



DATABASE BENCHMARKING

An unexpected Journey



Dirk Krautschick
PGDay Austria 04.09.2025

#whoami

Dirk Krautschick **Senior Solution Architect**

with Aiven since Nov 2023



18 years

DBA, Trainer, Consulting, Sales Engineering

PostgreSQL, Oracle, Kafka, Clickhouse, OpenSearch,...

Married, 2 Junior DBAs

Mountainbike, swimming, movies, music,
hifi/home cinema, 8 bit computing 

PostgreSQL User Group NRW

North Rhine-Westphalia, Germany

Founded Dec 2023

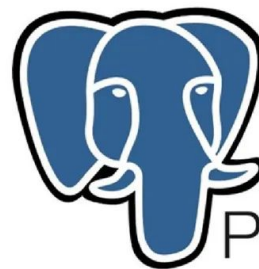
Feb 2024 Cologne, May 2024 Aachen, Oct 2024 Essen,

Feb 2025 Paderborn, Jun 2025 M'Gladbach,...

9th Oct 2025 Dortmund (Apps Associates)

Upcoming events in the pipeline, stay tuned!

Target is at least 3 meetups a year, all around NRW



PostgreSQL



Disclaimer

Inspiration and motivation for trying

I am NOT an deep dive benchmarking guru ...

... just a desperate database guy ...like you!? :-)

Unexceptional Open Source!!!

Everything what looks like advertising won't be meant like such...

...but this time I need to give some brief context of my company
to explain my journey...

One cloud data platform

STREAM

Event streaming



Aiven for
Apache Kafka®
and Kafka® Connect

STORE

Relational databases



Aiven for
PostgreSQL® Aiven for
MySQL

Key-value database



Aiven for
Valkey Aiven for
Dragonfly

Data warehouse



Aiven for
ClickHouse®

Time series database



Aiven for
Metrics

Search engine



Aiven for
OpenSearch®

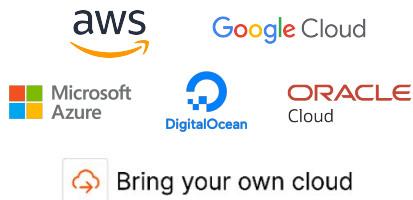
Data visualization



Aiven for
Grafana®

UNIFIED PLATFORM

Host



Deploy



Integrate



One cloud data platform

STREAM

Event streaming



Aiven for
Apache Kafka®
and Kafka® Connect

STORE

Relational databases



Aiven for
PostgreSQL®
Aiven for
MySQL

Key-value database



Aiven for
Valkey
Aiven for
Dragonfly

Data warehouse



Aiven for
ClickHouse®

Time series database



Aiven for
Metrics

Search engine



Aiven for
OpenSearch®

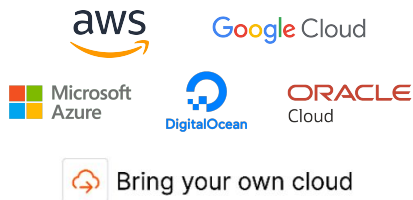
Data visualization



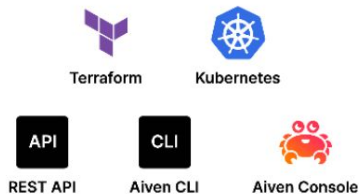
Aiven for
Grafana®

UNIFIED PLATFORM

Host



Deploy



Integrate



The Request

Can we do a full comparison of the performance of PostgreSQL ...
...everywhere!!!

Several t-shirt sizes... ("define yourself something what make sense")

All major cloud vendors

PostgreSQL v15, v16, v17,...

Hyperscaler offerings (AWS RDS, Azure FlexServer, GCP CloudSQL,...)

"What about things like e.g. AWS Aurora, GCP AlloyDB,...?"

The Request

Can we do a full comparison of the performance of PostgreSQL ...
...everywhere!!!

Several t-shirt sizes... ("define yourself something what make sense")

All major cloud vendors

PostgreSQL v15, v16, v17,...

Hyperscaler offerings (AWS RDS, Azure FlexServer, GCP CloudSQL,...)

"What about things like e.g. AWS Aurora, GCP AlloyDB,...?"

...???



Measurement

Well....what does "performance" means in that context?

How to measure that "performance" sustainable and reproducible?

What is the best workload using for those measurements?

Once upon a time....

...somewhere in Mönchengladbach, Germany,
...around 1997-1999,
...during my electronics engineer apprenticeship

"Wer misst, misst Mist!"

Udo Brockers, R.I.P. (* 05.09.1959, + 25.06.2012)

1:1 Translation:

"Who measures, measures manure!"



Using performance analytic skills?

One idea was using regular performance analysis

Investigating same load on test targets

Relying on existing metric sources (pg_stat_statements, pg_wait_sampling,...etc.)

Creating profiling reports with PG_PROFILE with comparing option

<https://www.youtube.com/watch?v=I57TNi6Y728>

But again, which workload?



