目录

[GreenPlum测试报告 2](#_Toc293850305)

[1、配置信息 2](#_Toc293850306)

[1.1硬件配置 2](#_Toc293850307)

[1.2测试环境 2](#_Toc293850308)

[2、测试案例综述 3](#_Toc293850309)

[2.1案例设计 3](#_Toc293850310)

[2.2测试数据 3](#_Toc293850311)

[3、测试结果 18](#_Toc293850312)

[3.1串行测试 18](#_Toc293850313)

[3.1.1 行存储模式 18](#_Toc293850314)

[3.1.1.1 数据入库测试 18](#_Toc293850315)

[3.1.1.2 系统性能指标 20](#_Toc293850316)

[3.1.1.3 查询性能指标 41](#_Toc293850317)

[3.1.2 列存储模式 47](#_Toc293850318)

[3.1.2.1数据入库测试 47](#_Toc293850319)

[3.1.2.2 系统性能指标 48](#_Toc293850320)

[3.1.2.3 查询性能指标 62](#_Toc293850321)

[3.2并行测试（没有数据入库，只有并行查询） 70](#_Toc293850322)

[3.2.1 行存储模式 70](#_Toc293850323)

[3.2.1.1 模拟并发5用户 70](#_Toc293850324)

[3.2.1.2 模拟并发10用户 84](#_Toc293850325)

[3.2.2 列存储模式 94](#_Toc293850326)

[3.2.2.1 模拟并发5用户 94](#_Toc293850327)

[3.2.2.2 模拟并发10用户---未测试 98](#_Toc293850328)

[3.3并行测试（数据入库的同时，并行查询） 98](#_Toc293850329)

[3.3.1 行存储模式 98](#_Toc293850330)

[3.3.1.1数据入库测试 98](#_Toc293850331)

[3.3.1.2 模拟并发5用户---未测试 99](#_Toc293850332)

[3.3.1.3 模拟并发10用户 99](#_Toc293850333)

[3.3.2 列存储模式 110](#_Toc293850334)

[3.3.2.1数据入库测试 110](#_Toc293850335)

[3.3.2.2 模拟并发5用户 110](#_Toc293850336)

[3.3.2.3 模拟并发10用户---未测试 117](#_Toc293850337)

[3.4分区轮换测试 117](#_Toc293850338)

[3.5 segment instance 无mirror且宕instance测试 117](#_Toc293850339)

[3.6主备master切换测试 118](#_Toc293850340)

[3.7 segment instance有mirror宕instance测试 118](#_Toc293850341)

[3.7.1 测试前环境 118](#_Toc293850342)

[3.7.2 测试1 119](#_Toc293850343)

[3.7.3 测试2 119](#_Toc293850344)

[3.7.4 测试3 119](#_Toc293850345)

[4、结论 119](#_Toc293850346)

# GreenPlum测试报告

## 1、配置信息

### 1.1硬件配置

**数据库Segment host ：4 台**

型号：华赛T3500

内存：8\*4G = 32 GB

CPU：2\*4 core，Xeon(R) CPU E5430 2.66GHz

网卡：4个千兆网卡

硬盘：22\*1T SATA / 7.2K转

Disk Array：2 \* 1T SATA RAID 1 操作系统

5\*1T SATA RAID 1+0 data (共4组RAID)

**数据库master host ：1 台**

型号：华赛T3500

内存：8\*4G = 32G

CPU：2\*4 core，Xeon(R) CPU E5430 2.66GHz

网卡：5个千兆网卡

硬盘：4\*1T SATA / 7.2K转

Disk Array：2 \* 1T SATA RAID 1 操作系统

2\*1T SATA RAID 1+0 data

**千兆网络交换机一台：**

4台segment host 使用4块千兆网口和1台master host做内联；

Master host 使用第五个千兆网口做public ip

**操作系统：Suse 11 SP1**

### 1.2测试环境

1. 总共32个PostGreSql实例，每台支持8个，所有表的数据几乎被等分在32个数据库实例中，无segment instance的镜像。

## 2、测试案例综述

### 2.1案例设计

1. 此次测试主要可以从两个角度来分析：串行和并行以及在此基础上的数据入库和sql查询；
2. 测试过程中记录性能指标（cpu/memory/io/network）；
3. 分区轮换测试；
4. 在segment instance无mirror的情况下，宕掉一个instance能否继续使用GP；
5. 主备mater部署及切换；
6. 在配置了segment instance级别的mirror之后，宕一个instance，测试GP能否继续使用，且要求不影响数据的准确性；在此基础上，继续测试宕掉整台设备看是否GP还能继续使用，且不影响数据的准确性。

### 2.2测试数据

共5张表（ne\_cell\_c / ne\_cell\_c\_bj / ne\_bts\_c / mod\_cdr\_hw / mod\_do\_hw）

其中：ne\_cell\_c 共65558条记录 21.3M；

ne\_cell\_c\_bj 共10199条记录 3.27M；

Ne\_bts\_c 共18064条记录 4.08M；

Mod\_cdr\_hw（700列） 广州（city\_id=20）一天的数据，28,018,657条记录，44.8GByte；

Mod\_do\_hw（459列） 北京（city\_id=10）一天的数据，73,027,740条记录，97GByte；

9个查询sql（在测试时根据情况又添加了一些sql，具体请看测试结果中的sql文本）：

-------------------------------------------------------------------------------------------------------------------

select \* from mod\_cdr\_hw

where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')

and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')

and access\_cell=1039;

select OMCID,

CITY\_ID,

NE\_SYS\_ID,

RELATED\_CELL,

RELATED\_BTS,

RELATED\_BSC,

RELATED\_OMC,

HANDLE\_INDEX,

CALL\_REF\_NUM,

TIME\_STAMP,

CHR\_VER,

CHR\_TYPE,

RELEASE\_CFC,

RELEASE\_CAUSE,

IMSI,

ESN,

CALL\_STATE,

DIALED\_DIGITS,

BAND\_CLASS,

PG\_DURATION,

ACCESS\_TIME,

ACCESS\_TIME\_MSEC,

SRV\_CONN\_START\_TIME,

ASSIGN\_DURATION,

PREAMBLE\_DURATION,

NEGO\_DURATION,

CALL\_DURATION,

CALL\_OR\_CALLED\_OR\_HHO,

SERVICE\_OPTION,

FINAL\_SERVICE\_OPTION,

MOBILE\_PROTOCOL\_REV,

MOBILE\_PROTOCOL\_REV\_INUSE,

SCM,

QOS\_INDEX,

USER\_QOS\_GRADE,

REV\_FCH\_GATING,

PCF\_IP,

IWF\_CPU\_ID,

MD\_PATH\_CH\_NO,

PSTN\_PATH\_CH\_NO,

PCM,

A\_E1\_SLOT,

SERVICE\_ONE\_WAY\_DELAY,

AUTH\_MODE,

SLOT\_CYCLE\_INDEX,

SIGNAL\_TYPE,

ALERT\_PITCH,

SIGNAL,

PRIVACY\_CURR\_TYPE,

MOB\_TERM,

DCCH\_SUPPORTED,

FCH\_SUPPORTED,

OTD\_SUPPORTED,

QPCH\_SUPPORTED,

GEO\_LOCATION\_TYPE,

FCH\_FRM\_SIZE\_CAP,

DCCH\_FRM\_SIZE\_CAP,

FCH\_FWD\_RC\_CAP,

DCCH\_FWD\_RC\_CAP,

FCH\_REV\_RC\_CAP,

DCCH\_REV\_RC\_CAP,

CALL\_NOTPROC\_COUNT,

ACCESS\_CHNANNEL,

ASSIGN\_CHANNEL,

ACCPROBE\_HO\_COUNT,

ADDT\_PILOT\_COUNT,

ACC\_HO\_FLAG,

ACCESS\_CELL,

ACCESS\_SECTOR,

ACCESS\_STRENGTH,

ADDTPN1\_CELL\_ID,

ADDTPN1\_SECTOR\_ID,

ADDTPN1\_PN\_OFFSET,

ADDTPN1\_STRENGTH,

ADDTPN2\_CELL\_ID,

ADDTPN2\_SECTOR\_ID,

ADDTPN2\_PN\_OFFSET,

ADDTPN2\_STRENGTH,

PPU\_CPU\_ID,

MS\_PARA\_METER\_TYPE,

MS\_PARA\_METER1\_VALUE,

MS\_PARA\_METER2\_VALUE,

MS\_PARA\_METER3\_VALUE,

MS\_PARA\_METER4\_VALUE,

MS\_PARA\_METER5\_VALUE,

MS\_PARA\_METER6\_VALUE,

MS\_PARA\_METER7\_VALUE,

MS\_PARA\_METER8\_VALUE,

MS\_PARA\_METER9\_VALUE,

MS\_PARA\_METER10\_VALUE,

MS\_PARA\_METER11\_VALUE,

MS\_PARA\_METER12\_VALUE,

MS\_PARA\_METER13\_VALUE,

MS\_PARA\_METER14\_VALUE,

MS\_PARA\_METER15\_VALUE,

MS\_PARA\_METER16\_VALUE,

MS\_PARA\_METER17\_VALUE,

MS\_PARA\_METER18\_VALUE,

CCM\_RESERVED

from mod\_cdr\_hw

where access\_time >

To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')

and access\_time <

To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')

and access\_cell = 1039;

--------------------------------------------------------------------------------------------------------------------------

select \* from mod\_cdr\_hw where access\_cell=1039;

--------------------------------------------------------------------------------------------------------------------------

1. select \* from mod\_cdr\_hw

where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')

and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')

and release\_cfc='4';

---------------------------------------------------------------------------------------------------------------------------

1. select \* from mod\_cdr\_hw

where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')

and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')

and imsi='460036271583050';

----------------------------------------------------------------------------------------------------------------------------

1. select \* from mod\_cdr\_hw

where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')

and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')

and imsi='460030912862245';

-----------------------------------------------------------------------------------------------------------------------------

1. select \* from mod\_cdr\_hw

where access\_time > To\_Date('2010-12-28 9:00:00', 'yyyy-mm-dd hh24:mi:ss')

and access\_time < To\_Date('2010-12-28 10:00:00', 'yyyy-mm-dd hh24:mi:ss');

------------------------------------------------------------------------------------------------------------------------------

1. select b.\* from mod\_cdr\_hw b, ne\_cell\_c c

where access\_cell=1038 and c.ne\_sys\_id = b.related\_cell;

-------------------------------------------------------------------------------------------------------------------------------

1. select \* from mod\_cdr\_hw b, ne\_cell\_c c

where b.related\_bts =1080810030078000 and c.ne\_sys\_id = b.related\_cell;

--------------------------------------------------------------------------------------------------------------------------------

13) select omcid , city\_id , ne\_sys\_id , related\_cell , related\_bts , related\_bsc , related\_omc , handle\_index , call\_ref\_num , time\_stamp , chr\_ver , chr\_type , release\_cfc , release\_cause , imsi , esn , call\_state , dialed\_digits , band\_class , pg\_duration , access\_time , access\_time\_msec , srv\_conn\_start\_time , assign\_duration , preamble\_duration , nego\_duration , call\_duration , call\_or\_called\_or\_hho , service\_option , final\_service\_option , mobile\_protocol\_rev , mobile\_protocol\_rev\_inuse , scm , qos\_index , user\_qos\_grade , rev\_fch\_gating , pcf\_ip , iwf\_cpu\_id , md\_path\_ch\_no , pstn\_path\_ch\_no , pcm , a\_e1\_slot , service\_one\_way\_delay , auth\_mode , slot\_cycle\_index , signal\_type , alert\_pitch , signal , privacy\_curr\_type , mob\_term , dcch\_supported , fch\_supported , otd\_supported , qpch\_supported , geo\_location\_type , fch\_frm\_size\_cap , dcch\_frm\_size\_cap , fch\_fwd\_rc\_cap , dcch\_fwd\_rc\_cap , fch\_rev\_rc\_cap , dcch\_rev\_rc\_cap , call\_notproc\_count , access\_chnannel , assign\_channel , accprobe\_ho\_count , addt\_pilot\_count , acc\_ho\_flag , access\_cell , access\_sector , access\_strength , addtpn1\_cell\_id , addtpn1\_sector\_id , addtpn1\_pn\_offset , addtpn1\_strength , addtpn2\_cell\_id , addtpn2\_sector\_id , addtpn2\_pn\_offset , addtpn2\_strength , ppu\_cpu\_id , ms\_para\_meter\_type , ms\_para\_meter1\_value , ms\_para\_meter2\_value , ms\_para\_meter3\_value , ms\_para\_meter4\_value , ms\_para\_meter5\_value , ms\_para\_meter6\_value , ms\_para\_meter7\_value , ms\_para\_meter8\_value , ms\_para\_meter9\_value , ms\_para\_meter10\_value , ms\_para\_meter11\_value , ms\_para\_meter12\_value , ms\_para\_meter13\_value , ms\_para\_meter14\_value , ms\_para\_meter15\_value , ms\_para\_meter16\_value , ms\_para\_meter17\_value , ms\_para\_meter18\_value , ccm\_reserved , access\_carr\_fwd\_load from mod\_cdr\_hw where imsi='460036271583050';

-----------------------------------------------------------------------------------------------------------------------------

14）select omcid , city\_id , ne\_sys\_id , related\_cell , related\_bts , related\_bsc , related\_omc , handle\_index , call\_ref\_num , time\_stamp , chr\_ver , chr\_type , release\_cfc , release\_cause , imsi , esn , call\_state , dialed\_digits , band\_class , pg\_duration , access\_time , access\_time\_msec , srv\_conn\_start\_time , assign\_duration , preamble\_duration , nego\_duration , call\_duration , call\_or\_called\_or\_hho , service\_option , final\_service\_option , mobile\_protocol\_rev , mobile\_protocol\_rev\_inuse , scm , qos\_index , user\_qos\_grade , rev\_fch\_gating , pcf\_ip , iwf\_cpu\_id , md\_path\_ch\_no , pstn\_path\_ch\_no , pcm , a\_e1\_slot , service\_one\_way\_delay , auth\_mode , slot\_cycle\_index , signal\_type , alert\_pitch , signal , privacy\_curr\_type , mob\_term , dcch\_supported , fch\_supported , otd\_supported , qpch\_supported , geo\_location\_type , fch\_frm\_size\_cap , dcch\_frm\_size\_cap , fch\_fwd\_rc\_cap , dcch\_fwd\_rc\_cap , fch\_rev\_rc\_cap , dcch\_rev\_rc\_cap , call\_notproc\_count , access\_chnannel , assign\_channel , accprobe\_ho\_count , addt\_pilot\_count , acc\_ho\_flag , access\_cell , access\_sector , access\_strength , addtpn1\_cell\_id , addtpn1\_sector\_id , addtpn1\_pn\_offset , addtpn1\_strength , addtpn2\_cell\_id , addtpn2\_sector\_id , addtpn2\_pn\_offset , addtpn2\_strength , ppu\_cpu\_id , ms\_para\_meter\_type , ms\_para\_meter1\_value , ms\_para\_meter2\_value , ms\_para\_meter3\_value , ms\_para\_meter4\_value , ms\_para\_meter5\_value , ms\_para\_meter6\_value , ms\_para\_meter7\_value , ms\_para\_meter8\_value , ms\_para\_meter9\_value , ms\_para\_meter10\_value , ms\_para\_meter11\_value , ms\_para\_meter12\_value , ms\_para\_meter13\_value , ms\_para\_meter14\_value , ms\_para\_meter15\_value , ms\_para\_meter16\_value , ms\_para\_meter17\_value , ms\_para\_meter18\_value , ccm\_reserved , access\_carr\_fwd\_load ,access\_carr\_fwd\_basic\_thresh , access\_carr\_fwd\_badleg\_ratio , access\_carr\_channle\_count , access\_carr\_rvs\_equ\_user , access\_carr\_rvs\_rssi , access\_carr\_rvs\_badleg\_ratio , access\_carr\_access\_load , access\_carr\_access\_fail\_rate , init\_rf\_conn\_count , init\_rf\_conn1\_cell , init\_rf\_conn1\_sector , init\_rf\_conn2\_cell , init\_rf\_conn2\_sector , init\_rf\_conn3\_cell , init\_rf\_conn3\_sector , last\_rf\_conn\_time\_vos , last\_rf\_conn\_count , last\_rf\_conn1\_cell , last\_rf\_conn1\_sector , last\_rf\_conn1\_str , last\_rf\_conn1\_onewaydelay , last\_rf\_conn2\_cell , last\_rf\_conn2\_sector , last\_rf\_conn2\_str , last\_rf\_conn2\_onewaydelay , last\_rf\_conn3\_cell , last\_rf\_conn3\_sector , last\_rf\_conn3\_str , last\_rf\_conn3\_onewaydelay , fwd\_fer0 , fwd\_fer1 , fwd\_fer2 , fwd\_fer3 , fwd\_fer4 , fwd\_fer5 , fwd\_fer6 , fwd\_fer7 , fwd\_fer8 , fwd\_fer9 , rev\_totframe0 , rev\_totframe1 , rev\_totframe2 , rev\_totframe3 , rev\_totframe4 , rev\_totframe5 , rev\_totframe6 , rev\_totframe7 , rev\_totframe8 , rev\_totframe9 , rev\_errframe0 , rev\_errframe1 , rev\_errframe2 , rev\_errframe3 , rev\_errframe4 , rev\_errframe5 , rev\_errframe6 , rev\_errframe7 , rev\_errframe8 , rev\_errframe9 , rev\_leg1\_totframe0 , rev\_leg1\_totframe1 , rev\_leg1\_totframe2 , rev\_leg1\_totframe3 , rev\_leg1\_totframe4 , rev\_leg1\_totframe5 , rev\_leg1\_totframe6 , rev\_leg1\_totframe7 , rev\_leg1\_totframe8 , rev\_leg1\_totframe9 , rev\_leg1\_errframe0 , rev\_leg1\_errframe1 , rev\_leg1\_errframe2 , rev\_leg1\_errframe3 , rev\_leg1\_errframe4 , rev\_leg1\_errframe5 , rev\_leg1\_errframe6 , rev\_leg1\_errframe7 , rev\_leg1\_errframe8 , rev\_leg1\_errframe9 , rev\_leg2\_totframe0 , rev\_leg2\_totframe1 , rev\_leg2\_totframe2 , rev\_leg2\_totframe3 , rev\_leg2\_totframe4 , rev\_leg2\_totframe5 , rev\_leg2\_totframe6 , rev\_leg2\_totframe7 , rev\_leg2\_totframe8 , rev\_leg2\_totframe9 , rev\_leg2\_errframe0 , rev\_leg2\_errframe1 , rev\_leg2\_errframe2 , rev\_leg2\_errframe3 , rev\_leg2\_errframe4 , rev\_leg2\_errframe5 , rev\_leg2\_errframe6 , rev\_leg2\_errframe7 , rev\_leg2\_errframe8 , rev\_leg2\_errframe9 from mod\_cdr\_hw , rev\_leg3\_totframe0 where imsi='460036271583050';

