阿里数仓POC评测报告

V0.0.2

**最近修订记录：**

|  |  |  |  |
| --- | --- | --- | --- |
| 修订日期 | 版本 | 创建人 | 修订记录 |
| 20170918 | v0.0.1 | 彭仲颐 | 创建文档 |
| 20170927 | V0.0.2 | 彭仲颐 | 添加拓展阿里数仓节点的评测结果 |
|  |  |  |  |
|  |  |  |  |

目录

[1.硬件性能报告 4](#_Toc494272689)

[1.1 硬件环境信息 4](#_Toc494272690)

[1.2 内网环境测试 4](#_Toc494272691)

[1.3磁盘性能测试 4](#_Toc494272692)

[1.4 内存性能测试 4](#_Toc494272693)

[2. SQL测试场景 5](#_Toc494272694)

[2.1 多种导入方式比较 5](#_Toc494272695)

[2.2 一般SQL测试语句与实际消耗 5](#_Toc494272696)

[2.3测试环境表现 6](#_Toc494272697)

[2.4 巨型SQL测试语句与实际消耗 6](#_Toc494272698)

[2.5 测试环境表现 7](#_Toc494272699)

[3.用户习惯 7](#_Toc494272700)

[3.1.函数或功能变更(Oracle至PostgreSQL) 7](#_Toc494272701)

[4.生产环境建议配置 9](#_Toc494272702)

[4.1.硬件选型标准 9](#_Toc494272703)

[4.GreenPlum使用手册 9](#_Toc494272704)

[4.1 目录 9](#_Toc494272705)

## 1.硬件环境信息

### 1.1 硬件性能信息

|  |  |  |  |
| --- | --- | --- | --- |
| 测试对象 | Vertica数仓  (测试环境) | 阿里数仓  (云上) | 阿里数仓  (云上升级版) |
| 节点数 | 2 | 2 | **4** |
| 合计逻辑CPU | **12core** | 2core | 8core |
| 合计内存 | 24G | 16G | **64G** |
| 磁盘 | SAS 转速10k | SSD | **SSD** |

### 1.2 内网性能测试

|  |  |  |
| --- | --- | --- |
| 节点发起方 | 节点接收方 | 通信速度 |
| V001 | V002 | 10.97MB/S |
| 平均值:10.68MB/S | 中位值:10.96MB/S |  |

阿里云不提供对外的数据测试结果

### 1.3 磁盘性能测试

|  |  |
| --- | --- |
| 磁盘写入测试 | |
| 写入量/写入时间 | 1513.46(GB)/573.98(S) |
| V001写入均速 | 922.95MB/S |
| V002写入均速 | 897.86MB/S |
| 磁盘读取测试 | |
| 读取量/读取时间 | 1513.46(GB)/421.22(S) |
| V001读取均速 | 1212.24MB/S |
| V002读取均速 | 1213.78MB/S |

阿里云不提供对外的数据测试结果

### 1.4 内存性能测试

|  |  |
| --- | --- |
| 内存带宽测试 | |
| V001带宽测试均速 | 11000.55MB/S |
| V002带宽测试均速 | 11003.02MB/S |

阿里云不提供对外的数据测试结果

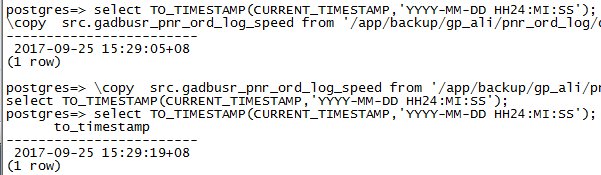
## 2. SQL测试场景

### 2.1 多种导入方式比较

阿里云现有五种数据导入方式

|  |  |  |
| --- | --- | --- |
| 数据导入方式 | 方式与限制 | 速度 |
| OSS 外部表同步数据 | 每张表一个oss的外部文件,上传导入繁琐 | 39559  行/秒 |
| Copy | 必须在psql命令行下使用  (因为没有OS的文件创建权限) | 15567  行/秒 |
| DTS数据同步 | 收费服务 | N/A |
| MYSQL导入 | 没有PostgreSQL环境暂时不测试 | N/A |
| PostgreSQL导入 | 没有PostgreSQL环境暂时不测试 | N/A |

