# PostgreSQL数据库选型测试报告

[PostgreSQL数据库选型测试报告 1](#_Toc25996)

[一、MySQL & PostgreSQL数据库特性对比 2](#_Toc21868)

[二、MySQL & PostgreSQL TPC-C/TPC-H 压测 3](#_Toc10120)

[2.1环境配置 3](#_Toc2669)

[2.2专业术语 3](#_Toc23235)

[2.3 测试场景 5](#_Toc12453)

[2.4 测试过程与测试结果： 5](#_Toc1151)

[2.5 测试结论： 10](#_Toc18911)

[三、 目前云平台mysql数据库面临的主要问题 11](#_Toc12973)

[3.1 目前云平台mysql数据库主要问题： 11](#_Toc6759)

[3.2 PostgreSQL以上业务场景对比测试 11](#_Toc22012)

[四、PG高可用方案 16](#_Toc32613)

[4.1 PostgreSQL常用的高可用工具 16](#_Toc4447)

[4.2 Patroni 16](#_Toc2242)

[4.3 分布式插件Citus 18](#_Toc21207)

# 一、MySQL & PostgreSQL数据库特性对比

|  |  |  |
| --- | --- | --- |
| 项目 | MySQL | PostgreSQL |
| 适用场景 | OLTP | OLTP/OLAP |
| 开源协议 | GNU General Public License (GPLv2) | PostgreSQL is released under the PostgreSQL License, a liberal Open Source license, similar to the BSD or MIT licenses. |
| 流行程度 | 目前最流行的开源数据库，DB-Engines网站排名第二  2019年度数据库 | DB-Engines目前排名第四，上升趋势非常明显，全球最先进的开源数据库。  2017、2018、2020年度数据库 |
| 线程模型 | 单进程多线程 | 多进程 |
| 主从复制 | Binlog逻辑复制  逻辑复制，延迟大(大数据量dml、load data等)  逻辑复制容易主从数据不一致 | 物理复制  物理复制，延迟小  物理复制，主备数据严格一致 |
| 优化器 | 基于CBO的优化器  统计信息简单  8.0之前只支持nest loop join  复杂查询性能不佳 | 功能更完备的CBO优化器  统计信息丰富  支持nest loop,merge sort和hash join  复杂查询性能很好，支持序列、窗口函数、CTE等 |
| 并行查询 | 不支持并行查询 | 支持并行查询 |
| 分区表 | Mysql分区表问题多，通常规范中不推荐使用 | 分区表功能成熟，大量用于生产系统 |
| 可扩展性 | PostgreSQL是高度可扩展的，您可以添加和拥有数据类型，运算符，索引类型和功能语言。 | 无法扩展 |
| 高可用方案 | 依赖第三方工具，  如mha/orachestrator等 | 依赖第三方工具，  如patroni/repmgr/pg\_pool等 |
| 分布式方案 | mycat | Citus/pgxc/pgxl  Greenplum/Yellobrick/TBase/GaussDB/ |
| 总结 | MySQL是单进程多线程的开源数据库，流行程度高，适合高并发、中小规模数据量下的OLTP，多表关联性能很低，通常单表数据量不超过500万行，多表关联通常不能超过3个表。数据量大之后需要分库分表但有关联问题。 | PostgreSQL是多进程的开源数据库，流行程度比MySQL低，完全开源，功能强大，优化器成熟。能够处理OLTP和OLAP的HTAP数据库，大数量和多表关联下仍然有很好的性能，分区表功能完善。  PostgreSQL更接近Oracle，都是多进程、物理复制、支持并行查询，是很多公司脱O迁出的首选数据库。 |
| 适用场景 | 业务形态简单、高并发、数据量不大的OLTP | 中大规模数据量的HTAP  复杂查询、多表关联、分析函数等的OLAP  Oracle、DB2等商业数据库迁移首选数据库 |

# 二、MySQL & PostgreSQL TPC-C/TPC-H 压测

## 2.1环境配置

|  |  |
| --- | --- |
| 类别 | 说明 |
| OS | Docker CentOS 7.6-1.3.0镜像 |
| CPU | 32C |
| Memory | 180G |
| 存储 | 8TB |
| MySQL版本 | 5.7.25 |
| PostgreSQL版本 | 12.5 |

## 2.2专业术语

TPC: Transaction Processing Performance Council,是一个非盈利组织，成立于1988年，这个组织主要的功能是定义事务处理、数据库的基准，这个基准用于评估服务器的性能，并且把服务器评估的结果发布在TPC的官方网站上。简单的讲TPC就是系统评测皇家科学院。

TPC-C: TPC-C is an On-Line Transaction Processing Benchmark.

TPC-C is measured in transactions per minute (tpmC).

TPC-C是业界常用的一套benchmark，由TPC委员会制定发布，用于评测数据库的联机交易处理（OLTP）能力。主要涉及10张表，包含五类业务事务模型（NewOrder–新订单的生成、Payment–订单付款、OrderStatus–最近订单查询、Delivery–配送、StockLevel–库存缺货状态分析）。

TPC-C事务模型

TPC-C需要处理的交易事务主要为以下几种：

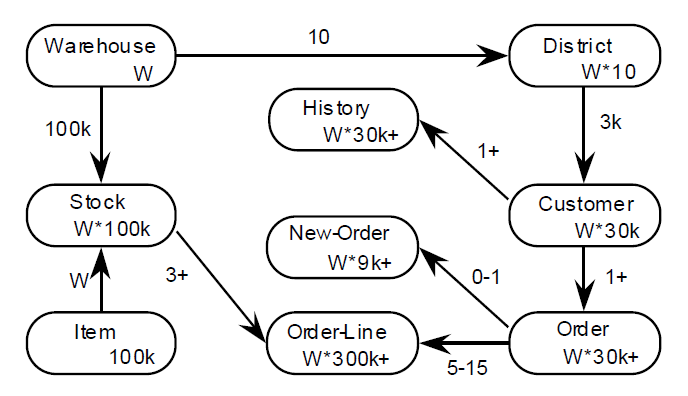
1、新订单（New-Order） ：客户输入一笔新的订货交易；

2、支付操作（Payment） ：更新客户帐户余额以反映其支付状况；

3、发货（Delivery） ：发货（模拟批处理交易）；

4、订单状态查询（Order-Status） ：查询客户最近交易的状态；

5、库存状态查询（Stock-Level） ：查询仓库库存状况，以便能够及时补货。



TPC-C通过tpmC值（Transactions per Minute）来衡量系统最大有效吞吐量（MQTh，Max Qualified Throughput），其中Transactions以NewOrder Transaction为准，即最终衡量单位为每分钟处理的新订单数

