



# TPC Benchmark<sup>TM</sup> C

## Full Disclosure Report

---

**Tencent Database Dedicated Cluster (with  
1650 TDSQL Data Nodes)**

*Using*

*Tencent TDSQL v10.3 Enterprise Pro Edition  
with Partitioning and Physical Replication*

First Edition

March 28, 2023

First Edition – March 28, 2023

Copyright © 2023 Tencent Cloud Computing (Beijing) Co., Ltd. and/or its affiliates. All rights reserved.

Tencent Cloud Computing (Beijing) Co., Ltd., as the Sponsor of this benchmark test, believes that the pricing information in this document is accurate as of the publication date, and is subject to change without notice. The Sponsor assumes no responsibility for any errors that may appear in this document. The pricing information is believed to accurately reflect the current prices as of the publication date. However, the Sponsor provides no warranty of the pricing information in this document.

The performance information in this document is for guidance only. Benchmark results are highly dependent on many factors including workload, hardware, operating environments, specific application requirements, and system design and implementation. Relative system performance may vary significantly as a result of these and other factors. The Sponsor does not warrant or represent that a user can or will achieve the same or similar performance expressed in transaction per minute (tpmC®) or normalized price/performance (\$/tpmC®). No warranty on system performance is either expressed or implied.

TDSQL is a registered trademark of Tencent and/or its affiliates.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation.

TPC Benchmark, TPC-C, and tpmC are trademarks of the Transaction Processing Performance Council.

All other products mentioned herein are trademarks or registered trademarks of their respective owners.

©Copyright 2023 Tencent Cloud Computing (Beijing) Co., Ltd.

All rights reserved. This product and related documentation are protected by copyright and distributed under licenses restricting its use, copying, distribution, and de-compilation. No part of this product or related documentation may be reproduced in any form by any means without the prior written authorization of Tencent Cloud Computing (Beijing) Co., Ltd. and its licensors, if any.

THIS PUBLICATION IS PROVIDED WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. TENCENT CLOUD COMPUTING (BEIJING) CO., LTD. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT.


## Abstract

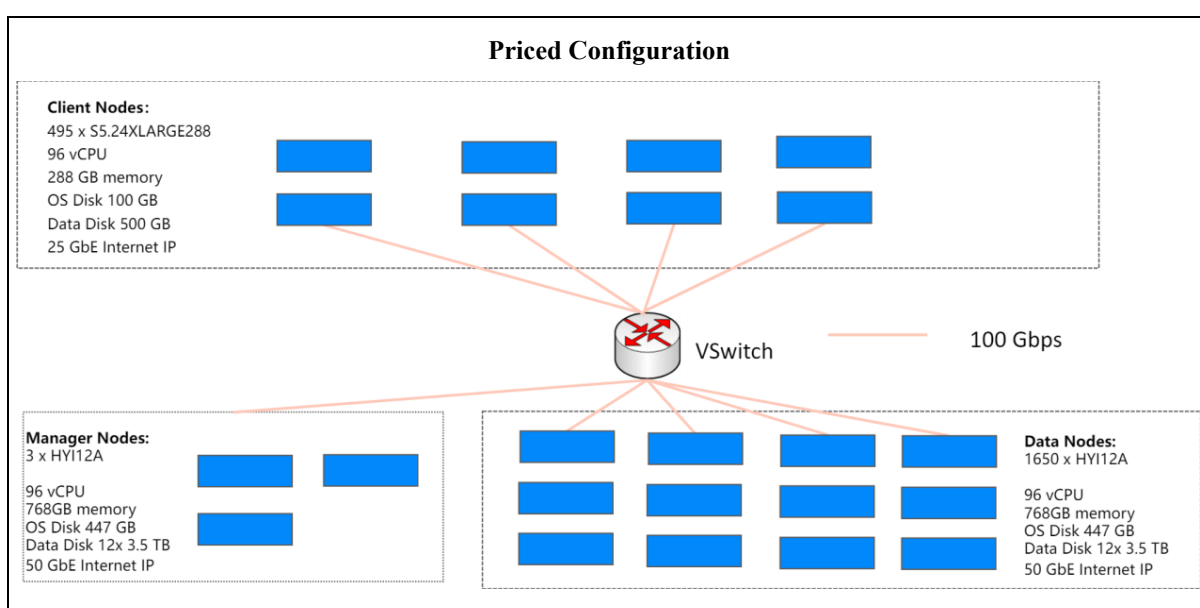
This report documents the methodology and results of the TPC Benchmark™ C test conducted on the following environment as measured by Tencent Cloud Computing (Beijing) Co., Ltd. The benchmark configuration, environment and methodology used to produce and validate the test results, and pricing model used to calculate the price/performance, were audited by Doug Johnson of InfoSizing to verify compliance with the relevant TPC specifications.

System	Processors	Database Environment	Operating System
Tencent Database Dedicated Cluster (with 1650 TDSQL Data Nodes)	Intel(R) Xeon(R) Platinum 8255C, 2.5GHz, 96 vCPU	Tencent TDSQL v10.3 Enterprise Pro Edition with Partitioning and Physical Replication	Tencent tlinux 2.2


## TPC Benchmark C Metrics

Total System Cost	TPC-C® Throughput	Price / Performance	Availability Date
Three-year cost including hardware, software, and maintenance	Maximum Qualified Throughput expressed as transaction per minute - C (tpmC)	Total System Cost / tpmC	Date for which all components, hardware and software are available for purchase
<b>1,031,746,953 CNY</b>	<b>814,854,791 tpmC</b>	<b>1.27 CNY/tpmC</b>	<b>June 18, 2023</b>

 <b>Tencent Cloud</b>	<b>Tencent Database Dedicated Cluster</b> <b>(with 1650 TDSQL Data Nodes)</b>		<b>TPC-C 5.11.0</b> <b>TPC-Pricing 2.8.0</b>	
			<b>Report Date</b> <b>March 28, 2023</b>	
<b>Total System Cost</b>	<b>TPC-C Throughput</b>	<b>Price/Performance</b>	<b>Availability Date</b>	
1,031,746,953 CNY	814,854,791 tpmC	1.27 CNY/tpmC	June 18, 2023	
<b>Database Server Processors/Cores/Threads</b>	<b>Database Manager</b>	<b>Operating System</b>	<b>Other Software</b>	<b>Number of Users</b>
Intel Xeon Platinum 8255C (2.50GHz) 3,300/79,200/158,400	Tencent TDSQL v10.3 Enterprise Pro Edition with Partitioning and Physical Replication	Tencent tlinux 2.2	Nginx 1.16.1	640,035,000



System Component	1,650 Data Nodes		3 Manager Nodes		495 Client Nodes		Total
<b>Processors /Cores/Threads</b>	2/48/96	Intel Xeon Platinum 8255C (2.50GHz)	2/48/96	Intel Xeon Platinum 8255C (2.50GHz)	2/48/96	Intel Xeon Platinum 8361HC (2.60GHz)	4,296/103,104/206,208
<b>Memory</b>	1	768 GB	1	768 GB	1	288 GB	1,412,064 GB
<b>OS Disk</b>	1	447 GB	1	447 GB	1	100 GB	788,391 GB
<b>Data Disk</b>	1	12x 3.5 TB NVMe SSD	1	12x 3.5 TB NVMe SSD	1	500 GB	69,673,500 GB
<b>Total Disk Storage</b>		70,037,550 GB		127,341 GB		297,000 GB	70,461,891 GB

 Tencent Cloud	<b>Tencent Database Dedicated Cluster</b> <b>(with 1650 TDSQL Data Nodes)</b>					TPC-C 5.11.0 TPC-Pricing 2.8.0	
						Report Date March 28, 2023	
Description	Part Number	Src	Discount	Unit Price	Qty	Ext.Price (CNY)	
<b>Tencent Database Dedicated Cluster</b>							
<b>HY112A (3-year-price)</b>		1		1,143,072.00	1,653	1,889,498,016.00	
- Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50GHz x2 (96 vCPU)	included	1		0.00	1,653		
- Memory (768G)	included	1		0.00	1,653		
- System Disk (447GB)	included	1		0.00	1,653		
- NVMe SSD Disk (12x 3.5 TB)	included	1		0.00	1,653		
- Tencent tlinux 2.2	included	1		0.00	1,653		
- Tencent TDSQL v10.3 Enterprise Pro Edition with Partitioning and Physical Replication	included	1		0.00	1,653		
- Mellanox Technologies MT27710 Family [ConnectX-4 Lx] x2	included	1		0.00	1,653		
- Tencent Cloud Online Support Service 7x24 for 3 years	included	1		0.00	1,653		
					<b>Sub-Total</b>	<b>1,889,498,016.00</b>	
<b>Tencent Cloud Virtual Machine</b>							
<b>S5.24XLARGE288 (3-year-price)</b>		1		397,440.00	495	196,732,800.00	
- Intel(R) Xeon(R) Platinum 8361HC CPU @ 2.60GHz x2 (96 vCPU)	included	1		0.00	495		
- Memory (288G)	included	1		0.00	495		
- System Disk (100GB)	included	1		0.00	495		
- Data Disk (500GB)	included	1		0.00	495		
- Tencent tlinux 2.6	included	1		0.00	495		
- Mellanox MCX562A-ACAI	included	1		0.00	495		
- Tencent Cloud Online Support Service 7x24 for 3 years	included	1		0.00	495		
					<b>Sub-Total</b>	<b>196,732,800.00</b>	
<b>Other Services</b>							
Professional Edition Support Service for 3 years 7x24hrs		2		6,300,000.00	1	6,300,000.00	
- Nginx support included	included	2		0.00	1		
					<b>Sub-Total</b>	<b>6,300,000.00</b>	

<b>Other Components</b>						
ThinkPad X1 Carbon 2022 Laptop (includes 2 spares)	21CBA003CD	3		12,499.00	3	37,497.00
					<b>Sub-Total</b>	<b>37,497.00</b>
<b>Discount</b>						
HYI12A (Tencent Database Dedicated Cluster)		1	50%	-571,536.00	1,653	-944,749,008.00
S5.24XLARGE288(CVM)		1	59%	-234,489.60	495	-116,072,352.00
					<b>Sub-Total</b>	<b>-1,060,821,360.00</b>
<b>Src</b>					<b>Total</b>	<b>1,031,746,953.00</b>
1. Tencent Cloud					<b>3-Y Cost of Ownership (CNY)</b>	<b>1,031,746,953.00</b>
2. TengCloud Future					<b>tpmC</b>	<b>814,854,791</b>
3. lenovo.com.cn					<b>CNY/tpmC</b>	<b>1.27</b>
<b>Audited by Doug Johnson of InfoSizing.</b>						
<i>Prices used in TPC Benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing section of the TPC benchmark specifications. If you find that stated prices are not available according to these terms, please inform the TPC at <a href="mailto:pricing@tpc.org">pricing@tpc.org</a>. Thank you.</i>						



# Tencent Database Dedicated Cluster (with 1650 TDSQL Data Nodes)

TPC-C 5.11.0  
TPC-Pricing 2.8.0

Report Date  
March 28, 2023

## Numerical Quantities:

MQTH (computed Maximum Qualified Throughput)	814,854,791 tpmC
Ramp-up Time	20 min
Measurement Interval (MI)	480 min
Checkpoints in MI per Node (Min/Avg/Max)	27,688/27,761.46/53,040
Maximum Checkpoint Interval (all nodes)	10.620 sec
Number of transactions (all types) completed in Measurement Interval	869,951,950,220

Response Time (Second)	Min	Avg	90th	Max
New-Order	0.101	0.107	0.114	13.610
Payment	0.101	0.104	0.106	12.329
Order-Status	0.101	0.104	0.104	11.328
Delivery	0.101	0.102	0.102	3.480
Stock-Level	0.102	0.109	0.116	13.825
Delivery (Deferred)	0.005	0.009	0.012	10.653
Menu	0.101	0.102	0.103	3.573
Emulated Display Delay:	0.100s			

Transaction Mix	Number	Percent
New-Order	391,130,299,744	44.960%
Payment	374,166,209,676	43.010%
Order-Status	34,885,214,326	4.010%
Delivery	34,885,339,177	4.010%
Stock-Level	34,884,887,297	4.010%

Keying Times (Second)	Min	Avg	Max
New-Order	18.001	18.001	18.106
Payment	3.001	3.001	3.108
Order-Status	2.001	2.001	2.098
Delivery	2.001	2.001	2.096
Stock-Level	2.001	2.001	2.096

Think Times (Second)	Min	Avg	Max
New-Order	0.002	12.001	120.048
Payment	0.002	12.001	120.046
Order-Status	0.002	10.001	100.031
Delivery	0.002	5.001	50.037
Stock-Level	0.002	5.001	50.028

# Table of Content

---

- 0 General Items ..... 15
  - 0.1 Application Code and Definition Statements ..... 15
  - 0.2 Benchmark Sponsor ..... 15
  - 0.3 Parameter Settings..... 15
  - 0.4 Configuration Diagrams ..... 15
- 1 Clause 1: Logical Database Design Related Items ..... 18
  - 1.1 Table Definitions..... 18
  - 1.2 Physical Organization of Database ..... 18
  - 1.3 Insert and Delete Operations ..... 18
  - 1.4 Horizontal or Vertical Partitioning ..... 18
  - 1.5 Replication or Duplication ..... 19
- 2 Clause 2: Transaction and Terminal Profiles ..... 20
  - 2.1 Random Number Generation ..... 20
  - 2.2 Input/Output Screens..... 20
  - 2.3 Priced Terminal Feature ..... 20
  - 2.4 Presentation Managers ..... 20
  - 2.5 Transaction Statistics ..... 20
  - 2.6 Queuing Mechanism ..... 21
- 3 Clause 3: Transaction and System Properties Related Items ..... 22
  - 3.1 Transaction System Properties (ACID)..... 22
  - 3.2 Atomicity ..... 22
    - 3.2.1 Completed Transaction ..... 22
    - 3.2.2 Aborted Transaction..... 22
  - 3.3 Consistency ..... 23
  - 3.4 Isolation Tests (1.12)..... 23
    - 3.4.1 Isolation Test 1 ..... 23
    - 3.4.2 Isolation Test 2..... 24
    - 3.4.3 Isolation Test 3..... 24
    - 3.4.4 Isolation Test 4..... 24
    - 3.4.5 Isolation Test 5..... 24
    - 3.4.6 Isolation Test 6..... 25
    - 3.4.7 Isolation Test 7..... 25
    - 3.4.8 Isolation Test 8..... 25
    - 3.4.9 Isolation Test 9..... 25



3.5 Durability .....	26
3.5.1 Instantaneous Interruption, Memory Failure, Network Failure, Disk Failure, Loss of Log, Loss of Database Tables .....	26
3.5.2 Power Failure, Full cluster failure.....	26
4 Clause 4: Scaling and Database Population .....	27
4.1 Cardinality of Tables.....	27
4.2 Distribution of tables and logs .....	27
4.3 Data model and database interface.....	27
4.4 The mapping of database partitions/replications.....	28
4.5 60 Day Space Computation.....	28
5 Clause 5: Performance Metrics and Response Time Related Items .....	30
5.1 Measured tpmC .....	30
5.2 Response Times .....	30
5.3 Keying and Think Times.....	30
5.4 Response Time Frequency Distribution Curves.....	30
5.5 Think Time Frequency Distribution.....	33
5.6 Response Times versus Throughput .....	33
5.7 Throughput versus Elapsed Time.....	34
5.8 Steady State Determination.....	34
5.9 Work Performance During Steady State .....	34
5.10 Measurement Interval.....	35
5.11 Transaction Mix Regulation.....	35
5.12 Transaction Mix .....	35
5.13 Percentage of New-Order Transactions .....	35
5.14 Number of Order-lines per New-Order .....	35
5.15 Percentage of Remote Order-lines per New-Order .....	36
5.16 Percentage of Remote Payments .....	36
5.17 Percentage of access customer by C_LAST for Payment and Order-Status .....	36
5.18 Percentage of Skipped Delivery Transactions.....	36
5.19 Checkpoints.....	36
6 Clause 6: SUT, Driver and Communications Related Items .....	38
6.1 RTE Description .....	38
6.2 Number of Terminal Connection Lost .....	38
6.3 Emulated Components .....	38
6.4 Configuration Diagrams.....	39
6.5 Network Configuration .....	39
6.6 Operator Intervention .....	39
7 Clause 7: Pricing Related Items .....	40
7.1 Hardware and software Price .....	40

