertising FOR SYSTEM_TIME AS OF PROCTIME() AS ad on (o.ad_id=ad.ad_me
LEFT JOIN advertising_divided FOR SYSTEM_TIME AS OF PROCTIME() AS ra on (o.ad_id=
WHERE rowNum = 1 group by reg_date,o.game_id,o.channel_id,o.game_channel_id,o.ad_
```

游戏评论用户数

> 本场景按照三分钟维度的滚动窗口统计评论游戏的用户数。
> 用户评论游戏后产生日志数据，Flink对Json格式的日志数据进行解析并清洗，获取app_id、游戏评论时间`day`、游戏评论的用户id等信息，以app_id和`day`进行分组，通过三分钟的滚动窗口函数进行聚合，统计得到对应的游戏评论用户数。

埋点数据样例

```
{
    "app_id":"",
     "body":{
        "lid":"",
        "affcode":"",
    }
    "app_table":"",
    "log_t":"",

}
```

输入表

```
CREATE TABLE log_input (
  `message` VARCHAR,
```

```
  ts AS case when JSON_VALUE(`message`, '$.log_t') is NULL then TO_TIMESTAMP('197
  WATERMARK wk1 FOR ts as withOffset(ts, 180000)  --Watermark计算方法,偏移1分钟
) WITH (
  type='sls',
  ...
);
```

输出表

```
create table total_comments (
  app_id VARCHAR,
  comment_name VARCHAR,
  comment_type VARCHAR,
  kpi_type_val VARCHAR,
  comment_value bigint,
  `day` VARCHAR,
  createtime timestamp,
  PRIMARY KEY (app_id,comment_name,comment_type,kpi_type_val,`day`)
) with (
  type = 'rds',
  ...
);
```

业务代码

解析Json数据并进行清洗

```
CREATE VIEW user_session AS
SELECT CAST(TO_DATE(cast(now() as VARCHAR),'yyyyMMdd') as VARCHAR) as `day`,ts,
JSON_VALUE(`message`, '$.app_id') as app_id,
JSON_VALUE(JSON_VALUE(`message`, '$.body'), '$.lid') as lid,  --游戏评论的用户id
JSON_VALUE(JSON_VALUE(`message`, '$.body'), '$.affcode') as affcode
from log_input
where JSON_VALUE(`message`, '$.app_table') = 'user_session'
and JSON_VALUE(`message`, '$.body') is not null
and JSON_VALUE(`message`, '$.body') <> ''
and CHAR_LENGTH(cast(JSON_VALUE(`message`, '$.log_t') as varchar)) = 13
and JSON_VALUE(`message`, '$.app_id') is not NULL
and JSON_VALUE(`message`, '$.app_id') <> ''
and JSON_VALUE(JSON_VALUE(`message`, '$.body'), '$.affcode') is not null
and JSON_VALUE(JSON_VALUE(`message`, '$.body'), '$.affcode') <> 'PRESSURE_TEST'
and JSON_VALUE(JSON_VALUE(`message`, '$.body'), '$.lid') is not null
and JSON_VALUE(JSON_VALUE(`message`, '$.body'), '$.lid') <> '';
```

统计3分钟维度的评论次数

```
INSERT INTO total_comments
SELECT
app_id,
```

```
'comment_name' as comment_name,
'comment' as comment_type,
affcode as comment_type_val,
count(DISTINCT lid) as comment_value,
`day`,
CURRENT_TIMESTAMP as createtime
from  user_session
GROUP BY `day`,TUMBLE(ts, INTERVAL '3' MINUTE),app_id,affcode;
```

场景三：热门游戏TOP3

> 本场景是用于计算每天的热门游戏的排行榜。
> 在游戏商城前端下载页面进行埋点，将埋点数据同步至DataHub，以time和game_app分组，计算单天内每个游戏的总下载次数。对下载次数进行topn排序，得到下载次数最多的三个游戏作为最热门游戏。

输入表

```
CREATE TABLE source_table(
  game_app VARCHAR ,--游戏名称
  `time` VARCHAR    --时间（本场景为天）
)WITH (
  TYPE='datahub',
  ...
);
```

输出表

```
CREATE TABLE result_table(
Ranking BIGINT,
`time` VARCHAR,
game_app VARCHAR,
number BIGINT,
primary key(`time`,game_app)
)WITH (
  TYPE='rds',
  ...
);
```

业务代码

```
INSERT INTO result_table
SELECT
Ranking,
`time`,
game_app,
number
FROM (
```

```
  SELECT *,
     ROW_NUMBER() OVER (PARTITION BY `time` ORDER BY number desc) AS Ranking
  FROM (
        SELECT
       `time` AS `time`,
        COUNT(game_app) AS number,
        game_app
        FROM  source_table
        GROUP BY `time`,game_app
    )a
)
WHERE Ranking <= 3
```