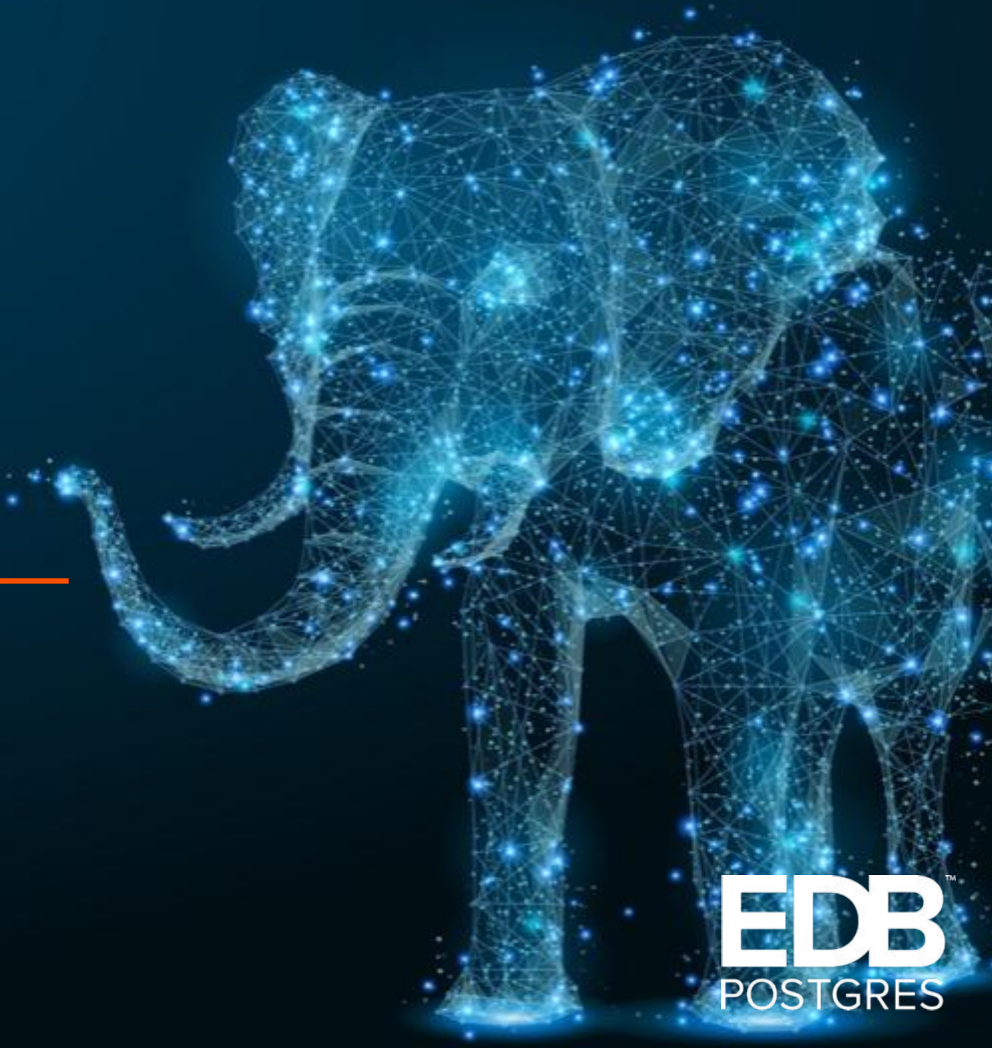
60000 TPS

How many CPUs ???



The results of an interesting research

Sebastiaan Mannem

EDB POSTGRES

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

But first

Who Am I:

DBA for 15 years

SQL Server

Oracle

Postgres

sebas@mannem.nl

sebastiaan.mannem@enterprisedb.com

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

But first

Who Am I:

Solutions Architect

EnterpriseDB

Professional Services

sebas@mannem.nl

sebastiaan.mannem@enterprisedb.com

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

But first

Who Am I:

Developer Open Source Software

https://docs.ansible.com/ansible/latest/modules/postgresql_pg_hba_module.html

https://github.com/bolcom/pgcdfga

https://github.com/bolcom/pg_replication_activity

https://github.com/sebasmannem/pg_cpu_test

sebas@mannem.nl

sebastiaan.mannem@enterprisedb.com

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

But first

Who Am I:

41 years old

Father of Three

Husband of one

sebas@mannem.nl

sebastiaan.mannem@enterprisedb.com

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Question: How much CPU is required to run 60000 TPS?

Answer 1: The Johnny5 answer

Please tell me

- What…
- How…
- When…
- etc.

Will this ever end in an advice?

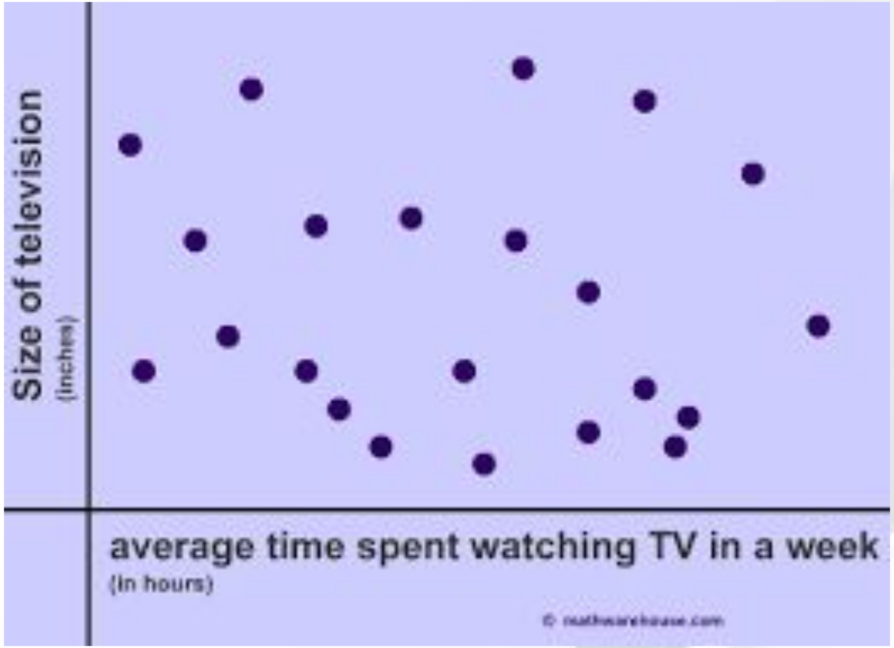
 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Question: How much CPU is required to run 60000 TPS?

Answer2: There is no correlation

There is no ~~correlation~~

CPU != bottleneck



CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Question: How much CPU is required to run 60000 TPS?

Answer 3: Another customer tested, and

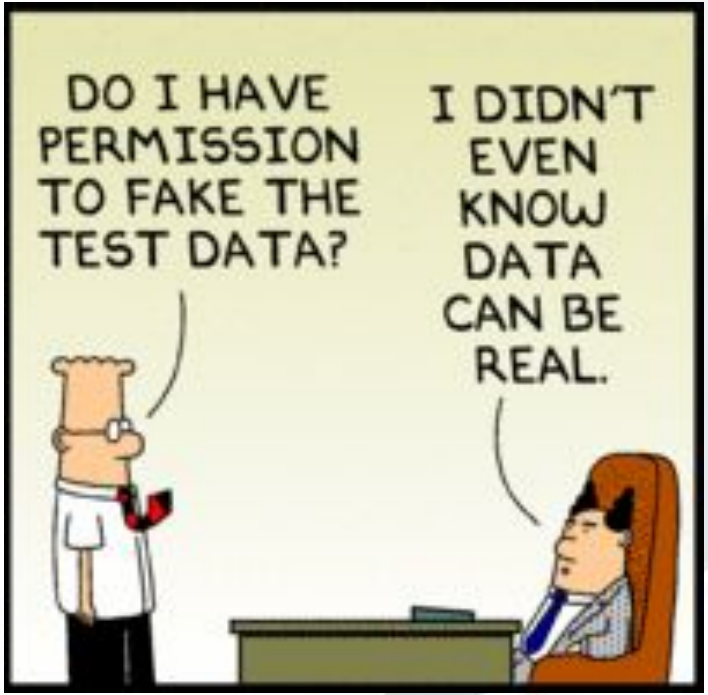
Doubtfully representative

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.
EDB POSTGRES

Question: How much CPU is required to run 60000 TPS?

