EDB Partner Program
GlobalConnect Technology

Swarm64 DA

EDB GlobalConnect Technology Partner Implementation Guide

EDB POWER TO POSTGRES

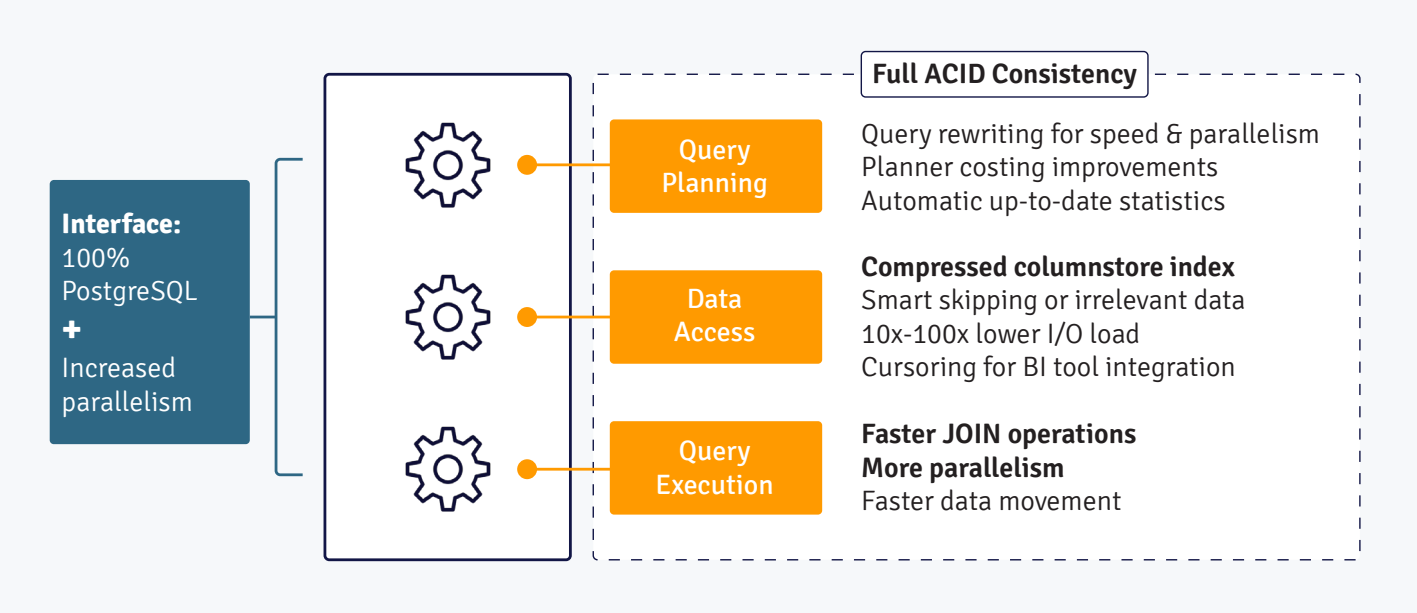
Contents

1 Partner information

Partner Name	Swarm64
Partner Product	Swarm64 DA
Web Site	www.swarm64.com
Version & Platform	5.4.0, Available platforms: Ubuntu 18.04 or 20.04, Centos 7, 8, alternatively can be deployed in a Docker container
Product Description	Swarm64 DA is an extension to PostgreSQL that accelerates PostgreSQL for analytical and hybrid OLTP/ analytical workloads
Date	March 10, 2021

2 Solution summary

Swarm64 DA is a PostgreSQL extension. Swarm64 DA extends PostgreSQL by providing a collection of tools to increase the performance of your workloads. The default PostgreSQL functionality remains available while Swarm64 DA uses various official PostgreSQL mechanisms to add functionality: modules, hooks and data access methods. Swarm64 DA accelerates PostgreSQL queries at every stage: during the query planning, when accessing the data and during query execution. The Swarm64 DA features at each stage are outlined in the following image:



For more details,read the technical overview.

