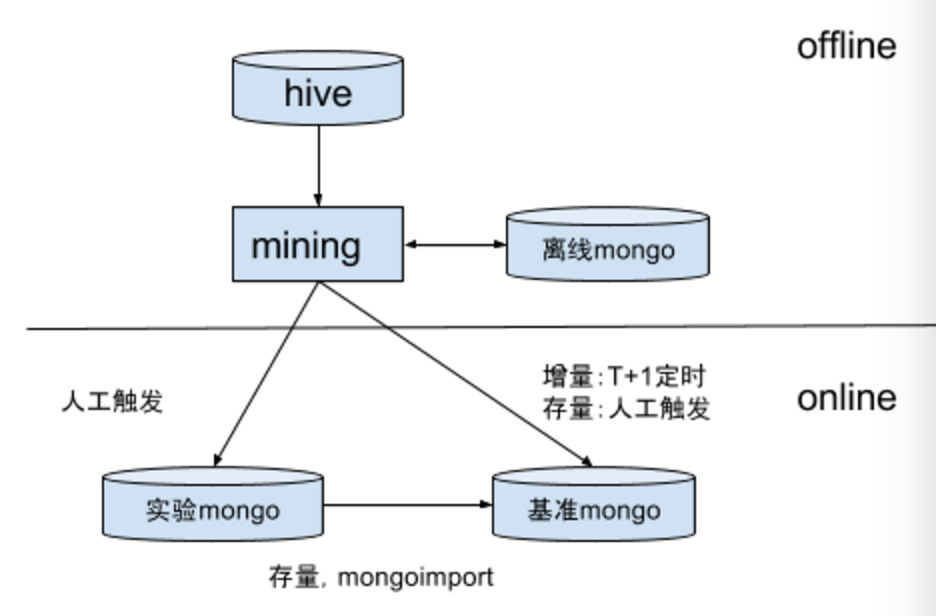
LBS推荐数据处理方案

**一、推荐数据存储**



**线上部分：**

使用mongodb，分2个库：

1. StartPosRecomm，正式库
2. StartPosRecomm-exp，实验库，按city灰度

实验库暂不使用，按城市预留开关即可

collection分2类，按city分表：

1. BaseResults，基准结果集，不频繁更新，定期restore或批量更新
2. IncrResuls，增量结果集，T+1的方式批量更新(替换)

增量结果集仅离线使用，根据增量变化阈值，触发离线基准数据的变更，同时使用线上api更新线上数据（线上每个城市仅保留基准结果集1个collection）

collection数据格式如下：

备注：标黄的部分仅离线部分存储

{

   "\_id" : "wx4eqwd525gb",

   "city" : "bj",

   "regeo" : "XXX大厦",

   "radius" : 200,

   "num" : 3,

   "recomms" : [

       {

           "lat" : 39.8663774224232,

           "lng" : 116.354408229915,

           "name" : "XXX-推荐上车点",

           "score" : 0.3,

           "tag" : "DB",

           "status" : 0,

           "count" : 100,

           "show" : 10000,

           "pick" : 3,

           "ext" : "some addon info"

       }

   ],

   “create\_time” : timestamp,

   "update\_time" : timestamp

}

备注：红字部分仅离线部分存储

|  |  |  |
| --- | --- | --- |
| 一级节点 | 二级节点 | 备注 |
| \_id |  | 期望上车点坐标geohash |
| city |  | 城市标记 |
| regeo |  | 期望上车点regeo名称 |
| radius |  | 推荐半径，和startPos组合区分层级 |
| num |  | 推荐结果数量 |
| recomms |  | 候选集合，list类型 |
|  | lat | 推荐点纬度 |
|  | lng | 推荐点经度 |
|  | name | 推荐点名称 |
|  | score | 推荐点权重，后端计算生成，排序使用 |
|  | tag | 标准位，字符串，推荐策略标记：  DB，依据百度点矫正  AC01，高德聚类结果，版本01 |
|  | status | 标志位，int，供参考：  0，基准点  1，增量点  2，新增点  3，再次命中  4，推荐被选择  5，人工干预  -1，删除 |
|  | count | optional，累计上车次数 |
|  | show | optional，展现次数 |
|  | pick | optional，累计被选中次数 |
|  | ext | optional，附加信息，字符串，仅在实验库中使用 |
| create\_time |  | 结果整体更新时间 |
| update\_time |  | 结果整体更新时间 |

**离线部分：**

1、采用Mysql (mongo占用内存多，DM1资源不够)，存储方式参考线上部分，存储方式参考线上部分

2、推荐数据更新至线上前，在离线部分进行备份，以便回滚

3、离线部分提供一个简单的人工干预机制，可手工删除、修改推荐点数据

**二、更新机制**

**线上：**

仅使用基准结果集BaseResults，不定期通过接口更新（离线触发）

**离线：**

分基准结果集BaseResults、增量结果集IncrResuls，共2个collection：

1. BaseResults更新，(周期不定，离线数据积累合适时触发更新)：
2. IncrResuls更新方式：
3. T+1接入昨日的端回传数据，计算点击率变化（供存量排序调权）
4. 每月计算一次上月的全量点（供存量增加新点使用）

**三、服务封装**

1. 功能需求
   1. 支持灰度城市配置，实验城市推荐结果从实验库取
   2. ~~基准集合与增量集合融合，优先取基准，增量做补充~~
   3. geohash扩大周边半径，融合推荐结果
   4. 按半径距离过滤推荐点
   5. 距离+score综合调权，二次排序
   6. 推荐点命名（可通过开关控制，取离线结果还是实时请求）
   7. 最终推荐结果的二级缓存，内存/redis
   8. 服务降级，无法获取推荐数据时特殊处理
2. 日志需求
   1. 推荐点展现及被选择的情况记录，字段参考：

expect\_lat # 触发推荐的期望上车坐标

expect\_lng

city

radius

num

timestamp

isPick # 该批次展现点是否被选中，0/1

recomms\_json = [

{

"lat" : 39.8663774224232,

"lng" : 116.354408229915,

"score" : 0.3, #最终展现的权重

"tag" : "DB", #透传

“pick” : 1 # 选中

}

{

"lat" : 39.8663774224232,

"lng" : 116.354408229915,

"score" : 0.3,

"tag" : "DB",

“pick” : 0 # 展现未选中

}

{

"lat" : 39.8663774224232,

"lng" : 116.354408229915,

"score" : 0.3,

"tag" : "DB",

“pick” : 0 # 展现未选中

}

]

选中的判断方式需要再进一步和端上讨论

1. 接口需求，mongo CRUD
   1. 基准/增量结果集查询接口
      1. 输入：期望开始点geohash
      2. 输出：bson对象
      3. 是否支持批量：是
   2. 基准/增量结果集批量更新(含新增)接口
      1. 输入：待更新的bson对象list
      2. 输出：更新成功个数
   3. 基准/增量结果集批量删除接口
      1. 输入：期望开始点geohash的list
      2. 输出：删除成功个数

**四、计划**

数据部分

1-13：基准数据定版

1-20 : 增量更新部分调通

节后：增量更新部分与kafka对接，根据数据积累情况，调整计算策略后正式上线

服务部分

XX-XX：接口改造

XX-XX：提供mongo crud

XX-XX ： 服务上线

备注：

增量数据有2种方式体现效果：

1. 短周期merge到基准数据中，比如3天
2. 基准和增量按权重混合展现，基准较长周期更新，比如30天

建议根据数据积累情况分析后再定