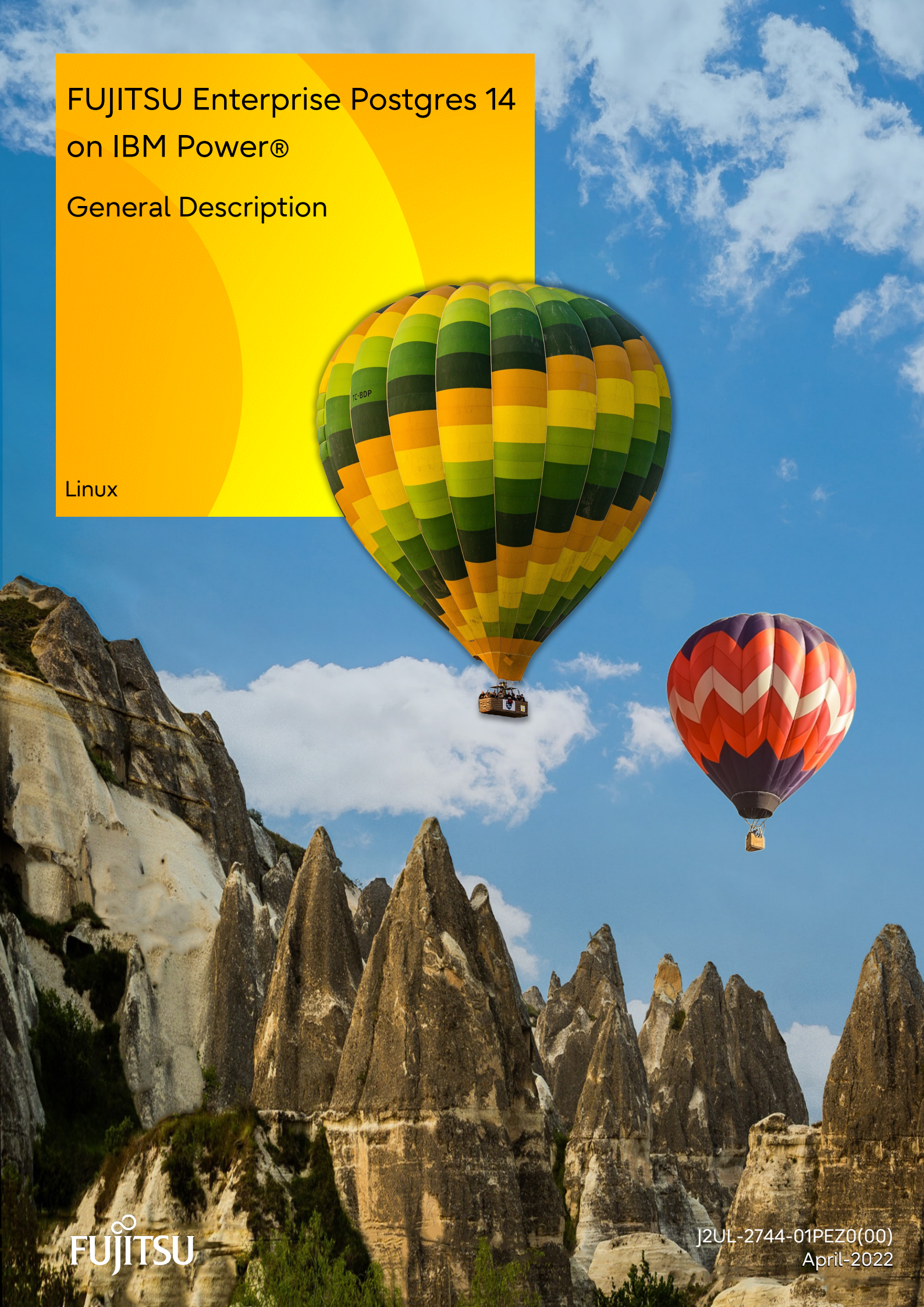
FUJITSU Enterprise Postgres 14 on IBM Power®

General Description

Linux



FUJITSU

J2UL-2744-01PEZ0(00)
April-2022

Preface

Purpose of this document

This document explains the FUJITSU Enterprise Postgres concepts to those who are to operate databases using it.

This document explains the features of FUJITSU Enterprise Postgres.

Intended readers

This document is intended for people who are:

- Considering installing FUJITSU Enterprise Postgres
- Using FUJITSU Enterprise Postgres for the first time
- Wanting to learn about the concept of FUJITSU Enterprise Postgres
- Wanting to see a functional overview of FUJITSU Enterprise Postgres

Readers of this document are also assumed to have general knowledge of:

- Computers
- Jobs
- Linux

Structure of this document

This document is structured as follows:

Chapter 1 FUJITSU Enterprise Postgres Basics

Explains the features of FUJITSU Enterprise Postgres.

Appendix A List of Features

Lists the main features provided by FUJITSU Enterprise Postgres.

Appendix B OSS Supported by FUJITSU Enterprise Postgres

Explains the OSS supported by FUJITSU Enterprise Postgres.

Appendix C Features that can be Used on Servers Other than the Database Server

Explains features that can be used on servers other than the database server.

Export restrictions

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Issue date and version

Edition 1.0: April 2022

Copyright

Contents

Chapter 1 FUJITSU Enterprise Postgres Basics

FUJITSU Enterprise Postgres maintains the operating methods, interfaces for application development and SQL compatibility of PostgreSQL, while providing expanded features for enhanced reliability and operability.

This chapter explains the functionality extended by FUJITSU Enterprise Postgres.

Refer to "Appendix A List of Features" for feature differences between editions.

Additionally, FUJITSU Enterprise Postgres supports various open source software (OSS). Refer to "Appendix B OSS Supported by FUJITSU Enterprise Postgres" for information on OSS supported by FUJITSU Enterprise Postgres.

FUJITSU Enterprise Postgres has the following features:

- Flexible database recovery
  Not only does FUJITSU Enterprise Postgres recover data to its most recent form when a failure occurs, which is essential for databases, but it can also recover to any point in time. Additionally, backup/recovery can be performed using any copy technology.

- Simple GUI-based installation and operation management
  FUJITSU Enterprise Postgres uses GUI to simplify cumbersome database operations, and allows databases to be used intuitively.

- High reliability by using database multiplexing
  Database multiplexing protects important data and enables highly reliable database operation.

- Seamless migration from Oracle databases
  FUJITSU Enterprise Postgres provides a compatibility feature with Oracle databases that localizes the correction of existing applications and allows easy migration to FUJITSU Enterprise Postgres.

- Storage Data Protection using Transparent Data Encryption
  Information can be protected from data theft by encrypting data to be stored in the database.

- Data masking for improved security
  The data masking feature changes the returned data for queries from applications, to prevent exposing actual data. This improves security for handling confidential data such as personal information.

- Audit logs for improved security
  Audit logs can be used to counter security threats such as unauthorized access and misuse of privileges for the database.

- Enhanced query plan stability
  The following features can control SQL statement query plans:

  - Optimizer hints
  - Locked statistics

  These features are used for curbing performance deterioration caused by changes in SQL statement query plans, such as with mission-critical jobs that emphasize performance stability over improved SQL statement processing performance.

