FeedProxy & 金门串讲

杨张婧
商业推荐研发部

2020/09/08

目录

CONTENT

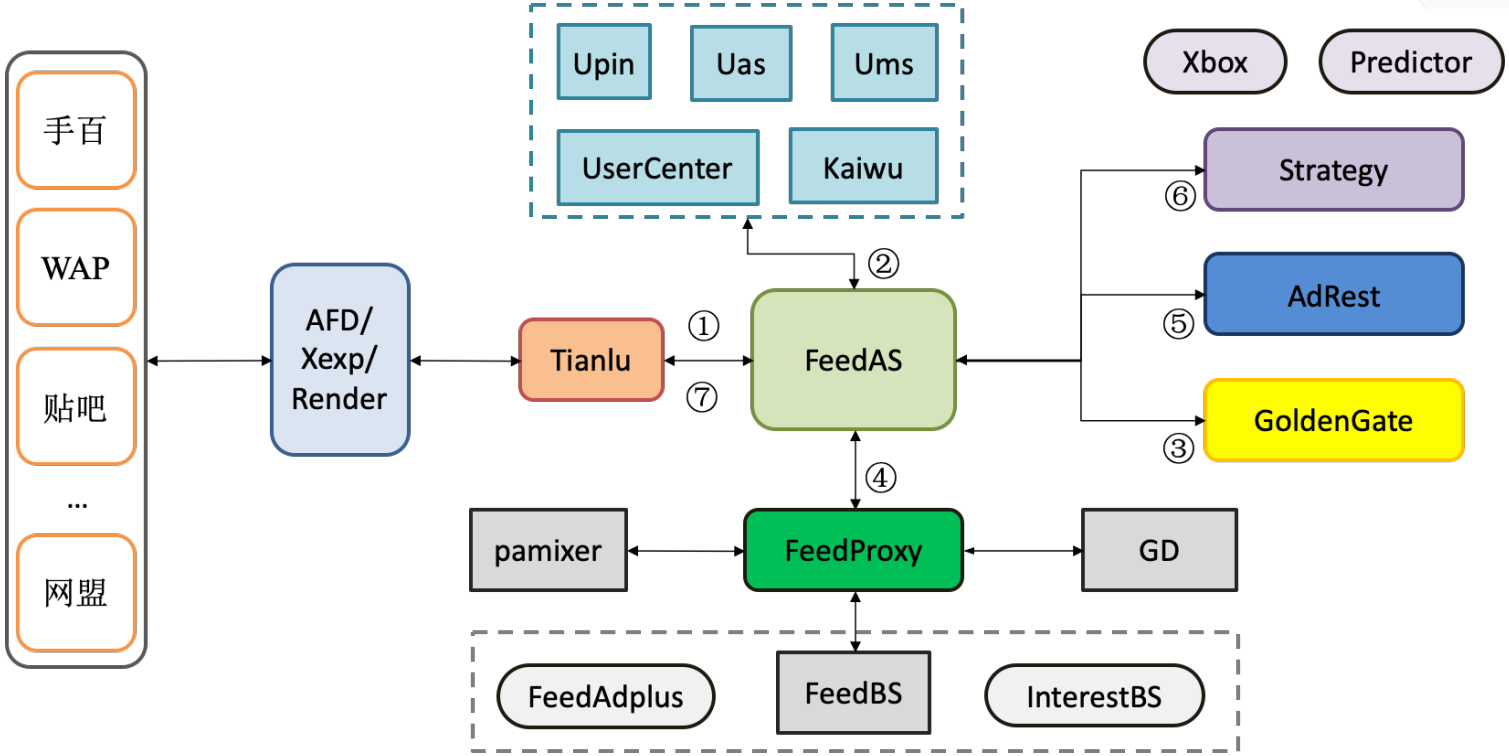
01 …… 系统简介

02 …… Baidu-rpc

03 …… Feedproxy

04 …… 金门

01 系统简介


手百
WAP
贴吧
...
网盟
AFD/
Xexp/
Render
Tianlu
①
⑦
FeedAS
②
Upin
Uas
Ums
UserCenter
Kaiwu
Xbox
Predictor
⑥
Strategy
⑤
AdRest
③
GoldenGate
④
pamixer
FeedProxy
GD
FeedAdplus
FeedBS
InterestBS

- **金门**：具有更普遍的适用场景
  - 抽象为通用推荐平台
  - 对外提供query推荐服务

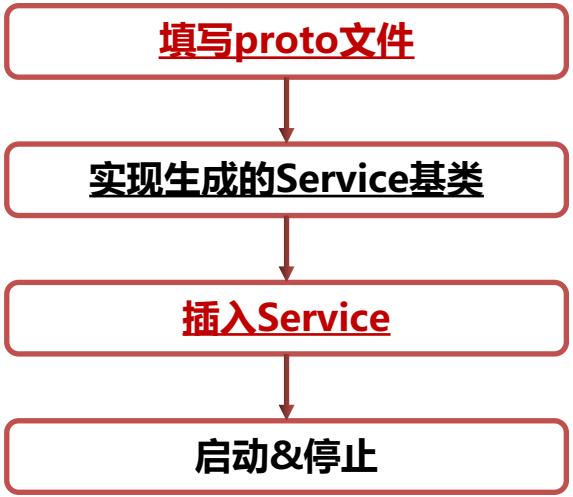
- **抽象模块**
  - 框架
  - 两个策略子框架
  - 公共策略库
  - 工具包

- **代码**
  - /im-goldengate/framework
  - /im-goldengate-deploy/feedgoldengate

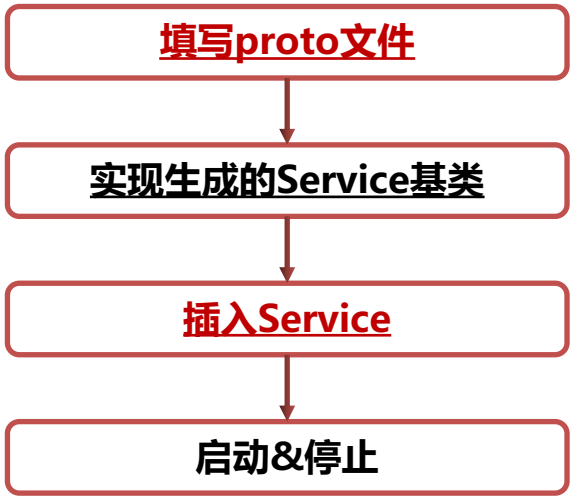
02 Baidu-rpc

- **RPC概念**
  - RPC把网络交互类比为“client访问server上的函数”
- **baidu-rpc**
  - baidu-rpc是访问和搭建百度内服务的统一RPC框架

- **Server基础实现**



- Server基础实现




```
service FeedProxyService {
    rpc FeedProxyRpc(FeedProxyRequest) returns (FeedProxyResponse);
}
```

```
void FeedProxyServiceImpl::FeedProxyRpc(google::protobuf::RpcController* cntl_base,
        const FeedProxyRequest* request,
        FeedProxyResponse* response,
        google::protobuf::Closure* done) {
```

```
if (server.AddService(&feedproxy_impl, baidu::rpc::SERVER_DOESNT_OWN_SERVICE) != 0)
```

```
if (server.Start(pd->port(), &options) != 0)
server.Stop(FLAGS_closewait_ms)
```

