

# MySQL Hochverfügbarkeitslösungen

Lenz Grimmer

⟨lenz@grimmer.com⟩

2009-05-04

amoocon 2009, Rostock, Germany

# Agenda

- Konzepte & Aspekte
- MySQL Replikation
- DRBD / Heartbeat
- MySQL Cluster
- Weitere HV-Applikationen

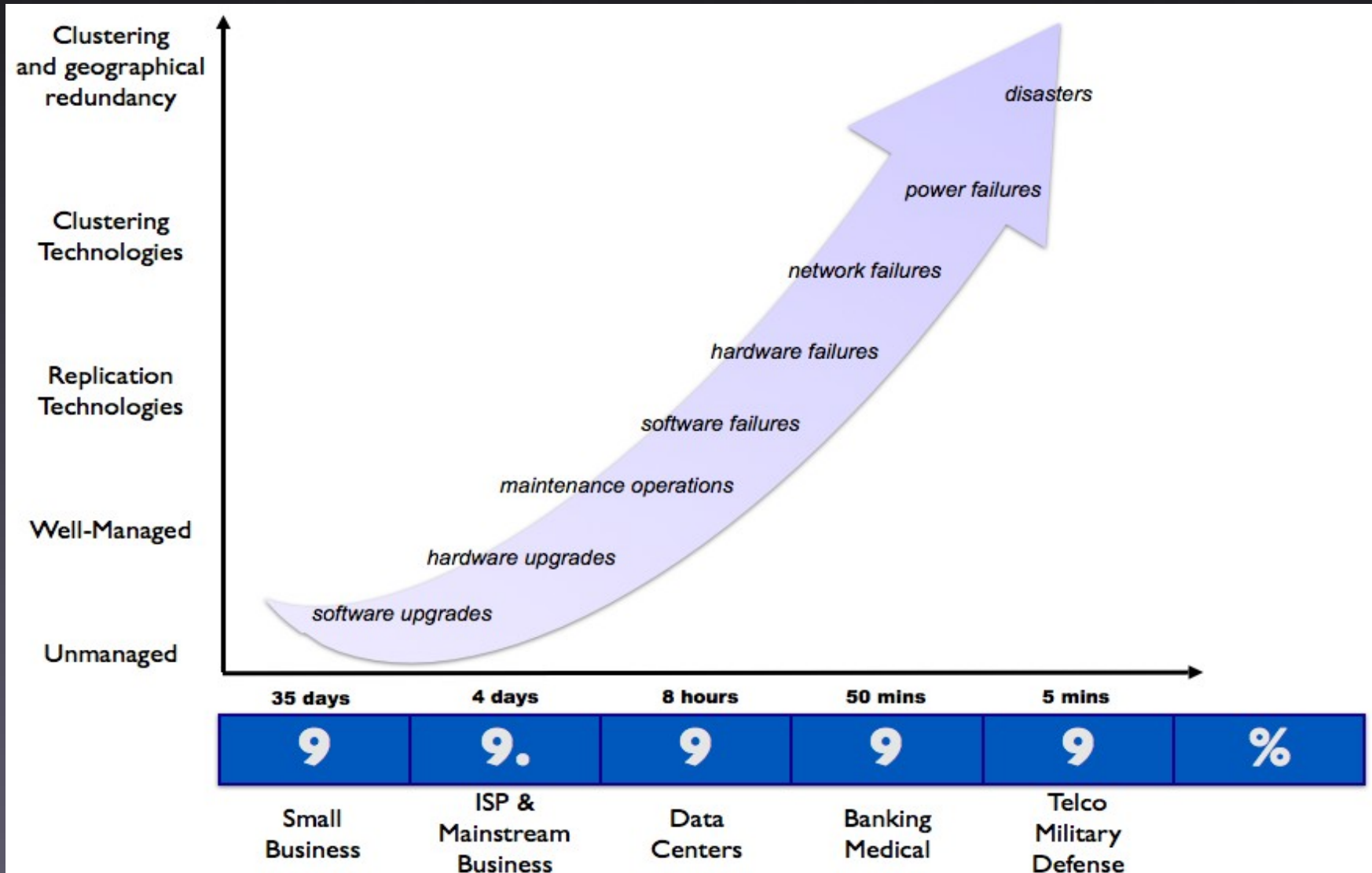
# Warum Hochverfügbarkeit?