3 Configuration

3.1 Prerequisites

- Tune your operating system for optimal performance by following the Swarm64 DA guidelines in the Swarm64 DA documentation
- Ensure that EDB Postgres Advanced Server is installed and running.
- For CentOS 7, you need to install the following repository:

```
sudo yum install -y epel-release centos-release-sc
```

3.2 Installation and configuration of Swarm64 DA

The following steps outlined below are for CentOS. See the Swarm64 documentation for full details on installation and configuration for CentoOS and other supported operating systems.

1. Install the Swarm64 DA repository

```
curl -s
https://packagecloud.io/install/repositories/swarm64/swarm64da/script.
rpm.sh | sudo bash
```

2. Install the corresponding Swarm64 DA package for your EDB Postgres Advanced Server installation.

   EDB Postgres Advanced Server 12

```
sudo yum install swarm64da-epas-12
```

3. Add ***swarm64da*** to ***shared_preload_libraries*** in your ***postgresql.conf*** configuration file. For example:

```
shared_preload_libraries = 'swarm64da'
```

   For other recommended settings, see the section *Recommended EPAS settings* in the Swarm64 documentation.

4. Restart the EDB Postgres Advanced Server.

5. Switch to a user with access to the database:

```
sudo su - enterprisedb
```

6. Connect to the database:

```
edb-psql -d postgres
```

7. Load the Swarm64 DA extension:

```
CREATE EXTENSION swarm64da;
```

A NOTICE is printed when executing the first SQL command through a connection to a database for which the swarm64da extension has not been created yet. This behavior is controlled by the following configuration parameter:

`swarm64da.enable_notice_not_created_extension(boolean)`

Enables or disables the reminder to create the extension. The default is on.

8. Verify that the Swarm64 DA extension is loaded correctly and look up its version number:

```
SELECT * FROM swarm64da.get_version();
```

9. Install a valid license.

```
SELECT swarm64da.load_license('<path-to-license-file-on-server>');
```

See also Install a license for Swarm64 DA in the Swarm64 DA documentation.

10. Check and follow the Swarm64 DA settings advisor and documented recommendations to tune your database system for optimal performance.

```
SELECT * FROM swarm64da.show_advice();
```

See also Settings advisor and Recommended EPAS settings in the Swarm64 DA documentation.

11. Create columnstore indexes.

    Columnstore indexes are necessary to maximize the performance benefits. Create an index for each large table covering the most common accessed columns. See Columnstore index and Using a columnstore index on a small TPC-H data set.

3.3 Integration Views

The following is a sample EDB Postgres Advanced Server configuration optimized for use with Swarm64 DA. Note this configuration is typically stored in a file separate from postgresql.conf. A reference is made in postgresql.conf to this location.

Create a directory within the PostgreSQL working directory:

```
[root@r640-2 pg12-host]# mkdir conf.d
```

Create a new file in the conf.d directory:

```
Create a new file in the conf.d directory:
```

Specify include_dir in postgresql.conf:

```
#------------------------------------------------------------------------------
# CONFIG FILE INCLUDES
#------------------------------------------------------------------------------

# These options allow settings to be loaded from files other than the default
# postgresql.conf. Note that these are directives, not variable assignments,
# so they can usefully be given more than once.

include_dir = 'conf.d'                  # include files ending in '.conf' fro
                                        # a directory, e.g., 'conf.d'
#include_if_exists = '...'              # include file only if it exists
#include = '...'                        # include file
```

3.4 Settings defined in s64_settings.conf

NOTE: Sample settings used in testing; values may be adjusted as require

# 25% of total available memory 	
shared_buffers = 96GB
	# SSDs have a much faster random access, for them a more appropiate value is 2 , if seq_page_cost is the default
	random_page_cost = 2.0

# 1% or more of total memory 	
temp_buffers = 4GB
	# Reduce it to 500 to encourage parallel plans, which for large queries are always faster
	parallel_setup_cost = 500

