MySQLによるHA・スケールアウトソリューション

松信 嘉範(MATSUNOBU Yoshinori)

MySQL株式会社

シニアコンサルタント

ymatsunobu@mysql.com

Agenda

- MySQL社の紹介
- HA・スケールアウト技術概要
- MySQLによるHA・スケールアウト ソリューションの解説
  - レプリケーション
  - HA構成
  - パーティショニング
  - MySQL Cluster

MySQL社の紹介

- 1995年に設立、日本法人(MySQL KK)は2006年2月に設立
- 社員数約400名、世界24カ国に在籍
- 海外での圧倒的な人気
  - 1,000万以上のインストールベース
  - 5万ダウンロード/日
- TCOの削減を実現
  - 浮いた費用を用いて品質を向上させるアプローチも可能
- 豊富な周辺ツール
- 多数のパートナー企業
  - 住商情報システム様、HP様、NTTコムウェア様、スマートスタイル様、

MySQLを利用している企業

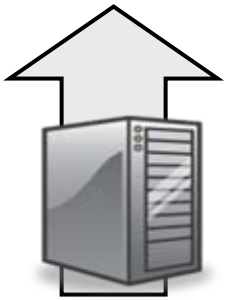
- **High Volume Websites**
  - Web 2.0
  - Dynamic content
  - eCommerce
  - ”Look to Book”
  - Session Management
  - Gaming & entertainment
  - Scale Out
- **Enterprise**
  - Data Warehousing
  - High-Volume OLTP
  - Scale Out
- **Embedded**
  - Bundled in software applications & hardware components

中・大規模アプリケーションに求められるもの

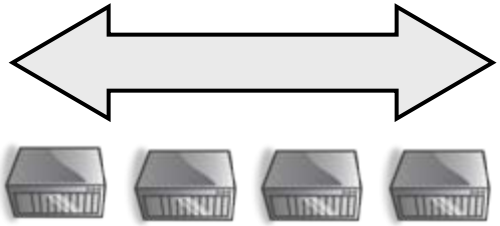
- 可用性の向上
  - MTBF(平均故障間隔) / (MTTR(平均修復間隔) + MTBF) の向上
    - 停止につながる障害を減らすこと
    - 万一障害が発生した場合は回復時間を短く済ませること
  - 冗長化構成が基本
- 負荷分散
  - 大量のリクエストを一定時間以内に処理する
  - 垂直型(スケールアップ)と水平型(スケールアウト)

スケールアップ 対 スケールアウト

- *スケールアップ*
  - 垂直型
  - CPU、メモリ、ディスクの増強によって
    パフォーマンスを向上
  - 価格が高くなる傾向にある



- *スケールアウト*
  - 水平型
  - 安価なハードウェアの冗長構成
  - マシンの増加によって
    パフォーマンス向上
  - アプリケーション側での対処も必要
  - 電源やスペースの費用



- 価格と性能のバランス
  - 1GB Memory×16台 vs 16GB Memory×1台

HA構成の種類

- Active/Passive(Standby)型
  - サービスを提供するDBサーバは1個(Active機)
  - Passive機はActive機を監視
  - Active機の停止を検知したら、Passive機はサービスを起動・再開
  - 負荷分散にはならない
  - fsckやクラッシュリカバリの時間は停止時間になる
  - 起動後はしばらくキャッシュにデータが無い状態が続く
  - サービスレベルとしては、Four Nines (99.99%)に相当
  - Heartbeat + SAN/DRBD + MySQL
  - Active/Passive + Passive/Activeの構成もある

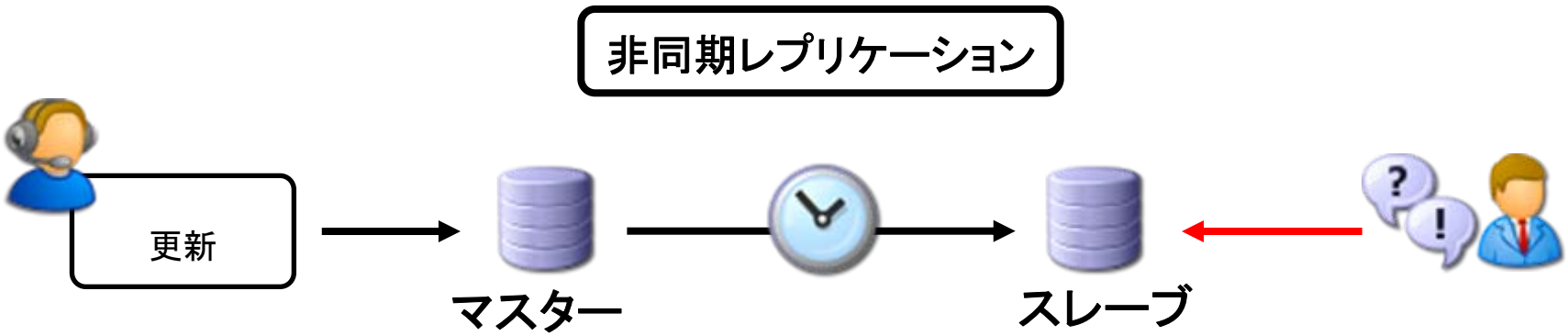
- Active/Active型
  - 同時に複数のDBサーバがサービスを提供
  - 停止検知の仕組みはActive/Passive型と同等だが、サービスはすぐ提供できる
  - サービスレベルとしては、Five Nines (99.999%)に相当
  - MySQL Cluster

MySQLによるHA・スケールアウト ソリューションの紹介

- MySQL レプリケーション
  - MySQLの標準機能だけで実現可能
  - 参照処理はスケールアウト可能
- サードベンダーのクラスタソフト等との組み合わせ
  - Heartbeat + MySQLレプリケーション
  - L/B + MySQL レプリケーション
  - Heartbeat + SAN + MySQL
  - Heartbeat + DRBD + MySQL
- パーティショニング
- MySQL Cluster
  - MySQLの機能だけで実現可能
  - HA、スケールアウト両方を実現
  - 欧米での導入実績が豊富
  - 5.1で大幅な機能強化

レプリケーションとは

データベースの複製を別ノード上に作成する技術



レプリケーションの分類

- 片方向レプリケーションか、双方向レプリケーションか
- シングルマスターか、マルチマスターか
- 非同期レプリケーションか、同期レプリケーションか
- 物理レプリケーションか、論理レプリケーションか
- Active/Passive か Active/Active か

MySQL レプリケーションの特徴

特徴

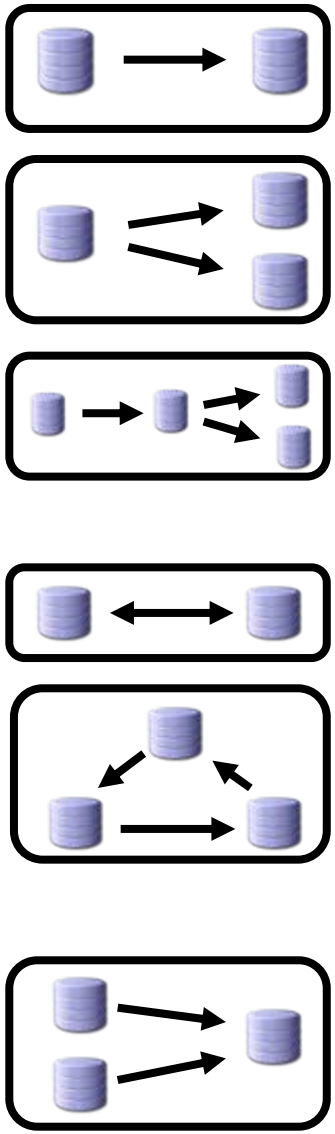
- マスター-スレーブ型
- 非同期レプリケーション
- マスターからSQL文をスレーブに転送して実行 (5.1からは行コピー型と選択/共存可能)

