Measuring and Reducing PostgreSQL Transaction Latency

Fabien Coelho

MINES ParisTech, PSL Research University

pgDay Paris – March 23, 2017
Postgres Latency

Talk Outline

1 Introduction
- Subject
- Typical Web Application
- Transaction Performance Definitions
- pgbench

2 Performance Comparisons
- Two Connection Costs
- Latency Pitfalls
- Benchmarking with Rate and Limit
- Three Storage Options
- Two Protocol Impacts
- Four Query Combination Tricks
- Reducting Server Distance
- Performance Scalability
- Miscellaneous Settings

3 Conclusion
- Latency and Throughput Wrap-Up
- Lessons Learned
- Contributions to Postgres
Small OLTP

- CRUD queries
- data fit in shared buffers
- RW, RO

OnLine Transaction Processing

... WHERE pk=?
small, few GB
pgbench builtins

Focus

- performance with emphasis on latency

and Motivation

interactive web app
Subject

Small OLTP

- CRUD queries
- data fit in shared buffers
- RW, RO

OnLine Transaction Processing

... WHERE pk=?
small, few GB
pgbench builtins

Focus and Motivation

- performance with emphasis on latency
- interactive web app
Subject

Small OLTP

- CRUD queries
- data fit in shared buffers
- RW, RO

OnLine Transaction Processing

... WHERE pk=?

small, few GB

pgbench builtin

Focus

- performance with emphasis on latency
- experiments & measures

and Motivation

interactive web app
do not assume!
### Small OLTP

- CRUD queries
- Data fit in shared buffers
- RW, RO

**OnLine Transaction Processing**

\[
\text{latency performance: } RW \times 63, \ RO \times 219
\]

### Focus and Motivation

- Performance with emphasis on latency
- Experiments & measures

Interactive web app do not assume!
Typical Web Application

3-Tier Architecture

Client user acts on user-agent, sends to
Server process request, database operations to
Database stores and retrieves data

Database Operations

- Connection
- Request-Response cycles

TCP/IP, SSL & AAA
Typical Web Application

3-Tier Architecture

**Client** user acts on user-agent, sends to

**Server** process request, database operations to

**Database** stores and retrieves data

Database Operations

- Connection
- Request-Response **cycles**

\[ \text{TCP/IP, SSL & AAA} \]

\[ \text{transfer, parse, plan, execute, transfer back} \]
## Transaction Performance

### Definitions

<table>
<thead>
<tr>
<th>Throughput</th>
<th>operations per time unit</th>
<th>usual approach, load measured in <strong>tps</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency</td>
<td>time for one operation</td>
<td><em>must fit application requirements</em></td>
</tr>
</tbody>
</table>

### Comments

- correlated
- max vs enough
- sensitive to many settings
- throughput bottleneck & latency additivity

*time & operations*

- **tx/s**
- **ms/tx**

*usual approach, load measured in tps*

Comments and contradictory and vice-versa net, soft & hard
Transaction Performance

Definitions

**Throughput**
operations per time unit

*usual approach, load measured in* **tps**

**Latency**
time for one operation

*must fit application requirements*

Comments

- correlated
- max vs enough
- sensitive to many settings
- throughput bottleneck & latency additivity

**time & operations**

<table>
<thead>
<tr>
<th>tx/s</th>
<th>ms/tx</th>
</tr>
</thead>
</table>

**And contradictory**

**And vice-versa**

**Net, soft & hard**
## Transaction Performance

### Definitions

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Throughput</strong></td>
<td>operations per time unit</td>
<td>tx/s</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>time for one operation</td>
<td>ms/tx</td>
</tr>
</tbody>
</table>

*usual approach, load measured in tps*

*must fit application requirements*

### Comments

- correlated
- max vs enough
- sensitive to many settings
- throughput bottleneck & latency additivity

*and contradictory* and vice-versa

*net, soft & hard*
**Transaction Performance**

<table>
<thead>
<tr>
<th>Definitions</th>
<th>time &amp; operations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Throughput</strong></td>
<td>operations per time unit ( tx/s )</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>time for one operation ( ms/tx )</td>
</tr>
</tbody>
</table>

**Comments**
- correlated
- max vs enough
- sensitive to many settings
- throughput bottleneck & latency additivity

and contradictory and vice-versa net, soft & hard
Transaction Performance

Definitions

<table>
<thead>
<tr>
<th>Throughput</th>
<th>operations per time unit</th>
<th>$tx/s$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>usual approach, load measured in tps</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Latency</th>
<th>time for one operation</th>
<th>$ms/tx$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>must fit application requirements</em></td>
<td></td>
</tr>
</tbody>
</table>

Comments

- correlated and contradictory
- max vs enough and vice-versa
- sensitive to many settings net, soft & hard
- throughput bottleneck & latency additivity deep voodoo!
Available Features

- **input**: SQL-like scripts with minimal client-side language
- **options**: time to run, prepared, reconnections, ...
- **parallelism**: threads, clients, asynchronous calls
- **output**: statistical performance data

Caveats

- long enough
- several times
- pedal-to-the-metal max speed test

---

warm-up, checkpoint and vacuum reproducibility not representative
Available Features

- **input** SQL-like scripts with minimal client-side language
- **options** time to run, prepared, reconnections, ...
- **parallelism** threads, clients, asynchronous calls
- **output** statistical performance data

Caveats

- long enough
- several times
- pedal-to-the-metal max speed test

- warm-up, checkpoint and vacuum
- reproducibility
- not representative
Default TPC-B-like Transaction

TPC-B-like banking transaction

```sql
-- random ids and amount
\set aid random(1, 100000 * :scale)
\set bid random(1, 1 * :scale)
\set tid random(1, 10 * :scale)
\set delta random(-5000, 5000)

-- actual transaction
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;
```
Default TPC-B-like Transaction

TPC-B-like banking transaction

--- random ids and amount
\set aid random(1, 100000 * :scale)
\set bid random(1, 1 * :scale)
\set tid random(1, 10 * :scale)
\set delta random(-5000, 5000)
--- actual transaction
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 -b tcpb-like
Performance Comparisons

Two Connection Costs
### Connection Costs

**pgbench**

**postgres**

- **Client**
- **LAN**
- **Server**

**Initialization**

**Postgres 9.6.1**

**pgbench**

**-i**

**-s 100**

**1.5 GB**
Connection Costs

pgbench -C

---

**Initialization**

pgbench -i -s 100

---

Postgres 9.6.1

**Client**

8 cores, 16 GB

**LAN**

1 Gbps

**Server**

16 cores, 32 GB, HDD
Connection Costs

**pgbench**

```
pgbench -C
```

---

**Initialization and Benchmarks**

```
pgbench -i -s 100
```

1.5 GB

```
pgbench -T 2000 -C "host=server sslmode=require"
```

36.1 tps

---

**Client**

8 cores, 16 GB

**LAN**

1 Gbps

**Server**

16 cores, 32 GB, HDD
Connection Costs

**Initialization and Benchmarks**

Postgres 9.6.1

```bash
pgbench -i -s 100
1.5 GB
```

```bash
pgbench -T 2000 -C "host=server sslmode=require"
36.1 tps
```

```bash
pgbench -T 2000 -C "host=server sslmode=disable"
56.4 tps
```
**Connection Costs**

```
pgbench -C
```

**Initialization and Benchmarks**

Postgres 9.6.1

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

Client 8 cores, 16 GB
LAN 1 Gbps
Server 16 cores, 32 GB, HDD

Initialization and Benchmarks

Postgres 9.6.1

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

- connection AAA 8.2 ms
- SSL negotiation 10.0 ms
- transfers and transactions 9.5 ms
Performance Comparisons

Latency Pitfalls
Latency Comparison – 9.5 vs 9.6

<table>
<thead>
<tr>
<th>Version 9.5.5</th>
<th>Version 9.6.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>throughput</td>
</tr>
<tr>
<td>329.4 tps</td>
<td>326.4 tps</td>
</tr>
<tr>
<td>average latency</td>
<td>average latency</td>
</tr>
<tr>
<td>24.3 ms</td>
<td>24.4 ms</td>
</tr>
</tbody>
</table>

pgbench -j 4 -c 8
Latency Comparison – 9.5 vs 9.6

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

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

pgbench -j 4 -c 8
Latency Comparison – 9.5 vs 9.6

**Version 9.5.5**
- throughput: 329.4 tps
- average latency: 24.3 ms
- latency std. dev.: 79.5 ms

**Version 9.6.1**
- throughput: 326.4 tps
- average latency: 24.4 ms
- latency std. dev.: 20.3 ms
What is happening?

- transaction surges are absorbed
- then data are written disk
Latency Comparison – 9.5 vs 9.6

**Version 9.5.5**

**Version 9.6.1**

What is happening?

- transaction surges are absorbed *in-memory + WAL*
- then data are written disk *checkpoint*

Instant TPS
What is happening?

- transaction surges are absorbed
- then data are written disk

Buy Now, Pay Later!

