FROM INTRODUCTION TO PRACTICE

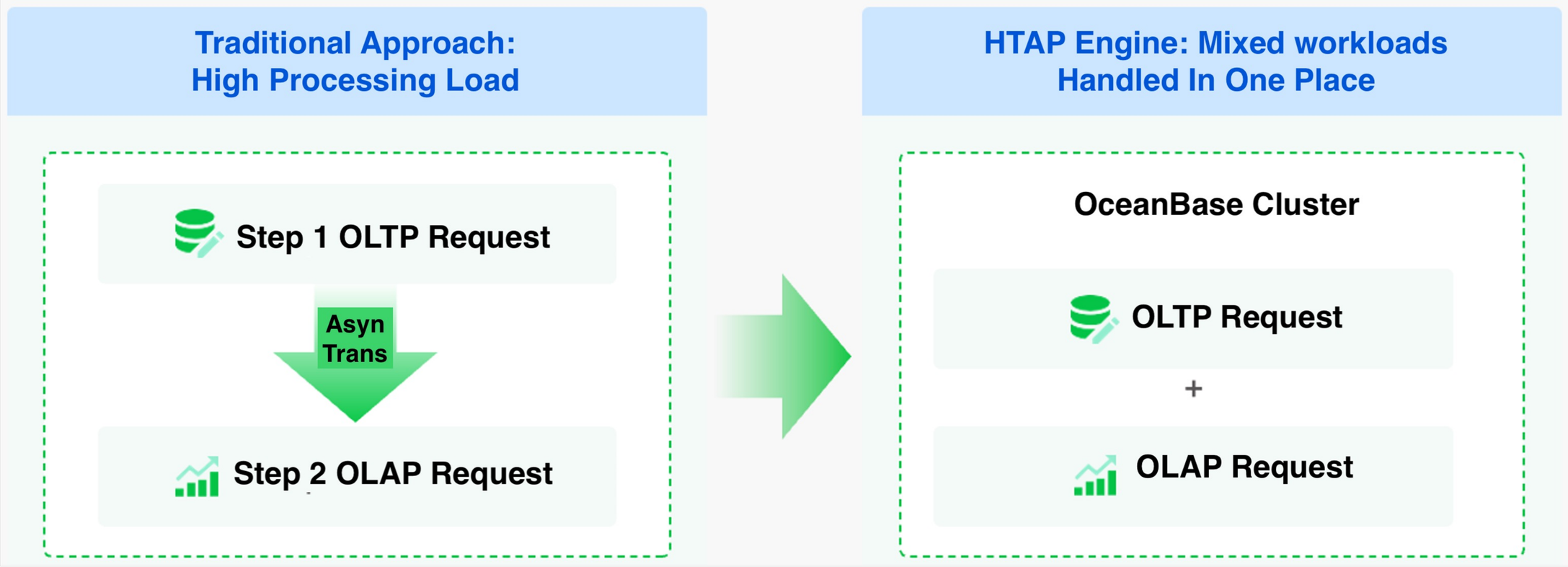
Lesson 3:
OceanBase Performance Test

Peng Wang
OceanBase Global Technical Evangelist

Agenda

- OceanBase Testing Overview
- Factors Affecting OceanBase Performance
- Benchmark Testing and Tuning
- Other Common Test Points

Performance Testing



OceanBase provides high-performance HTAP capabilities through native distributed technology, truly processing transactions and real-time analysis simultaneously through "one system". "One piece of data" is used for different workloads, fundamentally maintaining data consistency and minimizing data redundancy, helping enterprises significantly reduce total costs.

Parallel Import

Import Method	Implementation	Advantage	Applicable Scenarios
Load data	Implemented through batch insert	Stable performance	The amount of data to be imported is smaller or slightly larger than the memory
Bypass import	Write data directly to sstable	Better performance in large data volume scenarios	Importing large amounts of data
Load local data	Send the file content to the server through the local protocol for import	Import files can be placed on the client	The data file is not on the observer side

Through the parallel execution framework of the OceanBase database, multi-node databases can be written concurrently by multiple machines, greatly improving the data import speed. This is critical for users who must handle large amounts of data migration or synchronization tasks.

Data Compression

Data storage volume for the same business
OceanBase is only for MySQL/Oracle database

1/4-1/3

Significantly improve the stability and security of business systems
Effectively reduce storage costs

70%-90%

Data compression is a key means to reduce the storage space occupied by massive data. OceanBase's high-compression distributed storage engine abandons the fixed-length data block storage of traditional databases, adopts LSM-Tree-based storage architecture and adaptive compression technology, creatively solves the problem that traditional databases cannot balance ‘performance’ and ‘compression ratio’, and further reduces storage costs based on distributed storage technology based on data log separation methods, achieving high performance and low storage costs. The storage engine based on LSM-Tree uses encoding compression to greatly reduce storage costs.

Test High Availability

In 2014, OceanBase v0.5 was released.