TPC-C性能衡量指标tpmC

流量指标(Throughput,简称tpmC)：按照TPC组织的定义，流量指标描述了系统在执行支付操作、订单状态查询、发货和库存状态查询这4种交易的同时，每分钟可以处理多少个新订单交易。所有交易的响应时间必须满足TPC-C测试规范的要求，且各种交易数量所占的比例也应该满足TPC-C测试规范的要求。在这种情况下，流量指标值越大说明系统的联机事务处理能力越高。

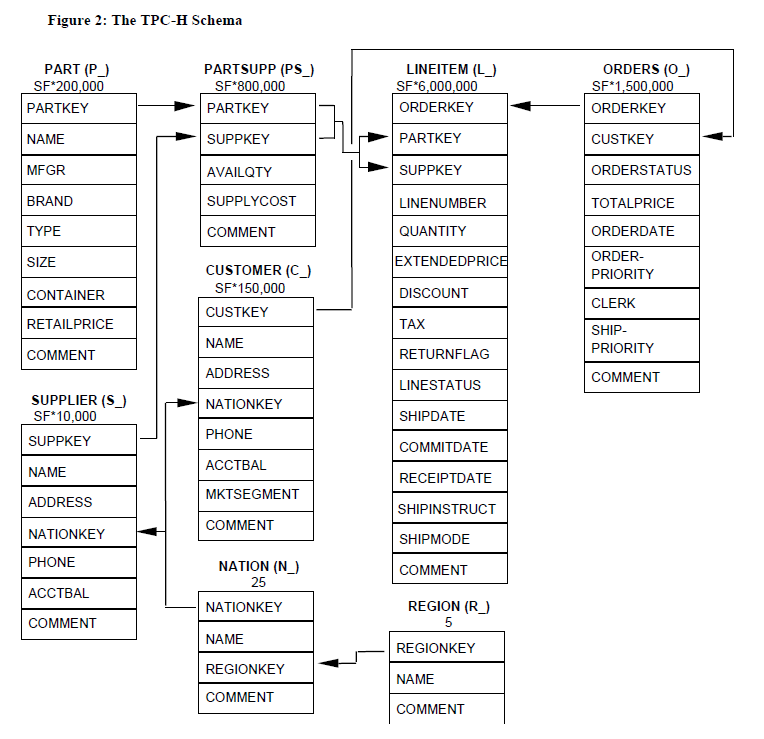
TPC-H:

TPC-H is a Decision Support Benchmark

TPC-H 是业界常用的一套 Benchmark，由 TPC 委员会制定发布，用于评测数据库的分析型查询能力。TPC-H 查询包含 8 张数据表、22 条复杂的 SQL 查询，大多数查询包含若干表 Join、子查询和 Group-by 聚合等。

简单来说，TPC-H就是通过22个复杂SQL查询来评估数据库OLAP的性能。

TPC-H表模型：



## 2.3 测试场景

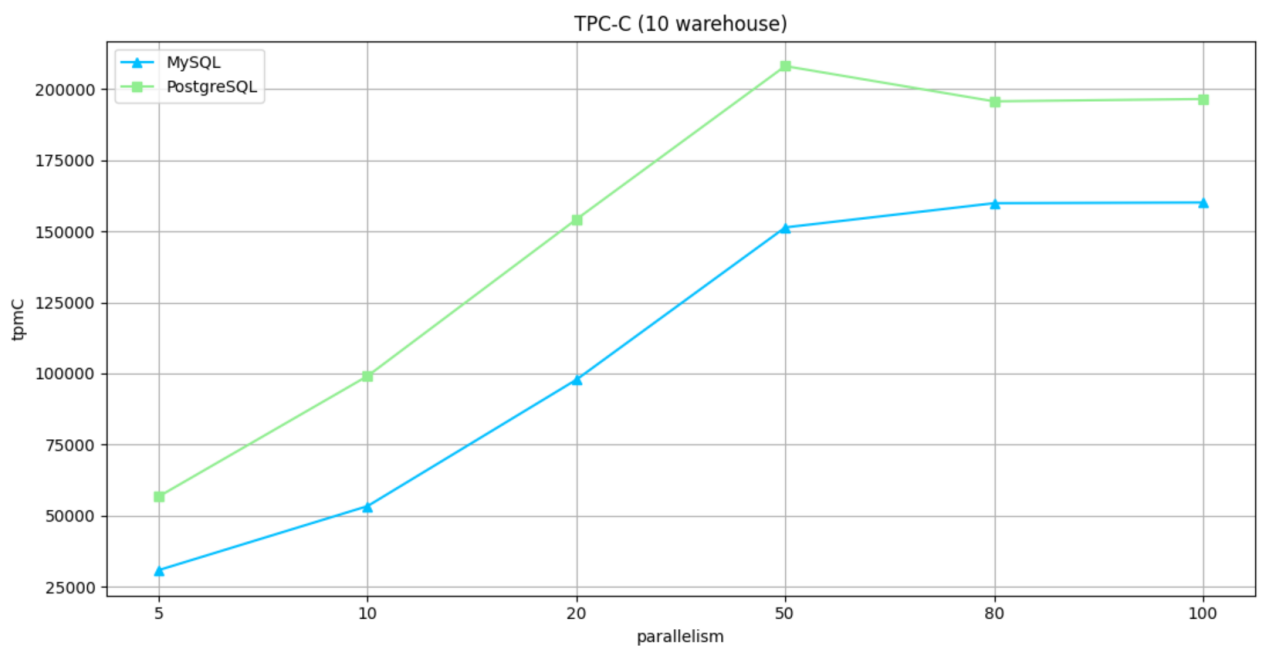
通过BenchmarkSQL的TPC-C模型压测MySQL/PostgreSQL 在不同数据量(warehouse)不同并发度下的OLTP处理能力

