PostgreSQL Wartungsstrategien

Jens Wilke

PGConf.DE

11. November 2011

Wartungsstrategien

- Warum Wartung?
- Autovacuum Tuning
- Repairtools

Warum Wartung?

- Statistiken
  pg_statistic
  ANALYZE
- MVCC (Multiversion Concurrency Control)
  Wiederverwendung von Speicherplatz
  gelöschter Zeilenversionen

Warum Wartung?

- MVCC (Multiversion Concurrency Control)

INSERT INTO foo VALUES ('foo'),('bar');

xmin	xmax	col1
1	0	foo
1	0	bar

DELETE FROM foo where col1 = 'bar';

xmin	xmax	col1
1	3	bar
1	0	foo

Warum Wartung?

- MVCC (Multiversion Concurrency Control)
  INSERT INTO foo VALUES ('foo'),('bar');

xmin	xmax	col1
1	0	foo
1	0	bar

DELETE FROM foo where col1 = 'bar';

xmin	xmax	col1
1	3	bar
1	0	foo

Warum Wartung?

- MVCC (Multiversion Concurrency Control)
  INSERT INTO foo VALUES ('foo'),('bar');

xmin	xmax	col1
1	0	foo
1	0	bar

  DELETE FROM foo where col1 = 'bar';

xmin	xmax	col1
1	3	bar
1	0	foo

Warum Wartung?

- VACUUM
  - Freespace Map
  - Transaktionsnummernüberlauf
- Strategien zur Reduzierung von Vacuum
  TRUNCATE/Partitionierung
- Für massiv fragmentierte Tabellen oder Indexe ist Vacuum i.d.R. nicht ausreichend
  - Table-Bloat
  - Index-Bloat
- Monitoring
  check_postgres.pl Bloat Check
  pgstattuple

Autovacuum Daemon

- Autovacuum/Autoanalyze
- regelmäßige/permanente Wartung
- tägliches manuelles Vacuum Analyze ist in vielen Fällen nicht ausreichend
- Autoanalyze Tuning ist bei kippenden Plänen notwendig
- Autovacuum Tuning ist erforderlich, wenn es nicht nachkommt oder wenn die Worker eine zu hohe Systemlast verursachen und meist nur bei größeren Tabellen sinnvoll.
- Problematik langlaufender Transaktionen
- Statistics collector und ANALYZE: pg_stat_user_tables und pg_class

Autovacuum Config Parameter

- postgresql.conf
  (defaults)
  - track_counts = on
  - autovacuum = on
  - maintenance_work_mem = 16MB
  - autovacuum_max_workers = 3
  - autovacuum_vacuum_cost_limit = 200
  - autovacuum_vacuum_cost_delay = 20ms
  - log_autovacuum_min_duration = -1
  - autovacuum_naptime = 1min

Autovacuum Parameter

- Storage Parameter (pg_class.reloptions) und/oder postgresql.conf
  - autovacuum_enabled = on (Storage Parameter)
  - default_statistics_target = 100
  - autovacuum_vacuum_scale_factor = 0.2
  - autovacuum_analyze_scale_factor = 0.1
  - autovacuum_vacuum_threshold = 50
  - autovacuum_analyze_threshold = 50
- Storage Parameter ändern
  ALTER TABLE ... SET
  (autovacuum_analyze_scale_factor=0.05);

Autovacuum Tuning

autovacuum

```
UPDATES + DELETES >=
reltuples * autovacuum_vacuum_scale_factor + autovacuum_vacuum_threshold
```

autoanalyze

```
INSERTS + UPDATES + DELETES >=
reltuples * autovacuum_analyze_scale_factor +
autovacuum_analyze_threshold
```

Autovacuum Tuning

Statistik

```
SELECT s.*, c.reloptions FROM pg_stat_user_tables s
JOIN pg_class c ON s.relid = c.oid ORDER BY n_live_tup DESC;
```

```
-[ RECORD 1 ]-----+-----------------------------------------------
 schemaname       | foo
 relname          | bar
 [...]            |
 n_tup_ins        | 4826182
 n_tup_upd        | 5510336
 n_tup_del        | 1962041
 n_tup_hot_upd    | 1679879
 n_live_tup       | 12566817
 n_dead_tup       | 48725
 last_vacuum      |
 last_autovacuum  | 2011-10-10 19:40:55.15632+02
 last_analyze     |
 last_autoanalyze | 2011-10-10 21:22:51.942734+02
 reloptions       | {autovacuum_vacuum_scale_factor=0.1,
                  | autovacuum_analyze_scale_factor=0.05}
```

Autovacuum Logging

```
2011-10-24 17:52:08 CEST LOG: automatic vacuum of table ''foo.public.bar'':
index scans: 1
pages: 0 removed, 27452 remain
tuples: 27126 removed, 460599 remain
system usage: CPU 2.37s/5.68u sec elapsed 504.47 sec
```

```
2011-10-25 02:52:20 CEST LOG: automatic vacuum of table ''foo.public.bar'':
index scans: 1
pages: 0 removed, 27452 remain
tuples: 942 removed, 455541 remain
system usage: CPU 1.16s/3.09u sec elapsed 373.97 sec
```

Repairtools für die Reduktion massiver Bloat

- lockende Tools und/oder mit Downtime
  - dump und restore
  - CLUSTER vs VACUUM FULL 8.4/9.0
  - ALTER TABLE foo ALTER int_col SET DATA TYPE integer;
  - REINDEX
- nicht oder marginal lockende Tools ohne Downtime
  - CREATE INDEX CONCURRENTLY
  - compact_table (ctid based)
  - pg_reorg/Trigger based Replication