ALIBABA CLOUD

阿里云

云数据库 PolarDB
性能白皮书

文档版本：20220422

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.性能白皮书（OLAP） 05
1.1. 性能测试方法（OLAP） 05
1.2. PolarDB MySQL引擎 HTAP性能 12
1.3. 并行查询性能 14
2.性能白皮书（OLTP） 19
2.1. 性能测试方法（OLTP） 19
2.2. PolarDB MySQL引擎8.0.2版本性能（集群版） 22
2.3. PolarDB MySQL引擎8.0.1版本性能（集群版） 29
2.4. PolarDB MySQL引擎与RDS MySQL性能对比 34
2.5. PolarDB MySQL引擎5.7版本性能（集群版） 36
2.6. PolarDB MySQL引擎5.6版本性能（集群版） 43
2.7. PolarDB MySQL引擎8.0.1版本历史库与集群版性能对比 48

1.性能白皮书（OLAP）

1.1. 性能测试方法（OLAP）

本文档基于TPC-H测试PolarDB MySQL引擎8.0版本集群版的OLAP负载性能，您可以按照本文介绍自行测试对比，快速了解数据库系统的性能。

注意事项

本文的TPC-H的实现基于TPC-H的基准测试，并不能与已发布的TPC-H基准测试结果相比较，本文中的测试并不符合TPC-H基准测试的所有要求。

测试环境

- 测试的ECS和PolarDB均在同一地域、同一可用区。
- 网络类型均为VPC网络。

> **说明** ECS实例和PolarDB集群需保证在同一个VPC中。

- 测试用PolarDB集群如下：
  - 节点规格为polar.mysql.x8.4xlarge（32核256 GB独享规格）。
  - 版本为MySQL 8.0.1和MySQL 8.0.2。
  - 产品系列为集群独享版。
  - 节点数量：
    - 并行查询测试节点数量为两个（一个主节点、一个只读节点）。
    - HTAP性能测试节点数量为三个（一个主节点、一个只读节点、一个只读列存节点）。
  - 并行查询测试使用的链接地址为主地址，如何查看PolarDB主地址请参见申请集群地址和主地址。
- 测试用ECS实例信息如下：
  - 实例规格为ecs.c5.4xlarge。
  - 实例挂载1000 GB高效云盘。
  - 实例所使用的镜像为CentOS 7.0 64位。

测试工具

TPC-H是业界常用的一套基准，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。

安装TPC-H

1. 在ECS上安装TPC-H。

   > **说明**
   > - 本文使用的TPC-H版本为TPC-H_Tools_v2.18.0。
   > - TPC-H需要完成注册后才可以下载。

2. 打开*dbgen*目录。

```
cd dbgen
```

3. 复制 `makefile` 文件。

```
cp makefile.suite makefile
```

4. 修改 `makefile` 文件中的CC、DATABASE、MACHINE、WORKLOAD等参数定义。
   i. 打开 `makefile` 文件。

   ```
   vim makefile
   ```

   ii. 修改CC、DATABASE、MACHINE、WORKLOAD参数的定义。

   ```
   ################
   ## CHANGE NAME OF ANSI COMPILER HERE
   ################
   CC      = gcc
   # Current values for DATABASE are: INFORMIX, DB2, ORACLE,
   #                                  SQLSERVER, SYBASE, TDAT (Teradata)
   # Current values for MACHINE are:  ATT, DOS, HP, IBM, ICL, MVS,
   #                                  SGI, SUN, U2200, VMS, LINUX, WIN32
   # Current values for WORKLOAD are:  TPCH
   DATABASE= MYSQL
   MACHINE = LINUX
   WORKLOAD = TPCH
   ```

   iii. 按Esc键，然后输入 `:wq` 保存并退出。
5. 修改 `tpcd.h` 文件，并添加新的宏定义。
   i. 打开 `tpcd.h` 文件。

   ```
   vim tpcd.h
   ```

   ii. 添加如下宏定义。

   ```
   #ifdef MYSQL
   #define GEN_QUERY_PLAN ""
   #define START_TRAN "START TRANSACTION"
   #define END_TRAN "COMMIT"
   #define SET_OUTPUT ""
   #define SET_ROWCOUNT "limit %d;\n"
   #define SET_DBASE "use %s;\n"
   #endif
   ```

   iii. 按Esc键，然后输入 `:wq` 保存并退出。
6. 对文件进行编译。

```
make
```

编译完成后该目录下会生成两个可执行文件：

- `dbgen` ：数据生成工具。在使用InfiniDB官方测试脚本进行测试时，需要用该工具生成tpch相关表数据。
- `qgen` ：SQL生成工具。生成初始化测试查询，由于不同的seed生成的查询不同，为了结果的可重

复性，请使用附件提供的22个查询。

7. 使用TPC-H生成测试数据。

```
./dbgen -s 100
```

dbgen参数 `-s` 的作用是指定生成测试数据的仓库数。

8. 使用TPC-H生成查询。

> **说明** 为了测试结果可重复，您可以忽略下面生成查询的步骤，使用附件的22个查询进行测试。

i. 将 `qgen` 与 `dists.dss` 复制到*queries*目录下。

```
cp qgen queries
cp dists.dss queries
```

ii. 使用以下脚本生成查询。

```
#!/usr/bin/bash
for i in {1..22}
do
  ./qgen -d $i -s 100 > db"$i".sql
done
```

测试方法

1. 在ECS上连接PolarDB数据库，具体操作请参见连接数据库集群。
2. 创建数据库。

```
create database tpch100g
```

3. 创建表。

```
source ./dss.ddl
```

> **说明** `dss.ddl` 在TPC-H中*dbgen*目录下。

4. 加载数据。

i. 在ECS上创建 `load.ddl` 文件。

```
Touch load.ddl
```

ii. 打开 `load.ddl` 文件。

```
vim load.ddl
```

iii. 添加并保存如下脚本内容。

```
load data local INFILE 'customer.tbl' INTO TABLE customer FIELDS TERMINATED BY '|';
load data local INFILE 'region.tbl' INTO TABLE region FIELDS TERMINATED BY '|';
load data local INFILE 'nation.tbl' INTO TABLE nation FIELDS TERMINATED BY '|';
load data local INFILE 'supplier.tbl' INTO TABLE supplier FIELDS TERMINATED BY '|';
load data local INFILE 'part.tbl' INTO TABLE part FIELDS TERMINATED BY '|';
load data local INFILE 'partsupp.tbl' INTO TABLE partsupp FIELDS TERMINATED BY '|';
load data local INFILE 'orders.tbl' INTO TABLE orders FIELDS TERMINATED BY '|';
load data local INFILE 'lineitem.tbl' INTO TABLE lineitem FIELDS TERMINATED BY '|';
```

iv. 连接PolarDB数据库加载 `load.ddl` 文件数据。

```
source ./load.ddl
```

5. 创建主外键。

```
source ./dss.ri
```

以创建的数据库 `tpch100g` 为例，将TPC-H的 `dss.ri` 文件中的内容替换并保存为如下内容。

```
use TPCH100G;
-- ALTER TABLE REGION DROP PRIMARY KEY;
-- ALTER TABLE NATION DROP PRIMARY KEY;
-- ALTER TABLE PART DROP PRIMARY KEY;
-- ALTER TABLE SUPPLIER DROP PRIMARY KEY;
-- ALTER TABLE PARTSUPP DROP PRIMARY KEY;
-- ALTER TABLE ORDERS DROP PRIMARY KEY;
-- ALTER TABLE LINEITEM DROP PRIMARY KEY;
-- ALTER TABLE CUSTOMER DROP PRIMARY KEY;
-- For table REGION
ALTER TABLE REGION
ADD PRIMARY KEY (R_REGIONKEY);
-- For table NATION
ALTER TABLE NATION
ADD PRIMARY KEY (N_NATIONKEY);
ALTER TABLE NATION
ADD FOREIGN KEY NATION_FK1 (N_REGIONKEY) references REGION(R_REGIONKEY);
COMMIT WORK;
-- For table PART
ALTER TABLE PART
ADD PRIMARY KEY (P_PARTKEY);
COMMIT WORK;
-- For table SUPPLIER
ALTER TABLE SUPPLIER
ADD PRIMARY KEY (S_SUPPKEY);
ALTER TABLE SUPPLIER
ADD FOREIGN KEY SUPPLIER_FK1 (S_NATIONKEY) references NATION(N_NATIONKEY);
COMMIT WORK;
-- For table PARTSUPP
ALTER TABLE PARTSUPP
ADD PRIMARY KEY (PS_PARTKEY,PS_SUPPKEY);
COMMIT WORK;
-- For table CUSTOMER
ALTER TABLE CUSTOMER
```

```
ADD PRIMARY KEY (C_CUSTKEY);
ALTER TABLE CUSTOMER
ADD FOREIGN KEY CUSTOMER_FK1 (C_NATIONKEY) references NATION(N_NATIONKEY);
COMMIT WORK;
-- For table LINEITEM
ALTER TABLE LINEITEM
ADD PRIMARY KEY (L_ORDERKEY,L_LINENUMBER);
COMMIT WORK;
-- For table ORDERS
ALTER TABLE ORDERS
ADD PRIMARY KEY (O_ORDERKEY);
COMMIT WORK;
-- For table PARTSUPP
ALTER TABLE PARTSUPP
ADD FOREIGN KEY PARTSUPP_FK1 (PS_SUPPKEY) references SUPPLIER(S_SUPPKEY);
COMMIT WORK;
ALTER TABLE PARTSUPP
ADD FOREIGN KEY PARTSUPP_FK2 (PS_PARTKEY) references PART(P_PARTKEY);
COMMIT WORK;
-- For table ORDERS
ALTER TABLE ORDERS
ADD FOREIGN KEY ORDERS_FK1 (O_CUSTKEY) references CUSTOMER(C_CUSTKEY);
COMMIT WORK;
-- For table LINEITEM
ALTER TABLE LINEITEM
ADD FOREIGN KEY LINEITEM_FK1 (L_ORDERKEY)  references ORDERS(O_ORDERKEY);
COMMIT WORK;
ALTER TABLE LINEITEM
ADD FOREIGN KEY LINEITEM_FK2 (L_PARTKEY,L_SUPPKEY) references
        PARTSUPP(PS_PARTKEY,PS_SUPPKEY);
COMMIT WORK;
```

6. （可选）创建索引。

```
#!/usr/bin/bash
host=$1
port=$2
user=$3
password=$4
db=$5
sqls=("create index i_s_nationkey on supplier (s_nationkey);"
"create index i_ps_partkey on partsupp (ps_partkey);"
"create index i_ps_suppkey on partsupp (ps_suppkey);"
"create index i_c_nationkey on customer (c_nationkey);"
"create index i_o_custkey on orders (o_custkey);"
"create index i_o_orderdate on orders (o_orderdate);"
"create index i_l_orderkey on lineitem (l_orderkey);"
"create index i_l_partkey on lineitem (l_partkey);"
"create index i_l_suppkey on lineitem (l_suppkey);"
"create index i_l_partkey_suppkey on lineitem (l_partkey, l_suppkey);"
"create index i_l_shipdate on lineitem (l_shipdate);"
"create index i_l_commitdate on lineitem (l_commitdate);"
"create index i_l_receiptdate on lineitem (l_receiptdate);"
"create index i_n_regionkey on nation (n_regionkey);"
"analyze table supplier"
"analyze table part"
"analyze table partsupp"
"analyze table customer"
"analyze table orders"
"analyze table lineitem"
"analyze table nation"
"analyze table region")
for sql in "${sqls[@]}"
do
    mysql -h$host -P$port -u$user -p$password -D$db  -e "$sql"
done
```

说明

- 为了更有效地衡量并行查询带来的性能提升，您可以通过如下查询将使用到的索引数据预载到内存池中。

```
#!/bin/bash
host=$1
port=$2
user=$3
password=$4
dbname=$5
MYSQL="mysql -h$host -P$port -u$user -p$password -D$dbname"
if [ -z ${dbname} ]; then
    echo "dbname not defined."
    exit 1
fi
table_indexes=(
        "supplier PRIMARY"
        "supplier i_s_nationkey"
        "part PRIMARY"
        "partsupp PRIMARY"
        "partsupp i_ps_partkey"
        "partsupp i_ps_suppkey"
        "customer PRIMARY"
        "customer i_c_nationkey"
        "orders PRIMARY"
        "orders i_o_custkey"
        "orders i_o_orderdate"
        "lineitem PRIMARY"
        "lineitem i_l_orderkey"
        "lineitem i_l_partkey"
        "lineitem i_l_suppkey"
        "lineitem i_l_partkey_suppkey"
        "lineitem i_l_shipdate"
        "lineitem i_l_commitdate"
        "lineitem i_l_receiptdate"
        "nation i_n_regionkey"
        "nation PRIMARY"
        "region PRIMARY"
)
for table_index in "${table_indexes[@]}"
do
    ti=($table_index)
    table=${ti[0]}
    index=${ti[1]}
    SQL="select count(*) from ${table} force index(${index})"
    echo "$MYSQL -e '$SQL'"
    $MYSQL -e "$SQL"
done
```

- HTAP性能测试中，并未对TPC-H各个表建立二级索引，具体请参见PolarDB MySQL引擎HTAP性能。故该步骤不适用于HTAP性能测试场景。

7. 运行查询。

```
#!/usr/bin/env bash
host=$1
port=$2
user=$3
password=$4
database=$5
resfile=$6
echo "start test run at"`date "+%Y-%m-%d %H:%M:%S"`|tee -a ${resfile}.out
for (( i=1; i<=22;i=i+1 ))
do
queryfile="Q"${i}".sql"
start_time=`date "+%s.%N"`
echo "run query ${i}"|tee -a ${resfile}.out
mysql -h ${host}  -P${port} -u${user} -p${password} $database -e" source $queryfile;" |
tee -a ${resfile}.out
end_time=`date "+%s.%N"`
start_s=${start_time%.*}
start_nanos=${start_time#*.}
end_s=${end_time%.*}
end_nanos=${end_time#*.}
if [ "$end_nanos" -lt "$start_nanos" ];then
        end_s=$(( 10#$end_s -1 ))
        end_nanos=$(( 10#$end_nanos + 10 ** 9))
fi
time=$(( 10#$end_s - 10#$start_s )).`printf "%03d\n" $(( (10#$end_nanos - 10#$start_nan
os)/10**6 ))`
echo ${queryfile} "the "${j}" run cost "${time}" second start at"`date -d @$start_time
"+%Y-%m-%d %H:%M:%S"`" stop at"`date -d @$end_time "+%Y-%m-%d %H:%M:%S"` >> ${resfile}.
time
done
```

后续步骤

- 并行查询性能测试结果请参见并行查询性能。
- HATP性能测试结果请参见PolarDB MySQL引擎 HTAP性能。

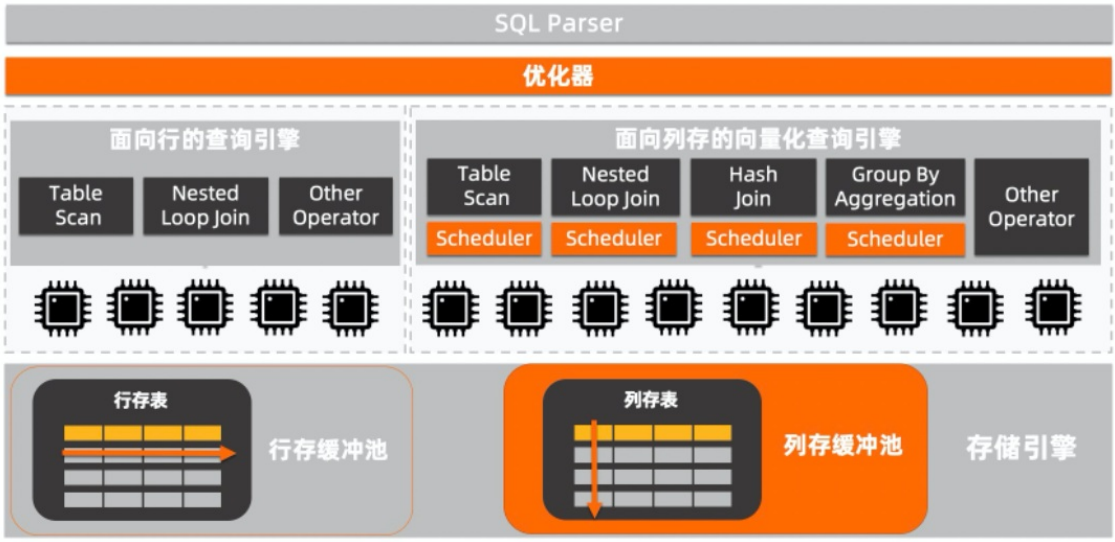
1.2. PolarDB MySQL引擎 HTAP性能

本文档基于TPC-H测试PolarDB MySQL引擎8.0.1版本集群版开启HTAP功能时的OLAP负载性能。

PolarDB MySQL引擎HTAP介绍

PolarDB MySQL引擎引入列存索引来增强OLAP场景大数据量复杂查询的处理能力。依托于列存索引，PolarDB MySQL引擎实现了一体化的实时事务处理和实时数据分析的能力，成为一站式HTAP数据库产品解决方案。