TPC-H压测MySQL/PostgreSQL OLAP处理能力

## 2.4 测试过程与测试结果：

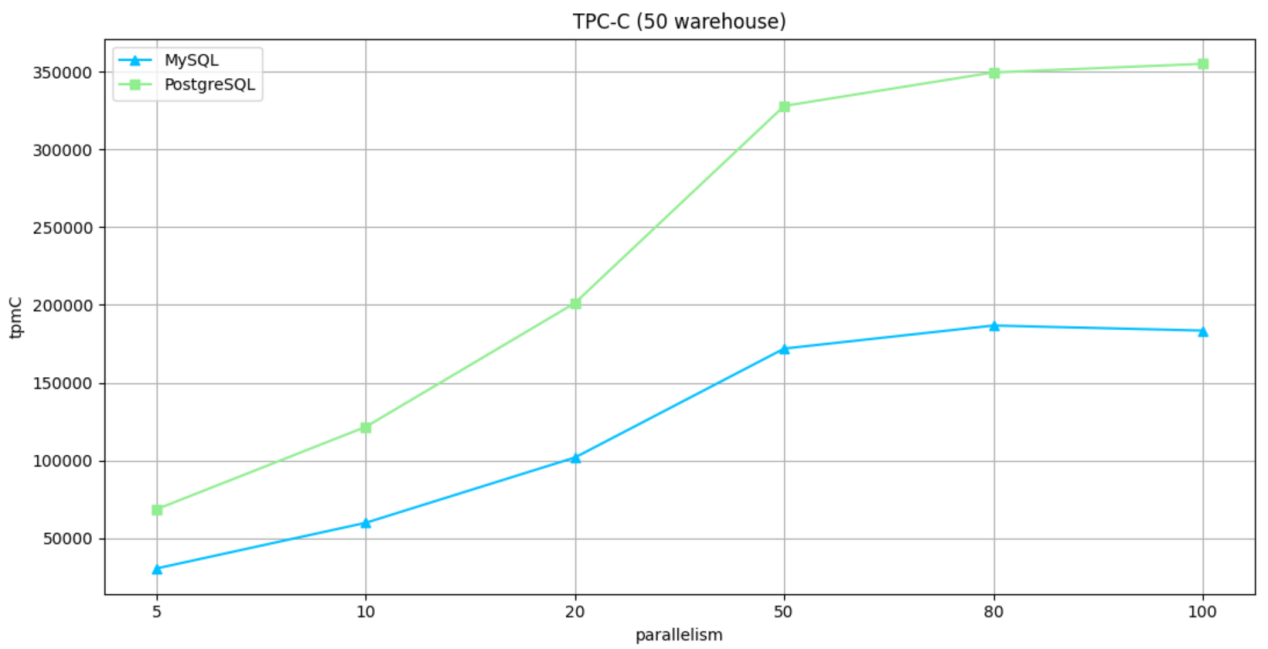
TPC-C 10 warehouse(数据量 customer 300000, 数据量 300 0000)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 并发度 | tpmC-MySQL | tpmC-MySQL(备注) | tpmC-PostgreSQL | tpmC-PostgreSQL(备注) |
| 5 | 30872.68 |  | 56751.44 |  |
| 10 | 53372.98 |  | 99111.11 | CPU 16% |
| 20 | 97824.88 |  | 154200.91 | CPU 34.3% |
| 50 | 151354.23 | CPU 72% | 208079.38 | CPU 77.2% |
| 80 | 159871.29 | CPU 86% | 195675.97 | CPU 92% |
| 100 | 160104.39 | CPU 89.6% | 196503.94 | CPU 100% |



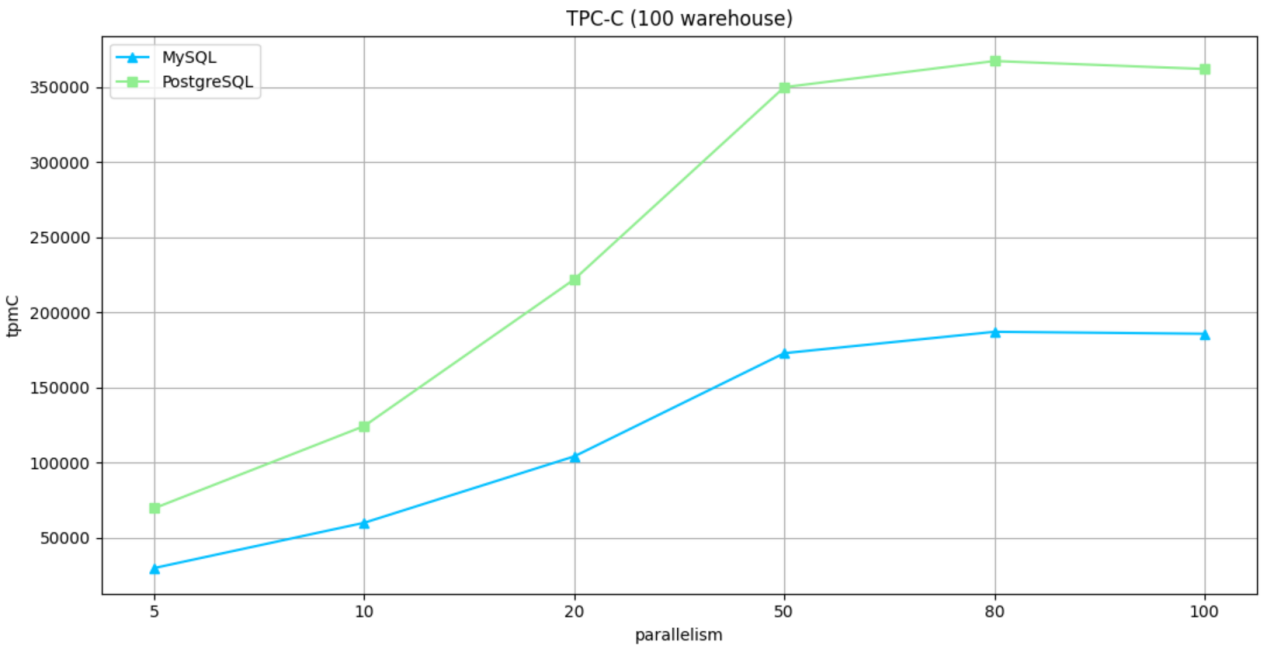
TPC-C 50 warehouse(数据量 customer 150 0000, 数据量 1500 0000 )

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 并发度 | tpmC-MySQL | tpmC-MySQL(备注) | tpmC-PostgreSQL | tpmC-PostgreSQL(备注) |
| 5 | 30630.98 |  | 68698.88 | CPU 9.8% |
| 10 | 59976.6 | CPU 20% | 121670.08 | CPU 19.6% |
| 20 | 101944.25 | CPU 40% | 201264.77 | CPU 39% |
| 50 | 171949.63 | CPU 86% | 327904.45 | CPU 85.3% |
| 80 | 186754.5 | CPU 100% | 349437.28 | CPU 100% |
| 100 | 183519.52 | CPU 100% | 354960.95 | CPU 100% |



