2ndQuadrant®
PostgreSQL

Ist Dein PostgreSQL logisch genug für bidirektionalität?

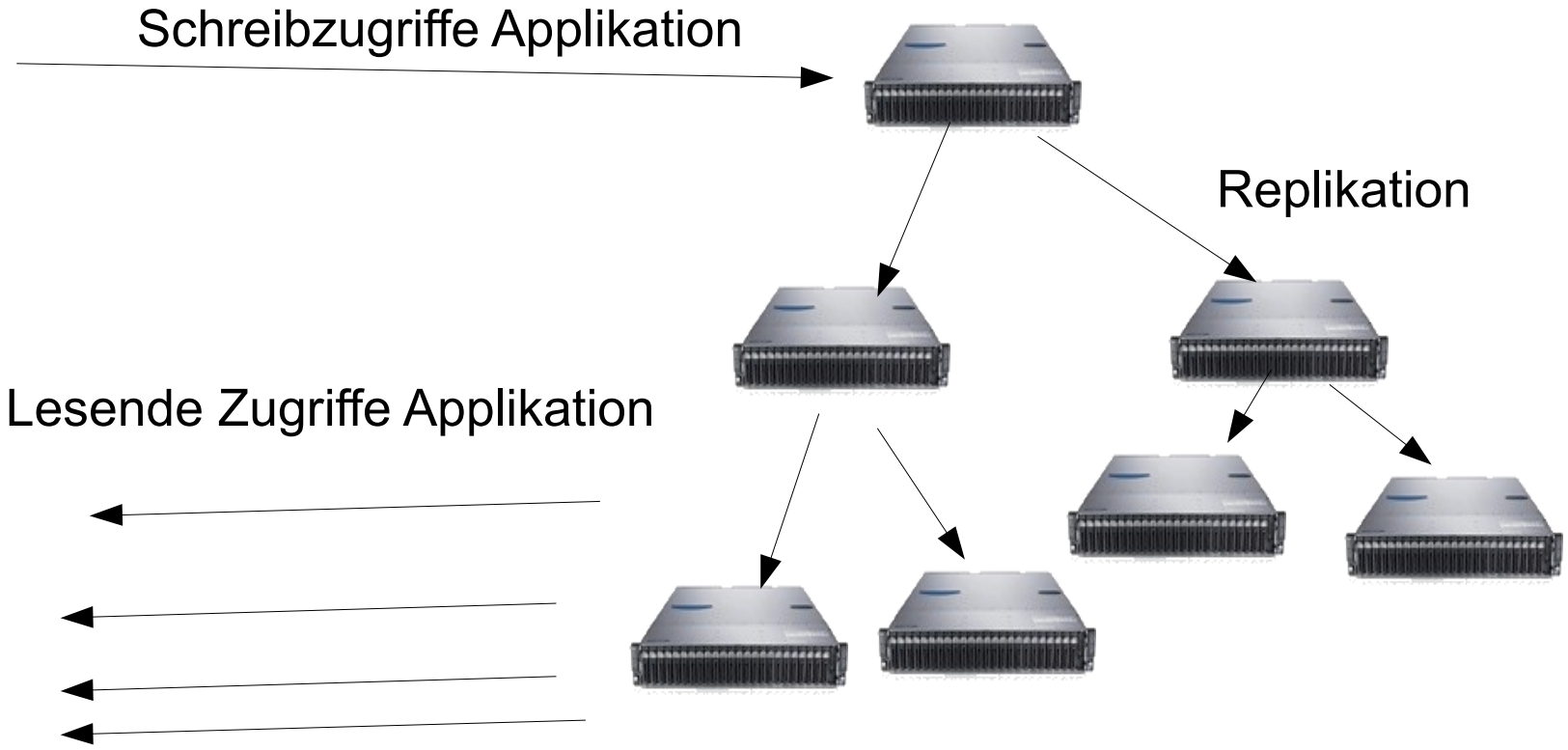
2018-06-29, Swiss PGDay 2018

Harald Armin Massa

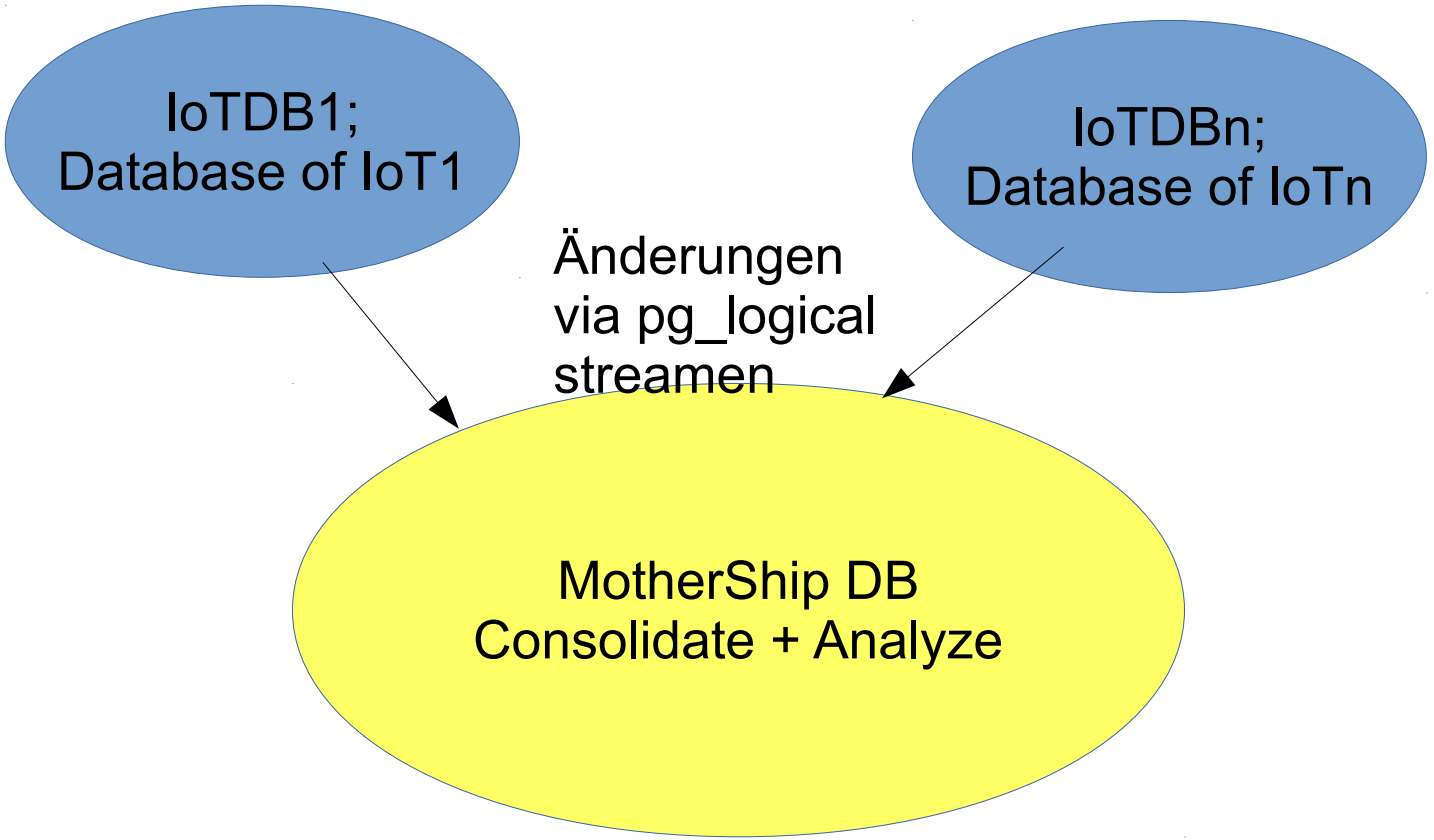
Replikation, warum?

