PostgreSQL repliziert: Ein Überblick

Michael Renner

Netways OSDC
29. & 30. April 2009

Vorstellung

- Michael Renner
- Sysadmin, DBA, ScrumMaster
- Web-Affin
- Open Source
- Hang zur Perfektion

Eine Gebrauchsanleitung

- Fragen, ja bitte
- Zu Schnell/langsam → do tell!
- Workshop! Oder so.
- Slides gibt’s nach dem Workshop

Die nächsten zwei Stunden

- Eine Einleitung
- Replikation ist böse... ...aber notwendig
- (Datenbank-)Theorie
- Replikations-Ansätze
- Pause?
- Praxis
- Zukunft
- Best Common Practices

Oder auch ganz anders!

- Workshop!
  - Wer hat Oracle im Einsatz?
    - Postgres?
    - MySQL?
    - Was andres?
  - Wer repliziert schon Daten?
    - Was/Wie?
    - Gibt es Probleme?
  - Hat jemand vor Replikation einzuziehen?
- Weiter mit dem Rahmenprogramm?
  - Oder lieber was anderes?

Eine Begriffserklärung

- Replikation
  - „Datenverteilung“
- Was darfs denn sein?
  - „Cluster“?
    - Automatische Segmentierung & Verteilung?
      - vulgo: Sharding, Horizontale Skalierung, Federated Databases, etc.
    - Master-Master?
    - High Availability?
      - 99,999% Verfügbarkeit?
        - Five nines: ~5 Minuten Downtime/Jahr
- DWIM!

Die Haltung des Core-Teams

29.5.2008, pgsql-hackers, Tom Lane:

- Historically the project policy has been to avoid putting replication into core PostgreSQL, [..]
- However, it is becoming clear that this policy is hindering acceptance of PostgreSQL to too great an extent, [..]
- [..] it is time to include a simple, reliable basic replication feature in the core system.

„Was bisher geschah“

- 7.4.3 (Juni 2004)
  - Slony-I (Trigger-Basiert)
  - pgpool (Middleware)
- 8.2 (Dezember 2006)
  - Log Shipping
- 8.4devel (Mai 2008)
  - Manifest des Core Teams
- 8.4 (Mai 2009)
  - :(
  - Skytools & Freunde

DO NOT USE REPLICATION

Populäre Irrtümer

Some people, when confronted with a problem, think

„I know, I'll use database replication.“

Now they have two problems.

Frei nach Jamie Zawinski's „I'll use regular expressions“

Gern durchgeführte Simplifizierung der LAMP-Generation.

Do NOT use replication

- Replication ain't a Silver Bullet
  - Zuerst das Problem verstehen, dann nach Lösungen suchen.
    - Wie lang kann ich jetzt skalieren?
    - Wo sind die Bottlenecks?
    - Was für Lösungswege gibt es?
    - Wie lange skalieren diese?

Do NOT use replication

- Gesteigerte Komplexität
  - In der Architektur
    - DB Connection Handling (Reader vs. Writer)
    - Synchrone & Asynchrone Replikation
    - Was passiert wo?
  - Im Operating
    - Monitoring
    - Maintenance
    - Backup & Recovery

Do NOT use replication

- Es muss nicht immer ein RDBMS sein
  - Caching
    - memcached
    - Framework-spezifische Lösungen
    - auch das ist schon Daten-Replikation
  - Spezielle Datenbanken
    - CouchDB
    - TokyoCabinet
  - Alternative Kommunikationswege
    - Message Queues
    - Group Communication

Do NOT use replication

- Ist die Datenbank und Applikation überhaupt optimiert?
  - Indizes
  - Queries & Applikation
  - Hardware
  - Postgres Konfiguration

Gute Gründe für Replikation

Gute Gründe für Replikation

- Redundanz & Hochverfügbarkeit
  - Redundanz
    - „Mehr als eins“
  - Hochverfügbarkeit
    - Synchrone Replikation
    - Automatisches Failover

Gute Gründe für Replikation

- Datenverteilung
  - Knoten mit identen Daten
    - Skaliert nur bei konstantem „Working Set“ und gleichbleibenden Write-Aufkommen
  - Knoten mit unterschiedlichen Daten
    - zB Zentrale ↔ Filialen

Gute Gründe für Replikation

- Horizontale Skalierung
  - Kombination aus Partitioning & Replikation
  - Notwendige Dinge
    - Klassifizierung der Daten & Definition der Segmente
      - zB: Stored Procedures, Applikations-Code
    - Verteilung der Daten
      - zB: dblink, plproxy, pgQ, etc.
    - Verteilen der Anfragen
      - zB: Middleware, Stored Procedures, etc.
- Keine fertigen Lösungen!