How to measure ideally?

What do we want?

- Absolute results

- Comparing results

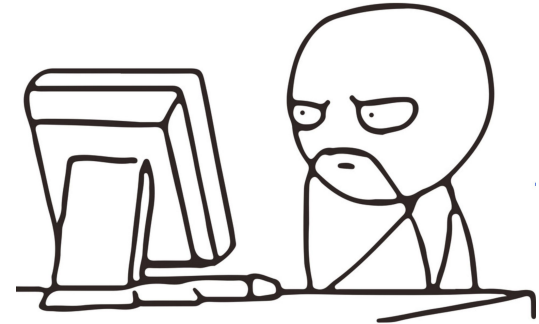
What are the metrics?

How do we want to measure at all?

- Use case specific measurements

- Application based measurements

- Generic measurements



Let's introduce... pgbench

Ähm...

not this "pgbench" :-)



Let's introduce... pgbench



pgbench

In contrib with PostgreSQL v7.0

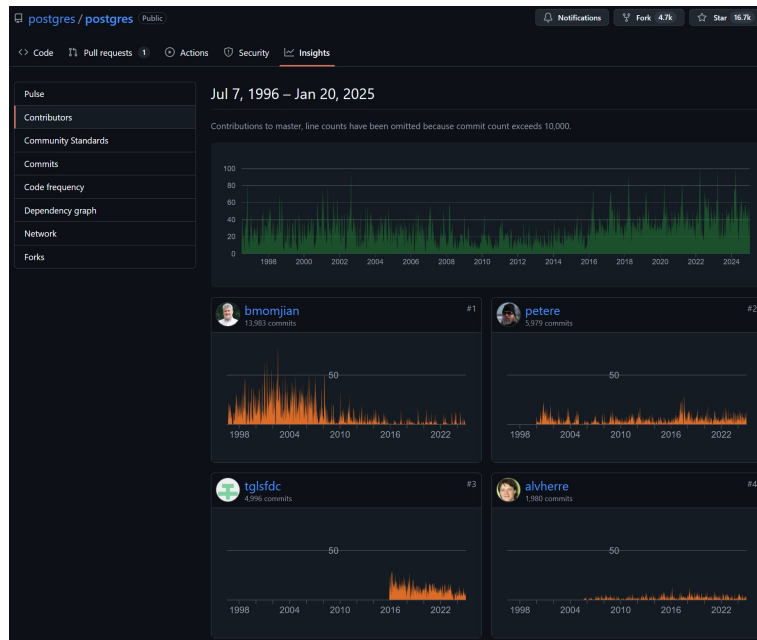
With PostgreSQL at least since 8.2

PostgreSQL License

<https://github.com/postgres/postgres/tree/master/src/bin/pgbench>

Actual Release v17.5 (May 2025)

C-based



Let's introduce... pgbench

Methodology

So called "TPC-B sort of" based scenario!?

5 select, update and insert command per transaction

4 Tables with the amount or rows (scale factor 1)

table	# of rows
pgbench_branches	1
pgbench_tellers	10
pgbench_accounts	100000
pgbench_history	0

 pgbench_accounts	 pgbench_tellers
123 aid	123 tid
123 bid	123 bid
123 abalance	123 tbalance
A-Z filler	A-Z filler
 pgbench_branc...	 pgbench_history
123 bid	123 tid
123 bbalance	123 bid
A-Z filler	123 aid
	123 delta
	 mtime
	A-Z filler

pgbench – An easy quick start...

Installation

...please....if you don't already have PostgreSQL somewhere...?

GET POSTGRESQL!!!!!!!!!!!!111eleven

<https://www.postgresql.org/download/>



pgbench – An easy quick start...

Initialization of pgbench

```
# $PGHOME/bin/pgbench -i
```

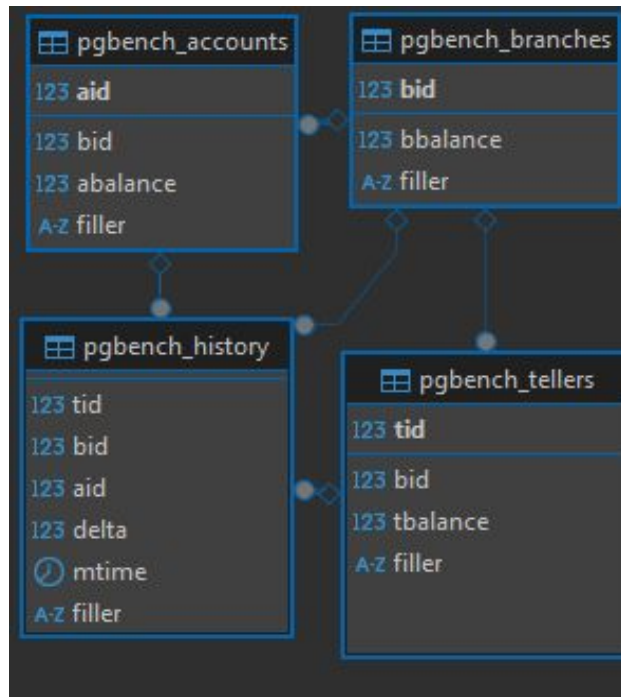
(if necessary, connection parameters like usual)

Consider some parameters....

foreign keys (`--foreign-keys`)

scale factor (`--scale, -s`)

partitioning (`--partition-method, -partitions`)



pgbench - Demo

DEMO

pgbench – An easy quick start...