- Irgendetwas kann und wird ausfallen
- Notwendigkeit von Wartungsarbeiten
- Ausfallzeiten sind teuer
  - Website-Besucher kommen nicht wieder
  - Sie verlieren €€€
  - Ihr Chef beschwert sich
- HV zu einem bereits vorhandenen System hinzuzufügen ist schwierig

# Was ist HV-Clustering?

- Redundanz
- Ein Dienst fällt aus → ein anderer übernimmt
- Übernahme der IP Adresse und des Dienstes
- Failover vs. Failback vs. Switchover
- **Nicht geeignet** für mehr Leistung
- **Nicht geeignet** für mehr Durchsatz (Lastverteilung)

# Hochverfügbarkeitslevel



# Eliminierung des SPOF

- Was ausfallen wird
  - Festplatten
  - Lüfter
- Was ausfallen kann
  - Absturz der Applikation
    - OOM-Situationen, Kernel-Panics
    - Netzwerkverbindungen/-kabel
    - Stromversorgungen

# HA Komponenten

- Heartbeat
  - Überwachung der hochverfügbar zu machenden Dienste
  - Kann einzelne Server, Dienste, Netzwerkverbindungen etc. prüfen
- HA Monitor
  - Konfiguration der Dienste/Applikationen
  - Regelt ordnungsgemäßen Start und Beenden
  - Erlaubt manuelle Übernahme
- Shared storage (SAN) / Replikation

# Replikation vs. SAN

- Konsistenz der replizierten Daten
- Synchrone vs. asynchrone Replikation
- Geteiltes Speichermedium kann zum SPOF werden
- “Split brain”-Situationen können zu Problemen führen (z.B. Gleichzeitiges Einhängen eines Dateisystems)
- SAN/NAS I/O Overhead

# Split-Brain

- Ausfall der Kommunikations → separate Cluster-Partitionen
- “Split-brain” Situation: mehr als eine Partition will die Kontrolle über ihren Teil des Clusters übernehmen
- <http://linux-ha.org/BadThingsWillHappen>
- Vermeidung mittels “Fencing“ oder Moderation/Arbitrierung

# Regeln zur Hochverfügbarkeit

- Auf Ausfälle vorbereitet sein
- Sicherstellen daß keine **wichtigen** Daten verloren gehen
- Keep it simple, stupid (KISS)
- Komplexität ist der Feind der Ausfallsicherheit
- **Automatisieren** was sinnvoll ist
- Häufig **Tests** durchführen!

# MySQL Replikation

- Unidirektional
- Anweisungs- oder zeilenbasiert (MySQL 5.1)
- Bestandteil des MySQL-Servers
- Einfach zu bedienen und einzurichten
- Ein Master, viele Slaves
- Asynchron – Slaves können nachlaufen

