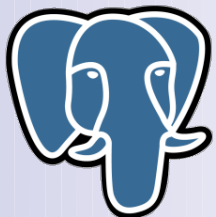


HAクラスタで  
PostgreSQLを高可用化(後編)  
～レプリケーション編～

2012年9月29日

しくみ+アプリケーション勉強会

松尾 隆利  
NTT OSSセンタ



# はじめに

---

- 今回は『HAクラスタでPostgreSQLを高可用化』の**後編**です
  - ストリーミングレプリケーション構成のクラスタリングのお話がメインです
  - 前編の入門編を復習されているのを前提にお話します。
    - ・ [http://linux-ha.sourceforge.jp/wp/wp-content/uploads/pacemaker\\_20120526JPUG.pdf](http://linux-ha.sourceforge.jp/wp/wp-content/uploads/pacemaker_20120526JPUG.pdf)
  - レプリケーション自体のお話も省きます。

# 開発経緯

---

## □ 昔～昔……

- PostgreSQL 8.3 + 独自パッチで同期レプリケーション機能を実装し、Heartbeat(Pacemakerの前身)でHAクラスタ化を実現するPG-REXというプロジェクトがあったそうなの……

## □ 2010年

- PostgreSQL 9.0 で非同期レプリケーション実装  
→ 同期レプリケーションを実現する独自パッチ + Pacemaker用RA開発でクラスタリング実現
- 9.0の非同期レプリケーションでも動くように手直しし、Pacemakerコミュニティへ投稿 → **リジェクト**



## □ 2011年

- PostgreSQL 9.1 で同期レプリケーション実装  
→ 新たにPacemaker用RAを開発しクラスタリング実現
- Pacemakerコミュニティ意見も反映し投稿！ → **絶賛！ (tremendous job !!)**

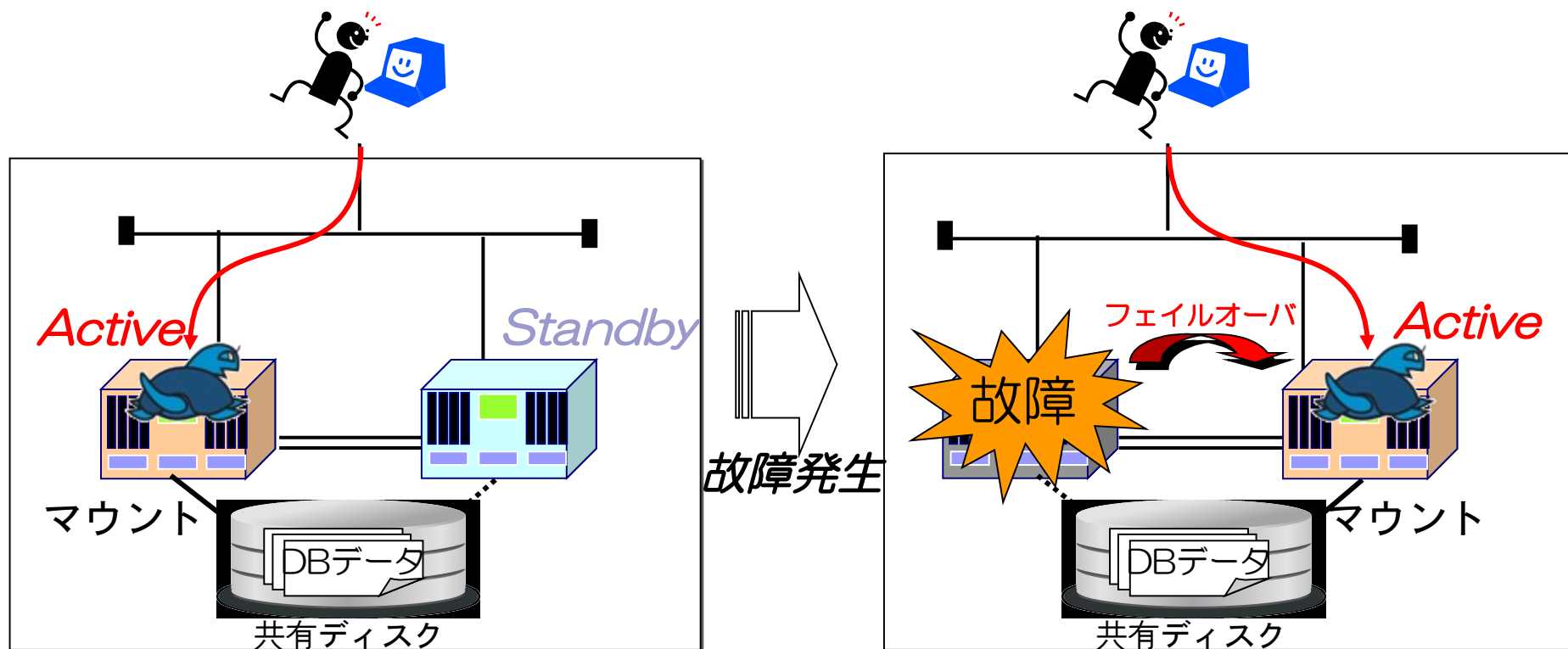


## □ 2012年

- 4月13日 コミュニティのリポジトリにマージ
- 5月16日 resource-agents 3.9.3 として無事リリース  
※Linux-HA Japan のリポジトリパッケージ 1.0.12-1.2 に同梱

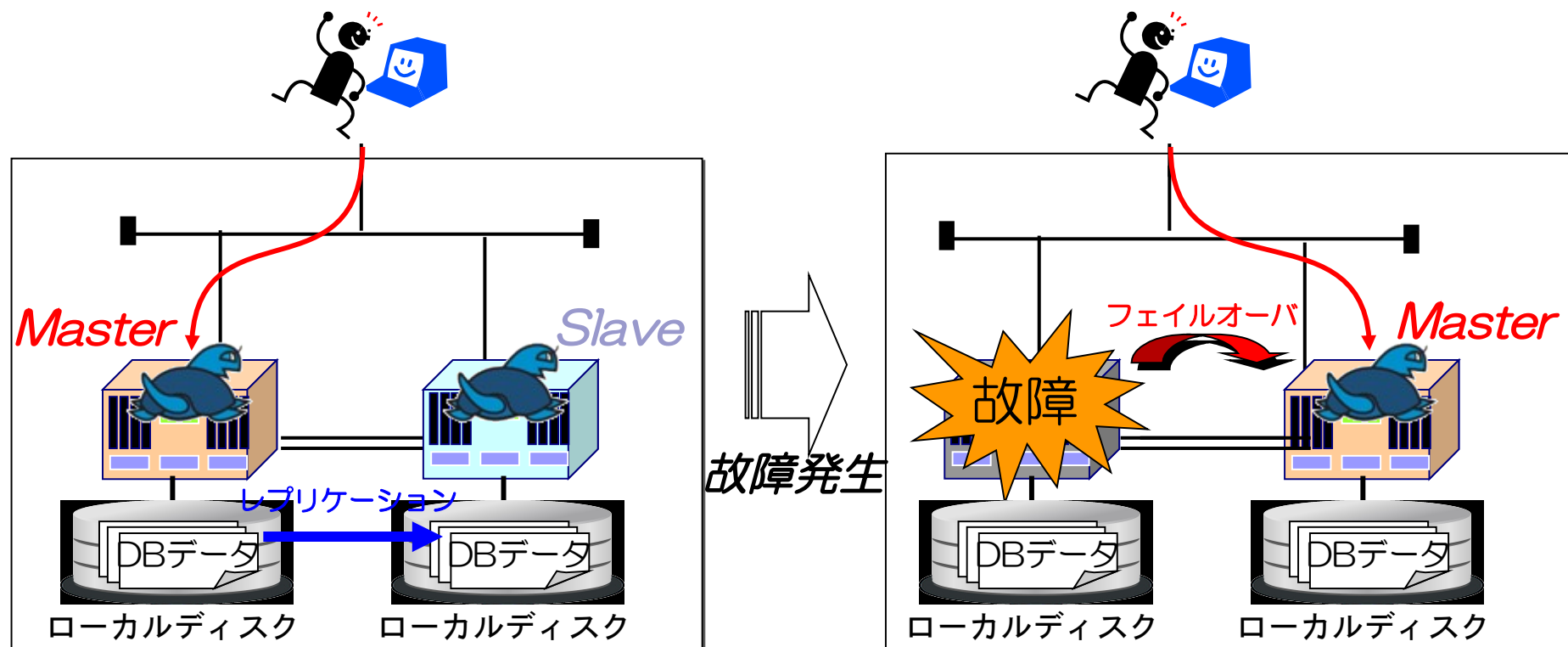
# 前編の構成 ～Active/Standby 構成～

- PostgreSQLのデータは共有ディスク上に配置し、2台のサーバ間で共有
- 通常はActiveサーバでサービスを提供し、Activeサーバ故障時はStandbyサーバがActiveとなりサービスを提供












# 今回の構成 ～Master/Slave 構成～

- PostgreSQLのデータはそれぞれのサーバのローカルディスク上に配置しPostgreSQLのストリーミングレプリケーション機能を用いて共有
- 通常はMasterサーバでサービスを提供し、Masterサーバ故障時はSlaveサーバがMasterとなりサービスを継続



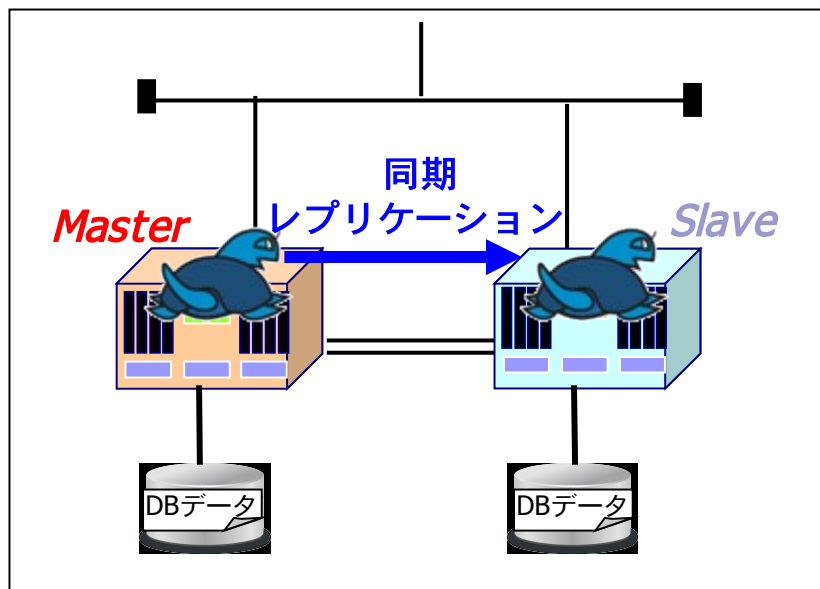
# Active/Standby vs Master/Slave



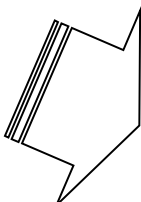
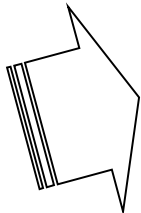
	Active/Standby	Master/Slave
ハードウェア費用	共有ディスク(相当のもの)必須	
運用のしやすさ	 データは1箇所のみ	2箇所のデータの整合性を考慮
データの安全性	最新データは 共有ディスク上のみ	 最新データは2箇所に分散
サービス継続性	 フェイルオーバー時に リカバリに時間を要する	 Slave故障がサービスに影響 (同期レプリケーション使用時)
DB性能	 レプリケーションの オーバーヘッドなし	 共有ディスク構成をとれない 某高速ストレージを活用可
負荷分散	構成上不可能	 ReadOnlyクエリを Slaveで処理可能
実績		これから……

