PgBench – Work in Progress

Talk Outline

1. PgBench
   - History
   - Capabilities
   - Caveats

2. Performance
   - Overheads
   - Loading
   - Connection Costs
   - SSL Costs
   - Index
   - Fill Factor

3. Future
   - Needs
   - CommitFest
   - Conclusion
Pgbench History
Simple tool based on TPC-B  

- external... then  
- initialization and scale  
- 2 benchmarks  

Tatsuo Ishii 2000

Contrib/  
-i -s 10  
-t 10000 -c 4

Visible and invisible developments  

- Initializing  
- Scripting  
- Running  
- Reporting

Using...  
- Debugging  
- Refactoring  
- Testing

2000-
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Author</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPY</td>
<td>initialization</td>
<td>Takahiro Itagaki</td>
<td>2007</td>
</tr>
<tr>
<td>FILLFACTOR</td>
<td>taux de remplissage</td>
<td>Pavan Deolasee</td>
<td>2007</td>
</tr>
<tr>
<td>UNLOGGED</td>
<td>tables</td>
<td>Robert Haas</td>
<td>2011</td>
</tr>
<tr>
<td>TABLESPACE</td>
<td>on tables or index</td>
<td>–</td>
<td>2011</td>
</tr>
<tr>
<td>--foreign-key</td>
<td>declarations</td>
<td>Jeff Janes</td>
<td>2012</td>
</tr>
<tr>
<td>-I</td>
<td>initialization steps</td>
<td>Masahiko Sawada</td>
<td>2017</td>
</tr>
</tbody>
</table>
PgBench History

Scripting

- \text{-N} \quad \text{simple update} \quad \text{Tatsuo Ishii} \quad 2002
- \text{-f} \quad \text{script file} \quad \text{Tomoaki Sato} \quad 2005
  \text{//set} \quad \text{basic arithmetic} \quad – \quad 2006
  \text{//sleep} \quad \text{slepping} \quad \text{Jan Wieck} \quad 2007
  \text{//shell} \quad \text{shelling...} \quad \text{Michaël Paquier} \quad 2009
\text{gaussian} \quad \text{random} \quad \text{Mitsumasa Kondo} \quad 2014
\text{exponential} \quad \text{random} \quad \text{Fabien Coelho} \quad 2014
\text{expression} \quad \text{integer arithmetic} \quad \text{Robert Haas} \quad 2015
\text{double} \quad \text{arithmetic and functions} \quad \text{Fabien Coelho} \quad 2016
\text{non-ascii} \quad \text{variable names} \quad – \quad 2017
PgBench History

PgBench WIP
F. Coelho

PgBench
History
Capabilities
Caveats

Performance
Overheads
Loading
Connection
SSL
Index
Fill Factor
Future
Needs
CommitFest
Conclusion

Running

PgBench History

- C  connection  Tatsuo Ishii  2001
- M  query mode  Takahiro Itagaki  2008
- T  run time  –  2008
- j  threading  –  2009
- R  throttling  Fabien Coelho  2013
- L  latency limit  –  2014
- f/b  ... weighted scripts  –  2016
### PgBench History

**-l**  logging  
**-r**  per statement stats  
**--sampling-rates**  sample stats  
**--aggregate-interval**  aggregated stats  
**-P**  progress  
**-f ...**  per script stats

---

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Author</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>-l</td>
<td>logging</td>
<td>Neil Conway</td>
<td>2002</td>
</tr>
<tr>
<td>-r</td>
<td>per statement stats</td>
<td>Florian Pflug</td>
<td>2010</td>
</tr>
<tr>
<td>--sampling-rates</td>
<td>sample stats</td>
<td>Tomas Vondra</td>
<td>2012</td>
</tr>
<tr>
<td>--aggregate-interval</td>
<td>aggregated stats</td>
<td>--</td>
<td>2013</td>
</tr>
<tr>
<td>-P</td>
<td>progress</td>
<td>Fabien Coelho</td>
<td>2013</td>
</tr>
<tr>
<td>-f ...</td>
<td>per script stats</td>
<td>--</td>
<td>2016</td>
</tr>
</tbody>
</table>
PgBench Capabilities
PgBench Capabilities

Initialize a database

- create and fill, with scaling
- options: PK, FK, unlogged, fillfactor, tablespace...