列存索引特性在PolarDB MySQL引擎中的功能架构图如下：



更多信息请参见列存索引。

注意事项

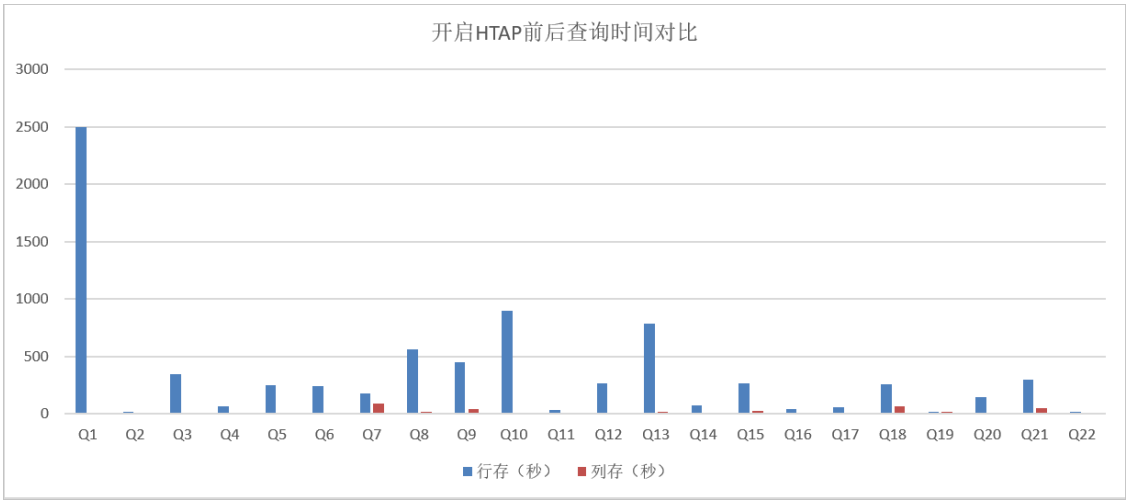
- HTAP功能按需开启，在同一集群内实现了OLTP和OLAP计算资源的隔离，开启HTAP功能后可以通过新增只读列存节点来服务业务中的OLAP查询请求。关于如何新增只读列存节点，请参见添加只读列存节点。
- PolarDB HTAP 性能测试时，需要额外创建一个只读数据库代理自定义地址指向该只读列存节点。请参见OLTP与OLAP业务独立场景下行存和列存手动分流，即这个指向只读列存节点的只读数据库代理自定义地址进行测试。
- HTAP性能测试，无需对TPC-H中各个表建立二级索引。

测试结果

开启HTAP功能（列存）和关闭HTAP功能（行存）的测试结果如下：

查询	行存（秒）	列存（秒）	加速比
Q1	2495.05	5.3	470.8
Q2	13.18	5.54	2.4
Q3	345.56	4.04	85.5
Q4	63.78	2.27	28.1
Q5	251.48	10.97	22.9
Q6	241.7	0.42	575.5
Q7	174.91	88.58	2.0
Q8	560.82	14.51	38.7
Q9	450.68	38.92	11.6

查询	行存（秒）	列存（秒）	加速比
Q10	895.75	8.44	106.1
Q11	30.03	6.17	4.9
Q12	266.14	7.1	37.5
Q13	780.74	16.38	47.7
Q14	72.04	2.56	28.1
Q15	261.77	24.99	10.5
Q16	40.69	4.72	8.6
Q17	57.75	9.09	6.4
Q18	257.66	65.98	3.9
Q19	19.17	18.16	1.1
Q20	143.97	1.71	84.2
Q21	293.99	49.24	6.0
Q22	18.81	2.77	6.8



1.3. 并行查询性能

本文档基于TPC-H测试PolarDB MySQL引擎8.0.1和8.0.2版本集群版的OLAP负载性能，您可以按照本文介绍自行测试对比，快速了解数据库系统的性能。

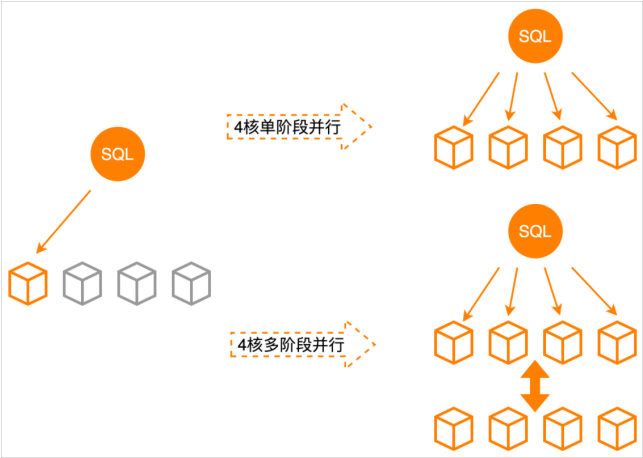
并行查询简介

PolarDB MySQL引擎8.0版本集群版推出并行查询（Parallel Query）框架。当您的查询数据量到达一定阈值，就会自动启动并行查询框架，从而使查询耗时指数级下降。具体请参见并行查询（Parallel Query）。

PolarDB MySQL引擎8.0.1版本中，并行查询在存储层将数据分片到不同的线程上，多个线程并行计算，将结果流水线汇聚到总线程，最后由总线程做简单归并将结果返回给用户，提高查询效率。

PolarDB MySQL引擎8.0.2版本中，并行查询将线性加速能力提升了一个等级，引入了多阶段并行计算能力。基于代价优化为更为灵活的并行执行计划，改进了8.0.1版本并行查询可能存在的Leader单点瓶颈和Worker负载不均衡的问题。具体来说，在存储层仍然将数据分片到不同的线程上，多个线程并行计算，并可能分发到下一个阶段的并行Worker组中，最终并行计算结果汇总到Leader并返回给用户，大幅提升了并行执行能力。

并行查询利用多核CPU的并行处理能力，以8核32 GB独享规格的集群为例，并行查询示意图如下所示。



下文将介绍8.0.1和8.0.2版本并行查询参数取值分别设置为如下表所示时，PolarDB集群负载性能测试方法（OLAP）与执行结果。关于如何配置参数，具体操作步骤请参见设置集群参数和节点参数。

- 8.0.1

参数	默认值	优化后的值
loose_parallel_degree_policy	REPLICA_AUTO	TYPICAL
loose_max_parallel_degree	0	16

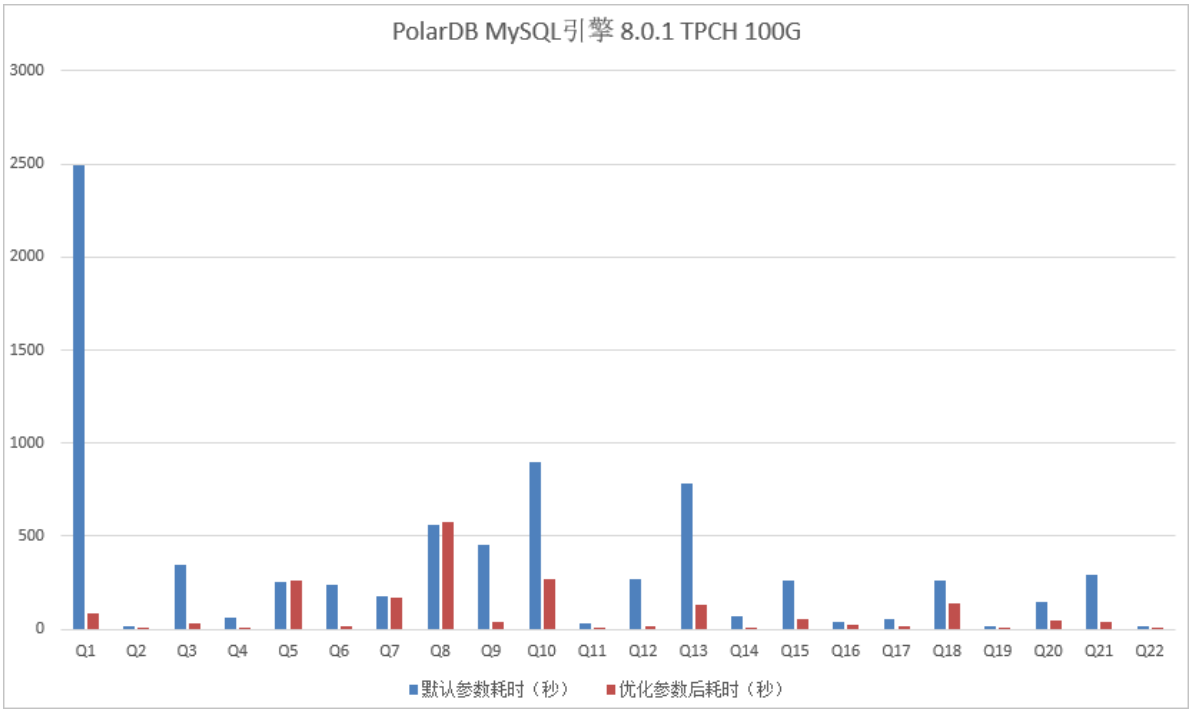
- 8.0.2

参数	默认值	优化后的值
loose_parallel_degree_policy	REPLICA_AUTO	TYPICAL
loose_parallel_query_switch	''	'force_run_using_planning_dop=on'
loose_optimizer_switch	''	'hash_join_cost_based=off'
loose_max_parallel_degree	0	16

执行结果

- 8.0.1版本的测试结果如下所示：

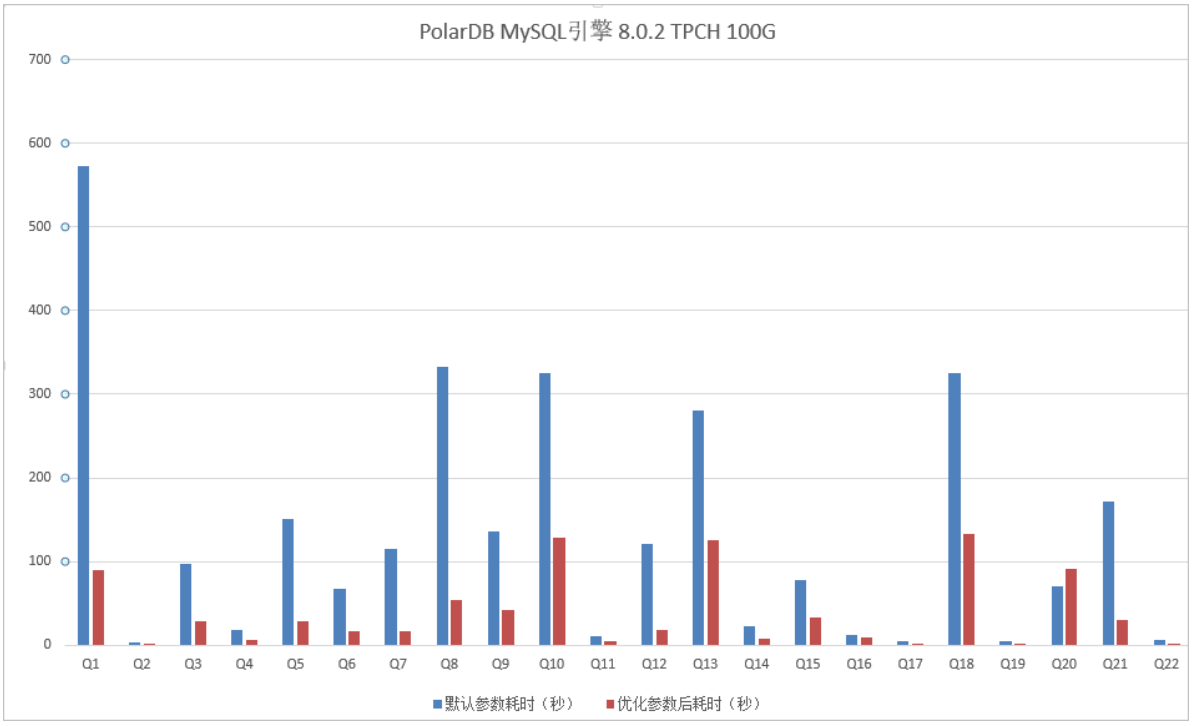
查询	默认值耗时（秒）	优化后的值耗时（秒）
Q1	2495.05	86.63
Q2	13.18	1.46
Q3	345.56	28.67
Q4	63.78	6.26
Q5	251.48	258.55
Q6	241.7	16.4
Q7	174.91	171.85
Q8	560.82	572.63
Q9	450.68	42.17
Q10	895.75	270.29
Q11	30.03	11.6
Q12	266.14	17.65
Q13	780.74	127.1
Q14	72.04	7.57
Q15	261.77	56.55
Q16	40.69	24.29
Q17	57.75	13.04
Q18	257.66	136.79
Q19	19.17	1.52
Q20	143.97	49.72
Q21	293.99	37.79
Q22	18.81	2.15



- 8.0.2版本的测试结果如下所示：

查询	默认值耗时（秒）	优化后的值耗时（秒）
Q1	573.34	89.65
Q2	3.62	0.66
Q3	96.89	28.52
Q4	18.66	6.41
Q5	150.93	28.93
Q6	68.13	17.13
Q7	114.56	17.39
Q8	333.36	54.32
Q9	136.26	42.15
Q10	325.19	128.55
Q11	11.49	4.53
Q12	121.68	17.54
Q13	281.1	126.2
Q14	23.34	7.59
Q15	77.22	33.02

查询	默认值耗时（秒）	优化后的值耗时（秒）
Q16	12.15	8.96
Q17	4.51	1.13
Q18	325.05	133.07
Q19	5.37	1.5
Q20	70.31	90.75
Q21	171.81	29.87
Q22	6.05	2.16



2.性能白皮书（OLTP）

2.1. 性能测试方法（OLTP）

本文档介绍以SysBench工具测试PolarDB MySQL引擎集群OLTP负载性能，您可以按照本文介绍自行测试对比，快速了解数据库系统的性能。

测试工具

SysBench是一个跨平台且支持多线程的模块化基准测试工具，用于评估系统在运行高负载的数据库时相关核心参数的性能表现。使用SysBench是为了绕过复杂的数据库基准设置，甚至在没有安装数据库的前提下，快速了解数据库系统的性能。

测试环境

- 测试的ECS和PolarDB MySQL引擎均在同一地域、同一可用区。
- 网络类型均为VPC网络。

> **说明** ECS实例和PolarDB MySQL引擎集群需保证在同一个VPC中。

- 测试用PolarDB MySQL引擎集群如下：
  - 只读、只写以及读写性能测试使用的是两节点集群（一主一只读），多个只读节点性能将依次使用一主一只读到一主八只读的集群进行测试。
  - 使用的连接串为集群地址，如何查看PolarDB MySQL引擎集群地址请参见申请集群地址和主地址。
  - 为了充分利用数据库集群资源，性能白皮书测试数据是在已升级的数据库代理规格上进行测试，数据库代理规格均升级到**标准配置*4**。具体升级说明请参见升级数据库代理的规格。
- 测试用ECS实例信息如下：
  - 实例规格为ecs.c5.4xlarge。
  - 实例所使用的镜像为CentOS 7.0 64位。
  - 测试用ECS实例的数量，根据集群规格来定，8核规格使用1个ECS实例，16核规格使用2个ECS实例，以此类推。

测试场景

- 对不同独享规格的一主一只读节点集群的只读、只写、读写性能以及一主多只读节点集群的只读性能进行OLTP性能测试。
- 对不同通用规格的一主一只读节点集群的只读、只写、读写性能进行OLTP性能测试。

衡量指标

- TPS（Transactions Per Second）：即数据库每秒执行的事务数，以COMMIT成功次数为准。
- QPS（Queries Per Second）：即数据库每秒执行的SQL数（含INSERT、SELECT、UPDATE、DELETE等）。

安装SysBench

1. 在ECS中执行如下命令安装SysBench。

```
yum install gcc gcc-c++ autoconf automake make libtool bzr mysql-devel git mysql
git clone https://github.com/akopytov/sysbench.git
##从Git中下载sysbench
cd sysbench
##打开sysbench目录
git checkout 1.0.18
##切换到sysbench 1.0.18版本
./autogen.sh
##运行autogen.sh
./configure --prefix=/usr --mandir=/usr/share/man
make
##编译
make install
```

2. 执行如下命令配置SysBench client，使内核可以利用所有的CPU核数处理数据包（默认设置为2核），同时减少CPU核数之间的上下文切换。

```
sudo sh -c 'for x in /sys/class/net/eth0/queues/rx-*; do echo ffffffff>$x/rps_cpus; don
e'
```

> **说明** `ffffffff` 表示使用32个核。请根据实际配置修改，例如ECS为8核，则输入 `ff` 。

```
sudo sh -c "echo 32768 > /proc/sys/net/core/rps_sock_flow_entries"
sudo sh -c "echo 4096 > /sys/class/net/eth0/queues/rx-0/rps_flow_cnt"
sudo sh -c "echo 4096 > /sys/class/net/eth0/queues/rx-1/rps_flow_cnt"
```

测试方法

> **说明** 以下测试方法中以PolarDB MySQL引擎集群版为例进行说明。

1. 获取PolarDB MySQL引擎集群地址和端口。具体操作请参见申请集群地址和主地址。
2. 关闭PolarDB MySQL引擎集群地址的**主库是否接受读**功能，具体操作请参见配置数据库代理。
3. 在ECS上执行如下命令，以在PolarDB MySQL引擎集群中创建数据库 `testdb` 为例。

```
mysql -h XXX -P XXX -u XXX -p XXX -e 'create database testdb'
```

