



DRBD と Heartbeat による フェイルオーバー型クラスタの構築

日本 PostgreSQL ユーザ会北海道支部

矢地重隆

syachi@brownmush.net



- 予算がない
 - ・ 商用のクラスタリングツールまで買う余裕が無い
 - ・ 高価なハードウェア
- 運用のことは後回し
 - ・ 開発の時点では二重化の話なんて無かった
 - ・ いまさら既存システムには手を入れられない
- サーバは2台用意されている
 - ・ でも、障害発生時の切り換えは手動だったり…



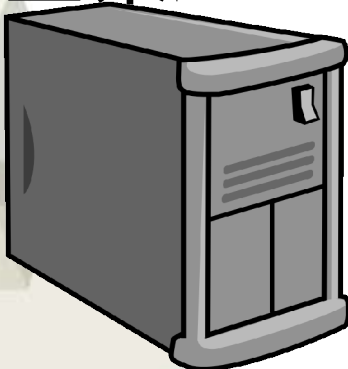
フェイルオーバー型クラスタとは

- 複数のコンピュータを同時に稼働させ、そのうち1台が「主系」、残りを「待機系」として利用
- 通常は主系が全てのサービスを提供
- 主系にトラブルがあれば、待機系が代わりにサービスを提供

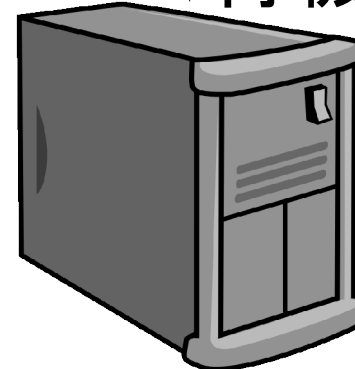
クライアント

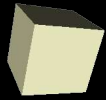


主系