- Skalierung
- unterschiedliche Nutzung der Daten
  - OLTP / OLAP
  - Unterschiedliche Orte (z.B. Auskunft im Web)
- Verfügbarkeit

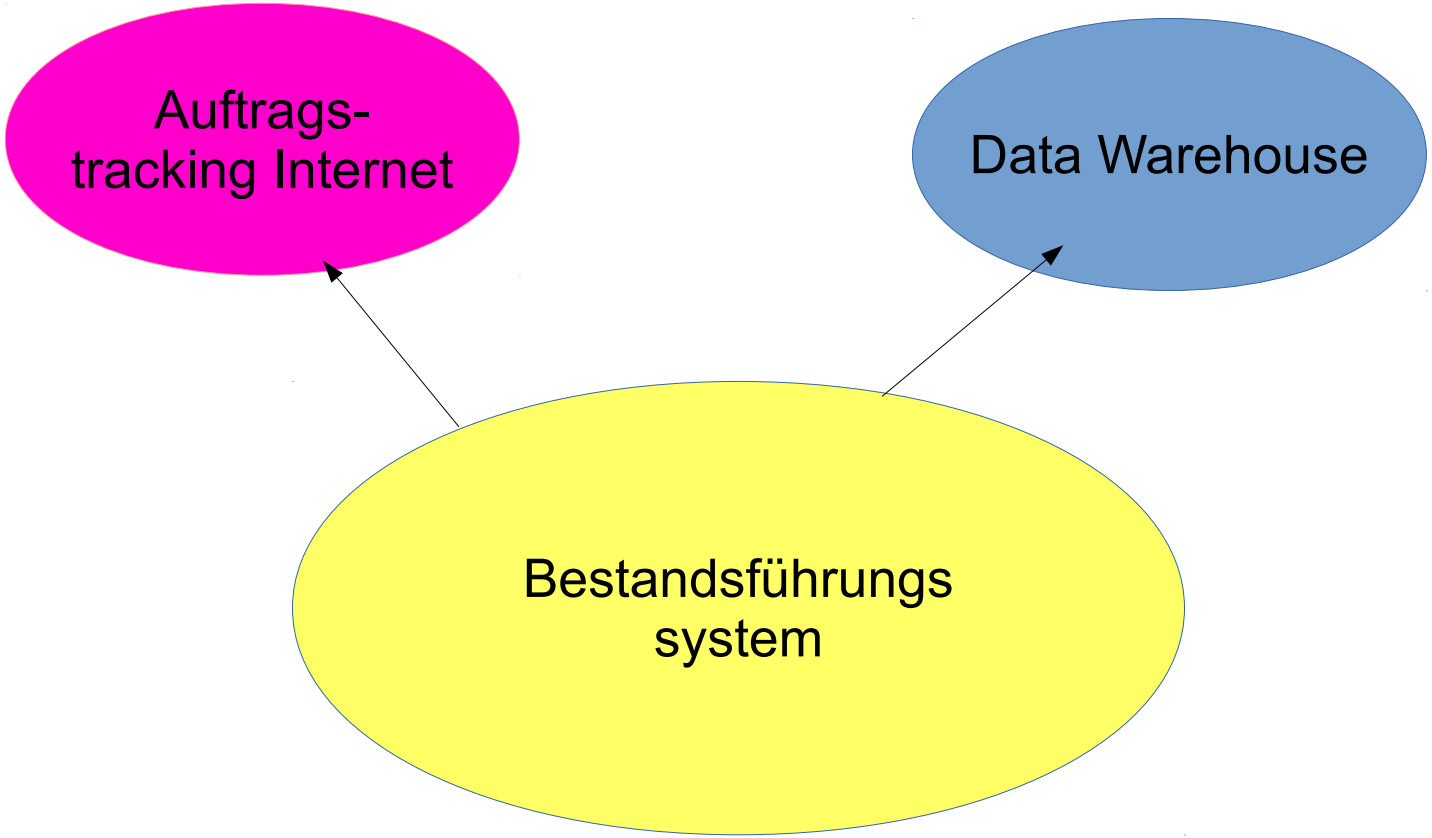
Leseskalierung



Konsolidierung



Verteilung


Auftrags-
tracking Internet
Data Warehouse
Bestandsführungs
system

Replikationsentwürfe

Statement
based Replicationm

Replikation von
(DML Statements

(UPDATE,
INSERT,
DELETE)

