

openGauss
2.1.0

Technical White Paper

Issue	01
Date	2021-09-30



Copyright © Huawei Technologies Co., Ltd. 2021. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions



HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: <https://www.huawei.com>

Email: support@huawei.com

Contents

1 Product Positioning..... 1

2 Application Scenario..... 2

3 Technical Characteristics.....3

4 Software Architecture..... 5

5 Deployment Solutions..... 7

6 Typical Networking.....12

7 Hardware and Software Configuration Requirements.....16

8 Core Database Technologies..... 19

8.1 Basic Functions Oriented to Application Development..... 19

8.2 High Performance.....21

8.3 HA..... 28

8.4 Maintainability.....30

8.5 Database Security..... 33

8.6 AI Capabilities..... 39

9 Technical Specifications..... 41

10 Glossary..... 43

1 Product Positioning

openGauss is a HA relational database that supports the SQL2003 standard and primary/standby deployment.

- Multiple storage modes support composite service scenarios. The in-place update storage engine is introduced.
- The NUMA data structure supports high performance.
- HA is supported by PAXO consistency log replication protocol, primary/standby mode, and cyclic redundancy check (CRC).
- Security features are supported, such as fully-encrypted computing and ledger database, to provide comprehensive end-to-end data security protection.
- The Table Access Method API layer supports multiple storage engines.

openGauss is a multi-core-oriented open-source relational database that provides ultimate performance, full-link service and data security, AI-based tuning, and efficient O&M capabilities. This leading database at enterprise level is developed in collaboration with global partners and is released under the Mulan Permissive Software License v2. openGauss deeply integrates Huawei's years of R&D experience in the database field and continuously builds competitive features based on enterprise-level scenario requirements.

2 Application Scenario

- Transaction applications

Applications need to process highly concurrent online transactions containing a large volume of data, such as e-commerce, finance, O2O, telecom customer relationship management (CRM), and billing.

- IoT data

In IoT scenarios, such as industrial monitoring, remote control, smart cities, smart homes, and IoV, challenges come from a large number of sensors and monitoring devices, high sampling frequency, additional storage modes, and concurrent operation and analysis.

3 Technical Characteristics

Compared with other open-source databases, openGauss has the following characteristics:

- High performance
 - Provides the multi-core architecture-oriented concurrency control technology and Kunpeng hardware optimization, and achieves that the TPC-C benchmark performance reaches 1,800,000 tpmC in Kunpeng 2-socket servers.
 - Uses NUMA-Aware data structures as the key kernel structures to adapt to the trend of using multi-core NUMA architecture on hardware.
 - Provides the SQL bypass intelligent fast engine technology.
 - The Ustore storage engine is provided for frequent update scenarios.
- High availability (HA)
 - Supports multiple deployment modes, such as primary/standby synchronization, primary/standby asynchronization, and cascaded standby server deployment.
 - Supports data page cyclic redundancy check (CRC), and automatically restores damaged data pages through the standby node.
 - Recovers the standby node in parallel and promotes it to primary to provide services within 10 seconds.
 - Log replication and primary selection framework are provided based on the Paxos distributed consistency protocol.
- High security

Supports security features such as fully-encrypted computing, access control, encryption authentication, database audit, and dynamic data masking to provide comprehensive end-to-end data security protection.
- Easy O&M
 - Provides AI-based intelligent parameter tuning and index recommendation to automatically recommend AI parameters.
 - Provides slow SQL diagnosis and multi-dimensional self-monitoring views to help you understand system performance in real time.
 - Provides SQL time forecasting that supports online auto-learning.
- Fully open

- Adopts the Mulan Permissive Software License, allowing code to be freely modified, used, and referenced.
- Fully opens database kernel capabilities.
- Provides excessive partner certifications, training systems, and university courses.

4 Software Architecture

openGauss is a standalone database where data is stored on a single physical node and data access tasks are pushed to service nodes. In this way, high concurrency of servers enables quick data processing. In addition, data can be copied to the standby server through log replication, ensuring high reliability and scalability.

openGauss can be deployed in primary/standby mode. #EN-US_CONCEPT_0000001208072581/en-us_concept_0283139007_en-us_topic_0237080634_en-us_topic_0231764167_fig5205420191411 shows the openGauss logical architecture.

Figure 4-1 openGauss logical architecture

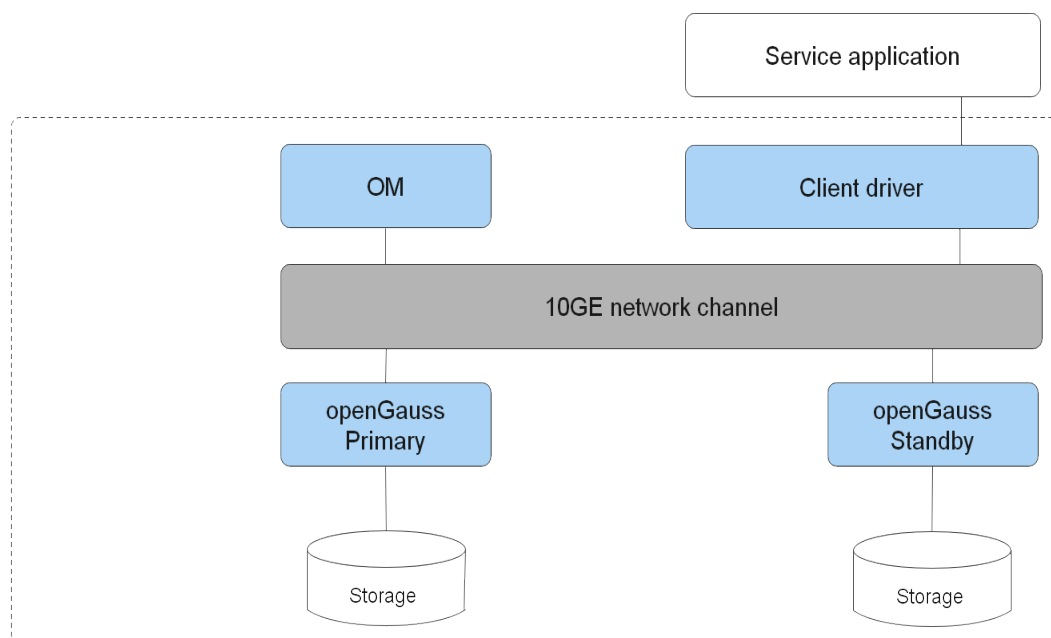


Table 4-1 Architecture description

Name	Description
OM	Operation Manager (OM) provides management interfaces and tools for routine maintenance and configuration management of the database.
Client driver	Client driver receives access requests from the application layer and returns execution results. It communicates with openGauss instances, sends application SQL commands, and receives openGauss execution results.
openGauss (primary/standby)	openGauss primary/standby DN stores service data, executes data query tasks, and returns execution results. openGauss supports one primary and multiple standbys. You are advised to deploy them on different physical nodes.
Storage	Functions as the server's local storage resources to store data permanently.

5 Deployment Solutions

openGauss can be deployed in standalone mode or with one primary node and multiple standby nodes.

Common Concepts

- Standalone
There is only one database instance.
- Primary/Standby
There are primary and standby database instances in the system. The primary instance supports read and write, and the standby instance supports read-only.
- One primary and multiple standbys
The system has one primary node and multiple standby nodes. Up to 8 standby nodes are supported.
- Hot/Cold backup
Cold backup: There is only a simple backup set that cannot provide services.
Hot backup: Backup databases can provide services for external systems.

Deployment Modes

For details about the standalone and primary/standby deployment modes, see [Table 5-1](#).

Table 5-1 Deployment modes in openGauss

Deployment Mode	Technical Solution	High Availability (HA)	Basic Configuration Requirement	Service Scenario	Characteristics	Specifications
Standalone	Standalone	HA is not supported.	Single equipment room	Physical machine	<ul style="list-style-type: none">No system reliability and availability requirements.Applicable to trial use and commissioning scenarios.	<ul style="list-style-type: none">RTO and RPO are uncontrollable.Instance-level DR is not supported. The system is unavailable when instance faults occur.Lost instance data cannot be restored.
Primary/Standby	Primary and standby nodes	Instance faults can be withstood.	Single equipment room	Physical machine	<ul style="list-style-type: none">No network delay between nodes.Instance faults in the database can be withstood.Applicable to scenarios without high reliability requirements.	<ul style="list-style-type: none">RPO = 0Instance fault RTO < 10sAZ-level DR is not supported.The primary and standby nodes in maximum availability mode are recommended.
One primary and multiple standby	One primary and multiple standby (Quorum/Paxos)	Instance faults can be withstood.	Single equipment room	Physical machine	<ul style="list-style-type: none">No network delay between nodes.Instance faults in the database can be withstood.	<ul style="list-style-type: none">RPO = 0Instance fault RTO < 10sAZ-level DR is not supported.The primary/standby synchronization mode is recommended.2 to 4 copies.

Hardware and Software Specifications

openGauss supports the following CPUs and OSs:

Table 5-2 openGauss software and hardware specifications

Delivery Mode	CPU	OS
Open source and offline	x86	CentOS 7.6 or openEuler 20.3 LTS
	Kunpeng	openEuler 20.03 LTS or Kirin V10

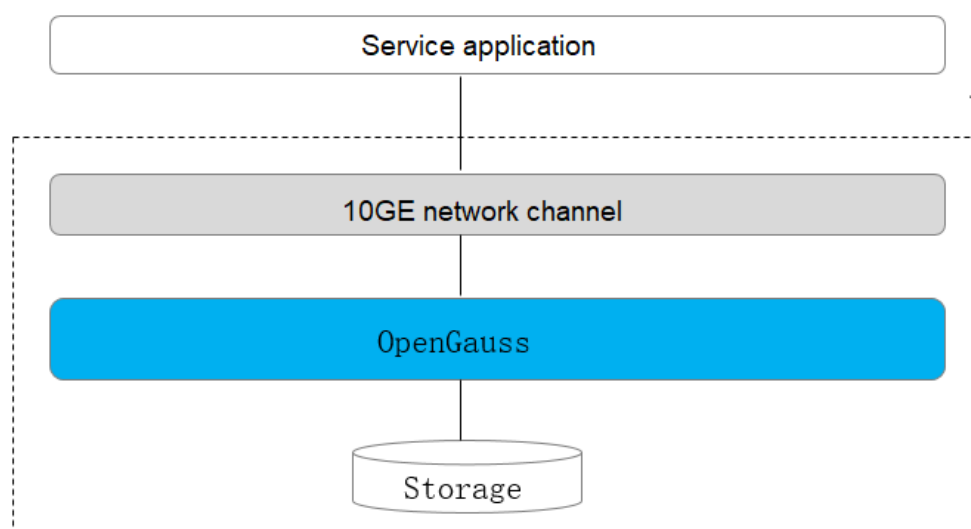
Introduction to Deployment Solutions

The overall deployment solution can be classified into three types: standalone, primary/standby, and one primary and multiple standbys.

- Standalone deployment

The standalone deployment does not ensure reliability or availability. There is only one data copy. Once data is damaged or lost, only physical backup can be used to restore data. Therefore, this deployment mode applies to scenarios such as experiencing databases and commissioning syntax functions in the test environment. You are not advised to use this mode on the commercial live network.