- Increased aggregation performance using the in-memory feature
  The following features help speed up scans even when aggregating many rows.

  - Vertical Clustered Index (VCI)

  - In-memory data

- High-speed data load
  Data from files can be loaded at high speed into FUJITSU Enterprise Postgres tables using the high-speed data load feature.

- High availability by using Connection Manager
  With the Connection Manager features, replication operation can be continued without being aware of the connection destination of the applications.

- Memory usage reduction using Global Meta Cache
  The Global Meta Cache feature loads some of meta cahe information in shared memory. This reduces overall system memory usage.

- Memory usage reduction using Local Meta Cache Limit
  Reduce memory consumption by limiting the metacache that is kept in local memory.

1.1 Flexible Database Recovery

Threats such as data corruption due to disk failure and incorrect operations are unavoidable in systems that use databases. The ability to reliably recover corrupted databases without extensive damage to users when such problems occur is an essential requirement in database systems.

FUJITSU Enterprise Postgres provides the following recovery features that flexibly respond to this requirement:

- Media recovery, which recovers up to the most recent point in time
- Point-in-time recovery, which can recover up to a specific point in time
- Backup/recovery that can integrate with various copy technologies

Media recovery, which recovers up to the most recent point in time

When a disk failure occurs, media recovery can recover data to how it was immediately before the failure.

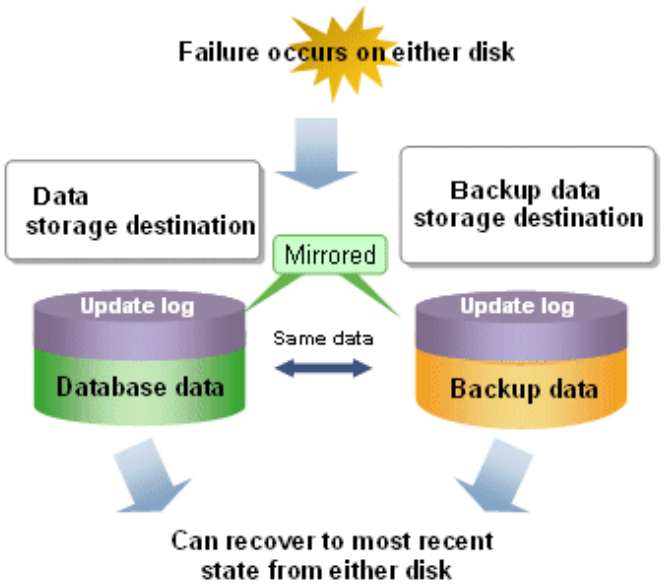
In order to recover the database, FUJITSU Enterprise Postgres accumulates a history of database update operations, such as data additions and deletions, as an update log.

FUJITSU Enterprise Postgres retains a duplicate (mirror image) of the update log after backup execution on the data storage destination and on the backup data storage destination. Therefore, the data on one disk can be used to recover to the most recent state of the database even if a disk failure has occurred on the other.

Media recovery is executed using either a GUI tool provided with FUJITSU Enterprise Postgres (WebAdmin) or server commands.

Information

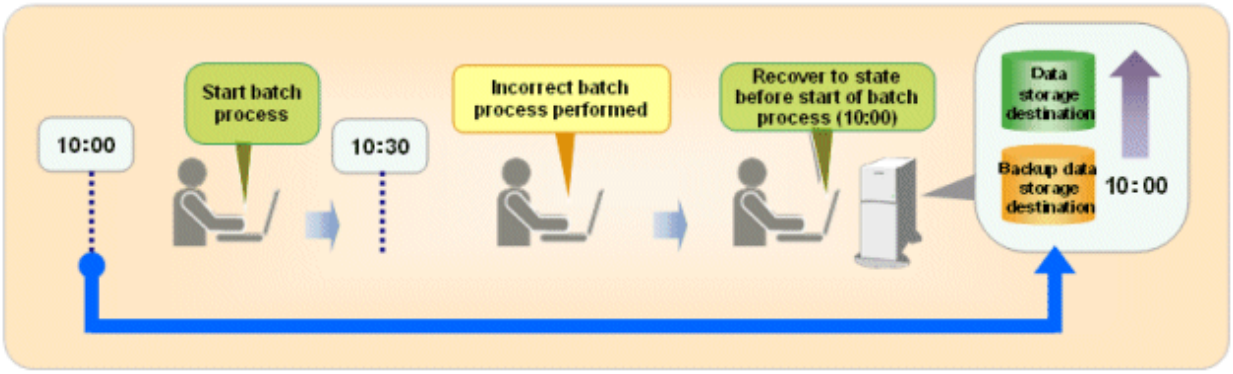
Recovery using WebAdmin requires less time and effort, since WebAdmin automatically determines the scope of the operation.



Point-in-time recovery, which can recover up to a specific point in time

Point-in-time recovery can be used to recover a database that has been updated by an incorrect operation, for example, by specifying any date and time before the incorrect operation.

Point-in-time recovery is executed using FUJITSU Enterprise Postgres server commands.



Backup/recovery that can integrate with various copy technologies

It is possible to back up to the backup data storage destination, or to any backup destination using any copy technology implemented by user commands.

See

Refer to "Backup/Recovery Using the Copy Command" in the Operation Guide for information on backup/recovery using user commands.

1.2 Simple GUI-Based Installation and Operation Management

FUJITSU Enterprise Postgres provides WebAdmin, which is a GUI tool for a range of tasks, from database installation to operation management. This allows the databases to be used simply and intuitively.

WebAdmin can be used for FUJITSU Enterprise Postgres setup, creating and monitoring a streaming replication cluster, database backups, and for recovery. Depending on the configuration, WebAdmin can be used to manage FUJITSU Enterprise Postgres instances in a single server, or instances spread across multiple servers.

- Setup

  To perform setup using WebAdmin, you must create an instance. An instance is a set of server processes that manage a database cluster (database storage area on the data storage destination disk). Instances can be created easily and with only minimal required input, because the tool automatically determines the optimal settings for operation.