> **说明** 请将本命令和后续步骤命令中的 `XXX` 替换为PolarDB MySQL引擎集群的集群地址、端口号、用户名和密码，具体参数说明如下。

参数	说明
`-h`	PolarDB MySQL引擎集群的集群地址。
`-P`	PolarDB MySQL引擎集群的端口号。
`-u`	PolarDB MySQL引擎集群的用户名。
`-p`	上述用户名对应的密码。

4. 使用SysBench测试PolarDB MySQL引擎集群版一主一只读节点的只读性能，整个过程将持续10分钟。

```
sysbench --db-driver=mysql --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-p
assword=XXX --mysql-db=sbtest --table_size=25000 --tables=250 --events=0 --time=600  ol
tp_read_only prepare
##准备数据
sysbench --db-driver=mysql --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-p
assword=XXX --mysql-db=sbtest --table_size=25000 --tables=250 --events=0 --time=600  -
-threads=XXX --percentile=95 --range_selects=0 --skip-trx=1 --report-interval=1 oltp_re
ad_only run
##运行workload
sysbench --db-driver=mysql --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-p
assword=XXX --mysql-db=sbtest --table_size=25000 --tables=250 --events=0 --time=600  -
-threads=XXX --percentile=95 --range_selects=0 oltp_read_only cleanup
##清理
```

5. 使用SysBench测试PolarDB MySQL引擎集群版一主一只读节点的写入性能，整个过程将持续10分钟。

```
sysbench --db-driver=mysql --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-p
assword=XXX --mysql-db=sbtest --table_size=25000 --tables=250 --events=0 --time=600  ol
tp_write_only prepare
##准备数据
sysbench --db-driver=mysql --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-p
assword=XXX --mysql-db=sbtest --table_size=25000 --tables=250 --events=0 --time=600  -
-threads=XXX --percentile=95 --report-interval=1 oltp_write_only run
##运行workload
sysbench --db-driver=mysql --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-p
assword=XXX --mysql-db=sbtest --table_size=25000 --tables=250 --events=0 --time=600  -
-threads=XXX --percentile=95  oltp_write_only cleanup
##清理
```

6. 使用SysBench测试PolarDB MySQL引擎集群版一主一只读节点的混合读写性能，整个过程将持续10分钟。

```
sysbench --db-driver=mysql --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-p
assword=XXX --mysql-db=sbtest --table_size=250000 --tables=25 --events=0 --time=600  ol
tp_read_write prepare
##准备数据
sysbench --db-driver=mysql  --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-
password=XXX --mysql-db=sbtest --table_size=250000 --tables=25 --events=0 --time=600
--threads=XXX --percentile=95 --report-interval=1 oltp_read_write run
##运行workload
sysbench --db-driver=mysql  --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-
password=XXX --mysql-db=sbtest --table_size=250000 --tables=25 --events=0 --time=600
--threads=XXX --percentile=95  oltp_read_write cleanup
##清理
```

7. 使用SysBench测试PolarDB MySQL引擎集群版一主多只读节点的只读性能（对一主一只读节点到一主八只读节点的集群依次测试）。

```
sysbench --db-driver=mysql --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-p
assword=XXX --mysql-db=sbtest --table_size=25000 --tables=250 --events=0 --time=600  ol
tp_read_only prepare
##准备数据
sysbench --db-driver=mysql --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-p
assword=XXX --mysql-db=sbtest --table_size=25000 --tables=250 --events=0 --time=600   -
-threads=XXX --percentile=95 --report-interval=1 oltp_read_only  --db-ps-mode=disable
--skip-trx=1  run
##运行workload
sysbench --db-driver=mysql --mysql-host=XXX --mysql-port=XXX --mysql-user=XXX --mysql-p
assword=XXX --mysql-db=sbtest --table_size=25000 --tables=250 --events=0 --time=600   -
-threads=XXX --percentile=95  oltp_read_only cleanup
##清理
```

后续步骤

- PolarDB MySQL引擎8.0.2版本测试结果请参见PolarDB MySQL引擎8.0.2版本性能（集群版）。
- PolarDB MySQL引擎8.0.1版本测试结果请参见PolarDB MySQL引擎8.0.1版本性能（集群版）。
- PolarDB MySQL引擎5.7版本测试结果请参见PolarDB MySQL 5.7性能（集群版）。
- PolarDB MySQL引擎5.6版本测试结果请参见PolarDB MySQL 5.6性能（集群版）。
- PolarDB MySQL引擎对比RDS MySQL测试结果请参见PolarDB MySQL引擎与RDS MySQL性能对比。

2.2. PolarDB MySQL引擎8.0.2版本性能（集群版）

本文介绍PolarDB MySQL引擎8.0.2版本集群版的OLTP负载性能测试结果。

PolarDB MySQL引擎8.0.2版本优化增强了并行查询和Hash Join能力。

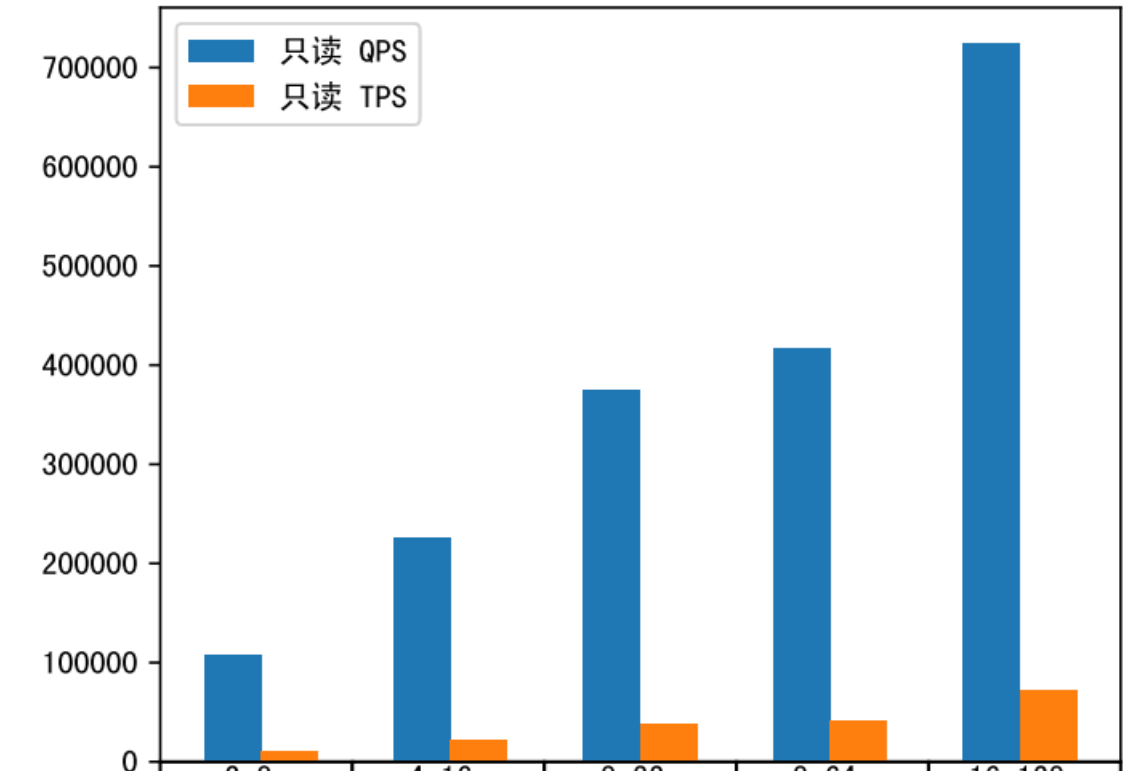
- 并行查询：并行查询重磅升级到2.0版本，对复杂分析查询进行了更全面的支持。
  - 多阶段的执行框架，SQL语句被拆解为多个计算算子。每个算子可以分布到多个worker上并行完成，算子间以流水线方式高效执行；
  - 支持全算子并行；
  - 算子并行方式多样化，如shuffle join、一阶段group aggregation、基于代价决定并行方式等。更多信息请参见并行查询（Parallel Query）。
- Hash Join：Hash Join的并行查询支持简单并行Hash Join、并行Hash Join和Shuffle Hash Join。更多信息请参见Hash Join的并行执行。

> **说明** 具体测试步骤请参见性能测试方法（OLTP）。

独享规格

- **主节点+单只读节点**

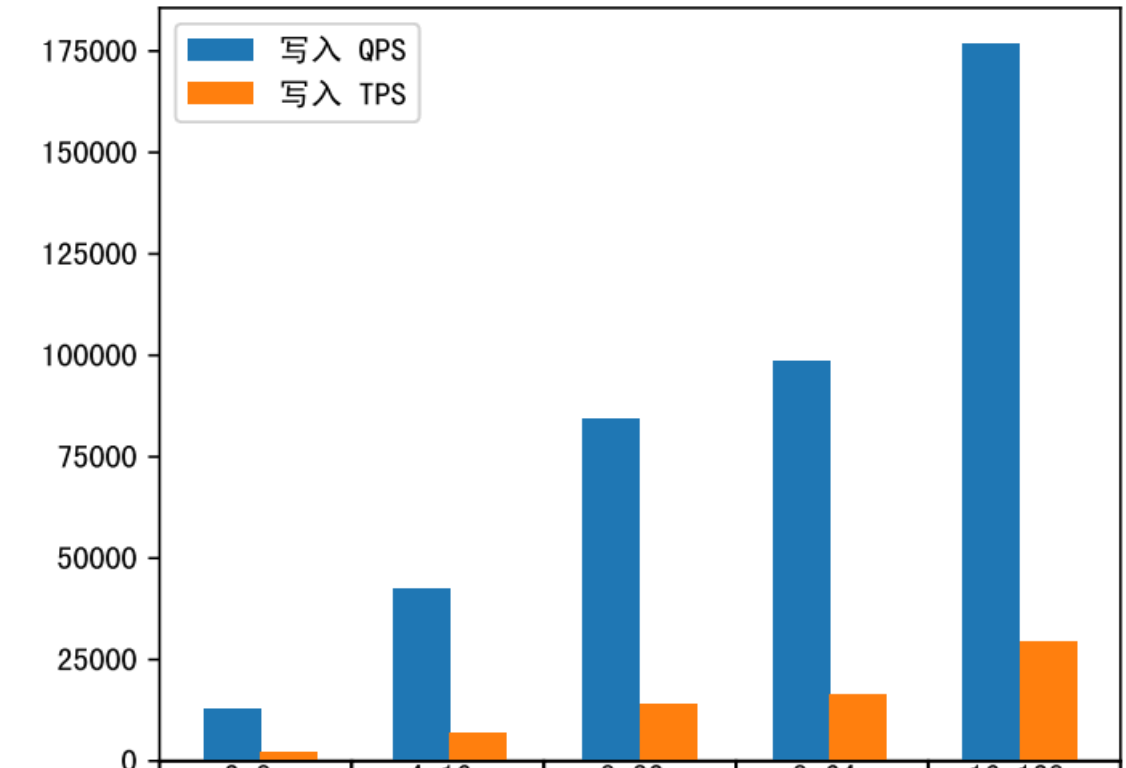
- 各规格只读性能测试结果



	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	64	128	512	512	512
只读 QPS	107812.56	226548.03	374547.44	416608.06	724369.37
只读 TPS	10781.26	22654.8	37454.75	41660.8	72436.93

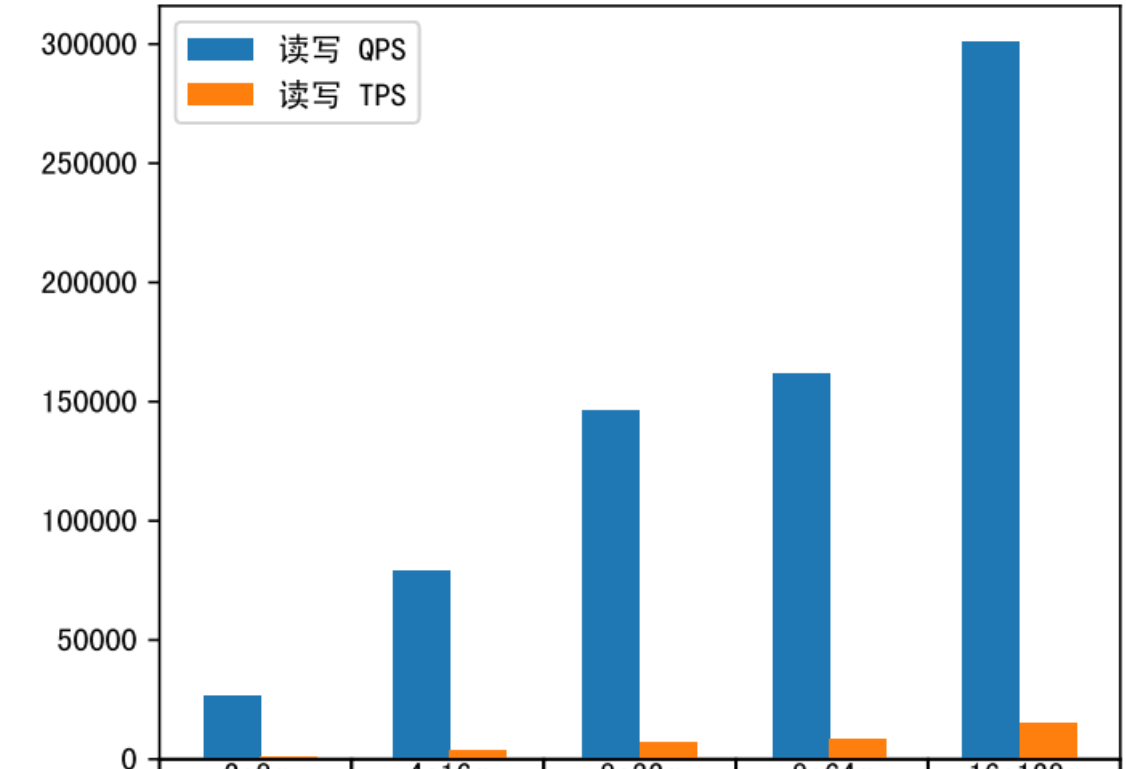
说明 上述测试中未开启所有范围的查询，即设置了range_selects参数为0，故性能较高。

- 各规格写入性能测试结果



	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	128	256	256	256	256
写入 QPS	12755.17	42369.39	84506.44	98563.51	176802.71
写入 TPS	2125.86	7061.57	14084.41	16427.25	29467.13

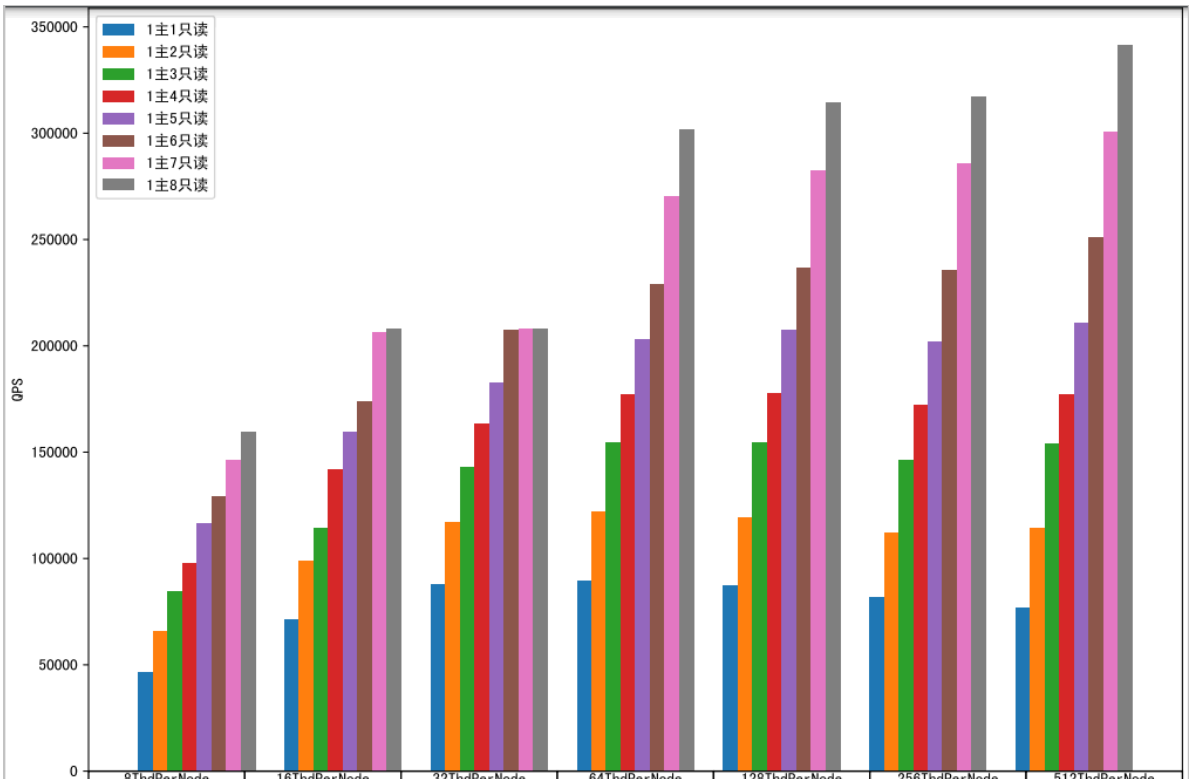
- 各规格混合读写性能测试结果



	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	256	512	1024	1024	1024
读写 QPS	26597.41	79320.18	146350.43	161591.61	300974.82
读写 TPS	1329.87	3966.01	7317.52	8079.58	15048.74

