



PERCONA
LIVE ONLINE
MAY 12 - 13th
2021

HammerDB

A Better Way to Benchmark Your Open Source Database

Hello!

Steve Shaw

Open source database @ Intel

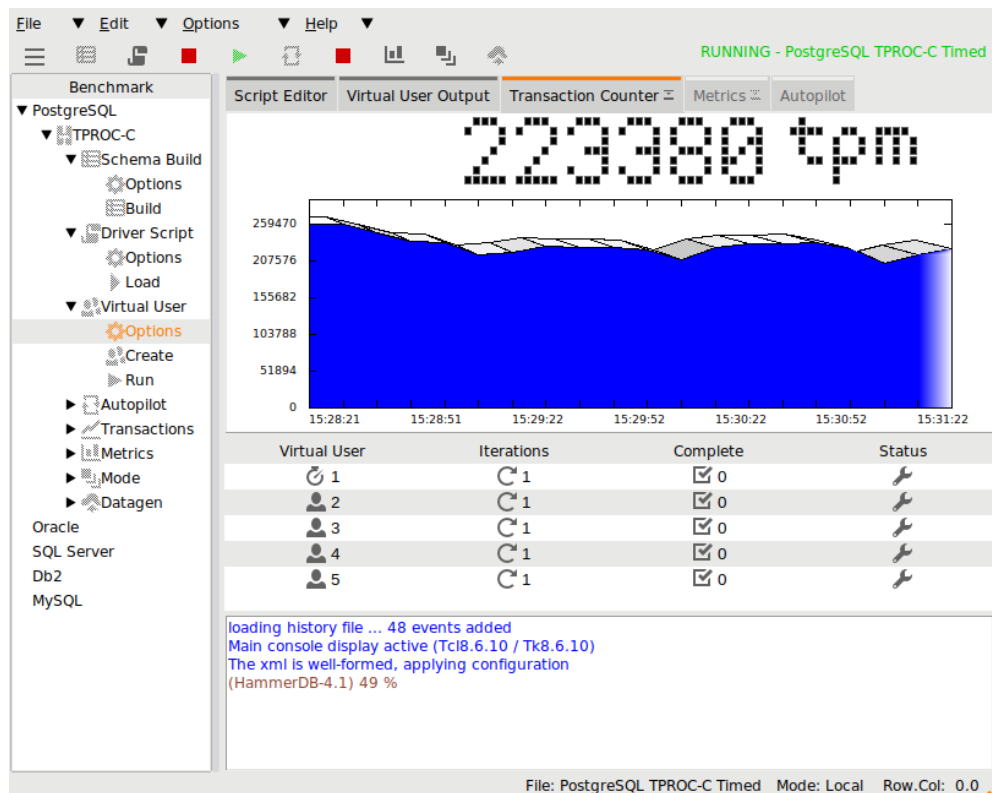
Main developer of HammerDB

Introduction

The background features abstract, flowing shapes in shades of orange and red. On the left, there are overlapping orange waves. On the right, there are overlapping red waves that transition from a darker red at the bottom to a lighter pink at the top. The overall effect is a modern, minimalist design.

What is HammerDB?

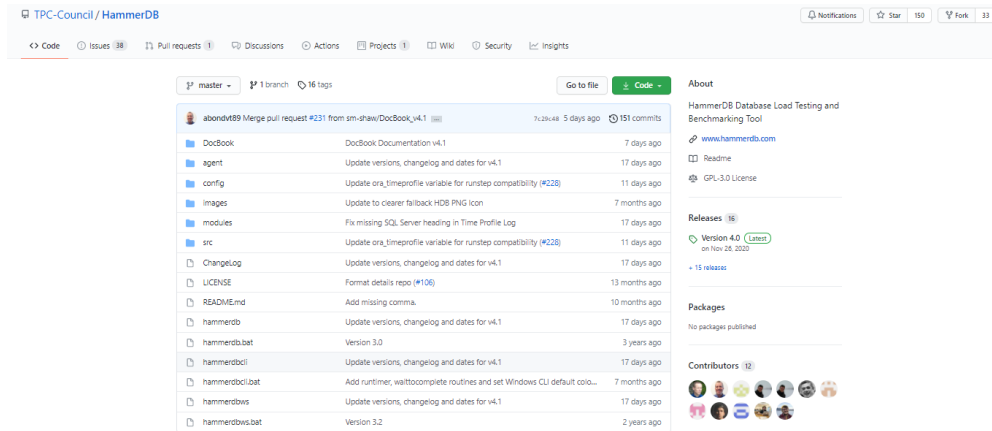
- Not a database!
- Leading open source tool for benchmarking relational databases
- Interfaces
 - Graphical
 - Command Line
 - Web REST interfaces
- Industry standard benchmarks
- High performance and scalability



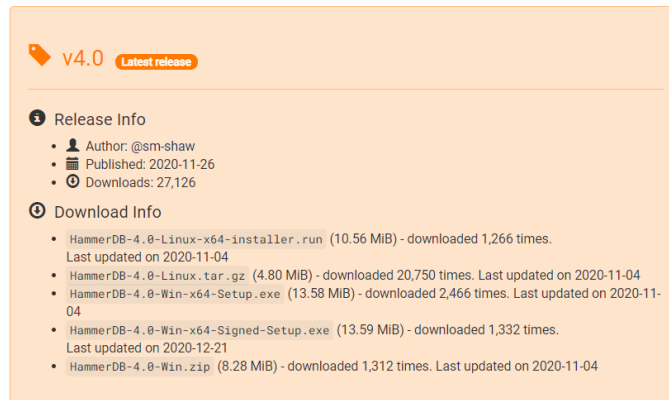


Open Source

- Hosted by TPC Council since 2019
 - Industry standard body for database benchmarks
- TPC-OSS subcommittee
 - Oversees and approves changes
- v4.1 Released on 22nd April 21
- Source code on GitHub
- Binaries @ GitHub Releases
 - <https://www.hammerdb.com/download.html>
- Client natively supports Linux and Windows on x64
 - GUI & CLI on both Linux and Windows
- GitHub Release Downloads @
 - <https://www.hammerdb.com/stats.html>
- Test databases on any platform



📥 GitHub Release Downloads 83,829



Supported Databases

- HammerDB supports the most popular relational databases
- Commercial and open source
- Metrics enable comparison across database engines

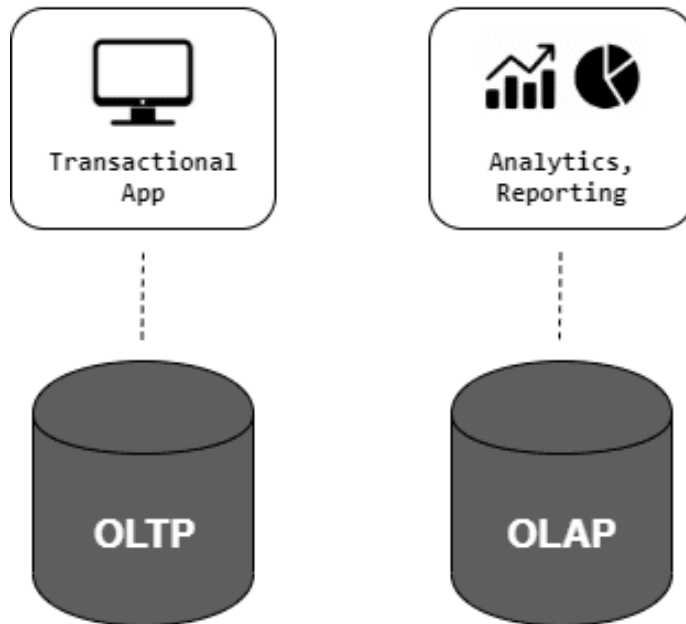
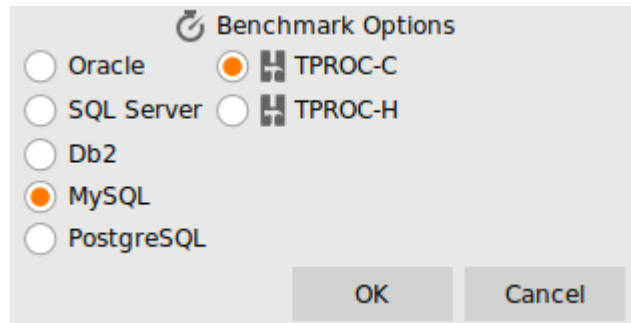


370 systems in ranking, April 2021

Rank			DBMS	Database Model	Score		
Apr 2021	Mar 2021	Apr 2020			Apr 2021	Mar 2021	Apr 2020
1.	1.	1.	Oracle +	Relational, Multi-model ⓘ	1274.92	-46.82	-70.51
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	1220.69	-34.14	-47.66
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	1007.97	-7.33	-75.46
4.	4.	4.	PostgreSQL +	Relational, Multi-model ⓘ	553.52	+4.23	+43.66
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	469.97	+7.58	+31.54
6.	6.	6.	IBM Db2 +	Relational, Multi-model ⓘ	157.78	+1.77	-7.85

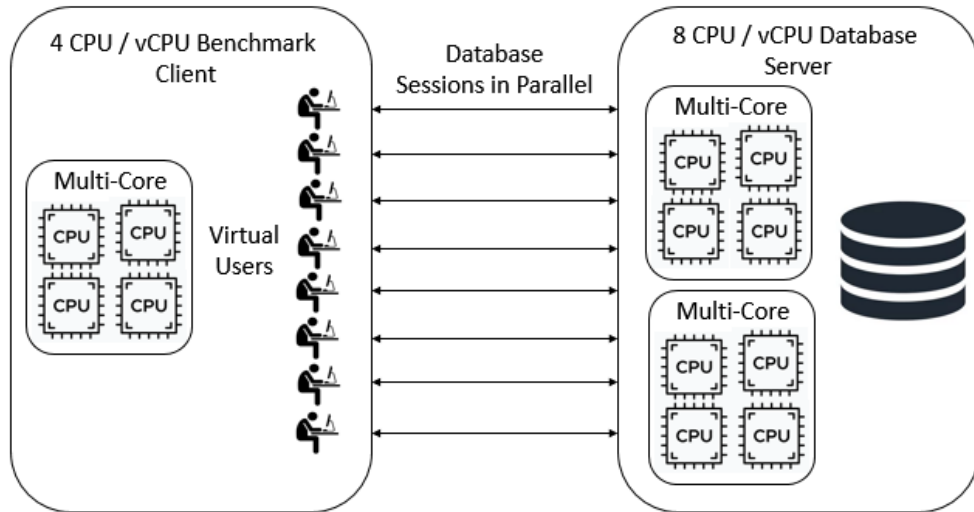
Supported Workloads

- TPROC-C = OLTP
 - Transactional workloads. Row oriented, high read and write throughput.
 - Derived from TPC-C
- TPROC-H = OLAP
 - Analytic, Decision Support
 - Focus on ETL
 - high bandwidth reads & minimal writes.
 - Derived from TPC-H
- Using TPCC/TPC-C, TPCH/TPC-H for derived workloads not permitted (trademark violation)



Key Database Benchmarking Concepts

- Parallel benchmarking software
 - Concurrency control must be in database, not in client
- Complex workloads designed to scale and test RDBMS concepts
 - Locking and latching
- Cross reference workloads across multiple database engines
 - Validate concepts
- HammerDB up to 6-7M NOPM on commercial database engines on 2 socket servers
 - High confidence levels that bottlenecks are in database software not HammerDB



HammerDB Programming Languages

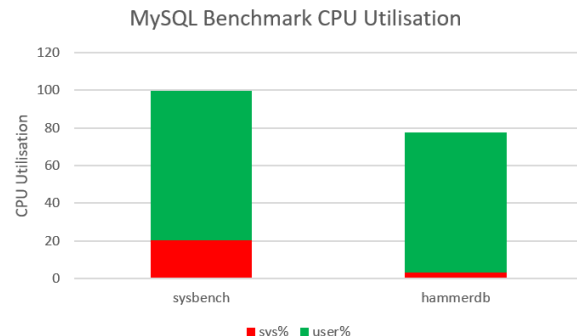
- Designed for High Performance and Scalability
- Database commands in SQL
- Application logic in stored procedures
- Database Interfaces in C
- More time in the database, less time in the 'round trip'
- More system resources for the database, less resources for the benchmark client
 - 3% for HammerDB
 - 20% in sysbench in socket/network layer

Database	Programming Interface
Oracle	OCI
SQL Server	ODBC
Db2	CLI
PostgreSQL	Libpq
MySQL	MySQL Native Driver

HammerDB Database Interfaces

Database	Application Logic
Oracle	PL/SQL
SQL Server	T-SQL
Db2	SQL PL
PostgreSQL	PL/pgSQL
MySQL	stored program language

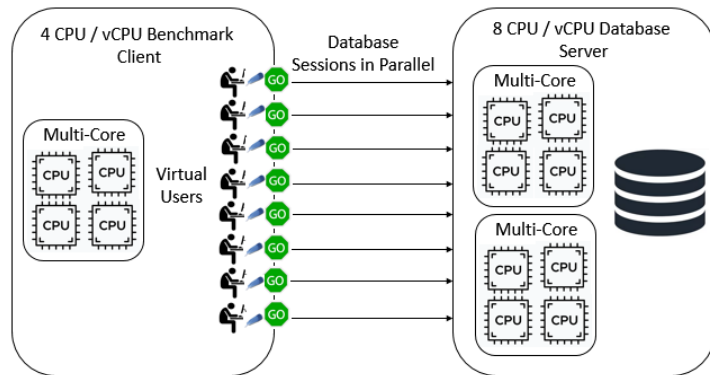
HammerDB Stored Procedures.