-------------------------------------------------------------------------------------------------------------------------------

15) select omcid , city\_id , ne\_sys\_id , related\_cell , related\_bts , related\_bsc , related\_omc , handle\_index , call\_ref\_num , time\_stamp , chr\_ver , chr\_type , release\_cfc , release\_cause , imsi , esn , call\_state , dialed\_digits , band\_class , pg\_duration , access\_time , access\_time\_msec , srv\_conn\_start\_time , assign\_duration , preamble\_duration , nego\_duration , call\_duration , call\_or\_called\_or\_hho , service\_option , final\_service\_option , mobile\_protocol\_rev , mobile\_protocol\_rev\_inuse , scm , qos\_index , user\_qos\_grade , rev\_fch\_gating , pcf\_ip , iwf\_cpu\_id , md\_path\_ch\_no , pstn\_path\_ch\_no , pcm , a\_e1\_slot , service\_one\_way\_delay , auth\_mode , slot\_cycle\_index , signal\_type , alert\_pitch , signal , privacy\_curr\_type , mob\_term , dcch\_supported , fch\_supported , otd\_supported , qpch\_supported , geo\_location\_type , fch\_frm\_size\_cap , dcch\_frm\_size\_cap , fch\_fwd\_rc\_cap , dcch\_fwd\_rc\_cap , fch\_rev\_rc\_cap , dcch\_rev\_rc\_cap , call\_notproc\_count , access\_chnannel , assign\_channel , accprobe\_ho\_count , addt\_pilot\_count , acc\_ho\_flag , access\_cell , access\_sector , access\_strength , addtpn1\_cell\_id , addtpn1\_sector\_id , addtpn1\_pn\_offset , addtpn1\_strength , addtpn2\_cell\_id , addtpn2\_sector\_id , addtpn2\_pn\_offset , addtpn2\_strength , ppu\_cpu\_id , ms\_para\_meter\_type , ms\_para\_meter1\_value , ms\_para\_meter2\_value , ms\_para\_meter3\_value , ms\_para\_meter4\_value , ms\_para\_meter5\_value , ms\_para\_meter6\_value , ms\_para\_meter7\_value , ms\_para\_meter8\_value , ms\_para\_meter9\_value , ms\_para\_meter10\_value , ms\_para\_meter11\_value , ms\_para\_meter12\_value , ms\_para\_meter13\_value , ms\_para\_meter14\_value , ms\_para\_meter15\_value , ms\_para\_meter16\_value , ms\_para\_meter17\_value , ms\_para\_meter18\_value , ccm\_reserved , access\_carr\_fwd\_load ,access\_carr\_fwd\_basic\_thresh , access\_carr\_fwd\_badleg\_ratio , access\_carr\_channle\_count , access\_carr\_rvs\_equ\_user , access\_carr\_rvs\_rssi , access\_carr\_rvs\_badleg\_ratio , access\_carr\_access\_load , access\_carr\_access\_fail\_rate , init\_rf\_conn\_count , init\_rf\_conn1\_cell , init\_rf\_conn1\_sector , init\_rf\_conn2\_cell , init\_rf\_conn2\_sector , init\_rf\_conn3\_cell , init\_rf\_conn3\_sector , last\_rf\_conn\_time\_vos , last\_rf\_conn\_count , last\_rf\_conn1\_cell , last\_rf\_conn1\_sector , last\_rf\_conn1\_str , last\_rf\_conn1\_onewaydelay , last\_rf\_conn2\_cell , last\_rf\_conn2\_sector , last\_rf\_conn2\_str , last\_rf\_conn2\_onewaydelay , last\_rf\_conn3\_cell , last\_rf\_conn3\_sector , last\_rf\_conn3\_str , last\_rf\_conn3\_onewaydelay , fwd\_fer0 , fwd\_fer1 , fwd\_fer2 , fwd\_fer3 , fwd\_fer4 , fwd\_fer5 , fwd\_fer6 , fwd\_fer7 , fwd\_fer8 , fwd\_fer9 , rev\_totframe0 , rev\_totframe1 , rev\_totframe2 , rev\_totframe3 , rev\_totframe4 , rev\_totframe5 , rev\_totframe6 , rev\_totframe7 , rev\_totframe8 , rev\_totframe9 , rev\_errframe0 , rev\_errframe1 , rev\_errframe2 , rev\_errframe3 , rev\_errframe4 , rev\_errframe5 , rev\_errframe6 , rev\_errframe7 , rev\_errframe8 , rev\_errframe9 , rev\_leg1\_totframe0 , rev\_leg1\_totframe1 , rev\_leg1\_totframe2 , rev\_leg1\_totframe3 , rev\_leg1\_totframe4 , rev\_leg1\_totframe5 , rev\_leg1\_totframe6 , rev\_leg1\_totframe7 , rev\_leg1\_totframe8 , rev\_leg1\_totframe9 , rev\_leg1\_errframe0 , rev\_leg1\_errframe1 , rev\_leg1\_errframe2 , rev\_leg1\_errframe3 , rev\_leg1\_errframe4 , rev\_leg1\_errframe5 , rev\_leg1\_errframe6 , rev\_leg1\_errframe7 , rev\_leg1\_errframe8 , rev\_leg1\_errframe9 , rev\_leg2\_totframe0 , rev\_leg2\_totframe1 , rev\_leg2\_totframe2 , rev\_leg2\_totframe3 , rev\_leg2\_totframe4 , rev\_leg2\_totframe5 , rev\_leg2\_totframe6 , rev\_leg2\_totframe7 , rev\_leg2\_totframe8 , rev\_leg2\_totframe9 , rev\_leg2\_errframe0 , rev\_leg2\_errframe1 , rev\_leg2\_errframe2 , rev\_leg2\_errframe3 , rev\_leg2\_errframe4 , rev\_leg2\_errframe5 , rev\_leg2\_errframe6 , rev\_leg2\_errframe7 , rev\_leg2\_errframe8 , rev\_leg2\_errframe9,rev\_leg3\_totframe0 ,rev\_leg3\_totframe1 , rev\_leg3\_totframe2 , rev\_leg3\_totframe3 , rev\_leg3\_totframe4 , rev\_leg3\_totframe5 , rev\_leg3\_totframe6 , rev\_leg3\_totframe7 , rev\_leg3\_totframe8 , rev\_leg3\_totframe9 , rev\_leg3\_errframe0 , rev\_leg3\_errframe1 , rev\_leg3\_errframe2 , rev\_leg3\_errframe3 , rev\_leg3\_errframe4 , rev\_leg3\_errframe5 , rev\_leg3\_errframe6 , rev\_leg3\_errframe7 , rev\_leg3\_errframe8 , rev\_leg3\_errframe9 , rev\_ebnt0 , rev\_ebnt1 , rev\_ebnt2 , rev\_ebnt3 , rev\_ebnt4 , rev\_ebnt5 , rev\_ebnt6 , rev\_ebnt7 , rev\_ebnt8 , rev\_ebnt9 , rev\_leg1\_ebnt0 , rev\_leg1\_ebnt1 , rev\_leg1\_ebnt2 , rev\_leg1\_ebnt3 , rev\_leg1\_ebnt4 , rev\_leg1\_ebnt5 , rev\_leg1\_ebnt6 , rev\_leg1\_ebnt7 , rev\_leg1\_ebnt8 , rev\_leg1\_ebnt9 , rev\_leg2\_ebnt0 , rev\_leg2\_ebnt1 , rev\_leg2\_ebnt2 , rev\_leg2\_ebnt3 , rev\_leg2\_ebnt4 , rev\_leg2\_ebnt5 , rev\_leg2\_ebnt6 , rev\_leg2\_ebnt7 , rev\_leg2\_ebnt8 , rev\_leg2\_ebnt9 , rev\_leg3\_ebnt0 , rev\_leg3\_ebnt1 , rev\_leg3\_ebnt2 , rev\_leg3\_ebnt3 , rev\_leg3\_ebnt4 , rev\_leg3\_ebnt5 , rev\_leg3\_ebnt6 , rev\_leg3\_ebnt7 , rev\_leg3\_ebnt8 , rev\_leg3\_ebnt9 , fwd\_tch\_fer\_ratio\_0\_to\_3 , fwd\_tch\_fer\_ratio\_3\_to\_7 , fwd\_tch\_fer\_ratio\_7\_to\_15 , fwd\_tch\_fer\_ratio\_more\_than\_15 , fwd\_tch\_expect\_fer , fwd\_totframe , fwd\_errframe , fwd\_sch\_totframe , fwd\_sch\_errframe , fwd\_quality , rvs\_quality , last\_rf\_higa1\_count , last\_rf\_higa1\_temp , last\_rf\_higa1\_intervals , last\_rf\_higa1\_begin , last\_rf\_higa1\_end , last\_rf\_higa2\_count , last\_rf\_higa2\_temp , last\_rf\_higa2\_intervals , last\_rf\_higa2\_begin , last\_rf\_higa2\_end , last\_rf\_higa3\_count , last\_rf\_higa3\_temp , last\_rf\_higa3\_intervals , last\_rf\_higa3\_begin , last\_rf\_higa3\_end , last\_rf\_setp\_count , last\_rf\_setp\_temp , last\_rf\_setp\_intervals , last\_rf\_setp\_begin , last\_rf\_setp\_end , first\_psmm\_time , first\_psmm\_act\_str , first\_psmm\_pn\_count , first\_psmm\_cell1 , first\_psmm\_sector1 , first\_psmm\_pn1\_strength , first\_psmm\_pn1\_onewaydelay , first\_psmm\_maho\_pn1\_keep , first\_psmm\_pn1\_pilotset,first\_psmm\_cell2 from mod\_cdr\_hw where imsi='460036271583050';

------------------------------------------------------------------------------------------------------------------------------

16) select omcid , city\_id , ne\_sys\_id , related\_cell , related\_bts , related\_bsc , related\_omc , handle\_index , call\_ref\_num , time\_stamp , chr\_ver , chr\_type , release\_cfc , release\_cause , imsi , esn , call\_state , dialed\_digits , band\_class , pg\_duration , access\_time , access\_time\_msec , srv\_conn\_start\_time , assign\_duration , preamble\_duration , nego\_duration , call\_duration , call\_or\_called\_or\_hho , service\_option , final\_service\_option , mobile\_protocol\_rev , mobile\_protocol\_rev\_inuse , scm , qos\_index , user\_qos\_grade , rev\_fch\_gating , pcf\_ip , iwf\_cpu\_id , md\_path\_ch\_no , pstn\_path\_ch\_no , pcm , a\_e1\_slot , service\_one\_way\_delay , auth\_mode , slot\_cycle\_index , signal\_type , alert\_pitch , signal , privacy\_curr\_type , mob\_term , dcch\_supported , fch\_supported , otd\_supported , qpch\_supported , geo\_location\_type , fch\_frm\_size\_cap , dcch\_frm\_size\_cap , fch\_fwd\_rc\_cap , dcch\_fwd\_rc\_cap , fch\_rev\_rc\_cap , dcch\_rev\_rc\_cap , call\_notproc\_count , access\_chnannel , assign\_channel , accprobe\_ho\_count , addt\_pilot\_count , acc\_ho\_flag , access\_cell , access\_sector , access\_strength , addtpn1\_cell\_id , addtpn1\_sector\_id , addtpn1\_pn\_offset , addtpn1\_strength , addtpn2\_cell\_id , addtpn2\_sector\_id , addtpn2\_pn\_offset , addtpn2\_strength , ppu\_cpu\_id , ms\_para\_meter\_type , ms\_para\_meter1\_value , ms\_para\_meter2\_value , ms\_para\_meter3\_value , ms\_para\_meter4\_value , ms\_para\_meter5\_value , ms\_para\_meter6\_value , ms\_para\_meter7\_value , ms\_para\_meter8\_value , ms\_para\_meter9\_value , ms\_para\_meter10\_value , ms\_para\_meter11\_value , ms\_para\_meter12\_value , ms\_para\_meter13\_value , ms\_para\_meter14\_value , ms\_para\_meter15\_value , ms\_para\_meter16\_value , ms\_para\_meter17\_value , ms\_para\_meter18\_value , ccm\_reserved , access\_carr\_fwd\_load ,access\_carr\_fwd\_basic\_thresh , access\_carr\_fwd\_badleg\_ratio , access\_carr\_channle\_count , access\_carr\_rvs\_equ\_user , access\_carr\_rvs\_rssi , access\_carr\_rvs\_badleg\_ratio , access\_carr\_access\_load , access\_carr\_access\_fail\_rate , init\_rf\_conn\_count , init\_rf\_conn1\_cell , init\_rf\_conn1\_sector , init\_rf\_conn2\_cell , init\_rf\_conn2\_sector , init\_rf\_conn3\_cell , init\_rf\_conn3\_sector , last\_rf\_conn\_time\_vos , last\_rf\_conn\_count , last\_rf\_conn1\_cell , last\_rf\_conn1\_sector , last\_rf\_conn1\_str , last\_rf\_conn1\_onewaydelay , last\_rf\_conn2\_cell , last\_rf\_conn2\_sector , last\_rf\_conn2\_str , last\_rf\_conn2\_onewaydelay , last\_rf\_conn3\_cell , last\_rf\_conn3\_sector , last\_rf\_conn3\_str , last\_rf\_conn3\_onewaydelay , fwd\_fer0 , fwd\_fer1 , fwd\_fer2 , fwd\_fer3 , fwd\_fer4 , fwd\_fer5 , fwd\_fer6 , fwd\_fer7 , fwd\_fer8 , fwd\_fer9 , rev\_totframe0 , rev\_totframe1 , rev\_totframe2 , rev\_totframe3 , rev\_totframe4 , rev\_totframe5 , rev\_totframe6 , rev\_totframe7 , rev\_totframe8 , rev\_totframe9 , rev\_errframe0 , rev\_errframe1 , rev\_errframe2 , rev\_errframe3 , rev\_errframe4 , rev\_errframe5 , rev\_errframe6 , rev\_errframe7 , rev\_errframe8 , rev\_errframe9 , rev\_leg1\_totframe0 , rev\_leg1\_totframe1 , rev\_leg1\_totframe2 , rev\_leg1\_totframe3 , rev\_leg1\_totframe4 , rev\_leg1\_totframe5 , rev\_leg1\_totframe6 , rev\_leg1\_totframe7 , rev\_leg1\_totframe8 , rev\_leg1\_totframe9 , rev\_leg1\_errframe0 , rev\_leg1\_errframe1 , rev\_leg1\_errframe2 , rev\_leg1\_errframe3 , rev\_leg1\_errframe4 , rev\_leg1\_errframe5 , rev\_leg1\_errframe6 , rev\_leg1\_errframe7 , rev\_leg1\_errframe8 , rev\_leg1\_errframe9 , rev\_leg2\_totframe0 , rev\_leg2\_totframe1 , rev\_leg2\_totframe2 , rev\_leg2\_totframe3 , rev\_leg2\_totframe4 , rev\_leg2\_totframe5 , rev\_leg2\_totframe6 , rev\_leg2\_totframe7 , rev\_leg2\_totframe8 , rev\_leg2\_totframe9 , rev\_leg2\_errframe0 , rev\_leg2\_errframe1 , rev\_leg2\_errframe2 , rev\_leg2\_errframe3 , rev\_leg2\_errframe4 , rev\_leg2\_errframe5 , rev\_leg2\_errframe6 , rev\_leg2\_errframe7 , rev\_leg2\_errframe8 , rev\_leg2\_errframe9,rev\_leg3\_totframe0 ,rev\_leg3\_totframe1 , rev\_leg3\_totframe2 , rev\_leg3\_totframe3 , rev\_leg3\_totframe4 , rev\_leg3\_totframe5 , rev\_leg3\_totframe6 , rev\_leg3\_totframe7 , rev\_leg3\_totframe8 , rev\_leg3\_totframe9 , rev\_leg3\_errframe0 , rev\_leg3\_errframe1 , rev\_leg3\_errframe2 , rev\_leg3\_errframe3 , rev\_leg3\_errframe4 , rev\_leg3\_errframe5 , rev\_leg3\_errframe6 , rev\_leg3\_errframe7 , rev\_leg3\_errframe8 , rev\_leg3\_errframe9 , rev\_ebnt0 , rev\_ebnt1 , rev\_ebnt2 , rev\_ebnt3 , rev\_ebnt4 , rev\_ebnt5 , rev\_ebnt6 , rev\_ebnt7 , rev\_ebnt8 , rev\_ebnt9 , rev\_leg1\_ebnt0 , rev\_leg1\_ebnt1 , rev\_leg1\_ebnt2 , rev\_leg1\_ebnt3 , rev\_leg1\_ebnt4 , rev\_leg1\_ebnt5 , rev\_leg1\_ebnt6 , rev\_leg1\_ebnt7 , rev\_leg1\_ebnt8 , rev\_leg1\_ebnt9 , rev\_leg2\_ebnt0 , rev\_leg2\_ebnt1 , rev\_leg2\_ebnt2 , rev\_leg2\_ebnt3 , rev\_leg2\_ebnt4 , rev\_leg2\_ebnt5 , rev\_leg2\_ebnt6 , rev\_leg2\_ebnt7 , rev\_leg2\_ebnt8 , rev\_leg2\_ebnt9 , rev\_leg3\_ebnt0 , rev\_leg3\_ebnt1 , rev\_leg3\_ebnt2 , rev\_leg3\_ebnt3 , rev\_leg3\_ebnt4 , rev\_leg3\_ebnt5 , rev\_leg3\_ebnt6 , rev\_leg3\_ebnt7 , rev\_leg3\_ebnt8 , rev\_leg3\_ebnt9 , fwd\_tch\_fer\_ratio\_0\_to\_3 , fwd\_tch\_fer\_ratio\_3\_to\_7 , fwd\_tch\_fer\_ratio\_7\_to\_15 , fwd\_tch\_fer\_ratio\_more\_than\_15 , fwd\_tch\_expect\_fer , fwd\_totframe , fwd\_errframe , fwd\_sch\_totframe , fwd\_sch\_errframe , fwd\_quality , rvs\_quality , last\_rf\_higa1\_count , last\_rf\_higa1\_temp , last\_rf\_higa1\_intervals , last\_rf\_higa1\_begin , last\_rf\_higa1\_end , last\_rf\_higa2\_count , last\_rf\_higa2\_temp , last\_rf\_higa2\_intervals , last\_rf\_higa2\_begin , last\_rf\_higa2\_end , last\_rf\_higa3\_count , last\_rf\_higa3\_temp , last\_rf\_higa3\_intervals , last\_rf\_higa3\_begin , last\_rf\_higa3\_end , last\_rf\_setp\_count , last\_rf\_setp\_temp , last\_rf\_setp\_intervals , last\_rf\_setp\_begin , last\_rf\_setp\_end , first\_psmm\_time , first\_psmm\_act\_str , first\_psmm\_pn\_count , first\_psmm\_cell1 , first\_psmm\_sector1 , first\_psmm\_pn1\_strength , first\_psmm\_pn1\_onewaydelay , first\_psmm\_maho\_pn1\_keep , first\_psmm\_pn1\_pilotset,first\_psmm\_cell2, first\_psmm\_sector2 , first\_psmm\_pn2\_strength , first\_psmm\_pn2\_onewaydelay , first\_psmm\_pn2\_keep , first\_psmm\_pn2\_pilotset , first\_psmm\_cell3 , first\_psmm\_sector3 , first\_psmm\_pn3\_strength , first\_psmm\_pn3\_onewaydelay , first\_psmm\_pn3\_keep , first\_psmm\_pn3\_pilotset , first\_psmm\_cell4 , first\_psmm\_sector4 , first\_psmm\_pn4\_strength , first\_psmm\_pn4\_onewaydelay , first\_psmm\_pn4\_keep , first\_psmm\_pn4\_pilotset , first\_psmm\_cell5 , first\_psmm\_sector5 , first\_psmm\_pn5\_strength , first\_psmm\_pn5\_onewaydelay , first\_psmm\_pn5\_keep , first\_psmm\_pn5\_pilotset , first\_psmm\_cell6 , first\_psmm\_sector6 , first\_psmm\_pn6\_strength , first\_psmm\_pn6\_onewaydelay , first\_psmm\_pn6\_keep , first\_psmm\_pn6\_pilotset , first\_sho\_time , first\_sho\_cause , first\_sho\_post\_agst , first\_sho\_result , last\_maho\_time , init\_maho\_act\_str , last\_maho\_pn\_count , last\_maho\_cell1 , last\_maho\_sector1 , last\_maho\_pn1\_strength , last\_maho\_pn1\_onewaydelay , last\_maho\_pn1\_keep , last\_maho\_pn1\_pilotset , last\_maho\_cell2 , last\_maho\_sector2 , last\_maho\_pn2\_strength , last\_maho\_pn2\_onewaydelay , last\_maho\_pn2\_keep , last\_maho\_pn2\_pilotset , last\_maho\_cell3 , last\_maho\_sector3 , last\_maho\_pn3\_strength , last\_maho\_pn3\_onewaydelay , last\_maho\_pn3\_keep , last\_maho\_pn3\_pilotset , last\_maho\_cell4 , last\_maho\_sector4 , last\_maho\_pn4\_strength , last\_maho\_pn4\_onewaydelay , last\_maho\_pn4\_keep , last\_maho\_pn4\_pilotset , last\_maho\_cell5 , last\_maho\_sector5 , last\_maho\_pn5\_strength , last\_maho\_pn5\_onewaydelay , last\_maho\_pn5\_keep , last\_maho\_pn5\_pilotset , last\_maho\_cell6 , last\_maho\_sector6 , last\_maho\_pn6\_strength , last\_maho\_pn6\_onewaydelay , last\_maho\_pn6\_keep , last\_maho\_pn6\_pilotset , last\_sho\_time , last\_sho\_cause , last\_sho\_result , last\_sho\_post\_agst , last\_psmm\_time , init\_psmm\_act\_str , last\_psmm\_pn\_count , last\_psmm\_cell1 , last\_psmm\_sector1 , last\_psmm\_pn1\_strength , last\_psmm\_pn1\_onewaydelay , last\_psmm\_pn1\_keep , last\_psmm\_pn1\_pilotset , last\_psmm\_cell2 , last\_psmm\_sector2 , last\_psmm\_pn2\_strength , last\_psmm\_pn2\_onewaydelay , last\_psmm\_pn2\_keep , last\_psmm\_pn2\_pilotset , last\_psmm\_cell3 , last\_psmm\_sector3 , last\_psmm\_pn3\_strength , last\_psmm\_pn3\_onewaydelay , last\_psmm\_pn3\_keep , last\_psmm\_pn3\_pilotset , last\_psmm\_cell4 , last\_psmm\_sector4 ,last\_psmm\_pn4\_strength from mod\_cdr\_hw where imsi='460036271583050';

