# Cm数据采集 cm\_collect.py

**Ftp 文件处理部分,这个部分大概耗时为4分钟。**

1. 清理表数据,查询mes\_beforeafter\_setting a WHERE a.`groupname` = 'BEFORE\_COLLECT\_DATA\_CLEAN\_CM\_FTP' 数据并清空这些表的数据,对于兰州来说只需要关系lte\_hw的表就可以了

[

"cm\_lte\_enbfunction\_eric\_bak",

"cm\_lte\_enbfunction\_hw\_bak",

"cm\_lte\_enbfunction\_zte\_bak",

"cm\_lte\_eutrancelltdd\_erci\_bak",

"cm\_lte\_eutrancelltdd\_hw\_bak",

"cm\_lte\_eutrancelltdd\_zte\_bak",

"cm\_lte\_eutranrelationtdd\_eric\_bak",

"cm\_lte\_eutranrelationtdd\_hw\_bak",

"cm\_lte\_eutranrelationtdd\_zte\_bak",

"cm\_lte\_gsmrelation\_hw\_bak",

"cm\_lte\_gsmrelation\_zte\_bak",

"cm\_lte\_cellrfshutdown\_hw\_bak",

"cm\_lte\_celllowpower\_hw\_bak",

"cm\_lte\_cellalgoswitch\_hw\_bak",

"cm\_lte\_interratcellshutdown\_hw\_bak",

"cm\_lte\_enodebautopoweroff\_hw\_bak",

"cm\_lte\_cellshutdown\_hw\_bak",

"cm\_lte\_lst\_cagroupcell\_hw\_bak",

"cm\_lte\_dsp\_cell\_hw\_bak",

"cm\_lte\_lst\_cell\_hw\_bak"

]

1. 查询mes\_ftp\_command 中group\_name = cm的数据并按照每个server\_name启动一个线程处理(如果mes\_ftp\_command 都对应到mes\_servers表中的一条记录则总是单线程在处理)
2. 下载文件并按照目标表生成文件，需要生11个文件,原始文件命名为mes\_ftp\_command.name,文件命名为MainDB-[tableName]\_[Timestamp].data

{".\*CaGroupCell":"cm\_lte\_lst\_cagroupcell\_hw\_bak"}

{".\*CELLALGOSWITCH":"cm\_lte\_CELLALGOSWITCH\_hw\_bak"}

{".\*CELLLOWPOWER":"cm\_lte\_CELLLOWPOWER\_hw\_bak"}

{".\*CELLRFSHUTDOWN":"cm\_lte\_CELLRFSHUTDOWN\_hw\_bak"}

{".\*CELLSHUTDOWN":"cm\_lte\_CELLSHUTDOWN\_hw\_bak"}

{".\*EnbFunction":"cm\_lte\_enbfunction\_hw\_bak"}

{".\*ENODEBAUTOPOWEROFF":"cm\_lte\_ENODEBAUTOPOWEROFF\_hw\_bak"}

{".\*EutranCellTdd":"cm\_lte\_eutrancelltdd\_hw\_bak"}

{".\*EutranRelationTdd":"cm\_lte\_eutranrelationtdd\_hw\_bak"}

{".\*INTERRATCELLSHUTDOWN":"cm\_lte\_INTERRATCELLSHUTDOWN\_hw\_bak"}

{".\*EutranCellTdd":"cm\_lte\_dsp\_cell\_hw\_bak"}

1. 将文件数据导入到目标表中

# 由于指令错误，导致第一个表没有数据，其他的都有数据

SELECT \* FROM cm\_lte\_lst\_cagroupcell\_hw\_bak;

SELECT \* FROM cm\_lte\_CELLALGOSWITCH\_hw\_bak;

SELECT \* FROM cm\_lte\_CELLLOWPOWER\_hw\_bak;

SELECT \* FROM cm\_lte\_CELLRFSHUTDOWN\_hw\_bak;

SELECT \* FROM cm\_lte\_CELLSHUTDOWN\_hw\_bak;

SELECT \* FROM cm\_lte\_enbfunction\_hw\_bak;

SELECT \* FROM cm\_lte\_ENODEBAUTOPOWEROFF\_hw\_bak;

SELECT \* FROM cm\_lte\_eutrancelltdd\_hw\_bak;

SELECT \* FROM cm\_lte\_eutranrelationtdd\_hw\_bak;

SELECT \* FROM cm\_lte\_INTERRATCELLSHUTDOWN\_hw\_bak;

#在步骤5中该表的数据会被移动到cm\_lte\_dsp\_cell\_hw表中

SELECT \* FROM cm\_lte\_dsp\_cell\_hw\_bak;

1. 将部分bak表中的数据移动到非bak表，并清空bak表

[

"cm\_lte\_enbfunction\_eric",

"cm\_lte\_enbfunction\_hw",

"cm\_lte\_enbfunction\_zte",

"cm\_lte\_eutrancelltdd\_erci",

"cm\_lte\_eutrancelltdd\_hw",

"cm\_lte\_eutrancelltdd\_zte",

"cm\_lte\_eutranrelationtdd\_eric",

"cm\_lte\_eutranrelationtdd\_hw",

"cm\_lte\_eutranrelationtdd\_zte",

"cm\_lte\_cellrfshutdown\_hw",

"cm\_lte\_celllowpower\_hw",

"cm\_lte\_cellalgoswitch\_hw",

"cm\_lte\_interratcellshutdown\_hw",

"cm\_lte\_enodebautopoweroff\_hw",

"cm\_lte\_cellshutdown\_hw",

"cm\_lte\_lst\_cagroupcell\_hw",

"cm\_lte\_dsp\_cell\_hw",

"cm\_lte\_lst\_cell\_hw"

]

**TELNEt部分，这个部分如果单线程的情况，可能运行一个小时后超时的问题，2个线程大概需要运行30~60分钟。**

1.清空表数据mes\_beforeafter\_setting a WHERE a.`groupname` = 'BEFORE\_COLLECT\_DATA\_CLEAN\_CM\_FIRST';

[

"param\_lte\_hw\_login\_bak",

"param\_td\_hw\_login\_bak",

"param\_lte\_subnetwork\_zte\_bak",

"param\_td\_subnetwork\_zte\_bak"]

2. 执行，将数据倒入到bak表中,

通过mes\_log\_command a WHERE a.`group\_name` = 'FIRST'查询需要执行的telnet指令，根据log\_parser字段到mes\_log\_parser中查询数据需要写入的表,其实就是上一个步骤中清理的表SELECT \* FROM mes\_log\_parser a WHERE a.`name` = 'HW\_LTE\_LST\_NE';

3. 将bak表中的数据移动到非bak表中，mes\_beforeafter\_setting a WHERE a.`groupname` = 'AFTER\_COLLECT\_DATA\_IMPORTDATA\_CM\_FIRST';

[

"param\_lte\_hw\_login",

"param\_td\_hw\_login",

"param\_lte\_subnetwork\_zte",

"param\_td\_subnetwork\_zte"]

4. [这个步骤是针对zte设备]查询mes\_beforeafter\_setting a WHERE a.`groupname` = 'BEFORE\_COLLECT\_DATA\_CLEAN\_CM\_SECOND';数据并清空对应的表数据

["cm\_td\_tcell\_zte\_bak"]

5. [这个步骤是针对zte设备] 如果是华为的设备这个步骤可以不用

1. [这个步骤是针对zte设备]将 cm\_td\_tcell\_zte\_bak 表中的数据移动到cm\_td\_tcell\_zte表中mes\_beforeafter\_setting a WHERE a.`groupname` = 'AFTER\_COLLECT\_DATA\_IMPORTDATA\_CM\_SECOND';

["cm\_td\_tcell\_zte"]

1. 清空表数据，mes\_beforeafter\_setting a WHERE a.`groupname` = 'BEFORE\_COLLECT\_DATA\_CLEAN\_CM\_PARA';