Answer 4: You need to test

YES



 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

So let's test

60000 TPS, how many CPU's?

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Let's test

- Impact of increasing threads
- Impact of adding cores
- Is there a difference between processor architectures
- Impact of Preparing
- Impact of Transaction Control
- Difference between query types
- Impact of Storage options
  - SSD
  - Datadirectory on tmpfs
  - Wal directory on tmpfs
  - fsync = off
- Can we tune to get more TPS
- Impact of different programming languages

DEATH BY POWERPOINT

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

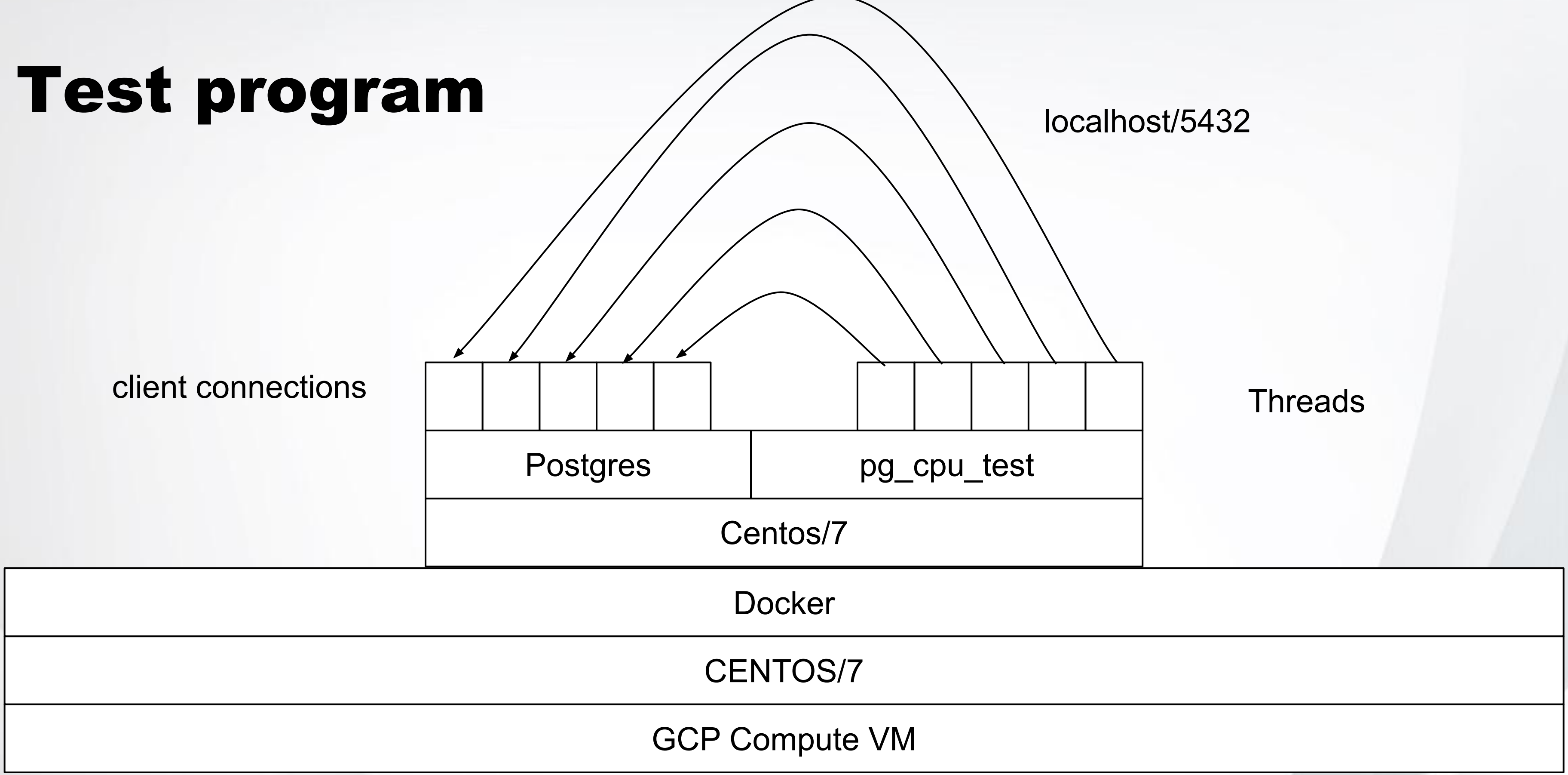
Let's test

- Impact of increasing threads and adding cores
- Impact of Preparing / Transaction Control / query types
- Impact of Storage options
- Programming languages

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Test program



 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

pg_cpu_test

https://github.com/sebasmannem/pg_cpu_test

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Load

- **Definition of a Transaction**
  - Complies to ACID
- **Test query types:**
  - empty, simple, read, write
  - compare the results
- **Test with and without Transaction Control**
  - and compare the results
- **Test with and without Prepare**
  - and compare the results

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.
EDB POSTGRES

Definition: Types of Queries

Without Transaction Control:

- **Simple Query:**
  - select $1;
- **Read:**
  - select col from table where col = $1;
- **Update:**
  - Update table set col = $1 where col = $1;

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Definition: Types of Queries

With Transaction Control:

- **Empty**
  - begin;
  - commit;
- **Simple**
  - begin;
  - select $1;
  - commit;
- **Read**
  - begin;
  - select col from table where col = $1;
  - commit;
- **Write**
  - begin;
  - update table set col = $1 where col = $1;
  - commit;

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.
EDB POSTGRES

Let's test

Adding cores

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

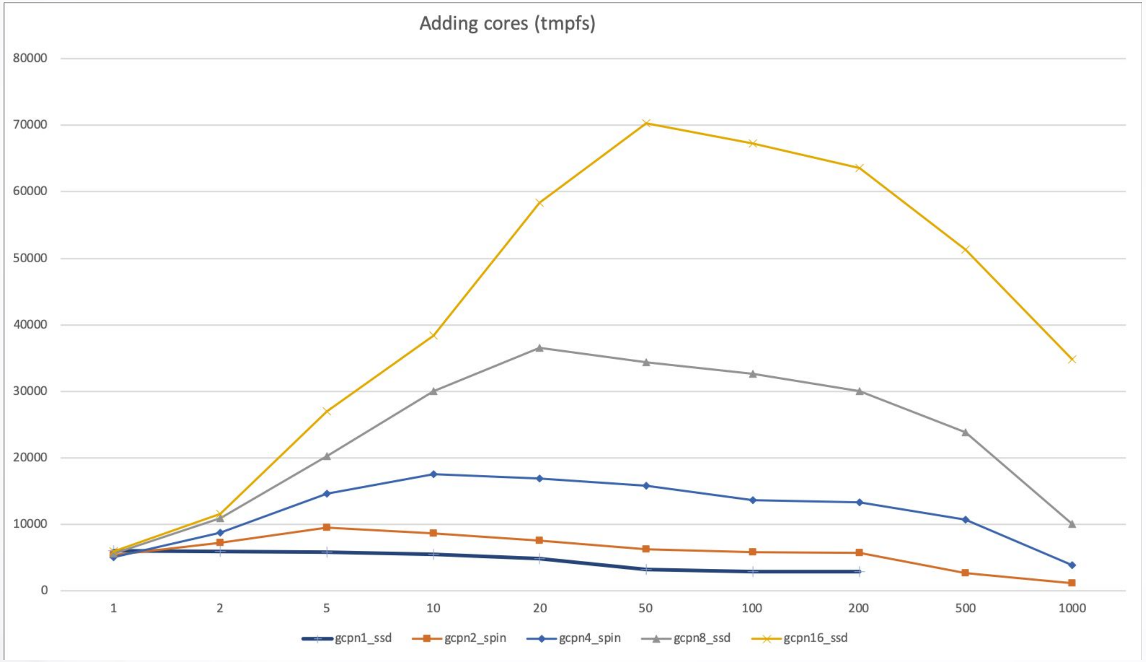
Let's test: Adding cores

- Run on GCP
- Run on n1, n2, n4, n8, n16
- Run on tmpfs, and normal storage
- Run Prepared, Transactional, write queries
- Run with 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000 threads