長所

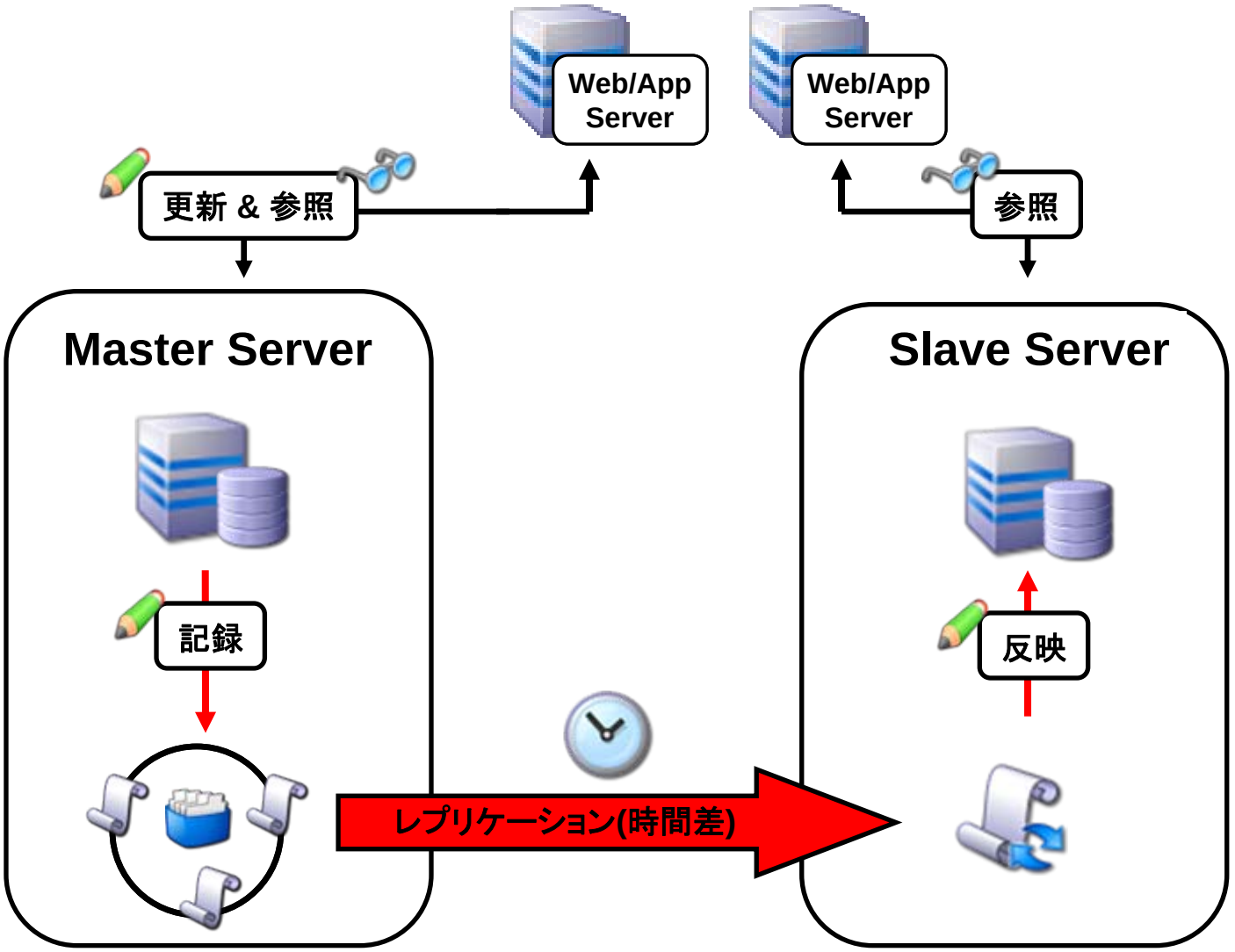
- 追加のソフトウェアが不要
- 参照処理をスケールアウト可能
- 仕組みが単純で、アーキテクチャ選択の幅が広い

