

Pacemaker + PostgreSQL レプリケーション構成(PG-REX)の 運用性向上 ～スロットの覚醒～

2016年 2月 27日
OSC2016 Tokyo/Spring

Linux-HA Japan
竹下 雄大



本日の内容

- Pacemakerってなに？
- PG-REXってなに？
- レプリケーションスロットの概要
- PG-REXでレプリケーションスロット
- リポジトリパッケージPacemaker-1.1.14-1.1と今後のスケジュール

Pacemakerはオープンソースの
HAクラスタソフトです

Pacemakerってなに？

High **A**vailability = 高可用性
つまり サービス継続性

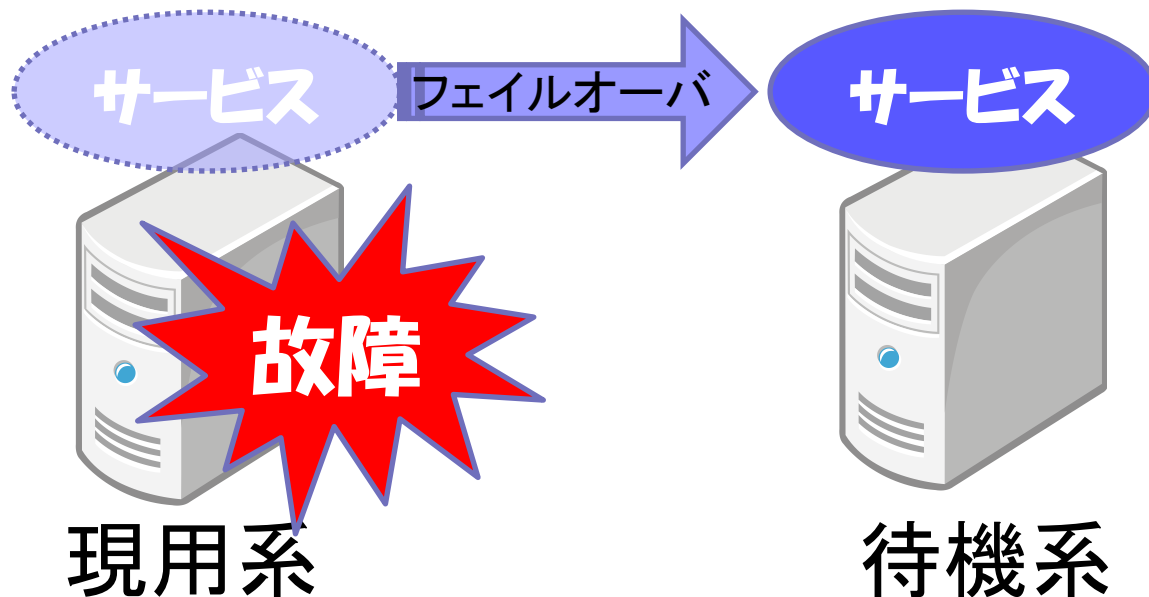
一台のコンピュータでは得られない高い信頼性を得るために、
複数のコンピュータを結合(クラスタ化)し、
ひとまとまりとする...