*in-memory + WAL checkpoint*
Latency Comparison – 9.5 vs 9.6

Postgres 9.5 Checkpoint
- data writes spread over some time
- OS choose when to actually write
- until fsync is called…

Postgres 9.6 Checkpoint

random I/O
30s delay on Linux
Latency Comparison – 9.5 vs 9.6

Postgres 9.5 Checkpoint

- data writes spread over some time
- OS choose when to actually write
- until fsync is called...

Postgres 9.6 Checkpoint

random I/O
30s delay on Linux
Latency Comparison – 9.5 vs 9.6

Checkpointing

Postgres 9.5 Checkpoint

- data writes spread over some time
- OS choose when to actually write
- until fsync is called...

Postgres 9.6 Checkpoint

random I/O

30s delay on Linux

I/O storm – on low-end HDD
Latency Comparison – 9.5 vs 9.6

**Checkpointing**

<table>
<thead>
<tr>
<th>Postgres 9.5 Checkpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>data writes spread over some time</td>
</tr>
<tr>
<td>OS choose when to actually write</td>
</tr>
<tr>
<td>until <code>fsync</code> is called...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Postgres 9.6 Checkpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sorted</strong> data writes spread over some time</td>
</tr>
<tr>
<td><strong>flush</strong> instructions sent regularly (256 kB)</td>
</tr>
</tbody>
</table>

**Postgres 9.5 Checkpoint**
- Data writes spread over some time
- OS choose when to actually write
- Until `fsync` is called...

**Postgres 9.6 Checkpoint**
- **Sorted** data writes spread over some time
- **Flush** instructions sent regularly (256 kB)

---

**Checkpointing**

**Latency Comparison – 9.5 vs 9.6**

**Postgres 9.5 Checkpoint**
- Data writes spread over some time
- OS choose when to actually write
- Until `fsync` is called...

**Postgres 9.6 Checkpoint**
- Sorted data writes spread over some time
- Flush instructions sent regularly (256 kB)

---

**Checkpointing**

**Latency Comparison – 9.5 vs 9.6**

**Postgres 9.5 Checkpoint**
- Data writes spread over some time
- OS choose when to actually write
- Until `fsync` is called...

**Postgres 9.6 Checkpoint**
- Sorted data writes spread over some time
- Flush instructions sent regularly (256 kB)

---

**Checkpointing**

**Latency Comparison – 9.5 vs 9.6**

**Postgres 9.5 Checkpoint**
- Data writes spread over some time
- OS choose when to actually write
- Until `fsync` is called...

**Postgres 9.6 Checkpoint**
- Sorted data writes spread over some time
- Flush instructions sent regularly (256 kB)

---

**Checkpointing**

**Latency Comparison – 9.5 vs 9.6**

**Postgres 9.5 Checkpoint**
- Data writes spread over some time
- OS choose when to actually write
- Until `fsync` is called...

**Postgres 9.6 Checkpoint**
- Sorted data writes spread over some time
- Flush instructions sent regularly (256 kB)
Latency Comparison – 9.5 vs 9.6

Checkpointing

Postgres 9.5 Checkpoint

- data writes spread over some time
- OS choose when to actually write
- until `fsync` is called...

Postgres 9.6 Checkpoint

- **sorted** data writes spread over some time
- **flush** instructions sent regularly (256 kB)
- when `fsync` is called

---

Postgres 9.5 Checkpoint

- random I/O
- 30s delay on Linux
- I/O storm – on low-end HDD

Postgres 9.6 Checkpoint

- sequential I/O
- `checkpoint_flush_after`
- ok!
Performance Comparisons

Benchmarking with Rate and Limit
Postgres Latency

F. Coelho

Introduction

Subject

Application

Definitions

pgbench

Performance

Connection

Latency

Rate & Limit

Storage

Protocol

Combinations

Distance

Scalability

Miscellaneous

Conclusion

Wrap-Up

Lessons

Contributions

---

**Rate (tps) and Limit (ms)**

pgbench -R 100 -L 100 -N

<table>
<thead>
<tr>
<th>Pg 9.5</th>
<th>basic checkpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ slow &amp; skipped</td>
<td></td>
</tr>
<tr>
<td>■ latency</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pg 9.6</th>
<th>sorted checkpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ slow &amp; skipped</td>
<td></td>
</tr>
<tr>
<td>■ latency</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pg 9.6</th>
<th>sorted &amp; flushed checkpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ slow &amp; skipped</td>
<td></td>
</tr>
<tr>
<td>■ latency</td>
<td></td>
</tr>
</tbody>
</table>
Rate (tps) and Limit (ms)

Pg 9.5  *basic checkpoint*

- slow & skipped
- latency

Pg 9.6  *sorted checkpoint*

- slow & skipped
- latency

Pg 9.6  *sorted & flushed checkpoint*

- slow & skipped
- latency

pgbench -R 100 -L 100 -N
**Pg 9.5** *basic checkpoint*
- slow & skipped: 24.0%
- latency: $15.6 \pm 158.3$ ms

**Pg 9.6** *sorted checkpoint*
- slow & skipped
- latency

**Pg 9.6** *sorted & flushed checkpoint*
- slow & skipped
- latency
Postgres Latency

F. Coelho

Introduction
Subject Application Definitions pgbench

Performance
Connection Latency Rate & Limit Storage Protocol Combinations Distance Scalability Miscellaneous

Conclusion
Wrap-Up Lessons Contributions

Pg 9.5  basic checkpoint
- slow & skipped 24.0%
- latency $15.6 \pm 158.3$ ms

Pg 9.6  sorted checkpoint
- slow & skipped
- latency

Pg 9.6  sorted & flushed checkpoint
- slow & skipped
- latency

pgbench -R 100 -L 100 -N
Rate (tps) and Limit (ms)

**Pg 9.5**  
*basic checkpoint*
- slow & skipped: 24.0%
- latency: 15.6 ± 158.3 ms

**Pg 9.6**  
*sorted checkpoint*
- slow & skipped: 2.7%
- latency: 3.6 ± 24.6 ms

**Pg 9.6**  
*sorted & flushed checkpoint*
- slow & skipped
- latency

pgbench -R 100 -L 100 -N
**Rate (tps) and Limit (ms)**

**Pg 9.5** *basic checkpoint*
- slow & skipped: 24.0%
- latency: $15.6 \pm 158.3$ ms

**Pg 9.6** *sorted checkpoint*
- slow & skipped: 2.7%
- latency: $3.6 \pm 24.6$ ms

**Pg 9.6** *sorted & flushed checkpoint*
- slow & skipped
- latency

```
pgbench -R 100 -L 100 -N
```
Rate (tps) and Limit (ms)

Pg 9.5  \textit{basic checkpoint}
- slow & skipped: 24.0%
- latency: $15.6 \pm 158.3$ ms

Pg 9.6  \textit{sorted checkpoint}
- slow & skipped: 2.7%
- latency: $3.6 \pm 24.6$ ms

Pg 9.6  \textit{sorted & flushed checkpoint}
- slow & skipped: 0.5%
- latency: $2.6 \pm 13.8$ ms

\texttt{pgbench -R 100 -L 100 -N}
Performance Comparisons

Three Storage Options
CREATE TABLE pgbench_accounts(...) WITH (FILLFACTOR = 100);

FILLFACTOR Storage Parameter

FILLFACTOR Usage

- MVCC: UPDATE = DELETE + INSERT  
  up to 3 pages changes
- some free space available in page  
  1 inside page change
- **but** more pages/costs for other operations  
  **trade-off**

FILLFACTOR = 100

- throughput 406.9 tps
- latency 19.7 ± 12.3 ms

FILLFACTOR = 95

- throughput
- latency
CREATE TABLE pgbench_accounts(...) WITH (FILLFACTOR = 100);

**FILLFACTOR Usage**

- MVCC: UPDATE = DELETE + INSERT  
  *up to 3 pages changes*
- some free space available in page  
  *1 inside page change*
- **but** more pages/costs for other operations  
  *trade-off*

**FILLFACTOR = 100**

- throughput  
  *406.9 tps*
- latency  
  *19.7 ± 12.3 ms*

**FILLFACTOR = 95**

- throughput
- latency
CREATE TABLE pgbench_accounts(...) WITH (FILLFACTOR = 100);

FILLFACTOR Usage

- MVCC: UPDATE = DELETE + INSERT
- some free space available in page
- but more pages/costs for other operations

FILLFACTOR = 100
- throughput 406.9 tps
- latency 19.7 ± 12.3 ms

FILLFACTOR = 95
- throughput 416.8 tps
- latency 19.2 ± 8.3 ms
Hardware

HDD vs SSD

<table>
<thead>
<tr>
<th>Hard Disk Drive</th>
<th>Solid State Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mechanics</td>
<td>Electronics</td>
</tr>
<tr>
<td>Fast sequential I/O</td>
<td>Fast sequential I/O</td>
</tr>
<tr>
<td>Slow random I/O</td>
<td>Fast random I/O</td>
</tr>
</tbody>
</table>

```
pgbench -j 4 -c 8 -T 2500 -M prepared ...
```

Postgres 9.6

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD</td>
<td>406.9</td>
<td>19.7 ± 12.3 ms</td>
</tr>
<tr>
<td>SSD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hardware

