## 定时器

由Linux命令执行定时任务，运行 offset\_main.py ，定时任务周期暂定1小时。

## 主函数

* 函数名：offset\_main
* 所在文件名：offset\_main.py
* 功能：读取配置参数，从Oracle数据库读取数据，进行优化计算，检查是否存在异常，如有异常，则进行报警，将OFFSET优化结果写入Oracle数据库，将前端需要用的数据表写入Postgresql数据库。
* 输入参数：无
* 返回值：无
* 行动结果：
  + 将OFFSET优化结果写入Oracle数据库
    - 测试环境：用户bigdataifuser,密码bigdataifuser,Host 10.69.2.137:1521,SID rptdev
    - 生产环境：用户bigdataifuser,密码bigdataifuser,Host 10.69.2.15:1521,SID rptdb
    - 写入方式：追加
    - 表名：mes\_bigdataif\_maskoffset
    - 字段及描述如下
      * TIMEKEY为计算所用glass中最近的生产开始时间？
    - 
  + 将OFFSET优化结果写入Postgresql数据库（用于页面2的展示）
    - 环境：原来为参数如下，重装系统后需要重建，以下简称本地Postgresql环境
    - PSQL\_DB = 'VISIONOX' PSQL\_USER = 'postgres' PSQL\_PASS = 'postgres' PSQL\_HOST = '10.68.2.182' PSQL\_PORT = int(5432)
    - 写入方式：追加（对多余历史数据的删除由前端来做）
    - 表名：OFFSET
    - 字段及描述：
      * Oracle数据库中mes\_bigdataif\_maskoffset表从TIMEKEY到EVAOFFSETTHETA字段，加上产品名（Oracle数据库，EDADB.EDA\_EVA\_PPA表，PRODUCT\_ID字段）
  + 若产生异常，执行发邮件操作，且将相关数据写入Postgresql数据库（用于页面4的展示）
    - 环境：本地Postgresql
    - 写入方式：追加（对多余历史数据的删除由前端来做）
    - 表名：ALARM
    - 字段及描述：
      * PRODUCT\_ID
      * EVENT\_TIME
      * GLASS\_ID
      * EVA\_CHAMBER
      * MASK\_ID
      * MASK\_SET
      * PORT
      * LINE
      * KEY：报警类型，有下面几类：
        + PPA\_X：当某个PPA\_X绝对值大于50，则报警
        + PPA\_Y：当某个PPA\_X绝对值大于50，则报警
        + PPA\_Difference：当3张glass的PPA差异过大，则报警
        + OFFSET\_X：当delta\_OFFSET\_X绝对值大于2.5时，则报警
        + OFFSET\_Y：当delta\_OFFSET\_Y绝对值大于2.5时，则报警
        + OFFSET\_THETA：当delta\_OFFSET\_THETA绝对值大于36时，则报警
      * VALUE：报警量对应的值
  + 将优化前、后的PPA总结数据写入Postgresql数据库（用于页面1中4幅图的展示，以及页面3中表格的展示）
    - 环境：本地Postgresql
    - 写入方式：追加（对多余历史数据的删除由前端来做）
    - 表名：PPA\_SUMMARY
    - 字段及描述：
      * PRODUCT\_ID
      * EVENT\_TIME
      * GLASS\_ID
      * EVA\_CHAMBER
      * MASK\_ID
      * MASK\_SET
      * PORT
      * LINE
      * Ratio\_Type：目前有X4.5、Y4.5、X6、Y6四种，可根据配置而变化
      * Ratio\_Value：与Ratio\_Type对应的良率
      * PPA\_X\_Max
      * PPA\_X\_Min
      * PPA\_Y\_Max
      * PPA\_Y\_Min
  + 将优化前、后的PPA数据写入Postgresql数据库（用于页面3中两幅图的展示）
    - 环境：本地Postgresql
    - 写入方式：追加（对多余历史数据的删除由前端来做）
    - 表名：EVA\_ALL
    - 字段及描述：
      * PRODUCT\_ID
      * EVENT\_TIME
      * GLASS\_ID
      * EVA\_CHAMBER
      * MASK\_ID
      * MASK\_SET
      * PORT
      * LINE
      * POS\_X
      * POS\_Y
      * X\_Label：由于不同产品的排布会发生变化，此处不再用字母标注，而直接用整数标注，POS\_X最**小**的标记为1，随着POS\_X增大，数值逐渐增加
      * Y\_Label：POS\_Y最**大**的标记为1，随着POS\_Y减小，数值逐渐增加
      * PPA\_X：实际PPA\_X值
      * PPA\_Y：
      * PPA\_X\_After：在实际PPA\_X值基础上，再进行优化后的值
      * PPA\_Y\_After
      * OFFSET\_X：实际OFFSET\_X值
      * OFFSET\_Y
      * OFFSET\_tht
* 调用函数：collect\_data、clean\_data、cal\_optimized\_offset
* 被调用函数：无

