

# PostgreSQL 9.6

PGConf,UK 2016  
London, UK

Magnus Hagander  
*magnus@hagander.net*

# Magnus Hagander

- Redpill Linpro
  - Infrastructure services
  - Principal database consultant
- PostgreSQL
  - Core Team member
  - Committer
  - PostgreSQL Europe

**PostgreSQL 9.6**

# Development schedule

- June 30, 2015 - branch 9.5
- July 2015 - CF1
- September 2015 - CF2
- November 2015 - CF3
- January 2016 - CF4
- March 2016 - CF5
- June 2016 - Beta2!

# Current status

## Beta 2

- Testing and fixes
- May still be removed
- *Please help!*

# New Features

- DBA and administration
- Developer and SQL features
- Replication and backup
- Performance

# idle in transaction timeout

- Simple: kill idle in transaction sessions

```
postgres=# set idle_in_transaction_session_timeout = 5000;  
SET  
postgres=# begin;  
BEGIN  
postgres=# FATAL: terminating connection due to idle-in-transacti
```

# pg\_stat\_activity

- Now has much better wait information
- Not just a boolean
- *waiting* column is now gone
  - Update your scripts!



# pg\_stat\_activity

```
postgres=# SELECT * FROM pg_stat_activity WHERE wait_event IS NOT
-[ RECORD 1 ]-----+-----
pid           | 4026
...
state_change  | 2016-04-14 14:33:10.621561+02
wait_event_type | Lock
wait_event    | transactionid
state         | active
...
query        | select * from a for update;
```

# pg\_blocking\_pids

- Returns array of pids that are blocking x
- Use on a process in waiting state
  - Shows who to blame

```
postgres=# select * from pg_blocking_pids(4026);
 pg_blocking_pids
-----
 {4021}
(1 row)
```

# Utility command progress

```
postgres=# SELECT * FROM pg_stat_progress_vacuum ;
-[ RECORD 1 ]-----+-----
pid          | 4021
datid        | 12407
datname      | postgres
relid        | 16402
phase        | scanning heap
heap_blks_total | 4425
heap_blks_scanned | 27
heap_blks_vacuumed | 0
index_vacuum_count | 0
max_dead_tuples | 291
num_dead_tuples | 0
```

# System information

- View: `pg_config`
  - Same info as binary *pg\_config*
- Functions: `pg_control_*`
  - Same info as `pg_controldata`

# Vacuum of frozen pages

- Track all-frozen pages
- Avoid VACUUM on all-frozen pages
  - Anti-wraparound autovac
  - Manual freeze
  - COPY FREEZE
- Much lighter on mostly-read tables

# postgres\_fdw

- Use remote extensions
  - Whitelist per server
  - Manually install on remote!
  - Use functions/operators locally

```
ALTER SERVER foo OPTIONS (extensions 'pgcrypto,tablefunc')
```

# New Features

- DBA and administration
- Developer and SQL features
- Replication and backup
- Performance

# Phrase searching

```
postgres=# SELECT plainto_tsquery('quick fox') @@  
           to_tsvector('the quick brown fox jumped');  
?column?  
-----  
t  
(1 row)
```

```
postgres=# SELECT phraseto_tsquery('quick fox') @@  
           to_tsvector('the quick brown fox jumped');  
?column?  
-----  
f  
(1 row)
```



# Phrase searching

```
postgres=# SELECT tsquery('quick <-> fox') @@  
           to_tsvector('the quick brown fox jumped');  
?column?  
-----  
f  
(1 row)
```

```
postgres=# SELECT tsquery('quick <2> fox') @@  
           to_tsvector('the quick brown fox jumped');  
?column?  
-----  
t  
(1 row)
```

# New Features

- DBA and administration
- Developer and SQL features
- Replication and backup
- Performance

# wal\_level=replica

- Same as old *hot\_standby*
- *archive* has been retired
  - If specified, maps to *replica*

# pg\_stat\_wal\_receiver

- On standbys only
- "Mirror" of *pg\_stat\_replication*
- Zero or one rows

# Replication slots

- *pg\_basebackup*
  - Can now create slot
  - Only used for replication
- *pg\_create\_physical\_replication\_slot*
  - Can now reserve WAL directly