COMMIT based
Replication

Replikation von
(geschriebenen)
Daten

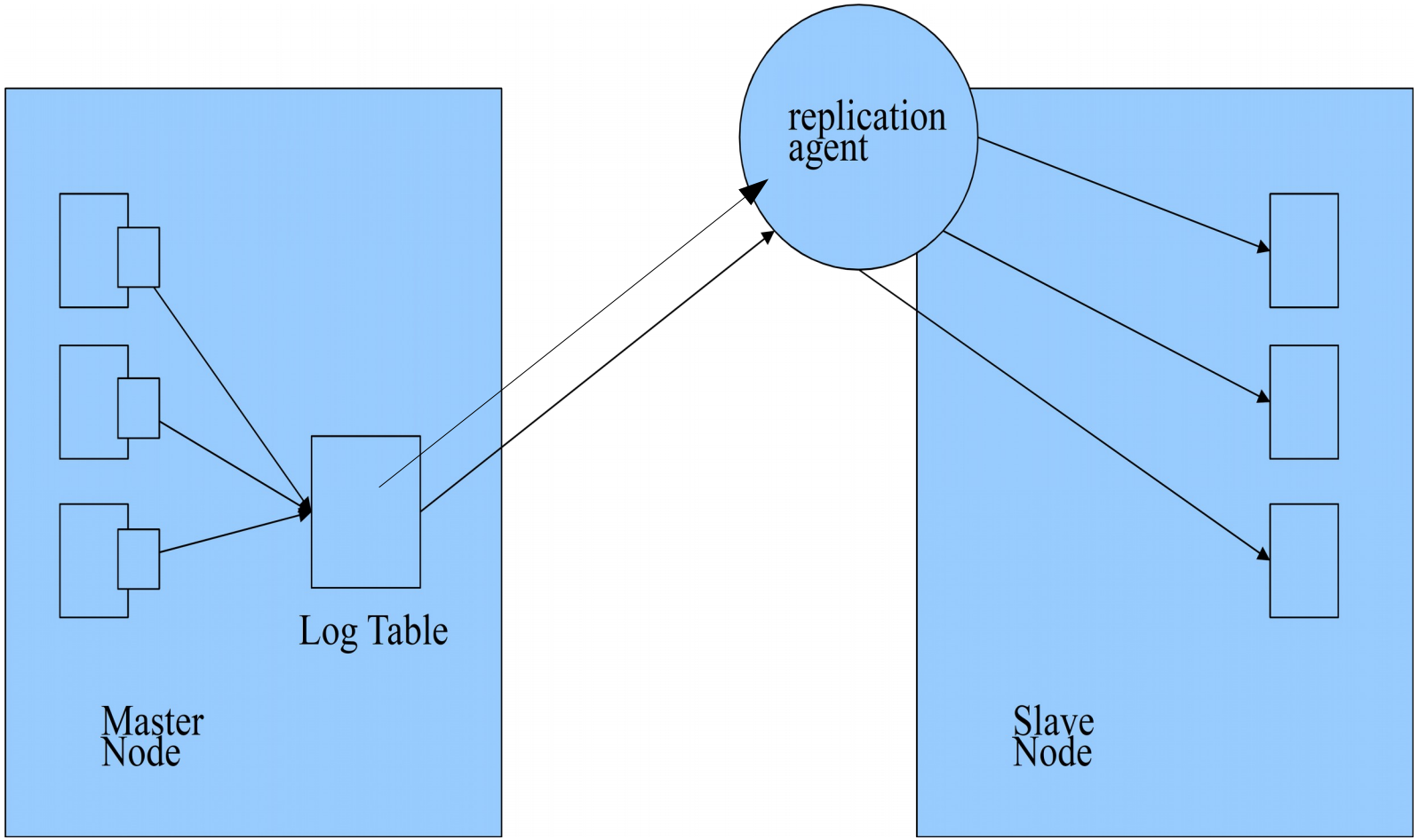
Statement basierte Replikation

- Beispiele
  - MySQL, Pg_pool
- Vorteile
  - Switchover / Failover leicht
  - Multimaster möglich
  - Minimaler Performanceimpact
  - Konzeptionell einfach
- Probleme
  - Datenkonsistenz
    - Volatile Funktionen (… values (now()...))
    - Statement-Reihenfolge bei Concurrency

Replikation basierend auf Datenänderungen

- Beispiele
  - Slony, Londiste, Bucardo
  - PostgreSQL LogShipping Replication
  - PostgreSQL Physical Streaming Replication
  - PostgreSQL LOGICAL Streaming Replication
- Vorteile
  - Datenkonsistenz
    - (volatile Funktionen)
    - Concurrency
- Nachteile
  - Multi Master knifflig
  - Performance-Impact

