General configuration and tuning recommendations for EDB Postgres Advanced Server and PostgreSQL on Windows

Last updated: March 17, 2022

These recommendations for tuning for EDB Postgres Advanced Server (EPAS) and PostgreSQL represent a starting point. Benchmarks and other measurements are required for proper tuning. This document is not intended to be an exhaustive list of configuration settings or recommendations, only the most important parameter settings that are not already the default values.

For optimal tuning we recommend engaging with EDB's Professional Services team or a qualified EDB partner.

Operating system-level recommendations:

Data drives

We recommend having separate drives/disk arrays with dedicated IOPs for the EPAS/PG Cluster. For better performance, we recommend using SSDs. One of them will be used for the PGDATA directory, and the other one will be used to store WAL (Write Ahead Log). Please note that this can easily be setup during installation, or even post-installation.

Depending on the usage of indexes and the amount of indexes needed for user-defined tables, as per workload, we recommend having separate drive(s) for indexes.

Installing EPAS/PG

EDB provides interactive installers for recent Windows versions for both EPAS and PostgreSQL.

Configuration & Authentication

max_connections

The optimal maximum number for max_connections is roughly 4 times the number of CPU cores. This formula often gives a very small number which isn't practical in most real world cases, thus we recommend using the formula: `GREATEST(4 x CPU cores, 100)`. Beyond this number, a connection pooler such as pgbouncer should be used.

Note: If you don't need that many connections as given by the above formula, then it is recommended to reduce the value of this parameter.

password_encryption

It is recommended that people should use `scram-sha-256` for this parameter.

Resource Usage

shared_buffers

This parameter has the most variance of all. Some workloads work best with very small values (such as 512MB or 1GB) even with very large database volumes. Other workloads require large values. Recommended max value for this parameter on Windows is 4GB..

work_mem

The recommended starting point for work_mem is `((Total RAM - shared_buffers)/(16 x CPU cores))`.

maintenance_work_mem

This determines the maximum amount of memory used for maintenance operations like VACUUM, CREATE INDEX, ALTER TABLE ADD FOREIGN KEY, and data-loading operations. These may increase the I/O on the database servers while performing such activities, so allocating more memory to them may lead to these operations finishing more quickly. The calculated value of `15% x (Total RAM - shared_buffers) / autovacuum_max_workers` up to 1GB is a good start.

Process title

update_process_title

Please turn this parameter to off on Windows, for performance.

Write-Ahead Log

wal_compression

When this parameter is on, the PostgreSQL server compresses a full-page image written to WAL when full_page_writes is on or during a base backup. Set this parameter to 'on'

wal_log_hints

This parameter is required in order to use pg_rewind. Set it to 'on'.

wal_buffers

This controls the amount of memory space available for back ends to place WAL data prior to sync. WAL segments are 16MB each by default, so buffering a segment is very inexpensive memory-wise. And larger buffer sizes have been observed to have a potentially very positive effect on performance in testing. Set this parameter to `64MB`.

checkpoint_timeout

Longer timeouts reduce overall WAL volume but make crash recovery take longer.
The recommended value is a minimum of `15 minutes`.

checkpoint_completion_target

This determines the amount of time in which PostgreSQL aims to complete a checkpoint. This means a checkpoint need not result in an I/O spike and instead aims to spread the writes over a certain period of time. The recommended value is `0.9`.

max_wal_size

This parameter can be difficult to set as its optimal value is a function of `checkpoint_timeout` and the maximum WAL throughput of the system. However, if set too small it can dramatically reduce performance so it is well worth taking the time to tune it. The risk of setting it too high is that you run out of disk space, but otherwise performance won't be adversely affected. Note that `max_wal_size` is a **soft** limit.

If you have lots of disk space available for WAL (hundreds of gigabytes or more), then set it to a high value. On a very high performance system, 200GB or even higher may not be unreasonable.

If your disk space is constrained, set `max_wal_size` to the highest value you can, that will avoid the risk of running out of space, leaving a little headroom. If your WAL is on a dedicated disk partition (which is always recommended), this may be 50-75% of the partition size" to be safe(r).

In order to more precisely tune `max_wal_size`, it is recommended that you monitor the `checkpoints_timed` and `checkpoints_req` values in the `pg_stat_bgwriter` view. If `max_wal_size` is too small, the ratio of requested checkpoints to timed checkpoints will rise, indicating that checkpoints are occurring because there is not enough space to hold the WAL segments required to reach the next timed checkpoint.

It's perfectly fine to have some requested checkpoints as a result of occasional activity spikes, however this should not be the norm. Increase `max_wal_size` as needed to minimize the number of requested checkpoints, without exhausting the available disk space.

archive_mode

Because changing this requires a restart, it should be set to 'on'.

archive_command

A valid archive_command is required if archive_mode is on. Until archiving is ready to be configured, a default of 'cd.' on Windows is recommended.

Query Tuning

random_page_cost

If using SSD disks, the recommended value is `1.5`.

effective_cache_size

This should be the sum of `shared_buffers` and the `free + cached memory` as per the Resource Monitor tool on Windows.

cpu_tuple_cost

Specifies the relative cost of processing each row during a query. It is currently set to `0.01`, but this is likely to be lower than optimal and should be increased to `0.03` for a more realistic costing.

Reporting and Logging

logging_collector

This parameter should be on if log_destination includes `stderr` or `csvlog`.

log_directory

If the logging_collector is on, this should be set to someplace outside of the data directory. This way, the logs are not part of base backups.

log_checkpoints

This should be set to on.

log_line_prefix

The prefix should at least contain the time, the proc id, the line number, the user and database, and the application name.
Suggested value: '`%m [%p]: u=[%u] db=[%d] app=[%a] c=[%h] s=[%c:%l] tx=[%v:%x]`'
Note: Don't forget the space at the end

log_lock_waits

Set to `on`. This parameter is essential in diagnosing slow queries.

log_statement

Set to '`ddl`'. In addition to leaving a basic audit trail, this will help determine at what time a catastrophic human error occurred, such as dropping the wrong table.

log_temp_files

Set to `0`. This will log all temporary files created, suggesting that work_mem is incorrectly tuned.

log_autovacuum_min_duration

Monitoring autovacuum activity will help in tuning it. Suggested value: `0`.

timed_statistics (EPAS)

Controls the collection of timing data for the Dynamic Runtime Instrumentation Tools Architecture (DRITA) feature. When set to on, timing data is collected. Set this parameter to on.

Autovacuum

autovacuum_max_workers

This is the number of workers that autovacuum has. Default value 3 and requires a DB restart to be updated. Please note that each table can have only one worker working on it. So increasing workers only helps in parallel and more frequent vacuuming across tables. The default value is low, therefore it is recommended to increase this value to `5`.

autovacuum_vacuum_cost_limit

To prevent excessive load on the DB due to the autovacuum, there is an I/O quota imposed by Postgres. So every read/write causes depletion of this quota and once it is exhausted the autovacuum sleeps for a fixed time. This configuration increases the quota limit, therefore increasing the amount of I/O that the vacuum can do. The default value is low, so we recommend increasing this value to `5000`.

Client Connection Defaults

idle_in_transaction_session_timeout

Sessions that remain idle in a transaction can hold locks and prevent vacuum.
Suggested value: `10 minutes`.

lc_messages

Log analyzers only understand untranslated messages. Set this to 'C'.

shared_preload_libraries

Adding pg_stat_statements is low overhead and high value. This is recommended but optional.