Run the benchmark

```
# $PGHOME/bin/pgbench
```

```
(if necessary, connection parameters like usual)
```

Consider some parameters....

time or transactions (`--time`, `--transactions`)

clients (`--client`)

threads (`--jobs`)

custom scripts

...

<https://www.postgresql.org/docs/current/pgbench.html>



pgbench – Example script

```
#!/bin/bash
# redirect stdout/stderr to a file
DATE=$(date +"%Y%m%d%H%M")
exec >benchrun$DATE.log 2>&1
```

```
SCALE=100
CLIENT_CNT=50
BENCH_TIME=300
THREADS=10
```

```
echo "pgbench -i -s $SCALE"
echo "pgbench -c $CLIENT_CNT -j $THREADS -P 60 -r -T $BENCH_TIME "
echo "-----"
echo $(date -u)
echo "----- INIT -----"
pgbench -i -s $SCALE "postgres://<USER>:<PASSWORD>@<HOST>:<PORT>/<DATABASE>"
echo $(date -u)
echo "----- PGBENCH -----"
pgbench -c $CLIENT_CNT -j $THREADS -P 60 -r -T $BENCH_TIME "postgres://<USER>:<PASSWORD>@<HOST>:<PORT>/<DATABASE>"
echo $(date -u)
```

pgbench – The result explained

```
----- INIT -----  
dropping old tables...  
creating tables...  
generating data (client-side)...  
vacuuming...  
creating primary keys...  
done in 649.74 s ( drop tables 0.00 s,  
                  create tables 0.01 s,  
                  client-side generate 478.74 s,  
                  vacuum 0.87 s,  
                  primary keys 170.13 s).
```

pgbench – The result explained

```
----- PGBENCH -----  
pgbench (17.5)  
starting vacuum...end.  
transaction type: <builtin: TPC-B (sort of)>  
scaling factor: 2500  
query mode: simple  
number of clients: 32  
number of threads: 2  
maximum number of tries: 1  
duration: 300 s  
number of transactions actually processed: 922031  
number of failed transactions: 0 (0.000%)  
latency average = 10.389 ms  
latency stddev = 91.565 ms  
initial connection time = 360.173 ms  
...
```

pgbench – The result explained

...

tps = 3076.984799 (without initial connection time)

statement latencies in milliseconds and failures:

0.000	0	\set aid random(1, 100000 * :scale)
0.000	0	\set bid random(1, 1 * :scale)
0.000	0	\set tid random(1, 10 * :scale)
0.000	0	\set delta random(-5000, 5000)
0.353	0	BEGIN;
7.374	0	UPDATE pgbench_accounts SET abalance = abalance + ...
0.517	0	SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
0.605	0	UPDATE pgbench_tellers SET tbalance = tbalance + ...
0.557	0	UPDATE pgbench_branches SET bbalance = bbalance + ...
0.569	0	INSERT INTO pgbench_history (tid, bid, aid, delta, ...
0.406	0	END;

Benefits and limitations of pgbench

It's just there, it's easy, it works!

Quick example load for testing, with many *(often unused!)* options

Missing tooling and utilization

What about

- Other specific workloads?

- OLAP/Analytics?

- Other database engines?

The wise elders of benchmarking...



Brief History of DB benchmarking...

- 1985 Debit/Credit (Jim Gray)
 "A Measure of Transaction Processing Power"
 <https://infolab.usc.edu/csci599/Fall2008/papers/c-2a.pdf>
 First database benchmark approach
- 1988 Transaction Processing Performance Council (TPC) was founded
 <https://www.tpc.org>
- 2000 PostgreSQL 7 (with pgbench in contrib)
- 2006 PostgreSQL 8.2 (with pgbench included)
- 2003 HammerDB 1.0

What the heck is that TPC???



Transaction Processing Performance Council

<https://www.tpc.org>

Non-profit corporation since 1988

Definition of industry standards for benchmarking

1989 TPC-A (Debit/Credit) and **1990 TPC-B** (DB version of TPC-A)

1992 TPC-C

1999 TPC-H

Several replacements

<https://www.tpc.org/information/about/history5.asp>



TPC...ok, nice...but...?



Large company/vendor driven

Expensive and provided reports not relevant anymore

But...