HDD vs SSD

Hard Disk Drive
- mechanics
- fast sequential I/O
- **slow** random I/O

Solid State Disk
- electronics
- fast sequential I/O
- **fast** random I/O

```plaintext
gbench -j 4 -c 8 -T 2500 -M prepared ... 
```

Postgres 9.6

<table>
<thead>
<tr>
<th></th>
<th>HDD</th>
<th>SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>tps</td>
<td>406.9</td>
<td>0</td>
</tr>
<tr>
<td>latency</td>
<td>19.7 ± 12.3 ms</td>
<td>0</td>
</tr>
</tbody>
</table>

Hardware Hardware

- **Hardware**
- **HDD**: Hard Disk Drive
- **SSD**: Solid State Disk

**HDD vs SSD**

- **HDD**
  - **mechanics**
  - Fast sequential I/O
  - **Slow** random I/O

- **SSD**
  - **Electronics**
  - Fast sequential I/O
  - Fast random I/O

```plaintext
gbench -j 4 -c 8 -T 2500 -M prepared ... 
```

**Postgres 9.6**

<table>
<thead>
<tr>
<th></th>
<th>HDD</th>
<th>SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>tps</td>
<td>406.9</td>
<td>0</td>
</tr>
<tr>
<td>latency</td>
<td>19.7 ± 12.3 ms</td>
<td>0</td>
</tr>
</tbody>
</table>
Hardware

HDD vs SSD

Hard Disk Drive

- mechanics
- fast sequential I/O
- slow random I/O

Solid State Disk

- electronics
- fast sequential I/O
  fast random I/O

pgbench -j 4 -c 8 -T 2500 -M prepared ...

Postgres 9.6

<table>
<thead>
<tr>
<th></th>
<th>HDD</th>
<th>SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>tps</td>
<td>406.9</td>
<td>4,764.9</td>
</tr>
<tr>
<td>latency</td>
<td>19.7 ± 12.3 ms</td>
<td>1.7 ± 2.4 ms</td>
</tr>
</tbody>
</table>
Hardware

**Hard Disk Drive**
- mechanics
- fast sequential I/O
- slow random I/O

**Solid State Disk**
- electronics
- fast sequential I/O
- fast random I/O

pgbench -j 4 -c 8 -T 2500 -M prepared ...

**Postgres 9.6**

<table>
<thead>
<tr>
<th>Storage</th>
<th>tps</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HDD</strong></td>
<td>406.9</td>
<td>19.7 ± 12.3 ms</td>
</tr>
<tr>
<td><strong>SSD</strong></td>
<td>4,764.9</td>
<td>1.7 ± 2.4 ms</td>
</tr>
</tbody>
</table>

checkpoint full page write effect

HDD vs SSD
CREATE UNLOGGED TABLE pgbench_accounts(...);
CREATE UNLOGGED TABLE pgbench_accounts(...);

<table>
<thead>
<tr>
<th>Standard</th>
<th>ACID</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>406.9 tps</td>
</tr>
<tr>
<td>latency</td>
<td>19.7 ± 12.3 ms</td>
</tr>
</tbody>
</table>

UNLOGGED

| throughput |
| latency    |
CREATE UNLOGGED TABLE pgbench_accounts();

<table>
<thead>
<tr>
<th>Standard</th>
<th>ACID</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>406.9 tps</td>
</tr>
<tr>
<td>latency</td>
<td>19.7 ± 12.3 ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UNLOGGED</th>
<th>good luck!</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>5,310.7 tps</td>
</tr>
<tr>
<td>latency</td>
<td>1.5 ± 0.3 ms</td>
</tr>
</tbody>
</table>

Can you loose your data?
CREATE UNLOGGED TABLE pgbench_accounts(...);

<table>
<thead>
<tr>
<th>Standard</th>
<th>ACID</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>406.9 tps</td>
</tr>
<tr>
<td>latency</td>
<td>19.7 ± 12.3 ms</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UNLOGGED</th>
<th>good luck!</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>5,310.7 tps</td>
</tr>
<tr>
<td>latency</td>
<td>1.5 ± 0.3 ms</td>
</tr>
</tbody>
</table>

**NO!**
Performance Comparisons

Two Protocol Impacts
Read-Only In-Cache Test

```sql
\set aid random(1, 100000 * :scale)
\set tid random(1, 10 * :scale)
\set bid random(1, :scale)
BEGIN;
SELECT abalance FROM pgbench_accounts WHERE aid=:aid;
SELECT tbalance FROM pgbench_tellers WHERE tid=:tid;
SELECT bbalance FROM pgbench_branches WHERE bid=:bid;
COMMIT;
```

<table>
<thead>
<tr>
<th>Operations</th>
<th>Queries on 3 tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 transfers</td>
<td>network protocol</td>
</tr>
<tr>
<td>2 parse query</td>
<td>syntax analysis</td>
</tr>
<tr>
<td>3 plan query</td>
<td>optimization</td>
</tr>
<tr>
<td>4 execute query</td>
<td>cheap if in cache</td>
</tr>
</tbody>
</table>
Read-Only In-Cache Test

\set aid random(1, 100000 * :scale)
\set tid random(1, 10 * :scale)
\set bid random(1, :scale)

BEGIN;
SELECT abalance FROM pgbench_accounts WHERE aid=:aid;
SELECT tbalance FROM pgbench_tellers WHERE tid=:tid;
SELECT bbalance FROM pgbench_branches WHERE bid=:bid;
COMMIT;

Operations

1. transfers
2. parse query
3. plan query
4. execute query

Queries on 3 tables

network protocol
syntax analysis
optimization
cheap if in cache
SSL Costs

- negotiation and re-negotiation
- cryptographic functions
- certificate

Benefits

- Confidentiality
- Integrity
- Authentication

sslmode=require

- throughput: 709.7 tps
- latency: 1.407 ± 0.132 ms

sslmode=disable

- throughput
- latency
### SSL Costs

| Negotiation and re-negotiation | Cryptographic functions | Certificate |

### Benefits

- Confidentiality
- Integrity
- Authentication

### Protocol

#### SSL Costs

<table>
<thead>
<tr>
<th>SSL</th>
<th>time &amp; €</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL</td>
<td>negotiation and re-negotiation</td>
</tr>
<tr>
<td>SSL</td>
<td>cryptographic functions</td>
</tr>
<tr>
<td>SSL</td>
<td>certificate</td>
</tr>
</tbody>
</table>

#### Benefits

- Confidentiality
- Integrity
- Authentication

### Performance

<table>
<thead>
<tr>
<th>sslmode=disable</th>
<th>sslmode=require</th>
</tr>
</thead>
<tbody>
<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>clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td></td>
</tr>
<tr>
<td>latency</td>
<td></td>
</tr>
</tbody>
</table>
### Protocol

<table>
<thead>
<tr>
<th>SSL Costs</th>
<th>Benefits</th>
</tr>
</thead>
<tbody>
<tr>
<td>negotiation and re-negotiation</td>
<td>Confidentiality</td>
</tr>
<tr>
<td>cryptographic functions</td>
<td>Integrity</td>
</tr>
<tr>
<td>certificate?</td>
<td>Authentication</td>
</tr>
</tbody>
</table>

#### SSL Costs

- `sslmode=require`  
  - throughput: 709.7 tps
  - latency: 1.407 ± 0.132 ms

- `sslmode/disable`  
  - throughput
  - latency
SSL Costs

- negotiation and re-negotiation
- cryptographic functions
- certificate

Benefits

- Confidentiality
- Integrity
- Authentication

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

sslmode=require

SSL

- throughput: 709.7 tps
- latency: 1.407 ± 0.132 ms

sslmode=disable

clear

- throughput
- latency
### Protocol

#### SSL or not

<table>
<thead>
<tr>
<th>SSL Costs</th>
<th>time &amp; €</th>
</tr>
</thead>
<tbody>
<tr>
<td>- negotiation and re-negotiation</td>
<td></td>
</tr>
<tr>
<td>- cryptographic functions</td>
<td></td>
</tr>
<tr>
<td>- certificate</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Snake Oil!</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Confidentiality</td>
<td></td>
</tr>
<tr>
<td>- Integrity</td>
<td></td>
</tr>
<tr>
<td>- Authentication</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>sslmode=require</th>
<th>SSL</th>
</tr>
</thead>
<tbody>
<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>clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>781.6 tps</td>
</tr>
<tr>
<td>latency</td>
<td>1.277 ± 0.034 ms</td>
</tr>
</tbody>
</table>
Protocol

--- prepare once in session

PREPARE Abal(INT) AS
  SELECT abalance
  FROM pgbench_accounts
  WHERE aid=$1;

--- execute multiple times...

EXECUTE Abal(1);
EXECUTE Abal(5432);
EXECUTE Abal(18);

### Simple vs Prepared

<table>
<thead>
<tr>
<th>ro3.sql</th>
<th>simple</th>
</tr>
</thead>
<tbody>
<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>ro3.sql</th>
<th>prepared</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td></td>
</tr>
<tr>
<td>latency</td>
<td></td>
</tr>
</tbody>
</table>
Protocol

-- prepare once in session
PREPARE Abal(INT) AS
  SELECT abalance
  FROM pgbench_accounts
  WHERE aid=$1;

-- execute multiple times...
EXECUTE Abal(1);
EXECUTE Abal(5432);
EXECUTE Abal(18);

Prepare
- temporary one-cmd function
- factor out parse cost
- keep plan and execute
- pgbench -M prepared...