7.2 Total 3-Year Cost.....	40
7.3 Availability Date .....	40
7.4 Hardware and Software Support .....	40
7.5 Statement of measured tpmC and Price/Performance .....	40
7.6 Country Specific Pricing .....	41
7.7 Orderability Date.....	41
8 Clause 8: Auditor Attestation.....	42
8.1 Auditor Information .....	42
8.2 Attestation Letter.....	42
Appendix A: Source Code File List .....	45
Appendix B: Database design .....	46
B.1 Table creation.....	46
B.2 Procedure creation.....	48
Appendix C: Configuration Options .....	55
C.1 Machine type I OS configuration .....	55
C.2 Machine type II OS configuration .....	56
C.3 Nginx configuration.....	57
C.4 Data Node database configuration.....	58
Appendix D: Price References .....	59
D.1 Tencent Cloud Virtual Machine S5.24XLARGE288.....	59
D.2 Tencent Database Dedicated Cluster HYI12A.....	59
D.3 ThinkPad X1 Carbon 2022 Laptop Price .....	60
D.4 TengCloud Service Support Price .....	61

## List of Tables

---

Table 1: Machine type I.....	15
Table 2: Machine type II. ....	16
Table 3: Summary of sharding .....	19
Table 4: Statistics for transactions and terminals .....	21
Table 5: Table cardinality. ....	27
Table 6: 60-day space computations (Note: All numbers about space are in GB). ....	28
Table 7: Parameter configuration of RTE.....	38
Table 8: Statement of tpmC and price/performance. ....	41

## List of Figures

---

Figure 1: Measured configuration. ....	16
Figure 2: Priced configuration.....	17
Figure 3: The procedures of durability test. ....	26
Figure 4: Database schema and partitions. ....	28
Figure 5: Frequency distribution of response times for Payment.....	31
Figure 6: Frequency distribution of response times for New-order.....	31
Figure 7: Frequency distribution of response times for Order-status .....	32
Figure 8: Frequency distribution of response times for Delivery-interactive.....	32
Figure 9: Frequency distribution of response times for Stock-level.....	33
Figure 10: Frequency distribution of think times for New-order .....	33
Figure 11: New-order response time versus throughput.....	34
Figure 12: Throughput versus elapsed time. ....	34

# Preface

---

TPC Benchmark<sup>TM</sup> C Standard Specification was developed by Transaction Processing Performance Council (TPC). It was released on August 13, 1992 and updated with revision 5.11 on February 2010.

This is the full disclosure report for benchmark testing of the TDSQL v10.3 Enterprise Pro Edition with Partitioning and Physical Replication on Tencent Cloud Service with 1,650 TDSQL Data Nodes to the TPC Benchmark<sup>TM</sup> C Standard Specification, Revision 5.11.

The TPC Benchmark<sup>TM</sup> C Full Disclosure Report is organized as following:

- *The main body of the document lists each item in Clause 8 of the TPC Benchmark<sup>TM</sup> C Standard and explains how each specification is satisfied.*
- *Appendix A contains the application source code file list.*
- *Appendix B contains the SQL queries used to create tables and procedures.*
- *Appendix C contains the configuration information for all the softwares and operating systems.*
- *Appendix D contains the Price References including those from the third party.*

# Introduction

---

The TPC Benchmark™ C Standard Specification requires test sponsors to publish, and make available to the public, a full disclosure report for the results to be considered compliant with the Standard.

This report is intended to satisfy the Standard's requirement for full disclosure. It documents the compliance of the benchmark tests required in the TPC Benchmark™ C results for the Tencent Cloud Dedicate Cluster (with 1,650 TDSQL Data Nodes) running TDSQL v10.3 Enterprise Pro Edition with Partitioning and Physical Replication.

In the *Standard Specification*, the main headings in Clause 8 are keyed to the other clauses. The headings in this report use the same sequence, so that they correspond to the titles or subjects referred to in Clause 8.

Each section in this report begins with the text of the corresponding item from Clause 8 of the Standard Specification, printed in italic type. The plain type text that follows explains how the tests comply with the TPC-C Benchmark.

# 0 General Items

## 0.1 Application Code and Definition Statements

The application program (as defined in clause 2.1.7) must be disclosed. This includes, but is not limited to, the code implementing the five transactions and the terminal input output functions.

Appendix A contains the file list of application source code for the five TPC-C transactions, including the input and output from the terminal, the interaction and communication with databases, and the implementation of deferred delivery. All files can be found in the supporting file package.

## 0.2 Benchmark Sponsor

A statement identifying the benchmark sponsor(s) and other participating companies must be provided.

The benchmark test was sponsored by Tencent Cloud Computing (Beijing) Co., Ltd. The implementation was developed and engineered by TDSQL team, Database R&D Department, Tencent Inc.

## 0.3 Parameter Settings

Settings must be provided for all customer-tunable parameters and options which have been changed from the defaults found in actual products, including by not limited to:

- Database tuning options
- Recover/commit options
- Consistency/locking options
- Operating system and application configuration parameters

Appendix C contains the tuned parameters for the operating systems, RTEs, Nginx servers, and database systems.

## 0.4 Configuration Diagrams

Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences.

Two types of machine instances/nodes are selected for the test. We list all the hardware configurations in the following.

Table 1: Machine type I.

VM instances	For RTE and Client nodes
CPU	Intel(R) Xeon(R) Platinum 8631HC
Processors/cores/thread	2/48/96
Memory	288 GB
OS disk (for OS)	100 GB
Local storage (for Data)	500 GB

We have in total 990 instances for machines type I where 495 instances are for RTE nodes and the other 495 instances for Client nodes.

Table 2: Machine type II.

Physical machines	For Manager and Data nodes
CPU	Intel(R) Xeon(R) Platinum 8255C
Processors/cores/thread	2/48/96
Memory	768 GB
OS disk	447 GB
Local storage (for Databases)	12x 3.5 TB NVMe SSD

We have in total 1653 nodes for machines type II where 1,650 nodes are for Data nodes and the other 3 nodes for Manager nodes.

There are four types of roles, the Remote Terminal Emulator (RTE) simulating clients, and System Under Test (SUT) includes three parts. (i) The Client nodes for web services receive client requests and communicate with the databases. (ii) The Manager nodes handle meta information like routing tables. (iii) The Data node is the deployment of the database for storing and processing data. In general, there are 495 instances for RTEs, also 495 instances for Client nodes, 3 machines for Manager nodes, and 1,650 machines for the Data nodes.

Figure 1 shows the overview of the system architecture. The highest level is the 495 RTE instances representing the clients. In the middle level, the 495 client instances receive the requests and then transform some parts of the job into SQL queries and pass to the databases. Then by the TDSQL proxies in the Client node, we find the right partitions of data in the specific Data nodes. In the low level, 3 Manager nodes store and update meta data, and 1,650 Data nodes manage and process the transactional data.

All tested machines are in a network zone arranged by a cluster of switches. Each ethernet port for test machines supports data transfer rates up to 25 Gbps.

Figure 2 shows the priced configuration of SUT, including the Client nodes, Manager nodes, and Data nodes. We exclude the RTE for calculating the cost of the test.

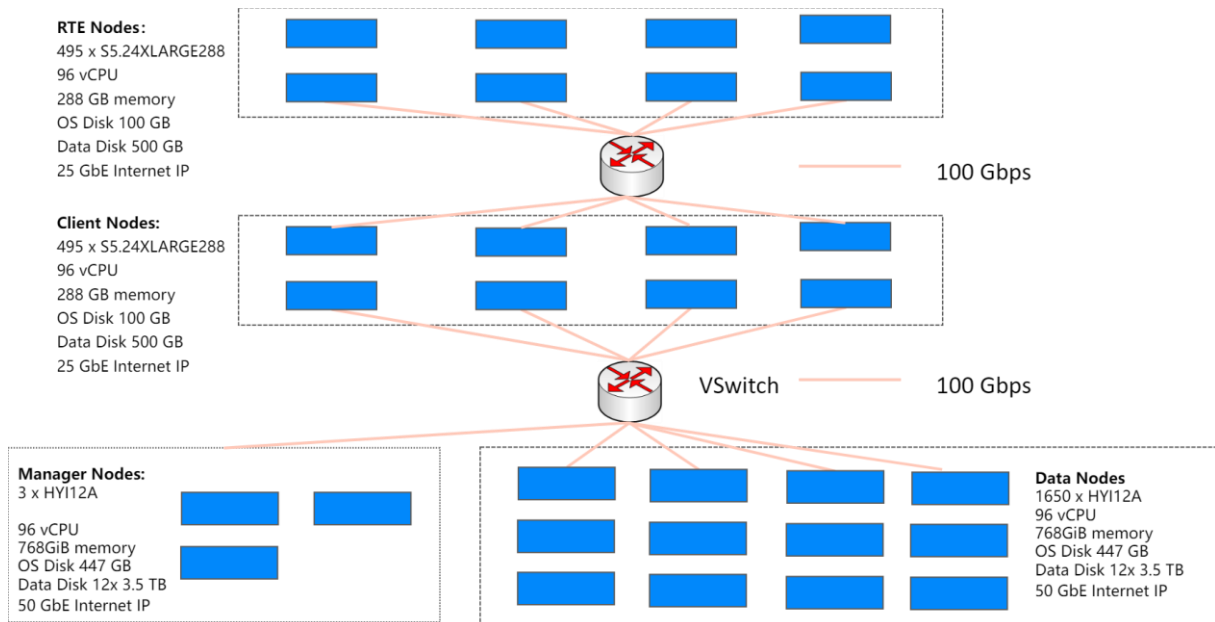


Figure 1: Measured configuration.



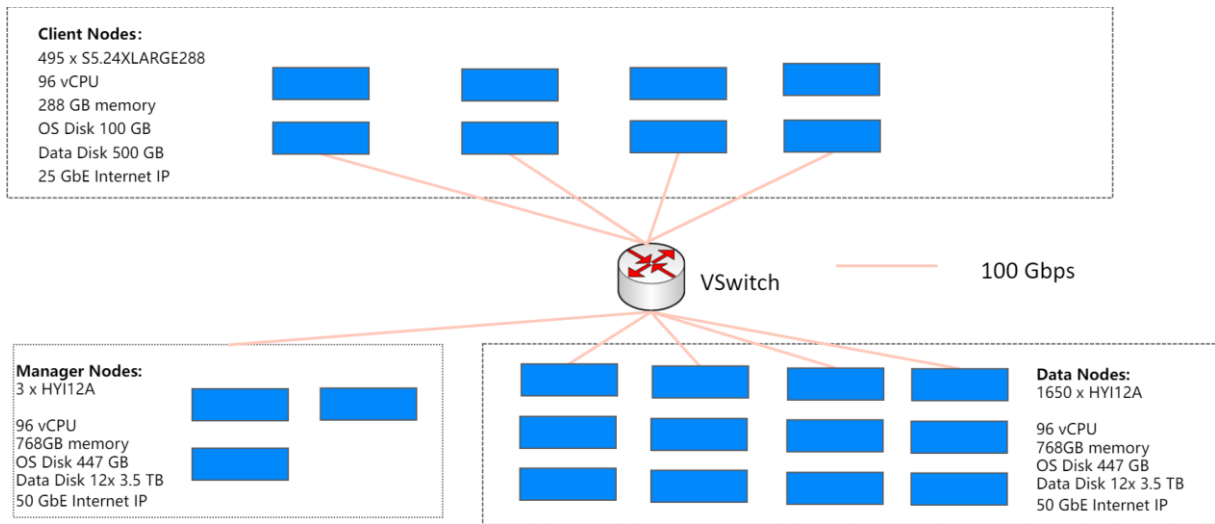


Figure 2: Priced configuration

# 1 Clause 1: Logical Database Design Related

## Items

### 1.1 Table Definitions

*Listing must be provided for all table definition statements and all other statements used to set up the database.*

Appendix B contains the scripts used to create tables and procedures.

### 1.2 Physical Organization of Database

*The physical organization of tables and indices within the database must be disclosed.*

Figure 2 and section 0.4 describe the overall SUT environment and system architecture. Each Data node has 12 SSD disks (3.5 TB NVMe each), which are formatted and managed by Tencent tlinux 2.2. TDSQL v10.3 Enterprise Pro Edition with Partitioning and Physical Replication. All database objects including table, index, and log are stored on these local disks. Appendix B contains the scripts used to create tables and indices. During table creation, the *shardkey* is specified so that the tables are partitioned.

### 1.3 Insert and Delete Operations

*It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C transaction mix. Furthermore, any restrictions in the SUT database implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the minimum key value for these new rows.*

All insert and delete functions were verified to be fully operational during the entire benchmark by the auditor.

### 1.4 Horizontal or Vertical Partitioning

*While there are a few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark, any such partitioning must be disclosed.*

The ITEM table is a global table that is fully replicated on every data node. All other tables are partitioned horizontally. The partitions were specified when creating tables by configuring *shardkey* parameter in the TDSQL database system.

Table 3: Summary of sharding

Table name	Table type	Shard Key
Warehouse	sharded table	w_id
District	sharded table	d_w_id
Customer	sharded table	c_w_id
New_orders	sharded table	no_w_id
Orders	sharded table	o_w_id
Order_line	sharded table	ol_w_id
Stock	sharded table	s_w_id
History	sharded table	h_c_w_id
Item	broadcast table	-

sharded table: The table is split into all sets using the shardkey field

broadcast table: All sets have a copy of the full table data

## 1.5 Replication or Duplication

*Replication of tables, if used, must be disclosed. Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance.*

ITEM table is fully replicated across all 1,650 Data nodes. All other tables are replicated into 3 replicas in 3 different nodes, called primary, replication, and log replicas.

The primary replica contains persistent data, page cache, continuous checkpoints and redo/undo logs. The replication replica contains persistent data, page cache, continuous checkpoints and redo/undo logs. The log replica only contains the redo logs. Replication replica has a similar memory consumption as the primary one. The log replica does not consume much memory, and it involves consensus voting but cannot be the leader.

## 2 Clause 2: Transaction and Terminal Profiles

### 2.1 Random Number Generation

*The method of verification for the random number generation must be described.*

Fastrand package in Go is used to implement a cryptographically secure pseudorandom number generator. The generator is seeded using the system's default entropy source, and thereafter produces random values via repeated hashing.

### 2.2 Input/Output Screens

*The actual layout of the terminal input/output screens must be disclosed.*

All screen layouts were verified by the auditor to validate that they followed the requirements of the specifications. The source code for generating the screens is available and provided. Appendix A describes the list of application source code files.

### 2.3 Priced Terminal Feature

*The method used to verify that the emulated terminals provide all the features described in Clause 2.2.2.4 must be explained. Although not specifically priced, the type and model of the terminals used for the demonstration in 8.1.3.3 must be disclosed and commercially available (including supporting software and maintenance).*

The mechanism of emulated terminals was verified by the auditor so that each required feature was implemented.

### 2.4 Presentation Managers

*Any usage of presentation managers or intelligent terminals must be explained.*

There is no intelligent terminal. The database and SUT send back the test result in HTML, which can be displayed by any Web browser. Appendix A describes the source code files for HTML display generation.