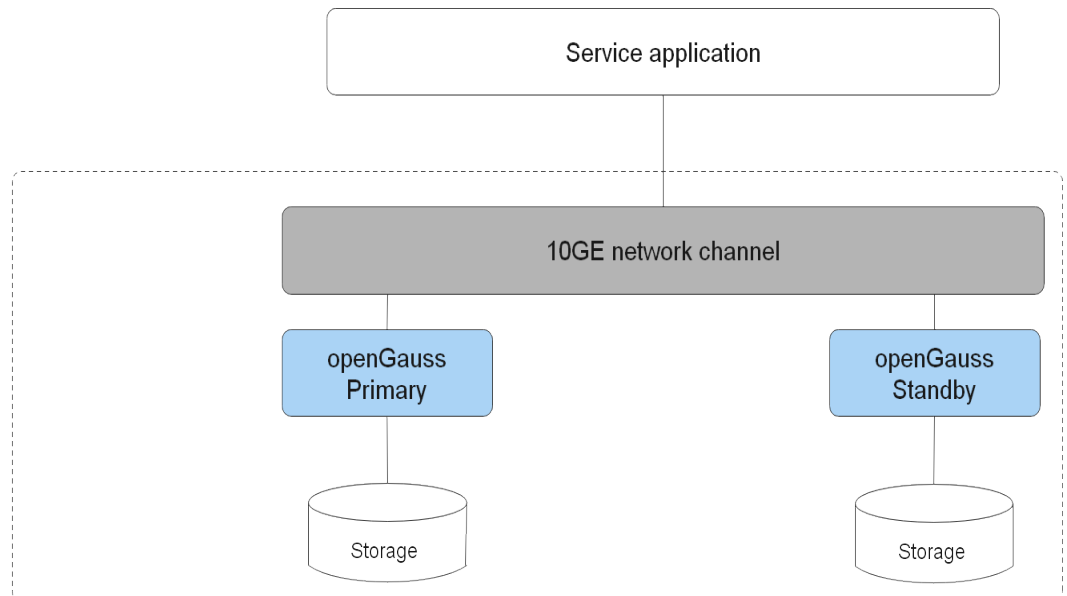
Figure 5-1 Standalone deployment



- Primary/standby deployment

The primary/standby mode is equivalent to two data copies, one for the primary node and the other for the standby node. The standby node receives logs and plays back the logs.

Figure 5-2 Primary/standby deployment



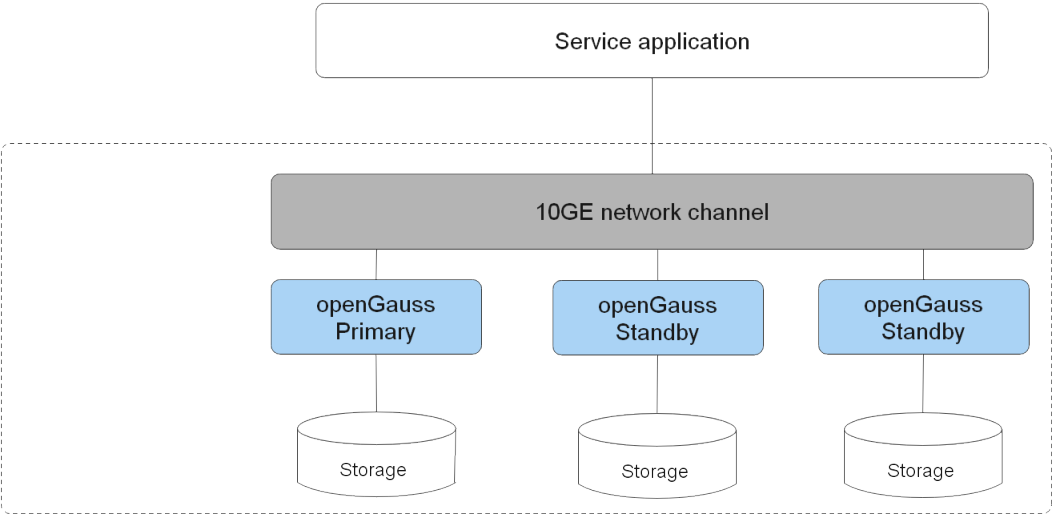
- One primary and multiple standbys deployment

The multi-copy deployment mode provides the capability of defending against instance-level faults. This mode is applicable to scenarios where equipment room DR is not required but some hardware faults need to be prevented.

Generally, the multi-copy deployment mode is one primary and two standbys. There are three copies in total. The reliability of the three copies is 99.99%, which can meet the reliability requirements of most applications.

- In primary/standby quorum replication, data is synchronized to at least one standby node to ensure the maximum performance.
- If any of primary and standby nodes is faulty, services will not be affected.
- There are three copies of data. If one node is faulty, the system still has two copies of data. In addition, any standby node can be promoted to primary.
- The primary and standby nodes cannot be deployed on the same physical machine.

Figure 5-3 One primary and multiple standbys deployment



6 Typical Networking

To ensure the security of application data, you are advised to divide the openGauss typical network into two independent networks: front-end service network and data management and storage network.

Figure 6-1 Typical networking

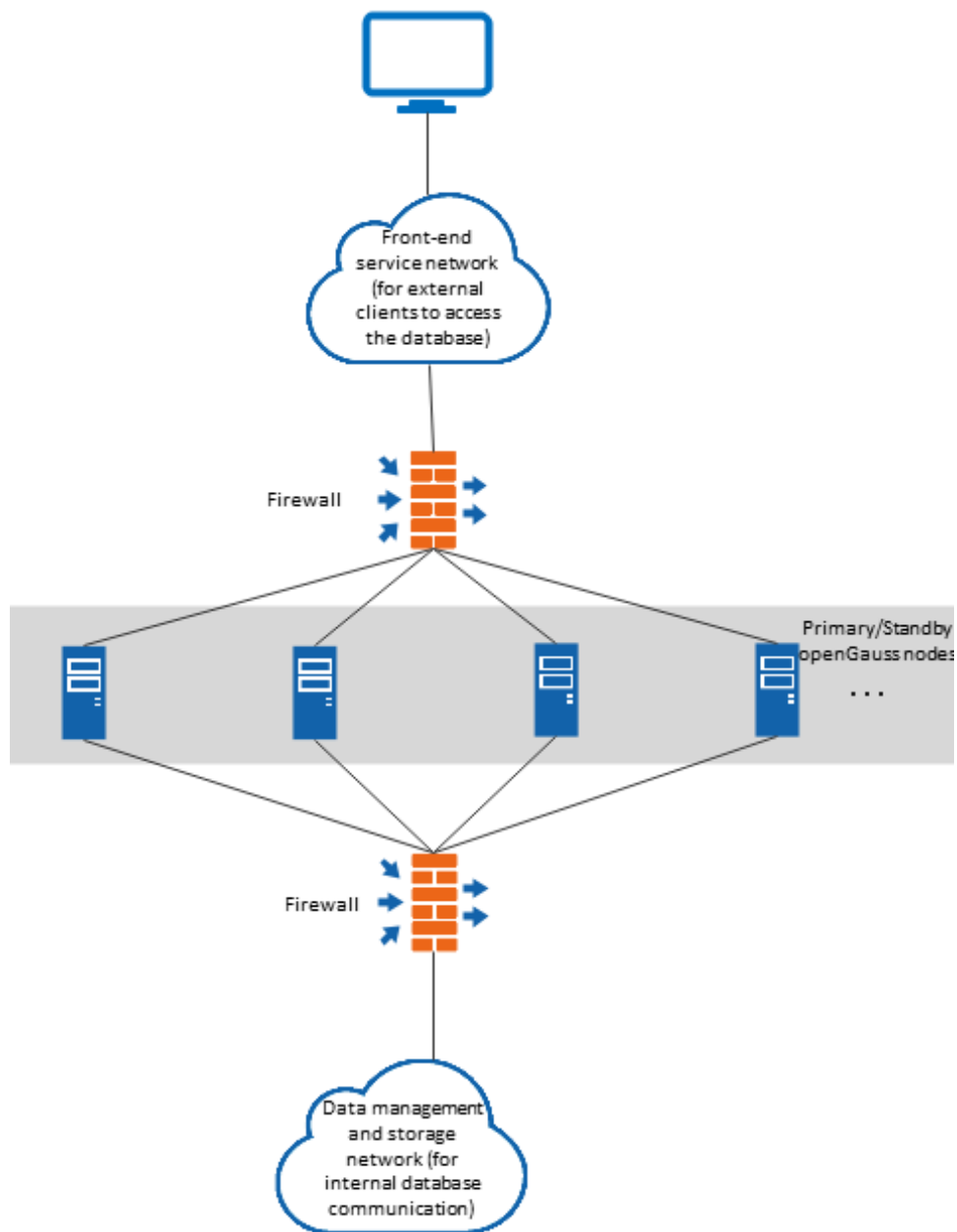


Table 6-1 describes the network division.

Table 6-1 Network division

Type	Description
Database management and storage network	The database administrator uses this network to invoke OM scripts to manage and maintain openGauss instances. It is also used for openGauss primary/standby communication networking. The database management and storage network are also used for applications to execute system monitoring.

Type	Description
Front-end service network	External clients access openGauss through this network.

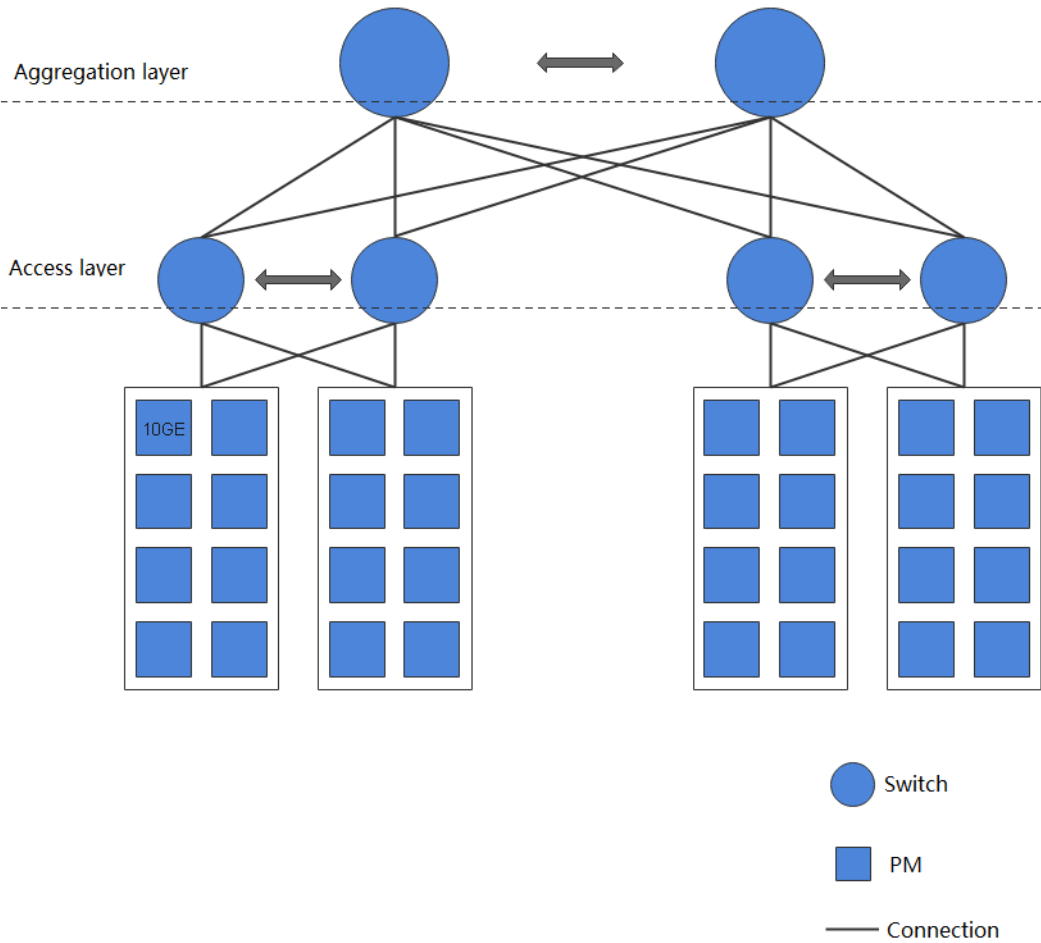
The typical networking has the following advantages:

- The service network is isolated from the database management and storage network, effectively protecting the security of back-end storage data.
- The isolation between the service network and database management and storage network prevents attackers from managing database servers through the Internet, improving system security.

