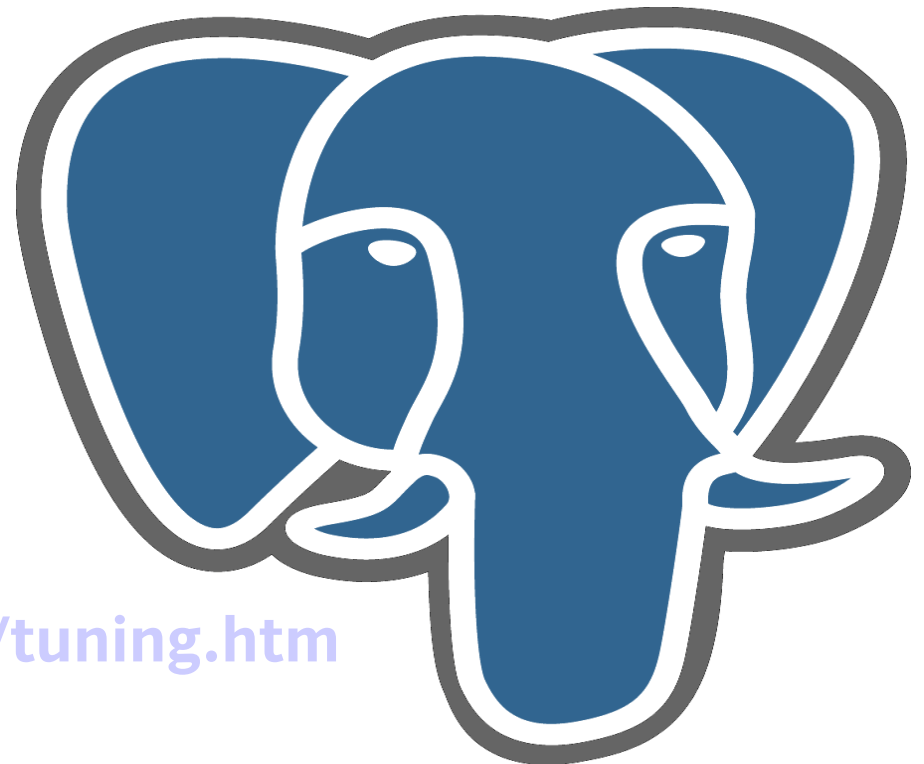
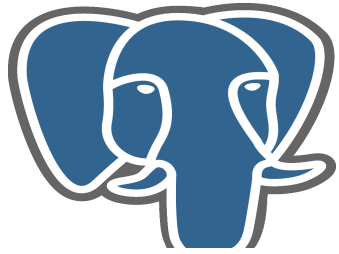