### 2.5 Transaction Statistics

*The percentage of home and remote order-lines in the New-Order transactions must be disclosed*

*The percentage of New-Order transactions that were rolled back as a result of an unused item number must be disclosed.*

*The number of items per orders entered by New-Order transactions must be disclosed*

*The percentage of home and remote Payment transactions must be disclosed.*

*The percentage of Payment and Order-Status transactions that used non-primary key (C\_LAST) access to the database must be disclosed.*

*The percentage of Delivery transactions that were skipped as a result of an insufficient number of rows in the NEW- ORDER table must be disclosed.*

*The percentage of Delivery transactions that were skipped as a result of an insufficient number of rows in the NEW- ORDER table must be disclosed.*

The mix (i.e., percentages) of transaction types seen by the SUT must be disclosed.

The following table describes the percentage of the transactions mentioned.

Table 4: Statistics for transactions and terminals

New Order	
Percentage of Home order-line	99.000%
Percentage of Remote order-line	1.000%
Percentage of Rolled Back Transactions	1.000%
Avg. Number of Items per Transactions	10.000
Payment	
Percentage of Home Transactions	85.000%
Percentage of Remote Transactions	15.000%
Access by C_LAST (Non-primary key)	
Percentage of Payment Transactions	60.000%
Percentage of Order-Status Transactions	60.000%
Delivery	
Skipped Transaction	0.000%
Transaction Mix	
New-Order	44.960%
Payment	43.010%
Order-status	4.010%
Delivery	4.010%
Stock-level	4.010%

## 2.6 Queuing Mechanism

The queuing mechanism used to defer the execution of the Delivery transaction must be disclosed.

We have implemented the Delivery Queue in the SUT. When a request comes from the RTE, the web server Nginx in the Client node redirects the request to SUT. SUT gets the request and puts it into a Delivery Queue and returns a response. There are also some routines in SUT to check if there is any request in the Delivery Queue and execute the deferred Delivery transaction and record the Delivery result into the result file.

# 3 Clause 3: Transaction and System Properties

## Related Items

### 3.1 Transaction System Properties (ACID)

*The results of the ACID tests must be disclosed along with a description of how the ACID requirements were met. This includes disclosing which case was followed for the execution of Isolation Test 7.*

Atomicity, Consistency, Isolation, and Durability (ACID) properties are required for the SUT. A set of tests must be conducted to meet the ACID requirements. This section describes the tests conducted and shows TDSQL compliances with the standard described in the specification.

### 3.2 Atomicity

*The system under test must guarantee that the database transactions are atomic. The system will either perform all individual operations on the data or will assure that no partially completed operations leave any effects on the data.*

#### 3.2.1 Completed Transaction

*Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number as specified in Clause 2.5.1.2) and verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have been changed appropriately*

The atomicity test was executed twice on the TDSQL cluster. In the first case (simulating local scenarios), the payment transaction was executed on one Data node of the TDSQL cluster, which means the randomly selected warehouse, district, and customer tables were stored on the same node. In the second case (simulating remote scenarios), the transaction was executed on two Data nodes of the TDSQL cluster, which means the randomly selected warehouse, district, and customer tables were stored on two different nodes.

The following steps were performed to verify the atomicity of the completed Payment transaction.

A row was randomly selected from the warehouse, district, and customer tables in the TDSQL database cluster, and a balance was noted (B1). A payment transaction was started with the selected warehouse, district, and customer above and a known amount of balance (b0), commit the transaction. It was verified that the balance after the payment transaction (B2) is equal to the balance before the transaction minus the changed amount of balance, which means  $B2 = B1 - b0$ .

#### 3.2.2 Aborted Transaction

*Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number as specified in Clause 2.5.1.2) and substitute a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have NOT been changed.*

The atomicity test was executed twice on the TDSQL cluster. In the first case (simulating local scenarios), the payment transaction was executed on one Data node of the TDSQL cluster, which means the randomly selected warehouse, district, and customer tables were stored on the same node. In the second case (simulating remote scenarios), the transaction was executed on two Data nodes of the TDSQL cluster, which means the randomly selected warehouse, district, and customer tables were stored on two different nodes.

The following steps were performed to verify the atomicity of the aborted Payment transaction:

A row was randomly selected from the warehouse, district, and customer tables in the TDSQL database, and a balance was

noted (B1). A payment transaction was started with the selected warehouse, district, and customer above and a known amount of balance (b0), rollback the transaction. It was verified that the balance after the payment transaction (B2) is equal to the balance before the transaction, which means  $B2 = B1$ .

## 3.3 Consistency

*Consistency is the property of the application that requires any execution of a database transaction to take the database from one consistent state to another, assuming that the database is initially in a consistent state.*

*Verify that the database is initially consistent by verifying that it meets the consistency conditions defined in Clauses 3.3.2.1 to 3.3.2.4. Describe the steps used to do this in sufficient detail so that the steps are independently repeatable.*

The specification requires explicitly demonstrating the following four consistency tests.

1. The sum of year-to-date balances (D\_YTD) for all districts within a specific warehouse is equal to the year-to-date balance (W\_YTD) of that warehouse.
2. For each district within a warehouse, the next available Order ID (D\_NEXT\_O\_ID) minus one is equal to the most recent Order ID [ $\max(O\_ID)$ ] in the Order table for the associated district and warehouse. It's also equal to the most recent Order ID [ $\max(NO\_O\_ID)$ ] of the New-Order table for the associated district and warehouse. Those relationship can be illustrated as follows:  
 $D\_NEXT\_O\_ID - 1 = \max(O\_ID) = \max(NO\_O\_ID)$   
where ( $D\_W\_ID = O\_W\_ID = NO\_W\_ID$ ) and ( $D\_ID = O\_D\_ID = NO\_D\_ID$ )
3. For each district within a warehouse, the value of the most recent order ID [ $\max(NO\_O\_ID)$ ] minus the first Order ID [ $\min(NO\_O\_ID)$ ] plus one for the New-Order table is equal to the total number of rows in that New-Order table. The relationship can be illustrated as follows:  
 $\max(NO\_O\_ID) - \min(NO\_O\_ID) + 1 = \text{number of rows in this New-Order table for this district.}$
4. For each district in a warehouse, the sum of order line counts [ $\sum(O\_OL\_CNT)$ ] for the Order table associated with this district equals to the total number of rows in the Order-Line table associated with the same District. The relationship can be illustrated as follows:  
 $\sum(O\_OL\_CNT) = \text{number of rows in the Order-Line table for this district}$   
where ( $O\_W\_ID = OL\_W\_ID$ ) and ( $O\_D\_ID = OL\_D\_ID$ )

These consistency conditions were tested by four independent functions using a go program to issue queries to the TDSQL cluster. The four consistency tests were performed twice, one was after loading data and the other was after the performance run reported. The results of the queries verified that the database was consistent for all four tests.

## 3.4 Isolation Tests (1.12)

*The Benchmark Standard Specification in Clause 3.4.2 defines nine tests used to demonstrate the required level of transaction isolation is met in the SUT database.*

These nine isolation tests are performed on the TDSQL cluster with 1,650 sets configured for 64,003,500 warehouses. The isolation level is Repeatable Read (RR) for all nine tests.

### 3.4.1 Isolation Test 1

*This test demonstrates isolation for read-write conflicts of Order-Status and New-Order transactions when the New-Order transaction is committed.*

The test proceeds as follows.

1. An Order-Status transaction T0 was executed and committed for a randomly selected customer C. The committed state of T0 was confirmed.
2. A New-Order transaction T1 for customer C was started, executed, and slept 20 seconds right before commit.
3. An Order-Status transaction T2 was started, executed, and committed before T1 committed. The transaction T2 wasn't blocked by T1. T2 returned the same order that T0 had returned.
4. T1 was committed.
5. An Order-Status transaction T3 for customer C was started and T3 returned the order inserted by T1.

## 3.4.2 Isolation Test 2

*This test demonstrates isolation for read-write conflicts of Order-Status and New-Order transactions when the New-Order transaction is rolled back.*

The test proceeds as follows.

1. The Order-Status transaction T0 was executed and committed for a randomly selected customer C. The committed state of T0 was confirmed.
2. A New-Order transaction T1 for customer C was started, executed, and slept 20 seconds right before rollback.
3. An Order-Status transaction T2 was started, executed, and committed before T1 committed. The transaction T2 wasn't blocked by T1. T2 returned the same order that T0 had returned.
4. T1 was allowed to ROLLBACK.
5. An Order-Status transaction T3 for customer C was started and T3 returned the same order that T0 had returned.

## 3.4.3 Isolation Test 3

*This test demonstrates isolation for write-write conflicts of two New-Order transactions when both transactions are committed.*

The test proceeds as follows.

1. A transaction T0 was started to retrieve the D\_NEXT\_O\_ID of a randomly selected district D and committed.
2. A New-Order transaction T1 was started for a randomly selected customer C in district D and slept 20 seconds before commit.
3. A New-Order transaction T2 was started for customer C and waited.
4. T1 was committed after waiting 20 seconds, and returned the order number, which was the same as the D\_NEXT\_O\_ID retrieved in T0.
5. T2 was committed. The returned order number of T2 was one greater than the order number returned by T1.
6. A transaction T3 retrieved the D\_NEXT\_O\_ID of district D again. It was one greater than the order number returned by T2.

## 3.4.4 Isolation Test 4

*This test demonstrates isolation for write-write conflicts of two New-Order transactions when one transaction is rolled back.*

The test proceeds as follows.

1. A transaction T0 was started to retrieve the D\_NEXT\_O\_ID of a randomly selected District D.
2. A New-Order transaction T1 was started for a randomly selected customer C in District D, and slept 20 seconds before rollback.
3. A New-Order transaction T2 was started for customer C and waited.
4. T1 was aborted after waiting 20 seconds, and returned the order number, which was the same as the D\_NEXT\_O\_ID retrieved in T0.
5. T2 was committed. The returned order number of T2 was the same as the D\_NEXT\_O\_ID retrieved in T0.
6. A transaction T3 retrieved the D\_NEXT\_O\_ID of District D again. It was one greater than the order number returned by T2.

## 3.4.5 Isolation Test 5

*This test demonstrates isolation for write-write conflicts of Payment and Delivery transactions when Delivery transaction is committed.*

The test proceeds as follows.

1. A transaction T0 executed a query to find out the customer C who is to be updated by the next Delivery transaction for a randomly selected warehouse W and district D, and retrieved the C's C\_BALANCE.
2. A Delivery transaction T1 was started and executed for warehouse W, and slept 20 seconds before commit.
3. A Payment transaction T2 was started for customer C, and waited.
4. T1 was committed after waiting 20 seconds. T2 was committed.
5. A transaction T3 retrieved the C\_BALANCE of customer C again. The C\_BALANCE reflected the results of both T1 and T2, meaning  $C\_BALANCE\ in\ T3 = C\_BALANCE\ in\ T0 + Delivery\ amount\ in\ T1 - Payment\ amount\ in\ T2$ .



## 3.4.6 Isolation Test 6

*This test demonstrates isolation for write-write conflicts of Payment and Delivery transactions when the Delivery transaction is rolled back.*

The test proceeds as follows.

1. A transaction T0 was started to execute a query to find out the customer C who is to be updated by the next Delivery transaction for a randomly selected warehouse W and district D, and retrieved the C's C\_BALANCE.
2. A Delivery transaction T1 was started and executed for warehouse W, and slept 20 seconds before rollback.
3. A Payment transaction T2 was started for customer C, and waited.
4. T1 was aborted after waiting 20 seconds. T2 was committed.
5. A transaction T3 retrieved the C\_BALANCE of customer C again. The C\_BALANCE reflected the results of only T2, meaning  $C\_BALANCE\ in\ T3 = C\_BALANCE\ in\ T0 - Payment\ amount\ in\ T2$ .

## 3.4.7 Isolation Test 7

*This test demonstrates repeatable reads for the New-Order transaction while an interactive transaction updates the prices of some items.*

The test proceeds as follows.

1. A transaction T0 was started to retrieve the I\_PRICE of two randomly selected items X and Y.
2. A New-Order transaction T1 with a group of items including items X and Y was started. T1 slept 20 seconds after retrieving the price of items X before retrieving the prices of item Y and of item X the second time.
3. A transaction T2 was started to increase the price of items X and Y by 10% and committed.
4. T1 retrieved the prices of X and Y again. The price of items X and Y matched those retrieved in T0.
5. T1 was committed.
6. A transaction T4 retrieved the items X and Y again. The values were matched the values set by T2, increased 10%.

## 3.4.8 Isolation Test 8

*This test demonstrates isolation for phantom protection between New-Order and Delivery transactions.*

The test proceeds as follows.

1. A transaction T0 was started to update the District ID (NO\_D\_ID) of all New-Order rows into D' (different than D) for a randomly selected warehouse W and district D, and committed.
2. A Delivery transaction T1 was started for warehouse W. T1 slept 20 seconds after retrieving the NEW\_ORDER table for warehouse W and district D. And no qualified row was found.
3. A New-order transaction T2 was started and committed for warehouse W and district D, without being blocked by T1. And no qualified row was found.
4. T1 was committed.
5. T3 retrieved the NEW\_ORDER table again and found the rows of district D inserted by T2.

## 3.4.9 Isolation Test 9

*This test demonstrates isolation for phantom protection between New-Order and Order-Status transactions.*

The test proceeds as follows.

1. An Order-Status transaction T1 was started for randomly selected customer C.
2. T1 slept 20 seconds after retrieving the most recently order for the customer C.
3. A New-Order transaction T2 was started for customer C and committed without being blocked by T1.
4. T1 was continued and retrieved the most recently order for the customer C again. The result was the same as the one found in step 2.
5. T1 was committed.
6. A transaction T3 retrieved the most recently order for the customer C and found that the result was the new one inserted by T2.

## 3.5 Durability

The tested system must guarantee durability: the ability to preserve the effects of committed transactions and insure database consistency after recovery from any one of the failures listed in Clause 3.5.3.

### 3.5.1 Instantaneous Interruption, Memory Failure, Network Failure, Disk Failure, Loss of Log, Loss of Database Tables

To demonstrate the durability of TDSQL under the scene of hardware failure such as memory failure, disk failure, network failure or software failure such as loss of files, interruption of instance. The test was executed by the following steps:

1. Checked consistency of the database and passed it. Obtained the sum of D\_NEXT\_O\_ID from all rows in the DISTRICT table, which is used to determine the current total number of orders, recorded as Count1.
2. The RTE was started with full user load.
3. The test was ramped up and ran in steady state at a tpmC greater than 90% of the reported tpmC for at least 20 minutes.
4. The power cord was pulling out of the electrical outlet for primary Manager node.
5. The TDSQL database cluster recovered automatically within several seconds. The duplication Manager node replaced it automatically.
6. The RTE does not report some errors. The test ran in steady state at a tpmC greater than 90% of the reported tpmC for at least 5 minutes.
7. The power cord was pulling out of the electrical outlet for primary Data node.
8. The TDSQL database cluster recovered automatically within several seconds. The duplication Data node replaced it automatically.
9. The RTE reported some failure transaction and recover to steady state in several seconds. The test continued in steady state for more than 5 minutes.
10. Logon Tencent Cloud control manager and destroyed a randomly selected primary Client node.
11. The client node did not recover.
12. The RTE reported failure transactions.
13. The RTE recorded all successful and rollbacked New-Order transactions in a *No\_detail* file during testing period.
14. Obtained the sum of D\_NEXT\_O\_ID from all rows in the DISTRICT table again, recorded as Count2, giving the ending total number of orders.
15. Checked consistency of the database and passed it. All successful New-Order Transactions in the *No\_detail* file are verified in the database and all rollbacked transactions in the *No\_detail* file are verified not in the database.
16. The difference between the counts, i.e., Count2-Count1, was compared with the sum of the committed transactions in the success file on RTE, as S0.
17. Count2-Count1  $\geq$  S0, and there is no committed transaction were lost.