Network exclusiveness and 1:1 bandwidth convergence ratio are the basic requirements for openGauss network performance. Therefore, in the production system, the back-end storage network shown in [Figure 6-1](#) must meet the requirements of exclusiveness and 1:1 bandwidth convergence ratio. For example, in [Figure 6-2](#), the Fat-tree networking is used. To achieve a convergence ratio of 1:1, the bandwidth doubles each time the switching network layer is increased by one layer. In the figure, each bold line indicates the 80GE bandwidth, that is, the sum of the bandwidth upper limits of eight physical machines. At the access layer, each switch provides 160GE downlink bandwidth and 160GE uplink bandwidth. The convergence ratio is 1:1. The access bandwidth of each switch at the aggregation layer is 320GE.

For the test system, the preceding requirements can be lowered.

Figure 6-2 Database management and storage network



7

Hardware and Software Configuration Requirements

Software Requirements

Table 7-1 Software requirements

Software	Configuration Description
OS	<ul style="list-style-type: none">• Arm:<ul style="list-style-type: none">– openEuler 20.3LTS (recommended)– Kirin V10• x86:<ul style="list-style-type: none">– openEuler 20.3LTS– CentOS 7.6
File system	The ext4 file system is recommended for EulerOS. It is recommended that the number of remaining inodes be greater than 1.5 billion.
Tool	bzip2
Python	<ul style="list-style-type: none">• openEuler: supports Python 3.7.X.• CentOS: supports Python 3.6.X.• Kirin: supports Python 3.7.X.

Hardware Requirements

Table 7-2 Hardware Requirements

Item	Configuration Description
Minimum memory	<p>Minimum 32 GB of memory is required for function debugging. In performance tests and commercial deployment, the single-instance deployment is performed. It is recommended that the memory be greater than 128 GB.</p> <p>Complex queries require much memory but the memory may be insufficient in high-concurrency scenarios. In this case, it is recommended that a large-memory server or load management be used to restrict concurrences on the system.</p>
CPU	<p>Minimum one 8-core 2.0 GHz CPU is required for function debugging.</p> <p>In performance tests and commercial deployment, the single-instance deployment is performed. It is recommended that one 16-core 2.0 GHz CPU be used.</p> <p>You can set CPUs to hyper-threading or non-hyper-threading mode, but ensure the setting is consistent across all the primary and standby nodes.</p> <p>NOTE Currently, openGauss supports only the CPUs of Kunpeng servers and x86_64-based universal PC servers.</p>
Hard disk	<p>Hard disks used for installing the instance must meet the following requirements:</p> <ul style="list-style-type: none">• At least 1 GB is used to install the instance application package.• About 300 MB is used for each host to store metadata.• More than 70% of available disk space is reserved to store data. <p>You are advised to configure the system disk to RAID 1 and data disk to RAID 5 and plan four groups of RAID 5 data disks for installing openGauss instance. RAID configuration is not described in this document. You can configure RAID by following instructions in the hardware vendors' manuals or using common methods found on the Internet. Set Disk Cache Policy to Disabled to avoid data loss in an unexpected power-off.</p> <p>openGauss supports using an SSD with the SAS API deployed in RAID mode as the primary storage device of the database.</p>

Item	Configuration Description
Network	<p>Minimum 300 Mbit/s Ethernet is required.</p> <p>You are advised to bond two NICs for redundancy. The configuration is not described in this document. You can configure it by following instructions in the manual provided by the hardware manufacturer or using the methods provided on the Internet.</p> <p>If bonds are configured for the primary and standby networks, ensure the consistency among these bond modes, because inconsistent bond modes may cause primary and standby communication exceptions.</p>

8 Core Database Technologies

[8.1 Basic Functions Oriented to Application Development](#)

[8.2 High Performance](#)

[8.3 HA](#)

[8.4 Maintainability](#)

[8.5 Database Security](#)

[8.6 AI Capabilities](#)

8.1 Basic Functions Oriented to Application Development

- Standard SQL

openGauss supports standard SQL statements. The SQL standard is an international standard and is updated periodically. SQL standards are classified into core features and optional features. Most databases do not fully support SQL standards. SQL features are built by database vendors to maintain customers and push up application migration costs. New SQL features are increasingly different among vendors. Currently, there is no authoritative SQL standard test.

openGauss supports most of the SQL:2011 core features and some optional features. For details about the feature list, see "SQL Reference > SQL Syntax" in the *Developer Guide*.

The introduction of standard SQL provides a unified SQL interface for all database vendors, reducing the learning costs of users and openGauss application migration costs.

- Standard Development Interfaces

Standard ODBC and JDBC interfaces are provided to ensure quick migration of user services to openGauss.

Currently, the standard ODBC 3.5 and JDBC 4.0 interfaces are supported. The ODBC interface supports SUSE Linux, Windows 32-bit, and Windows 64-bit platforms. The JDBC interface supports all platforms.

- **Multiple Storage Engines**
openGauss is based on the unified transaction mechanism, log system, concurrency control system, metadata information, and cache management, provides Table Access Method API, and supports different storage engines. Currently, the Astore and Ustore storage engines are supported.
- **Transaction Support**
Transaction support refers to the system capability to ensure the atomicity, consistency, isolation, and durability (ACID) features of global transactions. Transaction support and data consistency assurance are the basic functions of most databases and the prerequisites for a database to satisfy transaction-based application requirements.
 - **Atomicity**
A transaction is comprised of an indivisible unit of work. Operations performed in a transaction must be all finished or have not been performed.
 - **Consistency**
Transactions must be consistent within a system no matter when or how many concurrent transactions are ongoing.
 - **Isolation**
Transactions are isolated for execution, as if each of them is the only operation performed during the specified period planned by the system. If there are two transactions that are executed within the same period of time and performing the same function, the transaction isolation makes each of them regard itself as the only transaction using the system.
 - **Durability**
After a transaction is complete, the changes made by the transaction to the database are permanently stored in the database and will not be rolled back.

The default transaction isolation level is READ COMMITTED, ensuring no dirty data will be read.

Transactions are categorized into single-statement transactions and transaction blocks. Their basic interfaces are as follows:

 - Start transaction;
 - Commit;
 - Rollback;

Set transaction (used for setting the isolation level, read/write mode, and delay mode). For details about the syntax, see the *Developer Guide*.
- **Support for Functions and Stored Procedures**
Functions are important database objects. They encapsulate SQL statement sets used for certain functions so that the statements can be easily invoked. A stored procedure is a combination of SQL and PL/SQL. Stored procedures can move the code that executes business rules from the application to the database. Therefore, the code storage can be used by multiple programs at a time.
 - a. Allows customers to modularize program design and encapsulate SQL statement sets, easy to invoke.

- b. Caches the compilation results of stored procedures to accelerate SQL statement set execution.
- c. Allows system administrators to restrict the permission for executing a specific stored procedure and controls access to the corresponding type of data. This prevents access from unauthorized users and ensures data security.
- d. To process SQL statements, the stored procedure process assigns a memory fragment to store context association. Cursors are handles or pointers to context areas. With cursors, stored procedures can control alterations in context areas.
- e. Six levels of exception information are supported to facilitate the debugging of stored procedures. Stored procedure debugging is a debugging method. During the development of a stored procedure, you can trace the process executed by the stored procedure step by step and find the error cause or program bug based on the variable value to improve the fault locating efficiency. You can set breakpoints and perform independent debugging.

openGauss supports functions and stored procedures in the SQL standard, which enhances the usability of stored procedures. For details about how to use the stored procedures, see the *Developer Guide*.

- PG Interface Compatibility

Compatible with PostgreSQL clients and interfaces.

- SQL Hints

SQL hints are supported, which can override any execution plan and thus improve SQL query performance.

In plan hints, you can specify a join order; join, stream, and scan operations; and the number of rows in a result to tune an execution plan, improving query performance.

- Copy Interface for Error Tolerance

openGauss provides the encapsulated copy error tables for creating functions and allows users to specify error tolerance options when using the **Copy From** statement. In this way, errors related to parsing, data format, and character set during the execution of the **Copy From** statement are recorded in the error table instead of being reported and interrupted. Even if a small amount of data in the target file of **Copy From** is incorrect, the data can be imported to the database. You can locate and rectify the fault in the error table later.

8.2 High Performance

CBO Optimizer

The openGauss optimizer is a typical Cost-based Optimization (CBO). By using CBO, the database calculates the number of tuples and the execution cost for each execution step under each execution plan based on the number of table tuples, column width, NULL record ratio, and characteristic values, such as distinct, MCV, and HB values, and certain cost calculation methods. The database then selects the execution plan that takes the lowest cost for the overall execution or for the return of the first tuple.

The CBO optimizer can select the most efficient execution plan among multiple plans based on the cost to meet customer service requirements to the maximum extent.

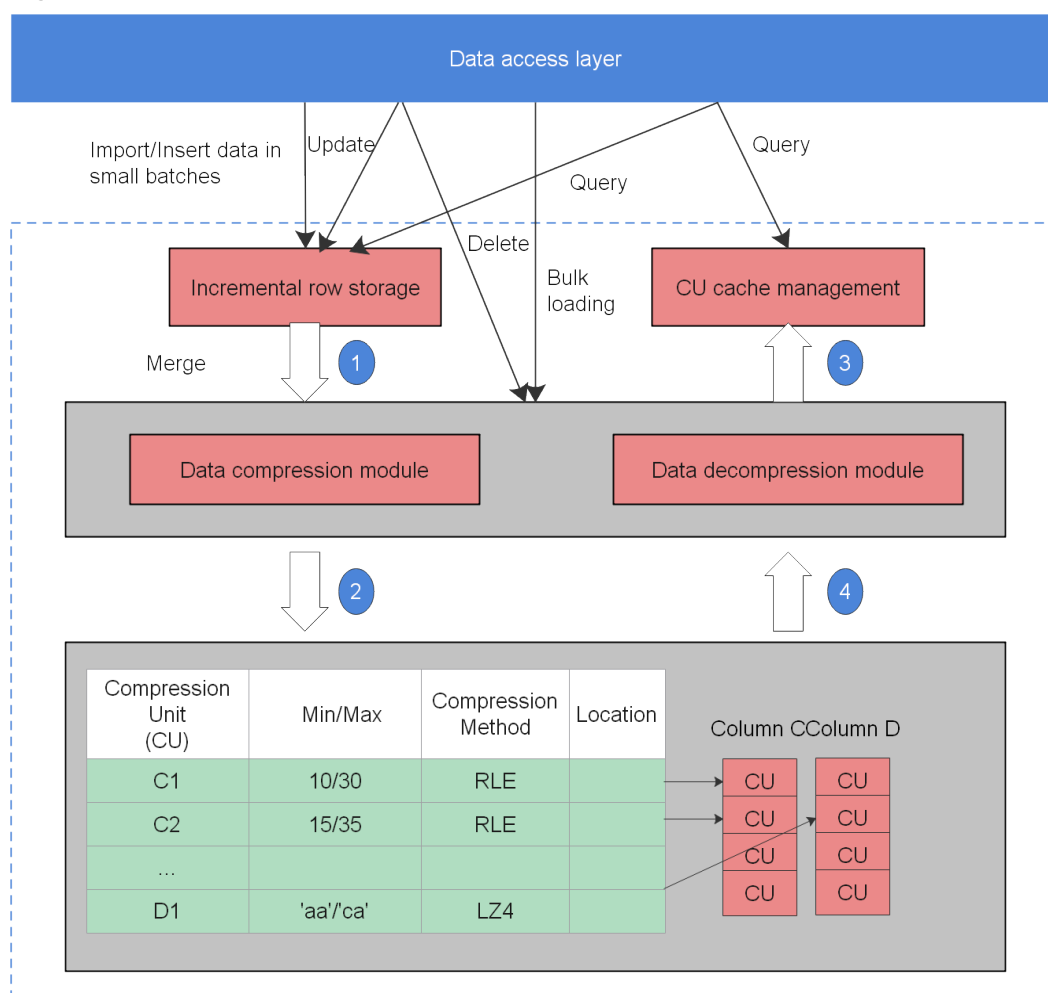
Hybrid Row-Column Storage

openGauss supports both the row-store and column-store models. Users can choose a row-store or column-store table based on their needs.