## 数据获取与新周期判断

* 函数名：collect\_data
* 功能：数据预处理，判断是否进入了新周期，确定是否进行下一步计算
* 输入参数：
  + Oracle\_data\_endurance：从Oracle数据库抽取数据距现在的时间（默认与定时任务的周期相同）
  + Post\_data\_endurance：从Postgresql数据库抽取数据距现在的时间（默认为2天）
  + Dummy工单过滤条件（默认条件待反馈）
* 返回值：
  + EVA\_ALL：与Postgresql数据库EVA\_ALL表中字段相同，但仅包含进入了新周期的组最近（默认2天）数据
* 行动结果：将新数据追加写入Postgresql数据库EVA\_ALL表中（若有重复记录，则跳过该记录）
* 算法：
  + 依据：
  + 在一个Cycle内，同一产品，同一Chamber、同一LINE对应的MASK\_SET应相同，如果MASK\_SET发生了改变，则进入了下一Cycle。（对LINE进行分组，但不对PORT进行分组，因为某些PORT可能未被抽到）
  + 这里用PPA数据来判断是否进入了新周期，若进入了新周期，则直接进行OFFSET计算，否则不进行计算。若利用工艺数据判断，由于PPA数据相对工艺数据有延时，且延时不完全确定，则依然难以确定开始计算OFFSET的时间。
  + 过程：
    1. 从Oracle的EDADB.EDA\_EVA\_PPA表中抽取最近的数据，进行预处理，包括过滤Dummy工单，增加X\_Label、Y\_Label（产品名在此确定）
    2. 从Oracle的edadb.EDC\_PROCESS\_DATA\_VIEW表中抽取最近的数据，进行预处理，包括过滤Dummy工单，目的是找到PPA数据中Glass对应的Mask、Port、OFFSET\_X、OFFSET\_Y、OFFSET\_tht、EVENTTIME
    3. 将二者合并（交集），得到新数据表
    4. 从Postgresql数据库EVA\_ALL表中抽取最近（默认2天）数据，得到历史数据表【初始EVA\_ALL表的获得，单独写程序得到】
    5. 在历史数据表中，找到同一产品、同一Chamber、同一LINE的最后一条数据对应的MASK\_SET
    6. 对每一产品、Chamber、LINE组合，将新数据表的MASK\_SET与历史数据表最后一条数据的MASK\_SET做差集，看结果是否为空
       - 若不为空：
         1. 将差集对应记录追加写入Postgresql数据库EVA\_ALL表中（除了PPA\_X\_After、PPA\_Y\_After字段没有值，不填）
         2. 计算差集对应记录汇总数据，追加写入Postgresql数据库PPA\_SUMMARY表中
         3. 将新数据与历史数据表合并，作为EVA\_ALL的返回值的一部分
       - 若为空，该产品、Chamber、Port组合不执行操作，不作为EVA\_ALL的返回值的一部分
* 调用函数：analysis\_pmaskid、analysis\_ppa
* 被调用函数：offset\_main