ためのソフトウェアです

Pacemakerってなに？

HAクラスタを導入すると、
故障で現用系でサービスが運用できなくなったときに、自
動で待機系でサービスを起動させます

→このことを「**フェイルオーバー**」と言います



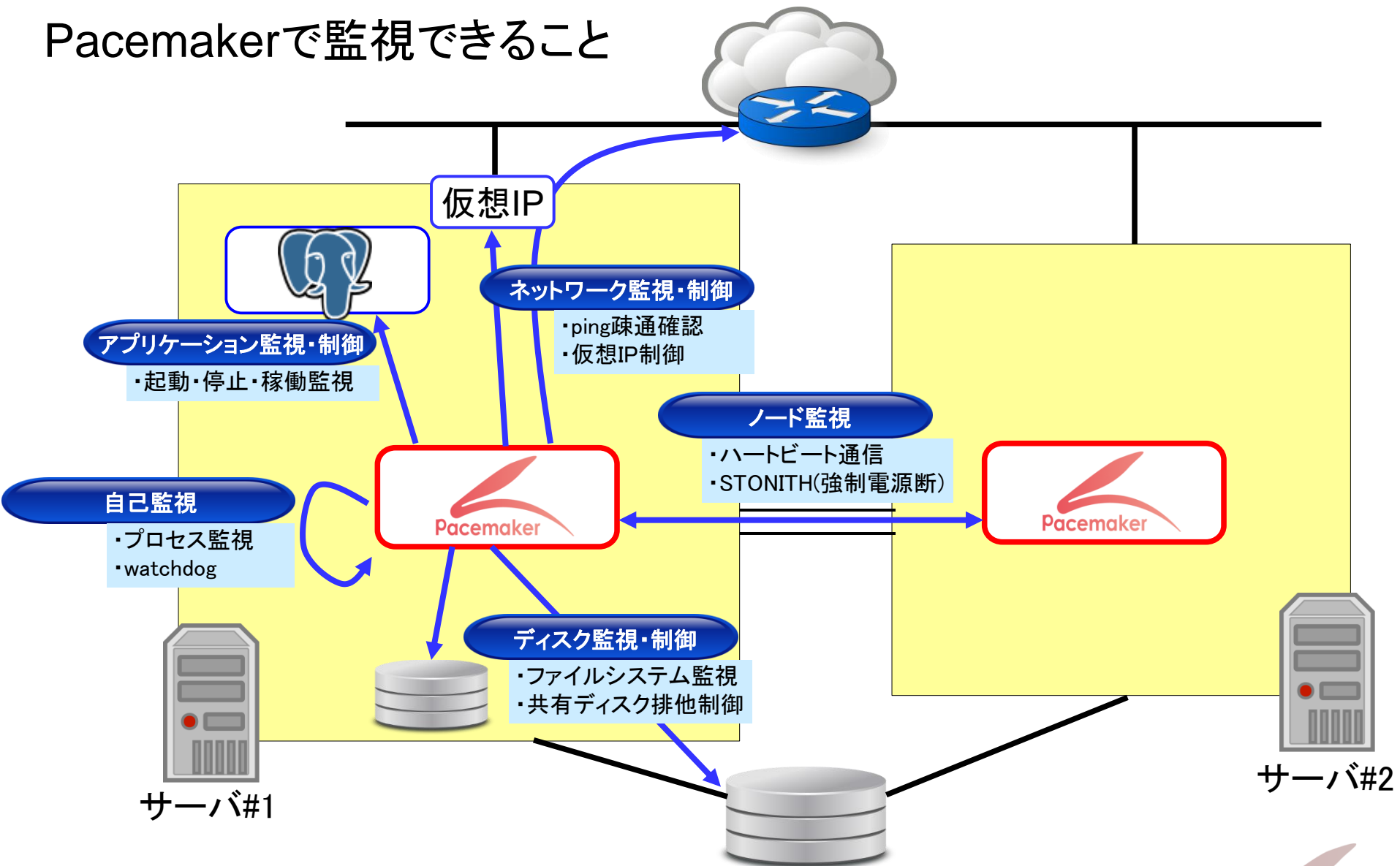
Pacemakerってなに？



Pacemaker は
このHAクラスタソフトとして
実績のある「Heartbeat」と
呼ばれていたソフトの後継です

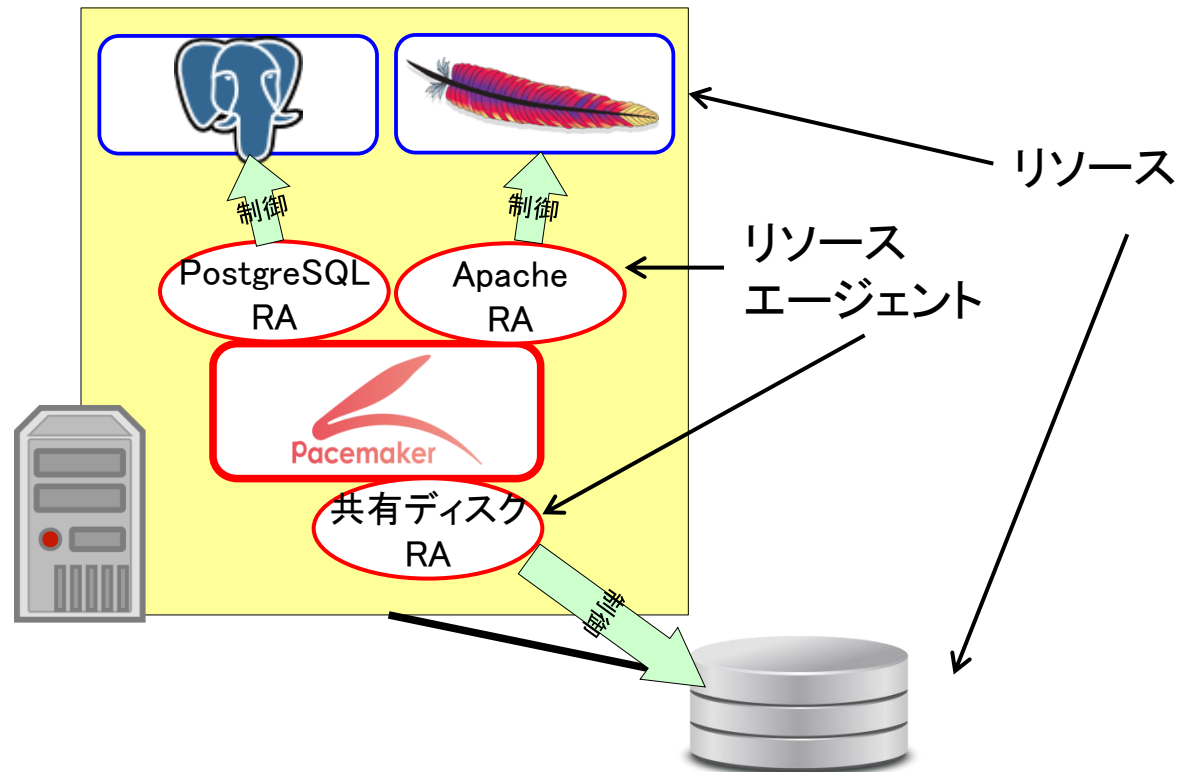
Pacemakerってなに？

Pacemakerで監視できること



Pacemakerってなに？

- Pacemakerが起動/停止/監視を制御する対象を**リソース**と呼ぶ
 - 例: Apache、PostgreSQL、共有ディスク、仮想IPアドレス...
- リソースの制御は**リソースエージェント(RA)**を介して行う
 - RAが各リソースの操作方法の違いをラップし、Pacemakerで制御できるようにしている
 - 多くはシェルスクリプト



Active/Standby と Master/Slave

- Pacemakerでは2つのクラスタ構成をとることができます
- Active/Standby(ACT/SBY)
 - 基本的なクラスタ構成で、ほぼ全てのミドルウェア/アプリケーションで構成可能
 - ミドルウェア/アプリケーションはACTサーバでのみ稼働
 - ACTが故障した場合、SBYサーバがACTに昇格(フェイルオーバー)
 - データは共有ディスク(相当のもの)に保存
- Master/Slave
 - ストリーミングレプリケーションに対応したミドルウェア/アプリケーションでのみ構成可能
 - ミドルウェア/アプリケーションは全サーバで稼働
 - 更新はMasterサーバでのみ可能
 - データは各サーバのローカルディスクに保存
 - ストリーミングレプリケーションにより、MasterからSlaveへ最新データを転送
 - レプリケーションの同期/非同期はミドルウェアに依存

PG-REXってなに？

PG-REXってなに？

□ PG-REXとは・・・

- PostgreSQLのストリーミングレプリケーション機能とPacemakerを組み合わせたMaster/Slave構成による高可用ソリューション