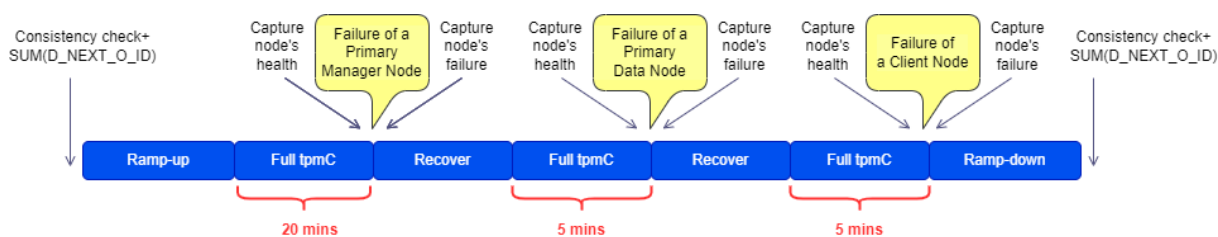


Figure 3: The procedures of durability test.

### 3.5.2 Power Failure, Full cluster failure

TDSQL uses three-replica strong synchronous replication for a single SET, where a SET consists of primary, replication, and log replicas. And each transaction log will be written to at least two replicas to ensure that the transaction data can be fully recovered from the replication node in case of the failure of the primary node while guaranteeing the reliability and correctness of the SET data; The two-phase commit protocol is adopted for distributed transactions. For pending transactions, the supplementary commit or rollback will be automatically performed to ensure that the complete ACID feature can be met in distributed scenarios.

This full cluster failure test is justified by documentation. Tencent Cloud's physical machines all comply with 30 minutes of Uninterruptible Power Supply (UPS) requirement.

# 4 Clause 4: Scaling and Database Population

## 4.1 Cardinality of Tables

The cardinality (e.g., the number of rows) of each table, as it existed at the start of the benchmark run (see Clause 4.2), must be disclosed. If the database was over-scaled and inactive rows of the WAREHOUSE table were deleted (see Clause 4.2.2), the cardinality of the WAREHOUSE table as initially configured and the number of rows deleted must be disclosed.

The TDSQL database was built with 64,003,500 warehouses. The following table shows the initial cardinality of the tables after the table population and the cardinality prior to the measurement run.

Table 5: Table cardinality.

Table	Initial Row Count	Row Count prior to Measured Run
Warehouse (WARE)	64,003,500	64,003,500
District (DIST)	640,035,000	640,035,000
Customer (CUST)	1,920,105,000,000	1,920,105,000,000
History (HIST)	1,920,105,000,000	1,920,105,000,000
Order (ORDR)	1,920,105,000,000	1,920,105,000,000
New order (NORD)	576,031,500,000	576,031,500,000
Order line (ORDL)	19,201,053,950,825	19,201,053,950,825
Stock (STOK)	6,400,350,000,000	6,400,350,000,000
Item (ITEM)	100,000	100,000

## 4.2 Distribution of tables and logs

The distribution of tables and logs across all media must be explicitly depicted for the tested and priced systems.

The TDSQL cluster automatically distributes the data evenly to different data nodes based on the hash scheme. All data are stored on Data nodes. It automatically splits a large table horizontally into different data sets by setting the shard key when the table is created. Spreading and accessing data are all based on the hashing of the sharded keys. Each data set of the TDSQL cluster has one primary replica, one replication replica, and one log replica. The primary replica and replication/log replica use a strong synchronization mechanism to synchronize data. Primary and replication replicas have the data and log files, while the log replica only has the log files. So in case any of primary replica failed due to Data node failed, the replication replica will be the new leader to its failed primary replica. The log replica involves the consensus election but cannot be the leader. Tables except ITEM are partitioned horizontally by the shard key. The ITEM table does not have a shard key. It is a global table stored on every data set in the cluster.

The database data files and logs are stored on a set of 12 3.5TB disks configured as RAID0 in each data node.

## 4.3 Data model and database interface

A statement must be provided that describes:

1. The data model implemented by the DBMS used (e.g., relational, network, hierarchical).
2. The database interface (e.g., embedded, call level) and access language (e.g., SQL, DL/1, COBOL read/write) used to implement the TPC-C transactions. If more than one interface/access language is used to implement TPC-C, each interface/access language must be described and a list of which interface/access language is used with which transaction type must be disclosed.

The TDSQL is a distributed relational database management system. The RTE module and the SUT's Web request processing

are both implemented in the Go language. The components of the database are programmed in C/C++. The Web Server uses standard SQL, including DML and stored procedures, to interact with the database.

The Client node receives all requests and distributes the requests to the back-end database layer according to the shard key. The Data node that receives the request executes the SQL statements and returns the results to the Client node. The Client node aggregates all the returns from the database layer and returns to the RTE via Web Server.

## 4.4 The mapping of database partitions/replications

*The mapping of database partitions/replications must be explicitly described.*

All tables are horizontally partitioned, and each partition has three replicas spread in different nodes, except for the ITEM table, which has the full copy on all data nodes. Each Data node has a set of three replicas, and each set is responsible for 96 partitions, so in total in the cluster, there exist 158400 unique partitions. Specifically, the rows of the horizontally partitioned tables are hashed by the shard key field, and then the corresponding partition ID is obtained by the modulo of 158400, and then the corresponding set is obtained by Partition\_ID/96. Then the IP address of the specific database is calculated by querying information from the routing table of the set and the physical machines.

Warehouse		District		Customer		History		Stock	
sharded key: w_id	w_id	sharded key: d_w_id	d_w_id	sharded key: c_w_id	c_w_id	sharded key: h_c_w_id	h_c_id	sharded key: s_w_id	s_w_id
	w_name		d_id		c_d_id		h_c_d_id		s_i_id
	w_street_1		d_name		c_id		h_c_w_id		s_quantity
	w_street_2		d_street_1		c_first		h_d_id		s_dist_01
	w_city		d_street_2		c_middle		h_w_id		s_dist_02
	w_state		d_city		c_last		h_date		s_dist_03
	w_zip		d_state		c_street_1		h_amount		s_dist_04
	w_tax		d_zip		c_street_2		h_data		s_dist_05
	w_ytd		d_tax		c_city		Order_line		s_dist_06
<b>Orders</b>		d_state	sharded key: ol_w_id	ol_w_id	s_dist_07				
sharded key: o_w_id	o_w_id	d_ytd		ol_d_id	s_dist_08				
	o_d_id	sharded key: no_w_id	ol_o_id	s_dist_09					
	o_id		no_w_id	ol_number	s_dist_10				
	o_c_id	no_d_id	ol_i_id	s_ytd					
	o_entry_d	no_o_id	ol_supply_w_id	s_order_cnt					
	o_carrier_id	Item is broadcast table, all sets has the same data		ol_delivery_d	s_remote_cnt				
	o_ol_cnt			ol_quantity	s_data				
	o_all_local			ol_amount					
		d_next_o_id	c_zip	ol_dist_info					
			c_phone						
			c_since						
			c_credit						
			c_credit_lim						
			c_discount						
			c_balance						
			c_ytd_payment						
			c_payment_cnt						
			c_delivery_cnt						
			c_data						

Figure 4: Database schema and partitions.

## 4.5 60 Day Space Computation

*Details of the 60-day space computations along with proof that the database is configured to sustain 8 hours of growth for the dynamic tables (Order, Order-Line, and History) must be disclosed (see Clause 4.2.3).*

The details of the 60-day space computations of 1,650 Data nodes are described as follows.

Table 6: 60-day space computations (Note: All numbers about space are in GB).

#Warehouse	64,003,500	tpmC	814,854,791
#Node	1,650	Storage per Node	42,000

Table statistics							
Table	Rows	Data	Index	Data-free	5% Space	8-H Space	Total Space
Warehouse	64,003,500	15.00		0.00	0.75		15.75
District	640,035,000	138.82		3.22	7.10		149.14
Customer	1,920,105,000,000	2,378,710.37	255,114.84	1,655.68	131,774.04		2,767,254.94
Item	100,000	30.74		22.56	2.66		55.96
Stock	6,400,350,000,000	4,159,728.86		1,662.94	208,069.59		4,369,461.39
New-orders	576,031,500,000	35,739.62		1,672.80	1,870.62		39,283.04
Orders	1,920,105,000,000	172,919.90	115,766.69	1,593.42		35,224.23	325,504.24
Order-line	19,201,053,950,825	2,646,499.23		1,655.62		539,098.66	3,187,253.51
History	1,920,105,000,000	273,400.46		1,608.87		55,692.37	330,701.70
Total	-	9,667,183.01	370,881.53	9,875.11	341,724.77	630,015.26	11,019,679.68
Space computation							
Free Space		9,875.11		Storage per Node		42,000	
Dynamic Space		3,092,819.59		Total Storage		69,300,000	
Static Space		6,945,244.94		60-Day Space		44,746,160.54	
Daily Growth		630,015.26		Remaining Space		24,553,839.46	
Daily Spread		0					

# 5 Clause 5: Performance Metrics and Response

## Time Related Items

### 5.1 Measured tpmC

*Measure tpmC must be reported.*

The measured tpmC is 814,854,791.

### 5.2 Response Times

*Ninetieth percentile, maximum and average response times must be reported for all transaction types as well as for the menu response time.*

The detailed statistics are reported as part of the Numerical Quantities.

### 5.3 Keying and Think Times

*The minimum, the average, and the maximum keying and think times must be reported for all transaction types.*

The detailed statistics are reported as part of the Numerical Quantities.

### 5.4 Response Time Frequency Distribution Curves

*Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type.*

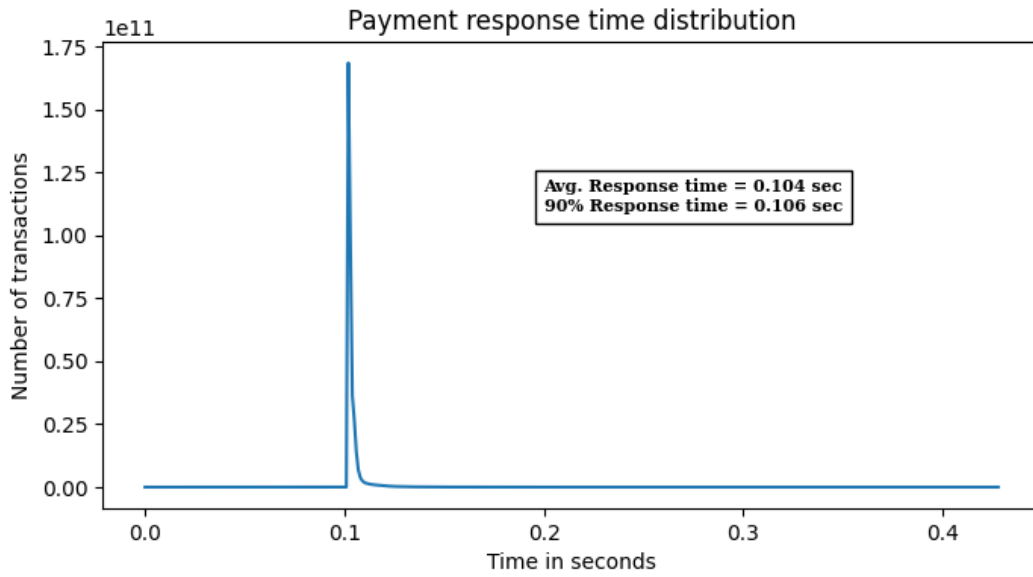


Figure 5: Frequency distribution of response times for Payment

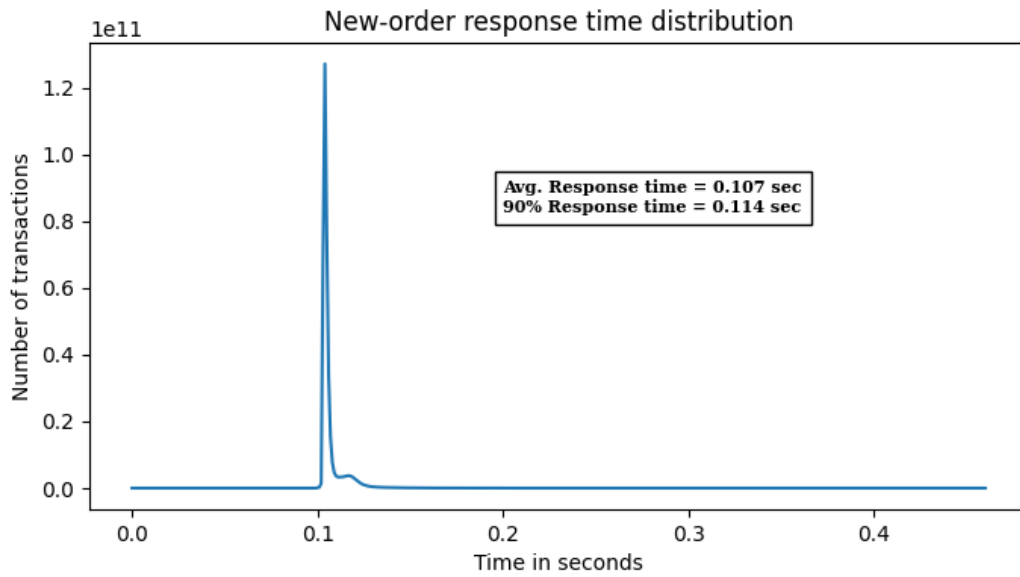


Figure 6: Frequency distribution of response times for New-order.