03 金门

main.cpp

作用：入口函数

- register_plugin：
  - register_query_mining_plugin： REGISTER_QUERY_MINING
  - register_query_strategy_plugin： REGISTER_QUERY_STRATEGY
- gflags & comlog：设置命令行参数gflags.conf与comlog日志配置comlog.conf
- 创建rpc server：
  - 添加GoldenGateServiceImpl服务
  - register_query_module
  - 添加KgServiceImpl服务
- 初始化ProcessData
- 启动Server
- 重载线程
- 停止 & 销毁数据

GoldenGate

作用：GoldenGateServiceImpl定义的服务

- parse request ：调用ParseRequest类的process方法，解析请求数据
  - parse_request_user_data
    - cntl->request_user_data();内容转换为string
    - im_request_header.ParseFromString(user_data_raw_string)
  - parse_ovlexp
  - parse_message
- do_process ：调用QueryModuleManager类方法
  - schedule
    - _parse_request
    - _input_extract
    - 插件执行_t_map[module_name]->process()
    - _pack_response
- post_process ： 打印日志

schedule

作用：核心函数

- _parse_request
- _input_extract：
  - InputExtractorManager::instance().process();
- _t_map[module_name]->process()
  - 执行所有注册插件的方法
- _pack_response
  - PackResponse::process
    - pack_response_user_data
    - pack_response_message

Baidu 百度

_t_map[module_name]->process()

_t_map

类名： TManager

- std::map<std::string, T*> _t_map
- register_T：关键函数
  - register_plugin： _t_map[name] = plugin
- register_conf：调用插件的对应函数
  - TMap::iterator it = _t_map.begin()
  - it->second->register_conf()
- Tmanager：父类
  - QueryModuleManager
  - QueryMiningManager
  - QueryStrategyManager

注册插件

函数：register_plugin

- QueryModuleManager
  - REGISTER_QUERY_MODULE
    - static QueryModuleRegister name##_register(#name, &name##_query_module);
      - QueryModuleRegister类
        - 构造函数：QueryModuleManager::instance().register_plugin(name, module)
- QueryMiningManager
  - REGISTER_QUERY_MINING
    - static QueryMiningRegister name##_register(#name, &name##_query_module);
      - QueryMiningRegister类
        - 构造函数： QueryMiningRegister ::instance().register_plugin(name, query_mining_plugin)
- QueryStrategyManager
  - REGISTER_QUERY_STRATEGY -> QueryStrategyRegister

register_query_module

创建rpc server：
添加GoldenGateServiceImpl服务
register_query_module
添加KgServiceImpl服务

函数：注册框架

- REGISTER_QUERY_MODULE(input_extractor, InputExtractorModule)
- REGISTER_QUERY_MODULE(query_mining, QueryMiningModule)
- REGISTER_QUERY_MODULE(query_strategy, QueryStrategyModule)

- InputExtractorModule：数据抽取
- QueryMiningModule： 基础挖掘
- QueryStrategyModule：策略管理

InputExtractorManager::process

作用：执行数据抽取插件

- 从_input_extractor_map中获取插件
- it->second(req_data)：执行插件

- InputExtractorManager只是一个框架，具体的执行内容是要看注册进来的extractor的函数如何执行的
- 具体的extractor函数通过REGISTER_INPUT_EXTRACTOR宏定义注册

金门
数据抽取：具体策略

extract_common.cpp	extract_original_query
	extract_baiduid
	extract_charge_name
	extract_deviceid
	extract_huichuan_multi_query
	extract_cuid
	extract_passport
	extract_title
	extract_tieba_title
	extract_content_title
	extract_article_with_update_time
	extract_sp_page_id
	extract_sp_app_id
	extract_sp_title
	extract_sp_tags
	extract_sp_category
	extract_sp_sub_category
	extract_sp_category_id
	extract_sp_sub_category_id
	extract_uc_unitid_clk_list

extract_upin_profile_query.cpp	extract_upin_profile_query
extract_kaiwu_query.cpp	extract_kaiwu_partial_query
	extract_kaiwu_qp_intent
	extract_kaiwu_query
	extract_kaiwu_wise_query
	extract_kaiwu_wise_query_with_timestap
	extract_kaiwu_feed_title_list
	extract_kaiwu_feed_tag_list
	extract_haokan_title_list
extract_upin_fcr_ss_info.cpp	extract_upin_fcr_ss_info
	extract_upin_fcr_gs_wise_ss_info
	extract_upin_fcr_gs_pc_ss_info
	extract_history_query_upin_wise_info
	extract_upin_app_list
extract_upin_intent_query.cpp	extract_upin_intent_query
extract_intent_model_query.cpp	extract_intent_model_origin_query
extract_upin_feed_session_title.cpp	extract_all_info_from_xbox
	extract_upin_feed_session_title
extract_upin_id_mapping_info.cpp	extract_upin_title_profile_info
	extract_upin_id_mapping_baiduid
	extract_upin_id_mapping_cuid

金门
数据抽取：具体策略

- 类别：
  - common 类插件，主要是提取original_query、baiduid、cuid、charge_name、device_id、title、tieba_title 等
  - kaiwu 类插件，主要提取kaiwu_query、kaiwu_patial_query、kaiwu_wise_query、kaiwu_feed_title_list
  - upin 类插件，主要是提取 upin_fcr_ss_info、upin_feed_session_title、upin_id_mapping_baidu\cuid 等
  - 其他类插件，这部分主要都是从 xbox 获取数据，例如 intent_model_query、history_query_upin_wise_info、rewite_titleid_query、kaiwu_video_title_session 等

- 一般调用ADD_QM_DATA
  - mutable_query_mining_data
    - 把数据放进：query_mining_data，它的数据类型是QueryMiningData
    - 定义于proto文件中
    - 以key-value的形式放入

QueryMiningManager::process()

作用：执行基础挖掘插件

- prepare_data
  - build_branch_map：对所有branch进行映射
  - prepare_current_branch_vec_by_groupid
  - 如果没有groupid执行下两句
  - find_current_desc： 从src_vec里找到第一个conf中存在的src_id，得到其对应的配置描述；对默认的default_srcid也来一遍上述操作
  - prepare_current_branch_vec：根据上一步得到的qm_hit_desc获取branch信息；判断分支是否有效、是否生效；拷贝分支
- Schedule
  - pre_process：获取当前branch向量，current_branch_vec；执行plugin的pre_process
  - do_process：执行plugin的do_process
  - post_process：执行plugin的post_process
- REGISTER_QUERY_MINING

金门

基础挖掘：具体策略

branch_name	
intent_xbox_first	cf_password_v2
ann_service_feed_session	feed_session
otpa	title_query
eureka	external_title
search_keyword	feed_app_list
query_mining_predictor_app_list	user_kg_xbox
original_query	sequence_word_title
session_query	rewrite_unitid_query
query_profile	ad_title_rewrite
trans_search_upin_7days	feed_title_session
trans_search_kaiwu_30days	truncation_query
intent_model_only_wide	intent_model_wanbo
sequence_word_merge	