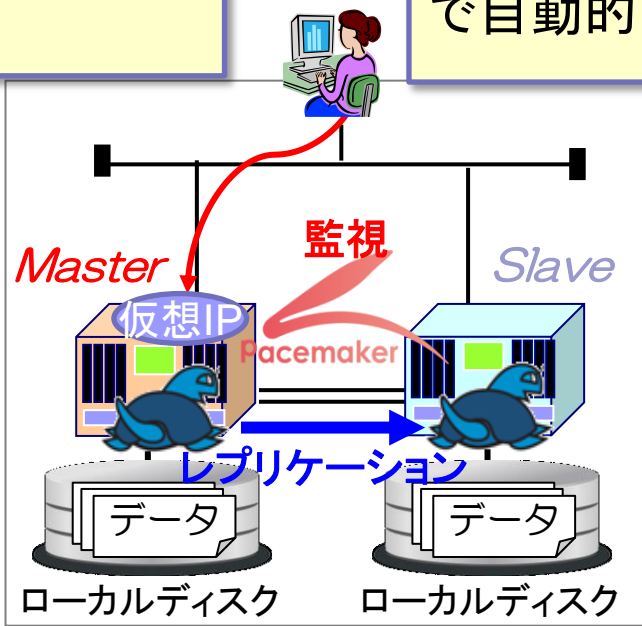


PostgreSQLのストリーミングレプリケーション機能を用いてデータを常に両系にコピー



故障をPacemakerが監視・検知。
SlaveをMasterに昇格させることで自動的にサービスを継続。

- 共有ディスクは不要
- 更新はMaster側のみ可能。
Slaveは参照のみ可能。
- "同期レプリケーション"により、更新データは即座にSlaveに送信される。
(送信後、トランザクション完了)



- Pacemakerは両系で動作。
- 故障時は、SlaveをMasterに昇格後、仮想IPを移動しサービス継続。
- Slave故障時はレプリケーション切り離しを実行。
(トランザクション中断を最小限に)

PG-REXのメリット・デメリット

□ メリット

- ACT/SBY構成と比較し、**コストが安い**
 - 共有ディスク不要、ローカルディスクでOK
- データが2箇所に分散しているため、耐故障性に優れる
 - ただし、バックアップ用途には不適
- 高速なフェイルオーバー
 - Pacemaker-1.1系+PostgreSQL 9.3以降で**30秒**以内！
 - ACT/SBY構成の場合、負荷状況などによって、10分以上かかるケースも・・・
- 参照負荷分散可能
 - 参照系クエリをSlaveで処理可能

□ デメリット

- ACT/SBY構成より性能が低下
 - レプリケーションのオーバーヘッド、ディスクのI/O性能
- 運用がやや難解
 - 2箇所のデータの整合性を考慮
 - 複雑な運用手順
 - 故障時の復旧(クラスタへの再組込み)に時間を要する

PG-REXをもっと使いやすく！

□ 運用性改善への取り組み

- 課題1:2箇所のデータの整合性を考慮

- 対応

- 属性値により、MasterとSlaveの同期状態を確認

- 属性値、ロックファイルによる古いデータでのpromote(Masterへの昇格)抑止

- 課題2:複雑な運用手順

- 対応

- PG-REX運用ツールによるSlave起動の自動化

- pg_basebackupによる全コピー

- アーカイブログの取得

- Pacemakerの起動

} PG-REX運用ツールにより
ワンコマンドで実行

- **pgsql RAにてレプリケーションスロットへ対応**

- **Linux-HA Japanで実装**

NEW!

