

openGauss
3.0.0

Release Notes

Issue	01
Date	2022-03-31



Copyright © Huawei Technologies Co., Ltd. 2022. 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 User Notice..... 1

2 Versions..... 2

3 Features..... 5

4 Important Notes..... 8

5 Known Issues..... 9

6 Common Vulnerabilities and Exposures (CVEs).....10

7 Resolved Issues..... 11

8 Source Code..... 12

9 Contribution..... 13

10 Acknowledgement..... 14

1 User Notice

openGauss is an open-source, secure, and reliable relational OLTP database with ultimate performance. It is released with the Mulan PSL v2 protocol, allowing users to copy, use, modify, and distribute the source code.

The version number of openGauss is named in *X.Y.Z* format. *X* indicates the version for architecture changes, *Y* indicates the version released every year, and *Z* indicates the patch version. Generally, a *Y* version is released every year. A new *X* version is released for major architecture or feature changes. The preliminary openGauss lifecycle plan is 3.5 years.

2 Versions

openGauss 3.0.0 is the second release version of openGauss, and its lifecycle is 3.5 years. This release contains two database server installation packages: enterprise edition and lite edition.

This document applies only to the enterprise edition.

The enterprise edition is compatible with the earlier versions. Main functions are as follows:

- Inherited functions:
 - SQL standard syntax, UPSERT, data type, XML type, table, temporary table, global temporary table, foreign table, view, materialized view, index, foreign key, generalized inverted index (GIN), sequence, function, trigger, ROWNUM, MEDIAN aggregate function, JSONB data type, GB18030 character set and UPSERT subquery.
 - Stored procedure, commit/rollback in stored procedure, omission of parameter parentheses () from the stored procedure or function calling, stored procedure debugging, and autonomous transaction.
 - Security features such as authentication, permission management, network communication security, database audit, fully encrypted database, dynamic masking, Chinese national cryptographic algorithm, tamper-proof ledger database, built-in role and permission management, and transparent encryption.
 - HA functions such as primary/standby two-node cluster, cascaded standby node, logical replication, ultimate RTO, standby node scale-out, and Paxos-based distributed consensus framework (DCF).
 - Range partitioning, global partitioned indexes, list partitioning, hash partitioning, and interval partitioning (automatic partitioning based on range partitioning).
 - Full physical backup, logical backup, standby node backup, incremental backup and restoration, and point-in-time recovery (PITR).
 - Memory-optimized table (MOT), NUMA-aware high-performance capability, and parallel query.
 - AI capabilities: parameter self-tuning, slow SQL discovery, Predictor (prediction of AI query time), Anomaly-detection (database metric collection, forecast, and exception detection), and DeepSQL (in-database AI algorithms).

- Storage features such as delayed standby node replay, logical replication on standby nodes, Xlog archiving on standby nodes, I/O write amplification optimization on standby nodes, gray upgrade, hash index, unique primary key constraint for column-store tables, Ustore storage engine, and segment-page storage.
- Database running metrics in WDRs, intelligent index recommendation enhancement, and automatic eviction of unique SQL statements.
- Load balancing and read/write isolation on the JDBC client. The gsql client supports automatic completion of the readline command. It supports dblink, Ubuntu, CMake script compilation, container-based deployment, IPv6 protocol, and postgis plug-ins.
- New functions:
 - Row-store execution to vectorized execution
 - Delay of entering the maximum availability mode
 - Parallel logical decoding
 - Cluster Manager (CM)
 - global syscache
 - Publication-Subscription
 - Foreign key lock enhancement
 - Row-store table compression
 - Open-source Data Studio
 - MySQL to openGauss migration tool Chameleon
 - Using ShardingSphere to build a distributed database
 - Deploying a distributed database using Kubernetes
 - Supporting ANY permission management
 - DBMind componentization
 - XGBoost, multiclass, and PCA supported by in-database AI algorithms
- Fixed defects:
 - **I4VUXG**: Fixed the data loss issue of unlogged tables.
 - **I4SF5P**: Fixed the core dump issue occurred by running create extension dblink after the database is compiled and installed in the release version, and the dblink module is compiled and installed.
 - **I4S74D**: Fixed the issue of failing to insert data (5/5) into a row-store compressed table using Jmeter when the data volume is greater than 1 GB. The compression type is set to 2.
 - **I4N81J**: Fixed the issue of failing to synchronize the UPDATE and DELETE operations to subscribers.
 - **I4YPJQ**: Fixed the issue of failing to insert varchar constants into MOTs using JDBC.
 - **I4PF6G**: Fixed the issue of TPC-C execution failure during foreign key lock enhancement and gray upgrade from 2.0.0 to 2.2.0 (not committed).
 - **I4WPD1**: Fixed the issue of failing to execute simplified installation because the **openGauss-2.1.0-CentOS-64bit.tar.bz2** file is missing in the decompressed installation package.