- **主节点+多只读节点**

只读性能测试结果



	8ThdPerNode	16ThdPerNode	32ThdPerNode	64ThdPerNode	128ThdPerNode	256ThdPerNode	512ThdPerNode
1主1只读	46375.68	71553.85	88066.93	89769.27	87498.04	81613.59	76796.31
1主2只读	65912.62	99059.10	117133.95	122361.34	119496.65	112431.88	114458.09
1主3只读	84890.58	114268.30	143296.38	154696.98	154760.32	146523.80	154068.01
1主4只读	98099.33	141895.77	163608.23	177411.93	177690.14	172346.43	177194.92
1主5只读	116745.41	159371.23	182678.36	203295.71	207277.09	202183.96	210778.83
1主6只读	129531.25	173688.25	207338.30	229294.68	236874.48	235927.28	251262.45
1主7只读	146612.09	206590.52	207879.73	270259.03	282639.67	286000.87	300847.28
1主8只读	159786.06	207909.52	207858.05	301673.81	314603.34	317439.03	341647.88

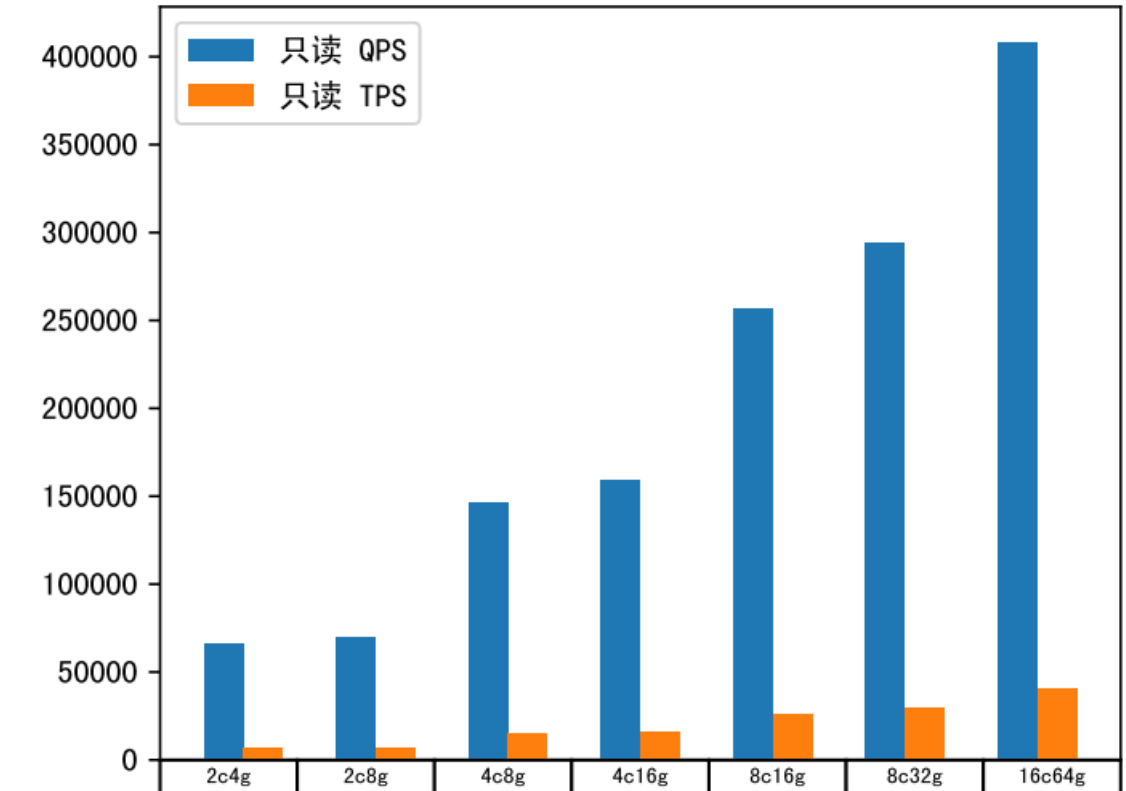
说明

- 上图显示的是使用集群连接地址测试的只读QPS结果。
- 上述测试中未指定range_selects参数，即默认开启了所有范围的查询。
- 上述测试的集群规格为4核16 GB。

通用规格

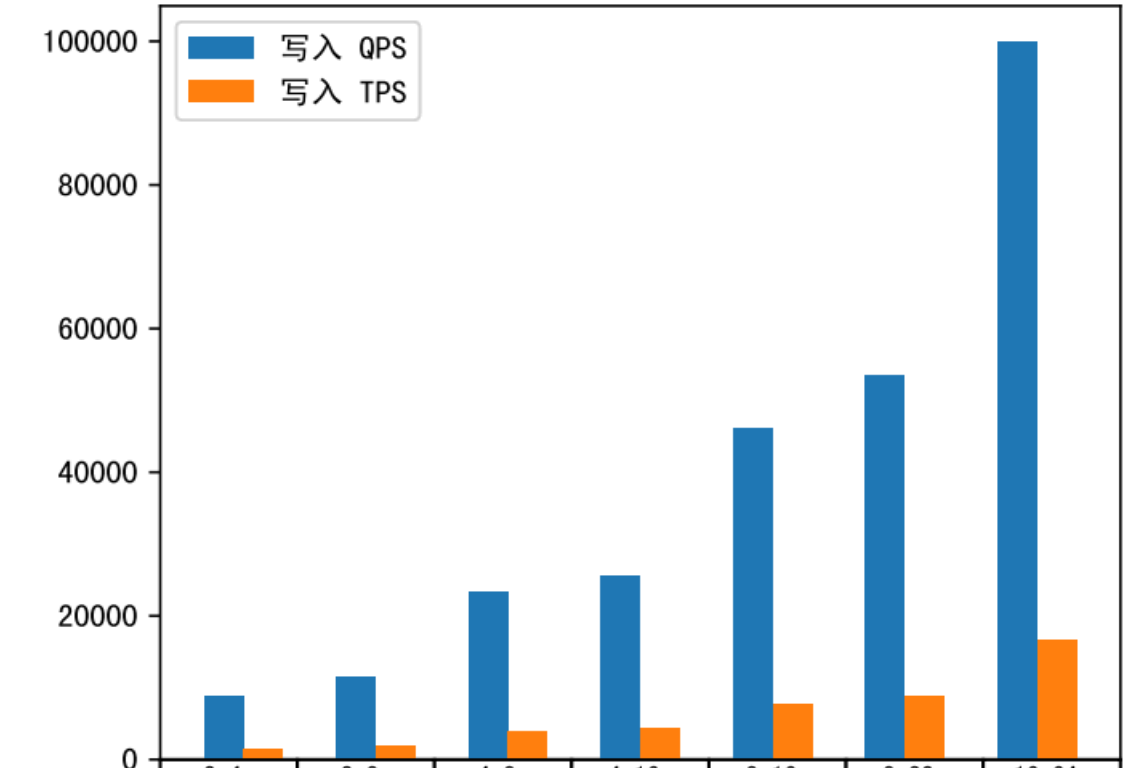
主节点+单只读节点

- 各规格只读性能测试结果



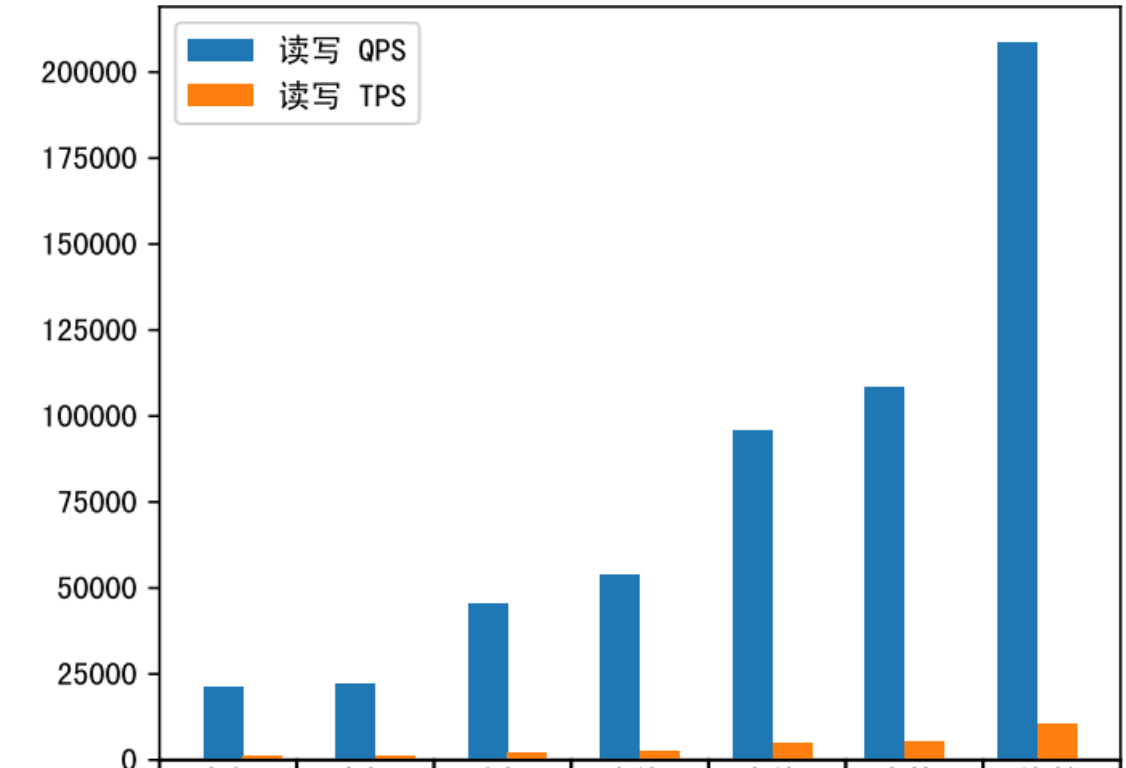
	2c4g	2c8g	4c8g	4c16g	8c16g	8c32g	16c64g
线程数	32	32	64	64	128	128	128
只读 QPS	66307.43	69996.55	146483.78	159040.86	256753.33	293871.38	408016.08
只读 TPS	6630.74	6999.65	14648.38	15904.09	25675.33	29387.14	40801.6

- 各规格写入性能测试结果



	2c4g	2c8g	4c8g	4c16g	8c16g	8c32g	16c64g
线程数	32	16	16	32	32	32	64
写入 QPS	8841.97	11461.45	23306.34	25622.56	46135.52	53594.97	99937.71
写入 TPS	1473.66	1910.24	3884.39	4270.43	7689.25	8932.5	16656.29

- 各规格混合读写性能测试结果



	2c4g	2c8g	4c8g	4c16g	8c16g	8c32g	16c64g
线程数	64	128	64	512	64	64	128
读写 QPS	21210.01	22371.09	45530.95	54062.54	95892.01	108591.99	208591.12
读写 TPS	1060.5	1118.55	2276.55	2703.13	4794.6	5429.6	10429.56

2.3. PolarDB MySQL引擎8.0.1版本性能（集群版）

本文介绍PolarDB MySQL引擎8.0.1版本集群版的OLTP负载性能测试结果。

PolarDB MySQL引擎8.0.1版本优化器支持CBO和RBO两种模式。

- CBO（Cost-Based Optimization）：根据统计信息或实时采样，结合代价模型评估出代价最小的执行方案。
- RBO（Rule-Based Optimization）：使用预定规则锁定执行方案。

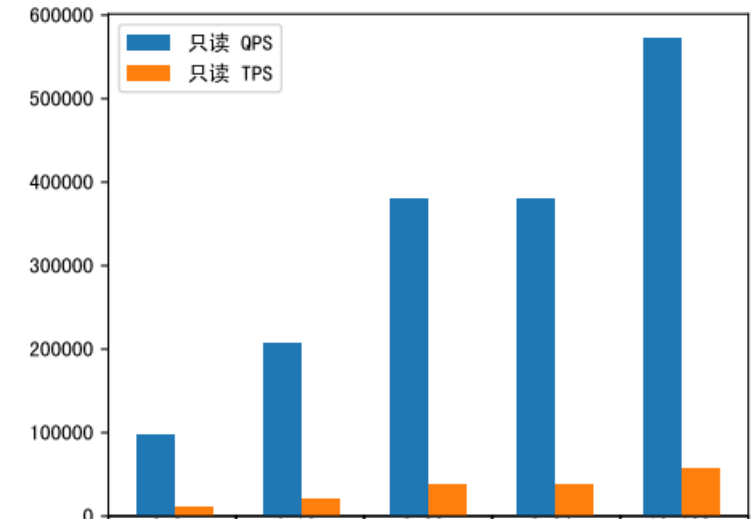
PolarDB MySQL引擎8.0.1版本优化器主要模式为CBO，并支持将RBO作为特定情况下的补充模式。您可以通过ANALYZE命令收集优化器所需要的统计信息，也可以让优化器自动收集统计信息，统计信息包括表大小和记录数、密度向量（density vector）、单列混合直方图（hybrid histogram）、NDV和null ratio。CBO代价模型参数可以通过系统表配置适应软硬件环境，同时可以锁定查询的执行路径或控制CBO搜索范围，防止产出错误的执行方案。

> **说明** 具体测试步骤请参见性能测试方法（OLTP）。

独享规格

- **主节点+单只读节点**

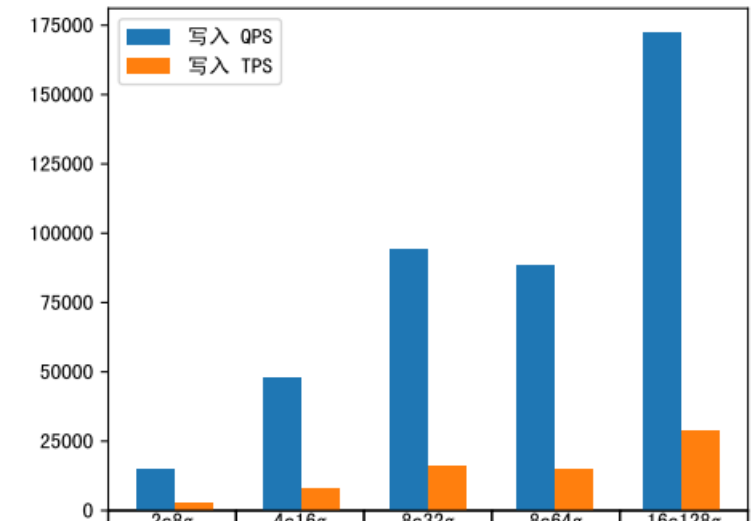
- 各规格只读性能测试结果



	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	128	256	512	512	1024
只读 QPS	96410.3	206659.62	379730.87	380502.71	572733.25
只读 TPS	9641.03	20665.96	37973.09	38050.27	57273.33

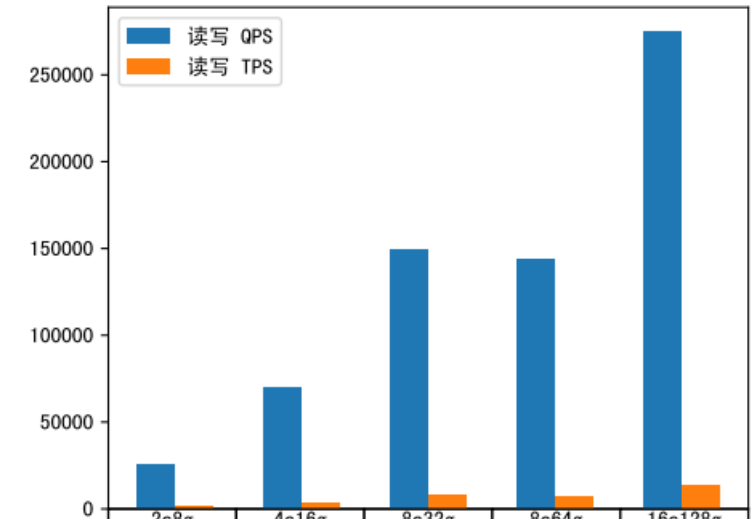
说明 上述测试中未开启所有范围的查询，即设置了range_selects参数为0，故性能较高。

- 各规格写入性能测试结果



	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	128	128	256	256	256
写入 QPS	15021.9	47539.63	94321.03	88057.82	172457.11
写入 TPS	2503.65	7923.27	15720.17	14676.3	28742.85