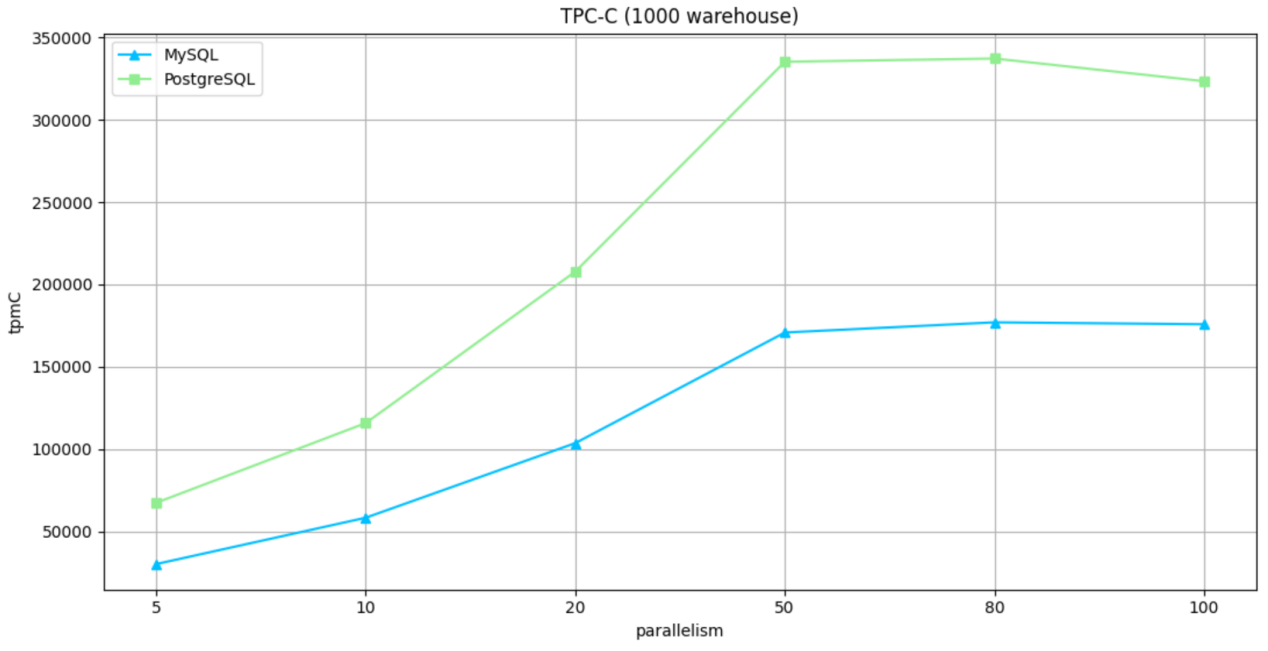
TPC-C 100 warehouse(数据量 customer 300 0000, 数据量 3000 0000)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 并发度 | tpmC-MySQL | tpmC-MySQL(备注) | tpmC-PostgreSQL | tpmC-PostgreSQL(备注) |
| 5 | 29656.35 | CPU 10.5% | 69456.03 | CPU 9.7% |
| 10 | 59816.2 | CPU 21.2% | 124215.02 | CPU 24.2% |
| 20 | 104020.01 | CPU 40.8% | 221906.19 | CPU 40.2% |
| 50 | 172759.54 | CPU 87.5% | 349877.03 | CPU 87.8% |
| 80 | 186974.33 | CPU 100% | 367292.42 | CPU 100% |
| 100 | 185676.76 | CPU 100% | 361991.76 | CPU 100% |



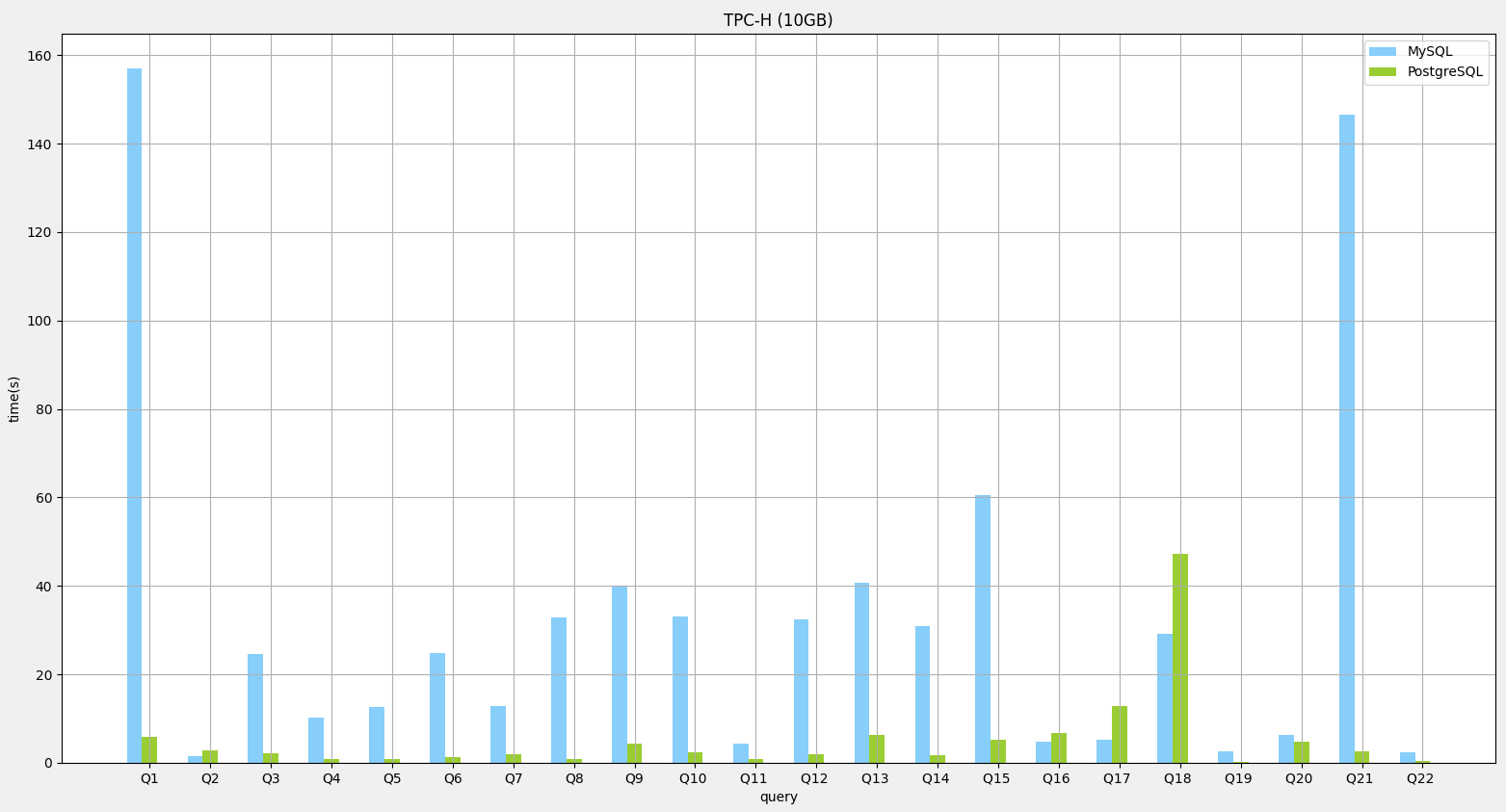
TPC-C 1000 warehouse(数据量 customer 300 0000, 数据量 30000 0000)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 并发度 | tpmC-MySQL | tpmC-MySQL(备注) | tpmC-PostgreSQL | tpmC-PostgreSQL(备注) |
| 5 | 30121.84 | CPU 10.5% | 67347.31 | CPU 9.3% |
| 10 | 58283.22 | CPU 20.8% | 115771.29 | CPU 19.6% |
| 20 | 103594.32 | CPU 41.6% | 207786.85 | CPU 37.9% |
| 50 | 170775.55 | CPU 86.8% | 335236.92 | CPU 86.6% |
| 80 | 176972.6 | CPU 93.6% | 337200.53 | CPU 93.7% |
| 100 | 175798.1 | CPU 92.8% | 323425.16 | CPU 92.3% |