# MySQL Replikation (2)

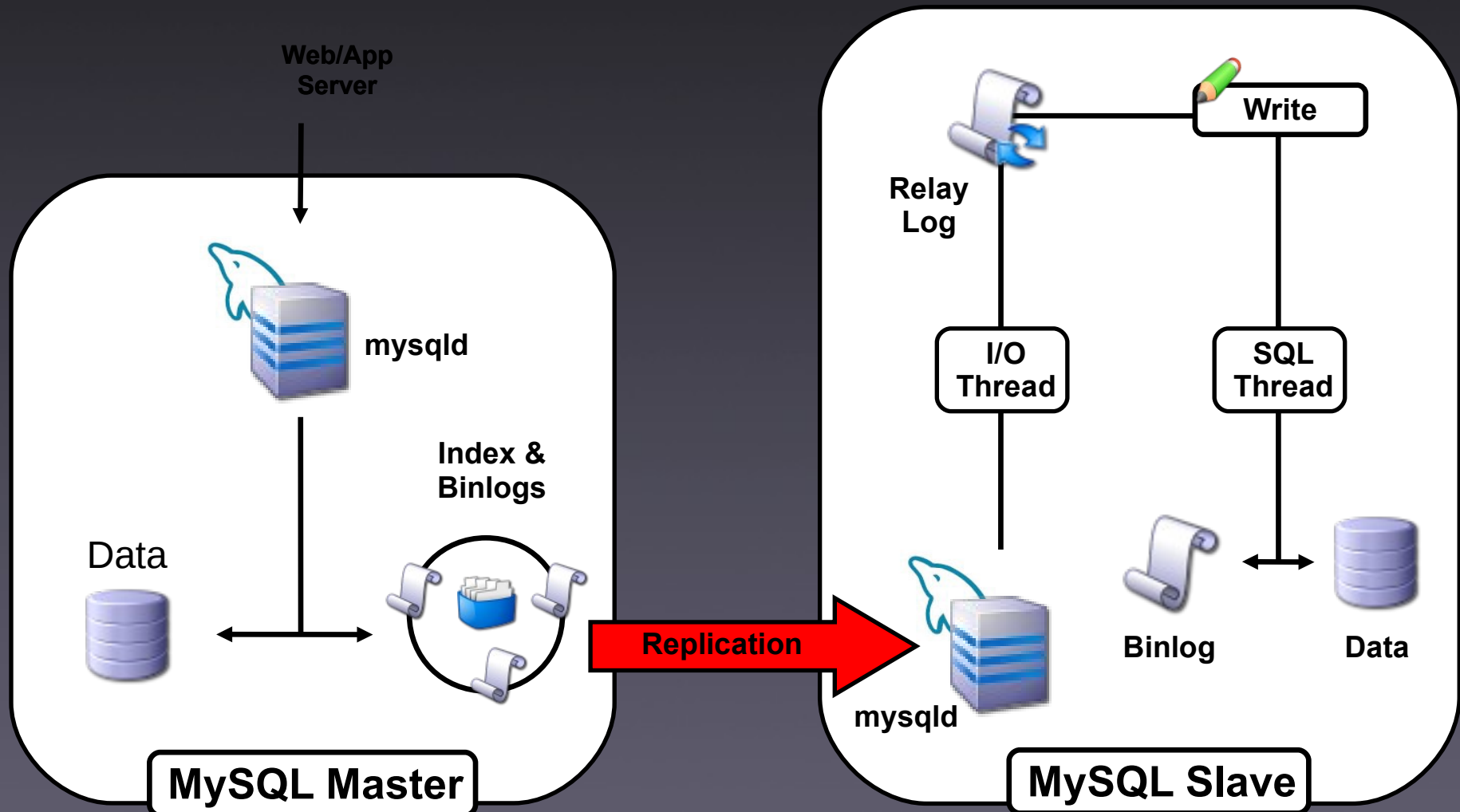
- Master verwaltet Binärlogs & index
- Replikation auf Slave ist (noch) single-threaded

<http://forge.mysql.com/wiki/ReplicationFeatures/ParallelSlave>

- Kein automatisiertes Fail-over
- Kein Heartbeat, kein Monitoring

# MySQL Replikation - Überblick

 **Read & Write** 



# Anweisungsbasierte Replikation

- Pro
  - Bewährt (seit MySQL 3.23 verfügbar)
  - Kleinere Logdateien
  - Auditing der tatsächlichen SQL-Anweisungen
  - Kein Primärschlüssel bei replizierten Tabellen erforderlich
- Kontra
  - Nicht-deterministische Funktionen und UDFs
  - LOAD\_FILE(), UUID(), USER(), FOUND\_ROWS() (RAND() and NOW() gehen)

# Zeilenbasierte Replikation

- Pro

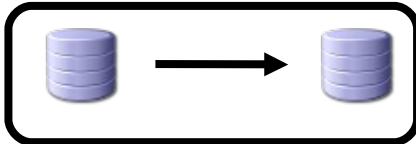
- Alle Veränderungen können repliziert werden
- Verfahren ähnlich zu anderen DBMSen
- Erfordert weniger Locks für bestimmte INSERT, UPDATE oder DELETE Anweisungen

- Kontra

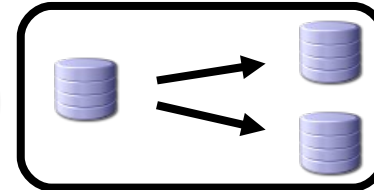
- Mehr Daten müssen gelogged werden
- Logfile-Größe (Auswirkungen auf Backup/Restore)
- Replizierte Tabellen benötigen expliziten Primärschlüssel
- Mögliche Ergebnis-Differenzen bei Bulk-INSERTs