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Adding cores (tmpfs)



 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

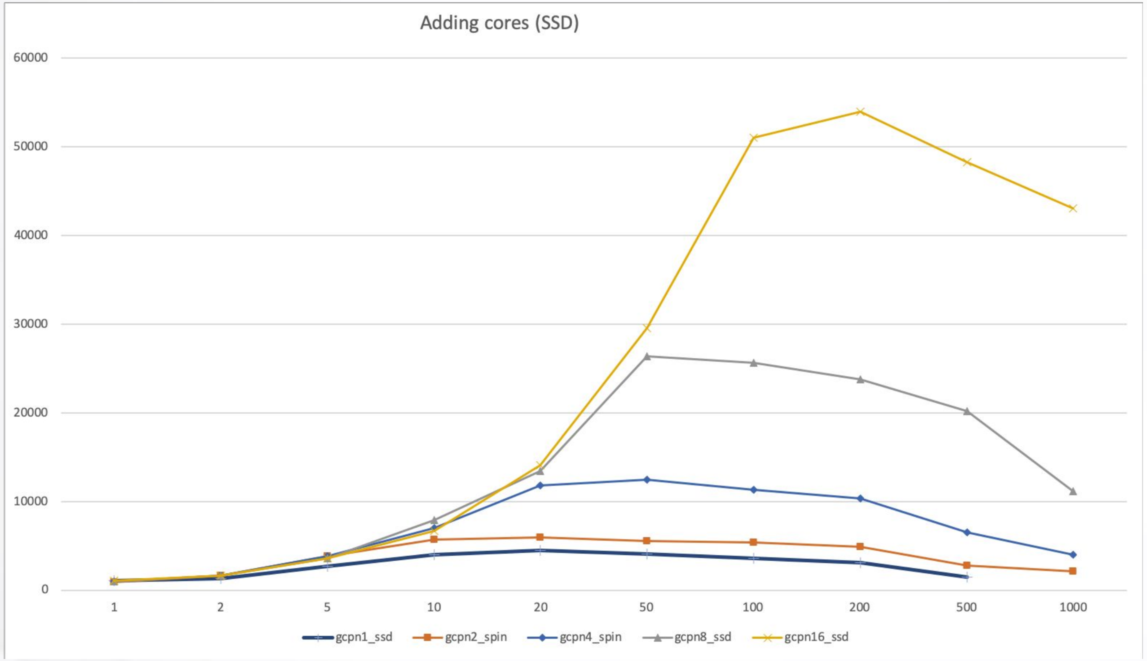
Correlate

#cores	Threads Peak	TPS Peak
2	5	10.000
4	10	18.000
8	20	37.000
13	38	60.000
16	50	70.000
32	100	140.000
64	200	280.000
128	500	500.000

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Adding cores (SSD)



CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Correlate

#cores	Threads Peak	TPS Peak
1	20	5000
2	20	6000
4	50	12.000
8	50	37.000
16	200	54.000
18	225	60.000

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Let's test

Poolers

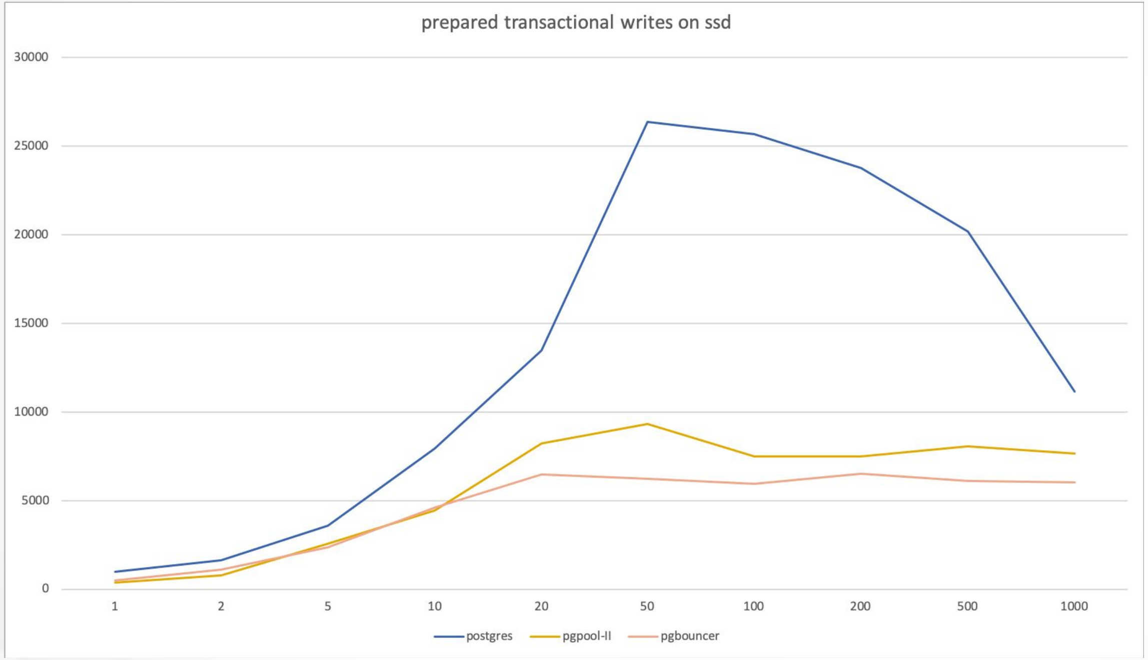
CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Let's test: Adding cores

- Run on GCP
- Run on n8
- Run on normal storage
- Run Prepared, Transactional, write queries
- Run with 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000 threads

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Poolers



 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Let's test

Transaction Control

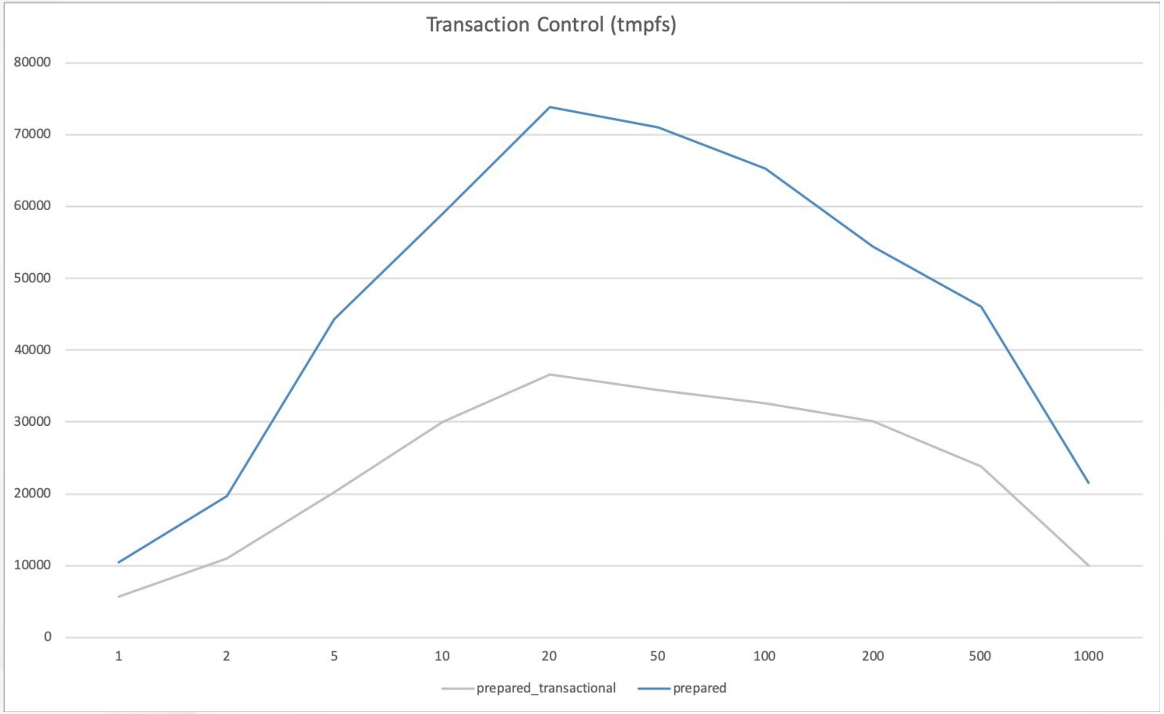
CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Let's test: Transaction Control