-------------------------------------------------------------------------------------------------------------------------------

17) select omcid , city\_id , ne\_sys\_id , related\_cell , related\_bts , related\_bsc , related\_omc , handle\_index , call\_ref\_num , time\_stamp , chr\_ver , chr\_type , release\_cfc , release\_cause , imsi , esn , call\_state , dialed\_digits , band\_class , pg\_duration , access\_time , access\_time\_msec , srv\_conn\_start\_time , assign\_duration , preamble\_duration , nego\_duration , call\_duration , call\_or\_called\_or\_hho , service\_option , final\_service\_option , mobile\_protocol\_rev , mobile\_protocol\_rev\_inuse , scm , qos\_index , user\_qos\_grade , rev\_fch\_gating , pcf\_ip , iwf\_cpu\_id , md\_path\_ch\_no , pstn\_path\_ch\_no , pcm , a\_e1\_slot , service\_one\_way\_delay , auth\_mode , slot\_cycle\_index , signal\_type , alert\_pitch , signal , privacy\_curr\_type , mob\_term , dcch\_supported , fch\_supported , otd\_supported , qpch\_supported , geo\_location\_type , fch\_frm\_size\_cap , dcch\_frm\_size\_cap , fch\_fwd\_rc\_cap , dcch\_fwd\_rc\_cap , fch\_rev\_rc\_cap , dcch\_rev\_rc\_cap , call\_notproc\_count , access\_chnannel , assign\_channel , accprobe\_ho\_count , addt\_pilot\_count , acc\_ho\_flag , access\_cell , access\_sector , access\_strength , addtpn1\_cell\_id , addtpn1\_sector\_id , addtpn1\_pn\_offset , addtpn1\_strength , addtpn2\_cell\_id , addtpn2\_sector\_id , addtpn2\_pn\_offset , addtpn2\_strength , ppu\_cpu\_id , ms\_para\_meter\_type , ms\_para\_meter1\_value , ms\_para\_meter2\_value , ms\_para\_meter3\_value , ms\_para\_meter4\_value , ms\_para\_meter5\_value , ms\_para\_meter6\_value , ms\_para\_meter7\_value , ms\_para\_meter8\_value , ms\_para\_meter9\_value , ms\_para\_meter10\_value , ms\_para\_meter11\_value , ms\_para\_meter12\_value , ms\_para\_meter13\_value , ms\_para\_meter14\_value , ms\_para\_meter15\_value , ms\_para\_meter16\_value , ms\_para\_meter17\_value , ms\_para\_meter18\_value , ccm\_reserved , access\_carr\_fwd\_load ,access\_carr\_fwd\_basic\_thresh , access\_carr\_fwd\_badleg\_ratio , access\_carr\_channle\_count , access\_carr\_rvs\_equ\_user , access\_carr\_rvs\_rssi , access\_carr\_rvs\_badleg\_ratio , access\_carr\_access\_load , access\_carr\_access\_fail\_rate , init\_rf\_conn\_count , init\_rf\_conn1\_cell , init\_rf\_conn1\_sector , init\_rf\_conn2\_cell , init\_rf\_conn2\_sector , init\_rf\_conn3\_cell , init\_rf\_conn3\_sector , last\_rf\_conn\_time\_vos , last\_rf\_conn\_count , last\_rf\_conn1\_cell , last\_rf\_conn1\_sector , last\_rf\_conn1\_str , last\_rf\_conn1\_onewaydelay , last\_rf\_conn2\_cell , last\_rf\_conn2\_sector , last\_rf\_conn2\_str , last\_rf\_conn2\_onewaydelay , last\_rf\_conn3\_cell , last\_rf\_conn3\_sector , last\_rf\_conn3\_str , last\_rf\_conn3\_onewaydelay , fwd\_fer0 , fwd\_fer1 , fwd\_fer2 , fwd\_fer3 , fwd\_fer4 , fwd\_fer5 , fwd\_fer6 , fwd\_fer7 , fwd\_fer8 , fwd\_fer9 , rev\_totframe0 , rev\_totframe1 , rev\_totframe2 , rev\_totframe3 , rev\_totframe4 , rev\_totframe5 , rev\_totframe6 , rev\_totframe7 , rev\_totframe8 , rev\_totframe9 , rev\_errframe0 , rev\_errframe1 , rev\_errframe2 , rev\_errframe3 , rev\_errframe4 , rev\_errframe5 , rev\_errframe6 , rev\_errframe7 , rev\_errframe8 , rev\_errframe9 , rev\_leg1\_totframe0 , rev\_leg1\_totframe1 , rev\_leg1\_totframe2 , rev\_leg1\_totframe3 , rev\_leg1\_totframe4 , rev\_leg1\_totframe5 , rev\_leg1\_totframe6 , rev\_leg1\_totframe7 , rev\_leg1\_totframe8 , rev\_leg1\_totframe9 , rev\_leg1\_errframe0 , rev\_leg1\_errframe1 , rev\_leg1\_errframe2 , rev\_leg1\_errframe3 , rev\_leg1\_errframe4 , rev\_leg1\_errframe5 , rev\_leg1\_errframe6 , rev\_leg1\_errframe7 , rev\_leg1\_errframe8 , rev\_leg1\_errframe9 , rev\_leg2\_totframe0 , rev\_leg2\_totframe1 , rev\_leg2\_totframe2 , rev\_leg2\_totframe3 , rev\_leg2\_totframe4 , rev\_leg2\_totframe5 , rev\_leg2\_totframe6 , rev\_leg2\_totframe7 , rev\_leg2\_totframe8 , rev\_leg2\_totframe9 , rev\_leg2\_errframe0 , rev\_leg2\_errframe1 , rev\_leg2\_errframe2 , rev\_leg2\_errframe3 , rev\_leg2\_errframe4 , rev\_leg2\_errframe5 , rev\_leg2\_errframe6 , rev\_leg2\_errframe7 , rev\_leg2\_errframe8 , rev\_leg2\_errframe9,rev\_leg3\_totframe0 ,rev\_leg3\_totframe1 , rev\_leg3\_totframe2 , rev\_leg3\_totframe3 , rev\_leg3\_totframe4 , rev\_leg3\_totframe5 , rev\_leg3\_totframe6 , rev\_leg3\_totframe7 , rev\_leg3\_totframe8 , rev\_leg3\_totframe9 , rev\_leg3\_errframe0 , rev\_leg3\_errframe1 , rev\_leg3\_errframe2 , rev\_leg3\_errframe3 , rev\_leg3\_errframe4 , rev\_leg3\_errframe5 , rev\_leg3\_errframe6 , rev\_leg3\_errframe7 , rev\_leg3\_errframe8 , rev\_leg3\_errframe9 , rev\_ebnt0 , rev\_ebnt1 , rev\_ebnt2 , rev\_ebnt3 , rev\_ebnt4 , rev\_ebnt5 , rev\_ebnt6 , rev\_ebnt7 , rev\_ebnt8 , rev\_ebnt9 , rev\_leg1\_ebnt0 , rev\_leg1\_ebnt1 , rev\_leg1\_ebnt2 , rev\_leg1\_ebnt3 , rev\_leg1\_ebnt4 , rev\_leg1\_ebnt5 , rev\_leg1\_ebnt6 , rev\_leg1\_ebnt7 , rev\_leg1\_ebnt8 , rev\_leg1\_ebnt9 , rev\_leg2\_ebnt0 , rev\_leg2\_ebnt1 , rev\_leg2\_ebnt2 , rev\_leg2\_ebnt3 , rev\_leg2\_ebnt4 , rev\_leg2\_ebnt5 , rev\_leg2\_ebnt6 , rev\_leg2\_ebnt7 , rev\_leg2\_ebnt8 , rev\_leg2\_ebnt9 , rev\_leg3\_ebnt0 , rev\_leg3\_ebnt1 , rev\_leg3\_ebnt2 , rev\_leg3\_ebnt3 , rev\_leg3\_ebnt4 , rev\_leg3\_ebnt5 , rev\_leg3\_ebnt6 , rev\_leg3\_ebnt7 , rev\_leg3\_ebnt8 , rev\_leg3\_ebnt9 , fwd\_tch\_fer\_ratio\_0\_to\_3 , fwd\_tch\_fer\_ratio\_3\_to\_7 , fwd\_tch\_fer\_ratio\_7\_to\_15 , fwd\_tch\_fer\_ratio\_more\_than\_15 , fwd\_tch\_expect\_fer , fwd\_totframe , fwd\_errframe , fwd\_sch\_totframe , fwd\_sch\_errframe , fwd\_quality , rvs\_quality , last\_rf\_higa1\_count , last\_rf\_higa1\_temp , last\_rf\_higa1\_intervals , last\_rf\_higa1\_begin , last\_rf\_higa1\_end , last\_rf\_higa2\_count , last\_rf\_higa2\_temp , last\_rf\_higa2\_intervals , last\_rf\_higa2\_begin , last\_rf\_higa2\_end , last\_rf\_higa3\_count , last\_rf\_higa3\_temp , last\_rf\_higa3\_intervals , last\_rf\_higa3\_begin , last\_rf\_higa3\_end , last\_rf\_setp\_count , last\_rf\_setp\_temp , last\_rf\_setp\_intervals , last\_rf\_setp\_begin , last\_rf\_setp\_end , first\_psmm\_time , first\_psmm\_act\_str , first\_psmm\_pn\_count , first\_psmm\_cell1 , first\_psmm\_sector1 , first\_psmm\_pn1\_strength , first\_psmm\_pn1\_onewaydelay , first\_psmm\_maho\_pn1\_keep , first\_psmm\_pn1\_pilotset,first\_psmm\_cell2, first\_psmm\_sector2 , first\_psmm\_pn2\_strength , first\_psmm\_pn2\_onewaydelay , first\_psmm\_pn2\_keep , first\_psmm\_pn2\_pilotset , first\_psmm\_cell3 , first\_psmm\_sector3 , first\_psmm\_pn3\_strength , first\_psmm\_pn3\_onewaydelay , first\_psmm\_pn3\_keep , first\_psmm\_pn3\_pilotset , first\_psmm\_cell4 , first\_psmm\_sector4 , first\_psmm\_pn4\_strength , first\_psmm\_pn4\_onewaydelay , first\_psmm\_pn4\_keep , first\_psmm\_pn4\_pilotset , first\_psmm\_cell5 , first\_psmm\_sector5 , first\_psmm\_pn5\_strength , first\_psmm\_pn5\_onewaydelay , first\_psmm\_pn5\_keep , first\_psmm\_pn5\_pilotset , first\_psmm\_cell6 , first\_psmm\_sector6 , first\_psmm\_pn6\_strength , first\_psmm\_pn6\_onewaydelay , first\_psmm\_pn6\_keep , first\_psmm\_pn6\_pilotset , first\_sho\_time , first\_sho\_cause , first\_sho\_post\_agst , first\_sho\_result , last\_maho\_time , init\_maho\_act\_str , last\_maho\_pn\_count , last\_maho\_cell1 , last\_maho\_sector1 , last\_maho\_pn1\_strength , last\_maho\_pn1\_onewaydelay , last\_maho\_pn1\_keep , last\_maho\_pn1\_pilotset , last\_maho\_cell2 , last\_maho\_sector2 , last\_maho\_pn2\_strength , last\_maho\_pn2\_onewaydelay , last\_maho\_pn2\_keep , last\_maho\_pn2\_pilotset , last\_maho\_cell3 , last\_maho\_sector3 , last\_maho\_pn3\_strength , last\_maho\_pn3\_onewaydelay , last\_maho\_pn3\_keep , last\_maho\_pn3\_pilotset , last\_maho\_cell4 , last\_maho\_sector4 , last\_maho\_pn4\_strength , last\_maho\_pn4\_onewaydelay , last\_maho\_pn4\_keep , last\_maho\_pn4\_pilotset , last\_maho\_cell5 , last\_maho\_sector5 , last\_maho\_pn5\_strength , last\_maho\_pn5\_onewaydelay , last\_maho\_pn5\_keep , last\_maho\_pn5\_pilotset , last\_maho\_cell6 , last\_maho\_sector6 , last\_maho\_pn6\_strength , last\_maho\_pn6\_onewaydelay , last\_maho\_pn6\_keep , last\_maho\_pn6\_pilotset , last\_sho\_time , last\_sho\_cause , last\_sho\_result , last\_sho\_post\_agst , last\_psmm\_time , init\_psmm\_act\_str , last\_psmm\_pn\_count , last\_psmm\_cell1 , last\_psmm\_sector1 , last\_psmm\_pn1\_strength , last\_psmm\_pn1\_onewaydelay , last\_psmm\_pn1\_keep , last\_psmm\_pn1\_pilotset , last\_psmm\_cell2 , last\_psmm\_sector2 , last\_psmm\_pn2\_strength , last\_psmm\_pn2\_onewaydelay , last\_psmm\_pn2\_keep , last\_psmm\_pn2\_pilotset , last\_psmm\_cell3 , last\_psmm\_sector3 , last\_psmm\_pn3\_strength , last\_psmm\_pn3\_onewaydelay , last\_psmm\_pn3\_keep , last\_psmm\_pn3\_pilotset , last\_psmm\_cell4 , last\_psmm\_sector4 ,last\_psmm\_pn4\_strength,last\_psmm\_pn4\_onewaydelay , last\_psmm\_pn4\_keep , last\_psmm\_pn4\_pilotset , last\_psmm\_cell5 , last\_psmm\_sector5 , last\_psmm\_pn5\_strength , last\_psmm\_pn5\_onewaydelay , last\_psmm\_pn5\_keep , last\_psmm\_pn5\_pilotset , last\_psmm\_cell6 , last\_psmm\_sector6 , last\_psmm\_pn6\_strength , last\_psmm\_pn6\_onewaydelay , last\_psmm\_pn6\_keep , last\_psmm\_pn6\_pilotset , first\_sho\_fail\_time , first\_sho\_fail\_cause , last\_ho\_block\_time , last\_ho\_block\_cause , last\_ho\_block\_pn\_count , last\_ho\_block\_pn , last\_sho\_fail\_time , last\_sho\_fail\_cause , one\_pilot\_count , one\_pilot\_duration , two\_pilots\_count , two\_pilot\_duration , three\_pilots\_count , threr\_pilot\_duration , four\_pilots\_count , four\_pilot\_duration , five\_pilots\_count , five\_pilot\_duration , six\_pilots\_count , six\_pilot\_duration , psmm\_received\_count , ppsmm\_received\_count , sho\_count , sho\_fail\_count , loc\_s\_add\_count , loc\_s\_add\_fail\_count , loc\_sr\_add\_count , loc\_sr\_add\_fail\_count , loc\_s\_drop\_count , loc\_s\_drop\_fail\_count , loc\_sr\_drop\_count , loc\_sr\_drop\_fail\_count , ext\_s\_add\_count , ext\_s\_add\_fail\_count , ext\_s\_drop\_count , ext\_s\_drop\_fail\_count , hho\_type , hho\_tgt\_freq , hho\_tgt\_count , hho\_tgt1\_market\_id , hho\_tgt1\_switch\_number , hho\_tgt1\_cell , hho\_tgt1\_sector , hho\_tgt2\_cell , hho\_tgt2\_sector , hho\_tgt3\_cell , hho\_tgt3\_sector , hho\_result , hho\_maho\_time , hho\_maho\_act\_count , hho\_maho\_act\_str , hho\_maho\_pn\_count , hho\_maho\_cell1 , hho\_maho\_sector1 , hho\_maho\_pn1\_strength , hho\_maho\_pn1\_onewaydelay , hho\_maho\_pn1\_pilotset , hho\_maho\_cell2 , hho\_maho\_sector2 , hho\_maho\_pn2\_strength , hho\_maho\_pn2\_onewaydelay , hho\_maho\_pn2\_pilotset , hho\_maho\_cell3 , hho\_maho\_sector3 , hho\_maho\_pn3\_strength , hho\_maho\_pn3\_onewaydelay , hho\_maho\_pn3\_pilotset , hho\_maho\_cell4 , hho\_maho\_sector4 , hho\_maho\_pn4\_strength , hho\_maho\_pn4\_onewaydelay , hho\_maho\_pn4\_pilotset , hho\_maho\_cell5 , hho\_maho\_sector5 , hho\_maho\_pn5\_strength , hho\_maho\_pn5\_onewaydelay , hho\_maho\_pn5\_pilotset , hho\_maho\_cell6 , hho\_maho\_sector6 , hho\_maho\_pn6\_strength , hho\_maho\_pn6\_onewaydelay , hho\_maho\_pn6\_pilotset , ics\_count , ics\_fail\_count ,ics\_begin\_time from mod\_cdr\_hw where imsi='460036271583050';