# Replikationstopologien

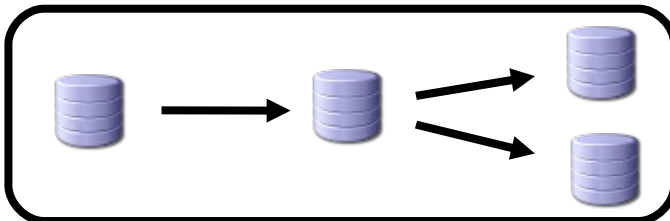
Master > Slave



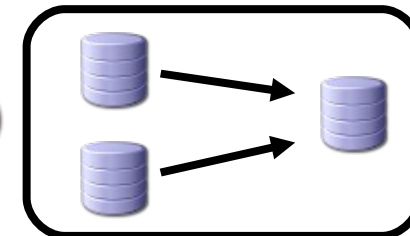
Master > Slaves



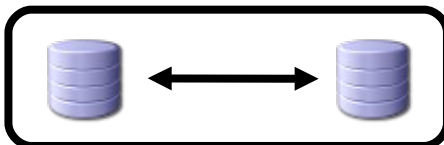
Master > Slave > Slaves



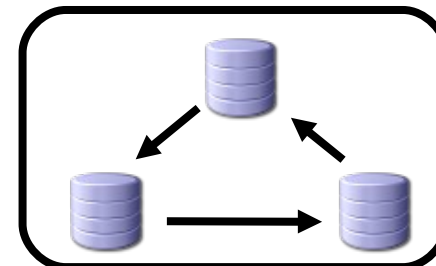
Masters > Slave (Multi-Source)



Master < > Master (Multi-Master)



Ring (Multi-Master)



# Master-Master-Replikation

- Zwei Server sind sowohl Master als auch Slave zueinander
- Vereinfacht Failover
- Nicht geeignet um Schreiblast zu verteilen
  - Änderungen erfolgen auf beiden Systemen
  - Keines von beiden hat volle Autorität
- **Nicht** auf beide Master schreiben!
- Sharding oder Partitionierung (z.B. MySQL Proxy) eignen sich besser

# MySQL Replikation als eine HV-Lösung

- Was passiert wenn der Master ausfällt?
  - Nicht viel
  - Applikation funktioniert nicht mehr
  - Slave hat nichts mehr zu replizieren
- Was passiert wenn der Slave ausfällt?
  - Nicht viel
  - Daten werden nicht mehr repliziert
- Das klingt aber nicht nach Hochverfügbarkeit?
  - Korrekt! Replikation kann **Bestandteil** einer HV-Lösung sein, implementiert jedoch nicht alle Teile einer HV-Konfiguration!

