Deep Tuning of PostgreSQL 17 in Ubuntu 24.04 LTS

1.1k views

asked in PostgreSQL by superadmin

Deep Tuning of PostgreSQL 17 in Ubuntu 24.04 LTS

postgresql

ha postgresql

streaming-replication

replication

ubuntu 24.04 lts

answer

1 Answer

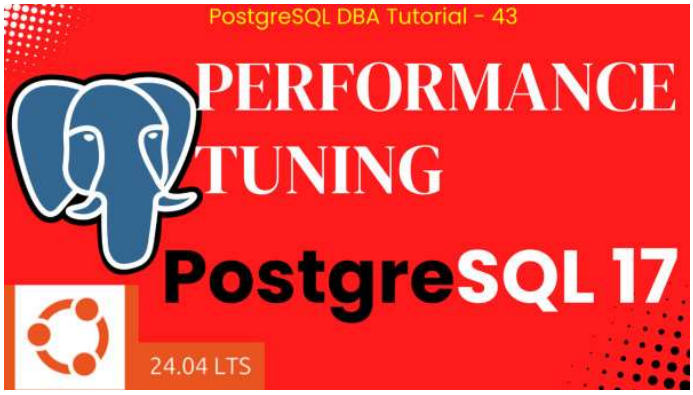
answered by superadmin

inchirags@gmail.com Chirag's PostgreSQL DBA Tutorial https://www.chirags.in

****

Deep Tuning of PostgreSQL 17 in Ubuntu 24.04 LTS

****



PostgreSQL server:

```
Server IP: 192.168.224.135
```
Copy

For deep tuning of PostgreSQL 17 on Ubuntu 24.04 LTS, the focus will be on optimizing settings for high performance and reliability based on your specific system's hardware resources, workload, and query patterns. Deep tuning involves adjusting parameters related to memory, disk I/O, caching, and query execution. Let's explore advanced tuning options for PostgreSQL.

1. Memory Tuning

Memory settings are critical for PostgreSQL performance, especially for high-load environments.

1.1 Shared Buffers

shared_buffers is the amount of memory dedicated to caching data for PostgreSQL. A good starting point is setting this to 25% of the system's RAM.

shared_buffers = 25% of your total RAM (e.g., 4GB if you have 16GB RAM)

For example:

```
shared_buffers = 4GBCopy
```

1.2 Work Mem

work_mem specifies the amount of memory allocated for internal sort operations and hash tables before using temporary disk files. This setting affects complex queries (e.g., those with ORDER BY, DISTINCT, JOIN operations).

If you expect complex queries, set this based on your available RAM and the number of concurrent connections.

```
work_mem = 16MBCopy
```

You can increase this value based on your workload.

1.3 Maintenance Work Mem

This setting controls the memory available for maintenance operations such as VACUUM, CREATE INDEX, and ALTER TABLE. For a system with larger data sets, increasing this can speed up these tasks.

```
maintenance_work_mem = 1GBCopy
```

1.4 Effective Cache Size

The effective_cache_size setting estimates how much of the operating system's file system cache PostgreSQL can use. It helps the query planner estimate whether a particular index will fit into memory. Typically, you can set this to around 50-75% of your total system memory.

```
effective_cache_size = 12GB  # If you have 16GB of RAM, 75% = 12GBCopy
```

2. Disk I/O Tuning

PostgreSQL is highly dependent on disk I/O. Optimizing how PostgreSQL reads and writes to disk is crucial for performance.

