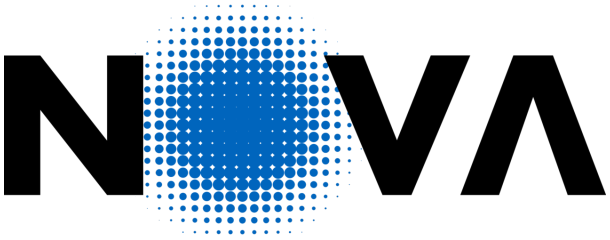

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

NOVA UNIVERSITY OF LISBON

MSC IN COMPUTER SCIENCE

PERFORMANCE COMPARISON BETWEEN DBMSS UNDER TPROC-C WORKLOADS

José Costa (62637)
Rodrigo Albuquerque (70294)
Rodrigo Silva (70567)

DATABASES SYSTEMS

JUNE 9, 2025

Contents

1 Introduction

This project aims to benchmark and compare the performance of different database systems using the TPC-C workload, a standard for evaluating OLTP (Online Transaction Processing) environments.

Automated scripts are used to run identical tests across multiple databases, each configured with similar settings to ensure a fair comparison.

Key metrics such as Transactions Per Minute (TPM) are measured under varying levels of concurrent users.

The results help identify the strengths and limitations of each database in handling transactional workloads.

In this project, the databases used were PostgreSQL, MySQL and MariaDB.

In total, we ran 54 tests:

- 4 tests scaling the number of virtual users (2 4 8 12) and warehouses (VU*5) on all PCs and databases (48 tests in total);
- 1 test with the number of virtual users set to the same as the number of threads in that PC, warehouses set to VU*5, *allwarehouse* = true on all databases on just one PC (3 test in total);
- 1 test with the number of virtual users set to the same as the number of threads in that PC, warehouses set to VU*5, with default config on all databases on just one PC (3 test in total).

2 Overview of HammerDB

HammerDB is a free, open-source tool for benchmarking the performance of relational databases [1].

It supports popular databases like Oracle, SQL Server, PostgreSQL, MySQL, and more. HammerDB uses industry-standard workloads such as TPROC-C and TPROC-H to simulate real-world database activity.

It offers both a graphical interface and command-line options, making it suitable for developers, DBAs, and system administrators to test, compare, and tune database performance.

In some cases we used HammerDB in docker containers to run the tests, which allows for easy setup and isolation of the testing environment.

In another case, we used the Windows version of HammerDB to run the tests on a Windows machine.

2.1 Overview of TPROC-C

TPROC-C is a benchmark designed to evaluate the performance of database management systems (DBMS) using a transactional workload. It simulates a typical online transaction processing (OLTP) environment, focusing on operations like inserts, updates, and deletes across multiple tables.

2.2 TPROC-C vs TPROC-H

TPROC-H is a benchmark designed for data warehousing and analytical workloads, while TPROC-C is focused on transactional processing. TPROC-H emphasizes complex queries and large data sets, whereas TPROC-C simulates real-time transactions with a focus on insert, update, and delete operations.

3 Problem & DBMS Summary

Modern applications progressively depend on robust, scalable database systems to effectively manage workloads in a transactional manner. Choosing the right Database Management System (DBMS) is critical to achieve the best possible performance, especially where concurrency is high and the workload is varied. With so many DBMSs to select from, with their own strengths, configurations, and community support, getting the right one can be problematic.

The main objective of this study is to provide a clear comparison of how three of the most popular open-source DBMSs like,PostgreSQL, MySQL, and MariaDB perform under TPROC-C workloads. This

is a close approximation of OLTP environments and thus is suitable to use in evaluating systems for high-throughput transaction processing.

A brief overview of the DBMSs under test is provided below:

- **PostgreSQL**: It is renowned for support of sophisticated query capabilities, extensibility, and strict support for data integrity features.
- **MySQL**: Extensively used in web development, valued as being easy to use and swift, with excellent ecosystem and support.
- **MariaDB**: A fork of MySQL with an emphasis on improved speed, open development, and additional storage engines.