Run scripts

- psql-like, 3 builtins or custom, weighted, prepared, throttled
- parallelism: threads, clients, re-connections...

Measure and report performance

- tps, latency, timeout; per script, per command...
- detailed, sampled or aggregated; stdout or file
Benchmarking Caveats

Beware

- long enough
- several times
- representative

warm-up, checkpoint and vacuum reproducibility pedal-to-the-metal?
# Benchmarking vs Performance Testing

## Benchmarking

- standard schema and transaction
- maximum load
- report transaction per second
- latency should good enough...

## Performance Testing

- YOUR schema and transaction
- YOUR load...
- load must be processed
- latency must match application constraints

---

**Benchmarking**

- *System comparison*
  - *pedal to the metal*
  - *tps*
  - *s*

**Performance Testing**

- *Does it work for me?*
  - *throttling*
Deceptive Performance

Version 9.5.5
- throughput: 329.4 tps
- average latency: 24.3 ms
- latency stddev: 79.5 ms

Version 9.6.1
- throughput: 326.4 tps
- average latency: 24.4 ms
- latency stddev: 20.3 ms

```sql
pgbench -j 4 -c 8
```
What is happening?

- transaction surges are absorbed
- then data are written disk

Buy Now, Pay Later!

*in-memory + WAL checkpoint*
Performance
## Tool Overheads

```
pgbench -T 10 -P 1 -f script.sql

<table>
<thead>
<tr>
<th>Command</th>
<th>Throughput</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep zero</td>
<td>13.4 Mtps</td>
<td>75 ns</td>
</tr>
<tr>
<td>\sleep 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set a variable</td>
<td>9.5 Mtps</td>
<td>105 ns</td>
</tr>
<tr>
<td>\set i 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Empty command</td>
<td>97,222 tps</td>
<td>10.3 µs</td>
</tr>
<tr>
<td>;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Empty SELECT</td>
<td>51,631 tps</td>
<td>19.4 µs</td>
</tr>
<tr>
<td>SELECT;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
### Impact of schema on loading time

<table>
<thead>
<tr>
<th>steps</th>
<th>(d)rop (t)able (g)enerate (v)accum (p)primary and (f)oreign key</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>pgbench -i -s 100 -I 'dtgv'</td>
<td>18 s</td>
<td></td>
</tr>
<tr>
<td>pgbench -i -s 100 -I 'dtgvp'</td>
<td>29 s</td>
<td></td>
</tr>
<tr>
<td>pgbench -i -s 100 -I 'dtgvpf'</td>
<td>32 s</td>
<td></td>
</tr>
<tr>
<td>pgbench -i -s 100 -I 'dtpgf'</td>
<td>39 s</td>
<td></td>
</tr>
<tr>
<td>pgbench -i -s 100 -I 'dtpfgv'</td>
<td>103 s</td>
<td></td>
</tr>
</tbody>
</table>

### Impact summary

- Primary key: 50-100%
- Foreign key: 20-300%
**Connection Costs**

```
pgbench -C
```

Diagram showing the interaction between `pgbench` and `postgres` with labels:
- **Client**
- **LAN**
- **Server**

**Postgres 9.6.1**

**Initialization and Benchmarks**

- `pgbench -i -s 100`
  - **1.5 GB**
- `pgbench -T 2000 -C "host=server ssldmode=require"
  - **36.1 tps**
- `pgbench -T 2000 -C "host=server ssldmode=disable"
  - **56.4 tps**
- `pgbench -T 2000  "host=server ssldmode=disable"
  - **105.4 tps**

- **connection AAA**
  - **8.2 ms**
- **SSL negociation**
  - **10.0 ms**
- **transfers and transactions**
  - **9.5 ms**
### SSL or not?

**SSL Costs**

- negotiation and re-negotiation
- cryptographic functions
- certificate

**Benefits**

- Confidentiality
- Integrity
- Authentication

