PostGreSQL Database Server Benchmarking : `pgbench`

Abdallah Ali Z.A. IBRAHIM

Mar, 2017

NADI

CSC | COMPUTER SCIENCE AND COMMUNICATIONS RESEARCH UNIT

UNIVERSITÉ DU LUXEMBOURG

Agenda

- **Why benchmark?**
- **Performance Tuning**
- **Benchmarking Basics**
  - A way for a successful DB Bench
  - DB Bench common mistakes
  - Standard Database Benchmarks
  - Common Database Benchmark Metrics
- **Tune PostgreSQL Performance**
- **pgbench: Benchmarking PostgreSQL**
  - Why pgbench
  - Install pgbench on Postgresql client
  - Initializing benchmark test
  - Start Testing the PostGreSQL
  - Pgbench: Options
  - Pgbench: Scripts
- **PostgreSQL: Other Benchmark Tools**
- **Further Reading**
- **Conclusions**

why benchmark?

- Benchmark can be used beforehand to uncover hidden problems
  - Number of connections limit of 100 for a server that is expecting high load
  - There is not enough resources to handle that many clients

Performance Tuning

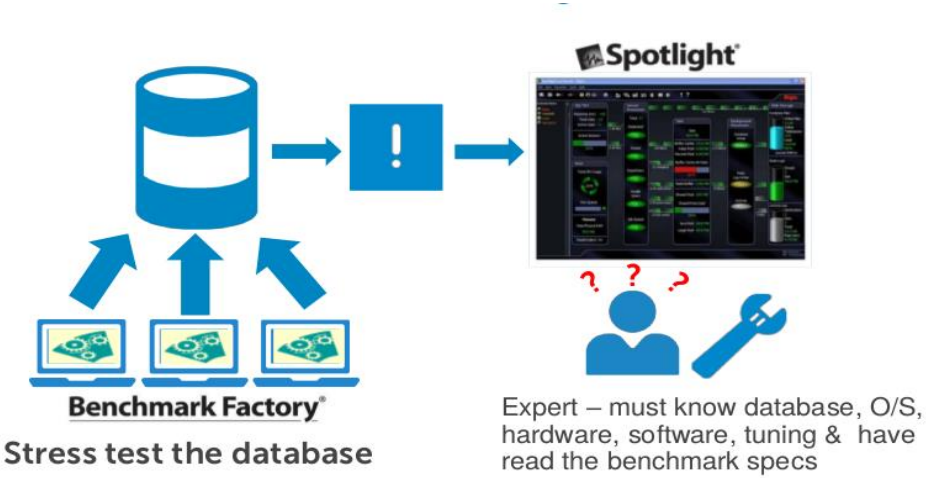
- When it comes to performance tuning an environment,

➢ Often the first place to start is with the database
  - Most of applications rely heavily on database

➢ Databases can be one of the most complex areas to tune
  - Tuning DB performance often involves tuning more than DB itself.
  - It often requires making hardware, OS, or even application modifications.

A way for a successful DB Bench

- Benchmark your DB server from the clients(remotely) side:
  - Stress the server by increasing workload
- Monitor the server Hardware(memory, I/O, CPU,..etc)
  - By using a monitor tool
  - You will see the affect of the stress
  - You will know what you need for high performance DB



DB Benchmarking Common Mistakes

- Wrong Person Doing the Heavy Lifting
- Trying to make do without the right tools
- Wrong number of virtual users (Don't Cut Corners)

Standard Database Benchmarks

- Standard Database Benchmarks have been around for over 20 years
- Transaction Processing Performance Council (TPC.org)
  - Non-profit corporation founded to define transaction processing and database benchmarks and to disseminate objective

Full Members

actian	cisco	cloudera	DELL	DataCore SOFTWARE
FUJITSU	Hewlett Packard Enterprise	HITACHI	HUAWEI	IBM
inspur	intel	Lenovo	Microsoft	ORACLE
Pivotal	redhat	SAP	Teradata a division of NCR	TTA
vmware				

Associate Members

IDEAS	SDSC	Telecommunications Technology Association		CAICT 中国信息通信研究院

CSC | COMPUTER SCIENCE AND COMMUNICATIONS RESEARCH UNIT

uni.lu UNIVERSITÉ DU LUXEMBOURG 7

TPC Benchmarks

- **TPC.org**

TPC-B: measures throughput in terms of how many transactions per second a system can perform.