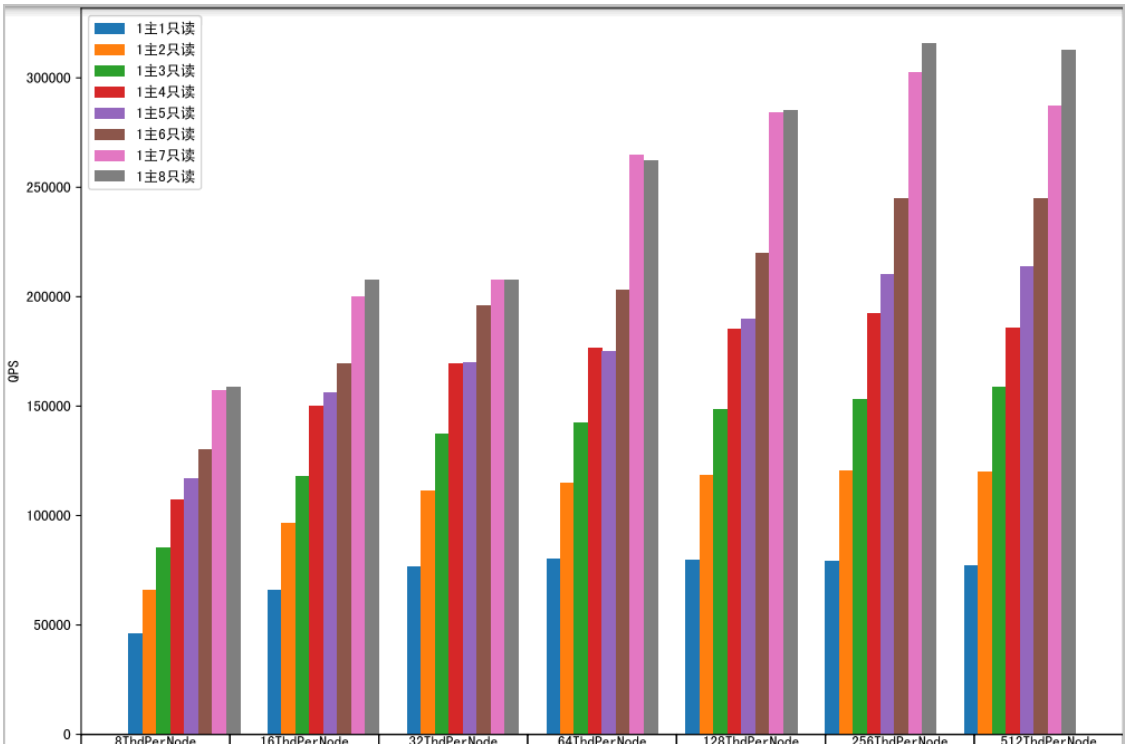
- 各规格混合读写性能测试结果



	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	256	1024	256	512	256
读写 QPS	25730.93	69445.43	148895.91	143372.35	275162.5
读写 TPS	1286.55	3472.28	7444.8	7168.62	13758.13

- **主节点+多只读节点**

  只读性能测试结果



	8ThdPerNode	16ThdPerNode	32ThdPerNode	64ThdPerNode	128ThdPerNode	256ThdPerNode	512ThdPerNode
1主1只读	46313.18	65944.70	76927.89	80522.90	79578.62	79481.48	77480.72
1主2只读	65874.95	96533.42	111192.51	115190.40	118652.75	120331.36	120107.70
1主3只读	85419.29	118281.75	137354.56	142516.11	148612.33	153223.33	158701.68
1主4只读	107335.35	150401.77	169366.97	176855.09	185506.29	192703.76	185824.62
1主5只读	117202.99	156237.19	170281.05	175216.62	189717.96	210509.14	213679.90
1主6只读	130426.55	169468.25	195849.25	203319.27	220140.46	244840.80	245097.87
1主7只读	157295.89	200136.66	207890.45	264793.67	284424.51	302631.53	287264.71
1主8只读	158618.41	207936.56	207856.81	262493.68	285463.91	316003.14	312566.46

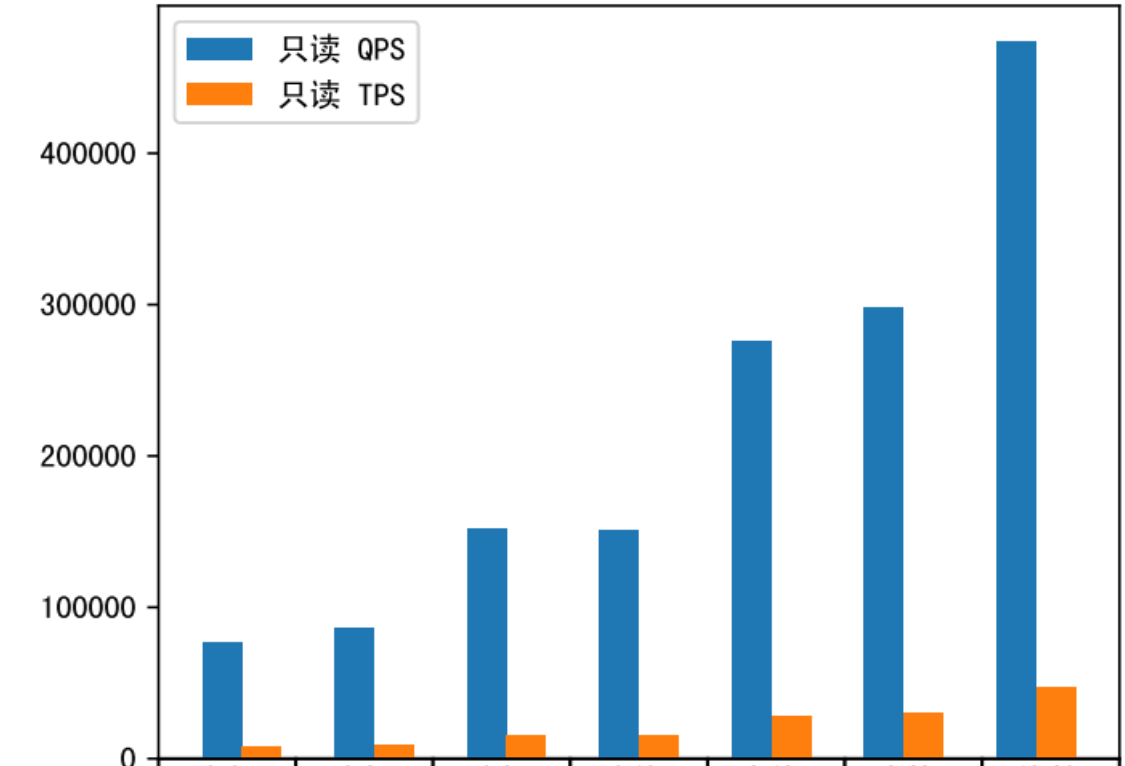
说明

- 上图显示的是使用集群连接地址测试的只读QPS结果。
- 上述测试中未指定range_selects参数，即默认开启了所有范围的查询。
- 上述测试的集群规格为4核16 GB。

通用规格

主节点+单只读节点

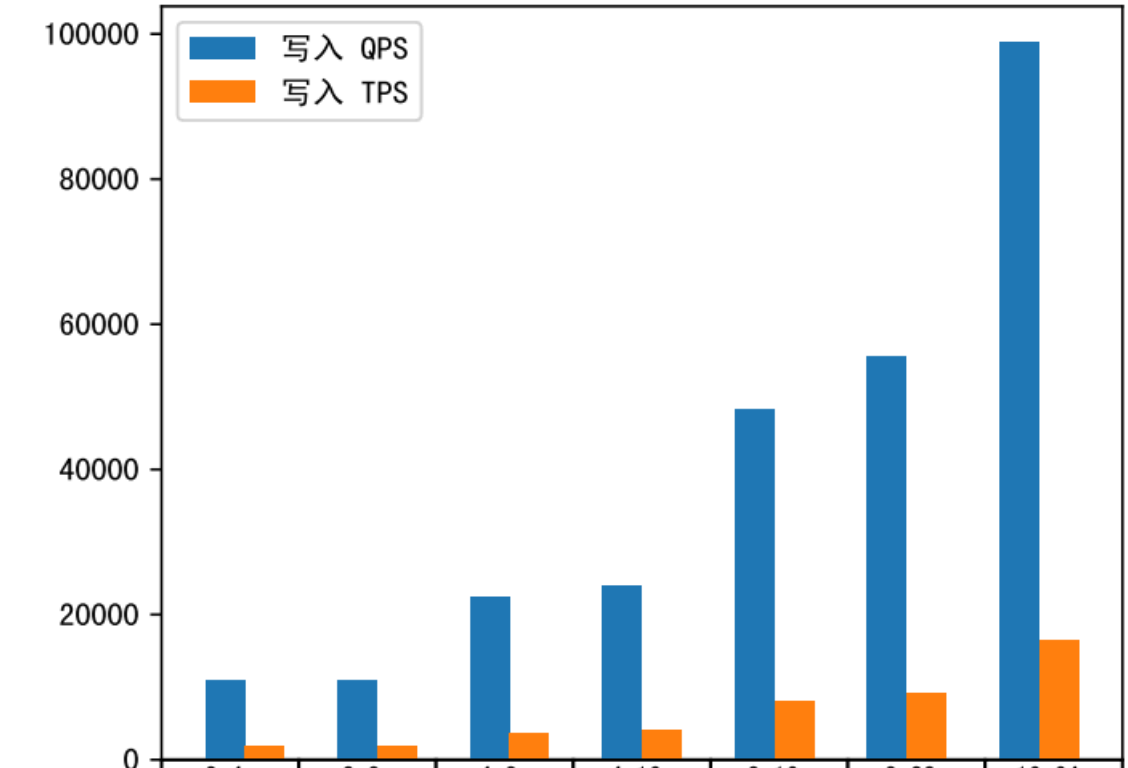
- 各规格只读性能测试结果



	2c4g	2c8g	4c8g	4c16g	8c16g	8c32g	16c64g
线程数	128	128	256	256	256	512	256
只读 QPS	76408.28	85870.0	152259.48	150893.92	275488.59	298054.39	473888.87
只读 TPS	7640.83	8587.0	15225.95	15089.39	27548.86	29805.44	47388.89

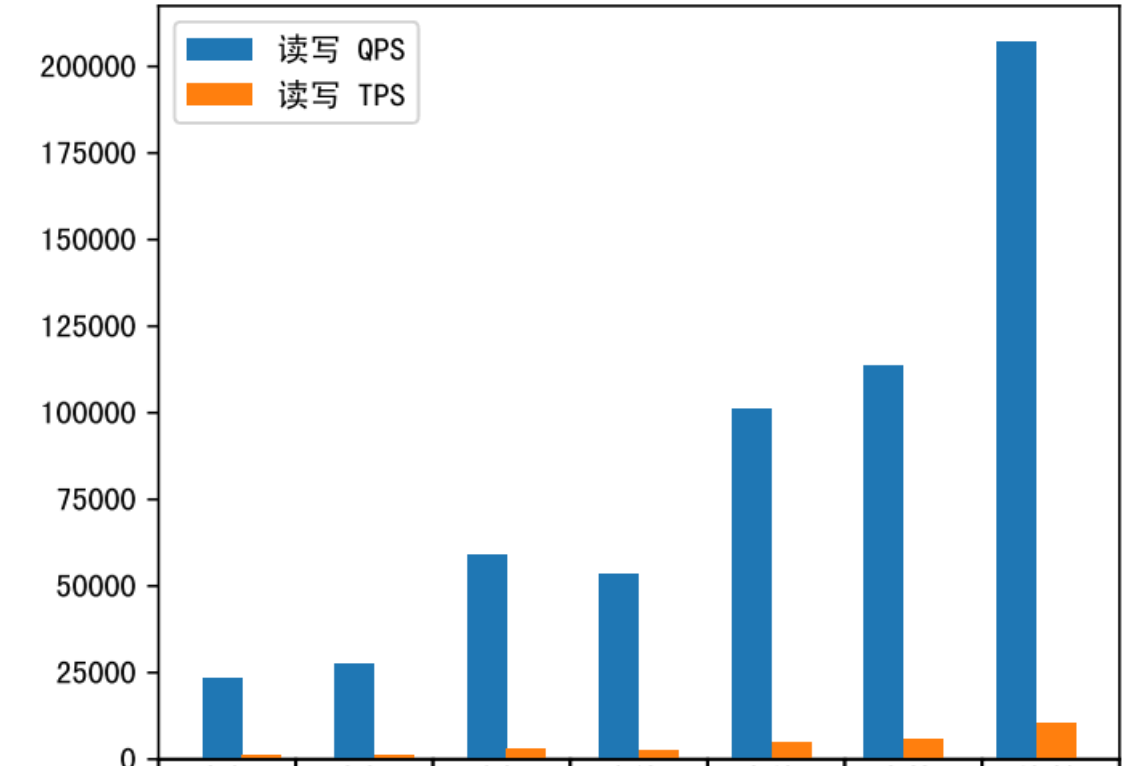
说明 上述测试中未开启所有范围的查询，即设置了range_selects参数为0，故性能较高。

- 各规格写入性能测试结果



	2c4g	2c8g	4c8g	4c16g	8c16g	8c32g	16c64g
线程数	32	16	32	16	32	32	64
写入 QPS	10970.78	10981.63	22506.58	24005.77	48208.44	55512.93	98868.22
写入 TPS	1828.46	1830.27	3751.1	4000.96	8034.74	9252.16	16478.03

- 各规格混合读写性能测试结果



	2c4g	2c8g	4c8g	4c16g	8c16g	8c32g	16c64g
线程数	128	256	512	512	64	128	128
读写 QPS	23574.52	27351.14	59187.2	53315.01	100908.78	113663.65	207112.15
读写 TPS	1178.73	1367.56	2959.36	2665.75	5045.44	5683.18	10355.62

2.4. PolarDB MySQL引擎与RDS MySQL性能对比

本文介绍PolarDB MySQL引擎与RDS MySQL在相同场景中OLTP负载性能的对比结果。

相对RDS MySQL，PolarDB MySQL引擎在如下几方面进行了优化，提高了集群整体性能：

- 采用了领先硬件技术，包括使用3DXpoint存储介质的Optane存储卡、NVMeSSD和RoCE RDMA网络等。
- 基于新的硬件，实现了一整套在用户态运行的IO和网络协议栈，获得了更高的性能、更低的延迟。
- 通过锁优化、IO路径优化、针对大表优化等内核层面的优化，实现了并发场景下的更优性能。

> **说明** 测试所用的产品为：
> - 不同规格的PolarDB MySQL引擎集群版独享规格集群。
> - 不同规格的RDS MySQL高可用版独享型规格实例（本地盘）。
>
> 具体测试方法，请参见性能测试方法（OLTP）。

注意事项

在您对PolarDB和RDS进行性能对比前，请了解以下注意事项，以便能获得比较准确、合理的性能对比结果：

- 使用相同规格配置的PolarDB和RDS进行性能对比。
- 使用相同版本的PolarDB和RDS进行性能对比。

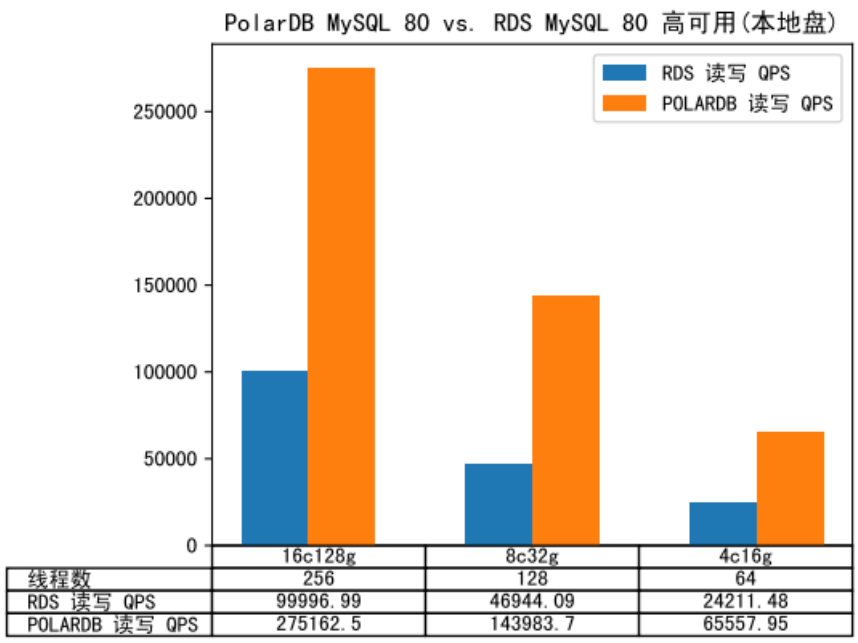
因为不同版本的实现机制不一样，例如PolarDB MySQL引擎8.0.1针对多核数CPU做优化，单独抽象出来Log_writer、log_fluser、log_checkpoint、log_write_notifier等线程，但在CPU核数较少的情况下性能则不如MySQL 5.6或MySQL 5.7。不推荐使用PolarDB MySQL引擎5.6版本和RDS MySQL 5.7或MySQL 8.0进行对比，因为PolarDB MySQL引擎5.6的优化器比较旧，不如新版本。

- 推荐使用模拟线上压力的场景进行实际性能对比，或者使用Sysbench进行对比，这样获得的数据更接近线上实际场景。
- 在对比读性能的时候，不推荐您使用单条SQL进行比较。

  因为PolarDB是计算存储分离的架构，所以单条语句有网络延迟的影响，导致读性能不如RDS。但是线上数据库的缓存命中率基本都在99%以上，因此只有第一次的读会调用I/O，因此读取性能会降低，后续数据都在缓存池（Buffer Pool）中，并不需要调用I/O，因此性能是一样的。
- 在对比写性能的时候，同样不推荐您使用单条SQL进行比较，推荐模拟线上环境进行压力测试。