Each DBMS was installed with identical hardware and software configurations to provide a level playing field for the tests. Experiments were designed to highlight differences in how each system handles transaction loads, concurrency, and configuration parameters. Through comparison of performance in a systematic manner across controlled tests, this study aims to guide database selection on the grounds of empirical evidence and not assumptions.

4 Benchmark Description

The benchmark used in the research is the TPROC-C workload using HammerDB. TPROC-C is designed to simulate an OLTP environment in the average case and is composed of transactions containing new orders, orders to make payment, checking orders status, delivering orders, and updating stock status. The aforementioned operations can be likened to real-world application environments.

4.1 Benchmark Goals

The main objectives of the benchmark are to:

- Compare the throughputs of the DBMSs at different levels of concurrency in terms of Transactions Per Minute, or TPM.
- Monitor the scalability of the systems as virtual users and additional warehouses get created.
- Measure consistency across repeated experiments as well as with different setups.

4.2 Test Parameters

To ensure consistency and fairness, the same configuration template was used for all tests, with the sole variations being the number of virtual users, warehouses, and the specific DBMS being tested. The significant parameters were:

- **Virtual Users (VU)**: Simulated clients making postings simultaneously. Applying the values 2, 4, 8, and 12 to perform the scaling test.
- **Warehouses**: A TPROC-C scale unit. Five times the number of virtual users.
- **Test Duration**: Each test was executed in medium one hour.
- **Ramp-Up and Cool-Down Time**: Confirmed with HammerDB setup to allow systems to reach steady state before measurement.

4.3 Execution Environment

All the testing was automated with custom scripts offering the same setup routines and execution across the systems. The environment included combinations of Windows and Linux systems, based on the setup. HammerDB was run in some cases in Docker containers to offer isolation to the test environment and ensure repeatability.

4.4 Metrics Collected

A critical measurement during the benchmarking was the Transaction Per Minute (TPM), as posted by HammerDB. The measurement is indicative of the system capability to process new orders and follow-up transactions in one minute. Secondary measures included CPU usage and memory use.

4.5 Limitations

Even though control over variables, consistency across runs, and bias to a variable as limited as possible was the focus of this testing, there are several limitations:

- Variability in network latency, and relative performance of hardware and storage may be negligible in each of the testing environments.
- Unless otherwise stated, default DBMS tuning parameters were used that are not reflective of, nor necessarily represent, the best performance that could be achieved for each database management system.
- The intent and focus were on relative performance under a specified workload, not full optimization for each system.

5 Methodology

5.1 Hardware and Software Setup

PC	**1**	**2**	**3**	**4**
OS	Windows 11	Windows 11	Linux (Unraid)	MacOS Sequoia
CPU	AMD Ryzen 5 3600	Intel i7-13700H	Intel i3-10100F	Apple M1
Cores	6	14 (6P 8E)	4	8
Threads	12	20	8	8
RAM	16GB	16GB	32GB	16GB
Disk	SSD M.2 NVMe	SSD M.2 NVMe	SSD M.2 NVMe	SSD M.2 NVMe
Read	2500 MB/s	3500 MB/s	3500 MB/s	3400 MB/s
Write	2100 MB/s	2700 MB/s	3300 MB/s	2800 MB/s
Test type	Bare metal	Docker	Docker	Docker

Table 1: Hardware used in the benchmarks

The benchmark tests were run on four different PCs that each met the same specifications for the purpose of consistent performance measurement.

- **PC 1:** All components, including HammerDB and the database systems, were installed directly on the host operating system (bare metal). This installation aimed to assess performance without the overhead associated with running in a container.
- **PCs 2, 3, and 4:** In this case, HammerDB and every DBMS were run within Docker containers. This offered a consistent method that provided isolation between services while ensuring consistency across machines.

We used as our orchestration tool Docker Compose, allowing us to apply and deploy the same setup as configured in multiple machines with minimal processing.

For MySQL and MariaDB, we created custom Dockerfiles to support the loading of external configuration files. This is because the databases enforced read-only access to the configuration file by default, unlike PostgreSQL, inside the container.