["cm\_gsm\_lst\_bscdstpa\_hw\_bak",

"cm\_gsm\_lst\_g2gncell\_hw\_bak",

"cm\_gsm\_lst\_gcell\_hw\_bak",

"cm\_gsm\_lst\_gcellbasicpara\_hw\_bak",

"cm\_gsm\_lst\_gext2gcell\_hw\_bak",

"cm\_gsm\_rldep\_eric\_bak",

"cm\_gsm\_rldep\_ext\_eric\_bak",

"cm\_gsm\_rlnrp\_eric\_bak",

"cm\_gsm\_rlstp\_eric\_bak",

"cm\_gsm\_rlsvp\_eric\_bak",

"cm\_lte\_dsp\_cell\_hw\_bak",

"cm\_lte\_lst\_cell\_hw\_bak",

"cm\_lte\_lst\_cagroupcell\_hw\_bak",

"cm\_lte\_powersaving\_zte\_bak",

"cm\_lte\_querycellstatus\_zte\_bak",

"cm\_td\_lst\_tcell\_hw\_bak",

"cm\_td\_lst\_tgsmcell\_hw\_bak",

"cm\_td\_lst\_tgsmncell\_hw\_bak",

"cm\_td\_lst\_tncell\_hw\_bak",

"cm\_td\_lst\_tnrnccell\_hw\_bak",

"cm\_td\_nodebid\_zte\_bak",

"cm\_td\_rncid\_hw\_bak",

"cm\_td\_tcarrier\_hw\_bak",

"cm\_td\_tcarrier\_zte\_bak",

"cm\_td\_texternalgsmcell\_zte\_bak",

"cm\_td\_texternalltecell\_zte\_bak",

"cm\_td\_tgsmncell\_zte\_bak",

"cm\_td\_utrancell\_zte\_bak"]

1. 查询mes\_log\_command 中a.`group\_name` = 'PARA'对应的命令，并将指令返回的log日志写入到数据库中，写入的表对应的是mes\_log\_parser中配置的表；即mes\_log\_command.log\_parser = mes\_log\_parser.name

对于华为的设备来说只需要使用以下的配置即可

SELECT \* FROM mes\_log\_command a WHERE a.`group\_name` = 'PARA' AND a.`manufacturers` = 'LTE\_HW' AND a.`enabled` = 1

一共执行三条命令

REG NE:IP=[ip];#DSP CELL:;#UNREG NE:IP=[ip];#; -- > HW\_LTE\_DSP\_CELL\_H#HW\_LTE\_DSP\_CELL\_V#HW\_LTE\_DSP\_CELL

数据写入目标表配置

SELECT \* FROM mes\_log\_parser a WHERE a.`name` IN ('HW\_LTE\_DSP\_CELL\_H','HW\_LTE\_DSP\_CELL\_V','HW\_LTE\_DSP\_CELL');

对应的表为 cm\_lte\_dsp\_cell\_hw\_bak

REG NE:IP=[ip];#LST CAGROUPCELL:;#UNREG NE:IP=[ip];#; --> HW\_LTE\_LST\_CAGROUPCELL#HW\_LTE\_LST\_CAGROUPCELL\_H

写入的表

SELECT \* FROM mes\_log\_parser a WHERE a.`name` IN (' HW\_LTE\_LST\_CAGROUPCELL ',' HW\_LTE\_LST\_CAGROUPCELL\_H ');

对应的表为cm\_lte\_lst\_cagroupcell\_hw\_bak

REG NE:IP=[ip];#LST CELL:;#UNREG NE:IP=[ip];#; --> HW\_LTE\_LST\_CELL\_H#HW\_LTE\_LST\_CELL\_V

写入的表

SELECT \* FROM mes\_log\_parser a WHERE a.`name` IN (' HW\_LTE\_LST\_CELL\_H ',' HW\_LTE\_LST\_CELL\_V ');

对应的表为

cm\_lte\_lst\_cell\_hw\_bak

1. 读取mes\_mrdetail\_model配置，生成rst\_\*\_mr\_detail 格式的表

综述:

Cm采集将向以下表中写入数据

**FTP写入**

cm\_lte\_lst\_cagroupcell\_hw

cm\_lte\_cellalgoswitch\_hw

cm\_lte\_celllowpower\_hw

cm\_lte\_cellrfshutdown\_hw

cm\_lte\_cellshutdown\_hw

cm\_lte\_enbfunction\_hw

cm\_lte\_enodebautopoweroff\_hw

cm\_lte\_eutrancelltdd\_hw

cm\_lte\_eutranrelationtdd\_hw

cm\_lte\_interratcellshutdown\_hw

cm\_lte\_dsp\_cell\_hw

**Fisrt步骤写入**

param\_lte\_hw\_login

**SECOND处理的是ZTE的数据，兰州不考虑**

cm\_td\_tcell\_zte

**PARA步骤写入**

cm\_lte\_dsp\_cell\_hw\_bak

cm\_lte\_lst\_cagroupcell\_hw\_bak

cm\_lte\_lst\_cell\_hw\_bak

**improveMrData将要生成rst\_\*\_mr\_detail 格式的表数据**

rst\_l2l\_mr\_detail

该数据的生成需要依赖以下表数据, rst\_l2l\_mro\_hw 需要在mr解析的时候生成，因此cm采集前应该已经做过mr的采集和解析了。基站的基础信息也是需要倒入的。

cm\_lte\_eutranrelationtdd\_hw

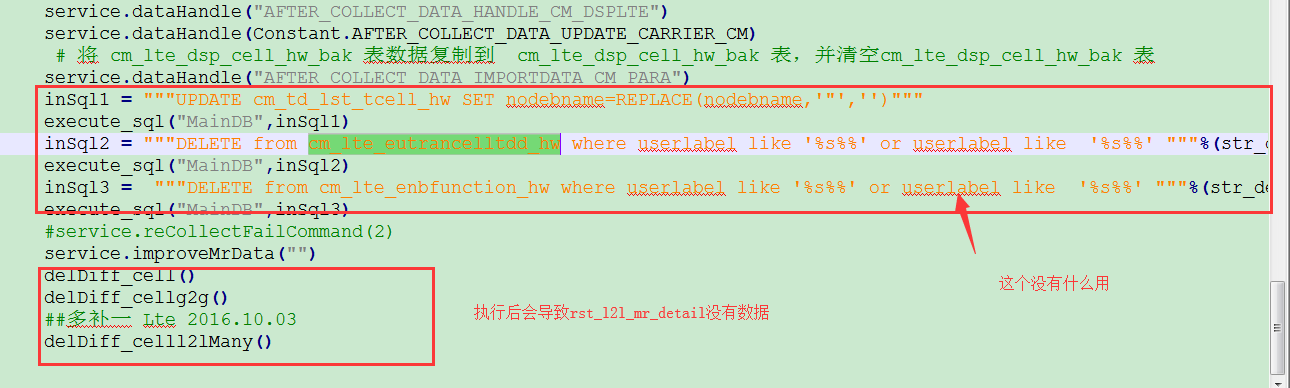
cm\_lte\_enbfunction\_hw

cm\_lte\_eutrancelltdd\_hw

rst\_l2l\_mro\_hw

mes\_base\_info\_lte

1. **mes\_log\_command，mes\_ftp\_command表中的filter，远程目录都配置都需要根据实际情况来配置**
2. **需要将mes\_appsetting表中aos.send\_flag 设置为true，否则采集不到telnet日志，全是test send**
3. **需要删除以下代码**



脚本：cm\_collect.py

配置：

目录：

输入：

ftp文件 && telnet 返回

输出：

ftp：  
cm\_lte\_enbfunction\_hw\_bak  
cm\_lte\_eutrancelltdd\_hw\_bak  
cm\_lte\_eutranrelationtdd\_hw\_bak  
  
telnet：  
cm\_lte\_dsp\_cell\_hw\_bak  
param\_lte\_hw\_login\_bak  
cm\_lte\_lst\_cagroupcell\_hw\_bak  
cm\_lte\_lst\_cell\_hw\_bak  
  
rst\_l2l\_mr\_detail

# Mro数据采集 mro\_collect\_lte\_hw.py

读取SELECT \* FROM mes\_ftp\_command a WHERE a.`group\_name` = 'LTE\_MRO' AND a.`enabled` = 1配置，下载文件并将文件写入到pm\_lte\_mro\_hw表中。

脚本： mro\_collect\_lte\_hw.py

配置：

目录：

输入： ftp文件

输出：

pm\_lte\_mro\_hw

# Mro文件解析 test\_mr\_paser\_sm.py

解析已经下载的mro文件

1. 根据正则表达式匹配指定目录下的mro文件，并生成scv文件，每个文件500行，每一行都是一个mro文件名称
2. 使用LteHWCellCollectOutPutHandle解析文件并生成csv文件
3. 将csv文件数据导入到表中，LTE\_NCELL\_INFO 格式的文件数据导入到rst\_l2l\_hw\_hz\_sm表中，其他格式的文件数据导入到rst\_l2l\_hw\_nc\_sm表中

