PostgreSQL Hardware und RAM-Tuning

Dipl.-Inf. Susanne Ebrecht

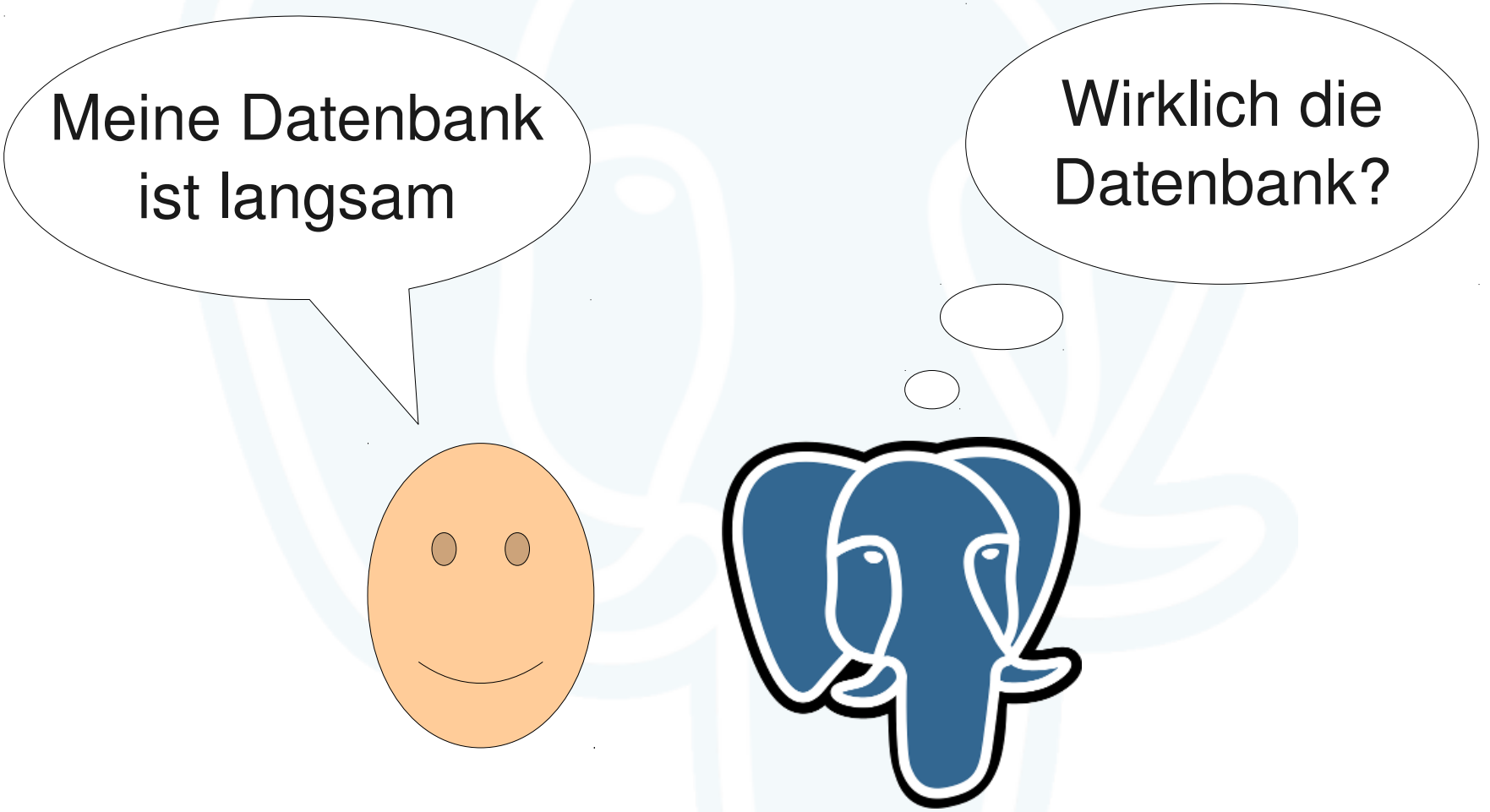
FrOSCon 2011

2ndQuadrant
Professional PostgreSQL

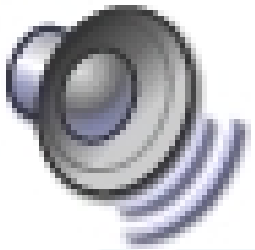
Hinweis

Zweck der Folien ist, den Vortrag zu untermalen. Die Folien allein sind nicht aussagekräftig. Die Folien unterliegendem dem Copyright der 2ndQuadrant Ltd. und dürfen nur mit schriftlicher Genehmigung der Autorin verteilt werden.

Wer ist schuld?



