ApsaraDB for RDS

Performance White Paper

MORE THAN JUST CLOUD | Alibaba Cloud

Performance White Paper

For MySQL

Product overview

ApsaraDB for RDS is a stable, reliable, and automatically scalable online database service. Based on the Apsara distributed file system of Alibaba Cloud and the full SSD high-performance storage, RDS provides a complete set of solutions for disaster tolerance, backup, recovery, monitoring, and migration to free you from the burden of database O&M. RDS supports a variety of engines such as MySQL, SQL Server, PostgreSQL, and PPAS which is highly compatible with Oracle.

RDS supports two billing methods: Pay-As-You-Go and Subscription. You can configure the RDS instance specifications according to the service load. In terms of the two billing methods:

- You can upgrade and downgrade the specifications of a Pay-As-You-Go instance at any time.
- You can upgrade the specifications of a Subscription instance at any time, and upgrade and downgrade the specifications upon renewal.

In the case of extreme service load, you can upgrade the instance to the RDS dedicated host specifications to handle unexpected conditions.

Test approach

Test environment

All tests are performed in Zone B of China East 2 (Shanghai).

The ECS instance for testing is Generation II.

The instance configuration is 8-Core 16 GB.

The network type is classic network.

The image for stress testing is CentOS 7.0 (64-bit).

Test tool

SysBench

SysBench is a cross-platform and multithread modular benchmarking tool, which is used to evaluate the performance of core parameters of high-load databases. SysBench helps you quickly know the database system performance without complex database benchmark settings and even with no database installed.

Installation method

In this article, SysBench 0.5 is used for testing. Click Here to download it.

Run the following command to install SysBench.

```
# yum install gcc gcc-c++ autoconf automake make libtool bzr mysql-devel
# unzip sysbench-0.5.zip
# cd sysbench-0.5
# ./autogen.sh
# ./configure --prefix=/usr --mandir=/usr/share/man
# make
# make install
```

Test commands

Prepare data

```
sysbench --num-threads=32 --max-time=3600 --max-requests=999999999 --test= oltp.lua --oltp-table-size=10000000 --oltp-tables-count=64 --db-driver=mysql --mysql-table-engine=innodb --mysql-host= XXXX --mysql-port=3306 --mysql-user= XXXX --mysql-password= XXXX prepare
```

Perform stress testing

```
sysbench --num-threads=32 --max-time=3600 --max-requests=999999999 --test= oltp.lua --oltp-table-size=10000000 --oltp-tables-count=64 --db-driver=mysql --mysql-table-engine=innodb --mysql-host= XXXX --mysql-port=3306 --mysql-user= XXXX --mysql-password= XXXX run
```

Clear the environment

```
sysbench --num-threads=32 --max-time=3600 --max-requests=999999999 --test= oltp.lua --oltp-table-size=10000000 --oltp-tables-count=64 --db-driver=mysql --mysql-table-engine=innodb --mysql-host= XXXX --mysql-port=3306 --mysql-user= XXXX --mysql-password= XXXX cleanup
```

Test model

Database table structure

```
CREATE TABLE `sbtest` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
`k` int(10) unsigned NOT NULL DEFAULT '0',
`c` char(120) NOT NULL DEFAULT '',
`pad` char(60) NOT NULL DEFAULT '',
PRIMARY KEY (`id`),
KEY `k_1` (`k`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

Data format

```
id: 1
k: 3718516
c: 08566691963-88624912351-16662227201-46648573979-64646226163-77505759394-75470094713-41097360717-15161106334-50535565977
pad: 63188288836-92351140030-06390587585-66802097351-49282961843
```

SQL style

Query:

```
SELECT c FROM sbtest64 WHERE id=4957216
SELECT c FROM sbtest43 WHERE id BETWEEN 4573346 AND 4573346+99
SELECT SUM(K) FROM sbtest57 WHERE id BETWEEN 5034894 AND 5034894+99
SELECT DISTINCT c FROM sbtest50 WHERE id BETWEEN 4959831 AND 4959831+99 ORDER BY c
```

Write:

```
INSERT INTO sbtest3 (id, k, c, pad) VALUES (4974042, 4963580, '33958272865-80411528812-36334179010-84793024318-25708692091-43736213170-37853797624-40480626242-32131452190-24509204411', '07716658989-39745043214-17284860193-80004426880-14154945098')
```

Update:

```
UPDATE sbtest11 SET k=k+1 WHERE id=5013989
UPDATE sbtest14 SET c='10695174948-02130015518-68664370682-70336600207-55943744221-72419172189-36252607855-75106351226-86920614936-86254476316' WHERE id=5299388
```

Delete:

```
DELETE FROM sbtest33 WHERE id=5002332
```

Test indicators

TPS

Transactions per second (TPS) indicates the number of transactions processed by the database per

second. The counted transactions are committed transactions.

QPS

Queries per second (QPS) indicates the number of SQL statements (such as INSERT, SELECT, UPDATE, and DELETE) that the database runs per second.

For MySQL 5.6

Instance types

Common instance

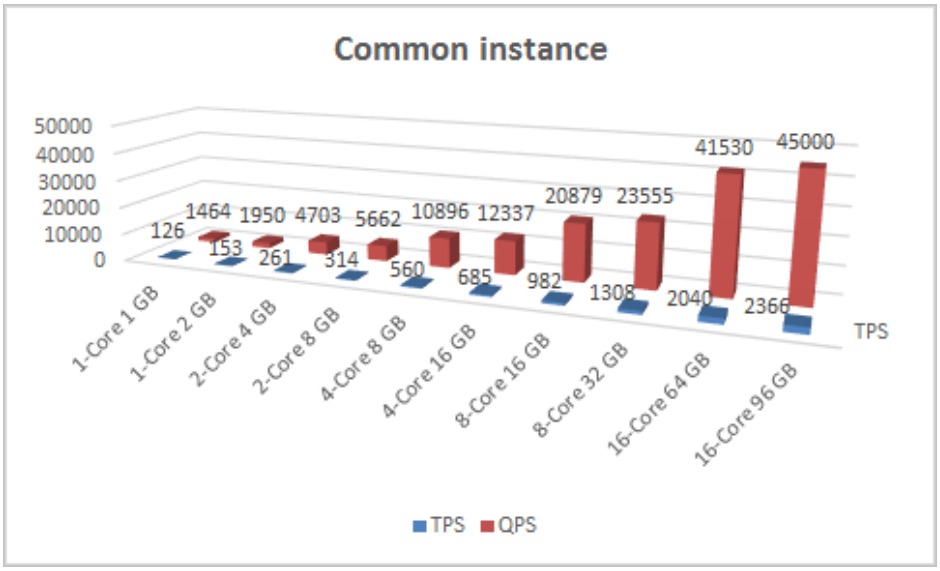
Type ID	Number of CPU cores	Memory (GB)	Number of connections	IOPS	TPS	QPS
rds.mysql.t1.small	1	1	300	600	126	1464
rds.mysql.s1.small	1	2	600	1000	153	1950
rds.mysql.s2.large	2	4	1200	2000	261	4703
rds.mysql.s2.xlarge	2	8	2000	4000	314	5662
rds.mysql.s3.large	4	8	2000	5000	560	10896
rds.mysql.m1.medium	4	16	4000	7000	685	12337
rds.mysql.c1.large	8	16	4000	8000	982	20879
rds.mysql.c1.xlarge	8	32	8000	12000	1308	23555
rds.mysql.c2.xlarge	16	64	16000	14000	2040	41530

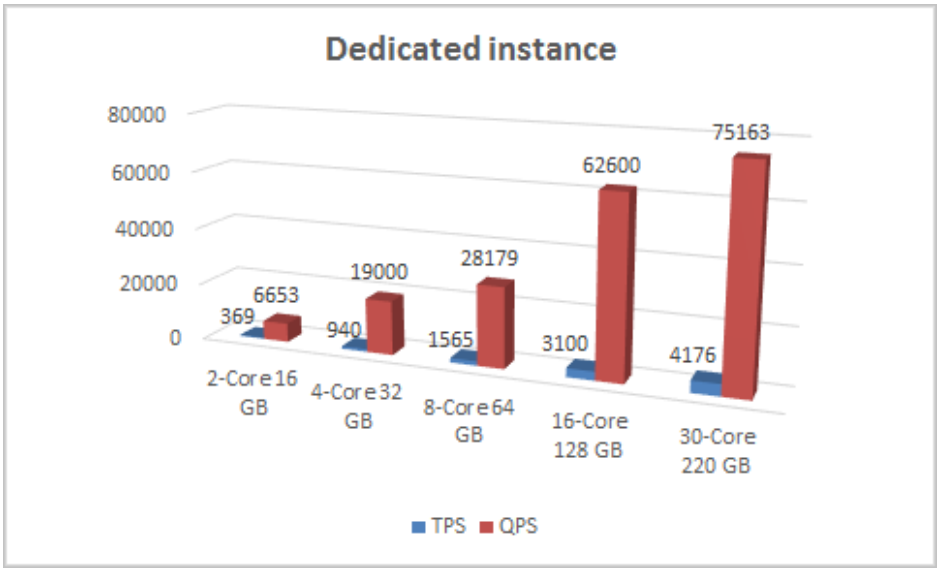
rds.mysql.c2.xlp2	16	96	24000	16000	2366	45000

Dedicated instance