Trigger basierend



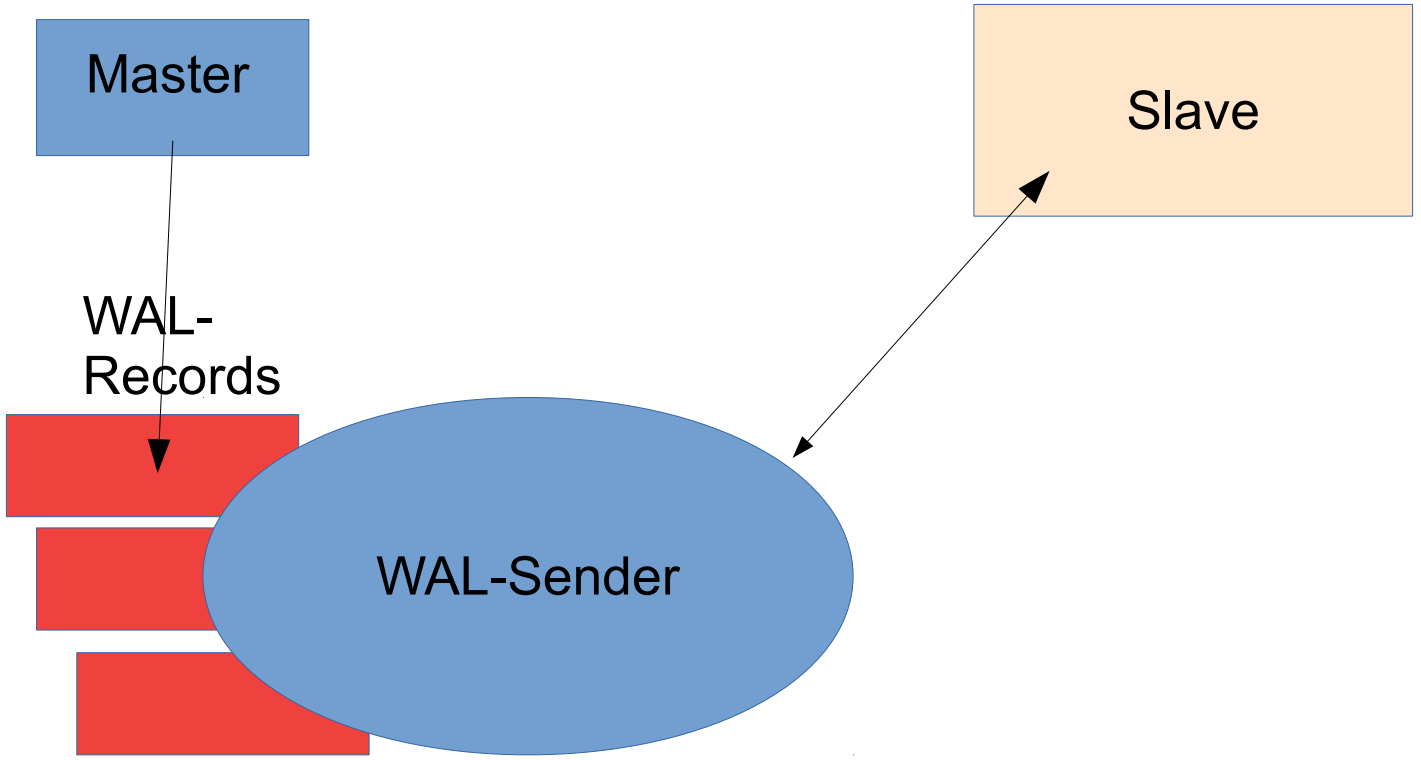
Vor- und Nachteile, Triggerbasierend

- Konsolidieren und Verteilen möglich
- Zwischen unterschiedlichen Versionen von PostgreSQL möglich
- (Sogar other → PostgreSQL ist möglich)
- Multimaster theoretisch möglich (?Bucardo?)

  ABER
- Data werden 4x geschrieben

  WAL, Datafile, QueueTable, WAL(QT)
- Kosten der Trigger, Kosten des Parsens
- Schemaänderungen = Schmerzen

Physical Streaming Replication



2ndQuadrant PostgreSQL

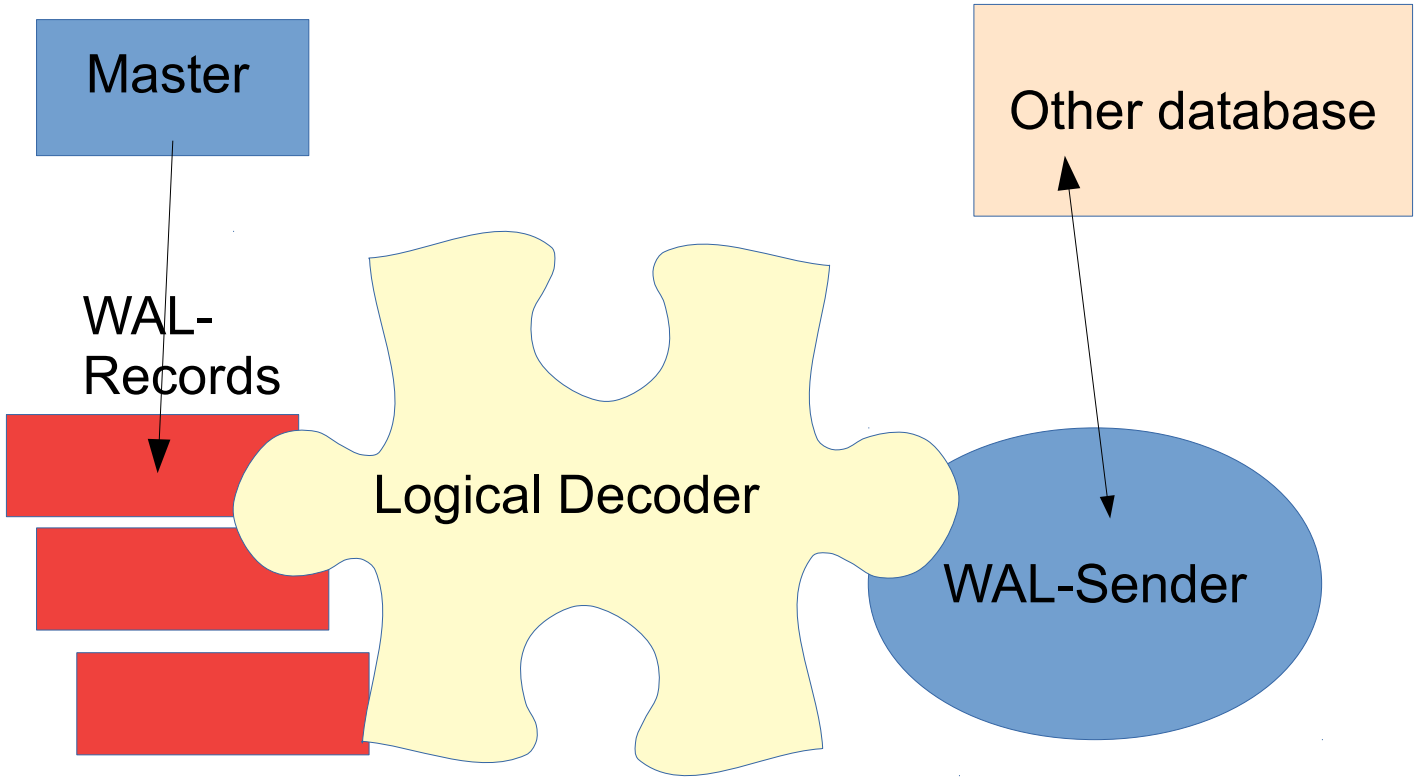
Physical Streaming Replication (binary...)

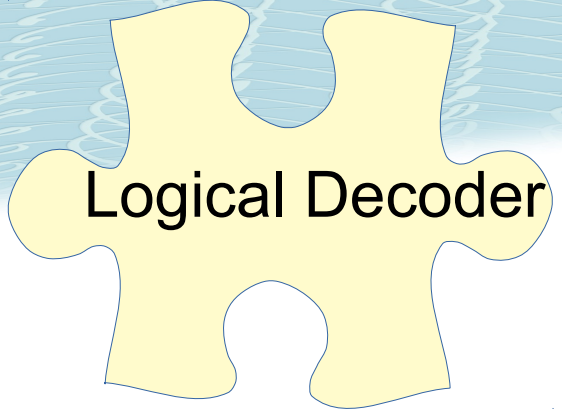
- Gesamter Cluster
- Schemaänderungen schmerzfrei
- wenig Last auf Master
- Kein Parsingoverhead
- Slave ist Read Only
- Sichern der WAL erlaubt PITR und DR

ABER:

- Kein Multimaster möglich
- Kein Konsolidieren oder Verteilen möglich
- Transfer administrativen Traffics (Vacuum, index)

LOGICAL Streaming Replication





2ndQuadrant®
PostgreSQL

- Liest (physical, binary) WAL-Traffic
- Anreichern der Daten
- Logischer zwischencode, “SQL-Like”
- Ergebnis der volatilen Funktionen
- Nur Daten, keine Struktur
- Keine Replikation von administrativem Traffic (Vacuum, Index)

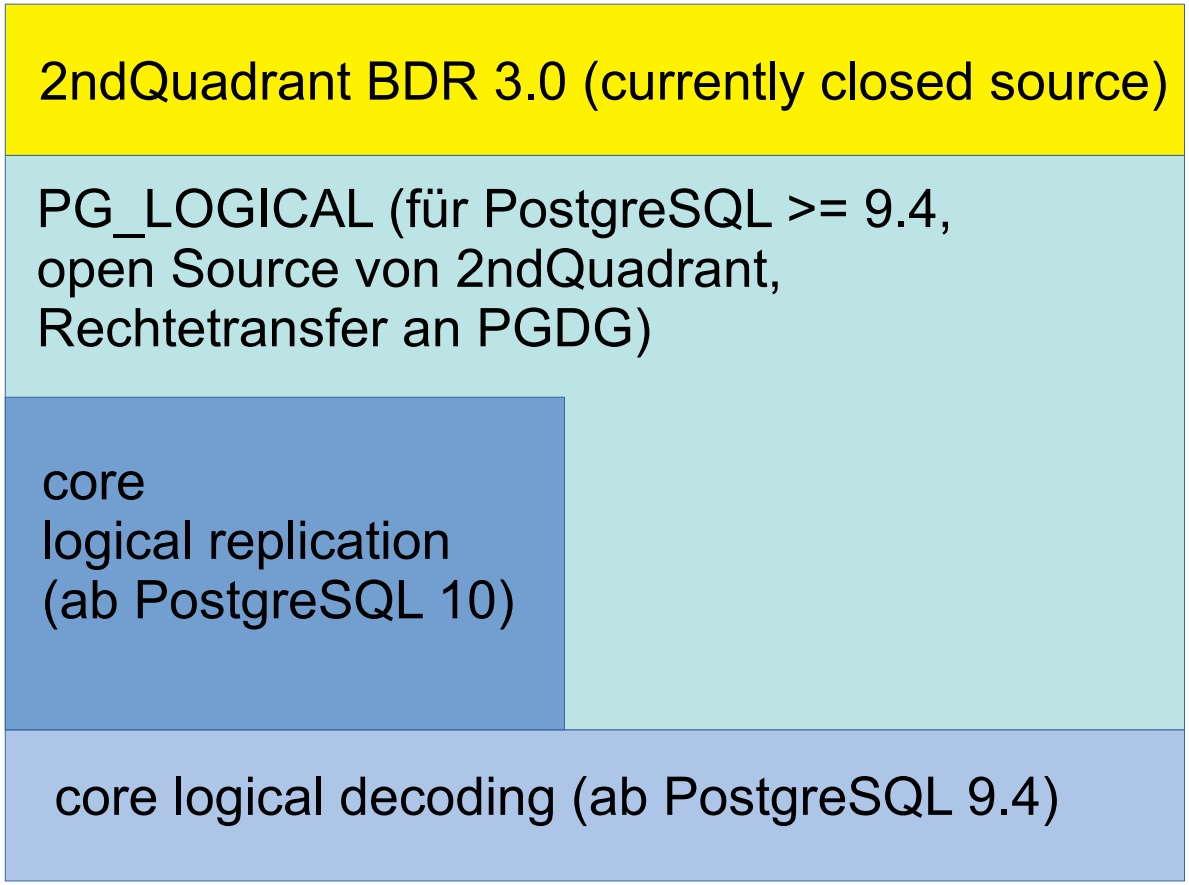
Logische Streaming-Replikation, was geht

- Datenformat weitgehend Versionsunabhängig
  - NZDT Upgrades zwischen Major versionen
- Selektive Replikation von Gruppen von Tabellen
  - Zusammenführen (Konsolidierung)
  - Verteilung
- Zeilenfilterung auf Sender oder Empfängerseite
- Asynchron + synchron möglich

Herausforderungen

- **DDL (=Schemaänderungen)**

Die Komponenten



Unterschiede, was und wie: pg_logical

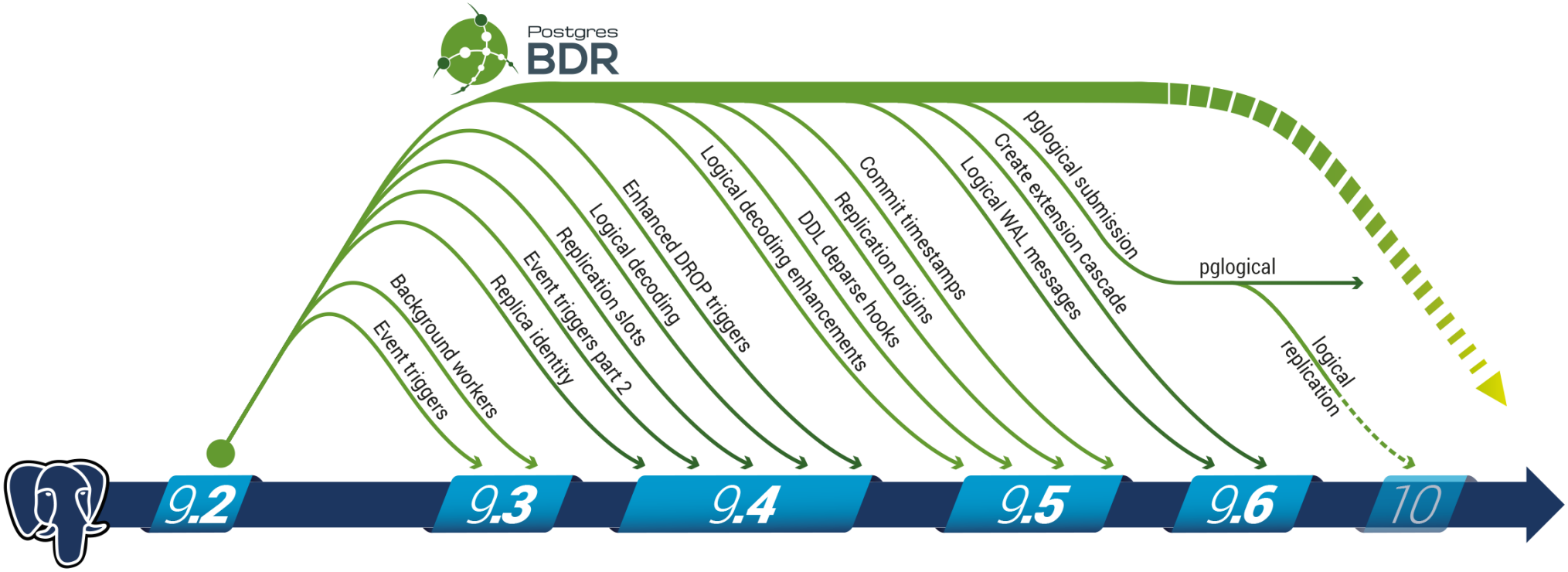
- pg_logical ist Obermenge von in-core logical replication, Mehrfunktionen
  - Unterstützung der manuellen DDL-Replikation
  - Transfer der Tabellenstruktur
  - Filterfunktionen
  - Mehr Konflikt-Behandlungs Optionen
- Logical decoding, pg_logical entstanden aus BDR Projekt