- Run on GCP (n8)
- Run on tmpfs / with SSD
- Run Prepared, write
- Run with 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000 threads

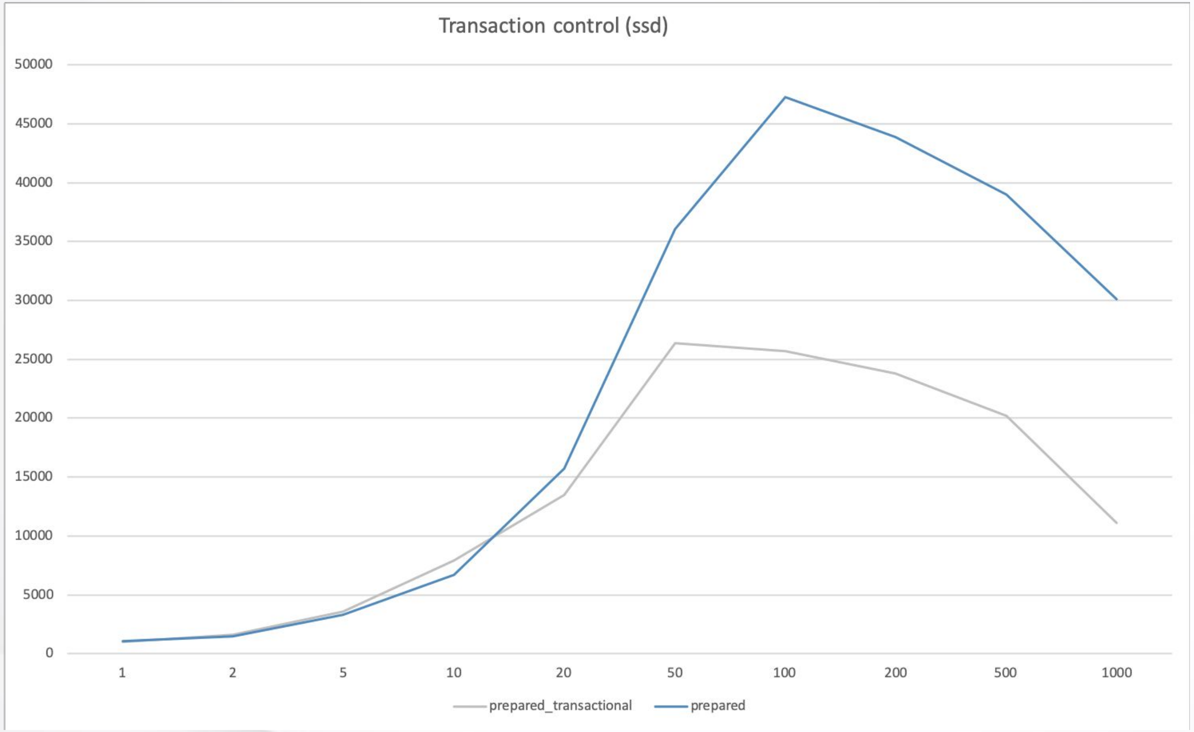
CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

With / without Transaction Control (tmpfs)



CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

With / without Transaction Control (SSD)



 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Let's test

Impact of storage

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

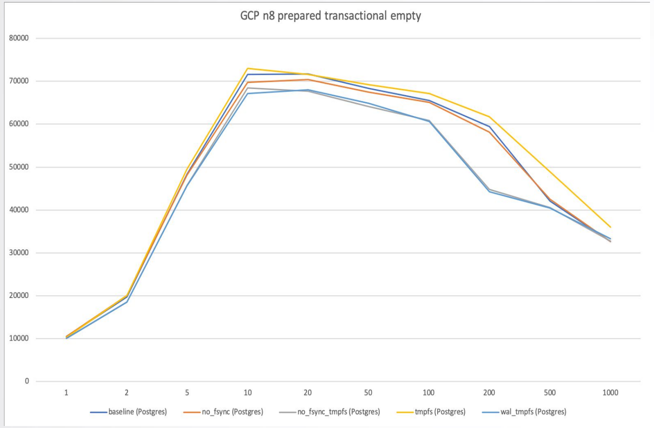
Let's test: Running on tmpfs / fsync=off

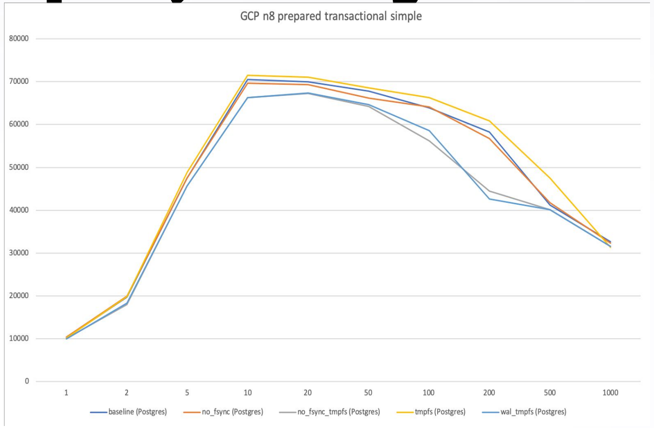
- **Run on GCP (n8)**
- **Run Prepared**
- **Run with 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000 threads**
- **Run with these storage options**
  - SSD
  - Datadirectory and WAL on tmpfs
  - WAL on tmpfs
  - fsync = off
  - DATA and WAL on tmpfs, fsync=off

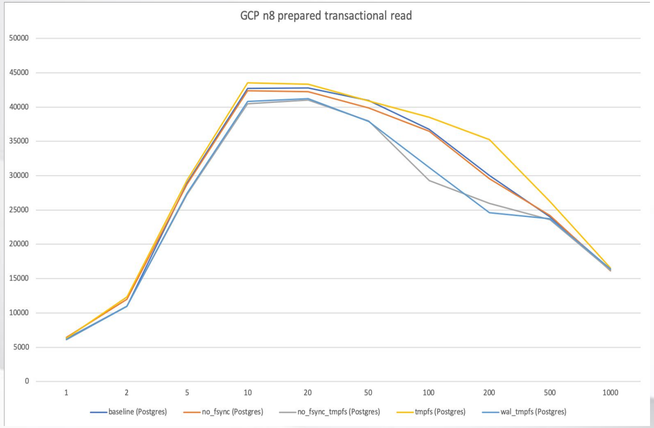
CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Running on SSD, tmpfs, nofsync