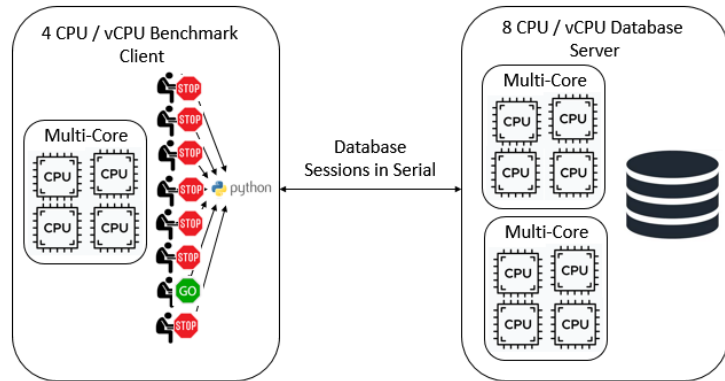
Tcl as Glue Language

- Python GIL limits to single-threading
- Tcl as glue language for truly parallel multithreading
- Tcl compiles into bytecode at runtime for high performance
- Co-routines used for event-driven scaling only to prevent bottleneck
- GUI & CLI on same codebase
- Native Tk GUI including 4k UHD scaling

Tcl Multithreading

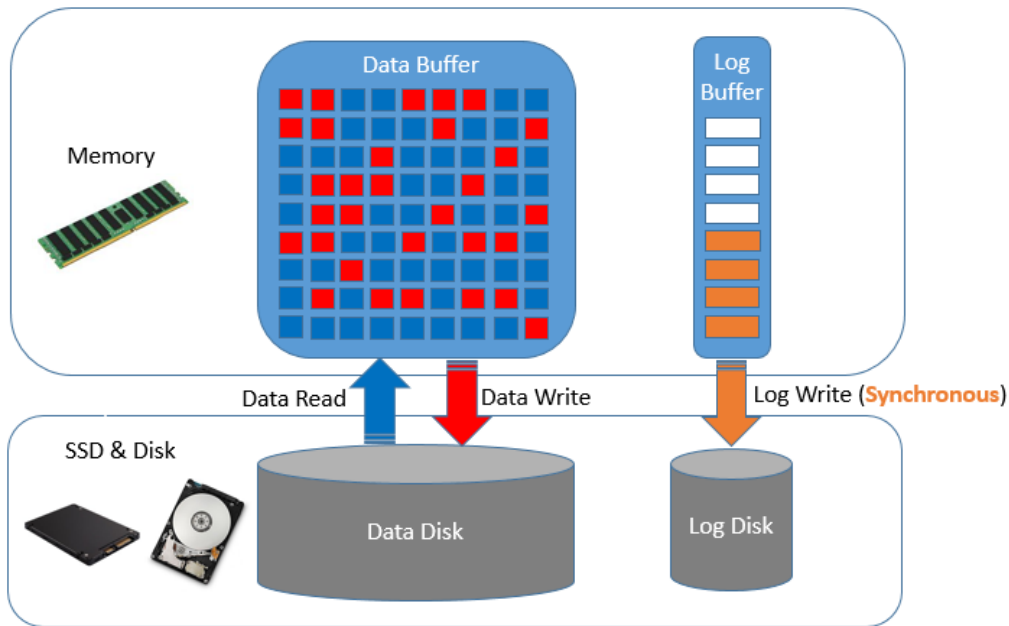


Python Multithreading with GIL



Cached vs Scaled Benchmarks

- Default Mode Cached workload
 - Testing goals
- Cached
 - Less than 5000 warehouses
 - All data in memory
 - 10s to hundreds of sessions
 - CPU & Memory Intensive
 - WAL & Redo Disk Write Intensive
 - Maximum performance at minimal configuration
- Scaled (Event driven scaling)
 - More than 5000 warehouses
 - Event driven scaling
 - Thousands of sessions
 - Middleware needed
 - Larger disk and networking requirement
 - Data Disk Read and Write Intensive
- Perfectly Scaled Configuration = Cached performance



System Configuration

The background features abstract, flowing shapes in shades of orange and red. On the left, there are overlapping orange waves. On the right, there are overlapping red waves. These waves meet in the center, creating a gradient effect. The overall composition is clean and modern, with the text 'System Configuration' centered in a large, black, sans-serif font.

CPU

- Some Linux releases default to CPU powersave mode

- 33% lower performance

```
sudo vi /etc/default/cpufrequtils  
GOVERNOR="performance"
```

```
systemctl restart cpufrequtils  
systemctl disable ondemand
```

```
sudo ./cpupower frequency-info  
analyzing CPU 0:  
  driver: intel_pstate
```

...

```
available cpufreq governors: performance powersave  
current policy: frequency should be within 800 MHz an  
3.40 GHz.
```

The governor "**performance**" may decide which speed to use

within this range.

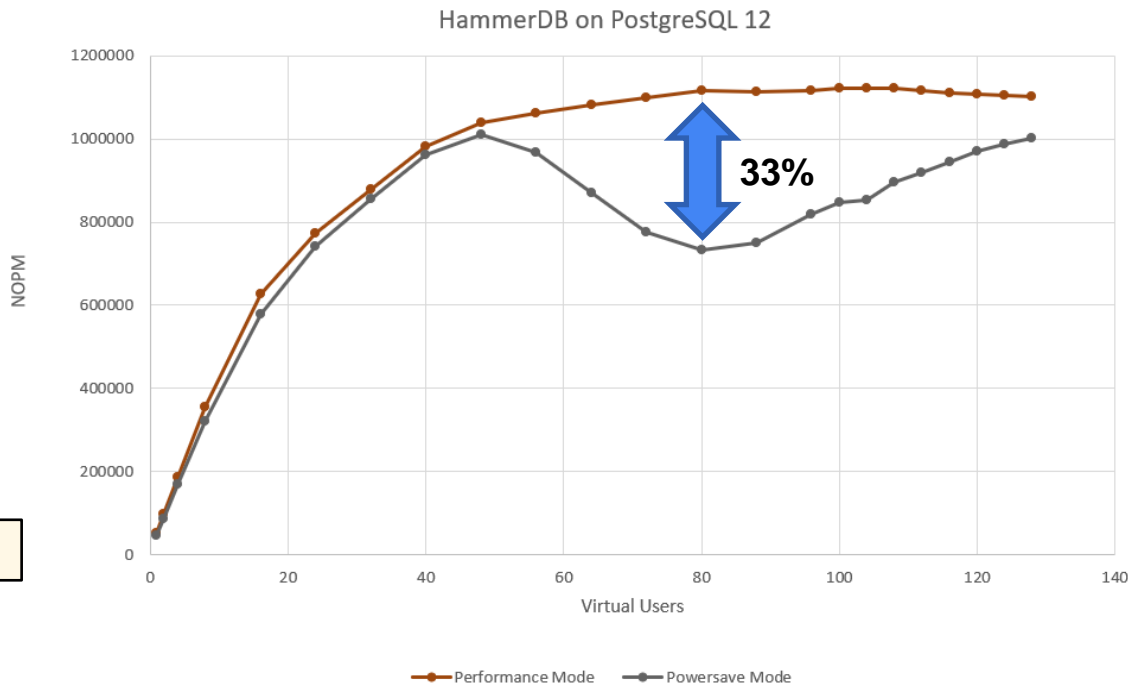
current CPU frequency: Unable to call hardware

current CPU frequency: 1.03 GHz (asserted by call to kern

boost state support:

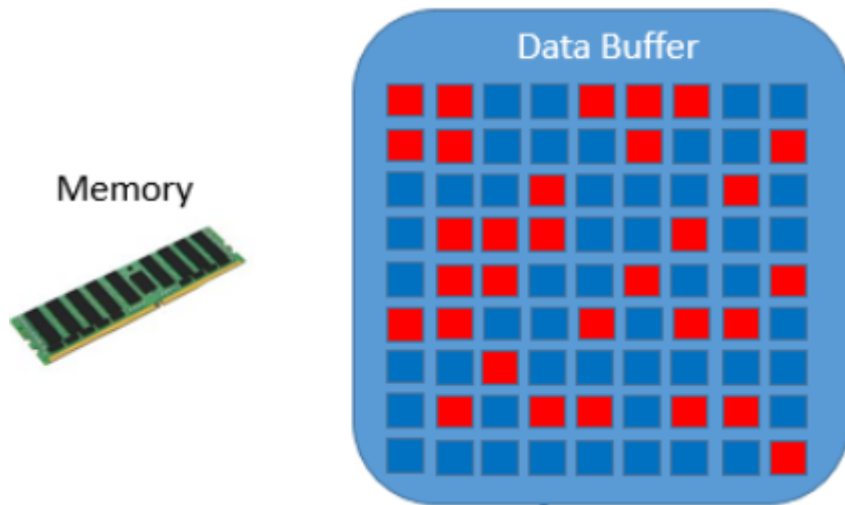
Supported: yes

Active: yes



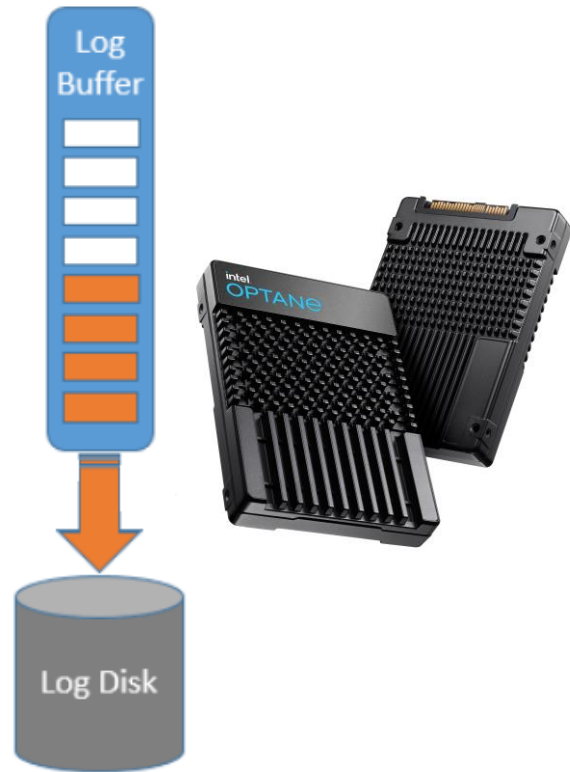
Memory

- For default cached workload
- Size Buffer Pool / Cache large enough to cache the TPROC-C schema
- Use 1GB Huge Pages for PostgreSQL
- MySQL/MariaDB InnoDB
 - `innodb_buffer_pool_size=64000M`
- PostgreSQL
 - `shared_buffers = 64000MB`
 - `huge_pages = on`



I/O WAL and Redo Performance

- Use Highest Performance SSDs for WAL/Redo
 - Intel Optane
 - Low latency writes
- Ensure partitions correctly aligned
- Use 1GB redo log / WAL segment size
- MySQL
 - `innodb_log_file_size=1024M`
 - `innodb_log_files_in_group=32`
- PostgreSQL
 - `initdb -D ./data --wal-segsize=1024`
- Synchronous Commit
 - What components are you testing?
- MySQL
 - `innodb_flush_log_at_trx_commit=0/1`
- PostgreSQL
 - `wal_level = minimal/replica`
 - `synchronous_commit = off/on`



Loading Database Client Libraries

- HammerDB needs access to client libraries to load interface
- CLI librarycheck command

```
hammerdb>librarycheck
Checking database library for MySQL
Success ... loaded library mysqltcl for MySQL
Checking database library for PostgreSQL
Success ... loaded library Pgtcl for PostgreSQL
```

- Export `LD_LIBRARY_PATH`

```
export LD_LIBRARY_PATH=/opt/postgresql-13.2/pgsql/lib/:$LD_LIBRARY_PATH
```

- Use `ldd` on HammerDB interfaces to verify library used
 - Up to v4.1 for MariaDB you need the MySQL client, from v4.2 MariaDB clients used