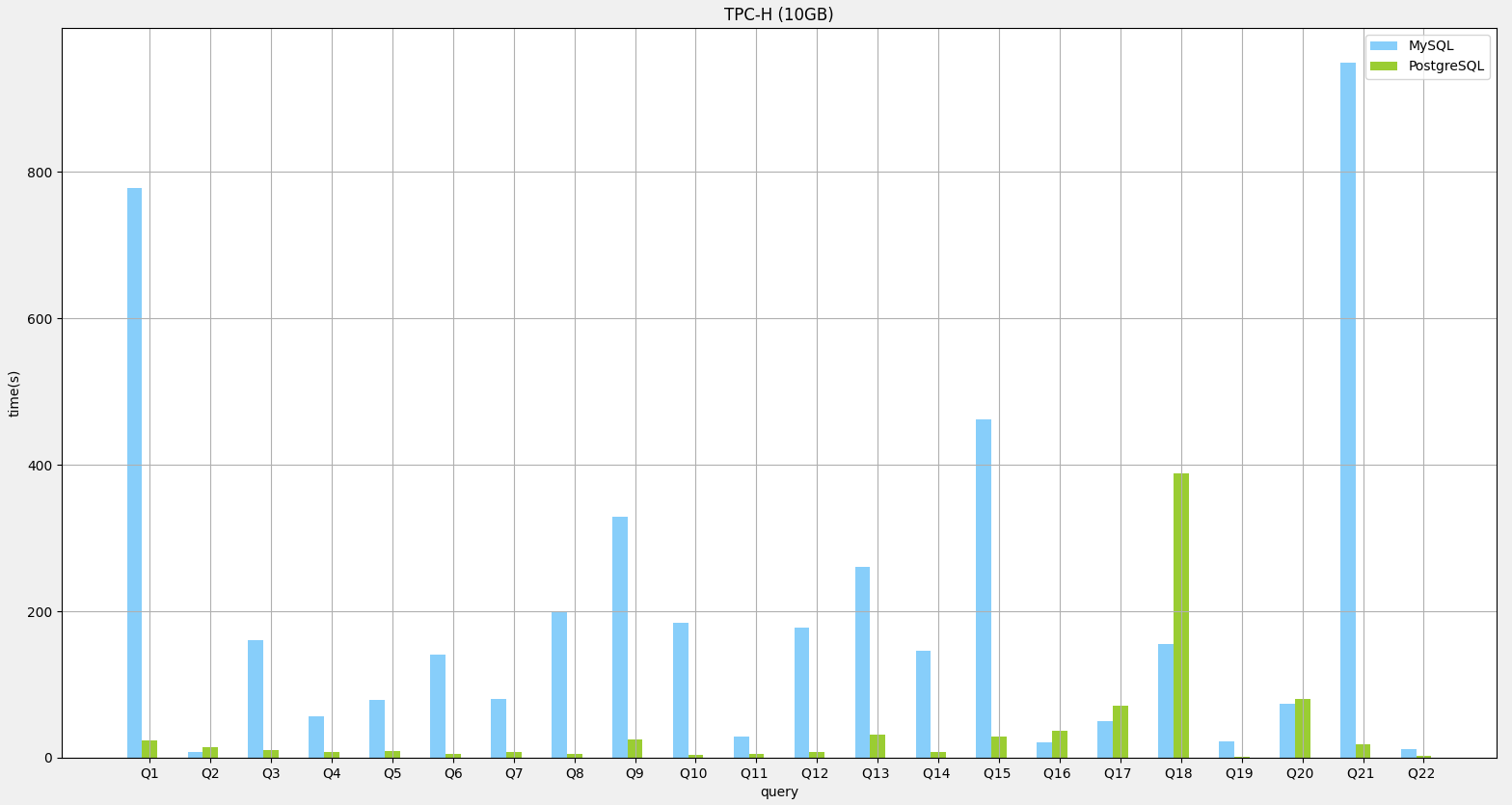
TPC-H 10G(Scalar Factor = 10),数据量(LINEITEM表约6000万行，ORDERS表1500万行，PARSUPP表800万行)

|  |  |  |
| --- | --- | --- |
| Query | MySQL（单位：秒） | PostgreSQL（单位：秒） |
| Q01 | 157 | 5.90 |
| Q02 | 1.53 | 2.89 |
| Q03 | 24.60 | 2.15 |
| Q04 | 10.12 | 0.80 |
| Q05 | 12.51 | 0.92 |
| Q06 | 24.90 | 1.33 |
| Q07 | 12.81 | 1.91 |
| Q08 | 32.76 | 0.86 |
| Q09 | 40.08 | 4.39 |
| Q10 | 33.16 | 2.40 |
| Q11 | 4.34 | 0.77 |
| Q12 | 32.32 | 1.84 |
| Q13 | 40.67 | 6.28 |
| Q14 | 30.95 | 1.67 |
| Q15 | 60.45 | 5.24 |
| Q16 | 4.69 | 6.63 |
| Q17 | 5.28 | 12.75 |
| Q18 | 29.14 | 47.30 |
| Q19 | 2.55 | 0.17 |
| Q20 | 6.30 | 4.67 |
| Q21 | 146.52 | 2.59 |
| Q22 | 2.33 | 0.44 |