Column-store is recommended if a table contains many columns (called a wide table) but its query involves only a few columns. Row-store is recommended if a table contains only a few columns and a query involves most of the columns.

Figure 8-1 shows the column-store model.

Figure 8-1 Column-store



In a wide table containing a huge amount of data, a query usually only involves certain columns. In this case, the query performance of the row-store engine is poor. For example, a single table containing the data of a meteorological agency has 200 to 800 columns. Among these columns, only 10 are frequently accessed. In this case, the vectorized execution technology and column-store engine can significantly improve performance by saving storage space.

Row-store tables and column-store tables have their own advantages and disadvantages. You are advised to select a table based on the site requirements.

- **Row-store table**
Row-store tables are created by default. Data is stored by row. Row-store supports adding, deleting, modifying, and querying data of a complete row. Therefore, this storage model applies to scenarios where data needs to be updated frequently.
- **Column-store table**
Data is stored by column. The I/O of data query in a single column is small, and column-store tables occupy less storage space than row-store tables. This storage model applies to scenarios where data is inserted in batches, less updated, and queried for statistical analysis. The performance of single point query and single record insertion in a column-store table is poor.

The principles for selecting row-store and column-store tables are as follows:

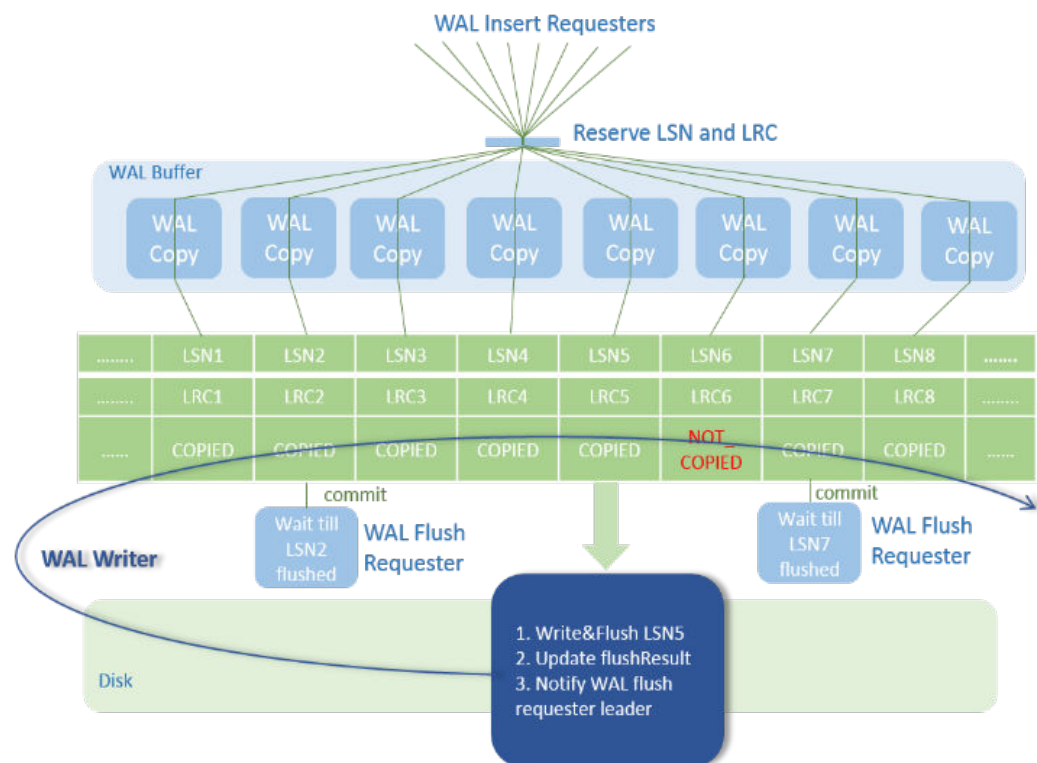
- **Update frequency**
If data is frequently updated, use a row-store table.
- **Data insertion frequency**
If a small amount of data is frequently inserted each time, use a row-store table. If a large amount of data is inserted at a time, use a column-store table.
- **Number of columns**
If a table is to contain many columns, use a column-store table.
- **Number of columns to be queried**
If only a small number of columns (less than 50% of the total) is queried each time, use a column-store table.
- **Compression ratio**
The compression ratio of a column-store table is higher than that of a row-store table. High compression ratio consumes more CPU resources.

In-place Update Storage

The in-place update storage engine solves the problems of space expansion and large tuples of the Append update storage engine. The design of efficient rollback segments is the basis of the in-place update storage engine.

Xlog Lockless Update and Parallel Page Playback

Figure 8-2 Xlog lock less Design



This feature optimizes the WalInsertLock mechanism by using log sequence numbers (LSNs) and log record counts (LRCs) to record the copy progress of each backend and canceling the WalInsertLock mechanism. The backend can directly copy logs to the WalBuffer without contending for the WalInsertLock. In addition, a dedicated WALWriter thread is used to write logs, and the backend thread does not need to ensure the Xlog flushing. After the preceding optimization, the WalInsertLock contention and WalWriter dedicated disk write threads are canceled. The system performance can be further improved while the original XLog function remains unchanged. This feature optimizes the Ustore in-place update WALs and Ustore DML operation parallel playback and distribution. Prefixes and suffixes are used to reduce the update WALs. The playback thread is divided into multiple types to solve the problem that most Ustore DML WALs are replayed on multiple pages. In addition, the Ustore data page playback is distributed based on blkno to improve the degree of parallel playback.

Adaptive Compression

Currently, mainstream databases usually use the data compression technology. Various compression algorithms are used for different data types. If pieces of data of the same type have different characteristics, their compression algorithms and results will also be different. Adaptive compression chooses the suitable compression algorithm for data based on the data type and characteristics, achieving high performance in compression ratio, import, and query.

Importing and frequently querying a huge amount of data are the main application scenarios. When you import data, adaptive compression greatly

reduces the data volume, increases I/O operation efficiency several times, and clusters data before storage, achieving fast data import. In this way, only a small number of I/O operations is required and data is quickly decompressed in a query. Data can be quickly retrieved and the query result is quickly returned.

Currently, the database has implemented various compression algorithms, including RLE, DELTA, BYTEPACK/BITPACK, LZ4, ZLIB, and LOCAL DICTIONARY. The following table lists data types and the compression algorithms suitable for them.

-	RLE	DELTA	BITPACK/ BYTEPACK	LZ4	ZLIB	LOCAL DICTIONARY
Smallint/int/bigint/Oid Decimal/real/double Money/time/date/ timestamp	√	√	√	√	√	-
Tinterval/interval/Time with time zone/	-	-	-	-	√	-
Numeric/char/varchar/ text/nvarchar2 Other supported data types	√	√	√	√	√	√

For example, large integer compression of mobile number-like character strings, large integer compression of the numeric type, and adjustment of the compression algorithm compression level are supported.

Partition

In the openGauss system, data is partitioned horizontally in an instance using a specified policy. This operation splits a table into multiple partitions that are not overlapped.

In common scenarios, a partitioned table has the following advantages over a common table:

- High query performance: You can specify partitions when querying partitioned tables, improving query efficiency.
- High availability: If a certain partition in a partitioned table is faulty, data in the other partitions is still available.
- Easy maintenance: To fix a partitioned table having a faulty partition, you only need to fix the partition.
- Balanced I/O: Partitions can be mapped to different disks to balance I/O and improve the overall system performance.

Currently, openGauss supports range partitioned tables, list partitioned tables, and hash partitioned tables.

- In a range partitioned table, data within a certain range is mapped to each partition. The range is determined by the partition key specified when the partitioned table is created. This partitioning mode is most commonly used.
With the range partitioning function, the database divides a record, which is to be inserted into a table, into multiple ranges using one or multiple columns and creates a partition for each range to store data. Partition ranges do not overlap.
- In a list partitioned table, data is mapped to each partition based on the key values contained in each partition. The key values contained in a partition are specified when the partition is created.
The list partitioning function divides the key values in the records to be inserted into a table into multiple lists (the lists do not overlap in different partitions) based on a column of the table, and then creates a partition for each list to store the corresponding data.
- In a hash partitioned table, data is mapped to each partition using the hash algorithm, and each partition stores records with the same hash value.
The hash partitioning function uses the internal hash algorithm to divide records to be inserted into a table into partitions based on a column of the table.

If you specify the **PARTITION** parameter when running the **CREATE TABLE** statement, data in the table will be partitioned.

Users can modify partition keys as needed during table creation to make the query result stored in the same or least partitions (called partition pruning), so as to obtain consecutive I/O to improve the query performance.

In actual services, time is often used as a filter criterion for query objects. Therefore, you can select the time column as the partition key. The key value range can be adjusted based on the total data volume and the data volume queried at a time.

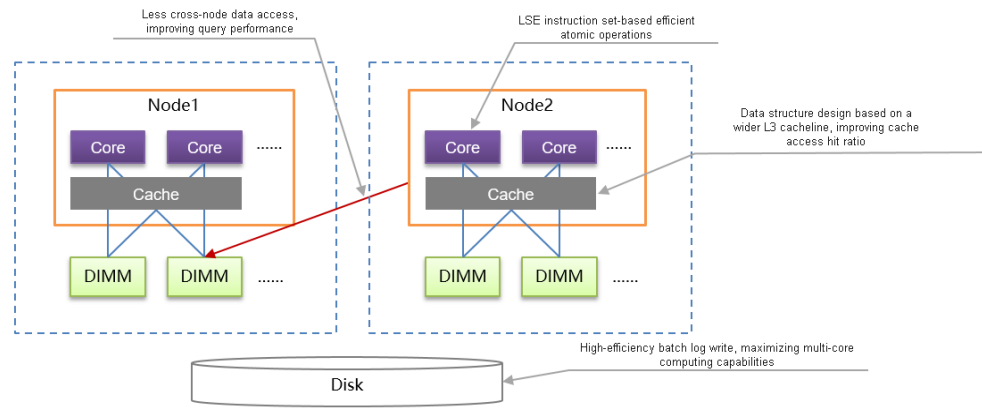
SQL by pass

In a typical OLTP scenario, simple queries account for a large proportion. This type of queries involves only single tables and simple expressions. To accelerate such query, the SQL bypass framework is proposed. After simple mode judgment is performed on such query at the parse layer, the query enters a special execution path and skips the classic execution framework, including operator initialization and execution, expression, and projection. Instead, it directly rewrites a set of simple execution paths and directly invokes storage APIs, greatly accelerating the execution of simple queries.

Kunpeng NUMA Architecture Optimization

The following figure shows the Kunpeng NUMA architecture optimization.

Figure 8-3 Kunpeng NUMA architecture optimization



1. Based on the multi-core NUMA architecture of the Kunpeng processor, openGauss optimizes the NUMA architecture to reduce the cross-core memory access latency and maximize the multi-core Kunpeng computing capability. The key technologies include redo log batch insertion, NUMA distribution of hotspot data, and Clog partitions, greatly improving the processing performance of the TP system.
2. Based on the ARMv8.1 architecture used by the Kunpeng chip, openGauss uses the LSE instruction set to implement efficient atomic operations, effectively improving the CPU usage, multi-thread synchronization performance, and XLog write performance.
3. Based on the wider L3 cache line provided by the Kunpeng chip, openGauss optimizes hotspot data access, effectively improving the cache access hit ratio, reducing the cache consistency maintenance overhead, and greatly improving the overall data access performance of the system.

High Concurrency of the Thread Pool

In the OLTP field, a database needs to process a large quantity of client connections. Therefore, the processing capability in high-concurrency scenarios is one of the important capabilities of the database.