それぞれ一長一短。サービスの要件に応じて選択すること。

# レプリケーション構成のHAクラスタ化 3大機能

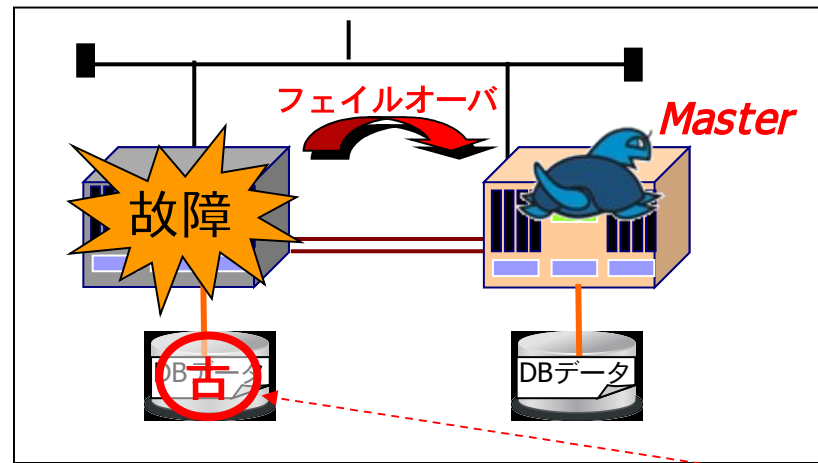


Master  
故障発生

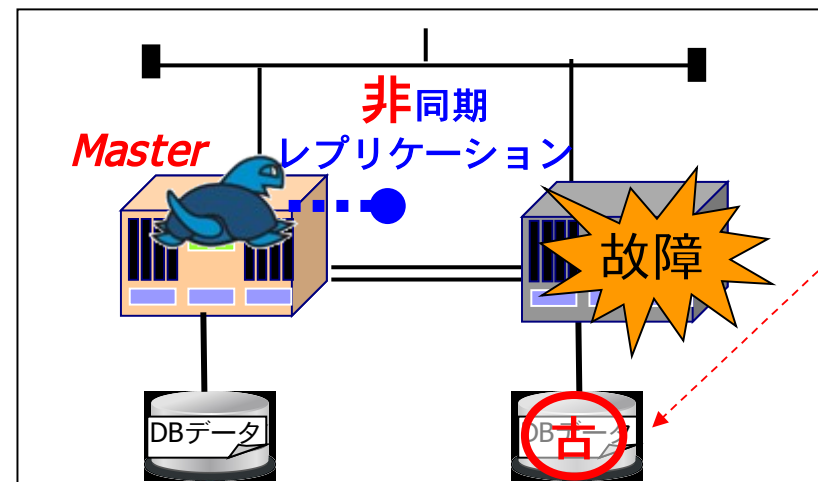


Slave  
故障発生

## ①フェイルオーバー

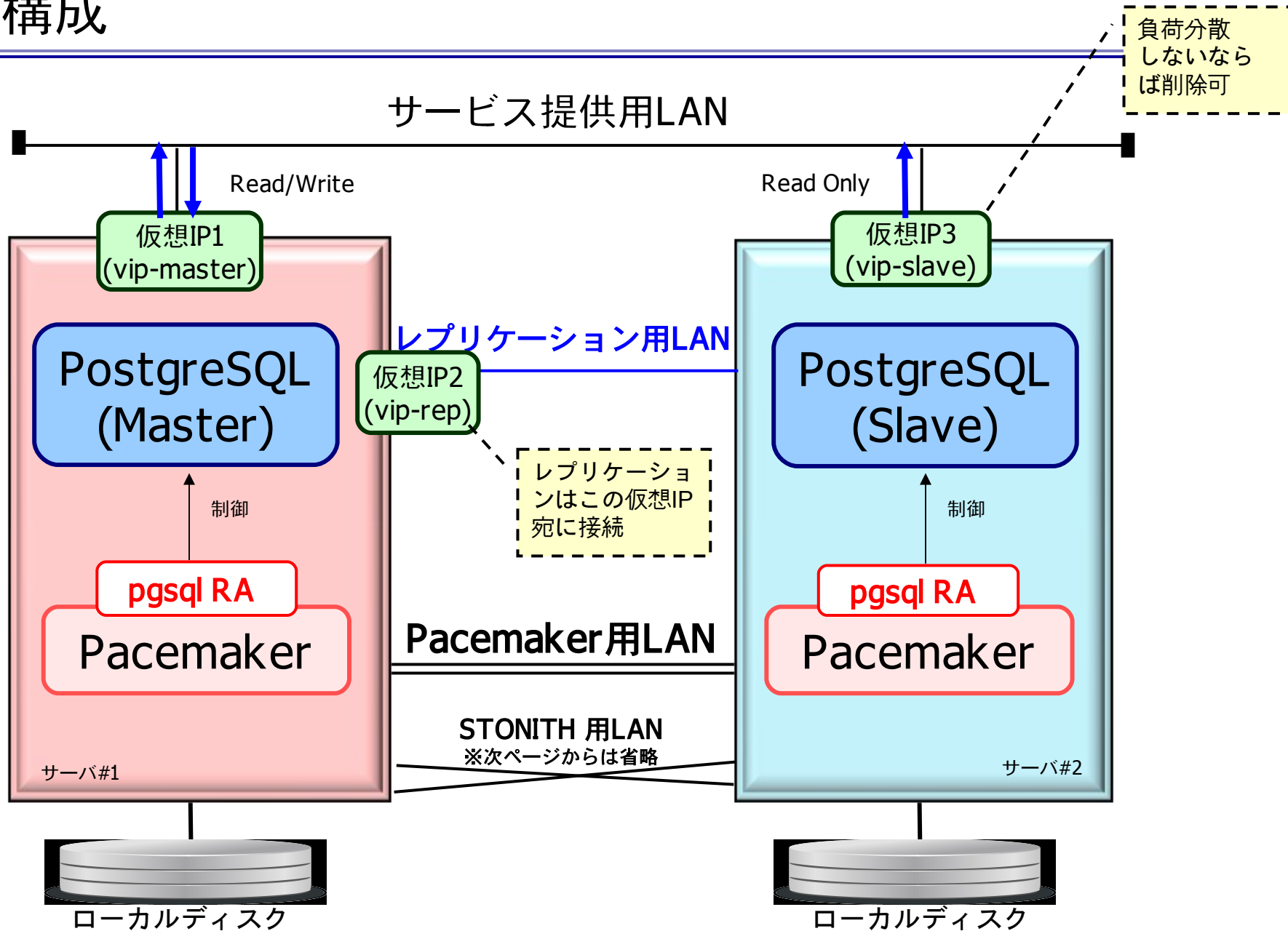


## ②同期・非同期の切替



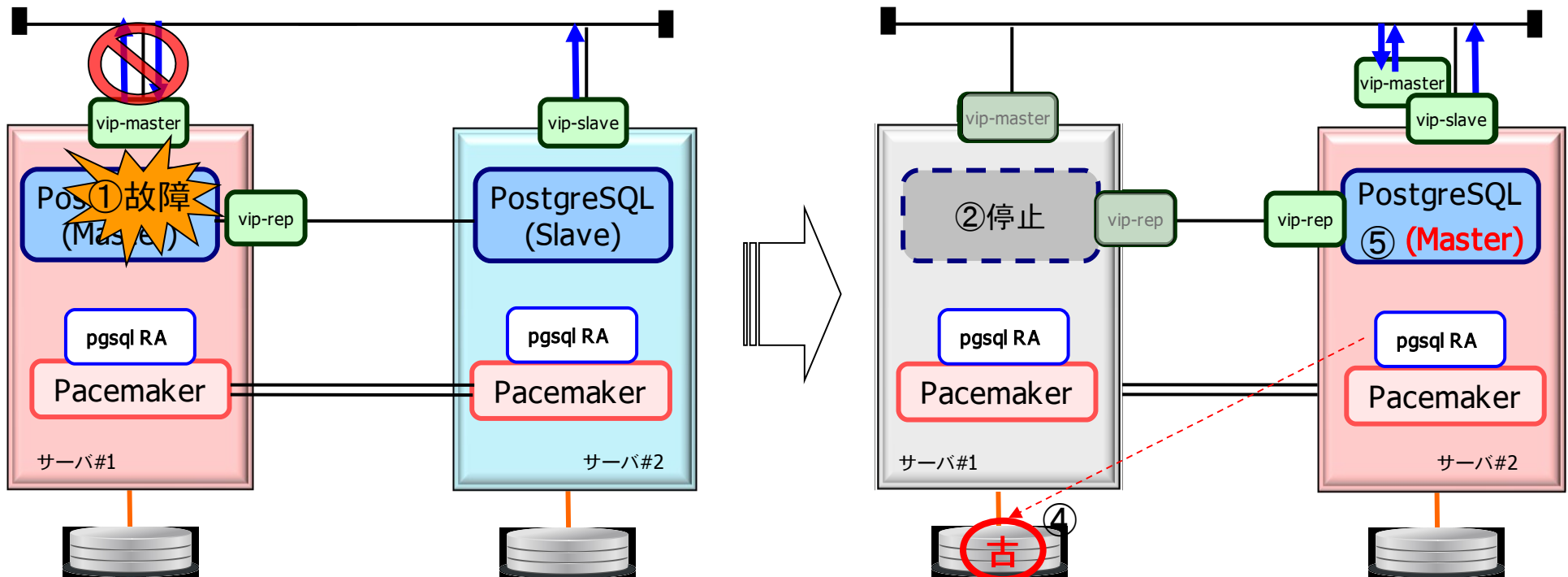
③ データの状態管理

# 基本構成





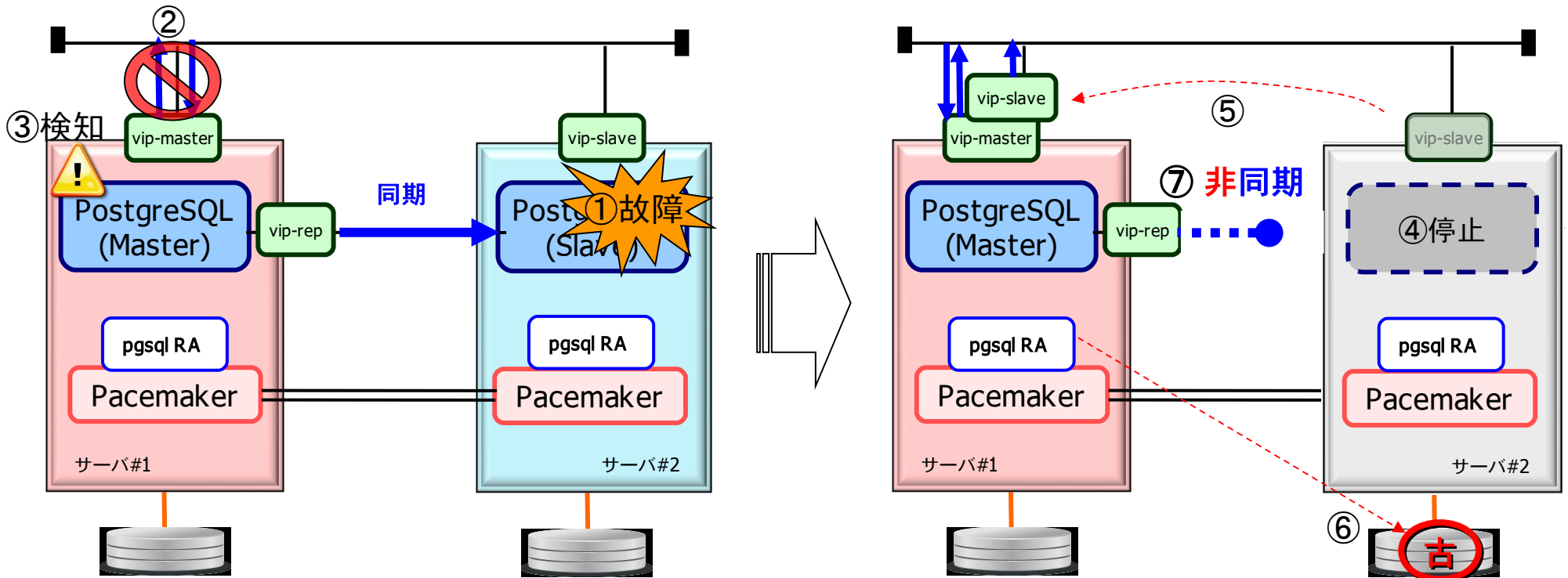
# 基本動作1 : Masterのフェイルオーバー



① #1のPostgreSQLの故障を検知

- ② #1のPostgreSQLを停止
- ③ 仮想IP(vip-master, vip-rep, vip-slave)を停止
- ④ #1のデータが古いことを記録
- ⑤ #2のPostgreSQLを**Masterに昇格(promote)**
- ⑥ #2で仮想IP(vip-master, vip-rep, vip-slave)を起動

# 基本動作2：同期・非同期の切替



- ① Slaveの故障発生