“Fair use” for research possible

Workload examples are adapted and/or established

Easy to use



Well, what other gear do we have?

sysbench

<https://github.com/akopytov/sysbench>

oltpbench (now BenchBase)

<https://github.com/oltpbenchmark/oltpbench> <https://github.com/cmu-db/benchbase>

pgbench-tools (soon “pgbent”)

<https://github.com/gregs1104/pgbench-tools>

YCSB (Yahoo! Cloud Serving Benchmark)

<https://github.com/brianfrankcooper/YCSB>

Let's introduce... HammerDB



HammerDB

Hosted by the TPC-Council

Steve Shaw to be mentioned

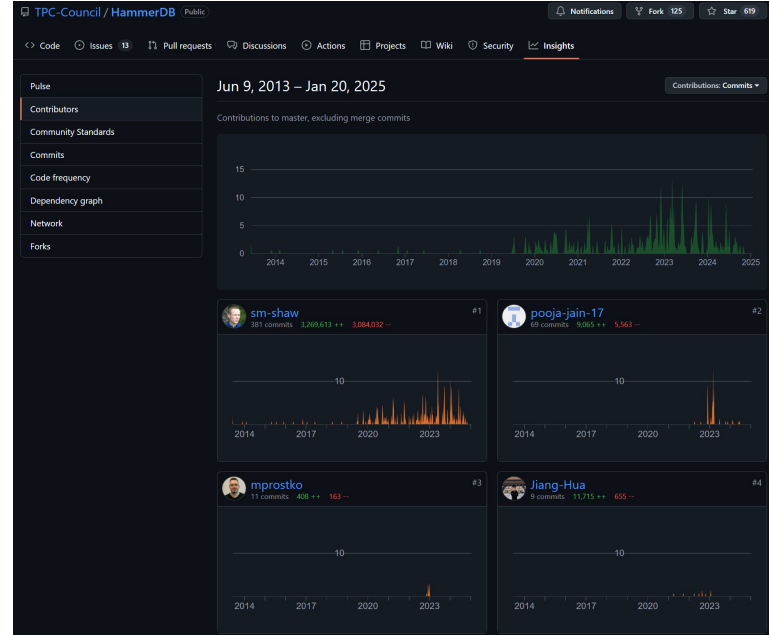
GNU License 3

<https://github.com/TPC-Council/HammerDB>

Release v1.0.0 (July 2003)

Actual Release v5.0 (Apr 2025)

