Wie PostgreSQL Ihre Daten Sicher Hält

Michael Banck <michael.banck@credativ.de>

PGConf.DE, 2019-05-10

Wie PostgreSQL Ihre Daten Sicher Hält

credativ

- Security
- Fehler-Toleranz und Integrität von Transaktionen
- Daten Prüfsummen

Wie Sie PostgreSQL's Daten sicher halten:

- Konsistenzprüfungen
- Backup Validierung

Wie PostgreSQL Ihre Daten Nicht Sicher Hält

credativ

- Locale-basierte Sortierungs-Probleme bei unkoordinierten Updates der C-Bibliothek
  - Sortierungs-Reihenfolge der C-Bibliothek kann sich ändern
  - Indexe finden Daten nicht mehr bzw. lassen Duplikate zu
  - `REINDEX` für Indexe mit Text-basierten Spalten (`text`, `varchar`, `char`) nötig
  - ICU-Bibliothek momentan keine generelle Lösung
  - `https://wiki.postgresql.org/wiki/Locale_data_changes`
- Datenverlust via `fsync()` bei überprovisioniertem Storage
  - Linux entfernt bei z.B. aus Platzmangel fehlerhaftem `fsync()` Daten aus dem Cache
  - Erneutes `fsync()` ist erfolgreich, aber gewünschte Daten wurden nicht geschrieben
  - In neuesten Point-Releases erfolgt Crash-Recovery bei `fsync()`-Fehlern
  - `https://fosdem.org/2019/schedule/event/postgresql_fsync/`
- Bis Version 9.6 hieß das Transaktionslog Verzeichnis `pg_xlog`
  - Wurde manchmal als unessentielles Log-Verzeichnis wahrgenommen und bei Speichermangel gelöscht

Security

- Default-mäßig werden nur Verbindungen von Localhost oder angemeldeten Benutzern akzeptiert
- Host-Based-Access-Controls (HBA) erlauben feingranulare Freigabe von Verbindungs-Rechten

Aber:

- Transparente Verschlüsselung problematisch, insbesondere Key-Handling
  - `pgcrypto`-Erweiterung erlaubt Verschlüsselung einzelner Spalten
  - Externe Projekte erlauben vollständige Verschlüsselung

Fehler-Toleranz

- *"I manage thousands of databases (PostgreSQL, SQL Server, and MySQL), and this past weekend we had a massive power surge that knocked out two APC cabinets. [...] Long story short,* **every single PostgreSQL machine survived the failure with zero data corruption**. *I had a few issues with SQL Server machines, and virtually every MySQL machine has required data cleanup and table scans and tweaks to get it back to "production" status."*

Joshua Drake,
`http://archives.postgresql.org/pgsql-advocacy/2011-04/msg00085.php`

Fehler-Toleranz

- *''I had exactly the same experience 3 years ago. Complete power failure (the stand-by generator took fire) in one small datacenter (around 500 machines). We had Oracle, SQL Server, DB2, MySQL, Progress, and of course PostgreSQL.* **The only database engine that restarted with no operation required was PostgreSQL.** *There were very minimal problems with Oracle (typing recover on some instances), but we had quite a few problems with the other engines.''*

Marc Cousin,
`http://archives.postgresql.org/pgsql-advocacy/2011-04/msg00086.php`

Integrität von Transaktionen

credativ

- Absturz eines Backends führt zur Beendigung aller aktuellen Verbindungen
  - Verhinderung von Datenverlust durch Speicher-Korrumpierung in anderen Sessions
- Transaktionen werden zunächst sicher im Transaktionslog gespeichert
  - Bei Absturz erfolgt Crash Recovery
- Replikation erlaubt Versand der Transaktionslog an sicheren zweiten Standort
  - (Per-Commit) Synchrone Replikation erlaubt sichere Speicherung auf mehreren Servern
  - Verschiedene Stufen von Persistenz-Garantien möglich