TPC-H 50G(Scalar Factor = 50),数据量(LINEITEM表约3亿行，ORDERS表7500万行，PARSUPP表4000万行)

|  |  |  |
| --- | --- | --- |
| Query | MySQL（单位：秒） | PostgreSQL（单位：秒） |
| Q01 | 778.42 | 23.42 |
| Q02 | 7.35 | 14.23 |
| Q03 | 160.87 | 10.00 |
| Q04 | 56.07 | 7.80 |
| Q05 | 79.27 | 8.88 |
| Q06 | 140.67 | 4.87 |
| Q07 | 80.00 | 7.88 |
| Q08 | 198.67 | 5.68 |
| Q09 | 329.18 | 25.29 |
| Q10 | 184.32 | 13.25 |
| Q11 | 28.16 | 4.86 |
| Q12 | 178.12 | 7.87 |
| Q13 | 259.97 | 31.60 |
| Q14 | 146.12 | 7.60 |
| Q15 | 462.34 | 28.50 |
| Q16 | 20.87 | 37.07 |
| Q17 | 50.52 | 71.14 |
| Q18 | 155.30 | 388.95 |
| Q19 | 21.66 | 0.95 |
| Q20 | 73.77 | 80.04 |
| Q21 | 949.47 | 18.15 |
| Q22 | 11.64 | 3.12 |



附录：苏宁易购MySQL/PG压测数据对比

<https://www.sohu.com/a/433657649_411876>

## 2.5 测试结论：

TPC-C的OLTP测试中，PostgreSQL在不同数据量不同并发度下有更高tpmC.

TPC-H的OLAP测试中，PostgreSQL在大部分复杂查询下比MySQL都有更低的响应时间，尤其是在大数据量以及多表关联的复制查询，PostgreSQL表现出更好的性能。

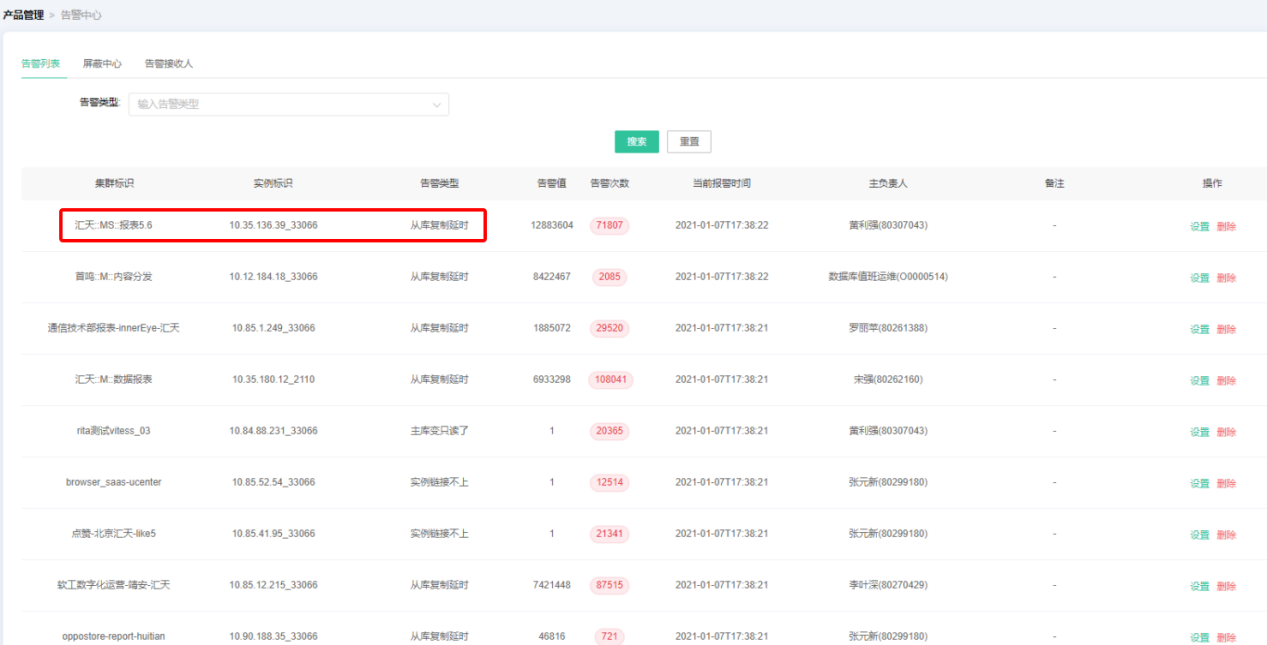
# 目前云平台mysql数据库面临的主要问题

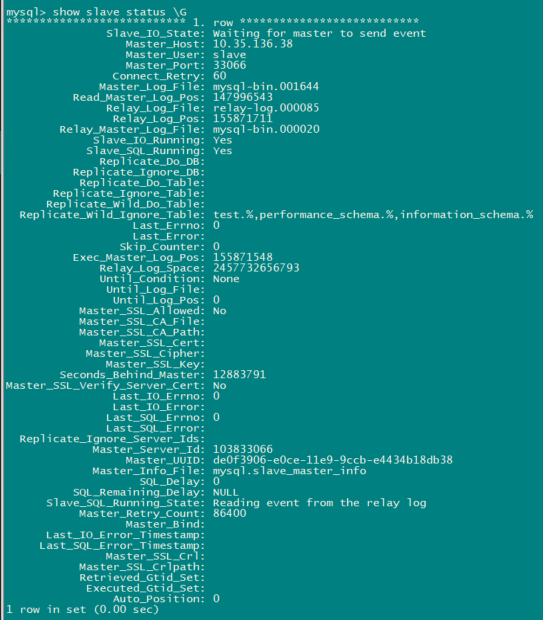
## 3.1 目前云平台mysql数据库主要问题：

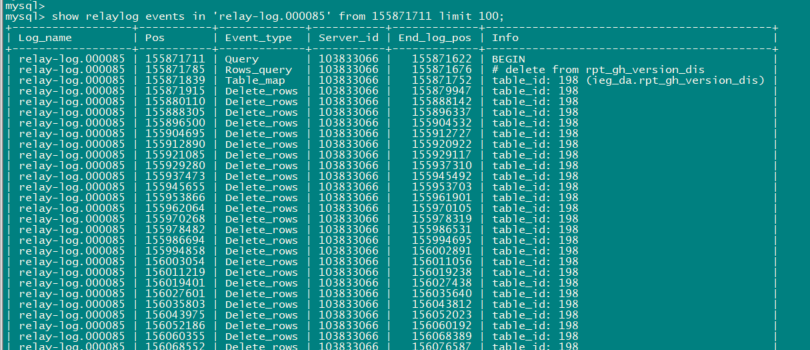
* 大表大批量dml操作，表上没有主键、索引，主从复制延迟，丢失高可用，无法迁移上云
* Load data导致主从复制延迟，binlog文件过大，写入性能低
* 大表查询、复杂查询、多表关联性能差
* 使用分区表的open file过大报错及全局MDL锁