BDR 3

- Nutzt pg_logical als Transportschicht
- Excerpte der Mehrinhalte:
  - DDL Support: Capture, Synchronize
  - Globale Sequenzen
  - Eager + efficient consistency
  - Tooling für Very High Availability via Shadow Master
  - Plugable für Konfliktresolution

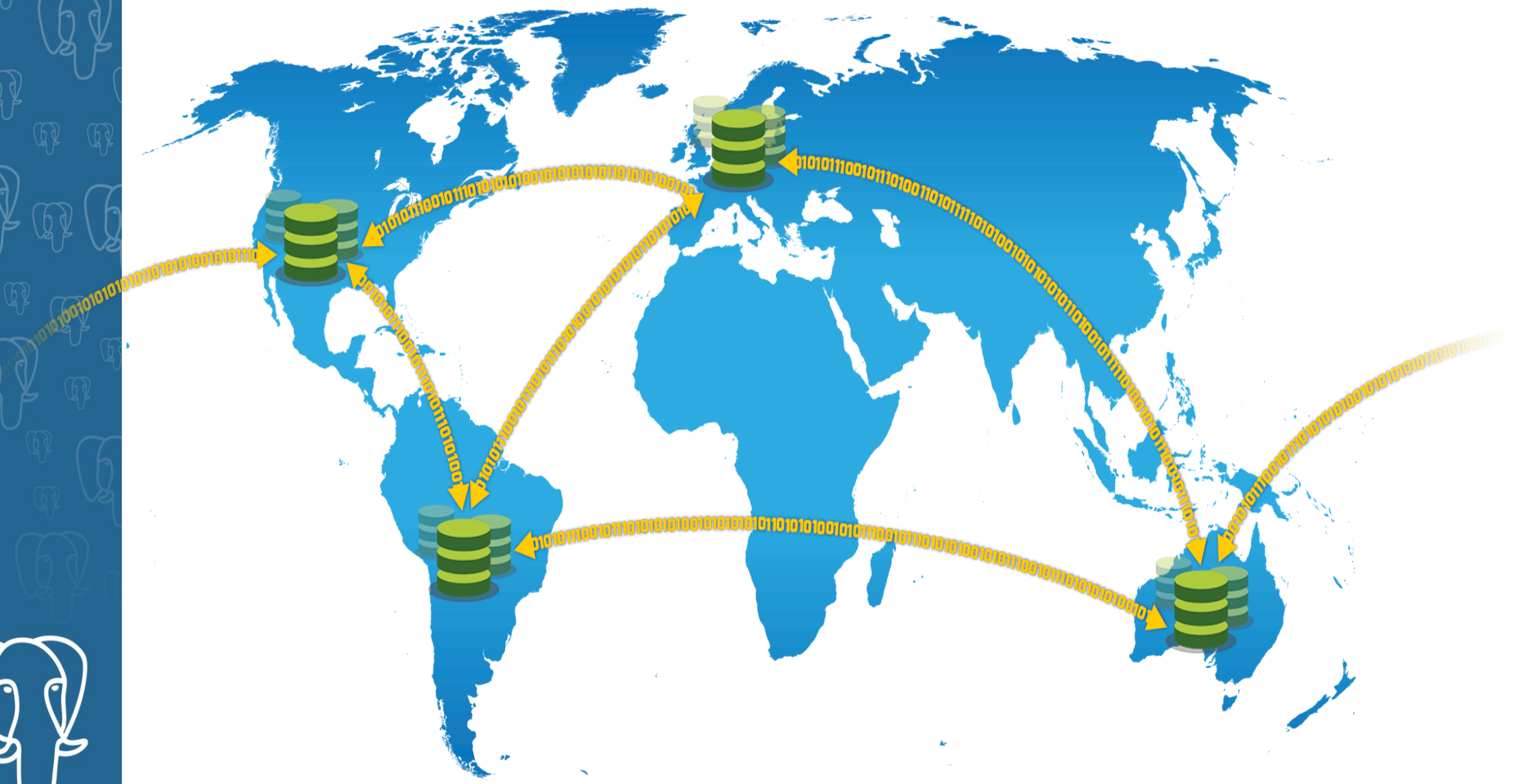
2ndQuadrant
PostgreSQL

BDR-Armadillo:
Infrastruktur für logische Replikation



2ndQuadrant®
PostgreSQL

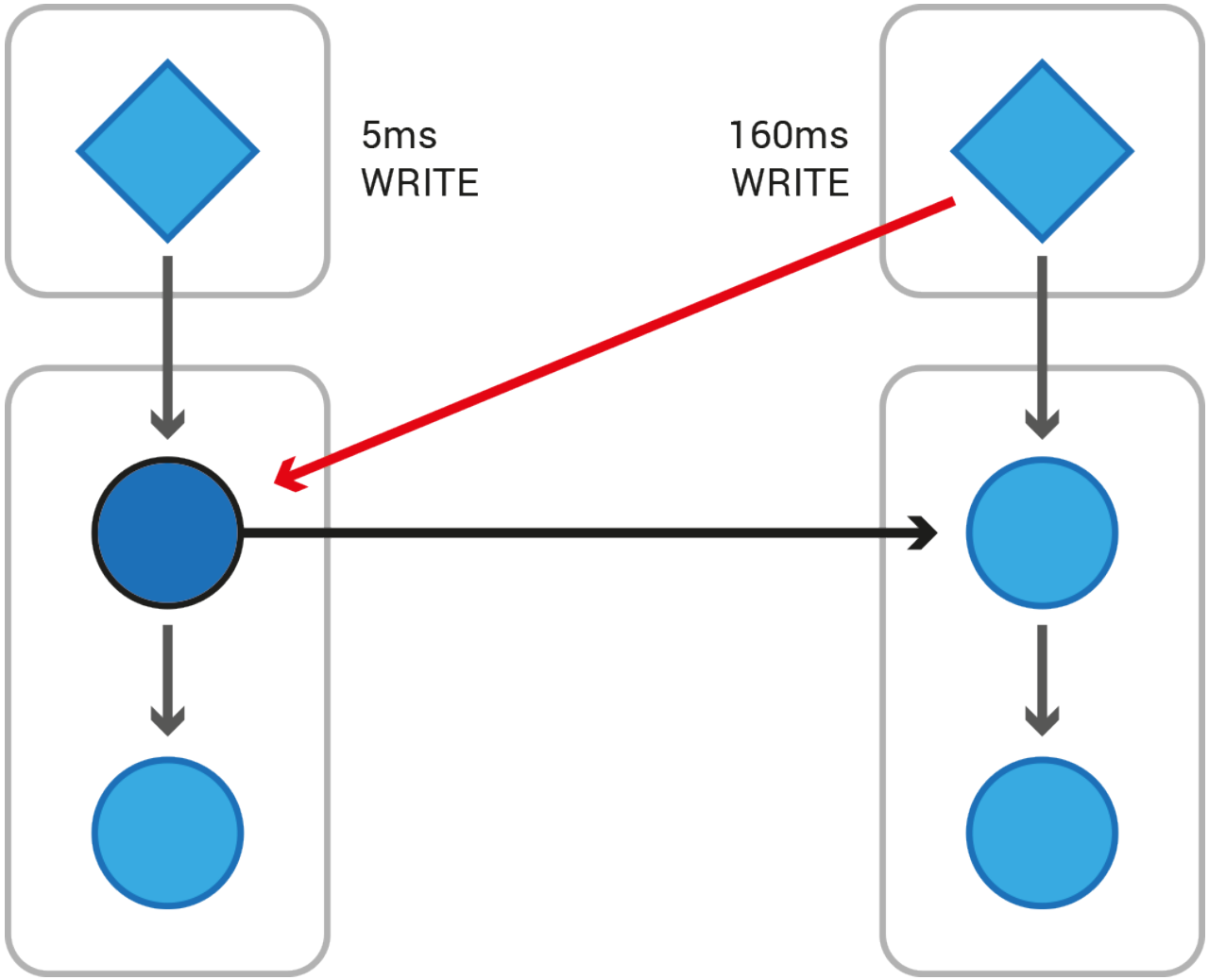
Georedundanz