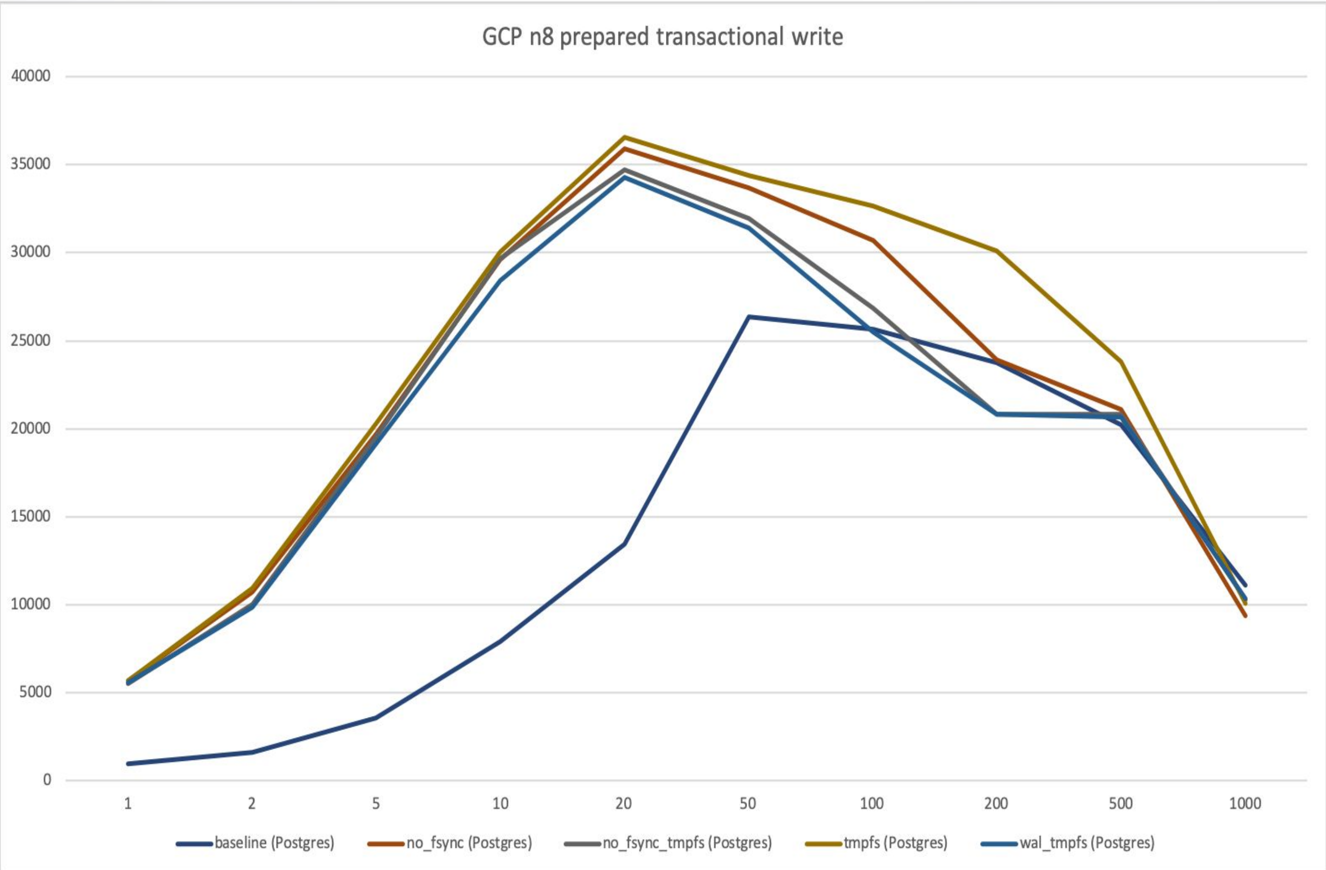




CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Running on SSD, tmpfs, nofsync (writes)



 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

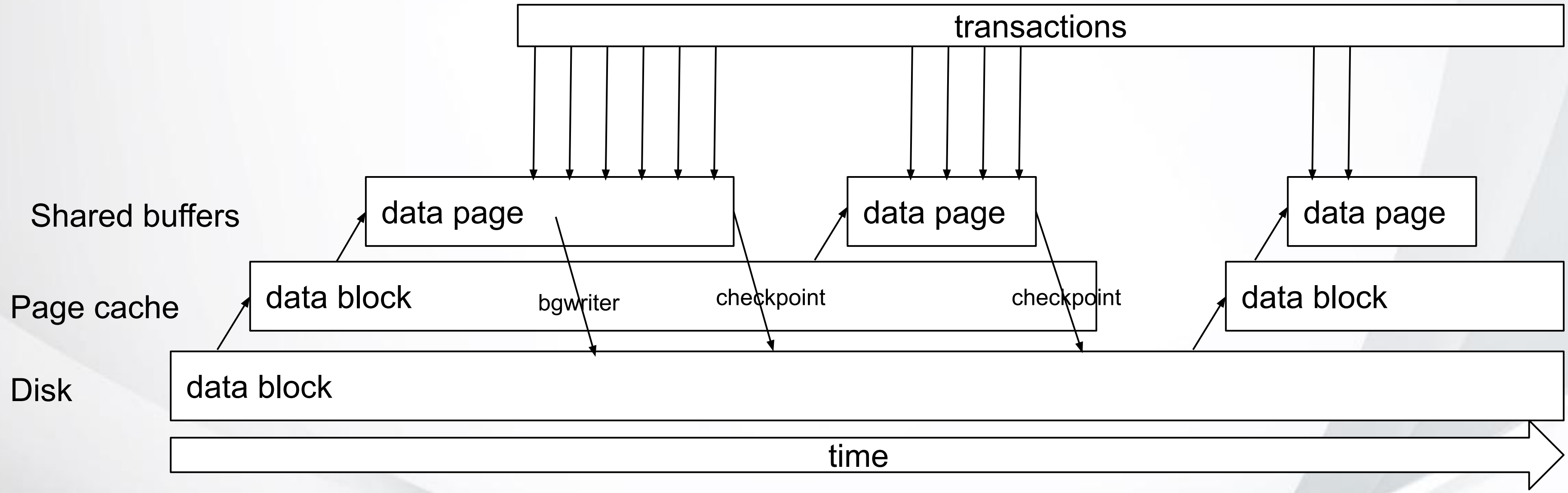
Let's test

Can we tune parameters?

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Can we tune to get more TPS?

How fast is your fsync flush

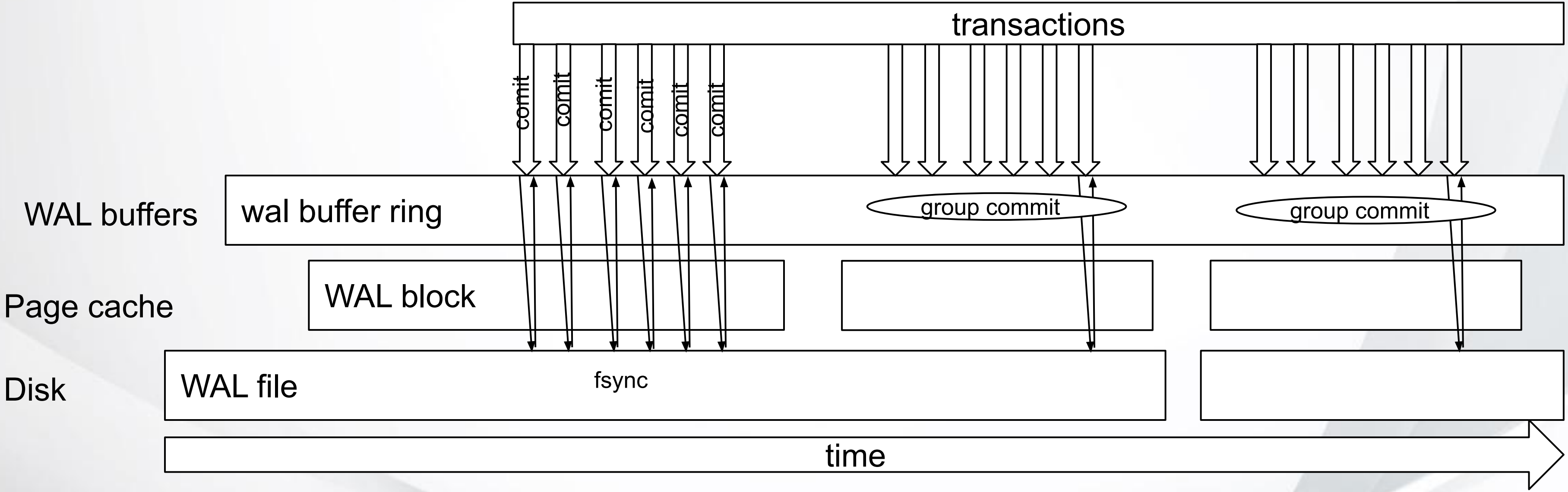


CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Can we tune to get more TPS?