---------------------------------------------------------------------------------------------------------------------------

18) select omcid , city\_id , ne\_sys\_id , related\_cell , related\_bts , related\_bsc , related\_omc , handle\_index , call\_ref\_num , time\_stamp , chr\_ver , chr\_type , release\_cfc , release\_cause , imsi , esn , call\_state , dialed\_digits , band\_class , pg\_duration , access\_time , access\_time\_msec , srv\_conn\_start\_time , assign\_duration , preamble\_duration , nego\_duration , call\_duration , call\_or\_called\_or\_hho , service\_option , final\_service\_option , mobile\_protocol\_rev , mobile\_protocol\_rev\_inuse , scm , qos\_index , user\_qos\_grade , rev\_fch\_gating , pcf\_ip , iwf\_cpu\_id , md\_path\_ch\_no , pstn\_path\_ch\_no , pcm , a\_e1\_slot , service\_one\_way\_delay , auth\_mode , slot\_cycle\_index , signal\_type , alert\_pitch , signal , privacy\_curr\_type , mob\_term , dcch\_supported , fch\_supported , otd\_supported , qpch\_supported , geo\_location\_type , fch\_frm\_size\_cap , dcch\_frm\_size\_cap , fch\_fwd\_rc\_cap , dcch\_fwd\_rc\_cap , fch\_rev\_rc\_cap , dcch\_rev\_rc\_cap , call\_notproc\_count , access\_chnannel , assign\_channel , accprobe\_ho\_count , addt\_pilot\_count , acc\_ho\_flag , access\_cell , access\_sector , access\_strength , addtpn1\_cell\_id , addtpn1\_sector\_id , addtpn1\_pn\_offset , addtpn1\_strength , addtpn2\_cell\_id , addtpn2\_sector\_id , addtpn2\_pn\_offset , addtpn2\_strength , ppu\_cpu\_id , ms\_para\_meter\_type , ms\_para\_meter1\_value , ms\_para\_meter2\_value , ms\_para\_meter3\_value , ms\_para\_meter4\_value , ms\_para\_meter5\_value , ms\_para\_meter6\_value , ms\_para\_meter7\_value , ms\_para\_meter8\_value , ms\_para\_meter9\_value , ms\_para\_meter10\_value , ms\_para\_meter11\_value , ms\_para\_meter12\_value , ms\_para\_meter13\_value , ms\_para\_meter14\_value , ms\_para\_meter15\_value , ms\_para\_meter16\_value , ms\_para\_meter17\_value , ms\_para\_meter18\_value , ccm\_reserved , access\_carr\_fwd\_load ,access\_carr\_fwd\_basic\_thresh , access\_carr\_fwd\_badleg\_ratio , access\_carr\_channle\_count , access\_carr\_rvs\_equ\_user , access\_carr\_rvs\_rssi , access\_carr\_rvs\_badleg\_ratio , access\_carr\_access\_load , access\_carr\_access\_fail\_rate , init\_rf\_conn\_count , init\_rf\_conn1\_cell , init\_rf\_conn1\_sector , init\_rf\_conn2\_cell , init\_rf\_conn2\_sector , init\_rf\_conn3\_cell , init\_rf\_conn3\_sector , last\_rf\_conn\_time\_vos , last\_rf\_conn\_count , last\_rf\_conn1\_cell , last\_rf\_conn1\_sector , last\_rf\_conn1\_str , last\_rf\_conn1\_onewaydelay , last\_rf\_conn2\_cell , last\_rf\_conn2\_sector , last\_rf\_conn2\_str , last\_rf\_conn2\_onewaydelay , last\_rf\_conn3\_cell , last\_rf\_conn3\_sector , last\_rf\_conn3\_str , last\_rf\_conn3\_onewaydelay , fwd\_fer0 , fwd\_fer1 , fwd\_fer2 , fwd\_fer3 , fwd\_fer4 , fwd\_fer5 , fwd\_fer6 , fwd\_fer7 , fwd\_fer8 , fwd\_fer9 , rev\_totframe0 , rev\_totframe1 , rev\_totframe2 , rev\_totframe3 , rev\_totframe4 , rev\_totframe5 , rev\_totframe6 , rev\_totframe7 , rev\_totframe8 , rev\_totframe9 , rev\_errframe0 , rev\_errframe1 , rev\_errframe2 , rev\_errframe3 , rev\_errframe4 , rev\_errframe5 , rev\_errframe6 , rev\_errframe7 , rev\_errframe8 , rev\_errframe9 , rev\_leg1\_totframe0 , rev\_leg1\_totframe1 , rev\_leg1\_totframe2 , rev\_leg1\_totframe3 , rev\_leg1\_totframe4 , rev\_leg1\_totframe5 , rev\_leg1\_totframe6 , rev\_leg1\_totframe7 , rev\_leg1\_totframe8 , rev\_leg1\_totframe9 , rev\_leg1\_errframe0 , rev\_leg1\_errframe1 , rev\_leg1\_errframe2 , rev\_leg1\_errframe3 , rev\_leg1\_errframe4 , rev\_leg1\_errframe5 , rev\_leg1\_errframe6 , rev\_leg1\_errframe7 , rev\_leg1\_errframe8 , rev\_leg1\_errframe9 , rev\_leg2\_totframe0 , rev\_leg2\_totframe1 , rev\_leg2\_totframe2 , rev\_leg2\_totframe3 , rev\_leg2\_totframe4 , rev\_leg2\_totframe5 , rev\_leg2\_totframe6 , rev\_leg2\_totframe7 , rev\_leg2\_totframe8 , rev\_leg2\_totframe9 , rev\_leg2\_errframe0 , rev\_leg2\_errframe1 , rev\_leg2\_errframe2 , rev\_leg2\_errframe3 , rev\_leg2\_errframe4 , rev\_leg2\_errframe5 , rev\_leg2\_errframe6 , rev\_leg2\_errframe7 , rev\_leg2\_errframe8 , rev\_leg2\_errframe9,rev\_leg3\_totframe0 ,rev\_leg3\_totframe1 , rev\_leg3\_totframe2 , rev\_leg3\_totframe3 , rev\_leg3\_totframe4 , rev\_leg3\_totframe5 , rev\_leg3\_totframe6 , rev\_leg3\_totframe7 , rev\_leg3\_totframe8 , rev\_leg3\_totframe9 , rev\_leg3\_errframe0 , rev\_leg3\_errframe1 , rev\_leg3\_errframe2 , rev\_leg3\_errframe3 , rev\_leg3\_errframe4 , rev\_leg3\_errframe5 , rev\_leg3\_errframe6 , rev\_leg3\_errframe7 , rev\_leg3\_errframe8 , rev\_leg3\_errframe9 , rev\_ebnt0 , rev\_ebnt1 , rev\_ebnt2 , rev\_ebnt3 , rev\_ebnt4 , rev\_ebnt5 , rev\_ebnt6 , rev\_ebnt7 , rev\_ebnt8 , rev\_ebnt9 , rev\_leg1\_ebnt0 , rev\_leg1\_ebnt1 , rev\_leg1\_ebnt2 , rev\_leg1\_ebnt3 , rev\_leg1\_ebnt4 , rev\_leg1\_ebnt5 , rev\_leg1\_ebnt6 , rev\_leg1\_ebnt7 , rev\_leg1\_ebnt8 , rev\_leg1\_ebnt9 , rev\_leg2\_ebnt0 , rev\_leg2\_ebnt1 , rev\_leg2\_ebnt2 , rev\_leg2\_ebnt3 , rev\_leg2\_ebnt4 , rev\_leg2\_ebnt5 , rev\_leg2\_ebnt6 , rev\_leg2\_ebnt7 , rev\_leg2\_ebnt8 , rev\_leg2\_ebnt9 , rev\_leg3\_ebnt0 , rev\_leg3\_ebnt1 , rev\_leg3\_ebnt2 , rev\_leg3\_ebnt3 , rev\_leg3\_ebnt4 , rev\_leg3\_ebnt5 , rev\_leg3\_ebnt6 , rev\_leg3\_ebnt7 , rev\_leg3\_ebnt8 , rev\_leg3\_ebnt9 , fwd\_tch\_fer\_ratio\_0\_to\_3 , fwd\_tch\_fer\_ratio\_3\_to\_7 , fwd\_tch\_fer\_ratio\_7\_to\_15 , fwd\_tch\_fer\_ratio\_more\_than\_15 , fwd\_tch\_expect\_fer , fwd\_totframe , fwd\_errframe , fwd\_sch\_totframe , fwd\_sch\_errframe , fwd\_quality , rvs\_quality , last\_rf\_higa1\_count , last\_rf\_higa1\_temp , last\_rf\_higa1\_intervals , last\_rf\_higa1\_begin , last\_rf\_higa1\_end , last\_rf\_higa2\_count , last\_rf\_higa2\_temp , last\_rf\_higa2\_intervals , last\_rf\_higa2\_begin , last\_rf\_higa2\_end , last\_rf\_higa3\_count , last\_rf\_higa3\_temp , last\_rf\_higa3\_intervals , last\_rf\_higa3\_begin , last\_rf\_higa3\_end , last\_rf\_setp\_count , last\_rf\_setp\_temp , last\_rf\_setp\_intervals , last\_rf\_setp\_begin , last\_rf\_setp\_end , first\_psmm\_time , first\_psmm\_act\_str , first\_psmm\_pn\_count , first\_psmm\_cell1 , first\_psmm\_sector1 , first\_psmm\_pn1\_strength , first\_psmm\_pn1\_onewaydelay , first\_psmm\_maho\_pn1\_keep , first\_psmm\_pn1\_pilotset,first\_psmm\_cell2, first\_psmm\_sector2 , first\_psmm\_pn2\_strength , first\_psmm\_pn2\_onewaydelay , first\_psmm\_pn2\_keep , first\_psmm\_pn2\_pilotset , first\_psmm\_cell3 , first\_psmm\_sector3 , first\_psmm\_pn3\_strength , first\_psmm\_pn3\_onewaydelay , first\_psmm\_pn3\_keep , first\_psmm\_pn3\_pilotset , first\_psmm\_cell4 , first\_psmm\_sector4 , first\_psmm\_pn4\_strength , first\_psmm\_pn4\_onewaydelay , first\_psmm\_pn4\_keep , first\_psmm\_pn4\_pilotset , first\_psmm\_cell5 , first\_psmm\_sector5 , first\_psmm\_pn5\_strength , first\_psmm\_pn5\_onewaydelay , first\_psmm\_pn5\_keep , first\_psmm\_pn5\_pilotset , first\_psmm\_cell6 , first\_psmm\_sector6 , first\_psmm\_pn6\_strength , first\_psmm\_pn6\_onewaydelay , first\_psmm\_pn6\_keep , first\_psmm\_pn6\_pilotset , first\_sho\_time , first\_sho\_cause , first\_sho\_post\_agst , first\_sho\_result , last\_maho\_time , init\_maho\_act\_str , last\_maho\_pn\_count , last\_maho\_cell1 , last\_maho\_sector1 , last\_maho\_pn1\_strength , last\_maho\_pn1\_onewaydelay , last\_maho\_pn1\_keep , last\_maho\_pn1\_pilotset , last\_maho\_cell2 , last\_maho\_sector2 , last\_maho\_pn2\_strength , last\_maho\_pn2\_onewaydelay , last\_maho\_pn2\_keep , last\_maho\_pn2\_pilotset , last\_maho\_cell3 , last\_maho\_sector3 , last\_maho\_pn3\_strength , last\_maho\_pn3\_onewaydelay , last\_maho\_pn3\_keep , last\_maho\_pn3\_pilotset , last\_maho\_cell4 , last\_maho\_sector4 , last\_maho\_pn4\_strength , last\_maho\_pn4\_onewaydelay , last\_maho\_pn4\_keep , last\_maho\_pn4\_pilotset , last\_maho\_cell5 , last\_maho\_sector5 , last\_maho\_pn5\_strength , last\_maho\_pn5\_onewaydelay , last\_maho\_pn5\_keep , last\_maho\_pn5\_pilotset , last\_maho\_cell6 , last\_maho\_sector6 , last\_maho\_pn6\_strength , last\_maho\_pn6\_onewaydelay , last\_maho\_pn6\_keep , last\_maho\_pn6\_pilotset , last\_sho\_time , last\_sho\_cause , last\_sho\_result , last\_sho\_post\_agst , last\_psmm\_time , init\_psmm\_act\_str , last\_psmm\_pn\_count , last\_psmm\_cell1 , last\_psmm\_sector1 , last\_psmm\_pn1\_strength , last\_psmm\_pn1\_onewaydelay , last\_psmm\_pn1\_keep , last\_psmm\_pn1\_pilotset , last\_psmm\_cell2 , last\_psmm\_sector2 , last\_psmm\_pn2\_strength , last\_psmm\_pn2\_onewaydelay , last\_psmm\_pn2\_keep , last\_psmm\_pn2\_pilotset , last\_psmm\_cell3 , last\_psmm\_sector3 , last\_psmm\_pn3\_strength , last\_psmm\_pn3\_onewaydelay , last\_psmm\_pn3\_keep , last\_psmm\_pn3\_pilotset , last\_psmm\_cell4 , last\_psmm\_sector4 ,last\_psmm\_pn4\_strength,last\_psmm\_pn4\_onewaydelay , last\_psmm\_pn4\_keep , last\_psmm\_pn4\_pilotset , last\_psmm\_cell5 , last\_psmm\_sector5 , last\_psmm\_pn5\_strength , last\_psmm\_pn5\_onewaydelay , last\_psmm\_pn5\_keep , last\_psmm\_pn5\_pilotset , last\_psmm\_cell6 , last\_psmm\_sector6 , last\_psmm\_pn6\_strength , last\_psmm\_pn6\_onewaydelay , last\_psmm\_pn6\_keep , last\_psmm\_pn6\_pilotset , first\_sho\_fail\_time , first\_sho\_fail\_cause , last\_ho\_block\_time , last\_ho\_block\_cause , last\_ho\_block\_pn\_count , last\_ho\_block\_pn , last\_sho\_fail\_time , last\_sho\_fail\_cause , one\_pilot\_count , one\_pilot\_duration , two\_pilots\_count , two\_pilot\_duration , three\_pilots\_count , threr\_pilot\_duration , four\_pilots\_count , four\_pilot\_duration , five\_pilots\_count , five\_pilot\_duration , six\_pilots\_count , six\_pilot\_duration , psmm\_received\_count , ppsmm\_received\_count , sho\_count , sho\_fail\_count , loc\_s\_add\_count , loc\_s\_add\_fail\_count , loc\_sr\_add\_count , loc\_sr\_add\_fail\_count , loc\_s\_drop\_count , loc\_s\_drop\_fail\_count , loc\_sr\_drop\_count , loc\_sr\_drop\_fail\_count , ext\_s\_add\_count , ext\_s\_add\_fail\_count , ext\_s\_drop\_count , ext\_s\_drop\_fail\_count , hho\_type , hho\_tgt\_freq , hho\_tgt\_count , hho\_tgt1\_market\_id , hho\_tgt1\_switch\_numeric , hho\_tgt1\_cell , hho\_tgt1\_sector , hho\_tgt2\_cell , hho\_tgt2\_sector , hho\_tgt3\_cell , hho\_tgt3\_sector , hho\_result , hho\_maho\_time , hho\_maho\_act\_count , hho\_maho\_act\_str , hho\_maho\_pn\_count , hho\_maho\_cell1 , hho\_maho\_sector1 , hho\_maho\_pn1\_strength , hho\_maho\_pn1\_onewaydelay , hho\_maho\_pn1\_pilotset , hho\_maho\_cell2 , hho\_maho\_sector2 , hho\_maho\_pn2\_strength , hho\_maho\_pn2\_onewaydelay , hho\_maho\_pn2\_pilotset , hho\_maho\_cell3 , hho\_maho\_sector3 , hho\_maho\_pn3\_strength , hho\_maho\_pn3\_onewaydelay , hho\_maho\_pn3\_pilotset , hho\_maho\_cell4 , hho\_maho\_sector4 , hho\_maho\_pn4\_strength , hho\_maho\_pn4\_onewaydelay , hho\_maho\_pn4\_pilotset , hho\_maho\_cell5 , hho\_maho\_sector5 , hho\_maho\_pn5\_strength , hho\_maho\_pn5\_onewaydelay , hho\_maho\_pn5\_pilotset , hho\_maho\_cell6 , hho\_maho\_sector6 , hho\_maho\_pn6\_strength , hho\_maho\_pn6\_onewaydelay , hho\_maho\_pn6\_pilotset , ics\_count , ics\_fail\_count ,ics\_begin\_time ,ics\_begin\_tgt\_count , ics\_begin\_src\_count , ics\_begin\_tgt\_cell , ics\_begin\_tgt\_sector , ics\_begin\_src1\_cell , ics\_begin\_src1\_sector , ics\_begin\_src2\_cell , ics\_begin\_src2\_sector , ics\_end\_time , ics\_end\_tgt\_count , ics\_end\_src\_count , ics\_end\_tgt\_cell , ics\_end\_tgt\_sector , ics\_end\_src1\_cell , ics\_end\_src1\_sector , ics\_end\_src2\_cell , ics\_end\_src2\_sector , sch\_fwd\_rc3\_cap , sch\_fwd\_rc4\_cap , sch\_fwd\_dynamic\_apply\_count , sch\_fwd\_extend\_apply\_count , sch\_fwd\_release\_apply\_count , sch\_fwd\_upgrade\_rate\_count , sch\_fwd\_lower\_rate\_count , sch\_fwd\_keep\_rate\_count , sch\_fwd\_1x\_req\_up\_rate , sch\_fwd\_2x\_sch\_up\_rate , sch\_fwd\_4x\_sch\_up\_rate , sch\_fwd\_8x\_sch\_up\_rate , sch\_fwd\_16x\_sch\_up\_rate , pilot\_0\_3\_duration , pilot\_3\_6\_duration , pilot\_6\_9\_duration , pilot\_9\_15\_duration , pilot\_15\_32\_duration , sdrop\_count , last\_sdrop\_time , fwd\_fch\_assigned\_rc\_type , rev\_ratereduction , rev\_tch\_target\_fer , rrm\_reserved , voice\_abnormal\_fmr , pwr\_abnormal\_flag , abnormal\_flag\_rsvd , dpus\_cpu\_id , dsp\_id , dsp\_occupancy , voice\_count\_per\_dsp , data\_count\_per\_dsp , eib\_count , bytes\_fwd , bytes\_rvs , is\_only\_eighth\_frame\_up , is\_only\_eighth\_frame\_down , is\_trau\_to\_mux\_msg\_fail , is\_trau\_to\_tc\_msg\_fail , is\_mux\_to\_trau\_msg\_fail , is\_mux\_to\_fpmdc\_msg\_fail , is\_fpmdc\_to\_mux\_msg\_fail , is\_fpmdc\_to\_bts\_msg\_fail , only\_eighth\_frame\_up\_count , only\_eighth\_frame\_down\_count , time\_adjust\_abnormal\_count , rev\_errframe , rev\_sch\_errframe , fer\_3\_percent\_count , fer\_3\_to\_7\_percent\_count , fer\_7\_to\_15\_percent\_count , fer\_over\_15\_percent\_count , fwd\_lostt\_on\_sig\_3\_rate , fwd\_lostt\_on\_sig\_3\_to\_7 , fwd\_lostt\_on\_sig\_7\_to\_15 , fwd\_lostt\_on\_sig\_over\_15 , rev\_frame\_all\_sig\_3 , rev\_frame\_all\_sig\_3\_to\_7 , rev\_frame\_all\_sig\_7\_to\_15 , rev\_frame\_all\_sig\_over\_15 , rx\_frame\_from\_tc\_1\_percent , rx\_frame\_from\_tc\_2\_percent , rx\_frame\_from\_tc\_4\_percent , rx\_frame\_from\_tc\_8\_percent , rx\_frame\_from\_tc\_null , rx\_frame\_from\_tc\_other , tx\_frame\_to\_tc\_1\_percent , tx\_frame\_to\_tc\_2\_percent , tx\_frame\_to\_tc\_4\_percent , tx\_frame\_to\_tc\_8\_percent , tx\_frame\_to\_tc\_null , tx\_frame\_to\_tc\_other , sch\_rev\_1x\_average\_fer , sch\_rev\_2x\_average\_fer , sch\_rev\_4x\_average\_fer , sch\_rev\_8x\_average\_fer , sch\_rev\_16x\_average\_fer , sch\_rev\_32x\_average\_fer , sig\_leg\_pwr\_rev\_higa\_count , ppp\_session\_on , ppu\_rp\_frame\_number , ppu\_rp\_slot\_number ,bpu\_rp\_frame\_number from mod\_cdr\_hw where imsi='460036271583050';