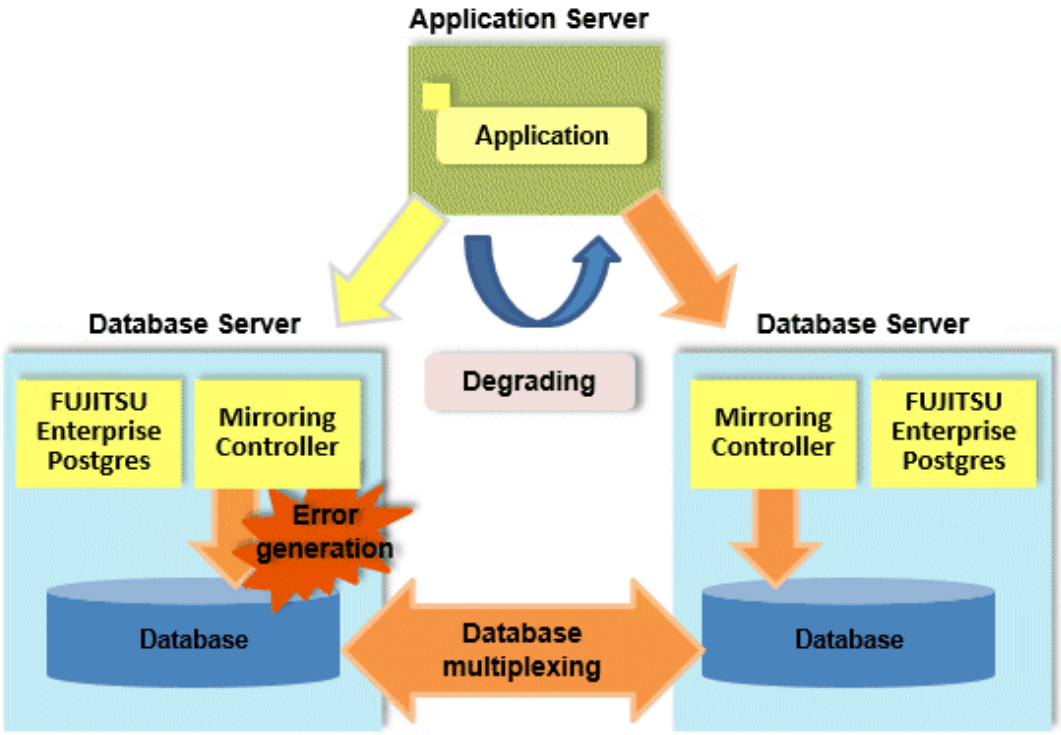
- Database backup/recovery

  Database backup and recovery can be performed using simple GUI operations.

  In particular, FUJITSU Enterprise Postgres can automatically identify and isolate the location of errors. This simplifies the recovery process and enables faster recovery.

1.3 High Reliability with Database Multiplexing

It is vital for systems that use databases to protect data from damage or loss caused by a range of factors such as hardware and software errors. Database multiplexing protects important data and enables highly reliable database operation.



FUJITSU Enterprise Postgres not only mirrors a database using the PostgreSQL streaming replication feature, but also provides simplified switchover and standby disconnection features as well as a feature to detect faults in elements that are essential for the continuity of database process, disk, network, and other database operations.

Even if a switchover is performed, the client automatically distinguishes between the primary and standby servers, so applications can be connected transparently regardless of the physical server.

The Mirroring Controller option enables the primary server (the database server used for the main jobs) to be switched automatically to the standby server if an error occurs in the former.

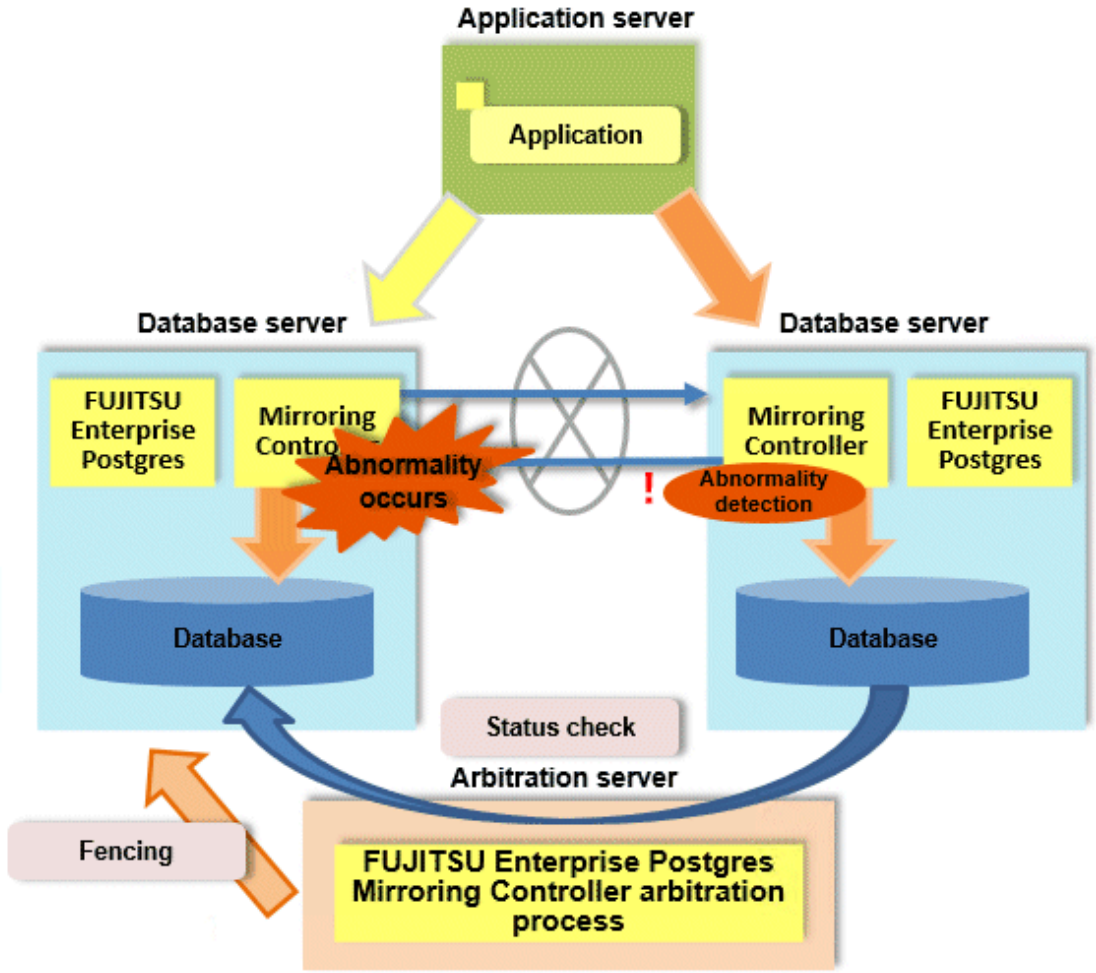
In addition, by using the data on the standby server, reference jobs such as data analysis and form output can be performed in parallel to the jobs on the primary server.

Operation using the arbitration server

Mirroring Controller may not be able to correctly determine the status of the other server if there is a network issue between database servers or a server is in an unstable state. As a result, both servers will temporarily operate as primary servers, so it may be possible to perform updates from either server.

The Server Assistant is a feature that objectively checks the status of database servers as a third party and isolates (fences) unstable servers in such cases.

In database multiplexing mode, the Server Assistant is made available by adding a new server (arbitration server) on which the Server Assistant is installed. Using an arbitration server can prevent the issue mentioned above (both servers temporarily operating as primary servers) and enables highly reliable operation.



See

Refer to "Database Multiplexing Mode" in the Cluster Operation Guide (Database Multiplexing) for information on the database multiplexing.

1.4 Seamless Migration from Oracle Databases

FUJITSU Enterprise Postgres supports Orafce, to provide compatibility with Oracle databases.

Using the compatibility feature reduces the cost of correcting existing applications and results in easy database migration.