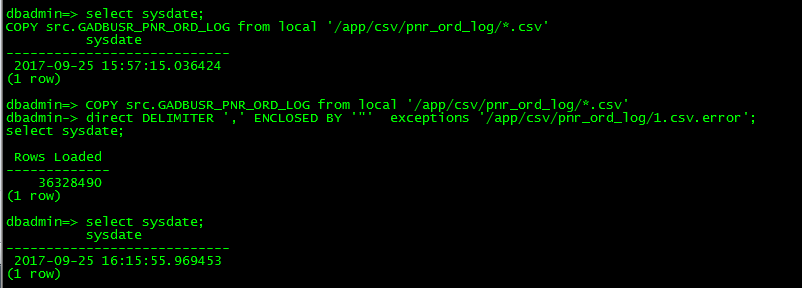
217938行数据,14秒



对比vertica的导入速度(单节点测试环境)

|  |  |  |
| --- | --- | --- |
| 数据导入方式 | 方式与限制 | 速度 |
| Copy | 远程主机或主机上文件导入 | 32436  行/秒 |

36328490行数据,1120秒



### 2.2 导入比较结论

传统copy导入数据场景下,Vertica的导入速度约是底层架构为GreenPlum的阿里数仓的1倍,且Vertica可在命令行或文件系统上操作,不受数据库服务的限制.

使用OSS作为中转进行导入,阿里数仓的速度与Vertica基本持平,但是OSS导入的操作繁琐,主要体现在步骤较多且需人工干预.

同时,在错误信息输出上,Vertica的分类明确,能精确定位到行及错误明细,阿里数仓由于GP的限制,只能限制出错率,且无现成的错误提示工具.

### 2.3 复合SQL测试语句与实际消耗

单表千万级数据做group by 统计查询

Select acct\_date, sum (ord\_amt) from src.GADBUSR\_PNR\_ORD\_LOG

Group by acct\_date;

GP格式变化:

Select acct\_date, sum (TO\_NUMBER (ord\_amt, 999999)) from src.GADBUSR\_PNR\_ORD\_LOG

Group by acct\_date;

单表批量SQL插入性能测试

Insert into src.GADBUSR\_PNR\_ORD\_LOG\_speed

Select \* from src.GADBUSR\_PNR\_ORD\_LOG;

Update效率测试

Update src.GADBUSR\_PNR\_ORD\_LOG

Set ord\_amt=ord\_amt\*0.4+fee\_amt\*acct\_date

Where acct\_date='20170619'

delete效率测试

Delete from src.GADBUSR\_PNR\_ORD\_LOG\_speed

Where acct\_date>='20170601'

### 2.4 复合测试环境表现

|  |  |  |  |
| --- | --- | --- | --- |
| 运行时间 | Vertica二节点测试 | 阿里数仓二节点测试 | 阿里数仓四节点测试 |
| Group By效率 | **1.236 secs** | 150.385 secs | 97.850secs |
| Insert 效率 | 722.569 secs | 1002.343 secs | **483.609secs** |
| Update效率 | **7.203 secs** | 214.59 secs | 104secs |
| Delete效率 | **1.414secs** | 779.615secs | 345.687secs |

### 2.5 复合测试结论

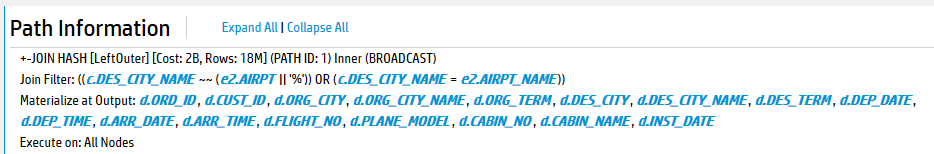
在group by聚合分析和update/delete操作中,同级别硬件的Vertica性能远超底层架构为GreenPlum的阿里数仓.Insert效率则是阿里数仓更好.

阿里数仓的性能可随着硬件和节点数拓展同步提升,也与文档中的说明一致.

### 2.6 巨型SQL测试语句与实际消耗



该查询在Vertica生产环境上的执行计划如下,其执行的cost高达20亿(2Billion)



相比之下,一般查询的cost在几百万左右(1~5Million),可见其消耗了大量的系统资源

该查询关联了6个表,又对其中2个表做了多次关联.