- ② Masterのトランザクション停止  
～ レプリケーションのタイムアウト待ち～
- ③ #1でレプリケーション切断を検知

```
SELECT * from pg_stat_replication
```

- ④ #2のPostgreSQLを停止
- ⑤ #2の仮想IP(vip-slave)を#1に付け替え
- ⑥ #2のデータが古いことを記録
- ⑦ #1のPostgreSQLを**非同期**に変更  
→ トランザクション再開

---

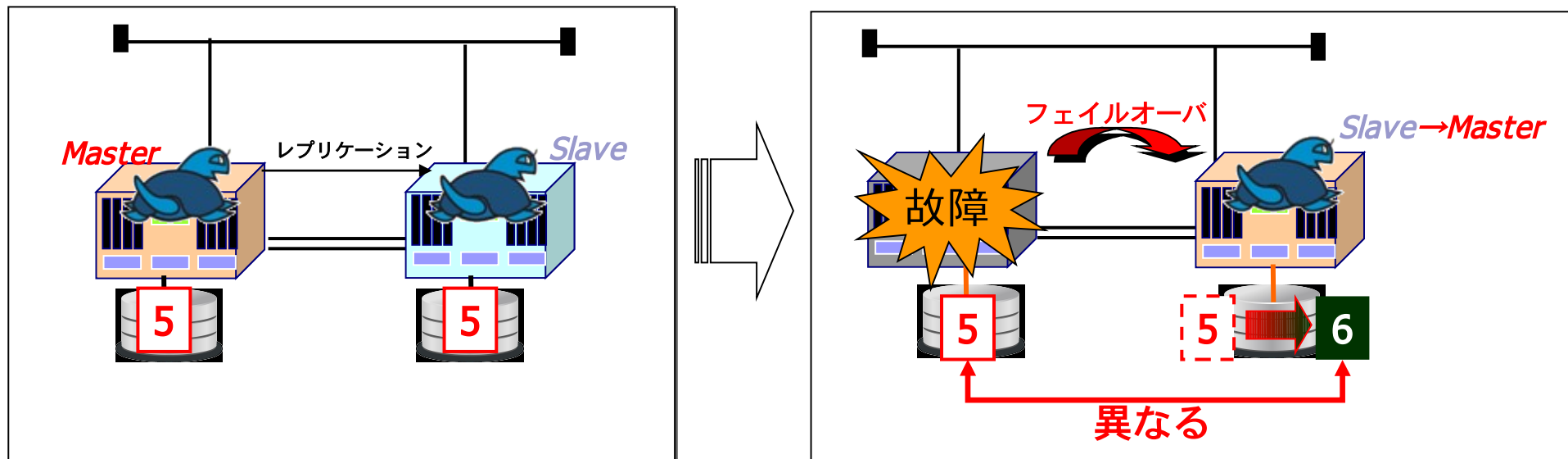
ここまでが基本動作  
次はフェイルオーバー後の復旧

# TimelineID



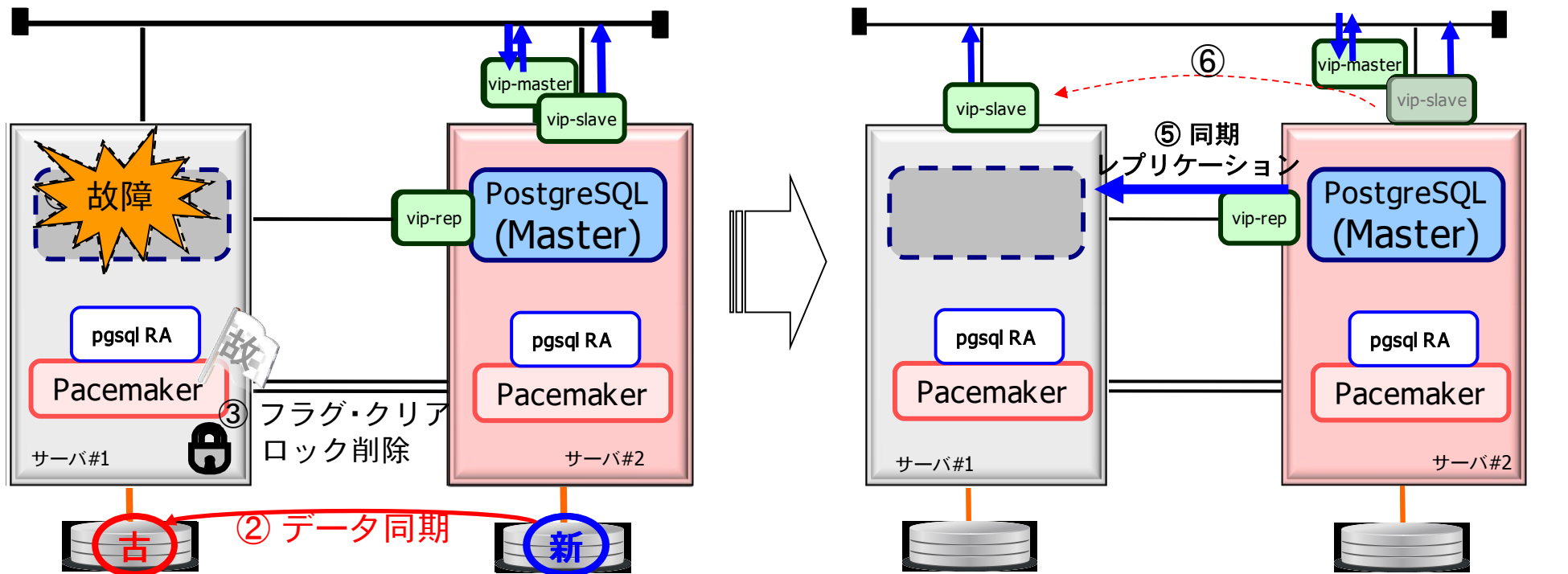
- ❑ PostgreSQLがSlaveからMasterへ昇格した際インクリメントされる数値
- ❑ TimelineIDが異なるとレプリケーション接続ができない

## フェイルオーバー時



TimelineIDをそろえるには  
新Masterのデータを旧Masterへコピーする必要あり

# 運用1: フェイルオーバー後の復旧



- ① 故障の復旧
- ② #1のデータを#2と同期  
→ TimelineIDがそろふ
- ③ #1のロックファイル削除と  
Pacemaker上の故障フラグを  
クリア