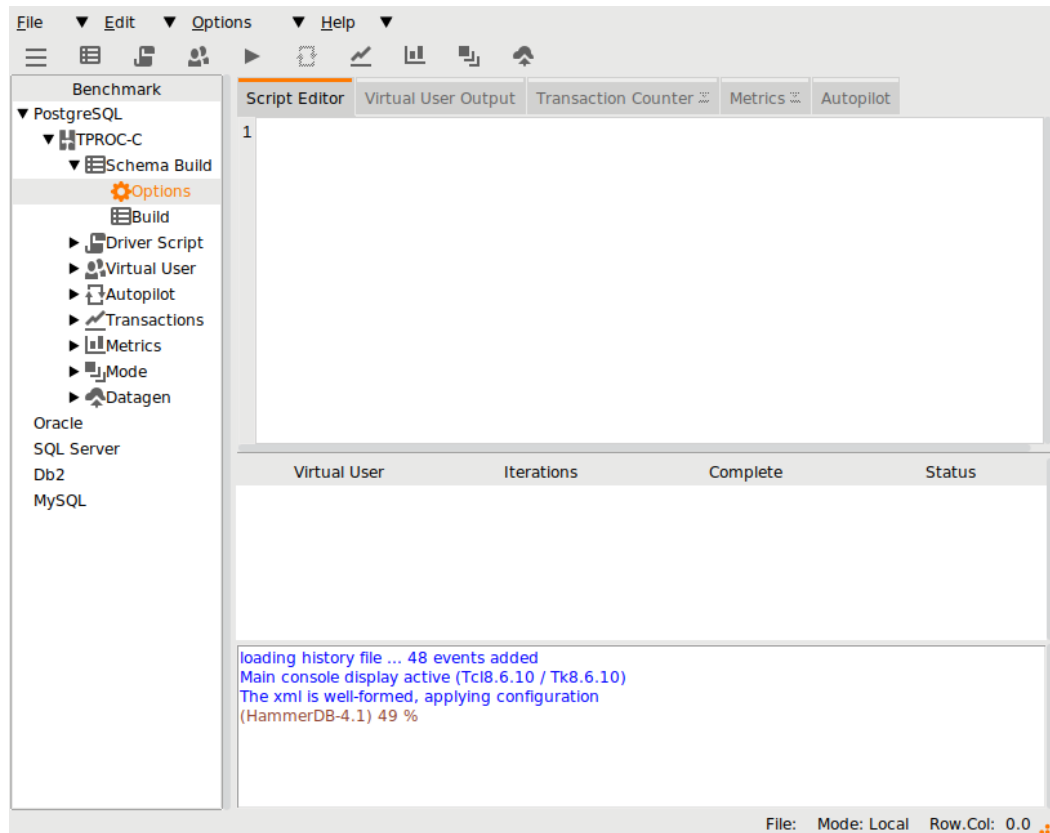
```
ldd libmysqltcl3.052.so
libmysqlclient.so.21 => /usr/lib/x86_64-linux-gnu/libmysqlclient.so.21
libpgtcl2.1.1.so
libpq.so.5 => /opt/postgresql-13.2/pgsql/lib/libpq.so.5 (0x00007f0e20ce5000)
```

Schema Build

The background features a series of overlapping, wavy, organic shapes in shades of orange and red. These shapes originate from the bottom left and flow towards the right, creating a sense of movement and depth. The colors transition from a bright orange on the left to a deeper red on the right, with some areas appearing as lighter, semi-transparent washes over the darker tones.

Schema Build

- Schema Build Options
 - Select options from menu
 - Configure with CLI commands
 - Same schema is built
- Options vary per database
 - MySQL Storage Engines
 - Partitioning
 - PostgreSQL stored procedures or functions
- Key Factors in Build Performance
 - CPU cores in client
 - I/O throughput on database



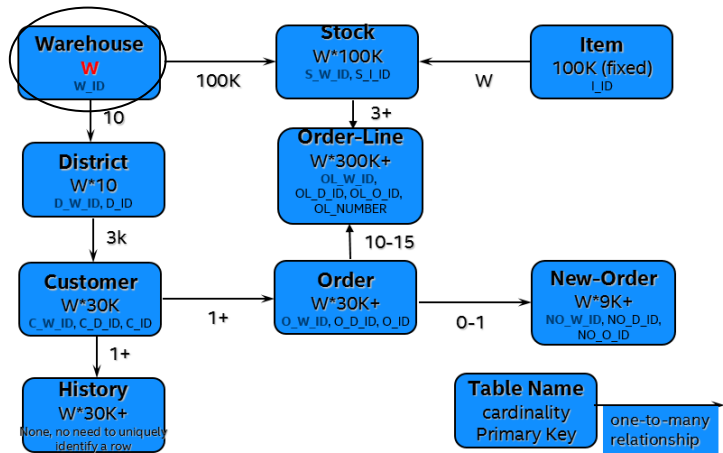
Schema Build Choices

- Schema Build
 - Creates tables
 - Creates and loads data
 - Creates Indexes
 - Creates functions/stored procedures
 - Gathers statistics
- Number of Warehouses
 - Define according to system scale
 - Entire schema scaled based on warehouse count
- Stored Procedures
 - New Order
 - Payment
 - Delivery
 - Stock Level
 - Order Status
- Virtual Users to Build Schema
 - Schema creates and loads data in parallel
 - Use number of CPU cores/threads on HammerDB client

Build Options

PostgreSQL Host :	localhost
PostgreSQL Port :	5432
PostgreSQL Superuser :	postgres
PostgreSQL Superuser Password :	postgres
PostgreSQL Default Database :	postgres
<input checked="" type="checkbox"/> TPROC-C PostgreSQL User :	tpcc
<input checked="" type="checkbox"/> TPROC-C PostgreSQL User Password :	tpcc
<input checked="" type="checkbox"/> TPROC-C PostgreSQL Database :	tpcc
<input checked="" type="checkbox"/> TPROC-C PostgreSQL Tablespace :	pg_default
EnterpriseDB Oracle Compatible :	<input type="checkbox"/>
PostgreSQL Stored Procedures :	<input type="checkbox"/>
Number of Warehouses :	1
Virtual Users to Build Schema :	1
Partition Order Line Table :	<input type="checkbox"/>

OK Cancel



How many warehouses?

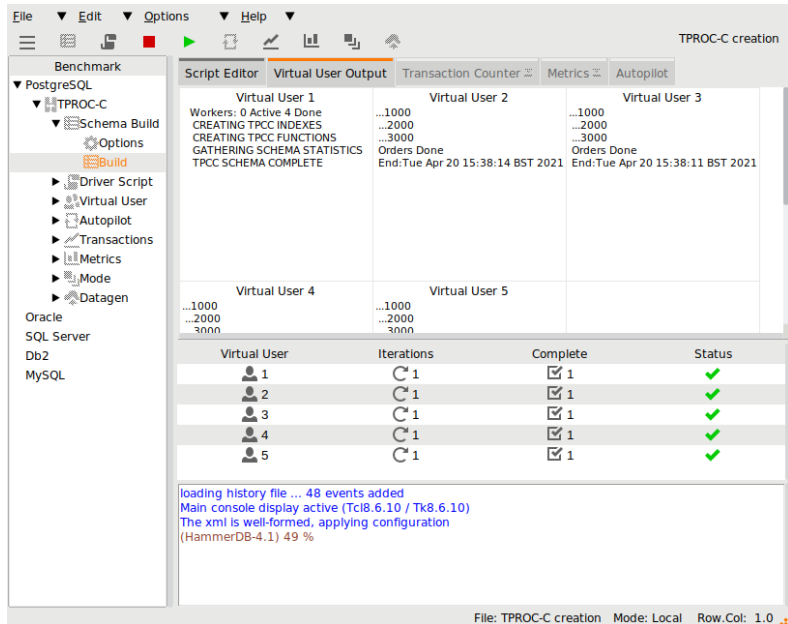
- Default Configuration
- Virtual Users chooses a home warehouse at random
- 90% of the workload satisfied from the home warehouse
 - Regardless of the number configured
 - Hot and cold data
- Configure enough warehouses to ensure an even spread of Virtual Users (eg 4X expected VU count)
- Overprovisioning warehouses will not increase performance or scalability
- Example on 2 socket 1000 warehouses
 - Takes 8-9 minutes to load
 - Depends on CPU and Disk and Virtual Users to build schema
- Warehouse Count Limits
 - 5000 warehouses in GUI
 - 30,000 in Datagen
 - No actual limit for advanced users, only interface limits



Running the Build

- Build Schema Command
 - GUI Build Option
 - CLI buildschema command

GUI



CLI Script

```
dbset db pg
dbset bm TPC-C
diset tpcc pg_count_ware 20
diset tpcc pg_num_vu 4
diset tpcc pg_superuser steve
diset tpcc pg_superuserpass postgres
buildschema
```

Run script

```
~/HammerDB-4.1$ ./hammerdbcli
HammerDB CLI v4.1
Copyright (C) 2003-2021 Steve Shaw
Type "help" for a list of commands
The xml is well-formed, applying configuration
hammerdb>source pgbuild.tcl
Database set to PostgreSQL
Benchmark set to TPC-C for PostgreSQL
Changed tpcc:pg_count_ware from 1 to 20 for PostgreSQL
Changed tpcc:pg_num_vu from 1 to 4 for PostgreSQL
Changed tpcc:pg_superuser from postgres to steve for PostgreSQL
Value postgres for tpcc:pg_superuserpass is the same as existing value postgres, no change made
Script cleared
Building 20 Warehouses with 5 Virtual Users, 4 active + 1 Monitor VU(dict value pg_num_vu is set to 4)
Ready to create a 20 Warehouse PostgreSQL TPROC-C schema
in host LOCALHOST:5432 under user TPCC in database TPCC?
Enter yes or no: replied yes
Vuser 1 created - WAIT IDLE
Vuser 2 created - WAIT IDLE
Vuser 3 created - WAIT IDLE
Vuser 4 created - WAIT IDLE
Vuser 5 created - WAIT IDLE
Vuser 1:RUNNING
Vuser 1:Monitor Thread
Vuser 1:CREATING TPCC SCHEMA
Vuser 1:CREATING DATABASE tpcc under OWNER tpcc
Vuser 2:RUNNING
Vuser 2:Worker Thread
Vuser 2:Waiting for Monitor Thread...
Vuser 1:CREATING TPCC TABLES
Vuser 1>Loading Item
Vuser 3:RUNNING
Vuser 3:Worker Thread
Vuser 3:Waiting for Monitor Thread...
Vuser 3>Loading 5 Warehouses start:6 end:10
Vuser 3:Start:Tue Apr 20 15:41:38 BST 2021
Vuser 3>Loading Warehouse
Vuser 3>Loading Stock Wid=6
Vuser 4:RUNNING
```

Datagen

GUI or CLI

- Bulk Loads to bypass database logging and network overhead

The screenshot displays the Datagen GUI interface. On the left, a sidebar shows a tree view with 'PostgreSQL' expanded, containing sub-items like 'TPROC-C', 'Schema Build', 'Driver Script', 'Virtual User', 'Autopilot', 'Transactions', 'Metrics', 'Mode', and 'Datagen'. Below this, database options for 'Oracle', 'SQL Server', 'Db2', and 'MySQL' are listed. The main window has tabs for 'Script Editor', 'Virtual User Output', 'Transaction Counter', 'Metrics', and 'Autopilot'. The 'Virtual User Output' tab is active, showing a table with columns 'Virtual User', 'Iterations', 'Complete', and 'Status'. Below this table, a log window displays messages such as 'loading history file ... 48 events added' and 'Main console display active (Tcl8.6.10 / Tk8.6.10)'. The status bar at the bottom indicates 'File: TPROC-C generation Mode: Local Row.Col: 0.0'.