金门

基础挖掘：具体策略

plugin_name
DmpQuery
PageAttentionQuery
ShortAttentionQuery
TiebaQuery
FeedAppList
VideoAttentionQuery
PredictorAsync
AppInfoQuery
Truncation
FeedTagQuery
OtpaPlugin
SearchQueryPlugin
AnnSearchPlugin
AnnServicePlugin
IntentXboxQuery

intent_xbox_query

XboxParallelRequestManager：
fill_request
parse_response

作用：请求xbox获取意图query

- pre_process
  - 开始请求xbox：has_request_xbox = true
  - 获取当前branch配置信息current_branch_vec
    - 解析自配置文件，通过prepare_current_branch_vec()
  - 获取当前意图xbox配置intent_xbox_params，为每个intent_xbox_params请求设置req_id：
    - 获取request_type
    - 从_xbox_req_map请求request_type【由REGISTER_XBOX_REQUEST_MAP注册进的】
    - 设置请求信息set_request_info：【branch_params、intent_xbox_params】
    - 注册请求register_request
  - 发送请求send_request：
    - 填充请求fill_request

intent_xbox_query

XboxParallelRequestManager：
fill_request
parse_response

作用：请求xbox获取意图query

- do_process
  - 接收xbox的响应数据recv_response
    - 解析响应parse_response
  - 把xbox回应的intent_xbox_item压入query_list，不能超过max_num个
    - intent_xbox_item在XboxRequest里被放入的，由REGISTER_XBOX_REQUEST_MAP注册进的

Xbox请求

IntentXboxRequest
WideQueryXboxRequest
UserKgXboxRequest
EurekaXboxRequest

- IntentXboxRequest
  - fill_request：获得key
    - 根据key_type获取key_list
    - 如果参数中有不为空的has_suffix，那么就为key加上后缀 “____”+suffix
    - 为handler添加这个构造key
    - 如果没有走else分支，为handler直接添加原始key
  - parse_response：获得截断后的query列表
    - 从handler得到raw_result，调用_get_intentxbox_result
    - 从conf得到truncate_num
    - 从raw_result抽出intent_xbox_result，填充p_query_info信息
    - 将p_query_info压到intent_xbox_item（存放于线程数据）

Xbox请求

IntentXboxRequest
WideQueryXboxRequest
UserKgXboxRequest
EurekaXboxRequest

- WideQueryXboxRequest
  - fill_request：获得key
    - 从history_query_upin_wise_info_size取出max_key_num个信息作为key放到去重队列里
  - parse_response：获得截断后的query列表
    - 得到truncate_num
    - 判断xbox数据是否获取成功，从intent_xbox_result中抽取结果
    - 填充p_query_info信息
    - 将p_query_info压到intent_xbox_item
- UserKgXboxRequest
    - fill_request：调用fill_request_kg2，key_type和aql_name用于设置handler
    - parse_response：调用parse_response_kg2，获得user_kg_list，放进td->p_response->mutable_user_kg2()

Xbox请求

IntentXboxRequest
WideQueryXboxRequest
UserKgXboxRequest
EurekaXboxRequest

- EurekaXboxRequest
  - fill_request：获得key
    - 因为插件eureka的intent_xbox_params没有deviceid，所以整段逻辑跳过
    - 从线程数据里得到cuid。得到user_info设置好id和类型为handle添加key
  - parse_response：获得截断后的query列表
    - 从handler得到raw_result，调用_get_intentxbox_result
      - 从conf得到truncate_num
      - 从raw_result抽出res信息，填充user_info信息

interact_with_ann_service

作用：与ANN交互

- pre_process
  - _prepare_query：获取ann_query_list
  - 获取query
  - mining_stage_rebuild_one打开，ann_cache存在，如果没有搜索缓存has_search_cache，则调用append_query_for_search(query_list)，如果_cache_info->flag[idx] == CACHE_STATUS_SEARCH_MISS，则将query放入ann_query_list，需要与ann服务交互，否则不需要交互
- do_process
  - mining_stage_rebuild_one为1，直接返回
- post_process
  - mining_stage_rebuild_one和use_ann_service都打开，执行_send_ann_request。返回的ret_query放入ret_query_list。
  - _parse_ann_response()
  - _set_query_to_mining：存放score小于相似度阈值similarity_thr的

feed_app_list

作用：通过applist获取query

- do_process
  - 获取input_tractor抽取的query_mining_data
  - upin_app_list中的每个app_name查表applist_query
  - 把查到的applist_query压入query_res
    - 因为query_res是vector，需要next_query为每个app_name分段(一个app可以对应多个query)
  - 把query放入p_query_info，压入query_list

dmp_query

作用：dmp_query

- do_process
  - 获取branch_params的max_num
  - 从plugin_params获取key_type_size
  - 从query_mining_data中用key_type查表得到dmp_query列结果
  - 调用fill_query_info函数
    - 为query_info填充branch_params的信息：branch、 rank、 weight
  - 将dmp_query填充进query_info，压入query_list

predictor_async

作用：从观星获取在线预估query

- pre_process：发送观星请求
  - int ret = _p_thrd_data->predictor.predict_async_send(plugin_params.predictor_phase_name());
- do_process：接收观星异步请求
  - _p_thrd_data->predictor.predict_async_revc(plugin_params.predictor_phase_name())
- post_process：直接返回

search_keyword

作用：search_keyword，获取历史query

- pre_process：
  - fill_exact_search_query
    - 从plugin_params得到exact_key_scan_limit、search_type信息
    - 获得input_tractor抽取的query_mining_data
    - 从upin_fcr_gs_wise_ss_info里获取query：最近8小时的用户search query
    - 从history_query_upin_wise_info里获取query：最近7天的用户upin数据
    - 从history_query_upin_wise_info里获取query：从最近30天的开物数据里取7天
  - 计算出wide_key_scan_limit值
  - 根据wide_query_table注册xbox请求并发送
- do_process：
  - 接收xbox的响应
  - 如果线程数据有need_filter_trade标记过滤
- post_process
  - pack_search_query_list：把search_query_list信息填到线程数据里

otpa

作用：otpa

- pre_process：
  - 请求xbox的三张表：
    - lal_xbox_table_name
    - rt_xbox_table_name
    - trade_xbox_table_name
- do_process：
  - 接收xbox的返回数据recv_response
  - pack_lookalike_list
  - get_trade_package
  - sort
  - merge_package：合并trade_package_list、lal_xbox_response、rt_xbox_response的以及pack_lookalike_list
  - 对otpa_package_list执行random_shuffle
  - pack_opta_package_list

truncation

作用：截断

- do_process：直接返回
- post_process：
  - 从branch_params获取max_num=900
  - 根据max_num截断query
  - 删除压缩query列表，query_list->compact

QueryStrategyManager::process()