<table>
<thead>
<tr>
<th>ro3.sql</th>
<th>simple</th>
</tr>
</thead>
<tbody>
<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>ro3.sql</th>
<th>prepared</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td></td>
</tr>
<tr>
<td>latency</td>
<td></td>
</tr>
</tbody>
</table>
**Protocol**

---

**Simple vs Prepared**

---

--- **prepare once in session**

PREPARE Abal(INT) AS

`SELECT abalance`  
`FROM pgbench_accounts`  
`WHERE aid=$1;`

--- execute multiple times...

EXECUTE Abal(1);  
EXECUTE Abal(5432);  
EXECUTE Abal(18);

---

---

### Prepare

- temporary one-cmd function
- factor out `parse` cost
- keep `plan` and `execute`
- `pgbench -M prepared` ...

---

---

<table>
<thead>
<tr>
<th>ro3.sql</th>
<th><strong>simple</strong></th>
<th><strong>prepared</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>throughput</strong></td>
<td>709.7 tps</td>
<td>860.0 tps</td>
</tr>
<tr>
<td><strong>latency</strong></td>
<td>1.407 ± 0.132 ms</td>
<td>1.161 ± 0.082 ms</td>
</tr>
</tbody>
</table>
Performance Comparisons

Four Query Combination Tricks
Query Combination

-- update table
UPDATE pgbench_accounts
  SET abalance = abalance + :delta
  WHERE aid = :aid;

-- get updated data
SELECT abalance
  FROM pgbench_accounts
  WHERE aid = :aid;

-- combined
UPDATE pgbench_accounts
  SET abalance = abalance + :delta
  WHERE aid = :aid
  RETURNING abalance;

---

Standard

- throughput: 406.9 tps
- latency: 19.7 ± 12.3 ms

Combined Update

- throughput
- latency
Query Combination

-- update table
UPDATE pgbench_accounts
  SET abalance = abalance + :delta
  WHERE aid = :aid;

-- get updated data
SELECT abalance
  FROM pgbench_accounts
  WHERE aid = :aid;

-- combined
UPDATE pgbench_accounts
  SET abalance = abalance + :delta
  WHERE aid = :aid
RETURNING abalance;

---

UPDATE & SELECT

- return updated rows
- one parse, plan, execute

---

Standard
- throughput: 406.9 tps
- latency: 19.7 ± 12.3 ms

Combined Update
- throughput
- latency
Query Combination

-- update table
UPDATE pgbench_accounts
  SET abalance = abalance + :delta
  WHERE aid = :aid;

-- get updated data
SELECT abalance
  FROM pgbench_accounts
  WHERE aid = :aid;

-- combined
UPDATE pgbench_accounts
  SET abalance = abalance + :delta
  WHERE aid = :aid
  RETURNING abalance;

UPDATE RETURNING Option

- return updated rows
- one parse, plan, execute

Standard
- throughput 406.9 tps
- latency 19.7 ± 12.3 ms

Combined Update
- throughput
- latency
Query Combination

**UPDATE & SELECT**

-- update table
```
UPDATE pgbench_accounts
  SET abalance = abalance + :delta
  WHERE aid = :aid;
```

-- get updated data
```
SELECT abalance
  FROM pgbench_accounts
  WHERE aid = :aid;
```

-- combined
```
UPDATE pgbench_accounts
  SET abalance = abalance + :delta
  WHERE aid = :aid
RETURNING abalance;
```

**UPDATE RETURNING Option**

- return updated rows
- one parse, plan, execute

---

**Standard**

- throughput: 406.9 tps
- latency: 19.7 ± 12.3 ms

**Combined Update**

- throughput: 408.2 tps
- latency: 19.6 ± 8.7 ms
Client-combined SQL Queries

```
-- "ro3c.sql" pgbench script
\set aid random(1, 100000 * :scale)
\set tid random(1, 10 * :scale)
\set bid random(1, :scale)
BEGIN ;
SELECT abalance FROM pgbench_accounts WHERE aid=:aid ;
SELECT tbalance FROM pgbench_tellers WHERE tid=:tid ;
SELECT bbalance FROM pgbench_branches WHERE bid=:bid ;
COMMIT ;
```

<table>
<thead>
<tr>
<th>ro3.sql</th>
<th>standard</th>
<th>ro3c.sql</th>
<th>combined</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>709.7 tps</td>
<td>throughput</td>
<td></td>
</tr>
<tr>
<td>latency</td>
<td>1.407 ± 0.132 ms</td>
<td>latency</td>
<td></td>
</tr>
</tbody>
</table>

Combine

- embedded semi-colon ;
- request with multiple queries
- response with list of results
- avoid request-response loop
Client-combined SQL Queries

--- "ro3c.sql" pgbench script

```sql
\set aid random(1, 100000 * :scale)
\set tid random(1, 10 * :scale)
\set bid random(1, :scale)
BEGIN \\
SELECT abalance FROM pgbench_accounts WHERE aid=:aid \\
SELECT tbalance FROM pgbench_tellers WHERE tid=:tid \\
SELECT bbalance FROM pgbench_branches WHERE bid=:bid \\
COMMIT;
```

Combine with

- embedded semi-colon ;
- request with multiple queries
- response with list of results
- avoid request-response loop

<table>
<thead>
<tr>
<th>ro3.sql</th>
<th>standard</th>
</tr>
</thead>
<tbody>
<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>ro3c.sql</th>
<th>combined</th>
</tr>
</thead>
<tbody>
<tr>
<td>• throughput</td>
<td></td>
</tr>
<tr>
<td>• latency</td>
<td></td>
</tr>
</tbody>
</table>

---
Client-combined SQL Queries

-- "ro3c.sql" pgbench script
\set aid random(1, 100000 * :scale)
\set tid random(1, 10 * :scale)
\set bid random(1, :scale)
BEGIN ;
SELECT abalance FROM pgbench_accounts WHERE aid=:aid ;
SELECT tbalance FROM pgbench_tellers WHERE tid=:tid ;
SELECT bbalance FROM pgbench_branches WHERE bid=:bid ;
COMMIT;

<table>
<thead>
<tr>
<th>ro3.sql</th>
<th>standard</th>
</tr>
</thead>
<tbody>
<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>ro3c.sql</th>
<th>combined</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>1,311.5 tps</td>
</tr>
<tr>
<td>latency</td>
<td>0.748 ± 0.132 ms</td>
</tr>
</tbody>
</table>

- combined with \;
  - embedded semi-colon ;
  - request with multiple queries
  - response with list of results
  - avoid request-response loop
Server-Side SQL queries

CREATE TYPE Balances
    AS (abal INT, tbal INT, bbal INT);

CREATE FUNCTION getBalSQL(INT, INT, INT)
    RETURNS Balances AS $$
SELECT
    (SELECT abalance
     FROM pgbench.accounts WHERE aid=$1),
    (SELECT tbalance
     FROM pgbench.tellers WHERE tid=$2),
    (SELECT bbalance
     FROM pgbench.branches WHERE bid=$3)
$$ LANGUAGE SQL;

-- "ro3sf.sql" pgbench script
\set aid random(1, 100000 * :scale)
\set tid random(1, 10 * :scale)
\set bid random(1, :scale)
SELECT getBalSQL(:aid, :tid, :bid);

<table>
<thead>
<tr>
<th>ro3.sql</th>
<th>standard</th>
</tr>
</thead>
<tbody>
<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>ro3sf.sql</th>
<th>SQL call</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td></td>
</tr>
<tr>
<td>latency</td>
<td></td>
</tr>
</tbody>
</table>
Server-Side SQL queries

CREATE TYPE Balances
    AS (abal INT, tbal INT, bbal INT);

CREATE FUNCTION getBalSQL(INT, INT, INT)
    RETURNS Balances AS $$
SELECT
    (SELECT abalance
     FROM pgbench.accounts WHERE aid=$1),
    (SELECT tbalance
     FROM pgbench.tellers WHERE tid=$2),
    (SELECT bbalance
     FROM pgbench.branches WHERE bid=$3)
$$ LANGUAGE SQL;

--- "ro3sf.sql" pgbench script
\set aid random(1, 100000 * :scale)
\set tid random(1, 10 * :scale)
\set bid random(1, :scale)
SELECT getBalSQL(:aid, :tid, :bid);

```
ro3.sql  standard

- throughput  709.7 tps
- latency      1.407 ± 0.132 ms
```

```
ro3sf.sql  SQL call

- throughput
- latency
```
Server-Side SQL queries

```sql
CREATE TYPE Balances
  AS (abal INT, tbal INT, bbal INT);

CREATE FUNCTION getBalSQL(INT, INT, INT)
RETURNS Balances AS $$
SELECT
  (SELECT abalance
   FROM pgbench.accounts WHERE aid=$1),
  (SELECT tbalance
   FROM pgbench.tellers WHERE tid=$2),
  (SELECT bbalance
   FROM pgbench.branches WHERE bid=$3)
$$ LANGUAGE SQL;
```