Virtual User	Iterations	Complete	Status
1	1	1	✓
2	1	1	✓
3	1	1	✓
4	1	1	✓
5	1	1	✓

loading history file ... 48 events added
Main console display active (Tcl8.6.10 / Tk8.6.10)
The xml is well-formed, applying configuration
(HammerDB-4.1) 49 %

Schema Data in text files

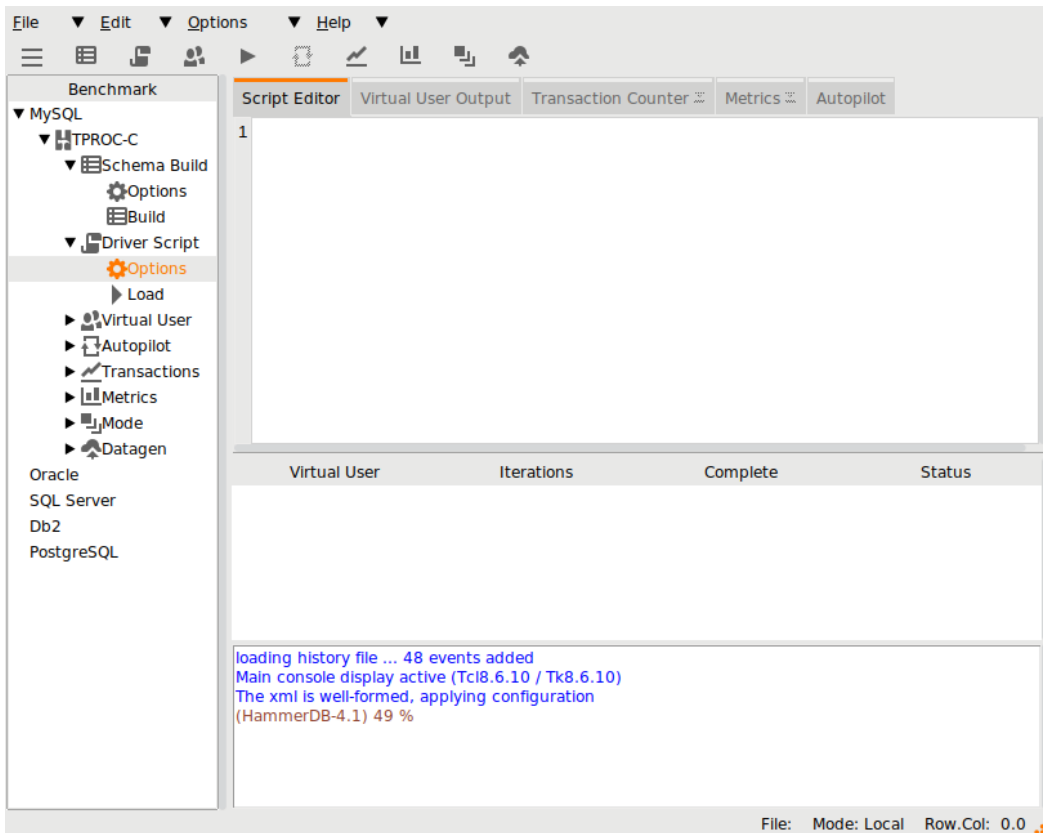
```
84868629 Apr 20 16:02 customer_1.tbl
84876579 Apr 20 16:02 customer_2.tbl
85005196 Apr 20 16:02 customer_3.tbl
84998551 Apr 20 16:02 customer_4.tbl
4557 Apr 20 16:01 district_1.tbl
4569 Apr 20 16:01 district_2.tbl
4606 Apr 20 16:01 district_3.tbl
4541 Apr 20 16:01 district_4.tbl
8523405 Apr 20 16:02 history_1.tbl
8585538 Apr 20 16:02 history_2.tbl
8823033 Apr 20 16:02 history_3.tbl
8825046 Apr 20 16:02 history_4.tbl
7556823 Apr 20 15:58 item_1.tbl
409500 Apr 20 16:03 new_order_1.tbl
418500 Apr 20 16:03 new_order_2.tbl
454500 Apr 20 16:03 new_order_3.tbl
454500 Apr 20 16:03 new_order_4.tbl
98030250 Apr 20 16:03 order_line_1.tbl
98708359 Apr 20 16:03 order_line_2.tbl
101014415 Apr 20 16:03 order_line_3.tbl
101028751 Apr 20 16:03 order_line_4.tbl
5997569 Apr 20 16:03 orders_1.tbl
6027850 Apr 20 16:03 orders_2.tbl
6147526 Apr 20 16:03 orders_3.tbl
6147400 Apr 20 16:03 orders_4.tbl
152901209 Apr 20 16:01 stock_1.tbl
152996867 Apr 20 16:01 stock_2.tbl
153408316 Apr 20 16:01 stock_3.tbl
153395417 Apr 20 16:01 stock_4.tbl
429 Apr 20 16:01 warehouse_1.tbl
471 Apr 20 16:01 warehouse_2.tbl
441 Apr 20 16:01 warehouse_3.tbl
419 Apr 20 16:01 warehouse_4.tbl
```

Running the Test

The background of the slide features abstract, flowing shapes in shades of orange and red. On the left, there are overlapping orange waves. On the right, there are overlapping red waves. These two sets of waves meet in the center, creating a gradient effect. The overall design is clean and modern.

Running the Test

- Driver Script Options
 - Test Loads a driver script
 - Options modify script loaded
- Test Script
 - Simple run
 - Small number of Virtual Users
 - Verify Schema Build
- Timed Script
 - Measured Test
 - Small to larger number of Virtual Users
 - Suppressed Output



Driver Script Options

- Connection Parameters
- Driver Script
- Total Transactions
 - Sets an upper limit for number of transactions for each Virtual User to run
- Rampup Time
 - Time for data to load into cache
- Test Duration
 - Timed period of test
- Advanced Options

The screenshot shows a 'Driver Options' dialog box with the following fields and controls:

- MySQL Host : 127.0.0.1
- MySQL Port : 3306
- MySQL Socket : /tmp/mysql.sock
- MySQL User : root
- MySQL User Password : mysql
- TPROC-C MySQL Database : tpcc
- TPROC-C Driver Script : ☐ Test Driver Script, ☒ Timed Driver Script
- Total Transactions per User : 1000000
- Exit on MySQL Error : ☐
- Keying and Thinking Time : ☐
- Prepare Statements : ☐
- Minutes of Rampup Time : 2
- Minutes for Test Duration : 5
- Use All Warehouses : ☐
- Time Profile : ☐
- Asynchronous Scaling : ☐
- Asynch Clients per Virtual User : 10
- Asynch Client Login Delay : 1000
- Asynchronous Verbose : ☐
- XML Connect Pool : ☐

At the bottom right are 'OK' and 'Cancel' buttons.

Virtual Users

- Configure and Create Virtual Users
 - Virtual Users run in parallel
 - Each Virtual User is OS thread
 - Runs independently

Virtual User Options

Virtual Users :

User Delay(ms) :

Repeat Delay(ms) :

Iterations :

☒ Show Output

☐ Log Output to Temp

☐ Use Unique Log Name

☐ No Log Buffer

☐ Log Timestamps

OK Cancel

MySQL TPROC-C Timed

File Edit Options Help

Benchmark

- MySQL
 - ▼ TPROC-C
 - Schema Build
 - Options
 - Build
 - Driver Script
 - Options
 - Load
 - Virtual User
 - Options
 - Create
 - Run
 - Autopilot
 - Transactions
 - Metrics
 - Mode
 - Dataset
 - Oracle
 - SQL Server
 - Db2
 - PostgreSQL

Script Editor Virtual User Output Transaction Counter Metrics Autopilot

Virtual User	Iterations	Complete	Status
Virtual User 1-MONITOR			
Virtual User 2			
Virtual User 3			
Virtual User 4			
Virtual User 5			

Virtual User	Iterations	Complete	Status
1	1	0	⌚
2	1	0	⌚
3	1	0	⌚
4	1	0	⌚
5	1	0	⌚

loading history file ... 48 events added
Main console display active (Tcl8.6.10 / Tk8.6.10)
The xml is well-formed, applying configuration
(HammerDB-4.1) 49 %

File: MySQL TPROC-C Timed Mode: Local Row.Col: 0.0

Running the Test

- Click Run to start
- Transaction Mix
 - New Order 45%
 - Payment 43%
 - Delivery 4%
 - Stock Level 4%
 - Order Status 4%
- Status shown of Virtual Users
 - Running
 - Complete
 - Error Status
- Press Stop to terminate Virtual Users

File Edit Options Help

Benchmark

- MySQL
 - TPROC-C
 - Schema Build
 - Options
 - Build
 - Driver Script
 - Options
 - Load
 - Virtual User
 - Options
 - Create
 - Run
 - Autopilot
 - Transactions
 - Metrics
 - Mode
 - Datagen
 - Oracle
 - SQL Server
 - Db2
 - PostgreSQL

Script Editor Virtual User Output Transaction Counter Metrics Autopilot

RUNNING - MySQL TPROC-C Timed

Virtual User 1-MONITOR
Beginning rampup time of 2 minutes

Virtual User 2
Processing 1000000 transactions with output suppressed...

Virtual User 3
Processing 1000000 transactions with output suppressed...

Virtual User 4
Processing 1000000 transactions with output suppressed...

Virtual User 5
Processing 1000000 transactions with output suppressed...

Virtual User	Iterations	Complete	Status
1	1	0	
2	1	0	
3	1	0	
4	1	0	
5	1	0	

loading history file ... 48 events added
Main console display active (Tcl8.6.10 / Tk8.6.10)
The xml is well-formed, applying configuration
(HammerDB-4.1) 49 %

File: MySQL TPROC-C Timed Mode: Local Row.Col: 0.0

Running a Test with the CLI

- Choose Options
- Load Script
- Workload is identical to that run by GUI (driver script is the same)

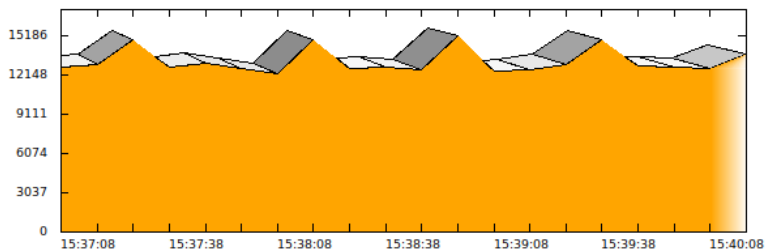
```
dbset db mysql
diset connection mysql_host localhost
diset connection mysql_port 3306
diset connection mysql_socket /tmp/mysql.sock
diset tpcc mysql_user root
diset tpcc mysql_pass mysql
diset tpcc mysql_driver timed
diset tpcc mysql_rampup 2
diset tpcc mysql_duration 5
loadscript
vuset vu 4
vucreate
vurun
runtimer 500
vudestroy
```

```
HammerDB CLI v4.1
Copyright (C) 2003-2021 Steve Shaw
Type "help" for a list of commands
The xml is well-formed, applying configuration
hammerdb>source myrun.tcl
Database set to MySQL
Changed connection:mysql_host from 127.0.0.1 to localhost for MySQL
Value 3306 for connection:mysql_port is the same as existing value 3306, no change made
Value /tmp/mysql.sock for connection:mysql_socket is the same as existing value /tmp/mysql.sock, no change made
Value root for tpcc:mysql_user is the same as existing value root, no change made
Value mysql for tpcc:mysql_pass is the same as existing value mysql, no change made
Clearing Script, reload script to activate new setting
Script cleared
Changed tpcc:mysql_driver from test to timed for MySQL
Value 2 for tpcc:mysql_rampup is the same as existing value 2, no change made
Value 5 for tpcc:mysql_duration is the same as existing value 5, no change made
Script loaded, Type "print script" to view
Vuser 1 created MONITOR - WAIT IDLE
Vuser 2 created - WAIT IDLE
Vuser 3 created - WAIT IDLE
Vuser 4 created - WAIT IDLE
Vuser 5 created - WAIT IDLE
5 Virtual Users Created with Monitor VU
Vuser 1:RUNNING
Vuser 1:Beginning rampup time of 2 minutes
Vuser 2:RUNNING
Vuser 2:Processing 1000000 transactions with output suppressed...
Vuser 3:RUNNING
Vuser 3:Processing 1000000 transactions with output suppressed...
Vuser 4:RUNNING
Vuser 4:Processing 1000000 transactions with output suppressed...
Vuser 5:RUNNING
Vuser 5:Processing 1000000 transactions with output suppressed...
```

Review Results

- Test Result Printed when compete
 - GUI
 - CLI
- Review Engine Throughput with Transaction Counter

13680 tpm



MySQL TPROC-C Timed

File Edit Options Help

Benchmark

- MySQL
 - TPROC-C
 - Schema Build
 - Options
 - Build
 - Driver Script
 - Options
 - Load
 - Virtual User
 - Options
 - Create
 - Run
 - Autopilot
 - Transactions
 - Metrics
 - Mode
 - Datagen
 - Oracle
 - SQL Server
 - Db2
 - PostgreSQL

Script Editor Virtual User Output Transaction Counter Metrics Autopilot

Virtual User 1-MONITOR
4 ... 5 ... Test complete, Taking end Transaction Count.
4 Active Virtual Users configured
TEST RESULT : System achieved 4351 NOPM from 13299 MySQL TPM

Virtual User 2
Processing 1000000 transactions with output suppressed...

Virtual User 3
Processing 1000000 transactions with output suppressed...

Virtual User 4
Processing 1000000 transactions with output suppressed...

Virtual User 5
Processing 1000000 transactions with output suppressed...