See

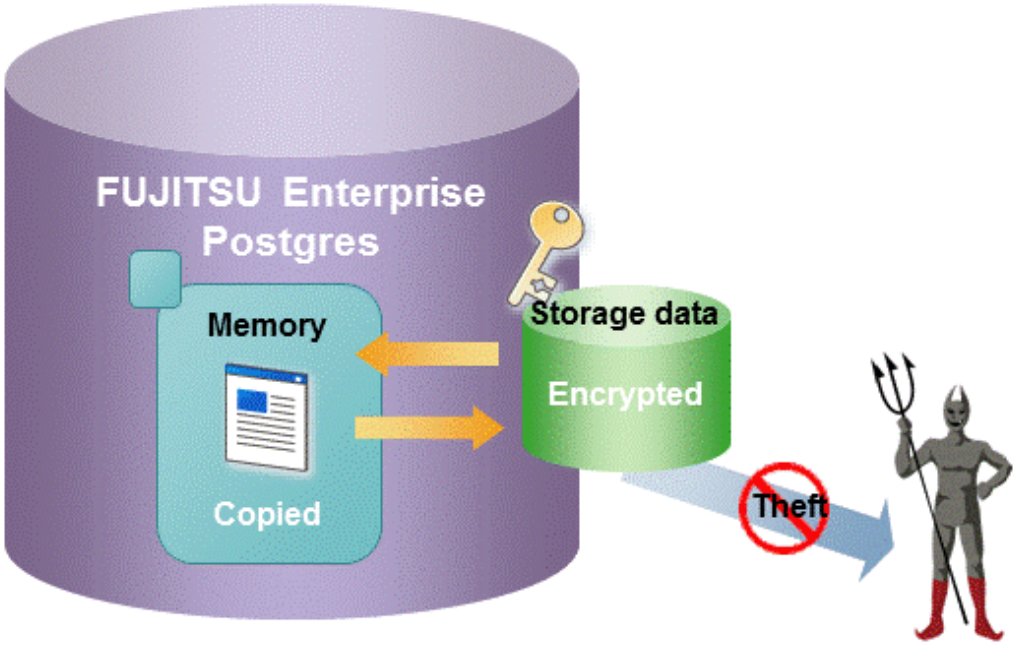
Refer to "Compatibility with Oracle Databases" in the Application Development Guide for information on compatible features.

1.5 Storage Data Protection Using Transparent Data Encryption

The encryption of data to be stored in a database is essential under the following encryption requirements of PCI DSS (Payment Card Industry Data Security Standard), the data security standard of the credit industry:

- Confidential information (such as credit card numbers) can be encrypted.
- The encryption key and data are managed as separate entities.
- The encryption key is replaced at regular intervals.

To satisfy these requirements, FUJITSU Enterprise Postgres provides a transparent data encryption feature. Note that PostgreSQL uses an encryption feature called pgcrypto, which can also be used in FUJITSU Enterprise Postgres, but requires applications to be modified. Therefore, we recommend using FUJITSU Enterprise Postgres's transparent data encryption feature.



See

Refer to "Protecting Storage Data Using Transparent Data Encryption" in the Operation Guide for information on stored data encryption.

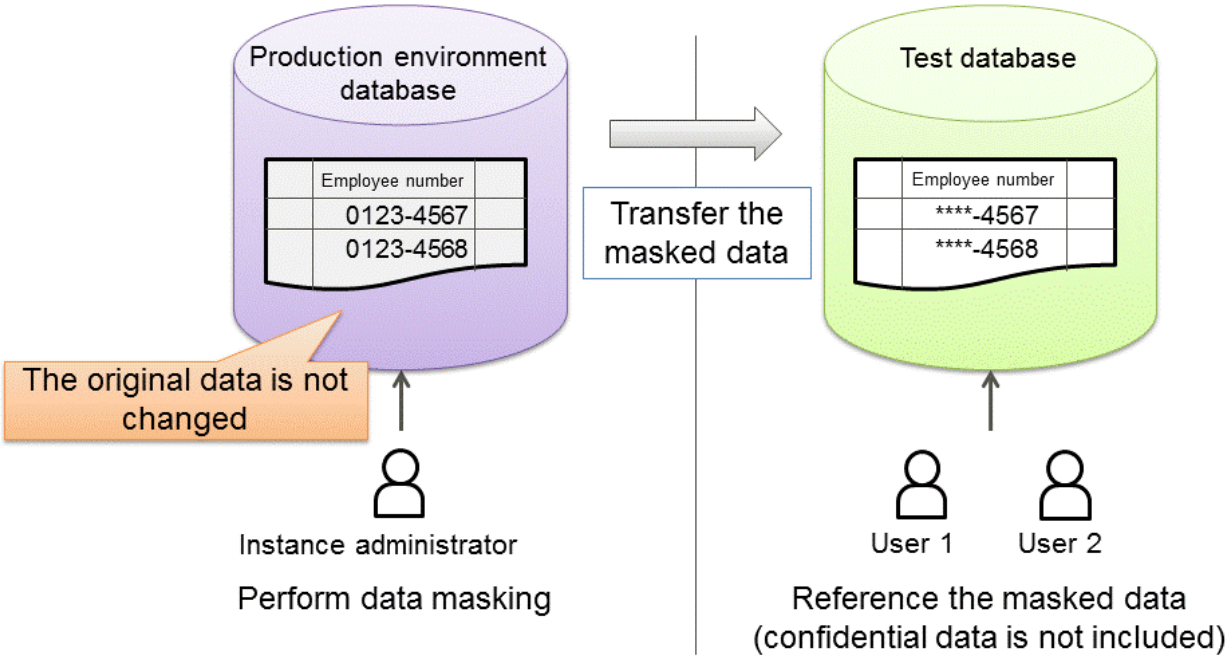
1.6 Data Masking for Improved Security

FUJITSU Enterprise Postgres provides a data masking feature that protects data to maintain security of data handled in systems.

The data masking feature changes the returned data for queries from applications and makes it available for reference without exposing the actual data.

For example, for a query of employee data, digits except the last four digits of an eight-digit employee number can be changed to "*" so that it can be used for reference.

Also, the data changed by the data masking feature can be transferred to a test database so that users who perform testing or development can reference the data. As production data should not be used in a test or development environment because of the risk of data leakage, this feature enables data that is similar to actual production data to be safely used in those environments.



See

Refer to "Data Masking" in the Operation Guide for information on data masking.

1.7 Security Enhancement Using Audit Logs

Details relating to database access can be retrieved in audit logs. The audit log feature can be used to counter security threats such as unauthorized access to the database and misuse of privileges.

In PostgreSQL, logs output as server logs can be used as audit logs by using the log output feature. There are, however, logs that cannot be analyzed properly, such as SQL runtime logs, which do not output the schema name. Additionally, because the output conditions cannot be specified in detail, log volumes can be large, which may lead to deterioration in performance.

The audit log feature of FUJITSU Enterprise Postgres enables retrieval of details relating to database access as an audit log by extending the feature to pgaudit. Additionally, audit logs can be output to a dedicated log file or server log. This enables efficient and accurate log monitoring.