Gute Gründe für Replikation

- Archivierung & Backup
  - Komplette Sicherung oft Zeit- und IO-Intensiv
  - Performance-Einbußen auf Live-System

Gute Gründe für Replikation

- Warehousing & Datenauswertung
  - „Business Intelligence“
  - Auswerten von Daten in einem (R)DBMS
  - Weg vom Live-System!
    - Lang laufende Queries
    - Oftmals keine Notwendigkeit für stundenaktuelle Daten

Gute Gründe für Replikation

- Datenrettung & Forensik
- Point In Time Recovery (PITR)
  - Erlaubt einen beliebigen Zeitpunkt in der Datenbankhistorie herzustellen
  - Nachvollziehbarkeit von Modifikationen
  - „Rollback“ zu einem Zustand vor einem „unglücklichen Query“

Theorie

Theoretisches

- Transaktion
- ACID
- Master & Slave
- Synchronität

Transaktion

- Eine feste Folge von Operationen, die vollständig oder garnicht ausgeführt wird

```
BEGIN;

UPDATE accounts SET balance = balance - 100.00
    WHERE name = 'Alice';

UPDATE accounts SET balance = balance + 100.00
    WHERE name = 'Bob';

COMMIT;
```

ACID

- Atomicity
- Consistency
- Isolation
- Durability

ACID

- Atomicity
  - Eine Transaktion wird komplett oder garnicht ausgeführt
- Consistency
  - Der Zustand der Datenbank ist immer konsistent
    - Augenmerk auf referentielle Integrität, Validierung von Daten

ACID

- Isolation
  - Transaktionen sind durch ein definiertes „Isolation Level“ voneinander geschützt
- Durability
  - Eine Transaktion die erfolgreich „committed“ wurde muss bestehen bleiben.

Master & Slave

- Master
  - Ist authorative Quelle für Daten
  - Darf datenverändernde Statements verarbeiten
- Slave
  - Darf (wenn überhaupt) nur lesende Statements verarbeiten
  - Bekommt Daten von einem anderen Knoten

Multi-Master

- Mehrere „Master“-Server
- Zumeist Synchrone Replikation
  - oder „Conflict Resolution“-Mechanismen
- Triviales Failover
- Skaliert nicht

Synchronität

- Synchrone Replikation
  - Commit returned erst, wenn zumindest ein synchron replizierter Knoten geantwortet hat
- Asynchrone Replikation
  - Commit auf Master darf sofort returnen
  - Keinerlei Garantie über Zustand & Aktualität von anderen Knoten