脚本：test\_mr\_paser\_sm.py

配置：

目录：

输入：

Mro ftp文件

输出:

LTE\_NCELL\_INFO 格式的文件🡪 rst\_l2l\_hw\_nc\_sm <目前使用的>

rst\_pm\_l2l\_hw\_hz  （脚本没有携带表后缀时）

非LTE\_NCELL\_INFO 格式的文件🡪 rst\_l2l\_hw\_hz\_sm<目前使用的>

rst\_pm\_l2l\_hw\_nc\_info （脚本没有携带表后缀时）

* **解析生成的****rst\_l2l\_hw\_hz\_sm将在重叠覆盖计算时用于生成rst\_l2l\_mro\_hw表数据。**

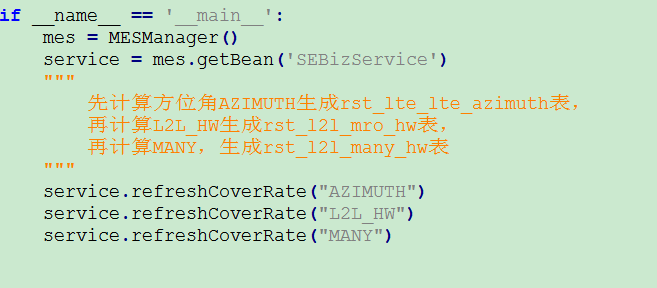
# 重叠覆盖度计算 mro\_over\_cal.py

概述

重叠覆盖度计算需要先计算方位角，在计算L2L\_HW生成rst\_l2l\_mro\_hw，最后计算MANY，生成rst\_l2l\_many\_hw表，因为计算MANY的时候需要依赖rst\_l2l\_mro\_hw数据。

基于mr的重叠覆盖计算根据邻区关系信息和mr数据计算筛选g2g,t2g,l2t,l2l覆盖情况；

基于方位角的重叠覆盖计算，根据小区的工参数据，通过经纬度计算两个小区的距离，筛选出小区距离小于50米的共站小区，输出g2g共站列表，t2g共站列表，l2t共站列表和l2l共站列表，再从四个共站列表中筛选，方位角偏差10度的为补偿小区。



流程说明：

需要先计算一补一再计算多补一的情况，根据数据的前后依赖关系配置脚本的执行顺序。

1. 通过groupname 查询mes\_over\_setting中的表数据，groupname为空时，先查询一补一再查询多补一
2. 遍历上一步的查询结果
3. 通过mes\_over\_setting.querysql查询数据并生成临时表，如果数据超过80000条则分批处理
4. 使用handle处理（简单计算）querysql查询到的数据并将数据写入到csv中
5. 将写入到csv文件的数据导入到mes\_over\_setting.res\_table中

脚本：mro\_over\_cal.py

配置：mes\_over\_setting

目录：

输入：

mes\_base\_info\_lte  🡪用于计算方位角生成rst\_lte\_lte\_azimuth数据

**rst\_l2l\_hw\_hz\_sm**  🡪 用于mr生成 rst\_l2l\_mro\_hw

rst\_l2l\_mro\_hw，**rst\_l2l\_hw\_hz\_sm 🡪** **rst\_l2l\_many\_hw**

输出：

rst\_lte\_lte\_azimuth

rst\_l2l\_mro\_hw

rst\_l2l\_many\_hw

# 大数据预测功能 bigdata\_forecast.py (rst\_bigdata\* 数据)

根据不同制式的pm\_gsm|td|lte\_\*\_his历史数据生成 rst\_bigdata\_pm开头的预测表数据。核心思路就是通过过去30天同一时段的pm均值来预测明天同一时段的pm值。

1. 查询mes\_bigdata\_forecast\_setting表配置，按需要配置
2. 组装过去同一时刻时间值，如’2018-08-29 00:15:00’, ’2018-08-30 00:15:00’……
3. 将时间作为条件查询pm\_lte\_eutrancelltdd\_hw\_his表数据获取均值
4. 生成文件，并将文件导入到mes\_bigdata\_forecast\_setting.res\_table 中

脚本：bigdata\_forecast.py

配置：

mes\_bigdata\_forecast\_setting.lte

目录：

输入：

pm\_lte\_eutrancelltdd\_hw\_his

输出：

rst\_bigdata\_pm\_lte\_eutrancelltdd\_hw

* **大数据预测功能生成的rst\_bigdata\_pm\_lte\_eutrancelltdd\_hw表数据将在计算历史Kpi时被用到**

# Pm数据采集 pm\_collect.py

1. 将数据移动到历史表表， mes\_beforeafter\_setting a WHERE a.`groupname` = 'BEFORE\_COLLECT\_DATA\_ADD\_BATCHID'

["pm\_gsm\_cellgprs\_eric",

"pm\_gsm\_cellqoseg\_eric",

"pm\_gsm\_cellqosg\_eric",

"pm\_gsm\_celtchf\_eric",

"pm\_gsm\_celtchfp\_eric",

"pm\_gsm\_celtchh\_eric",

"pm\_gsm\_cltch\_eric",

"pm\_gsm\_mrf\_4c0003ed\_hw",

"pm\_gsm\_trafdlgprs\_eric",

"pm\_lte\_eutrancelltdd\_eric",

"pm\_lte\_eutrancelltdd\_hw",

"pm\_lte\_eutrancelltdd\_zte",

"pm\_td\_carrier\_hw\_cell",

"pm\_td\_carrier\_zte\_cell",

"pm\_td\_utrancell\_hw",

"pm\_td\_utrancell\_zte"]

1. 清空频载数据，这个和lte的没有关系

["pm\_td\_carrier\_hw","pm\_td\_carrier\_zte"]

1. 文件下载，这里先查询了下mes\_bscname\_config表，不清楚有什么鸟用。查询mes\_ftp\_command中的pm配置,按照mes\_ftp\_command.server\_name进行分组，每个server\_name启动一个线程进行ftp文件下载
2. 将文件数据导入到数据库中 pm\_lte\_eutrancelltdd\_hw 表中，目标表在mes\_ftp\_command中进行配置
3. 更新["pm\_td\_utrancell\_hw","pm\_td\_carrier\_hw"]表数据,3G退网了这个没有什么用了
4. 更新pm\_td\_carrier\_hw\_cell,pm\_td\_carrier\_zte\_cell数据，这个没什么用了
5. 更新pm\_lte\_eutrancelltdd\_eric表数据，eric的需要关注

**Pm的大部分采集都是执行以上的步骤，之后休眠时段才会执行下面的步骤**

1. 查询mes\_zs\_open\_period数据，看当前是否为节能时段
2. 计算历史kpi，这里我们只看hw lte的场景(具体请参考PM->历史Kpi计算)

查询mes\_kpical\_set配置，获取kpi计算sql，如果配置中delflag为true则需要先将mes\_lte\_kpi表的数据移动到历史表，并删除mes\_lte\_kpi表中的数据。

