|  |  |
| --- | --- |
| l文档编号 | IAAS-DDS 002 |
| 文档版本 | 1.0 |
| 拟制人 | 段彬彬 |
| 日期 | 2016.10.21 |

**EayunCloud公有云业务管理平台**

**（回源流量统计设计）**

**概要设计说明书**

变更说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **变更位置** | **变更说明** | **作者** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目　　录

[变更说明 2](#_Toc465343398)

[目　　录 3](#_Toc465343399)

[概述 4](#_Toc465343400)

[业务分析 4](#_Toc465343401)

[回源流量统计需求整理 4](#_Toc465343402)

[ECSC 4](#_Toc465343403)

[ECMC 5](#_Toc465343404)

[计费模块 8](#_Toc465343405)

[若干问题确认及设计原则 8](#_Toc465343406)

[回源流量统计延时问题 8](#_Toc465343407)

[如何从OBS下载流量中剔除回源流量问题 9](#_Toc465343408)

[UpYun统计回源流量API 9](#_Toc465343409)

[涉及功能模块开发 11](#_Toc465343410)

[计划任务 11](#_Toc465343411)

[CdnBacksourceDetailGatherJob 11](#_Toc465343412)

[ObsDetailSummaryJob 11](#_Toc465343413)

[ObsSortDetailSummaryJob 12](#_Toc465343414)

[ObsDetailMonthSummaryJob 13](#_Toc465343415)

[CdnDetailGatherJob 15](#_Toc465343416)

[模块功能 15](#_Toc465343417)

[计费接口 15](#_Toc465343418)

[ECSC 17](#_Toc465343419)

[ECMC对象存储总览 20](#_Toc465343420)

[ECMC客户详情 30](#_Toc465343421)

[ECMC-CDN服务 37](#_Toc465343422)

# 概述

原版本中我们采集到的OBS下载流量中包含着一部分回源流量无法分离，因此给用户计算价钱时，实际多算了一部分钱。新版本中， 可通过调用UpYun的API，获得域名一段时间内的回源流量，因此需要对原版中中涉及到OBS下载流量的地方均要剔除掉回源流量。