作用：执行具体策略插件

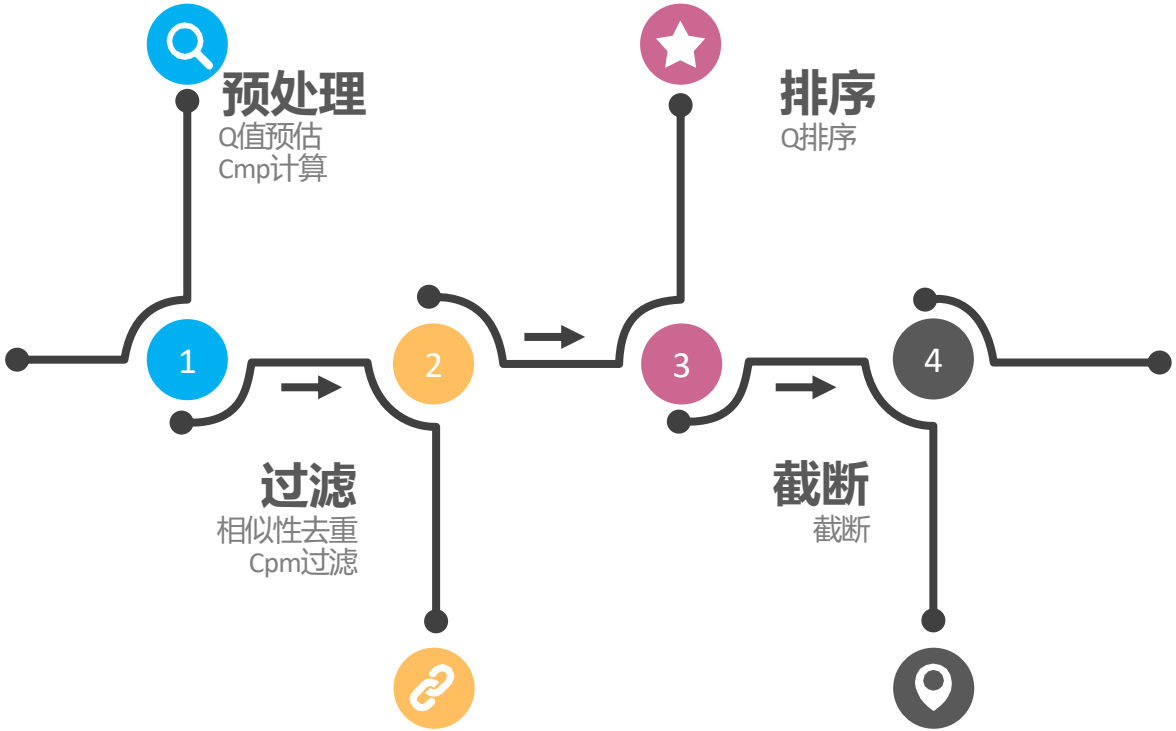
- prepare_data
  - prepare_current_strategy_vec_by_groupid
  - 如果没有groupid执行下两句
  - find_current_desc
  - prepare_current_strategy_vec
- Schedule
  - pre_process：执行plugin的pre_process
  - do_process：执行plugin的do_process
  - post_process：执行plugin的post_process

金门

策略管理：具体策略

strategy_name	plugin_name
query_source	query_source
feed_dedup_545_dedup	dedup
shoubai_app_cpm_and_tag	query_pvr_xbox
query_filter_by_cpm_id1	cpm_filter
query_filter_by_black_trade_id1	black_trade_filter
deep_intentq_cuid_v4	deep_intentq
interact_with_kg_shoubai	interact_with_kg
query_filter_by_intentq_1092	intentq_filter
feed_dedup_545_fre_control	dedup
deep_rsq_app_list	deep_rsq
deep_pvrq	deep_pvrq
feed_compute_545	compute_with_pvrq
query_sort_by_score_1092	query_sort
query_tag_filter_1092	query_tag_filter
max_intent_query_list_545	truncate_querylist

金门
策略管理：具体策略



query_source

作用：为query设置source信息

- do_process
  - query_list为0则直接返回
  - 创建source_map，key为query，value由branch决定
  - 为每个query设置source信息

```
feed_dedup_545_dedup
feed_dedup_545_fre_control
```

dedup

作用：去重

- do_process
  - query_list为0则直接返回
  - 若self_dedup生效
    - 执行self_dedup_query_list
      - 创建set：query_set
      - 查到重lazy_erase
      - 最后对结果compact
  - build_querylist_char_set：因为shown_dedup和upin_dedup为1，直接跳过
  - 从金门请求获得shown_info
  - 若shown_dedup生效
    - dedup_query_list
      - 计算string_similarity
      - 大于similar_thr去掉
    - 若upin_dedup
      - dedup_query_list

query_pvr_xbox

作用：使用query请求xbox，丰富query的信息

- do_process
  - 表名table_name设置为conf里的query_pvr_table_name
  - 新建xbox请求QueryPvrXboxRequest
    - 为请求设置set_truncate_num，来自conf
    - fill_request：线程数据中的query_list用query作为key【去重】
    - parse_response：
      - 得到返回的raw_result
      - rough_sort开关生效，从raw_result中抽出CpmTradeEpvqResult结果
      - 为query添加来自用CpmTradeEpvqResult结果的cpm、trade_id、post_epvq信息
      - user_preference_info_feed开关关闭
  - 新建请求QueryTagXboxRequest
    - fill_request：线程数据中的query_list用query作为key
    - parse_response：
      - 得到返回的raw_result
      - 从raw_result中抽出UserTagRes结果
      - 从抽出的结果中得到tag_info放到query的tag_list中

query_pvr_xbox

作用：使用query请求xbox，丰富query的信息

- do_process
  - 新建请求CuidToTagXboxRequest
    - fill_request：线程数据中的cuid作为key
    - parse_response：
      - 得到返回的raw_result
      - 从raw_result中抽出UserTagRes结果
      - 从抽出的结果中得到tag_info，limit信息放进user_tag里
  - 新建请求管理器XboxParallelRequestManager
  - user_tag_control
    - 注册三个请求
  - xbox_request_manager.wait_all_done

cpm_filter

作用：根据cpm信息截断query

- do_process
  - 从conf中得到cpm_thr
  - 没有explore_ratio
  - 遍历query_list，cpm小于cpm_thr的lazy_erase
  - query_list->compact

black_trade_filter

作用：截断在trade黑名单中的query

- do_process
  - 遍历query_list
  - 判断query的trade_id是否在black_trade_list，lazy_erase
  - query_list->compact

deep_intentq

作用：计算 intenq

- do_process
  - get_deep_intentq_vec
    - 新建请求DeepIntentqUserVecRequest
      - fill_request：用conf的deep_intentq_key_type请求user_req_val，user_req_val[0]作为请求的key
      - parse_response：从raw_result抽数据到user_vec_proto，设置user_sq以及user_vec
    - 新建请求DeepIntentqQueryVecRequest
      - fill_request：query太长的lazy_erase，以query作为key
      - parse_response：
        - query_sq、query_vec调整为raw_result大小，遍历raw_result，先将query_sq、query_vec清零，抽raw_result结果到query_vec_proto，若query_vec_proto权重大小和conf中deep_intentq_vec_len一致，设置query_sq、query_vec
    - 新建请求QueryRichnessXboxRequest：不被使用到
    - 新建请求管理器XboxParallelRequestManager
    - 用user_vec_table_name注册请求user_vec_request与query_vec_request
    - xbox_request_manager.wait_all_done