Tcl based



Let's introduce... HammerDB

Databases

PostgreSQL

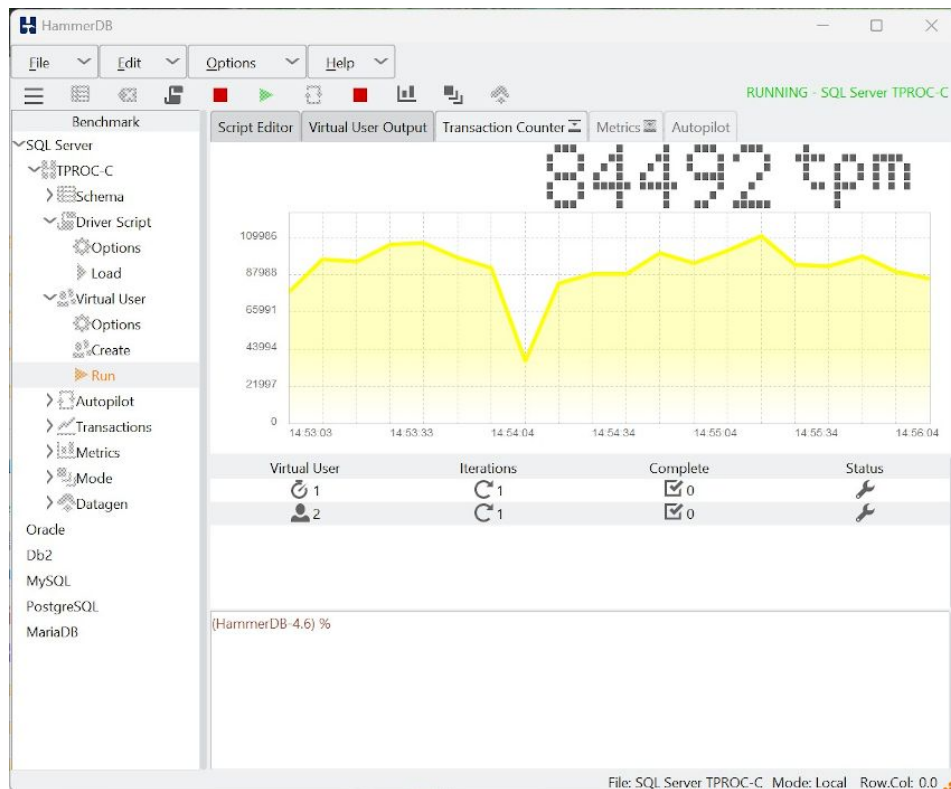
Oracle

SQL Server

DB2

MySQL

MariaDb



Let's introduce... HammerDB

Workloads

TPROC-C

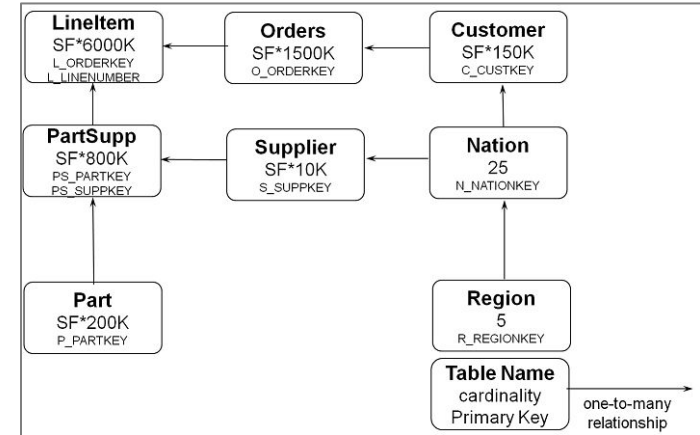
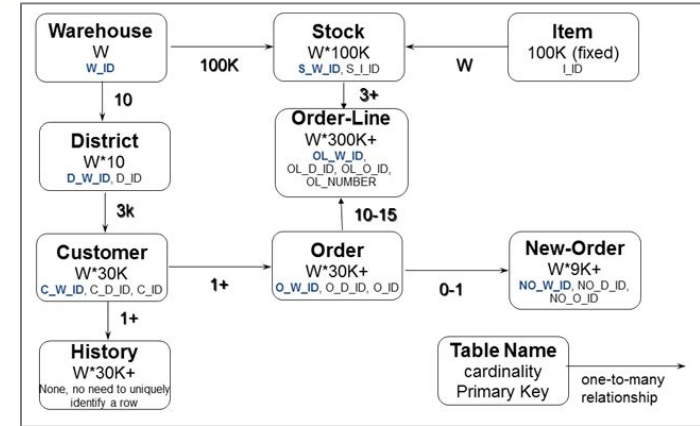
Based on TPC-C

classic OLTP

TPROC-H

Based on TPC-H

Analytics



Let's introduce... HammerDB

Warehouses

- like scaling factor in pgbench

- 1 wh = 100,000 items, 10 sales districts with 3000 customers each

- 5 warehouses are around 510 MByte (TPC-C)

Virtual Users

- like clients/transactions part like pgbench

HammerDB – An easy quick start...

Installation

<https://www.hammerdb.com/download.html>

```
# wget https://github.com/TPC-Council/HammerDB/releases/download/v5.0/HammerDB-5.0-Prod-Lin-RHEL8.tar.gz
```

```
/* (or select any other distribution... */
```

```
# tar -xzf HammerDB-5.0-Prod-Lin-RHEL8.tar.gz
```

Yeah....thats it!



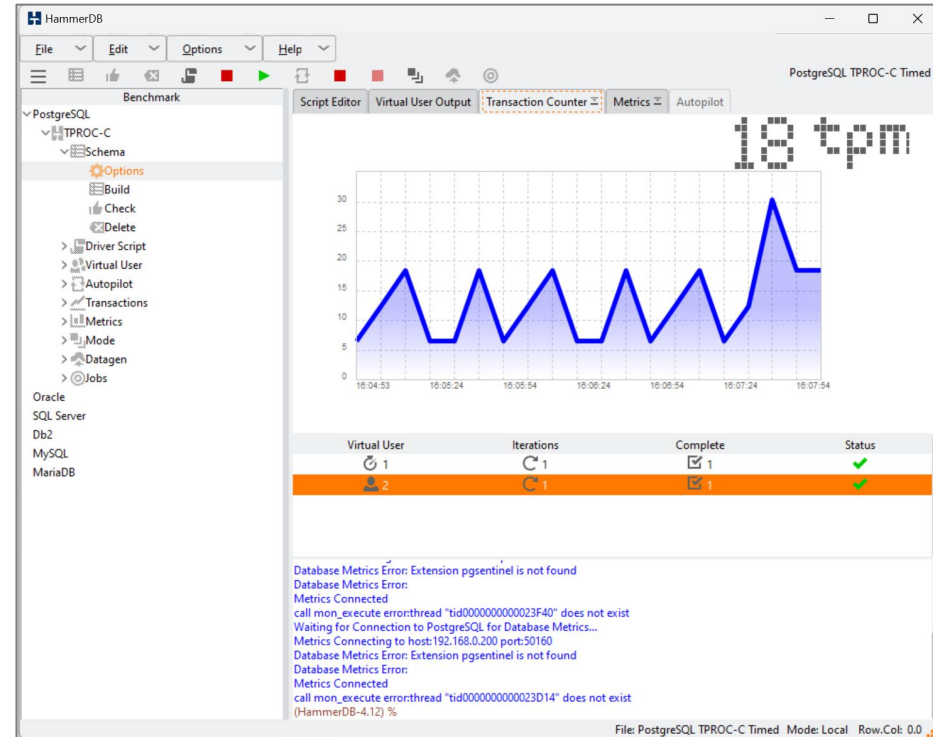
HammerDB – An easy quick start...