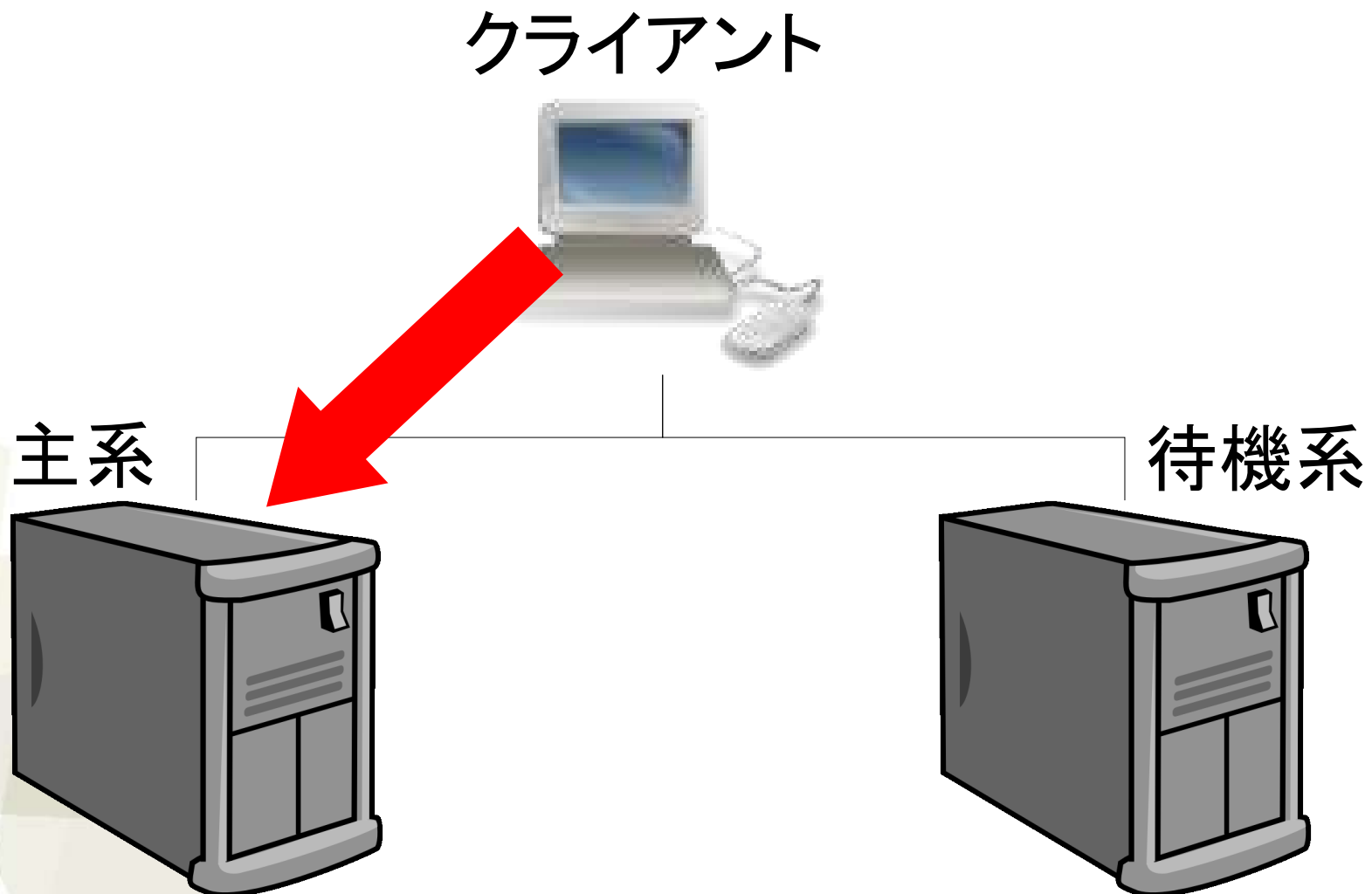


待機系





- 主系が全ての処理を担当
 - ・ クライアントは主系のサービスにアクセスする



- 待機系のマシンが処理を担当
 - ・ 主系に代わって待機系が処理を行う
 - ・ 主系から待機系に処理が移ることを「フェイルオーバー」と呼ぶ

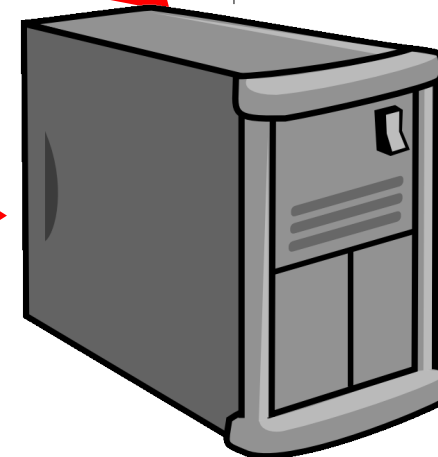
クライアント



主系



待機系



フェイルオーバー

■ HA (High Availability)

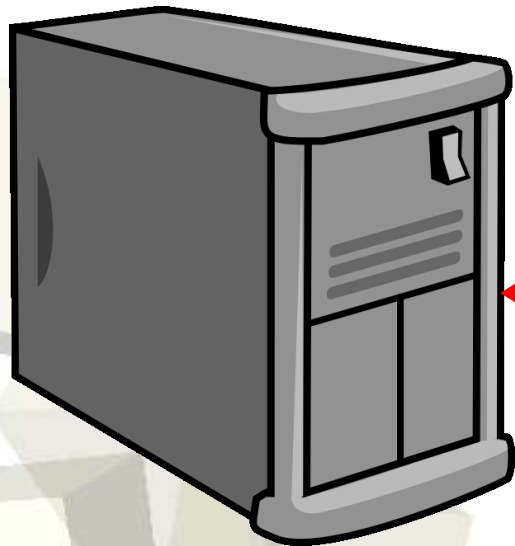
- ◆ 高可用性 = 高い稼働率
 - 「ダウンしない」ではなく「ダウンタイムが短い」
- ◆ 稼働率と停止時間
 - 99% → 約 3.6 日 / 年
 - 99.9% → 約 8.76 時間 / 年
 - 99.99% → 約 52 分 / 年
 - 99.999% → 約 5 分 / 年
 - 99.9999% → 約 30 秒 / 年
 - 99.99999% → 約 3 秒 / 年

フェイルオーバーを実現するには？ (1/3)