9 Bibliography

[1] Wikipedia contributors. *HammerDB — Wikipedia, The Free Encyclopedia*. [Online; accessed 25-May-2025]. 2025. URL: https://en.wikipedia.org/w/index.php?title=HammerDB&oldid=1275860580.

A TCL Scripts

A.1 MySQL TCL Script

Listing 1: TCL script for MySQL

```
#!/usr/bin/tclsh
# Set Database & Benchmark
dbset db mysql
dbset bm TPC-C

# DB configs
diset connection mysql_host 172.22.0.3
diset connection mysql_port 3306
diset connection mysql_socket "/var/run/mysqld/mysqld.sock"
diset tpcc mysql_user root
diset tpcc mysql_pass 1234
diset tpcc mysql_dbase tpcc

# Default for WH and VU
diset tpcc mysql_count_ware 50
diset tpcc mysql_num_vu 10

# Driver script options
diset tpcc mysql_timeprofile true
diset tpcc mysql_async_scale false
diset tpcc mysql_driver timed
# Ensure test is limited by time
diset tpcc mysql_total_iterations 1000000
# Timed duration
diset tpcc mysql_rampup 2
diset tpcc mysql_duration 8
# Distribute load
diset tpcc mysql_allwarehouse false

# Transactions options
tcset refreshrate 10
tcset logtotemp 1
tcset unique 1
tcset timestamps 1
# Run
foreach z {2 4 8 12} {
    set w [expr {$z * 5}]
    diset tpcc mysql_count_ware $w
    puts "Building Schema for $z TEST"
    # Delete possible previous data
    deleteschema
    vudestroy
    # Build and load
    buildschema
    vudestroy
    loadscript
    # Vuser options
    vuset delay 500
    vuset repeat 500
    vuset iterations 1
    vuset showoutput 0
    vuset logtotemp 1
    vuset unique 1
    vuset nobuff 0
    vuset timestamps 1
    puts "Starting $z VU TEST"
    tcstart
    vuset vu $z
    vucreate
    vurun
    puts "Waiting 1 for cleanup and collection"
    after 60000
    puts "Destroying VU"
    vudestroy
    tcstop
}
```

A.2 MariaDB TCL Script

Listing 2: TCL script for MariaDB

```
#!/usr/bin/tclsh
# Set Database & Benchmark
dbset db maria
dbset bm TPC-C

# DB configs
diset connection maria_host 172.22.0.4
diset connection maria_port 3306
diset connection mysql_socket "/var/run/mysqld/mysqld.sock"
diset tpcc maria_user root
diset tpcc maria_pass 1234
diset tpcc maria_dbase tpcc

# Default for WH and VU
diset tpcc maria_count_ware 50
diset tpcc maria_num_vu 10

# Driver script options
diset tpcc maria_timeprofile true
diset tpcc maria_async_scale false
diset tpcc maria_driver timed

# Ensure test is limited by time
diset tpcc maria_total_iterations 1000000

# Timed duration
diset tpcc maria_rampup 2
diset tpcc maria_duration 8
# Distribute load
diset tpcc maria_allwarehouse false

# Transactions options
tcset refreshrate 10
tcset logtotemp 1
tcset unique 1
tcset timestamps 1

# Run
foreach z {2 4 8 12} {
    set w [expr {$z * 5}]
    diset tpcc maria_count_ware $w
    puts "Building Schema for $z TEST"
    # Delete possible previous data
    deleteschema
    vudestroy
    # Build and load
    buildschema
    vudestroy
    loadscript
    # Vuser options
    vuset delay 500
    vuset repeat 500
    vuset iterations 1
    vuset showoutput 0
    vuset logtotemp 1
    vuset unique 1
    vuset nobuff 0
    vuset timestamps 1
    puts "Starting $z VU TEST"
    tcstart
    vuset vu $z
    vucreate
    vurun
    puts "Waiting 1 for cleanup and collection"
    after 60000
    puts "Destroying VU"
    vudestroy
    tcstop
}
```