PostgreSQL 8.4 Performance Features

Simon Riggs
2nd Quadrant
simon@2ndQuadrant.com

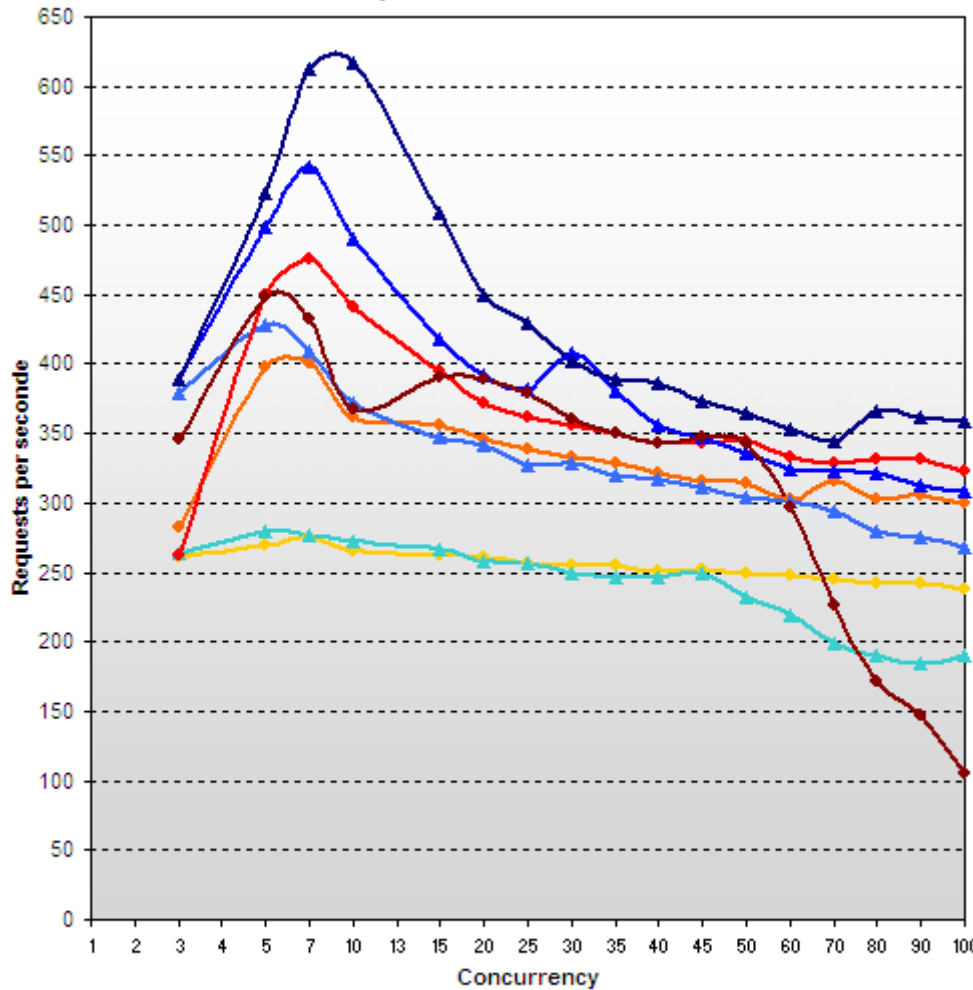
<http://www.2ndQuadrant.com/tuning.htm>





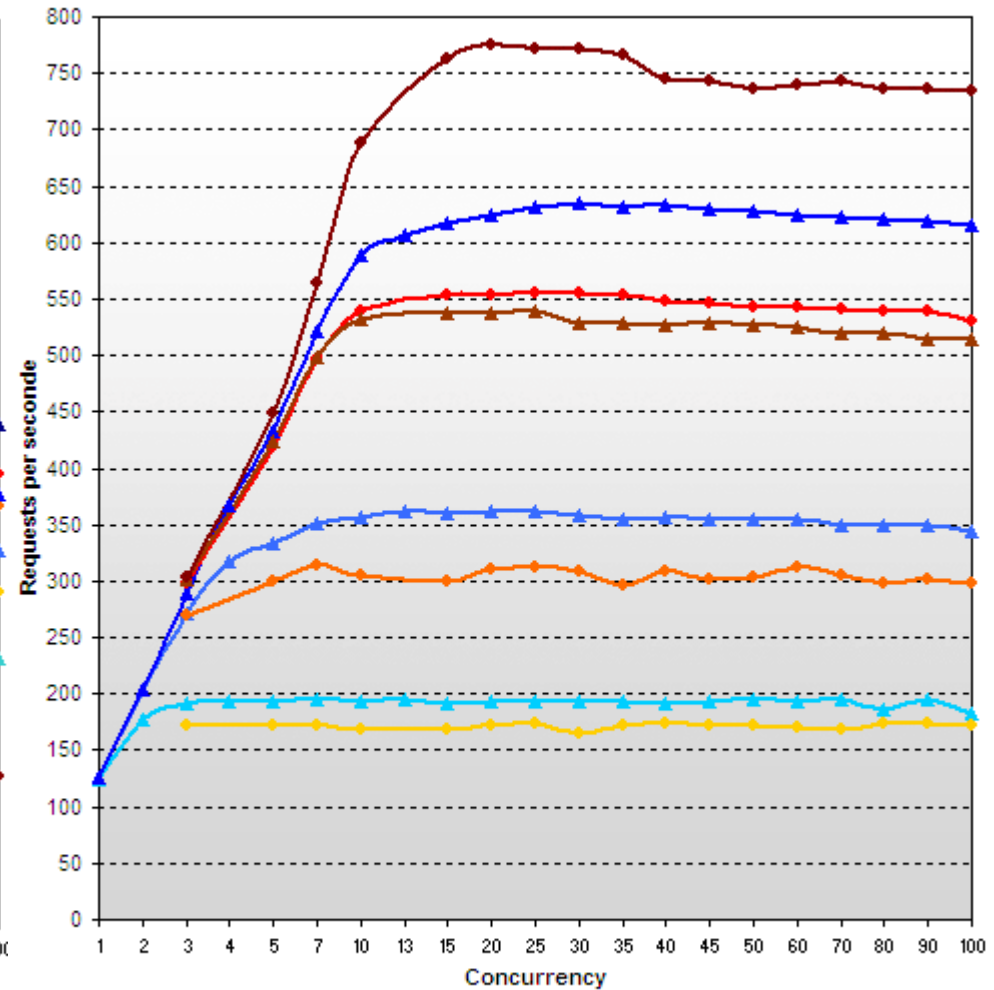
MySQL v PostgreSQL Scalability

Tweakers.net Database-simulatie
MySQL 5.0.20a vs. 5.0.32-bk

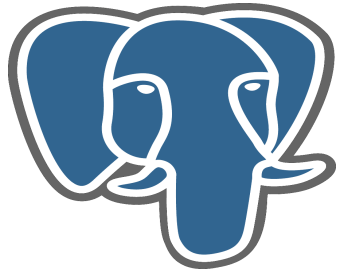


- 5.0.20a (single)
- 5.0.20a (1x dual)
- 5.0.20a (1x quad)
- 5.0.20a (2x quad)
- 5.0.32-bk (single)
- 5.0.32-bk (1x dual)
- 5.0.32-bk (1x quad)
- 5.0.32-bk (2x quad)

Tweakers.net Database-simulatie
Clovertown vs. Woodcrest - PostgreSQL 8.2-dev

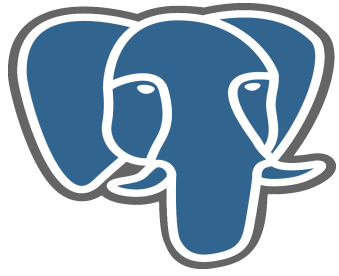


- Woodcrest 3,0GHz (single)
- Woodcrest 3,0GHz (1x dual)
- Woodcrest 3,0GHz (2x dual)
- Clovertown 2,66GHz (single)
- Clovertown 2,66GHz (1x dual)
- Clovertown 2,66GHz (2x dual)
- Clovertown 2,66GHz (1x quad)
- Clovertown 2,66GHz (2x quad)



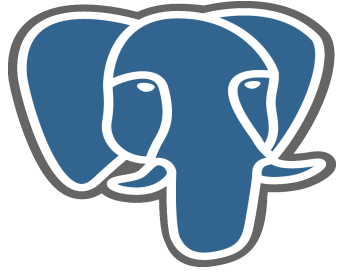
Performance Features