----------------------------------------------------------------------------------------------------------------------------

19) select \* from mod\_cdr\_hw where imsi='460036271583050';

## 3、测试结果

### 3.1串行测试

#### 3.1.1 行存储模式

##### 3.1.1.1 数据入库测试

###### 3.1.1.1.1 数据表无索引

下面是基准表（一倍容量）的情况：

|  |  |
| --- | --- |
| **mod\_cdr\_hw** | |
| 记录数 | 28,018,657 |
| 文本存储大小 | 48184548331Byte = 44.8GB |
| 压缩算法（x级压缩） | **Zlib 5级** |
| 入库之后大小 | 29G |
| 压缩比 | 29/44.8=0.647 = 64.7% |
| 入库速度 | 664秒 69M/s |
| 收集统计信息时间(no index) | 241秒（采用率为默认值 10%） |
| 收集统计信息时间(index) | 248秒 |

|  |  |
| --- | --- |
| **mod\_do\_hw** | |
| 记录数 | 7302774\*10 = 73027740 |
| 文本存储大小 | 10389112752=9.7GB \* 10 = 97GB |
| 压缩算法（x级压缩） | **Zlib 5级** |
| 入库之后大小 | 58G |
| 压缩比 | 58/97=0.598 = 59.8% |
| 入库速度 | 1086秒 91.5M/s |
| 收集统计信息时间(no index) | 189秒（采用率为默认值 10%） |
| 收集统计信息时间(index) | 194秒 |

|  |  |
| --- | --- |
| **mod\_cdr\_hw** | |
| 记录数 | 28,018,657 |
| 文本存储大小 | 48184548331Byte = 44.8GB |
| 压缩算法（x级压缩） | **quicklz 1级(该压缩算法只能设置为1)** |
| 入库之后大小 | 37G |
| 压缩比 | 37/44.8=0.826 = 82.6% |
| 入库速度 | 632秒 72.6M/s |
| 收集统计信息时间 | 247秒（采用率为默认值 10%） |

|  |  |
| --- | --- |
| **mod\_do\_hw** | |
| 记录数 | 7302774\*10 = 73027740 |
| 文本存储大小 | 10389112752=9.7GB \* 10 = 97GB |
| 压缩算法（x级压缩） | **quicklz 1级(该压缩算法只能设置为1)** |
| 入库之后大小 | 78G |
| 压缩比 | 78/97=0.804 = 80.4% |
| 入库速度 | 829秒 119.8M/s |
| 收集统计信息时间 | 186秒（采用率为默认值 10%） |

###### 3.1.1.1.2 数据表有索引

|  |  |
| --- | --- |
| **mod\_cdr\_hw** | |
| 记录数 | 28,018,657 |
| 文本存储大小 | 48184548331Byte = 44.8GB |
| 压缩算法（x级压缩） | **Zlib 5级** |
| 索引个数 | 3 |
| 索引情况 | Imsi /access\_cell/release\_cfc 每个字段上一个btree index |
| 索引总大小 | 908 MB | ind\_mod\_cdr\_hw\_release\_cfc  786 MB | ind\_mod\_cdr\_hw\_accesscell  1083 MB | ind\_mod\_cdr\_hw |
| 入库之后表大小 | 29G |
| 压缩比 | 29/44.8=0.647 = 64.7% |
| 入库速度 | 768秒 (44.8+2.71)/768= 83.4M/s |

|  |  |
| --- | --- |
| **mod\_do\_hw** | |
| 记录数 | 7302774\*10 = 73027740 |
| 文本存储大小 | 10389112752=9.7GB \* 10 = 97GB |
| 压缩算法（x级压缩） | **Zlib 5级** |
| 索引个数 | 1 |
| 索引情况 | Imsi 上一个btree |
| 索引总大小 | 2897 MB | ind\_mod\_do\_hw\_imsi |
| 入库之后大小 | 58G |
| 压缩比 | 58/97=0.598 = 59.8% |
| 入库速度 | 1242秒 (97+2.83)/1242= 82.3M/s |

##### 3.1.1.2 系统性能指标

入库时的性能指标（录像/截屏）-----针对无索引的数据入库

###### 3.1.1.2.1 master host

table: mod\_cdr\_hw

/tmp/nmonreport/mdw-3112-cdr.txt

**CPU**

**DISK**

**MEMORY**

**NetWork**

table: mod\_do\_hw

/tmp/nmonreport/mdw-3112-do.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

###### 3.1.1.2.2 segment host ---sdw1

table: mod\_cdr\_hw

/tmp/sdw1-3112-cdr.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

table: mod\_do\_hw

/tmp/sdw1-3112-do.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

###### 3.1.1.2.3 segment host ---sdw2

table: mod\_cdr\_hw

/tmp/sdw2-3112-cdr.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

table: mod\_do\_hw

/tmp/sdw2-3112-do.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

###### 3.1.1.2.4 segment host ---sdw3

table: mod\_cdr\_hw

/tmp/sdw3-3112-cdr.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

table: mod\_do\_hw

/tmp/sdw3-3112-do.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

###### 3.1.1.2.5 segment host ---sdw4

table: mod\_cdr\_hw

/tmp/sdw4-3112-cdr.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

table: mod\_do\_hw

/tmp/sdw4-3112-do.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

##### 3.1.1.3 查询性能指标

###### 3.1.1.3.1 数据表无索引

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sql 序号 | Sql 文本 | 返回结果集 | 首次查询耗时 | 二次查询耗时 |
| 1 | explain analyze select \* from r\_mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_cell=1039; | 28657 | 17.87s | 16.3s |
| 2 | Sql太长，详见测试数据 sql 2 | 28657 | 15.65s | 15.72s |
| 3 | explain analyze select \* from r\_mod\_cdr\_hw  where access\_cell=1039; | 28657 | 15.45s | 15.26s |
| 4 | explain analyze select \* from r\_mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and release\_cfc='4'; | 197344 | 18.7s | 18.65s |
| 5 | explain analyze select \* from r\_mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and imsi='460036271583050'; | 3725 | 15.7s | 15.6s |
| 6 | Sql 与 5 类似 |  |  |  |
| 7 |  |  |  |  |
| 8 | explain analyze select b.\* from r\_mod\_cdr\_hw b, r\_ne\_cell\_c c  where access\_cell=1038  and c.ne\_sys\_id = b.related\_cell; | 22229 | 17.17s | 17.18s |
| 9 | explain analyze select \* from r\_mod\_cdr\_hw b, r\_ne\_cell\_c c  where b.related\_bts =1080810030078000  and c.ne\_sys\_id = b.related\_cell; | 27001 | 17.56s | 16.87s |
| 10 | explain analyze select distinct access\_time from r\_mod\_do\_hw; | 1 | 38.55s | 35.3s |
| 11 | explain analyze select \* from r\_mod\_do\_hw where imsi='460036101389257'; | 120 | 33.5s | 33.45s |
| 12 | explain analyze select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 54770 | 40s | 37.7s |
| 13 | explain analyze select \* from r\_mod\_do\_hw\_lz where imsi='460036101389257'; | 120 | 25.6s | 25.5s |
| 14 | explain analyze select \* from r\_mod\_do\_hw\_lz b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 54770 | 30s | 28.8s |

###### 3.1.1.3.2 数据表有索引

**create index ind\_r\_mod\_cdr\_hw on r\_mod\_cdr\_hw(imsi);**

耗时：24.7s

所占空间：838MB

**create index ind\_r\_mod\_cdr\_hw\_accesscell on r\_mod\_cdr\_hw(access\_cell);**

耗时：17.3s

所占空间：597MB

**create index ind\_r\_mod\_cdr\_hw\_release\_cfc on r\_mod\_cdr\_hw(release\_cfc);**

耗时：20.4s

所占空间：597MB

**create index ind\_r\_mod\_do\_hw\_imsi on r\_mod\_do\_hw(imsi);**

耗时：51.4s

所占空间：2177MB

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sql 序号 | Sql 文本 | 返回结果集 | 首次查询耗时（no index） | 查询耗时(index) | 二次查询(index) |
| 1 | explain analyze select \* from r\_mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_cell=1039; | 28657 | 17.87s | 17s | 16.4s |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 | explain analyze select \* from r\_mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and release\_cfc='4'; | 197344 | 18.7s | 19s | 18.6s |
| 5 | explain analyze select \* from r\_mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and imsi='460036271583050'; | 3725 | 15.7s | 10.7s | 1.7s |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |
| 8 | explain analyze select b.\* from r\_mod\_cdr\_hw b, r\_ne\_cell\_c c  where access\_cell=1038  and c.ne\_sys\_id = b.related\_cell; | 22229 | 17.17s | 22.3s | 17.2s |
| 9 |  |  |  |  |  |
| 10 |  |  |  |  |  |
| 11 | explain analyze select \* from r\_mod\_do\_hw where imsi='460036101389257'; | 120 | 33.5s | 0.4s | 0.07s |
| 12 |  |  |  |  |  |
| 13 |  |  |  |  |  |
| 14 |  |  |  |  |  |

**以下是针对有索引字段查询时生成的执行计划**

uway=# explain analyze select \* from r\_mod\_cdr\_hw where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and access\_cell=1039;

QUERY PLAN

---------------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice1; segments: 32) (cost=0.00..1424419.48 rows=31 width=4420)

Rows out: 28657 rows at destination with 34 ms to first row, 16928 ms to end, start offset by 6.379 ms.

-> Append-only Scan on r\_mod\_cdr\_hw (cost=0.00..1424419.48 rows=31 width=4420)

Filter: access\_time::text > '2010-12-28'::text AND access\_time::text < '2010-12-29'::text AND access\_cell = 1039::numeric

Rows out: Avg 895.5 rows x 32 workers. Max 946 rows (seg15) with 75 ms to first row, 16229 ms to end, start offset by 211

1344 ms.

Slice statistics:

(slice0) Executor memory: 964K bytes.

(slice1) Executor memory: 1214K bytes avg x 32 workers, 1214K bytes max (seg0).

Statement statistics:

Memory used: 128000K bytes

Total runtime: 16941.112 ms

(11 rows)

Time: 16952.889 ms

uway=# explain analyze select \* from r\_mod\_cdr\_hw where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and release\_cfc='4';

QUERY PLAN ---------------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice1; segments: 32) (cost=0.00..1424419.48 rows=270 width=4420)

Rows out: 197344 rows at destination with 58 ms to first row, 19174 ms to end, start offset by 6.256 ms.

-> Append-only Scan on r\_mod\_cdr\_hw (cost=0.00..1424419.48 rows=270 width=4420)

Filter: access\_time::text > '2010-12-28'::text AND access\_time::text < '2010-12-29'::text AND release\_cfc = 4::numeric

Rows out: Avg 6167.0 rows x 32 workers. Max 6340 rows (seg28) with 78 ms to first row, 15840 ms to end, start offset by -

32230036278 ms.

Slice statistics:

(slice0) Executor memory: 964K bytes.

(slice1) Executor memory: 1214K bytes avg x 32 workers, 1214K bytes max (seg0).