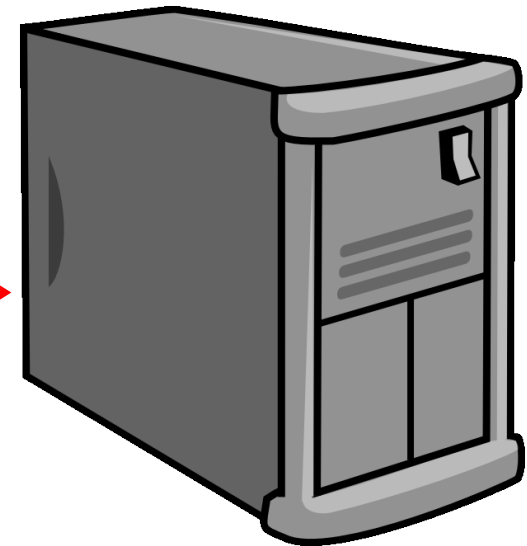
■ 互いの障害検出

- ・ 主系がダウンしたら待機系にフェイルオーバーする
- ・ 双方向で監視

主系



待機系



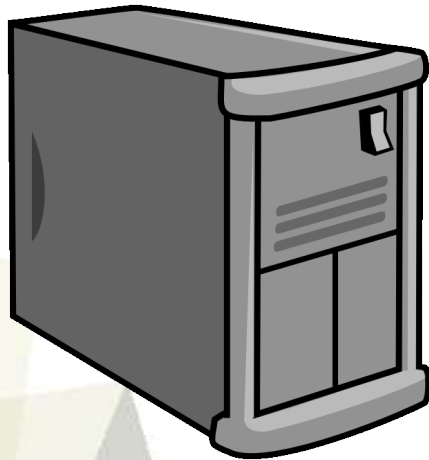
障害の検出



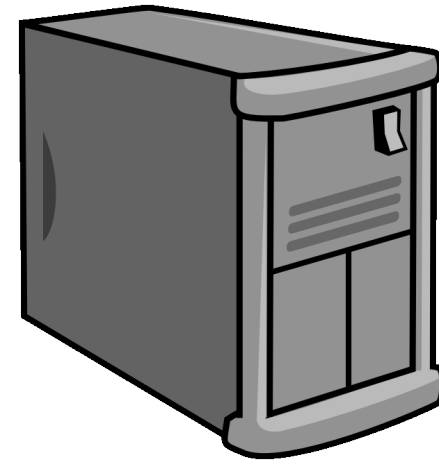
フェイルオーバーを実現するには？ (2/3)