- Executor Enhancements
- Optimizer Enhancements
- General Server Enhancements
- Recovery/ Warm Standby Performance
- VACUUM & MVCC Performance
- Bulk Loading and Restore
- Monitoring
- Usability & Control
- Full Text Search



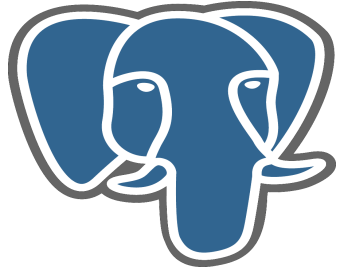
Executor Changes

- Hash Aggregation now used for **SELECT DISTINCT** and **UNION/INTERSECT/EXCEPT** when memory allows – improving speed over previous sort-based plans
- I/O read ahead for bitmap index scans improves speed of longer running queries with complex index access



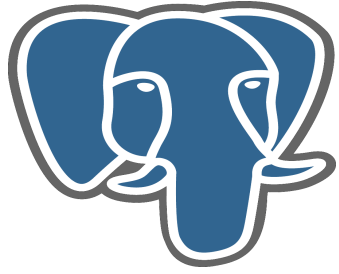
Hash Joins

- Multi-batch hash joins could occur when we underestimated the column stats for number of distinct items
- Multi-batch hashed plans now cope much better with skewed data distributions, common in most real world databases
- Multi-batch hash joins now minimise the amount of data sent to disk by projecting out unwanted columns beforehand