There is a GUI

Same on all OS

Easy to start with

But not that much intuitive IMHO :-)



HammerDB – An easy quick start...

There is a command line as well...

```
~/HammerDB-4.12 []# ./hammerdbcli  
HammerDB CLI v4.12  
Copyright (C) 2003-2024 Steve Shaw  
Type "help" for a list of commands  
Initialized new Jobs on-disk database /tmp/hammer.DB  
hammerdb> librarycheck
```

HammerDB – An easy quick start...

There is a command line as well...

```
~/HammerDB-4.12 []# ./hammerdbcli
HammerDB CLI v4.12
Copyright (C) 2003-2024 Steve Shaw
Type "help" for a list of commands
Initialized new Jobs on-disk database /tmp/hammer.DB
hammerdb> librarycheck
Checking database library for Oracle
Error: failed to load Oratcl - can't read "env(ORACLE_HOME)"
...
Checking database library for PostgreSQL
Success ... loaded library Pgtcl for PostgreSQL
Checking database library for MariaDB
...
hammerdb>
```



HammerDB

DEMO

HammerDB – An easy quick start...

Definition of the database type...

```
hammerdb> dbset db pg  
Database set to PostgreSQL
```

Definition of the workload...

```
hammerdb> dbset bm TPC-C  
Benchmark set to TPC-C for PostgreSQL
```

HammerDB – An easy quick start...

```
hammerdb> print dict
Dictionary Settings for PostgreSQL
connection {
  pg_host      = localhost
  pg_port      = 5432
  pg_sslmode   = prefer
}
tpcc          {
  pg_count_ware      = 1
  pg_num_vu           = 1
  pg_superuser       = postgres
  pg_superuserpass    = postgres
  pg_defaultdbase    = postgres
  pg_user             = tpcc
  pg_pass             = tpcc
  pg_dbase            = tpcc
  ...
}
```

HammerDB – An easy quick start...

Setting of the parameters...

```
hammerdb> diset connection pg_host 192.168.0.200
```

```
Changed connection:pg_host from localhost to 192.168.0.200 for PostgreSQL
```

```
hammerdb> diset connection pg_port 50178
```

```
Changed connection:pg_port from 5432 to 50178 for PostgreSQL
```

```
hammerdb> diset tpcc pg_num_vu 5
```

```
Changed tpcc:pg_num_vu from 1 to 5 for PostgreSQL
```

```
hammerdb> diset tpcc pg_count_ware 10
```

```
Changed tpcc:pg_count_ware from 1 to 10 for PostgreSQL
```

```
...
```

HammerDB – An easy quick start...

Creating the schema...

```
hammerdb> buildschema
Script cleared
Building 10 Warehouses with 6 Virtual Users, 5 active + 1 Monitor VU(dict value
pg_num_vu is set to 5)
Ready to create a 10 Warehouse PostgreSQL TPROC-C schema
in host 192.168.0.200:50178 sslmode PREFER under user TPCC in database TPCC?
Enter yes or no: replied yes
Vuser 1 created - WAIT IDLE
Vuser 2 created - WAIT IDLE
...
Vuser 1:GATHERING SCHEMA STATISTICS
Vuser 1:TPCC SCHEMA COMPLETE
Vuser 1:FINISHED SUCCESS
ALL VIRTUAL USERS COMPLETE
Schema Build jobid=6795743D62C903E243831353
hammerdb>
```

HammerDB – An easy quick start...

Let's roll...

```
hammerdb> vurun
Script loaded, Type "print script" to view
Vuser 1 created MONITOR - WAIT IDLE
Vuser 2 created - WAIT IDLE
2 Virtual Users Created with Monitor VU
Vuser 1:RUNNING
Vuser 1:DBVersion:17.2
Vuser 1:Beginning rampup time of 2 minutes
Vuser 2:RUNNING
...
Vuser 1:TEST RESULT : System achieved 20622 NOPM from 47596 PostgreSQL TPM
Vuser 1:FINISHED SUCCESS
Vuser 2:FINISHED SUCCESS
ALL VIRTUAL USERS COMPLETE
Benchmark Run jobid=6796AD8E62CA03E253833373
hammerdb>
```

About HammerDB results

NOPM (New orders per minute)

How fast are you going

Close relation to official tpmC

TPM (Transactions per minute)

How hard your engine is working

Comparison performance

NOPM can be compared between engines

TPM can only be compared across the same engine

TPM useful engineering metric to compare statistics

Back to my initial task...