---

```
-- "ro3sf.sql" pgbench script
\set aid random(1, 100000 * :scale)  
\set tid random(1, 10 * :scale)  
\set bid random(1, :scale)  
SELECT getBalSQL(:aid, :tid, :bid);
```
CREATE FUNCTION getBalPL(a INT, t INT, b INT)
RETURNS Balances AS $$
DECLARE
  abal INT; tbal INT; bbal INT;
BEGIN
  SELECT abalance INTO abal
  FROM pgbench.accounts WHERE aid=a;
  SELECT tbalance INTO tbal
  FROM pgbench.tellers WHERE tid=t;
  SELECT bbalance INTO bbal
  FROM pgbench.branches WHERE bid=b;
  RETURN (abal, tbal, bbal)::Balances;
END;
$$ LANGUAGE PLpgSQL;

-- "ro3pf.sql" pgbench script
\set aid random(1, 100000 * :scale)
\set tid random(1, 10 * :scale)
\set bid random(1, :scale)
SELECT getBalPL(:aid, :tid, :bid);

<table>
<thead>
<tr>
<th>ro3.sql</th>
<th>standard</th>
</tr>
</thead>
<tbody>
<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>ro3pf.sql</th>
<th>PL/pgSQL call</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td></td>
</tr>
<tr>
<td>latency</td>
<td></td>
</tr>
</tbody>
</table>
CREATE FUNCTION getBalPL(a INT, t INT, b INT)
RETURNS Balances AS $$
DECLARE
abal INT; tbal INT; bbal INT;
BEGIN
SELECT abalance INTO abal
FROM pgbench.accounts WHERE aid=a;
SELECT tbalance INTO tbal
FROM pgbench.tellers WHERE tid=t;
SELECT bbalance INTO bbal
FROM pgbench.branches WHERE bid=b;
RETURN (abal, tbal, bbal)::Balances;
END;
$$ LANGUAGE PLpgSQL;

-- "ro3pf.sql" pgbench script
\set aid random(1, 100000 * :scale)
\set tid random(1, 10 * :scale)
\set bid random(1, :scale)
SELECT getBalPL(:aid, :tid, :bid);

<table>
<thead>
<tr>
<th>ro3.sql</th>
<th>standard</th>
</tr>
</thead>
<tbody>
<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>ro3pf.sql</th>
<th>PL/pgSQL call</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td></td>
</tr>
<tr>
<td>latency</td>
<td></td>
</tr>
</tbody>
</table>
Server-Side PL/pgSQL queries

CREATE FUNCTION getBalPL(a INT, t INT, b INT) RETURNS Balances AS $$
DECLARE
  abal INT; tbal INT; bbal INT;
BEGIN
  SELECT abalance INTO abal
  FROM pgbench.accounts WHERE aid=a;
  SELECT tbalance INTO tbal
  FROM pgbench.tellers WHERE tid=t;
  SELECT bbalance INTO bbal
  FROM pgbench.branches WHERE bid=b;
  RETURN (abal, tbal, bbal)::Balances;
END;
$$ LANGUAGE PLpgSQL;

-- "ro3pf.sql" pgbench script
\set aid random(1, 100000 * :scale)
\set tid random(1, 10 * :scale)
\set bid random(1, :scale)
SELECT getBalPL(:aid, :tid, :bid);

ro3.sql

- throughput: 709.7 tps
- latency: 1.407 ± 0.132 ms

ro3pf.sql

- throughput: 2,485.5 tps
- latency: 0.400 ± 0.055 ms
Server-Side PL/pgSQL queries

```sql
CREATE FUNCTION getBalPL(a INT, t INT, b INT)
RETURNS Balances AS $$
DECLARE
    abal INT; tbal INT; bbal INT;
BEGIN
    SELECT abalance INTO abal
    FROM pgbench.accounts WHERE aid=a;
    SELECT tbalance INTO tbal
    FROM pgbench.tellers WHERE tid=t;
    SELECT bbalance INTO bbal
    FROM pgbench.branches WHERE bid=b;
    RETURN (abal, tbal, bbal)::Balances;
END;
$$ LANGUAGE PLpgSQL;
```

```sql
-- "ro3pf.sql" pgbench script
\set aid random(1, 100000 * :scale)
\set tid random(1, 10 * :scale)
\set bid random(1, :scale)
SELECT getBalPL(:aid, :tid, :bid);
```

PL/pgSQL caches plans!

<table>
<thead>
<tr>
<th>ro3.sql</th>
<th>standard</th>
</tr>
</thead>
<tbody>
<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>ro3pf.sql</th>
<th>PL/pgSQL call</th>
</tr>
</thead>
<tbody>
<tr>
<td>throughput</td>
<td>2,485.5 tps</td>
</tr>
<tr>
<td>latency</td>
<td>0.400 ± 0.055 ms</td>
</tr>
</tbody>
</table>
Performance Comparisons

Reducing Server Distance
## Interconnection

<table>
<thead>
<tr>
<th>Interconnection Type</th>
<th>Latency</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LAN</strong> Local Area Network</td>
<td><strong>Ethernet</strong></td>
<td><strong>100.3 tps</strong></td>
</tr>
<tr>
<td><strong>LO</strong> loopback interface</td>
<td><strong>localhost</strong></td>
<td></td>
</tr>
<tr>
<td><strong>IPC</strong> Inter-Process Communication</td>
<td><strong>Unix domain socket</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TPC-B-Like</th>
<th>on HDD</th>
<th>Read-Only 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LAN</strong></td>
<td><strong>100.3 tps</strong></td>
<td><strong>LAN</strong></td>
</tr>
<tr>
<td><strong>LO</strong></td>
<td></td>
<td><strong>709.7 tps</strong></td>
</tr>
<tr>
<td><strong>IPC</strong></td>
<td></td>
<td><strong>1.4 ms</strong></td>
</tr>
</tbody>
</table>
Client-Server Distance

Interconnection

**LAN**  Local Area Network

**LO**  loopback interface

**IPC**  Inter-Process Communication

---

**TPC-B-Like on HDD**

<table>
<thead>
<tr>
<th>Connection</th>
<th>Reads/s</th>
<th>Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>100.3 tps</td>
<td>9.9 ms</td>
</tr>
<tr>
<td>LO</td>
<td>114.5 tps</td>
<td>8.7 ms</td>
</tr>
<tr>
<td>IPC</td>
<td>113.5 tps</td>
<td>8.8 ms</td>
</tr>
</tbody>
</table>

**Read-Only 3**

<table>
<thead>
<tr>
<th>Connection</th>
<th>Reads/s</th>
<th>Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>709.7 tps</td>
<td>1.4 ms</td>
</tr>
<tr>
<td>LO</td>
<td>709.7 tps</td>
<td>1.4 ms</td>
</tr>
<tr>
<td>IPC</td>
<td>709.7 tps</td>
<td>1.4 ms</td>
</tr>
</tbody>
</table>

---

*Ethernet*

*localhost*

*Unix domain socket*
### Client-Server Distance

#### Interconnection

<table>
<thead>
<tr>
<th>Interconnection</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LAN</strong> Local Area Network</td>
<td><strong>Ethernet</strong></td>
</tr>
<tr>
<td><strong>LO</strong> loopback interface</td>
<td><strong>localhost</strong></td>
</tr>
<tr>
<td><strong>IPC</strong> Inter-Process Communication</td>
<td><strong>Unix domain socket</strong></td>
</tr>
</tbody>
</table>

#### TPC-B-Like on SSD

<table>
<thead>
<tr>
<th>Interconnection</th>
<th>Read-Only 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LAN</strong></td>
<td>709.7 tps 1.4 ms</td>
</tr>
<tr>
<td><strong>LO</strong></td>
<td>1,133.3 tps 0.9 ms</td>
</tr>
<tr>
<td><strong>IPC</strong></td>
<td>1,243.1 tps 0.8 ms</td>
</tr>
</tbody>
</table>
# Client-Server Distance

## Interconnection

<table>
<thead>
<tr>
<th>Interconnection Type</th>
<th>LAN (Local Area Network)</th>
<th>LO (loopback interface)</th>
<th>IPC (Inter-Process Communication)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet</td>
<td>403.8 tps 2.4 ms</td>
<td>1,133.3 tps 0.9 ms</td>
<td>1,243.1 tps 0.8 ms</td>
</tr>
<tr>
<td>localhost (LO)</td>
<td>709.7 tps 1.4 ms</td>
<td>2,515.3 tps 0.4 ms</td>
<td>3,607.6 tps 0.3 ms</td>
</tr>
<tr>
<td>Unix domain socket</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## TPC-B-Like on SSD