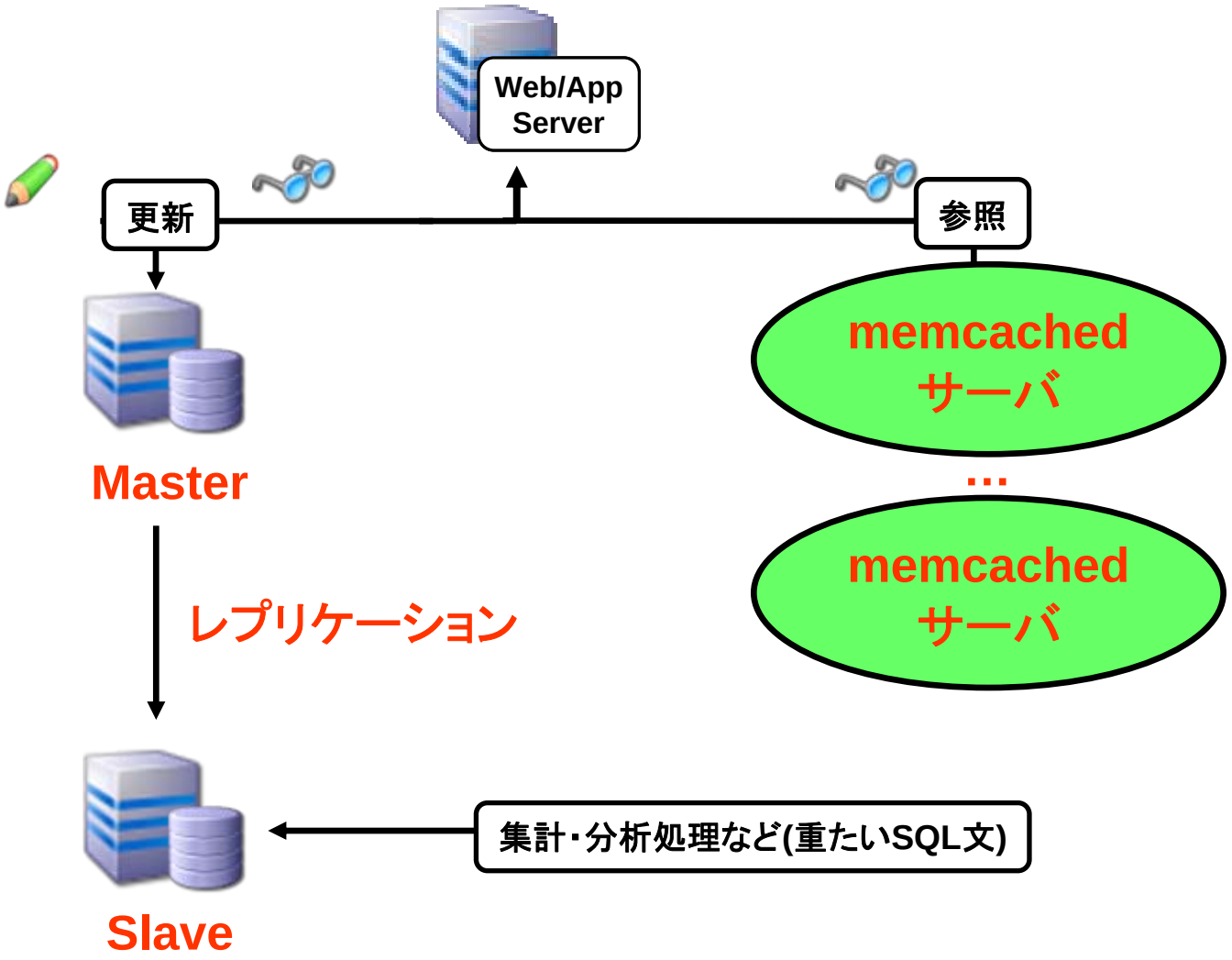
制限事項

- 1個のスレーブが持てるマスターは1個だけ
- 障害時に、スレーブ側への反映が未完了の場合がある
- フェイルオーバーのための処理が手動

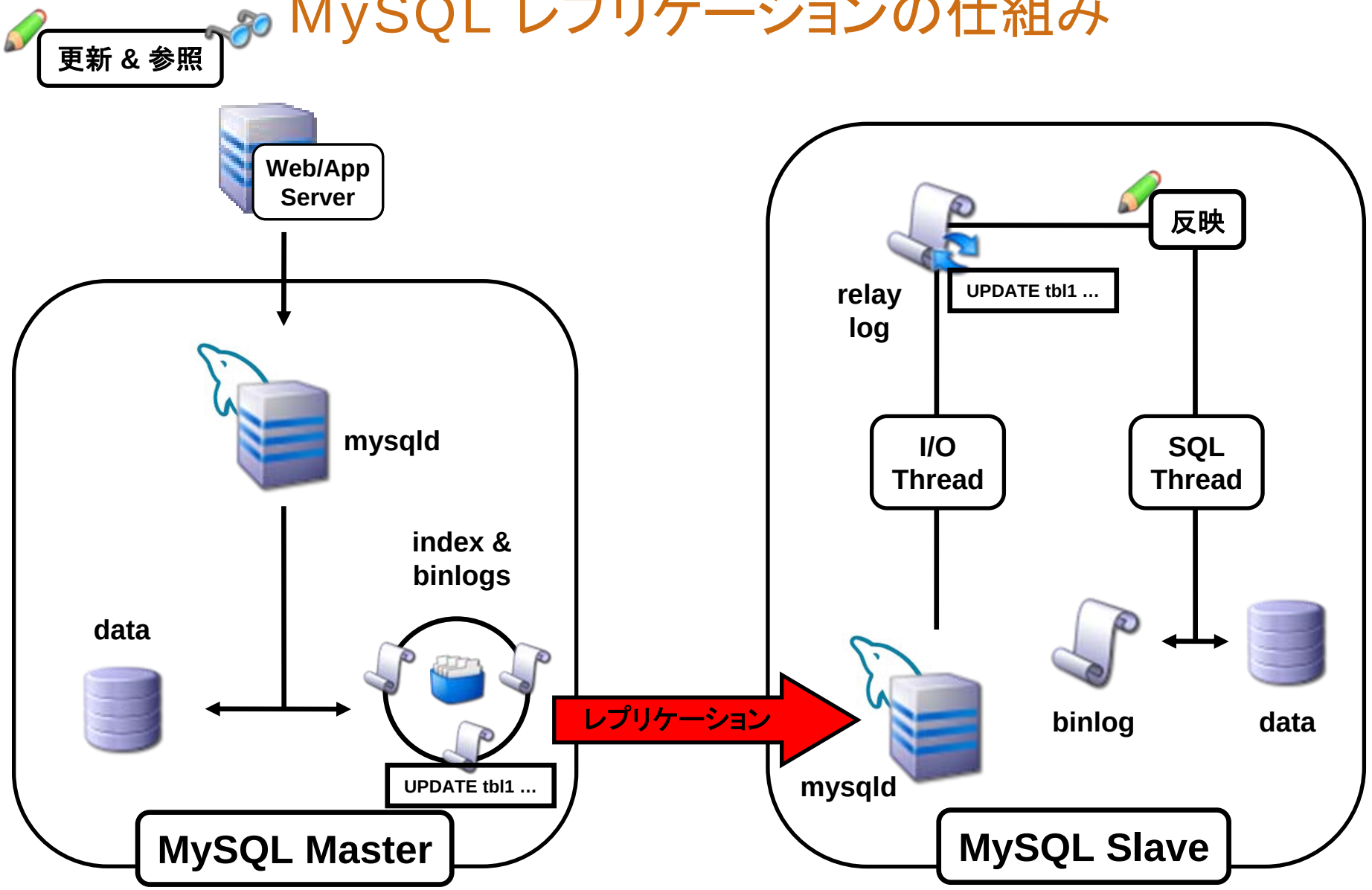


MySQL レプリケーションの流れ

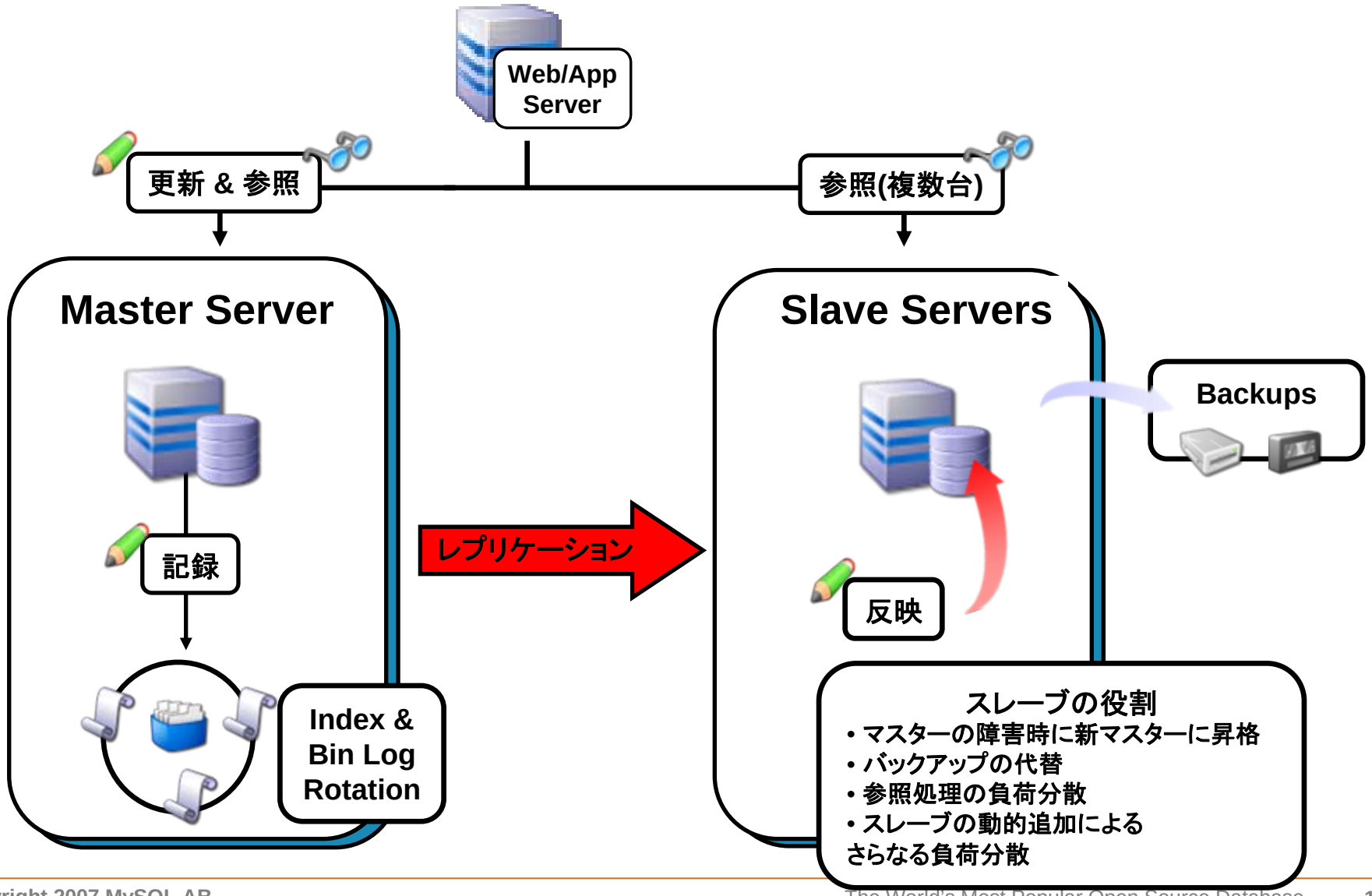



Web/App
Server
更新
参照
memcached
サーバ
...
memcached
サーバ
Master
レプリケーション
集計・分析処理など(重たいSQL文)
Slave

MySQL レプリケーションの仕組み



参照処理のスケールアウト



レプリケーションの手順

- マスター
  - 全体バックアップ
    - InnoDBオンリーであれば
      $ mysqldump --user=root --password=xx -socket=xx --single-transaction --master-data=2 --flush-logs --hex-blob --default-character-set=cp932 --all-databases > xx.sql

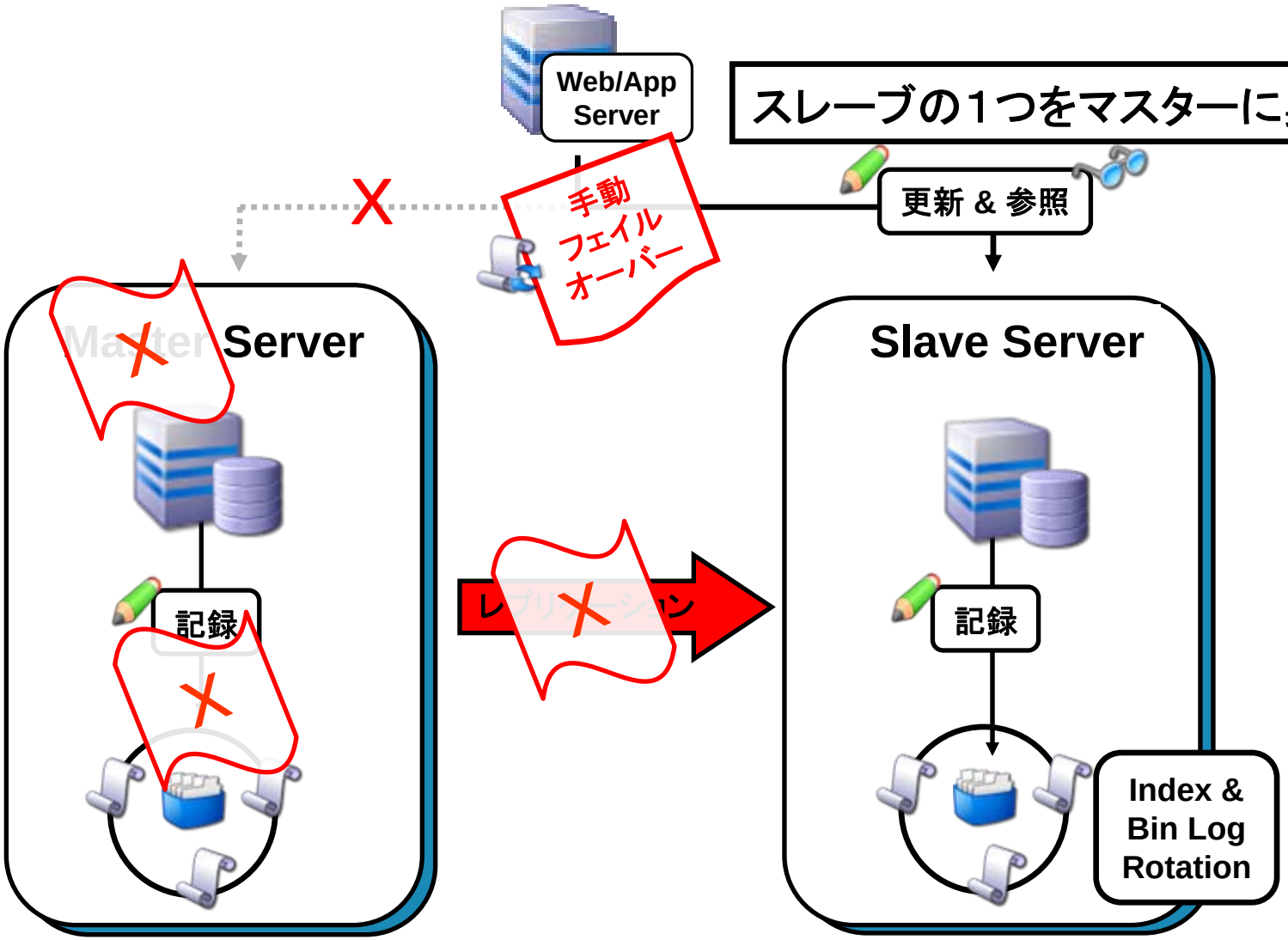
- スレーブ
  - 全体バックアップのコピー・リストア
  - mysqldを起動
    - リードオンリーの指定: read_only
    - サーバーID を一意に指定: server-id=N
  - マスターへの接続

```
mysql> CHANGE MASTER TO MASTER_HOST='hostname',
    -> MASTER_PORT=port, MASTER_USER='repl_user',
    -> MASTER_PASSWORD='repl_pass',
    -> MASTER_LOG_FILE='binlog_file_path';
```