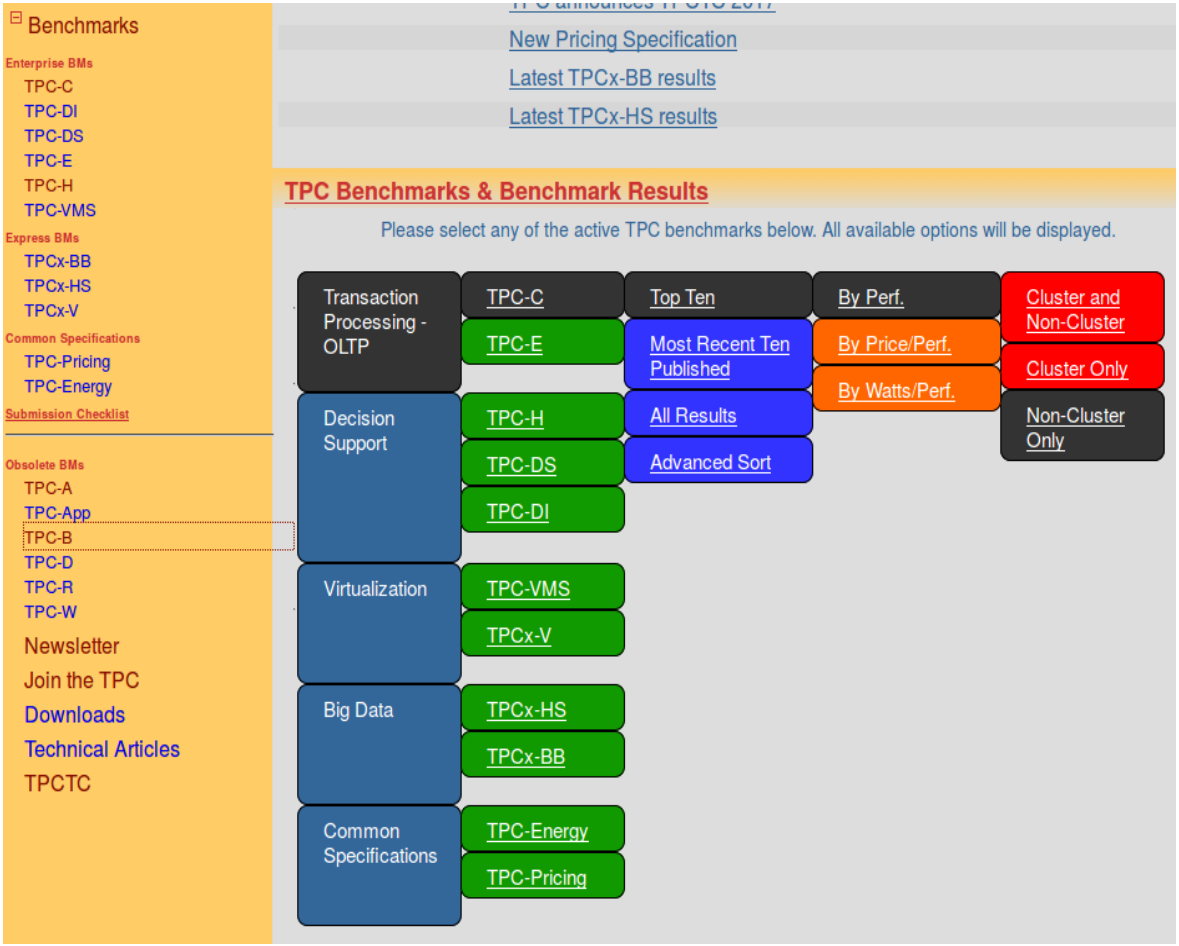
TPC-C: Older OLTP benchmark. Basic "order entry" type application.