SQL-Optimierung






TUNE YOUR SQL!!!

... TUNE YOUR SQL!!

... TUNE ...

... YOUR ...

... SQL!

Hardware-Anschaffung

Gesetz der Hardwareanschaffung

$1 Jeder ist sich selbst am nächsten!

- **Traue Niemandem!!**

$2 Traue keiner Statistik, die Du nicht selbst gefälscht hast!

- **Eigene Benchmarks sind zwingend erforderlich!**

$3 Achte darauf, dass Hardware, die die Erwartungen nicht erfüllt, zurückgegeben werden kann!

- **Erst ausgiebig testen, dann kaufen!**
- **Einkäufern auf die Finger hauen!**

CPU

- PostgreSQL nutzt nur eine CPU je Query
- Passt die komplette Query-Ausführung ins RAM, ist die CPU der Engpass
- CPU und RAM sollten schnell sein
- Schnelle Datenübermittlung zwischen Prozessor und RAM

Mehr oder schnellere Cores?

- Beobachtung mit Tools wie top
- wenig Prozesse nutzen je eine CPU
  - schnellere Cores
- Viele zeitgleiche Prozesse auf den CPUs
  - mehr Cores

RAM

- Abhängig von der Menge der Datensätze, die in den häufig anfallenden Operation zu bearbeiten sind.
- Viel hilft nicht immer viel
- Passt alles in den RAM
  - schnellere Prozessoren
- Zu groß für realisierbares RAM
  - schnellere Festplatten

Konventionelle Festplatten

- U/min sind teuer
- Plattenmechanik als Faktor für Random I/O
- Unterstützt das Betriebssystem Command-Queuing?
- Mehrplattensysteme (wie RAID) höherer Nettodurchsatz als Einzelplatten
- Physikalische Hardware als Begrenzung
- Benchmark early - Benchmark often!
  - sowohl Hardware wie DB-Anwendung

SSD

- Urteil ist anwendungsabhängig
- **Niemals ohne Batterie !!!**
- Benchmarken! Benchmarken! Benchmarken!
- Preis / Leistung prüfen