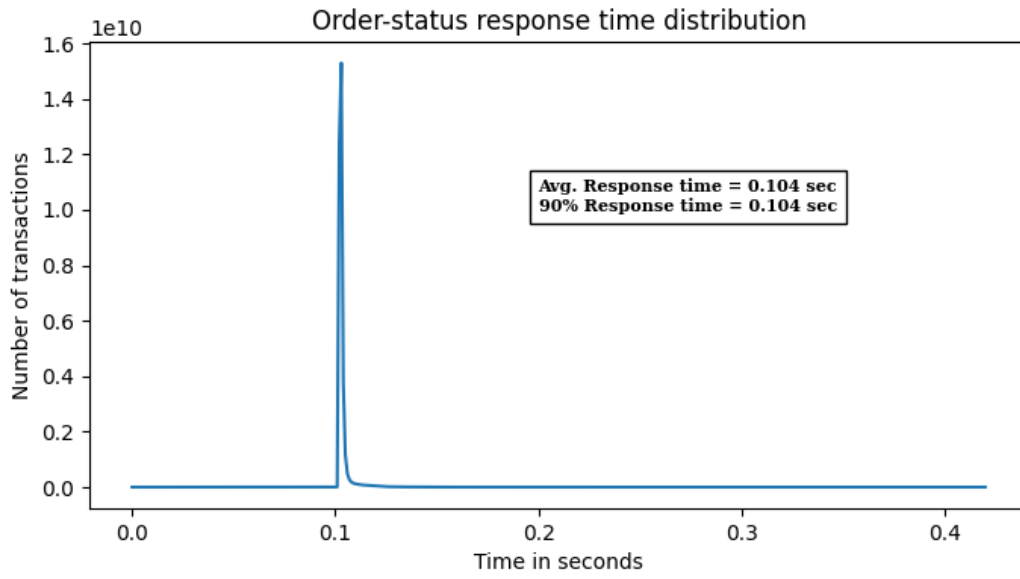


Figure 7: Frequency distribution of response times for Order-status

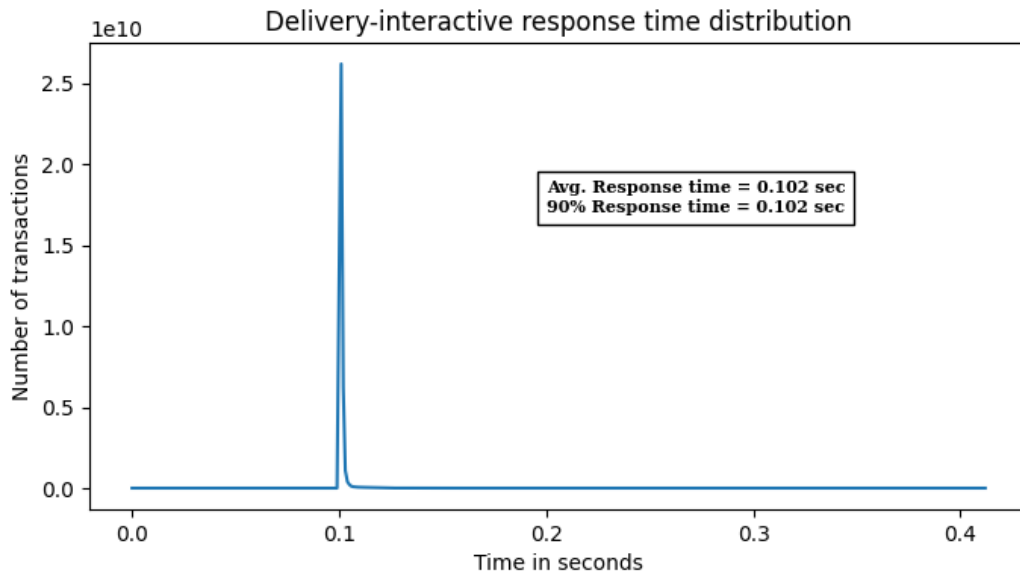


Figure 8: Frequency distribution of response times for Delivery-interactive



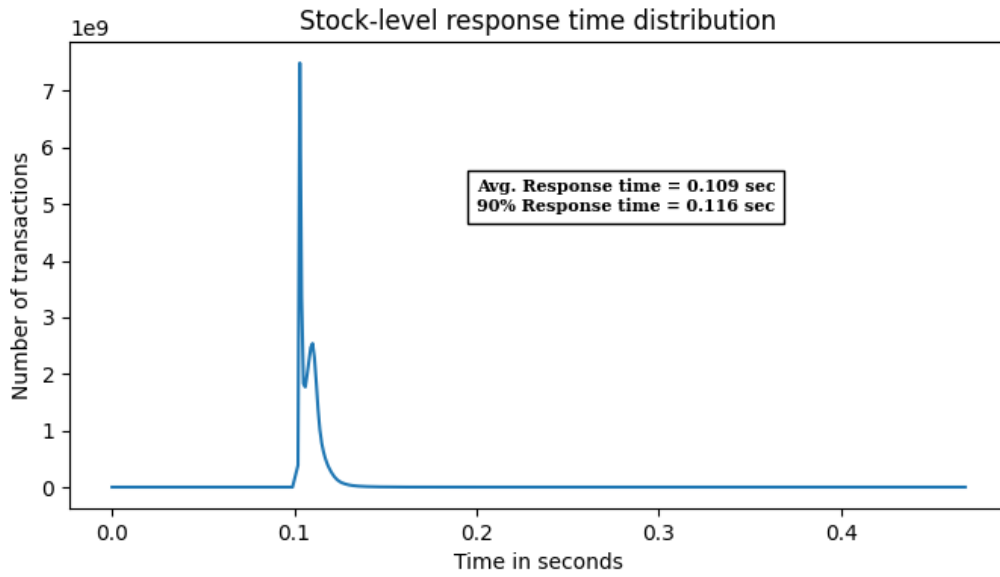


Figure 9: Frequency distribution of response times for Stock-level

## 5.5 Think Time Frequency Distribution

Think Time frequency distribution curves (see Clause 5.6.3) must be reported for the New-Order transaction.



Figure 10: Frequency distribution of think times for New-order

## 5.6 Response Times versus Throughput

The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New-Order transaction.

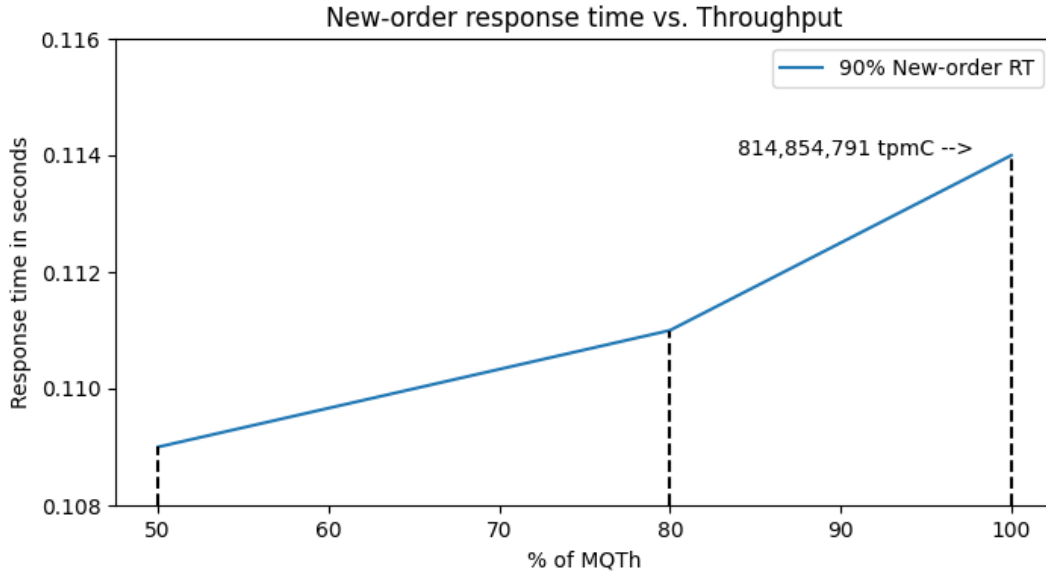


Figure 11: New-order response time versus throughput

## 5.7 Throughput versus Elapsed Time

A graph of throughput versus elapsed time (see Clause 5.6.4) must be reported for the New-Order transaction.

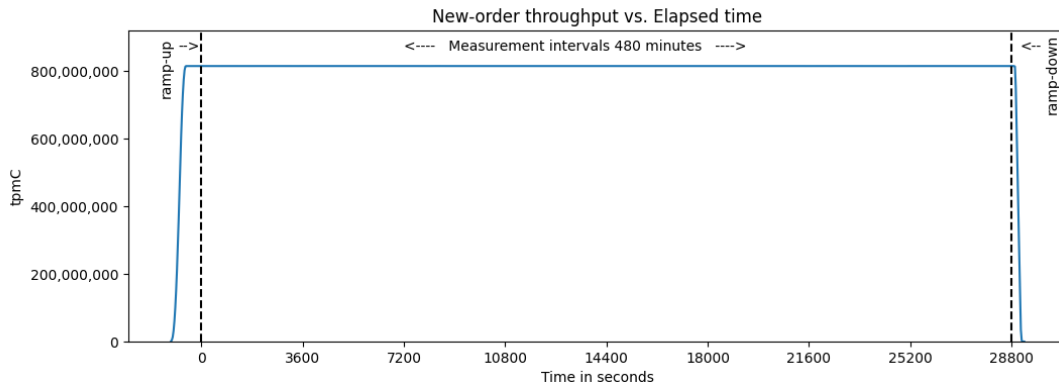


Figure 12: Throughput versus elapsed time.

## 5.8 Steady State Determination

The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval (see Clause 5.5) must be described.

Figure 12 depicts the New-order throughput versus elapsed time. It is the ramp-up period when the benchmark begins. With around 20 minutes of ramp-up period, the throughput starts to increase and reaches a steady state. We consider the valid 480-minute measurement time of steady state from at 20 minutes to at 500 minutes after benchmark begins. With around 20 minutes of ramp-down period, the throughput starts to decrease and down to 0 tpmC. The whole period of steady state remains the volatility of tpmC less than 1%.

## 5.9 Work Performance During Steady State

A description of how the work normally performed during a sustained test (for example checkpointing, writing redo/undo log

*records, etc.), actually occurred during the measurement interval must be reported.*

During the period of the sustained test, TDSQL guarantees all ACID properties required by the benchmark specification. The RTE submit the transaction requests via HTTP to Web server in the Client node. The transactions are one of five standard transactions by the requirement specification. The RTE will wait some time for simulating inputting data and thinking of purchase. The RTE also will record the time it submits the data to the server, and it receives the response from the server. The RTE will repeat the above steps till the test completes.

For the SUT side, TDSQL cluster receives the transactions and executes queries in a transaction sequentially. The TDSQL will guarantee all ACID properties. At the time of committing transactions, the data are synchronized to replication (data and redo logs) and log (only redo logs) replicas. The transactions are labeled as committed only if it receives synchronized ACK from its replication or log replica. The checkpoint time interval in TDSQL is set to 1 second for this test. So, when fault happens in memory, it can easily recover from redo log, and when fault happens in machine, the replication can be switch to primary (leader) to provide service.

## 5.10 Measurement Interval

*A statement of the duration of the measurement interval for the reported Maximum Qualified Throughput (tpmC) must be included.*

Exact 8 hours (8hr = 480min = 28800sec) are reported for the steady performance of Maximum Quantifies Throughput (tpmC) as the measurement interval.

## 5.11 Transaction Mix Regulation

*The method of regulation of the transaction mix (e.g., card decks or weighted random distribution) must be described. If weighted distribution is used and the RTE adjusts the weights associated with each transaction type, the maximum adjustments to the weight from the initial value must be disclosed.*

The weighted distribution algorithm in RTE controls the transaction mix percentage concerning the specification requirements. The weight of each transaction was assigned before the test and were fixed during the entire test.

## 5.12 Transaction Mix

*The percentage of the total mix for each transaction type must be disclosed.*

The detailed statistics are reported as part of the Numerical Quantities.

## 5.13 Percentage of New-Order Transactions

*The percentage of New-Order transactions rolled back as a result of invalid item number must be disclosed.*

The detailed statistics are reported as part of the Numerical Quantities.

## 5.14 Number of Order-lines per New-Order

*The average number of order-lines entered per New-Order transaction must be disclosed.*

The detailed statistics are reported as part of the Numerical Quantities.

## 5.15 Percentage of Remote Order-lines per New-Order

*The percentage of remote order-lines entered per New-Order transaction must be disclosed.*

*a) Percentage of Remote Payments*

*The percentage of remote payment transactions must be disclosed.*

*b) Percentage of access customer by C\_LAST for Payment and Order-Status*

*The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed.*

*c) Percentage of Skipped Delivery Transactions*

*The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed.*

*d) Checkpoints*

*The number of checkpoints in the Measurement Interval, the time in seconds from the start of the Measurement Interval to the first checkpoint and the Checkpoint Interval must be disclosed*

The detailed statistics are reported as part of the Numerical Quantities.

## 5.16 Percentage of Remote Payments

*The percentage of remote payment transactions must be disclosed.*

The detailed statistics are reported as part of the Numerical Quantities.

## 5.17 Percentage of access customer by C\_LAST for Payment and Order-Status

*The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed.*

The detailed statistics are reported as part of the Numerical Quantities.

## 5.18 Percentage of Skipped Delivery Transactions

*The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed.*

The detailed statistics are reported as part of the Numerical Quantities.

## 5.19 Checkpoints

*The number of checkpoints in the Measurement Interval, the time in seconds from the start of the Measurement Interval to the first checkpoint and the Checkpoint Interval must be disclosed.*

Checkpoints in measurement interval per Node (Min/Avg/Max) 27688/27761.46/53040.  
The current Maximum Checkpoint Interval (All Nodes) is 10.620 second.

The check point history is from information\_schema (INNODB\_CHECKPOINT\_HISTORY) in the table of each node. Each

database node performs checkpoint independently. As the server appends redo log rather than writing log in cycle, it has unlimited log spaces in theory, and a background thread lazily advances checkpoint periodically (by configuring `innodb_log_checkpoint_every`). The current value for `innodb_log_checkpoint_every` is 1 second. Also, we can define a soft limit of uncheckpointed log space such that If reaching some limits, the background page cleaner threads will do more aggressive page flushing, so checkpoint can be advanced (by configuring `innodb_log_max_checkpoint_files`). The current value for `innodb_log_max_checkpoint_files` is 8 GB.

# 6 Clause 6: SUT, Driver and Communications

## Related Items

### 6.1 RTE Description

*If the RTE is commercially available, then its inputs must be specified. Otherwise, a description must be supplied of what inputs (e.g. scripts) to the RTE had been used.*

The RTE module and the SUT’s Web request processing are both internally implemented by the Go language. Each RTE machine starts processes and keeps long HTTP connections to Nginx Servers. The RTE involves several jobs. The RTE first generates random parameter values that meet the specification requirement such as keying and think time. Then it establishes the simulation model and initializes the request to interact with SUT. The RTE is also responsible for collecting the required statistics such as response time, and complete or abort state. The RTE runs in the whole period of the test run, including ramp-up, measurement interval, and ramp-down time. Table 5 shows the main parameter configuration of RTE.

Table 7: Parameter configuration of RTE.

Parameter	Value	Description
warehouse-count	64,003,500	current warehouse number
http-conn-count	640,035,000	simulating client number
RTE nodes/instances	495	RTE node/instance number
data-path	"./data"	path of statistical data
http-idle-duration	"1h"	http connection max idle duration
ramp-up-time	"20m"	warmup time
ramp-down-time	"10m"	ramp down time
measurement-interval	"8h"	time of steady running
print-interval	"10s"	print interval, less than 0s will not print
emulated-display-delay	"100ms"	Emulated Display Delay

### 6.2 Number of Terminal Connection Lost

*The number of terminal connections lost during the Measurement Interval must be disclosed (see Clause 6.6.2).*

There is no terminal connection lost during the measurement interval.

### 6.3 Emulated Components

*It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system. The results of the test described in Clause 6.6.3.4 must be disclosed.*

The SUT provides the service to the emulated system via HTTP. Each RTE machine starts 24 processes, and each process keeps 53875 long HTTP connections to SUT's Nginx servers.

## 6.4 Configuration Diagrams

*A complete functional diagram of both the benchmark configuration and the configuration of the proposed (target) system must be disclosed. A detailed list of all software and hardware functionality being performed on the Driver System, and its interface to the SUT must be disclosed (see Clause 6.6.3.6).*

The measured and priced configurations are shown in Figure 1 and Figure 2, respectively.

## 6.5 Network Configuration

*The network configurations of both the tested services and the proposed (target) services, which are being represented, and a thorough explanation of exactly which parts of the proposed configuration are being replaced with the Driver System must be disclosed (see Clause 6.6.4).*

All machines are in one network zone with a cluster of 100-Gbps switches.

## 6.6 Operator Intervention

*If the configuration requires operator intervention, the mechanism and the frequency of this intervention must be disclosed.*

No human intervention is involved during the entire measurement interval. The TDSQL cluster is highly automatic and smartly distributed to maintain steady performance with high transactional throughput.

# 7 Clause 7: Pricing Related Items

## 7.1 Hardware and software Price

*A detailed list of hardware and software used in the priced system must be reported. Each separately orderable item must have vendor part number, description, release/revision level, and either general availability status or committed delivery date. If package pricing is used, vendor part number of the package and a description uniquely identifying each of the components of the package must be disclosed. Pricing source(s) and effective date(s) of price(s) must also be reported.*

The detailed list of hardware and software used in the price system is reported as part of the Executive Summary. The Third-Party Pricing Information is provided in Appendix D

## 7.2 Total 3-Year Cost

*The total 3-year price of the entire Priced Configuration must be reported, including: hardware, software, and maintenance charges. The justification of any Discounts applied must be disclosed in the price sheet. Sufficient detail of what items are being discounted and by how much they are being discounted must be provided so that the Discount amount used in the computation of the total system cost can be independently reproduced.*

The total 3-year price of the entire Price Configuration is reported as part of the Executive Summary.