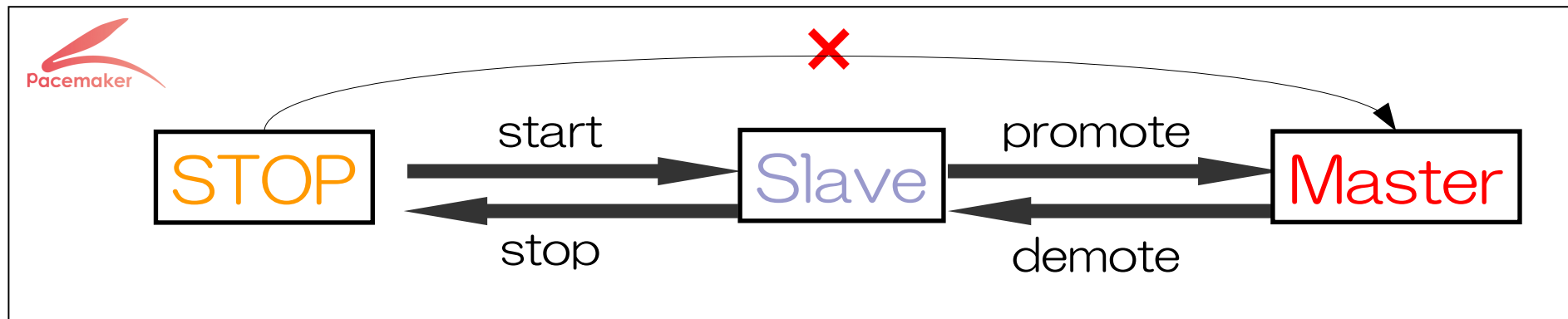
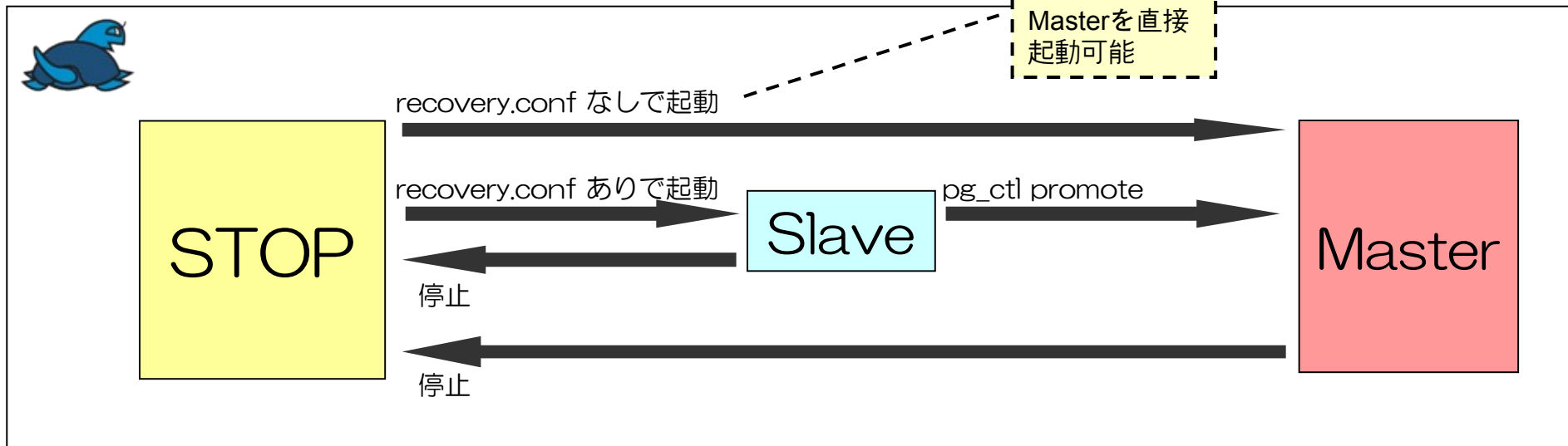
(手動)

- ④ #1のPostgreSQLをSlaveで起動
- ⑤ レプリケーション開始  
→ 非同期で接続→同期に切替
- ⑥ #2の仮想IP(vip-slave)を#1に付け替え

---

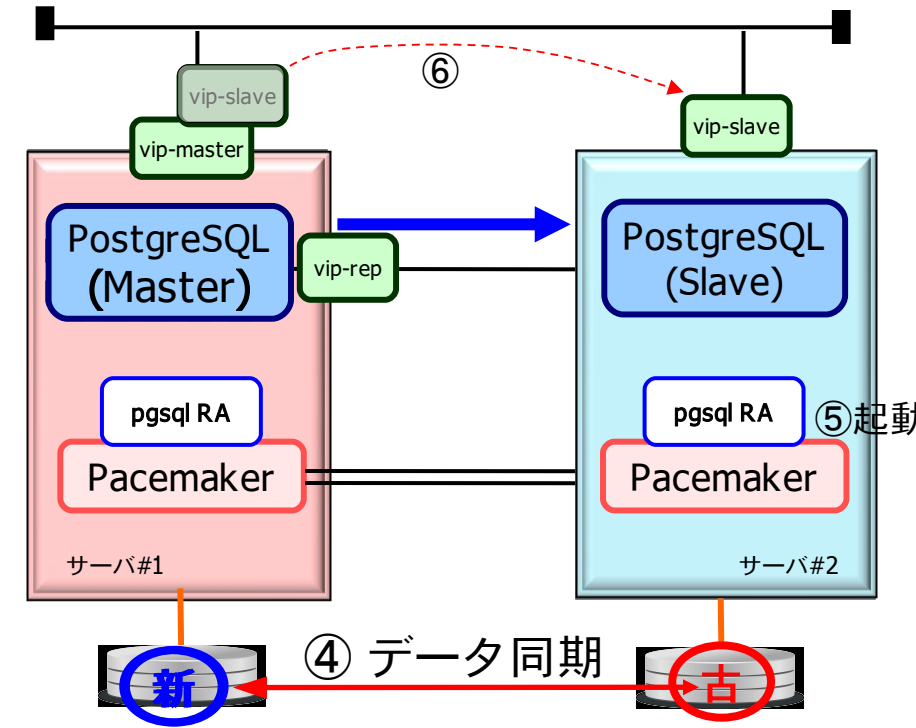
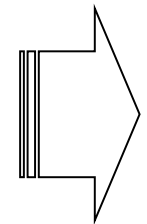
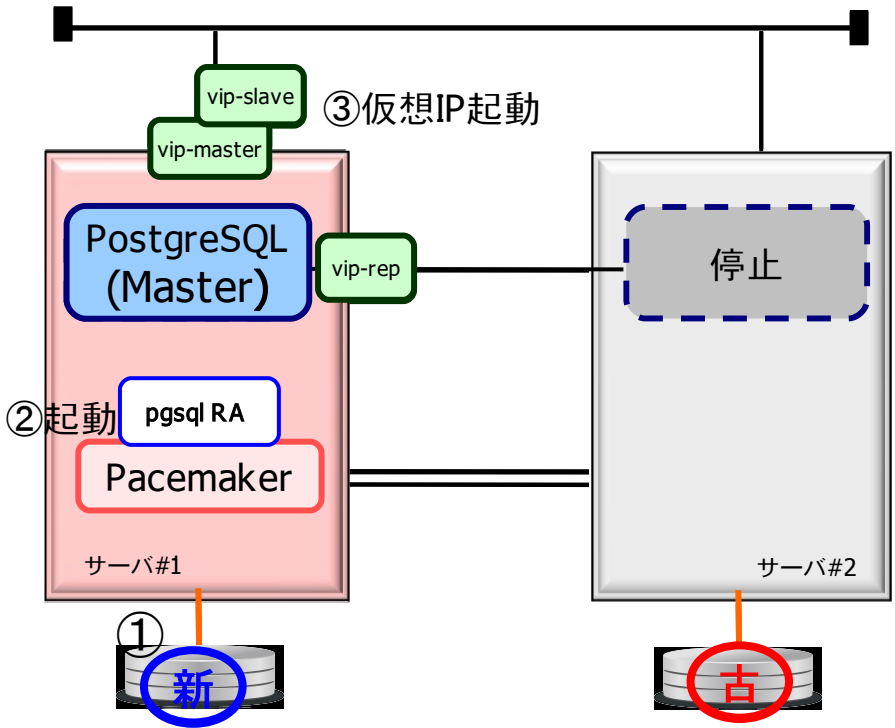
次は起動方法

# PostgreSQLとPacemakerの状態遷移



PostgreSQLのMasterは必ずSlaveを経由(recovery.confはRAが作成)して起動される  
→ Master起動時もTimelineIDがインクリメントされる

# 運用2：起動



- ① データが新しい方のサーバーを選択
- ② 選択したサーバのPacemakerを起動  
→ Slaveで起動 → Masterに遷移  
→ TimelineIDがずれる
- ③ 仮想IPが#1で起動

- (手動) ④ #2のデータを#1と同期  
→ TimelineIDがそろう
- (手動) ⑤ Pacemaker起動  
→ レプリケーション開始
- ⑥ #1の仮想IP(vip-slave) を#2に付け替え



---

ここからは状態遷移詳細

# その前に用語を整理



PostgreSQLとPacemakerの状態遷移は一致しないため、  
二者を区別するために以下を使い分け

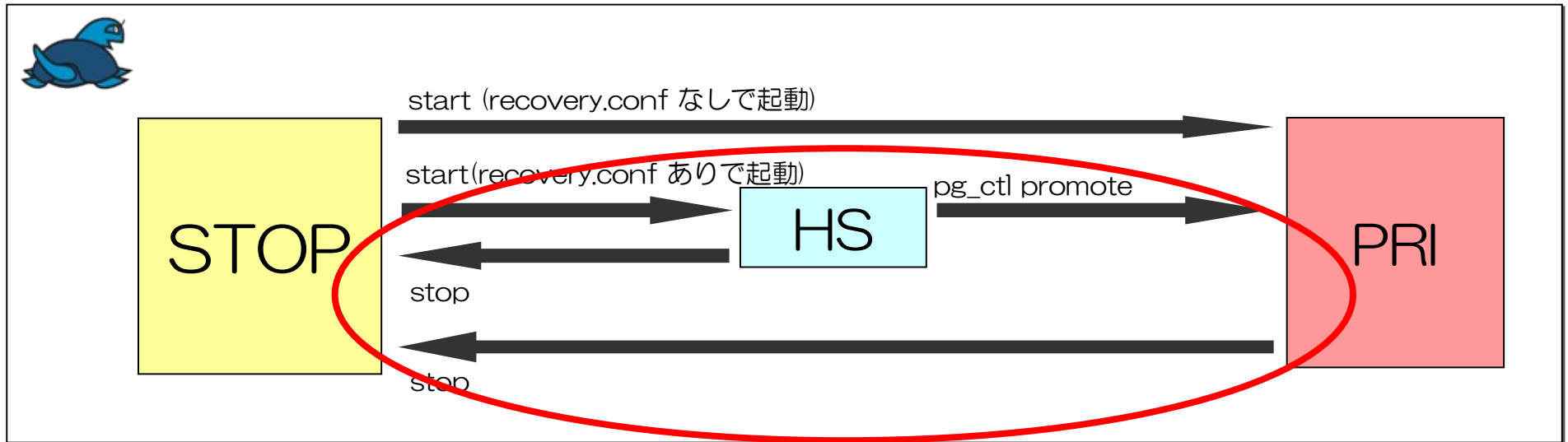
## □PostgreSQLの状態を表す用語

- PRI : Primary(Master)状態
- HS : Hot Standby(Slave)状態
- STOP : 停止している状態

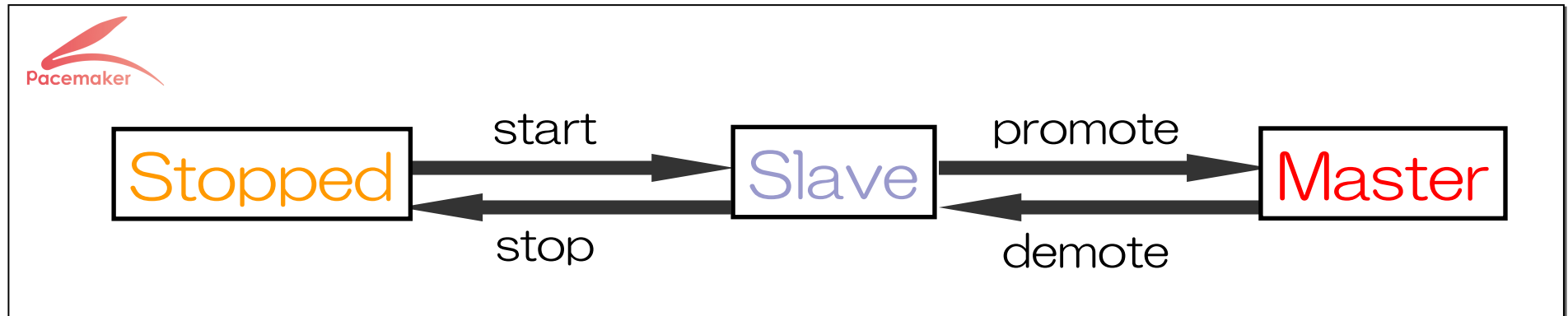
## □Pacemakerの状態を表す用語

- Master : アプリケーションをMasterとして管理している状態
- Slave : アプリケーションをSlaveとして管理している状態
- Stopped : アプリケーションを停止として管理している状態