# Multiple sync standbys

- Requires more than one server to ack commit
- Increase availability in case of multi-node failure

```
synchronous_standby_names = 'node1'
```

```
synchronous_standby_names = '3 (node1, node2, node3, node4)'
```

# `synchronous_commit = 'remote_apply'`

- Waits for full WAL apply on standby
- Slower than 'on'
  - But not necessarily much
- Guarantees data available for slave read
- Can be combined with multiple sync

# New Features

- DBA and administration
- Developer and SQL features
- Replication and backup
- Performance



# Faster time datatypes output

- timestamp, date and time
- Much faster output functions
- Copy up to 2x faster!
  - Single table, single column timestamp

# Locking changes

- Even more...
- For high concurrency loads
- Also better tracing

# Relation extension

- Used to extend by one block
  - Much blocking in write intensive loads
- Now extends multiple blocks at once
  - 20 \* number of waiters

# Checkpoint sorting

- I/O at checkpoints no longer random
  - Sorted by tablespace
  - Then relfilenode
  - Then fork
  - Then block
- Much more sequential writing

# Kernel writeback config

- Issues with large write caches
- OS would buffer writes "too long"
- And flush all at once
  - Causing I/O storms
- Could be configured on global level
  - `/proc/sys/vm/dirty_background_ratio` etc

# Kernel writeback config

- Now configurable in postgresql.conf
- Platform dependent
- Enabled by default on Linux only
  - for now
- Usually better to "flush early"
  - Exception workload:
    - Bigger than shared\_buffers
    - Smaller than OS cache

# Kernel writeback config

- `checkpoint_flush_after`
  - Default: 256Kb
- `bgwriter_flush_after`
  - Default: 512Kb
- `backend_flush_after`
  - Default: 128Kb

# postgres\_fdw

- Control fetch\_size
  - Per table or per server
  - (Used to be 100)



# postgres\_fdw

- Push down joins
  - Normal joins
  - Not anti/semi
- Push down ordering
  - Triggers remote ORDER BY
- Make direct updates and deletes
  - No SELECT FOR UPDATE

# Parallelism

# Parallelism

- CPU intensive workloads
- Previously, single query=single core
- But we have many cores now...

# Parallelism

- Many different parts
- Many still remaining
- But already very useful!

# Parallel seq scans

- Scan a single table using multiple workers
- Increase throughput
- Functions can be pushed down
  - Filtering functions
  - Target functions
  - If marked parallel safe
- Foundation for many others

# Parallel aggregates

- Aggregates often CPU-bound
- Partial aggregation in worker
- Final combination in parent
- Requires aggregate-specific support
  - Most built-in
  - Except string, json, xml, arrays
  - And not ordered-sets

# Parallel joins

- Based in parallel seq scan
- Each "partition" joined individually
  - In a separate worker
- Not all joins
  - >Only NestLoop and Hash
  - Other restrictions

# Controlling parallelism

- `max_worker_processes = n`
  - Global
- `max_parallel_degree = n`
  - Max per individual query
  - Limited by `max_worker_processes`



# Controlling parallelism

- `parallel_setup_cost = n`
- `parallel_tuple_cost = n`
- `force_parallel_mode = n`

# Controlling parallelism

- ALTER TABLE .. SET (parallel\_degree = n)
  - Default determines by relation size
- ALTER FUNCTION .. PARALLEL SAFE
- ALTER FUNCTION ... COST

**Ok, one last thing**

**Anybody used Oracle?**

ORA-01555: snapshot too old

**Yup, we have that now**

# Snapshot too old

- Configured by time
- Terminates old transactions
  - If *repeatable\_read* or higher
  - Prevents bloat buildup
- `old_snapshot_threshold = <minutes>`
  - Default is **off**

```
postgres=# SELECT * FROM c;  
ERROR:  snapshot too old
```

# There's always more

- Lots of smaller fixes
- Performance improvements
- etc, etc
- Can't mention them all!

# What's *your* biggest feature?

- Parallelism
- Vacuum freeze
- Snapshot Too Old
- Multiple sync standbys
- postgres\_fdw improvements
- Wait/lock monitoring
- Other?

# Thank you!

Magnus Hagander

magnus@hagander.net

@magnushagander

<http://www.hagander.net/talks/>

This material is licensed