Integrität von Transaktionen - Transaktionslog

credativ

- Transaktionslog (Write-Ahead Log, WAL) wird als WAL-Buffer im Speicher und als WAL-Dateien auf Platte geführt
- Bei jedem Commit wird das WAL gesynct bevor der Client Erfolgsmeldung erhält
- WAL-Segmente sind 16 MB groß, prä-alloziert und recycelt
- WAL-Einträge sind mit Prüfsummen (`crc32`) versehen
- `walwriter`-Prozess schreibt periodisch die WAL-Buffer auf Platte
- Geänderte Heap-Daten werden durch periodische Checkpoints persistent ins Datenverzeichnis geschrieben

Integrität von Transaktionen - Crash Recovery

credativ

- Der Postmaster-Prozess überprüft beim Hochfahren ob Postgres zuletzt geordnet beendet wurde
- Im Fehlerfall wird zunächst Recovery-Modus aktiviert
- WAL-Dateien seit dem letzten Checkpoint (Redo-Punkt) werden nachgespielt
- Danach entweder weiterhin im Recovery-Modus (`standby_mode`) oder Wechsel in Produktivbetrieb

Daten Checksummen

- In allen unterstützten Postgres-Versionen verfügbar
- Prüfsummen im Header von Tabellen- und Index-Daten Blöcken
- Checksummen werden beim Schreiben/Lesen vom Storage gesetzt/überprüft
  - Bei nicht erfolgreicher Checksummen-Validierung erfolgt ein Fehler
  - Option `ignore_checksum_failure` ermöglicht Degradierung zu Warnung
- Impliziert `wal_log_hints`, erhöhtes Transaktionslog-Aufkommen
- Ab Version 12 Statistiken über Checksummen-Fehler in `pg_stat_database`

Daten Checksummen - Aktivierung

- Nicht der Standard, muss bei Instanz-Initialisierung eingeschaltet werden
  - `initdb -k`
- Ab Version 12 Offline Aktivierung via `pg_checksums` möglich
  - Jeder Block muss neu geschrieben werden
  - Keine Absicherung bei versehentlichem Instanz-Start
  - Replikations-Setups nicht-trivial
- Offline-Deaktivierung schnell (Ändern von `pg_control`)

Daten Checksummen - Überprüfung

- Implizite Überprüfung bei Verwendung von Tabellen/Index-Daten
- `pg_dump` überprüft Tabellen-Daten
- `pg_basebackup` überprüft seit Version 11 Checksummen
  - Option `--no-verify-checksums` verhindert dies
  - Checksummen-Fehler werden als Warnungen geloggt
- `pg_verify_checksums` überprüft seit Version 11 Checksummen
  - Instanz muss heruntergefahren sein
  - Base Backups müssen zunächst hoch- und wieder heruntergefahren werden
- pgBackRest überprüft Checksummen beim Backup und Restore
- Externe Tools:
  - `https://github.com/google/pg_page_verification`
  - `https://github.com/uptimejp/postgres-toolkit/`
  - `https://github.com/michaelpq/pg_plugins/tree/master/pg_checksums`

Daten Checksummen - pg_checksums

https://github.com/credativ/pg_checksums

- Fork/Merge von pg_verify_checksums/pg_checksums
- In Debian 10 (buster) bzw. apt.postgresql.org enthalten
- Online Überprüfung von Checksummen
- Offline (De-)Aktivierung von Checksummen
- Mit allen unterstützten Postgres-Versionen kompatibel
- Fortschrittsanzeige und I/O-Drosselung

Daten Checksummen - Zukünftige Arbeiten

credativ

- Online (De-)Aktivierung von Checksummen
  - Wurde bereits für Version 11 committet, wegen Problemen wieder entfernt
  - Globale Synchronisierung der schreibenden Backends problematisch
- Online Überprüfung von Checksummen
  - Kooperation mit dem Server nötig für In-Core Implementation
- Checksummen für Commit-Log (`pg_xact`), `pg_subtrans`, `pg_multixact` etc.
  - `https://commitfest.postgresql.org/19/1682/`
- Daten-Checksummen als Default (optional abschaltbar)