## 7.3 Availability Date

*The Committed delivery date for general availability (availability date) of products used in the price calculations must be reported. The Availability Date must be reported on the first page of the Executive Summary and with a precision of one day. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all Components are committed to be Generally Available. Each Component used in the Priced Configuration is considered to be Available on the Availability Date unless an earlier date is specified.*

All products in the price calculation are available on June 18, 2023.

## 7.4 Hardware and Software Support

Tencent Cloud provides Enterprise-level Client Services, including supporting all CVM instances and physical machines with 7x24 hours phone services.

TDSQL team provides professional support of database systems with 7x24 hours phone services and Technical Support.

OpenResty Inc. provides Nginx support service with 7x24 hours services.

## 7.5 Statement of measured tpmC and Price/Performance

*A statement of the measured tpmC, as well as the respective calculations for 3 -year pricing, price/performance (price/tpmC), and the availability date must be included.*



Table 8: Statement of tpmC and price/performance.

Total System Cost	TPC-C® Throughput	Price/Performance	Availability Date
Three-year cost of hardware, software, and maintenance	Maximum Qualified Throughput by transaction per minute - C (tpmC)	Total System Cost/tpmC	Date for all hardware, software, and maintenance available to purchase
<b>1,031,746,953 CNY</b>	<b>814,854,791 tpmC</b>	<b>1.27 CNY/tpmC</b>	<b>June 18, 2023</b>

## 7.6 Country Specific Pricing

*Additional Clause 7 related items may be included in the Full Disclosure Report for each country specific priced configuration. Country specific pricing is subject to Clause 7.1.7.*

All components of the Priced Configuration are priced by Renminbi or Chinese Yuan (standard code CNY), which is the official currency of People's Republic of China.

## 7.7 Orderability Date

*For each of the components that are not orderable on the report date of the FDR, the following information must be included in the FDR:*

- *Name and part number of the item that is not orderable*
- *The date when the component can be ordered (on or before the Availability Date)*
- *The method to be used to order the component (at or below the quoted price) when that date arrives*
- *The method for verifying the price*

All components of the Priced Configuration are available on June 18, 2023.

# 8 Clause 8: Auditor Attestation

## 8.1 Auditor Information

*The auditor's name, address, phone number, and a copy of the auditor's attestation letter indicating compliance must be included in the Full Disclosure Report*

This benchmark was audited by:  
PerfLabs, Inc. DBA InfoSizing  
Doug Johnson  
63 Lourdes Drive Leominster, MA 01453, USA  
Phone: +1 (978) 343-6562  
[www.sizing.com](http://www.sizing.com)

## 8.2 Attestation Letter

The Auditor's Attestation Letter is included in the following pages.

Anqun Pan  
TEG Database R&D Department  
Tencent Inc.  
No. 16 Gaoxin South Avenue 10  
Shenzhen, China

March 14, 2023

I verified the TPC Benchmark™ C v5.11.0 performance of the following configuration:

Platform: Tencent Database Dedicated Cluster (with 1,650 TDSQL Data Nodes)  
Operating System: Tencent tlinux 2.2  
Database Manager: Tencent TDSQL v10.3 Enterprise Pro Edition with Partitioning and Physical Replication

The results were:

**Performance Metric** 814,854,791 tpmC  
**Number of Users** 640,035,000

<u>Server</u>	<u>Tencent Database Dedicated Cluster</u>															
Nodes	1,650 Data Nodes, 3 Manager Nodes, 495 Client Nodes															
CPUs	2x Intel® Xeon® Platinum 8255C (2.50 GHz, 24-core) (Data/Mgr. Nodes) 2x Intel® Xeon® Platinum 8361HC (2.60 GHz, 24-core) (Client Nodes)															
Memory	768 GB (Data/Mgr. Nodes) 288 GB (Client Nodes)															
Storage	<table border="1"> <thead> <tr> <th>Qty</th> <th>Size</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>447 GB</td> <td>OS Disk (Data/Mgr. Nodes)</td> </tr> <tr> <td>12</td> <td>3.5 TB</td> <td>NVMe SSD (Data/Mgr. Nodes)</td> </tr> <tr> <td>1</td> <td>100 GB</td> <td>OS Disk (Client Nodes)</td> </tr> <tr> <td>1</td> <td>500 GB</td> <td>Data Disk (Client Nodes)</td> </tr> </tbody> </table>	Qty	Size	Type	1	447 GB	OS Disk (Data/Mgr. Nodes)	12	3.5 TB	NVMe SSD (Data/Mgr. Nodes)	1	100 GB	OS Disk (Client Nodes)	1	500 GB	Data Disk (Client Nodes)
Qty	Size	Type														
1	447 GB	OS Disk (Data/Mgr. Nodes)														
12	3.5 TB	NVMe SSD (Data/Mgr. Nodes)														
1	100 GB	OS Disk (Client Nodes)														
1	500 GB	Data Disk (Client Nodes)														

In my opinion, these performance results were produced in compliance with the TPC requirements for the benchmark.

The following verification items were given special attention:

- The transactions were correctly implemented.
- The database records were the proper size.
- The database was properly scaled and populated.
- The ACID properties were met.

- Input data was generated according to the specified percentages.
- The transaction cycle times included the required keying and think times.
- The reported response times were correctly measured.
- At least 90% of all delivery transactions met the 80 Second completion time limit.
- All 90% response times were under the specified maximums.
- The measurement interval was representative of steady state conditions.
- The reported measurement interval was over 120 minutes.
- Checkpoint intervals were under 30 minutes.
- The 60-day storage requirement was correctly computed.
- The system pricing was verified for major components and maintenance.

**Additional Audit Notes:**

None.

Respectfully Yours,

A handwritten signature in cursive script that reads "Doug Johnson". The signature is written in black ink and has a long, sweeping horizontal line extending to the right.

**Doug Johnson, Certified TPC Auditor**

63 Lourdes Dr. | Leominster, MA 01453 | 978-343-6562 | [www.sizing.com](http://www.sizing.com)

# Appendix A: Source Code File List

The source code and scripts used to implement the benchmark are provided as a soft appendix. This soft appendix includes the following files:

```
|-- load
| |-- load_data
| |   |-- src
| | |   |-- load_main.cc
| | |   |-- load_set_main.cc
| | |   |-- murmurhash.h
| | |   |-- parse_port.h
| | |   |-- spt_proc.c
| | |   |-- spt_proc.h
| | |   |-- tpc.h
| |   |-- sql
| | |   |-- create-procedure-sut.sql
| | |   |-- create_table.sql
|-- tpc-for-tdsql-sut
| |-- cmd
| |   |-- sut-server
| | |   |-- main.go
| |-- conf
| |   |-- sut.toml.template
|-- nginx_for_tpcc_template
| |-- html
| |   |-- 50x.html
| |   |-- check.js
|-- pkg
| |-- logutil
| |   |-- logutil.go
| |-- typeutil
| |   |-- bfw.go
| |   |-- duration.go
| |   |-- string_byte.go
|-- server
| |-- conf
| |   |-- conf.go
|-- driver
| |-- detail_procedure.go
|-- http_tdsq_handler.go
|-- http_tdsq_router.go
|-- server.go
|-- tdsq_conn.go
|-- tdsq_delivery_goroutine_pool.go
|-- tdsq_delivery_processor.go
|-- tdsq_delivery_task_queue.go
|-- tdsq_new_order_processor.go
|-- tdsq_order_status_processor.go
|-- tdsq_payment_processor.go
|-- tdsq_stock_level_processor.go
|-- tpc_para_pools.go
|-- tpc_txn.go
|-- util.go
|-- utils
| |-- buffer.go
| |-- buffer_test.go
| |-- byte_slice_pool.go
| |-- bytes_buffer_pool.go
| |-- nosaferand.go
| |-- zeroalloc.go
```

17 directories, 39 files

# Appendix B: Database design

The schema of the database system is provided as follows.

## B.1 Table creation

```
CREATE TABLE if not exists warehouse (  
  w_id INT NOT NULL,  
  w_name VARCHAR(10),  
  w_street_1 VARCHAR(20),  
  w_street_2 VARCHAR(20),  
  w_city VARCHAR(20),  
  w_state CHAR(2),  
  w_zip CHAR(9),  
  w_tax DECIMAL(4 , 4 ),  
  w_ytd DECIMAL(12 , 2 ),  
  PRIMARY KEY (w_id)  
) ENGINE=INNODB DEFAULT CHARSET=latin1 COLLATE=latin1_bin shardkey=w_id;
```

```
CREATE TABLE if not exists district (  
  d_w_id INT NOT NULL,  
  d_id TINYINT NOT NULL,  
  d_name VARCHAR(10),  
  d_street_1 VARCHAR(20),  
  d_street_2 VARCHAR(20),  
  d_city VARCHAR(20),  
  d_state CHAR(2),  
  d_zip CHAR(9),  
  d_tax DECIMAL(4 , 4 ),  
  d_ytd DECIMAL(12 , 2 ),  
  d_next_o_id INT,  
  PRIMARY KEY (d_w_id , d_id)  
) ENGINE=INNODB DEFAULT CHARSET=latin1 COLLATE=latin1_bin shardkey=d_w_id;
```

```
CREATE TABLE if not exists customer (  
  c_w_id INT NOT NULL,  
  c_d_id TINYINT NOT NULL,  
  c_id INT NOT NULL,  
  c_first VARCHAR(16) NOT NULL,  
  c_middle CHAR(2),  
  c_last VARCHAR(16) NOT NULL,  
  c_street_1 VARCHAR(20),  
  c_street_2 VARCHAR(20),  
  c_city VARCHAR(20),  
  c_state CHAR(2),  
  c_zip CHAR(9),  
  c_phone CHAR(16),  
  c_since DATETIME,  
  c_credit CHAR(2),  
  c_credit_lim DECIMAL(12 , 2 ),  
  c_discount DECIMAL(4 , 4 ),  
  c_balance DECIMAL(12 , 2 ),  
  c_ytd_payment DECIMAL(12 , 2 ),  
  c_payment_cnt SMALLINT,  
  c_delivery_cnt INT,  
  c_data TEXT,  
  KEY `idx_customer` (`c_w_id` , `c_d_id` , `c_last` , `c_first`),  
  PRIMARY KEY (c_w_id , c_d_id , c_id)  
) ENGINE=INNODB DEFAULT CHARSET=latin1 COLLATE=latin1_bin shardkey=c_w_id;
```

```

CREATE TABLE  if not exists history (
  h_c_id INT,
  h_c_d_id TINYINT,
  h_c_w_id INT,
  h_d_id TINYINT,
  h_w_id INT,
  h_date DATETIME,
  h_amount DECIMAL(6 , 2 ),
  h_data VARCHAR(24),
) ENGINE=INNODB DEFAULT CHARSET=latin1 COLLATE=latin1_bin shardkey=h_c_w_id;

CREATE TABLE  if not exists new_orders (
  no_w_id INT NOT NULL,
  no_d_id TINYINT NOT NULL,
  no_o_id INT NOT NULL,
  PRIMARY KEY (no_w_id , no_d_id , no_o_id)
) ENGINE=INNODB DEFAULT CHARSET=latin1 COLLATE=latin1_bin shardkey=no_w_id;

CREATE TABLE  if not exists orders (
  o_w_id INT NOT NULL,
  o_d_id TINYINT NOT NULL,
  o_id INT NOT NULL,
  o_c_id INT NOT NULL,
  o_entry_d DATETIME,
  o_carrier_id TINYINT,
  o_ol_cnt TINYINT,
  o_all_local TINYINT,
  KEY `idx_orders` (`o_w_id` , `o_d_id` , `o_c_id` , `o_id`),
  PRIMARY KEY (o_w_id , o_d_id , o_id)
) ENGINE=INNODB DEFAULT CHARSET=LATIN1 COLLATE = LATIN1_BIN shardkey=o_w_id;

CREATE TABLE  if not exists order_line (
  ol_w_id INT NOT NULL,
  ol_d_id TINYINT NOT NULL,
  ol_o_id INT NOT NULL,
  ol_number TINYINT NOT NULL,
  ol_i_id INT,
  ol_supply_w_id INT,
  ol_delivery_d DATETIME,
  ol_quantity TINYINT,
  ol_amount DECIMAL(6 , 2 ),
  ol_dist_info CHAR(24),
  PRIMARY KEY (ol_w_id , ol_d_id , ol_o_id , ol_number)
) ENGINE=INNODB DEFAULT CHARSET=latin1 COLLATE=latin1_bin  shardkey=ol_w_id ;

CREATE TABLE  if not exists item (
  i_id INT NOT NULL,
  i_im_id INT,
  i_name VARCHAR(24),
  i_price DECIMAL(5 , 2 ),
  i_data VARCHAR(50),
  PRIMARY KEY (i_id)
) ENGINE=INNODB DEFAULT CHARSET=latin1 COLLATE=latin1_bin  shardkey=noshardkey_allset;

CREATE TABLE  if not exists stock (
  s_w_id INT NOT NULL,
  s_i_id int NOT NULL,
  s_quantity SMALLINT,
  s_dist_01 CHAR(24),
  s_dist_02 CHAR(24),
  s_dist_03 CHAR(24),

```

```

s_dist_04 CHAR(24),
s_dist_05 CHAR(24),
s_dist_06 CHAR(24),
s_dist_07 CHAR(24),
s_dist_08 CHAR(24),
s_dist_09 CHAR(24),
s_dist_10 CHAR(24),
s_ytd DECIMAL(8,0),
s_order_cnt SMALLINT,
s_remote_cnt SMALLINT,
s_data VARCHAR(50),
PRIMARY KEY (s_w_id, s_i_id)
) ENGINE=INNODB DEFAULT CHARSET=LATIN1 COLLATE = LATIN1_BIN shardkey=s_w_id;

```

## B.2 Procedure creation

```

/*sets:allsets*/DROP PROCEDURE IF EXISTS new_order_tpcc;
/*sets:allsets*/DROP PROCEDURE IF EXISTS payment_tpcc;
/*sets:allsets*/DROP PROCEDURE IF EXISTS order_status_tpcc;
/*sets:allsets*/DROP PROCEDURE IF EXISTS delivery_tpcc;
/*sets:allsets*/DROP PROCEDURE IF EXISTS stock_level_tpcc;

-- new order
delimiter $$
/*sets:allsets*/CREATE PROCEDURE new_order_tpcc(
  in param_w_id int,
  in param_d_id int,
  in param_c_id int,
  in param_o_ol_cnt int,
  in param_o_all_local int,
  in param_itemid varchar(512), -- Encoding for item id: concat_ws(",", item1, item2, ...)
  in param_supware varchar(512), -- Encoding for item warehouse: concat_ws(",", warehouse1, ...)
  in param_qyt varchar(512)) -- Encoding for item qty: concat_ws(",", q1, q2, ...)
label:begin

  -- declare variables
  declare var_ol_number int default 1; -- cycles
  declare var_i_price decimal(5,2) default NULL;
  declare var_i_name varchar(24) default NULL;
  declare var_i_data varchar(50) default NULL;
  declare var_s_quantity smallint default NULL;
  declare var_s_data varchar(50) default NULL;
  declare var_ol_dist_info varchar(24) default NULL;
  declare var_bg varchar(2) default NULL;
  declare var_ol_amount decimal(6,2) default NULL;
  declare var_ol_supply_w_id int default NULL; -- item warehouse
  declare var_ol_i_id int default NULL; -- item id
  declare var_ol_quantity int default NULL; -- item qty
  declare var_remote int default NULL; -- item is remote supply

  declare err int;
  declare n_var_datetime char(81);
  declare n_var_c_last varchar(16);
  declare n_var_c_credit char(2);
  declare n_var_c_discount decimal(4,2);
  declare n_var_d_next_o_id int;
  declare n_var_w_tax decimal(4,2);
  declare n_var_d_tax decimal(4,2);
  declare n_var_amount_total decimal(8,2);
  declare n_item_data text(501);

  -- defining error handling
  -- declare continue HANDLER FOR SQLWARNING,NOT FOUND,SQLException
  declare exit HANDLER FOR NOT FOUND,SQLException