Virtual User	Iterations	Complete	Status
1	1	1	✓
2	1	1	✓
3	1	1	✓
4	1	1	✓
5	1	1	✓

loading history file ... 48 events added
Main console display active (Tcl8.6.10 / Tk8.6.10)
The xml is well-formed, applying configuration
(HammerDB-4.1) 49 %

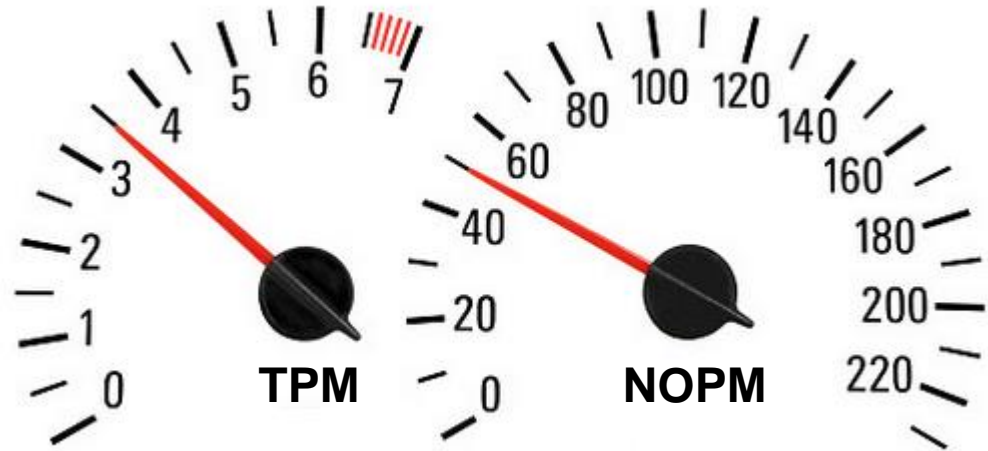
File: MySQL TPROC-C Timed Mode: Local Row.Col: 0.0

User 1:Test complete, Taking end Transaction Count.
User 1:4 Active Virtual Users configured
User 1:TEST RESULT : System achieved 4335 NOPM from 13058 MySQL TPM
User 1:FINISHED SUCCESS
User 4:FINISHED SUCCESS
Timer: 7 minutes elapsed
User 2:FINISHED SUCCESS
User 5:FINISHED SUCCESS
User 3:FINISHED SUCCESS
ALL VIRTUAL USERS COMPLETE
runtimer returned after 424 seconds
vudestroy success

Understanding Results: NOPM vs TPM

```
Vuser 1:Test complete, Taking end Transaction Count.  
Vuser 1:140 Active Virtual Users configured  
Vuser 1:TEST RESULT : System achieved 1722391 NOPM from 5216847 MySQL TPM
```

- **NOPM**
 - **How fast you are going**
 - Close relation to official tpmC
- **TPM**
 - **How hard your engine is working**
- **Comparing performance**
 - NOPM can be compared between engines
 - TPM can only be compared across the same engine
 - TPM useful engineering metric to compare statistics



GUI Automation: Autopilot

- GUI Automation
- Run Unattended Test Sequence
- Define sequence of tests
 - Increased Virtual User Count
- Log Output

Autopilot Options

☐ Autopilot Disabled
☒ Autopilot Enabled

Minutes per Test in Virtual User Sequence : 10

Active Virtual User Sequence (Space Separated) : 1 2 4 8 12 16 20 24

☒ Show Virtual User Output
☐ Log Virtual User Output to Temp
☐ Use Unique Log Name
☐ No Log Buffer
☐ Log Timestamps

OK Cancel

File Edit Options Help

Benchmark

- MySQL
 - TPROC-C
 - Schema Build
 - Driver Script
 - Options
 - Load
 - Virtual User
 - Autopilot
 - Options
 - Autopilot
 - Transactions
 - Metrics
 - Mode
 - Datagen
- Oracle
- SQL Server
- Db2
- PostgreSQL

Script Editor Virtual User Output Transaction Counter Metrics Autopilot

RUNNING - MySQL TPROC-C Timed

00d:00h:00m:18s

Autopilot Sequence 1 2 4 8 12 16 20 24 started at 16:22:50_04/22/2021
1 Active Virtual User Test started at 16:22:50_04/22/2021 with Monitor VU
Beginning rampup time of 2 minutes

Virtual User	Iterations	Complete	Status
1	1	0	
2	1	0	

loading history file ... 48 events added
Main console display active (Tcl8.6.10 / Tk8.6.10)
The xml is well-formed, applying configuration
(HammerDB-4.1) 49 %

File: MySQL TPROC-C Timed Mode: Local Row.Col: 0.0

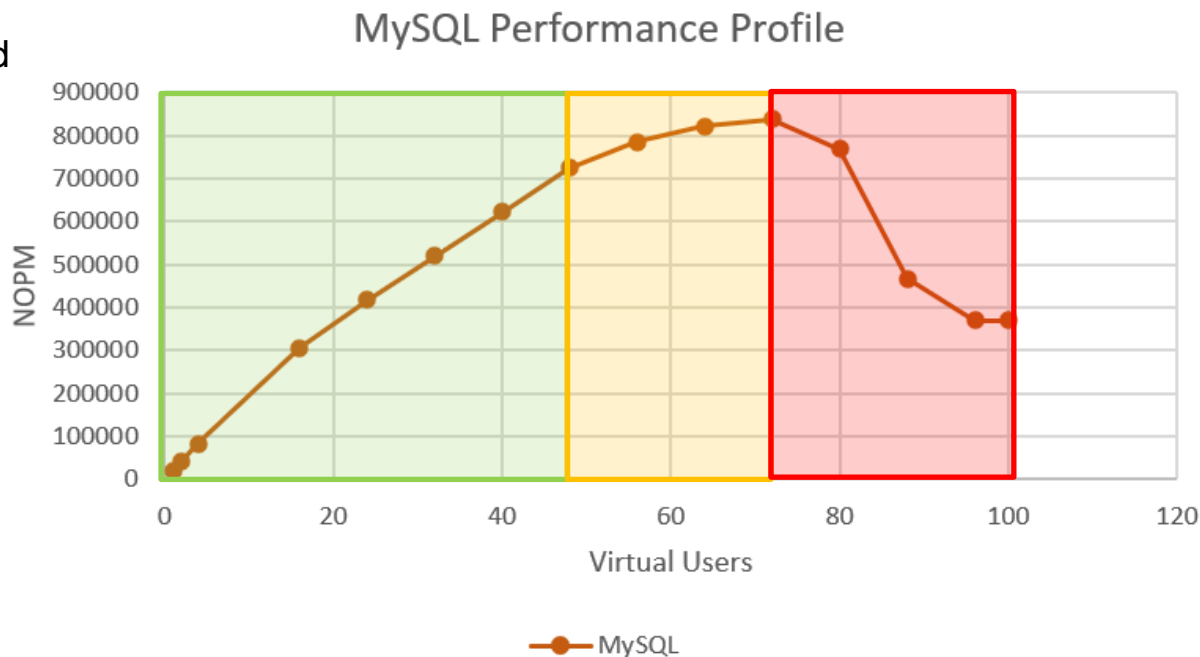
CLI Automation: Scripting

- CLI supports full TCL syntax
- Simple foreach loop for test sequence
- Can modify any parameters desired
- Log output

```
dbset db mysql
diset connection mysql_host localhost
diset connection mysql_port 3306
diset connection mysql_socket /tmp/mysql.sock
diset tpcc mysql_user root
diset tpcc mysql_pass mysql
diset tpcc mysql_driver timed
diset tpcc mysql_rampup 2
diset tpcc mysql_duration 5
loadscript
foreach z {1 2 4 8 12 16 20 24} {
  puts "$z VU TEST"
  vuset vu $z
  vucreate
  vurun
  runtimer 500
  vudestroy
}
```

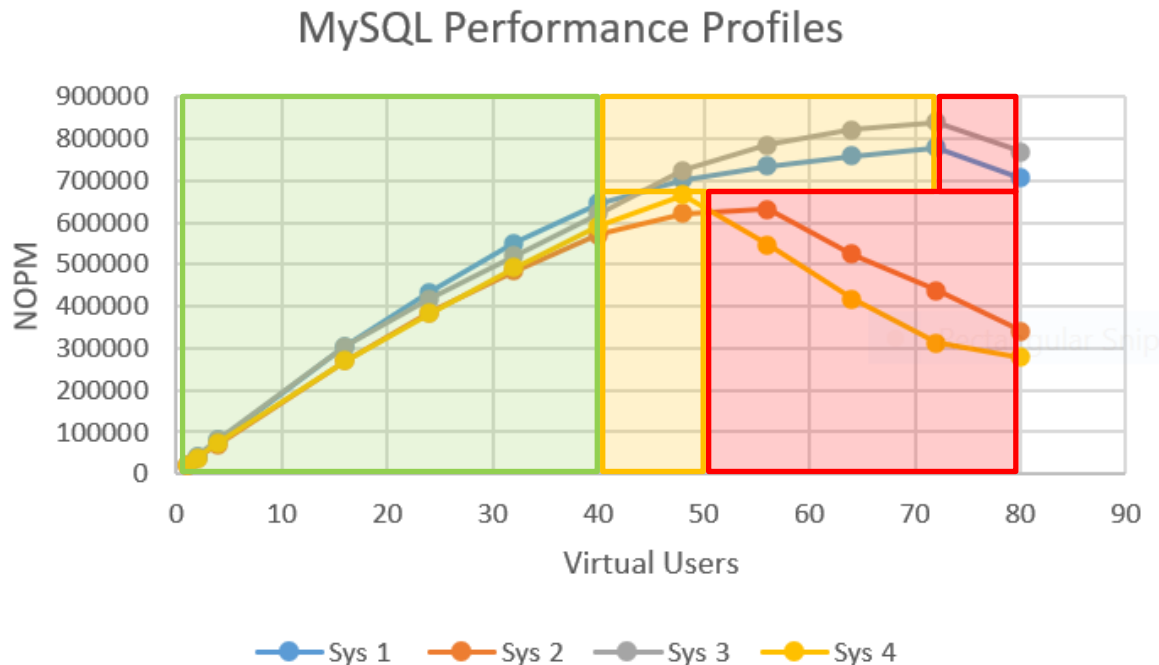
Performance Profiles

- Run Multiple Tests
 - Increasing Virtual User Load
- Example 56 (2 x 28) cores
- (Near) Linear Scale
 - Up to CPU cores/threads
 - Dependence on Database software
- Performance Plateau
 - Capture Peak Performance
 - Highest CPU Utilisation
- Contention
 - Increasing response times
 - Flat to lower performance



Comparing Performance

- Different Systems have different profiles
 - Not predictable on CPU Count
 - Database engines differ
- MySQL example
 - Linear scale is the same
 - Sys 1 + 3 extended performance plateau
 - Sys 2 + 4 show contention earlier
- Plan for differing levels of capacity



Advanced Testing Features

The background of the slide features abstract, wavy shapes in shades of orange and red. On the left side, there are overlapping orange shapes that curve upwards. On the right side, there are overlapping red and pink shapes that curve upwards, creating a sense of movement and depth.

Advanced Testing Features

- **Use All Warehouses**
 - Increase physical I/O to the data area
- **Connect Pooling**
 - Direct parts of the workload to different nodes in the same cluster
 - For example read/write and read-only nodes
- **Event Driven Scaling**
 - Co-routine based
 - Implements keying and thinking time
 - Scales to thousands of sessions
- **Time Profiling**
 - Capture Virtual User response times
- **Step Workloads**
 - Variable throughput by adding and removing Virtual Users
- **Advanced Features not mutually exclusive**
 - Can use some or all of the advanced features at the same time

Use All Warehouses

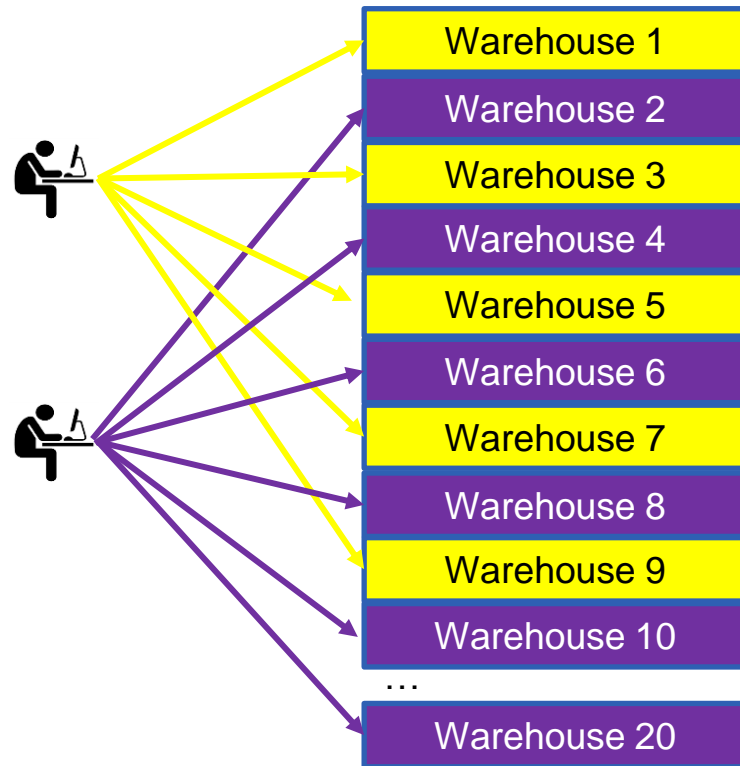
- All Warehouses divided between Virtual Users

- New warehouse selected per transaction
- More physical I/O

The screenshot shows the PostgreSQL TPROC-C benchmark interface. The top bar indicates 'RUNNING - PostgreSQL TPROC-C Timed'. The left sidebar contains a 'Benchmark' menu with options like 'TPROC-C', 'Schema Build', 'Driver Script', 'Virtual User', 'Options', 'Create', 'Run', 'Autopilot', 'Transactions', 'Metrics', 'Mode', and 'Datenen'. The main area displays the 'Script Editor' with three virtual users: Virtual User 1-MONITOR, Virtual User 2, and Virtual User 3. Below the script editor is a table showing the progress of these users.