Daten Konsistenz

- Kein generelles Tool verfügbar, welches (Daten-)Konsistenz prüft
- `pg_dump` für Tabellen-Daten
- `amcheck` für Index-Daten
- Entwicklungsbedarf vorhanden

Daten Konsistenz - pg_dump

credativ

- Für die Konsistenz von Tabellen-Daten kann pg_dump nach /dev/null verwendet werden
- Paralleler Dump nach /dev/null nicht möglich
  - https://www.credativ.de/blog/postgresql-11-checksummen-und-basebackups
  - https://share.credativ.com/~mba/patches/parallel-pgdump-dev-null/
- Relativ schwergewichtig
- Konsistenz-Prüfung Page Header und erfolgreiche Dekodierung der Daten
- Überprüft keine Indexe, insbesondere auch nicht falsche doppelte Datensätze

Daten Konsistenz - amcheck

- Untersucht Index auf Konsistenz-Fehler, auch im Hinblick auf Tabellen-Daten
- Bisher nur BTree-Indexe unterstützt, Support für GIST-Indexe unter Entwicklung
- Kein CLI-Programm verfügbar, nur SQL-Interface

```
SELECT bt_index_check(index => c.oid, heapallindexed => i.indisunique) FROM pg_index i
JOIN pg_opclass op ON i.indclass[0] = op.oid JOIN pg_am am ON op.opcmethod = am.oid
JOIN pg_class c ON i.indexrelid = c.oid JOIN pg_namespace n ON c.relnamespace = n.oid
WHERE am.amname = 'btree' AND n.nspname != 'pg_catalog' AND c.relkind = 'i'
AND c.relpersistence != 't' AND c.relkind = 'i' AND i.indisready AND i.indisvalid;
```

- Enthalten in PostgreSQL "contrib" seit Version 10
- amcheck_next: https://github.com/petergeoghegan/amcheck
  - Support für Versionen 9.4-9.6
  - Potenziell neuere Features

Validierung von Backups

- Backups ohne Validierung sind deutlich weniger Wert
- Im Notfall unklar ob Backup korrupt, Prozess falsch oder sonstiges Problem
- Backups von Produktions-Instanzen sollten mindestens einmal die Woche automatisch validiert werden

Validierung von Backups - Logische Backups

credativ

- pg_dump basiert
- Validiert bei erfolgreichem Einspielen (pg_restore)
  - Einspielen in eine separate Restore-Test-Instanz üblich
  - Restore-Test-Instanz sollte gleiche Version von PostgreSQL/Betriebssystem haben
- Indexe werden nicht gedumpt, diese können in Produktion weiterhin korrupt sein
- Bei großen Instanzen (ca. $>$ 0,5 TB) Dumps unhandlicher und seltener

Validierung von Backups - Physikalische Backups

- `pg_basebackup`, pgBackRest etc. basiert
- Base Backups benötigen entsprechende Transaktionslogs (XLOGs)
- Einspielen des Base Backups und Hochfahren mit `recovery.conf` für XLOGs nicht ausreichend
  - Daten-Korruption wird nicht getestet
  - Evtl. Probleme beim Handling treten potenziell erst später auf
- Logischer Dump der Datenbanken nach `/dev/null`
- Im Idealfall erneutes Einspielen des Dumps in weitere Instanz
- Zusätzlicher wöchentlicher logischer Dump (evtl. nach `/dev/null`) ratsam

Danke für die Aufmerksamkeit - Kontakt

credativ

- Fragen?
- Michael Banck <michael.banck@credativ.de>
- http://www.credativ.de
- http://www.credativ.de/postgresql-competence-center
- http://www.credativ.de/jobs
- http://www.credativ.de/blog
- http://github.com/credativ