Type ID	Number of CPU cores	Memory (GB)	Number of connections	IOPS	TPS	QPS
mysql.x8.medium.2	2	16	2500	4500	369	6653
mysql.x8.large.2	4	32	5000	9000	940	19000
mysql.x8.xlarge.2	8	64	10000	18000	1565	28179
mysql.x8.2xlarge.2	16	128	20000	36000	3100	62600
rds.mysql.st.d13	30	220	64000	20000	4176	75163

Test result





For SQL Server

Product overview

ApsaraDB for RDS is a stable, reliable, and automatically scalable online database service. Based on the Apsara distributed file system of Alibaba Cloud and the full SSD high-performance storage, RDS provides a complete set of solutions for disaster tolerance, backup, recovery, monitoring, and migration to free you from the burden of database O&M. RDS supports a variety of engines such as MySQL, SQL Server, PostgreSQL, and PPAS which is highly compatible with Oracle.

RDS supports two billing methods: Pay-As-You-Go and Subscription. You can configure the RDS instance specifications according to the service load. In terms of the two billing methods:

- You can upgrade and downgrade the specifications of a Pay-As-You-Go instance at any time.
- You can upgrade the specifications of a Subscription instance at any time, and upgrade and downgrade the specifications upon renewal.

In the case of extreme service load, you can upgrade the instance to the RDS dedicated host specifications to handle unexpected conditions.

Test approach

Test environment

All tests are performed in Zone B of China East 1 (Hangzhou).

The ECS instance for testing is C1, compute instance type family.

The configuration of the ECS instance is 8-core 16 GB.

The network type is classic network.

The image for stress testing is Windows Server 2012 Standard Edition (64-bit).

Test tool

Introduction to HammerDB

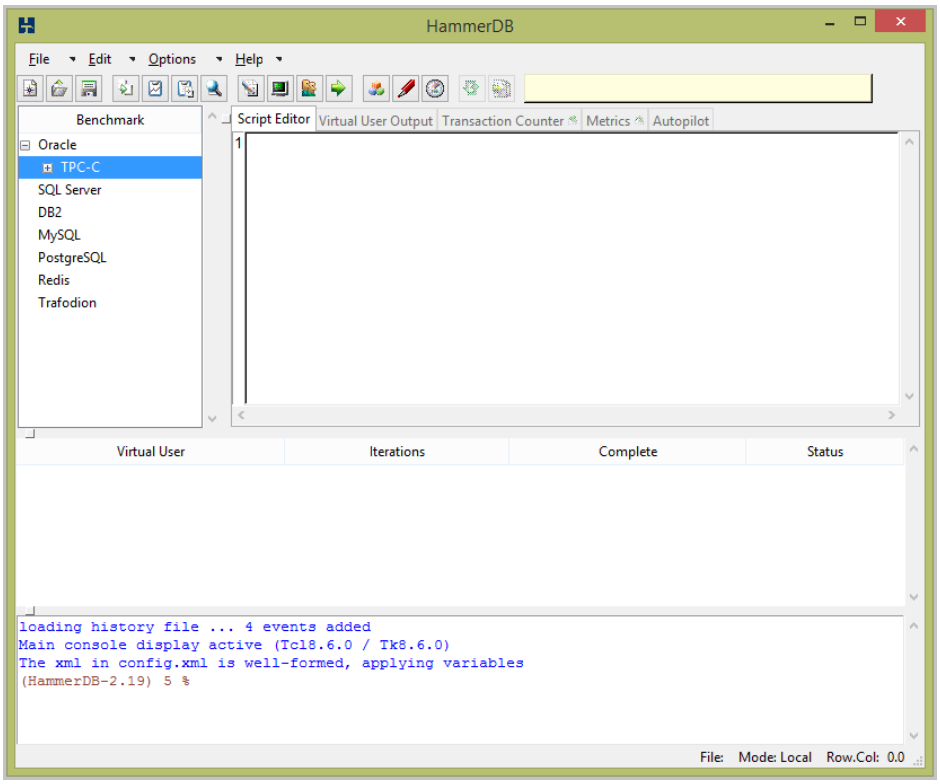
HammerDB is an open-source database load testing and benchmarking tool. It can be used to test the database systems running in all types of operating systems. It can run in Linux and Windows. HammerDB features automatic running, multithread support, and dynamic scripts extension. Currently, HammerDB