对于hw lte来说，需要依赖

rst\_bigdata\_pm\_lte\_eutrancelltdd\_hw,

cm\_lte\_eutrancelltdd\_hw,

cm\_lte\_enbfunction\_hw三个表的数据

1. 将当前休眠小区写入到历史表mes\_l2l\_currentsleep🡪mes\_l2l\_currentsleep\_his
2. 删除上一时段未下发的记录,删除mes\_l2l\_currentsleep表中数据，和上一个步骤组成一个备份步骤
3. 休眠小区监控
4. 将mes\_adjust\_command数据写入到mes\_adjust\_command\_his表中，并删除mes\_adjust\_command表数据
5. 查询mes\_montior\_setting监控流程配置生成mes\_lte\_nonsleep表数据，生成mes\_lte\_nonsleep表数据需要依赖三个表进行关联，分别是

mes\_lte\_real\_kpi,

mes\_l2l\_many\_currentsleep,

mes\_l2l\_currentsleep

1. 如果mes\_lte\_nonsleep数据中存在差指标的，则更新到黑名单中mes\_lte\_black
2. 查询mes\_notify\_setting表数据，如果mes\_notify\_setting表配置的是差小区，则生成所有的小区的唤醒指令,唤醒时需要查询mes\_notify\_setting表数据并依赖三个表的数据

mes\_lte\_nonsleep,mes\_l2l\_many\_currentsleep,cm\_lte\_dsp\_cell\_hw

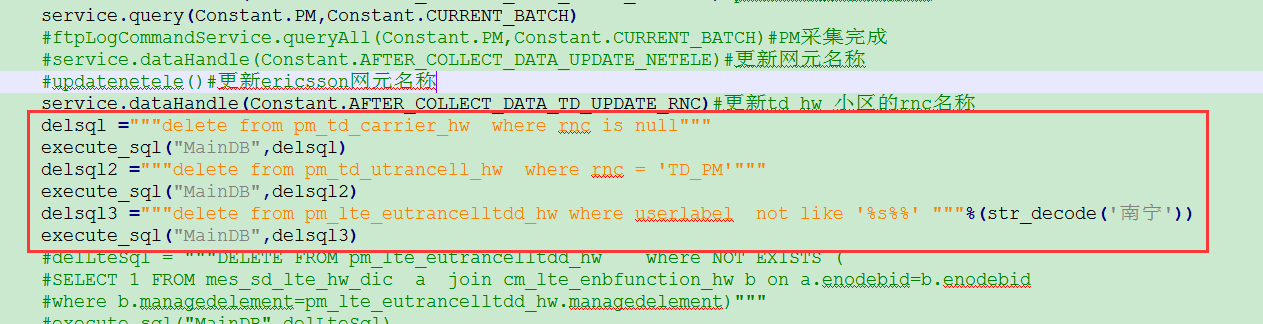
1. 唤醒休眠小区，查询mes\_adjust\_command表获取需要唤醒的小区，如果唤醒成功向mes\_lte\_commandlog写入唤醒记录，如果唤醒失败，记录黑名单和告警mes\_lte\_black,mes\_lte\_selfalarm
2. 将唤醒失败的mes\_adjust\_command\_his表数据，写入到mes\_l2l\_currentsleep\_fail中

**上一个部分是休眠时段对小区的监控流程和小区的唤醒流程，下面的步骤是休眠小区筛选，冲突处理，指令生成，休眠指令下发流程**

1. 刷新节能小区(理论上，9-14步骤和后面的步骤不会同时执行),查询mes\_celllist\_ref表配置，生成rst\_\*\_energyref\_mr类型的表，对hw lte而已就是将select \* from rst\_l2l\_mr\_detail where sstate=1 and nstate=1的查询结果写入得到rst\_l2l\_energyref\_mr表中
2. 休眠小区筛选
3. 冲突处理
4. 指令生成
5. 休眠指令下发

**注意的地方**

1.需要删除或者注释掉如下图被选中的内容



脚本：pm\_collect.py

配置：

目录：

输入：

ftp文件

mes\_lte\_real\_kpi,

mes\_l2l\_many\_currentsleep,

mes\_l2l\_currentsleep

输出：

Pm采集

pm\_lte\_eutrancelltdd\_hw

Kpi计算

mes\_lte\_kpi

小区监控

mes\_lte\_nonsleep

唤醒小区

mes\_lte\_commandlog

mes\_lte\_black,mes\_lte\_selfalarm

mes\_l2l\_currentsleep\_fail

mes\_adjust\_command\_his

刷新节能小区

rst\_l2l\_energyref\_mr

# PM->历史Kpi计算

通过rst\_bigdata\_pm\_lte\_eutrancelltdd\_hw , cm\_lte\_eutrancelltdd\_hw, cm\_lte\_enbfunction\_hw 表数据的关联生成mes\_lte\_kpi表数据，该脚本在代码里面已经写死，只能在0~6点之间执行，其他时间都不会执行。

脚本：bigdata\_next.py

配置：mes\_kpical\_set.BIGDATA\_KPI\_CAL\_HIS

目录：

输入：

rst\_bigdata\_pm\_lte\_eutrancelltdd\_hw JOIN  
cm\_lte\_eutrancelltdd\_hw JOIN  
cm\_lte\_enbfunction\_hw

输出：

mes\_lte\_kpi

* **该步骤输出的mes\_lte\_kpi表数据将在休眠小区筛选时用到。**

# PM->休眠小区筛选

筛选休眠小区：

1. 查询busytype值，该值由mes\_zs\_coverage.base\_overlay\_degree确认，如果base\_overlay\_degree=1则为true，否则为false
2. 如果busytype值为true，则group\_name = azimuth\_l2l,为false则group\_name = mr\_l2l
3. 使用上一步骤得到的group\_name值作为参数查询mes\_sleepsel\_setting表数据
4. 如果mes\_sleepsel\_setting.del\_flag = 1 则先删除mes\_sleepsel\_setting.res\_table数据
5. 通过mes\_sleepsel\_setting.querysql查询数据
6. 通过mes\_sleepsel\_setting.service\_handle处理数据
7. 生成csv文件
8. 将csv文件数据导入到mes\_sleepsel\_setting.res\_table中

脚本： pm\_collect.py

配置： mes\_sleepsel\_setting.'%l2l'

目录：

输入：

rst\_l2l\_energyref\_azimuth JOIN  
mes\_lte\_kpi  
  
rst\_l2l\_energyref\_mr JOIN  
mes\_lte\_kpi  
  
rst\_l2l\_energyref\_many JOIN  
mes\_lte\_kpi

输出：

mes\_l2l\_sleep\_azimuth  
  
mes\_l2l\_sleep\_mr  
  
mes\_l2l\_sleep\_many

mes\_l2l\_mobile\_table;

# PM->休眠小区冲突处理

由于小区既有可能是休眠小区，又可能是补偿小区业务场景的存在，导致，需要对小区休眠做一个优先级，尤其是在不同制式的场景下，如小区GSM-A是 TD-B的休眠小区，又是Lte的补偿小区时，应该把GSM优先作为休眠小区而不是做为补偿小区，通常情况下休眠优先级从高到低依次是GSM>TD>LTE。

相同制式，小区是休眠小区，也是补偿小区，则该小区只做休眠小区，不做补偿小区考虑，需要将作为补偿小区的一组数据删除，避免被补偿的小区休眠导致业务异常。

# PM->休眠指令生成

脚本：pm\_collect.py

配置：mes\_sleepexe\_setting.'l2l'

目录：

输入：

mes\_l2l\_sleep\_azimuth  
mes\_l2l\_sleep\_many  
mes\_l2l\_sleep\_mr  
  
cm\_lte\_powersaving\_zte  
  
cm\_lte\_dsp\_cell\_hw  
param\_lte\_hw\_login  
cm\_lte\_lst\_cagroupcell\_hw  
mes\_l2l\_currentsleep

输出：

mes\_adjust\_command

# PM->唤醒指令生成

系统可以通过2种方式生产唤醒指令，一个是通过执行notifyall.py 唤醒所有的小区来生产休眠指令并唤醒小区；另一个方式是通过pm里面的监控来生产唤醒指令。

使用配置

SELECT \* FROM mes\_notify\_setting a WHERE a.`enabled` = 1 AND a.`groupname` = 'lte';

里面用到的查询sql

SELECT

l2l.src\_enodebid,

l2l.src\_localcellid,

l2l.src\_vender,

l2l.omm,

l2l.subnetwork,

l2l.userlabel,

l2l.ip,

l2l.dest\_enodebid,

l2l.dest\_localcellid,

l2l.dest\_vender,

non.starttime,

'l2l' bus\_type,l2l.cagroupid,l2l.preferredpcellpriority

FROM

mes\_l2l\_currentsleep l2l

JOIN mes\_lte\_nonsleep non ON non.enodebid = l2l.dest\_enodebid

AND non.localcellid = l2l.dest\_localcellid

LEFT JOIN mes\_l2l\_many\_currentsleep l2ml

ON l2l.dest\_enodebid = l2ml.dest\_enodebid

AND l2l.dest\_localcellid = l2ml.dest\_localcellid

WHERE 1=1

# 加了下面这2个条件会没有数据，应该是当前的数据问题，不存在弱小区

#and ismakeup = 1

# and kpiweak=1

AND l2ml.dest\_enodebid IS NULL

AND l2l.src\_vender = '华为'