# Replikation & Hochverfügbarkeit

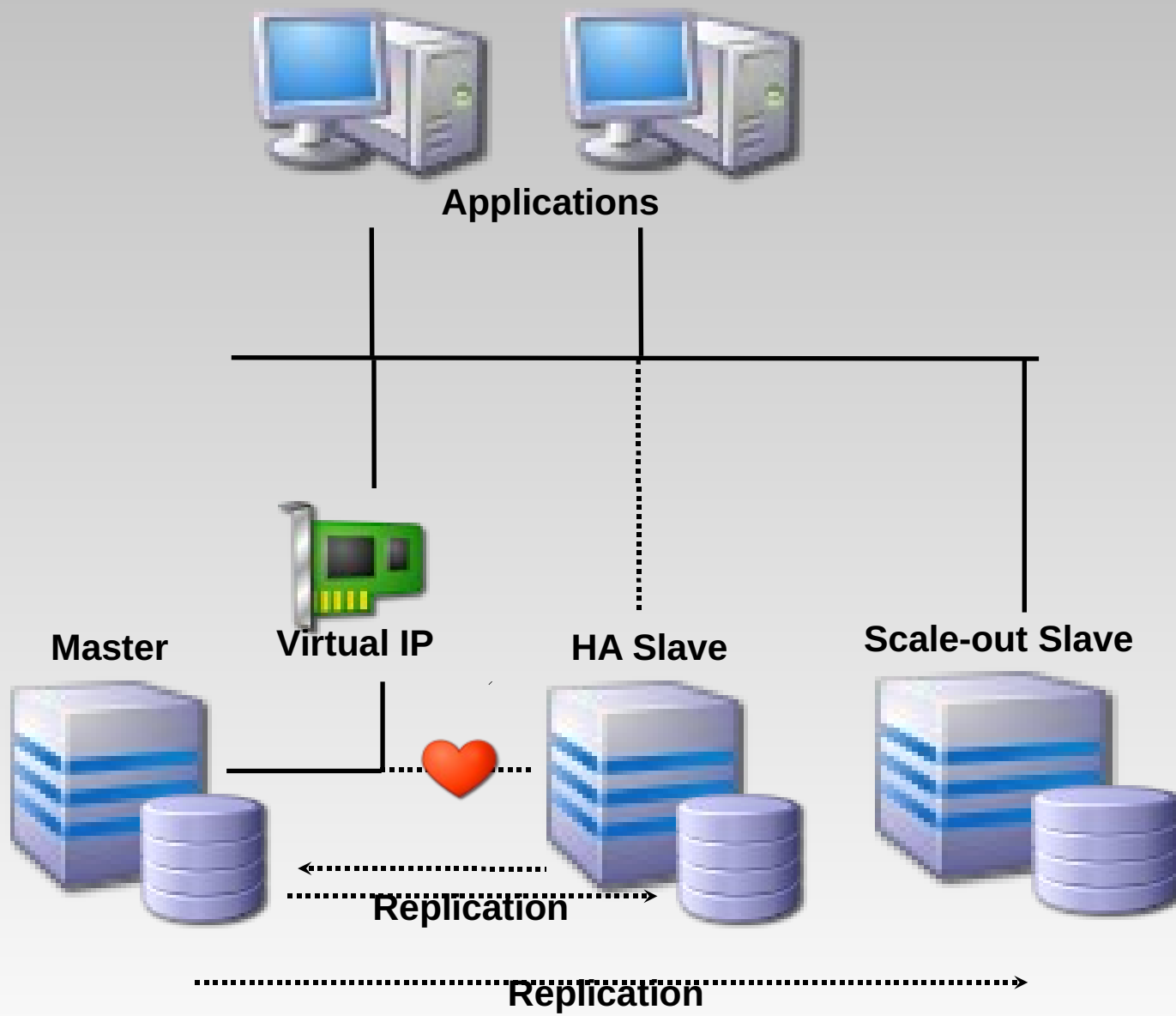
- Kombiniert mit Heartbeat
- Übernahme der virtuellen IP-Adresse
- Slave wird zum Master befördert
- Nebeneffekt: Lastverteilung & Backup
- Failback kompliziert
- Keine automatische Konfliktauflösung
- Korrektes Failover muß gescriptet werden

# Linux-HA / Heartbeat

- Unterstützt 2 oder mehr Knoten (v2)
- Wenig Abhängigkeiten/ Anforderungen
- Überwachung von Ressourcen
- Aktiver Fencing-Mechanismus: STONITH
- Erkennung eines Knotenausfalls in Sekundenbruchteilen
- <http://linux-ha.org/>

# Heartbeat (2)

- Richtlinien-basierte Ressourcenverwaltung, Abhängigkeiten & Einschränkungen
- Zeitbasierte Regeln
- Unterstützt sehr viele Applikationen (incl. MySQL)
- Grafische Oberfläche