## 异常值处理与多周期折算

* 函数名：clean\_data
* 功能：
  1. 将PPA大于50的数据作为异常值处理，进行报警
  2. 剔除同一周期多张glass的PPA差别大的情况，进行报警
  3. 若最近一周期抽检不足3片，将之前抽检的PPA折算到最近一周期内
* 输入参数：
  + EVA\_ALL：collect\_data函数的返回值
  + PPA\_outlier\_threshold：PPA异常的阈值，默认为50
  + PPA\_Difference\_nsigma：多张PPA差异过大的阈值，其中标准差前的系数，默认为2
  + PPA\_Difference\_rate：多张PPA差异过大的阈值，其中最多允许的异常的比例，默认为0.4
* 返回值：
  + PPA\_use：可直接用于优化计算的PPA数据
* 行动结果：若发现异常，则触发报警行动，且将异常数据追加写入Postgresql数据库ALARM表中
* 算法：
  + 依据：每个产品、每个Chamber、每个LINE（特定PORT被抽到的次数是随机的，但只要抽取次数足够多，每个PORT都一定会被抽到）中产生的Glass都会被抽检，产生PPA数据
  + 过程：
    1. 检查EVA\_ALL数据是否为空（当每个产品、Chamber、LINE组合都未进入新周期时发生），若为空，停止程序，否则继续下面的算法
    2. 对EVA\_ALL数据，按产品、Chamber、LINE、PORT进行分组，组内按时间由近到远排序
    3. 对记录（从0开始）编号，连续相同的MASK\_SET用同一编号，不同MASK\_SET则继续编号，该编号即为周期序号，例如ABBBBAAABBAAABBBAA，编号为011112223344455566
    4. 检查编号为1（编号为0的Mask是新周期的开始，故不使用）的MASK\_SET记录数是否大于等于3，若是采用全部记录；若不是，则顺次向后寻找MASK\_SET相同的记录，直到找到3条记录为止
    5. 对找到的记录，检查PPA是否有异常，若有，进行报警，且将报警信息写入Postgresql数据库ALARM表中
    6. 将异常的PPA用其上下两点PPA的均值代替，若自身为端点，则直接用上或下方相邻点PPA代替（若某一列连续出现异常，则用该列均值替代，若该列所有均为异常，舍去该glass的数据）
    7. 通过周期序号及实际OFFSET值，将找到记录的PPA折算到周期序号为1的PPA，折算规则为：先通过OFFSET值计算各周期间的delta\_OFFSET值，然后按照PPA与delta\_OFFSET的递推关系得到更近周期的PPA
    8. 检查折算后各Glass的PPA差异，若存在异常，则报警，且将报警信息写入Postgresql数据库ALARM表中，补充新的Glass，直到补足3片。异常规则暂定如下：
    - 对每一Glass，将数据分为该Glass和剩余其他Glass两组，对每个点位，计算剩余其他的PPA的均值和标准差，检查该Glass对应点位PPA的值是否在区间内（标准差前倍数可配置），如果在该区间内，判该点位正常，否则为异常，若异常点位超过40%（可配置），判该Glass异常
    1. 将折算后的PPA数据作为PPA\_use的返回值
* 调用函数：alarm
* 被调用函数：offset\_main