Synchronität

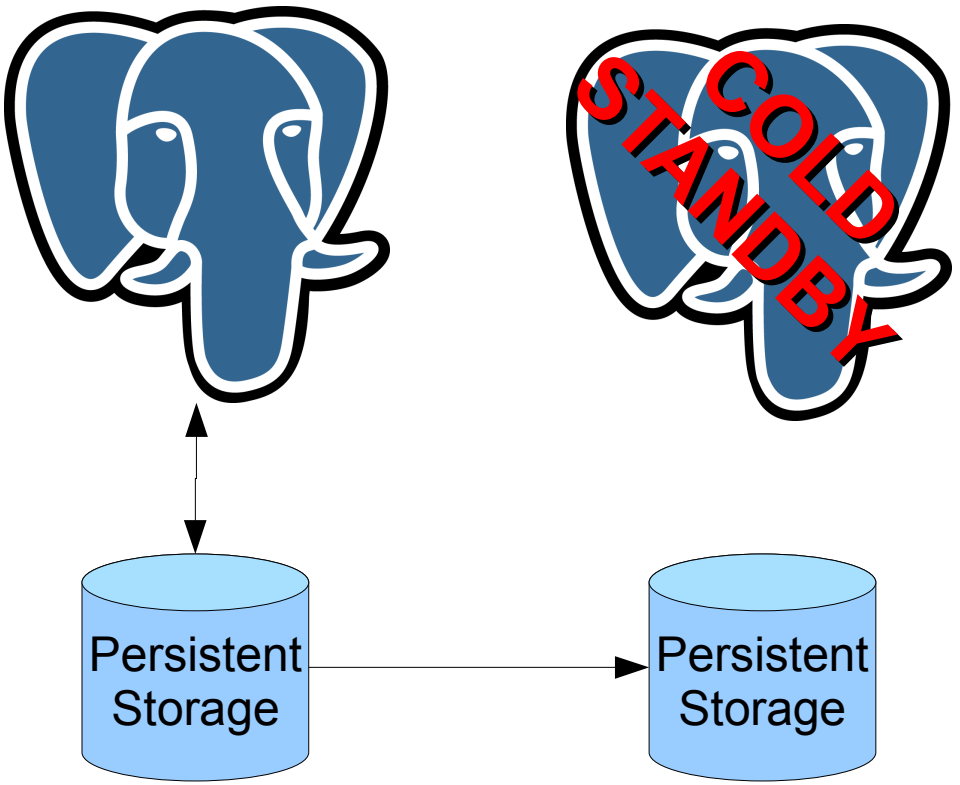
- Synchrone Replikation
  - Langsamer als Single-Server
  - Benötigt niedrige Latenzen
- Asynchrone Replikation
  - Slave Lag!

Über das Verteilen von Daten

Ansätze

- Storage & OS-basierte Methoden
- Log-Shipping Systeme
- Triggerbasierte Systeme
- Middleware
- Forks & Patches von und für PostgreSQL