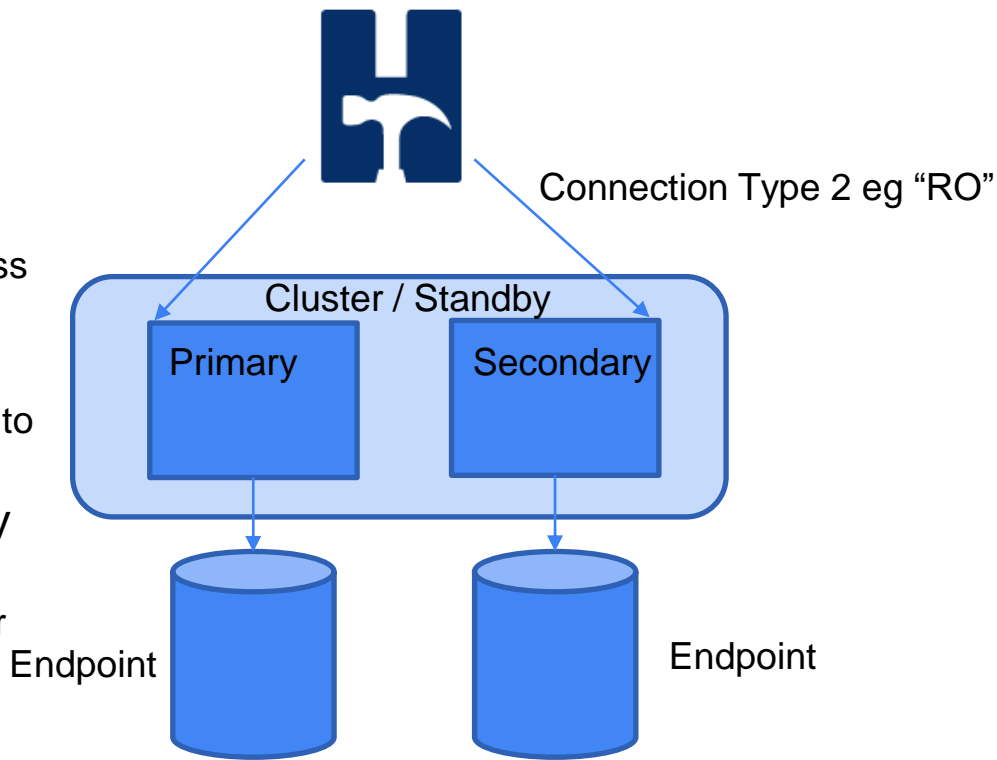
Virtual User	Iterations	Complete	Status
1	1	0	
2	1	0	
3	1	0	

At the bottom, a status bar shows 'File: PostgreSQL TPROC-C Timed Mode: Local Row.Col: 0.0'.



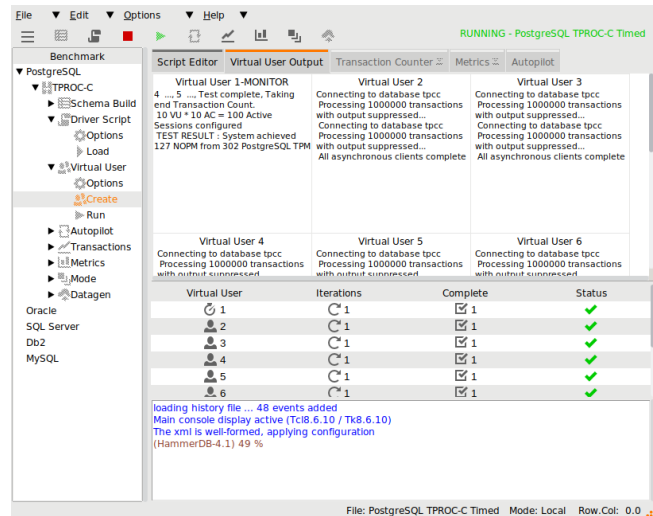
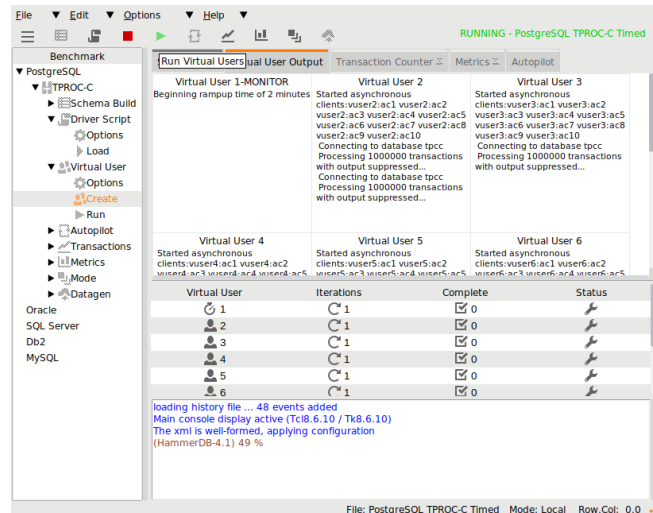
Connect Pooling for Clusters

- Define in XML Configuration
 - Multiple connections instances in cluster
 - Which transactions are directed to which nodes
 - Policy on how to allocate transactions across pool of connections eg round robin
- Example RW/RO nodes
 - Define RW transactions to primary and RO to standby
- Reports NOPM and TPM from Primary
 - Also reports client side TPM
 - Detailed view of transactions processed per node



Event Driven Scaling

- Default Workload is Cached
- Scaled Workloads
 - Large Session Counts
 - Fixed Throughput
 - Keying and Thinking Time delays
- Requires larger storage and networking
 - Requires middleware
 - HammerDB connects to middleware
 - Middleware connects to database
- Multiple Sessions per Virtual User
 - Uses co-routines to make key and think asynchronous
 - Appx 1 NOPM per session
- Example 1000 warehouses
 - 10,000 Sessions
 - 10,000 NOPM



Time Profiling for Response Times

- 2 Time Profiling Packages
 - Xtprof – all virtual users
 - Etprof – first active virtual user
- Xtprof
 - Profile of all virtual user response times
 - Summary of all virtual users

```
+-----+
>>>> SUMMARY OF 4 ACTIVE VIRTUAL USERS : MEDIAN ELAPSED TIME : 179324ms
>>>> PROC: NEWORD
CALLS: 253013 MIN: 0.414ms AVG: 1.375ms MAX: 121.556ms TOTAL: 347988.879ms
P99: 4.628ms P95: 2.553ms P50: 1.092ms SD: 1951.917 RATIO: 48.514%
>>>> PROC: PAYMENT
CALLS: 252926 MIN: 0.183ms AVG: 0.822ms MAX: 123.252ms TOTAL: 208144.667ms
P99: 3.319ms P95: 1.780ms P50: 0.544ms SD: 1855.444 RATIO: 29.018%
>>>> PROC: DELIVERY
CALLS: 25081 MIN: 0.589ms AVG: 2.663ms MAX: 34.680ms TOTAL: 66813.642ms
P99: 7.967ms P95: 4.789ms P50: 2.352ms SD: 1244.199 RATIO: 9.315%
>>>> PROC: SLEV
CALLS: 25412 MIN: 0.699ms AVG: 1.782ms MAX: 41.734ms TOTAL: 45289.974ms
P99: 4.685ms P95: 3.030ms P50: 1.594ms SD: 811.589 RATIO: 6.314%
>>>> PROC: OSTAT
CALLS: 25286 MIN: 0.149ms AVG: 0.696ms MAX: 123.539ms TOTAL: 17610.867ms
P99: 3.018ms P95: 1.556ms P50: 0.400ms SD: 2373.871 RATIO: 2.455%
+-----+
```

File Edit Options Help

COMPLETE

Benchmark

▼ PostgreSQL

▼ TPROC-C

► Schema Build

▼ Driver Script

Options

Load

▼ Virtual User

Options

Create

Run

Autopilot

Transactions

Metrics

Mode

Datagen

Oracle

SQL Server

Db2

MySQL

Script Editor

Virtual User 1-MONITOR

4 Active Virtual Users configured

TEST RESULT : System achieved 84735 NOMP from 194844

PostgreSQL TPM

Gathering timing data from Active Virtual Users... Calculating timings... Writing timing data to /tmp/hdbxtprofile.log

Transaction Counter

Virtual User 2

Initializing xtprof time profiler Processing 1000000 transactions with output suppressed...

Virtual User 3

Initializing xtprof time profiler Processing 1000000 transactions with output suppressed...

Virtual User 4

Initializing xtprof time profiler Processing 1000000 transactions with output suppressed

Virtual User 5

Initializing xtprof time profiler Processing 1000000 transactions with output suppressed

Virtual User	Iterations	Complete	Status
1	1	1	✓
2	1	1	✓
3	1	1	✓
4	1	1	✓
5	1	1	✓

loading history file ... 48 events added

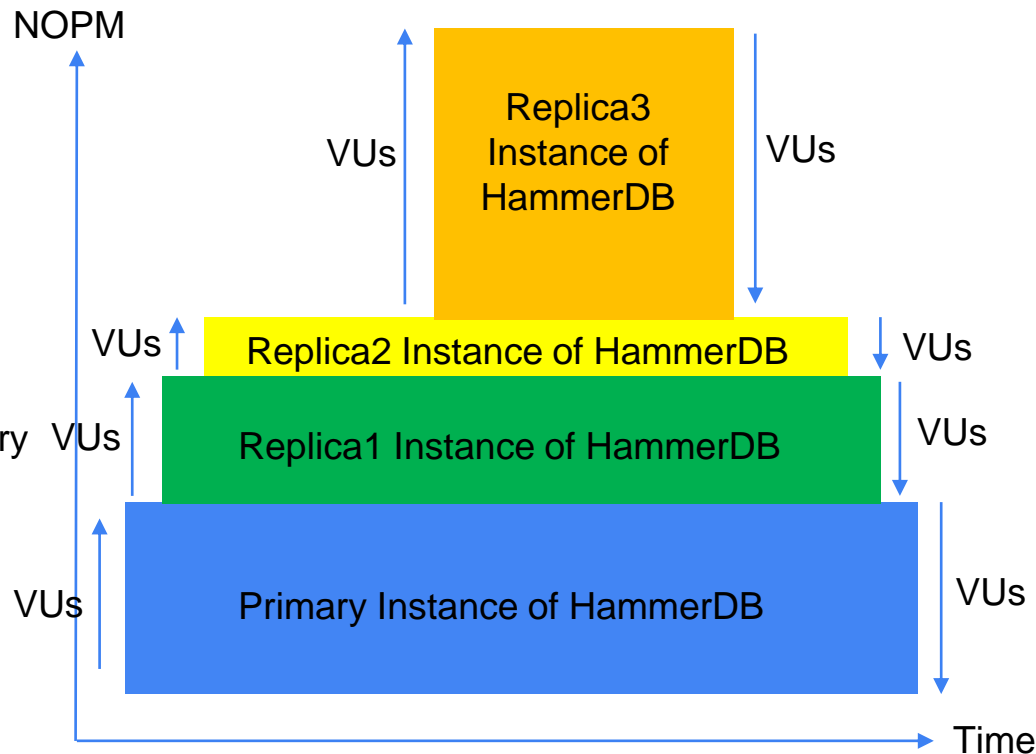
Main console display active (Tcl8.6.10 / Tk8.6.10)

The xml is well-formed, applying configuration (HammerDB-4.1) 49 %

File: PostgreSQL TPROC-C Timed Mode: Local Row.Col: 0.0

Variable Step Workloads

- **Creates Variable Load**
 - Define in XML
 - Pyramid of HammerDB Instances
- **Runs in CLI only**
 - Primary Instance of HammerDB
 - Replica instances created automatically and connect to primary
 - Timed delay of replica starts
- **Start with “steprun” command**
- **Measure response times**
 - Variations in performance