supports a variety of databases, including mainstream databases such as Oracle, SQL Server, DB2, TimesTen, MySQL, MariaDB, PostgreSQL, Greenplum, Postgres Plus Advanced Server, Redis, and Trafodion SQL on Hadoop. HammerDB has a built-in baseline workload compliant with the TPC-C industrial standard.

Installation method

In this article, HammerDB 2.19 is used for testing. Click Here to download it.

Install HammerDB as instructed by the installation wizard.

After HammerDB is installed, the following page appears:



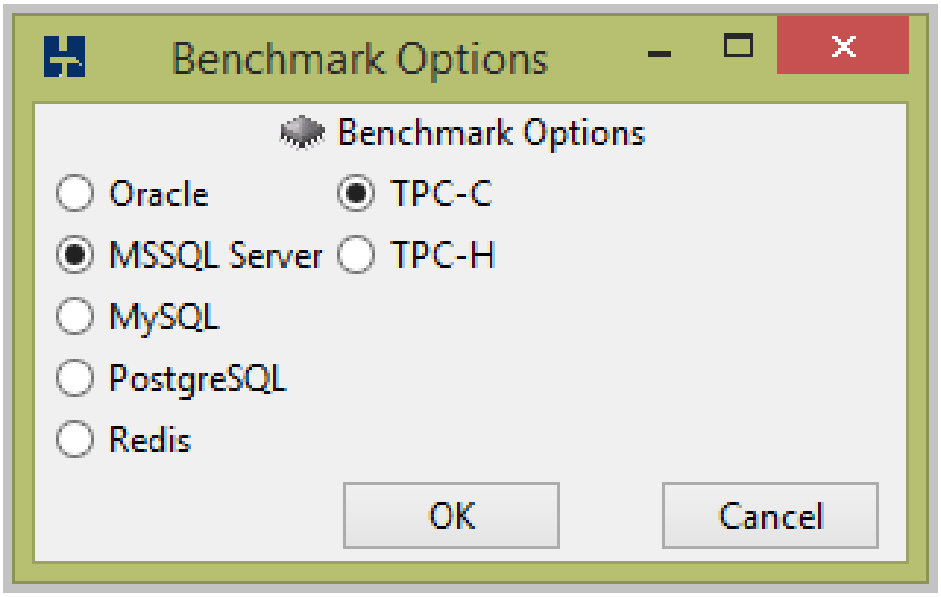
Test approach

Complete the following operations before you perform a TPC-C test.

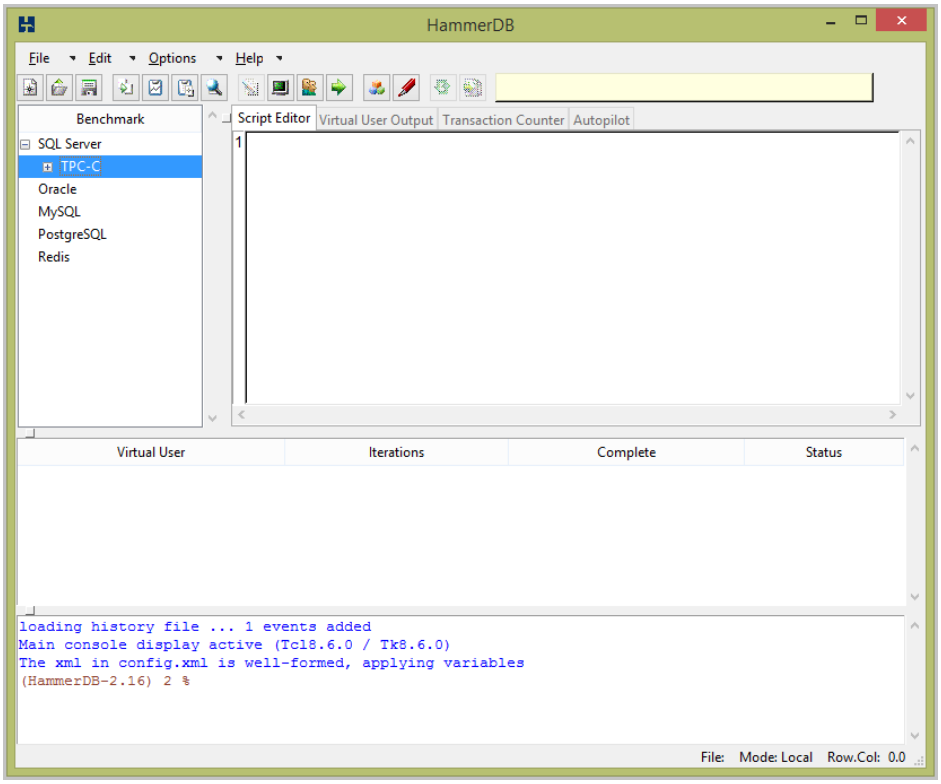
> **Note:** We recommend that you use the test result data in the first 30 minutes to 2 hours. With increasing data volumes, the TPC-C mode encounters performance bottlenecks, in which case you must add indexes to tables [dbo].[STOCK], [dbo].[ORDER_LINE], and [dbo].[ORDERS] to guarantee normal testing.

Open **HammerDB**.

Select **SQL SERVER** and **TPC-C**.