  - レプリケーションの開始

```
mysql> START SLAVE;
```

マスターのフェイルオーバー



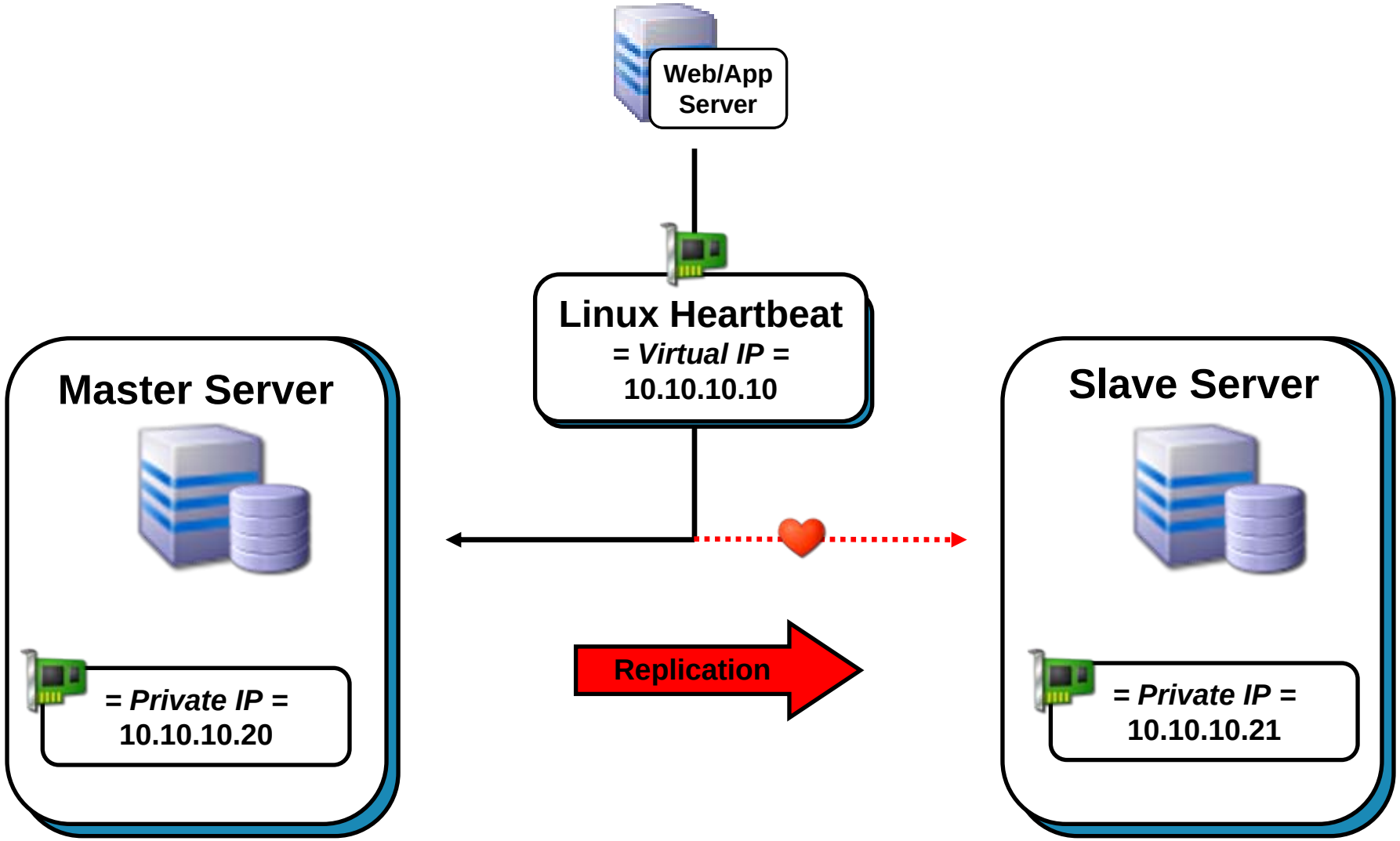
マスターのフェイルオーバーの難しさ

- 障害を検知しフェイルオーバーする仕組みを標準では持っていない
  - クラスタリングソフトを使う、手動で作りこむ
  - アプリケーションの接続先も変えないといけない
- マスターの全更新情報(バイナリログ)のスレーブへの転送が完了していない可能性がある
  - 非同期レプリケーションの宿命
  - 未転送のものを捨てるか、手動で転送・適用するか
  - 捨てる場合、フェイルバックが難しい

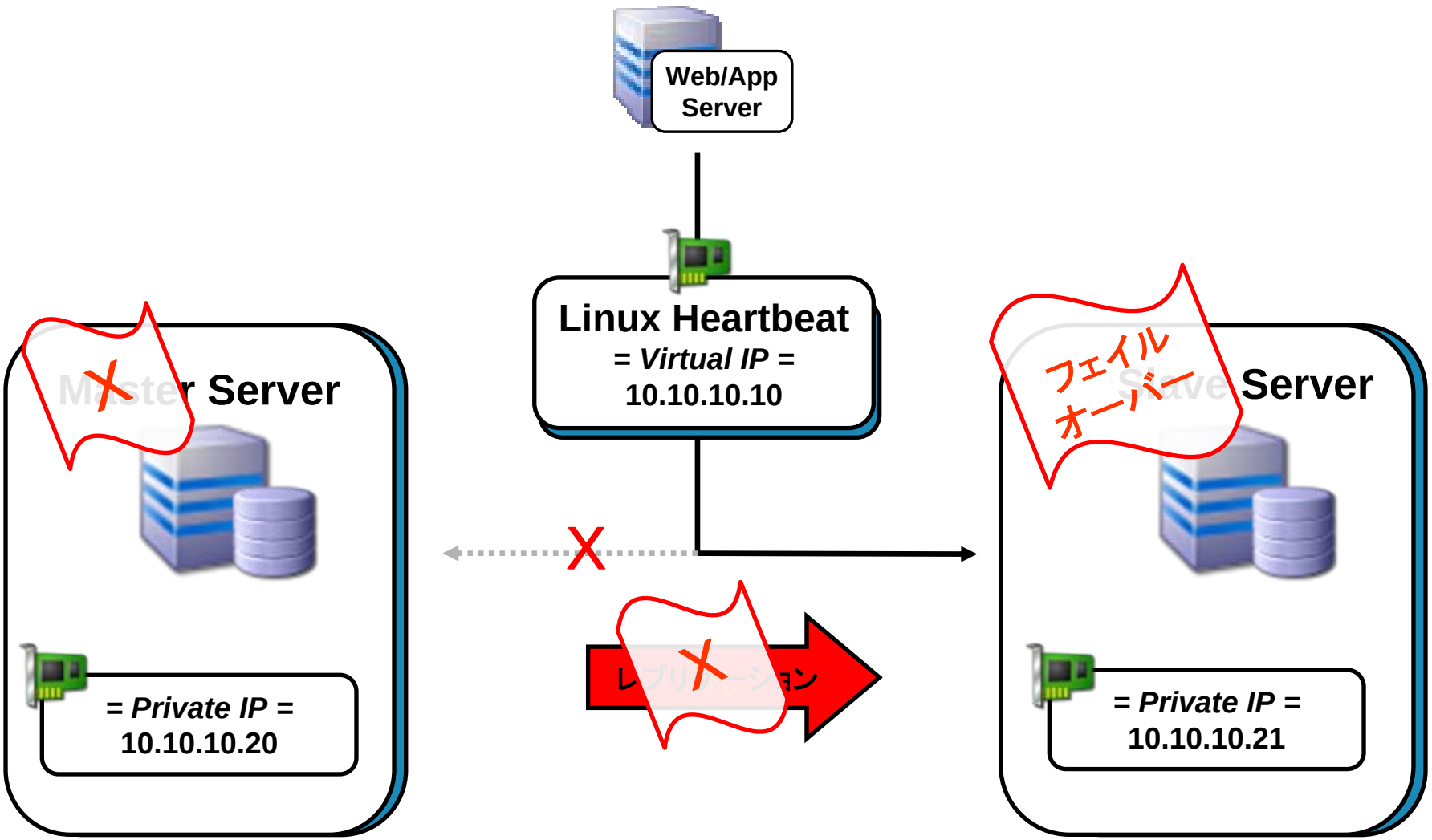
Heartbeat + MySQLレプリケーション

- クラスタソフト(Heartbeat)の使用
  - Heart-Beat プロトコル
    - ノード間で生存確認メッセージの送受信
    - ハートビート送受信の失敗がフェイルオーバー開始の合図
    - シリアルケーブルによるハートビート送受信が可能
  - Virtual IPの管理
    - IPアドレスの引継ぎ
    - 管理は自動
  - オープンソース
- 自動フェイルオーバーが可能
- レプリケーションは非同期