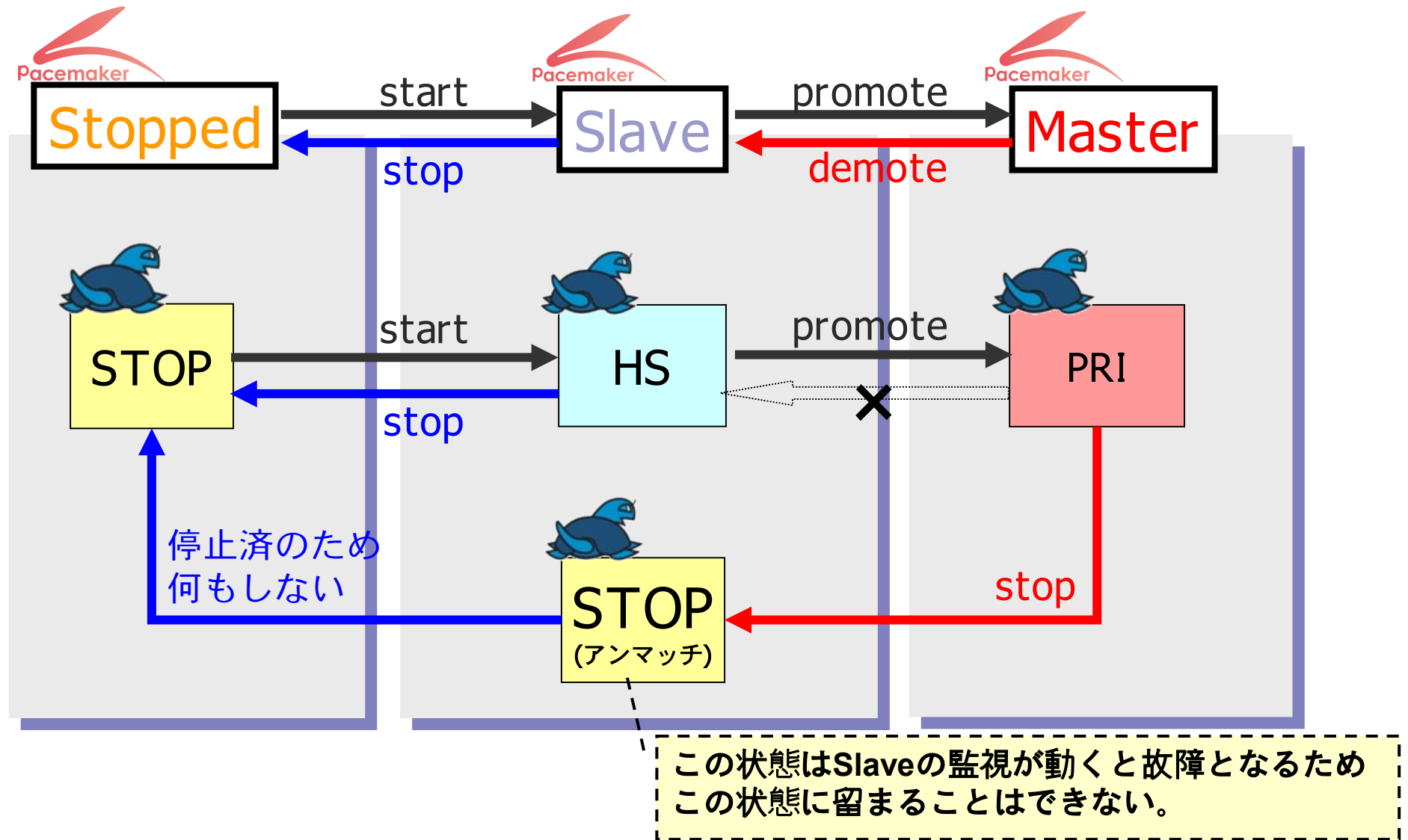
# PostgreSQLとPacemakerの状態遷移 (用語区別版)



この部分をPacemakerの状態遷移とマッピング



# 状態遷移のマッピング



# HSの複数状態

## □ PostgreSQLのHSには複数の状態を管理

1. PRI へ未接続

... HS:alone

2. PRI へ接続中

a. 同期レプリケーション以外 ... HS:(others)