How fast is your fsync flush FOR WAL



CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Introduction of group commit

CommitDelay

Vadim B. Mikheev <vadim4o@yahoo.com>:

- https://github.com/postgres/postgres/commit/b58c0411bad414a5dbde8b38f615867c68adf55c
- https://github.com/postgres/postgres/commit/a7fcadd10ab67a9cc938eb2818aae33d5be0238a
- https://github.com/postgres/postgres/commit/db2faa943a0d3517f9e9641b9012d81ecc870ff6
- https://github.com/postgres/postgres/commit/5b0740d3fcd55f6e545e8bd577fe8ccba2be4987
- https://github.com/postgres/postgres/commit/f0e37a85319e6c113ecd3303cddeb6edd5a6ac44
- https://github.com/postgres/postgres/commit/a70e74b060ab2769523ad831f571cb80122121d3
- https://github.com/postgres/postgres/commit/741510521caea7e1ca12b4db0701bbc2db346a5f

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Probably the bottleneck is storage

Group commit

Introduced in 7.1

```
commit 741510521caea7e1ca12b4db0701bbc2db346a5f
Author: Vadim B. Mikheev <vadim4o@yahoo.com>
Date:   Thu Nov 30 01:47:33 2000 +0000

    XLOG stuff for sequences.
    CommitDelay in guc.c

 src/backend/access/transam/rmgr.c |  15 ++------
 src/backend/access/transam/xact.c |   4 +--
 src/backend/access/transam/xlog.c |  40 +++++++++++++--------
 src/backend/commands/sequence.c   | 180
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++-------------
 src/backend/utils/misc/guc.c      |  13 ++++---
 src/include/access/htup.h         |   6 ++--
 src/include/access/rmgr.h         |   3 +-
 src/include/access/xlog.h         |   8 ++++-
 src/include/catalog/catversion.h  |   4 +--
 src/include/commands/sequence.h   |  35 ++++++++++++++++--
 10 files changed, 241 insertions(+), 67 deletions(-)
```


CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Probably the bottleneck is storage

Group commit

```
commit e620ee35b249b0af255ef788003d1c9edb815a35
Author: Simon Riggs <simon@2ndQuadrant.com>
Date:   Wed Dec 8 18:48:03 2010 +0000

    Optimize commit_siblings in two ways to improve group commit.
    First, avoid scanning the whole ProcArray once we know there
    are at least commit_siblings active; second, skip the check
    altogether if commit_siblings = 0.

    Greg Smith

 doc/src/sgml/config.sgml            | 17 ++++++++++++-----
 src/backend/access/transam/xact.c   |  2 +-
 src/backend/storage/ipc/procarray.c | 17 ++++++++++++-----
 src/backend/utils/misc/guc.c        |  2 +-
 src/include/storage/procarray.h     |  2 +-
 5 files changed, 27 insertions(+), 13 deletions(-)
```

Introduced in 9.1

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Probably the bottleneck is storage

Group commit

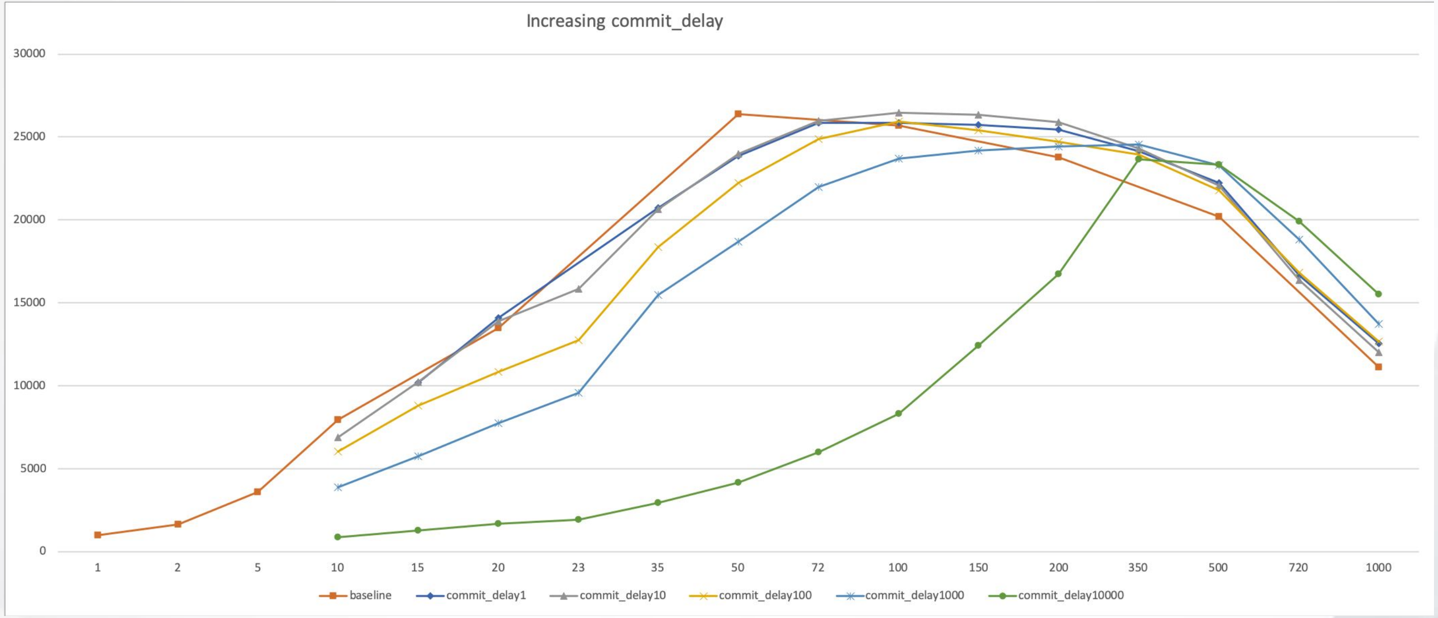
```
commit f11e8be3e812cdbbc139c1b4e49141378b118dee
Author: Robert Haas <rhaas@postgresql.org>
Date:   Mon Jul 2 10:26:31 2012 -0400

    Make commit_delay much smarter.

    Instead of letting every backend participating in a group commit wait
    independently, have the first one that becomes ready to flush WAL wait
    for the configured delay, and let all the others wait just long enough
    for that first process to complete its flush.  This greatly increases
    the chances of being able to configure a commit_delay setting that
    actually improves performance.

    As a side consequence of this change, commit_delay now affects all WAL
    flushes, rather than just commits.  There was some discussion on
    pgsql-hackers about whether to rename the GUC to, say, wal_flush_delay,
    but in the absence of consensus I am leaving it alone for now.

    Peter Geoghegan, with some changes, mostly to the documentation, by me.

 doc/src/sgml/config.sgml          | 35 +++++++++++++++++++-----------------
 doc/src/sgml/wal.sgml             |  4 +---
 src/backend/access/transam/xact.c | 19 -------------------
 src/backend/access/transam/xlog.c | 59 ++++++++++++++++++++++++++++++++++++++++---------------------
 4 files changed, 58 insertions(+), 59 deletions(-)
```