Heartbeat + MySQLレプリケーション



Heartbeat + MySQLレプリケーション(フェイルオーバー時)



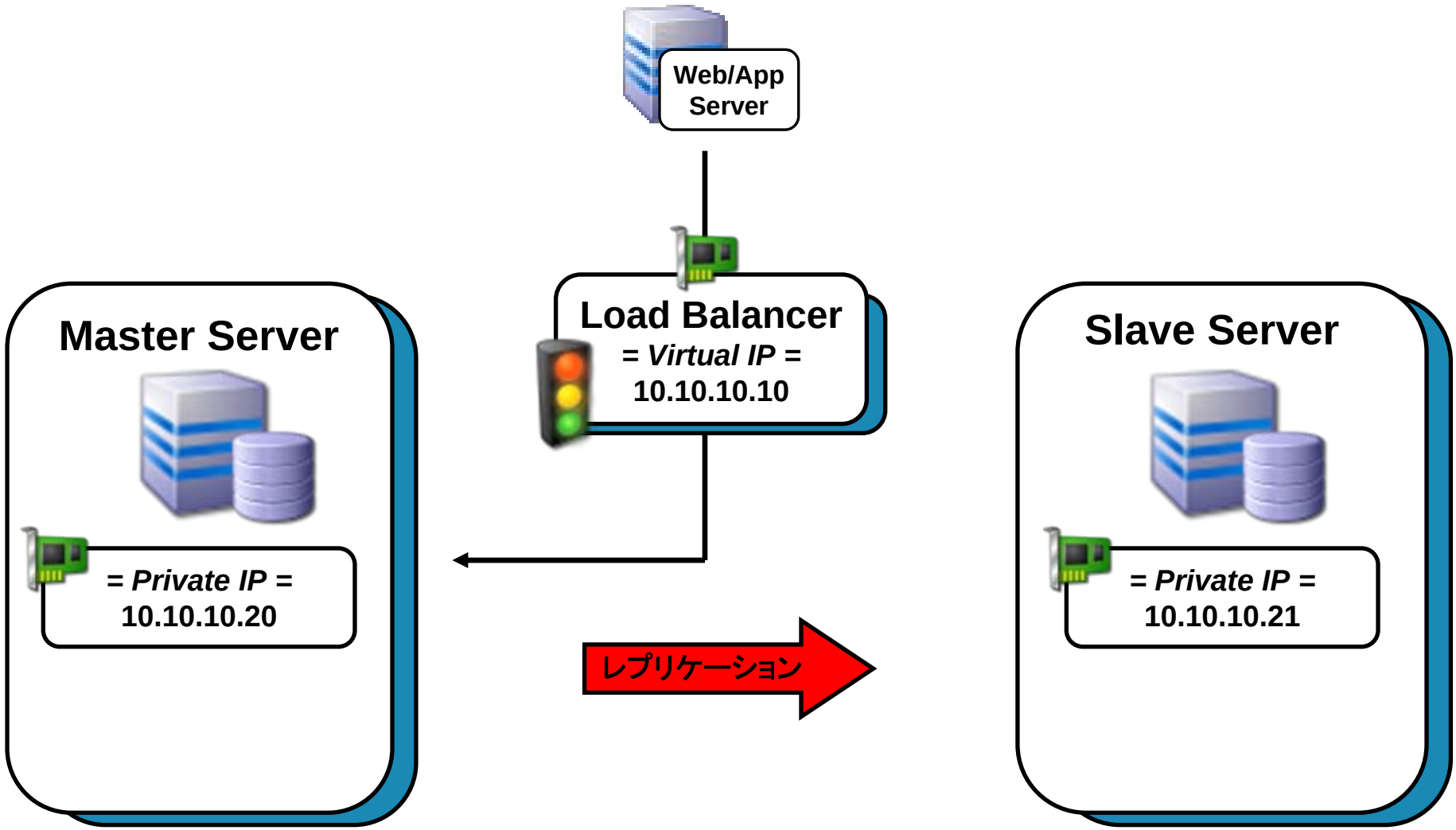
Load Balancer + MySQLレプリケーション

- HAとスケールアウトの両方の目的で活用可能

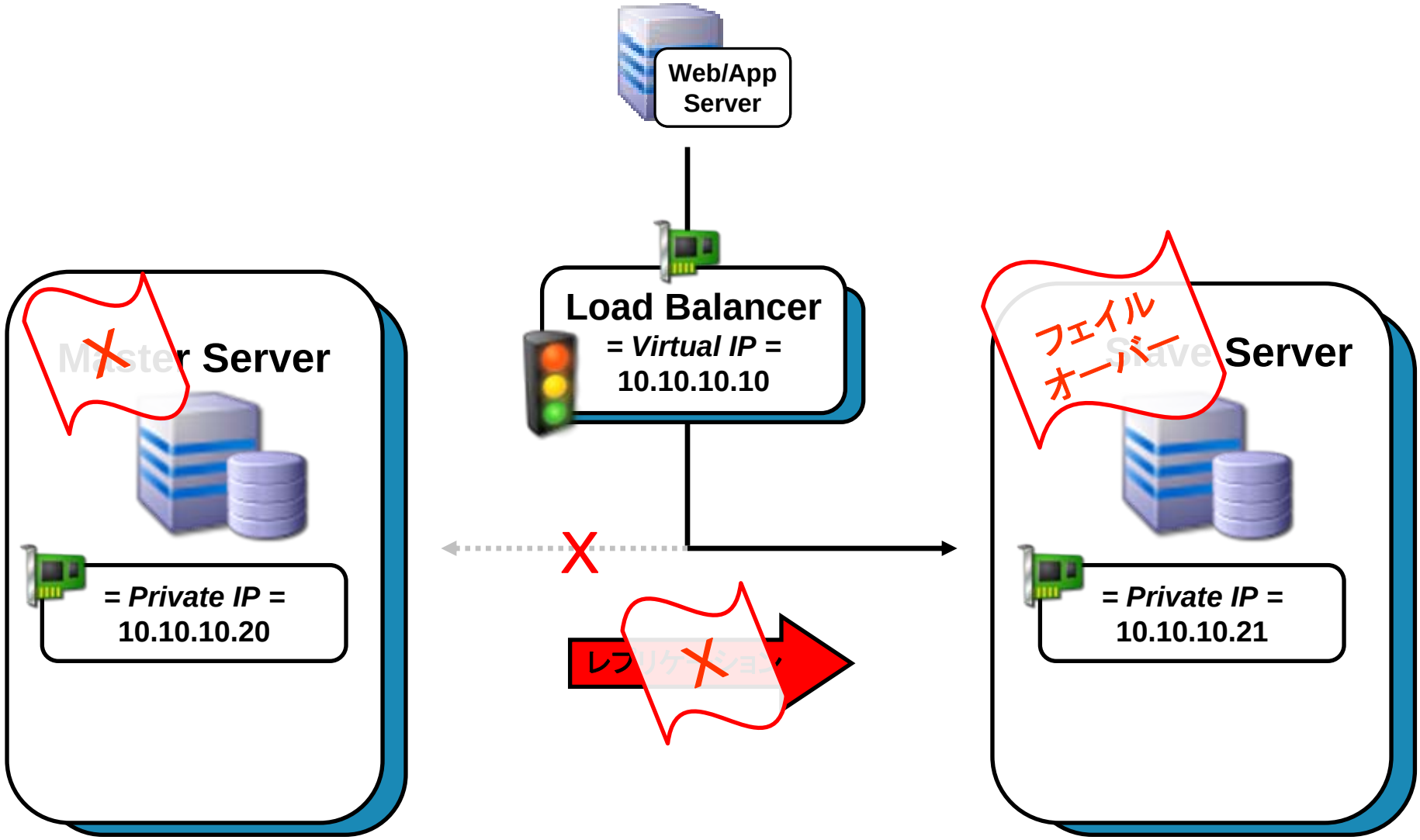
- スケールアウト目的
  - 複数台のスレーブの前に置いて負荷分散

- HA目的
  - 「クラスタソフト + MySQLレプリケーション」構成と類似
    - ハードウェアベースでのVirtual IP管理(ソフトウェアL/Bもある)
  - 特徴
    - 100:0の振り分けルール
    - 自動フェイルオーバー
    - 非同期レプリケーション

L/B + MySQLレプリケーション(HA目的)



L/B + MySQLレプリケーション(HA目的)