TPC-H: Data warehousing queries. 22 queries – not star schema design.

TPC-E: Newer OLTP benchmark. Simulates workload of brokerage firms.

TPC-DS: New, evolving complete DW. Spec still evolving – not yet in BMF.

TPC-VMS: New, evolving virtual DB's. Spec still evolving – not yet in BMF.



TPC-B

- Measures throughput in terms of how many transactions per second a system can perform.
  - Companies use it to benchmark the PostgreSQL.

- Be Sure to Read the Specifications
  - Understand basic database design
  - Implementation options up to you
  - Sizing options for data load and concurrency
  - How to interpret results and/or gauge success

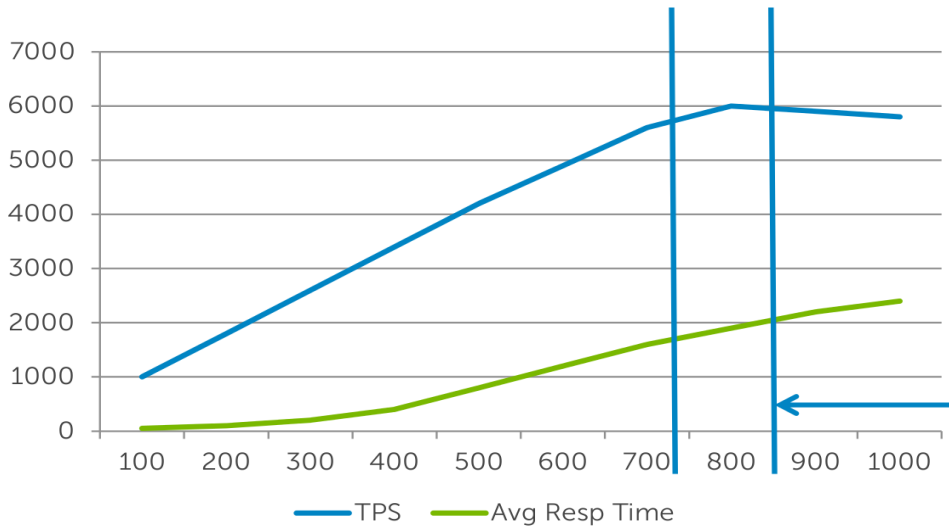
  - Majority of failures due to insufficient preparation, unrealistic expectations,and uncertain metrics for success
  - Page: http://www.tpc.org/tpcb/default.asp

Common Database Benchmark Metrics

- Transactions per Seconds (TPS)
  - Gets far too much attention
  - Meaningless to most users
  - Sort of like automobile RPM's (how fast internal engine is working, not how fast car is moving or how soon we'll arrive)
  - Misconception that TPS equates to IOPS (I/O Operations per Second), ignores database memory caching and logging
- Avg. Response Time (RT)
  - Gets far too less attention
  - Meaningful to most users
  - Sort of like MPH (or KPH) (how fast car actually is moving so infers how soon we'll arrive or amount of fuel we'll use)

True Point of Saturation: TPS vs. Avg. RT

- When examined **Avg. RT** in conjunction with **TPS**, then a generally observable and clear pattern often emerges



Looking for inflection point where TPS is still increasing or just decreasing and close to max where average response time is below customer defined SLA