# 1-3% of total memory 	
work_mem = 6GB
	# Reduce it to 0.01 to encourage parallel plans, which for large queries are always faster
	parallel_tuple_cost = 0.01

# Up to 5% of total memory 	
maintenance_work_mem = 16GB
	# Lower values encourage parallel plans, which for large queries are always faster
	min_parallel_table_scan_size=0

# At least 128 when using SSDs to enable more simultaneous IO requests. 	
effective_io_concurrency = 128
	# With a dedicated server to host the database this value can be up 80-90% of the total memory available
	effective_cache_size = 288GB

# Set this to at least the number of available virtual cores in your system times the number of expected concurrent complex queries 	
# It is a global system limitation in the amount of concurrent work 	
max_worker_processes = 1024
	# A value of 2500 or higher is recommended, as with better statistics the planner will generate more optimal plans
	# Run ANALYZE after increasing this value to update the statistics inmediately
	default_statistics_target = 2500

# Set this to the number of virtual cores in the system 	
# This is the maximum degree of parallelism that a query can achieve 	
max_parallel_workers_per_gather = 52
	# When working with partitioned tables, disable jit for optimal performance with Swarm64 DA
	jit = on

# Set this to half of the number of virtual cores in the system to accelerate some maintenance operations, such as vacuuming or index building 	
max_parallel_maintenance_workers = 32
	# Must be disabled so the Shuffle node can be used by the planner
	parallel_leader_participation = off

Settings defined in s64_settings.conf CONTINUED

NOTE: Sample settings used in testing; values may be adjusted as require

```
# This is the subset of workers from max_
worker_processes that can be dedicated to
support parallel queries
# If you expect a low number of queries but
very parallel, this value can be as high as max_
worker_processes
max_parallel_workers = 1000
```

```
# Automatic vacuuming must be active to keep the
columnstore indexes up-to-date
autovacuum = on
```

```
# Set this to -1 for compatibility with the
Columnstore index
old_snapshot_threshold = -1
```

```
# Needs to be increased as the number of tables and
columnstore indexes grows
autovacuum_max_workers = 15
```

```
# Set it to 100GB or more to accelerate the data
ingestion, this slows down the crash recovery of
the database
max_wal_size = 100GB
```

```
# Reduce it to 1 second in situations where there is a
constant ingestion, so that the columnstore indexes are
constantly updated
autovacuum_naptime = '1s'
```

```
# Set it to 4GB or more to accelerate the data
ingestion
min_wal_size = 4GB
```

```
# Set it to zero so the automatic vacuuming runs
constantly with the ingest
# Otherwise it is relative to the scale of the table, which
delays the update of the indexes
autovacuum_vacuum_scale_factor = 0
```

```
# In some workloads due to wrong estimations
the plan selected by the planner can be
suboptimal
# Disabling the materialization might produce
queries with faster runtime
# In some situations this might impact
negatively.
enable_material = off
# Disable partition aware join for optimal
performance with Swarm64 DA
enable_partitionwise_join = off
```

```
# Set it to zero so the automatic analyze runs constantly
with the ingest
# Otherwise it is relative to the scale of the table, which
delays the update of the indexes
autovacuum_analyze_scale_factor = 0
```

```
# Disable partition aware aggregate for optimal
performance with Swarm64 DA
enable_partitionwise_aggregate = off
```

4 Certification environment

Certification Test Date:	March 10, 2021
OS	CentOS Linux 7 (Core)
Memory	377G
Processor	Intel® Xeon® Processor SP Family (“Skylake”)
CPU(s)	64
Core(s) per socket	16
Socket(s)	2
Storage	1 TB (SSD-based storage)
Swarm64 DA	5.4.0
EDB Advanced Server	12.6.7

5 Appendix

Evaluate Swarm64 DA using the TPC-H benchmark.

Swarm64 DA

EDB GlobalConnect Technology Partner Implementation Guide

EnterpriseDB Corporation
34 Crosby Drive
Suite 201
Bedford, MA 01730