- ハードディスクのデータを同期する
 - ・ データベース、セッションファイル等
- 手法
 - ・ NFS、rsync、共有ディスク

主系



待機系

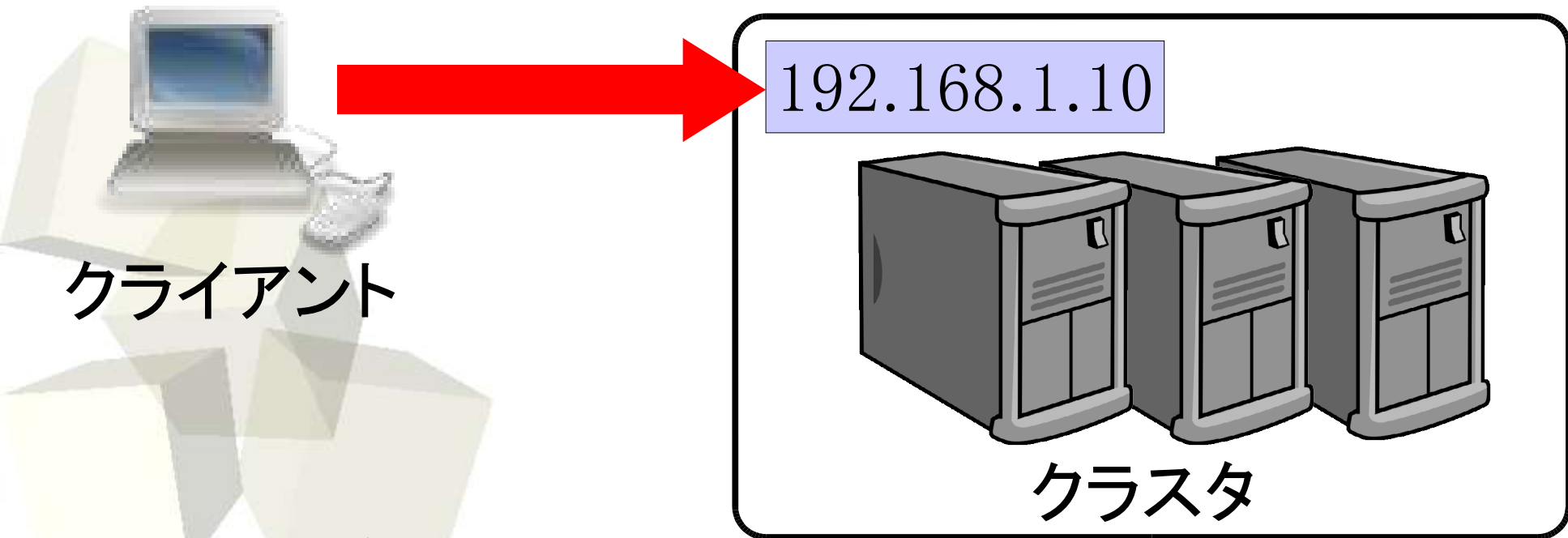


データの同期



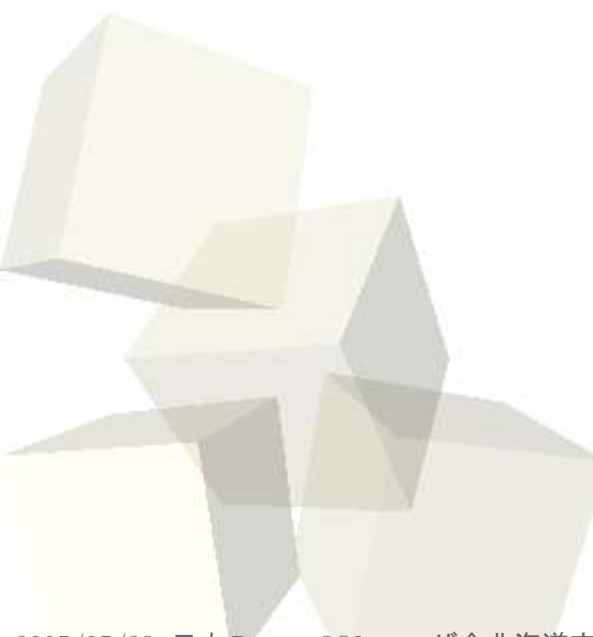
フェイルオーバーを実現するには？ (3/3)

- 複数のサーバを1台に見せる
 - ・ フェイルオーバー後もクライアントの設定を変えずにアクセス可能にする
 - ・ クライアントから見える IP アドレスは変えない
 - ・ クライアントの設定は変更なし





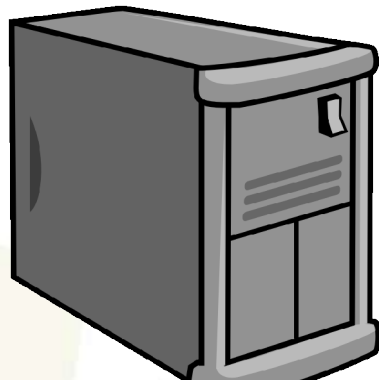
- フェイルオーバー型クラスタの構築
 - ◆ DRBD
 - <http://www.drbd.org/>
 - GPL
 - ◆ Heartbeat
 - <http://www.linux-ha.org/>
 - GPL/LGPL



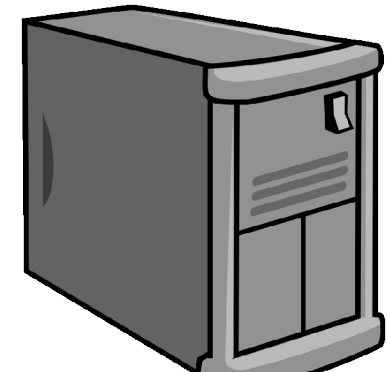
■ ネットワーク経由での RAID1 を提供

- 主系に書き込まれたディスクの内容をネットワーク経由で待機系に同期（ミラーリング）する
- 仮想的な共有ディスクを構築
- mount(write) はどちらか 1 台のみ行うことができる