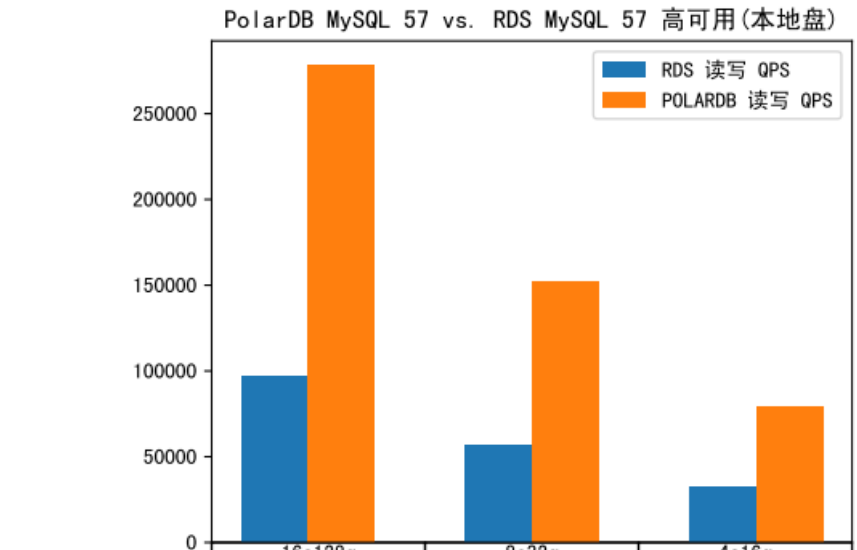
  如果要对比RDS性能，请使用PolarDB（主节点+只读节点）和RDS（主实例+半同步的只读实例）进行对比。这是因为PolarDB的架构在写入数据的时候默认采用Quorum机制，即写入数据时默认写入到三副本里面的大多数（在三个副本中的两个或两个以上写入成功，就认为写操作成功了）。PolarDB已经在存储层面做了数据冗余，并保证三副本强同步高可靠，因此使用RDS MySQL的半同步复制（而不是异步复制）进行对比更合理。

PolarDB MySQL引擎8.0.1对比RDS MySQL 8.0



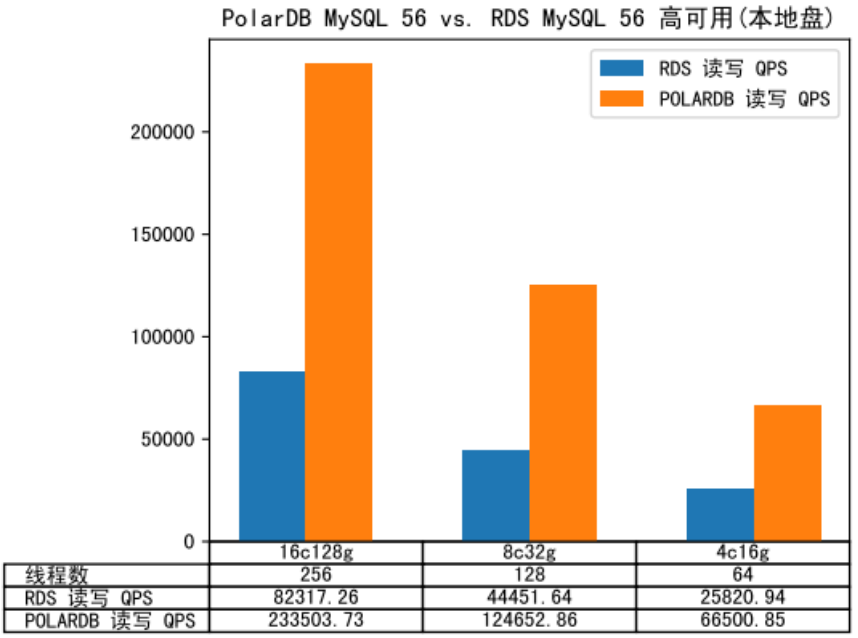
	16c128g	8c32g	4c16g
线程数	256	128	64
RDS 读写 QPS	99996.99	46944.09	24211.48
POLARDB 读写 QPS	275162.5	143983.7	65557.95

PolarDB MySQL引擎5.7对比RDS MySQL 5.7



	16c128g	8c32g	4c16g
线程数	256	128	64
RDS 读写 QPS	97131.79	56175.8	31942.05
POLARDB 读写 QPS	278515.15	151488.12	78920.79

PolarDB MySQL引擎5.6对比RDS MySQL 5.6



	16c128g	8c32g	4c16g
线程数	256	128	64
RDS 读写 QPS	82317.26	44451.64	25820.94
POLARDB 读写 QPS	233503.73	124652.86	66500.85

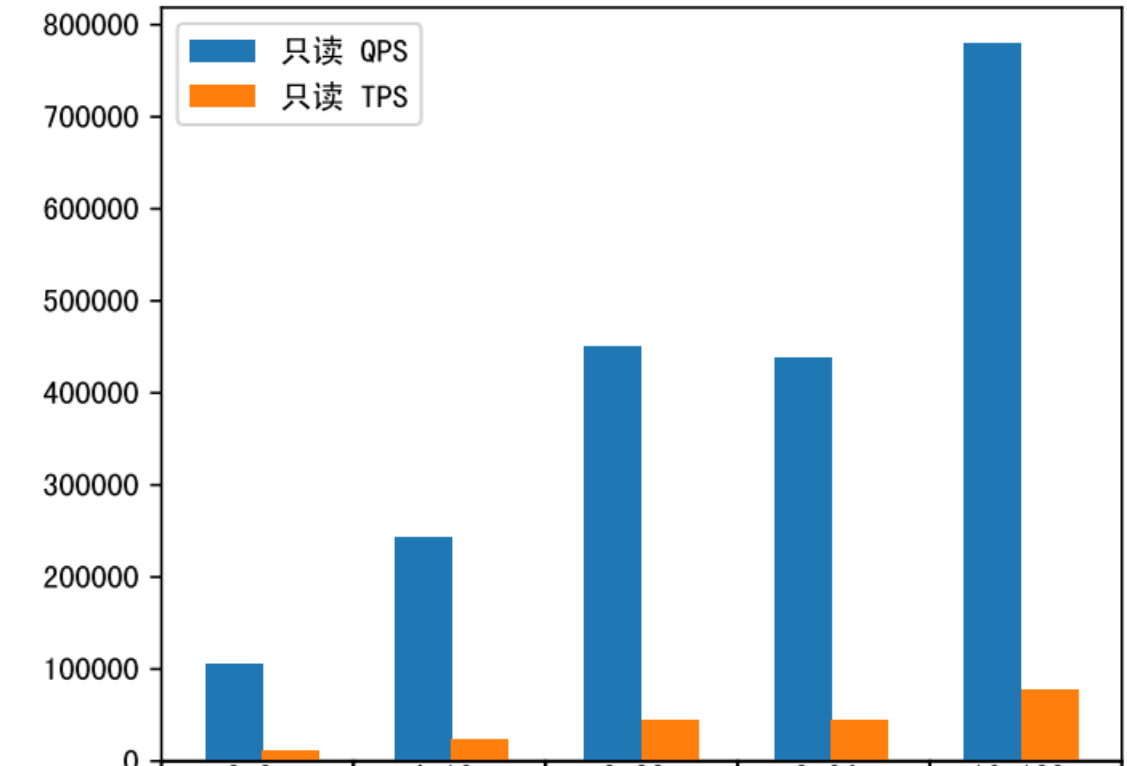
2.5. PolarDB MySQL引擎5.7版本性能（集群版）

本文介绍PolarDB MySQL引擎5.7版本集群版OLTP负载性能测试结果。

说明 具体测试步骤请参见性能测试方法（OLTP）。

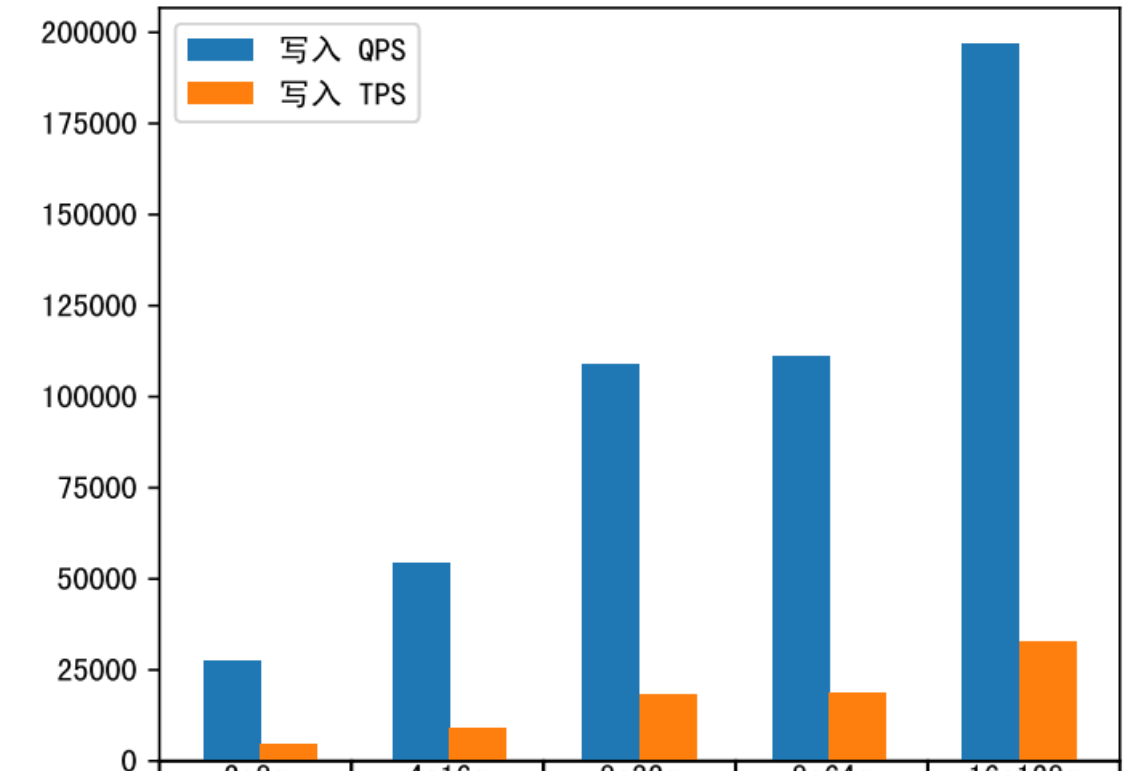
独享规格

- **主节点+单只读节点**
  - 各规格只读性能测试结果



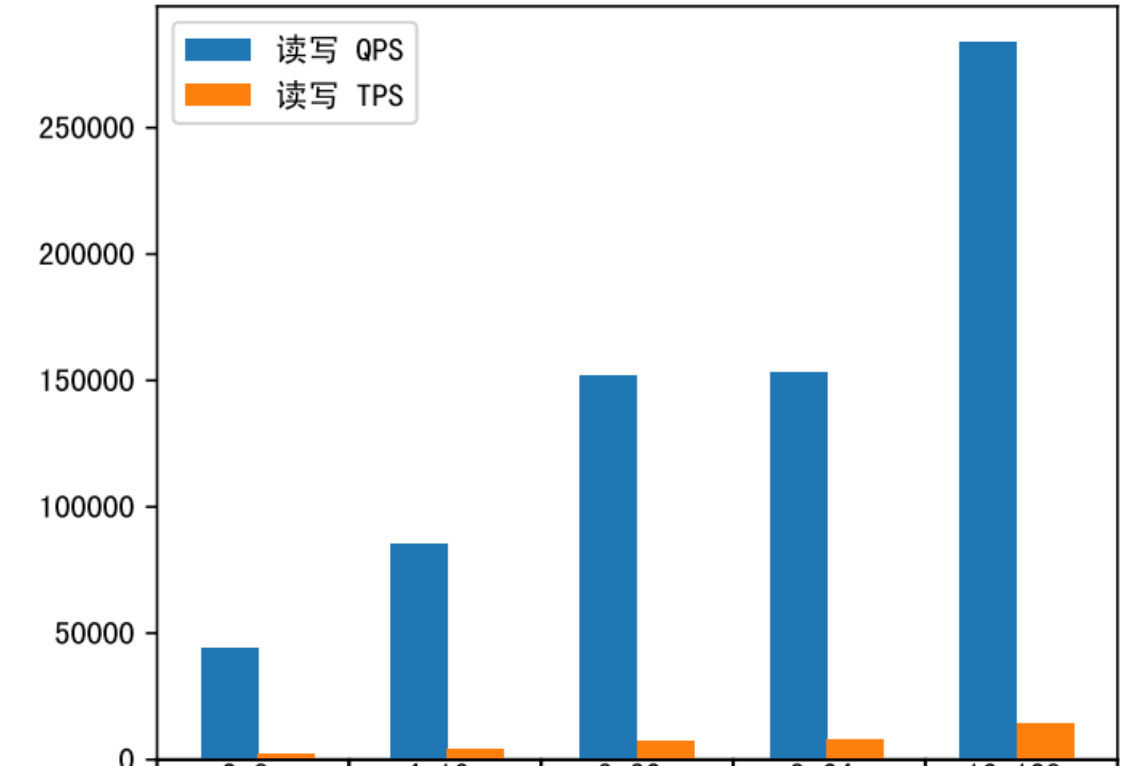
	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	512	256	512	512	512
只读 QPS	104994.93	243710.73	449538.36	437532.14	779728.33
只读 TPS	10499.49	24371.07	44953.84	43753.21	77972.83

- 各规格写入性能测试结果



	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	128	64	128	128	256
写入 QPS	27633.0	54408.35	108832.02	110912.13	196885.72
写入 TPS	4605.5	9068.06	18138.67	18485.36	32814.29

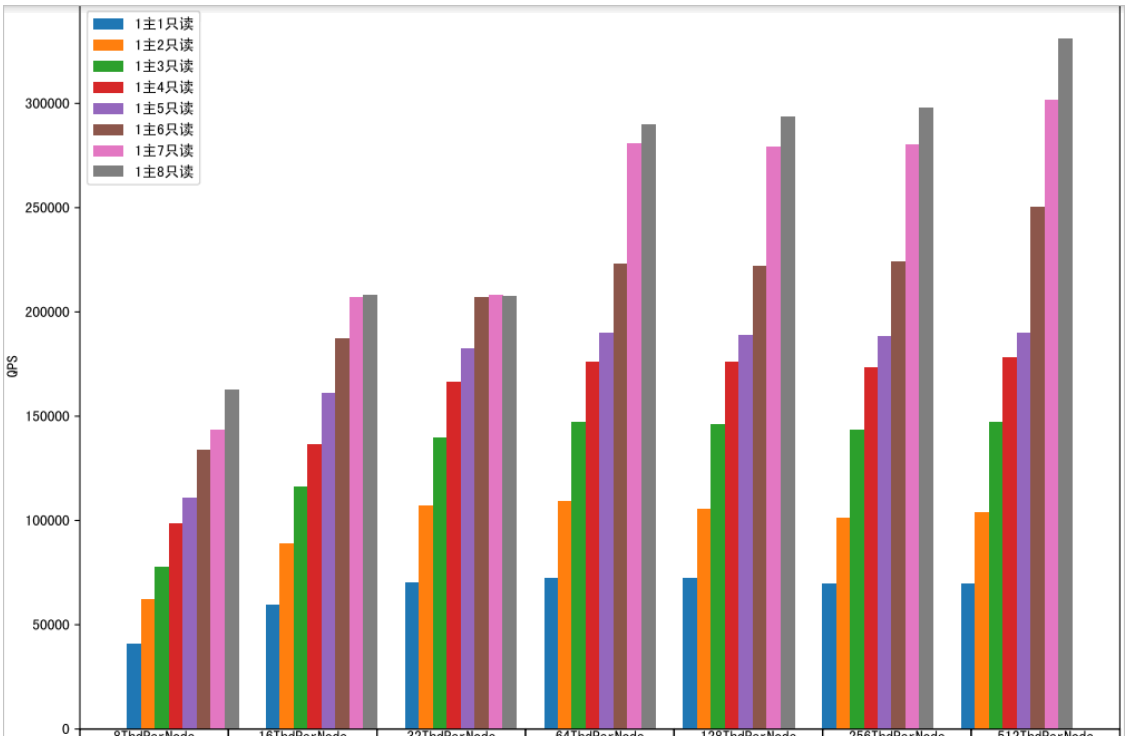
- 各规格混合读写性能测试结果



	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	128	128	256	2048	512
读写 QPS	44262.04	85057.68	151845.61	152952.18	283858.93
读写 TPS	2213.1	4252.88	7592.28	7647.61	14192.95

- **主节点+多只读节点**

只读性能测试结果



	8ThdPerNode	16ThdPerNode	32ThdPerNode	64ThdPerNode	128ThdPerNode	256ThdPerNode	512ThdPerNode
1主1只读	40810.09	59557.95	70371.69	72458.21	72530.58	69617.41	69604.09
1主2只读	62357.91	88841.81	107038.99	109323.13	105459.34	101114.97	103699.22
1主3只读	77784.57	115992.37	139997.38	147414.05	146049.62	143428.12	147317.57
1主4只读	98446.26	136730.19	166743.16	176339.00	176036.31	173224.82	178152.89
1主5只读	111065.09	161263.12	182757.78	189881.62	189046.27	188532.87	189887.12
1主6只读	134026.20	187287.44	207222.52	223044.08	221993.21	224309.89	250410.96
1主7只读	143277.28	207149.00	207914.38	280602.47	279175.15	280135.47	301752.53
1主8只读	162809.09	207950.30	207882.52	289943.75	293507.60	297951.13	331253.15