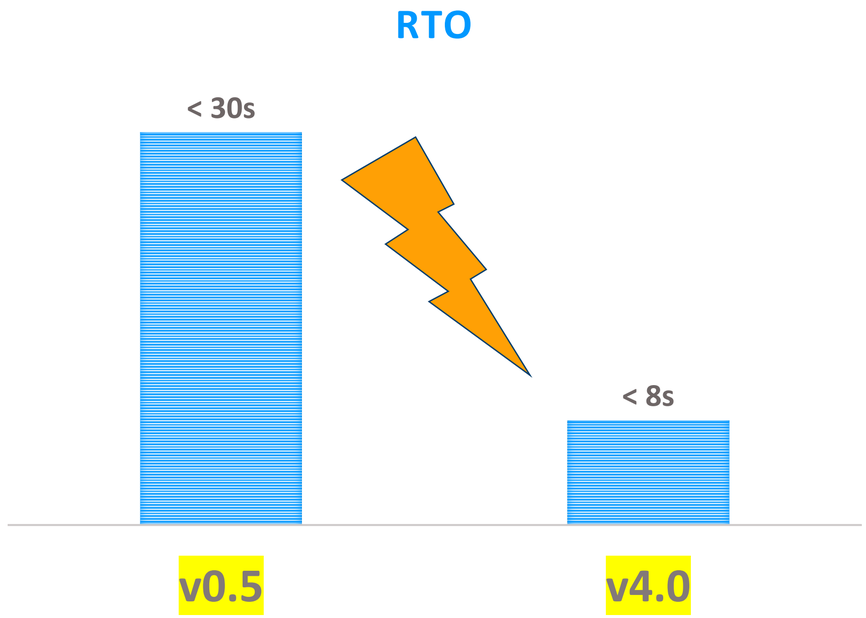
- **RPO=0**
- **RTO < 30s**

Becoming the industry standard

In 2022, OceanBase v4.0 was released

- **RPO=0**
- **RTO < 8s**

The First database in the industry to achieve an RTO of under 8 seconds



- The database system plays an important role in data storage and query in the application architecture. The high availability of enterprise data is crucial to ensuring business continuity.
- High availability is also a key factor in database testing.
- OceanBase database implements a multi-copy disaster recovery solution based on the Paxos protocol, providing users with high availability capabilities with RPO = 0 and RTO < 8s in the event of a minority failure.

Tips: RTO and RPO are important indicators for measuring the performance and requirements of disaster recovery systems.

Agenda

- OceanBase Testing Overview
- Factors Affecting OceanBase Performance
- Benchmark Testing and Tuning
- Other Common Test Points

Factors Affecting OceanBase Performance

How to achieve better performance in common Benchmarks?

- Operating System Parameters
- Resource Allocation
- Database Parameters
- Compaction
- Statistics Collection

Operating System Parameters

Type	Configuration Items	Description	Recommendation
Network Parameters	net.core.somaxconn	The maximum length of the socket listening queue. If you frequently establish connections, you need to increase this value.	The default value is 128, and the recommended value is 2048.
Network Parameters	net.core.netdev_max_backlog	The buffer queue length processed by the protocol stack. If it is set too small, it may cause packet loss.	The recommended setting is 10000
Network Parameters	net.core.rmem_default	The default length of the receive buffer queue	The recommended configuration is 16777216
Network Parameters	net.core.wmem_default	The default length of the send buffer queue	The recommended configuration is 16777216
Network Parameters	net.core.rmem_max	Maximum length of the receive buffer queue	The recommended configuration is 16777216
Network Parameters	net.core.wmem_max	Maximum length of the send buffer queue	The recommended configuration is 16777216
Network Parameters	net.ipv4.ip_local_port_range	The local TCP/UDP port range, the local uses the port in this range to initiate a connection with the remote end	The recommended port range is [3500, 65535]
Network Parameters	net.ipv4.tcp_rmem	The size of the socket receive buffer, which is the minimum value, default value, and maximum value.	The recommended minimum, default, and maximum values are 4096, 87380, and 16777216 respectively.
Network Parameters	net.ipv4.tcp_wmem	The size of the socket send buffer, which is the minimum value, default value, and maximum value.	The recommended minimum, default, and maximum values are 4096, 65536, and 16777216 respectively.
Network Parameters	net.ipv4.tcp_max_syn_backlog	Number of connections in SYN_RECVD state	The recommended configuration is 16384
Network Parameters	net.ipv4.tcp_fin_timeout	Duration of the FIN-WAIT-2 state after the socket is actively disconnected	The recommended setting is 15
Network Parameters	net.ipv4.tcp_tw_reuse	Allow reuse of sockets in TIME WAIT state	The recommended configuration is 1
Network Parameters	net.ipv4.tcp_slow_start_after_idle	Disable slow start of TCP connections from Idle state to reduce network latency in some situations	The recommended setting is 0
Virtual Memory Configuration	vm.swappiness	Prioritize the use of physical memory	The recommended setting is 0
Virtual Memory Configuration	vm.max_map_count	The number of virtual memory regions that a process can have	The recommended setting is 655360
AIO Configuration	fs.aio-max-nr	Number of asynchronous I/O requests	The recommended setting is 1048576

Host standardization check items : https://en.oceanbase.com/docs/common-ocp-10000000001187522

Resource Allocation – Software and Hardware Resources

Disk Partitioning

- System log, Transaction log, Data file partition
- **clog Recycling**: log_disk_utilization_threshold
- **clog Stop Writing**: log_disk_utilization_limit_threshold
- **Log Flow Limiting**: syslog_io_bandwidth_limit
- **Log Recovery**: enable_syslog_recycle max_syslog_file_count

Tenant Resources

- CPU
- Memory
- IOPS (v4.x newly supports IPOS isolation)
- LOG_DISK_SIZE (4.x Added support for tenant log disk isolation)

Primary Zone

- The default is RANDOM, which controls the distribution of partition leaders.
- Scaling tenants by adjusting zone priorities
- v4.x only supports tenant-level Primary Zone and no longer supports table-level, DB-level, and TableGroup-level configurations.