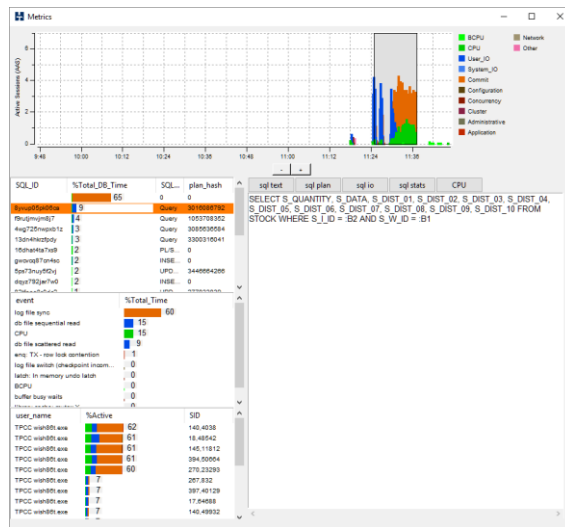
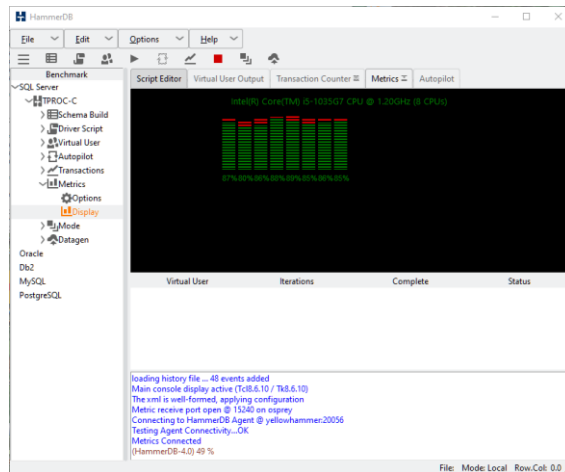


Performance Monitoring

The background of the slide features abstract, flowing shapes in shades of orange and red. On the left side, there are overlapping wavy shapes in various tones of orange. On the right side, there are similar wavy shapes in shades of red and pink, which appear to flow upwards and outwards, creating a sense of movement and depth.

CPU & Database Metrics

- HammerDB Graphical CPU Monitor
- Visualization of CPU load
- eg 50% CPU Average
 - Could be 50% of all cores at 50%
 - Could be 50% of cores at 100% and 50% at 0
- Detect CPU imbalance
- System & User CPU Utilization
- Identify Interrupt bottlenecks on individual cores
- GUI Database Metrics for PostgreSQL in progress



- MySQL 8.0.20+ recommended
 - Improved Lock Scheduling
- Monitor InnoDB storage engine
- Innotop

- MySQL 8.0.20+ recommended
 - Improved Lock Scheduling
- Monitor InnoDB storage engine
- Innotop

```
[RO] InnoDB Buffers (? for help)

Switch to a different mode:
A Dashboard          I InnoDB I/O Info      Q Query List
B InnoDB Buffers     K InnoDB Lock Waits    R InnoDB Row Ops
C Command Summary    L Locks                S Variables & Status
D InnoDB Deadlocks   M Replication Status    T InnoDB Txns
F InnoDB FK Err      O Open Tables           U User Statistics

Actions:
d Change refresh interval          p Pause innotop
i Toggle incremental status display q Quit innotop
n Switch to the next connection

Other:
TAB Switch to the next server group / Quickly filter what you see
! Show license and warranty         = Toggle aggregation
# Select/create server groups       @ Select/create server connections
$ Edit configuration settings      \ Clear quick-filters

Press any key to continue
```

```

[RO] Query List (7 for help)
localhost, 21h, 8.91M QPS, 144/128/0 con/run/cac thds, 8.0.22

When Load Cxns QPS Slow Se/In/Up/De% QCacheHit KCacheHit BpsIn BpsOut
Now 0.00 143 8.91M 0 9/ 3/ 5/ 0 0.00% 100.00% 27.76M 49.72M
Total 0.00 3.91k 151.96k 0 9/ 3/ 5/ 0 0.00% 100.00% 1.59M 846.54k

Cmd ID State User Host DB Time Query
Daemon 5 Waiting on empty q event_sc localhost 21:22:00
Query 691 updating root localhost tpcc 00:00 DELETE FROM new_order WHERE no_w_id = d_w_id AND no_d_id = d_d_id AND no_o_id = d_o_id
Query 692 statistics root localhost tpcc 00:00 SELECT SUM(ol_amount) INTO d_ol_total FROM order_line WHERE ol_o_id = d_no_o_id AND
Query 693 root localhost tpcc 00:00 CALL NEWORD(733,1000,10,1039,12,@edisc,@last,@credit,@dtax,@wtax,@next_o_id,str_to
Query 694 updating root localhost tpcc 00:00 UPDATE stock SET s_quantity = no_s_quantity WHERE s_i_id = no_ol_i_id AND s_w_id =
Query 695 Opening tables root localhost tpcc 00:00 INSERT INTO order_line (ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id, ol_supply_w
Query 696 statistics root localhost tpcc 00:00 SELECT c_discount, c_last, c_credit, w_tax INTO no_c_discount, no_c_last, no_c_cre
Query 697 updating root localhost tpcc 00:00 UPDATE order_line SET ol_delivery_d = timestamp WHERE ol_o_id = d_no_o_id AND ol_d
Query 698 root localhost tpcc 00:00 CALL PAYMENT(600,9,600,9,@p_c_id,1,1672,@p_c_last,@p_w_street,1,@p_w_street,2,@p_w
Query 699 executing root localhost tpcc 00:00 SELECT COUNT(DISTINCT (s_i_id)) INTO stock_count FROM order_line, stock WHERE ol_w
Query 700 System lock root localhost tpcc 00:00 UPDATE order_line SET ol_delivery_d = timestamp WHERE ol_o_id = d_no_o_id AND ol_d
Query 701 waiting for handle root localhost tpcc 00:00 COMMIT
Query 702 closing tables root localhost tpcc 00:00 SELECT c_discount, c_last, c_credit, w_tax INTO no_c_discount, no_c_last, no_c_cre
Query 703 statistics root localhost tpcc 00:00 SELECT c_discount, c_last, c_credit, w_tax INTO no_c_discount, no_c_last, no_c_cre
Query 704 executing root localhost tpcc 00:00 SELECT COUNT(DISTINCT (s_i_id)) INTO stock_count FROM order_line, stock WHERE ol_w
Query 705 updating root localhost tpcc 00:00 UPDATE district SET d_ytd = d_ytd + p_h amount WHERE d_w_id = p_w_id AND d_id = p
Query 706 closing tables root localhost tpcc 00:00 CALL NEWORD(630,1000,2,1797,14,@edisc,@last,@credit,@dtax,@wtax,@next_o_id,str_to
Query 707 executing root localhost tpcc 00:00 SELECT COUNT(DISTINCT (s_i_id)) INTO stock_count FROM order_line, stock WHERE ol_w
Query 708 init root localhost tpcc 00:00 CALL NEWORD(993,1000,4,2540,15,@edisc,@last,@credit,@dtax,@wtax,@next_o_id,str_to
Query 709 updating root localhost tpcc 00:00 UPDATE order_line SET ol_delivery_d = timestamp WHERE ol_o_id = d_no_o_id AND ol_d
Query 710 optimizing root localhost tpcc 00:00 SELECT count(c_id) INTO namecnt FROM customer WHERE c_last = p_c_last AND c_d_id =
Query 711 update root localhost tpcc 00:00 INSERT INTO new_order (no_o_id, no_d_id, no_w_id) VALUES (o_id, no_d_id, no_w_id)
Query 712 System lock root localhost tpcc 00:00 SELECT s_quantity, s_data, s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05

```

PostgreSQL

- PostgreSQL 13+ recommended
 - Improved throughput
- `pg_stat_statements` / `pg_sentinel`

%	AAS	backend_type	wait_event_type	wait_event
----	-----	-----	-----	-----
48	28.00	client backend	CPU	CPU
12	6.82	client backend	LWLock	XactSLRU
11	6.18	client backend	LWLock	WALInsert
9	5.41	client backend	IPC	ProcArrayGroupUpdate
6	3.71	client backend	Client	ClientRead
6	3.65	client backend	IPC	XactGroupUpdate
5	2.82	client backend	Lock	extend
2	0.94	client backend	LWLock	ProcArray
1	0.35	client backend	IPC	CheckpointDone

HPE LinuxKI

- LinuxKI Toolset
(Trace-based performance analysis tool)
- System level analysis
 - Beyond database only statistics

MySQL - HammerDB

1.3.3 Trace Events of Top 10 Processes

[\[Prev Subsection\]](#)[\[Next Subsection\]](#)---[\[Prev Section\]](#)[\[Next Section\]](#)[\[Table of Contents\]](#)

Analyzing Pid: [3180](#) Trace Records: 4327871 cmd: /usr/sbin/mysqld (mysqld)

Freq	Percent	Trace_type	64bit	ElapsedT	Max	Ave	Errors
1898956	43.88%	sys_enter					
1307788	30.22%	sched_yield	1	0.713	0.0044	0.000001	0
469256	10.84%	futex	1	3.808	0.0037	0.000008	1417
340574	7.87%	sched_wakeup					
188886	4.36%	sched_switch					
121911	2.82%	nanosleep	1	9.256	0.0067	0.000076	0
499	0.01%	hardclock					

MySQL - Sysbench

1.3.3 Trace Events of Top 10 Processes

[\[Prev Subsection\]](#)[\[Next Subsection\]](#)---[\[Prev Section\]](#)[\[Next Section\]](#)[\[Table of Contents\]](#)

Analyzing Pid: [6134](#) Trace Records: 1786621 cmd: /usr/sbin/mysqld (mysqld)

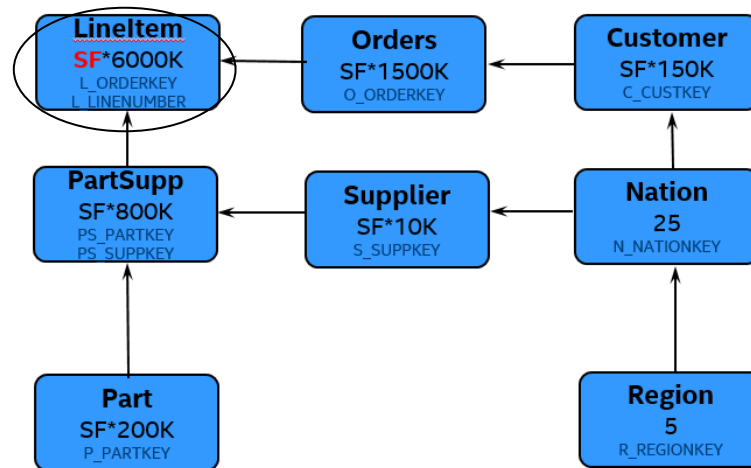
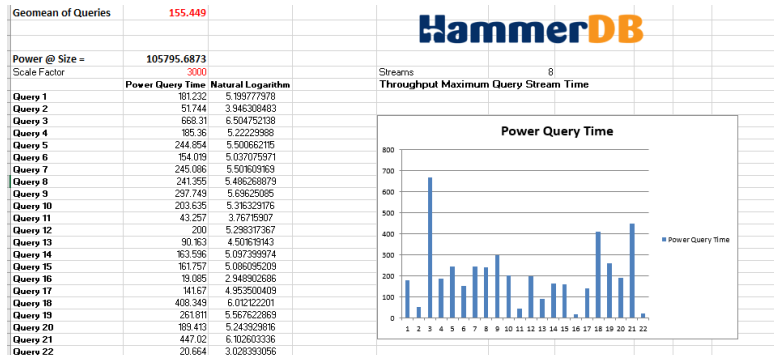
Freq	Percent	Trace_type	64bit	ElapsedT	Max	Ave	Errors
742430	41.55%	sys_enter					
444924	24.90%	recvfrom	1	0.578	0.0008	0.000001	148226
151720	8.49%	sched_wakeup					
149126	8.35%	sched_switch					
148348	8.30%	sendto	1	0.365	0.0005	0.000002	0
148226	8.30%	ppoll	1	5.559	0.0055	0.000038	0
914	0.05%	hardclock					
821	0.05%	futex	1	0.085	0.0054	0.000104	146
111	0.01%	sched_yield	1	0.002	0.0004	0.000021	0

Next Steps

The background features abstract, wavy shapes in shades of orange and red. On the left, there are overlapping orange shapes. On the right, there are overlapping red and pink shapes that curve upwards towards the top right corner.

Analytic Testing

- TPROC-H for Analytics
- Cloud Queries
- Stream of 22 Complex Queries
- PostgreSQL Parallel Query
- Columnstores
- More complex skillset required



Published Benchmarks

- Has someone already done a performance study you can use?
 - <https://www.hammerdb.com/benchmarks.html>
- Have you published your findings for other people to use?
- Making database performance data open source benefits all



[Home](#) [About](#) [Download](#) [Documentation](#) **[Benchmarks](#)** [Support](#) [GitHub](#) [Stats](#) [Blog](#)

HammerDB Blog

Automating CLI Tests on Windows
The information in this post is a duplicate of this GitHub Issue <https://github.com/TPC-Council/HammerDB/issues/84>. The ...