The simplest processing mode for external connections is the per-thread-per-connection mode, in which a user connection generates a thread. This mode features simple processing thanks to its architecture. However, in high-concurrency scenarios, there are too many threads, causing heavy workload in thread switchover and large conflict between the lightweight lock areas of the database. As a result, the performance (throughput) deteriorates sharply and the SLA of user performance cannot be met.

Therefore, a thread resource pooling and reuse technology needs to be used to resolve this problem. The overall design idea of the thread pool technology is to pool thread resources and reuse them among different connections. After the system is started, a fixed number of working threads are started based on the current number of cores or user configuration. A working thread serves one or more connection sessions. In this way, the session and thread are decoupled. The number of worker threads is fixed. Therefore, frequent thread switchover does not occur in case of high concurrency. The database layer schedules and manages sessions.

Parallel Query

The Symmetric Multi-Processing (SMP) parallel technology of openGauss uses the multi-core CPU architecture of a computer to implement multi-thread parallel computing, fully using CPU resources to improve query performance. In complex query scenarios, a single query execution takes long time and the system concurrency is low. Therefore, the SMP parallel execution technology is used to implement operator-level parallel execution, which effectively reduces the query execution time and improves the query performance and resource utilization. The overall implementation of the SMP parallel technology is as follows: For query operators that can be executed in parallel, data is sliced, multiple working threads are started for computation, and then the results are summarized and returned to the frontend. The data interaction operator **Stream** is added to SMP parallel execution to implement data interaction between multiple working threads, ensuring the correctness of the query and completing the overall query.

Dynamic Build and Execution

Based on the query execution plan tree, with the library functions provided by the LLVM, openGauss moves the process of determining the actual execution path from the executor phase to the execution initialization phase. In this way, problems such as function calling, logic condition branch determination, and a large amount of data reading that are related to the original query execution are avoided, to improve the query performance.

8.3 HA

Primary/Standby

To ensure that a fault can be rectified, data needs to be written into multiple copies. Multiple copies are configured for the primary and standby nodes, and logs are used for data synchronization. In this way, openGauss has no data lost when a node is faulty or the system restarts after a stop, meeting the ACID feature requirements. The primary/standby environment supports two modes: primary/standby, and one primary and multiple standbys. In primary/standby mode, if the standby node needs to redo logs, it can be promoted to primary. In the one primary and multiple standbys mode, all standby nodes need to redo logs and can be promoted to primary. The primary/standby mode is mainly used for OLTP systems with general reliability to save storage resources. The one primary and multiple standbys mode provides higher DR capabilities and is suitable for the OLTP system with higher availability transaction processing.

The **switchover** command can be used to trigger a switchover between the primary and standby nodes. If the primary node is faulty, the **failover** command can be used to promote the standby node to the primary.

To ensure that the failover time is controllable, you can enable the log flow control function to control the rate of sending logs to the standby node. This ensures that the logs accumulated on the standby node will be replayed within the time configured for flow control. After flow control is enabled, the rate of sending logs to the standby node is dynamically adjusted. As a result, the overall transaction performance deteriorates.

In scenarios such as initial installation or backup and restoration, data on the standby node needs to be rebuilt based on the primary node. In this case, the build function is required to send the data and WALs of the primary node to the standby node. When the primary node is faulty and joins again as a standby node, the build function needs to be used to synchronize data and WALs with those of the new primary node. Build includes full build and incremental build. Full build depends on primary node data for rebuild. The amount of data to be copied is large and the time required is long. Incremental build copies only differential files. The amount of data to be copied is small and the time required is short. Generally, the incremental build is preferred for fault recovery. If the incremental build fails, the full build continues until the fault is rectified.

In addition to streaming replication in primary/standby mode, openGauss also supports logical replication. In logical replication, the primary database is called the source database, and the standby database is called the target database. The source database parses the WAL file based on the specified logical parsing rule and parses the DML operation into certain logical change information (standard SQL statements). The source database sends standard SQL statements to the target database. After receiving the SQL statements, the target database applies them to implement data synchronization. Logical replication involves only DML operations. Logical replication can implement cross-version replication, heterogeneous database replication, dual-write database replication, and table-level replication.

Logical Backup

openGauss provides the logical backup capability to back up data in user tables to local disk files in text or CSV format and restore the data in homogeneous or heterogeneous databases.

Physical Backup

openGauss provides the physical backup capability to back up data of the entire instance to local disk files in the internal database format, and restore data of the entire instance in a homogeneous database.

Physical backup is classified into full backup and incremental backup. The difference is as follows: Full backup includes the full data of the database at the backup time point. The time required for full backup is long (in direct proportion to the total data volume of the database), and a complete database can be restored. An incremental backup involves only incremental data modified after a specified time point. It takes a short period of time (in direct proportion to the incremental data volume and irrelevant to the total data volume). However, a complete database can be restored only after the incremental backup and full backup are performed. openGauss supports both full and incremental backup modes.

Flashback Restoration

The flashback function is used to restore dropped tables from the recycle bin. Like in a Window OS, dropped table information is stored in the recycle bin of databases. The MVCC mechanism is used to restore data to a specified point in time or system change number (SCN).

Ultimate RTO

After the ultimate RTO function is enabled, multi-level pipelines are established for Xlog log playback to improve the concurrency and log playback speed.

When the service load is heavy, the playback speed of the standby node cannot catch up with that of the primary node. After the system runs for a long time, logs are accumulated on the standby node. If a host is faulty, data restoration takes a long time and the database is unavailable, which severely affects system availability. The ultimate recovery time object (RTO) is enabled to reduce the data recovery time after a host fault occurs and improve availability.

Logical Replication

openGauss provides the logical decoding function to reversely parse physical logs into logical logs. Logical replication tools such as DRS convert logical logs to SQL statements and replay the SQL statements in the peer database. In this way, data can be synchronized between heterogeneous databases. Currently, unidirectional and bidirectional logical replication between the openGauss database and the MySQL or Oracle database is supported. DNs reversely parse physical logs to logical logs. Logical replication tools such as DRS extract logical logs from DNs, convert the logs to SQL statements, and replay the SQL statements in MySQL. Logical replication tools also extract logical logs from a MySQL database, reversely parse the logs to SQL statements, and replay the SQL statements in openGauss. In this way, data can be synchronized between heterogeneous databases.

Point-In-Time Recovery (PITR)

PITR uses basic hot backup, WALs, and WAL archive logs for backup and recovery. When replaying a WAL record, you can stop at any point in time, so that there is a snapshot of the consistent database at any point in time. That is, you can restore the database to the state at any time since the backup starts. During recovery, openGauss supports specifying the recovery stop point as TID, time, and LSN.

High Availability Based on the Paxos Protocol (DCF)

After DCF is enabled, DNs support Paxos-based replication and quorum, achieving high availability and disaster recovery. DNs support automatic primary node selection and log replication. The replication process supports compression and stream control to prevent high bandwidth usage. Node types based on Paxos roles are provided and can be adjusted.

8.4 Maintainability

Workload Diagnosis Report

The workload diagnosis report (WDR) generates a performance report between two different time points based on the system performance snapshot data at two different time points. The report is used to diagnose database kernel performance faults.

WDR depends on the following two components:

- **SNAPSHOT:** The performance snapshot can be configured to collect a certain amount of performance data from the kernel at a specified interval and store the data in the user tablespace. Any snapshot can be used as a performance baseline for comparison with other snapshots.
- **WDR Reporter:** This tool analyzes the overall system performance based on two snapshots, calculates the changes of more specific performance indicators between the two time periods, and generates summarized and detailed performance data. For details, see [Table 8-1](#) and [Table 8-2](#).

Table 8-1 Summarized diagnosis report

Diagnosis Type	Description
Database Stat	<p>Evaluates the load and I/O status of the current database. Load and I/O are the most important indicators of the TP system.</p> <p>The statistics include the number of sessions connected to the database, number of committed and rolled back transactions, number of read disk blocks, number of disk blocks found in the cache, number of rows returned, captured, inserted, updated, and deleted through database query, number of conflicts and deadlocks, usage of temporary files, and I/O read/write time.</p>
Load Profile	<p>Evaluates the current system load from the time, I/O, transaction, and SQL dimensions.</p> <p>The statistics include the job running elapse time, CPU time, daily transaction quality, logical and physical read volume, read and write I/O times and size, login and logout times, SQL, transaction execution volume, and SQL P85 and P90 response time.</p>
Instance Efficiency Percentages	<p>Evaluates the cache efficiency of the current system.</p> <p>The statistics include the database cache hit ratio.</p>
Events	<p>Evaluates the performance of key system kernel resources and key events.</p> <p>The statistics include the number of times that the key time of the database kernel occurs and the waiting time.</p>
Wait Classes	<p>Evaluates the performance of key events in the system.</p> <p>The statistics include the release of the data kernel in the main types of waiting events, such as STATUS, LWLOCK_EVENT, LOCK_EVENT, and IO_EVENT.</p>
CPU	<p>Includes time release of the CPU in user mode, kernel mode, wait I/O, and idle mode.</p>
IO Profile	<p>Includes the number of database I/O times, database I/O data volume, number of redo I/O times, and redo I/O volume.</p>

Diagnosis Type	Description
Memory Statistics	Includes maximum process memory, used process memory, maximum shared memory, and used shared memory.

Table 8-2 Detailed diagnosis report

Diagnosis Type	Description
Time Model	Evaluates the performance of the current system in the time dimension. The statistics include time consumed by the system in each phase, including the kernel time, CPU time, execution time, parsing time, compilation time, query rewriting time, plan generation time, network time, and I/O time.
SQL Statistics	Diagnoses SQL statement performance problems. The statistics include normalized SQL performance indicators in multiple dimensions: elapsed time, CPU time, rows returned, tuple reads, executions, physical reads, and logical reads. The indicators can be classified into execution time, number of execution times, row activity, and cache I/O.
Wait Events	Diagnoses performance of key system resources and key time in detail. The statistics include the performance of all key events in a period of time, including the number of events and the time consumed.
Cache IO Stats	Diagnoses the performance of user tables and indexes. The statistics include read and write operations on all user tables and indexes, and the cache hit ratio.
Utility status	Diagnoses the performance of backend jobs. The statistics include the performance of backend operations such as page operation and replication.
Object stats	Diagnoses the performance of database objects. The statistics include user tables, tables on indexes, index scan activities, insert, update, and delete activities, number of valid rows, and table maintenance status.
Configuration settings	Determines whether the configuration is changed. It is a snapshot that contains all current configuration parameters.

Benefits:

- WDR is the main method for diagnosing long-term performance problems. Based on the performance baseline of a snapshot, performance analysis is performed from multiple dimensions, helping DBAs understand the system load, performance of each component, and performance bottlenecks.
- Snapshots are also an important data source for subsequent performance problem self-diagnosis and self-optimization suggestions.

Slow SQL Diagnosis

Slow SQL records information about all jobs whose execution time exceeds the threshold.

Historical slow SQL provides table-based and function-based query interfaces. You can query the execution plan, start time, end time, query statement, row activity, kernel time, CPU time, execution time, parsing time, compilation time, query rewriting time, plan generation time, network time, I/O time, network overhead, and lock overhead. All information is anonymized.

Slow SQL provides detailed information required for slow SQL diagnosis. You can diagnose performance problems of specific slow SQL statements offline without reproducing the problem. The table-based and function-based APIs help users collect statistics on slow SQL indicators and connect to third-party platforms.

8.5 Database Security

Access Control

Access control is to manage users' database access control permissions, including database system permissions and object permissions.

Role-based access control is supported. Roles and permissions are associated. Permissions are assigned to roles and then roles are assigned to users, implementing user access control permission management. The login access control is implemented by using the user ID and authentication technology. The object access control is implemented by checking the object permission based on the user permission on the object. You can assign the minimum permissions required for completing tasks to related database users to minimize database usage risks.

An access control model based on separation of permissions is supported. Database roles are classified into system administrator, security administrator, and audit administrator. The security administrator creates and manages users, the system administrator grants and revokes user permissions, and the audit administrator audits all user behaviors.