Resource Allocation - Table Resources

Partition Table

- Improving partition availability through partition management
- Reduce shared resource contention in TP scenarios
- Improving query performance in AP scenarios using partition pruning
- Partitions are used as storage units and load-balancing units to make data distribution more even through load-balancing.

Table Group

- By clustering and distributing partitions with the same rules, Partition Wise Join can be implemented, greatly optimizing read and write performance.
- After v4.x, the partitioning mode of a table in a Table Group is no longer restricted by partition attributes.
- SHARDING = NONE, all partitions of all tables are gathered on the same server, and there is no restriction on the partition type of the table
- SHARDING ≠ NONE. Table Group requires that all tables have the same partitioning method to implement 'Partition Wise Join'.

Index

- Local index: The local index of the partition table is only mapped to the main table data in its partition
- Global index: globally unique, the partitioning rules of the index can be inconsistent with the partitioning regulations of the main table
- Although global indexes are unique, they may make each write a distributed transaction across machines, affecting write performance in high-concurrency scenarios.

Parameter Tuning

To improve the user experience and ease of use of the database, so that every developer can get better performance when using the database, OceanBase database has done a lot of performance optimization work after V4.x. This gives users a better database performance experience in scenarios based on basic parameter tuning. Here we only provide tuning suggestions for some basic parameters. On this basis, if you want to improve the performance of the database further, you can also further perform some personalized parameter tuning based on the operating environment and business scenarios.

OLTP Scenario

Type	Description	Command
OBServer Parameters	Disable SQL auditing	ALTER system SET enable_sql_audit=false;
	Disable performance event information collection	ALTER system SET enable_perf_event=false;
	Set the system log level to ERROR to reduce logs	ALTER system SET syslog_level='ERROR';
	Turn off trace logging	ALTER system set enable_record_trace_log=false;
OBProxy Parameters	Increase obproxy runtime memory limit	ALTER proxyconfig SET proxy_mem_limited='4G';
	Disable the compression protocol function of the proxy	ALTER proxyconfig set enable_compression_protocol= false;

OLAP Scenario

Type	Description	Command
OBServer Parameters	Set the SQL workspace memory percentage to the entire tenant memory	SET GLOBAL ob_sql_work_area_percentage = 80;
	Set the maximum SQL execution time (Unit: microseconds)	SET GLOBAL ob_query_timeout = 36000000000;
	Set transaction timeout (Unit: microseconds)	SET GLOBAL ob_trx_timeout = 36000000000;
	Set the maximum network packet size (Unit: Byte)	SET GLOBAL max_allowed_packet = 67108864;
	The number of parallel execution threads that a tenant can apply for on each node	SET GLOBAL parallel_servers_target = 624;

Compaction and Statistics Collection

Compaction

- The compaction will merge the SSTable and MemTable of the current major version with the full static data of the previous major version to generate new full data
- Compression is performed during the compaction, saving storage space and greatly improving query performance.
- Based on the LSM-Tree architecture storage system, compression has almost no impact on data writing performance

Statistics Collection

- Statistics refers to optimizer statistics, which is a data set that describes the table and column information in the database and helps the optimizer generate the optimal execution plan
- Statistics are stored in internal tables as normal data, and a cache of statistics is maintained locally to improve the optimizer's access speed to statistics
- After V4.x, statistics collection and daily compaction are decoupled. The daily compaction process no longer collects statistics, and the execution plan is no longer affected by the daily compaction

Agenda

- OceanBase Testing Overview
- Factors Affecting OceanBase Performance
- Benchmark Testing and Tuning
- Other Common Test Points

Sysbench

Sysbenh Features

- A scriptable multi-threaded benchmark tool based on LuaJIT, which can implement database testing in different scenarios through custom Lua scripts
- Features: Single table structure, simple fields, single table operation
- The test scenario is simple, without any business scenario, and only tests the basic performance of SQL.
- Performance indicators: TPS, QPS, RT