See

Refer to "Audit Log Feature" in the Security Operation Guide for details.

1.8 Enhanced Query Plan Stability

FUJITSU Enterprise Postgres estimates the cost of query plans based on SQL statements and database statistical information, and selects the least expensive query plan. However, like other databases, FUJITSU Enterprise Postgres does not necessarily select the most suitable query plan. For example, it may suddenly select unsuitable query plan due to changes in the data conditions.

In mission-critical systems, stable performance is more important than improved performance, and changes in query plans case to be avoided. In this situation, the following features can stabilize query plans:

- Optimizer hints

  You can use pg_hint_plan to specify a query plan in each individual SQL statement.

- Locked statistics

  You can use pg_dbms_stats to lock statistical information per object, such as a database, schema, or table.

See

Refer to "Optimizer Hints" in the Application Development Guide for information on optimizer hints.

Refer to "Locked Statistics" in the Application Development Guide or information on locked statistics.

Note

Use the features provided when FUJITSU Enterprise Postgres is installed for optimizer hints and locked statistical information. FUJITSU Enterprise Postgres does not support other similar open-source features.

1.9 Increased Aggregation Performance Using the In-memory Feature

FUJITSU Enterprise Postgres provides the in-memory feature, which uses columnar index and memory-resident data. This reduces disk I/Os and enhances aggregation performance.

Columnar index

Many aggregation processes may require a large portion of data in a particular column. However, traditional row data structure reads unnecessary columns, resulting in inefficient use of memory and CPU cache, and slower processing. FUJITSU Enterprise Postgres provides a type of columnar index, VCI (Vertical Clustered Index). This addresses the above issues, and enhances aggregation performance.

VCI provides the following benefits:

- Minimizes impact on existing jobs, and can perform aggregation using job data in real time.
- Provided as an index, so no application modification is required.
- Stores data also on the disk, so aggregation jobs can be quickly resumed using a VCI even if a failure occurs (when an instance is restarted).
- If the amount of memory used by VCI exceeds the set value, aggregation can still continue by using VCI data on the disk.

It also provides the features below:

- Disk compression
  Compresses VCI data on the disk, minimizing required disk space. Even if disk access is required, read overhead is low.
- Parallel scan
  Enhances aggregation performance by distributing aggregation processes to multiple CPU cores and then processing them in parallel.

In-memory data

The following features keep VCI data in memory and minimize disk I/Os on each aggregation process.

- Preload feature
  Ensures stable response times by loading VCI data to memory before an application scans it after the instance is restarted.
- Stable buffer feature
  Reduces disk I/Os by suppressing VCI data eviction from memory by other job data.

Purposes of this feature

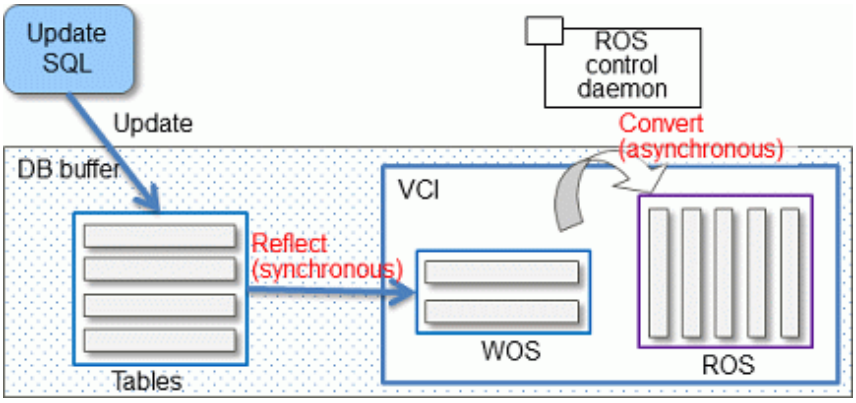
This feature has a data structure that can efficiently use the newly added resources, and aims to enhance the existing aggregation processing in normal operations to be faster than parallel scan. It shares the same purpose of enhancing aggregation performance with the parallel scan feature that is provided separately, but differs in that it speeds up nightly batch processes by utilizing available resources.

VCI architecture

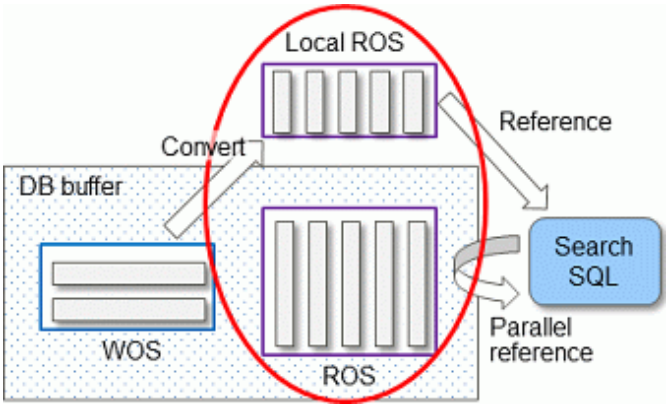
This section briefly explains VCI architecture as it contains basic terminology required, for example, when setting parameters.

Update and aggregation operations to enable real time use of job data are described.

VCI has write buffer row-based WOS (Write Optimized Store) in addition to the columnar data structure ROS (Read Optimized Store). Converting each update into a columnar index has a significant impact on the update process response times. Therefore, data is synchronously reflected to the row-based WOS when updating. After a certain amount of data is stored in WOS, the ROS control daemon asynchronously converts it to ROS. As above, the entire VCI is synchronized with the target table column, minimizing update overhead.



The same scan results can be obtained without a VCI by using WOS in conjunction with ROS. More specifically, WOS is converted to Local ROS in local memory for each aggregation process, and aggregated with ROS.



See

Refer to "Installing and Operating the In-memory Feature" in the Operation Guide for information on installation and operation of VCI.

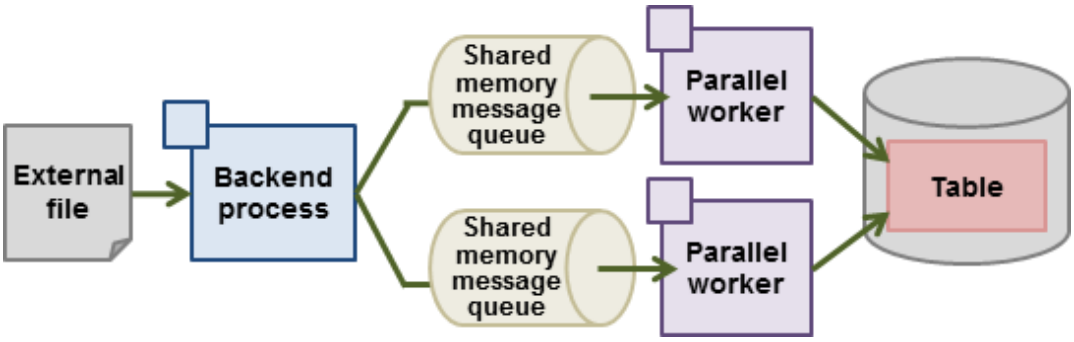
Refer to "Scan Using a Vertical Clustered Index (VCI)" in the Application Development Guide for information on scan using a VCI.