By default, the role-based access control model is used. You can set parameters to determine whether to enable the access control model based on separation of permissions.

Separation of Control and Access Permissions

For the system administrator, the control and access permissions on table objects are separated to improve data security of common users and restrict the object access permissions of administrators.

This feature applies to the following scenarios: An enterprise has multiple business departments using different database users to perform service operations. Database maintenance departments at the same level use the database administrator to perform O&M operations. The business departments require that administrators can only perform control operations (**DROP**, **ALTER**, and **TRUNCATE**) on data of each department and cannot perform access operations (**INSERT**, **DELETE**, **UPDATE**, **SELECT**, and **COPY**) without authorization. That is, the control permissions of database administrators for tables need to be isolated from their access permissions to improve the data security of common users.

The system administrators can specify the **INDEPENDENT** attribute when creating a user, indicating that the user is a private user. Database administrators (including initial users and other administrators) can control (**DROP**, **ALTER**, and **TRUNCATE**) objects of private users but cannot access (**INSERT**, **DELETE**, **UPDATE**, **SELECT**, **COPY**, **GRANT**, **REVOKE**, and **ALTER OWNER**) the objects without authorization.

Built-in Database Role Permission Management

openGauss provides a group of default roles whose names start with **gs_role_**. These roles are provided to access to specific, typically high-privileged operations. You can grant these roles to other users or roles within the database so that they can use specific functions. These roles should be given with great care to ensure that they are used where they are needed. [Table 8-3](#) describes the permissions of built-in roles.

Table 8-3 Built-in role permissions

Role	Permission
gs_role_copy_files	Permission to run the copy... to/from filename command. However, the GUC parameter enable_copy_server_files must be set first to enable the function of copying server files.
gs_role_signal_backend	Permission to invoke the pg_cancel_backend , pg_terminate_backend , and pg_terminate_session functions to cancel or terminate other sessions. However, this role cannot perform operations on sessions of the initial user or PERSISTENCE user.
gs_role_tablespace	Permission to create a tablespace.
gs_role_replication	Permission to invoke logical replication functions, such as kill_snapshot , pg_create_logical_replication_slot , pg_create_physical_replication_slot , pg_drop_replication_slot , pg_replication_slot_advance , pg_create_physical_replication_slot_extern , pg_logical_slot_get_changes , pg_logical_slot_peek_changes , pg_logical_slot_get_binary_changes and pg_logical_slot_peek_binary_changes .
gs_role_account_lock	Permission to lock and unlock users. However, this role cannot lock or unlock the initial user or PERSISTENCE user.

Role	Permission
gs_role_directory_create	Permission to create directory objects. However, this role needs to enable the GUC parameter enable_access_server_directory first.
gs_role_directory_drop	Permission to delete directory objects. However, this role needs to enable the GUC parameter enable_access_server_directory first.

Database Encryption Authentication

The password encryption method based on the RFC5802 mechanism is used for authentication.

The unidirectional, irreversible Hash encryption algorithm PBKDF2 is used for encryption and authentication, effectively defending against rainbow attacks.

The password of the created user is encrypted and stored in the system catalog. During the entire authentication process, passwords are encrypted for storage and transmission. The hash value is calculated and compared with the value stored on the server to verify the correctness.

The message processing flow in the unified encryption and authentication process effectively prevents attackers from cracking the username or password by capturing packets.

Database Audit

Audit logs record user operations performed on database startup and stopping, connection, and DDL, DML, and DCL operations. The audit log mechanism enhances the database capability of tracing illegal operations and collecting evidence.

You can set parameters to specify the statements or operations for which audit logs are recorded.

Audit logs record the event time, type, execution result, username, database, connection information, database object, database instance name, port number, and details. You can query audit logs by start time and end time and filter audit logs by recorded field.

Database security administrators can use the audit logs to reproduce a series of events that cause faults in the database and identify unauthorized users, unauthorized operations, and the time when these operations are performed.

Equality Query in a Fully-encrypted Database

Compared with a common database, a fully-encrypted database inherits its basic functions and provides the capability of computing data ciphertext. In the fully-encrypted database solution, data encryption and decryption are performed only on the client, and the server processes only ciphertext data. In addition to the user client medium, ciphertext data is only exposed to ciphertext-only attacks. To address issues such as database privacy leakage and third-party trust, you can use a fully-encrypted database solution that provides full data lifecycle protection.

For a server that processes only ciphertext data, ensure that the performance impact is within the acceptable range. The server needs to inherit the capabilities of the native database, such as generating execution plans, processing transaction consistency, and retrieving and computing data. For a client that requires additional encryption capabilities, provide capabilities such as identifying and encrypting/decrypting private data, rewriting SQL statements that contain private data, and managing multi-level keys on the premise that encryption/decryption automation and transparency are ensured.

Network Communication Security

SSL can be used to encrypt communication data between the client and server, ensuring communication security between the client and server.

The TLS 1.2 protocol and a highly secure encryption algorithm suite are adopted. [Table 8-4](#) lists the supported encryption algorithm suites.

Table 8-4 Encryption algorithm suites

OpenSSL Suite Name	IANA Suite Name	Security
DHE-RSA-AES128-GCM-SHA256	TLS_DHE_RSA_WITH_AES_128_GCM_SHA256	HIGH
DHE-RSA-AES256-GCM-SHA384	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384	HIGH
DHE-RSA-AES128-CCM	TLS_DHE_RSA_WITH_AES_128_CCM	HIGH
DHE-RSA-AES256-CCM	TLS_DHE_RSA_WITH_AES_256_CCM	HIGH
ECDHE-RSA-AES256-GCM-SHA384	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	HIGH
ECDHE-RSA-AES128-GCM-SHA256	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	HIGH
ECDHE-ECDSA-AES256-GCM-SHA384	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	HIGH
ECDHE-ECDSA-AES128-GCM-SHA256	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	HIGH

Row-Level Security

The row-level security (RLS) feature enables database access control to be accurate to each row of data tables. When different users perform the same SQL query operation, the read results may be different according to the RLS policy.

You can create an RLS policy for a data table. The policy defines an expression that takes effect only for specific database users and SQL operations. When a database user accesses the data table, if a SQL statement meets the specified RLS policy of the data table, the expressions that meet the specified condition will be combined by using **AND** or **OR** based on the attribute type (**PERMISSIVE** | **RESTRICTIVE**) and applied to the execution plan in the query optimization phase.

RLS is used to control the visibility of row-level data in tables. By predefining filters for data tables, the expressions that meet the specified condition can be applied to execution plans in the query optimization phase, which will affect the final execution result. Currently, RLS supports the following SQL statements: SELECT, UPDATE, and DELETE.

Resource Labels

The resource label feature classifies database resources based on user-defined rules to implement resource classification and management. Administrators can configure resource labels to configure security policies, such as auditing or data masking, for a group of database resources.

Resource labels can be used to group database resources based on features and application scenarios. You can manage all database resources with specified labels, which greatly reduces policy configuration complexity and information redundancy and improves management efficiency.

Currently, resource labels support the following database resource types: schema, table, column, view, and function.

Dynamic Data Masking

To prevent unauthorized users from sniffing privacy data, the dynamic data masking feature can be used to protect user privacy data. When an unauthorized user accesses the data for which a dynamic data masking policy is configured, the database returns the anonymized data to protect privacy data.

Administrators can create dynamic data masking policies on data columns. The policies specify the data masking methods for specific user scenarios. After the dynamic data masking function is enabled, the system matches user identity information (such as the access IP address, client tool, and username) with the masking policy when a user accesses data in the sensitive column. After the matching is successful, the system masks the sensitive data in the query result of the column based on the masking policy.

The purpose of dynamic data masking is to flexibly protect privacy data by configuring the filter, and specifying sensitive column labels and corresponding masking functions in the masking policy without changing the source data.

Unified Auditing

Unified auditing allows administrators to configure audit policies for database resources or resource labels to simplify management, generate audit logs, reduce redundant audit logs, and improve management efficiency.

Administrators can customize audit policies for configuring operation behaviors or database resources. The policies are used to audit specific user scenarios, user behaviors, or database resources. After the unified auditing function is enabled, when a user accesses the database, the system matches the corresponding unified audit policy based on the user identity information, such as the access IP address, client tool, and username. Then, the system classifies the user behaviors based on the access resource label and user operation type (DML or DDL) in the policy to perform unified auditing.

The purpose of unified auditing is to change the existing traditional audit behavior into specific tracking audit behavior and exclude other behaviors from the audit, thereby simplifying management and improving the security of audit data generated by the database.

Password Strength Verification

To harden the security of customer accounts and data, do not set weak passwords. You need to specify a password when initializing the database, creating a user, or modifying a user. The password must meet the strength requirements. Otherwise, the system prompts you to enter the password again.

The account password complexity policy restricts the minimum number of uppercase letters, lowercase letters, digits, and special characters in a password, the maximum and minimum length of a password, the password cannot be the same as the username or the reverse of the username, and the password cannot be a weak password. This policy enhances user account security.

Weak passwords are easy to crack. The definition of weak passwords may vary with users or user groups. Users can define their own weak passwords.

The **password_policy** parameter specifies whether to enable the password strength verification mechanism. The default value is **1**, indicating that the password strength verification mechanism is enabled.

Data Encryption and Storage

Imported data is encrypted before stored.

This feature provides data encryption and decryption APIs for users and uses encryption functions to encrypt sensitive information columns identified by users, so that data can be stored in tables after being encrypted.

If you need to encrypt the entire table, you need to write an encryption function for each column. Different attribute columns can use different input parameters.

If a user with the required permission wants to view specific data, the user can decrypt required columns using the decryption function API.

Transparent Data Encryption

Transparent data encryption (TDE) encrypts data when the database writes the data to the storage medium and automatically decrypts the data when reading the data from the storage medium. This prevents attackers from reading data in the data file without database authentication, solving the static data leakage problem. This function is almost transparent to the application layer. You can determine whether to enable the transparent data encryption function as required.

The three-layer key structure is used to implement the key management mechanism, including the root key (RK), cluster master key (CMK), and data encryption key (DEK). CMKs are encrypted and protected by RKs, and DEKs are encrypted and protected by CMKs. DEKs are used to encrypt and decrypt user data. Each table corresponds to a DEK.

Table-level encryption is supported. When creating a table, you can specify whether to encrypt the table and the encryption algorithm to be used. The

encryption algorithm can be AES-128-CTR or SM4-CTR, which cannot be changed once specified. If an encrypted table is created, the database automatically applies for a DEK for the table and saves the encryption algorithm, key ciphertext, and corresponding CMK ID in the **reloptions** column of the pg_class system catalog in keyword=value format.

You can change the encryption status of an encrypted table, that is, from an encrypted table to a non-encrypted table or from a non-encrypted table to an encrypted table. If the encryption function is not enabled when a table is created, the table cannot be switched to an encrypted table.

For encrypted tables, DEK rotation is supported. After the key rotation, the data encrypted using the old key is decrypted using the old key, and the newly written data is encrypted using the new key. The encryption algorithm is not changed during key rotation.

The key management service is provided by the external KMS. The current version can interconnect with HUAWEI CLOUD KMS.

Currently, only row-store tables can be encrypted. For more feature constraints, see the *Feature Description*.

Ledger Database

To prevent database O&M personnel from stealing, tampering with, and erasing traces of the database, you can use the ledger database feature to perform comprehensive audit and trace the history. When a tamper-proof user table is modified, the database records the modification behavior to the history table where only data can be appended. In this way, the operation history can be recorded and the operation source can be traced.

The ledger database stores and verifies historical operations by generating data hash digests. Ledgers refer to user history tables and global blockchain tables. For table-level data modification operations, the system records the operation information and hash digest in a global blockchain table. In addition, each tamper-proof user table corresponds to a user history table to record the hash digest of row-level data changes. You can determine whether the user table is tampered by recalculating the hash digest and verifying the hash digest consistency.