Storage & OS Methoden



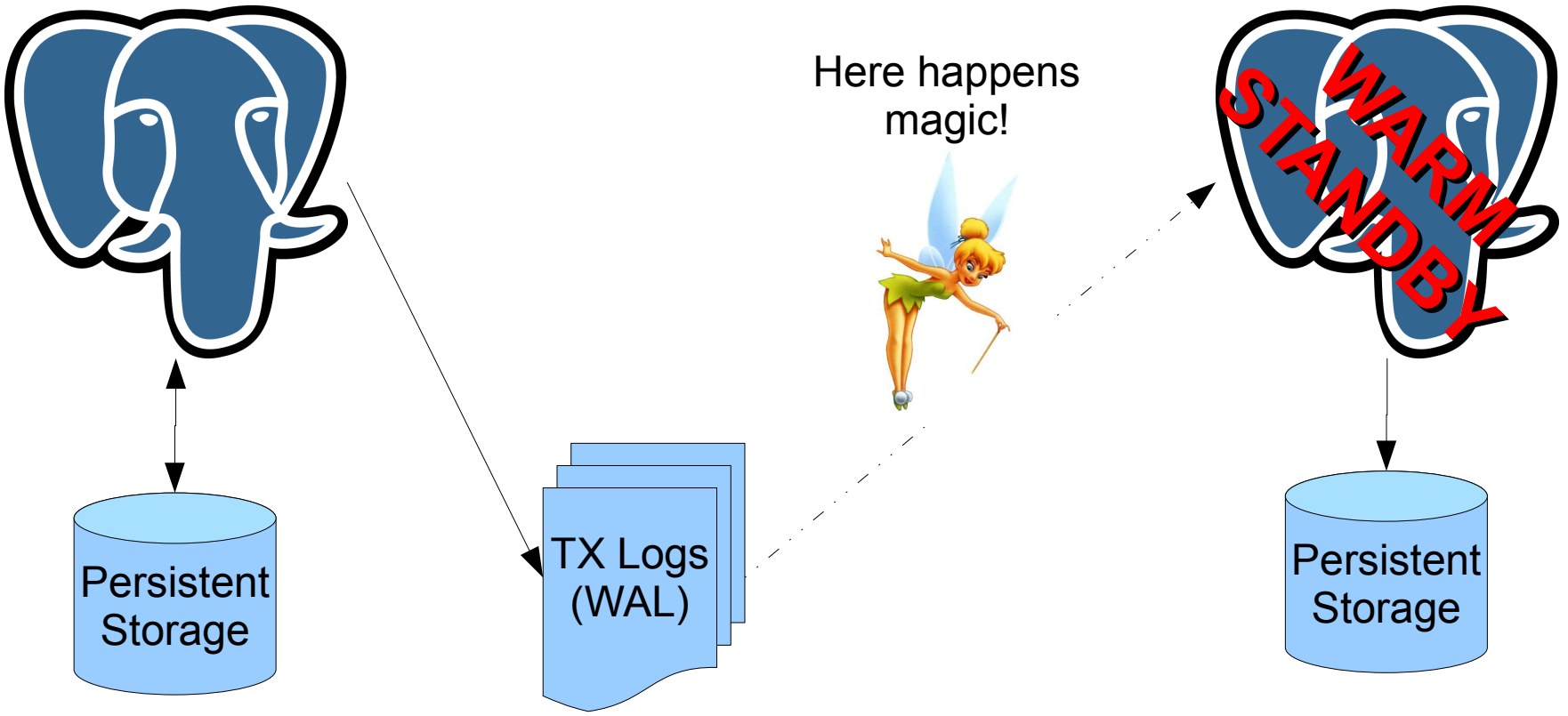
Storage & OS Methoden

- Replizieren des Blockdevices
  - Verschiedene Lösungen für (a)synchrone Replikation, zB:
    - DRBD
    - EMC MirrorView
    - NetApp SyncMirror
  - ACID & synchrone Replikation auf Block-Ebene garantieren jederzeit idente Knoten

Storage & OS Methoden

- Vorteile
  - Datenbankunabhängig
  - Ermöglicht einfaches „Site-Failover“
- Nachteile
  - Unflexibelste Lösung
  - Erfordert Storage-Systeme oder OS-Support
  - Meistens Cold Standby