```plaintext
pgbench -j 1 -c 1 -D scale=100 -f ro3.sql -T 30 "host=server ..."
```

<table>
<thead>
<tr>
<th>sslmode=mandatory</th>
<th>clear</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SSL</strong></td>
<td></td>
</tr>
<tr>
<td>throughput</td>
<td>709.7 tps</td>
</tr>
<tr>
<td>latency</td>
<td>1.407 ± 0.132 ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sslmode=disable</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>clear</strong></td>
<td></td>
</tr>
<tr>
<td>throughput</td>
<td>781.6 tps</td>
</tr>
<tr>
<td>latency</td>
<td>1.277 ± 0.034 ms</td>
</tr>
</tbody>
</table>
### Select Only Index

<table>
<thead>
<tr>
<th>With primary key</th>
<th>17,225 tps</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialization</td>
<td></td>
</tr>
</tbody>
</table>

```
pgbench -T 10 -P 1 -S
```

<table>
<thead>
<tr>
<th>No primary key</th>
<th>23 tps</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialization</td>
<td></td>
</tr>
</tbody>
</table>

```
pgbench -i -s 10 -I "dtgvp"
```

<table>
<thead>
<tr>
<th>With hash index</th>
<th>18,289 tps</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialization</td>
<td></td>
</tr>
</tbody>
</table>

```
pgbench -i -s 10 -I "dtgvp"
```

- plus non unique hash index

```
CREATE INDEX ah ON pgbench_accounts USING HASH(aid);
```
Fill Factor

Update intensive load

- UPDATE = DELETE + INSERT
- induce about 3 page writes
- or keep some free space available

Initialization

CREATE TABLE pgbench_accounts(...)
  WITH (FILLFACTOR = 95);

...
Fill Factor

```
pgbench -f update.sql ...
```

**Only UPDATE script**

```
\set naccounts 100000 * :scale
\set aid random(1, :naccounts)
\set delta random(-5000, 5000)
UPDATE pgbench_accounts
    SET abalance = abalance + :delta, filler = NOW()::TEXT
WHERE aid = :aid;
```
Future
PgBench Future

Where to stop?
Needed Capabilities

TPC-B Simple Banking Example  version 2.0, June 1994

- schema creation and initialization
- simple but non trivial SQL scripting
  - SQL commands (SELECT INSERT UPDATE COMMIT...)
  - variables, expressions, (uniform) random numbers, \textbf{if}...
- constant performance monitoring and reporting
  stability, steady-state, statistics, errors...
- explicit anti-cheating constraints
  
\textbf{1.3.2 application must retrieve the balance}

Cannot be implemented with PgBench (yet)
<table>
<thead>
<tr>
<th>Needed Capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Missing features in development</strong></td>
</tr>
<tr>
<td>cset/gset</td>
</tr>
<tr>
<td>boolean</td>
</tr>
<tr>
<td>if/endif</td>
</tr>
<tr>
<td><strong>Other features in development</strong></td>
</tr>
<tr>
<td>pow</td>
</tr>
<tr>
<td>ppoll</td>
</tr>
<tr>
<td>stats</td>
</tr>
<tr>
<td>perm</td>
</tr>
<tr>
<td>error</td>
</tr>
</tbody>
</table>
TPC-B Real Transaction Profile

```
\set tbid random(1, :scale)
\set tid 10 * (:tbid - 1) + random(1, 10)
\if :scale = 1 OR random(0, 99) < 85 -- same branch
  \set bid :tbid
\else -- other branch
  \set bid 1 + (:tbid + random(1, :scale - 1)) % :scale
\endif
\set aid :bid * 100000 + random(1, 100000)
\set delta random(-999999, 999999)
BEGIN;
UPDATE pgbench_accounts
  SET abalance = abalance + :delta WHERE aid = :aid
  RETURNING abalance AS balance \gset
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;
```
CommitFest

Patch Catchup Process

Work in slow Progress

- submit patch
- get a review and review others
- get a decision
- wait for a committer...
- and maybe get one

commitfest.postgresql.org
Conclusion

Easy to use and improving tool

- write your custom script
- run it against your data
- for your load

Need test data?

- directives on SQL declarations
- generators for many types and constraints

pgbench

datafiller