Statement statistics:

Memory used: 128000K bytes

Total runtime: 19222.335 ms

(11 rows)

Time: 19227.216 ms

uway=# explain analyze select \* from r\_mod\_cdr\_hw where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and imsi='460036271583050';

QUERY PLAN

---------------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice1; segments: 32) (cost=128.28..16021.83 rows=1 width=4420)

Rows out: 3725 rows at destination with 195 ms to first row, 10714 ms to end, start offset by 6.140 ms.

-> Bitmap Append-Only Row-Oriented Scan on r\_mod\_cdr\_hw (cost=128.28..16021.83 rows=1 width=4420)

Recheck Cond: imsi = '460036271583050'::bpchar

Filter: access\_time::text > '2010-12-28'::text AND access\_time::text < '2010-12-29'::text

Rows out: Avg 116.4 rows x 32 workers. Max 149 rows (seg25) with 430 ms to first row, 4784 ms to end, start offset by -32

230036147 ms.

-> Bitmap Index Scan on ind\_r\_mod\_cdr\_hw (cost=0.00..128.27 rows=6 width=0)

Index Cond: imsi = '460036271583050'::bpchar

Bitmaps out: Avg 1.0 x 32 workers. Max 1 (seg0) with 35 ms to end, start offset by -7563 ms.

Work\_mem used: 56K bytes avg, 89K bytes max (seg0).

Slice statistics:

(slice0) Executor memory: 1003K bytes.

(slice1) Executor memory: 1928K bytes avg x 32 workers, 1928K bytes max (seg0). Work\_mem: 89K bytes max.

Statement statistics:

Memory used: 128000K bytes

Total runtime: 10721.569 ms

(16 rows)

Time: 10726.714 ms

uway=# explain analyze select b.\* from r\_mod\_cdr\_hw b, r\_ne\_cell\_c c where access\_cell=1038 and c.ne\_sys\_id = b.related\_cell;

QUERY PLAN ---------------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice2; segments: 32) (cost=1284532.87..1314248.15 rows=517 width=4420)

Rows out: 22229 rows at destination with 14969 ms to first row, 17523 ms to end, start offset by 4814 ms.

-> Hash Join (cost=1284532.87..1314248.15 rows=517 width=4420)

Hash Cond: c.ne\_sys\_id = b.related\_cell

Rows out: Avg 694.7 rows x 32 workers. Max 755 rows (seg25) with 14841 ms to first row, 15403 ms to end, start offset by

-32230031362 ms.

Executor memory: 3124K bytes avg, 3388K bytes max (seg25).

Work\_mem used: 3124K bytes avg, 3388K bytes max (seg25).

(seg25) Hash chain length 125.8 avg, 267 max, using 6 of 4111 buckets.

-> Broadcast Motion 32:32 (slice1; segments: 32) (cost=0.00..22982.72 rows=65558 width=12)

Rows out: Avg 65558.0 rows x 32 workers at destination. Max 65558 rows (seg0) with 0.064 ms to first row, 128 ms to

end, start offset by 12298 ms.

-> Seq Scan on r\_ne\_cell\_c c (cost=0.00..1348.58 rows=2049 width=12)

Rows out: Avg 2048.7 rows x 32 workers. Max 2056 rows (seg17) with 352 ms to first row, 353 ms to end, start

offset by 8921 ms.

-> Hash (cost=1284326.20..1284326.20 rows=517 width=4420)

Rows in: Avg 694.7 rows x 32 workers. Max 755 rows (seg25) with 14825 ms to end, start offset by -32230031362 ms.

-> Append-only Scan on r\_mod\_cdr\_hw b (cost=0.00..1284326.20 rows=517 width=4420)

Filter: access\_cell = 1038::numeric

Rows out: Avg 694.7 rows x 32 workers. Max 755 rows (seg25) with 67 ms to first row, 14821 ms to end, start o

ffset by -32230031362 ms.

Slice statistics:

(slice0) Executor memory: 2135K bytes.

(slice1) Executor memory: 1429K bytes avg x 32 workers, 1429K bytes max (seg0).

(slice2) Executor memory: 10231K bytes avg x 32 workers, 10231K bytes max (seg0). Work\_mem: 3388K bytes max.

Statement statistics:

Memory used: 128000K bytes

Total runtime: 22341.101 ms

(24 rows)

Time: 22347.867 ms

uway=# explain analyze select \* from r\_mod\_do\_hw where imsi='460036101389257';

QUERY PLAN

---------------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice1; segments: 32) (cost=387.82..223363.61 rows=73 width=2758)

Rows out: 120 rows at destination with 77 ms to first row, 383 ms to end, start offset by 3.469 ms.

-> Bitmap Append-Only Row-Oriented Scan on r\_mod\_do\_hw (cost=387.82..223363.61 rows=73 width=2758)

Recheck Cond: imsi = '460036101389257'::bpchar

Rows out: Avg 3.8 rows x 32 workers. Max 7 rows (seg4) with 221 ms to first row, 282 ms to end, start offset by -7640 ms.

-> Bitmap Index Scan on ind\_r\_mod\_do\_hw\_imsi (cost=0.00..387.24 rows=73 width=0)

Index Cond: imsi = '460036101389257'::bpchar

Bitmaps out: Avg 1.0 x 32 workers. Max 1 (seg0) with 151 ms to end, start offset by -7641 ms.

Work\_mem used: 28K bytes avg, 57K bytes max (seg4).

Slice statistics:

(slice0) Executor memory: 785K bytes.

(slice1) Executor memory: 1474K bytes avg x 32 workers, 1513K bytes max (seg0). Work\_mem: 57K bytes max.

Statement statistics:

Memory used: 128000K bytes

Total runtime: 392.537 ms

(15 rows)

Time: 399.117 ms

#### 3.1.2 列存储模式

##### 3.1.2.1数据入库测试

###### 3.1.2.1.1数据表无索引

下面是基准表（一倍容量）的情况：

|  |  |
| --- | --- |
| **mod\_cdr\_hw** | |
| 记录数 | 28,018,657 |
| 文本存储大小 | 48184548331Byte = 44.8GB |
| 压缩算法（x级压缩） | Zlib 5级 |
| 入库之后大小 | 7.61G |
| 压缩比 | 7.61/44.8 = 0.17 = 17% |
| 入库速度 | 507秒 90.5M/s |
| 收集统计信息时间（no index） | 252秒（采用率为默认值 10%） |
| 收集统计信息时间(index) | 260秒 |

|  |  |
| --- | --- |
| **mod\_do\_hw** | |
| 记录数 | 7302774\*10 = 73027740 |
| 文本存储大小 | 10389112752=9.7GB \* 10 = 97GB |
| 压缩算法（x级压缩） | Zlib 5级 |
| 入库之后大小 | 12G |
| 压缩比 | 12/97 = 0.124 = 12.4% |
| 入库速度 | 823秒 120.7M/s |
| 收集统计信息时间 | 204秒（采用率为默认值 10%） |
| 收集统计信息时间(index) | 216秒 |

###### 3.1.2.1.2数据表有索引

下面是基准表（一倍容量）的情况：

|  |  |
| --- | --- |
| **mod\_cdr\_hw** | |
| 记录数 | 28,018,657 |
| 文本存储大小 | 48184548331Byte = 44.8GB |
| 压缩算法（x级压缩） | Zlib 5级 |
| 索引个数 | 3 |
| 索引情况 | Imsi /access\_cell/release\_cfc 每个字段上一个btree index |
| 索引总大小 | 908 MB | ind\_mod\_cdr\_hw\_release\_cfc  786 MB | ind\_mod\_cdr\_hw\_accesscell  1083 MB | ind\_mod\_cdr\_hw |
| 入库之后表大小 | 7.85G |
| 压缩比 | 7.85/44.8 = 0.17 = 17% |
| 入库速度 | 578秒 (44.8+2.71)/578=84.2M/s |

|  |  |
| --- | --- |
| **mod\_do\_hw** | |
| 记录数 | 7302774\*10 = 73027740 |
| 文本存储大小 | 10389112752=9.7GB \* 10 = 97GB |
| 压缩算法（x级压缩） | Zlib 5级 |
| 索引个数 | 1 |
| 索引情况 | Imsi 上一个btree |
| 索引总大小 | 2897 MB | ind\_mod\_do\_hw\_imsi |
| 入库之后大小 | 12G |
| 压缩比 | 12/97 = 0.124 = 12.4% |
| 入库速度 | 885秒 (97+2.83)/885= 115.5M/s |

##### 3.1.2.2 系统性能指标

入库时的性能指标（录像/截屏）-----针对无索引的数据入库

###### 3.1.2.2.1 master host

table: mod\_cdr\_hw

/tmp/nmonreport/mdw-3122-cdr.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

table: mod\_do\_hw

/tmp/nmonreport/mdw-3122-do.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

###### 3.1.2.2.2 segment host ---sdw1

table: mod\_cdr\_hw

/tmp/sdw1-3122-cdr.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

table: mod\_do\_hw

/tmp/sdw1-3122-do.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

###### 3.1.2.2.3 segment host ---sdw2

table: mod\_cdr\_hw

/tmp/sdw2-3122-cdr.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

table: mod\_do\_hw

/tmp/sdw2-3122-do.txt

监控与上类似，在此省略。

###### 3.1.2.2.4 segment host ---sdw3

table: mod\_cdr\_hw

/tmp/sdw3-3122-cdr.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

table: mod\_do\_hw

/tmp/sdw3-3122-do.txt

监控与上类似，在此省略。

###### 3.1.2.2.5 segment host ---sdw4

table: mod\_cdr\_hw

/tmp/sdw4-3122-cdr.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

table: mod\_do\_hw

/tmp/sdw4-3122-do.txt

监控与上类似，在此省略。

##### 3.1.2.3 查询性能指标

###### 3.1.2.3.1 数据表无索引

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sql 序号 | Sql 文本 | 返回结果集 | 首次查询耗时 | 二次查询耗时 |
| 1 | explain analyze select \* from mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_cell=1039; | 28657 | 31.48s | 31.33s |
| 2 | Sql太长，详见sql 2 | 28657 | 5.984s | 6s |
| 3 | explain analyze select \* from mod\_cdr\_hw  where access\_cell=1039; | 28657 | 30.35s | 30s |
| 4 | explain analyze select \* from mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and release\_cfc='4'; | 197344 | 34.26s | 33.79s |
| 5 | explain analyze select \* from mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and imsi='460036271583050'; | 3725 | 31s | 31s |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 | explain analyze select b.\* from mod\_cdr\_hw b, r\_ne\_cell\_c c  where access\_cell=1038  and c.ne\_sys\_id = b.related\_cell; | 22229 | 33s | 33s |
| 9 | explain analyze select \* from mod\_cdr\_hw b, r\_ne\_cell\_c c  where b.related\_bts =1080810030078000  and c.ne\_sys\_id = b.related\_cell; | 27001 | 32s | 32s |
| 10 | explain analyze select distinct access\_time from mod\_do\_hw; | 1 | 4.7s | 4.6s |
| 11 | explain analyze select \* from mod\_do\_hw where imsi='460036101389257'; | 120 | 58.75s | 51.7s |
| 12 | explain analyze select \* from mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 54770 | 62s | 53s |
| **后增查询，为了测试列存是否符合线性关系** | | | | |
| 13 | select 100列 from mod\_cdr\_hw where imsi=’ 460036271583050’  sql详见测试数据 sql 13 | 3725 | 7.6s |  |
| 14 | 200列 sql14 | 3725 | 13.7s |  |
| 15 | 300列 sql15 | 3725 | 19.4s |  |
| 16 | 400列 sql16 | 3725 | 25.3s |  |
| 17 | 500列 sql17 | 3725 | 30s |  |
| 18 | 600列 sql18 | 3725 | 36.2s |  |
| 19 | 699列 sql19 | 3725 | 41.5s |  |

**以下是查询耗时最长的执行计划：**

12: Gather Motion 32:1 (slice3; segments: 32) (cost=529.46..6935218.25 rows=593 width=7673)

Rows out: 54770 rows at destination with 730 ms to first row, 52948 ms to end, start offset by 278 ms.

-> Hash Join (cost=529.46..6935218.25 rows=593 width=7673)

Hash Cond: b.related\_cell = c.ne\_sys\_id

Rows out: Avg 18256.7 rows x 3 workers. Max 24720 rows (seg20) with 681 ms to first row, 21538 ms to end, start offset by

4691 ms.

Executor memory: 92K bytes avg, 102K bytes max (seg10).

Work\_mem used: 92K bytes avg, 102K bytes max (seg10).

(seg10) Hash chain length 1.0 avg, 2 max, using 337 of 4111 buckets.

(seg20) Hash chain length 1.0 avg, 3 max, using 318 of 4111 buckets.

-> Redistribute Motion 32:32 (slice1; segments: 32) (cost=0.00..6934389.69 rows=777 width=2757)

Hash Key: b.related\_cell

Rows out: Avg 18256.7 rows x 3 workers at destination. Max 24720 rows (seg20) with 396 ms to first row, 20063 ms to

end, start offset by 4977 ms.

-> Append-only Columnar Scan on mod\_do\_hw b (cost=0.00..6933892.60 rows=777 width=2757)

Filter: related\_bts = 1080010000032000::numeric

Rows out: Avg 1711.6 rows x 32 workers. Max 1778 rows (seg3) with 679 ms to first row, 51865 ms to end, start

offset by -6521 ms.

-> Hash (cost=401.97..401.97 rows=319 width=4916)

Rows in: Avg 318.7 rows x 32 workers. Max 353 rows (seg10) with 140 ms to end, start offset by 1878796 ms.

-> Redistribute Motion 32:32 (slice2; segments: 32) (cost=0.00..401.97 rows=319 width=4916)

Hash Key: c.ne\_sys\_id

Rows out: Avg 318.7 rows x 32 workers at destination. Max 353 rows (seg10) with 0.071 ms to first row, 138 ms

to end, start offset by 1878796 ms.

-> Seq Scan on r\_ne\_cell\_c\_bj c (cost=0.00..197.99 rows=319 width=4916)

Rows out: Avg 318.7 rows x 32 workers. Max 324 rows (seg5) with 0.072 ms to first row, 0.150 ms to end,

start offset by -6519 ms.

Slice statistics:

(slice0) Executor memory: 1594K bytes.

(slice1) Executor memory: 61003K bytes avg x 32 workers, 61003K bytes max (seg7).

(slice2) Executor memory: 1583K bytes avg x 32 workers, 1583K bytes max (seg0).

(slice3) Executor memory: 1569K bytes avg x 32 workers, 1627K bytes max (seg0). Work\_mem: 102K bytes max.

Statement statistics:

Memory used: 128000K bytes

Total runtime: 53235.824 ms

(30 rows)

Time: 53240.630 ms

###### 3.1.2.3.2 数据表有索引

**create index ind\_mod\_cdr\_hw on mod\_cdr\_hw(imsi);**

耗时：47s

所占空间：838MB

**create index ind\_mod\_cdr\_hw\_accesscell on mod\_cdr\_hw(access\_cell);**

耗时：9s

所占空间：597MB

**create index ind\_mod\_cdr\_hw\_release\_cfc on mod\_cdr\_hw(release\_cfc);**

耗时：2s

所占空间：597MB

**create index ind\_mod\_do\_hw\_imsi on mod\_do\_hw(imsi);**

耗时：81.5s

所占空间：2177MB

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sql 序号 | Sql 文本 | 返回结果集 | 首次查询耗时(no index) | 查询耗时(index) | 二次查询(index) |
| 1 | explain analyze select \* from mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_cell=1039; | 28657 | 31.48s | 28s | 12s |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 | explain analyze select \* from mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and release\_cfc='4'; | 197344 | 34.26s | 43s | 34s |
| 5 | explain analyze select \* from mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and imsi='460036271583050'; | 3725 | 31s | 19s | 2.8s |
| 6 |  |  |  |  |  |
| 7 |  |  |  |  |  |
| 8 | explain analyze select b.\* from mod\_cdr\_hw b, r\_ne\_cell\_c c  where access\_cell=1038  and c.ne\_sys\_id = b.related\_cell; | 22229 | 33s | 30.8s | 13.8s |
| 9 |  |  |  |  |  |
| 10 |  |  |  |  |  |
| 11 | explain analyze select \* from mod\_do\_hw where imsi='460036101389257'; | 120 | 58.75s | 22.6s | 0.6s |
| 12 |  |  |  |  |  |

**以下是针对有索引字段查询时生成的执行计划：**

uway=# explain analyze select \* from mod\_cdr\_hw where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and access\_cell=1039;

QUERY PLAN

---------------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice1; segments: 32) (cost=1345.24..1535922.24 rows=31 width=4421)

Rows out: 28657 rows at destination with 228 ms to first row, 12018 ms to end, start offset by 99 ms.

-> Bitmap Append-Only Column-Oriented Scan on mod\_cdr\_hw (cost=1345.24..1535922.24 rows=31 width=4421)

Recheck Cond: access\_cell = 1039::numeric

Filter: access\_time::text > '2010-12-28'::text AND access\_time::text < '2010-12-29'::text

Rows out: Avg 895.5 rows x 32 workers. Max 960 rows (seg6) with 216 ms to first row, 11442 ms to end, start offset by -76

72 ms.

-> Bitmap Index Scan on ind\_mod\_cdr\_hw\_accesscell (cost=0.00..1344.99 rows=524 width=0)

Index Cond: access\_cell = 1039::numeric

Bitmaps out: Avg 1.0 x 32 workers. Max 1 (seg0) with 0.932 ms to end, start offset by -7659 ms.

Work\_mem used: 218K bytes avg, 218K bytes max (seg0).

Slice statistics:

(slice0) Executor memory: 1003K bytes.

(slice1) Executor memory: 96457K bytes avg x 32 workers, 96457K bytes max (seg0). Work\_mem: 218K bytes max.

Statement statistics:

Memory used: 128000K bytes

Total runtime: 12121.626 ms

(16 rows)

Time: 12126.295 ms

uway=# explain analyze select \* from mod\_cdr\_hw where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and release\_cfc='4';

QUERY PLAN ---------------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice1; segments: 32) (cost=0.00..2981525.48 rows=236 width=4421)

Rows out: 197344 rows at destination with 6987 ms to first row, 43425 ms to end, start offset by 96 ms.