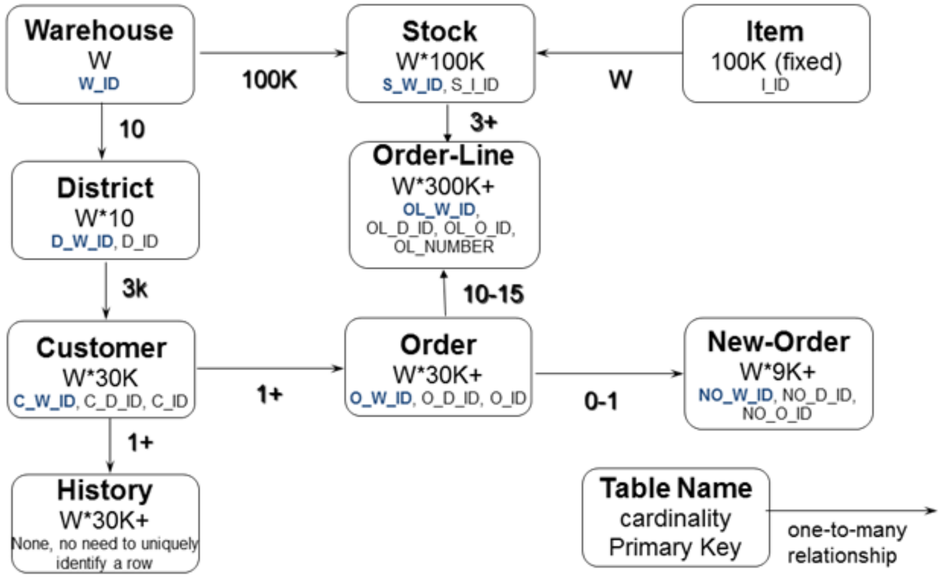
```
local stmt_defs = {
   point_selects = {
      "SELECT c FROM sbtest%u WHERE id=?",
      t.INT},
   simple_ranges = {
      "SELECT c FROM sbtest%u WHERE id BETWEEN ? AND ?",
      t.INT, t.INT},
   sum_ranges = {
      "SELECT SUM(k) FROM sbtest%u WHERE id BETWEEN ? AND ?",
      t.INT, t.INT},
   order_ranges = {
      "SELECT c FROM sbtest%u WHERE id BETWEEN ? AND ? ORDER BY c",
      t.INT, t.INT},
   distinct_ranges = {
      "SELECT DISTINCT c FROM sbtest%u WHERE id BETWEEN ? AND ? ORDER BY c",
      t.INT, t.INT},
   index_updates = {
      "UPDATE sbtest%u SET k=k+1 WHERE id=?",
      t.INT},
   non_index_updates = {
      "UPDATE sbtest%u SET c=? WHERE id=?",
      {t.CHAR, 120}, t.INT},
   deletes = {
      "DELETE FROM sbtest%u WHERE id=?",
      t.INT},
   inserts = {
      "INSERT INTO sbtest%u (id, k, c, pad) VALUES (?, ?, ?, ?)",
      t.INT, t.INT, {t.CHAR, 120}, {t.CHAR, 60}},
}
```

TPC-C

TPC-C Features

- OLTP Benchmark, Commodity Sales Model
- 9 tables, adjust the data size by the number of warehouses W
- 5 types of transactions (new-order, payment, order-status, delivery, stock-level)
- Performance indicator: tpmC (the number of new-order transactions per minute)

TPC-C Schema



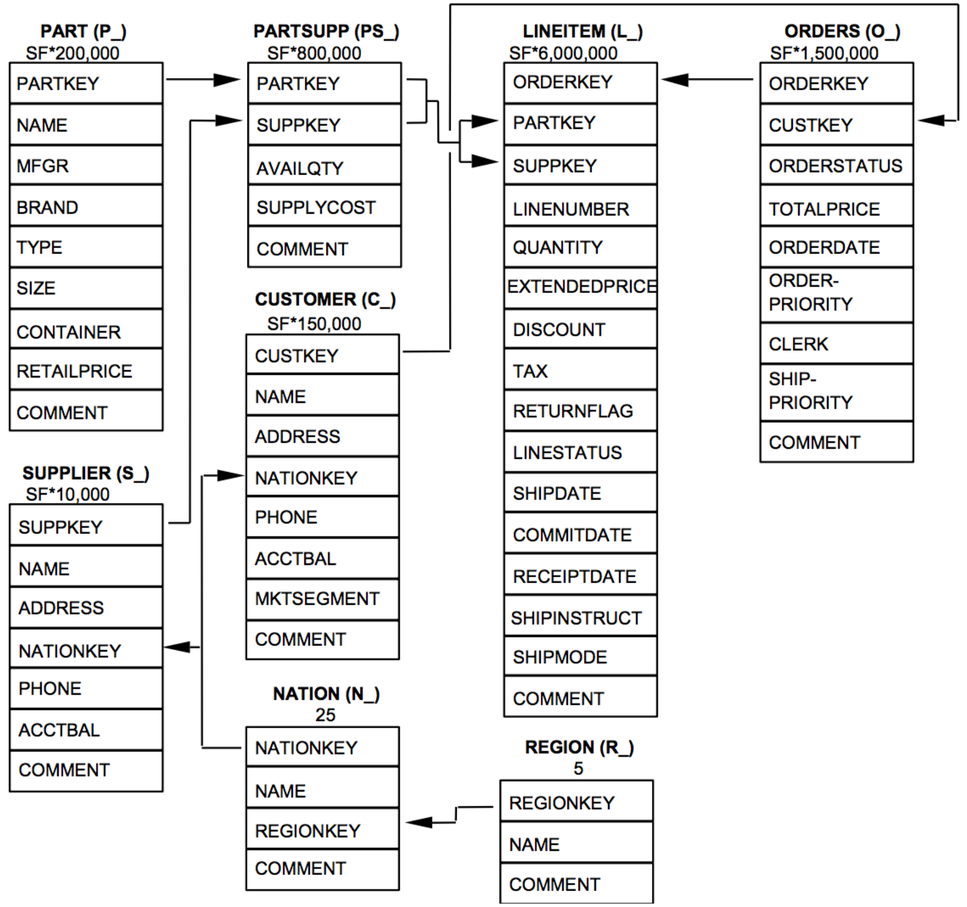
Transactions	Transaction Complexity Level	Read and Write Frequency	Response Speed Requirements	Proportion
New-order	Medium-sized	Frequent reading and writing	Requiring fast response	45%
Payment	Lightweight	Frequent reading and writing	Fast response required	43%
OrderStatus	Medium-sized	Low read-only frequency	Fast response required	4%
Delivery	Heavyweight	Low read and write frequency	More relaxed response time	4%
Stock-level	Heavyweight	Low read-only frequency	More relaxed response time	4%