主系



待機系



仮想的な
共有ディスク

ミラーリング

- フェイルオーバーの仕組みを提供
 - ・ LAN ケーブルやシリアルケーブルを通して、互いの死活監視を行う
 - ・ 主系がダウンした場合に、フェイルオーバーを発生させ、主系のサービス資源を待機系に移動させる
 - 主系の IP アドレスを待機系に移動
 - 待機系で共有ディスクのマウント
 - 待機系で PostgreSQL の起動

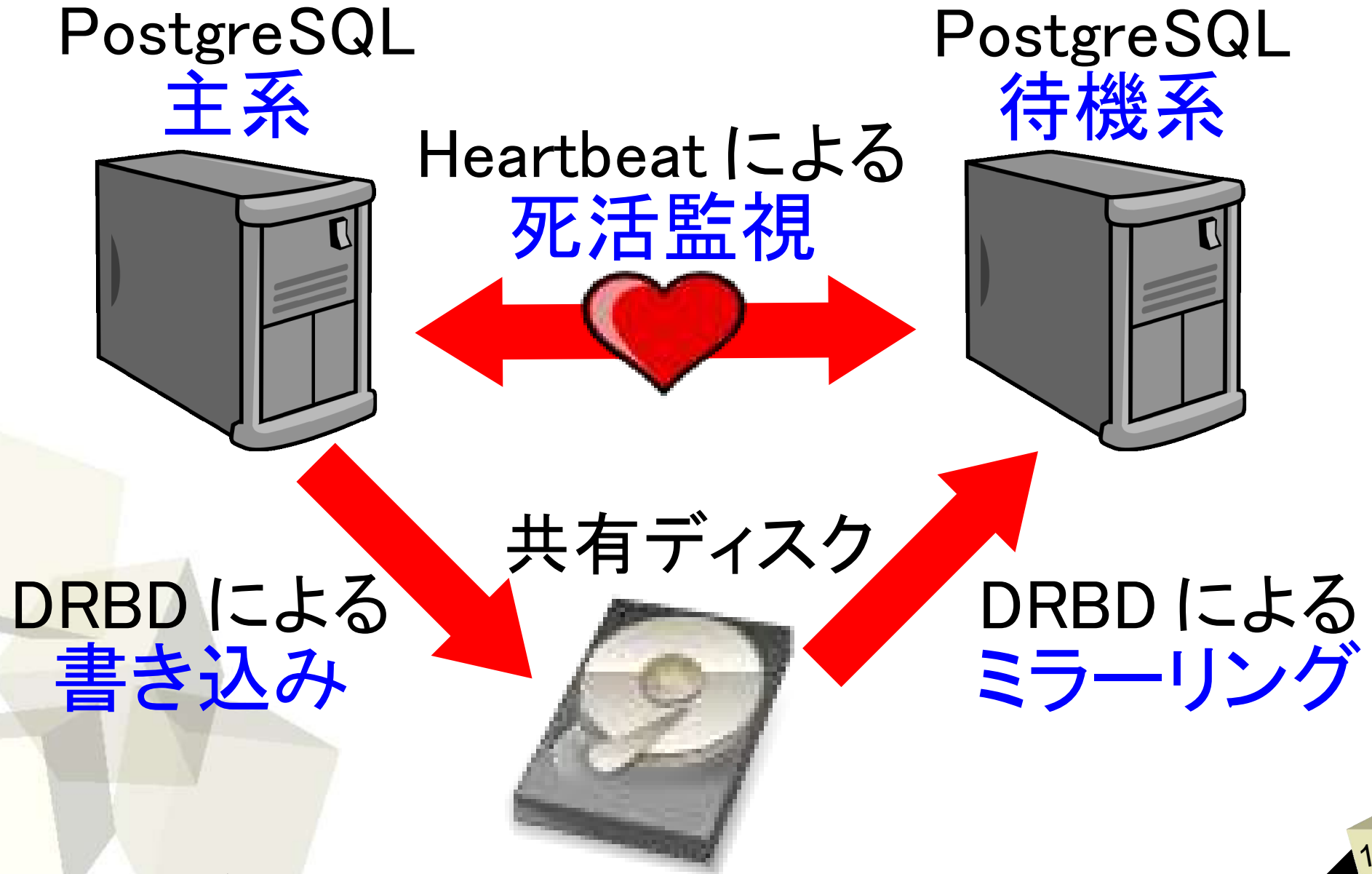
主系

待機系





■ 概念図





ネットワーク構成例

- DRBD は eth1 を使用
- Heartbeat は eth0 と eth1 の 2 系統を使用



クライアント

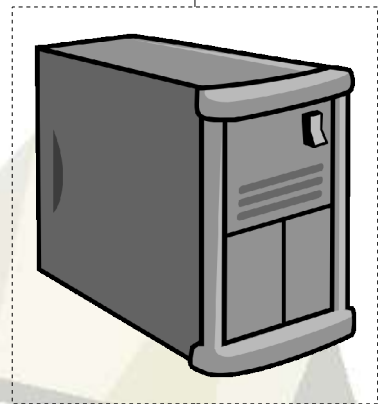
gateway

192.168.1.254

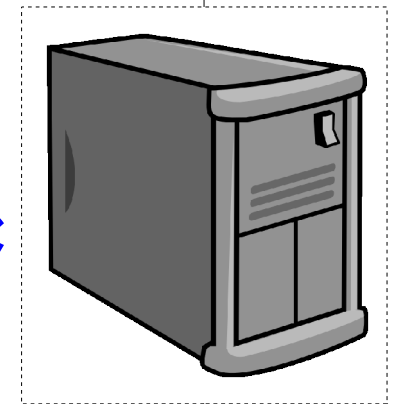


eth0 192.168.1.11