-> Append-only Columnar Scan on mod\_cdr\_hw (cost=0.00..2981525.48 rows=236 width=4421)

Filter: access\_time::text > '2010-12-28'::text AND access\_time::text < '2010-12-29'::text AND release\_cfc = 4::numeric

Rows out: Avg 6167.0 rows x 32 workers. Max 6297 rows (seg29) with 11408 ms to first row, 42316 ms to end, start offset b

y -32230036097 ms.

Slice statistics:

(slice0) Executor memory: 964K bytes.

(slice1) Executor memory: 90623K bytes avg x 32 workers, 90623K bytes max (seg0).

Statement statistics:

Memory used: 128000K bytes

Total runtime: 43564.882 ms

(11 rows)

Time: 43569.843 ms

uway=# explain analyze select \* from mod\_cdr\_hw where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and imsi='460036271583050';

QUERY PLAN

---------------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice1; segments: 32) (cost=128.34..16992.27 rows=1 width=4421)

Rows out: 3725 rows at destination with 7013 ms to first row, 15510 ms to end, start offset by 3488 ms.

-> Bitmap Append-Only Column-Oriented Scan on mod\_cdr\_hw (cost=128.34..16992.27 rows=1 width=4421)

Recheck Cond: imsi = '460036271583050'::bpchar

Filter: access\_time::text > '2010-12-28'::text AND access\_time::text < '2010-12-29'::text

Rows out: Avg 116.4 rows x 32 workers. Max 141 rows (seg18) with 10864 ms to first row, 13703 ms to end, start offset by

7275 ms.

-> Bitmap Index Scan on ind\_mod\_cdr\_hw (cost=0.00..128.33 rows=6 width=0)

Index Cond: imsi = '460036271583050'::bpchar

Bitmaps out: Avg 1.0 x 32 workers. Max 1 (seg0) with 38 ms to end, start offset by -4195 ms.

Work\_mem used: 57K bytes avg, 81K bytes max (seg21).

Slice statistics:

(slice0) Executor memory: 1003K bytes.

(slice1) Executor memory: 96448K bytes avg x 32 workers, 96457K bytes max (seg0). Work\_mem: 81K bytes max.

Statement statistics:

Memory used: 128000K bytes

Total runtime: 18999.447 ms

(16 rows)

Time: 19004.289 ms

uway=# explain analyze select b.\* from mod\_cdr\_hw b, r\_ne\_cell\_c c where access\_cell=1038 and c.ne\_sys\_id = b.related\_cell;

QUERY PLAN ---------------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice2; segments: 32) (cost=1536051.82..1565786.65 rows=524 width=4421)

Rows out: 22229 rows at destination with 19118 ms to first row, 25332 ms to end, start offset by 5465 ms.

-> Hash Join (cost=1536051.82..1565786.65 rows=524 width=4421)

Hash Cond: c.ne\_sys\_id = b.related\_cell

Rows out: Avg 694.7 rows x 32 workers. Max 739 rows (seg8) with 19064 ms to first row, 23954 ms to end, start offset by 2

116851 ms.

Executor memory: 3124K bytes avg, 3339K bytes max (seg8).

Work\_mem used: 3124K bytes avg, 3339K bytes max (seg8).

(seg8) Hash chain length 123.2 avg, 275 max, using 6 of 2063 buckets.

-> Broadcast Motion 32:32 (slice1; segments: 32) (cost=0.00..22982.72 rows=65558 width=12)

Rows out: Avg 65558.0 rows x 32 workers at destination. Max 65558 rows (seg0) with 0.104 ms to first row, 1880 ms t

o end, start offset by 20249 ms.

-> Seq Scan on r\_ne\_cell\_c c (cost=0.00..1348.58 rows=2049 width=12)

Rows out: Avg 2048.7 rows x 32 workers. Max 2056 rows (seg17) with 38 ms to first row, 40 ms to end, start of

fset by 9297 ms.

-> Hash (cost=1535842.42..1535842.42 rows=524 width=4421)

Rows in: Avg 694.7 rows x 32 workers. Max 739 rows (seg8) with 19013 ms to end, start offset by 2116886 ms.

-> Bitmap Append-Only Column-Oriented Scan on mod\_cdr\_hw b (cost=1349.18..1535842.42 rows=524 width=4421)

Recheck Cond: access\_cell = 1038::numeric

Rows out: Avg 694.7 rows x 32 workers. Max 739 rows (seg8) with 7354 ms to first row, 18904 ms to end, start

offset by 2116886 ms.

-> Bitmap Index Scan on ind\_mod\_cdr\_hw\_accesscell (cost=0.00..1344.99 rows=524 width=0)

Index Cond: access\_cell = 1038::numeric

Bitmaps out: Avg 1.0 x 32 workers. Max 1 (seg0) with 15 ms to end, start offset by -2131 ms.

Work\_mem used: 218K bytes avg, 218K bytes max (seg0).

Slice statistics:

(slice0) Executor memory: 2173K bytes.

(slice1) Executor memory: 1429K bytes avg x 32 workers, 1429K bytes max (seg0).

(slice2) Executor memory: 104917K bytes avg x 32 workers, 104930K bytes max (seg3). Work\_mem: 3339K bytes max.

Statement statistics:

Memory used: 128000K bytes

Total runtime: 30802.399 ms

(28 rows)

Time: 30824.174 ms

uway=# explain analyze select \* from mod\_do\_hw where imsi='460036101389257';

QUERY PLAN

---------------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice1; segments: 32) (cost=387.83..227131.54 rows=73 width=2757)

Rows out: 120 rows at destination with 5782 ms to first row, 19279 ms to end, start offset by 3269 ms.

-> Bitmap Append-Only Column-Oriented Scan on mod\_do\_hw (cost=387.83..227131.54 rows=73 width=2757)

Recheck Cond: imsi = '460036101389257'::bpchar

Rows out: Avg 3.9 rows x 31 workers. Max 8 rows (seg4) with 8044 ms to first row, 18427 ms to end, start offset by -4377

ms.

-> Bitmap Index Scan on ind\_mod\_do\_hw\_imsi (cost=0.00..387.25 rows=73 width=0)

Index Cond: imsi = '460036101389257'::bpchar

Bitmaps out: Avg 1.0 x 32 workers. Max 1 (seg0) with 79 ms to end, start offset by -4367 ms.

Work\_mem used: 27K bytes avg, 57K bytes max (seg4).

Slice statistics:

(slice0) Executor memory: 785K bytes.

(slice1) Executor memory: 61909K bytes avg x 32 workers, 62461K bytes max (seg3). Work\_mem: 57K bytes max.

Statement statistics:

Memory used: 128000K bytes

Total runtime: 22548.456 ms

(15 rows)

Time: 22556.316 ms

### 3.2并行测试（没有数据入库，只有并行查询）

#### 3.2.1 行存储模式

##### 3.2.1.1 模拟并发5用户

###### 3.2.1.1.1 数据表无索引

* sql-1 情况：

|  |  |  |  |
| --- | --- | --- | --- |
| Sql | 所有session执行相同sql | 查询耗时 | 原串行耗时 |
| Session 1 | explain analyze select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 172s | 40s |
| Session 2 | ….. | 173s |  |
| Session 3 | ….. | 172s |  |
| Session 4 | …… | 169s |  |
| Session 5 | ….. | 169s |  |
| **合计（累计）** |  | **173s** | **200s** |

* sql-2 情况：

|  |  |  |  |
| --- | --- | --- | --- |
| Sql | 所有session执行不同sql | 查询耗时 | 原串行耗时 |
| Session 1 | explain analyze select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 128s | 40s |
| Session 2 | explain analyze select \* from r\_mod\_cdr\_hw b, r\_ne\_cell\_c c  where b.related\_bts =1080810030078000  and c.ne\_sys\_id = b.related\_cell; | 99s | 17.56s |
| Session 3 | explain analyze select \* from r\_mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and release\_cfc='4'; | 94s | 18.7s |
| Session 4 | explain analyze select \* from r\_mod\_do\_hw\_lz b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 122s | 30s |
| Session 5 | explain analyze select \* from r\_mod\_do\_hw where imsi='460036101389257'; | 125s | 33.5s |
| **合计（累计）** |  | **128s** | **140s** |

###### 3.2.1.1.2 数据表有索引

**ind\_r\_mod\_cdr\_hw on r\_mod\_cdr\_hw(imsi);** 所占空间：838MB

**ind\_r\_mod\_cdr\_hw\_accesscell on r\_mod\_cdr\_hw(access\_cell);** 所占空间：597MB

**ind\_r\_mod\_cdr\_hw\_release\_cfc on r\_mod\_cdr\_hw(release\_cfc);** 所占空间：597MB

**ind\_r\_mod\_do\_hw\_imsi on r\_mod\_do\_hw(imsi);** 所占空间：2177MB

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sql | 所有session执行不同sql | 查询耗时（no index） | 原串行耗时(no index) | 原串行耗时(index) | 查询耗时(index) |
| Session 1 | explain analyze select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 128s | 40s | / | 107.6s |
| Session 2 | explain analyze select \* from r\_mod\_cdr\_hw b, r\_ne\_cell\_c c  where b.related\_bts =1080810030078000  and c.ne\_sys\_id = b.related\_cell; | 99s | 17.56s | / | 105s |
| Session 3 | explain analyze select \* from r\_mod\_cdr\_hw  where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss')  and release\_cfc='4'; | 94s | 18.7s | 19s | 105s |
| Session 4 | explain analyze select \* from r\_mod\_do\_hw\_lz b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 122s | 30s | / | 106s |
| Session 5 | explain analyze select \* from r\_mod\_do\_hw where imsi='460036101389257'; | 125s | 33.5s | 22.6s | 1s |
| **合计（累计）** |  | **128s** | **140s** |  | **107.6s** |

**Session 3执行的sql:**

QUERY PLAN -------------------------------------------------------------------------------------------------------------------------- Gather Motion 32:1 (slice1; segments: 32) (cost=0.00..1424419.48 rows=270 width=4420)

Rows out: 197344 rows at destination with 0.038 ms to first row, 104425 ms to end, start offset by 690 ms.

-> Append-only Scan on r\_mod\_cdr\_hw (cost=0.00..1424419.48 rows=270 width=4420)

Filter: access\_time::text > '2010-12-28'::text AND access\_time::text < '2010-12-29'::text AND release\_cfc = 4::numeric

Rows out: Avg 6167.0 rows x 32 workers. Max 6340 rows (seg28) with 4.797 ms to first row, 53403 ms to end, start offset by -32230035716 ms.

Slice statistics:

(slice0) Executor memory: 964K bytes.

(slice1) Executor memory: 1214K bytes avg x 32 workers, 1214K bytes max (seg0).

Statement statistics:

Memory used: 128000K bytes

Total runtime: 105161.289 ms

(11 rows)

**Session 5执行的sql:**

QUERY PLAN

------------------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice1; segments: 32) (cost=387.82..223363.61 rows=73 width=2758)

Rows out: 120 rows at destination with 6.923 ms to first row, 383 ms to end, start offset by 654 ms.

-> Bitmap Append-Only Row-Oriented Scan on r\_mod\_do\_hw (cost=387.82..223363.61 rows=73 width=2758)

Recheck Cond: imsi = '460036101389257'::bpchar

Rows out: Avg 3.8 rows x 32 workers. Max 7 rows (seg4) with 41 ms to first row, 105 ms to end, start offset by -6914 ms.

-> Bitmap Index Scan on ind\_r\_mod\_do\_hw\_imsi (cost=0.00..387.24 rows=73 width=0)

Index Cond: imsi = '460036101389257'::bpchar

Bitmaps out: Avg 1.0 x 32 workers. Max 1 (seg0) with 0.269 ms to end, start offset by -6916 ms.

Work\_mem used: 28K bytes avg, 57K bytes max (seg4).

Slice statistics:

(slice0) Executor memory: 785K bytes.

(slice1) Executor memory: 1475K bytes avg x 32 workers, 1513K bytes max (seg0). Work\_mem: 57K bytes max.

Statement statistics:

Memory used: 128000K bytes

Total runtime: 1044.431 ms

(15 rows)

###### 3.2.1.1.3 系统性能指标

性能指标（录像/截屏）-----针对无索引时的并行查询

3.2.1.1.3.1 master host

/tmp/nmonreport/mdw-3211-sql-1.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.2.1.1.3.2 segment host---sdw1

/tmp/sdw1-3211-sql-1.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.2.1.1.3.3 segment host---sdw2

/tmp/sdw2-3211-sql-1.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.2.1.1.3.4 segment host---sdw3

/tmp/sdw3-3211-sql-1.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.2.1.1.3.5 segment host---sdw4

/tmp/sdw4-3211-sql-1.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

##### 3.2.1.2 模拟并发10用户

###### 3.2.1.2.1数据表无索引

* sql-1 情况：

|  |  |  |  |
| --- | --- | --- | --- |
| Sql | 所有session执行相同sql | 查询耗时 | 原串行耗时 |
| Session 1 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 337.5s | 40s |
| Session 2 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000187100 and c.ne\_sys\_id = b.related\_cell; | 336.7s | 40s |
| Session 3 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000228100 and c.ne\_sys\_id = b.related\_cell; | 334s | 40s |
| Session 4 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000229200 and c.ne\_sys\_id = b.related\_cell; | 334s | 40s |
| Session 5 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000269100 and c.ne\_sys\_id = b.related\_cell; | 334s | 40s |
| Session 6 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000306100 and c.ne\_sys\_id = b.related\_cell; | 335s | 40s |
| Session 7 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010032028100 and c.ne\_sys\_id = b.related\_cell; | 330s | 40s |
| Session 8 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010032029200 and c.ne\_sys\_id = b.related\_cell; | 331s | 40s |
| Session 9 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010032034200 and c.ne\_sys\_id = b.related\_cell; | 332s | 40s |
| Session 10 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010032111100 and c.ne\_sys\_id = b.related\_cell; | 331s | 40s |
| **合计（累计）** |  | **337.5s** | **400s** |

* sql-2 情况(根据测试情况来看，此部分已略掉，结果可参考并发5用户)：

|  |  |  |  |
| --- | --- | --- | --- |
| Sql | 所有session执行不同sql | 查询耗时 | 原串行耗时 |
| Session 1 | explain analyze select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; |  | 40s |
| Session 2 | explain analyze select \* from r\_mod\_cdr\_hw b, r\_ne\_cell\_c c where b.related\_bts =1080810030078000 and c.ne\_sys\_id = b.related\_cell; |  | 17.56s |
| Session 3 | explain analyze select \* from r\_mod\_cdr\_hw where access\_time > To\_Date('2010-12-28 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and access\_time < To\_Date('2010-12-29 00:00:00', 'yyyy-mm-dd hh24:mi:ss') and release\_cfc='4'; |  | 18.7s |
| Session 4 | explain analyze select \* from r\_mod\_do\_hw\_lz b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; |  | 30s |
| Session 5 | explain analyze select \* from r\_mod\_do\_hw where imsi='460036101389257'; |  | 33.5s |
| Session 6 |  |  |  |
| Session 7 |  |  |  |
| Session 8 |  |  |  |
| Session 9 |  |  |  |
| Session 10 |  |  |  |

###### 3.2.1.2.2数据表有索引---未测试

可参考 3.2.1.1.2节 模拟并发5用户—数据表有索引部分的测试结果。

###### 3.2.1.2.3 系统性能指标

性能指标（录像/截屏）-----针对无索引时的并行查询

3.2.1.2.3.1 master host

/tmp/nmonreport/mdw-3212-sql-1.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.2.1.2.3.2 segment host---sdw1

/tmp/sdw1-3212-sql-1.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.2.1.2.3.3 segment host---sdw2

/tmp/sdw2-3212-sql-1.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.2.1.2.3.4 segment host---sdw3

/tmp/sdw3-3212-sql-1.txt

**CPU**

**DISK**

可参考sdw2的图例，与之类似。

**MEMORY**

**NETWORK**

可参考sdw2的图例，与之类似。

3.2.1.2.3.5 segment host---sdw4

/tmp/sdw4-3212-sql-1.txt

**CPU**

**DISK**

可参考sdw2的图例，与之类似。

**MEMORY**

**NETWORK**

可参考sdw2的图例，与之类似。

#### 3.2.2 列存储模式

##### 3.2.2.1 模拟并发5用户

###### 3.2.2.1.1 数据表无索引

* sql-1 情况：

|  |  |  |  |
| --- | --- | --- | --- |
| Sql | Sql 文本 | 查询耗时 | 原串行耗时 |
| Session 1 | explain analyze select distinct access\_time from mod\_do\_hw; | 18s | 4.7s |
| Session 2 | explain analyze select distinct ne\_sys\_id from mod\_do\_hw; | 49s | 4.7s |
| Session 3 | explain analyze select distinct related\_bsc from mod\_do\_hw; | 40s | 4.7s |
| Session 4 | explain analyze select distinct release\_cfc from mod\_do\_hw; | 16 | 4.7s |
| Session 5 | explain analyze select distinct call\_ref\_num from mod\_do\_hw; | 77s | 4.7s |
| **合计（累计）** |  | **77s** | **4.7\*5=23.5s** |

* sql-2 情况：

|  |  |  |  |
| --- | --- | --- | --- |
| Sql | Sql 文本 | 查询耗时 | 原串行耗时 |
| Session 1 | explain analyze select b.\* from mod\_cdr\_hw b, r\_ne\_cell\_c c where access\_cell=1038 and c.ne\_sys\_id = b.related\_cell; | 159s | 33s |
| Session 2 | explain analyze select \* from mod\_cdr\_hw b, r\_ne\_cell\_c c where b.related\_bts =1080810030078000 and c.ne\_sys\_id = b.related\_cell; | 160s | 32s |
| Session 3 | explain analyze select distinct access\_time from mod\_do\_hw; | 6s | 4.7s |
| Session 4 | explain analyze select \* from mod\_do\_hw where imsi='460036101389257'; | 178s | 58.75s |
| Session 5 | explain analyze select \* from mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 177s | 62s |
| **合计（累计）** |  | **178s** | **190.45s** |

###### 3.2.2.1.2 数据表有索引

**ind\_mod\_cdr\_hw on mod\_cdr\_hw(imsi);** 所占空间：838MB

**ind\_mod\_cdr\_hw\_accesscell on mod\_cdr\_hw(access\_cell);** 所占空间：597MB

**ind\_mod\_cdr\_hw\_release\_cfc on mod\_cdr\_hw(release\_cfc);** 所占空间：597MB

**ind\_mod\_do\_hw\_imsi on mod\_do\_hw(imsi);** 所占空间：2177MB

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sql | Sql 文本 | 查询耗时(no index) | 原串行耗时(no index) | 原串行耗时(index) | 查询耗时(index) |
| Session 1 | explain analyze select b.\* from mod\_cdr\_hw b, r\_ne\_cell\_c c where access\_cell=1038 and c.ne\_sys\_id = b.related\_cell; | 159s | 33s | 30.8s | 68s |
| Session 2 | explain analyze select \* from mod\_cdr\_hw b, r\_ne\_cell\_c c where b.related\_bts =1080810030078000 and c.ne\_sys\_id = b.related\_cell; | 160s | 32s | / | 103s |
| Session 3 | explain analyze select distinct access\_time from mod\_do\_hw; | 6s | 4.7s | / | 8.6s |
| Session 4 | explain analyze select \* from mod\_do\_hw where imsi='460036101389257'; | 178s | 58.75s | 22.6s | 26.6s |
| Session 5 | explain analyze select \* from mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 177s | 62s | / | 110s |
| **合计（累计）** |  | **178s** | **190.45s** |  | **110s** |

**Session1执行的sql:**

QUERY PLAN

--------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice2; segments: 32) (cost=1536051.82..1565786.65 rows=524 width=4421)

Rows out: 22229 rows at destination with 37454 ms to first row, 63142 ms to end, start offset by 4905 ms.