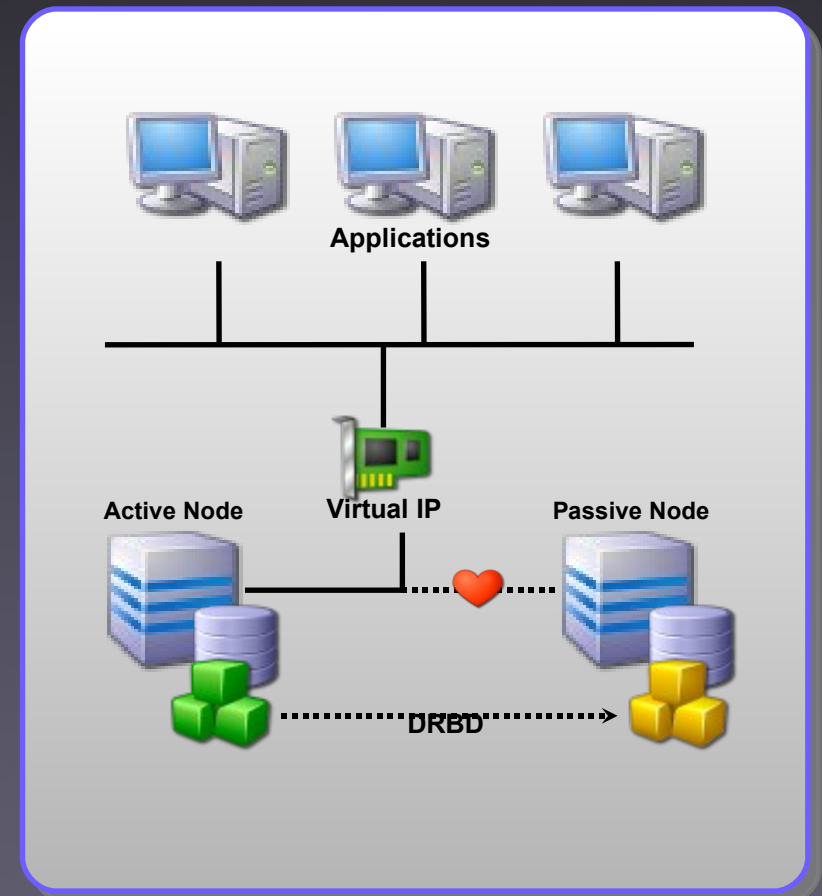
# DRBD

- Distributed Replicated Block Device
- “RAID-1 über das Netzwerk”
- Synchrone/asynchrone Block-Replizierung
- Automatische Resynchronisierung
- Applications-agnostisch
- Kann lokale I/O-Fehler maskieren
- Aktiv/passiv-Konfiguration vorgegeben
- Dual-primary Modus (benötigt ein Cluster Dateisystem wie GFS or OCFS2)
- <http://drbd.org/>



# DRBD im Detail

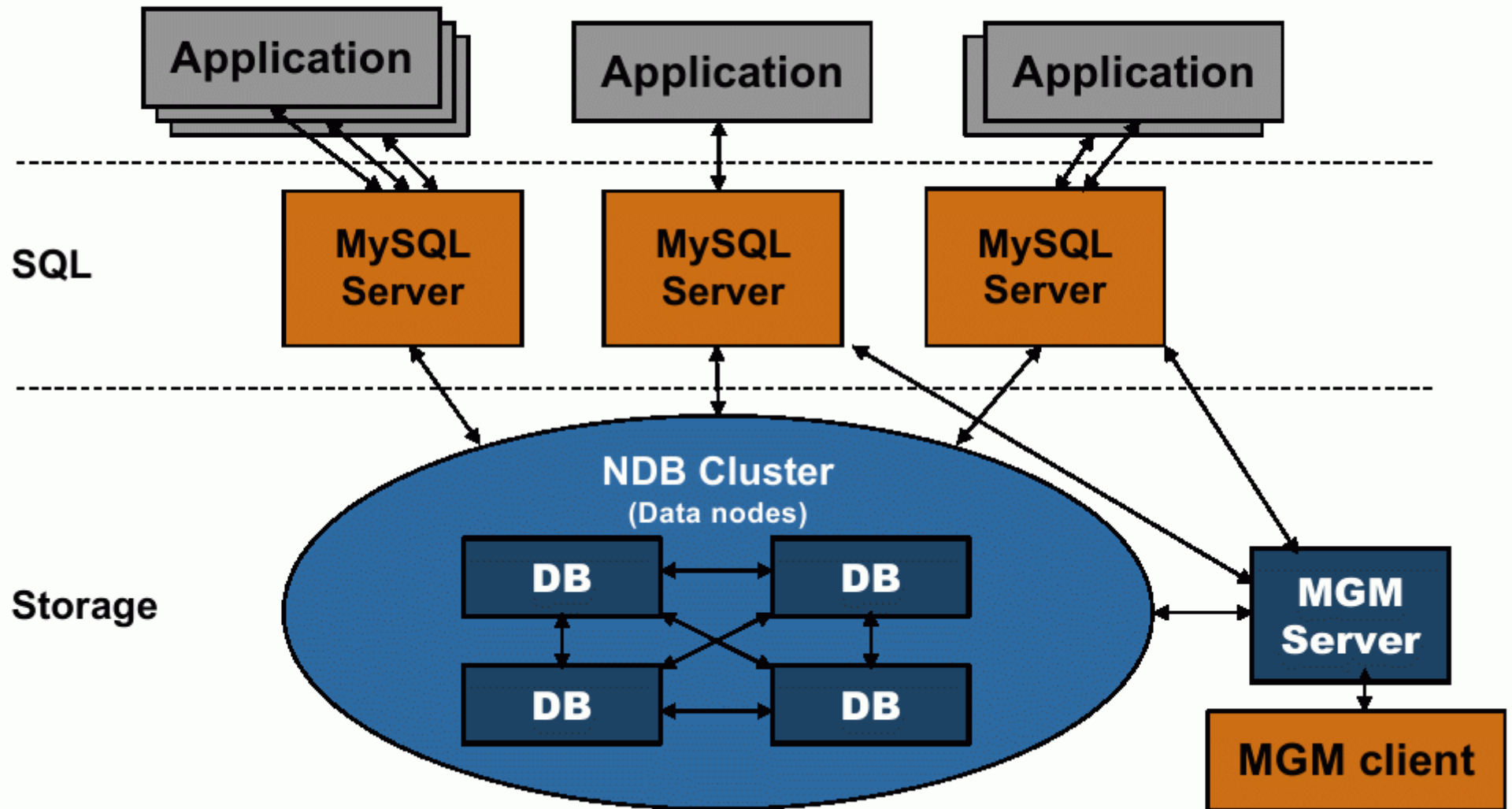
- DRBD repliziert Datenblöcke zwischen zwei Plattenpartitionen
- DRBD kann mit Linux-HA und anderen HV-Lösungen gekoppelt werden
- MySQL läuft normal auf dem Primärknoten
- MySQL ist nicht aktiv auf dem Sekundärknoten
- DRBD ist nur für Linux verfügbar