How to Graph HammerDB Response Times
The HammerDB workload derived from TPC-C contains a feature to record the response times for the stored procedures that ...

HammerDB v3.3 event driven scaling
HammerDB v3.3 includes a new feature called event driven scaling to enable the scaling of virtual users to thousands of ...

[rss feed](#)

HammerDB Benchmarks

2021

- [Accelerated Database Performance on Red Hat Enterprise Linux 8.3 with Intel Ice Lake](#)
- [Design Guide — MySQL InnoDB Cluster on Dell EMC PowerStore T](#)
- [Ecosystem Readiness for 3rd Generation AMD EPYC Processors](#)
- [Analyze Microsoft SQL Server Databases up to 1.38x as Fast with Amazon™ EC2 M5n Instances Featuring 2nd Gen Intel® Xeon® Scalable Processors](#)
- [How to get the biggest bang for your buck with SQL Server on Azure VMs](#)
- [Understanding Clones in VMware vSphere 7](#)
- [Performance Optimizations in VMware vSphere 7.0 U2 CPU Scheduler for AMD EPYC Processors](#)
- [Improve PostgreSQL® Database Performance by up to 1.21x with Google Cloud™ N2 Virtual Machine Instances Featuring 2nd Gen Intel® Xeon® Scalable Processors](#)
- [Accelerate Oracle Database with the Next-Gen FlashArray/iX](#)
- [Improve MySQL™ Database Performance by up to 1.21x with Google Cloud™ N2 High-Memory Virtual Machines Featuring 2nd Gen Intel® Xeon® Scalable Processors](#)
- [Improve MySQL™ Database Performance up to 1.66x with Amazon™ EC2 M5n Instances Featuring 2nd Gen Intel® Xeon® Scalable Processors](#)
- [On MySQL™ Workloads, New Microsoft® Azure® Dv4 Virtual Machines with 2nd Gen Intel® Xeon® Scalable Processors Outperformed Dv3 VMs by up to 1.63x](#)
- [Get 1.63x More MySQL™ Performance by Selecting Newer Microsoft® Azure® Edv4 Virtual Machines Featuring 2nd Gen Intel® Xeon® Scalable Processors](#)
- [Handle up to 1.64x the MySQL™ Database Transactions Per Minute with Amazon™ EC2 R5 Instances Featuring 2nd Gen Intel® Xeon® Scalable Processors](#)
- [AMD EPYC™ 7003 Series CPUs Set New Standard as Highest Performance Server Processor](#)
- [Better performance for less: AWS continues to beat Azure on SQL Server price/performance](#)
- [Deliver better performance for transactional database workloads at a lower cost by choosing an Amazon EC2 R5b instance](#)
- [Benchmarking PostgreSQL with NOPM: The Daily 500 Users](#)
- [Understanding Clones in VMware vSphere 7](#)
- [Running Microsoft SQL Server 2019 on OpenShift using Red Hat OpenShift Container Storage](#)
- [Benchmarking Amazon RDS Graviton2](#)
- [ESG Technical Validation: Maximize Database Efficiency and Performance in a VMware Environment using 32G NVMe over Fibre Channel](#)
- [AMD Powered Supremio A+ Servers Accelerate Oracle Database 19c Performance](#)

Tweets by @hammerdbresult

HammerDB Retweeted

Louis Imershein
@LouisPTC

With #SQLServer, "The HammerDB OLTP performance on Ice Lake benefited from larger L3 cache and higher CPU counts, achieving between 35-55% better transactions per minute (TPM) than the Cascade Lake at higher user counts." #sqlfriends sprout11RNPDpy5aVr


Accelerated Database P...
This post compares the p...
[radhat.com](#)

[Embed](#) [View on Twitter](#)

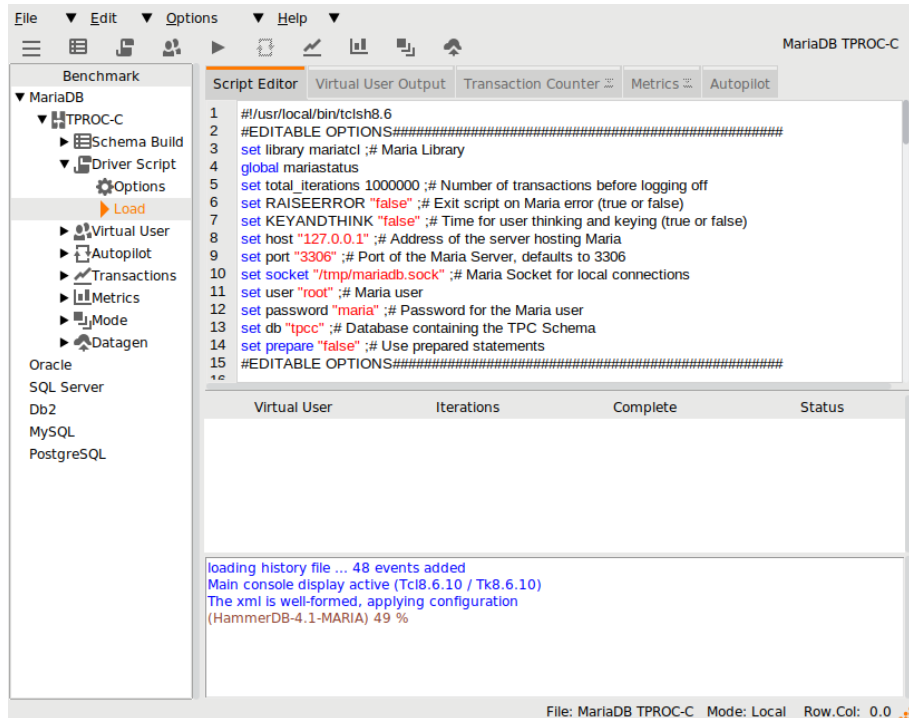
Up Next v4.2

Add MariaDB as a separate database #54 #234

New issue

 Open Jiang-Hua wants to merge 10 commits into TPC-Council:master from Jiang-Hua:master 

- MariaDB as separate database
 - TPROC-C and TPROC-H
 - Current support requires MySQL client library
 - Future support MariaDB client
 - Opportunity to diversify workload



The screenshot shows the MariaDB TPROC-C benchmark interface. The left sidebar contains a tree view with the following structure:

- Benchmark
 - ▼ MariaDB
 - ▼ TPROC-C
 - Schema Build
 - ▼ Driver Script
 - Options
 - Load
 - Virtual User
 - Autopilot
 - Transactions
 - Metrics
 - Mode
 - Datagen
 - Oracle
 - SQL Server
 - Db2
 - MySQL
 - PostgreSQL

The main window is titled "MariaDB TPROC-C" and contains a "Script Editor" tab. The script editor shows a configuration script for the benchmark. The script includes the following lines:

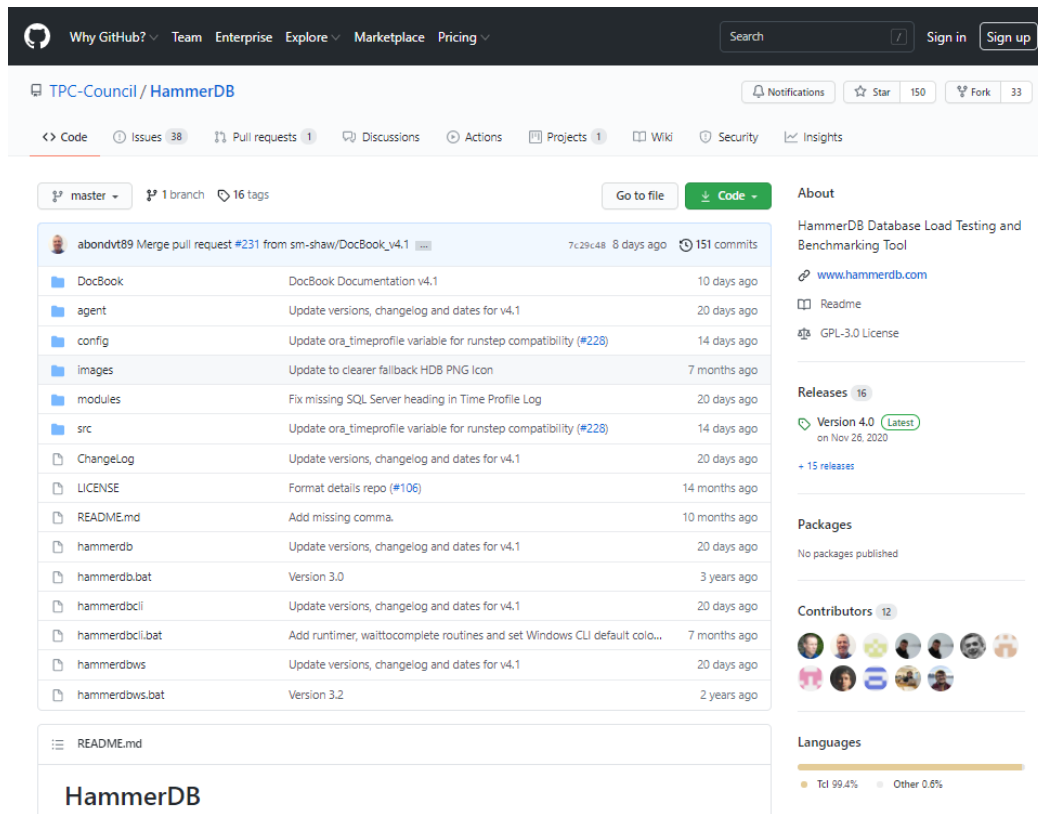
```
1 #!/usr/local/bin/tclsh8.6
2 #EDITABLE OPTIONS#####
3 set library mariatcl ;# Maria Library
4 global mariastatus
5 set total_iterations 1000000 ;# Number of transactions before logging off
6 set RAISEERROR "false" ;# Exit script on Maria error (true or false)
7 set KEYANDTHINK "false" ;# Time for user thinking and keying (true or false)
8 set host "127.0.0.1" ;# Address of the server hosting Maria
9 set port "3306" ;# Port of the Maria Server, defaults to 3306
10 set socket "/tmp/mariadb.sock" ;# Maria Socket for local connections
11 set user "root" ;# Maria user
12 set password "maria" ;# Password for the Maria user
13 set db "tpcc" ;# Database containing the TPC Schema
14 set prepare "false" ;# Use prepared statements
15 #EDITABLE OPTIONS#####
```

Below the script editor is a table with the following columns: Virtual User, Iterations, Complete, and Status. The table is currently empty.

At the bottom of the interface, there is a status bar that reads: "File: MariaDB TPROC-C Mode: Local Row.Col: 0.0".

Contribute to HammerDB on GitHub

- Contribute to HammerDB
- All source code open source
- Documentation open source
 - Docbook format
 - Edit with any XML editor
- Issues
- Discussions
- Binary releases



The screenshot displays the GitHub repository page for **TPC-Council/HammerDB**. The repository is in the **master** branch and has 16 tags. The file list includes:

File/Folder	Description	Last Commit
DocBook	DocBook Documentation v4.1	10 days ago
agent	Update versions, changelog and dates for v4.1	20 days ago
config	Update ora_timeprofile variable for runstep compatibility (#228)	14 days ago
images	Update to clearer fallback HDB PNG icon	7 months ago
modules	Fix missing SQL Server heading in Time Profile Log	20 days ago
src	Update ora_timeprofile variable for runstep compatibility (#228)	14 days ago
Changelog	Update versions, changelog and dates for v4.1	20 days ago
LICENSE	Format details repo (#106)	14 months ago
README.md	Add missing comma.	10 months ago
hammerdb	Update versions, changelog and dates for v4.1	20 days ago
hammerdb.bat	Version 3.0	3 years ago
hammerdbcli	Update versions, changelog and dates for v4.1	20 days ago
hammerdbcli.bat	Add runtime, waittocomplete routines and set Windows CLI default colo...	7 months ago
hammerdbws	Update versions, changelog and dates for v4.1	20 days ago
hammerdbws.bat	Version 3.2	2 years ago

The sidebar on the right provides additional information:

- About:** HammerDB Database Load Testing and Benchmarking Tool. Website: www.hammerdb.com. License: GPL-3.0 License.
- Releases:** 16 releases. Latest release: **Version 4.0** (Latest) on Nov 26, 2020.
- Packages:** No packages published.
- Contributors:** 12 contributors.
- Languages:** Tcl 99.4%, Other 0.6%.

Thanks!

Any questions?

THANK YOU !



PERCONA
LIVEONLINE
MAY 12 - 13th
2021