- 課題3:Slaveの復旧に時間がかかる

- 対応

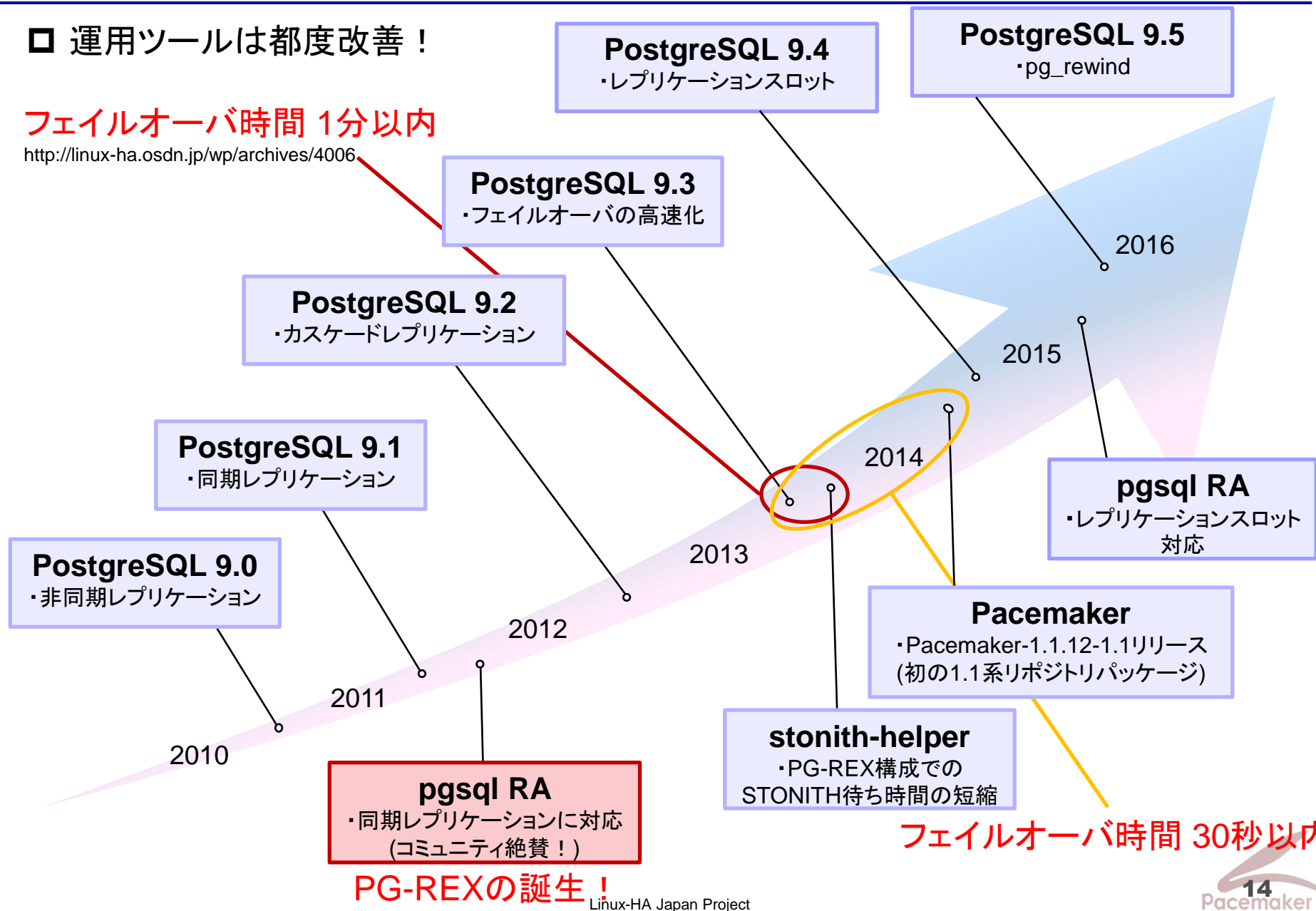
- **pgsql RAにてレプリケーションスロットへ対応**

May the PG-REX be with PostgreSQL !

□ 運用ツールは都度改善 !