deep_intentq

作用：计算 intenq

- do_process
  - compute_softmax：核心是计算intentq
    - 如果user_vec数量不等于deep_intentq_vec_len直接返回
    - 设置min_size为query_vec和query_sq最小的size
    - mul为user_sq*query_sq
    - 若mul小于0跳过
    - xy+=user_vec[i]*query_vec[i][j]计算cosin
    - 计算y0，y1并判断有没有超过EXP_MAX_INPUT
    - 计算intentq：intentq = y1 / (y0 + y1);
    - intentq行业打平：用trade_id作为key，查表，intentq乘以系数
  - record_intentq_log：记录intentq个数，总和以及平均值
  - pack_deep_intentq_user_vec
    - 因为conf中 deepintentq_res_module为1，走predict first分支
    - 若有user_vec_online存这个，没有存user_vec
    - 结果存到线程的add_deepintentq_res

interact_with_kg

作用：请求kg意图

- pre_process：直接返回
- do_process：
  - _create_rpc_channel
  - _fill_kg_request
    - _fill_kg_req_base_attr
    - _fill_kg_req_mining_list
    - _fill_kg_ovlexp
- post_process：
  - WaitMethodDone
  - _parse_kg_response

intentq_filter

作用：根据intentq 过滤 query

- do_process：
  - 从conf读intentq_thr
  - 遍历query_list
  - 若intentq小于intentq_thr
    - 若在表_intentq_white_dict_copy中跳过
    - 不在就lazy_erase
  - query_list->compact

deep_rsq

作用：计算epvq

- do_process：
  - get_deep_rsq_weight
    - 以conf的deep_intentq_softmax_tag查表得到weight_{0,1}
    - 再append上softmax_tag得到去查表得到bias_{0,1}
  - get_deep_rsq_vec
    - 新建xbox管理器xbox_request_manager
    - 新建xbox请求DeepRsqQueryVecRequest
      - fill_request：以query_vec_tag作为key
      - parse_response：query_sq、query_vec调整为raw_result大小
      - 遍历raw_result，先将query_sq、query_vec清零
      - 抽raw_result结果到query_vec_proto
      - 若query_vec_proto权重大小和conf中deep_intentq_vec_len一致
      - 设置query_sq、query_vec
    - 填充请求信息、注册请求、wait_all_done
  - compute_softmax

deep_pvrq

作用：计算pvrq

- do_process：
  - get_deep_pvrq_vec
    - 新建xbox管理器xbox_request_manager
    - 新建xbox请求DeepPvrqQueryVecRequest
      - fill_request：以query名为key
      - parse_response：
        - query_vec调整为raw_result大小
        - 遍历raw_result，先将query_vec清零
        - 抽raw_result结果到query_vec_proto
        - 设置query_vec
    - 新建xbox请求DeepPvrqUserVecRequest
      - fill_request：得到cuid通过conf的deep_intentq_key_type调用seek_querymining_data，以cuid为key
      - parse_response：抽raw_result结果到query_vec_proto、设置user_vec
    - 填充请求信息、注册请求、 wait_all_done
  - compute_softmax： default 沿用intentq配置

deep_pvrq

作用：计算pvrq

- do_process：
  - get_deep_pvrq_vec
    - 新建xbox管理器xbox_request_manager
    - 新建xbox请求DeepPvrqQueryVecRequest
      - fill_request：以query名为key
      - parse_response：
        - query_vec调整为raw_result大小
        - 遍历raw_result，先将query_vec清零
        - 抽raw_result结果到query_vec_proto
        - 设置query_vec
    - 新建xbox请求DeepPvrqUserVecRequest
      - fill_request：得到cuid通过conf的deep_intentq_key_type调用seek_querymining_data，以cuid为key
      - parse_response：抽raw_result结果到query_vec_proto、设置user_vec
    - 填充请求信息、注册请求、 wait_all_done
  - compute_softmax： default 沿用intentq配置

compute_with_pvrq

作用：计算query的score

- do_process：
  - 从conf得到weight_intentq、weight_cpm、weight_composed、t_cpm
  - epvq_compute生效，从conf得到t_epvq
  - compute_with_pvrq生效，从conf得到weight_pvrq、weight_epvq
  - 设置word_profile_tmp
  - 分割word_profile_tmp得到word_profile_list
  - 计算每个query的score
  - (*it)->score = (*it)->weight * ((*it)->intentq * weight_intentq + (*it)->epvq * weight_epvq + (*it)->pvrq * weight_pvrq)

query_sort

作用：按照score为query排序

- do_process：
  - conf中query_sort_by_rank没有：返回
  - conf中query_sort_by_score有：为query排序
    - std::stable_sort(query_list->begin(), query_list->end(), compare_score);

query_tag_filter

作用：根据tag过滤query

- do_process：
  - 从conf中得到truncate_num
  - 用线程数据中user_tag设置tag_info
    - tag_info[it->first] = truncate_num * it->second;
  - 如果tag_info中查到query的某个tag在，并且map结果小于等于0，则过滤该query

truncate_querylist

作用：截断query

- do_process：
  - 从conf得到truncate_num
    - truncate_num : 120
  - 如果count大于truncate_num就会被lazy_erase

tail_query_explore

作用：末位query探索机制

- do_process：
  - 从conf得到explore_t、truncate_num、max_expand_word_num
  - 将max_explore_num设置为max_explore_num与query_list_size - truncate_num更小的那个
  - 设置keep_query_num = truncate_num - max_explore_num
  - 若query_list_size <= truncate_num直接返回
  - to_which_model: 1，是INTENTQ_TYPE，以intentq作为query的score
  - 如果query有intentq，final_score设为score^explore_t
  - 累计final_score到sum_score
  - 计算前n个query的final_score占总和的百分比
  - 随机产生prob，选出最接近该概率的query_score_list的select_index
  - 遍历query_list如果count超过keep_query_num，explore_index也不在select_index中则删除该广告

04 FeedProxy

Main.cpp

说明：入口函数

- flags & comlog：命令行参数配置和comlog日志配置
- 初始化ProcessData
- register_module
- 重载线程
- 创建rpc server
  - 添加FeedProxyServiceImpl服务
- 启动server
- 监测exit信号后退出

FeedProxyRpc

作用： FeedProxyServiceImpl定义的服务

- 核心是执行WorkFlow::instance().do_process函数，送入参数cntl，request，response
- 上述执行完就是收尾工作
  - 发送do_process的耗时给feedas
  - 发送response：done->Run()
  - 处理日志
  - notice_log
  - monitor监控
  - fengsui日志
  - 清空线程结束

