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
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

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

<table>
<thead>
<tr>
<th>pid</th>
<th>4026</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>state_change</td>
<td>2016-04-14 14:33:10.621561+02</td>
</tr>
<tr>
<td>wait_event_type</td>
<td>Lock</td>
</tr>
<tr>
<td>wait_event</td>
<td>transactionid</td>
</tr>
<tr>
<td>state</td>
<td>active</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>query</td>
<td>select * from a for update;</td>
</tr>
</tbody>
</table>
```
pg_blocking_pids

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

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

<table>
<thead>
<tr>
<th>pid</th>
<th>4021</th>
</tr>
</thead>
<tbody>
<tr>
<td>datid</td>
<td>12407</td>
</tr>
<tr>
<td>datname</td>
<td>postgres</td>
</tr>
<tr>
<td>relid</td>
<td>16402</td>
</tr>
<tr>
<td>phase</td>
<td>scanning_heap</td>
</tr>
<tr>
<td>heap_blks_total</td>
<td>4425</td>
</tr>
<tr>
<td>heap_blks_scanned</td>
<td>27</td>
</tr>
<tr>
<td>heap_blks_vacuumed</td>
<td>0</td>
</tr>
<tr>
<td>index_vacuum_count</td>
<td>0</td>
</tr>
<tr>
<td>max_dead_tuples</td>
<td>291</td>
</tr>
<tr>
<td>num_dead_tuples</td>
<td>0</td>
</tr>
</tbody>
</table>
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
postres_fdw

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

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

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

```sql
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

synonymous_standby_names = 'node1'

synonymous_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
  - $20 \times$ 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`

```sql
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/