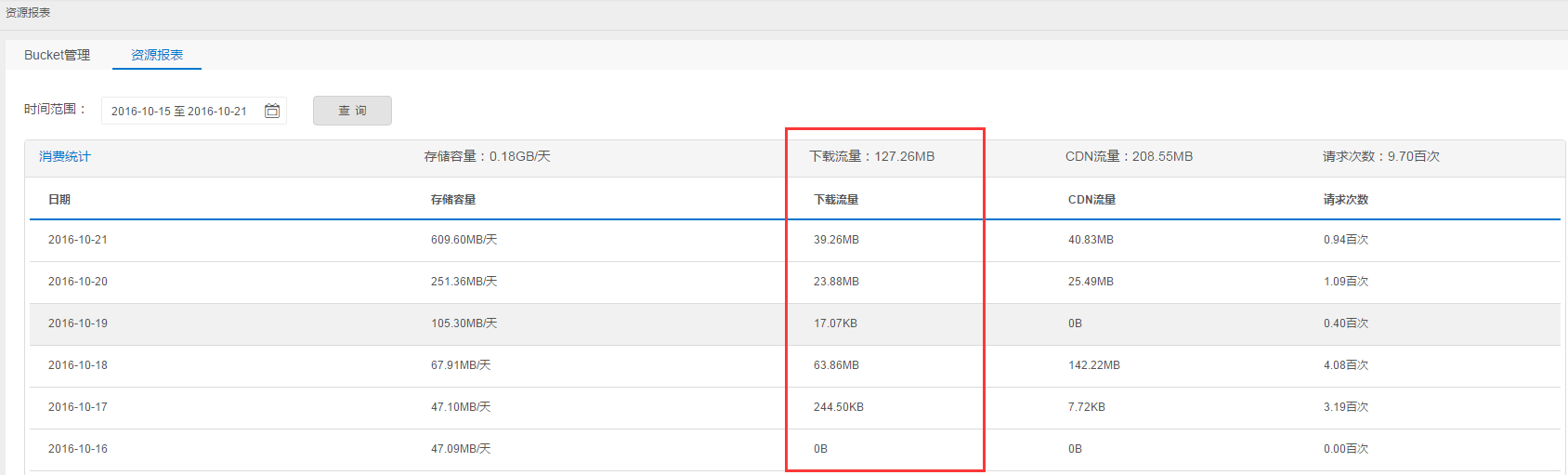
# 业务分析

## 回源流量统计需求整理

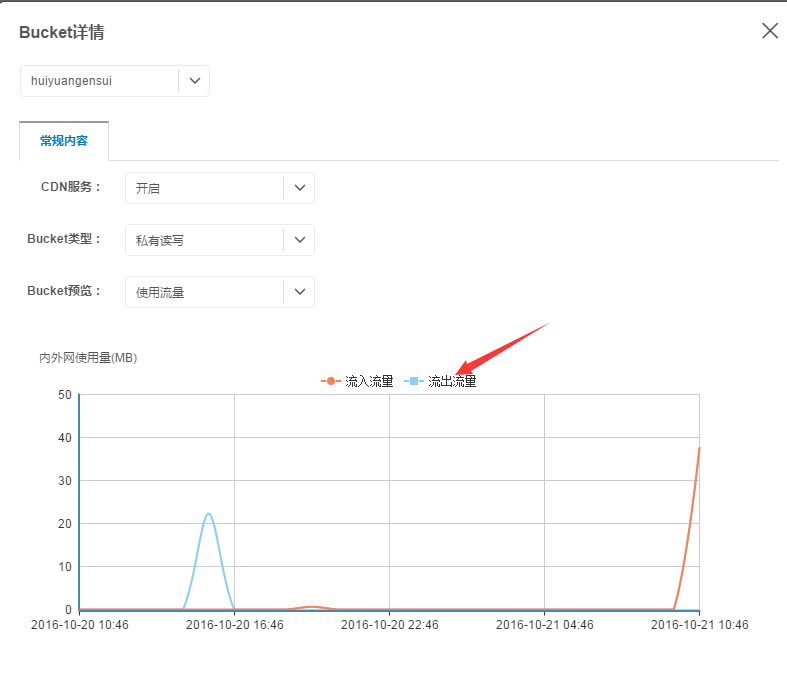
### ECSC

①、目前对象存储资源报表消费统计界面不做修改，还是按原来的需求而定；

②、下图，资源报表中，若用户已开启CDN，从下载流量中剔除掉回源流量的部分



③、如下图，bucket详情中，若用户已开启CDN，则显示的使用流量折线图中从流出流量中剔除掉相应的回源流量



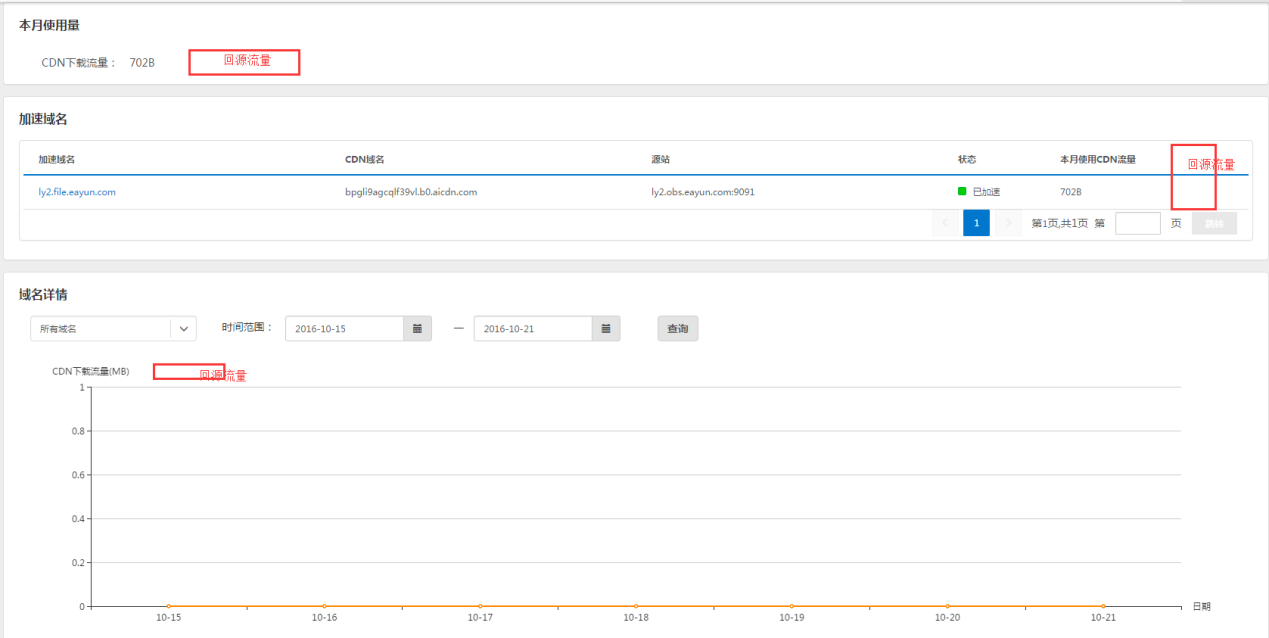
### ECMC

**ECMC CDN服务部分：**

①、在CDN服务页面增加本月回源流量显示

②、列表增加各域名本月回源流量

③、折线图图表增加回源流量信息



**对象存储总览部分：**

24小时新增总下载流量

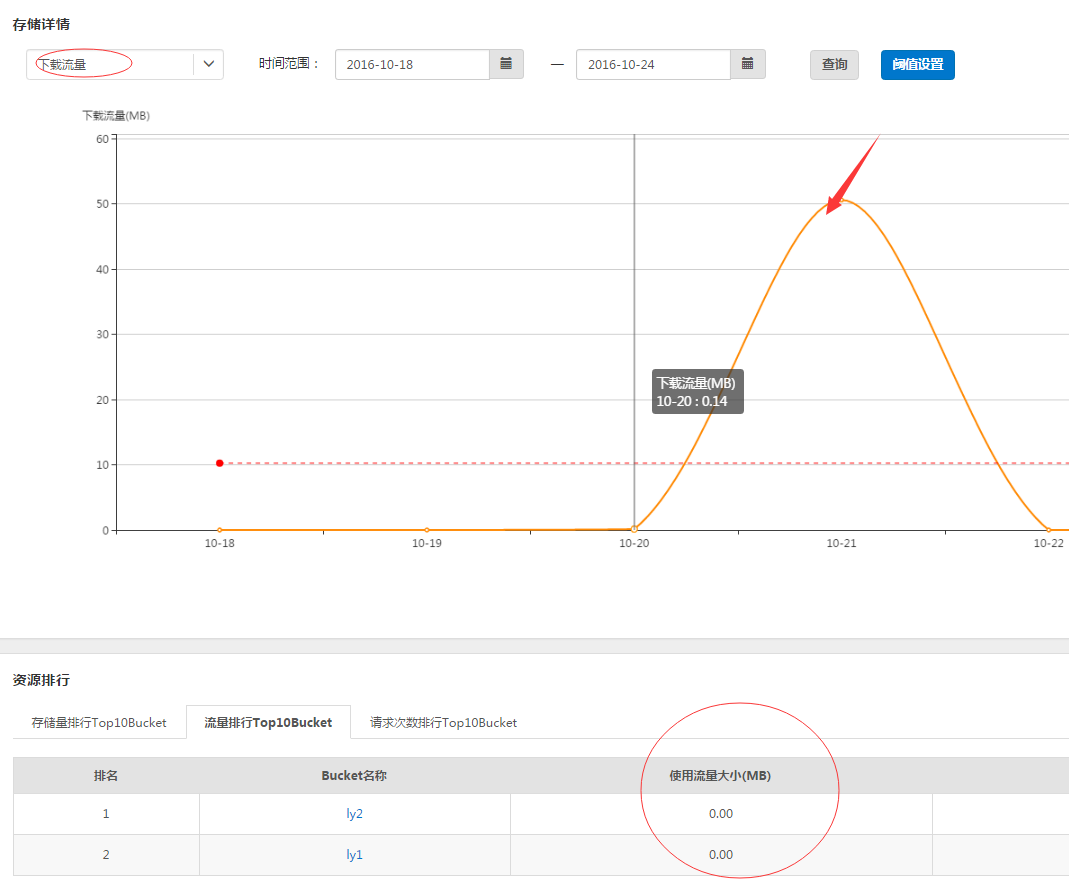
内网下载总流量(MB)

详情下载流量曲线

流量（下载）排行榜

以上四项中剔除掉回源流量





**客户详情部分：**

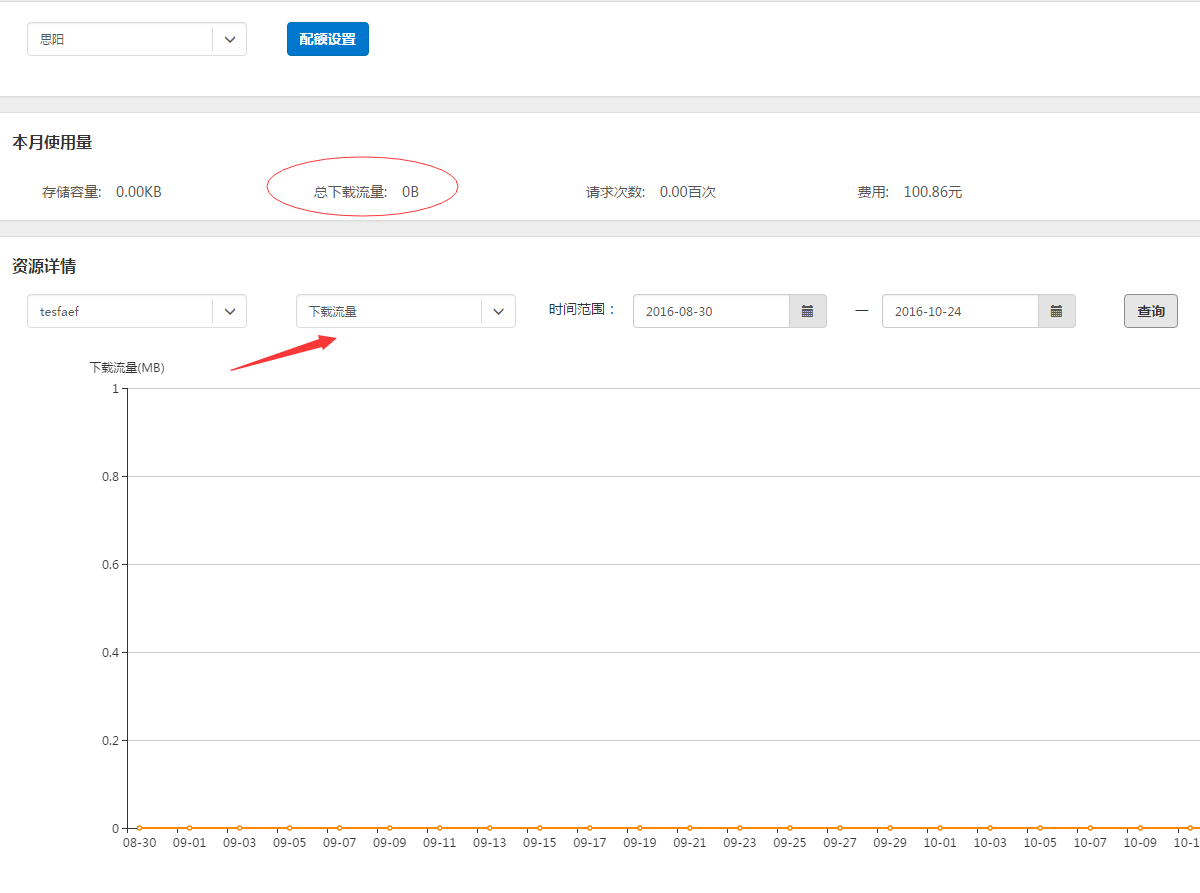
本月使用总下载流量

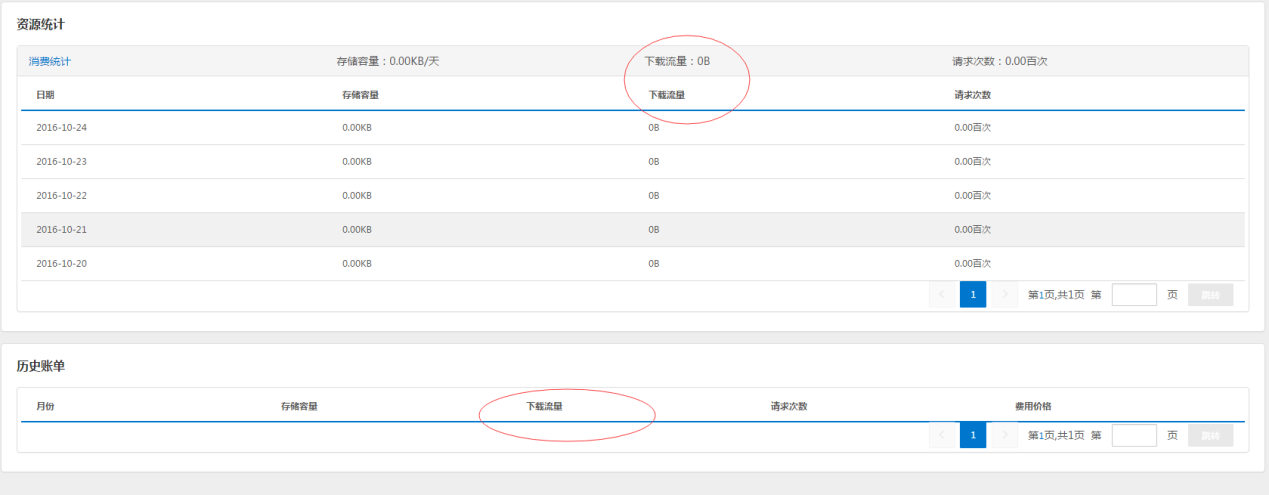
资源详情下载流量曲线

资源统计下载流量

历史账单下载流量

以上四项中剔除掉回源流量





### 计费模块

从目前的计费逻辑中扣除掉回源流量的部分

## 若干问题确认及设计原则

### 回源流量统计延时问题

①、问题描述：因UpYun查询回源流量的API为每5分钟返回一个结果值，且返回有延时情况。

如：正常情况为9:55-59分产生的回源流量，一般会在10:00分的时候统计出来，"timestamp"记为10::00。但延时情况下会发生两种错误情况：

一种为这部分数据有可能在10:05分才会统计出来，"'timestamp"仍为10:00。

第二种情况为58、59分左右产生的数据可能会记录在10:05分的结果中。（10:05分代表的应该是10:00-05分的回源流量）

②、问题影响：由于以上的延时情况，造成某一个小时内的OBS下载流量可能会比回源流量还要小，因此造成计费不准确。

③解决办法：

**经由与产品讨论后暂定解决办法：因OBS计费为每小时计费一次，如该时段内下载流量大于或等于回源流量，即相减结果>=0，则按照正常的计费逻辑扣费；如果该时段内的下载流量小于回源流量，即相减结果<0，则其相减结果按照为0扣费。**

**讨论人：段彬彬、陈浩、张帆、路兴、耿杰**

### 如何从OBS下载流量中剔除回源流量问题

因为上文的扣费确认逻辑，即下载流量小于回源流量时按相减结果为0计算，对页面显示影响：

①、ECSC中bucket详情中流出流量（即下载流量）曲线图：此曲线图为每小时一个点，每点的下载流量数据都减去该点的回源流量值，正则正常显示，负则显示为0

②、ECSC中资源报表统计：此项统计为有当天的数据时，从当天每小时的集合中取值，

当天之前的数据，从统计出的每天的集合中取值。剔除回源时，要每小时每小时的取值比较，正则相加，负则为0再相加。

③、

④、

# UpYun统计回源流量API

请求地址：https://api.upyun.com/flow/common\_data/

查询条件： { start\_time: '2015-10-01 00:00:00', end\_time: '2015-10-05 12:00:00', flow\_source: 'backsource', }