2.1 Checkpoint Settings

Checkpoints are moments when all data is flushed to disk. Optimizing these reduces disk I/O pressure during peak times.

checkpoint_segments: Controls the number of log segments between checkpoints. Increasing this reduces the frequency of checkpoints.

checkpoint_timeout: Increases the time between checkpoints.

checkpoint_completion_target: Controls how aggressively PostgreSQL writes dirty buffers to disk between checkpoints. A higher value smooths disk I/O load.

```
checkpoint_timeout = 15min        # Default is 5 minutes; 15 minutes can be a good starting point
checkpoint_completion_target = 0.9  # Default is 0.5; higher values reduce I/O spikes
checkpoint_warning = 30s
max_wal_size = 2GB                # Defines the max size of the WAL (write-ahead log)Copy
```

2.2 WAL Settings

The write-ahead log (WAL) settings control how PostgreSQL writes transactions to disk.

wal_buffers: Defines the amount of shared memory for the WAL.

wal_writer_delay: Controls how often WAL is flushed to disk. Increasing this slightly can reduce disk I/O without impacting durability.

```
wal_buffers = 16MB        # Increase from default 1MB
wal_writer_delay = 500ms   # Increase from default 200ms for better performanceCopy
```

2.3 Random Page Cost

random_page_cost tells PostgreSQL how expensive it is to read a page randomly from disk versus sequentially. If you have fast SSDs, you can lower this to improve index usage.

```
random_page_cost = 1.1     # Default is 4; 1.1 or 1.5 is recommended for SSDsCopy
```

2.4 Sequential Page Cost

seq_page_cost is the cost of reading a page sequentially. Lowering this value makes PostgreSQL prefer sequential scans.

```
seq_page_cost = 1.0Copy
```

3. Connection Tuning

Tuning connection settings is crucial for systems with many concurrent users.

3.1 Max Connections

max_connections defines the maximum number of concurrent connections to the database. Having too many connections can cause resource contention. Use a connection pooler like pgbouncer to manage connections efficiently.

```
max_connections = 200     # Based on system resources and usage patterns
```
Copy

3.2 Connection Pooling

Consider using pgbouncer or another connection pooler to handle large numbers of short-lived connections efficiently.

4. Autovacuum Tuning

The autovacuum process automatically reclaims storage by cleaning up dead rows. However, its default configuration can lead to poor performance if not tuned.

4.1 Autovacuum Settings

autovacuum_vacuum_cost_delay: Increase this to reduce the impact of autovacuum on performance.

autovacuum_max_workers: The number of autovacuum processes that can run in parallel.

autovacuum_vacuum_threshold and autovacuum_analyze_threshold: Lower these values to trigger more frequent vacuums and analyze runs.

```
autovacuum_vacuum_cost_delay = 10ms       # Lower this to decrease autovacuum impact
autovacuum_max_workers = 6                # Increase for larger systems
autovacuum_vacuum_scale_factor = 0.2      # Set lower to vacuum more frequently
autovacuum_analyze_scale_factor = 0.1     # Set lower to analyze more frequently
```
Copy

5. Query Planner Tuning

5.1 Enable JIT Compilation

PostgreSQL 17 supports Just-In-Time (JIT) compilation for faster query execution. Enabling this can improve performance for complex queries.

```
jit = on
```
Copy

5.2 Parallel Queries

PostgreSQL can run queries in parallel across multiple CPUs, which can speed up large scans and joins. Tuning the number of workers is crucial for improving query performance.

```
max_parallel_workers_per_gather = 4  # Set based on available CPU cores
max_worker_processes = 8
max_parallel_workers = 8
```
Copy

6. Logging and Monitoring

Tuning PostgreSQL's logging can help track slow queries and detect bottlenecks.

6.1 Log Slow Queries

Log any query that exceeds a certain duration to help identify performance issues.

```
log_min_duration_statement = 1000    # Logs queries taking longer than 1 second
log_checkpoints = on
log_lock_waits = on
log_temp_files = 0
```
Copy

6.2 Enable Stats Collection

Collecting statistics on database activity can provide insights into query patterns and index usage.

```
track_io_timing = on
track_activity_query_size = 2048
track_functions = all
```
Copy

7. Kernel and OS Level Tuning

7.1 Kernel Shared Memory

Adjust the shared memory settings for PostgreSQL at the OS level.

sudo sysctl -w kernel.shmmax=17179869184 # Set based on total system memory

sudo sysctl -w kernel.shmall=4194304

7.2 File Descriptors

Increase the maximum number of file descriptors allowed for PostgreSQL. You can configure this by editing /etc/security/limits.conf:

```
sudo vi /etc/security/limits.conf

* soft nofile 65536
* hard nofile 65536
```
Copy

Then, configure PostgreSQL's max_files_per_process:

```
max_files_per_process = 10000
```
Copy

8. Additional Considerations

Partitioning: For large tables, consider using table partitioning to split them into smaller, more manageable pieces.