## 计算最优OFFSET

* 函数名：cal\_optimized\_offset
* 功能：根据各种配置，计算最优OFFSET，将相关结果写入两个数据库中
* 输入参数：
  + PPA\_use：经过清洗的PPA数据
  + PPA\_threshold\_loose：PPA宽松管控线，默认值6.5
  + PPA\_threshold\_strict：PPA严格管控线，默认值4
  + weight\_x\_loose：优化目标中，PPA\_X绝对值小于宽松管控线的权重，默认值0.25
  + weight\_y\_loose：优化目标中，PPA\_Y绝对值小于宽松管控线的权重，默认值0.25
  + weight\_t\_loose：优化目标中，PPA\_X、PPA\_Y绝对值最大值小于宽松管控线的权重，默认值0
  + weight\_x\_strict：优化目标中，PPA\_X绝对值小于宽严格控线的权重，默认值0.25
  + weight\_y\_strict：优化目标中，PPA\_Y绝对值小于宽严格控线的权重，默认值0.25
  + weight\_t\_strict：优化目标中，PPA\_X、PPA\_Y绝对值最大值小于严格管控线的权重，默认值0（以上6个参数用1个向量也可）
  + delta\_OFFSET\_X\_low：改变OFFSET\_X的最小值，默认为0.5
  + delta\_OFFSET\_X\_high：改变OFFSET\_X的最大值，默认为2.5
  + delta\_OFFSET\_Y\_low：改变OFFSET\_Y的最小值，默认为0.5
  + delta\_OFFSET\_Y\_high：改变OFFSET\_Y的最大值，默认为2.5
  + delta\_OFFSET\_tht\_low：改变OFFSET\_tht的最小值，默认为4（delta\_OFFSET\_Y\_low \* 18000/pi/max(POS\_X)=0.5 \* 18000/670/pi）
  + delta\_OFFSET\_tht\_high：改变OFFSET\_tht的最大值，默认为36（delta\_OFFSET\_X\_high \* 18000/pi/max(POS\_Y)=2.5 \* 18000/400/pi）（以上6个参数用1个向量也可）
* 返回值：
  + OFFSET：最终表的数据，可能有用
* 行动结果：将计算的OFFSET结果写入Postgresql数据库OFFSET表中，同时写入Oracle数据库mes\_bigdataif\_maskoffset表中，当delta\_OFFSET值大于阈值时，进行报警
* 算法：
  + 过程：
    1. 分每个产品、每个Chamber、每个Port，按照设置的权重，计算最优delta\_OFFSET
    2. 若delta\_OFFSET\_X、Y、tht三者均小于等于低阈值，则令三者都为0
    3. 若delta\_OFFSET\_X、Y、tht三者任一大于高阈值，进行报警
    4. 由delta\_OFFSET计算OFFSET，写入对应数据库的表中
    5. 将PPA\_X\_After、PPA\_Y\_After字段写入Postgresql数据库EVA\_ALL表的对应字段中
* 调用函数：alarm、optimize\_offset
* 被调用函数：offset\_main

## 配置参数总结

* Oracle\_data\_endurance：从Oracle数据库抽取数据距现在的时间（默认与定时任务的周期相同，1小时）
* Post\_data\_endurance：从Postgresql数据库抽取数据距现在的时间（默认为2天）
* Dummy工单过滤条件（默认条件待反馈）
* PPA\_outlier\_threshold：PPA异常的阈值，默认为50
* PPA\_Difference\_nsigma：多张PPA差异过大的阈值，其中标准差前的系数，默认为2
* PPA\_Difference\_rate：多张PPA差异过大的阈值，其中最多允许的异常的比例，默认为0.4
* PPA\_threshold\_loose：PPA宽松管控线，默认值6.5
* PPA\_threshold\_strict：PPA严格管控线，默认值4
* weight\_x\_loose：优化目标中，PPA\_X绝对值小于宽松管控线的权重，默认值0.25
* weight\_y\_loose：优化目标中，PPA\_Y绝对值小于宽松管控线的权重，默认值0.25
* weight\_t\_loose：优化目标中，PPA\_X、PPA\_Y绝对值最大值小于宽松管控线的权重，默认值0
* weight\_x\_strict：优化目标中，PPA\_X绝对值小于宽严格控线的权重，默认值0.25
* weight\_y\_strict：优化目标中，PPA\_Y绝对值小于宽严格控线的权重，默认值0.25
* weight\_t\_strict：优化目标中，PPA\_X、PPA\_Y绝对值最大值小于严格管控线的权重，默认值0
* delta\_OFFSET\_X\_low：改变OFFSET\_X的最小值，默认为0.5
* delta\_OFFSET\_X\_high：改变OFFSET\_X的最大值，默认为2.5
* delta\_OFFSET\_Y\_low：改变OFFSET\_Y的最小值，默认为0.5
* delta\_OFFSET\_Y\_high：改变OFFSET\_Y的最大值，默认为2.5
* delta\_OFFSET\_tht\_low：改变OFFSET\_tht的最小值，默认为4
* delta\_OFFSET\_tht\_high：改变OFFSET\_tht的最大值，默认为36
* n\_teg\_x：glass teg 在x方向个数，默认为19
* n\_teg\_y：glass teg 在y方向个数，默认为6

## 一些逻辑确认

1. 目前OC3腔室没有用，后面可能会用，数据中是否有OC3的记录，还不确定。（之前PPA数据中是可以抽取到EVA*CHAMBER==OC*B'的数据的，但其PPA数据均为-9999或9999，可以一直考虑OC3，在其无数据时令OFFSET为0）
2. Dummy工单：productspecname字段，值为647MQC（还是字符串包含647MQC？还是包含647且包含MQC）