返回结果：

[

{

"time": 128,

"\_200": 1,

"bytes": 1029908,

"reqs": 1,

"bandwidth": 31858.49,

"rps": 0,

"kt": 1476783900,

"kts": "2016-10-18 17:45:00"

},

{

"kt": 1476784200,

"kts": "2016-10-18 17:50:00"

}

]

错误：

{

"error\_code": "23502",

"request": "GET /flow/common\_data/",

"message": "bucket or domain is not belong to you"

}

调用接口：

**public** String getBackSource(String domainId,Date from , Date to) **throws** Exception{

*log*.info("开始进行获取bucket回源流量,domainId:"+domainId+",from:"+from+",to:"+to);

List<NameValuePair> params = **new** ArrayList<NameValuePair>();

params.add(**new** BasicNameValuePair("start\_time", DateUtil.*dateToString*(from)));

params.add(**new** BasicNameValuePair("end\_time", DateUtil.*dateToString*(to)));

params.add(**new** BasicNameValuePair("flow\_source", "backsource"));

params.add(**new** BasicNameValuePair("query\_type", "bucket"));

params.add(**new** BasicNameValuePair("query\_value", domainId));

String url=CDNConstant.*CDN\_COMMONDATA*+"?"+URLEncodedUtils.*format*(params, HTTP.~~UTF\_8~~);

*log*.info("url:"+url);

HttpResponse res=doGet(url);

**int** statusCode = res.getStatusLine().getStatusCode();

String resData = EntityUtils.*toString*(res.getEntity());

*log*.info("response:"+resData);

JSONObject result = **new** JSONObject();

**try** {

JSONArray jsonArray = JSONArray.*parseArray*(resData);

result.put("data", jsonArray);

result.put("result", **true**);

result.put("message", "Succeed in get backsource.");

} **catch** (JSONException e) {

result = JSONObject.*parseObject*(resData);

**if**(result.containsKey("error\_code")){

result.put("result", **false**);

}

}

addCDNLog(**null**, domainId, url, CDNConstant.operationCdn.*QUERY\_BACKSOURCE*, **null**, statusCode, result);

**return** result.toJSONString();

}

TODO：CDN的添加日志接口，将API返回的全部信息存储到CDN日志里

# 涉及功能模块开发

## 计划任务

### CdnBacksourceDetailGatherJob

**新增每小时计划任务**，按照客户及bucket统计回源流量，每小时整点执行，统计时间向前推迟10分钟以确保采集到数据，如10:00执行采集8:50-9:50之间的回源流量。

**新增集合：**"cdn.backsource.1h"记录业务需要使用的回源流量信息。

"cdn.backsource.api.details"记录UpYun查询API返回的详细信息。

### ObsDetailSummaryJob

**原OBS每天汇总的计划任务**，增加代码汇总客户每天的所有回源流量（统计客户每天"cdn.backsource.1h"集合中的所有数据）。仍为每天凌晨15分执行, 与之前相同，记录的timestamp表示为该天的数据（与detail中表示的意义不同），如统计10月20日全天的数据，记录的时间为2016-10-20 00:00:00（mongo中因有时区差距，记录时间显示为2016-10-19 16:00:00）。

**新增集合**cdn.backsource.1d为一天内统计出的客户所有的bucket的回源流量总计。

原"obs.used.24h"集合增加一个属性"final\_data"，表示客户一天内所有原下载流量剔除掉回源流量后的值，根据计费时的原则：按照bucket名称和时间对查询出的下载流量集合和回源流量集合一一对比，将同一bucket同一时间段内统计到的下载流量减去回源流量，正数或0则正常计数，负数则记为0，最终相加为一天内的结果final\_data。

**long** finalData=0;

**for**(**int** i=0;i<jsonList.size();i++){

JSONObject obj=jsonList.get(i);

JSONArray categories = obj.getJSONArray("categories");

Date thisTime = obj.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

String bucketName = obj.getString("bucket");

**long** oneData = 0;

**for**(**int** j=0;j<categories.size();j++){

**long** bytesSent=categories.getJSONObject(j).getLong("bytes\_sent");

download+=bytesSent;

oneData+=bytesSent;

}

**long** oneBacksource=0;

**if**(**null**!=backJsonList && !backJsonList.isEmpty()){

**for**(**int** j=0;j<backJsonList.size();j++){

JSONObject backJson=backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

**if**(thisTime.getTime()==backThisTime.getTime()&&bucketName.equals(backBucketName)){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

}

**long** diff=(oneData-oneBacksource)>0?oneData-oneBacksource:0;

finalData = finalData+diff;

}

### ObsSortDetailSummaryJob

**原OBS下载流量排行Top10计划任务**，原逻辑为查询所有用户的所有bucket上周一周内的下载流量数据，按照用户和bucket分组相加，对结果做排序等处理，redis集合名为 "obs:obsSortByDownload"。

**增加代码逻辑：**对取到的原始数据做处理，每个bucket每小时的数据取值时，增加剔除回源流量逻辑，剔除原则为原流量减去回源流量，为负则记为0。

代码：

**for**(**int** i=0;i<jsonList.size();i++){//下载流量原始数据，所有bucket的每小时下载流量数据

JSONObject obj=jsonList.get(i);

**long** countRequest=0;

**long** download=0;

String owner=obj.getString("owner");

String bucketName=obj.getString("bucket");

JSONArray categories = obj.getJSONArray("categories");

Date thisTime = obj.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

**for**(**int** j=0;j<categories.size();j++){

**long** bytesSent=categories.getJSONObject(j).getLong("bytes\_sent");//原每小时下载流量

download+=bytesSent;

**long** ops=categories.getJSONObject(j).getLong("ops");

countRequest+=ops;

}

**long** backData = 0;

**if**(**null** != backJsonList && !backJsonList.isEmpty()){//回源流量原始数据，所有bucket的每小时回源流量数据

**for**(**int** j=0;j<backJsonList.size();j++){

JSONObject backJson = backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

String cusId = backJson.getString("cus\_id");

**if**(thisTime.getTime()==backThisTime.getTime() &&

bucketName.equals(backBucketName) && cusId.equals(owner)){

//找到对应bucket该时段的回源流量数据

backData = backJson.getLongValue("backsource");

**break**;

}

}

}

//比较，正则正常记录，负则为0

**long** diffData = (download > backData)?(download - backData):0;

JSONObject bucket=**new** JSONObject();

bucket.put("name", bucketName);

bucket.put("owner", owner);

bucket.put("countRequest", countRequest);

bucket.put("download", diffData);//记录新数据

newJsonList.add(bucket);

}

### ObsDetailMonthSummaryJob

原OBS每月下载流量统计，只用在ECMC历史账单显示模块，按照客户维度统计上个月内的所有bucket的下载流量总值。

**逻辑修改：**①原下载流量集合"obs.used.1month"增加一个属性"final\_data "，代表上个月所有下载流量剔除回源流量后的值，剔除原则仍为每小时数据的比较相减，正则正常相加，负则为0。

②、增加集合"cdn.backsource.1month"，存储客户上个月所有bucket的所有回源流量总值。

代码：

**for**(**int** i=0;i<jsonList.size();i++){//下载流量原始数据

JSONObject obj=jsonList.get(i);

**long** bytesSent=obj.getLong("download");

download+=bytesSent;

**long** ops=obj.getLong("countRequest");

countRequest+=ops;

Date thisTime = obj.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

String bucketName = obj.getString("bucket");

**long** backData = 0;

**if**(**null** != backJsonList && !backJsonList.isEmpty()){//回源流量原始数据

**for**(**int** j=0;j<backJsonList.size();j++){

JSONObject backJson=backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

**if**(thisTime.getTime()==backThisTime.getTime() &&

bucketName.equals(backBucketName)){

backData = backJson.getLongValue("backsource");

**break**;

}

}

}

**long** diffData = (bytesSent > backData)?(bytesSent - backData):0;

finalData+=diffData;

}

JSONObject obsUsed=**new** JSONObject();

obsUsed.put("countRequest", countRequest);

obsUsed.put("download", download);

obsUsed.put("timestamp", beforMonth);

obsUsed.put("owner", userId);

obsUsed.put("final\_data", finalData);

mongoTemplate.insert(obsUsed, MongoCollectionName.*OBS\_USED\_1MONTH*);

**long** backsource = 0;

**if**(**null** != backJsonList && !backJsonList.isEmpty()){

**for**(**int** j=0;j<backJsonList.size();j++){

JSONObject backJson=backJsonList.get(j);

**long** sourceData = backJson.getLongValue("backsource");

backsource += sourceData;