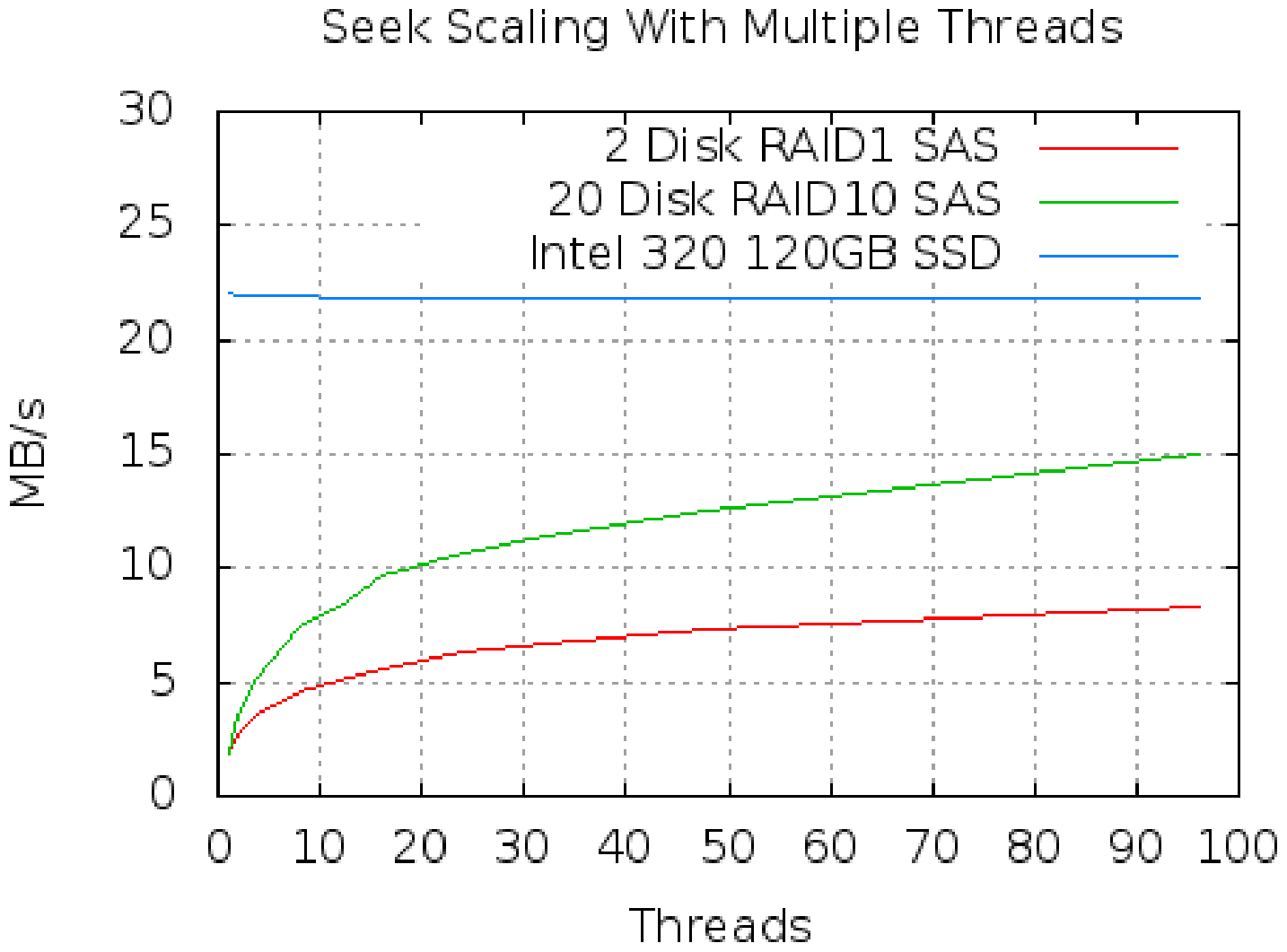
Festplattenfaktoren

- Sequenzielle Lesegeschwindigkeit
- Sequenzielle Schreibgeschwindigkeit
- Radom-I/O-Geschwindigkeit
- Commit-Rate

Testtools

- dd
- Bonnie++
- sysbench
- und weitere

Plattenvergleich



2ndQuadrant
Professional PostgreSQL

Wichtig ist ...

- Der Platten-Controller ist ausschlaggebend
- batteriegestützter write-cache
- Batterien sind lebenswichtig
- Read-Ahead rauf (z.B. auf 4096 Blöcke)
- dirty_ratio und dirty_background_ratio runter (auf 10% und 5% RAM oder weniger) bzw. dirty_bytes und dirty_background_bytes verwenden

Verbindungen

- max_connections
- superuser_reserved_connections in max_connections enthalten
- Verbindungen pro Core > 2 oder 3 => CPU veranstaltet Swap-Wettrennen
- PgBouncer und pgpool-II

Pufferspeicher

- Betriebssystem
- shared_buffers
- Linuxkernel anpassen
- shared_buffers mind. 256 MB
- shared_buffer Daumenwert: 1/4 RAM
- Max: 512 MB Windows, 8 GB Linux/Unix

Cache-Budget

- Mit wieviel Cache vom Betriebssystem kann PostgreSQL rechnen?
- Passen größere Indexe in den RAM / Cache?
- effective_cache_size
- ≅ RAM - schared_buffers
- rein informativ

Aufräumen

- dirty (pages)
- Checkpoints
- checkpoint_timeout (5 Minuten)
- checkpoint_segments
- Anzahl geschriebener WAL-Dateien (16 MB)
- Gute Auslastung => 3 zu häufig
- checkpoint_segments zwischen 32 und 128

Write Ahead Log

- Transaktionsprotokoll
- 64 kB als Puffer
- wal_buffers
- Massive Schreibanwendung => Cache schnell zu klein
  - Zwischen 1 und 16 MB

Wartungsarbeiten

- maintainance_work_mem
- Wartungsarbeiten
- Index-Erstellung
- Vacuum
- ≅ RAM / 16

Speicher am Arbeitsplatz

- work_mem
- Ausführung von Queries
- pro Ausführungsschritt
- RAM / (max_connections * 16)
- bis viermal mehr bei einfachen Queries
- üblich 4-128 MB
- änderbar pro Sitzung im laufenden Betrieb

Speicher für Vergängliches

- temp_buffers
- temporäre Tabellen
- nur wirksam, wenn vor der ersten Nutzung der temporären Tabelle gesetzt
- 8 MB voreingestellt
- änderbar pro Sitzung im laufenden Betrieb

Dauerhaft abgelegt?

- COMMIT-Meldung an den Client, erst nach Speicherung von WAL auf Platte
- sychronous_commit
- Tx-Daten im Cache sind bei Absturz verloren
- änderbar pro Sitzung im laufenden Betrieb

Plattenkopf-Wettspringen

- Random langsamer als sequenziell
- random_page_cost (4.0)
- Nicht auf tatsächlichen Wert sondern eher kleiner setzen
- gängige Random Daten (häufig genutzte Indexe) liegen meist eh im RAM
- üblich bei schnellen Systemen und Systemen mit viel RAM ist 2.0
- Passt DB in RAM => sehr klein (1.01)

Danke

Vielen Dank für die Aufmerksamkeit.

Fragen?