A.3 PostgreSQL TCL Script

Listing 3: TCL script for PostgreSQL

```
#!/usr/bin/tclsh
# Set Database & Benchmark
dbset db pg
dbset bm TPC-C

# DB configs
diset connection pg_host 172.22.0.2
diset connection pg_port 5432
diset connection pg_sslmode prefer
diset tpcc pg_superuser postgres
diset tpcc pg_superuserpass 1234
diset tpcc pg_defaultdb postgres
diset tpcc pg_user tpcc
diset tpcc pg_pass tpcc
diset tpcc pg_dbase tpcc
diset tpcc pg_tspace pg_default

# Default for WH and VU
diset tpcc pg_count_ware 50
diset tpcc pg_num_vu 10

# Driver script options
diset tpcc pg_timeprofile true
diset tpcc pg_async_scale false
diset tpcc pg_driver timed
# Ensure test is limited by time
diset tpcc pg_total_iterations 1000000
# Timed duration
diset tpcc pg_rampup 2
diset tpcc pg_duration 8
# Distribute load
diset tpcc pg_allwarehouse false
# Transactions options
tcset refreshrate 10
tcset logtotemp 1
tcset unique 1
tcset timestamps 1
# Run
foreach z {2 4 8 12} {
    set w [expr {$z * 5}]
    diset tpcc pg_count_ware $w
    puts "Building Schema for $z TEST"
    # Delete possible previous data
    deleteschema
    vudestroy
    # Build and load
    buildschema
    vudestroy
    loadscript
    # Vuser options
    vuset delay 500
    vuset repeat 500
    vuset iterations 1
    vuset showoutput 0
    vuset logtotemp 1
    vuset unique 1
    vuset nobuff 0
    vuset timestamps 1
    puts "Starting $z VU TEST"
    tcstart
    vuset vu $z
    vucreate
    vurun
    puts "Waiting 1 for cleanup and collection"
    after 60000
    puts "Destroying VU"
    vudestroy
    tcstop
}
```

B Docker

B.1 Docker Compose File

Listing 4: Docker Compose file for the databases and HammerDB

```
version: "3.9"

services:
  mysql:
    build:
      context: .
      dockerfile: Dockerfiles/MySQL/Dockerfile
    restart: always
    container_name: MySQL_SBD
    environment:
      MYSQL_ROOT_PASSWORD: 1234
    volumes:
      - mysql-volume:/var/lib/mysql
    networks:
      sbd_network:
        ipv4_address: 172.22.0.3

  postgres:
    image: postgres
    container_name: Postgres_SBD
    command: postgres -c config_file=/etc/postgresql/postgresql.conf
    environment:
      POSTGRES_PASSWORD: "1234"
    volumes:
      - postgres-volume:/var/lib/postgresql/data
      - ./Configs/Hard_Test/postgresql.conf:/etc/postgresql/postgresql.conf:ro
    networks:
      sbd_network:
        ipv4_address: 172.22.0.2

  mariadb:
    build:
      context: .
      dockerfile: Dockerfiles/MariaDB/Dockerfile
    restart: always
    container_name: MariaDB_SBD
    environment:
      MYSQL_ROOT_PASSWORD: 1234
      MYSQL_DATABASE: sbdDatabase
    volumes:
      - mariadb-volume:/var/lib/mysql:rw
    networks:
      sbd_network:
        ipv4_address: 172.22.0.4

  hammer-gui:
    image: tpcorg/hammerdb:latest-cloudtk
    container_name: HammerDB_SBD
    restart: always
    ports:
      - "8081:8081"
      - "8082:8082"
      - "8080:8080"
    depends_on:
      - mysql
      - postgres
      - mariadb
    volumes:
      - ./Scripts:/home/HammerDB-4.10/scripts/tcl-scripts
      - ./stats-logs:/exp-logs
    networks:
      sbd_network:
        ipv4_address: 172.22.0.5

volumes:
  mysql-volume:
```

```
    driver: local
  postgres-volume:
    driver: local
  mariadb-volume:
    driver: local

networks:
  sbd_network:
    driver: bridge
    ipam:
      config:
        - subnet: 172.22.0.0/16
```