Starting with pgbench

because it was just there and established! :-)

All scenarios

T-shirt size

Cloud vendor

Long run (60 min), short run (5 min)

My approach with pgbench

A “good enough” compute instance (VM)

- each cloud vendor

- same region/availability zone

- Preventing latency

All DBs with defaults

Scripted (based on the example) sequenced set of pgbench runs

To prevent abnormalities, >3 runs checking average result

Rough scale increase for t-shirt sizes (*4, *16, *32, *64)

- Scale (500, 1500, 2500 and 4500)

- Client count (4, 16, 32 and 64)

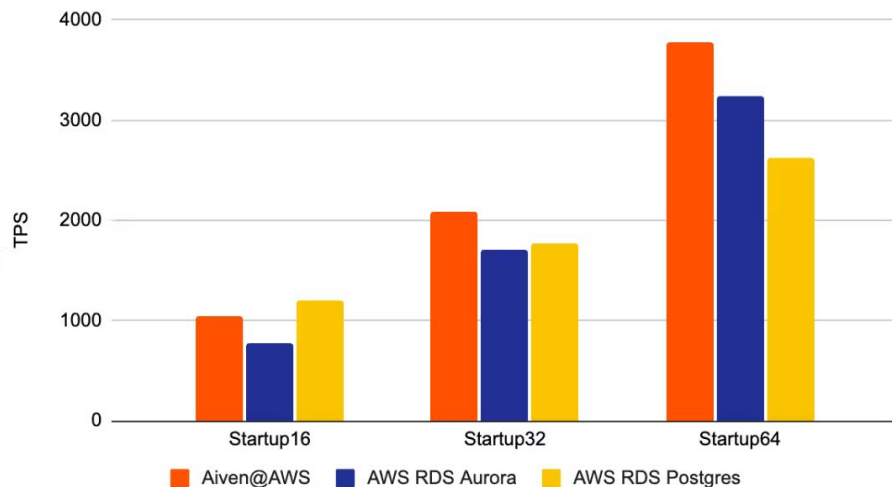
Some findings from me..

Huge differences between hyperscalers (compute instances)

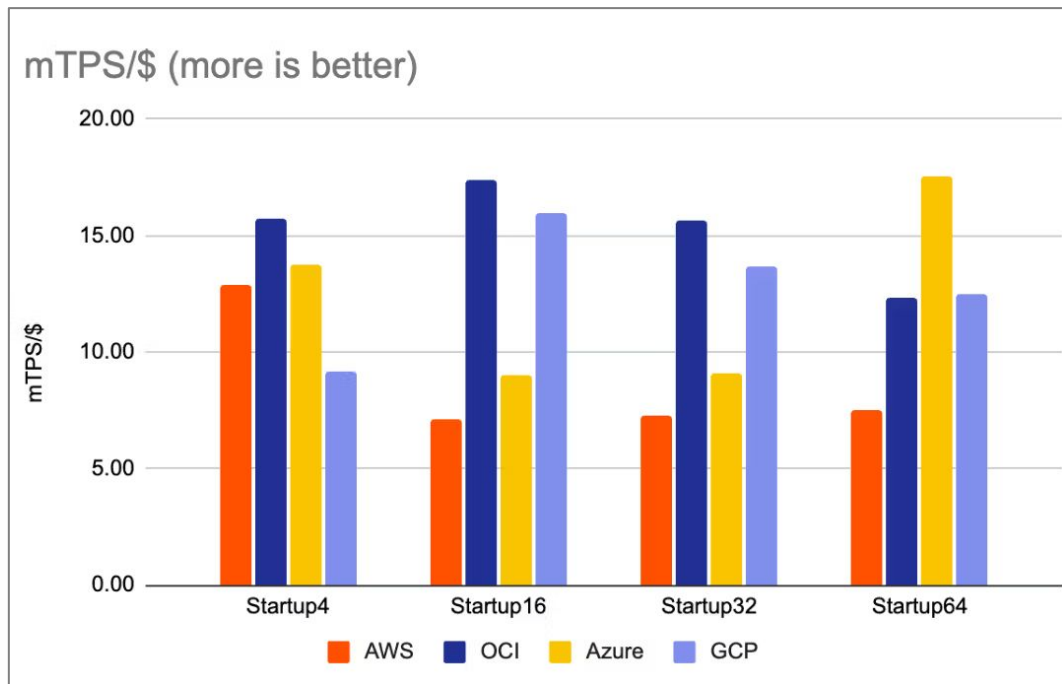
Transactions per second - startup16 plans



Transactions per Second - Aiven for Postgres and AWS RDS



More findings ...check the blog



<https://aiven.io/blog/aiven-for-postgresqlr-performance-benchmarks-across-cloud>



My next steps...

More research, long term results

Consolidating the results, creating blog post at Aiven

Switch to HammerDB

Analytical workload integration (e.g. to challenge AlloyDB Omni)

DB Compute instance verification and comparison

Digging deeper and more specific PG parameter comparison

Automatism, Scripting improvements

Collaboration with companies like dbtune, benchant, etc. ...