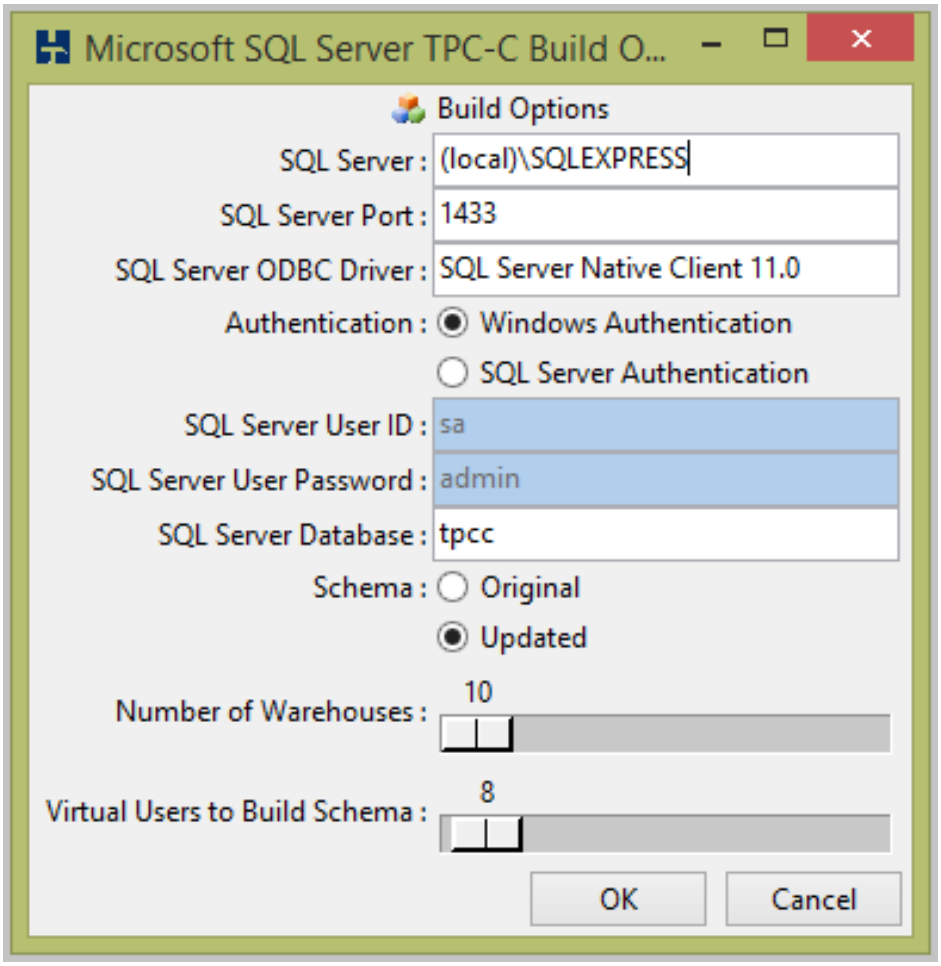


Prepare for building schema.

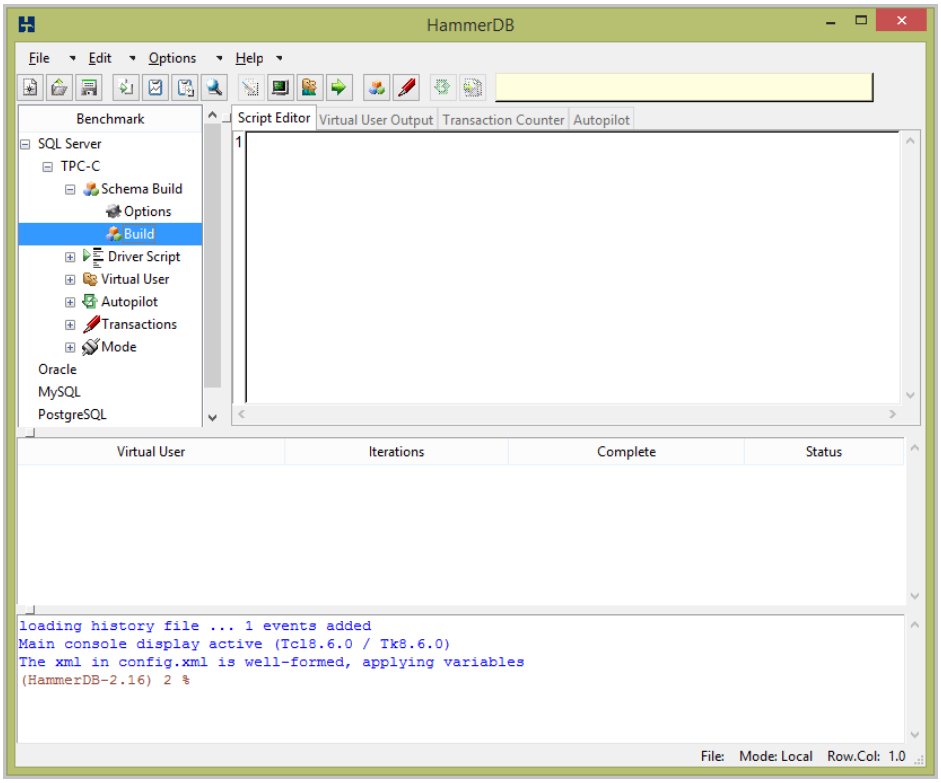


Set the connection information and initialize the repository. The value 10 applies to all specifications. Set and adjust the number of concurrent users based on stress to obtain the optimal test performance. Then double-click **Schema Build/Option**.

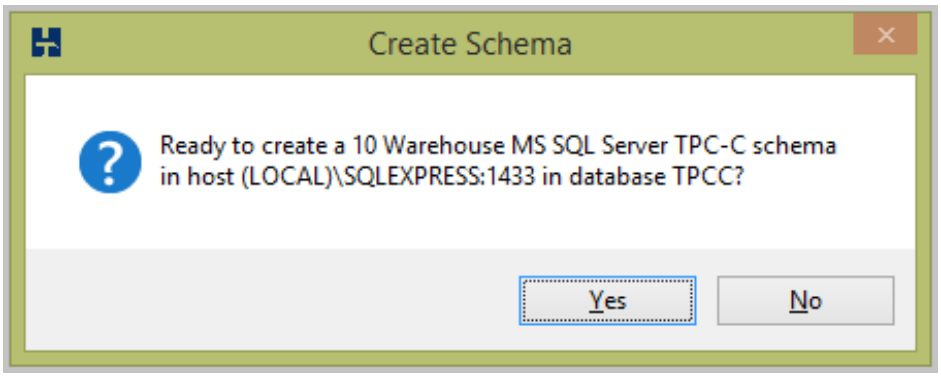
> **Note:** Though a port is specified during testing the RDS instance, you must specify the port on the SQL server, for example, **.sqlserver.rds.aliyuncs.com,3433.



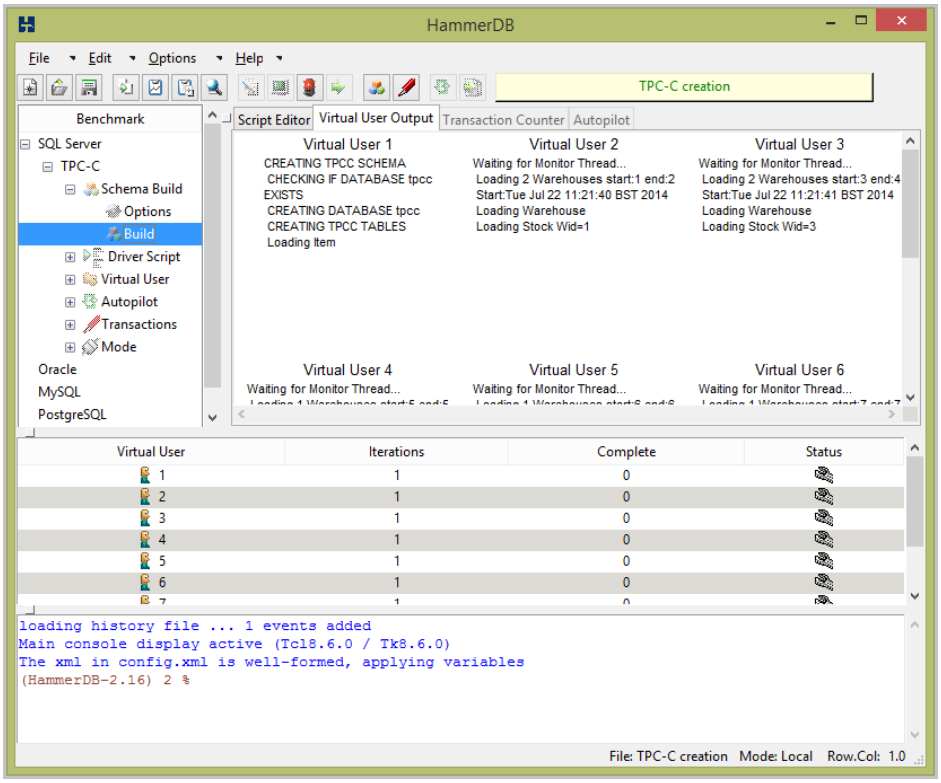
Double-click **Schema Build/Build**.



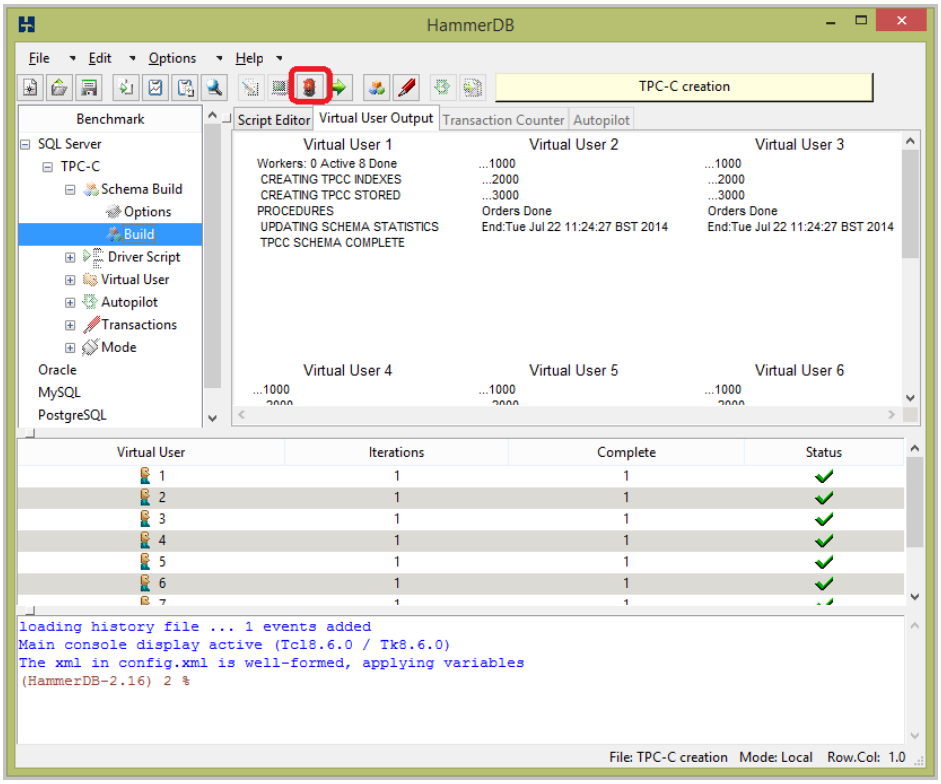
Click **YES** to create a schema.



Wait until the schema initialization is complete.

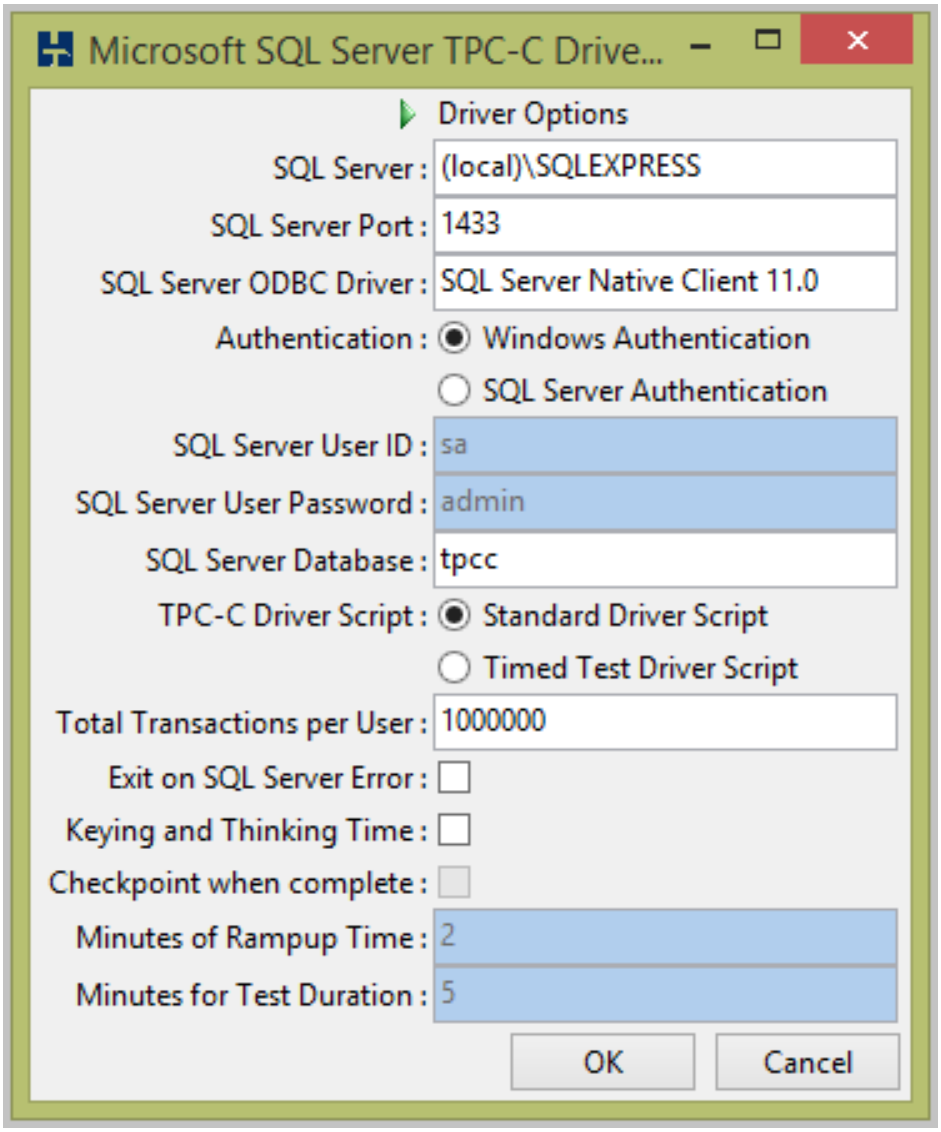


When all initialization items show **complete**, click the button marked by the red box to stop initialization.

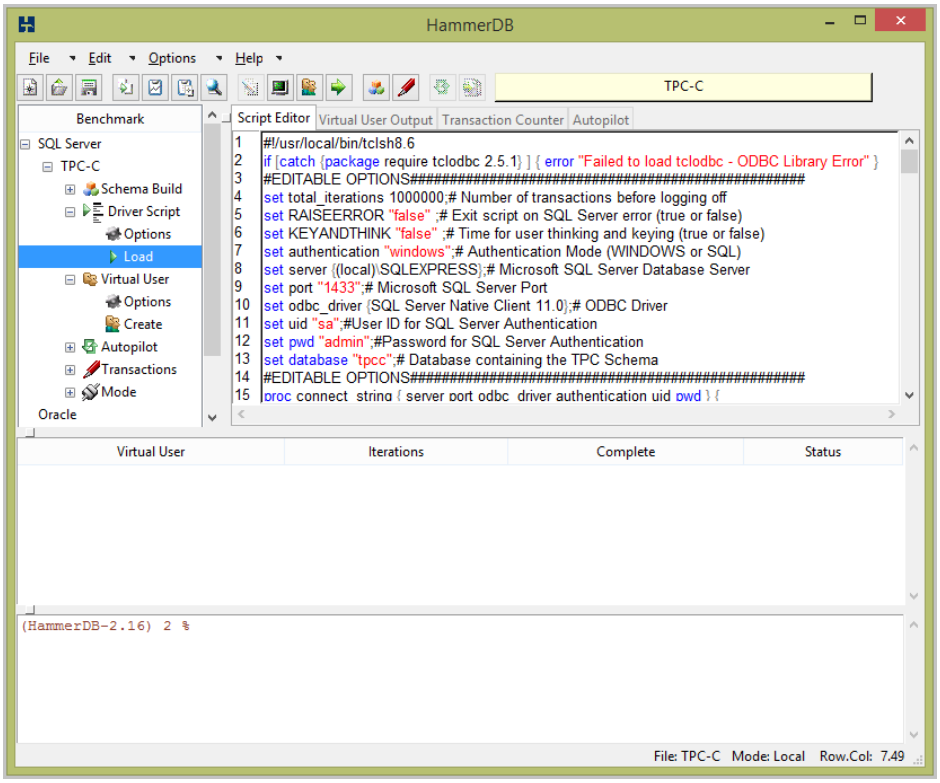


In the left-side navigation pane, select **Driver script/option** and verify that the database