TPC-H

TPC-H Features

- OLAP benchmarking, simulating transaction behavior between suppliers and buyers
- 8 tables, based on 3NF, of which the data volume of the nation and region tables is fixed, and the data volume of the remaining 6 tables is related to the scale factor SF (Scale Factor)
- 22 complex analytical queries. Including statistical queries (SUM, AVG), multi-table association queries, subqueries, complex condition queries, GROUP BY, ORDER BY, etc.
- Performance Indicators: RT

TPC-H Schema



Legend:

- The parentheses following each table name contain the prefix of the column names for that table;
- The arrows point in the direction of the one-to-many relationships between tables;
- The number/formula below each table name represents the cardinality (number of rows) of the table. Some are factored by SF, the Scale Factor, to obtain the chosen database size. The cardinality for the LINEITEM table is approximate (see Clause 4.2.5).

OBD One-button Test

Sysbench

obd test sysbench <deploy_name> --tenant=<tenant_name> --script-name=oltp_read_write.lua --table-size=1000000 --tables=30 --threads=32 --rand-type=uniform

TPC-C

obd test tpcc <deploy_name> --tenant= =<tenant_name> --warehouses=1000 --load-workers=40 --terminals=800 --run-mins=5

TPC-H

obd test tpch <deploy_name> --tenant =<tenant_name> --user=root --remote-tbl-dir=/data/tpch100 -s 100

Tips

- -v: Print out detailed test steps
- -O: Setting the performance tuning level
- --dt: In the TPC-H test, skip the data transmission step
- --test-only: In the TPC-H test, only running the test SQL

OBD TPCH Test Example

Test Command:

obd test tpch perf --tenant=perf --user=root --remote-tbl-dir=/data/tpch -s 1

```
[root@[redacted]~]# obd test tpch perf --tenant=perf --user=root --remote-tbl-dir=/data/tpch -s 1
Get local repositories and plugins ok
Open ssh connection ok
Cluster status check ok
Connect to observer [redacted] ok
Generate Data (Scale Factor: 1) ok
Send tbl to remote ([redacted]) ok
Optimize for stage test ok
Format DDL ok
Create table ok
Load data ok
Merge ok
Format SQL ok
Warmup ok
[2024-03-27 19:24:07]: start /root/tmp/db1.sql
[2024-03-27 19:24:07]: end /root/tmp/db1.sql, cost 0.10s
[2024-03-27 19:24:07]: start /root/tmp/db2.sql
[2024-03-27 19:24:07]: end /root/tmp/db2.sql, cost 0.07s
[2024-03-27 19:24:07]: start /root/tmp/db3.sql
[2024-03-27 19:24:08]: end /root/tmp/db3.sql, cost 0.05s
[2024-03-27 19:24:08]: start /root/tmp/db4.sql
[2024-03-27 19:24:08]: end /root/tmp/db4.sql, cost 0.04s
[2024-03-27 19:24:08]: start /root/tmp/db5.sql
[2024-03-27 19:24:08]: end /root/tmp/db5.sql, cost 0.08s
[2024-03-27 19:24:08]: start /root/tmp/db6.sql
[2024-03-27 19:24:08]: end /root/tmp/db6.sql, cost 0.02s
[2024-03-27 19:24:08]: start /root/tmp/db7.sql
[2024-03-27 19:24:08]: end /root/tmp/db7.sql, cost 0.10s
[2024-03-27 19:24:08]: start /root/tmp/db8.sql
[2024-03-27 19:24:08]: end /root/tmp/db8.sql, cost 0.10s
[2024-03-27 19:24:08]: start /root/tmp/db9.sql
[2024-03-27 19:24:08]: end /root/tmp/db9.sql, cost 0.21s
[2024-03-27 19:24:08]: start /root/tmp/db10.sql
[2024-03-27 19:24:08]: end /root/tmp/db10.sql, cost 0.07s
[2024-03-27 19:24:08]: start /root/tmp/db11.sql
[2024-03-27 19:24:08]: end /root/tmp/db11.sql, cost 0.05s
[2024-03-27 19:24:08]: start /root/tmp/db12.sql
[2024-03-27 19:24:08]: end /root/tmp/db12.sql, cost 0.05s
[2024-03-27 19:24:08]: start /root/tmp/db13.sql
[2024-03-27 19:24:08]: end /root/tmp/db13.sql, cost 0.10s
[2024-03-27 19:24:08]: start /root/tmp/db14.sql
[2024-03-27 19:24:08]: end /root/tmp/db14.sql, cost 0.03s
[2024-03-27 19:24:08]: start /root/tmp/db15.sql
[2024-03-27 19:24:08]: end /root/tmp/db15.sql, cost 0.10s
[2024-03-27 19:24:08]: start /root/tmp/db16.sql
[2024-03-27 19:24:09]: end /root/tmp/db16.sql, cost 0.09s
[2024-03-27 19:24:09]: start /root/tmp/db17.sql
[2024-03-27 19:24:09]: end /root/tmp/db17.sql, cost 0.04s
[2024-03-27 19:24:09]: start /root/tmp/db18.sql
[2024-03-27 19:24:09]: end /root/tmp/db18.sql, cost 0.05s
[2024-03-27 19:24:09]: start /root/tmp/db19.sql
[2024-03-27 19:24:09]: end /root/tmp/db19.sql, cost 0.05s
[2024-03-27 19:24:09]: start /root/tmp/db20.sql
[2024-03-27 19:24:09]: end /root/tmp/db20.sql, cost 0.08s
[2024-03-27 19:24:09]: start /root/tmp/db21.sql
[2024-03-27 19:24:09]: end /root/tmp/db21.sql, cost 0.13s
[2024-03-27 19:24:09]: start /root/tmp/db22.sql
[2024-03-27 19:24:09]: end /root/tmp/db22.sql, cost 0.07s
Total Cost: 1.68s
Recover ok
```