由于是全表查询,汇总了一下表的数量级

|  |  |
| --- | --- |
| 表名 | 行数 |
| dw3.dim\_trust\_user\_cust\_info | 约2千万行 |
| pnracct.RPT\_PASS\_INFO | 约8千万行 |
| pnracct.RPT\_ORD\_INFO | 约7千万行 |
| pnracct.RPT\_FLIGHT\_INFO | 约6千万行 |

(其余2表为万级,不统计入内,忽略)

Vertica生产环境配置:

|  |  |
| --- | --- |
| 配置情况 | 数量 |
| 节点数 | 6 |
| 内存总数 | 96G |
| 数据导入时间 | 5W/S |
| 消耗内存数 | 60G+(消耗超过阈值,查询转至磁盘执行) |
| 执行时间 | 387S |
| 文件句柄 | 1000+ |
| 线程数 | 50+ |

### 2.7 巨型SQL测试环境表现

|  |  |  |  |
| --- | --- | --- | --- |
|  | Vertica二节点测试 | 阿里数仓二节点测试 |  |
| 运行时间 | 1149.25s | 1231.98s |  |

### 2.8 巨型SQL测试结论

## 3.用户习惯

### 3.1.函数或功能变更(MySQL至PostgreSQL)

|  |  |  |
| --- | --- | --- |
| **特性** | **MySQL** | **PostgreSQL** |
| 存储过程和用户定义函数 | 支持 CREATE PROCEDURE 和 CREATE FUNCTION 语句。存储过程可以用 SQL 和 C++ 编写。用户定义函数可以用 SQL、C 和 C++ 编写。 | 没有单独的存储过程，都是通过函数实现的。用户定义函数可以用 PL/PgSQL（专用的过程语言）、PL/Tcl、PL/Perl、PL/Python 、SQL 和 C 编写。 |
| 开源问题 | 最重要的引擎InnoDB很早就由Oracle公司控制。目前整个MySQL数据库都由Oracle控制。 | BSD协议，没有被大公司垄断。 |
| 复杂查询 | 对复杂查询的处理较弱，查询优化器不够成熟 | 很强大的查询优化器，支持很复杂的查询处理。 |
| 多表连接 | 只有一种表连接类型:嵌套循环连接(nested-loop),不支持排序-合并连接(sort-merge join)与散列连接(hash join)。 | 都支持 |
| 函数索引与物化视图 | 不支持函数索引，只能在创建基于具体列的索引。不支持物化视图。 | 支持函数索引，同时还支持部分数据索引，通过规则系统可以实现物化视图的功能。 |
| Dblink跨库访问 | 不支持database link。有一种叫做Federated的存储引擎可以作为一个中转将查询语句传递到远程服务器的一个表上,不过,它功能很粗糙并且漏洞很多 | 有dblink，同时还有一个dbi-link的东西，可以连接到oracle和mysql上。 |