}

}

JSONObject backsourceJson=**new** JSONObject();

backsourceJson.put("backsource", backsource);

backsourceJson.put("timestamp", beforMonth);

backsourceJson.put("cus\_id", userId);

mongoTemplate.insert(backsourceJson, MongoCollectionName.*CDN\_BACKSOURCE\_1MONTH*);

### CdnDetailGatherJob

**原CDN下载流量每小时统计计划任务**，统计bucket的每小时CDN下载流量明细。

**逻辑修改：**①、原集合"obs.cdn.1h"增加一个属性"real\_time"，记录插入数据时的真实时间。

②、增加一个集合"obs.cdn.api.details "，记录API返回的完整信息。已有CDN的API日志，暂不考虑在这里增加新集合。

## 模块功能

### 计费接口

流量计费逻辑为：提供两个参数，一为月初到上次计费时间内所用的流量（原为下载流量，现为纯下载流量）和月初到本次计费时间内所用的流量，需要这两次的流量值来确定所在的区间，然后计费，因此需要新提供一个接口，返回月初到某个时间点的纯下载流量。

逻辑为：

//两个时间天数相差超过1天（因每天的计划任务在零点15分执行，因此endTime为0点时，查询不到昨天整天的记录）

**if**(endTime.after(DateUtil.*addDay*(startTime, **new** **int**[]{0,0,1}))){

//计算整天的纯下载流量

Aggregation.*match*(Criteria.*where*("final\_data").exists(**false**)),//没有新加属性的按照旧下载流量相加

Aggregation.*group*().sum("download").as("totalDownLoad"));

MongoCollectionName.*OBS\_USED\_24H*

Aggregation.*match*(Criteria.*where*("final\_data").exists(**true**)),//有新加属性的按照新加属性相加

Aggregation.*group*().sum("final\_data").as("totalFinalData"));

MongoCollectionName.*OBS\_USED\_24H*, JSONObject.**class**);

//计算今天的纯下载流量

//下载流量原始数据

Query(criatira),JSONObject.**class**,MongoCollectionName.*OBS\_USED\_1H*);

//回源流量原始数据

Query(backCriatira),JSONObject.**class**,MongoCollectionName.*CDN\_BACKSOURCE\_1H*);

**if**(**null** != jsonList && !jsonList.isEmpty()){

**for**(**int** i = 0;i < jsonList.size();i++){

JSONObject obj=jsonList.get(i);

JSONArray categories = obj.getJSONArray("categories");

Date thisTime = obj.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

String bucketName = obj.getString("bucket");

**long** oneData = 0;

**for**(**int** j=0;j<categories.size();j++){

**long** bytesSent=categories.getJSONObject(j).getLong("bytes\_sent");

oneData+=bytesSent;

}

**long** oneBacksource=0;

**if**(**null** != backJsonList && !backJsonList.isEmpty()){

**for**(**int** j = 0;j < backJsonList.size();j++){

JSONObject backJson=backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

**if**(thisTime.getTime()==backThisTime.getTime()&&bucketName.equals(backBucketName)){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

}

**long** diff=(oneData-oneBacksource)>0?oneData-oneBacksource:0;

backsource = backsource + diff;

}

}

}**else**{ //两个时间天数相差 小于或等于1天

/下载流量原始数据

Query(criatira),JSONObject.**class**,MongoCollectionName.*OBS\_USED\_1H*);

//回源流量原始数据

Query(backCriatira),JSONObject.**class**,MongoCollectionName.*CDN\_BACKSOURCE\_1H*);

**if**(**null** != jsonList && !jsonList.isEmpty()){

**for**(**int** i = 0;i < jsonList.size();i++){

JSONObject obj=jsonList.get(i);

JSONArray categories = obj.getJSONArray("categories");

Date thisTime = obj.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

String bucketName = obj.getString("bucket");

**long** oneData = 0;

**for**(**int** j=0;j<categories.size();j++){

**long** bytesSent=categories.getJSONObject(j).getLong("bytes\_sent");

oneData+=bytesSent;

}

**long** oneBacksource=0;

**if**(**null** != backJsonList && !backJsonList.isEmpty()){

**for**(**int** j = 0;j < backJsonList.size();j++){

JSONObject backJson=backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

**if**(thisTime.getTime()==backThisTime.getTime()&&bucketName.equals(backBucketName)){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

}

**long** diff=(oneData-oneBacksource)>0?oneData-oneBacksource:0;

backsource = backsource + diff;

}

}

}

**double** result = 0.0d;

result = (**double**)backsource/1024/1024/1024;

**return** result;

}

### ECSC

#### 对象存储-资源报表

**Warning：Job15分钟问题**

ObsUsedServiceImpl.getObsUsedList()

**对下载流量，原逻辑为：**按天统计客户下所有bucket的下载流量，今天的数据显示零点到现在的用量。今天之前的数据，从每天汇总集合中取。

代码逻辑修改：①、取今天的数据时，将每个bucket每小时的下载流量与回源流量一一对比，按照既定原则相加。

**long** download = 0;

**if** (**null** != usedlistToday && usedlistToday.size() > 0) {

**for** (**int** i = 0; i < usedlistToday.size(); i++) {

JSONObject obsUsed = usedlistToday.get(i);

Date timestamp = obsUsed.getDate("timestamp");

timestamp = DateUtil.*dateRemoveSec*(timestamp);

String bucketName = obsUsed.getString("bucket");

String time = DateUtil.*dateToStr*(timestamp);

**if** (time.equals(todayStr)) {

JSONArray categories = obsUsed

.getJSONArray("categories");

**long** oneData = 0;

**for** (**int** j = 0; j < categories.size(); j++) {

**long** Request = categories.getJSONObject(j).getLong(

"ops");

countRequest += Request;

**long** byteSet = categories.getJSONObject(j).getLong(

"bytes\_sent");

oneData += byteSet;

}

/\*\* 当天的数据统计，遍历每个bucket每个时间的统计，正则加，负则为0\*/

**long** backData=0;

**if**(**null**!=backJsonListToday&&!backJsonListToday.isEmpty()){

**for**(**int** j = 0; j < backJsonListToday.size(); j++){

JSONObject backJson = backJsonListToday.get(j);

Date backDate = backJson.getDate("timestamp");

String backBucketName = backJson.getString("bucket\_name");

backDate = DateUtil.*dateRemoveSec*(backDate);

**if**(timestamp.getTime()==backDate.getTime() && bucketName.equals(backBucketName)){

backData = backJson.getLongValue("backsource");

**break**;

}

}

}

**long** diffData = (oneData-backData)>0?oneData-backData:0;

download += diffData;

}

}

}

②、今天之前的数据，从每天汇总集合中取值

/\*\*整天的统计量：改为取新加的属性值\*/

**long** byteSet = **null**==obsUsed.getLong("final\_data")?

obsUsed.getLong("download"):obsUsed.getLong("final\_data");

#### bucket管理-详情流量曲线图

ObsBucketServiceImpl .getBucketUsedAndRequest()

**对使用流量，原逻辑为：**展示一个bucket流入和流出（即下载）流量向前推一天范围内的使用情况。

**代码逻辑修改：**（横坐标取点规则保持不变）将每个bucket每小时的下载流量与回源流量一一对比，按照既定原则相加。

Criteria criatira = **new** Criteria();

criatira.andOperator(Criteria.*where*("bucket").is(bucketName),

Criteria.*where*("owner").is(cusId),

Criteria.*where*("timestamp").gte(start),

Criteria.*where*("timestamp").lt(end));

List<JSONObject> jsonList = mongoTemplate.find(**new** Query(criatira),

JSONObject.**class**, MongoCollectionName.*OBS\_USED\_1H*);

/\*\*剔除掉回源流量：dbb\*/

Criteria backCriatira = **new** Criteria();

backCriatira.andOperator(Criteria.*where*("bucket\_name").is(bucketName),

Criteria.*where*("cus\_id").is(cusId),

Criteria.*where*("timestamp").gte(start),

Criteria.*where*("timestamp").lt(end));

List<JSONObject> backJsonList = mongoTemplate.find(**new** Query(backCriatira),

JSONObject.**class**, MongoCollectionName.*CDN\_BACKSOURCE\_1H*);

**if** (type.equals("used")) {// 查询流量的流入流出

BigDecimal download = **new** BigDecimal(0);

BigDecimal upload = **new** BigDecimal(0);

**for** (**int** j = 0; j < jsonList.size(); j++) {

JSONObject obj = jsonList.get(j);

JSONArray categories = obj.getJSONArray("categories");

Date thisTime = obj.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

**long** oneSent = 0;

**for** (**int** k = 0; k < categories.size(); k++) {

Long bytesSent = categories.getJSONObject(k).getLong(

"bytes\_sent");

oneSent+=bytesSent;

}

/\*\*剔除掉回源流量：dbb\*/

**long** backData=0;

**if**(**null**!=backJsonList && !backJsonList.isEmpty()){

**for** (**int** k = 0; k < backJsonList.size(); k++) {

JSONObject backJson = backJsonList.get(k);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

**if**(thisTime.getTime()==backThisTime.getTime()){

backData = backJson.getLongValue("backsource");

**break**;

}

}

}

**long** diffData = (oneSent-backData)>0?oneSent-backData:0;

download = download.add(**new** BigDecimal(diffData));

}

usedAndRequestBean.setNetIn(download.divide(**new** BigDecimal(1024 \* 1024d), 2, BigDecimal.*ROUND\_HALF\_EVEN*));

usedAndRequestBean.setTimestamp(end);

usedAndRequestBean.setType(type);

}

### ECMC对象存储总览

#### 24小时新增总下载流量

EcmcObsOverviewServiceImpl.getObs24Used()