Log-Shipping Systeme



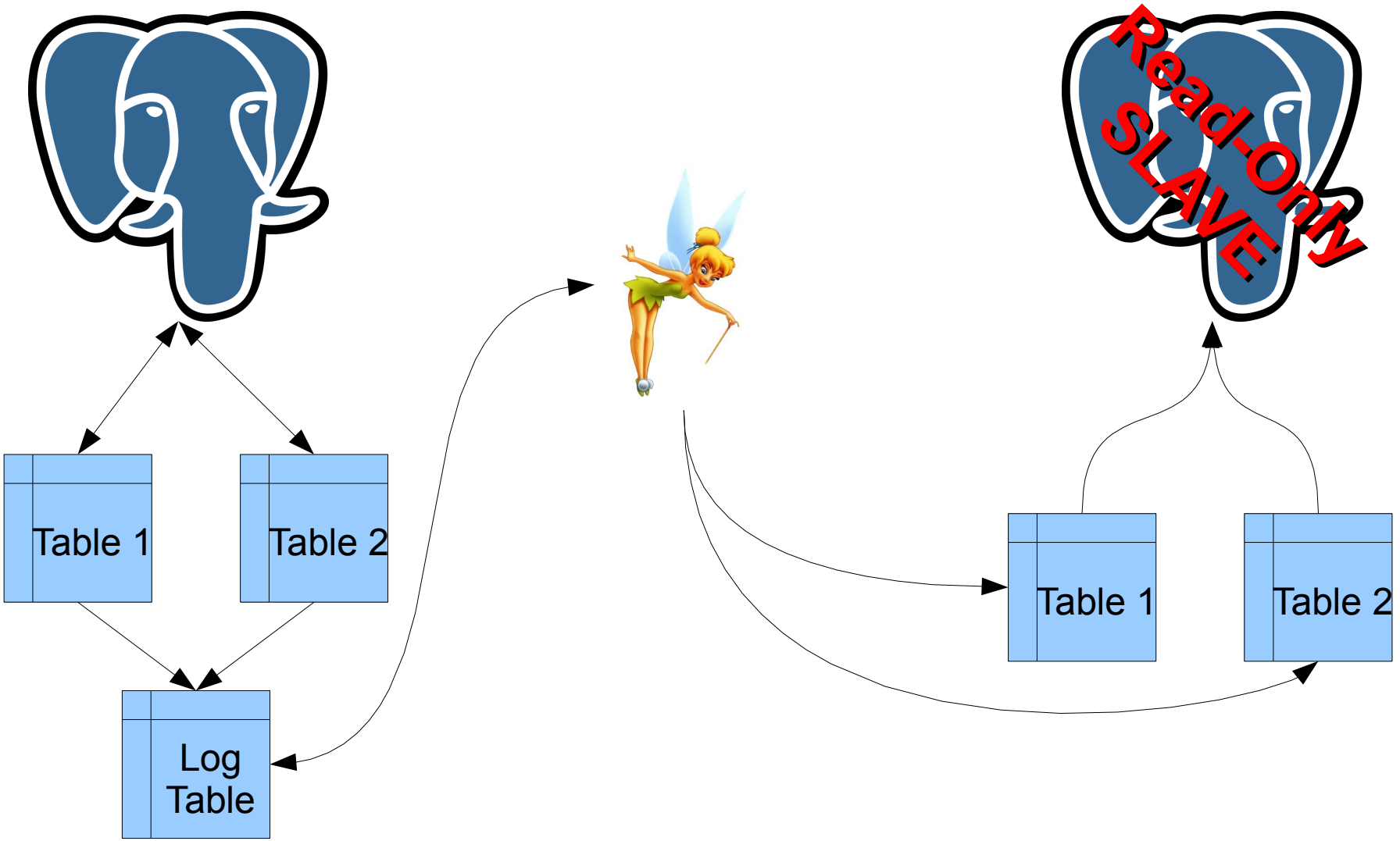
Log-Shipping Systeme

- Verwendet die Transaktionslogs der Datenbanksysteme
  - PostgreSQL WAL Shipping
  - Oracle Data Guard
  - DB2 HADR
- Auf Slave-Systemen werden kontinuierlich die Änderungen vom Master nachgezogen
- Row-basiert

Log-Shipping Systeme

- Vorteile
  - Geringer Overhead
  - Hohe Robustheit
- Nachteile
  - Kann nur komplette Datenbanksysteme replizieren
  - In Postgres nur asynchrone Warm Standby Slaves möglich (8.5!)