```



```

begin
  set err = -1;
  select
err,n_item_data,n_var_amount_total,n_var_c_credit,n_var_c_discount,n_var_c_last,n_var_d_next_o_id,n_var_d_tax,n_var_d
atetime,n_var_w_tax;
  rollback;
end;
START TRANSACTION;
set err = 0;

select CURRENT_TIMESTAMP() into n_var_datetime;
set n_var_amount_total=0;

-- query the user's discount, tax and so on
SELECT c_discount, c_last, c_credit, w_tax INTO n_var_c_discount, n_var_c_last, n_var_c_credit, n_var_w_tax
FROM customer, warehouse
  WHERE w_id = param_w_id AND c_w_id = w_id AND c_d_id = param_d_id AND c_id = param_c_id;

-- update the order-id and tax of the district
UPDATE district set  d_next_o_id = d_next_o_id + 1
  WHERE d_id = param_d_id AND d_w_id = param_w_id      txsql_returning  d_next_o_id-1,d_tax into
n_var_d_next_o_id,n_var_d_tax;

-- create order
INSERT INTO orders(o_id,o_d_id,o_w_id,o_c_id,o_entry_d,o_ol_cnt,o_all_local)
  VALUES (n_var_d_next_o_id, param_d_id, param_w_id, param_c_id, n_var_datetime, param_o_ol_cnt,
param_o_all_local);

-- create unshipped orders
INSERT INTO new_orders (no_o_id, no_d_id, no_w_id)
  VALUES (n_var_d_next_o_id, param_d_id, param_w_id);

-- circulation processing for every item
while var_ol_number <= param_o_ol_cnt do
  -- decoding warehouse, item-id, item-qty
  set var_ol_supply_w_id= substring_index(substring_index(param_supware, ',', var_ol_number), ',', -1);
  set var_ol_i_id= substring_index(substring_index(param_itemid, ',', var_ol_number), ',', -1);
  set var_ol_quantity= substring_index(substring_index(param_qty, ',', var_ol_number), ',', -1);
  set var_remote = 0;
  if var_ol_supply_w_id != param_w_id then
    set var_remote = 1;
  end if;

  -- if the number of affected rows is not 1, the next item execution is processed
  -- get stock info
  UPDATE stock set s_order_cnt = s_order_cnt + 1,s_ytd = s_ytd + var_ol_quantity,s_remote_cnt = s_remote_cnt +
var_remote,
  s_quantity = (case when s_quantity >= var_ol_quantity then s_quantity else s_quantity + 91 end ) - var_ol_quantity
  WHERE s_i_id = var_ol_i_id
    AND s_w_id = var_ol_supply_w_id
    txsql_returning s_quantity,s_data,
    (case param_d_id
      when 1 then s_dist_01
      when 2 then s_dist_02
      when 3 then s_dist_03
      when 4 then s_dist_04
      when 5 then s_dist_05
      when 6 then s_dist_06
      when 7 then s_dist_07
      when 8 then s_dist_08
      when 9 then s_dist_09
      else s_dist_10
    end ) as dist
  into var_s_quantity, var_s_data, var_ol_dist_info;

  -- get item info
  SELECT i_price, i_name, (case when locate("original", i_data) and locate("original", var_s_data) then 'B' else 'G' end)

```

```

as bg
    INTO var_i_price, var_i_name, var_bg FROM item WHERE i_id = var_ol_i_id;

-- calculate amount
set var_ol_amount= var_ol_quantity * var_i_price;
set n_var_amount_total=n_var_amount_total + var_ol_amount;

-- insert the order details
INSERT INTO order_line (ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id, ol_supply_w_id, ol_quantity, ol_amount,
ol_dist_info)
    VALUES (n_var_d_next_o_id, param_d_id, param_w_id, var_ol_number, var_ol_i_id, var_ol_supply_w_id,
var_ol_quantity, var_ol_amount, var_ol_dist_info);

    set n_item_data = concat_ws(',', n_item_data, var_ol_supply_w_id, var_ol_i_id, var_ol_quantity, var_i_name,
var_s_quantity, var_bg, var_i_price, var_ol_amount);
    set var_ol_number= var_ol_number+1;

end while;

set n_var_amount_total=n_var_amount_total * (1 + n_var_w_tax + n_var_d_tax) * (1 - n_var_c_discount);
if err = 0 then
    select
err,n_item_data,n_var_amount_total,n_var_c_credit,n_var_c_discount,n_var_c_last,n_var_d_next_o_id,n_var_d_tax,n_var_d
atetime,n_var_w_tax;
        commit;
    end if;
end $$
delimiter ;

-- payment
/*sets:allsets*/DROP PROCEDURE IF EXISTS payment_tpcc;
delimiter $$
/*sets:allsets*/ CREATE PROCEDURE payment_tpcc(in wid int,in account float,in did int,in byname int,in p_cwid int,in
p_cdid int,in p_cid_in int,in lastname char(17))
label:begin
    declare num int;
    declare err int;
    declare p_datetime char(81);
    declare p_wstreet1 varchar(20);
    declare p_wstreet2 varchar(20);
    declare p_wcity varchar(20);
    declare p_wstate char(2);
    declare p_wzip char(9);
    declare p_wname char(11);
    declare p_dstreet1 varchar(20);
    declare p_dstreet2 varchar(20);
    declare p_dcity varchar(20);
    declare p_dstate char(2);
    declare p_dzip char(9);
    declare p_dname char(11);
    declare p_cid int;
    declare p_cfirst varchar(16);
    declare p_cmiddle char(2);
    declare p_clast char(17);
    declare p_cstreet1 varchar(20);
    declare p_cstreet2 varchar(20);
    declare p_ccity varchar(20);
    declare p_cstate char(2);
    declare p_czip char(9);
    declare p_cphone char(16);
    declare p_ccredit char(2);
    declare p_ccreditlim bigint;
    declare p_cdiscount decimal(4,2);
    declare p_cbalance decimal(12,2);
    declare p_csince char(81);
    declare p_cdata text(502);

```

```

declare exit HANDLER FOR NOT FOUND,SQLException
begin
    set err = -1;
    select
err,p_cbalance,p_ccity,p_ccredit,p_ccreditlim,p_cdata,p_cdiscount,p_cfirst,p_cid,p_clast,p_cmiddle,p_cphone,p_csince,p_cstate,
p_cstreet1,p_cstreet2,p_czip,p_datetime,p_dcity,p_dname,p_dstate,p_dstreet1,p_dstreet2,p_dzip,p_wcity,p_wname,p_wstate,
p_wstreet1,p_wstreet2,p_wzip;
    rollback;
end;

START TRANSACTION;
set err = 0;
SELECT CURRENT_TIMESTAMP() into p_datetime;
set p_cid=p_cid_in;

if byname=1 then
    SELECT count(c_id) INTO num FROM customer WHERE c_w_id = p_cwid AND c_d_id = p_cdid AND c_last =
lastname;
    if num = 0 then
        set err = -1;
        select
err,p_cbalance,p_ccity,p_ccredit,p_ccreditlim,p_cdata,p_cdiscount,p_cfirst,p_cid,p_clast,p_cmiddle,p_cphone,p_csince,p_cstate,
p_cstreet1,p_cstreet2,p_czip,p_datetime,p_dcity,p_dname,p_dstate,p_dstreet1,p_dstreet2,p_dzip,p_wcity,p_wname,p_wstate,
p_wstreet1,p_wstreet2,p_wzip;
        rollback;
        leave label;
    end if;
    set num=floor(num/2);
    SELECT c_id into p_cid FROM customer WHERE c_w_id = p_cwid AND c_d_id = p_cdid AND c_last = lastname
ORDER BY c_first limit num,1;
end if;

UPDATE customer set c_balance= c_balance - account, c_ytd_payment = c_ytd_payment + account, c_payment_cnt =
c_payment_cnt + 1
    where c_w_id = p_cwid AND c_d_id = p_cdid AND c_id = p_cid
    txsql_returning c_first, c_middle, c_last, c_street_1, c_street_2, c_city, c_state, c_zip, c_phone, c_credit, c_credit_lim,
c_discount, c_balance, c_since
    into p_cfirst, p_cmiddle, p_clast, p_cstreet1, p_cstreet2, p_ccity, p_cstate, p_czip, p_cphone, p_ccredit, p_ccreditlim,
p_cdiscount, p_cbalance, p_csince ;

if p_ccredit="BC" then
    UPDATE customer SET c_data = substr(concat(p_cid,'',p_cdid,'',p_cwid,'',did,'',wid,'',account,'',c_data),1,500)
    WHERE c_w_id = p_cwid AND c_d_id = p_cdid AND c_id = p_cid txsql_returning substr(c_data, 1, 200) INTO
p_cdata;
else
    set p_cdata = "";
end if;

UPDATE district SET d_ytd = d_ytd + account
    WHERE d_w_id =wid AND d_id = did
    txsql_returning d_name, d_street_1, d_street_2, d_city, d_state, d_zip INTO p_dname, p_dstreet1, p_dstreet2, p_dcity,
p_dstate, p_dzip;
UPDATE warehouse SET w_ytd = w_ytd + account
    WHERE w_id =wid
    txsql_returning w_name, w_street_1, w_street_2, w_city, w_state, w_zip INTO p_wname, p_wstreet1, p_wstreet2,
p_wcity, p_wstate, p_wzip;

INSERT INTO history(h_c_d_id, h_c_w_id, h_c_id, h_d_id, h_w_id, h_date, h_amount, h_data)
    VALUES(p_cdid, p_cwid, p_cid, did, wid, p_datetime, account, concat(p_wname,'',p_dname) );

if err = 0 then
    select

```

```
err,p_cbalance,p_ccity,p_ccredit,p_ccreditlim,p_cdata,p_cdiscount,p_cfirst,p_cid,p_clast,p_cmiddle,p_cphone,p_csince,p_cstate,
```

```
p_cstreet1,p_cstreet2,p_czip,p_datetime,p_dcity,p_dname,p_dstate,p_dstreet1,p_dstreet2,p_dzip,p_wcity,p_wname,p_wstate,  
p_wstreet1,p_wstreet2,p_wzip;
```

```
    commit;  
    else  
        set err = -1;  
        rollback;  
    end if;  
end $$  
delimiter ;
```

```
-- order status
```

```
/*sets:allsets*/DROP PROCEDURE IF EXISTS order_status_tpcc;
```

```
delimiter $$
```

```
/*sets:allsets*/ CREATE PROCEDURE order_status_tpcc(  
    in wid int,  
    in d_id_arg int,  
    in c_id_arg int,  
    in bylastname int,  
    in c_last_arg varchar(16))
```

```
label:begin  
    declare itemnum int default 0;  
    declare done int default 0;  
    declare ol_i_id_arg int;  
    declare ol_supply_w_id_arg int;  
    declare ol_quantity_arg tinyint;  
    declare ol_amount_arg decimal(6,2);  
    declare ol_delivery_d_arg datetime;  
    declare namecnt int;  
    declare rowindex int;  
    declare c_id_arg_out int;  
    declare c_balance_arg decimal(12,2);  
    declare c_first_arg varchar(16);  
    declare c_middle_arg char(2);  
    declare c_last_arg_out varchar(16);  
    declare o_id_arg int;  
    declare o_entry_d_arg datetime;  
    declare o_carrier_id_arg tinyint;  
    declare item_data text;  
    declare err int default 0;  
  
    declare exit HANDLER FOR NOT FOUND,SQLException  
    begin  
        set err = -1;  
        select err, c_id_arg_out, c_balance_arg, c_first_arg, c_middle_arg, c_last_arg_out, o_id_arg, o_entry_d_arg,  
o_carrier_id_arg, item_data;  
        rollback;  
    end;  
    START TRANSACTION;  
    set err = 0;  
    -- if bylastname is not 0, get cid by c_last_arg  
    if bylastname != 0 then  
        SELECT count(c_id) INTO namecnt FROM customer WHERE c_w_id = wid AND c_d_id = d_id_arg AND c_last =  
c_last_arg;  
        if namecnt = 0 then #cannot find one record  
            set err = -1;  
            select err, c_id_arg_out, c_balance_arg, c_first_arg, c_middle_arg, c_last_arg_out, o_id_arg, o_entry_d_arg,  
o_carrier_id_arg, item_data;  
            rollback;  
            leave label;  
        end if;  
        set rowindex = floor(namecnt/2); #find which one customer  
        SELECT c_id into c_id_arg FROM customer WHERE c_w_id = wid AND c_d_id = d_id_arg AND c_last =  
c_last_arg order by c_first limit rowindex,1;
```

```

end if;

set c_id_arg_out = c_id_arg;
SELECT c_balance, c_first, c_middle, c_last into c_balance_arg,c_first_arg,c_middle_arg,c_last_arg_out
FROM customer WHERE c_w_id = wid AND c_d_id = d_id_arg AND c_id = c_id_arg;

-- find the most recent order for this customer
SELECT o_id, o_entry_d, COALESCE(o_carrier_id,0) into o_id_arg,o_entry_d_arg,o_carrier_id_arg
FROM orders WHERE o_w_id = wid AND o_d_id = d_id_arg AND o_c_id = c_id_arg order by o_id desc limit 1;

begin
DECLARE cur CURSOR FOR
SELECT ol_i_id, ol_supply_w_id, ol_quantity, ol_amount, ol_delivery_d
FROM order_line WHERE ol_w_id = wid AND ol_d_id = d_id_arg AND ol_o_id = o_id_arg;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
OPEN cur;
while done != 1 do
FETCH cur INTO ol_i_id_arg, ol_supply_w_id_arg, ol_quantity_arg, ol_amount_arg, ol_delivery_d_arg;
if done != 1 then
if isnull(ol_delivery_d_arg) then
set item_data = concat_ws(',', item_data, ol_i_id_arg, ol_supply_w_id_arg, ol_quantity_arg, ol_amount_arg,
'NULL');
else
set item_data = concat_ws(',', item_data, ol_i_id_arg, ol_supply_w_id_arg, ol_quantity_arg, ol_amount_arg,
ol_delivery_d_arg);
end if;
end if;
end while;
CLOSE cur;
end;

select err, c_id_arg_out, c_balance_arg, c_first_arg, c_middle_arg, c_last_arg_out, o_id_arg, o_entry_d_arg,
o_carrier_id_arg, item_data;
commit;
end $$
delimiter ;

-- delivery
/*sets:allsets*/drop procedure if exists delivery_tpcc;
delimiter $$
/*sets:allsets*/ CREATE PROCEDURE delivery_tpcc(in wid int,in o_carrier_id_arg int)
begin
declare var_ol_number int default 1; -- cycles
declare real_no_o_id int;
declare cid int;
declare dtime datetime;
declare total_ol_amount decimal(12,2);
declare delivery_order_ids varchar(128);
declare errinfo varchar(128);

select CURRENT_TIMESTAMP() into dtime;
START TRANSACTION;

myloop:
while var_ol_number <= 10 do
DELETE FROM new_orders WHERE no_w_id = wid AND no_d_id = var_ol_number order by no_o_id asc limit 1
txsql_returning no_o_id into real_no_o_id ;

if found_rows() > 0 then
UPDATE orders SET o_carrier_id = o_carrier_id_arg WHERE o_w_id = wid AND o_d_id = var_ol_number AND
o_id = real_no_o_id txsql_returning o_c_id into cid ;
if ROW_COUNT() != 1 then
set errinfo = concat("update orders can't find,for no_d_id:",var_ol_number);
leave myloop;
end if;