My next steps...

More research, long term results

Consolidating the results, creating blog post at Aiven

Switch to HammerDB

Analytical workload integration (e.g. to challenge AlloyDB Omni)

DB Compute instance verification and comparison

Digging deeper and more specific PG parameter comparison

Automatism, Scripting improvements

Collaboration with companies like dbtune, benchant, etc. ...

The journey continues...



DATABASE BENCHMARKING THE DESOLATION OF THE HYPERSCALERS

Dirk Krautschick
Coming soon ...



So, what are the takeaways?

Choose your tooling wisely

Effort vs. goals?

Always think about Udo! :-)

Measuring is not difficult

Context, comparison and structure is key

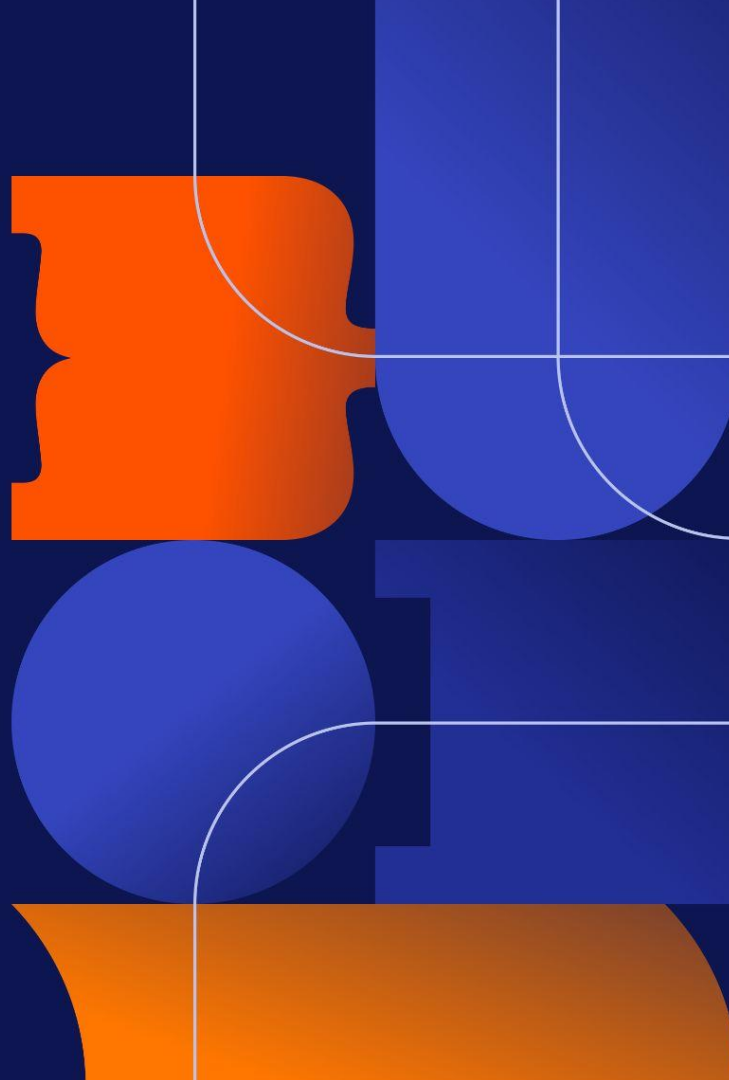
Define your objectives, metrics and foundations clearly...and up front!

Take care of reproduction...and DO reproductions



Questions?

**Please do ask or
catch me outside!**





Your Trusted Data & AI Platform

Streaming | Database Optimization |
Analytics | Search | Data Warehousing |
In-Memory Caching



One cloud data platform

STREAM

Event streaming



Aiven for
Apache Kafka®
and Kafka® Connect

STORE

Relational databases



Aiven for
PostgreSQL® Aiven for
MySQL

Key-value database



Aiven for
Valkey Aiven for
Dragonfly

Data warehouse



Aiven for
ClickHouse®

Time series database



Aiven for
Metrics

Search engine



Aiven for
OpenSearch®

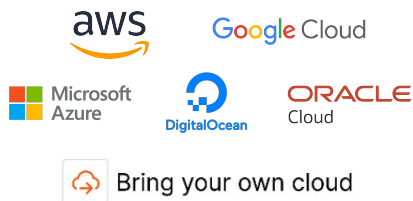
Data visualization



Aiven for
Grafana®

UNIFIED PLATFORM

Host



Deploy



Integrate



Customers

okta



snyk

 DOORDASH

priceline®

fiverr.

Norauto

DECATHLON

 GTL

ACTIVISION | BLIZZARD

 MIRAKL

 GOV.UK

goto financial

spare

Schibsted

 TOYOTA

 paf

ONRAD

adeo

ometria


WÄRTSILÄ