Common mistake to simply attempt maximize TPS – remember TPS is not IOPS

Tune PostgreSQL Performance

- As always one should experiment with what values work best for your environment.
  - max_connections = N
  - 140% of the average number of expected connections, N= 100 so avg= 140 max connections
  - shared_buffers = N
  - 1/4 to 1/2 physical memory
  - fsync = true|false
  - Setting this to false will speed up the file system but crashes or unexpeced stop will require a restore from backup, keep as fsync=true
  - work_mem = N
  - Default is 1M but 2M doesn't hurt, this can be set per

CSC | COMPUTER SCIENCE AND COMMUNICATIONS RESEARCH UNIT

uni.lu UNIVERSITÉ DU LUXEMBOURG

Benchmarking PostgreSQL: pgbench

- Why pgbench:
  - It is fast to setup.
  - It is simple to use.
  - Tests on pgbench are based on **TCP-B**
  - The test involve :
    - Select, update and insert commands per transaction
  - It is easy to test other cases by writing your own transaction script files

Pgbench: Installation

- If you test PostgreSQL server from:
  - Locally:
    - The pgbench already exist with the DB server
  - Remotely:
    - On the client side, issue this command:
      - $ sudo apt-get install postgresql-contrib
    - Test the setup by calling the help:
      - $ pgbench -? | [--help]

Pgbench: initializing the test

- Setting up the pgbench sample database
  - Initialize: to create tables for test
    - $ pgbench -i [other-options] dbname
  - Other options (remotely):
    - $ pgbench -i -h server-ip -p port dbname -s scaling-factor -F fillFactor
- Pgbench initializing test creates four tables:
- By default value -s = 1

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

- Pgbench destroying any existing tables of these names

Pgbench: initializing the test

- To increase the database size for test use:
- -s scaling factor:
  - To multiply the number of rows entered into each table and the database size, if -s = 50
  - DB size = 50 x 16 MB = 800

```
table                    # of rows
---------------------------------
pgbench_branches         50
pgbench_tellers          500
pgbench_accounts         5000000
pgbench_history          0
```

- -F fillFactor
  - Should be larger than the scaling factor
  - -F = 90 the DB size = 850 MB

CSC | COMPUTER SCIENCE AND COMMUNICATIONS RESEARCH UNIT

uni.lu UNIVERSITÉ DU LUXEMBOURG

Start testing PostgreSQL DB Server

Once initialization is done, run the benchmark without -i option:

- $ pgbench [options] dbname

- Example:
  - $ pgbench -h server-ip -p port -c connections -j threads -t transactions dbname
  - There are many options for pgbench(will be discussed in the next slide).

- Output:

```
postgres@client1:~$ pgbench -h 10.10.1.200 -p 5432 -c 10 -j 2 -t 10000 ali
Password:
starting vacuum...end.
transaction type: TPC-B (sort of)
scaling factor: 50
query mode: simple
number of clients: 10
number of threads: 2
number of transactions per client: 10000
number of transactions actually processed: 100000/100000
tps = 301.454305 (including connections establishing)
tps = 301.510476 (excluding connections establishing)
```

Pgbench: Options

- Initialization: initializing the test
  - -i -s -F and more ...
- Common: during benchmark and initialization
  - -h -p -U dbuser and more ...
- Benchmarking: benchmark arguments
  - -c Number of clients simulated
  - -j Number of worker threads within pgbench.
  - -t Number of transactions each client runs.
  - -L Transaction which last more than limit milliseconds are counted and reported separately
  - M Protocol to use for submitting queries to the

Pgbench: Scripts

- pgbench executes test scripts chosen randomly from a specified list.