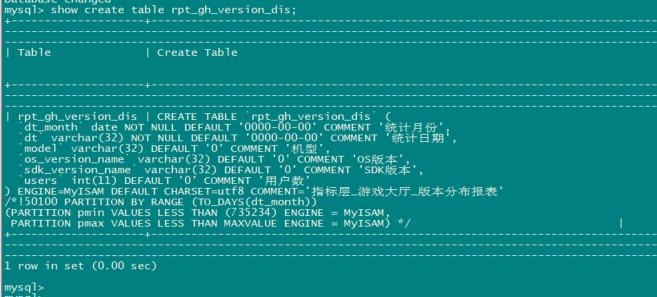
## 3.2 PostgreSQL以上业务场景对比测试

3.2.1 PG物理流复制，大表DML操作从库无复制延时问题



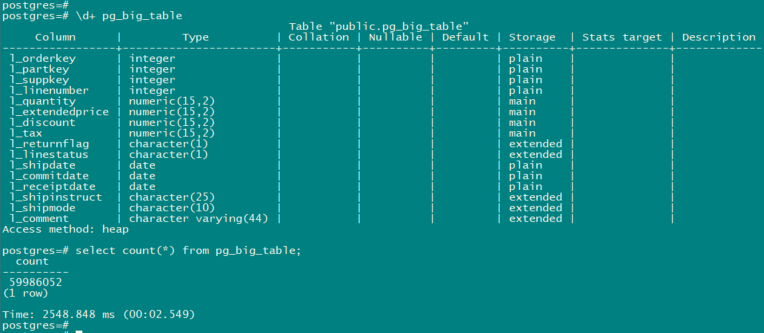




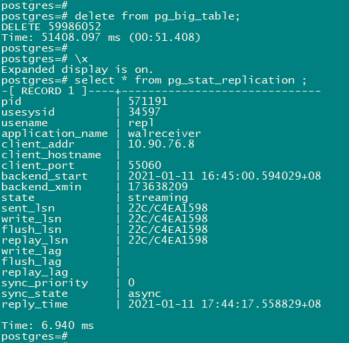


该表rpt\_gh\_version\_di有8725852行数据，无主键无索引，delete from rpt\_gh\_version\_dis造成主从复制延时，relaylog从2020-08-11开始堆积，堆积2.3T

PostgreSQL下主库端删除大表主从复制测试，验证如下：

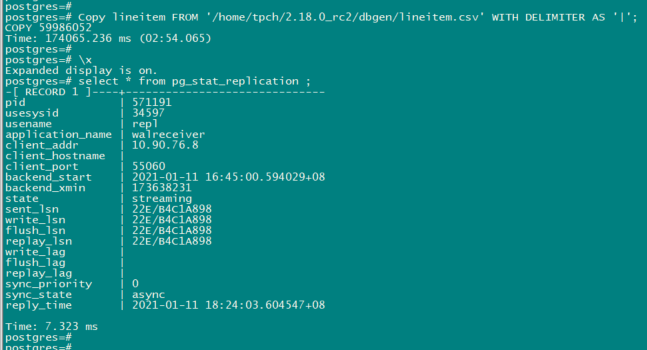


主库端删除大约6000万行的大表，删除完成后从库立即同步完成，PG物理流复制，主从复制无延迟。



3.2.2 PG物理流复制，load data从库无复制延时问题，load data性能更高

导入6000万行数据，从库无复制延时



MySQL & PG load data性能对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | MySQL耗时 | MySQL备注 | PG耗时 | PG备注 |
| 6000万 | 434.27秒  Query OK, 59986052 rows affected (7 min 14.27 sec)  Records: 59986052 Deleted: 0 Skipped: 0 Warnings: 0 | Binlog过大  主从复制延时 | 173.54秒  COPY 59986052  Time: 173542.533 ms (02:53.543) | 主从复制无延时 |

3.2.3 PG在支持复杂业务场景时有更好的性能

集群名称: 汇天-贷超-后台报表

主要问题：该集群为典型的OLAP报表系统，频繁地load data/存在大量的慢查询/主从复制延时明显。

对以下慢SQL迁移在PG上进行对比测试，性能得到明显提升。另外，PG load data相比MySQL也有更好性能。

慢SQL01:

alter table rpt\_loan\_channel\_admission\_convert\_d truncate partition p20210110;

慢SQL02:

SELECT db\_loan\_report.rpt\_wallet\_buscard\_use\_d.dayno AS col\_29541\_dayno, sum(use\_cards)/sum(use\_card\_users) AS met\_21688\_direct\_0

FROM db\_loan\_report.rpt\_wallet\_buscard\_use\_d

WHERE db\_loan\_report.rpt\_wallet\_buscard\_use\_d.dayno >= '2020-10-13 00:00:00' AND db\_loan\_report.rpt\_wallet\_buscard\_use\_d.dayno <= '2021-01-10 23:59:59' GROUP BY db\_loan\_report.rpt\_wallet\_buscard\_use\_d.dayno ORDER BY col\_29541\_dayno desc

LIMIT 500000;

慢SQL03:

SELECT count(\*) AS count\_1

FROM (SELECT db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.dt AS col\_32824\_dt, sum(db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.homepage\_uv) AS met\_24361\_direct\_0, sum(db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.certification\_uv) AS met\_24362\_direct\_0, sum(db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.verify\_cnt) AS met\_24363\_direct\_0, sum(verify\_cnt)/sum(certification\_uv) AS met\_24446\_direct\_0, sum(db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.credit\_apply\_dcnt) AS met\_24365\_direct\_0, sum(db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.credit\_pass\_dcnt) AS met\_24366\_direct\_0, sum(credit\_pass\_cnt)/sum(credit\_apply\_cnt) AS met\_24449\_direct\_0, sum(credit\_limit\_sum)/sum(credit\_pass\_dcnt) AS met\_24448\_direct\_0, sum(credit\_rate\_sum)/sum(credit\_pass\_dcnt) AS met\_24447\_direct\_0, sum(db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.loan\_apply\_dcnt) AS met\_24372\_direct\_0, sum(db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.loan\_success\_dcnt) AS met\_24374\_direct\_0, sum(loan\_success\_cnt)/sum(loan\_apply\_cnt) AS met\_24450\_direct\_0, sum(db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.loan\_success\_amount) AS met\_24377\_direct\_0