1.10 High-Speed Data Load

High-speed data load executes COPY FROM commands using multiple parallel workers. Because conversion of data from the external file to the appropriate internal format, table creation, and index creation are performed in parallel, it is possible to load large volumes of data at high speed.

Architecture of high-speed data load

High-speed data load is required for parameter setting and resource estimation, so a brief description of its architecture is provided below.



High-speed data load uses a single backend process collaborating with multiple parallel workers to perform data load in parallel. Data is exchanged between the backend process and parallel workers via shared memory message queues. The backend process distributes the loaded data of external files to multiple parallel workers. Each parallel worker then converts the data loaded from the shared memory message queue into the appropriate internal format, and inserts it into the table. If the table has indexes, their keys are extracted and inserted into the index page.

See

Refer to "High-Speed Data Load" in the Operation Guide for details.

1.11 High availability by using Connection Manager

The Connection Manager provides the following features. You can use these features to increase system availability.

Heartbeat monitoring feature

Detects kernel panics between the server running the client and the server running the instance, physical server failures, and inter-server network link downs, and notifies the client or instance. The client is notified as an error event through the SQL connection, and the instance will be notified in the form of a force collection of SQL connections with clients that are out of service.

Transparent connection support feature

When an application wants to connect to an instance of an attribute (Primary/Standby) configured with replication, it can do so without knowing which server the instance is running on.

Information

The available client drivers for Connection Manager are libpq (C language library), ECPG (embedded SQL in C), ODBC driver and JDBC driver.

See

Refer to the Connection Manager User's Guide for details.

1.12 Memory Usage Reduce with Meta cache Reduction and Limit

When executing SQL, you must refer to the definition of the table or index you want to access. These definitions are cached in the local memory of the backend process separately from the shared memory buffer of the database on shared_buffers because they are referenced each time SQL is executed. The direct definition is a tuple of system catalogs. The cache for this tuple is called "Catalog Cache". A structure that makes the definition easy to use is called a "Relations Cache". And in FUJITSU Enterprise Postgres, these two are collectively called "Meta Cache".

The meta cache will be kept indefinitely for performance reasons. In a large database, a single backend process accesses a large number of tables and so on, which results in a large meta-cache for the backend process. As a result, the sum of the local memory of the backend process may exceed the realistic memory size.

On the other hand, the feature to reduce the meta cache for the entire instance by sharing the meta cache between backend processes is the Global Meta Cache feature. The current Global Meta Cache feature only shares the catalog cache. Therefore, the metacache in Global Meta Cache now refers to the catalog cache.

What you still cannot share using the current Global Meta Cache feature and need to keep in local memory is the information (Meta cache header) and relation cache to access Global Meta Cache in shared memory. If you do not use the Global Meta Cache feature, keep the catalog and relation caches in local memory. The meta cache held in local memory is called the "Local Meta Cache". The feature to limit the size of a Local Meta Cache by removing it if it has not been accessed for a long time is the Local Meta Cache limit feature.

The Global Meta Cache feature and the Local Meta Cache limit feature can be used together to provide the strictest control over memory consumption. Of course, you can use only one or the other.

However, the Global Meta Cache feature has several percent overhead to access shared memory. The Local Meta Cache limit feature also causes the overhead of reholding the metacache because it may discard the previously held metacache. Therefore, consider using these features when your estimates do not allow for memory consumption.

1.12.1 Memory Usage Reduction Using Global Meta Cache

The Global Meta Cache feature cache the meta cache in shared memory. The meta cache on shared memory is called the Global Meta Cache (GMC).

Without this feature, the meta cache was cached in per-process memory. Therefore, there was a problem increase in memory usage in environments with large databases and large numbers of connections. The Global Meta Cache feature enables sharing of meta caches on shared memory, thereby reducing overall system memory usage.

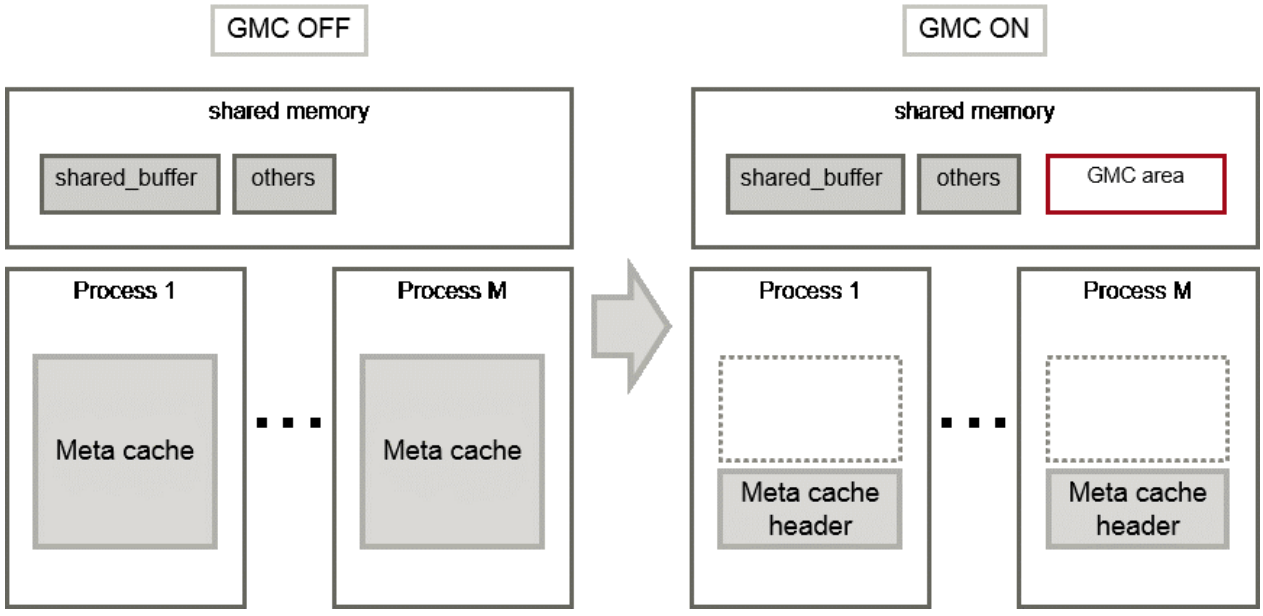
Meta cache

Processing a query involves parsing the query, creating the plan, executing the plan, and so on. PostgreSQL process accesses the system catalog to perform these steps. Once accessed, the system catalog tuples are cached in per-process memory. The direct definition is one tuple of system catalogs. Each process performs faster query processing by searching the meta cache instead of searching for the required tuples in the system catalog each time.

The meta cache usage increases in proportion to the number of tables and columns accessed. It is cached on a per-process basis, so the system's overall meta cache usage increases in proportion to the number of connections.

Architecture of Global Meta Cache feature

Describes the architecture of the Global Meta Cache feature.