Trigger-basierte Systeme



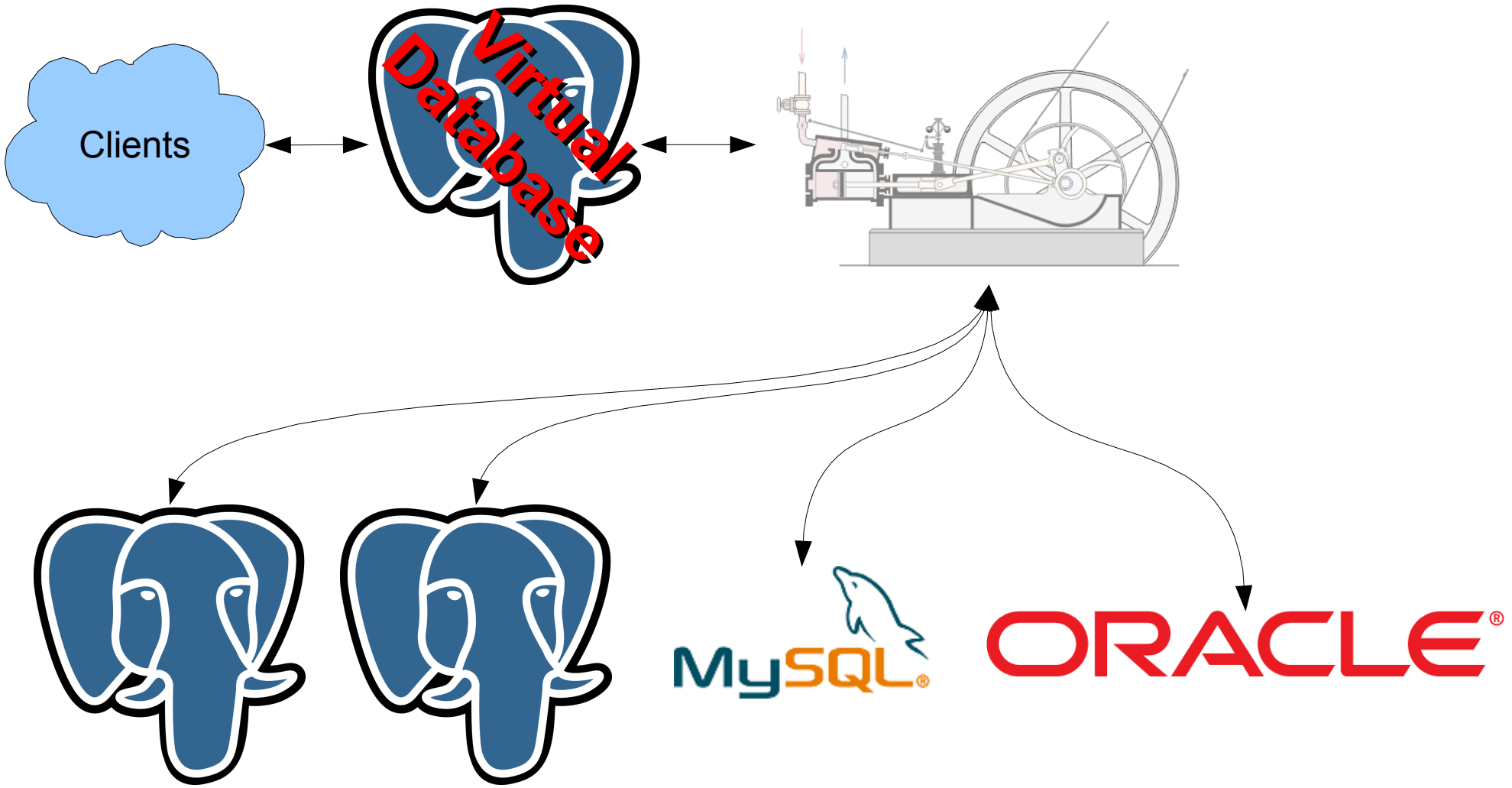
Trigger-basierte Systeme

- Verwenden Trigger & Stored Procedures
- Alle Änderungen an Tabelleninhalten werden in Log-Tables geschrieben
- Separate Daemons verteilen Daten an Slaves
- Erlaubt verschiedene Master für unterschiedliche Datenbankobjekte

Trigger-basierte Systeme

- Vorteile
  - Sehr flexibel
- Nachteile
  - Komplex
  - Schema-Änderungen nur über Umwege möglich
  - Hoher Overhead für Writes
    - ~2,5x Schreibmenge

Middleware



Middleware

- Sitzt zwischen Datenbank und Client
- Simuliert eine virtuelle Datenbank
- Komplett eigenständige Software
- Kann verschiedenste Aufgaben übernehmen
  - Replikation, Sharding
  - Connection Pooling & Brokering
  - Load Balancing, HA
  - Parallelized Query Execution

Middleware

- Vorteile
  - (Teilweise) Datenbankunabhängig
  - Kann Multi-Master-Szenarien umsetzen
- Nachteile
  - Sehr komplex
  - Eigener, separat zu entwickelnder & zu wartender Layer
  - (Teilweise) Ressourcenhungrig

Patches & Forks

Patches & Forks

- Komplexe Features die tiefgreifende Modifikationen an PostgreSQL erfordern
- Ein Auszug:
  - Greenplum: Greenplum Database
    - Sharding & Datawarehousing
  - Command Prompt: PostgreSQL + Replication
    - Eingebaute Async Master → Slave Replikation
  - EnterpriseDB: Postgres Plus
    - PL/SQL, Oracle ↔ Postgres Replication, Datawarehousing

Pause!

15 Minuten

Praxis

- Log Shipping
  - Vanilla PostgreSQL & pg_standby
- Trigger
  - Slony-I

Write Ahead Log

- Enthält alle Änderungen an Datenfiles
- Facts
  - Zu finden in $PGDATA/pg_xlog
  - Name enthält 3 Zähler á 32 Bit in Hex-Notation
    - zB 000000010000000000000008E
    - Timeline
    - Logid
    - Segmentnummer
  - 16MiB Files, in 8KiB Pages unterteilt

WAL cont'd

- WAL-Files werden
  - Gewechselt & „archiviert“
    - Wenn sie voll sind
    - Wenn archive_timeout abgelaufen ist
    - Manuelle Intervention
  - In pg_xlog recycled bzw. gelöscht
    - Sobald ein WAL-Segment gewechselt wurde...
    - ...und keine offenen Transaktionen auf sie verweisen