レプリケーションとフェイルオーバーの相性

- マスター→スレーブのフェイルオーバー
- スレーブの状態はマスターと同一になっていない可能性がある
- 未完了状態のスレーブをマスターに昇格すると…
- 最新の状態が失われる
- マルチマスター構成でも同様の問題がある

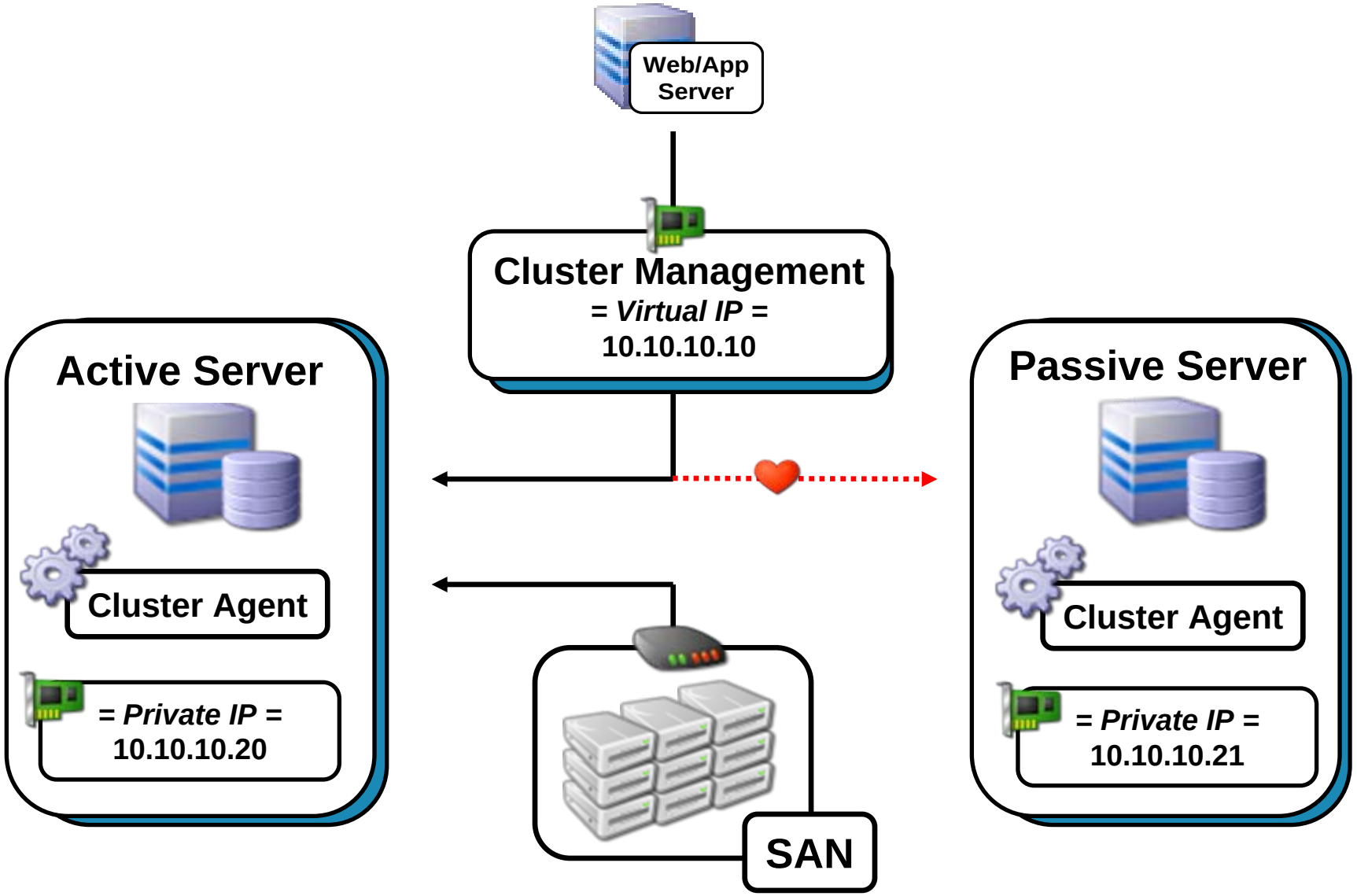
- 考慮事項が多く、難しい

- マスター→スレーブのフェイルオーバーではなく、Active/Standby構成のマスターのフェイルオーバー

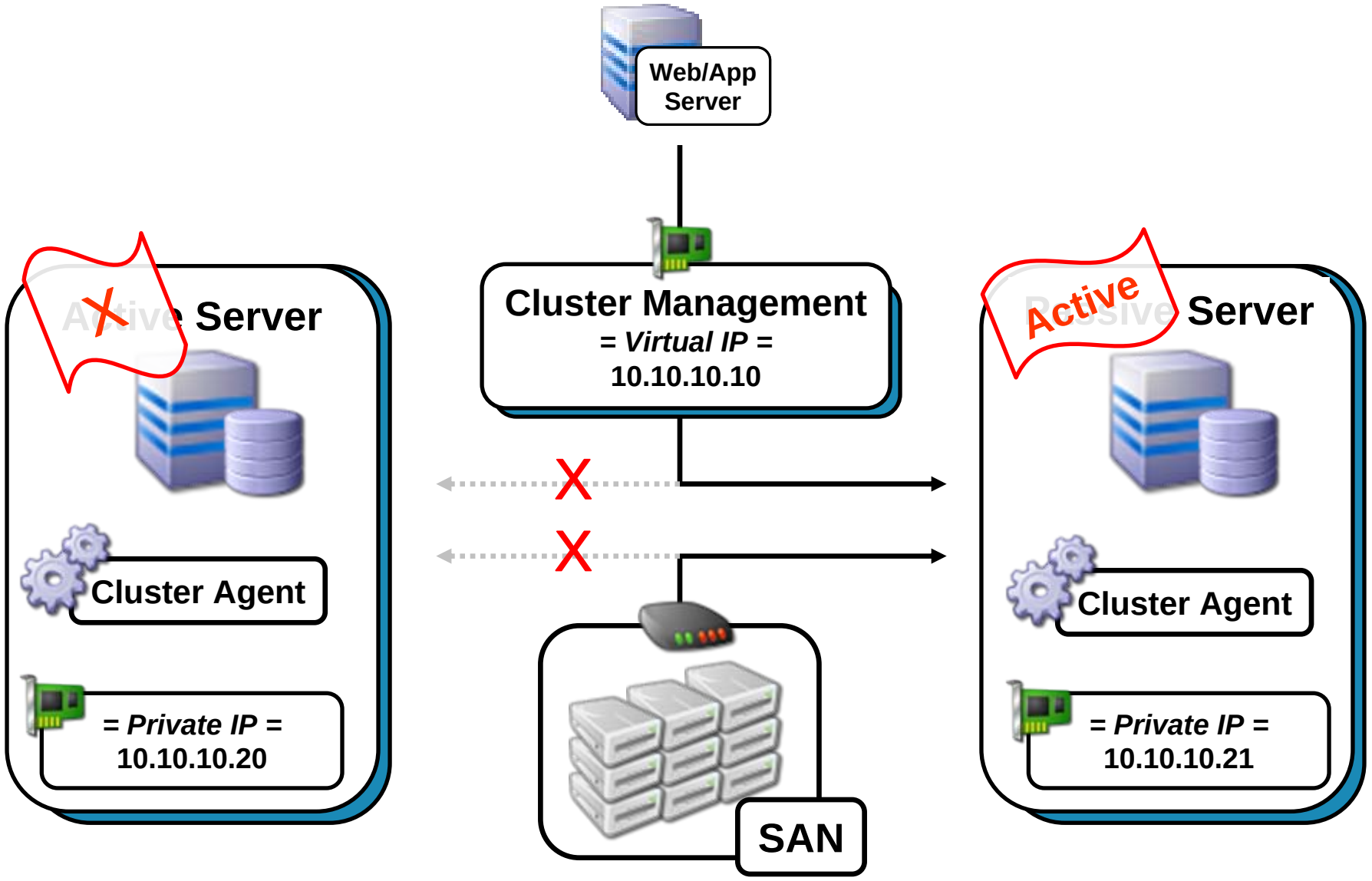
Heartbeat + SAN + MySQL

- Active/Passive型クラスタ構成
  - 同時に複数のmysqldが同じファイルにアクセスすることは不可
  - 負荷分散にならない
- 自動フェイルオーバー
  - Virtual IP
  - ファイルシステムのマウント
  - mysqldの起動
- データが1箇所なので、不整合の問題が発生しない
  - フェイルバックもより簡単になる
- その他の特徴
  - 初期設定はより複雑になる
  - ハードウェア費用が高価
  - クラッシュリカバリに関わるリカバリ時間の考慮が必要
  - フェイルオーバー直後はキャッシュが無いためパフォーマンスに注意
  - 多くのSANベンダーによる実績がある