1) Built-in scripts with -b scriptname[@weight]

- Each script may be given a relative weight specified after a @
  - The default weight for each script is 1
  - The scripts with 0 weight are ignored
- Scripts names: tpcb-like, simple-update and select-only
  - Use -S to invoke the select-only built-in script (select statement only !)
  - Use -N to invoke the simple-update built-in script (just reduce the update transactions).
- The default script is invoked by -b tpcb-like

```
1. BEGIN;
2. UPDATE pgbench_accounts SET abalance = abalance + :delta WHERE aid = :aid;
3. SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
4. UPDATE pgbench_tellers SET tbalance = tbalance + :delta WHERE tid = :tid;
5. UPDATE pgbench_branches SET bbalance = bbalance + :delta WHERE bid = :bid;
6. INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES (:tid,
:bid, :aid, :delta, CURRENT_TIMESTAMP);
7. END;
```

CSC | COMPUTER SCIENCE AND COMMUNICATIONS RESEARCH UNIT

UNIVERSITÉ DU LUXEMBOURG

Pgbench: Scripts(cont.)

2) User-provided custom scripts with -f filename[@weight] option:

- Custom benchmark scenarios
  - By replacing the default transaction script
  - A script file contains one or more SQL commands terminated by semicolons.
  - Script file meta commands begin with a backslash (\)
  - Use set and pgbench built-in functions
- A full example of the pgbench: scripts and functions is the full definition of the built-in TPC-B-like transaction is:

```
\set aid random(1, 100000 * :scale)
\set bid random(1, 1 * :scale)
\set tid random(1, 10 * :scale)
\set delta random(-5000, 5000)
BEGIN;
UPDATE pgbench_accounts SET abalance = abalance + :delta WHERE aid = :aid;
SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
UPDATE pgbench_tellers SET tbalance = tbalance + :delta WHERE tid = :tid;
UPDATE pgbench_branches SET bbalance = bbalance + :delta WHERE bid = :bid;
INSERT INTO pgbench_history (tid, bid, aid, delta, mtime) VALUES (:tid, :bid, :aid, :delta,
CURRENT_TIMESTAMP);
END;
```

Pgbench Documentation on GitHub

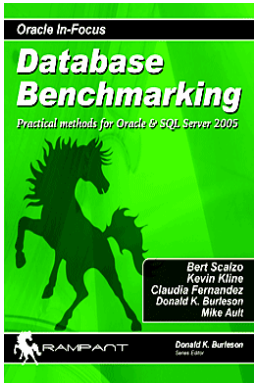
- For more information about pgbench like:
  - Custom Scripts
  - Built-in Functions
  - Per-Transaction Logging
  - Aggregated Logging
  - Per-Statement Latencies

- Browse the PostgreSQL- Bench on GitHub:
  - https://github.com/AbdallahCoptan/PostGreSQL-Bench

PostgreSQL: Other Benchmark tools

- **TPC-H:**
  - The official TPC-H benchmark at :
    - http://tpc.org/tpch/default.asp
  - Use the TPC-H-like(use only the dbgen a qgen parts) benchmark at :
    - https://github.com/AbdallahCoptan/pg_tpch
- **Pgbench-tools:**
  - Automates running PostgreSQL's built-in pgbench tool in a useful ways.
    - https://github.com/AbdallahCoptan/pgbench-tools
- **HammerDB:**

Further Reading



- GitHub:
  - https://github.com/AbdallahCoptan/PostGreSQL-Bench
- Tuning PostgreSQL with pgbench
  - https://blog.codeship.com/tuning-postgresql-with-pgbench/
- Book: Database Benchmarking: Practical Methods for Oracle & SQL Server
- Performance Tuning Postgresql
  - http://edoceo.com/howto/postgresql-performance

Conclusions

- Performance is also tied to the schema and design of the system and to the data that is stored.
- To successfully run a benchmark the tests must be done many times and then averaged out
- Pgbench is an excellent testing tool but,
- You need to tune the postgresql performance metrics with the **pgbench** test
- Also, the hardware performance
- Finally, Monitor your server during the test

Thank you for your attention

Abdallah Ali Zainelabden Abdallah IBRAHIM
abdallah.ibrahim@uni.lu
pcog.uni.lu

CSC | COMPUTER SCIENCE AND COMMUNICATIONS RESEARCH UNIT

UNIVERSITÉ DU LUXEMBOURG