PostgreSQL

2ndQuadrant PostgreSQL

PostgreSQL Single Master



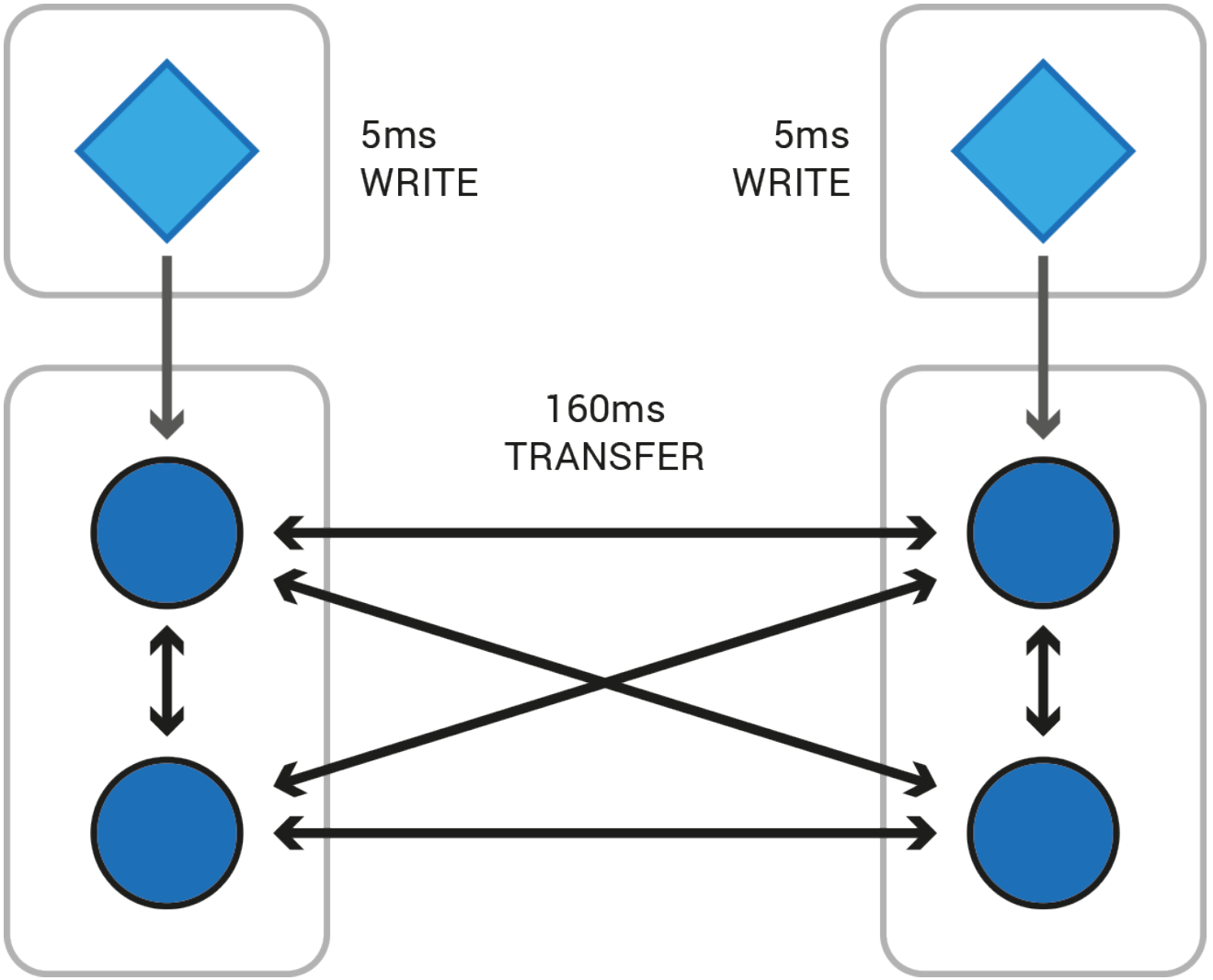
PostgreSQL

Konsistenz über weite Strecken

- PACELC als Erweiterung von CAP (Consistency, Availability, Partitiontol.)
  (Partition: Availability, Consitency
  Else: Latency, Consistency)
- Constiency:
  - Warten, bis Änderungen auf allen Knoten umgesetzt sind
- Latency
  - Kein Warten
  - Schnell für Nutzer
  - Konflikte müssen gelöst werden