Introduced in 9.3

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Changing commit_delay



 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Let's test

Different type of queries

EDB POSTGRES

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

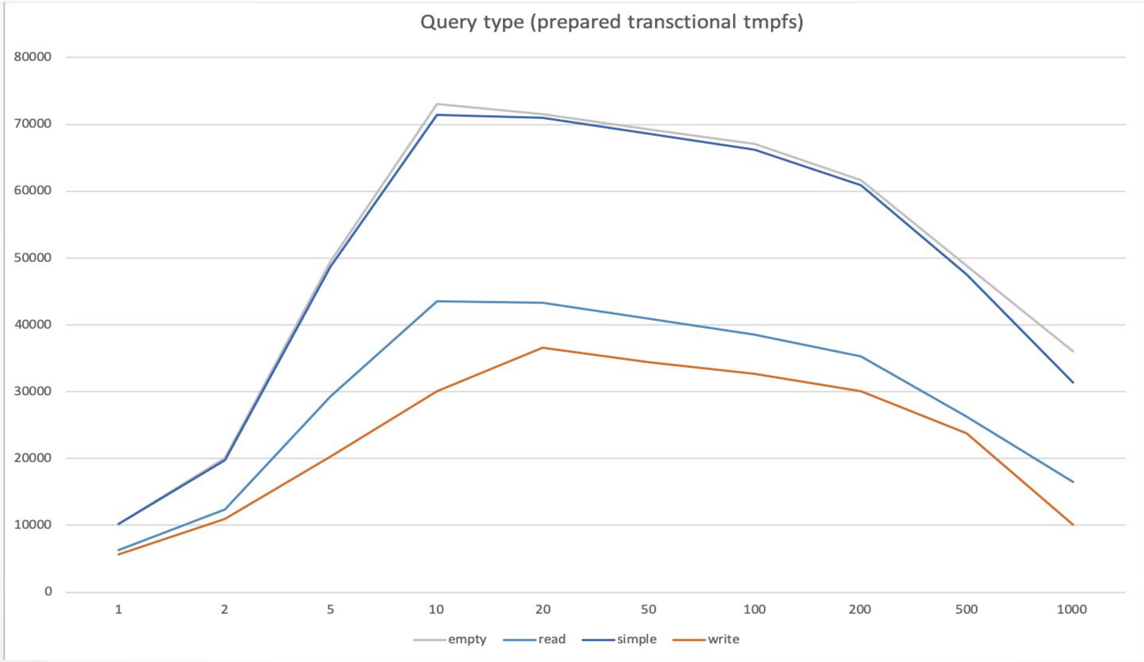
Let's test: Difference between query types

- Run on GCP (n8)
- Run on tmpfs
- Run Prepared
- Test with and without Transaction Control
- Run with 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000 threads

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

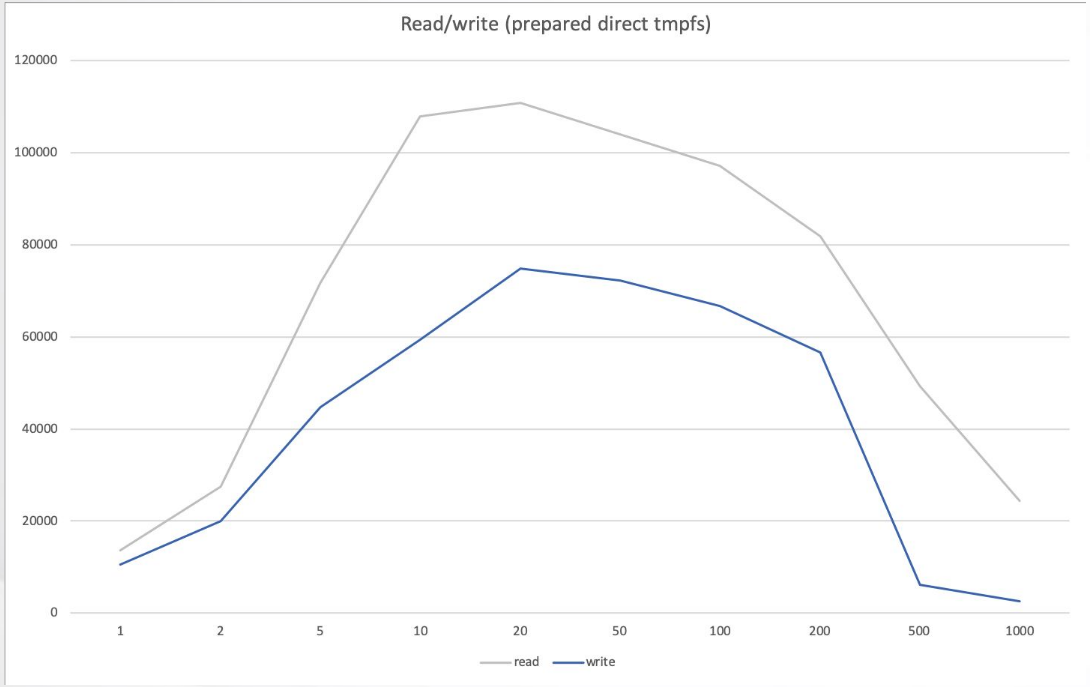
EDB POSTGRES

Query types (with Transaction Control)



CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

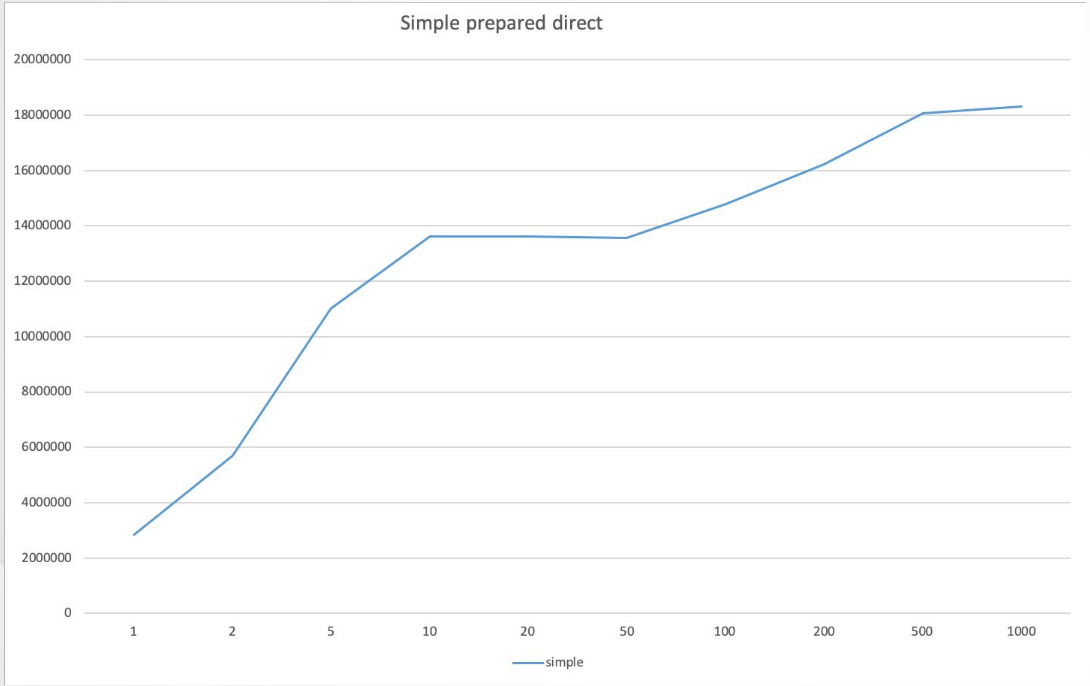
Query types (without Transaction Control, read/write)



Postgres reports same as threads
So, without Transaction Control, Postgres wraps reads and writes into a transaction.
But it is faster than using Transaction Control.

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Query types (without Transaction Control, simple)



- Could not read this from Postgres
- This is Queries per second

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Let's test

Impact of Programming language

EDB POSTGRES

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Efficient programming language

https://www.rust-lang.org/

Rust is blazingly fast and memory-efficient:
with no runtime or **garbage collector**, it can

- power performance-critical services,
- run on embedded devices, and
- easily integrate with other languages.