フェイルオーバー時間 1分以内

<http://linux-ha.osdn.jp/wp/archives/4006>



レプリケーションスロットの概要

※ちょっとだけPostgreSQLの話

レプリケーションスロットとは

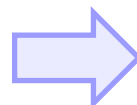
- PostgreSQL 9.4で実装された機能

- Masterの「スロット」にSlaveの最新のWAL位置を保持し、未反映WALの削除を防ぐ

- wal_keep_segmentsの個数を超えて保持する

- これまでは、wal_keep_segmentsの個数までWALを保持
 - 超過分は古いものから削除される

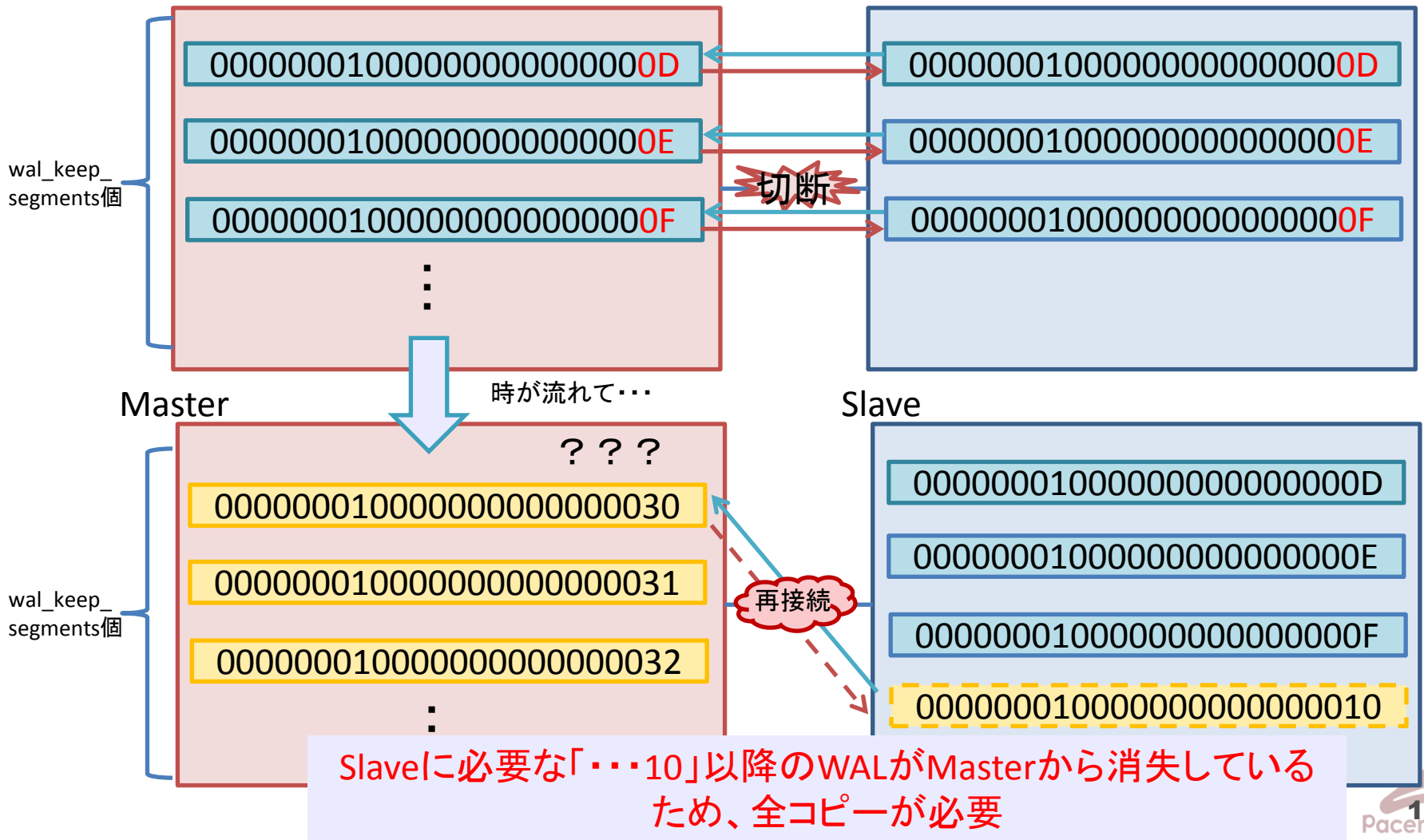
- Slaveの再接続前に、必要なWALが削除されることがなくなる

 **Slaveの再接続時にDBコピーが不要に！**

- Slaveで参照中のトランザクションを管理し、Slaveで参照中のデータが、MasterのVACUUMに伴い削除されてしまうこと（VACUUM処理との競合）を防ぐことができる。