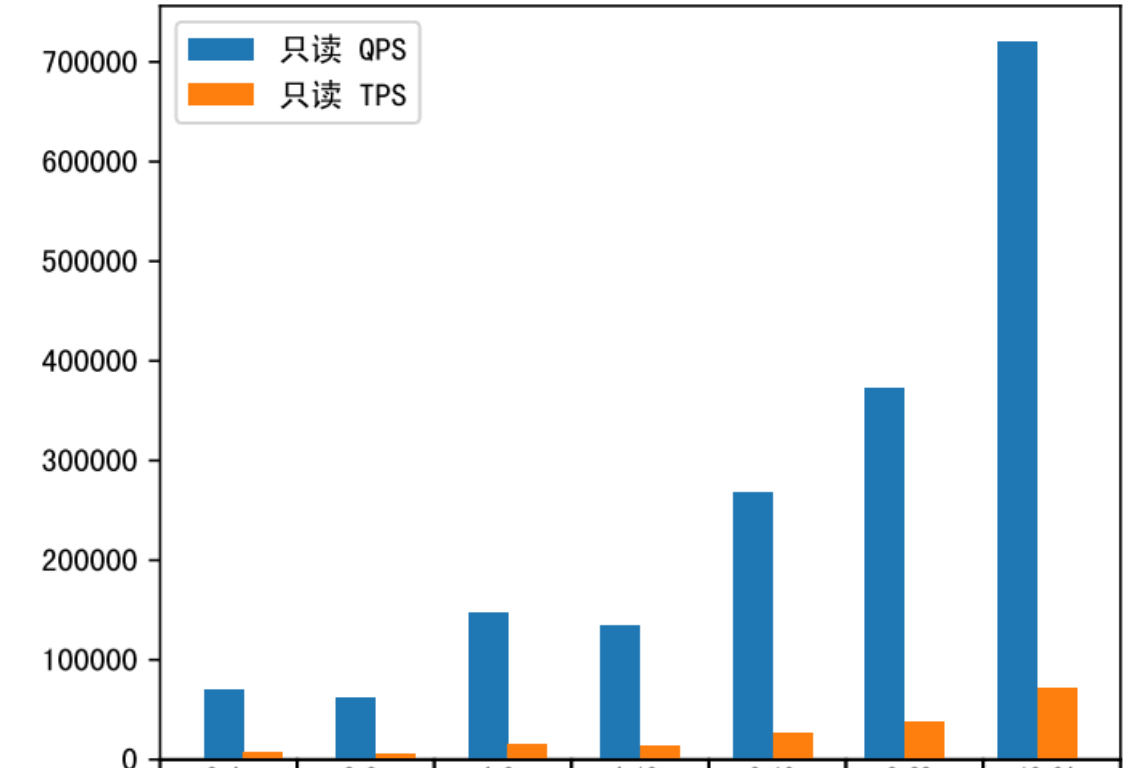
说明

- 上图显示的是使用集群连接地址测试的只读QPS结果。
- 上述测试中未指定range_selects参数，即默认开启了所有范围的查询。
- 上述测试的集群规格为4核16 GB。

通用规格

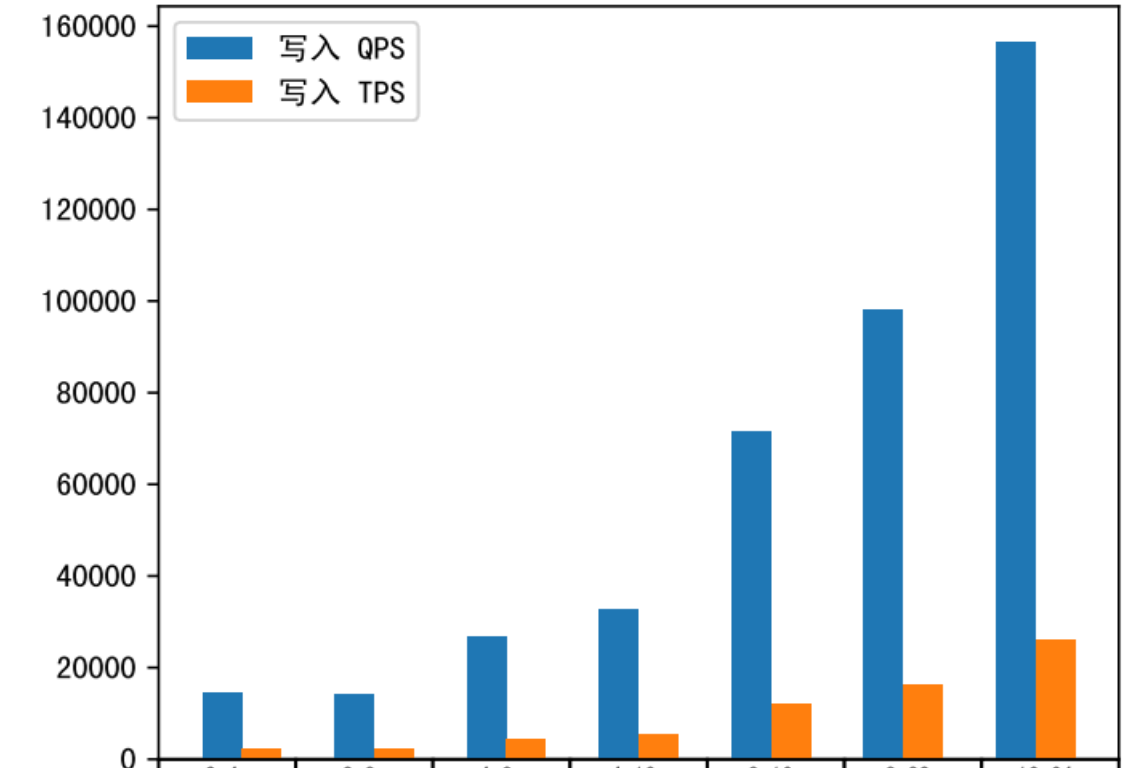
主节点+单只读节点

- 各规格只读性能测试结果



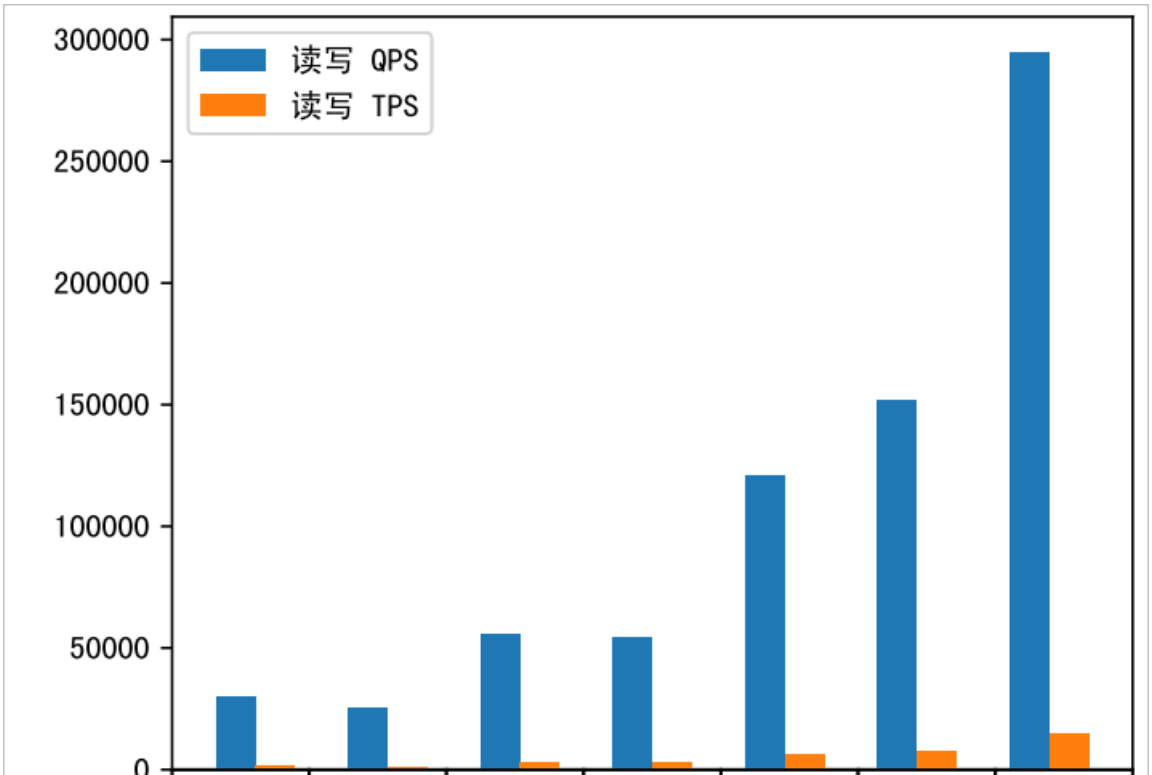
	2c4g	2c8g	4c8g	4c16g	8c16g	8c32g	16c64g
线程数	32	32	64	64	128	256	512
只读 QPS	70514.15	62214.15	146575.58	133643.69	268194.94	372419.72	720218.62
只读 TPS	7051.42	6221.41	14657.56	13364.37	26819.49	37241.97	72021.86

- 各规格写入性能测试结果



	2c4g	2c8g	4c8g	4c16g	8c16g	8c32g	16c64g
线程数	64	128	256	32	64	64	128
写入 QPS	14399.02	14295.58	26699.26	32771.73	71522.12	98013.36	156513.72
写入 TPS	2399.84	2382.6	4449.88	5461.95	11920.35	16335.56	26085.62

- 各规格混合读写性能测试结果



	2c4g	2c8g	4c8g	4c16g	8c16g	8c32g	16c64g
线程数	32	64	64	32	64	64	128
读写 QPS	30187.62	25239.98	55330.91	54259.42	120779.31	151630.55	294721.93
读写 TPS	1509.38	1262.0	2766.55	2712.97	6038.97	7581.53	14736.09

2.6. PolarDB MySQL引擎5.6版本性能（集群版）

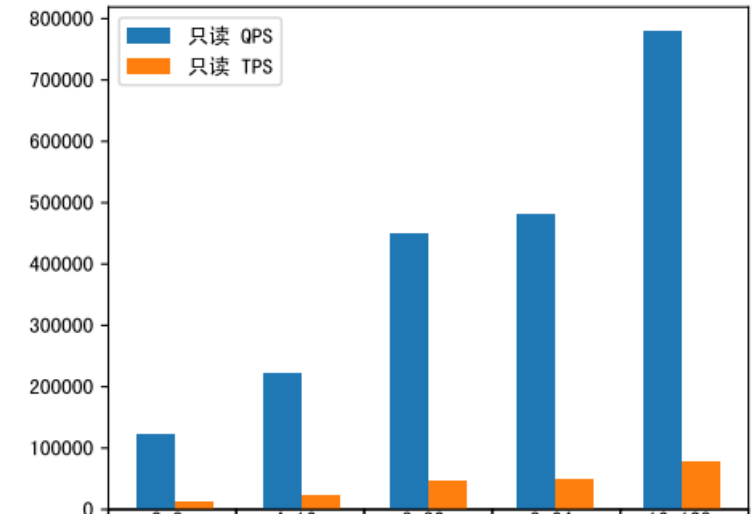
本文介绍PolarDB MySQL引擎5.6版本集群版OLTP负载性能测试结果。

> **说明** 具体测试步骤请参见性能测试方法（OLTP）。

独享规格

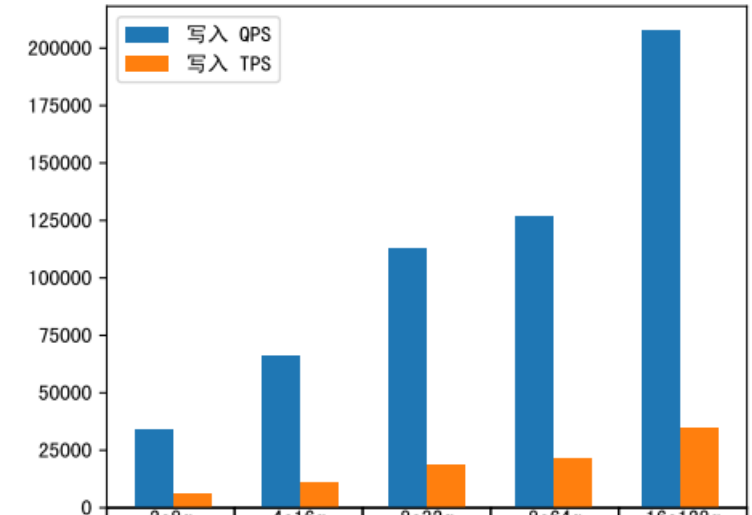
- **主节点+单只读节点**

- 各规格只读性能测试结果



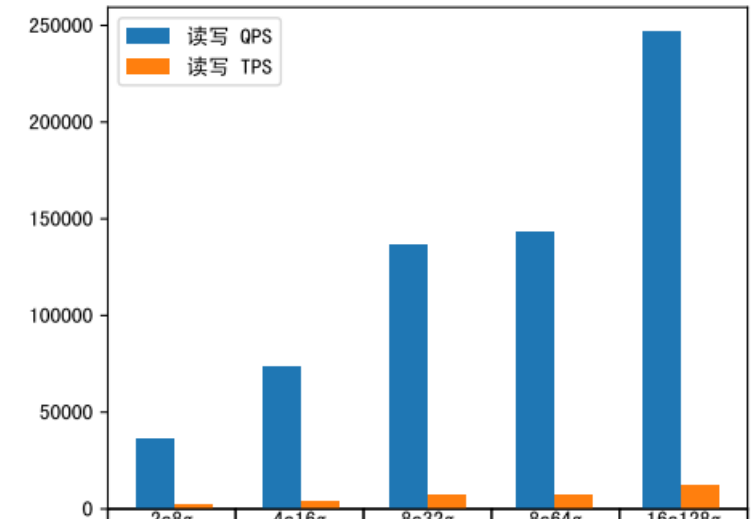
	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	64	256	512	512	512
只读 QPS	122709.11	221856.55	448376.37	480648.41	779927.61
只读 TPS	12270.91	22185.65	44837.63	48064.84	77992.76

- 各规格写入性能测试结果



	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	1024	64	128	256	256
写入 QPS	33645.45	66126.15	112311.65	126286.88	207812.58
写入 TPS	5607.58	11021.02	18718.61	21047.81	34635.43

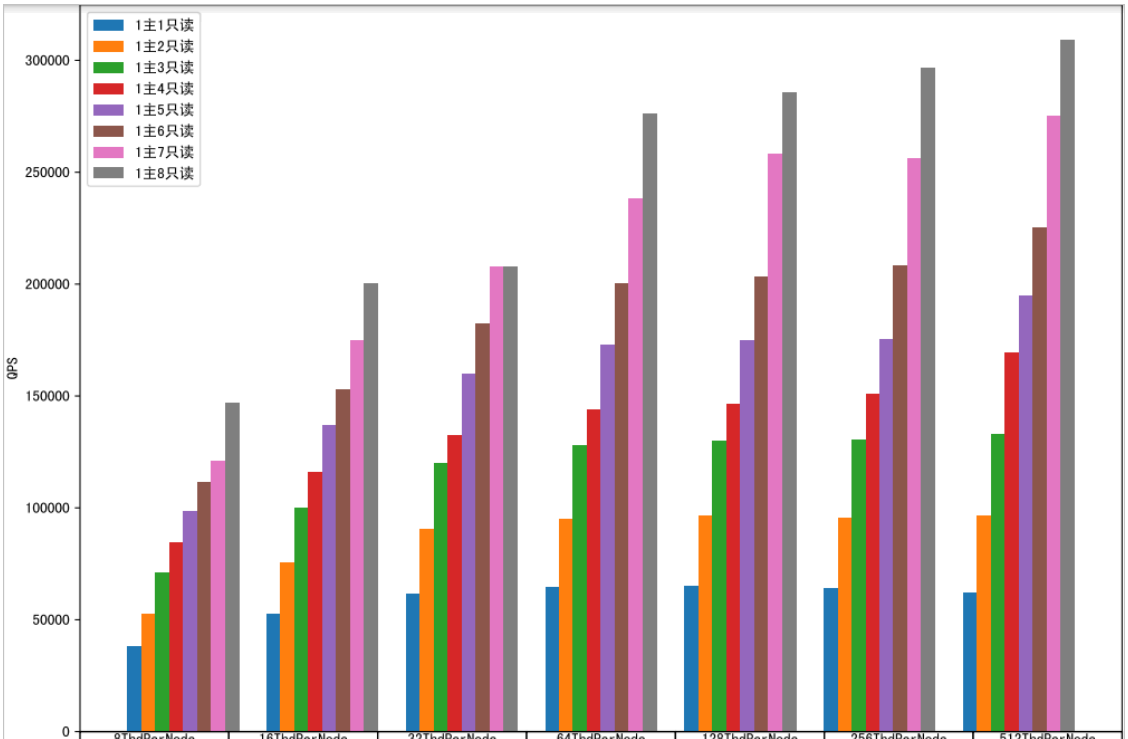
- 各规格混合读写性能测试结果



	2c8g	4c16g	8c32g	8c64g	16c128g
线程数	128	128	256	1024	512
读写 QPS	36261.94	73012.41	136715.53	142728.05	247012.28
读写 TPS	1813.1	3650.62	6835.78	7136.4	12350.62