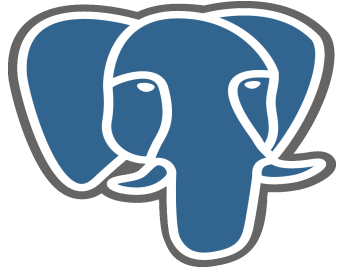
Optimizer Changes

- Subquery handling improved
- IN and EXISTS now comparable performance
- default_statistics_target = 100 (was 10)
 - note that sample sizes will be larger and ANALYZE will run for longer on many tables



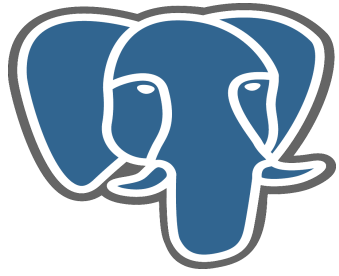
Server Changes

- `xxx_pattern_ops` now used for equality comparison, reducing number of indexes required when using a non-C locale
- hash indexes now much faster
 - but still unusable
- stats collection
 - stats file placed on separate drive for performance
 - rate limiting of access to stats data
- `float4`, `float8` and `int8` are now passed by value
 - reduce memory usage



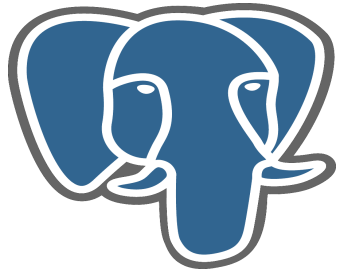
Out-of-line data storage (TOAST)

- Enhancements to TOAST facility in 8.4 allow
- Consider TOAST compression on values as short as 32 bytes (previously 256 bytes)
- Require 25% of space savings before using TOAST compression (previously 20%)
- Be more aggressive in storing **EXTERNAL** and **EXTENDED** column values in TOAST



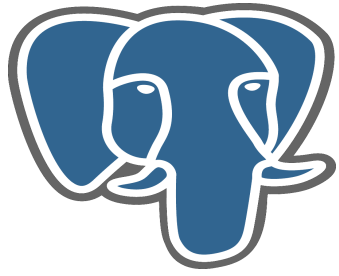
Recovery

- bgwriter now active during recovery
- Much improved performance of warm standby replication on I/O bound databases



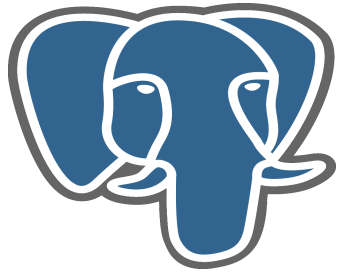
VACUUM & MVCC

- VACUUM performance improved
- Snapshot tracking allows earlier removal of rows during long extended transactions
- Can specify TOAST table setting to autovacuum



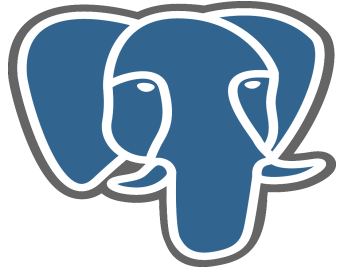
Utility Performance

- Parallel `pg_restore -j`
 - Many times faster
- COPY and Create Table as Select
 - 10-20% gain in many cases
- Memory reduction for AFTER triggers



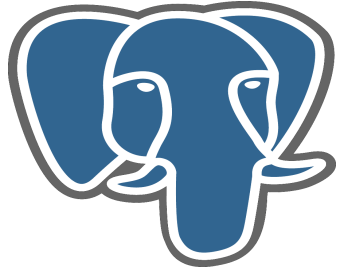
Monitoring

- track_functions
 - per function run-time statistics (except for SQL funcs)
- track_activity_query_size
- executor hook
- planner stats hook
- new Dtrace probes
- EXPLAIN VERBOSE



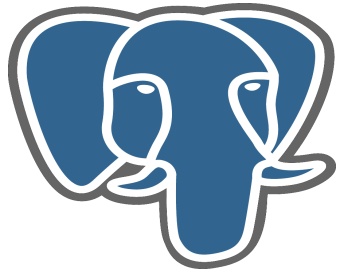
New add-on Modules

- Contrib
 - auto explain
 - pg_stat_statements
- PgFoundry
 - New prefix index module
 - pgloader significantly improved



User Control

- `cursor_tuple_fraction`
- `constraint_exclusion`
- New SQL features for `WITH RECURSIVE` and Window functions will also make more complex problems expressible as a single SQL statement and allowing better optimisation
- `suppress_redundant_updates_trigger()`
- `text_position()` improvements