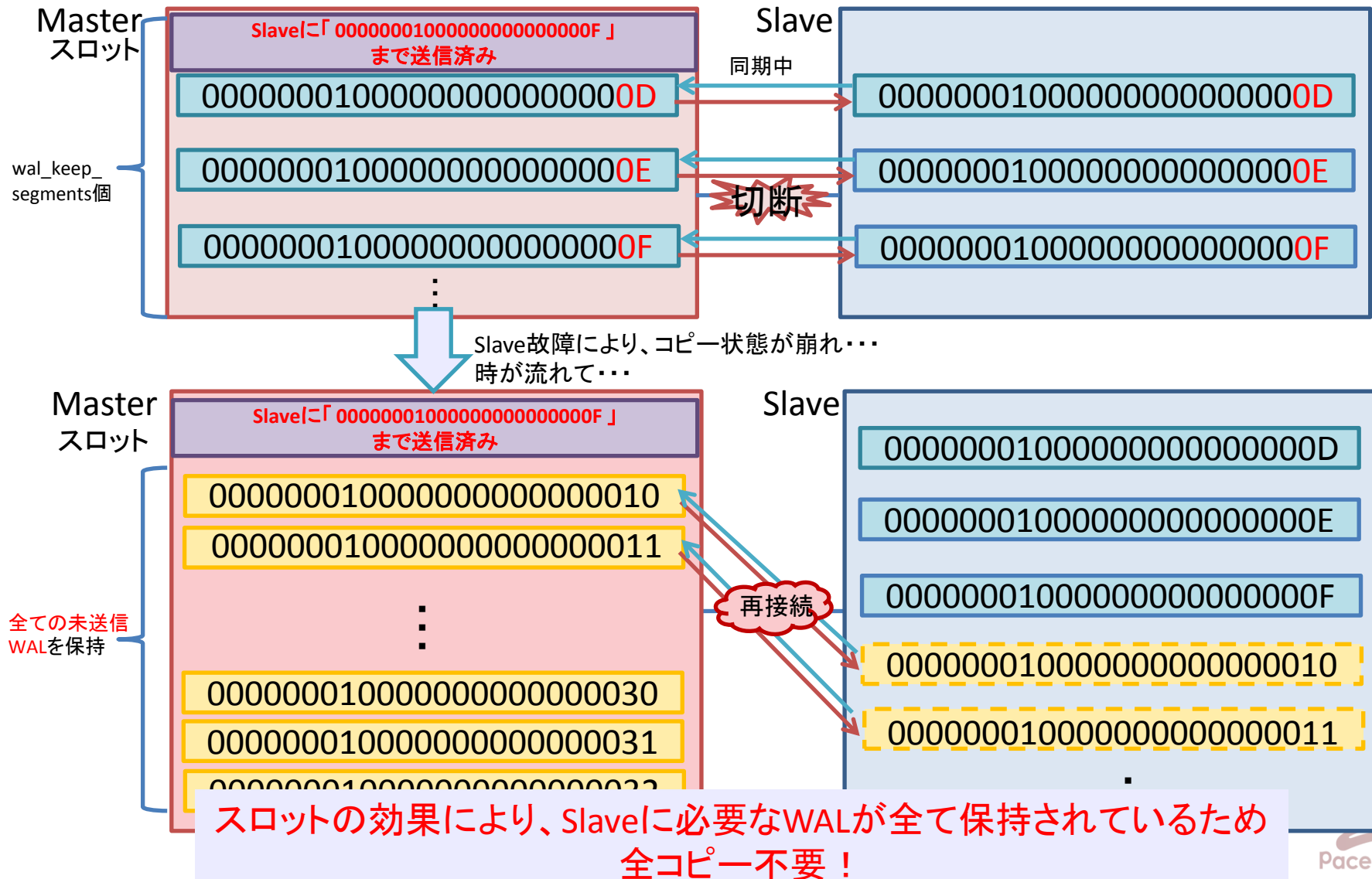
従来のレプリケーション

- 保存されるWALは基本的にはwal_keep_segments個まで
- Slaveの復旧に必要なWALが、Masterノードに存在しない場合、全コピーが必要



レプリケーションスロット利用時

- スロット利用かつSlaveへの未送信WALが存在する場合、当該WALが wal_keep_segments を超えて保持されるため、全コピー不要



レプリケーションスロットの使い方

❑ postgresql.conf (Master)

max_replication_slots = 1 ★1以上の値を設定するとスロットが有効になる

❑ スロットの作成(Master)

```
postgres=# select pg_create_physical_replication_slot('test_slot');
```



任意のスロット名

❑ recovery.conf(Slave)

```
primary_slot_name = 'test_slot' ★追加設定
```

❑ スロットへの接続確認(Master)

```
postgres=# select * from pg_replication_slots;
```

slot_name	plugin	slot_type	datoid	database	active	xmin	catalog_xmin	restart_lsn
test_slot		physical			t			0/3000420

(1 row)

❑ スロットの削除(Master)

```
postgres=# select pg_drop_replication_slot('test_slot');
```

PG-REXでレプリケーションスロット

(再掲)レプリケーションスロットとは

RAに実装すると便利では！？

□ PostgreSQL 9.4で実装された機能

□ Masterの「スロット」にSlaveの最新のWAL位置を保持し、未反映WALの削除を防ぐ

□ wal_keep_segmentsの個数を超えて保持する

- これまでは、wal_keep_segmentsの個数までWALを保持
- 超過分は古いものから削除される

□ Slaveの再接続前に、必要なWALが削除されることがなくなる

➡ Slaveの再接続時にDBコピーが不要に！

□ Slaveで参照中のデータが、Masterで参照中のデータと競合する。
実装しました！！
(resource-agents 3.9.7に含まれます)