When the GMC feature is on, the per-process meta cache is cached in the GMC area on shared memory. Reference to the GMC area and process-specific work information is cached in the memory of each process. PostgreSQL process searches the meta cache for each process and accesses the GMC based on the reference information. If there is no reference information in the process's memory, it searches the GMC area. If the GMC area also does not have a corresponding meta cache, it accesses the system catalog to create meta cache.

Also, sharing the meta cache does not cause any loss of data consistency. If the system catalog or table definition changes while a transaction is running, the cache deletion or creation does not affect outside of the process running the transaction. After the transaction commits, the GMC area cache is deleted or created. If other transactions are referencing the cache when GMC is tried to be deleted, the deletion is deferred until there are no more references. After a commit, a new transaction sees the new cache instead of the old one.

 See

Global Meta Cache feature is disabled by default. Refer to "Global Meta Cache" in the Operation Guide for information how to decide whether introduce it or not and usage.

1.12.2 Memory Usage Reduction Using Local Meta Cache Limit

Local Meta Cache Limit feature limits the size of a Local Meta Cache by removing it if it has not been accessed for a long time.

Of the definitions that SQL accesses, the main factors that make the Local Meta Cache bloat are tables and indexes. In addition, table column definitions are also maintained as a catalog cache.

For example, in a system where one long-lived connection is shared by various businesses, one connection (that is, backend process) will access many tables. If there are 3,000 such connections, and each connection accesses a table of 50,000, the total amount of memory consumed by the 3,000 backend processes may be a few terabytes.

In such a case, using this feature may reduce it to about several tens of gigabytes.

Architecture of Local Meta Cache Limit feature

When this feature is enabled, the caching strategy is to keep the cache as long as possible within the specified upper limit. If holding a new cache exceeds the limit, consider locality of reference and delete the cache from the one with the longest unreferenced time.

However, because the cache used by active transactions cannot be deleted, if a transaction uses a large number of caches, the cache may be held above the limit. In this case, delete the cache at the end of the transaction.

 See

Local Meta Cache limit feature is disabled by default. Refer to "Local Meta Cache Limit" in the Operation Guide for information how to decide whether introduce it or not and usage.

Appendix A List of Features

The following table lists the main features provided by FUJITSU Enterprise Postgres.

Category	Feature
Fujitsu-developed software technology	WebAdmin (Rapid setup, One-click recovery)
Improved reliability and availability	Database multiplexing
	Backup/recovery using user commands
	Connection Manager
Application development	Embedded SQL integration Java integration ODBC integration
	Features compatible with Oracle databases
Security	Storage data encryption
	Data masking
	Audit log
Performance	In-memory feature
	High-speed data load
	Global Meta Cache
	Local Meta Cache Limit
Performance tuning	Optimizer hints
	Fixed statistical information

Appendix B OSS Supported by FUJITSU Enterprise Postgres

The OSS supported by FUJITSU Enterprise Postgres is listed below.

OSS name	Version and level	Description	Reference
PostgreSQL	14.0	Database management system	PostgreSQL Documentation
orafce	3.17.0	Oracle-compatible SQL features	"Compatibility with Oracle Databases" in the Application Development Guide
Pgpool-II	4.2.6	Failover, connection pooling, load balancing, etc.	"Pgpool-II" in the Installation and Setup Guide for Server
oracle_fdw	2.4.0	Connection to the Oracle database server	"oracle_fdw" in the Installation and Setup Guide for Server
pg_statsinfo	13.0	Collection and accumulation of statistics	"pg_statsinfo" in the Installation and Setup Guide for Server
pg_hint_plan	13.1.3.7	Tuning (statistics management, query tuning)	- "pg_hint_plan" in the Installation and Setup Guide for Server - "Optimizer Hints" in the Application Development Guide
pg_dbms_stats	1.5.0		- "pg_dbms_stats" in the Installation and Setup Guide for Server - "Locked Statistics" in the Application Development Guide
pg_repack	1.4.7	Table reorganization	"pg_repack" in the Installation and Setup Guide for Server
pg_rman	1.3.13	Backup and restore management	"pg_rman" in the Installation and Setup Guide for Server
pgBadger	11.6	Log analysis	"pgBadger" in the Installation and Setup Guide for Server
pg_bigm	1.2	Full-text search (multibyte)	"pg_bigm" in the Installation and Setup Guide for Server
PostgreSQL JDBC driver	42.2.23	JDBC driver	"JDBC Driver" in the Application Development Guide
psqlODBC	13.02.0000	ODBC driver	"ODBC Driver" in the Application Development Guide

Appendix C Features that can be Used on Servers Other than the Database Server

This chapter explains the configuration and operating environment of features to be installed and used on servers other than the database server when used in conjunction with the FUJITSU Enterprise Postgres database server.

In this chapter, FUJITSU Enterprise Postgres programs are referred to as server programs.

Below are features to be installed and used on servers other than the database server:

- WebAdmin
- Server Assistant
- Pgpool-II (failover, connection pooling, and load balancing)

C.1 WebAdmin

If there is only one database server, WebAdmin is normally installed on the same server as the database (the WebAdmin program can be installed at the same time as the server program).

If there are multiple database servers, database server instances can be managed collectively if a dedicated WebAdmin server is used. In this case, the WebAdmin program is installed on the WebAdmin server, and the server program and WebAdmin program are installed on the database server.

 See

- Refer to "1.2 Simple GUI-Based Installation and Operation Management" for information on WebAdmin.
- Refer to "Determining the Preferred WebAdmin Configuration" in the Installation and Setup Guide for Server for information on the server configuration when using WebAdmin.
- Refer to "Required Operating System" in the Installation and Setup Guide for Server for information on the operating environment of WebAdmin.

C.2 Server Assistant

To use the Server Assistant, the Server Assistant program is installed on a dedicated server (arbitration server).

 See

- Refer to "Overview of Database Multiplexing Mode" in the Cluster Operation Guide (Database Multiplexing) for information on the Server Assistant and the server configuration.
- Refer to "Required Operating System" in the Installation and Setup Guide for Server Assistant for information on the operating environment of the Server Assistant.

C.3 Failover, Connection Pooling, and Load Balancing Features of Pgpool-II

Pgpool-II is software that is placed between the database server and database client to relay the connection.