B.2 Dockerfile for MySQL

Listing 5: Dockerfile for MySQL

```
FROM mysql:8.4

RUN mkdir -p /var/log/mysql && \
    chown -R mysql:mysql /var/log/mysql

COPY ./Configs/Hard_Test/mysql.cnf /etc/mysql/conf.d/mysql.cnf

RUN chmod 644 /etc/mysql/conf.d/mysql.cnf
```

B.3 Dockerfile for MariaDB

Listing 6: Dockerfile for MariaDB

```
FROM mariadb:11.4

COPY ./Configs/Hard_Test/my.cnf /etc/mysql/my.cnf

RUN chmod 644 /etc/mysql/my.cnf
```

C Configs

C.1 MySQL Configuration File

Listing 7: MySQL configuration file

```
[mysqld]

#------------------------------------------------------------------------------
# BASIC SETTINGS
#------------------------------------------------------------------------------
port = 3306
bind-address = 0.0.0.0
max_connections = 1000
max_connect_errors = 1000000
wait_timeout = 28800
interactive_timeout = 28800
connect_timeout = 10
back_log = 1500

#------------------------------------------------------------------------------
# CHARACTER SET & COLLATION
#------------------------------------------------------------------------------
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci

#------------------------------------------------------------------------------
# STORAGE AND INNODB ENGINE
#------------------------------------------------------------------------------
innodb_buffer_pool_size = 4G
innodb_buffer_pool_instances = 4
innodb_log_file_size = 512M
innodb_log_buffer_size = 64M
innodb_file_per_table = 1
innodb_flush_log_at_trx_commit = 2
innodb_flush_method = O_DIRECT
innodb_io_capacity = 2000
innodb_io_capacity_max = 4000
innodb_read_io_threads = 8
innodb_write_io_threads = 8
innodb_purge_threads = 4
innodb_doublewrite = 1
innodb_autoinc_lock_mode = 2
innodb_stats_persistent = 1
innodb_lru_scan_depth = 2048
innodb_adaptive_flushing = 1
innodb_adaptive_hash_index = 0
innodb_change_buffering = none

#------------------------------------------------------------------------------
# TEMPORARY TABLE & BUFFERS
#------------------------------------------------------------------------------
tmp_table_size = 256M
max_heap_table_size = 256M
sort_buffer_size = 1M
join_buffer_size = 1M
read_buffer_size = 512K
read_rnd_buffer_size = 2M

#------------------------------------------------------------------------------
# LOGGING
#------------------------------------------------------------------------------
slow_query_log = 1
long_query_time = 2
slow_query_log_file = /var/lib/mysql/mysql-slow.log
general_log = 0

#------------------------------------------------------------------------------
# BYNARY LOGGING
#------------------------------------------------------------------------------
skip-log-bin
sync_binlog = 0
```

```

#-------------------------------------------------------------------------------
# SECURITY & COMPATIBILITY
#-------------------------------------------------------------------------------
local_infile = 0
sql_mode = STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION
transaction_isolation = REPEATABLE-READ

#-------------------------------------------------------------------------------
# PERFORMANCE
#-------------------------------------------------------------------------------
performance_schema = OFF
max_prepared_stmt_count = 12800
table_open_cache = 2000
table_open_cache_instances = 4
open_files_limit = 65535
thread_cache_size = 50
thread_stack = 256K

#-------------------------------------------------------------------------------
# MONITORING
#-------------------------------------------------------------------------------
innodb_monitor_enable = '%'

#-------------------------------------------------------------------------------
# PLUGIN & AUTHENTICATION
#-------------------------------------------------------------------------------
require_secure_transport = OFF
caching_sha2_password_auto_generate_rsa_keys = ON
```