### 3.2.函数或功能变更(Oracle至PostgreSQL)

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **问题点** | **Oracle** | **PostgreSQL** |
| **1** | **DUAL** | SELECT 1+1 FROM DUAL | SELECT 1+1  或者 CREATE VIEW dual AS  SELECT 'X'::VARCHAR(1) AS DUMMY |
| **2** | **NEXTVAL** | SELECT TABLE1.NEXTVAL FROM   DUAL | SELECT NEXTVAL('TABLE1') FROM  DUAL |
| **3** | **ROWNUM** | WHERE ROWNUM<=5 | LIMIT 5 OFFSET 0 |
| **4** | **(+)** | SELECT \* FROM A\_TABLE A , B\_TABLE B  WHERE A.ID(+)=B.ID | SELECT \*  FROM A\_TABLE A  RIGHT OUTER JOIN  B\_TABLE B  ON A.ID=B.ID |
| **5** | **AS** | SELECT A.COL1 A\_COL1,            A.COL2 A\_COL2 FROM A\_TABLE A | SELECT A.COL1 AS A\_COL1,            A.COL2 AS A\_COL2 FROM A\_TABLE A |
| **6** | **NVL** | SELECT NVL(SUM(VALUE11),0) FS\_VALUE1,             NVL(SUM(VALUE21),0) FS\_VALUE2 FROM   FIELD\_SUM | SELECT COALESCE(SUM(VALUE11),0) AS FS\_VALUE1,            COALESCE(SUM(VALUE21),0) AS FS\_VALUE2 FROM   FIELD\_SUM |
| **7** | **TO\_ NUMBER** | SELECT COL1  FROM A\_TABLE ORDER BY TO\_NUMBER(COL1) | SELECT COL1  FROM A\_TABLE ORDER BY TO\_NUMBER(COL1,999999) [注：'999999' ---- 6位数为COL1字段的长度] |
| **8** | **DECODE** | SELECT DECODE(ENDFLAG,'1','A','B') ENDFLAG FROM TEST | SELECT  (CASE ENDFLAG  WHEN '1' THEN 'A' ELSE 'B' END) AS ENDFLAG FROM TEST |
| **9** | **时间 问题** | UPDATE A\_TABLE SET ENTREDATE=SYSDATE | UPDATE A\_TABLE SET ENTREDATE=TO\_TIMESTAMP(CURRENT\_TIMESTAMP,'YYYY-MM-DD HH24:MI:SS') |
| **10** | **||** | SELECT NULL||'-'||NULL AS VALUES1 FROM DUAL | SELECT COALESCE(NULL,'')||'-'||COALESCE(NULL,'') AS VALUES1 FROM DUAL |
| **11** | **CEIL** | SELECT CEIL(SYSDATE - TO\_DATE('20051027 14:56:10','YYYYMMDD HH24:MI:SS')) AS DAYS FROM DUAL | SELECT EXTRACT(DAY FROM  (TO\_TIMESTAMP(CURRENT\_TIMESTAMP,'YYYY-MM-DD-HH24-MI-SS') -TO\_TIMESTAMP('2005-10-27 14:56:10','YYYY-MM-DD-HH24-MI-SS') ))+1 AS DAYS FROM DUAL |
| **12** | **NULLIF** | 无NULLIF函数 | SELECT NULLIF(VALUE1,VALUE2) AS COL1 FROM DUAL [注]当VALUE1=VALUE2时，COL1=NULL |
| **13** | **CONCAT** | CONCAT(CHAR,CHAR) | 创建函数来解决 CREATE FUNCTION CONCAT(CHAR,CHAR) RETURNS CHAR AS          'SELECT $1 || $2' LANGUAGE 'sql'; |
| **14** | **ADD\_ MONTHS** | add\_months(date, int) | 创建函数来解决 CREATE FUNCTION add\_months(date, int)  RETURNS date AS  'SELECT ($1 + ( $2::text || ''months'')::interval)::date;'  LANGUAGE 'sql' |
| **15** | **LAST \_DAY** | LAST\_DAY(DATE) | 创建函数来解决 CREATE FUNCTION LAST\_DAY(DATE) RETURNS DATE AS         'SELECT date(substr(text($1 +                  interval(''1 month'')),1,7)||''-01'')-1'         LANGUAGE 'sql'; |
| **16** | **MONTHS \_BETWEEN** | MONTH\_BETWEEN(DATA,DATA) | 创建函数来解决 CREATE FUNCTION MONTH\_BETWEEN(DATA,DATA) RETURNS NUMERIC AS         'SELECT to\_number((date($1)-                            date($2)),''999999999'')/31'         LANGUAGE 'sql'; |
| **17** | **MINUS** | MINUS | 以EXCEPT来替代 |
| **18** | **子条件** |  | 在FROM子条件中字段须有列名， 处理方法用AS +别名 |

## 4.生产环境建议配置

### 4.1.硬件选型标准

1.依据:高性能原则(CPU,内存,磁盘,网络) 高可靠原则(单点故障问题) 可拓展性原则(预计何时升级)

2.CPU选型:并发主要由cpu核数决定

3.内存选型:用于存放join,排序等,要满足大部分sql的内存需求

4.磁盘选型:磁盘与内存的交互,磁盘内部的交互,都是较为关键的部分.SAS盘或SSD盘为佳,SATA盘转速较低,会有一定的性能瓶颈.做raid哪个级别也是需要考虑的问题

5.网络:千兆足以,注意冗余

## 4.GreenPlum使用手册

### 4.1 目录

第一章-GP基础操作

第二章-GP结构介绍

第三章-GP上线部署

第四章-GP数据库管理

第五章-GP数据库监控

第六章-GP数据库运维

第七章-GP数据库备份与恢复

第八章-GP数据库扩容;