スロット利用の効果

□ Slave故障時の復旧手順の簡略化

□ 従来

1. pg_basebackupコマンド等によるDBクラスタとWALの全コピー
 2. アーカイブログファイルの取得
 3. Pacemaker(Slave)を起動
- 時間のかかる処理

□ レプリケーション利用時

1. Pacemaker(Slave)を起動

簡単！

□ Slave故障時の復旧時間の短縮

□ DBクラスタ領域のコピー、アーカイブログファイルの取得が不要のため

- DBクラスタ領域が巨大なほど効果を発揮！

□ wal_keep_segmentsの見積が不要に

- ただし、ディスク容量の見積が必要

レプリケーションスロット利用時のcrm設定例

```
primitive prmPostgresql ocf:heartbeat:pgsql ¥
  params ¥
    pgctl="/usr/pgsql-9.4/bin/pg_ctl" ¥
    start_opt="-p 5432" ¥
    psql="/usr/pgsql-9.4/bin/psql" ¥
    pgdata="/var/lib/pgsql/data" ¥
    pgdba="postgres" ¥
    pgport="5432" ¥
    pgdb="template1" ¥
    rep_mode="sync" ¥
    node_list="pm01 pm02" ¥
    master_ip="192.168.0.70" ¥
    restore_command="/bin/cp /var/lib/pgsql/archive/%f %p" ¥
    repuser="repuser" ¥
    primary_conninfo_opt="keepalives_idle=60 keepalives_interval=5 keepalives_count=5" ¥
    stop_escalate="0" ¥
    xlog_check_count="0" ¥
    replication_slot_name="test" ¥
  op start interval="0s" timeout="300s" on-fail="restart" ¥
  op monitor interval="10s" timeout="60s" on-fail="restart" ¥
  op monitor role="Master" interval="9s" timeout="60s" on-fail="restart" ¥
  op promote interval="0s" timeout="300s" on-fail="restart" ¥
  op demote interval="0s" timeout="300s" on-fail="fence" ¥
  op notify interval="0s" timeout="60s" ¥
  op stop interval="0s" timeout="300s" on-fail="fence"
```

replication_slot_name="test" ¥



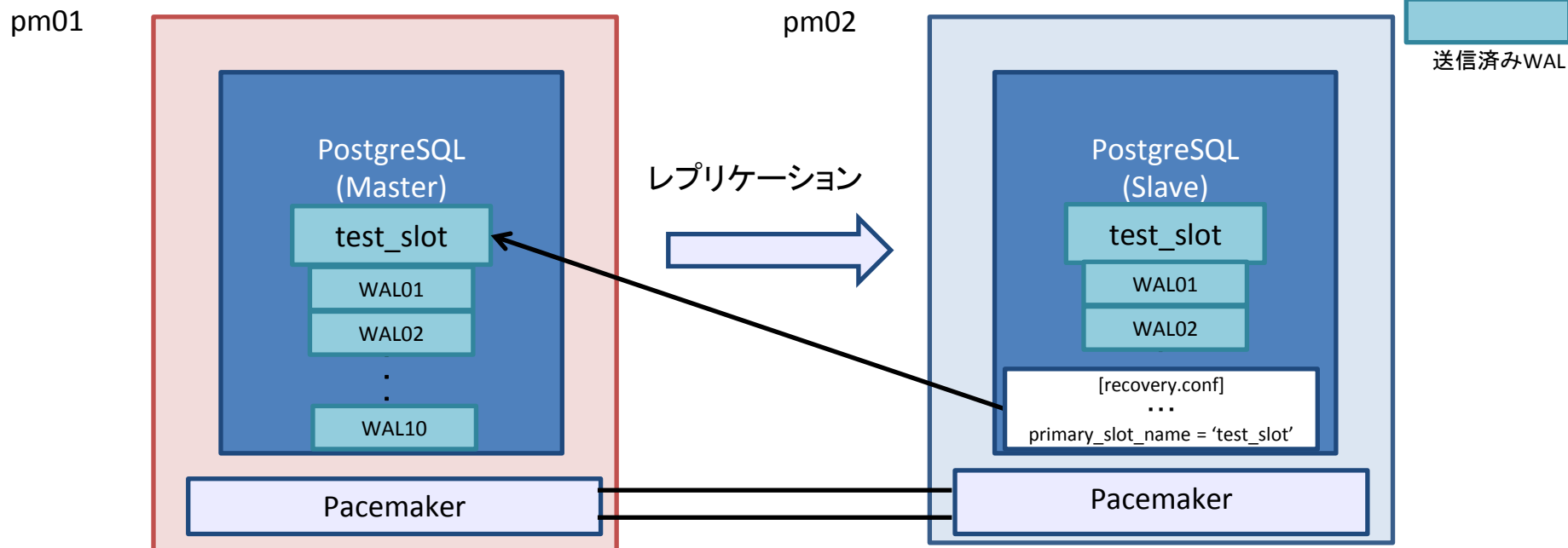
replication_slot_nameを追加するだけ

- ❑ replication_slot_nameに利用できる文字は下記
 - ❑ 英小文字、数字、アンダースコア
- ❑ resource-agents 3.9.7以降で使用可能
 - ❑ Pacemaker-1.1.14-1.1のリポジトリパッケージ同梱予定