OBD TPCH Test Example

Test Command: obd test tpch perf --tenant=perf --user=root --remote-tbl-dir=/data/tpch -s 1 -v

```
-- mkdir /root/tmp/s1
Generate Data (Scale Factor: 1) ok
-- local execute: cd /root/tmp/s1; /usr/tpc-h-tools/tpc-h-tools/bin/dbgen -s 1 -b /usr/tpc-h-tools/tpc-h-tools/dists.dss -- exited code 0
```

```
-- execute sql: select value from oceanbase.__all_virtual_sys_parameter_stat where name="enable_sql_audit". args: None
-- origin_value True(str) target_value False(bool) condition lambda n, o: n != o
-- execute sql: alter system set enable_sql_audit=%s. args: (False,)
-- execute sql: select value from oceanbase.__all_virtual_sys_parameter_stat where name="syslog_level". args: None
-- origin_value WDIAG(str) target_value PERF(str) condition lambda n, o: n != o
-- execute sql: alter system set syslog_level=%s. args: ('PERF',)
-- execute sql: select value from oceanbase.__all_virtual_sys_parameter_stat where name="enable_perf_event". args: None
-- origin_value True(str) target_value False(bool) condition lambda n, o: n != o
-- execute sql: alter system set enable_perf_event=%s. args: (False,)
-- execute sql: select value from oceanbase.__all_virtual_sys_parameter_stat where name="enable_record_trace_log". args: None
-- origin_value False(str) target_value false(str) condition lambda n, o: n != o
-- execute sql: alter system set enable_record_trace_log=%s. args: ('false',)
-- optimize SystemConfigEntrance success
```

Basic ideas for troubleshooting Benchmark performance issues - OLTP

Step 1: Use the Obdiag tool to perform a physical examination on OceanBase to check the overall health status of the cluster

OLTP Scenario:

1. When troubleshooting, you can set the primary to a single zone, run the oltp_point_select case in Sysbench, and observe whether the Observer's CPU can be fully utilized to simplify the basic problem scenario.

2. Check the hardware resource bottlenecks on the client -> proxy -> Observer link, including the CPU, memory, network bandwidth, disk, network latency, etc. of each link component

3. Check whether the test steps are consistent with the official website's standard steps. For example, when running oltp_read_write and oltp_write_only scenarios, if the user does not set -rand-type=uniform, or the range of table-size and tables is relatively small, the values of id or table name in the performance test will be too concentrated, which may easily trigger a deadlock.

4. Use “top -H -p pid” to observe the CPU usage. In a small-scale scenario, you can pay attention to the overhead of MINI_MERGE and MINOR_EXE (Minor Compaction Thread).

Basic ideas for troubleshooting Benchmark performance issues - OLAP

Step 1: Use the Obdiag tool to perform a physical examination on OceanBase to check the overall health status of the cluster.

OLAP scenario:

1. Check tenant resource configuration. Evaluate whether the tenant CPU and memory configuration is sufficient based on the amount of data related to the test

2. Observe the CPU water level and disk io load of the machine under parallel execution

3. Check whether the test steps are consistent with the official website standard steps, and the key OBServer parameters configuration in OLAP scenario. For example, ob_sql_work_area_percentage, parallel_servers_target, etc

4. By querying the view CDB_OB_MAJOR_COMPACTION, whether the STATUS field is IDLE, you can determine whether the merge task is completed normally

5. Check the DBA_OB_TASK_OPT_STAT_GATHER_HISTORY view to see if the STATUS field is SUCCESS to determine whether statistics collection is proceeding normally

6. Check whether the execution plan is normal, focus on the “dop” value in the execution plan, and determine whether the SQL query is executed in parallel normally.