<table>
<thead>
<tr>
<th>Interconnection Type</th>
<th>LAN (Local Area Network)</th>
<th>LO (loopback interface)</th>
<th>IPC (Inter-Process Communication)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>403.8 tps 2.4 ms</td>
<td>1,133.3 tps 0.9 ms</td>
<td>1,243.1 tps 0.8 ms</td>
</tr>
<tr>
<td>LO</td>
<td>709.7 tps 1.4 ms</td>
<td>2,515.3 tps 0.4 ms</td>
<td>3,607.6 tps 0.3 ms</td>
</tr>
<tr>
<td>IPC</td>
<td>1,243.1 tps 0.8 ms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Read-Only 3

<table>
<thead>
<tr>
<th>Interconnection Type</th>
<th>LAN (Local Area Network)</th>
<th>LO (loopback interface)</th>
<th>IPC (Inter-Process Communication)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>709.7 tps 1.4 ms</td>
<td>2,515.3 tps 0.4 ms</td>
<td>3,607.6 tps 0.3 ms</td>
</tr>
<tr>
<td>LO</td>
<td>1,133.3 tps 0.9 ms</td>
<td>2,515.3 tps 0.4 ms</td>
<td>3,607.6 tps 0.3 ms</td>
</tr>
<tr>
<td>IPC</td>
<td>1,243.1 tps 0.8 ms</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance Comparisons

Performance Scalability
# Clients Scalability

**Base**

*Read-Only 3 – remote SSL simple queries*

- **Best Throughput**

- **Best Latency**

- **Compromise**
# Clients Scalability

**Base**

- **Best Throughput**
  - 37,639 tps
  - 4.103 ms
  - 156/4

- **Best Latency**
  - **Compromise**

---

**Read-Only 3 – remote SSL simple queries**

![Graph showing scalability results](image-url)

- **Number of clients**
- **Throughput in thousands tps**
- **Latency in ms**

---

**Conclusion**

- Wrap-Up
- Lessons
- Contributions
# Clients Scalability

## Base

**Best Throughput**
- 37,639 tps
- 4.103 ms
- 156/4

**Best Latency**
- 5,748 tps
- 1.042 ms
- 6/1

**Compromise**
# Clients Scalability

**Base**

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Throughput</th>
<th>Latency</th>
<th>Clients</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Best Throughput</strong></td>
<td>37,639 tps</td>
<td>4.103 ms</td>
<td>156/4</td>
</tr>
<tr>
<td><strong>Best Latency</strong></td>
<td>5,748 tps</td>
<td>1.042 ms</td>
<td>6/1</td>
</tr>
<tr>
<td><strong>Compromise</strong></td>
<td>31,494 tps</td>
<td>1.837 ms</td>
<td>58/4</td>
</tr>
</tbody>
</table>

**Read-Only 3 – remote SSL simple queries**

![Graph showing scalability results](attachment:graph.png)
# Clients Scalability

**Best**

- **Best Throughput**
- **Best Latency**
- **Compromise**

---

*Read-Only 3 – remote noSSL prepared PL call*
# Clients Scalability

**Best Throughput**

181,503 tps 0.766 ms 140/4

**Best Latency**

**Compromise**

---

*Read-Only 3 – remote noSSL prepared PL call*
Best Throughput

181,503 tps 0.766 ms 140/4

Best Latency

39,232 tps 0.254 ms 10/2

Compromise

Read-Only 3 – remote noSSL prepared PL call
# Clients Scalability

**Best Throughput**
181,503 tps 0.766 ms 140/4

**Best Latency**
39,232 tps 0.254 ms 10/2

**Compromise**
156,945 tps 0.381 ms 60/4

Read-Only 3 – remote noSSL prepared PL call
Performance Comparisons

Miscellaneous Settings
### Application

<table>
<thead>
<tr>
<th>connection</th>
<th>persistence</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache</td>
<td>Memcached Redis</td>
</tr>
</tbody>
</table>

### Postgres configuration

<table>
<thead>
<tr>
<th>disk</th>
<th>block_size random_page_cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory</td>
<td>shared_buffers effective_cache_size huge_pages</td>
</tr>
<tr>
<td>checkpoint</td>
<td>_timeout _completion_target _flush_after</td>
</tr>
<tr>
<td>wal</td>
<td>max_wal_size</td>
</tr>
</tbody>
</table>
### Application

**application**
- connection persistence
- cache Memcached Redis

### Postgres configuration

**disk**
- block_size
- random_page_cost

**memory**
- shared_buffers
- effective_cache_size
- huge_pages

**checkpoint**
- timeout
- completion_target
- flush_after

**wal**
- max_wal_size
Miscellaneous Settings

OS & Hardware

tweak and choose

OS

- FS  XFS  ext4  Btrfs  ZFS, mount options
- IO  io scheduler, queue length, write delay, dirty bytes...
- others  NUMA, ...

Hardware

expensive is (probably) better

- diskS  tables wal logs, HDD-with-cache, SSD
- tweaking  read ahead, write flush
- RAID  with large caches, BBU
<table>
<thead>
<tr>
<th>OS</th>
<th>tweak and choose</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS</td>
<td>XFS, ext4, Btrfs, ZFS, mount options</td>
</tr>
<tr>
<td>IO</td>
<td>io scheduler, queue length, write delay, dirty bytes...</td>
</tr>
<tr>
<td>others</td>
<td>NUMA, ...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hardware</th>
<th>expensive is (probably) better</th>
</tr>
</thead>
<tbody>
<tr>
<td>disks</td>
<td>tables wal logs, HDD-with-cache, SSD</td>
</tr>
<tr>
<td>tweaking</td>
<td>read ahead, write flush</td>
</tr>
<tr>
<td>RAID</td>
<td>with large caches, BBU</td>
</tr>
</tbody>
</table>
Conclusion
Wrap-up

```bash
pgbench -j 1 -c 1 ...
```

<table>
<thead>
<tr>
<th></th>
<th>TPC-B-like</th>
<th>Read-Only 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tps</td>
<td>ms</td>
</tr>
<tr>
<td>HDD -c SSL</td>
<td>36.1</td>
<td>27.7</td>
</tr>
<tr>
<td>HDD -c noSSL</td>
<td>56.4</td>
<td>17.7</td>
</tr>
<tr>
<td>HDD SSL</td>
<td>105.4</td>
<td>9.5</td>
</tr>
<tr>
<td>SSD SSL</td>
<td>403.8</td>
<td>2.47</td>
</tr>
<tr>
<td>SSD noSSL</td>
<td>465.4</td>
<td>2.15</td>
</tr>
<tr>
<td>... + prepared</td>
<td>548.1</td>
<td>1.82</td>
</tr>
<tr>
<td>– returning</td>
<td>529.4</td>
<td>1.89</td>
</tr>
<tr>
<td>... + prepared</td>
<td>681.2</td>
<td>1.47</td>
</tr>
<tr>
<td>– combined</td>
<td>857.8</td>
<td>1.15</td>
</tr>
<tr>
<td>– SQL func</td>
<td>940.3</td>
<td>1.06</td>
</tr>
<tr>
<td>... + prepared</td>
<td>957.9</td>
<td>1.04</td>
</tr>
<tr>
<td>– PL func</td>
<td>1,279.4</td>
<td>0.78</td>
</tr>
<tr>
<td>... + prepared</td>
<td>1,323.2</td>
<td>0.75</td>
</tr>
<tr>
<td>localhost</td>
<td>1,907.6</td>
<td>0.52</td>
</tr>
<tr>
<td>socket</td>
<td>2,273.1</td>
<td>0.44</td>
</tr>
</tbody>
</table>

- connection
- HDD to SSD
- SSL to none
- simple to prepared
- combinations...
- remote to local
pgbench -j 1 -c 1 ...

- connection
- HDD to SSD
- SSL to none
- simple to prepared
- combinations...
- remote to local

×3 to ×14
Wrap-up

```
pgbench -j 1 -c 1 ...
```

<table>
<thead>
<tr>
<th>Connection</th>
<th>tps</th>
<th>ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD to SSD</td>
<td>36.1</td>
<td>27.7</td>
</tr>
<tr>
<td>SSL to none</td>
<td>56.4</td>
<td>17.7</td>
</tr>
<tr>
<td>Simple to Prepared</td>
<td>105.4</td>
<td>9.5</td>
</tr>
<tr>
<td>Remote to Local</td>
<td>403.8</td>
<td>2.47</td>
</tr>
<tr>
<td>Local</td>
<td>465.4</td>
<td>2.15</td>
</tr>
<tr>
<td>Remote</td>
<td>548.1</td>
<td>1.82</td>
</tr>
</tbody>
</table>

```
<table>
<thead>
<tr>
<th>Read-Only 3</th>
<th>tps</th>
<th>ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD -c SSL</td>
<td>52.7</td>
<td>18.96</td>
</tr>
<tr>
<td>HDD -c noSSL</td>
<td>110.1</td>
<td>9.08</td>
</tr>
<tr>
<td>HDD SSL</td>
<td>709.7</td>
<td>1.41</td>
</tr>
<tr>
<td>SSD SSL</td>
<td>695.1</td>
<td>1.44</td>
</tr>
<tr>
<td>SSD noSSL</td>
<td>820.1</td>
<td>1.22</td>
</tr>
<tr>
<td>... + prepared</td>
<td>974.0</td>
<td>1.02</td>
</tr>
<tr>
<td>returning</td>
<td>529.4</td>
<td>1.89</td>
</tr>
<tr>
<td>... + prepared</td>
<td>681.2</td>
<td>1.47</td>
</tr>
<tr>
<td>combined</td>
<td>857.8</td>
<td>1.15</td>
</tr>
<tr>
<td>SQL func</td>
<td>1,536.4</td>
<td>0.64</td>
</tr>
<tr>
<td>... + prepared</td>
<td>2,144.7</td>
<td>0.46</td>
</tr>
<tr>
<td>PL func</td>
<td>3,040.4</td>
<td>0.33</td>
</tr>
<tr>
<td>... + prepared</td>
<td>10,006.8</td>
<td>0.10</td>
</tr>
</tbody>
</table>

- 4 to =
###Wrap-up