b. 同期レプリケーション中 ... HS:sync

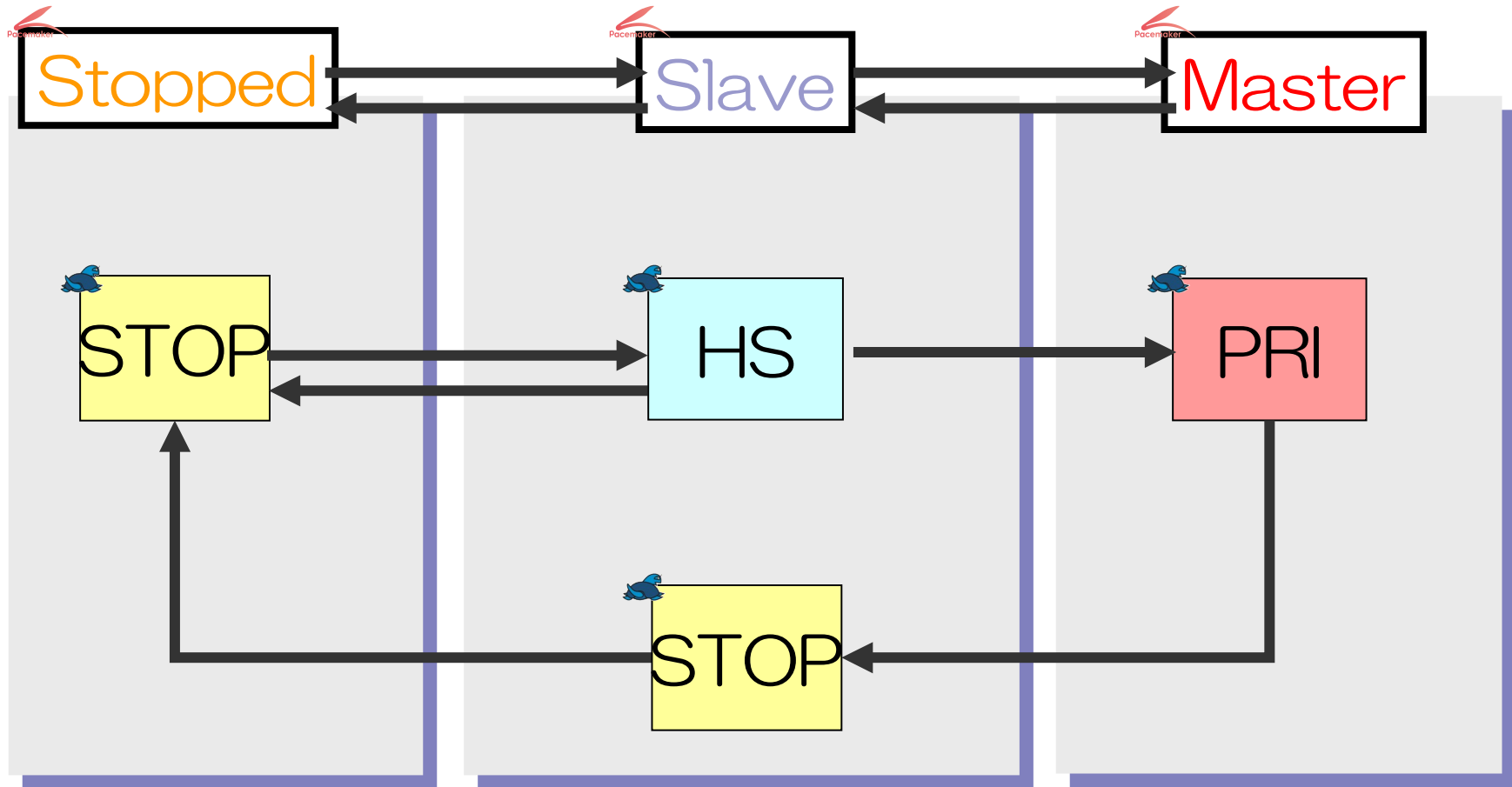
HS:(others) は以下の二つの状態をとる

- HS:connected
- HS:async

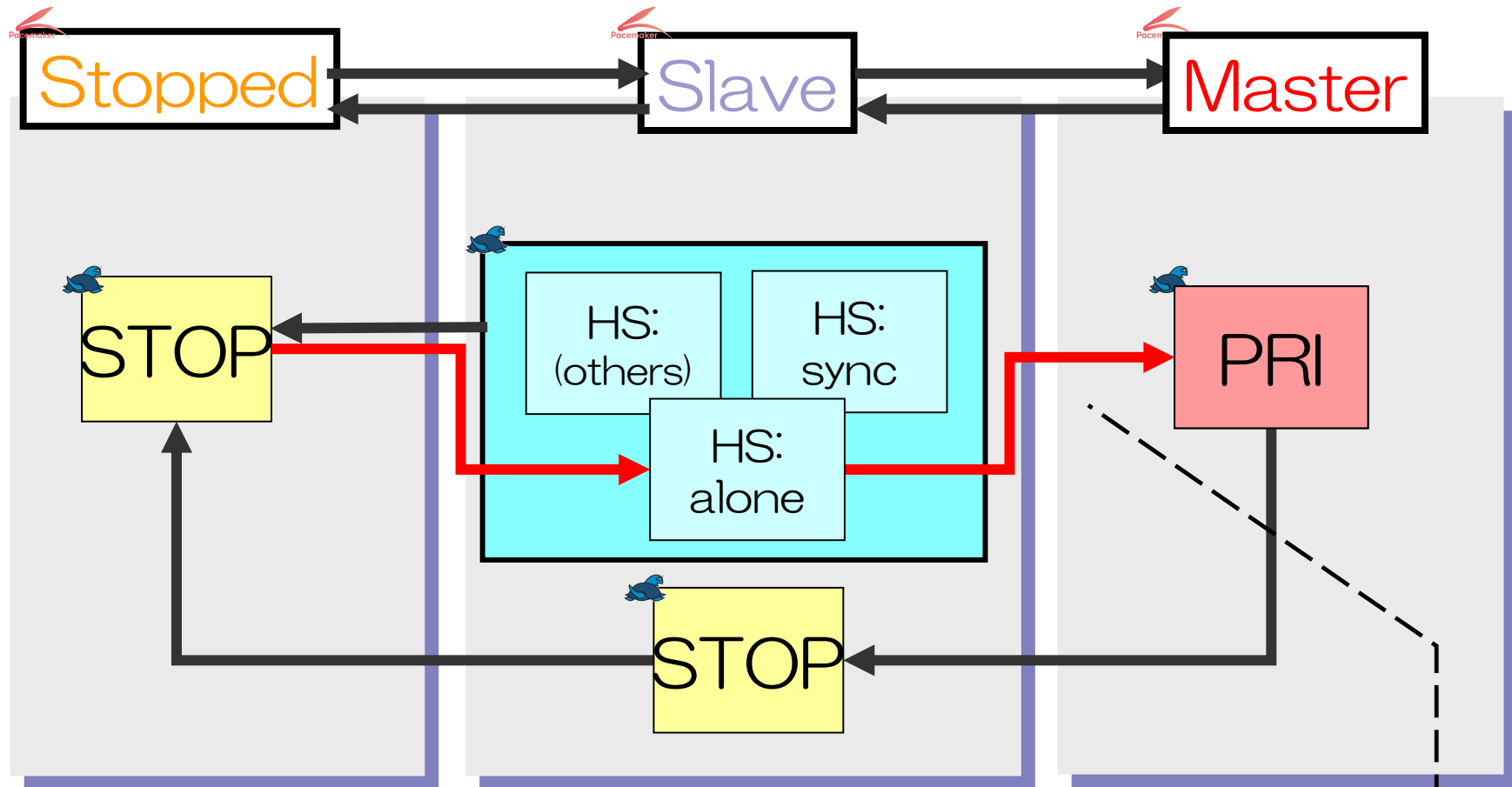
Pacemakerでこれら状態を“pgsql-status”属性として管理

※リソースID名をpgsqlとした場合

# 状態遷移のマッピング (改良前)



# 状態遷移のマッピング (改良後)



他のノードにPRIがおらず、自分のデータが最新の場合

# データの新旧判断方法

---

## HSのデータが最新かどうかを判断する方法

### □PRI存在時

- PRIとの接続状態を基にデータの状態を記録
  - ・ PRI存在時は自分がPRIに昇格することはないため記録のみ

### □PRI故障時

- PRI故障直前のPRIとの接続状態を基に判断

### □初期起動時 (PRIが一度も存在したことがない時)

- 他にHSがいる場合
  - ・ 他のHS間でデータの新旧を比較して判断
- 他にHSがいない場合
  - ・ 自分が最新だと判断



# データの状態

## □ PRI存在時に記録するHSのデータ状態

1. PRI へ未接続

.. DISCONNECT

2. PRI へ接続中

a. 同期レプリケーション以外 .. STREAMING|ASYNC (例)

b. 同期レプリケーション中 .. STREAMING|SYNC

古いデータ

最新データ

※ pg\_stat\_replication ビューから取得

□ PRI時

.. LATEST

Pacemakerでこれら状態を“pgsql-data-status”属性として管理

※リソースID名をpgsqlとした場合

# pgsql-status と pgsql-data-status 属性の違い

## □ pgsql-status

- PostgreSQL の現在の実行状態を示す
  - ・ 停止 : STOP
  - ・ HSで動作 : HS:alone, HS:async, HS:sync
  - ・ PRIで動作 : PRI
  - ・ 不明時 : UNKNOWN

### 用途

- ・ HSの現在の状態把握
- ・ HSの状態と他のリソースとの連携



## □ pgsql-data-status

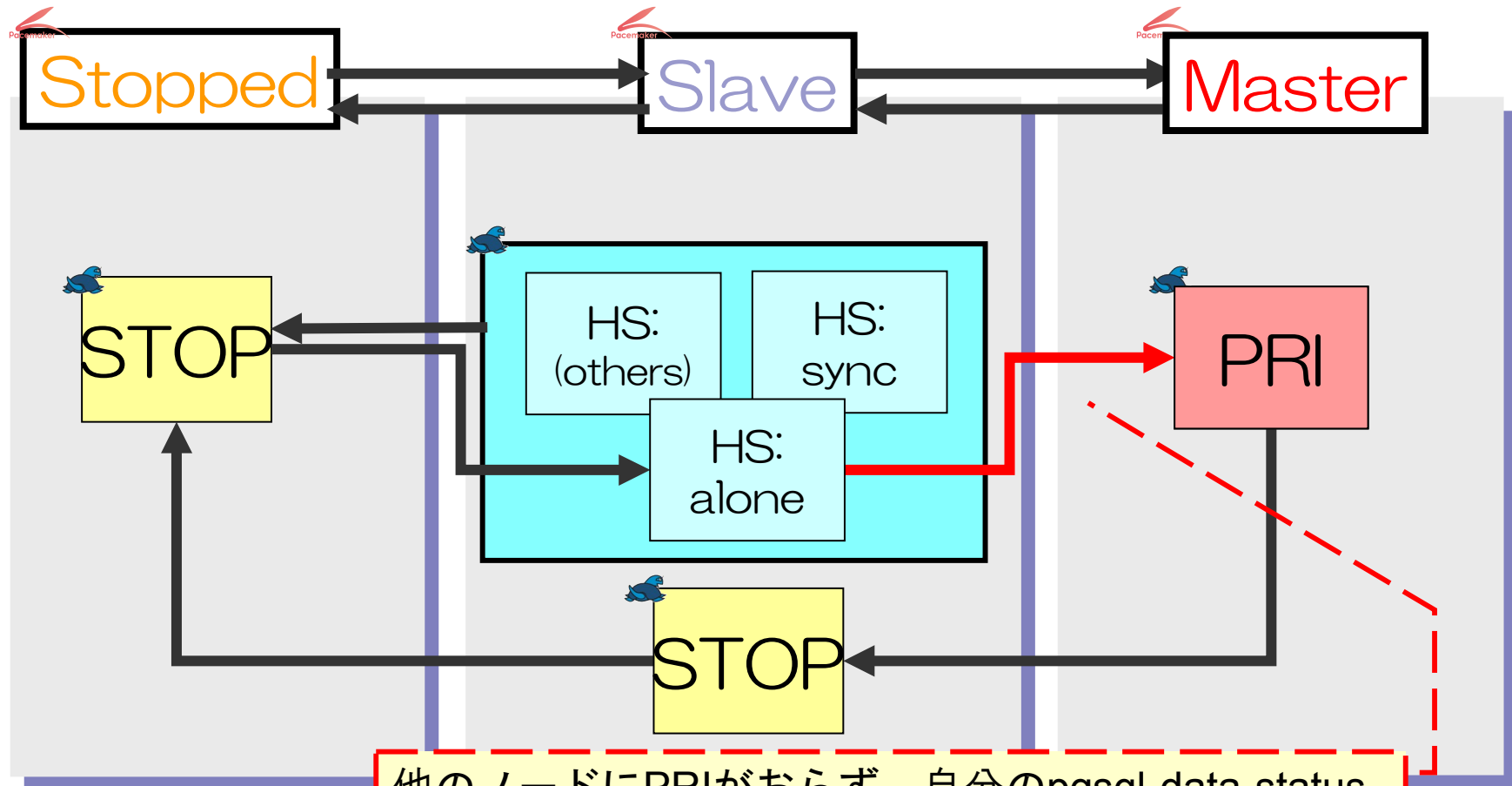
- PRI との関係によって決まるデータの状態。
  - ・ PRI未接続時 : DISCONNECTED
  - ・ HS時 : STREAMING|ASYNC, STREAMING|SYNC 等
  - ・ PRI時 : LATEST
- PRI 故障時、PRI故障直前の状態が残る
  - ・ PostgreSQL の実行状態と必ずしも一致しない
    - pgsql-status=HS:aloneで、pgsql-data-status = LATEST があり得る

### 用途

- ・ データの新旧状態記録
- ・ Masterへの昇格可否の判断



# 状態遷移のマッピング (条件整理後)

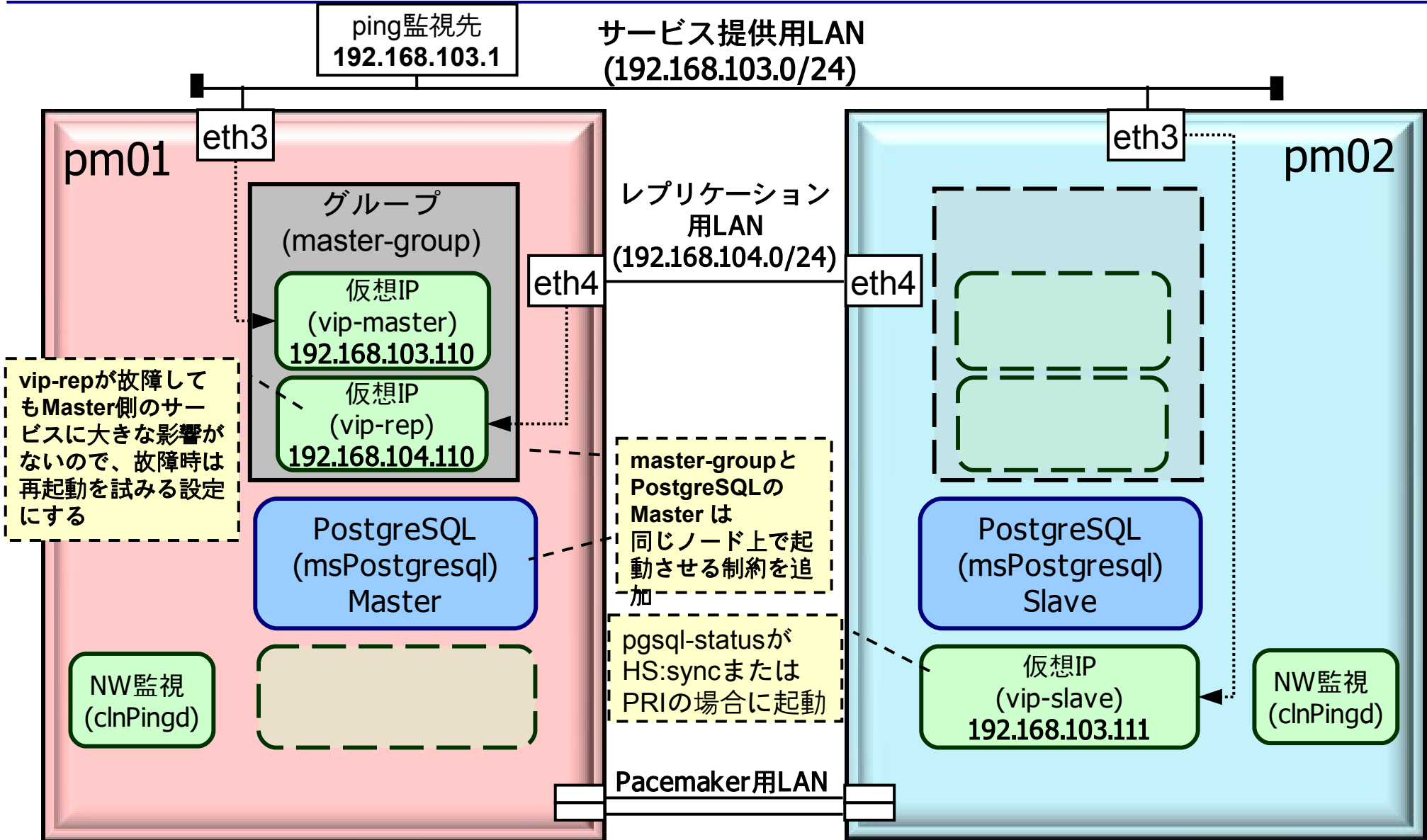


他のノードにPRIがおらず、自分のpgsql-data-statusが最新(LATEST or STREAMING|SYNC) の場合。初期起動時は他のノードとデータ新旧を比較した結果