Heartbeat + SAN + MySQL



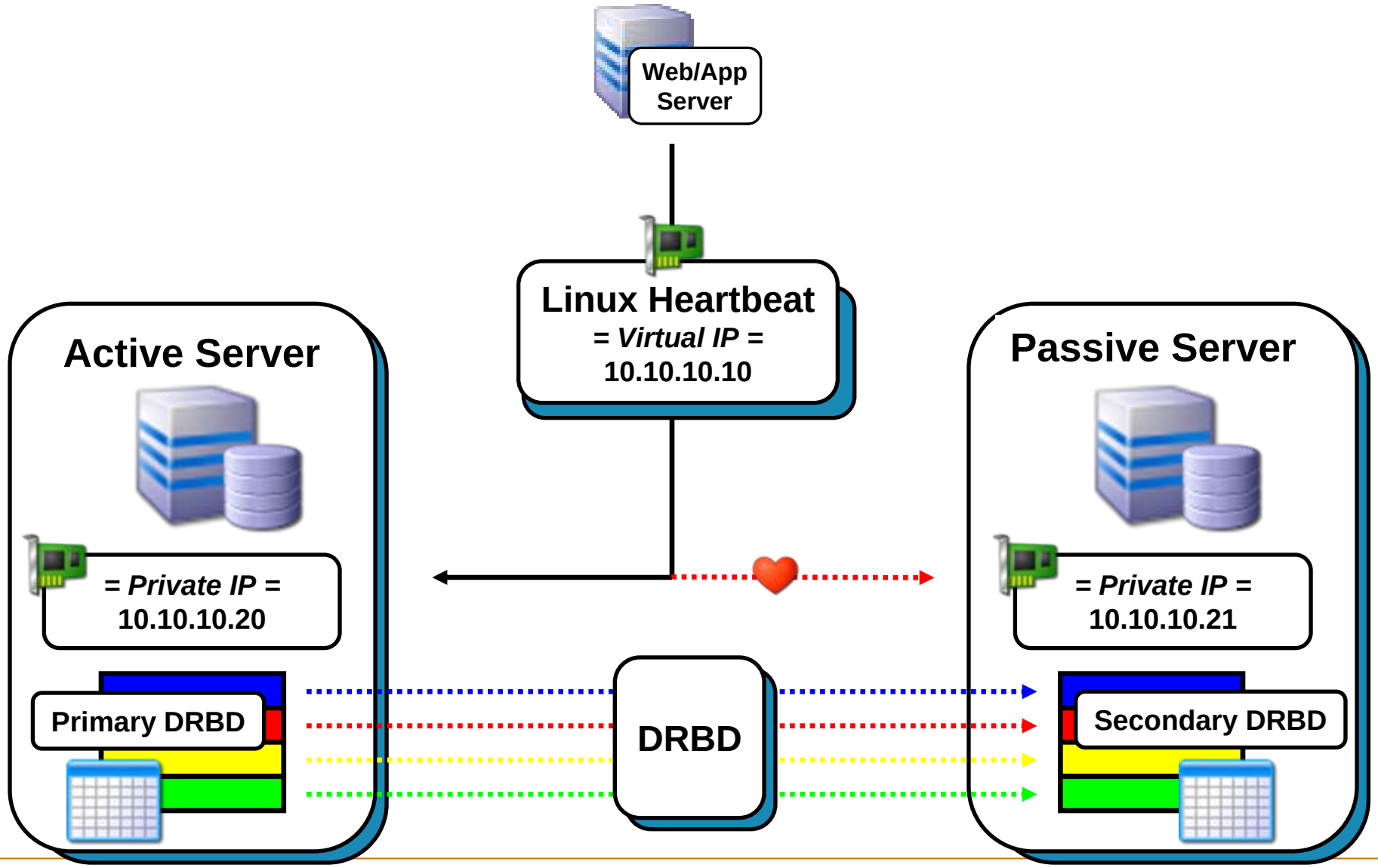
Heartbeat + SAN + MySQL



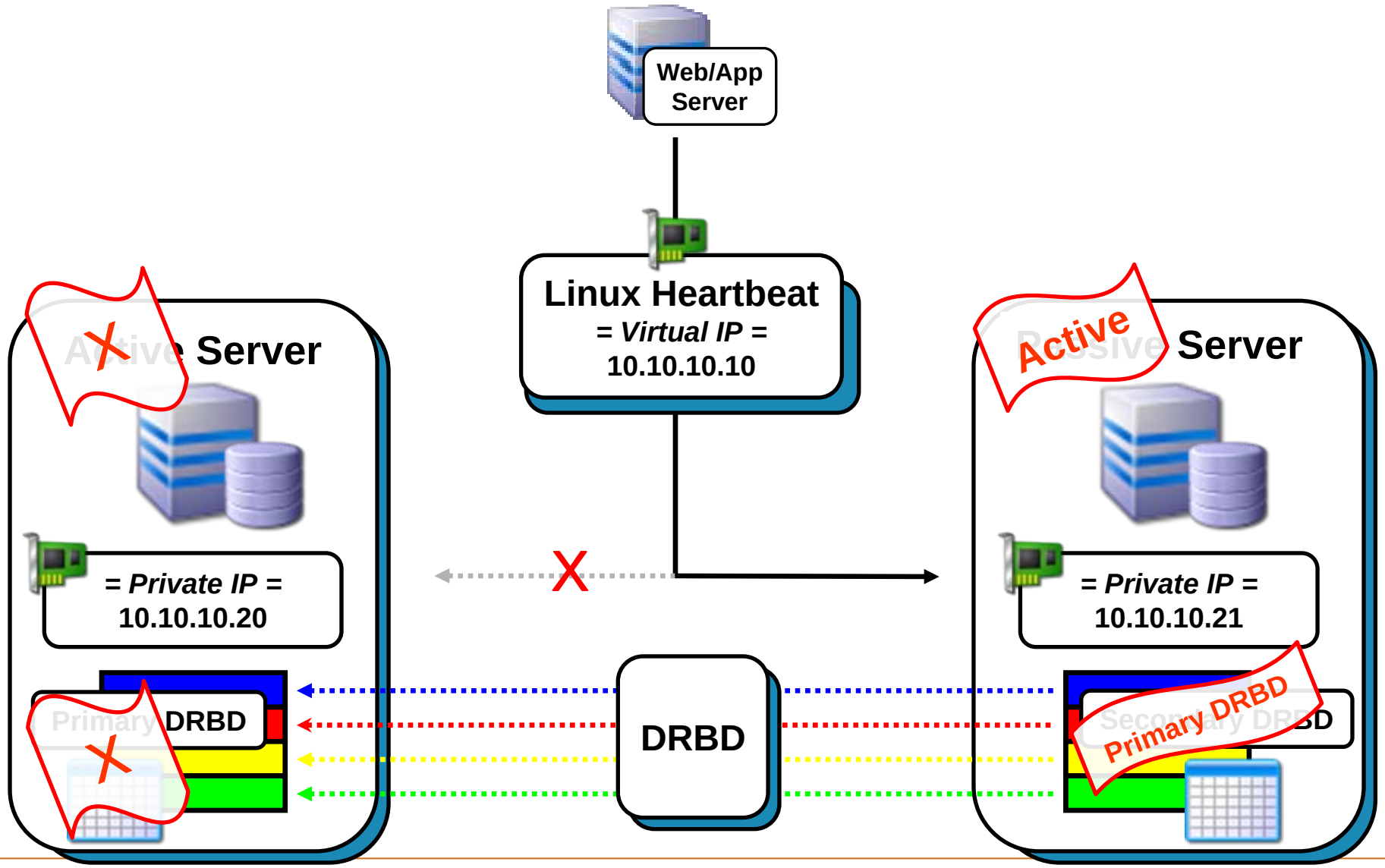
Heartbeat + DRBD + MySQL

- ディスクミラーソフト (DRBD)
  - http://www.drbd.org
  - ディスクをマシンごとに分散配置し、
    片方向物理レプリケーション (ネットワーク RAID1)
  - 特別なハードウェアが不要(通常のIPネットワーク上で動作)
  - 同期レプリケーション
  - オープンソース、MySQL Enterpriseではオプションでサポート
  - レプリケーション性能は良い(SQL文のレイヤーを回避)
- Active/Passive型クラスタ構成と同等で、
  負荷分散にはならない
  - ミラーされている側は、ファイルシステムのマウントすらできない
- 自動フェイルオーバーが可能
- 同期レプリケーションなので不整合の問題が発生しない
  - フェイルバックもより簡単になる