Each record in the ledger represents a given operation fact that has occurred. The content of the record can only be appended and cannot be modified. The consistency between the tamper-proof user table and the corresponding history table can be checked to identify and track the tampering behavior. In addition, the ledger database provides an API for checking the tamper-proof user table consistency and an API for restoring and archiving history tables to meet the requirements of tampering identification, data expansion and mitigation, and historical data restoration and archiving.

8.6 AI Capabilities

AI4DB

AI4DB includes intelligent parameter tuning and diagnosis, slow SQL discovery, index recommendation, time sequence prediction, and exception detection. It

provides users with more convenient O&M operations and performance improvement, and implements functions such as self-tuning, self-monitoring, and self-diagnosis.

DB4AI

DB4AI is compatible with the MADlib ecosystem, supports more than 70 algorithms, and delivers performance several times higher than that of MADlib on PostgreSQL. Advanced and common algorithm suites such as XGBoost, prophet, and GBDT are added to supplement the shortcomings of the MADlib ecosystem. The technology stack from SQL to machine learning is unified to implement one-click driving of SQL statements from data management to model training.

The fenced UDF and native DB4AI algorithm capabilities are provided, including the execution plan, operators, and SQL syntax in the database.

9 Technical Specifications

Technical Specifications	Maximum Value
Database capacity	Varying with the OS and hardware
Size of a table	32 TB
Size of data in each row	1 GB
Size of a single field in each record	1 GB
Number of records in each table	$2^{32} \times (8 \text{ KB/Row width})$. At the code level, a single table can contain a maximum of 2^{32} pages, and the size of each page is 8 KB. Assume that the current data row width is 1 KB. The number of records in a single table is $2^{32} \times 8 = 2^{35}$. The current page size is 8 KB, and each page contains eight rows of data.
Number of columns in each table	250 to 1600 (varying with the field type)
Number of indexes in each table	Unlimited
Number of columns contained in a composite index	32
Database name length	64
Object name length (excluding the database name)	64
Number of constraints in each table	Unlimited
Number of concurrent connections	10000
Number of partitions in a partitioned table	32767

Technical Specifications	Maximum Value
Size of each partition in a partitioned table	32 TB
Number of records in each partition of a partitioned table	2^{55}
Maximum LOB capacity	(1 GB – 8203) B
Maximum length of SQL text	1048576 B

10 Glossary

Table 10-1 Glossary

Terms	Description
A - E	
ACID	Atomicity, consistency, isolation, and durability (ACID) is a set of properties that guarantee that transactions are processed reliably in a database management system (DBMS).
AZ	Acronym for available zone which usually refers to an equipment room.
bgwriter	A background write thread created when the database starts. The thread is used to flush dirty pages out of the database to a permanent device (such as a disk).
bit	Short for binary digit. The smallest unit of information handled by a computer. One bit expresses a 1 or a 0 in a binary numeral, or a true or false logical condition, and is represented physically by an element such as a high or low voltage at one point in a circuit or a small spot on a disk magnetized one way or the other. A single bit conveys little information a human would consider meaningful. A group of 8 bits, however, makes up a byte, which can be used to represent many types of information, such as a letter of the alphabet, a decimal digit, or other character.
bloom filter	Bloom filter is a space-efficient binary vectorized data structure, conceived by Burton Howard Bloom in 1970, that is used to test whether an element is a member of a set. False positive matches are possible, but false negatives are not. In other words, a query returns either "possibly in set (possible error)" or "definitely not in set". In the cases, Bloom filter sacrificed the accuracy for time and space.
CEK	Acronym for column encryption key.

Terms	Description
CIDR	Acronym for classless inter-domain routing. Whereas classical network design for IPv4 sized the network prefix as one or more 8-bit groups, resulting in the blocks of Class A (8-bit), B (16-bit), or C (24-bit) addresses, CIDR allocates address space on any address bit boundary. A CIDR address is in the format of <i>IP address/Number of bits in a network ID</i> . For example, in 192.168.23.35/21 , 21 indicates that the first 21 bits are the network prefix and others are the host ID.
CLI	Acronym for command-line interface. A means of communication between a program and its user, based solely on textual input and output. Commands are input with the help of a keyboard or similar device and are interpreted and executed by the program. Results are output as text or graphics to the terminal.
CMK	Acronym for client master key in the full encryption scenario.
CU	Acronym for compression unit. It is the smallest storage unit in a column-store table.
core file	<p>A file that is created for further analysis when memory overwriting, assertion failures, or access to invalid memory occurs in a program, causing a process to fail.</p> <p>A core file stores memory dump data, and supports binary mode and specified ports. The name of a core file consists of the word "core" and the OS process ID.</p> <p>The core file is available regardless of the type of platform.</p>
core dump	A core dump, memory dump, or system dump consists of the recorded state of the working memory of a computer program at a specific time, generally when the program stops abnormally. In practice, other key pieces of program state are usually dumped at the same time, including the processor registers, which may include the program counter and stack pointer, memory management information, and other processor and OS flags and information. Core dumps are often used to assist in diagnosing and debugging errors in computer programs.
DBA	Acronym for database administrator who instructs or executes database maintenance operations.
DBLINK	An object of the path from one database to another database. Using DBLINK, you can query a remote database object.
DBMS	Acronym for database management system. It is a piece of system management software that allows users to access information in a database. It is a collection of programs that allows users to access, manage, and query data in a database. A DBMS can be classified as memory DBMS or disk DBMS based on the location of the data.
DCL	Acronym for data control language.

Terms	Description
DDL	Acronym for data definition language.
DEK	Acronym for data encryption key.
DML	Acronym for data manipulation language.
backup	A backup copy or an act of creating a backup refers to copying and archiving computer data for purposes of recovery in case the original copy of data is lost.
backup and restoration	A collection of concepts, procedures, and strategies to prevent data loss caused by invalid media or misoperations.
standby node	A node in the openGauss primary/standby solution. It functions as a backup of the primary node. If the primary node fails, the standby node is promoted to primary, ensuring uninterrupted data services.
crash	A crash (or system breakdown) is an event that a computer or program, such as a software application or an operating system, fails to function correctly. If a program experiences a crash, it often automatically exits. The program responsible may appear to freeze or hang until a crash reporting service reports the crash and any details relating to it. If the program is a critical part of the operating system, the entire system may crash or hang, often resulting in a kernel panic or fatal system error.
coding	Coding is representing data and information using code so that it can be processed and analyzed by a computer. Characters, digits, and other objects can be converted into digital code, or information and data can be converted into the required electrical pulse signals based on predefined rules.
encoding technology	A technology that presents data using a specific set of characters, which can be identified by computer hardware and software.
table	A table consists of columns and rows. Each column is referred to as a field. The value in each field represents a data type. For example, if a table contains three fields Name , City , and State , it has three columns Name , City , and State . In every row of the table, the Name column contains a name, the City column contains a city, and the State column contains a state.
tablespace	A tablespace is a logical storage structure that contains tables, indexes, large objects, and long data. A tablespace provides an abstract layer between physical data and logical data, and provides storage space for all database objects. When you create an object, you can specify which tablespace it belongs to.

Terms	Description
concurrency control	A DBMS service that ensures data integrity when multiple transactions are concurrently executed in a multi-user environment. In a multi-threaded openGauss environment, concurrency control ensures that database operations are safe and all database transactions remain consistent at any given time.
query	A request sent to the database with the purpose of updating, modifying, querying, or deleting information.
query operator	An iterator or a query tree node, which is a basic unit for the execution of a query. Execution of a query can be split into one or more query operators. Common query operators include scan, join, and aggregation.
durability	One of the ACID properties of a database transaction. After a transaction is complete, the changes made by the transaction to the database are permanently stored in the database and will not be rolled back.
stored procedure	A collection of SQL statements compiled and stored on a server in a large database system that can be executed using an interface (specifying the procedure name and parameters if any) to perform a specific operation.
operating system	An operating system (OS) is loaded by a boot program to a computer to manage other programs in the computer. Other programs are called applications or application programs.
Blob	Acronym for binary large object, a collection of binary data stored in a database. Blobs are typically videos, audio or other multimedia objects.
segment	A segment is a set of extents in a database. The smallest space scope of a database is an extent, which consists of data blocks. One or more segments comprise a tablespace.
F – J	
failover	Automatic switchover from a faulty primary node to its standby node. Reversely, automatic switchback from the standby node to the primary node is called failback.
FDW	Acronym for foreign data wrapper. It is an SQL interface provided by openGauss. It is used to access big data objects stored in remote data so that DBAs can integrate data from unrelated data sources and store them in public schema in the database.

Terms	Description
freeze	An operation automatically performed by the AutoVacuum Worker process when transaction IDs are exhausted. openGauss records transaction IDs in the row heading. When a transaction reads a row, the transaction ID in the row heading and the actual transaction ID are compared to determine whether this row is explicit. Transaction IDs are integers containing no symbols. If exhausted, transaction IDs are re-calculated outside of the integer range, causing the explicit rows to become implicit. To prevent such a problem, the freeze operation marks a transaction ID as a special ID. Rows marked with these special transaction IDs are explicit to all transactions.
GDB	Acronym for GNU debugger which is used to monitor the internal situation of a running program or rewind a crashed program to see what happened. GDB can perform the following operations (strengthening PDK functions) to detect bugs: <ul style="list-style-type: none"> • Starts a program and specifies anything that might affect its behavior. • Stops a program under a specific condition. • Checks what happens when a program stops. • Modifies the program content to rectify the fault and proceeds with the next one.
GIN index	Acronym for generalized inverted index. It is used for handling cases where the items to be indexed are composite values, and the queries to be handled by the index need to search for element values that appear within the composite items.
GNU	The GNU Project was publicly announced on September 27, 1983 by Richard Stallman, aiming at building an OS composed wholly of free software. GNU is a recursive acronym for "GNU's Not Unix!". Stallman announced that GNU should be pronounced as Guh-NOO. Unix is a widely used commercial OS. GNU's design is Unix-like, but GNU differs from Unix by being free software.
gsql	openGauss interaction terminal. It enables you to interactively enter and issue queries to openGauss, and view the query results. Queries can also be entered from files. In addition, gsql supports many meta commands and shell-like commands, allowing you to conveniently compile scripts and automate tasks.
GUC	Acronym for grand unified configuration which includes parameters for running databases, and the values of which affect database system behavior.
HA	Acronym for high availability which helps to minimize the duration of service interruptions caused by routine maintenance (planned) or sudden system breakdowns (unplanned), improving the system and application usability.

Terms	Description
HBA	Acronym for host-based authentication which allows hosts to authenticate on behalf of all or some of that particular host's users. Those accounts can be all of the accounts on a system or a subset designated by the Match directive. This type of authentication can be useful for managing computing and other fairly homogenous pools of machines. In all, three files on the server and one file on the client must be modified to prepare for host-based authentication.
IV	Acronym for initialization vector. An IV is a block used for random encryption in many encryption modes. Therefore, different ciphertexts may be generated by using the same plaintext and key.
server	A combination of hardware and software designed for providing clients with services. This word alone refers to a computer running a server OS, or software or dedicated hardware providing services.
isolation	One of the ACID properties of a database transaction. Isolation means that operations and data used in a transaction are isolated from those in other concurrent transactions. Concurrent transactions are independent of each other.
relational database	A database that conforms to the relational model. It processes data using mathematical concepts and methods such as the set algebra.
archive thread	A thread started when the archive function is enabled on a database. The thread is used to archive database logs to a specified path.
failover	Automatic substitution of a functionally equivalent system component for a failed one. The system component can be a processor, server, network, or database.
environment variable	Environment variables are used to define part of the environment in which a process runs. For example, it can be used to define a home directory, command search path, terminal being used, or the current time zone.
checkpoint	A mechanism that stores data at a certain time in the database memory to disks. openGauss periodically stores the data of committed transactions and data of uncommitted transactions to disks. The data and redo logs can be used for database restoration if a database restarts or breaks down.
encryption	A function hiding information content during data transmission to prevent the unauthorized use of the information.
node	Database nodes (or nodes) are physical and virtual servers that make up the openGauss database environment.