```bash
gbench -j 1 -c 1 ...
```

<table>
<thead>
<tr>
<th>TPC-B-like</th>
<th>Read-Only 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tps</td>
</tr>
<tr>
<td>HDD -c SSL</td>
<td>36.1</td>
</tr>
<tr>
<td>HDD -c noSSL</td>
<td>56.4</td>
</tr>
<tr>
<td>HDD SSL</td>
<td>105.4</td>
</tr>
<tr>
<td>SSD SSL</td>
<td>403.8</td>
</tr>
<tr>
<td>SSD noSSL</td>
<td>465.4</td>
</tr>
<tr>
<td>... + prepared</td>
<td>548.1</td>
</tr>
<tr>
<td>... - returning</td>
<td>529.4</td>
</tr>
<tr>
<td>... + prepared</td>
<td>681.2</td>
</tr>
<tr>
<td>... - combined</td>
<td>857.8</td>
</tr>
<tr>
<td>... - SQL func</td>
<td>940.3</td>
</tr>
<tr>
<td>... + prepared</td>
<td>957.9</td>
</tr>
<tr>
<td>... - PL func</td>
<td>1,279.4</td>
</tr>
<tr>
<td>... + prepared</td>
<td>1,323.2</td>
</tr>
<tr>
<td>localhost</td>
<td>1,907.6</td>
</tr>
<tr>
<td>socket</td>
<td>2,273.1</td>
</tr>
</tbody>
</table>

- connection
- HDD to SSD
- SSL to none
- simple to prepared
- combinations...
- remote to local

+15% to +18%
## Wrap-up

pgbench -j 1 -c 1 ...

<table>
<thead>
<tr>
<th></th>
<th>TPC-B-like</th>
<th>Read-Only 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tps</td>
<td>ms</td>
</tr>
<tr>
<td>HDD -c SSL</td>
<td>36.1</td>
<td>27.7</td>
</tr>
<tr>
<td>HDD -c noSSL</td>
<td>56.4</td>
<td>17.7</td>
</tr>
<tr>
<td>HDD SSL</td>
<td>105.4</td>
<td>9.5</td>
</tr>
<tr>
<td>SSD SSL</td>
<td>403.8</td>
<td>2.47</td>
</tr>
<tr>
<td>SSD noSSL</td>
<td>465.4</td>
<td>2.15</td>
</tr>
<tr>
<td>... + prepared</td>
<td>548.1</td>
<td>1.82</td>
</tr>
<tr>
<td>– returning</td>
<td>529.4</td>
<td>1.89</td>
</tr>
<tr>
<td>... + prepared</td>
<td>681.2</td>
<td>1.47</td>
</tr>
<tr>
<td>– combined</td>
<td>857.8</td>
<td>1.15</td>
</tr>
<tr>
<td>– SQL func</td>
<td>940.3</td>
<td>1.06</td>
</tr>
<tr>
<td>... + prepared</td>
<td>957.9</td>
<td>1.04</td>
</tr>
<tr>
<td>– PL func</td>
<td>1,279.4</td>
<td>0.78</td>
</tr>
<tr>
<td>... + prepared</td>
<td>1,323.2</td>
<td>0.75</td>
</tr>
<tr>
<td>localhost</td>
<td>1,907.6</td>
<td>0.52</td>
</tr>
<tr>
<td>socket</td>
<td>2,273.1</td>
<td>0.44</td>
</tr>
</tbody>
</table>

- connection
- HDD to SSD
- SSL to none
- simple to prepared
- combinations...
- remote to local

+2% to +28%
**Wrap-up**

```bash
gbench -j 1 -c 1 ...
```

<table>
<thead>
<tr>
<th></th>
<th><strong>TPC-B-like</strong></th>
<th><strong>Read-Only 3</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tps</td>
<td>ms</td>
</tr>
<tr>
<td>HDD -c SSL</td>
<td>36.1</td>
<td>27.7</td>
</tr>
<tr>
<td>HDD -c noSSL</td>
<td>56.4</td>
<td>17.7</td>
</tr>
<tr>
<td>HDD SSL</td>
<td>105.4</td>
<td>9.5</td>
</tr>
<tr>
<td>SSD SSL</td>
<td>403.8</td>
<td>2.47</td>
</tr>
<tr>
<td>SSD noSSL</td>
<td>465.4</td>
<td>2.15</td>
</tr>
<tr>
<td>... + prepared</td>
<td>548.1</td>
<td>1.82</td>
</tr>
<tr>
<td>– returning</td>
<td>529.4</td>
<td>1.89</td>
</tr>
<tr>
<td>... + prepared</td>
<td>681.2</td>
<td>1.47</td>
</tr>
<tr>
<td>– combined</td>
<td>857.8</td>
<td>1.15</td>
</tr>
<tr>
<td>– SQL func</td>
<td>940.3</td>
<td>1.06</td>
</tr>
<tr>
<td>... + prepared</td>
<td>957.9</td>
<td>1.04</td>
</tr>
<tr>
<td>– PL func</td>
<td>1,279.4</td>
<td>0.78</td>
</tr>
<tr>
<td>... + prepared</td>
<td>1,323.2</td>
<td>0.75</td>
</tr>
<tr>
<td>localhost</td>
<td>1,907.6</td>
<td>0.52</td>
</tr>
<tr>
<td>socket</td>
<td>2,273.1</td>
<td>0.44</td>
</tr>
</tbody>
</table>

- connection
- HDD to SSD
- SSL to none
- simple to prepared
- combinations...
- remote to local

×3 to ×4
### TPC-B-like

<table>
<thead>
<tr>
<th></th>
<th>tps</th>
<th>ms</th>
<th></th>
<th>tps</th>
<th>ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDD -c SSL</td>
<td>36.1</td>
<td>27.7</td>
<td>52.7</td>
<td>18.96</td>
<td></td>
</tr>
<tr>
<td>HDD -c noSSL</td>
<td>56.4</td>
<td>17.7</td>
<td>110.1</td>
<td>9.08</td>
<td></td>
</tr>
<tr>
<td>HDD SSL</td>
<td>105.4</td>
<td>9.5</td>
<td>709.7</td>
<td>1.41</td>
<td></td>
</tr>
<tr>
<td>SSD SSL</td>
<td>403.8</td>
<td>2.47</td>
<td>695.1</td>
<td>1.44</td>
<td></td>
</tr>
<tr>
<td>SSD noSSL</td>
<td>465.4</td>
<td>2.15</td>
<td>820.1</td>
<td>1.22</td>
<td></td>
</tr>
<tr>
<td>... + prepared</td>
<td>548.1</td>
<td>1.82</td>
<td>974.0</td>
<td>1.02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>529.4</td>
<td>1.89</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... + prepared</td>
<td>681.2</td>
<td>1.47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>... + prepared</td>
<td>857.8</td>
<td>1.15</td>
<td>1,536.4</td>
<td>0.64</td>
<td></td>
</tr>
<tr>
<td>... + prepared</td>
<td>940.3</td>
<td>1.06</td>
<td>1,818.1</td>
<td>0.55</td>
<td></td>
</tr>
<tr>
<td>... + prepared</td>
<td>957.9</td>
<td>1.04</td>
<td>2,144.7</td>
<td>0.46</td>
<td></td>
</tr>
<tr>
<td>... + prepared</td>
<td>1,279.4</td>
<td>0.78</td>
<td>2,778.0</td>
<td>0.36</td>
<td></td>
</tr>
<tr>
<td>localhost</td>
<td>1,907.6</td>
<td>0.52</td>
<td>10,006.8</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>socket</td>
<td>2,273.1</td>
<td>0.44</td>
<td>11,545.5</td>
<td>0.09</td>
<td></td>
</tr>
</tbody>
</table>

### Read-Only 3

- **connection**
- **HDD to SSD**
- **SSL to none**
- **simple to prepared**
- **combinations...**
- **remote to local**

**×1.7 to ×3.7**
### Wrap-up