connection information is correct.

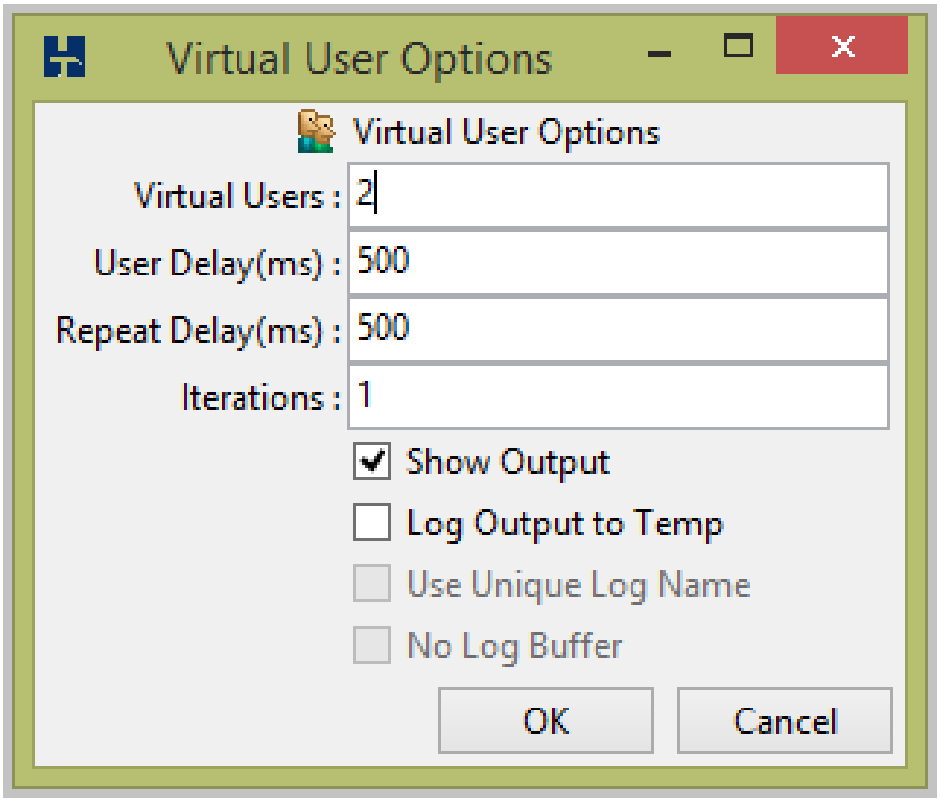


In the left-side navigation pane, select **Driver script/load**.

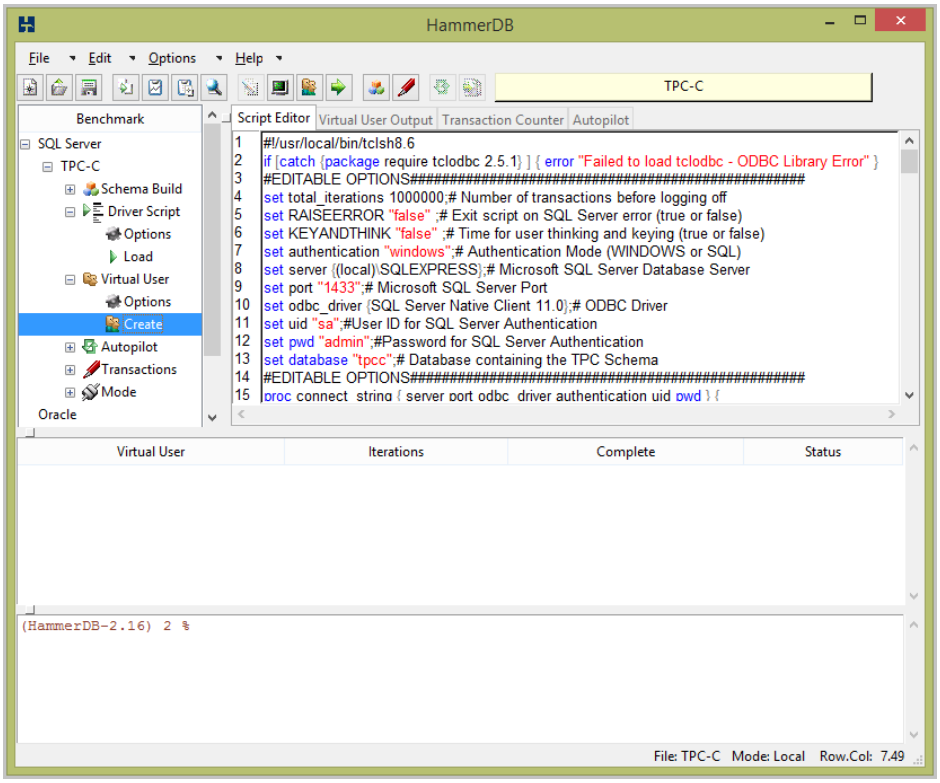


In the left-side navigation pane, select the Virtual User configuration, and select the number of users based on specifications. Wait until the database reaches the highest TPM.

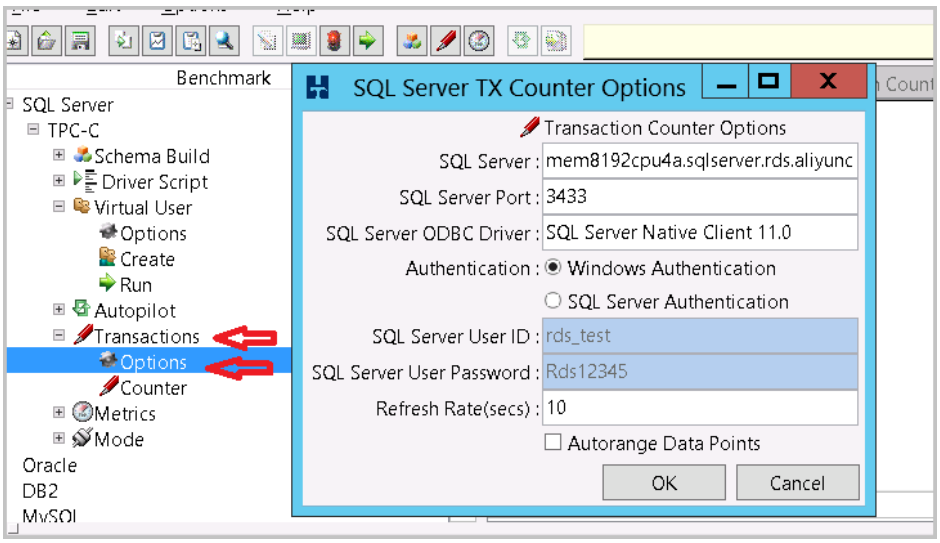
> **Note:** We do not recommend selecting the **show output** option, which may make the client unresponsive.



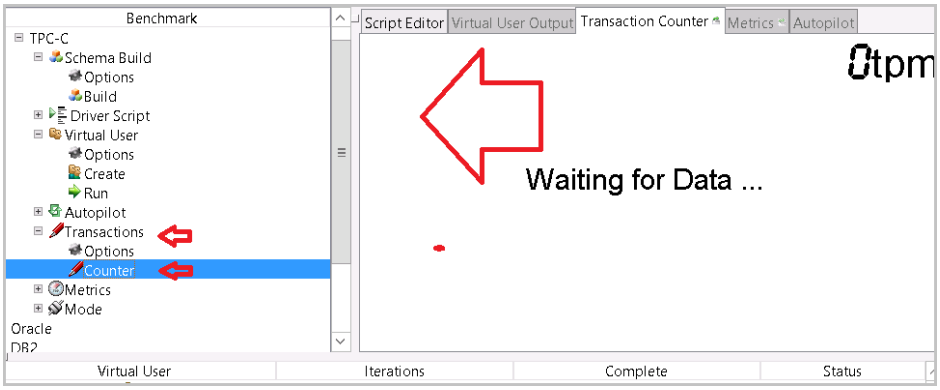
In the left-side navigation pane, select **Virtual User/Create**.



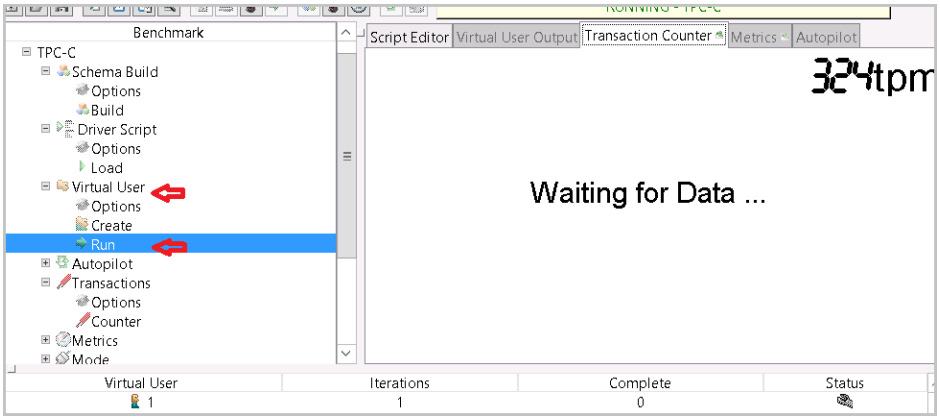
In the left-side navigation pane, select **Transactions/Option**.



In the left-side navigation pane, select **Transactions/Counter**.



In the left-side navigation pane, select **Virtual User/Run**.



Test model

Database table structure

For relevant information, see Introduction to Transactional (TPC-C) Testing for all Databases.

Table