AND non.starttime= '2018-10-19 05:30:00'

UNION ALL

SELECT

l2l.src\_enodebid,

l2l.src\_localcellid,

l2l.src\_vender,

l2l.omm,

l2l.subnetwork,

l2l.userlabel,

l2l.ip,

l2l.dest\_enodebid,

l2l.dest\_localcellid,

l2l.dest\_vender,

non.starttime,

'l2l\_many' bus\_type,cagroupid,preferredpcellpriority

FROM

mes\_l2l\_many\_currentsleep l2l

JOIN mes\_lte\_nonsleep non ON non.enodebid = l2l.dest\_enodebid

AND non.localcellid = l2l.dest\_localcellid

WHERE

ismakeup = 1

AND kpiweak=1

AND l2l.src\_vender = '华为'

AND non.starttime= '2018-10-19 05:30:00';

# PM->休眠指令下发

先休眠一补一的，再休眠多补一的 ;唤醒的时候刚好相反，需要先唤醒多补一的，再唤醒一补一的。通过 SELECT \* FROM mes\_adjust\_command a WHERE a.`group\_name` = 'SLEEP' AND a.`applied` = 0 AND a.`app\_name` = 'MULTINET' 查询数据,并循环;如果是唤醒查询语句为WHERE a.`group\_name` = ' NOTIFY' AND a.`applied` = 0 AND a.`app\_name` = 'MULTINET'。

执行指令 。

脚本：

配置：

目录：

输入：

mes\_adjust\_command

输出：

休眠成功 mes\_lte\_commandlog  
  
休眠失败 mes\_td\_black  
 mes\_subscribe\_lte\_selfalarm

# PM->Real Kpi计算

脚本：pm\_collect.py

配置：mes\_kpical\_set.KPI\_CAL\_REAL

目录：

输入：

pm\_lte\_eutrancelltdd\_hw JOIN

cm\_lte\_eutrancelltdd\_hw JOIN

cm\_lte\_enbfunction\_hw

输出：

mes\_lte\_real\_kpi

# 历史数据预测 8:00

每天8:00计算能时段（ 0:00-6:00）的业务模型，根据 cm\_\* 表和pm\_\*\_his表生成rst\_\* 表。

根据\*表的数据量进行自动适配查询三个月，一个月还是七天的数据。



根据历史数据的存储量使用不同的计算公式，如果系统有的n月（周）的数据则:

满足三个月历史数据(n > 3m)：

下一时段预测业务量=前第一个月同期下一时段业务流量均值\*0.3+前第二个月同期下一时段业务量均值\*0.2+前第3至6个月同期下一时段业务量均值\*0.1

满足2个月不到三个月(2m < n < 3m):

下一时段预测业务量=前第一个月同期下一时段业务流量均值\*0.3+前第二个月同期下一时段业务量均值\*0.2

满足1个月不到2个月(1m < n < 2m):

下一时段预测业务量=前第一个月同期下一时段业务流量均值\*0.4

满足一周不满足一个月(1w < n < 1m):

下一时段预测业务量=前第一个月同期下一时段业务流量均值\*0.4

不满足一周 (n < 1w):