C.2 MariaDB Configuration File

Listing 8: MariaDB configuration file

```
[mysqld]

#-------------------------------------------------------------------------------
# BASIC SETTINGS
#-------------------------------------------------------------------------------
port = 3306
bind-address = 0.0.0.0
max_connections = 1000
max_connect_errors = 1000000
wait_timeout = 28800
interactive_timeout = 28800
connect_timeout = 10
back_log = 1500

#-------------------------------------------------------------------------------
# CHARACTER SET & COLLATION
#-------------------------------------------------------------------------------
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci

#-------------------------------------------------------------------------------
# STORAGE AND INNODB ENGINE
#-------------------------------------------------------------------------------
innodb_buffer_pool_size = 4G
innodb_buffer_pool_instances = 4
innodb_log_file_size = 512M
innodb_log_buffer_size = 64M
innodb_file_per_table = 1
innodb_flush_log_at_trx_commit = 2
innodb_flush_method = O_DIRECT
innodb_io_capacity = 2000
innodb_io_capacity_max = 4000
innodb_read_io_threads = 8
innodb_write_io_threads = 8
innodb_purge_threads = 4
innodb_doublewrite = 1
innodb_autoinc_lock_mode = 2
```

```
innodb_stats_persistent = 1
innodb_lru_scan_depth = 2048
innodb_adaptive_flushing = 1
innodb_adaptive_hash_index = 0
innodb_change_buffering = none

#------------------------------------------------------------------------------
# TEMPORARY TABLE & BUFFERS
#------------------------------------------------------------------------------
tmp_table_size = 256M
max_heap_table_size = 256M
sort_buffer_size = 1M
join_buffer_size = 1M
read_buffer_size = 512K
read_rnd_buffer_size = 2M

#------------------------------------------------------------------------------
# LOGGING
#------------------------------------------------------------------------------
slow_query_log = 1
long_query_time = 2
slow_query_log_file = /var/lib/mysql/mysql-slow.log
general_log = 0

#------------------------------------------------------------------------------
# BYNARY LOGGING
#------------------------------------------------------------------------------
skip-log-bin
sync_binlog = 0

#------------------------------------------------------------------------------
# SECURITY & COMPATIBILITY
#------------------------------------------------------------------------------
local_infile = 0
sql_mode = STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION
transaction_isolation = REPEATABLE-READ

#------------------------------------------------------------------------------
# PERFORMANCE
#------------------------------------------------------------------------------
performance_schema = OFF
max_prepared_stmt_count = 12800
table_open_cache = 2000
table_open_cache_instances = 4
open_files_limit = 65535
thread_cache_size = 50
thread_stack = 256K

#------------------------------------------------------------------------------
# MONITORING
#------------------------------------------------------------------------------
innodb_monitor_enable = '%'
```

C.3 PostgreSQL Configuration File

Listing 9: PostgreSQL configuration file

```
#------------------------------------------------------------------------------
# FILE LOCATIONS
#------------------------------------------------------------------------------
data_directory = '/var/lib/postgresql/data'
hba_file = '/var/lib/postgresql/data/pg_hba.conf'
ident_file = '/var/lib/postgresql/data/pg_ident.conf'

#------------------------------------------------------------------------------
# CONNECTIONS AND AUTHENTICATION
#------------------------------------------------------------------------------
listen_addresses = '*'
port = 5432
max_connections = 100

#------------------------------------------------------------------------------
# RESOURCE USAGE
#------------------------------------------------------------------------------
shared_buffers = 512MB
work_mem = 64MB
maintenance_work_mem = 256MB
effective_cache_size = 2GB

#------------------------------------------------------------------------------
# WRITE-AHEAD LOG
#------------------------------------------------------------------------------
wal_level = replica
synchronous_commit = off
checkpoint_timeout = 15min
checkpoint_completion_target = 0.9
max_wal_size = 2GB
min_wal_size = 512MB

#------------------------------------------------------------------------------
# LOGGING (optional, but useful for debugging)
#------------------------------------------------------------------------------
logging_collector = on
log_directory = 'log'
log_filename = 'postgresql.log'
log_statement = 'none'
log_min_duration_statement = 1000
```