```bash
pgbench -j 1 -c 1 ...
```

<table>
<thead>
<tr>
<th></th>
<th><strong>TPC-B-like</strong></th>
<th></th>
<th><strong>Read-Only 3</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>tps</td>
<td>ms</td>
<td>tps</td>
</tr>
<tr>
<td>HDD -c SSL</td>
<td>36.1</td>
<td>27.7</td>
<td>52.7</td>
</tr>
<tr>
<td>HDD -c noSSL</td>
<td>56.4</td>
<td>17.7</td>
<td>110.1</td>
</tr>
<tr>
<td>HDD SSL</td>
<td>105.4</td>
<td>9.5</td>
<td>709.7</td>
</tr>
<tr>
<td>SSD SSL</td>
<td>403.8</td>
<td>2.47</td>
<td>695.1</td>
</tr>
<tr>
<td>SSD noSSL</td>
<td>465.4</td>
<td>2.15</td>
<td>820.1</td>
</tr>
<tr>
<td>... + prepared</td>
<td>548.1</td>
<td>1.82</td>
<td>974.0</td>
</tr>
<tr>
<td>– returning</td>
<td>529.4</td>
<td>1.89</td>
<td>–</td>
</tr>
<tr>
<td>... + prepared</td>
<td>681.2</td>
<td>1.47</td>
<td>–</td>
</tr>
<tr>
<td>– combined</td>
<td>857.8</td>
<td>1.15</td>
<td>1,536.4</td>
</tr>
<tr>
<td>– SQL func</td>
<td>940.3</td>
<td>1.06</td>
<td>1,818.1</td>
</tr>
<tr>
<td>... + prepared</td>
<td>957.9</td>
<td>1.04</td>
<td>2,144.7</td>
</tr>
<tr>
<td>– PL func</td>
<td>1,279.4</td>
<td>0.78</td>
<td>2,778.0</td>
</tr>
<tr>
<td>... + prepared</td>
<td>1,323.2</td>
<td>0.75</td>
<td>3,040.4</td>
</tr>
<tr>
<td>localhost</td>
<td>1,907.6</td>
<td>0.52</td>
<td>10,006.8</td>
</tr>
<tr>
<td>socket</td>
<td>2,273.1</td>
<td>0.44</td>
<td>11,545.5</td>
</tr>
</tbody>
</table>

- connection
- HDD to SSD
- SSL to none
- simple to prepared
- combinations...
- remote to local

× 63 to × 219

and scaling effects
Lessons

Things to Bring Home

**NoTPS** not only TPS
latency-throughput compromise

Performance experiment and measure
*pgbench* is improving...

Postgres version
sorted and flushed checkpoints

High costs

RW load ACID

RO load *pg* as a cache manager

*in-memory OLTP load*

**latency matters!**

do not assume!

Postgres version 9.6!

network, parse & plan

SSD ≫ HDD

SSD = HDD
## Lessons

**Things to Bring Home**  

<table>
<thead>
<tr>
<th>NoTPS</th>
<th>not only TPS</th>
<th>latency-throughput compromise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>experiment and measure</td>
<td>pgbench is improving...</td>
</tr>
</tbody>
</table>

**Postgres version**  

- sorted and flushed checkpoints
- High costs
- RW load ACID
- RO load pg as a cache manager

**9.6!**

- network, parse & plan
- SSD ≫ HDD
- SSD = HDD

---

**in-memory OLTP load**

- latency matters!
- do not assume!
## Lessons

<table>
<thead>
<tr>
<th>Things to Bring Home</th>
<th>in-memory OLTP load</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NoTPS</strong></td>
<td>not only TPS</td>
</tr>
<tr>
<td></td>
<td>latency-throughput compromise</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>experiment and measure</td>
</tr>
<tr>
<td></td>
<td>pgbench is improving...</td>
</tr>
<tr>
<td><strong>Postgres</strong></td>
<td>version</td>
</tr>
<tr>
<td></td>
<td>sorted and flushed checkpoints</td>
</tr>
<tr>
<td>High</td>
<td>costs</td>
</tr>
<tr>
<td>RW load</td>
<td>ACID</td>
</tr>
<tr>
<td>RO load</td>
<td>pg as a cache manager</td>
</tr>
</tbody>
</table>

- **latency matters!**
- **do not assume!**
- **9.6!**
- **network, parse & plan**
- **SSD ≫ HDD**
- **SSD = HDD**

---

**Postgres**

**Latency**

F. Coelho

**Introduction**

**Subject**

**Application**

**Definitions**

**pgbench**

**Performance**

**Connection**

**Rate & Limit**

**Storage**

**Protocol**

**Combinations**

**Distance**

**Scalability**

**Miscellaneous**

**Conclusion**

**Wrap-Up**

**Lessons**

**Contributions**
# Lessons

## Things to Bring Home

<table>
<thead>
<tr>
<th>NoTPS</th>
<th>not only TPS latency-throughput compromise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>experiment and measure pgbench is improving. . .</td>
</tr>
<tr>
<td>Postgres</td>
<td>version sorted and flushed checkpoints</td>
</tr>
<tr>
<td>High</td>
<td>costs</td>
</tr>
<tr>
<td>RW load</td>
<td>ACID</td>
</tr>
<tr>
<td>RO load</td>
<td>pg as a cache manager</td>
</tr>
</tbody>
</table>

### in-memory OLTP load

- **NoTPS** not only TPS latency-throughput compromise
- **Performance** experiment and measure pgbench is improving. . .
- **Postgres** version sorted and flushed checkpoints
- **High** costs
- **RW load** ACID
- **RO load** pg as a cache manager

---

**Lessons**

- **Things to Bring Home**
  - in-memory OLTP load
  - NoTPS not only TPS latency-throughput compromise
  - Performance experiment and measure pgbench is improving. . .
  - Postgres version sorted and flushed checkpoints
  - High costs
  - RW load ACID
  - RO load pg as a cache manager

**Conclusion**

- Wrap-Up Lessons Contributions

---

**Notes**

- Postgres version 9.6!
- Sorted and flushed checkpoints
- High costs
- RW load ACID
- RO load pg as a cache manager
- in-memory OLTP load
- latency matters!
- do not assume!
- network, parse & plan
- SSD >> HDD
- SSD = HDD
## Lessons

### Things to Bring Home

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NoTPS</strong></td>
<td>not only TPS</td>
</tr>
<tr>
<td></td>
<td>latency-throughput compromise</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>experiment and measure</td>
</tr>
<tr>
<td></td>
<td><em>pgbench</em> is improving...</td>
</tr>
<tr>
<td><strong>Postgres</strong></td>
<td>version</td>
</tr>
<tr>
<td></td>
<td>sorted and flushed checkpoints</td>
</tr>
<tr>
<td><strong>High</strong></td>
<td>costs</td>
</tr>
<tr>
<td><strong>RW load</strong></td>
<td>ACID</td>
</tr>
<tr>
<td><strong>RO load</strong></td>
<td><em>pg</em> as a cache manager</td>
</tr>
</tbody>
</table>

---

*in-memory OLTP load*

- **latency matters!**
- **do not assume!**
- **9.6!**
- **SSD >> HDD**
- **SSD = HDD**

*Things to Bring Home*
## Lessons

### Things to Bring Home

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NoTPS</strong></td>
<td>not only TPS</td>
<td><strong>latency matters!</strong></td>
</tr>
<tr>
<td></td>
<td>latency-throughput compromise</td>
<td></td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>experiment and measure</td>
<td><strong>do not assume!</strong></td>
</tr>
<tr>
<td></td>
<td><em>pgbench</em> is improving...</td>
<td></td>
</tr>
<tr>
<td><strong>Postgres</strong></td>
<td>version</td>
<td>9.6!</td>
</tr>
<tr>
<td></td>
<td>sorted and flushed checkpoints</td>
<td></td>
</tr>
<tr>
<td><strong>High</strong></td>
<td>costs</td>
<td></td>
</tr>
<tr>
<td><strong>RW load</strong></td>
<td>ACID</td>
<td><strong>network, parse &amp; plan</strong></td>
</tr>
<tr>
<td><strong>RO load</strong></td>
<td>pg as a cache manager</td>
<td><strong>SSD ≫ HDD</strong></td>
</tr>
</tbody>
</table>

#### Observations

- Hot vs. cold data placement
- SSD vs. HDD

#### Conclusion

- Lessons:
  - Things to Bring Home
  - Contributions

#### Miscellaneous

- Conclusion
  - Wrap-Up
  - Lessons
  - Contributions
## Contributions

**provided or provoked**

### About Core

- sorted checkpoints
- flushed checkpoints

### About pgbench

- expressions
- mixed and weighted scripts and builtins
- better statistics
- improved usability
- rate and limit load
- debug...

\set ...

- \texttt{-b/-f ...@...}
- \texttt{stddev, per script...}
- \texttt{-c/-j -P...}
- \texttt{-R -L}
Contributions

About Core

- sorted checkpoints
- flushed checkpoints

About pgbench

- expressions
- mixed and weighted scripts and builtins
- better statistics
- improved usability
- rate and limit load
- debug.

\set ...
- \texttt{-b/-f ...@...}

\texttt{stddev, per script...}
- \texttt{-c/-j -P...}

\texttt{-R -L}
Measuring and Reducing Postgres Transaction Latency

Fabien Coelho

MINES ParisTech, PSL Research University

pgDay Paris – March 23, 2017