WorkFlow::instance().do_process

作用： FeedProxyRpc核心函数

- 初始化线程数据，填充启动时间，填充请求td->request
- parse_request：解析请求
- data_prepare：设置需要请求pa_adplus模块，得到本次PV的pid
- interact_with_downstream：与下游交互
- product_strategy_process：产品策略处理
- pack_response：打包响应

parse_request

作用： WorkFlow::instance().do_process的一环

- set_current_time
- parse_request_user_data
  - 获得user_data_raw_string
  - 为线程数据设置qid
  - 设置ovlexp_binary
- parse_request_message
  - has_feed_info ：没有feed信息就返回
  - has_pamixer_info ：没有闪投信息就返回
  - has_common_info：parse_common_info
  - has_feed_info：parse_feed_info
    - 有广告质量等级低中高的划分
  - has_freq_control_info：parse_freq_control_info
    - 解析一系列的频控信息到线程数据
  - has_upin_info：parse_upin_info
  - has_geed_info：parse_geed_info
    - 很简短，就根据信息是否存在设置了gd_flow_type以及has_place_id
  - has_cache_info：parse_cache_info
- parse_fengsui_info

interact_with_downstream

作用： WorkFlow::instance().do_process的一环

- ModuleMgr* module_mgr = ModuleMgr::get_instance();
- module_mgr->run_all_modules();
  - BaseModule *p_module = _module_array[i];
  - p_module->**preprocess**()
  - p_module->**do_process**()

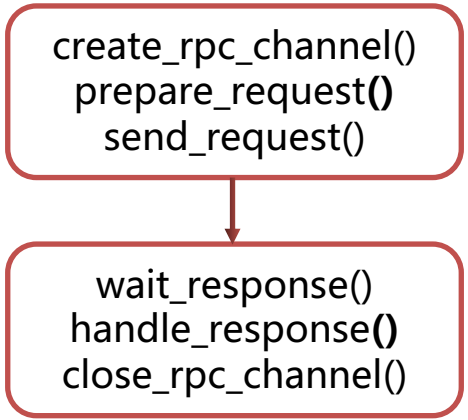
- 模块的添加
  - REGISTER_MODULE
    - ModuleCarrier::register_module
      - ModuleMgr::get_instance()->add_module
        - _module_array[pos] = p_module

```
p_module->preprocess()
p_module->do_process()
```

interact_with_downstream

作用： WorkFlow::instance().do_process的一环

- Preprocess：前处理
  - 创建rpc频道，解析请求，发送请求
- do_process：执行过程
  - 等待请求的模块结束，调用handle_response



下游Feedbs模块

interact_with_downstream

作用：下游模块

- Preprocess：
  - create_rpc_channel_new
  - prepare_request
    - set_feedbs_request
      - 填充proto中需求的字段
    - 设置cntl
  - end_request_new
- do_process：
  - wait：等待返回
  - handle_response
    - 记录bs_ip
    - 添加烽燧交互信息
  - parse_feedbs_response
    - 参考proto

product_strategy_process

作用： WorkFlow::instance().do_process的一环

- preprocess_advlist ：前处理广告列
  - 完成一些去重set的填充 ：bs_unit_set存放BS里没有重的，adplus_unit_set存放adplus里没有重的，dedup_unit_set，存放adplus和bs重复的
  - 添加广告列到ori_advlist中，包括bs_tg1_advlist、bs_tg2_advlist、adplus_advlist、pa_adplus_advlist、store_advlist、geed_advlist
  - 记录烽燧日志
  - 将ori_advlist，clone到simulation_original_advlist

product_strategy_process

作用： WorkFlow::instance().do_process的一环

- traverse_src_plugins：src级别的插件
  - 生成策略插件的管理器实例，StrategyPluginManager* plugin_manager = StrategyPluginManager::instance()
  - 调用src_id策略级插件 plugin_manager->run_src_plugins(src_id, td->ori_advlist) != 0
    - 读取策略文件的配置stra_desc
    - 得到插件配置plugin_desc
    - 判断插件是否alive
    - 判断src_id是否命中策略
    - 得到策略函数
    - 根据src_id得到策略的配置参数
    - 执行策略：function.func(src_id, paramsP, &adv_list, NULL)

product_strategy_process

作用： WorkFlow::instance().do_process的一环

- build_advlist：将ori_advlist按产品线拆分到不同的advlist中
  - 根据不同的product_id，将ori_advlist中广告分到不同的队列product_advlist
  - 标志mark，判断之前出现过没有
  - 如果productid_vec有，直接填充到对应的队列
  - 如果没有，新增对应的product_id和对应的队列（不同的广告队列不能超过MAX_ADVLIST_NUM【16】个）

product_strategy_process

作用： WorkFlow::instance().do_process的一环

- traverse_product_plugins：执行product级别的策略
  - 生成策略插件的管理器实例 StrategyPluginManager* plugin_manager = StrategyPluginManager::instance()
  - 对productid_vec中不同product_id对应的广告队列，分别执行run_product_plugins
    - 得到策略配置stra_desc，得到插件配置plugin_desc
    - 判断流量是否需要分level，以及当前plugin是否需要分level
    - 如果productid和level在插件的cond_group和cond_pid中，可以执行
    - 获取插件函数
    - 获取插件的配置参数
    - 运行插件

product_strategy_process

作用： WorkFlow::instance().do_process的一环

- merge_advlist：将过滤阶段后的advlist重新merge到ori_advlist中，以供打包返回上游
  - 清空ori_advlist
  - 遍历不同的产品线id，productid_vec
  - 将不同产品线广告列加入ori_advlist： td->ori_advlist.append_advlist(td->product_advlist[i]);

product_strategy_process

作用： WorkFlow::instance().do_process的一环

- pack_response：打包返回结果
  - pack_response_user_data：打包返回用户数据
    - 序列化，设置cntl内容
  - pack_response_message：打包返回消息
    - 对每一个广告：pack_one_res(response, cur_item)
    - 判断广告类型,为adv_item设置信息
  - pack_share_info
    - set_interest_ids
    - set_user_profile_index
    - 每一个term_list：add_term_list
  - pack_shitu_info
    - set_bs_tg1_res_num
    - set_bs_tg2_res_num
    - set_pa_res_num

src策略

作用： src_id级别的策略

- interact_with_bcsvr_send_async
  - send_bc_request(td->_budget_context, advlist)
    - 设置budget_request、 budget_response
    - 为请求设置id set_requestid(td->qid)
    - build_budget_id_set：总消费预算
      - 获取预算控制配置
      - 记录流量级的budget id
      - 添加产品粒度 mt_pt id
      - 记录产品级的budget id
      - 对流量分级
      - bes_multi_charge_contorl_budget
    - 添加到budget_id_fields
    - 分别以user_id、plan_id，进行去重判断，添加进budget_request中
  - budget_client.bbs_talk_send(&budget_request, &budget_response, &context->bc_td);
  - td->_budget_request_adv_num = advlist->size()

product策略

作用： product级别的策略

- pid == cond_pid ||((pid / PRODUCT_DIV_BASE_NEW) * PRODUCT_DIV_BASE_NEW) == cond_pid
  - const uint32_t PRODUCT_DIV_BASE_NEW = 1000

- 1000 : Feed
  - 1001 : feed shoubai
  - 1002 : feed wap
  - 1003 : feed shoubai-app-pb