D Runner Scripts

D.1 Batch Script

Listing 10: Batch script to run the TCL scripts

```
@echo off
setlocal

:: Set container names
set HAMMER_CONTAINER=HammerDB_SBD

echo Starting HammerDB benchmark runner...

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: Setup containers
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

:: Compose down to ensure a clean start
echo Stopping and removing existing containers, images and volumes...
docker compose down --rmi all -v

timeout /t 5

:: Compose up to start the containers
echo Starting containers...
docker compose up -d

timeout /t 10

echo Containers are up and running!

echo Starting HammerDB benchmark...

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: POSTGRESQL
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

echo Starting up PostgreSQL benchmark...

:: Start the stats logging in a new window using the temp file
start "PCStats" cmd /c PCStats.bat "postgres"

timeout /t 5 >nul

docker exec -i %HAMMER_CONTAINER% /home/HammerDB-4.10/hammerdbcli auto scripts/tcl-scripts/largeTestPG.tcl

for /f "tokens=2 delims=," %%i in ('tasklist /v /fo csv ^| findstr /i /c:"PCStats"') do (
    taskkill /PID %%~i /F >nul 2>&1
)

echo Benchmark and monitoring complete for PostgreSQL.

timeout /t

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: MYSQL
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

echo Starting up MySQL benchmark...

:: Start the stats logging in a new window using the temp file
start "PCStats" cmd /c PCStats.bat "mysql"

timeout /t 5 >nul

docker exec -i %HAMMER_CONTAINER% /home/HammerDB-4.10/hammerdbcli auto scripts/tcl-scripts/largeTestMySQL.
    ↪ tcl

for /f "tokens=2 delims=," %%i in ('tasklist /v /fo csv ^| findstr /i /c:"PCStats"') do (
    taskkill /PID %%~i /F >nul 2>&1
)
```

```

echo Benchmark and monitoring complete for MySQL.

timeout /t 30

::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
:: MARIADB
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

echo Starting up MariaDB benchmark...

:: Start the stats logging in a new window using the temp file
start "PCStats" cmd /c PCStats.bat "mariadb"

timeout /t 5 >nul

docker exec -i %HAMMER_CONTAINER% /home/HammerDB-4.10/hammerdbcli auto scripts/tcl-scripts/
    ↪ largeTestMariaDB.tcl

for /f "tokens=2 delims=," %%i in ('tasklist /v /fo csv ^| findstr /i /c:"PCStats"') do (
    taskkill /PID %%~i /F >nul 2>&1
)

echo Benchmark and monitoring complete for MariaDB.

echo Benchmark and monitoring complete.
endlocal
```

D.1.1 PCStats Script

Listing 11: PCStats script to monitor the system in Windows

```
@echo off
setlocal enabledelayedexpansion

set DB=%1

:: Format date and time safely for filename
for /f "tokens=1-3 delims=/" %%a in ("%date%") do (
    set mm=%%a
    set dd=%%b
    set yyyy=%%c
)
for /f "tokens=1-3 delims=:." %%a in ("%time%") do (
    set hh=%%a
    set min=%%b
    set ss=%%c
)
set "logfile=sys_usage_%mm%_%dd%_%hh%_%min%_%DB%.csv"

:: Set interval (in seconds)
set interval=10

:: Write CSV header
echo Timestamp,CPU_Usage_Percent,RAM_Used_MB > "%logfile%"

:loop
:: Get timestamp in ISO format using PowerShell
for /f %%x in ('powershell -command "Get-Date -Format yyyy-MM-dd_HH:mm:ss"') do set timestamp=%%x

echo Current timestamp: %timestamp%

:: Get CPU usage
for /f "skip=1" %%x in ('wmic cpu get loadpercentage') do (
    if not "%%x"=="" (
        set cpu=%%x
        goto :gotCPU
    )
)
:gotCPU

```

```
:: Get RAM usage (in MB)
for /f "skip=1 tokens=2,3 delims=," %%a in ('wmic OS get FreePhysicalMemory^,TotalVisibleMemorySize /
    ↪ format:csv') do (
    set "free=%%a"
    set "total=%%b"
)
set /a usedMB=(%total% - %free%) / 1024

:: Write to CSV
echo %timestamp%,%cpu%,%usedMB% >> "%logfile%"

:: Wait and repeat
timeout /t %interval% >nul
goto loop
```