---

# Pacemakerの設定例

# リソース構成例 (シンプル版)



# Pacemaker 設定例 (シンプル版) 1/2

```
property \  
  no-quorum-policy="ignore" \  
  stonith-enabled="false" \  
  crmd-transition-delay="0s" \  
  
rsc_defaults \  
  resource-stickiness="INFINITY" \  
  migration-threshold="1"
```

```
ms msPostgresql postgresql \  
  meta \  
    master-max="1" \  
    master-node-max="1" \  
    clone-max="2" \  
    clone-node-max="1" \  
    notify="true"
```

← PostgreSQLの  
Master/Slave化設定

```
group master-group \  
  vip-master \  
  vip-rep \  
  meta \  
    ordered="false"
```

← サービス用仮想IPと  
レプリケーション用仮想IPを  
master-groupとしてグループ化

```
clone clnPingd \  
  pingCheck
```

```
primitive vip-master ocf:heartbeat:IPAddr2 \  
  params \  
    ip="192.168.103.110" \  
    nic="eth3" \  
    cidr_netmask="24" \  
  op start timeout="60s" interval="0s" on-fail="restart" \  
  op monitor timeout="60s" interval="10s" on-fail="restart" \  
  op stop timeout="60s" interval="0s" on-fail="block"
```

← サービス用  
仮想IP(vip-master)設定

```
primitive vip-rep ocf:heartbeat:IPAddr2 \  
  params \  
    ip="192.168.104.110" \  
    nic="eth4" \  
    cidr_netmask="24" \  
  meta \  
    migration-threshold="0" \  
  op start timeout="60s" interval="0s" on-fail="stop" \  
  op monitor timeout="60s" interval="10s" on-fail="restart" \  
  op stop timeout="60s" interval="0s" on-fail="block"
```

← レプリケーション用  
仮想IP(vip-rep)設定

```
primitive vip-slave ocf:heartbeat:IPAddr2 \  
  params \  
    ip="192.168.103.111" \  
    nic="eth3" \  
    cidr_netmask="24" \  
  meta \  
    resource-stickiness="1" \  
  op start timeout="60s" interval="0s" on-fail="restart" \  
  op monitor timeout="60s" interval="10s" on-fail="restart" \  
  op stop timeout="60s" interval="0s" on-fail="block"
```

← HS (read only) 用  
仮想IP(vip-slave)設定

```
primitive postgresql ocf:heartbeat:postgresql \  
  params \  
    pgctl="/usr/postgresql-9.1/bin/pg_ctl" \  
    psql="/usr/postgresql-9.1/bin/psql" \  
    pgdata="/var/lib/postgresql/9.1/data/" \  
    rep_mode="sync" \  
    node_list="pm01 pm02" \  
    restore_command="cp /var/lib/postgresql/9.1/data/pg_archive/%f %p" \  
    primary_conninfo_opt="keepalives_idle=60 \  
      keepalives_interval=5 keepalives_count=5" \  
    master_ip="192.168.104.110" \  
    stop_escalate="0" \  
  op start timeout="30s" interval="0s" on-fail="restart" \  
  op stop timeout="30s" interval="0s" on-fail="block" \  
  op monitor timeout="30s" interval="11s" on-fail="restart" \  
  op monitor timeout="30s" interval="10s" on-fail="restart" role="Master" \  
  op promote timeout="30s" interval="0s" on-fail="restart" \  
  op demote timeout="30s" interval="0s" on-fail="block" \  
  op notify timeout="60s" interval="0s"
```

← PostgreSQL  
メイン設定

設定詳細は字が小さくて見えないと思うので、  
資料アップロード後確認してください

# Pacemaker 設定例 (シンプル版) 2/2

```
primitive pingCheck ocf:pacemaker:pingd \  
  params \  
    name="default_ping_set" \  
    host_list="192.168.103.1" \  
    multiplier="100" \  
  op start timeout="60s" interval="0s" on-fail="restart" \  
  op monitor timeout="60s" interval="2s" on-fail="restart" \  
  op stop timeout="60s" interval="0s" on-fail="ignore"
```

### Resource Location ###

```
location rsc_location-1 msPostgresql \  
  rule -inf: not_defined default_ping_set or default_ping_set lt 100
```

```
location rsc_location-2 vip-slave \  
  rule 200: pgsq-status eq HS:sync \  
  rule 100: pgsq-status eq PRI \  
  rule -inf: not_defined pgsq-status \  
  rule -inf: pgsq-status ne HS:sync and pgsq-status ne PRI
```

} **HS (read only) 用仮想IP (vip-slave)  
起動条件の設定**

### Resource Colocation ###

```
colocation rsc_colocation-1 inf: msPostgresql    clnPingd  
colocation rsc_colocation-2 inf: master-group    msPostgresql:Master ← Masterと仮想IPの同居制約設定
```

### Resource Order ###

```
order rsc_order-1 0: clnPingd      msPostgresql    symmetrical=false  
order rsc_order-2 inf: msPostgresql:promote master-group:start symmetrical=false  
order rsc_order-3 0: msPostgresql:demote master-group:stop  symmetrical=false } Masterと仮想IPの起動・停止順番設定
```

商用利用時は、STONITHやディスク監視等、  
その他必要なリソースは別途追加すること

---

重要な点だけピックアップ



# PostgreSQLメイン設定部 ~pgsql RA~

primitive pgsql ocf:heartbeat:pgsql ¥

params ¥

pgctl="/usr/pgsql-9.1/bin/pg\_ctl" ¥

psql="/usr/pgsql-9.1/bin/psql" ¥

pgdata="/var/lib/pgsql/9.1/data/" ¥

} 各コマンドやPostgreSQLデータの場所を指定

rep\_mode="sync" ¥

← 同期レプリケーション使用 (指定しない場合は通常のAct-Stadby動作)

node\_list="pm01 pm02" ¥

← レプリケーションに参加するサーバーの全ホスト名

restore\_command="cp /var/lib/pgsql/9.1/data/pg\_archive/%f %p" ¥

← recovery.conf の restore\_command 設定

primary\_conninfo\_opt="keepalives\_idle=60 ¥

keepalives\_interval=5 keepalives\_count=5" ¥

} recovery.confの primary\_conninfo に追加するオプション

master\_ip="192.168.104.110" ¥

← vip-repのIP

stop\_escalate="0" ¥ ← フェイルオーバーを高速化するために、Master停止時にimmediate shutdownを使用

op start timeout="30s" interval="0s" on-fail="restart" ¥

op stop timeout="30s" interval="0s" on-fail="block" ¥

op monitor timeout="30s" interval="11s" on-fail="restart" ¥

op monitor timeout="30s" interval="10s" on-fail="restart" role="Master" ¥

op promote timeout="30s" interval="0s" on-fail="restart" ¥

op demote timeout="30s" interval="0s" on-fail="block" ¥

op notify timeout="60s" interval="0s" ¥

} monitor(Master時)  
promote, demote  
notifyの動作定義

# 仮想IP vip-slave (ReadOnly用) の起動条件設定部

```
location rsc_location-2 vip-slave \
```

```
rule 200: pgsql-status eq HS:sync \ ← pgsql-status = HS:syncならば起動可
```

```
rule 100: pgsql-status eq PRI \ ← pgsql-status = PRIならば起動可
```

```
rule -inf: not_defined pgsql-status \
```

```
rule -inf: pgsql-status ne HS:sync and pgsql-status ne PRI
```

*HS:syncの方がスコアが大きい (200 > 100)ため、  
正常な同期レプリケーション時はSlave側で、  
それ以外はPRI (Master)側で起動。*

※vip-slaveが起動するノードが固定されないように、resource-stickinessは"1"にしておくこと

```
primitive vip-slave ocf:heartbeat:IPaddr2 \
  params \
    ip="192.168.103.111" \
    nic="eth3" \
    cidr_netmask="24" \
  meta \
    resource-stickiness="1" \
```

# 仮想IP vip-rep (レプリケーション用) の再起動設定

```
primitive vip-rep ocf:heartbeat:IPaddr2 \  
  params \  
    ip="192.168.104.110" \  
    nic="eth4" \  
    cidr_netmask="24" \  
  meta \  
    migration-threshold="0" \  
  op start timeout="60s" interval="0s" on-fail="stop" \  
  op monitor timeout="60s" interval="10s" on-fail="restart" \  
  op stop timeout="60s" interval="0s" on-fail="block"
```

レプリケーション用仮想IPは、監視で故障発生し  
ても  
再起動を試みる設定にする  
※0は回数制限なし

# Masterと仮想IP(vip-master, vip-rep)の起動・停止順番設定部

promote 後に 仮想IP を起動

```
order rsc_order-2 inf: msPostgresql:promote master-group:start symmetrical=false
                                仮想IPを含むグループ
order rsc_order-3 0: msPostgresql:demote master-group:stop symmetrical=false
```

demote(PostgreSQL停止)完了後に仮想IPを停止。  
先に仮想IPを停止すると、レプリケーション接続が切断され、  
レプリケーションの packets が届かなくなるのを防ぐため先にdemote  
する。

# 同期レプリケーション時のPacemaker表示例 (crm\_mon -Af)

Online: [ pm01 pm02 ]

**vip-slave (ocf::heartbeat:IPaddr2):** Started pm02

Resource Group: master-group

**vip-master (ocf::heartbeat:IPaddr2):** Started pm01

**vip-rep (ocf::heartbeat:IPaddr2):** Started pm01

Master/Slave Set: msPostgresql

**Masters:** [ pm01 ]

**Slaves:** [ pm02 ]

Clone Set: clnPingd

Started: [ pm01 pm02 ]