FeedProxy
具体策略

plugin_name		
adv_add_trade	ue_related_info_filter	feedbs_adv_matchtype_sort_and_truncate_new
sort_and_truncate_front	feedbs_adv_same_user_idea_limit	product_truncate
set_global_params	advlist_request_cache	set_idea_blacklist_status
deviceid_filter	ad_force_src_ids_filter	video_solo_filter
bes_dimension_filter	user_dislike_userid_filter	feedbs_idea_dedup
adv_add_brand_subject	user_dislike_trade2_filter	report_discript_trade_filter
adv_add_old_trade	user_dislike_filter	high_freq_dislike_filter
adv_quality_show_probility_control	user_dislike_trade1_filter	interact_with_xbox
adv_control_tag_extract	freq_control	query_sim_filter
user_control_ratio_tag_filter	ad_risk_level_control_new	user_dislike_brand_filter
tob_adv_filter	ad_region_blacklist_filter	xbox_freq_control
freq_control_new	interact_with_bcsvr_recv_and_parse_async	installed_app_filter
common_conf_filter	vbid_filter	adv_title_normalize
overseas_user_blacklist_filter	richq_control_v2	

adv_add_trade

作用： 为广告添加trade信息

- 对送入的广告列，每一个用对应的adv->user_id查表user_trade_new_dict
- 获得trade1，trade2以及trade3列表（MAX_TRADE3_NUM=8）

set_global_params

作用： 消费控制相关

- 设置消费控制跳过的ftype（从配置到线程数据）
- 根据src_id设置需要过滤的deviceid
- 解析feedbs返回的广告队列，如果有unit_add_time，设置新广告new_adv_flag标识

deviceid_filter

作用： 对于无deviceid的情况进行过滤

- 根据上一个插件set_global_params设置的is_deviceid_filter_src_id，判断要不要直接返回
- 根据请求的common_info获取device_id
- 如果device_id>0当前请求有，没必要过滤。
- 根据ACTIVE_TRANS_TYPE == (*iter)->trans_type && OCPC_BID_TYPE == (*iter)->bid_type，如果满足就删掉该条广告

bes_dimension_filter

作用： 根据性别、年龄、用户机型的维度过滤广告

- 获得mobile_model_name, age, gender, cmatch, tu_name信息
- 根据不同的组合查表（bes_dimension_filter_dict、bes_dimension_filter_userid_dict），只要能找得到就从队列里删除广告

adv_add_brand_subject

作用： 获取广告主品牌

- 对每条广告，用res_subject替换原有的subject，原本为null则不替换
  - PD()->get_feed_userid_subject_map().seek((*iter)->user_id, &res_subject)

adv_add_old_trade

作用：查词典获取广告主行业

- 对广告队列的每条广告，用user_id查表“user_trade_dict”，得到res_trade
  - adv->ecom3_trade1 = res_trade.trade1();
  - adv->ecom3_trade2 = res_trade.trade2();

adv_quality_show_probility_control

作用：根据广告质量控制展现概率

- 获取广告的质量等级，如果是BAD和LOW，把cur_pv_{0,1}_level_max_cpm分别改为
  - std::max((*iter)->cpm_transfer, cur_pv_{0,1}_level_max_cpm)
- 归一化cpm作为cur_pv_0_level_max_prob概率，如果
  - ADV_QUALITY_BAD < params->adv_quality_show_prob_t_size()
  - 以BAD广告作为广告展现系数，对cur_pv_0_level_max_prob进行指数调整
- 获取每条广告的质量等级，如果等级低于展现概率size就要进行调整。如果是BAD和LOW，广告展示概率门限就要取当前计算和之前计算出最小的那个
- 如果随机数大于adv_show_prob_thr，就把该条广告过滤掉

adv_control_tag_extract

作用：查词典获取广告主行业

- 在广告的control_ratio_tag_dedup里插入trade1_str与trade2_str

user_control_ratio_tag_filter

作用：过滤特定行业的广告

- 线程数据中的user_control_ratio_tag_dedup
- 如果出现在上一个插件adv_control_tag_extract设置的control_ratio_tag_dedup中，就删掉这条广告

tob_adv_filter

作用：tob广告过滤

- 新建三个set：match_type_set、 mining_type_set、 toB_ssg_trade2_set
- 如果配置参数中有has_tob_control_miningtype_list，并且分别有tob_control_matchtype_list、tob_control_miningtype_list、 tob_ssg_trade2_list，就将读到的配置（string）转换为set
- 遍历广告列，如果（不在match_type_set或mining_type_set）并且 不在toB_ssg_trade2_set ，就删掉

freq_control_new

作用：频控

- 调用freq_control
  - 通过get_freq_control_src_conf(src_id);得到src_conf
  - is_in_retarget_src_list判断src是否在retarget列表里
  - enable_game_expansion，分别通过UNITID_DIM，PLANID_DIM，USERID_DIM查表，找到了可以设置广告中的cpm_delta为cpm_delta
  - retarget_adv_ajust_bid
  - prog_freq_control_fix，根据广告的is_program_idea信息判断跳过正常频控与否
  - ocpc_deep_freq_control_dedup，如果不在retarget_jump_freq_user_set，在ocpc_deep_user_dedup_set中跳过
  - 各种去重...

common_conf_filter

作用：通用配置过滤

- 通过查表判断用黑名单过滤还是白名单过滤
- common_conf_filter_by_white_dict
  - 根据common_filter_data信息创建一系列的set，对每条广告看看有没有一个信息在set中中了，没有最后就删掉
- common_conf_filter_by_black_dict
  - 创建一系列set，但是是广告信息中一个字段就删掉

overseas_user_blacklist_filter

作用：海外用户黑名单过滤

- 根据adv->user_id查表OVERSEAS_USER_BLACKLIST（海外用户黑名单表），如果在黑名单中就删除该广告

ue_related_info_filter

作用： dislike 行业、实体过滤

- 新建4个黑名单列表set，dislick_trade2,eshow_control_trade2,ums_user_grade_trade2,user_dislike_entity
- 从request->common_info()中获取common_info.ue_related_info(i).{type(),.tags(j)}的信息填充三个set
- 如果广告里有need_retarget_adv_adjust_bid就可以跳过filter
- 看广告的trade2是否在前三个set中，是就删除广告。看广告实体非空以及是否在最后一个set中，是就删除

ad_force_src_ids_filter

作用： plan在srcid维度白名单过滤

- plan_id查表ad_force_src_ids_dict
- 找到，如果当前的src_id不在查到结果的src_id中，则可以投放，否则删除
- g_switches_on(add_user_blacklist) 打开，&& td->is_userid_blacklist_filter，user_id查表an_tou_black_list，查到删除
- ad_force_src_id打开，对于强制明投srd_id，如果词表中没有，则直接过滤，保证明投广告位仅投放词表中的plan

feedbs_adv_same_user_idea_limit

作用：相同创意过滤

- 新建两个map，same_user_idea_limit，video_same_user_idea_limit。根据不同的维度进行quota的设置，比如query，package，intent，lbs等。video_same_user_idea_limit也同理=> 对两个map进行quota对配置
- add_auto_targeting_matchtype开关打开了，输出same_user_idea的quota设置的limit信息
- enable_same_user_quota_balance关闭了，enable_same_user_quota_balance那map也就为空，for循环不执行
- 对每个广告，根据是否是video_adv，走不同的处理逻辑。以video_adv广告为例子，维护一个video_same_user_idea_cnt的map，以match_type以及user_id为key，获取数量。如果数量大于quota的限制，就删除该广告
- video_match_type_user_idea_cnt >= video_match_type_same_user_idea_limit
- 除了match_type，最后还要看total有没有超过limit