下一时段预测业务量=前第一个月同期下一时段业务流量均值\*0.5

脚本：his\_fcast.py

配置：mes\_fcast\_setting

目录：

输入：

cm\_gsm\_cellgprs\_eric && pm\_gsm\_cellgprs\_eric\_his

cm\_gsm\_cellqoseg\_eric && pm\_gsm\_cellqoseg\_eric\_his

cm\_gsm\_celtchf\_eric && pm\_gsm\_celtchf\_eric\_his

cm\_gsm\_cellqosg\_eric && pm\_gsm\_cellqosg\_eric\_his

cm\_gsm\_celtchfp\_eric && pm\_gsm\_celtchfp\_eric\_his

cm\_gsm\_celtchh\_eric && pm\_gsm\_celtchh\_eric\_his

cm\_gsm\_cltch\_eric && pm\_gsm\_cltch\_eric\_his

cm\_gsm\_mrf\_4c0003ed\_hw && pm\_gsm\_mrf\_4c0003ed\_hw\_his

cm\_td\_carrier\_hw\_cell && pm\_td\_carrier\_hw\_cell\_his

cm\_td\_carrier\_zte\_cell && pm\_td\_carrier\_zte\_cell\_his

cm\_td\_utrancell\_hw && pm\_td\_utrancell\_hw\_his

cm\_td\_utrancell\_zte && pm\_td\_utrancell\_zte\_his

cm\_lte\_eutrancelltdd\_hw && pm\_lte\_eutrancelltdd\_hw\_his

cm\_lte\_eutrancelltdd\_zte && pm\_lte\_eutrancelltdd\_zte\_his

cm\_lte\_eutrancelltdd\_eric && pm\_lte\_eutrancelltdd\_eric\_his

cm\_gsm\_trafdlgprs\_eric && pm\_gsm\_trafdlgprs\_eric\_his

输出：

rst\_pm\_gsm\_cellgprs\_eric

rst\_pm\_gsm\_cellqoseg\_eric

rst\_pm\_gsm\_celtchf\_eric

rst\_pm\_gsm\_cellqosg\_eric

rst\_pm\_gsm\_celtchfp\_eric

rst\_pm\_gsm\_celtchh\_eric

rst\_pm\_gsm\_cltch\_eric

rst\_pm\_gsm\_mrf\_4c0003ed\_hw

rst\_pm\_td\_carrier\_hw\_cell

rst\_pm\_td\_carrier\_zte\_cell

rst\_pm\_td\_utrancell\_hw

rst\_pm\_td\_utrancell\_zte

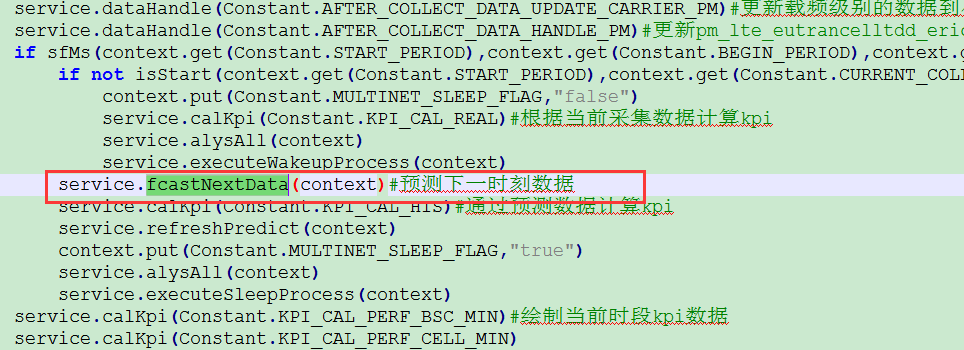
rst\_pm\_lte\_eutrancelltdd\_hw

rst\_pm\_lte\_eutrancelltdd\_zte

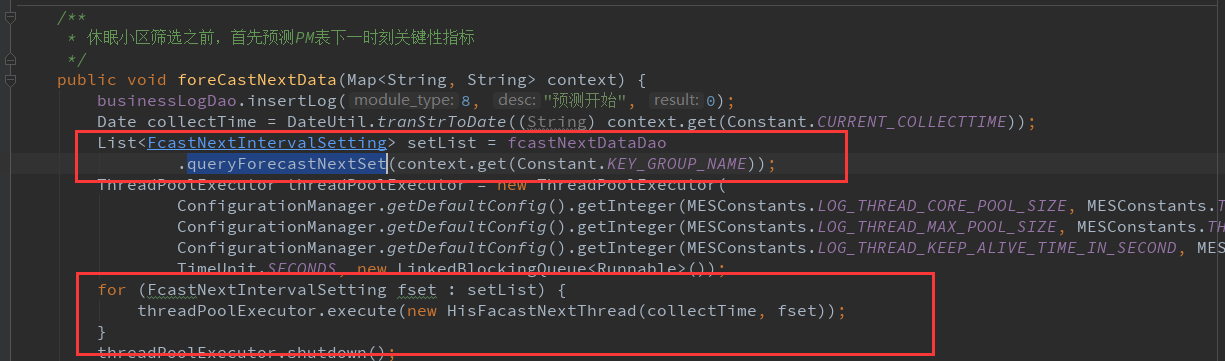
rst\_pm\_lte\_eutrancelltdd\_eric

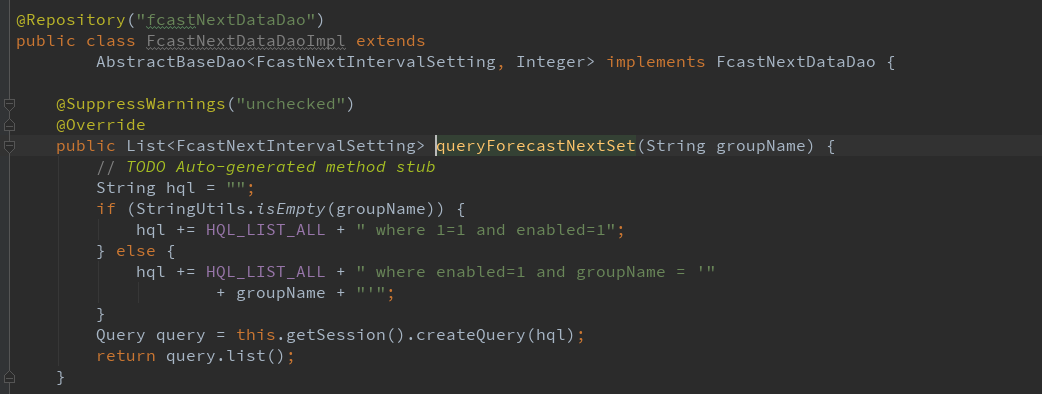
rst\_pm\_gsm\_trafdlgprs\_eric

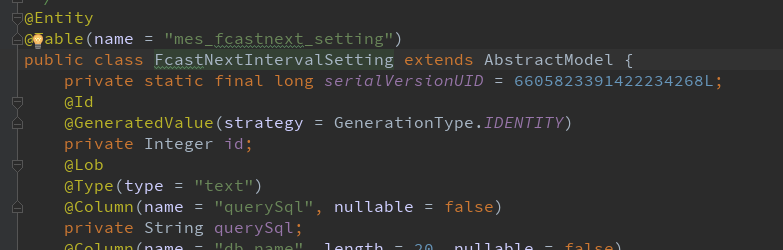
这里生产的rst\_pm\_lte\_eutrancelltdd\_hw表在pm中会使用到，在pm\_collect.py脚本中service.fcastNextData(context)#预测下一时刻数据，执行方法后生成rst\_fcastnext\_pm\_lte\_eutrancelltdd\_hw表数据,生产的数据在mes\_kpical\_set中用到id = 9用于计算kpi











# Busytype 值说明

mes\_zs\_coverage 表中base\_overlay\_degree = 1时Busytype=true，否则 Busytype=false

对于只有4g的场景下，建议只打开l2l的场景。

# rst\_l2l\_energyref\_\*生产规则

SELECT \* FROM mes\_celllist\_ref a WHERE a.`res\_table` LIKE 'rst\_l2l\_energyref\_%' AND a.`enabled` = 1;

查询出来的sql就是rst\_l2l\_energyref\_ 表对应的数据生成sql

# rst\_bigdata\_pm\_\* 表数据生成规则

所有的rst\_bigdata\_pm打头的表都是有bigdata\_forecast.py运行生成的。

# rst\_\*\_mr\_detail 表生成规则

通过rst\_\*\_mr\_detail 格式的表都是通过mes\_mrdetail\_model配置生成的，都是在执行cm\_collect.py脚本时通过service.improveMrData("")生成的。如果只有l2l场景则只需要执行service.improveMrData("l2l")即可。

# rst\_lte\_lte\_azimuth

rst\_lte\_lte\_azimuth 表数据由mes\_base\_info\_lte 自己关联后得到数据，数据生成规则配置为

SELECT \* FROM mes\_over\_setting a WHERE a.`result\_table` = 'rst\_lte\_lte\_azimuth';

rst\_l2l\_mro\_hw 数据由 rst\_l2l\_hw\_hz\_sm 生成

rst\_l2l\_many\_hw 数据由rst\_l2l\_hw\_hz\_sm a LEFT JOIN rst\_l2l\_mro\_hw 生成。

# mes\_fcastnext\_setting 作用？

SELECT \* FROM mes\_fcastnext\_setting a WHERE a.`source\_table` LIKE '%hw%' AND a.`source\_table` LIKE '%lte%';

# mes\_fcast\_setting 作用说明

用于历史数据预测，生成rst\_ 打头的表。

# 脚本作用说明

alarm.py

alystest.py

azimuth.py

azimuth\_converge.py

bigdata\_forecast.py

根据过去30天的pm数据预测下一天的pm数据生成rst\_bigdata\_pm 打头的表

bigdata\_next.py

生产历史Kpi数据,mes\_lte\_kpi数据

cell\_sleep\_time.py

checkcmdserver.py

cleanHisData.py

清楚历史数据

cm\_collect.py

cm 数据采集解析&& rst\_\*\_mr\_detail数据生成

his\_fcast.py

生成rst\_pm\_ 相关的表数据

mro\_collect\_\*.py

mro数据采集

mro\_over\_cal.py

计算重叠覆盖度，生成方位角和mr覆盖数据

notifyall.py

唤醒所有数据

parameters.py

提供几个公用方法

performance\_cal\_day.py

按天，小时，分钟维度计算kpi数据

pm\_collect.py

pm采集解析，休眠小区的筛选，监控，休眠，唤醒

test\_mr\_paser\_sm.py

mro数据解析，test\_mr\_paser\_\*.py的脚本都是用于解析mro，但是其他的几类暂时没有发现有什么用

# 还无法解释的坑

这里记录的都是目前还没有搞清楚的问题。

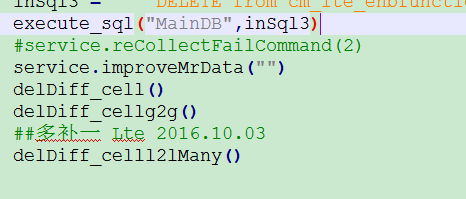
## 数据计算规则

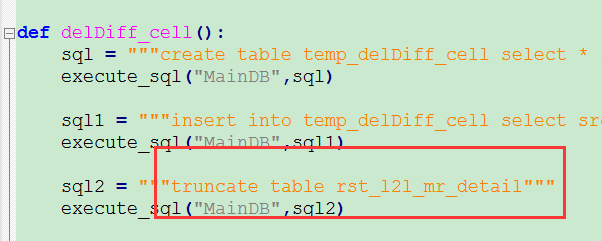
在代码里面涉及到数据计算规则不清楚，可参考的资料是之前的概要设计里面提到的几个公式。

## rst\_\*\_mr\_detail 数据先写入再删除

在cm\_collect.py脚本中，先执行service.improveMrData("") 通过mes\_mrdetail\_model配置将数据写入到了rst\_\*\_mr\_detail 表中，随后又执行了delDiff\*()方法将数据全部删除。

这里会体现一个基站距离的计算，当2个基站的距离小于500米的时候，数据才会被写入到rst\_\*\_mr\_detail表中，而且这里只会计算g2g,l2l的数据。





# Mr，CM，PM数据作用及采集频率

基于历史的mr数据筛选可休眠的小区，筛选出来的小区列表就是系统具备休眠条件的小区，一般来说这些都是相对固定的，即系统可休眠的小区基本上是固定的。是否休眠需要根据pm的实时数据来决定，如果小区当前pm指标出现异常则不能休眠小区。Cm为小区的工参数据，下斜角，载频等会经常发生变化，但是基本信息不变。

但是由于MR数据特别大，需要每天采集前一天的数据，月末再执行重叠覆盖度计算逻辑。由于MR数据的生成多数都存在很大的延时问题，故我们将前一天MR数据的采集时间定为第二天13点采集前一天的数据，即当天13点采集前一天全天的mr数据。

PM（话统）数据只需要采集节能时段的pm数据及可。

Cm（工参）：为保证小区基础工参的准确性，通过mml（telnet）指令和ftp文件方式采集，每天22点提前采集小区的工参数据，第一次部署为实时采集。

告警数据：用于逻辑判断，每15分钟检查一次最新的告警数据。

MR数据采集为每天采集前以前的，计算是按月为一周期；历史业务模型预测周期为每天，时间为每天8点对历史进行预测，生产历史kpi数据；每天23点更新小区节能状态表，结合黑白名单对小区状态进行监控；