```

```

UPDATE order_line SET ol_delivery_d = dtime WHERE ol_o_id = real_no_o_id AND ol_d_id = var_ol_number
AND ol_w_id = wid;
  if ROW_COUNT() < 1 then
    set errinfo = concat("update order_line can't find,for no_d_id:",var_ol_number);
    leave myloop;
  end if;

SELECT SUM(ol_amount) into total_ol_amount FROM order_line WHERE ol_o_id = real_no_o_id AND ol_d_id
= var_ol_number AND ol_w_id = wid;
  if ROW_COUNT() != 1 then
    set errinfo = concat("sum(total_ol_amount) can't find,for no_d_id:",var_ol_number);
    leave myloop;
  end if;

UPDATE customer SET c_balance = c_balance + total_ol_amount , c_delivery_cnt = c_delivery_cnt + 1 WHERE
c_id = cid AND c_d_id = var_ol_number AND c_w_id = wid;
  if ROW_COUNT() != 1 then
    set errinfo = concat("update customer can't find,for no_d_id:",var_ol_number);
    leave myloop;
  end if;

set delivery_order_ids = concat_ws(',', delivery_order_ids, real_no_o_id);
end if;

set var_ol_number = var_ol_number + 1;

end while myloop;

select delivery_order_ids, errinfo;
if isnull(errinfo) then
  commit;
else
  rollback;
end if;
end $$
delimiter ;

-- stock level
/*sets:allsets*/DROP PROCEDURE IF EXISTS stock_level_tpcc;
delimiter $$
/*sets:allsets*/ CREATE PROCEDURE stock_level_tpcc(in w_id_arg int,
  in d_id_arg int,
  in level_arg int)
begin
  declare low_stock int;

  START TRANSACTION;
  SELECT COUNT(DISTINCT (s_i_id)) into low_stock
  FROM order_line, stock, district
  WHERE d_w_id = ol_w_id AND ol_w_id = s_w_id AND d_w_id = w_id_arg AND d_id = ol_d_id AND d_id =
d_id_arg
  AND ol_i_id = s_i_id AND s_quantity < level_arg AND ol_o_id BETWEEN (d_next_o_id - 20) AND
(d_next_o_id - 1);

  select low_stock;
  commit;
end $$
delimiter ;

```

# Appendix C: Configuration Options

The settings that have been changed from the defaults found in the actual software products are provided as follows:

## C.1 Machine type I OS configuration

```
# Kernel info(/etc/sysctl.conf)
kernel.sysrq = 1
net.ipv6.conf.all.disable_ipv6=0
net.ipv6.conf.default.disable_ipv6=0
net.ipv6.conf.lo.disable_ipv6=0
kernel.printk = 5
fs.file-max=6553500
vm.max_map_count=655360
net.ipv4.ip_local_port_range=32768 61000
kernel.pid_max=98304
kernel.threads-max=8241675
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_window_scaling=1
net.ipv4.tcp_max_syn_backlog=4096
net.core.somaxconn=4096
net.core.netdev_max_backlog=2000
vm.swappiness=0
net.ipv4.tcp_keepalive_time=5
net.ipv4.tcp_keepalive_intvl=2
net.ipv4.tcp_keepalive_probes=5
net.ipv4.tcp_retries2=6
kernel.core_pattern=/data/coredump/core-%e-%p-%t
net.ipv4.conf.all.arp_announce=2
net.ipv4.conf.all.rp_filter=0
net.ipv4.ip_forward=0

# File system info(df -h)
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        139G   0  139G   0% /dev
tmpfs            139G  24K  139G   1% /dev/shm
tmpfs            139G 107M  139G   1% /run
tmpfs            139G   0  139G   0% /sys/fs/cgroup
/dev/vda1        99G   7.8G   87G   9% /
/dev/vdb         493G  30G  438G   7% /data
tmpfs            28G   0   28G   0% /run/user/0

# OS info(/etc/os-release)
NAME="Tencent tlinux"
VERSION="2.6"
ID="tlinux"
ID_LIKE="rhel fedora centos"
VERSION_ID="2.6"
PRETTY_NAME="Tencent tlinux 2.6"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:tlinux:linux:2"
HOME_URL="https://tlinux.qq.com/"

# File opener info (/etc/security/limits.conf)
# BEGIN TDSQL SET
*          -    nofile    1000000
# END TDSQL SET
*          -    nofile    400000
```

## C.2 Machine type II OS configuration

### Kernel info(/etc/sysctl.conf)

```
# System default settings live in /usr/lib/sysctl.d/00-system.conf.
# To override those settings, enter new settings here, or in an /etc/sysctl.d/<name>.conf file
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
kernel.printk = 4
fs.file-max=6553500
vm.max_map_count=655360
net.ipv4.ip_local_port_range=32768 61000
kernel.pid_max=98304
kernel.threads-max=8241675
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_window_scaling=1
net.ipv4.tcp_max_syn_backlog=4096
net.core.somaxconn=4096
net.core.netdev_max_backlog=2000
vm.swappiness=0
net.ipv4.tcp_keepalive_time=5
net.ipv4.tcp_keepalive_intvl=2
net.ipv4.tcp_keepalive_probes=5
net.ipv4.tcp_retries2=6
kernel.core_pattern=/data/coredump/core-%e-%p-%t
net.ipv4.conf.tunl0.arp_ignore=1
net.ipv4.conf.tunl0.arp_announce=2
net.ipv4.conf.all.arp_announce=2
net.ipv4.conf.tunl0.rp_filter=0
net.ipv4.conf.all.rp_filter=0
net.ipv4.ip_forward=0
fs.aio-max-nr = 1048576
```

### File system info(df -h)

Filesystem	Size	Used	Avail	Use%	Mounted on	dev	tmpfs	Size	Used	Avail	Use%	Mounted on	dev
tmpfs	377G	20K	377G	1%	/dev/shm								
tmpfs	377G	284M	377G	1%	/run								
tmpfs	377G	0	377G	0%	/sys/fs/cgroup								
/dev/sda1	20G	6.1G	13G	33%	/								
/dev/sda3	20G	1.8G	17G	10%	/usr/local								
/dev/sda2	511M	7.0M	505M	2%	/boot/efi								
/dev/sda4	401G	27G	353G	8%	/data								
tmpfs	76G	0	76G	0%	/run/user/0								
/dev/md1	42T	25T	18T	59%	/data1	#database data&log							

### OS info(/etc/os-release)

```
NAME="Tencent tlinux"
VERSION="2.2 (Final)"
ID="tlinux"
ID_LIKE="rhel fedora centos"
VERSION_ID="2.2"
PRETTY_NAME="Tencent tlinux 2.2 (Final)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:tlinux:linux:2"
HOME_URL="http://tlinux.oa.com/"
BUG_REPORT_URL="http://tapd.oa.com/tlinux/bugtrace/bugreports/my_view/"
```

### #File opener info (/etc/security/limits.conf)

```
BEGIN TDSQL SET
* - nofile 1000000
# END TDSQL SET
```

### RTE parameters for tpcc-tdsql

```
title = "RTE parameters"
# current warehouse number
warehouse-count = 64003500
```



```

# path of statistical data
data-path = "../data"
# http conn max idle duration
http-idle-duration = "1h"
# warmup time
ramp-up-time = "10m"
# time of stable running
measurement-interval = "8h"
# print interval, less than 0s will not print
print-interval = "10s"
# Emulated Display Delay
emulated-display-delay = "100ms"

```

## C.3 Nginx configuration

### Nginx configuration file nginx.cnf

```

#tcp_nopush    on;

keepalive_timeout 3600;
keepalive_requests 1000000;
#gzip on;

upstream fastcgi_backend {
    server 127.0.0.1:9000;
    keepalive 60;
}

server {
    listen      8080 reuseport backlog=102400;
    server_name location;

    #charset koi8-r;
    #access_log logs/host.access.log main;
gzip on;
# Compress all pages
gzip_min_length 0;
gzip_comp_level 9;
gzip_types text/plain application/javascript application/x-javascript text/javascript text/xml text/css;
gzip_vary on;
# Compress all returns
gzip_proxied any;

    location / {
        include fastcgi.conf;
        fastcgi_pass fastcgi_backend;
        fastcgi_keep_conn on;
        proxy_set_header X-Forwarded-For $remote_addr;
        fastcgi_connect_timeout 600;
        fastcgi_read_timeout 600;
        fastcgi_send_timeout 600;
    }

    #error_page 404          /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root html;
    }
    location = /check.js {
        root html;
    }
}

```

```
}  
}
```

## C.4 Data Node database configuration

### [mysqld]

```
binlog_expire_logs_seconds=0  
binlog_row_image=minimal  
binlog_transaction_dependency_tracking=COMMIT_ORDER  
binlog_write_threshold=0  
character_set_server=latin1  
collation_server=latin1_bin  
forbid_remote_change_sql_log_bin=0  
forbid_remote_drop_meta=off  
forbid_server_path_remote_access=off  
innodb_buffer_pool_in_core_file=off  
innodb_buffer_pool_instances=64  
innodb_buffer_pool_size=289910292480  
innodb_checksum_algorithm=innodb  
innodb_cleaner_lsn_age_factor=high_checkpoint  
innodb_fill_factor=95  
innodb_flush_log_at_trx_commit=1  
innodb_flush_neighbors=0  
innodb_io_capacity=10000  
innodb_io_capacity_max=20000  
innodb_lock_wait_timeout=5  
innodb_log_auto_purge=1  
innodb_log_buffer_size=536870912  
innodb_log_recent_closed_size=536870912  
innodb_log_recent_written_size=536870912  
innodb_lru_scan_depth=2048  
innodb_max_dirty_pages_pct=90  
innodb_page_hash_locks=256  
innodb_page_reserve_factor=256  
innodb_page_size=4096  
innodb_print_ddl_logs=on  
innodb_simplify_trx_in_innodb=on  
innodb_space_extend_fill_zero=0  
innodb_thread_concurrency=0  
innodb_use_cloned_view=1  
innodb_write_io_threads=48  
innodb_log_file_use_mmap=1  
engine_net_timeout=16  
local_infile=on  
lower_case_table_names=1  
max_binlog_size=524288000  
max_connections=100000  
max_prepared_stmt_count=1048576  
performance_schema=off  
query_prealloc_size=204800  
read_only=OFF  
sp_cache_range_info=1  
sqlasyn=1  
table_open_cache=40960  
table_open_cache_instances=32  
thread_handling=0  
thread_pool_max_threads=1000  
thread_pool_max_threads_per_group=10  
transaction_write_set_extraction=off  
txsql_enable_resource_statistics=off
```

# Appendix D: Price References

## D.1 Tencent Cloud Virtual Machine S5.24XLARGE288

已选实例 已选实例 S5.19XLARGE288 (标准型S5, 96核288GB) 您已选择广州三区, 如需更多配额, 可前往 [控制台申请](#)。

常见业务场景选型推荐

实例	规格	vCPU	内存	处理器型号	内网带宽	网络	参考费用
<input type="radio"/> 标准型S5 (8.7折)	S5.8XLARGE128	32核	128GB	Intel Xeon Cascade Lake 8255C...	12Gbps	250	3841.92元/月 4416元/月
<input type="radio"/> 标准型S5 (8.7折)	S5.12XLARGE96	48核	96GB	Intel Xeon Cascade Lake 8255C...	17Gbps	400	3841.92元/月 4416元/月
<input type="radio"/> 标准型S5 (8.7折)	S5.12XLARGE192	48核	192GB	Intel Xeon Cascade Lake 8255C...	17Gbps	400	5762.88元/月 6624元/月
<input type="radio"/> 标准型S5 (8.7折)	S5.16XLARGE256	64核	256GB	Intel Xeon Cascade Lake 8255C...	23Gbps	500	7683.84元/月 8832元/月
<input checked="" type="radio"/> 标准型S5 (8.7折)	S5.24XLARGE288	96核	288GB	Intel Cooper Lake(2.5GHz)	13Gbps	420	9604.8元/月 11040元/月

共 40 项

已选 S5.19XLARGE288 (标准型S5, 96核288GB)

时长 3年 数量 1

配置费用 162950.40元  
387440元

带宽费用 0.00元

下一步: 设置网络和主机

有奖调研

## D.2 Tencent Database Dedicated Cluster HYI12A

机型

虚拟机类型 物理机类型

机型	机型信息	vCpu	内存	磁盘总大小	数量(max)	费用
<input type="checkbox"/> HYIO2A (售罄)	CPU: Intel(R) Xeon(R) Gold 6133 CPU @ 2.50GHz x2 内存: 32GB x24 磁盘: NVMe SSD 4分区, 双15TB数据分区, 双3.7TB日志分区	80核	768GB	37500GB		--
<input checked="" type="checkbox"/> HYI12A	CPU: Intel(R) Xeon(R) Platinum 8255C CPU @ 2.50GHz x2 内存: 32GB x24 磁盘: NVMe SSD 4分区, 双14TB数据分区, 双7TB日志分区	96核	768GB	40000GB	1	31752.00元 / 台 / 月

自动续费  账户余额足够时, 到期后自动按月续费

购买时长 1个月 2个月 3个月 1年 2年 3年 更多

服务条款  我已阅读并同意 [《云数据库服务条款》](#)

配置费用 571536.00元  
1143072.00元

立即购买

## D.3 ThinkPad X1 Carbon 2022 Laptop Price

The screenshot shows the product page for the ThinkPad X1 Carbon 2022. The page layout includes a navigation bar at the top with the ThinkPad logo and various links. The main content area features a large product image of the laptop, a price tag of ¥11999, and a list of configuration options. The page is in Chinese and includes promotional banners and a sidebar with navigation icons.

**ThinkPad X1 Carbon 2022 英特尔酷睿i7 超轻旗舰本 03CD**  
i7-1260P/Windows 11 家庭中文版/16GB LPDDR5/512GB SSD/锐炬Xe显卡/14.0英寸2.2K IPS广视角 LED背光显示屏300nit (A面偏光处理) /FHD+RGB 红外摄像头/WiFi 6/内置SIM卡(12个月\*15G/月流量)

**¥11999** (原价 ¥12499)

限时12期免息  
50元首购礼金

第12代英特尔®酷睿™ i7

操作系统: Windows 11 家庭中文版 / Windows 11 专业版

选择配置:

12代i5/16G LPDDR5/512GB/2.2K屏	12代i7/16G LPDDR5/512GB/2.2K屏
12代i7/16G LPDDR5/1TB/2.8K屏	12代i7/32G LPDDR5/2TB/4K屏
12代i7/16G LPDDR5/1TB/触控屏	三体定制款/12代i7/32G LPDDR5/2T...

## D.4 TengCloud Service Support Price



### 软件服务报价单

该报价单仅针对于对下文所述公司的特定软件服务报价，仅在该报价单约束的服务、时间范围内生效。报价日期为 2023 年 1 月 31 日，该报价在未来 360 天内有效，超过时间请重新询价。

客户名称：腾讯云计算（北京）有限责任公司  
地址：北京市海淀区海淀大街 38 号银科大厦  
电话：010-62671188  
联系人：李书庚  
手机：13003932516  
邮箱：cycloneli@tencent.com

供货单位：贵州云腾未来科技有限公司  
地址：贵州省贵阳市观山湖区长岭南路 160 号高科一号 C 栋 16 楼  
公司电话：13909230888  
报价联系人：罗素群  
报价联系人手机：18228021853  
邮箱：luosq@tcfuture.tech

根据您的要求，我们为以下服务，做出如下报价：

服务项目	Nginx 支持服务
服务范围	Nginx 安装、问题调试、配置、以及后续维保服务
服务时间	7*24H 现场支持(北京、成都、深圳市内支持)
单价	35,000 元/人*月
预购时长	36 月
服务人数	5 人
总价	6,300,000 元，大写：陆佰叁拾万元整

付款说明：本报价仅接受人民币预付款。

税费说明：本报价包含增值税。

报价单位：  
时间：2023 年 1 月 31 日