192.168.1.12 eth0



主系



待機系

eth1 192.168.2.11

192.168.2.12 eth1



- ・クライアントは仮想 IP アドレス経由でサーバにアクセスする
- ・共有ディスクは主系のみマウントする



クライアント

gateway

192.168.1.254



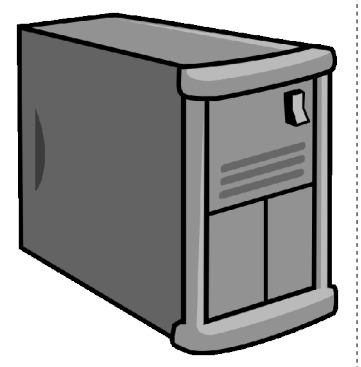
eth0:0 192.168.1.10
eth0 192.168.1.11

PostgreSQL
Apache

mount

共有ディスク

192.168.1.12 eth0



eth1 192.168.2.11

192.168.2.12 eth1



障害発生時

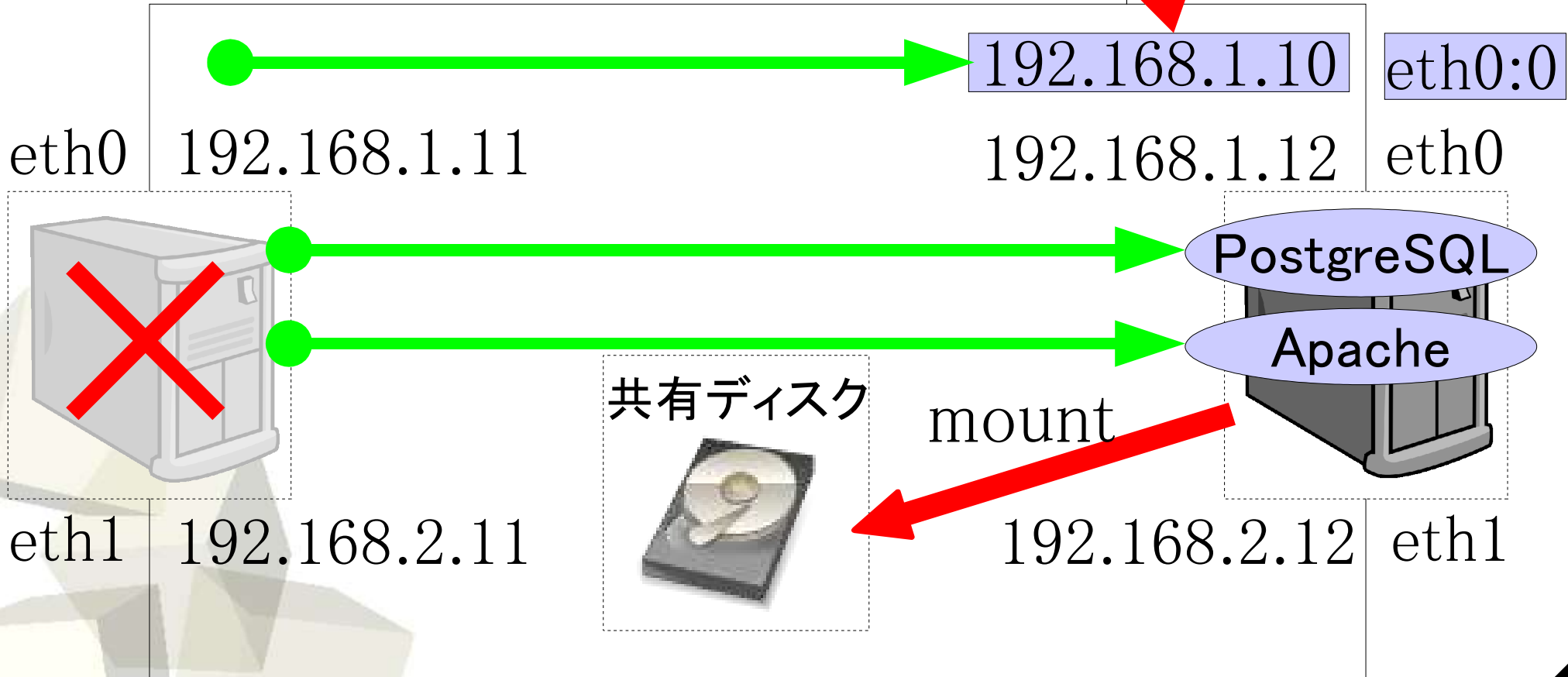
- ・仮想 IP アドレスが待機系に移動
- ・Postgres は待機系で起動する
- ・共有ディスクは待機系がマウント
- ・クライアント側の設定変更は必要ない