Heartbeat + DRBD + MySQL



Heartbeat + DRBD + MySQL



Active/Passive + Passive/Active

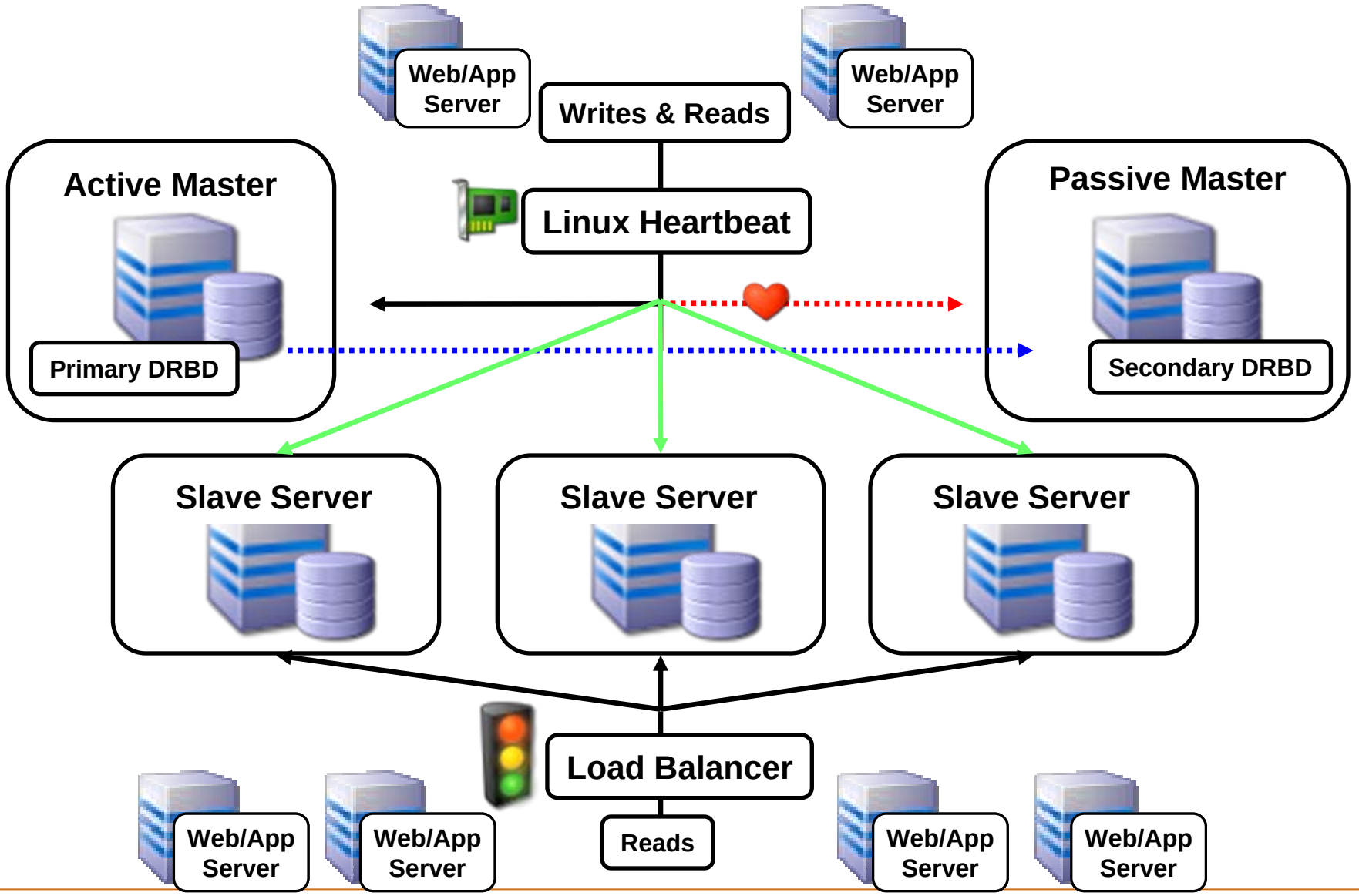
- 2台のマシンに、Active/Passiveアプリと、Passive/Activeアプリを用意
- それぞれのマシン上で、それぞれアプリが動いている状態
- マシンが無駄になることが無い
- フェイルオーバー時には、2つのアプリが1台のマシン上で同時に動く

まとめと応用

- 参照処理のスケールアウトのために
  MySQLレプリケーション
- マスターの自動フェイルオーバーのためにクラスタソフト
- マスターの不整合回避のために
  共有ディスク(SAN)または同期レプリケーション(DRBD)

- これらの組み合わせも可能(次項)

Heartbeat + DRBD + MySQLレプリケーション + L/B



ストレージエンジンによる違い

- InnoDB、NDB、Falcon
  - mysqldの起動時に自動でリカバリ(クラッシュリカバリ)
  - リカバリ時間はデータ量には依存しない。通常は非常に高速
  - Falconも同様
  - 仕組みは違うがNDBも自動でリカバリする
- MyISAM
  - テーブルが破損している場合がある
  - myisamchk –rq、REPAIR TABLEで復旧
  - 自動でのリカバリはしない(my.cnfで--myisam-recoverの設定)
  - リカバリ時間はデータ量に比例する
- MEMORY
  - mysqldの停止によって、レコードは消失する

MySQL Enterpriseによるサポート

- Heartbeat + DRBDのサポートをMySQLより提供
  - DRBDについてはビルド済みバイナリを提供
  - MySQLが問い合わせを一次受け、開発元のLinbitにリダイレクト
  - 24時間×7
  - MySQL Enterprise Gold以上でオプションとして提供
  - サブスクリプション費用は50%増

  - MySQL Enterprise Gold: 2,999USD/Year × 1.5
  - MySQL Enterprise Platinum: 4,999USD/Year × 1.5

大規模環境でよく使われるソリューション

- Heartbeat + DRBD + MySQLレプリケーション + L/B
- パーティショニング
- MySQL Cluster

パーティショニング

1.アクセス先ルックアップ

2.対象サーバにアクセス

ID範囲	対象サーバ
1～100万	Master/Slave1
～200万	Master/Slave2
～300万 (以下同様)	Master/Slave3 (以下同様)

・マッピングテーブル
・ハッシュ
・機能別

・レコード数による分割
・テーブルの種類による分割
・アプリの種類による分割

パーティショニングの考慮事項

- いつパーティショニングするか、タイミングを決める
  - レスポンスタイムの平均値
  - ロードアベレージ

- タイミングを遅らせる工夫
  - アプリが複雑になり、障害切り分けも難しくなるので、パーティショニングを避けられるのであれば避ける(遅らせる)
  - ある程度のスケールアップをする (特にメモリ)
    - 32bit機の2GB Memory <<<< 64bit機の16GB Memory
    - RAID 1+0
  - パフォーマンスチューニングをする
    - 50%～100%の性能向上は珍しくない
    - my.cnfのパラメータによる影響は大きい
    - SQL文の処理効率

- 5.1のパーティショニング機能
  - ハッシュ、主キー、レンジ、リスト、複合型
  - すべてのストレージエンジンでサポート
  - ディスクI/Oボトルネックの軽減に効果的

5.1のパーティショニング機能の効果

```
mysql> CREATE TABLE part_tab
    ->  (  c1 int ,c2 varchar(30) ,c3 date )
    ->      PARTITION BY RANGE (year(c3)) (PARTITION p0 VALUES LESS THAN (1995),
    ->      PARTITION p1 VALUES LESS THAN (1996) , PARTITION p2 VALUES LESS THAN (1997) ,
    ->      PARTITION p3 VALUES LESS THAN (1998) , PARTITION p4 VALUES LESS THAN (1999) ,
    ->      PARTITION p5 VALUES LESS THAN (2000) , PARTITION p6 VALUES LESS THAN (2001) ,
    ->      PARTITION p7 VALUES LESS THAN (2002) , PARTITION p8 VALUES LESS THAN (2003) ,
    ->      PARTITION p9 VALUES LESS THAN (2004) , PARTITION p10 VALUES LESS THAN (2010),
    ->      PARTITION p11 VALUES LESS THAN MAXVALUE );
mysql> create table no_part_tab (c1 int,c2 varchar(30),c3 date);
*** 800万行をロード ***
mysql> select count(*) from no_part_tab where c3 > date '1995-01-01' and c3 < date '1995-12-31';
+----------+
| count(*) |
+----------+
|   795181 |
+----------+
1 row in set (38.30 sec)
mysql> select count(*) from part_tab where c3 > date '1995-01-01' and c3 < date '1995-12-31';
+----------+
| count(*) |
+----------+
|   795181 |
+----------+
1 row in set (3.88 sec)
```

90% の性能改善

MySQL Cluster

- シェアードナッシング型クラスタ
- 同期レプリケーション
- インメモリで動作するため非常に高速
  - 5.1からはデータ領域をディスクに置くことも可能
- スケールアウト可能
- 高速なフェイルオーバー(秒単位のレベル)
- 追加のソフトウェアが不要
  - クラスタソフトの機能をMySQL Cluster自身が実装している
- 特殊なハードウェアが不要
- 詳しくは次のセッションで

MySQLコンサルティングサービス

- MySQLアーキテクチャデザイン
- MySQL Scale-out, HA and Replication Jumpstart
- MySQL Cluster Jumpstart
- MySQLパフォーマンスチューニング
- MySQLデータベース管理
- MySQLマイグレーション
- MySQL Time Hire
- その他、MySQLの専門知識が必要なもの全般

http://www-jp.mysql.com/consulting
consulting-jp@mysql.com

MySQL User Conference Tokyo

- 2007年9月11日(火)、12日(水)
- http://www-jp.mysql.com/news-and-events/news/article_1325.html
- 日本科学未来館(東京・お台場)にて開催
- Marten Mickos、Brian Akerらが来日予定
- 事前登録無料
- 間もなく申込み開始