2ndQuadrant PostgreSQL

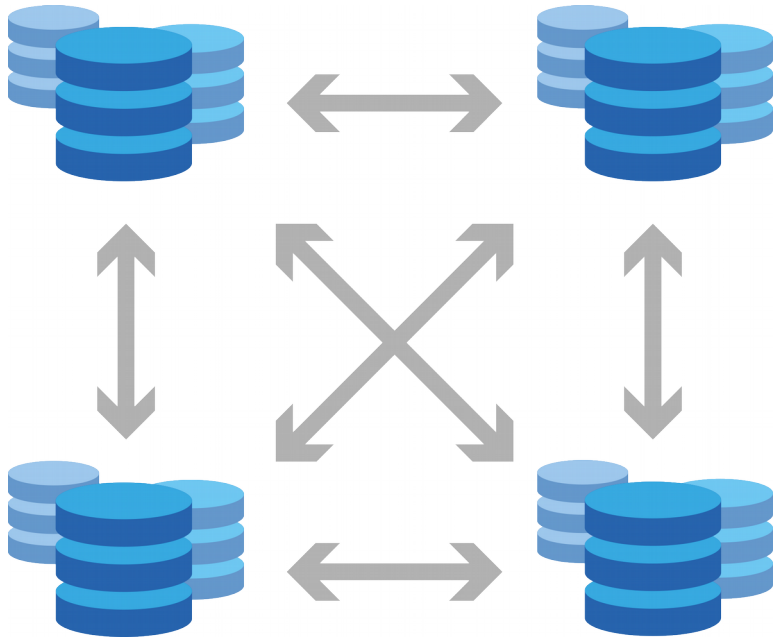
PostgreSQL-BDR Geo Cluster



PostgreSQL

mehr von BDR

- Multi-Master in mash
- BDR1 – 4 Jahre produktiv
- BDR3 – Architektur-Update
  - bessere Werkzeuge
  - andere Globale Seq.
  - Neue Dienstleistungen
- Muster-Architekturen VHA
- Viel in PG_LOGICAL ausgelagert



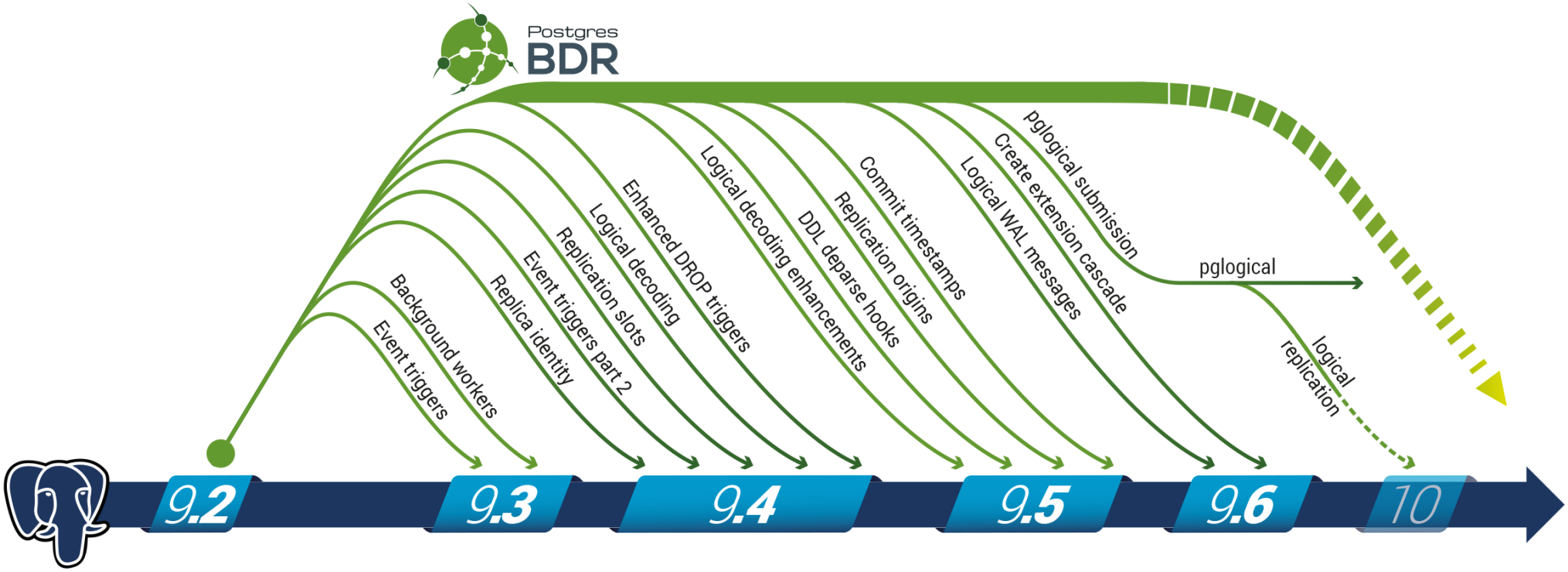
PostgreSQL

Woran wir arbeiten

- Konflikt-Behandlung per Spalte bei UPDATE
- Delta-Change UPDATEs
- CRDT “conflict free” replicated datatypes
  Konfliktfrei replizierte Datentypen
- Performanz
- Kaskadierende Replikation
- zusätzliche Konsistenz-Optionen

2ndQuadrant
PostgreSQL

BDR-Armadillo:
Infrastruktur für logische Replikation



Kontakt

2ndQuadrant Deutschland GmbH
Spielberger Str. 49
70435 Stuttgart

Harald Armin Massa
info@2ndquadrant.de
+49 711 400 557-00