- **主节点+多只读节点**

  只读性能测试结果



	8ThdPerNode	16ThdPerNode	32ThdPerNode	64ThdPerNode	128ThdPerNode	256ThdPerNode	512ThdPerNode
1主1只读	38165.11	52601.24	61460.87	64597.46	65098.47	64106.80	62269.36
1主2只读	52545.94	75447.29	90349.27	94997.69	96625.73	95504.80	96506.80
1主3只读	71068.31	100062.08	120195.60	128202.47	130236.80	130713.45	133017.93
1主4只读	84647.44	116128.55	132502.69	144035.12	146378.38	151111.51	169291.22
1主5只读	98673.20	136935.80	159866.34	172916.82	175024.83	175494.99	194850.22
1主6只读	111399.29	153161.95	182324.50	200496.06	203280.18	208417.29	225383.34
1主7只读	120858.20	174812.36	207786.41	238221.52	258426.15	256082.43	275473.98
1主8只读	147067.11	200417.56	207840.03	276093.11	285856.01	296640.78	309309.08

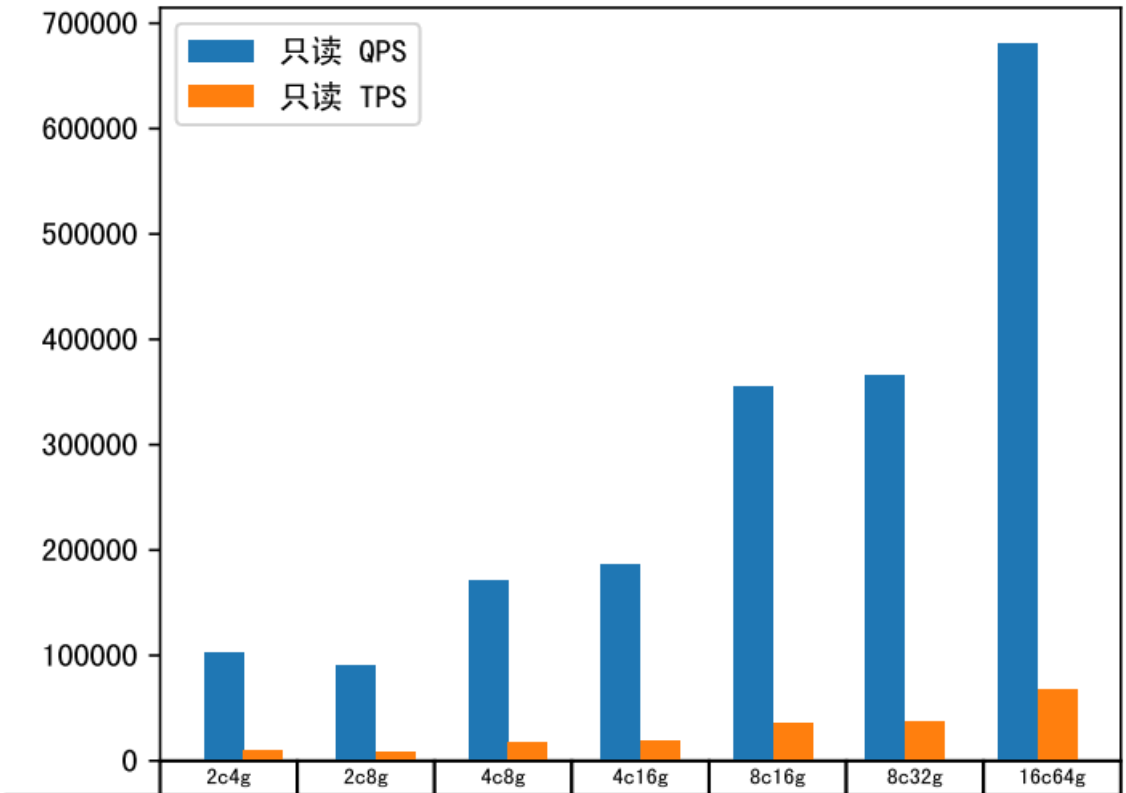
说明

- 上图显示的是使用集群连接地址测试的只读QPS结果。
- 上述测试中未指定range_selects参数，即默认开启了所有范围的查询。
- 上述测试的集群规格为4核16 GB。

通用规格

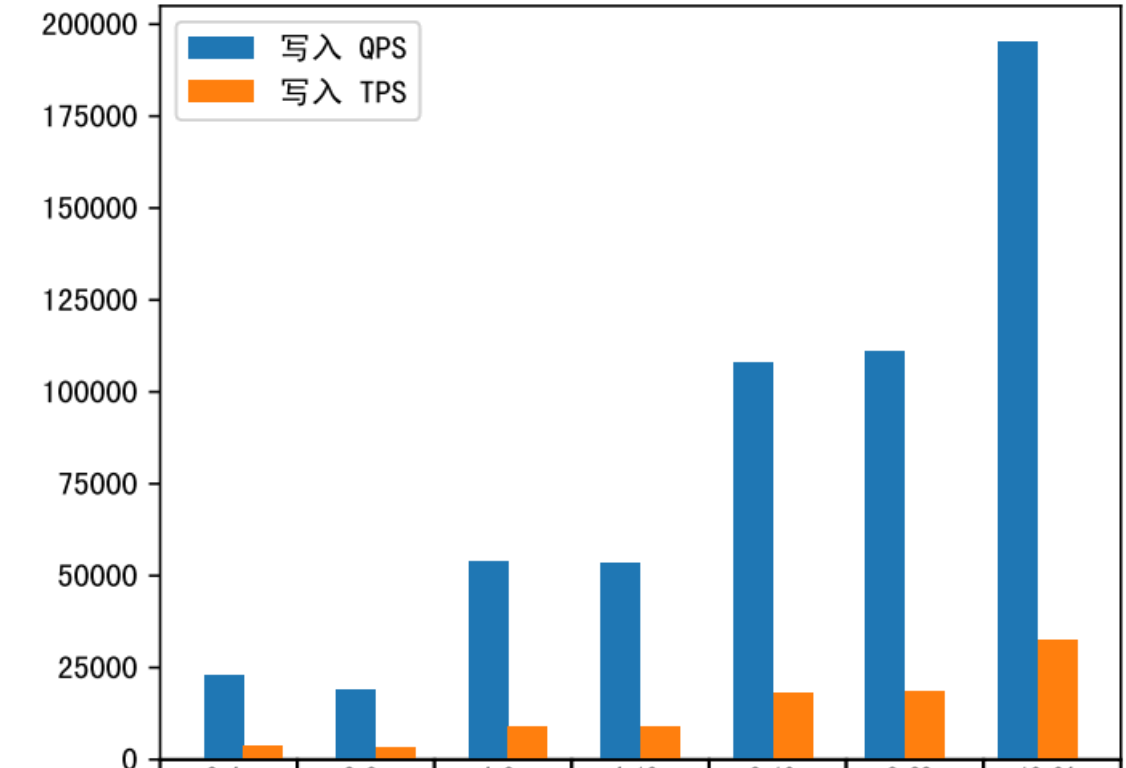
主节点+单只读节点

- 各规格只读性能测试结果



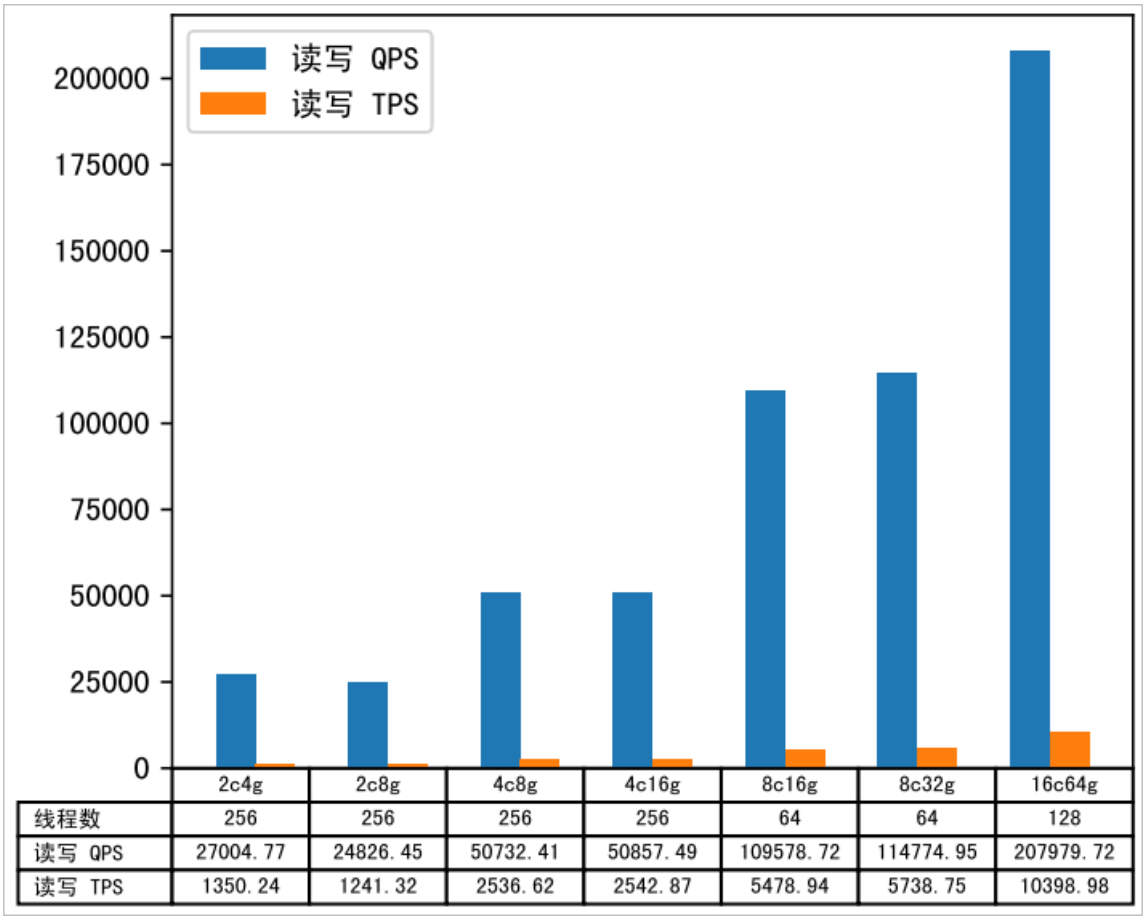
	2c4g	2c8g	4c8g	4c16g	8c16g	8c32g	16c64g
线程数	64	32	64	128	256	256	512
只读 QPS	102573.17	90140.64	171304.12	186309.44	354986.69	365125.59	680724.06
只读 TPS	10257.32	9014.06	17130.41	18630.94	35498.67	36512.56	68072.41

- 各规格写入性能测试结果



	2c4g	2c8g	4c8g	4c16g	8c16g	8c32g	16c64g
线程数	128	32	32	64	64	64	128
写入 QPS	23107.69	19021.9	53819.61	53318.05	108133.21	110957.26	195225.66
写入 TPS	3851.28	3170.32	8969.94	8886.34	18022.2	18492.88	32537.61

- 各规格混合读写性能测试结果



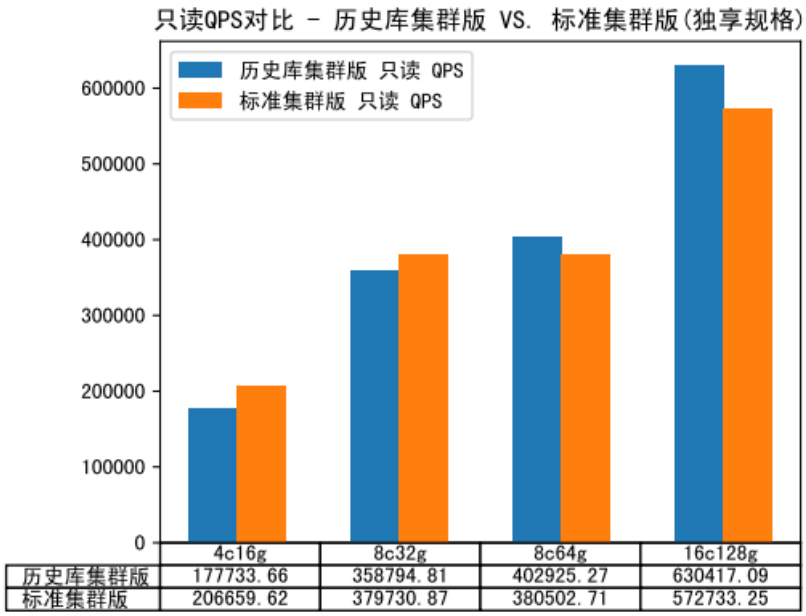
2.7. PolarDB MySQL引擎8.0.1版本历史库与集群版性能对比

本文介绍了PolarDB MySQL引擎8.0.1版本历史库集群版（X-Engine引擎）和集群版（InnoDB引擎）的OLTP负载性能测试结果。

PolarDB MySQL引擎历史库集群版使用阿里云自研存储引擎X-Engine，具备了强大的数据压缩能力，能够满足归档数据库低存储成本的要求，适合大数据量归档存储的场景。经阿里巴巴内部使用历史库的业务验证，在相同原始数据量的情况下，历史库集群版消耗的存储空间仅为InnoDB引擎数据库的1/5~1/3。更多详情，请参见历史库。

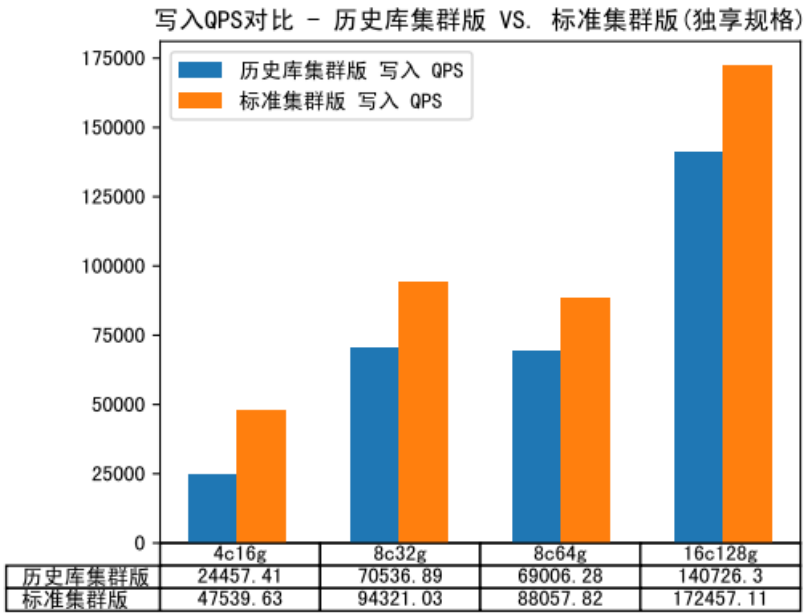
> **说明**
> - 历史库集群版和集群版使用了不同的引擎，因此为保证两种引擎均压测至极限性能，测试相同规格节点时，两种引擎使用的线程数略有差异。
> - 具体测试步骤请参见性能测试方法（OLTP）。

各规格只读性能对比

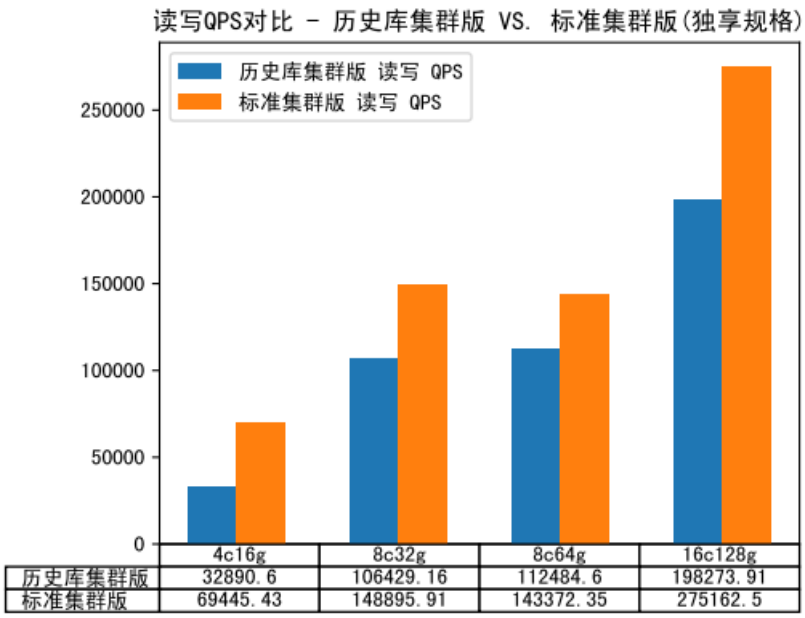


说明 上述测试中未开启所有范围的查询，即设置了range_selects参数为0，故性能较高。

各规格写入性能对比



各规格混合读写性能对比



	4c16g	8c32g	8c64g	16c128g
历史库集群版	32890.6	106429.16	112484.6	198273.91
标准集群版	69445.43	148895.91	143372.35	275162.5