Node Attributes:

\* Node pm01:

+ default\_ping\_set : 100

+ master-pgsql:0 : 1000

+ **pgsql-data-status** : LATEST

+ **pgsql-status** : PRI

+ **pgsql-master-baseline** : 0000000150000B0

+ pm02-eth1 : up

+ pm02-eth2 : up

\* Node pm02:

+ default\_ping\_set : 100

+ master-pgsql:1 : 100

+ **pgsql-data-status** : STREAMING|SYNC

+ **pgsql-status** : HS:sync

+ pm01-eth1 : up

+ pm01-eth2 : up

Migration summary:

\* Node pm01:

\* Node pm02:

} 仮想IPの  
状態

} PostgreSQLの  
Master/Slaveの状態

} pm01ノードの  
pgsql-status,  
pgsql-data-status  
の状態

← promote直前のxlogの位置

} pm02ノードの  
pgsql-status,  
pgsql-data-status  
の状態

---

# PostgreSQLの設定ポイント

# postgresql.conf (重要点のみ)

---

- listen\_address = \*
  - Slaveには仮想IP(vip-master)が存在しないため特定IPでListenはできない
- synchronous\_standby\_names はコメントアウト
  - Pacemakerが同期・非同期を切り替えるためユーザ設定不可
    - RAが/var/lib/pgsql/tmp/rep\_mode.conf をincludeする設定を自動追加
- restart\_after\_crash = off
  - PacemakerがPostgreSQLの状態管理をするため、自動再起動はoff
- replication\_timeout = 20s (ぐらい?)
  - 誤検知しない程度に短くする。Slave故障時にMasterのトランザクションが止まる時間に影響
- hot\_standby = on
  - Slaveを監視するために必須
- max\_standby\_streaming\_delay = -1  
max\_standby\_archive\_delay = -1
  - Slaveの監視クエリがキャンセルされるのを防ぐ

# postgresql.conf (その他)

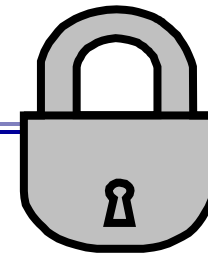
---

- wal\_level = hot\_standby
- wal\_keep\_segments = 64 (ぐらい?)
  - 小さくし過ぎると、レプリケーションに必要なwalがすぐに削除されてしまう
- wal\_receiver\_status\_interval = 5s (ぐらい?)
  - replication\_timeout より短くすること
- hot\_standby\_feedback = on
- archive\_mode = on



---

# ロックファイル



## □PRIを停止すると残るファイル

- データ不整合を避けるために導入
  - ・ 本ファイルが存在するとPostgreSQLは起動できない
  - ・ デフォルトパス：/var/lib/pgsql/tmp/PGSQL.lock
- HSがない場合、PRI正常停止時に削除される

### ★同期レプリケーションの意味★

- コミット成功は必ず両サーバにデータが存在することを保証
- つまりコミット失敗時はデータがコミットされたかどうかは不定
- 最悪両サーバ間でデータに差異が発生
- PostgreSQLはこの差異を検知できないため起動をロック

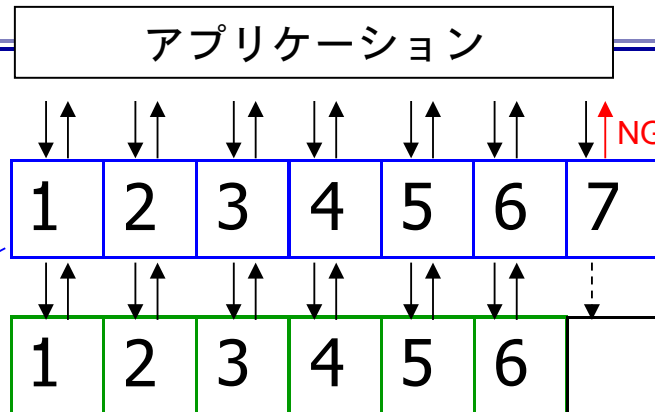
# 不整合の発生パターン例

## 問題が発生するパターン

- PRIに7のデータ書き込み
- PRIが7のデータをHSへ転送中にPRIが故障 (HSにパケット未達)



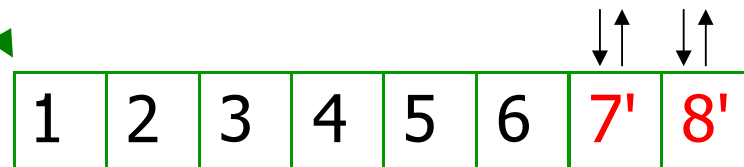
PRI



- フェイルオーバ発生
  - ・ HSは6までのデータでPRIに昇格し7', 8' のデータを書き込み

HS

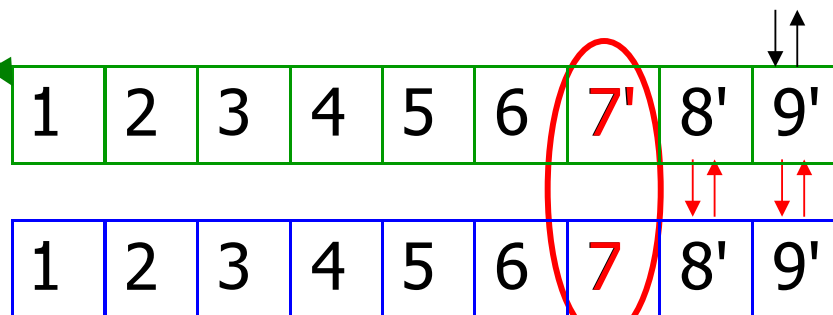
PRI



- 旧PRIをHSとして組み込み
  - ・ 旧PRIには転送し損ねた7のデータが存在するため、8'からレプリケーションが開始される

PRI

HS



不整合発生

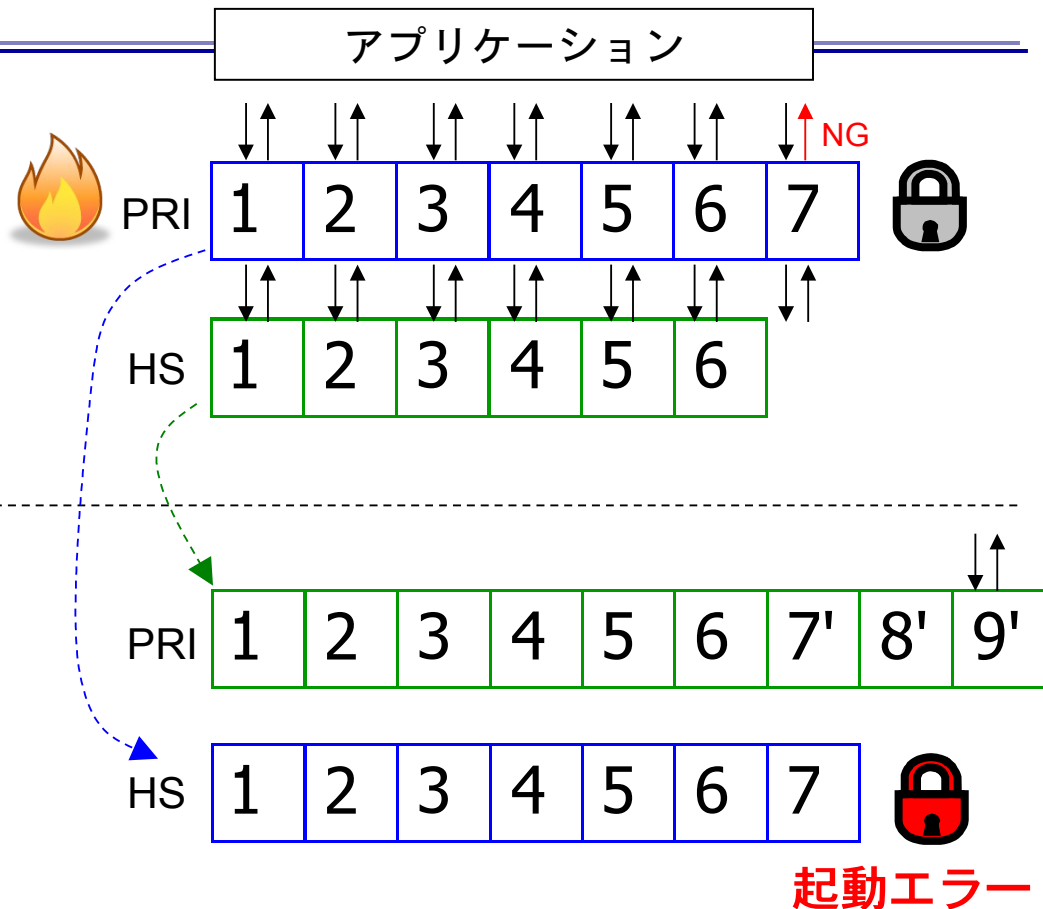
# ロックファイル導入後

## □ 回避パターン

- PRI へ昇格時にロックファイル作成
- PRIに7のデータ書き込み
- PRIが7のデータをHSへ転送中にPRIが故障 (HSにパケット未達)  
→ フェイルオーバー発生 (図省略)

- 旧PRIをHSとして組み込み

→ HS起動時にロックファイルがあるため起動エラーに。



ロックファイルがある場合は、**新PRIのデータを新HSへコピーし**  
ロックファイルを削除する手順が必要

# HAクラスタ化まとめ

---

## □ 3大機能

- Masterのフェイルオーバー
- レプリケーションの同期・非同期の切替
- データの状態管理

## □ Pacemaker設定のポイント

- レプリケーション用の仮想IP(vip-rep)が必要
- 必要ならばSlaveにReadOnlyクエリ負荷分散用仮想IPを設定可能

## □ 運用時の注意

- TimelineIDがずれているとレプリケーションできないため注意
- フェイルオーバー後はTimelineIDのずれに加えデータ不整合も発生する可能性あり
  - ・ ロックファイルがある場合、不整合が発生していないデータに入れ替えること

# 動作環境

---

## □ Pacemaker 1.0.12 以上推奨

- 1.0.11だとMaster/Slaveバグ回避設定が必要

## □ resource-agents 3.9.3 以上

- Linux-HA Japan Pacemakerリポジトリパッケージ 1.0.12-1.2 以上に同梱 (2012年7月リリース)
- 色々バグフィックスしているので、pgsql RAだけ最新に入れ替えた方が無難

## □ PostgreSQL 9.1 以上

- 9.0では動きません
- 9.2では動いたとの報告あり

## □ ソースコード (GitHub上のRA直リンク)

- URL : <https://github.com/ClusterLabs/resource-agents/blob/master/heartbeat/pgsql>

## □ ドキュメントおよび設定例 (GitHubのWiki)

- <https://github.com/t-matsuo/resource-agents/wiki/>

## □ Pacemakerダウンロード・インストール

- <http://linux-ha.sourceforge.jp/>