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Let's test: Impact of more efficient programming languages

Simple Query, no Transaction Control, Prepared:

GoLang

```
[root@6abd33baf7d3 /]# ~/test_with_go
2019/07/05 10:52:16 TPS: 33790.740819
2019/07/05 10:52:17 TPS: 33376.113974
2019/07/05 10:52:18 TPS: 35445.123329
2019/07/05 10:52:19 TPS: 34871.932790
2019/07/05 10:52:20 TPS: 35750.724915
2019/07/05 10:52:21 TPS: 34836.967868
2019/07/05 10:52:22 TPS: 34943.431673
2019/07/05 10:52:23 TPS: 37725.527713
2019/07/05 10:52:24 TPS: 34402.818432
2019/07/05 10:52:25 TPS: 34382.913243
```

About 34 thousand QPS

Rust

```
[root@6abd33baf7d3 /]# ~/test_with_rust
Initializing all threads
Connectstring: postgres://postgres@localhost:5432/postgres
Query: SELECT $1
SType: prepared
Date        time (sec)          | Sample period |          Threads            |            Postgres           |
                                |               | Average TPS | Total TPS     |        tps |            wal/s |
2019-07-05 10:52:32.757321         1.005000   352699.969   10580999.000           0.983              0.000
2019-07-05 10:52:33.774815         1.005000   346679.062   10400372.000           0.983              0.000
2019-07-05 10:52:34.788588         1.001000   368831.781   11064953.000           0.986              0.000
2019-07-05 10:52:35.805328         1.005000   309643.344    9289300.000           0.984              0.000
2019-07-05 10:52:36.826486         1.008000   332049.375    9961481.000           0.979              0.000
2019-07-05 10:52:37.852077         1.014000   341096.094   10232883.000           0.975              0.000
2019-07-05 10:52:38.874427         1.011000   335545.719   10066372.000           0.978              0.000
2019-07-05 10:52:39.903272         1.017000   332043.938    9961318.000           0.972              0.000
Let's give the threads some time to stop
Finished
[root@6abd33baf7d3 /]#
```

About 10 Million QPS (About 300 times more)

EDB POSTGRES

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Some final thoughts

Summarizing the results

EDB POSTGRES

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Some final thoughts

- **The 'happy zone'**
  - Reads: 5 - 100 parallel connections
  - Writes: 50 and 500 parallel connections
  - Higher than 200: System stability decreases

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Some final thoughts

- **Adding cores**
  - tps ↑
  - 'the happy zone' ↑

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

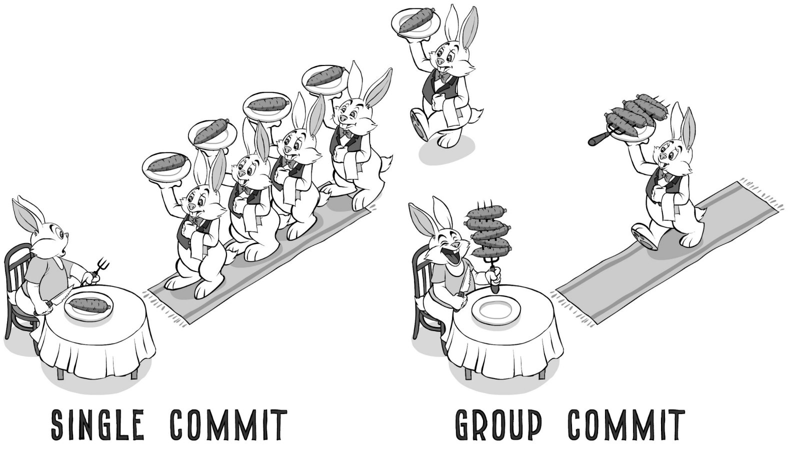
Some final thoughts

- **Compare Prepare vs Unprepared**
  - Preparing (150%) increases TPS by about 50% vs Unprepared (100%)

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Some final thoughts

- **Transaction Control**
  - Writing + Transaction Control = -50%
  - Storage ramp up: (8 core >30 threads)



 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

Some final thoughts

- **Compare running on disk vs tmpfs vs fsync=off**
  - Perf=True: tmpfs / wal on tmpfs / fsync=off
  - Running on disks (as we always do)
    - Adds a bottleneck, **but only for writes**
    - shifts the happy zone up
      - more connections helps writes on SSD more)
      - Happy zone 10 → 50

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Some final thoughts

- **Programming language**
  - Optimized for much little things
  - Rust: compiled, no garbage collection
  - Rust outperformed GoLang hugely in some cases

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

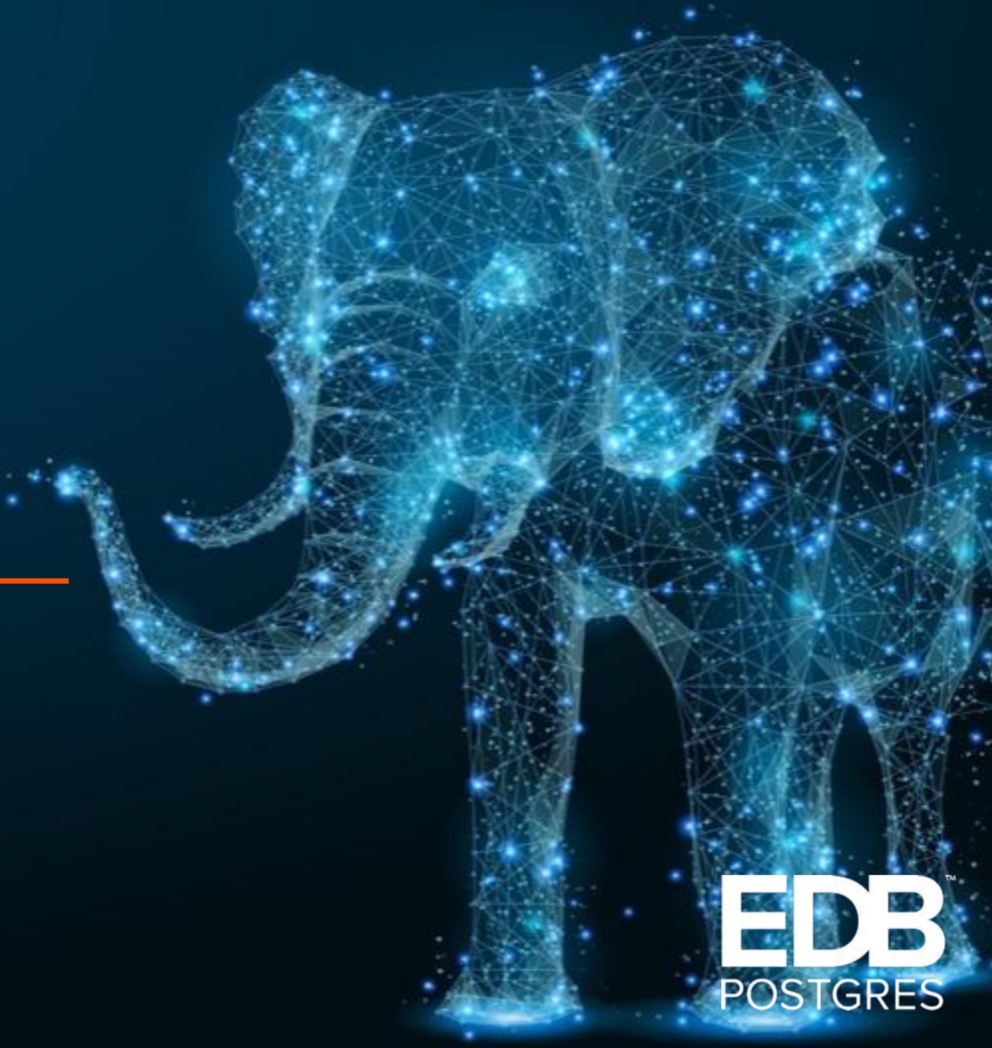
Answer (how many cpu, 60.000 TPS)

configuration	#cores	#threads
empty/simple	7	9
writes w/o transaction control	11	133
writes on tmpfs, no_fsync	13	38
read on SSD	14	11
Writes on SSD	**18**	**225**

CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

60000 TPS

How many CPUs ???



The results of an interesting research

Sebastiaan Mannem

EDB POSTGRES

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

QUESTIONS & DISCUSSION

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES

Solutions Architect

EnterpriseDB

Professional Services

sebas@mannem.nl

sebastiaan.mannem@enterprisedb.com

 CONFIDENTIAL © Copyright EnterpriseDB Corporation, 2019. All rights reserved.

EDB POSTGRES