**对下载流量，原逻辑为：**展示距当前时间相差24h内增加的下载流量，如当前时间为10-24 16:32分，展示10-23 16:00至10-24 16:00之间增加的数据，不包括前一个临界点（即10-23 16:00）的数据。

**代码逻辑修改：**

同查询出相同时间内相同bucket的回源流量值，取到同一客户、同一bucket、同一小时的回源流量，与下载流量相比，下载流量减去回源流量，正则继续相加，负则记为0相加。

**long** flowAgo = 0l;

**long** requestCountAgo = 0l;

**for** (JSONObject ult : usedlistToday) {

List<HashMap> list = (ArrayList<HashMap>) ult.get("categories");

Date thisTime = ult.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

String bucketName = ult.getString("bucket");

String owner = ult.getString("owner");

**long** oneData = 0l;

**for** (HashMap map : list) {

// 流量

oneData += Long.*parseLong*(map.get("bytes\_sent").toString());

// 请求次数

requestCountAgo += Long.*parseLong*(map.get("ops").toString());

}

**long** oneBacksource = 0l;

**if**(**null**!=backJsonList && !backJsonList.isEmpty()){

**for**(**int** j=0;j<backJsonList.size();j++){

JSONObject backJson=backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

String cusId = backJson.getString("cus\_id");

**if**(thisTime.getTime()==backThisTime.getTime() && bucketName.equals(backBucketName) && owner.equals(cusId)){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

}

**long** diffData = (oneData > oneBacksource)?(oneData - oneBacksource):0;

flowAgo += diffData;

}

#### 内网下载总流量(MB)

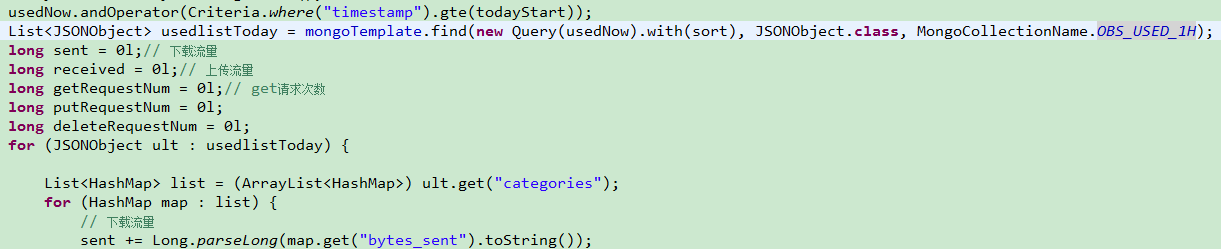
**Warning：Job15分钟问题**

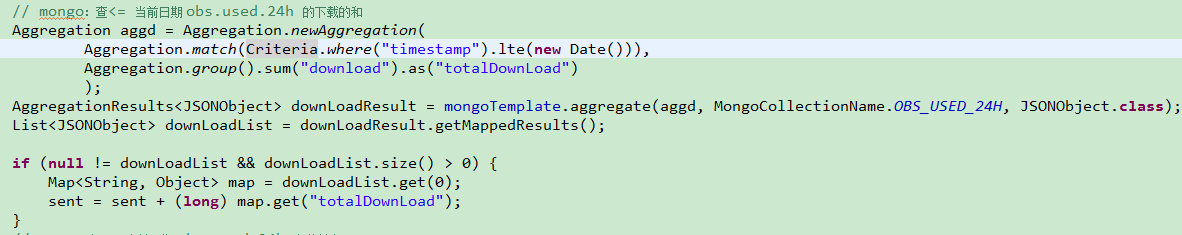
EcmcObsOverviewServiceImpl.getObsView()

**对下载流量，原逻辑为：**系统内所有客户所有bucket一直以来所有的下载流量。

**逻辑修改：**从每个bucket的每小时数据记录中扣除掉回源流量信息，原则仍为下载流量减去回源流量，正则相加，负则为0相加。

原代码逻辑：今天的数据值从"obs.used.1h"集合中取得，今天之前的值取"obs.used.24h"集合中的聚合。





代码修改：①、对今天的数据，查询出对应每个bucket的回源流量记录，一一比较根据上述的原则相加。

/\*\*从今天的数据中扣除回源流量 \*/

Criteria backCriatira = **new** Criteria();

backCriatira.andOperator(Criteria.*where*("timestamp").gte(todayStart));

List<JSONObject> backJsonList = mongoTemplate.find(**new** Query(backCriatira).with(sort),JSONObject.**class**,

MongoCollectionName.*CDN\_BACKSOURCE\_1H*);

**long** sent = 0l;// 下载流量

**long** received = 0l;// 上传流量

**long** getRequestNum = 0l;// get请求次数

**long** putRequestNum = 0l;

**long** deleteRequestNum = 0l;

**for** (JSONObject ult : usedlistToday) {

Date thisTime = ult.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

String bucketName = ult.getString("bucket");

String owner = ult.getString("owner");

List<HashMap> list = (ArrayList<HashMap>) ult.get("categories");

**long** oneData = 0;

**for** (HashMap map : list) {

// 下载流量

oneData += Long.*parseLong*(map.get("bytes\_sent").toString());

}

**long** oneBacksource=0;

**if**(**null**!=backJsonList && !backJsonList.isEmpty()){

**for**(**int** j=0;j<backJsonList.size();j++){

JSONObject backJson=backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

String cusId = backJson.getString("cus\_id");

**if**(thisTime.getTime()==backThisTime.getTime() && bucketName.equals(backBucketName)

&& owner.equals(cusId)){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

}

**long** diffData = (oneData > oneBacksource)?(oneData - oneBacksource):0;

sent += diffData;

}

②、对于今天之前的数据，取"obs.used.24h"集合中新加的属性"final\_data"的值的聚合,考虑到改版之前的数据记录中没有该属性，需做一些调整。

判断有新加属性"final\_data"的取"final\_data"的聚合，没有新加属性的取原属性"download"的聚合，把两次的结果相加。

/\*\*查询今天之前的数据\*/

// mongo：查<= 当前日期 obs.used.24h 的下载的和

Aggregation aggd = Aggregation.*newAggregation*(

Aggregation.*match*(Criteria.*where*("timestamp").lte(**new** Date())),

Aggregation.*match*(Criteria.*where*("final\_data").exists(**false**)),//没有新加属性的按照旧下载流量相加

Aggregation.*group*().sum("download").as("totalDownLoad")

);

AggregationResults<JSONObject> downLoadResult = mongoTemplate.aggregate(aggd,

MongoCollectionName.*OBS\_USED\_24H*, JSONObject.**class**);

List<JSONObject> downLoadList = downLoadResult.getMappedResults();

**if** (**null** != downLoadList && downLoadList.size() > 0) {

Map<String, Object> map = downLoadList.get(0);

sent = sent + (**long**) map.get("totalDownLoad");

}

Aggregation aggback = Aggregation.*newAggregation*(

Aggregation.*match*(Criteria.*where*("timestamp").lte(**new** Date())),

Aggregation.*match*(Criteria.*where*("final\_data").exists(**true**)),//有新加属性的按照新加属性相加

Aggregation.*group*().sum("final\_data").as("totalFinalData")

);

AggregationResults<JSONObject> backResult = mongoTemplate.aggregate(aggback,

MongoCollectionName.*CDN\_BACKSOURCE\_1D*, JSONObject.**class**);

List<JSONObject> backList = backResult.getMappedResults();

**if** (**null** != backList && !backList.isEmpty()) {

Map<String, Object> map = backList.get(0);

sent = sent + (**long**) map.get("totalFinalData");

}

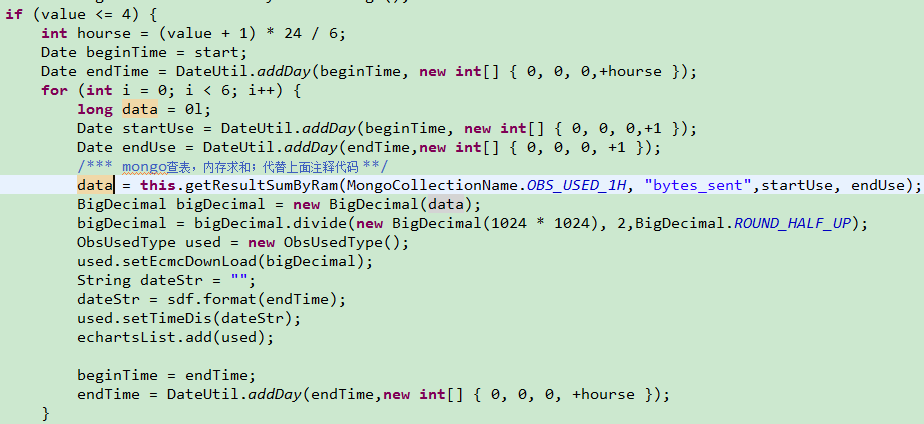
#### 详情下载流量曲线

EcmcObsOverviewServiceImpl.getChart ()