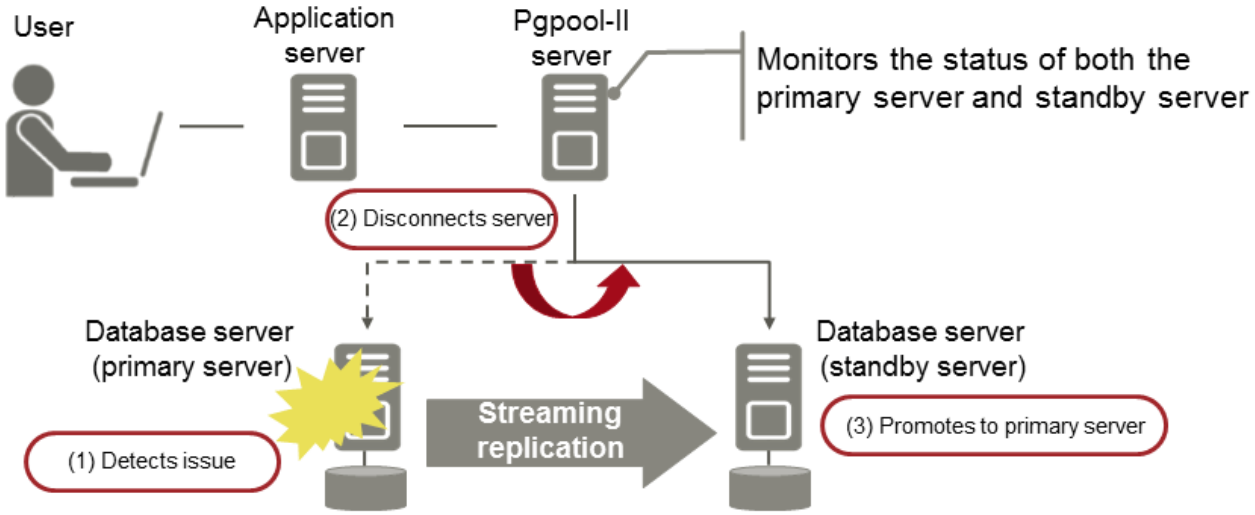
Pgpool-II provides the failover, connection pooling, and load balancing features for use during streaming replication.

Failover

In PostgreSQL, a database can be made redundant (building a high availability system) using synchronous streaming replication.

If the database server of either the primary server or standby server fails or is no longer accessible when using synchronous streaming replication, jobs will stop.

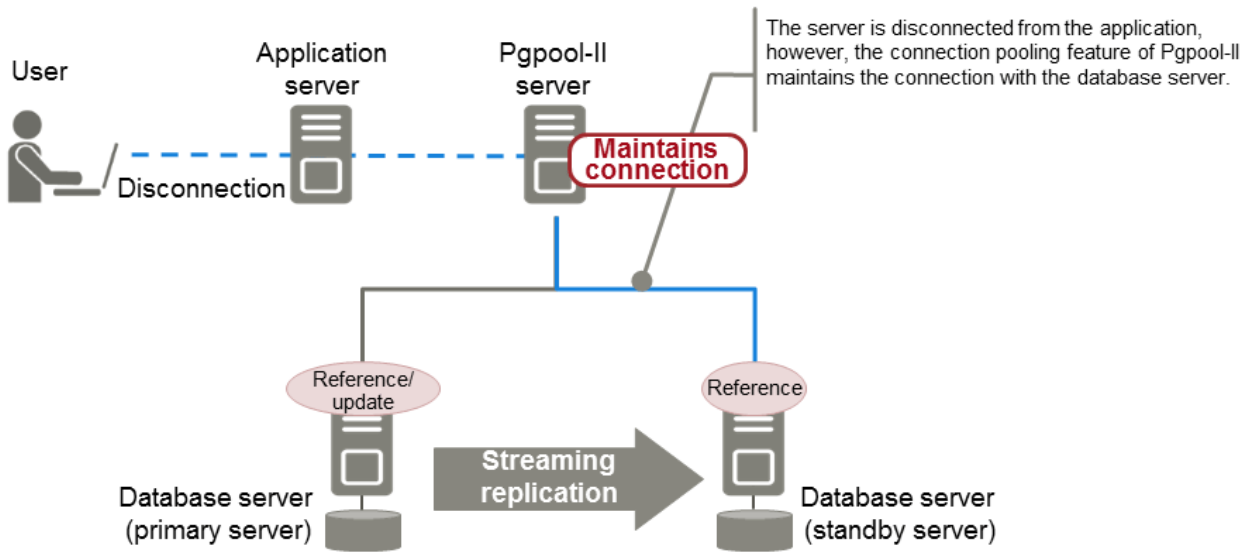
Failover monitors the status of each database and automatically disconnects the server when an error occurs. As a result, jobs can continue uninterrupted on the remaining server.



Connection pooling

This feature maintains (pools) the connection established with the database server, and reuses that connection each time a new connection with the same properties (user name, database, and protocol version) arrives.

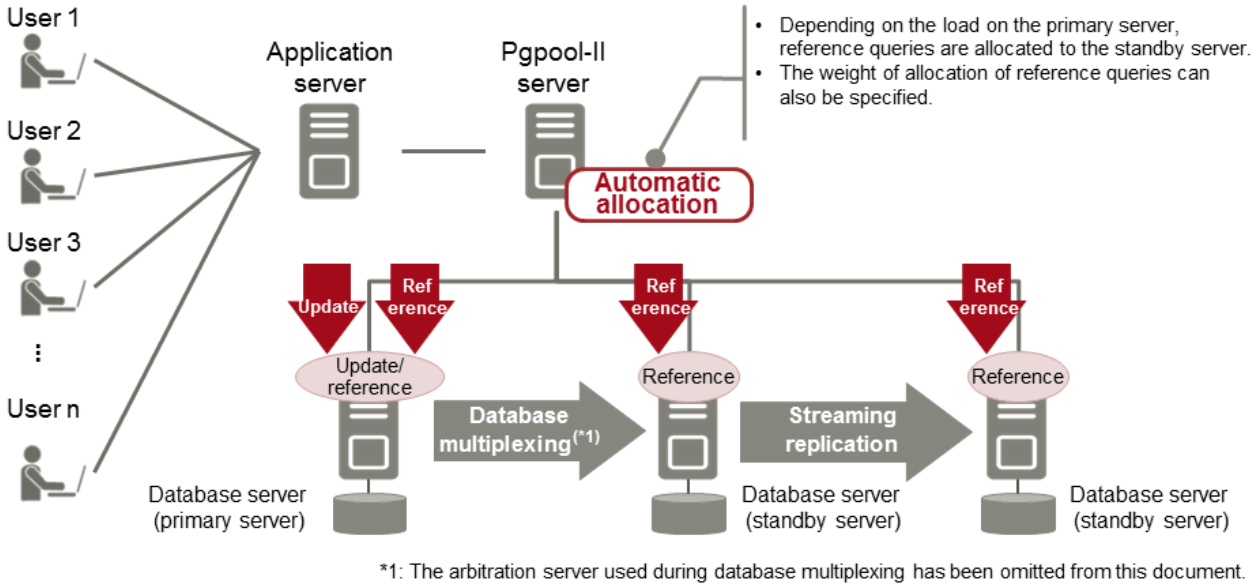
Connection pooling reduces the connection overhead for the database server, improving throughput of the whole system.



Load balancing

This feature distributes reference queries to multiple database servers, improving throughput of the whole system.

By combining load balancing with the FUJITSU Enterprise Postgres database multiplexing feature or the PostgreSQL streaming replication feature, load on the database server is reduced.



*1: The arbitration server used during database multiplexing has been omitted from this document.

See

- Refer to "System configuration when using Pgpool-II" in the Installation and Setup Guide for Server for information on the server configuration when using Pgpool-II.
- Refer to "Required Operating System" in the Installation and Setup Guide for Server for information on the operating environment of Pgpool-II.

Index