主逻辑流程分三部分：月粒度执行部分、天粒度执行部分、15分钟粒度执行部分，具体流程如下：

按月周期：

1、按月为周期，更新GSM/TD-SCDMA/LTE各网制式及设备的MR数据，如果没有MR数据的根据手动跟新粒度更新工参数据。

2、判断GSM/TD-SCDMA/LTE各网络制式及设备是否有MR采集数据，如果是有MR数据，则基于采集的MR数据结合邻区关系进行各网络对应的重叠覆盖度的计算；如果否没有MR数据，则基于手动提供的小区工参基础信息进行各网络的重叠覆盖度计算；

3、根据前面的重叠覆盖度计算结果，筛选输出GSM/TD-SCDMA/LTE各网络制式的节能小区表，节能小区包括休眠小区和补偿小区及对应关系；

按天周期：

4、按天为周期，在每天08:00开始对历史业务统计模型进行预测，根据存储的历史数据对节能时段（00:00-06:00）每个15分钟时段业务进行历史预测；

5、在每天23:00更新节能小区状态表，确保GSM/TD-SCDMA/LTE各网络小区状态表和现网运行和黑白名单状态保持一致；

15分钟周期：

6、15分钟粒度为周期，（1）每个周期开始时读取小区状态表和黑白名单；（2）同时对GSM/TD-SCDMA/LTE各网小区15分钟粒度监控小区KPI和告警，并根据监控情况实时处理告警、差小区、以及补偿唤醒小区流程；

分支一：

7、根据读取GSM/TD-SCDMA/LTE各网络小区状态和黑白名单信息，对于上次周期结果列表中开启的小区进行各网络休眠和补偿小区筛选；输出各网络的休眠小区和补偿小区列表；

8、根据各网络制式的节能优先级设置情况，对各网络的休眠小区和补偿小区冲突处理，确定优先节能的制式小区，消除既是休眠小区有作为补偿小区情况，最后输出可执行的休眠小区和对应的补偿小区列表。

9、根据最后输出的可执行的休眠小区和补偿小区信息列表，对休眠小区进行休眠操作执行。

分支二：

10、根据读取GSM/TD-SCDMA/LTE各网络小区状态和黑白名单信息，对于上次周期结果列表中关闭的小区实时监控其补偿小区；

11、如果监控到补偿小区有告警或业务负荷超过设置门限，则根据相应的唤醒规则对选择对应的唤醒小区，输出唤醒小区列表信息；

12、对唤醒小区列表信息执行唤醒操作；

合并：

13、根据本15分钟时段执行休眠和唤醒操作以及全网小区监控结果，更新节能小区状态表和黑白名单（黑名单添加需满足命令执行失败小区和连续N次出现指标差和高级别告警的小区，N默认为3次）。

# Pm数据prb利用率为0 ？

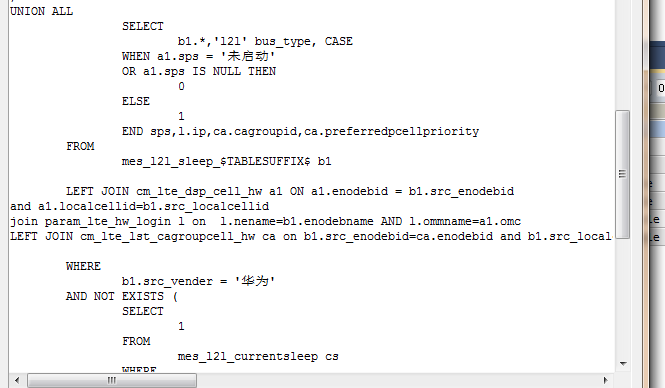
# Pm，cm，mr数据处理流 ？

那些任务使用到了pm，cm，mr数据，每个任务数据处理的目标和作用。

# cm\_lte\_dsp\_cell\_hw 数据来源及作用

Cm采集时会把数据存储到cm\_lte\_dsp\_cell\_hw\_bak表中，再通过备份的方式将数据移动到cm\_lte\_dsp\_cell\_hw表中。每次cm采集都会把cm\_lte\_dsp\_cell\_hw\_bak表清空通过将采集的cm\_lte\_dsp\_cell\_hw\_bak中的数据移动到cm\_lte\_dsp\_cell\_hw表中。

这个表的数据将在生成休眠指令的时候用到，进行关联查询数据。



# 数据验证 ？

Mr重叠覆盖数据，方位角覆盖数据，休眠小区筛选数据 验证问题

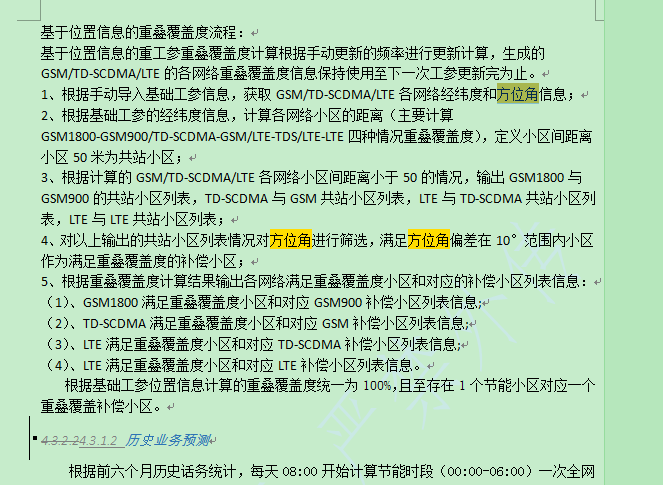
只能人工去看，或者找移动的要已经算出来的数据

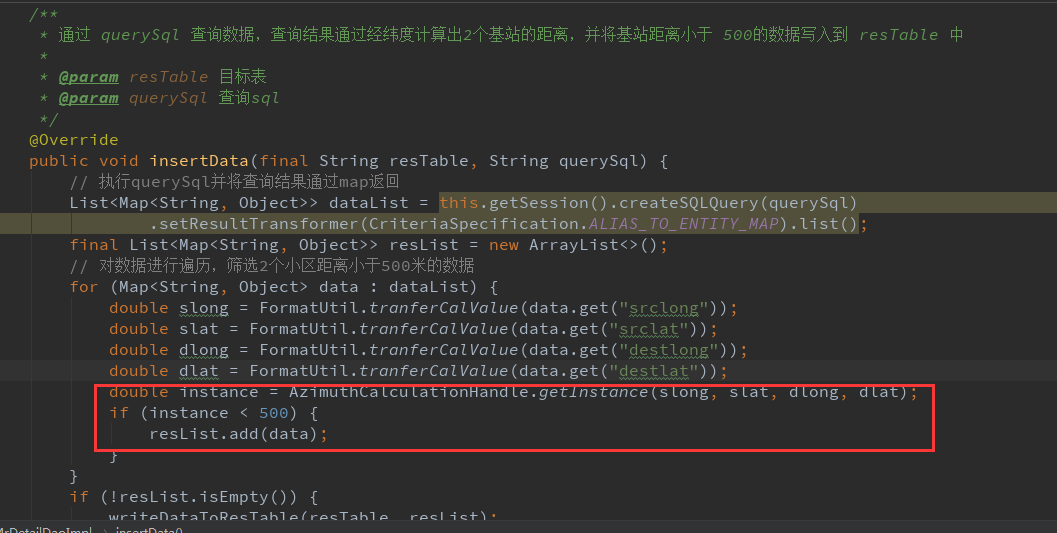
# 在系统结构说明里面，是有2个数据库的，一个读一个写，但是目前只看到了一个

目前来说这个没有这个东西，只有一个库。

# 共站小区问题

在提供的文档里面说明共站小区筛选条件是方位角小于10度，距离小于50米，但是在代码里面写的距离是500米





# 指令下发调试

**数据查询要求，只要这个sql可以查询到数据就可以生成休眠指令**

SELECT

b1.\*,'l2l' bus\_type, CASE

WHEN a1.sps = '未启动'