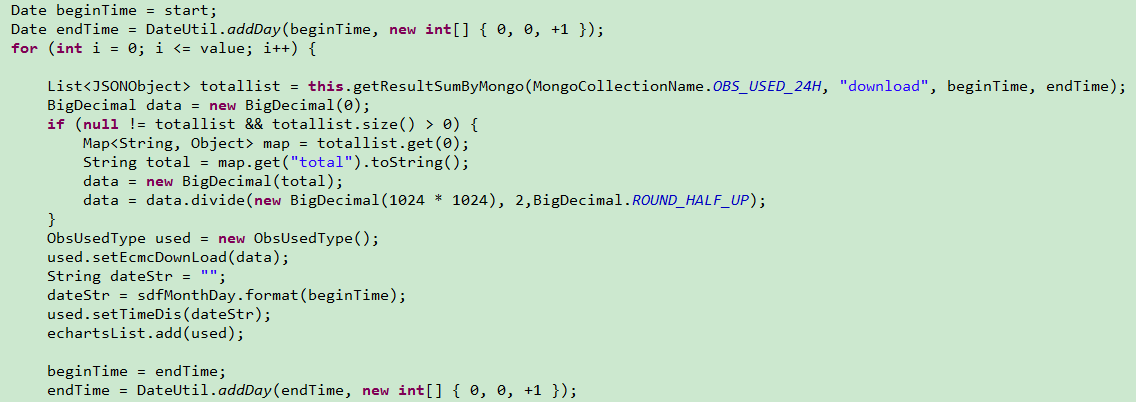
对下载流量，原逻辑为：选定时间范围内的所有用户的所有bucket的下载流量的数据信息，时间点选择根据范围自动确定，代表数据意义为一段时间内的值的和。

逻辑修改：根据原则，扣除回源流量。

**原代码逻辑：**①、时间相差天数<=4时，从"obs.used.1h"集合中取值



②、天数>4时，从"obs.used.24h"集合中取值，如需要取今天的数据，从1小时集合取。



**代码修改：**对原本采取的取点方法和逻辑、横坐标显示规则保持不变。

①、对于从集合"obs.used.1h"中取值的部分，查询出对应bucket的每小时的回源流量信息，一一对比，根据既定原则相加。

/\*\*\*\*\*\* 通过查mongo，内存求和 \*\*\*\*\*\*/

@SuppressWarnings({ "unchecked", "rawtypes" })

**private** **long** getResultSumByRam(String mongoTable, String sumField,

Date beginTime, Date endTime) {

Criteria criatira = **new** Criteria();

criatira.andOperator(Criteria.*where*("timestamp").gte(beginTime),Criteria.*where*("timestamp").lt(endTime));

List<JSONObject> jsonList = mongoTemplate.find(**new** Query(criatira),JSONObject.**class**, mongoTable);

List<JSONObject> backJsonList = **null**;

**if**(sumField.equals("bytes\_sent")){//对象存储总览详情-下载流量曲线数据显示

Criteria backCriatira = **new** Criteria();

backCriatira.andOperator(Criteria.*where*("timestamp").gte(beginTime), Criteria.*where*("timestamp").lt(endTime));

backJsonList = mongoTemplate.find(**new** Query(backCriatira),JSONObject.**class**,MongoCollectionName.*CDN\_BACKSOURCE\_1H*);

}

**long** data = 0l;

**for** (JSONObject json : jsonList) {

**long** count = 0l;

Date thisTime = json.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

String bucketName = json.getString("bucket");

String owner = json.getString("owner");

List<HashMap> list2 = (ArrayList<HashMap>) json.get("categories");

**for** (HashMap map : list2) {

count += Long.*parseLong*(map.get(sumField).toString());

}

**long** oneBacksource = 0;

**if**(**null** != backJsonList && !backJsonList.isEmpty()){

**for**(**int** j=0;j<backJsonList.size();j++){

JSONObject backJson=backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

String cusId = backJson.getString("cus\_id");

**if**(thisTime.getTime()==backThisTime.getTime() && bucketName.equals(backBucketName)

&& owner.equals(cusId)){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

count = (count > oneBacksource)?(count - oneBacksource):0;

}

data = data + count;

}

**return** data;

}

②、对于从"obs.used.24h"集合中取值的部分，取"obs.used.24h"集合中新加的属性"final\_data"的值的聚合,考虑到改版之前的数据记录中没有该属性，需做一些调整。

查询下载流量数据时条件内的集合数据，有新加属性"final\_data"的取"final\_data"聚合相加的值，没有新加属性的取原属性"download"聚合相加的值，两者相加为最终的值。

/\*\*总览详情-下载流量曲线图数据，扣除掉回源流量\*/

**if**(sumField.equals("download")){

Aggregation agg = Aggregation.*newAggregation*(

Aggregation.*match*(Criteria.*where*("timestamp").gte(beginTime)),

Aggregation.*match*(Criteria.*where*("timestamp").lt(endTime)),

Aggregation.*group*().sum("download").as("total"));

AggregationResults<JSONObject> totalResult = mongoTemplate.aggregate(agg, mongoTable, JSONObject.**class**);

totalList = totalResult.getMappedResults();

Aggregation aggback = Aggregation.*newAggregation*(

Aggregation.*match*(Criteria.*where*("timestamp").lte(**new** Date())),

Aggregation.*match*(Criteria.*where*("final\_data").exists(**true**)),//有新加属性的按照新加属性相加

Aggregation.*group*().sum("final\_data").as("total"));

AggregationResults<JSONObject> backResult = mongoTemplate.aggregate(aggback,

MongoCollectionName.*CDN\_BACKSOURCE\_1D*, JSONObject.**class**);

List<JSONObject> backList = backResult.getMappedResults();

**if**(**null** != backList && !backList.isEmpty()){

totalList.addAll(backList);

}

List<JSONObject> totallist = **this**.getResultSumByMongo(MongoCollectionName.*OBS\_USED\_24H*, "download", beginTime, endTime);

BigDecimal data = **new** BigDecimal(0);

**if** (**null** != totallist && totallist.size() > 0) {

Map<String, Object> map = totallist.get(0);

String total = map.get("total").toString();

data = **new** BigDecimal(total);

**if**(totallist.size() > 1){

Map<String, Object> mapback = totallist.get(1);

String totalback = mapback.get("total").toString();

BigDecimal databack = **new** BigDecimal(totalback);

data = data.add(databack);

}

data = data.divide(**new** BigDecimal(1024 \* 1024), 2,BigDecimal.*ROUND\_HALF\_UP*);

}

#### 流量（下载）排行榜

EcmcObsOverviewServiceImpl.getTop10()

**下载流量排行榜，原逻辑为：**按照bucket从本周一0点到现在使用的下载流量的大小由高到低排序，同时显示上一个整周内该bucket的使用流量排列序号。

**逻辑修改：**本周内的流量查询根据既定原则扣除掉回源流量，上一个整周内的流量排序已经过7天计划任务逻辑修改，已扣除过回源流量，不再做特别操作。

**此方法逻辑复杂，需要重新改造**

**代码修改：**

/\*\*

\* 根据传入的type:loadFlow或request，得到换算好的list loadFlow

\* 对应categoriesKey:bytes\_sent; request 对应categoriesKey:ops

\*/

@SuppressWarnings({ "rawtypes", "unchecked" })

**private** List<EcmcObsTopModel> getBucketListFromMongo(List<JSONObject> usedlistToday, List<EcmcObsTopModel> bucketList,

String type, String categoriesKey, Date lastDay) {

List<EcmcObsTopModel> list = **new** ArrayList<EcmcObsTopModel>();

List<JSONObject> backJsonList = **null**;

**if**(type.equals("loadFlow")){

Criteria backCriatira = **new** Criteria();

Sort backsort = **new** Sort(Direction.*DESC*, "timestamp");

backCriatira.andOperator(Criteria.*where*("timestamp").gte(lastDay));

backJsonList = mongoTemplate.find(**new** Query(backCriatira).with(backsort),

JSONObject.**class**,MongoCollectionName.*CDN\_BACKSOURCE\_1H*);

}

**for** (JSONObject sut : usedlistToday) {

**long** flow = 0l;

String bucketName = (String) sut.get("bucket");

Date thisTime = sut.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

EcmcObsTopModel otm = **new** EcmcObsTopModel();

**if** (!bucketName.equals("")) {

// list放入第一个值

**if** (list.size() == 0) {

otm.setBucketName(bucketName);

String obsId = sut.getString("owner");

// 流量(现在)

List<HashMap> flowList = (ArrayList<HashMap>) sut.get("categories");

**for** (HashMap map : flowList) {

flow += Long.*parseLong*(map.get(categoriesKey).toString());

}

/\*\*第一个bucket，扣除该时段的回源流量\*/

**long** oneBacksource = 0;

**if**(**null** != backJsonList && !backJsonList.isEmpty()){

**for**(**int** j = 0 ;j < backJsonList.size();j++){

JSONObject backJson = backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

String cusId = backJson.getString("cus\_id");

**if**(thisTime.getTime()==backThisTime.getTime() && bucketName.equals(backBucketName)

&& obsId.equals(cusId)){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

}

flow = (flow > oneBacksource)?(flow - oneBacksource):0;

otm.setSize(**new** BigDecimal(flow));

otm.setBelongUserId(obsId);

BaseCustomer cus = ecmcCustomerService.getCustomerById(obsId);

**if** (cus != **null** && cus.getCusId() != **null** && cus.getCusId().length() > 0) {

otm.setBelongUserId(obsId);

otm.setBelongUser(cus.getCusOrg());

list.add(otm);

}

} **else** {

**boolean** flag = **false**;

// 判断是否名称相同

**for** (EcmcObsTopModel obsTopModel : list) {

**if** (bucketName.equals(obsTopModel.getBucketName())) {// 判断现在有的list中是否含有当前的bucketName

// 流量(现在)

List<HashMap> flowList = (ArrayList<HashMap>) sut.get("categories");

String obsId = sut.getString("owner");

flow = obsTopModel.getSize().longValue();

**for** (HashMap map : flowList) {

flow = flow + Long.*parseLong*(map.get(categoriesKey).toString());

}

//**TODO** 这里判断可能有问题

/\*\*前面已有过的bucket，扣除该时段的回源流量\*/

**long** oneBacksource = 0;

**if**(**null** != backJsonList && !backJsonList.isEmpty()){

**for**(**int** j = 0 ;j < backJsonList.size();j++){

JSONObject backJson = backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

String cusId = backJson.getString("cus\_id");