-> Hash Join (cost=1536051.82..1565786.65 rows=524 width=4421)

Hash Cond: c.ne\_sys\_id = b.related\_cell

Rows out: Avg 694.7 rows x 32 workers. Max 739 rows (seg8) with 41787 ms to first row, 61068 ms to end, start offset by 2116684 ms.

Executor memory: 3124K bytes avg, 3339K bytes max (seg8).

Work\_mem used: 3124K bytes avg, 3339K bytes max (seg8).

(seg8) Hash chain length 123.2 avg, 275 max, using 6 of 2063 buckets.

-> Broadcast Motion 32:32 (slice1; segments: 32) (cost=0.00..22982.72 rows=65558 width=12)

Rows out: Avg 65558.0 rows x 32 workers at destination. Max 65558 rows (seg0) with 0.100 ms to first row, 11041 ms to end, start offset by 47458 ms.

-> Seq Scan on r\_ne\_cell\_c c (cost=0.00..1348.58 rows=2049 width=12)

Rows out: Avg 2048.7 rows x 32 workers. Max 2056 rows (seg17) with 0.039 ms to first row, 0.903 ms to end, start offset by 9736 ms.

-> Hash (cost=1535842.42..1535842.42 rows=524 width=4421)

Rows in: Avg 694.7 rows x 32 workers. Max 739 rows (seg8) with 41751 ms to end, start offset by 2116684 ms.

-> Bitmap Append-Only Column-Oriented Scan on mod\_cdr\_hw b (cost=1349.18..1535842.42 rows=524 width=4421)

Recheck Cond: access\_cell = 1038::numeric

Rows out: Avg 694.7 rows x 32 workers. Max 739 rows (seg8) with 12136 ms to first row, 41469 ms to end, start offset by 2116684 ms.

-> Bitmap Index Scan on ind\_mod\_cdr\_hw\_accesscell (cost=0.00..1344.99 rows=524 width=0)

Index Cond: access\_cell = 1038::numeric

Bitmaps out: Avg 1.0 x 32 workers. Max 1 (seg0) with 0.541 ms to end, start offset by -2120 ms.

Work\_mem used: 218K bytes avg, 218K bytes max (seg0).

Slice statistics:

(slice0) Executor memory: 2173K bytes.

(slice1) Executor memory: 1429K bytes avg x 32 workers, 1429K bytes max (seg0).

(slice2) Executor memory: 104917K bytes avg x 32 workers, 104930K bytes max (seg3). Work\_mem: 3339K bytes max.

Statement statistics:

Memory used: 128000K bytes

Total runtime: 68051.518 ms

(28 rows)

**Session4执行的sql:**

QUERY PLAN

--------------------------------------------------------------------------------------------------------------------------

Gather Motion 32:1 (slice1; segments: 32) (cost=387.83..227131.54 rows=73 width=2757)

Rows out: 120 rows at destination with 10520 ms to first row, 24830 ms to end, start offset by 1764 ms.

-> Bitmap Append-Only Column-Oriented Scan on mod\_do\_hw (cost=387.83..227131.54 rows=73 width=2757)

Recheck Cond: imsi = '460036101389257'::bpchar

Rows out: Avg 3.9 rows x 31 workers. Max 8 rows (seg4) with 11110 ms to first row, 21384 ms to end, start offset by -5768 ms.

-> Bitmap Index Scan on ind\_mod\_do\_hw\_imsi (cost=0.00..387.25 rows=73 width=0)

Index Cond: imsi = '460036101389257'::bpchar

Bitmaps out: Avg 1.0 x 32 workers. Max 1 (seg0) with 0.646 ms to end, start offset by -5640 ms.

Work\_mem used: 27K bytes avg, 57K bytes max (seg4).

Slice statistics:

(slice0) Executor memory: 785K bytes.

(slice1) Executor memory: 61909K bytes avg x 32 workers, 62461K bytes max (seg3). Work\_mem: 57K bytes max.

Statement statistics:

Memory used: 128000K bytes

Total runtime: 26595.714 ms

(15 rows)

##### 3.2.2.2 模拟并发10用户---未测试

此部分未测试，可参考 3.2.2.1节 列存储模式---模拟并发5用户 部分。

### 3.3并行测试（数据入库的同时，并行查询）

#### 3.3.1 行存储模式

##### 3.3.1.1数据入库测试

**入库表信息（同时入库以下两张表）：**

|  |  |
| --- | --- |
| **mod\_cdr\_hw** | |
| 记录数 | 28,018,657 |
| 文本存储大小 | 48184548331Byte = 44.8GB |
| 压缩算法（x级压缩） | Zlib 5级 |
| 入库之后大小 | 29G |
| 压缩比 | 29/44.8=0.647 = 64.7% |
| 入库速度 | 1420秒 32.3M/s |

|  |  |
| --- | --- |
| **mod\_do\_hw** | |
| 记录数 | 7302774\*10 = 73027740 |
| 文本存储大小 | 10389112752=9.7GB \* 10 = 97GB |
| 压缩算法（x级压缩） | Zlib 5级 |
| 入库之后大小 | 58G |
| 压缩比 | 58/97=0.598 = 59.8% |
| 入库速度 | 1784秒 55.7M/s |

##### 3.3.1.2 模拟并发5用户---未测试

此部分可参考 3.3.2.2节--列存储模式下的并发5用户测试部分。

##### 3.3.1.3 模拟并发10用户

###### 3.3.1.3.1 数据表无索引

* sql-1 情况：

|  |  |  |  |
| --- | --- | --- | --- |
| Sql | 所有session执行相同sql | 查询耗时 | 原串行耗时 |
| Session 1 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 381.2s | 40s |
| Session 2 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000187100 and c.ne\_sys\_id = b.related\_cell; | 383.7s |  |
| Session 3 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000228100 and c.ne\_sys\_id = b.related\_cell; | 377s |  |
| Session 4 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000229200 and c.ne\_sys\_id = b.related\_cell; | 380s |  |
| Session 5 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000269100 and c.ne\_sys\_id = b.related\_cell; | 385s |  |
| Session 6 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000306100 and c.ne\_sys\_id = b.related\_cell; | 382s |  |
| Session 7 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010032028100 and c.ne\_sys\_id = b.related\_cell; | 379s |  |
| Session 8 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010032029200 and c.ne\_sys\_id = b.related\_cell; | 378s |  |
| Session 9 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010032034200 and c.ne\_sys\_id = b.related\_cell; | 383s |  |
| Session 10 | select \* from r\_mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010032111100 and c.ne\_sys\_id = b.related\_cell; | 382s |  |
| **合计（累计）** |  | **385s** | **40s\*10=400s** |

###### 3.3.1.3.2 系统性能指标

性能指标（录像/截屏）-----针对无索引时的并行查询，与此同时入库正在同步进行。

3.3.1.3.2.1 master host

/tmp/nmonreport/mdw-3313-sql-2.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.3.1.3.2.2 master host---sdw1

/tmp/sdw1-3313-sql-2.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.3.1.3.2.3 master host---sdw2

/tmp/sdw2-3313-sql-2.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.3.1.3.2.4 master host---sdw3

/tmp/sdw3-3313-sql-2.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.3.1.3.2.5 master host---sdw4

/tmp/sdw4-3313-sql-2.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

#### 3.3.2 列存储模式

##### 3.3.2.1数据入库测试

**入库表信息（同时入库以下两张表）：**

|  |  |
| --- | --- |
| **mod\_cdr\_hw** | |
| 记录数 | 28,018,657 |
| 文本存储大小 | 48184548331Byte = 44.8GB |
| 压缩算法（x级压缩） | Zlib 5级 |
| 入库之后大小 | 7.61G |
| 压缩比 | 7.61/44.8 = 0.17 = 17% |
| 入库速度 | 1220秒 37.6M/s |

|  |  |
| --- | --- |
| **mod\_do\_hw** | |
| 记录数 | 7302774\*10 = 73027740 |
| 文本存储大小 | 10389112752=9.7GB \* 10 = 97GB |
| 压缩算法（x级压缩） | Zlib 5级 |
| 入库之后大小 | 12G |
| 压缩比 | 12/97 = 0.124 = 12.4% |
| 入库速度 | 1324 秒 75M/s |

##### 3.3.2.2 模拟并发5用户

###### 3.3.2.2.1 数据表无索引

|  |  |  |  |
| --- | --- | --- | --- |
| Sql | Sql 文本 | 查询耗时 | 原串行耗时 |
| Session 1 | explain analyze select b.\* from mod\_cdr\_hw b, r\_ne\_cell\_c c where access\_cell=1038 and c.ne\_sys\_id = b.related\_cell; | 197s | 33s |
| Session 2 | explain analyze select \* from mod\_cdr\_hw b, r\_ne\_cell\_c c where b.related\_bts =1080810030078000 and c.ne\_sys\_id = b.related\_cell; | 216s | 32s |
| Session 3 | explain analyze select distinct access\_time from mod\_do\_hw; | 21 | 4.7s |
| Session 4 | explain analyze select \* from mod\_do\_hw where imsi='460036101389257'; | 273s | 58.75s |
| Session 5 | explain analyze select \* from mod\_do\_hw b, r\_ne\_cell\_c\_bj c where b.related\_bts =1080010000032000 and c.ne\_sys\_id = b.related\_cell; | 273s | 62s |
| **合计（累计）** |  | **273s** | **190.45s** |

###### 3.3.2.2.2 系统性能指标

性能指标（录像/截屏）-----针对无索引时的并行查询，与此同时入库正在同步进行。

3.3.2.2.2.1 master host

/tmp/nmonreport/mdw-3322-sql-1.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.3.2.2.2.2 segment host---sdw1

/tmp/sdw1-3322-sql-1.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.3.2.2.2.3 segment host---sdw2

/tmp/sdw2-3322-sql-1.txt

**CPU**

**DISK**

**MEMORY**

**NETWORK**

3.3.2.2.2.4 segment host---sdw3

/tmp/sdw3-3322-sql-1.txt

监控图同上，在此省略。

3.3.2.2.2.5 segment host---sdw4

/tmp/sdw4-3322-sql-1.txt

监控图同上，在此省略。

##### 3.3.2.3 模拟并发10用户---未测试

此部分可参考 3.3.2.2节模拟并发5用户。

### 3.4分区轮换测试

Gp不支持job功能，只能自动配置系统任务。

### 3.5 segment instance 无mirror且宕instance测试

Gp中一旦出现有segment无效，则整个db将不可用。

报错信息如下：

ERROR: All segment databases are unavailable

\*\*\*\*\*\*\*\*\*\* 错误 \*\*\*\*\*\*\*\*\*\*

ERROR: All segment databases are unavailable

SQL 状态: 58M01

### 3.6主备master切换测试

**正常状态如下：**

Primary master : mdw

Standby mater : smdw

入库数据： ne\_cell\_c 65558条记录 3964k

Mod\_cdr\_hw(release\_cfc=’4’) 197344条记录 55M

**切换测试方案：**

Kill 掉mdw的postgre进程，导致gp down掉，之后用smdw启动gp

启动语法：

**Gpactivatestandby –d** *standby\_master\_datadir* **-f**

Gp启动之后，做如下查询验证：

ne\_cell\_c 65558条记录 3964k

Mod\_cdr\_hw(release\_cfc=’4’) 197344条记录 55M

Select \* from ne\_cell\_c;

数据没有发现问题。

### 3.7 segment instance有mirror宕instance测试

#### 3.7.1 测试前环境

Segment host : 3 台

Mater host : 1台

每个segment host有４个primary gpseg

4个 mirror gpseg

其中映射关系如下(轮回)：

　Sdw1 镜像 sdw4 的 4个gpseg

Sdw3 镜像 sdw1 的 4个gpseg

Sdw4 镜像 sdw3 的 4个gpseg

GP处于正常可用状态。

#### 3.7.2 测试1

测试方法：将交换机上的网线拔掉1根，此网线连接的是sdw4

**测试结果：GP仍可用，数据没有丢失。**

测试方法：再次拔掉1跟网线

**测试结果：GP不再可用，已经宕掉。**

#### 3.7.3 测试2

测试方法：将sdw4上的postgre进程全部kill掉

**测试结果：GP仍可用，数据没有丢失，此时原mirror的segment instance的role由mirror转变为primary。**

#### 3.7.4 测试3

测试方法：将sdw4机器上的eth0网卡down掉

**测试结果：GP仍可用，数据没有丢失，此时该网卡上fail的primary instance 由其mirror的 instance接管。**

测试方法：再次down掉sdw4上的eth1网卡

**测试结果：GP仍可用，数据没有丢失，此时该网卡上再次fail的primary instance 由其mirror的 instance接管。**

## 4、结论

1、**存储方式的选择**：默认情况下（采用zlib 5级压缩），行存储模式入库之后所占空间是文本时的60%，即节省了40%的空间，而列存储模式入库之后所占空间是文本时的15%，即节省了85%的空间。

2、**数据入库速度**：由于行存储的压缩效果较列存储要差很多，这也就导致了在使用行存储时CPU的消耗要明显小于列存储时对CPU的消耗，直接影响是行存储入库时间大于列存储入库时间。

3、**不同的存储方式对创建索引的影响**：在行存储和列存储上分别创建相同的单列btree索引，行存储上不同索引创建消耗的时间相差不大，而列存储上相差很明显，选择性强的消耗的时间长，选择性弱的消耗的时间短，这也正好体现了行列存储的本质区别。

4、**索引对数据入库的影响：**在行存储数据表上创建3个索引，有无索引对数据入库影响1/7的时间（数据表：44.8G，索引：2.71G），而在列存储数据表上创建3个索引，有无索引对数据入库影响1/7的时间（数据表：44.8G，索引：2.71G），此时没有什么区别。

5、**存储方式对查询的影响：**在行存储模式下，当全表扫描时，不同的查询列数查询耗用的时间没区别，这也体现了行存储的特性，而在**列存储模式**下，正好相反，不同的查询列数查询耗用的时间是成正比的，查询的列数越多，所消耗的时间越长，时间也成**线性关系**；还有一点，相同的查询语句（都是全列查询），行存储所需要的时间是列存储下耗用时间的1/2。

6、**索引对查询的影响：**将具有索引的列作为查询条件时，执行计划显示查询时并不一定每次都利用该索引去查询，如果没有利用索引查询，那么该查询耗时与之前没有区别，而一旦利用了索引，那么该查询将较之前的查询快几倍甚至几十倍。

7、**并行查询的影响**：并行查询时，无论数据是按照行存储还是按照列存储，并行查询的规律是一致的，即小于并行查询语句在串行方式下查询耗时之和，当较串行执行耗时节省的时间有限，如，串行耗时之和为140s，并行需要128s，串行耗时之和为400s，并行需要337s。

8、**资源优先分析：**由于GP使用资源的方式是尽可能发挥资源的最大优势，所以在发生资源瓶颈的时候，必然会出现资源竞争的情况，而当先执行数据入库之后，再执行并发的N个查询，此时资源由入库占有转为查询占有，也就是说**查询**具有资源利用的**高优先级**。

9、**资源使用分析：**此次测试采用了1个master host和4个segment host，如此一来，在资源耗用的时候，4个segment host所消耗的资源几乎是一致的，即负载均衡性特点非常明显。

10、**资源瓶颈分析：**由于数据入库时都是压缩的，导致cpu利用较高，经常100%，而磁盘和网卡利用率较低，在资源最繁忙的时候，磁盘的i/o和网卡的最高指标都较GP安装时候的测试具有较大差距，应该是数据在内存中达到一定数据量之后，才有入库的需求，而内存也需要大量的消耗，所以瓶颈主要体现在CPU和内存上。

11、**分区轮换影响：**由于GP不支持job功能，所以该部分功能只能利用操作系统来配置jobs，从而增加了操作系统方面的配置。

12、**segment的一个instance宕掉对GP系统的影响：**在无mirror的情况下，如果发生了一个segment的instance宕掉，将导致整个GP不可用，而在有mirror的时候，则可以避免GP的不可用，但缺点就是需要消耗一半的资源来做mirror，从而使有效的可用资源下降一倍。

最后，说明一下本次测试的数据量：44.8G\*8+97G\*8=1134.4G=1.108TB。