advlist_request_cache

作用：cache idea、 unit过滤

- advlist->sort(compare_adv_by_matchtype_priority_and_cpm);
  - compare_adv_by_matchtype_priority_and_cpm，用matchtype优先级和cpm比较广告大小，先比matchtype_priority，同级再比cpm
  - get_ocpc_cpm_by_bsroiq，计算cpm乘上调整系数
- enable_random_select_idea
- 如果idea_id命中_cache_ideaid_set，unit_id命中_cache_unitid_set，就删掉广告

user_dislike_userid_filter

作用：user_id维度dislike过滤

- 如果广告的user_id在td->user_dislike_userid_flt_set里，就删除广告

user_dislike_trade2_filter

作用：trade维度dislike过滤

- 建立trade2_blacklist，通过便利user_dislike_userid_dedup_set去重集合，用userid去查表USER_TRADE_NEW_DICT，把对应结果中的trade2插入到set里
- 如果广告的trade2在黑名单trade2_blacklist中，删除广告
- filter_no_trade_adv_on_dislike_stra打开，如果广告trade2是DEFAULT_TRADE且user_dislike_userid_dedup_set大于0，删除广告（for no trade info）

user_dislike_filter

作用：user_id维度dislike过滤

- filter_dislike_adv_permanent打开，如果cuid_permanent_dislike_found_result且user_permanent_dislike_filter结果为真，则扩产日志
- user_permanent_dislike_filter：
- filter_dislike_ideaid_permanent打开，idea_id在cuid_permanent_dislike.idea_list中，删除广告
- filter_dislike_brand_sign_permanent打开，brand_sign在cuid_permanent_dislike.brand_sign_list中，删除广告
- 如果idea_id在user_dislike_idea_dedup_set中，删除广告
- filter_dislike_adv_by_simi打开，如果广告与user_dislike_idea_dedup_set相似，也删掉

freq_control

作用：频控

- 用src_id 得到src_conf
- 获得从conf读retarget_src_list，判断src_id是否在这里
- 如果以ocpc进行竞价且是ocpc第二阶段的，如果unit_id，plan_id，user_id任意一个在对应词典中找到，则为广告设置cpm_delta
- 如果user_id在retarget_jump_freq_user_set中且不在ocpc_deep_user_dedup_set中，且_is_in_retarget_adv_ajust_bid_src_list不为空，则need_retarget_adv_adjust_bid设为真，need_retarget设为真【需要重定价】
- 如果is_program_idea为真【是程序化创意】跳过频控，否则正常频控【鼓励使用程序化创意】
- 如果user_id在retarget_jump_freq_user_set且不在ocpc_deep_user_dedup_set，且is_in_retarget_src_list不为空，则跳过ocpc_deep_freq_control
- enable_pa_white_entity打开，如果ad_type是PA_PRODUCT_AD_TYPE或PA_CAMPAIGN_AD_TYPE【是闪投广告或冠军广告】，is_store_adv或is_estore_adv，执行freq_control_winfo_dedup，成功该广告过
- ...
- 后面的以此类推，判断广告中了哪条执行完是true的（说明在子函数里已经删掉了），就continue到下一条广告

user_dislike_trade1_filter

作用：用户维度对一级行业的dislike次数进行广告过滤

- 如果广告在user_dislike_trade1_set中，就删掉

ad_risk_level_control_new

作用：风险等级过滤

- 对于非合约广告，通过user_id、entity_res.entity_id()、es_trade.trade2()三个维度，分别去查表，通过ad_risk_level_filter判断为true就能删除广告

ad_region_blacklist_filter

作用：区域黑名单过滤

- userid && pid || userid && cid查表AD_REGION_BLACKLIST命中

interact_with_bcsvr_recv_and_parse_async

作用：消费控制，主要包括消费速度和各个cmatch的消费比例控制

- _budget_parse_adv_num加上当前广告列数
- 若_budget_response_status为0，重新判断，设置_budget_response_status
- 在预算控制中执行分cmatch的预算控制 ：budget_id_budget_control
  - 如果当前budget-id为样式产品级别的budgetid且 当前广告不属于样式转置产品(如不是三图)范畴，则跳过当前budgetid的消费控制逻辑
  - 选择只投详情页的广告跳过详情页流量上的预算控制策略
  - plan下的消费在当前budget_id下已经超出ratio，删除广告
  - 在预算控制中执行分cmatch的消费控制 ：budget_id_charge_control
  - 如果当前budget-id为样式产品级别的budgetid且 当前广告不属于样式转置产品(如不是三图)范畴，则跳过当前budgetid的消费控制逻辑
  - 选择只投详情页的广告跳过详情页流量上的预算控制策略
  - 判断budget_id是否在_bes_quality_level_budget_id_set中，在的话不跳过消费控制逻辑
  - 当前budget_id下消费占比已经超出ratio，删除广告
  - 如果ocpc会有放量，不超过的ocpc广告不删除

vbid_filter

作用：超投控制

- enable_realtime_overcharge打开，调用realtime_overcharge_ctrl_flt，如果flt_res为真，删广告继续。
- 按plan和user消费情况得到show_prob，以1-show_prob的概率过滤
- 修改anti_reason_bid，如果adv->anti_reason_bid == 0 || adv->anti_mod_time == 0，跳过该广告
- 修改now_bid，用bsroiq调整。如果ow_bid <= adv->anti_reason_bid，跳过该广告
- 修改show_prob，show_prob >= 1.0跳过广告
- 计算is_hit，通过show_prob_filter_fun，如果is_hit==1,删除广告

richq_control_v2

作用：丰富度控制

- step1 : 按subject划分广告队列
- step2： 同subject内按照cpm排序
- step3 : 构建并查集，相同subject内的元素构建并查集
- step4： 相同subject内的元素保留前same_subject_adv_cnt_limit个

product_truncate

作用：截断

- 设置product_truncate_num为params->l1_time_trunc_num() || params->l2_time_trunc_num() 个
- 根据product_truncate_num设置不同quota，截断广告
- 存储product_truncate_num到_pid_shoubai_truncate_num

feedbs_adv_matchtype_sort_and_truncate_new

作用：广告排序与截断

- 阶段零：提取刷次 cache，advlist_request_cache
- 阶段一：排序，compare_adv_by_cpm
- g_switches_on(get_prog_mtq_from_bs) && g_switches_on(prog_adv_sort_by_mtq)失败跳过
- 阶段二: 按matchtype选取广告 ，获得matchtype和video的quota
- 阶段三：高优队列选取广告
  - 1. bes单选扶持
  - 2. 明示扶持
  - 3. 激励视频扶持
  - 6. 游戏客户支持
  - 7. 详情页相关扶持
  - 9. merge刷次cache
- 阶段四：截断&再排序
  - is_selected就删掉

THANKS!