**if**(thisTime.getTime()==backThisTime.getTime() && bucketName.equals(backBucketName)

&& obsId.equals(cusId)){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

}

flow = (flow > oneBacksource)?(flow - oneBacksource):0;

obsTopModel.setSize(**new** BigDecimal(flow));

otm.setBelongUserId(obsId);

BaseCustomer cus = ecmcCustomerService.getCustomerById(obsId);

**if** (cus != **null** && cus.getCusId() != **null** && cus.getCusId().length() > 0) {

otm.setBelongUser(cus.getCusOrg());

}

flag = **true**;

**break**;

}

}

**if** (!flag) {// 上面的for中没有这个bucket，则在此加入list

otm.setBucketName(bucketName);

// 流量(现在)

List<HashMap> flowList = (ArrayList<HashMap>) sut.get("categories");

String obsId = sut.getString("owner");

**for** (HashMap map : flowList) {

flow += Long.*parseLong*(map.get(categoriesKey).toString());

}

/\*\*新的bucket，扣除该时段的回源流量\*/

**long** oneBacksource = 0;

**if**(**null** != backJsonList && !backJsonList.isEmpty()){

**for**(**int** j = 0 ;j < backJsonList.size();j++){

JSONObject backJson = backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

String cusId = backJson.getString("cus\_id");

**if**(thisTime.getTime()==backThisTime.getTime() && bucketName.equals(backBucketName)

&& obsId.equals(cusId)){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

}

flow = (flow > oneBacksource)?(flow - oneBacksource):0;

otm.setSize(**new** BigDecimal(flow));

otm.setBelongUserId(obsId);

BaseCustomer cus = ecmcCustomerService.getCustomerById(obsId);

**if** (cus != **null** && cus.getCusId() != **null** && cus.getCusId().length() > 0) {

otm.setBelongUserId(obsId);

otm.setBelongUser(cus.getCusOrg());

list.add(otm);

}

}

}

}

}

// 单位转换

**for** (EcmcObsTopModel model : list) {

EcmcObsTopModel original = model;

**if** (type.equals("loadFlow")) {// 如果选择流量下载MB

original.setSize(original.getSize().divide(**new** BigDecimal(1024 \* 1024), 2,BigDecimal.*ROUND\_HALF\_UP*));

original.setSizeStr(original.getSize()+"");

} **else** {// 请求次数的换算单位：（万次）

original.setSize(original.getSize().divide(**new** BigDecimal(10000), 2, BigDecimal.*ROUND\_HALF\_UP*));

original.setSizeStr(original.getSize()+"");

}

bucketList.add(original);

}

**return** bucketList;

}

### ECMC客户详情

#### 本月使用总下载流量

EcmcObsCustomerServiceImpl.getObsInMonthUsed()

**TODO此处费用需求未定："100.86元"**

**Warning：Job15分钟问题**

**对于下载流量，原逻辑为：**本月初零点到当前时间，某客户的所有bucket的所有下载流量总和。（原代码逻辑，只取了每天计划任务中的数据，现在保持不变）

**逻辑修改：**只取每天计划任务中的数据，即取"obs.used.24h"集合中新加的属性"final\_data"的值的聚合,考虑到改版之前的数据记录中没有该属性，需做一些调整。

判断有新加属性"final\_data"的取"final\_data"的聚合，没有新加属性的取原属性"download"的聚合，把两次的结果相加。

// mongo求sum下载量

Aggregation aggDownLoad = Aggregation.*newAggregation*(

Aggregation.*match*(Criteria.*where*("owner").is(cusId)),

Aggregation.*match*(Criteria.*where*("timestamp").gte(begin)),

Aggregation.*match*(Criteria.*where*("timestamp").lt(today)),

Aggregation.*match*(Criteria.*where*("final\_data").exists(**false**)),//没有新加属性的按照旧下载流量相加

Aggregation.*group*().sum("download").as("totalDownLoad"));

AggregationResults<JSONObject> downLoadResult = mongoTemplate.aggregate(aggDownLoad, MongoCollectionName.*OBS\_USED\_24H*, JSONObject.**class**);

List<JSONObject> downLoadList = downLoadResult.getMappedResults();

**if** (**null** != downLoadList && downLoadList.size() > 0) {

Map<String, Object> map = downLoadList.get(0);

downLoad = (**long**) map.get("totalDownLoad");

}

//计算扣除掉回源流量后的下载流量

Aggregation aggback = Aggregation.*newAggregation*(

Aggregation.*match*(Criteria.*where*("cus\_id").is(cusId)),

Aggregation.*match*(Criteria.*where*("timestamp").gte(begin)),

Aggregation.*match*(Criteria.*where*("timestamp").lt(today)),

Aggregation.*match*(Criteria.*where*("final\_data").exists(**true**)),//有新加属性的按照新加属性相加

Aggregation.*group*().sum("final\_data").as("totalFinalData")

);

AggregationResults<JSONObject> backResult = mongoTemplate.aggregate(aggback,

MongoCollectionName.*CDN\_BACKSOURCE\_1D*, JSONObject.**class**);

List<JSONObject> backList = backResult.getMappedResults();

**if** (**null** != backList && !backList.isEmpty()) {

JSONObject json = backList.get(0);

**long** data = json.getLongValue("totalFinalData");

downLoad = downLoad + data;

}

#### 资源详情下载流量曲线

EcmcObsCustomerServiceImpl .getObsUsedView()

**对下载流量，原逻辑为：**展示指定客户、指定bucket在选定时间段内的下载流量曲线图。

**原代码逻辑：**时间相差天数<=4和>4时横坐标的取点不同，但都要从"obs.used.1h"集合中取值，因为每天汇总的集合里没有bucket的信息。

**逻辑修改：（**对原本采取的取点方法和逻辑、横坐标显示规则保持不变**）**从每小时集合"obs.used.1h"取值，将同一客户同一bucket同一时间的回源集合"cdn.backsource.1h"中的数据取出，一一比较，正则相加，负则为0相加。

/\*\*从下载流量中扣除回源流量\*/

Criteria backCriatira = **new** Criteria();

backCriatira.andOperator(

Criteria.*where*("cus\_id").is(cusId),Criteria.*where*("bucket\_name").is(bucketName),

Criteria.*where*("timestamp").gte(DateUtil.*addDay*(beginTime, **new** **int**[] { 0, 0, 0,+1 })),

Criteria.*where*("timestamp").lt(DateUtil.*addDay*(endTime, **new** **int**[] { 0, 0, 0,+1 })));

List<JSONObject> backJsonList = mongoTemplate.find(**new** Query(backCriatira),JSONObject.**class**,MongoCollectionName.*CDN\_BACKSOURCE\_1H*);

**long** data = 0l;

**for** (JSONObject json : jsonList) {

**long** flow = 0l;

Date thisTime = json.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

List<HashMap> list2 = (ArrayList<HashMap>) json.get("categories");

**for** (HashMap map : list2) {

// 下载流量

flow += Long.*parseLong*(map.get("bytes\_sent").toString());

}

**long** oneBacksource=0;

**if**(**null**!=backJsonList && !backJsonList.isEmpty()){

**for**(**int** j=0;j<backJsonList.size();j++){

JSONObject backJson=backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

**if**(thisTime.getTime()==backThisTime.getTime()){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

flow = (flow > oneBacksource)?(flow - oneBacksource):0;

}

data = data + flow;

}

BigDecimal bigDecimal = **new** BigDecimal(data);

————————————————————————

/\*\*从下载流量中扣除回源流量\*/

Criteria backCriatira = **new** Criteria();

backCriatira.andOperator(Criteria.*where*("cus\_id").is(cusId),

Criteria.*where*("bucket\_name").is(bucketName),

Criteria.*where*("timestamp").gte(beginTime),

Criteria.*where*("timestamp").lt(endTime));

List<JSONObject> backJsonList = mongoTemplate.find(**new** Query(backCriatira),

JSONObject.**class**,MongoCollectionName.*CDN\_BACKSOURCE\_1H*);

**for** (JSONObject json : usedList) {

**long** flow = 0;

Date thisTime = json.getDate("timestamp");

thisTime = DateUtil.*dateRemoveSec*(thisTime);

List<HashMap> list2 = (ArrayList<HashMap>) json.get("categories");

**for** (HashMap map : list2) {

flow += Long.*parseLong*(map.get("bytes\_sent").toString());

}

**long** oneBacksource=0;

**if**(**null**!=backJsonList && !backJsonList.isEmpty()){

**for**(**int** j=0;j<backJsonList.size();j++){

JSONObject backJson=backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

**if**(thisTime.getTime()==backThisTime.getTime()){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

flow = (flow > oneBacksource)?(flow - oneBacksource):0;

}

ObsUsedType out = **new** ObsUsedType();

out.setDownload(flow);

list.add(out);

}

**long** load = 0l;

**for** (ObsUsedType loadDown : list) {

load = load + loadDown.getDownload();

}

#### 资源统计下载流量

EcmcObsCustomerServiceImpl .getObsResources()

**Warning：Job15分钟问题**

**对下载流量，原逻辑为：**查询客户选定时间范围内的下载流量，按天统计。（接口中传入了bucket信息，但无用）。 当天的数据从集合"obs.used.1h"中取得，将得到数据相加；当天之前的数据取"obs.used.24h"集合中汇总好的值。

**代码修改：**从"obs.used.1h"中取数据的部分，先查询出回源流量集合"cdn.backsource.1h"的数据，将相同时间的数据一一比较，根据既定原则相加。

**long** download = 0;

//下载流量、请求次数

**if** (**null** != usedlistToday && usedlistToday.size() > 0) {

**for** (**int** i = 0; i < usedlistToday.size(); i++) {

JSONObject obsUsed = usedlistToday.get(i);