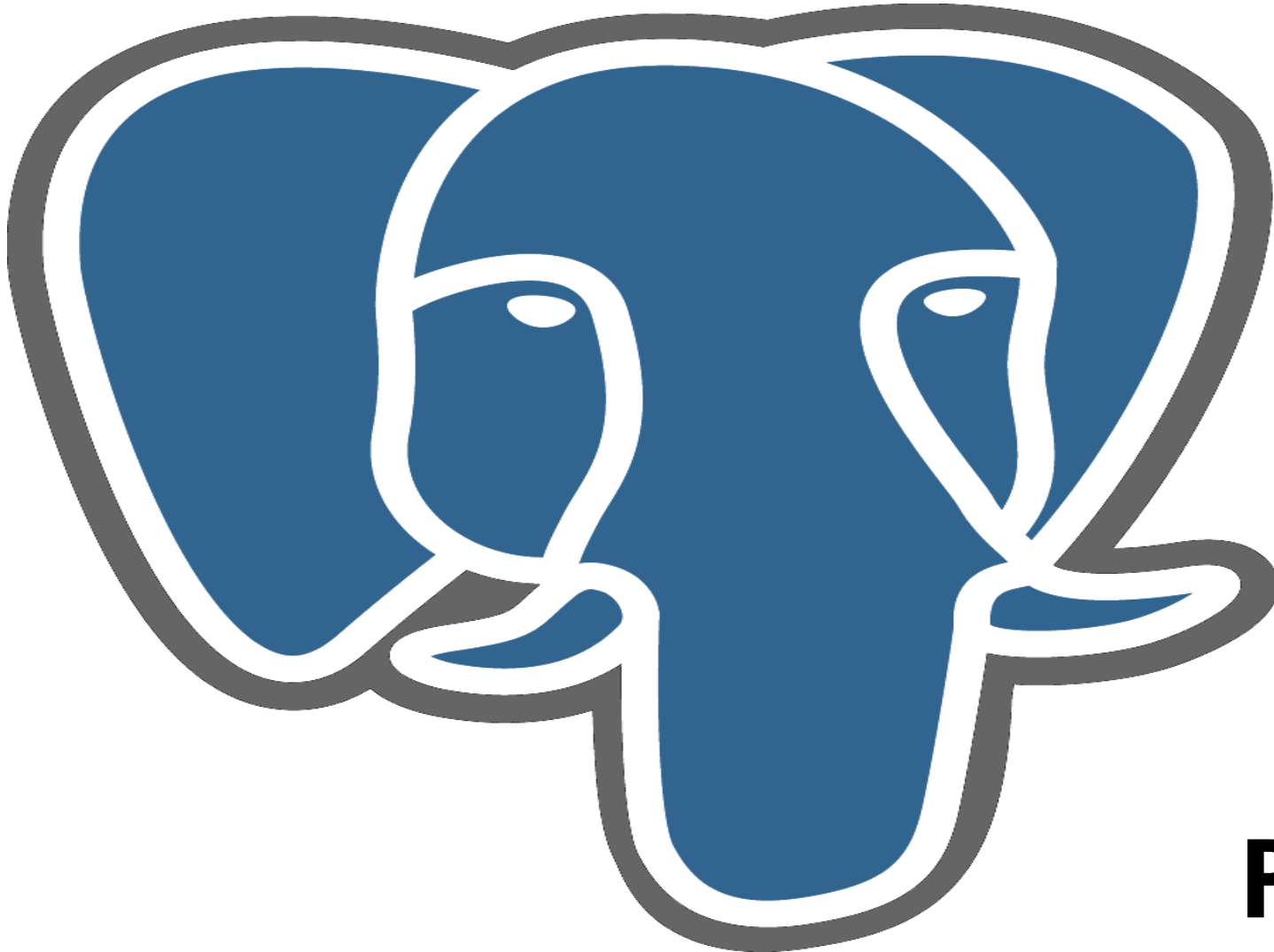
PostgreSQL 8.3

PostgreSQL 8.3 was

- Smaller
- Faster
- Smoother
- More scalable
- More tunable

PostgreSQL 8.4 gives

- Better plans
- Better use of memory
- Faster replication
- Faster restore
- Improved monitoring



PostgreSQL