```
[dbo].[CUSTOMER]
[dbo].[DISTRICT]:
[dbo].[HISTORY]:
[dbo].[ITEM]:
[dbo].[NEW_ORDER]
[dbo].[ORDER_LINE]
[dbo].[ORDERS]
[dbo].[STOCK]
[dbo].[WAREHOUSE]
```

Stored procedure

```
[dbo].[DELIVERY]
[dbo].[NEWORD]
[dbo].[OSTAT]
[dbo].[PAYMENT]
[dbo].[SLEV]
```

Test indicators

HammerDB only provides the TPM indicator but does not provide the TPS or BatchRequest indicators. The two indicators are provided in the following test.

TPM

Transactions per minute (TPM) indicates the number of transactions per minute.

TPS

Transactions per second (TPS) indicates the number of transactions processed by the database per second.

Batch Request

Batch Request indicates the number of requests processed in batches per second.

Note: Batch Request is not simply the sum of TPS and QPS.

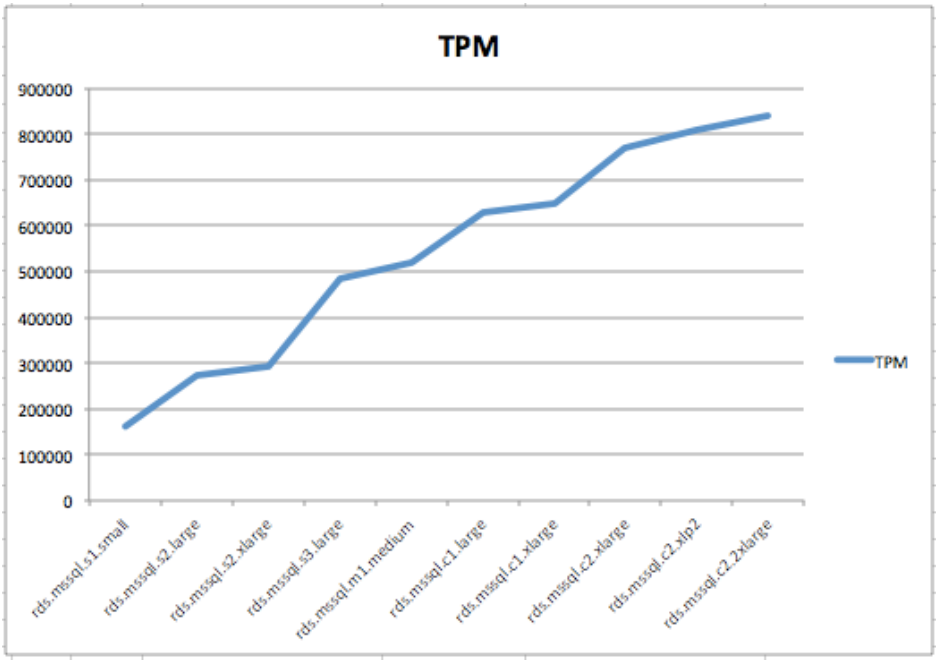
Test result

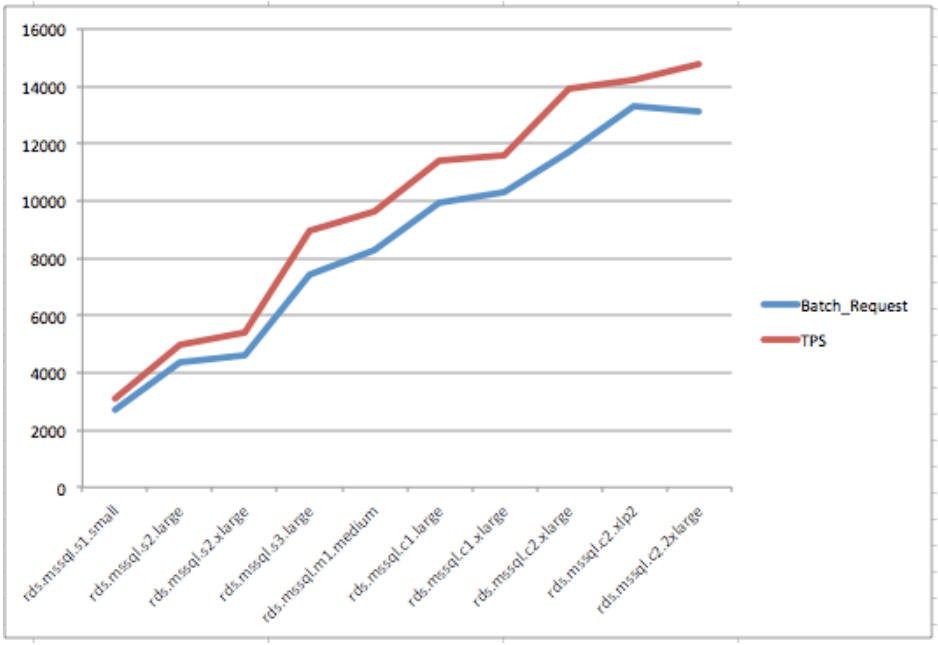
SQL Server 2008 R2 High-Availability Edition

Instance type

Type ID	CPU (unit: core)	Memory (unit: GB)	IOPS
rds.mssql.s1.small	1	2	1000
rds.mssql.s2.large	2	4	2000
rds.mssql.s2.xlarge	2	8	4000
rds.mssql.s3.large	4	8	5000
rds.mssql.m1.medium	4	16	7000
rds.mssql.c1.large	8	16	8000
rds.mssql.c1.xlarge	8	32	12000
rds.mssql.c2.xlarge	16	64	14000
rds.mssql.c2.xlp2	16	96	16000
rds.mssql.c2.2xlarge	16	128	16000

Test results





No.	Specifications	TPM	Batch_Request	TPS
1	rds.mssql.s1.small	162000	2680	3100
2	rds.mssql.s2.large	273000	4370	4980
3	rds.mssql.s2.xlarge	293000	4600	5400
4	rds.mssql.s3.large	483000	7450	8970
5	rds.mssql.m1.medium	517800	8260	9640

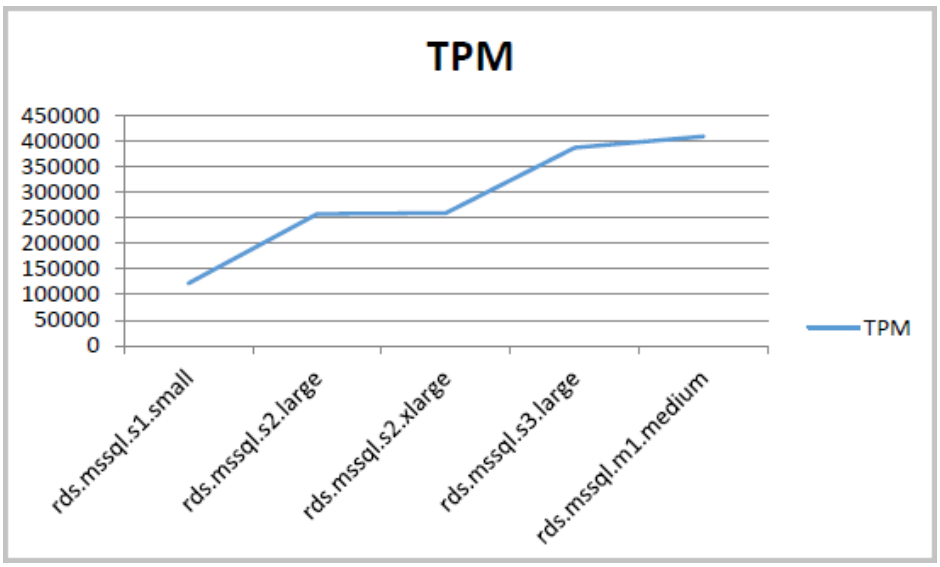
6	rds.mssql.c1.large	630000	9950	11400
7	rds.mssql.c1.xlarge	647000	10300	11600
8	rds.mssql.c2.xlarge	769000	11700	13900
9	rds.mssql.c2.xlp2	810000	13300	14200
10	rds.mssql.c2.2xlarge	840000	13100	14800

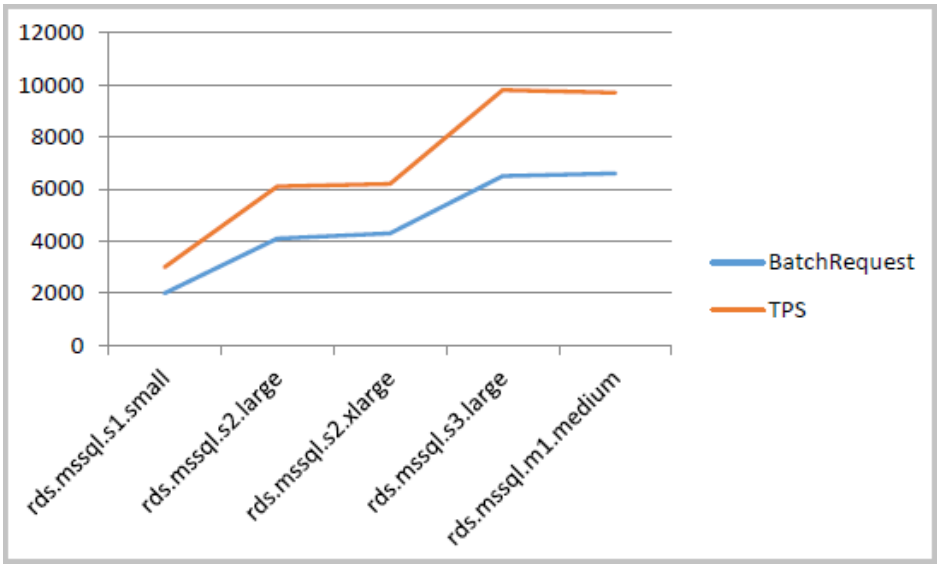
SQL Server 2012 Enterprise edition

Instance type

Type ID	CPU (unit: core)	Memory (unit: GB)
rds.mssql.s1.small	1	2
rds.mssql.s2.large	2	4
rds.mssql.s2.xlarge	2	8
rds.mssql.s3.large	4	8
rds.mssql.m1.medium	4	16

Test results





No.	Specifications	TPM	Batch_Request	TPS
1	rds.mssql.s1.small	122000	2000	3000
2	rds.mssql.s2.large	258000	4100	6100
3	rds.mssql.s2.xlarge	260000	4300	6200
4	rds.mssql.s3.large	388000	6500	9800
5	rds.mssql.m1.medium	410000	6600	9700