# MySQL Cluster

- “Shared-nothing”-Architektur
- Automatische Partitionierung
- Verteilte Fragmente
- Synchrone Replikation
- Schnelles, automatisches Fail-Over der Datenknoten
- Automatische Resynchronisierung
- Transparent für MySQL-Applikationen
- Unterstützt Transaktionen
- <http://mysql.com/products/database/cluster/>

# Cluster Components



# MySQL Master-Master Replication Manager

- Scriptsammlung zur Überwachung/Failover und Management
- Master-Master Replikationskonfiguration (ein aktiver beschreibbarer Master)
- Failover mittels Übernahme einer virtuellen IP-Adresse
- <http://code.google.com/p/mysql-master-master/>

# Flipper

- Perl Script zur Steuerung eines sich replizierenden Paares von MySQL-Servern (Master-Master)
- Automatisierter Switch-Over, manuell ausgelöst
- Kontrolliertes Herunterfahren einer Hälfte zu Wartungszwecken
- Die andere Hälfte übernimmt die Arbeit
- Verteilung von rollenbasierten IP Adressen ("nur-lesen", "beschreibbar") zwischen zwei Knoten im Master-Paar stellt sicher, daß beide Rollen verfügbar sind
- <http://provencal.com/software/flipper/>

# Red Hat Cluster Suite

- HV und Lastverteilung
- Unterstützt bis zu 128 Knoten
- Unterstützt Shared Storage
- Überwacht Dienste, Dateisysteme und Netzwerk-Status
- Fencing (STONITH) oder distributed lock manager
- Synchronisierung der Konfiguration
- [http://www.redhat.com/cluster\\_suite/](http://www.redhat.com/cluster_suite/)

# Solaris Cluster / OpenHA Cluster

- Provides failover and scalability services
- Solaris / OpenSolaris (Project Colorado)
- Kernel-level components plus userland
- Agents monitor applications
- Geo Edition to provide Disaster Recovery using Replication
- Open Source since June 2007
- <http://www.opensolaris.org/os/community/ha-clusters/>
- <http://opensolaris.org/os/project/ha-mysql/>

# Weitere Werkzeuge/Links

- **Maatkit**

<http://maatkit.sourceforge.net/>

- **Continuent Tungsten Replicator**

<https://community.continuent.com/community/tungsten-replicator>

- **Mon – scheduler and alert management**

<http://www.kernel.org/software/mon/>

- **Linux Cluster Information Center**

<http://www.lcic.org/ha.html>

Q & A

Fragen, Kommentare?

Vielen Dank!

Lenz Grimmer <lenz@grimmer.com>