Agenda

- OceanBase Testing Overview
- Factors Affecting OceanBase Performance
- Benchmark Testing and Tuning
- Other Common Test Points

Parallel Import

Based on the “customer” table in the TPC-H test, create an empty table “customer2” with the same table structure, and perform insert operations with PDML turned off and on to observe the effects:

Steps:

1. SET ob_query_timeout = 21600000000; (*Unit: microseconds*) Adjust the maximum SQL execution time to prevent transactions from timing out due to excessive size

2. Without parallelization, observing the execution results, we can see that without parallelization, it takes OceanBase about 110 seconds to insert 15 million rows of data in a single transaction.

```
obclient [test]> INSERT INTO customer2 SELECT * FROM customer;
Query OK, 15000000 rows affected (1 min 50.043 sec)
Records: 15000000  Duplicates: 0  Warnings: 0
```

3. After clearing the customer2, start PDML and execute the insert again. Generally speaking, if there are no multiple SQL statements executed concurrently, the parallelism of a single SQL statement can be set to the number of tenant CPUs.

```
obclient [test]> truncate table customer2;
Query OK, 0 rows affected (0.192 sec)

obclient [test]> INSERT /*+ parallel(22) enable_parallel_dml */ INTO customer2 SELECT * FROM customer;
Query OK, 15000000 rows affected (17.319 sec)
Records: 15000000  Duplicates: 0  Warnings: 0
```

It can be seen that after PDML is enabled, the time taken by OceanBase to insert 15 million rows of data into the same table is reduced to about 17.3 seconds. The performance improvement brought by the PDML feature is about 6.3 times.
This feature can help users in scenarios where batch data processing is required.

Data Compression

FROM INTRODUCTION TO PRACTICE

Based on the 'lineitem' table in the TPC-H test, the total data size of the 'lineitem' table before data import is 75G:

Steps:

1. After the import is complete, execute ALTER SYSTEM major freeze tenant=<your tenant name> to manually trigger the merge
2. Query the oceanbase.CDB_OB_MAJOR_COMPACTION table. When the STATUS in the table is IDLE, it means the merge is complete

```
obclient [oceanbase]> select * from oceanbase.CDB_OB_MAJOR_COMPACTION;
+-----------+--------------------+----------------------------+---------------------+--------------------+----------------------------+----------------------------+--------+----------+--------------+------+
| TENANT_ID | FROZEN_SCN         | FROZEN_TIME                | GLOBAL_BROADCAST_SCN | LAST_SCN           | LAST_FINISH_TIME           | START_TIME                 | STATUS | IS_ERROR | IS_SUSPENDED | INFO |
+-----------+--------------------+----------------------------+---------------------+--------------------+----------------------------+----------------------------+--------+----------+--------------+------+
|         1 | 1710698400187072313 | 2024-03-18 02:00:00.187072 |  1710698400187072313 | 1710698400187072313 | 2024-03-18 02:04:11.906878 | 2024-03-18 02:00:00.208338 | IDLE   | NO       | NO           |      |
|      1001 | 1710698403464687512 | 2024-03-18 02:00:03.464688 |  1710698403464687512 | 1710698403464687512 | 2024-03-18 02:03:25.045465 | 2024-03-18 02:00:03.496329 | IDLE   | NO       | NO           |      |
|      1002 | 1710698404806897471 | 2024-03-18 02:00:04.806897 |  1710698404806897471 | 1710698404806897471 | 2024-03-18 02:03:36.510177 | 2024-03-18 02:00:04.875337 | IDLE   | NO       | NO           |      |
+-----------+--------------------+----------------------------+---------------------+--------------------+----------------------------+----------------------------+--------+----------+--------------+------+
3 rows in set (0.007 sec)
```

3. Use the following query statement in the sys tenant to view the storage usage of data after importing to OceanBase. For example, we query the space usage of the leader replica for the 'lineitem' table:

```
obclient [oceanbase]> select t1.table_name,
    -> round(sum(t2.data_size)/1024/1024/1024,2) as data_size_gb,
    -> round(sum(t2.required_size)/1024/1024/1024,2) as required_size_gb
    -> from dba_ob_tenants t,cdb_ob_table_locations t1,cdb_ob_tablet_replicas t2
    -> where t.tenant_id=t1.tenant_id
    -> and t1.svr_ip=t2.svr_ip
    -> and t1.tenant_id=t2.tenant_id
    -> and t1.ls_id=t2.ls_id
    -> and t1.tablet_id=t2.tablet_id
    -> and t1.role='leader'
    -> and t.tenant_name='perf'
    -> and t1.database_name='test'
    -> and t1.table_name='lineitem'
    -> group by t1.table_name
    -> order by 3 desc;
+------------+--------------+------------------+
| table_name | data_size_gb | required_size_gb |
+------------+--------------+------------------+
| lineitem   |        22.15 |            23.11 |
+------------+--------------+------------------+
```

It can be seen that the total data size of the lineitem table before importing is 75G, the compressed data size is 22.15G, the actual disk space required is 23.11G, and the compression ratio is about 75/23.11=3.2

High Availability Testing

We can simulate continuous business requests through the oltp_read_write.lua test of Sysbench. During the stress test, we manually fail a leader by 'kill -9' and observe the continuous drop time of QPS or TPS:

Steps:

1. Deploy a 3-node cluster, set primary_zone='z1', and connect Sysbench directly to z2 or z3 Observer

Stress test command example: sysbench oltp_read_write.lua --mysql-host=xxx --mysql-port=xxx --mysql-user=root@mysql --mysql-db=sysbench --mysql-password=test --threads=400 --report-interval=1 --tables=10 --table_size=500000 --mysql-ignore-errors=1062,2013,4265,5066,6002,6213,6224,6222,4746,4012,4009,4250,4009,4038 --time=600 --db-ps-mode=disable run

2. 'kill -9 z1_observer' to observe the service recovery time

```
[ 13s ] thds: 400 tps: 190.00 qps: 3431.03 (r/w/o: 2700.03/380.00/351.00) lat (ms,99%): 1938.16 err/s: 0.00 reconn/s: 0.00
[ 14s ] thds: 400 tps: 452.00 qps: 8034.99 (r/w/o: 6234.99/885.00/915.00) lat (ms,99%): 2045.74 err/s: 0.00 reconn/s: 0.00
[ 15s ] thds: 400 tps: 464.00 qps: 8466.98 (r/w/o: 6591.98/945.00/930.00) lat (ms,99%): 1304.21 err/s: 0.00 reconn/s: 0.00
[ 16s ] thds: 400 tps: 404.00 qps: 7723.98 (r/w/o: 6036.98/865.00/822.00) lat (ms,99%): 1280.93 err/s: 0.00 reconn/s: 0.00
[ 17s ] thds: 400 tps: 515.99 qps: 8798.78 (r/w/o: 6818.83/971.98/1007.97) lat (ms,99%): 1479.41 err/s: 0.00 reconn/s: 0.00
[ 18s ] thds: 400 tps: 546.01 qps: 9694.12 (r/w/o: 7528.10/1071.01/1095.01) lat (ms,99%): 1149.76 err/s: 0.00 reconn/s: 0.00
[ 19s ] thds: 400 tps: 555.99 qps: 9973.84 (r/w/o: 7746.88/1109.98/1116.98) lat (ms,99%): 861.95 err/s: 0.00 reconn/s: 0.00
[ 20s ] thds: 400 tps: 551.02 qps: 10089.34 (r/w/o: 7873.26/1121.04/1095.04) lat (ms,99%): 846.57 err/s: 0.00 reconn/s: 0.00
[ 21s ] thds: 400 tps: 267.00 qps: 4857.99 (r/w/o: 3763.99/534.00/560.00) lat (ms,99%): 831.46 err/s: 0.00 reconn/s: 0.00
[ 22s ] thds: 400 tps: 0.00 qps: 0.00 (r/w/o: 0.00/0.00/0.00) lat (ms,99%): 0.00 err/s: 0.00 reconn/s: 0.00
[ 23s ] thds: 400 tps: 0.00 qps: 0.00 (r/w/o: 0.00/0.00/0.00) lat (ms,99%): 0.00 err/s: 0.00 reconn/s: 0.00
[ 24s ] thds: 400 tps: 0.00 qps: 0.00 (r/w/o: 0.00/0.00/0.00) lat (ms,99%): 0.00 err/s: 0.00 reconn/s: 0.00
[ 25s ] thds: 400 tps: 0.00 qps: 0.00 (r/w/o: 0.00/0.00/0.00) lat (ms,99%): 0.00 err/s: 0.00 reconn/s: 0.00
[ 26s ] thds: 400 tps: 0.00 qps: 211.00 (r/w/o: 90.00/0.00/121.00) lat (ms,99%): 0.00 err/s: 0.00 reconn/s: 178.00
[ 27s ] thds: 400 tps: 3.00 qps: 1765.86 (r/w/o: 1531.88/10.00/223.98) lat (ms,99%): 6594.16 err/s: 0.00 reconn/s: 174.99
[ 28s ] thds: 400 tps: 449.03 qps: 10079.67 (r/w/o: 8246.55/935.06/898.06) lat (ms,99%): 7895.16 err/s: 0.00 reconn/s: 2.00
[ 29s ] thds: 400 tps: 612.00 qps: 10188.00 (r/w/o: 7714.00/1268.00/1206.00) lat (ms,99%): 816.63 err/s: 0.00 reconn/s: 0.00
[ 30s ] thds: 400 tps: 550.03 qps: 10166.58 (r/w/o: 7971.46/1084.06/1111.06) lat (ms,99%): 831.46 err/s: 0.00 reconn/s: 0.00
[ 31s ] thds: 400 tps: 571.96 qps: 10246.31 (r/w/o: 7956.46/1139.92/1149.92) lat (ms,99%): 831.46 err/s: 0.00 reconn/s: 0.00
```

It can be seen that the stable TPS value is about 500, and it dropped from 21s to 27s. The process took about 7s, which means that RTO < 8s restored the service.

Key Points

- High-Performance HTAP Capabilities
- Parallel Import and Data Processing
- Data Compression
- High Availability Testing

OCEANBASE

FROM INTRODUCTION TO PRACTICE

Thank You!

OceanBase Official website:
https://oceanbase.github.io/

GitHub Discussions:
https://github.com/oceanbase/oceanbase/discussions