Indexes: Regularly review index usage and ensure you're using the right types of indexes (e.g., B-tree, GIN, GiST).

Database Maintenance: Schedule regular VACUUM and REINDEX operations to optimize performance.

9. Restart PostgreSQL to Apply Changes

After making changes to the configuration files, restart PostgreSQL to apply the new settings:

```
sudo systemctl restart postgresql
```
Copy

Conclusion

Tuning PostgreSQL 17 for deep performance optimization involves careful adjustment of memory, I/O, query execution, and connection settings. These changes should be tested based on your actual workload to ensure they're providing the desired improvements. Keep monitoring query performance and system resources to ensure that PostgreSQL remains well-optimized for your environment.

Let me know if you need further details or guidance!

For any doubts and query, please write on YouTube video comments section.

Note : Flow the Process shown in video.

Please, Subscribe and like for more videos:

https://youtube.com/@chiragstutorial

Don't forget to, Follow, Like, Share &, Comment

Tutorial Link :

https://www.chirags.in/tutorials/PostgreSQL_Database

Thanks & Regards,

Chitt Ranjan Mahto "Chirag"

Note: All scripts used in this demo will be available in our website.

Link will be available in description.

comment

Search... Search

Categories

All categories
Technology (10)
Asp.net C# (1)
Asp.Net MVC (1)
Python (3)
Laravel (28)
MongoDB (10)
Oracle (1)
MySQL (6)
MS SQL Server (2)

PostgreSQL (20)

Asp.Net Core MVC (2)

Windows (1)

Windows Server 2025 (6)

MariaDB (0)

Most popular tags

laravel postgresql laravel-10 replication ha postgresql mongodb laravel-11 mongodb database mongodb tutorial ubuntu 24.04 lts streaming-replication mysql database laravel postgresql backup laravel login register logout database mysql php laravel 11 - login with otp valid for 10 minutes. user and admin registration user and admin login multiauth technlogy asp.net asp.net c# mysql master slave replication centos linux laravel sql server schedule backup autobackup postgresql django python haproxy load balancer install self sign ssl laravel 11 gaurds zabbix 7 how to install graylog on ubuntu 24.04 lts | step-by-step asp.net core mvc .net mvc network upload c# ssl integration sql server on ubuntu 22.04 lts mssql server ms sql server sql server user access in postgres mysql password change cent os linux configure replica laravel 11 socialite login with google account google login kubernetes (k8s) install nginx load balancer install install and configure .net 8.0 in ubuntu 24.04 lts php in iis php with iis php tutorial chirags php tutorials chirags php tutorial chirags tutorial laravel 11 guards mongodb sharding metabase business analytics metabase postgresql 16 to postgresql 17 postgresql migration letsencrypt mongodb crud rocky linux laravel custom captcha laravel 11 captcha laravel captcha mongo dll php.ini debian 12 nginx apache nextcloud gitea in ubuntu git gitea npm error node js mysql ndb cluster mysql cluster ssl oracle login register logout in python debian windows shell batch file bat file time stamp date time shopping cart in laravel centos rhel swap memeory rhel 5.5

Related questions

Streaming Replication of PostgreSQL 17 in Ubuntu 24.04 LTS

Installation, Configuration & Tuning of PostgreSQL 17 and pgAdmin4 in Ubuntu 24.04 LTS

How to setup streaming replication in PostgreSQL 14 step by step on Ubuntu 22.04 LTS

PostgreSQL 16 - Promote Slave to Master and Master to Slave in Ubuntu 24.04 LTS step by step process

Installation & Configuration PGPool 2 and PostgreSQL-16 with Streaming Replication in Ubuntu 24.04 LTS

Powered by Chirags Tutorial QA