OR a1.sps IS NULL THEN

0

ELSE

1

END sps,l.ip,ca.cagroupid,ca.preferredpcellpriority

FROM

mes\_l2l\_sleep\_mr b

LEFT JOIN cm\_lte\_dsp\_cell\_hw a1 ON a1.enodebid = b1.src\_enodebid

and a1.localcellid=b1.src\_localcellid

join param\_lte\_hw\_login l on l.nename=b1.enodebname AND l.ommname=a1.omc

LEFT JOIN cm\_lte\_lst\_cagroupcell\_hw ca on b1.src\_enodebid=ca.enodebid and b1.src\_localcellid=ca.localcellid

WHERE

b1.src\_vender = '华为'

AND NOT EXISTS (

SELECT

1

FROM

mes\_l2l\_currentsleep cs

WHERE

cs.src\_enodebid = b1.src\_enodebid

AND cs.src\_localcellid = b1.src\_localcellid

)

数据查询业务概述

通过mes\_l2l\_sleep\_mr的src\_enodebid, src\_localcellid, enodebname去关联cm\_lte\_dsp\_cell\_hw，param\_lte\_hw\_login，cm\_lte\_lst\_cagroupcell\_hw三个cm的基础工参信息表，并且要求mes\_l2l\_currentsleep表中不存在数据时才会返回结果，否则返回的结果集为空。

问题描述：

SELECT \* FROM param\_lte\_hw\_login a , mes\_l2l\_sleep\_mr b WHERE a.`nename` = b.`enodebname`;

SELECT \* FROM param\_lte\_hw\_login a, cm\_lte\_dsp\_cell\_hw b WHERE a.`ommname` = b.`omc`;

# 这2个表关联不上数据 ， cm\_lte\_dsp\_cell\_hw 表中的 enodebid 都是空的

SELECT \* FROM mes\_l2l\_sleep\_mr a ,cm\_lte\_dsp\_cell\_hw b

WHERE a.`src\_enodebid` = b.`enodebid`

AND a.`src\_localcellid` = b.`localcellid`;

# 这个也关联不上数据, cm\_lte\_lst\_cagroupcell\_hw 表里的 enodebid 都是 0 所以无法关联

SELECT \* FROM cm\_lte\_lst\_cagroupcell\_hw a, mes\_l2l\_sleep\_mr b WHERE a.`enodebid` = b.`src\_enodebid` AND a.`localcellid` = b.`src\_localcellid`;

问题:

1. mes\_l2l\_sleep\_mr 生产过程排查，看下src\_enodebid，src\_localcellid 这些数据是从哪里来的
2. cm\_lte\_dsp\_cell\_hw的enodebid为什么是空的
3. cm\_lte\_lst\_cagroupcell\_hw的enodebid为什么是空的
4. cm\_lte\_dsp\_cell\_hw, cm\_lte\_lst\_cagroupcell\_hw 这2个表的数据是否有问题?区别是什么？

# 常用命令说明

查询小区的静态基础信息，小区名，小区号，频点等基础信息，cm

REG NE:IP=100.92.19.79;#LST CELL:LOCALCELLID=12;#UNREG NE:IP=100.92.19.79;

关闭小区

REG NE:IP=100.92.19.79;#DEA CELL:LOCALCELLID=12;#UNREG NE:IP=100.92.19.79;

查询小区的动态参数信息，逻辑小区的状态，运行状态信息

REG NE:IP=[ip];#DSP CELL:;#UNREG NE:IP=[ip];#;

查询参考信息功率，

REG NE:IP=[ip];#DSP BRDMFRINFO:CN=[cn],SRN=[srn],SN=[sn];#;

查询RRU的配置信息

REG NE:IP=[ip];#LST RRU:;#UNREG NE:IP=[ip];#;

查询小区的静态参数如：小区名、小区号、小区PCI等信息小区的基础公参即CM

REG NE:IP=[ip];#LST CELL:;#UNREG NE:IP=[ip];#;

查询小区是否为CA集小区即载波聚合小区

REG NE:IP=[ip];#LST CAGROUPCELL:;#UNREG NE:IP=[ip];#;

查询小区的告警日志即历史告警

REG NE:IP=[ip];#LST ALMAF:;#UNREG NE:IP=[ip];#;

查询本地小区状态

REG NE:IP=[ip];#DSP LOCELL:;#UNREG NE:IP=[ip];#;

查询小区物理单板拓扑关系

REG NE:IP=[ip];#DSP CELLPHYTOPO:;#UNREG NE:IP=[ip];#;

查询小区运行状态

REG NE:IP=[ip];#DSP CELL:;#UNREG NE:IP=[ip];#;

查询小区单板制造信息

REG NE:IP=[ip];#DSP BRDMFRINFO:CN=[cn],SRN=[srn],SN=[sn];#;

删除CAGR小区

#RMV CAGROUPCELL:CAGROUPID=[cagroupid],LOCALCELLID=[src\_localcellid],ENODEBID=[src\_enodebid];

**休眠小区唤醒**

休眠小区唤醒，先开载波聚合，添加辅载波的指令，再唤醒小区

REG NE:IP=100.92.19.79;#ADD CAGROUPCELL:CAGROUPID=1,LOCALCELLID=12,ENODEBID=606440,PREFERREDPCELLPRIORITY=0;#ACT CELL:LOCALCELLID=12;#UNREG NE:IP=100.92.19.79;

最后状态查询确认小区已经关闭了

REG NE:IP=100.92.19.79;#LST CELL:LOCALCELLID=12;#UNREG NE:IP=100.92.19.79;

# 下面这些可能是最全面的说明了

1. MRO 采集 (mro\_collect\_lte\_hw.py)

依赖：mr文件

生成：pm\_lte\_mro\_hw

1. MRO文件解析(test\_mr\_parser\_sm.py)

依赖： mr文件

生成：rst\_l2l\_hw\_hz\_sm

1. 重叠覆盖计算(mro\_over\_cal.py)

依赖：rst\_l2l\_hw\_hz\_sm (MRO文件解析步骤)

mes\_base\_info\_lte (需要提前导入)

生成：rst\_l2l\_mro\_hw

1. CM采集🡪 基础信息采集 ftp，telnet

依赖：cm文件，telnet结果

生成：param\_lte\_login && param\_lte\_login\_bak

Cm\_lte\_\*\_hw

1. CM采集🡪 improveMrData

依赖：

cm\_lte\_eutranrelationtdd\_hw

cm\_lte\_enbfunction\_hw

cm\_lte\_eutrancelltdd\_hw

rst\_l2l\_mro\_hw （重叠覆盖计算）

mes\_base\_info\_lte （基础导入）

生成： rst\_l2l\_mr\_detail

1. PM采集🡪 基础信息采集 (pm\_collect.py)

依赖： pm文件

生成： pm\_lte\_eutrancelltdd\_hw

1. 大数据预测(bigdata\_forecast.py)

依赖: pm\_lte\_eutrancelltdd\_hw

生成：res\_bigdata\_ pm\_lte\_eutrancelltdd\_hw

1. 历史KPI计算

依赖：res\_bigdata\_ pm\_lte\_eutrancelltdd\_hw（大数据预测）

Cm\_lte\_entrancelltdd\_hw

Cm\_lte\_enbfuncation\_hw

生成：mes\_lte\_kpi

1. 刷新节能小区

依赖：rst\_l2l\_mr\_detail (步骤五，cm采集时生成)

Cm\_lte\_entrancelltdd\_hw

Cm\_lte\_enbfuncation\_hw

Cm\_lte\_entranrelation\_hw

Rst\_l2l\_mro\_hw (步骤3 重叠覆盖计算)

生成：rst\_l2l\_energyref\_mr

1. 休眠小区筛选

依赖: rst\_l2l\_energyref\_mr ， mes\_lte\_kpi

生成：mes\_l2l\_sleep\_mr

1. 休眠指令生成

依赖：cm\_lte\_dsp\_cell\_hw

param\_lte\_hw\_login

cm\_lte\_lst\_cagroupcell\_hw

mes\_l2l\_sleep\_mr

生成： mes\_adjust\_command

1. 休眠指令下发

依赖： mes\_adjust\_command

只要mes\_adjust\_command里面有类型为sleep的数据，且开关打开就可以生成

1. 小区监控-唤醒指令生成

依赖：

mes\_l2l\_currentsleep

mes\_lte\_nonsleep

mes\_l2l\_many\_currentsleep

生成：

mes\_adjust\_command

1. 唤醒指令下发

依赖： mes\_adjust\_command

只要mes\_adjust\_command里面有类型为notify的数据，且开关打开就可以生成