D.2 Shell Script

Listing 12: Shell script to run the TCL scripts

```
#!/bin/bash

# Set your container names
HAMMER_CONTAINER=HammerDB_SBD

echo "Starting HammerDB benchmark runner..."

######################################################################
# Setup containers
######################################################################

# Compose down to ensure a clean start
echo "Stopping and removing existing containers, images and volumes..."

docker compose down --rmi all -v

sleep 5

# Compose up to start the containers
echo "Starting containers..."

docker compose up -d

sleep 10

echo "Containers are up and running!"

echo "Starting HammerDB benchmark..."

######################################################################
# POSTGRESQL
######################################################################

echo "Starting up PostgreSQL benchmark..."

# Start the stats logging in the background
./PCStats.sh "postgres" &
PCSTATS_PID=$!

# Wait 5 seconds for stats logger to initialize
sleep 5

# Run the benchmark via Docker
docker exec -i "$HAMMER_CONTAINER" /home/HammerDB-4.10/hammerdbcli auto scripts/tcl-scripts/largeTestPG.
    ↪ tcl

# Kill the PCStats logger
kill "$PCSTATS_PID"

echo "Benchmark and monitoring complete for PostgreSQL."

# Wait 30 seconds before exiting
```

```
sleep 30

#####################################################################
# MYSQL
#####################################################################

echo "Starting up MySQL benchmark..."

# Start the stats logging in the background
./PCStats.sh "mysql" &
PCSTATS_PID=$!

# Wait 5 seconds for stats logger to initialize
sleep 5

# Run the benchmark via Docker
docker exec -i "$HAMMER_CONTAINER" /home/HammerDB-4.10/hammerdbcli auto scripts/tcl-scripts/largeTestMySQL
    ↪ .tcl

# Kill the PCStats logger
kill "$PCSTATS_PID"

echo "Benchmark and monitoring complete for MySQL."

# Wait 30 seconds before exiting
sleep 30

#####################################################################
# MARIADB
#####################################################################

echo "Starting up MariaDB benchmark..."

# Start the stats logging in the background
./PCStats.sh "mariadb" &
PCSTATS_PID=$!

# Wait 5 seconds for stats logger to initialize
sleep 5

# Run the benchmark via Docker
docker exec -i "$HAMMER_CONTAINER" /home/HammerDB-4.10/hammerdbcli auto scripts/tcl-scripts/
    ↪ largeTestMariaDB.tcl

# Kill the PCStats logger
kill "$PCSTATS_PID"

echo "Benchmark and monitoring complete for MariaDB."

echo "Benchmark and monitoring complete."
```

D.2.1 PCStats Script

Listing 13: PCStats script to monitor the system in Linux

```
#!/bin/bash

DB="$1"
interval=10

# Create log file name based on current date and time
timestamp=$(date "+%m_%d_%H_%M")
logfile="sys_usage_${timestamp}_${DB}.csv"

# Write CSV header
echo "Timestamp,CPU_Usage_Percent,RAM_Used_MB" > "$logfile"

while true; do
    # Get current timestamp
    timestamp=$(date "+%Y-%m-%d_%H:%M:%S")
    echo "Current timestamp: $timestamp"
```

```
17
18      # Get CPU usage percentage (average over 1 second)
19      cpu=$(top -bn2 | grep "Cpu(s)" | tail -n 1 | awk -F'id,' -v prefix="" '{ split($1, vs, ","); cpu=100 -
            ↪  vs[length(vs)]; printf "%.0f", cpu }')
20
21      # Get RAM usage in MB
22      mem_total=$(free -m | awk '/Mem:/ {print $2}')
23      mem_used=$(free -m | awk '/Mem:/ {print $3}')
24
25      # Write to CSV
26      echo "$timestamp,$cpu,$mem_used" >> "$logfile"
27
28      # Wait for interval
29      sleep "$interval"
30  done
```