FROM db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2

WHERE db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.dt >= '2020-01-01 00:00:00' AND db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.dt <= '2020-12-31 00:00:00' AND db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.first\_channel\_name\_group IN ('unknown', 'other', '广告', '体系内合作', '软件商店', '浏览器', '短信', '游戏中心', 'OPPO商城', '我的OPPO') GROUP BY db\_loan\_report.rpt\_loan\_mart\_source\_report\_v2.dt) AS expr\_qry;

|  |  |  |  |
| --- | --- | --- | --- |
| SQL语句编号 | 表数据量 | MySQL慢日志中记录时间(单位秒) | PG库执行时间 |
| SQL01 | 155590 | Query\_time: 584.204737 Lock\_time: 372.631809 | 0.02  直接truncate对应分表或DETACH  truncate table rpt\_loan\_channel\_admission\_convert\_d\_p20210110; |
| SQL02 | 3827833 | 30.106177 | Time: 67.457 ms  0.067s |
| SQL03 | 9309832 | 31.442047 | Time: 71.208 ms  0.071s |

# 四、PG高可用方案

## 4.1 PostgreSQL常用的高可用工具



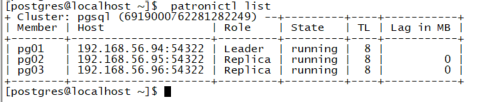
|  |  |  |  |
| --- | --- | --- | --- |
|  | PATRONI | PAF | REPMGR |
| 特点 | 支持REST APIs和HAproxy集成  支持回调脚本(callbacks scripts)  DCS(Etcd/zookeeper/consul/kubernets) | Pacemaker and Corosync stack. | 配置简单  不支持DCS  会在数据库中创建repmgr系统表  自动切换需要配置节点ssh互信 |

## 4.2 Patroni

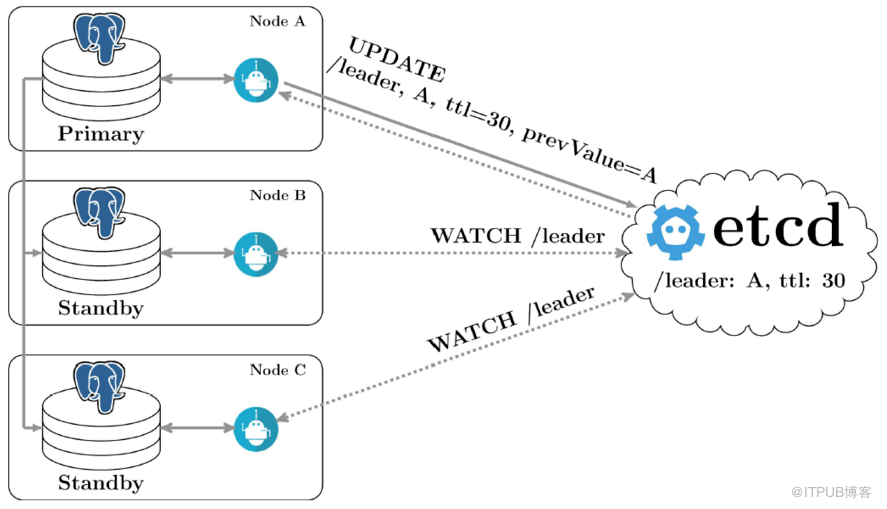
Patroni是目前PostgreSQL数据库国内应用广泛、成熟的开源高可用工具。

* 由Zalando公司开发(Zalando 是总部位于德国柏林的大型网络电子商城,PG重度用户)
* 100% Open source
* 支持自动、手动switchover/failover
* 通过DCS(etcd, zookeeper, Consul等)保存集群元数据，DCS自身高可用
* 支持配置callbacks scripts
* 支持Rest API
* 支持集成HAproxy

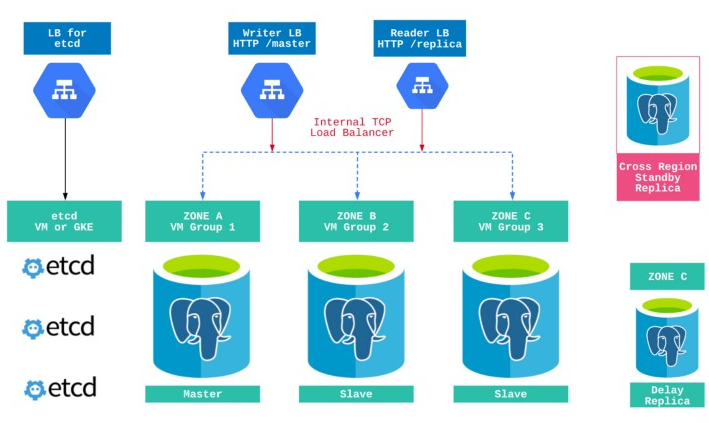
一主两从环境：



Patroni集群基本架构图：



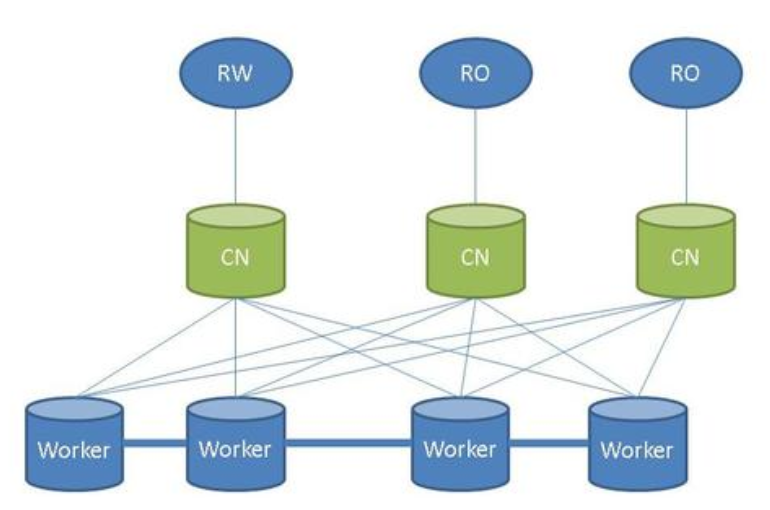
跨区高可用Patroni架构：



## 4.3 分布式插件Citus

Citus是开源的PostgreSQL分布式数据库插件，由协调器节点(coordinator)和Worker节点构成一个数据库集群，worker节点可以作为分片分库分表，支持水平扩容。应用将查询发送到协调器节点，协调器处理后发送至各个work节点并行执行，并返回最终数据。

<https://www.citusdata.com/>



Citus读写分离架构：

Master读写vip , standby读vip

CN节点连接读写vip为master

CN节点连接读vip为standby

patroni callback切换vip

CN节点主备切换，通过patroni回调函数修改

citus.use\_secondary\_nodes=always

Worker3\_standby

读vip

Worker2\_primary

读写vip

Worker3\_primary

读写vip

Worker1\_primary

读写vip

Worker2\_standby

读vip

Worker1\_standby

读vip

RO

RW

CN master

CN standby

CN standby

RO