- **I4L268**: Fixed the issue of incorrect system catalog **pg_partition** after the partitioned table is truncated for multiple times and then the **vacuum freeze pg_partition** command is executed.
- **I3HZJN**: Fixed the issue of incorrect date format when the **copy** command is executed.
- **I4HUXD**: Fixed the issue of failing to query the JSONB type.
- **I4QDN9**: Fixed the issue of returning a value for **select 1.79E+308*2,cume_dist() over(order by 1.0E128*1.2)** out of range.
- **I4PAVO**: Fixed the issue of failing to identify the **start with connect by record** subquery.
- **I4UY9A**: Fixed the issue of failing to create the default partition during list partitioning.
- **I4W3UB**: Fixed the issue of failing to obtain the view definition when the view is created using a user-defined type and the user-defined type is renamed.
- **I4WRMX**: Fixed the issue of failing to clear data in the **statement_history** table. When the database restarts and the **enable_stmt_track** parameter is disabled, no record should be found in the **statement_history** table.
- **I4WOBH**: Fixed the issue of failing to restart the database by setting GUC parameter **pagewriter_sleep** from **360000** to **2000**.

3 Features

- **Standard SQL support**
Supports SQL-92, SQL-99, SQL:2003, and SQL:2011 standards, GBK and UTF-8 character sets, SQL standard functions and analytic functions, and stored procedures.
- **Database storage management**
Tablespaces are supported. Different tables can be stored in different locations. The enterprise edition supports storage engines such as Ustore, Astore, and MOT.
- **Primary/standby deployment**
Supports the ACID feature of transactions, single-node recovery, HA data synchronization, and HA switchover. The enterprise edition provides the CM tool to support database instance status query, primary/standby switchover, and log management.
- **Application programming interface (API)**
Supports standard JDBC 4.0 and ODBC 3.5 features.
- **Management tools**
Provides the installation and deployment tool, instance start and stop tool, backup and restoration tool, scale-out and scale-in tool, and upgrade tool.
- **Security management**
Supports SSL network connections, user permission management, password management, security auditing, and other functions, to ensure data security at the management, application, system, and network layers.
- **AI**
Supports parameter self-tuning, slow SQL discovery, single query index recommendation, virtual index, workload index recommendation, database metric collection, forecast, and exception detection. The native AI engine in the database supports more than 10 high-performance machine learning algorithms.

New Features

This section describes the openGauss 3.0.0 enterprise edition. Compared with openGauss 2.1.0, openGauss 3.0.0 has the following new features:

- Row-store execution to vectorized execution
Row-store table queries are converted into vectorized execution plans for execution, improving the execution performance of complex queries.
- Delay of entering the maximum availability mode
When the maximum availability mode is enabled on the primary node and the primary node detects that the standby node exits (for example, due to network jitter), the primary node remains in the maximum protection mode within a specified time window. After the time window expires, the primary node enters the maximum availability mode.
- Parallel logical decoding
When JDBC or pg_recvlogical is used for decoding, you can set **parallel-decode-num** to a value greater than 1 and less than or equal to 20 to enable parallel decoding. In this way, one read thread, multiple decoding threads, and one sending thread are used to perform logical decoding, significantly improving the decoding speed.
- CM
Supports customized resource monitoring and provides capabilities such as monitoring of the primary/standby database status, network communication faults, file system faults, and automatic primary/standby switchover upon faults.
- Global SysCache
Decouples the system cache from sessions and binds them to threads to reduce the memory usage together with the thread pool feature. In addition, it is used to improve the cache hit rate and ensure stable performance.
- Publication-Subscription
Supports publication-subscription, which is implemented based on logical replication, with one or more subscribers subscribing to one or more publications on a publisher node. The subscriber pulls data from the publications they subscribe to. Data across database clusters can be synchronized in real time.
- Foreign key lock enhancement
Two types of row locks are added, which are extended from share and update locks to key share, share, no key update, and update locks. A non-primary key update obtains a no key update lock, and a row lock obtained by a foreign key trigger is a key share lock. The two types of locks do not conflict with each other, thereby improving concurrency of foreign key locks.
- Row-store table compression
Supports row-store table compression. A general compression algorithm is provided to implement transparent compression of table and index data pages and maintenance of page storage locations to achieve high compression and high performance. Disk persistence is implemented using two types of files: compressed address file (with the file name extension .pca) and compressed data file (with the file name extension .pcd).
- Open-source Data Studio
Data Studio is a universal and integrated development environment for developers and database administrators. It simplifies the development and management of the openGauss database and supports the following functions:

- Connect to the openGauss database over an integrated GUI-based development environment.
- Efficiently develop SQL.
- Manage or create database objects (databases, schemas, functions, stored procedures, tables, sequences, columns, indexes, constraints, views, users, roles, and tablespaces).
- Run SQL statements or SQL scripts.
- Create and execute a stored procedure.
- Add, delete, modify, and query table data.
- Import and export table data.
- Display and export DDL data.
- Import and export connection information.
- Format SQL statements.
- View SQL execution history.
- Display the execution plan and ER diagram.
- MySQL to openGauss migration tool Chameleon
Chameleon is Python-based. It supports real-time data replication from MySQL to openGauss. The tool can replicate initial full data and incremental data in real time to migrate them from MySQL to openGauss.
- Using ShardingSphere to build a distributed database
Supports the distributed middleware ShardingSphere to provide openGauss the distributed database capability. Up to 16 Kunpeng 920 nodes can be used for networking. The sharding performance is greater than 10 million tpmC.
- Deploying a distributed database using Kubernetes
Supports quick deployment of a distributed database. Patroni is used to implement planned switchover and automatic failover in case of faults. HAProxy is used to implement read and write load balancing between the primary and standby openGauss nodes. ShardingSphere is used to implement distributed capabilities. All functions are packaged into images and one-click deployment scripts are provided.

4 Important Notes

- For details about technical specifications, see the *Technical White Paper*.
- You are advised to deploy one primary node and two standby nodes to ensure reliability and availability.

5 Known Issues

- openGauss cannot monitor file permissions and slow disks. When the file permission is abnormal, the database exits and the corresponding information is recorded in logs. On a slow disk, the response to database operations is slower than usual.
- The read-only mode of the standby node and cascaded standby node is incompatible with the ultimate RTO feature. If the ultimate RTO feature is enabled, disable the read-only mode of the standby node and cascaded standby node.
- Memory Optimized Tables (MOTs) are incompatible with the incremental checkpoint feature. If MOTs are used, disable the incremental checkpoint function.
- LLVM does not support the ARM architecture. When the MOT TPC-C is imported, an LLVM error is reported. To avoid this problem, disable the JIT function using the **enable_mot_codegen** parameter. You can configure **force_mot_pseudo_codegen= true** to reduce the impact on TPC-C test performance when the JIT function is disabled.

6 Common Vulnerabilities and Exposures (CVEs)

This is the seventh release of openGauss. It does not involve any common vulnerabilities and exposures (CVEs).

7 Resolved Issues

This is the seventh release of openGauss. It does not involve any CVEs.

8 Source Code

openGauss contains the following 12 code repositories:

- Open-source software code repository: https://gitee.com/opengauss/openGauss-third_party
- JDBC driver code repository: <https://gitee.com/opengauss/openGauss-connector-jdbc>
- ODBC driver code repository: <https://gitee.com/opengauss/openGauss-connector-odbc>
- Database server code repository: <https://gitee.com/opengauss/openGauss-server>
- OM tool code repository: <https://gitee.com/opengauss/openGauss-OM>
- CM tool code repository: <https://gitee.com/opengauss/CM>
- DCF code repository: <https://gitee.com/opengauss/DCF>
- DCC code repository: <https://gitee.com/opengauss/DCC>
- Plug-in code repository: <https://gitee.com/opengauss/Plugin>
- MySQL-to-openGauss migration tool code repository: <https://gitee.com/opengauss/openGauss-tools-chameleon>
- Prometheus-exporter code repository: <https://gitee.com/opengauss/openGauss-prometheus-exporter>
- Document repository: <https://gitee.com/opengauss/docs>

9 Contribution

Participating in Contribution

As an openGauss user, you can contribute to the openGauss community in multiple ways. For details about how to contribute to the community, see [Contribution](#). Here, some methods are listed for reference.

Special Interest Groups (SIGs)

openGauss brings together people of common interest to form different SIGs. For details about existing SIGs, see the [SIG list](#).

You are welcome to join an existing SIG or create a SIG. For details about how to create a SIG, see [Special Interest Group \(SIG\)](#).

Mail List and Issues

You are welcome to actively help users solve problems raised in the [mail list](#) and issues (including [code repository issues](#)). In addition, you can submit an issue. All these will help the openGauss community develop better.

Documents

You can contribute to the community by submitting code. We also welcome your feedback on problems and difficulties, or suggestions on improving the usability and integrity of documents, for example, problems encountered when obtaining software or documents and difficulties encountered when using the system. You are welcome to pay attention to and help us improve the documentation module of the openGauss community.

10 Acknowledgement

We sincerely thank all the members who participated in and assisted in the openGauss project. It is your hard work to make the version released successfully and provide the possibility for the better development of openGauss.