WAL Shipping

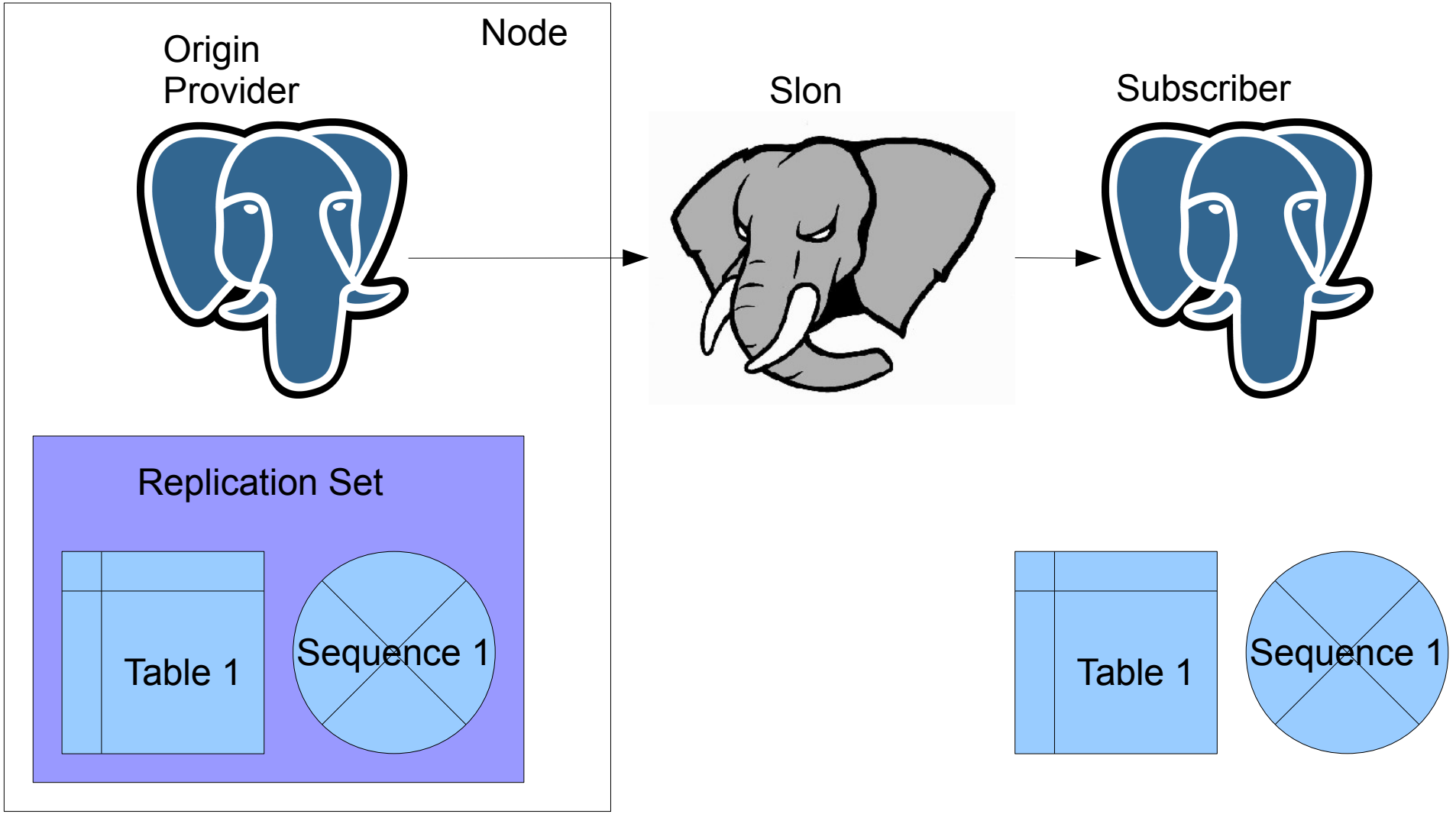
- Master muss
  - Archivieren
    - archive_mode & archive_command
- Slave benötigt
  - Base Backup
    - pg_start_backup()
  - recovery.conf
  - Alle WAL-Files seit dem Base-Backup

Slony-I

- Begrifflichkeiten
  - Cluster
  - Node
  - Replication Set
  - Origin, Provider, Subscriber
  - Slon daemons
  - slonik

Slony-I Schematisch

Cluster



Neu in 8.4

- On-Disk Free Space & Visibility Map
- Parallel Restore
- Auto-Explain
- Common Table Expressions & Recursive Joins
- Windowing Functions
- Performance...
- ...und noch einiges mehr...