クライアント

gateway

92.168.1.254



- HA 化によるデグレードが少ない
 - ◆ PostgreSQL の機能がフルに利用可能
 - クエリベースのレプリケーションに比べ制限が少ない
 - oid、now()、temp table 等、問題なく利用可能
 - 既存のソフトウェアからの移行がしやすい
- PostgreSQL 以外の HA 化も可能
 - ◆ Apache の DocumentRoot
 - ◆ Web アプリケーションのセッションファイル
 - ◆ アプリケーションのディスク上のデータ
- フェイルバック時にサービス停止がない
 - ◆ 切り離れたノードを復旧させる時に、データベースを停止させずに復旧することが出来る


- 負荷分散は出来ない
- 障害検知→フェイルオーバー完了までの時間、サービスが停止する
 - ◆ 障害検出にかかる時間
 - ◆ 仮想 IP アドレスの変更時間
 - ◆ アプリケーションの起動時間
 - PostgreSQL
 - Apache
 - 業務アプリケーション



■ 環境

- ◆ 今回は全て deb パッケージで構成
 - PostgreSQL 7.4.7
 - DRBD 0.7.10
 - Heartbeat 1.2.3
 - Debian GNU/Linux 3.1 (sarge)

■ 共有ディスクに配置するもの

- ◆ PostgreSQL のデータ領域
 - ◆ Apache の DocumentRoot
 - ◆ サンプルアプリケーション
- 

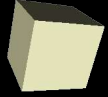
- Heartbeat だけではサービス監視が出来ない
 - ◆ Heartbeat は Heartbeat 同士の死活監視のみ行う
 - Heartbeat プロセスが落ちた場合のみ異常とみなす
 - 電源断
 - LAN ケーブル断線
 - ◆ 別の死活監視も合わせて行うのが望ましい
 - port 5432 の監視
 - 実際にクエリを投げて監視
 - プロセス監視

■ 構築のポイント

- ◆ アプリケーションの起動 / 停止は Heartbeat が行う
 - フェイルオーバーにかかわるアプリケーションは Heartbeat が起動 / 停止を担当
 - PostgreSQL、Apache 等
 - サーバ起動時に自動で起動させている場合は注意
- ◆ UID と GID
 - 主系と待機系で同じ ID を指定する
- ◆ ロックファイル等
 - PID ファイル等が残っていても大丈夫？

■ こんなことにもご注意を

- ◆ SPF (Single Point of Failure) = 一点障害
 - 一つの機器の障害がシステム全体の障害となる事
 - Gateway は？
 - 電源は？
 - 連携している他のサーバは？



ありがとうございました