Terms	Description
error correction	A technique that automatically detects and corrects errors in software and data flows to improve system stability and reliability.
process	An instance of a computer program that is being executed. A process consists of one or more threads. A process cannot use a thread occupied by another process.
PITR	Acronym for point-in-time recovery, a backup and restoration feature of openGauss. Data can be restored to a specified point in time if backup data and WAL logs are normal.
record	In a relational database, a record corresponds to data in each row of a table.
K – O	
KMC	Acronym for key management component.
KMS	Acronym for key management service.
KSF	Acronym for key store file.
logical replication	Data synchronization mode between primary and standby database nodes or between two databases. Different from physical replication which replays physical logs, logical replication transfers logical logs between two databases or synchronizes data through SQL statements in logical logs.
logical log	Logs recording database changes made through SQL statements. Generally, the changes are logged at the row level. Logical logs are different from physical logs that record changes of physical pages.
logical decoding	Logic decoding is a process of extracting all permanent changes in database tables into a clear and easy-to-understand format by decoding Xlogs.
logical replication slot	In a logical replication process, logic replication slots are used to prevent Xlogs from being reclaimed by the system or VACUUM . A logical replication slot in openGauss is an object that records logical decoding positions. It can be created, deleted, read, and pushed by invoking SQL functions.
MVCC	Acronym for multi-version concurrency control. It is a protocol that allows a tuple to have multiple versions, on which different query operations can be performed. One advantage is that read and write operations do not conflict.
NameNode	NameNode is the centerpiece of an HDFS, managing the namespace of the file system and client access to files.
OM	Acronym for operations management. It provides management interfaces and tools for routine maintenance and configuration management of the database.

Terms	Description
client	A computer or program that connects to or requests the services of another computer or program.
free space management	A mechanism for managing free space in a table. This mechanism enables the database system to record free space in each table and establish an easy-to-find data structure, accelerating operations (such as INSERT) performed on the free space.
junk tuple	A tuple that is deleted using the DELETE and UPDATE statements. When deleting a tuple, openGauss only marks the tuples that are to be cleared. The VACUUM thread will then periodically clear these junk tuples.
column	An equivalent concept of field. A database table consists of one or more columns.
logical node	Multiple logical nodes can be installed on the same physical node. A logical node is a database instance.
schema	Collection of database objects, including logical structures, such as tables, views, sequences, stored procedures, synonyms, indexes, and database links.
schema file	An SQL file that determines the database structure.
P – T	
page	Smallest memory unit for row storage in the relational object structure in openGauss. The default size of a page is 8 KB.
Paxos	Distributed consistency protocol.
PostgreSQL	An open-source relational DBMS developed by volunteers all over the world. PostgreSQL is not controlled by any companies or individuals. Its source code can be used for free.
postmaster	A thread started when the database service is started. It listens on connection requests from other nodes in the database or from clients. After receiving and accepting a connection request from the standby node, the primary node creates a WAL sender thread to interact with the standby node.
RHEL	Acronym for Red Hat Enterprise Linux.
redo log	A log that records operations on the database. Redo logs contain the information required for performing these operations again. If a database is faulty, redo logs can be used to restore the database to its pre-fault state.
RK	Acronym for root key.

Terms	Description
SCTP	Acronym for stream control transmission protocol. It is a transport-layer protocol defined by Internet Engineering Task Force (IETF) in 2000. The protocol ensures the reliability of datagram transport based on unreliable service transmission protocols by transferring SCN narrowband signaling over IP network.
savepoint	A savepoint marks the end of a sub-transaction (also known as a nested transaction) in a relational DBMS. The process of a long transaction can be divided into several parts. After a part is successfully executed, a savepoint will be created. If later execution fails, the transaction will be rolled back to the savepoint instead of being totally rolled back. This is helpful for recovering database applications from complicated errors. If an error occurs in a multi-statement transaction, the application can be restored by rolling back to the savepoint without terminating the entire transaction.
session	If a database receives a connection request from an application, a task is created for the connection. Sessions are managed by the session manager. They execute initial tasks and perform all user operations.
SMP	Acronym for symmetric multiprocessing. A group of processors (multiple CPUs) is integrated on a computer. These CPUs share the memory subsystem and bus structure. The OS must support multitasking and multithreading to ensure an SMP system achieves high performance. In databases, SMP means to concurrently execute queries using the multi-thread technology, efficiently using all CPU resources and improving query performance.
SQL	Acronym for structured query language. A standard database query language, consisting of data definition language (DDL), data manipulation language (DML), and data control language (DCL).
SSL	Acronym for secure sockets layer. It is a network security protocol introduced by Netscape. It is based on the TCP/IP and uses public key technology. SSL supports a wide range of networks and provides three basic security services, all of which use the public key technology. SSL ensures the security of service communication through the network by establishing a secure connection between a client and a server and then being able to securely send any data through this connection.
oversubscription ratio	The ratio of downlink bandwidth to uplink bandwidth of a switch. A high oversubscription ratio indicates a highly oversubscribed traffic environment and severe packet loss.
Table Access Method	The table access method layer decouples the execution engine from the storage engine to implement the pluggable capability of the storage engine.

Terms	Description
TCP	Acronym for transmission control protocol. It splits data into packets which are sent through the Internet protocol (IP), and checks and reassembles packets received through IP to obtain original information. TCP is a connection-oriented, reliable protocol that ensures information correctness in transmission.
trace	A specialized use of logging to record information about the way a program is executed. Programmers use the information for debugging. System administrators and technical support personnel can diagnose common problems by using software monitoring tools and based on this information.
strong consistency	A query cannot see any instantaneous intermediate state of a transaction.
full backup	Backup of the whole database.
full synchronization	A data synchronization mechanism specified in the openGauss primary/standby solution, which is used to synchronize all data from the primary node to a standby node.
log file	A file containing a record of to activities made in a computer.
transaction	A logical unit of work performed within a database management system against a database. A transaction consists of a limited database operation sequence, and must have ACID features.
data	A representation of facts or directives for manual or automatic communication, explanation, or processing. Data includes constants, variables, arrays, and strings.
data partition	The action of dividing a table into parts (partitions) whose data does not overlap within a database instance. Tables can be partitioned by range, where the target storage location is mapped based on the range of the values in the column that is specified in the tuple.
database	A collection of data that is stored together and can be accessed, managed, and updated. Data in a view in the database can be classified into the following types: numerals, full text, digits, and images.
database instance	A database instance consists of a process in openGauss and files controlled by the process. openGauss allows multiple database instances to be installed on one physical node. A database instance is also called a logical node.

Terms	Description
database primary/standby solution	openGauss provides a highly reliable HA solution. In this solution, each openGauss is called a primary or standby node. At the same time, only one openGauss is identified as the primary node. When the primary/standby system is deployed for the first time, the primary node performs full synchronization on the standby node, and then performs incremental synchronization on the standby node. When the primary/standby system is running, the primary node can receive data read and write operation requests and the standby node only synchronize logs.
database file	A binary file that stores user data and the data inside the database system.
data dictionary	A read-only collection of database tables and views containing reference information about the database. The information includes database design information, stored procedure information, user rights, user statistics, database process information, database increase statistics, and database performance statistics.
deadlock	A situation where different transactions are unable to proceed, because each holds a lock that the other needs.
index	An ordered data structure in DBMS to help quickly query and update data in database tables.
statistics	Information that is automatically collected by databases, including table-level information (number of tuples and number of pages) and column-level information (column value range distribution histogram). Statistics in databases are used to estimate the cost of query plans to find a plan with the lowest cost.
stop word	In computing, stop words are words which are filtered out before or after processing of natural language data (text), saving storage space and improving search efficiency.
U - Z	
Ustore	Alias for the in-place update storage engine which solves the problems of space expansion and large tuples of the Append update storage engine. The design of efficient rollback segments is the basis of the in-place update storage engine.
Undo Record	An undo record can be inserted, queried, and organized. It connects to the ustore through the northbound interface and connects to the buffer pool through the southbound interface.
Undo Space	Manages physical resources of undo records, including adding and deleting undo files.
Undo Zone	Be bound to service threads and manages the undo logical resources of each service thread.

Terms	Description
TransactionSlot	Records undo records by transaction granularity for transaction rollback and undo record recycling.
TIMECAPSULE	Flashback keyword. After the flashback technology is used, it takes only seconds to restore the submitted data before the database is modified. The restoration time is irrelevant to the database size.
RECYCLE BIN	After the recycle bin function is enabled, DROP TABLE can move a table and its sub-objects to the recycle bin.
PURGE	Clears objects in the recycle bin.
VACUUM	A thread that is periodically started by a database to remove junk tuples. Multiple VACUUM threads can be started concurrently by setting a parameter.
verbose	A verbose option specifies the information to be displayed.
WAL	Acronym for write-ahead logging, a standard method for logging a transaction. Corresponding logs must be written into a permanent device before a data file (carrier for a table and index) is modified.
WAL receiver	A thread created by a standby node during database replication. The thread is used to receive data and commands from the primary node and to tell the primary node that the data and commands have been acknowledged. Only one WAL receiver thread can run on one standby node.
WAL sender	A thread created on the primary node when the primary node has received a connection request from a standby node during database replication. This thread is used to send data and commands to the standby node and to receive responses from the standby node. Multiple WAL sender threads may run on one primary node. Each WAL sender thread corresponds to a connection request initiated by a standby node.
WAL writer	A thread for writing redo logs that are created when a database is started. This thread is used to write logs in the memory to a permanent device, such as a disk.
Xlog	A transaction log. A logical node can have only one .xlog file.
xDR	The x detail record is a general term that refers to call detail records (CDRs), user flow data records (UFDRs), transaction detail records (TDRs), and statistics detail records (SDRs) on the user and signaling planes.
physical node	A physical machine.

Terms	Description
system catalog	A system catalog stores meta information about a database, including user tables, indexes, columns, functions, and data types.
pushdown	In openGauss, a primary database node can send a query plan to multiple DNs for parallel execution. This behavior is called pushdown. It achieves better query performance than extracting data to a primary database node for query.
compression	Data compression, source coding, or bit-rate reduction involves encoding information that uses fewer bits than the original representation. Compression can be either lossy or lossless. Lossless compression reduces bits by identifying and eliminating statistical redundancy. No information is lost in lossless compression. Lossy compression reduces bits by identifying and removing less noticeable information. The process of reducing the size of a data file is commonly referred as data compression, although its formal name is source coding (coding done at the source of the data, before it is stored or transmitted).
consistency	One of the ACID properties of a database transaction. Data in the database must comply with integrity constraints.
metadata	Data that provides information about other data. Metadata describes the source, size, format, or other characteristics of data. In the data field, metadata helps to explain the content of a data warehouse.
atomicity	One of the ACID properties of a database transaction. A transaction is composed of an indivisible unit of work. Operations performed in a transaction must be all finished or have not been performed. If an error occurs during transaction execution, the transaction is rolled back to the state when it was not performed.
dirty page	A page that has been modified, where the changes are not yet written to a permanent device.
incremental backup	Incremental backup only saves data changed since the last valid backup.
incremental synchronization	A data synchronization mechanism in the openGauss primary/standby solution, which is used to synchronize inconsistent data from the primary node to a standby node.
primary node	A node that allows read and write operations and works with all standby nodes in the openGauss primary/standby system. At the same time, only one node in the primary/standby system is identified as the primary node.
subject term	A standardized word or phrase that describes the subject of an article.

Terms	Description
dump file	A specific type of trace file. A dump is typically a one-time output of diagnostic data in response to an event, whereas a trace tends to be continuous output of diagnostic data.
minimum restoration point	A method used by openGauss to ensure data consistency. During startup, openGauss checks consistency between the latest WAL logs and the minimum restoration point. If the record location of the minimum restoration point is greater than that of the latest WAL logs, the database fails to start.