8.5 and beyond

- (Synchronous) Native WAL replication
  - NTT Projekt
  - 8.5!
- Read-Only Logshipping Slaves
  - Simon Riggs/2nd Quadrant Projekt
  - 8.5!
- Bestrebungen in Richtung Sharding
  - Foreign Data Wrappers, SQL/MED, SkyTools, etc.
  - Infrastruktur, Werkzeuge, Best Common Practices...

Best Common Practices

- Verbindliche Richtlinien zum Einziehen von Datenbank-Replikation in Ihrer Infrastruktur
  - Planung (Jetzt & Zukunft)
  - Dokumentation (Architektur, Naming conventions)
  - Prozesse (DDL, Reintegration nach Ausfall, etc.)
  - Monitoring (Slave Lag, Performance, etc.)
  - Desaster-Szenarien (Warme Failover-Server, etc.)
  - Hardwarekosten ein Server:
    - entspricht ~80 Arbeitsstunden eines Angestellten
    - oder ~30 Arbeitsstunden eines externen Dienstleisters

Common Use Cases

- Oder: Was soll ich jetzt eigentlich verwenden?!
- Still no Silver Bullets
  - Log Shipping (pg_standby, walmgr, etc)
    - Backup
    - Warehousing
    - Failover
  - Slony
    - Online Slaves mit geringer Schreiblast
    - Dezentrale Datenquellen
- Alles andere: RESEARCH

Interessante Projekte

- Liste anbei
- TMTOWTDI
- Größte Userbasis derzeit hinter
  - Log shipping
  - Slony-I
  - Skytools

Community

- IRC
  - irc.freenode.net, #postgresql, #postgresql-de
- Mailinglisten
  - pgsql-general, pgsql-performance, pgsql-de-allgemein
- Web
  - http://postgresql.org/
  - http://wiki.postgresql.org/
  - http://pgug.de/

Danke!

Fragen?

michael.renner@amd.co.at
Robe @ #postgresql(-de), Freenode

Links

http://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling

http://www.postgresql.org/community/lists/

http://wiki.postgresql.org/wiki/Skytools

Replikationslösungen

Produkt/Projekt	Art	Lizenz	Status	Link
walmgr	Log Shipping	BSD	Production	https://developer.skype.com/SkypeGarage/DbProjects/SkyTools/WalMgr
synch rep	Log Shipping	BSD	alpha	http://wiki.postgresql.org/wiki/NTT's_Development_Projects#Synch_Rep
Hot Standby	Log Shipping	BSD	alpha	http://wiki.postgresql.org/wiki/Hot_Standby
Slony-I	Trigger-Basiert	BSD	Production	http://www.slony.info/
Bucardo	Trigger-Basiert	BSD	Production	http://bucardo.org/
Londiste & pgQ	Trigger-Basiert	BSD	Production	https://developer.skype.com/SkypeGarage/DbProjects/SkyTools
Sequioa & Tungsten	Middleware	Apache	Production	http://community.continuent.com/community/sequoia
pgpool-II	Middleware	BSD	Production	http://pgpool.projects.postgresql.org/
PgBouncer	Middleware	BSD	Production	https://developer.skype.com/SkypeGarage/DbProjects/PgBouncer
PostgreSQL + Replication	Fork	BSD?	Production	http://www.commandprompt.com/products/mammothreplicator/
Greenplum Database	Fork	kommerziell	Production	http://www.greenplum.com/products/greenplum-database/
Postgres Plus AS	Fork	kommerziell	Production	http://www.enterprisedb.com/products/postgres_plus_as/overview.do
PL/Proxy	Library	BSD	Production	https://developer.skype.com/SkypeGarage/DbProjects/PlProxy