Date timestamp = obsUsed.getDate("timestamp");

String time = format.format(timestamp);

Date thisTime = DateUtil.*dateRemoveSec*(timestamp);

String bucketName = obsUsed.getString("bucket");

**if** (time.equals(todayStr)) {

**long** oneData = 0;

JSONArray categories = obsUsed.getJSONArray("categories");

**for** (**int** j = 0; j < categories.size(); j++) {

**long** Request = categories.getJSONObject(j).getLong("ops");

countRequest += Request;

**long** byteSet = categories.getJSONObject(j).getLong("bytes\_sent");

oneData += byteSet;

}

**long** oneBacksource=0;

**if**(**null**!=backJsonList && !backJsonList.isEmpty()){

**for**(**int** j=0;j<backJsonList.size();j++){

JSONObject backJson=backJsonList.get(j);

Date backThisTime = backJson.getDate("timestamp");

backThisTime = DateUtil.*dateRemoveSec*(backThisTime);

String backBucketName = backJson.getString("bucket\_name");

**if**(thisTime.getTime()==backThisTime.getTime()&&bucketName.equals(backBucketName)){

oneBacksource = backJson.getLongValue("backsource");

**break**;

}

}

oneData = (oneData > oneBacksource)?(oneData - oneBacksource):0;

}

download += oneData;

}

}

}

——————————————————————————————

对从"obs.used.24h"集合中取值的部分，将原来取"download"的值改为取"final\_data"的值，如没有则仍取原值。

//以下两个find只为把mongo数据查出来然后赋值，so： 不能mongo自动求和

List<JSONObject> usedlist = mongoTemplate.find(**new** Query(criatira).with(sort), JSONObject.**class**,MongoCollectionName.*OBS\_USED\_24H*);

List<JSONObject> storageUsed = mongoTemplate.find(**new** Query(criatira).with(sort), JSONObject.**class**,MongoCollectionName.*OBS\_STORAGE\_24H*);

//第一种可能：只有下载流量与请求次数的数据

**if** (**null** != usedlist && usedlist.size() > 0 && (**null** == storageUsed || storageUsed.size() <= 0)) {

**for** (**int** i = 0; i < usedlist.size(); i++) {

JSONObject obsUsed = usedlist.get(i);

ObsUsedType type = **new** ObsUsedType();

**long** coutRequest = obsUsed.getLong("countRequest");

**long** byteSet = **null** != obsUsed.getLong("final\_data")?

obsUsed.getLong("final\_data"):obsUsed.getLong("download");

Date timestamp = obsUsed.getDate("timestamp");

String time = format.format(timestamp);

//转换单位：存储量、流量、请求次数

String requestCount = **this**.convertRequestCountUnit(coutRequest);

String loadFlow = **this**.convertFlowLoadUnit(byteSet);

type.setLoadDown(loadFlow);

type.setRequestCount(requestCount);

type.setUsedStorage("0KB");

type.setStorageUsed(0);

type.setCountRequest(coutRequest);

type.setDownload(byteSet);

type.setTimeDis(time);

type.setTime(timestamp);

obsList.add(type);

}

//第2种可能：只有存储容量的数据

}

//第3种可能：存储容量、下载流量、请求次数的数据都有

} **else** **if** (**null** != storageUsed && storageUsed.size() > 0&& **null** != usedlist && usedlist.size() > 0) {

**for** (**int** i = 0; i < usedlist.size(); i++) {

JSONObject obsUsed = usedlist.get(i);

ObsUsedType type = **new** ObsUsedType();

**long** coutRequest = obsUsed.getLong("countRequest");

**long** byteSet = **null** != obsUsed.getLong("final\_data")?

obsUsed.getLong("final\_data"):obsUsed.getLong("download");

Date timestamp = obsUsed.getDate("timestamp");

String time = format.format(timestamp);

type.setTime(timestamp);

type.setTimeDis(time);

type.setCountRequest(coutRequest);

type.setDownload(byteSet);

//转换单位：存储量、流量、请求次数

String requestCount = **this**.convertRequestCountUnit(coutRequest);

String loadFlow = **this**.convertFlowLoadUnit(byteSet);

type.setLoadDown(loadFlow);

type.setRequestCount(requestCount);

**for** (**int** j = 0; j < storageUsed.size(); j++) {

JSONObject stoageUsed = storageUsed.get(j);

Date times = stoageUsed.getDate("timestamp");

String timeDis = format.format(times);

**if** (time.trim().equals(timeDis.trim())) {

**double** storage = stoageUsed.getDouble("storageUsed");

//转换单位：存储量、流量、请求次数

String resultStorage = **this**.convertStorageUnit(storage);

type.setUsedStorage(resultStorage);

type.setStorageUsed(storage);

**break**;

} **else** {

type.setStorageUsed(0);

type.setUsedStorage("0KB");

}

}

obsList.add(type);

}

#### 历史账单下载流量

EcmcObsCustomerServiceImpl .getObsHistoryResources()

**TODO此处费用需求未定："100.86元"，整个需求模糊。**

**Warning：Job5分钟问题**

**对下载流量，原逻辑为：**直接查询出客户每月汇总"obs.used.1month"中的数据，显示出来。

**逻辑修改：**从"obs.used.1month"集合中取值的部分，将原来取"download"的值改为取"final\_data"的值，如没有则仍取原值。

**for** (**int** i = 0; i < usedlist.size(); i++) {

JSONObject obsUsed = usedlist.get(i);

ObsUsedType type = **new** ObsUsedType();

**long** coutRequest = obsUsed.getLong("countRequest");

**long** byteSet = **null**!=obsUsed.getLong("final\_data")?

obsUsed.getLong("final\_data"):obsUsed.getLong("download");

Date timestamp = obsUsed.getDate("timestamp");

String time = formats.format(timestamp);

//转换单位：存储量、流量、请求次数

String loadFlow = **this**.convertFlowLoadUnit(byteSet);

String requestCount = **this**.convertRequestCountUnit(coutRequest);

type.setLoadDown(loadFlow);

type.setRequestCount(requestCount);

type.setCost("100.86元");

type.setUsedStorage("0KB");

type.setCountRequest(coutRequest);

type.setDownload(byteSet);

type.setStorageUsed(0);

type.setTimeDis(time);

type.setTime(timestamp);

obsList.add(type);

}

### ECMC-CDN服务

这里新增的回源流量信息只显示纯回源流量，不牵涉其他。

#### 新增本月使用回源流量

EcmcObsCdnServiceImpl.getMonthDomainData()

**原CDN的逻辑：**显示客户本月的所有CDN下载流量总和

**新增逻辑：**显示客户本月所有的回源流量总和

//回源流量

Aggregation backAgg = Aggregation.*newAggregation*(

Aggregation.*match*(Criteria.*where*("cus\_id").is(cusId)),

Aggregation.*match*(Criteria.*where*("timestamp").gt(firstDay)),

Aggregation.*match*(Criteria.*where*("timestamp").lte(now)),

Aggregation.*group*().sum("backsource").as("sum")

);

AggregationResults<JSONObject> backResult = mongoTemplate.aggregate(

backAgg,MongoCollectionName.*CDN\_BACKSOURCE\_1H*, JSONObject.**class**);

List<JSONObject> backList = backResult.getMappedResults();

**long** backData = 0;

**if**(**null** != backList && !backList.isEmpty()){

JSONObject obj = backList.get(0);

backData = obj.getLongValue("sum");

}

type.setBacksource(backData);

type.setBacksourceStr(**this**.cdnFlowUnit(backData));

#### 域名本月使用回源流量

EcmcObsCdnServiceImpl .getMonthDomainList()

**原CDN逻辑：**显示该客户**本月内曾经**开通过CDN的加速域名列表，同时显示本月内使用的CDN流量。

**新增逻辑：**加上显示上述加速域名本月内的回源流量。

/\*\*

\* 查询某加速域名本月内使用的纯回源流量

\* **@param** domainId

\* **@return**

\*/

**private** **long** getMonthBackByDomain(String domainId){

Date now = **new** Date();

Calendar c = Calendar.*getInstance*();

c.add(Calendar.*MONTH*, 0);

c.set(Calendar.*DAY\_OF\_MONTH*,1);

c.set(Calendar.*HOUR\_OF\_DAY*, 0);

c.set(Calendar.*MINUTE*, 0);

c.set(Calendar.*SECOND*, 0);

Date firstDay = c.getTime();

Aggregation agg = Aggregation.*newAggregation*(

Aggregation.*match*(Criteria.*where*("domain\_id").is(domainId)),

Aggregation.*match*(Criteria.*where*("timestamp").gt(firstDay)),

Aggregation.*match*(Criteria.*where*("timestamp").lte(now)),

Aggregation.*group*().sum("backsource").as("sum")

);

AggregationResults<JSONObject> totalresult = mongoTemplate.aggregate(

agg,MongoCollectionName.*CDN\_BACKSOURCE\_1H*, JSONObject.**class**);

List<JSONObject> totallist = totalresult.getMappedResults();

**long** backData = 0;

**if**(totallist.size() > 0){

JSONObject obj = totallist.get(0);

backData = obj.getLongValue("sum");

}

**return** backData;

}

#### 域名/全部回源流量曲线

EcmcObsCdnServiceImpl .getDomainData()

**原CDN逻辑：**显示客户指定bucket(或全部bucket)在选定时间范围内的CDN下载流量曲线，

**新增逻辑：**在上述原则下，加一条曲线显示回源流量