スロット利用イメージ(1/3)

正常時

- Masterを起動
 - `$replication_slot_name`のスロットが作成される(既に同名のスロットが存在する場合は初期化する)
- Slaveを起動
 - 同様にスロットが作成される(※)
 - `recovery.conf`へ「`primary_slot_name = $replication_slot_name`」が追記され、Masterのスロットへ接続
 - 以降、MasterのスロットにSlaveの最新のWAL位置が記録される



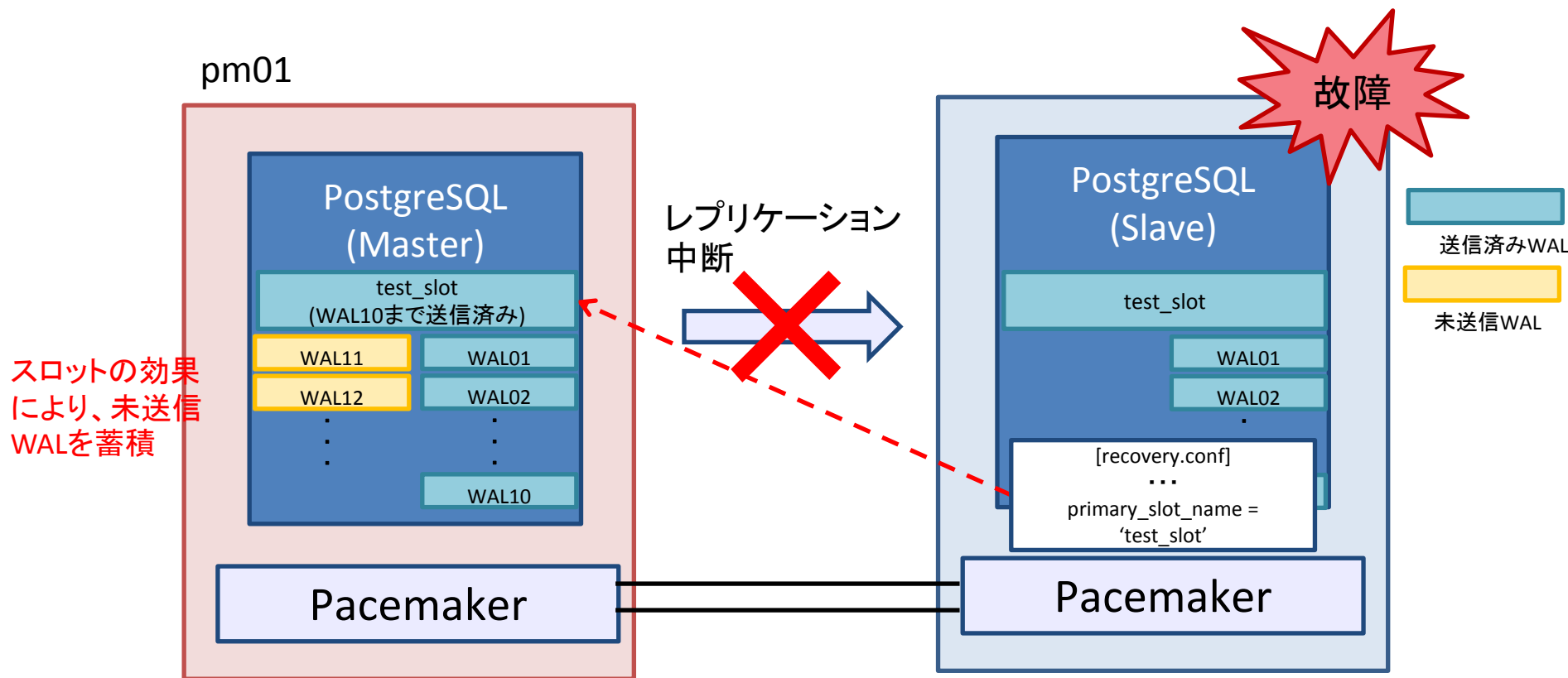
- 正常時は、MasterとSlaveの差分がないため、Masterに余分なWALの蓄積はない
- Masterのスロットは常に最新のWAL位置で更新される

※ 実装の簡易化と将来の拡張(カスケードレプリケーション対応)の可能性を考慮し、Slaveでもスロットを作成している

スロット利用イメージ(2/3)

Slave故障発生時

- PostgreSQL(Slave)プロセス故障、レプリケーションLAN故障、Slave電源断など

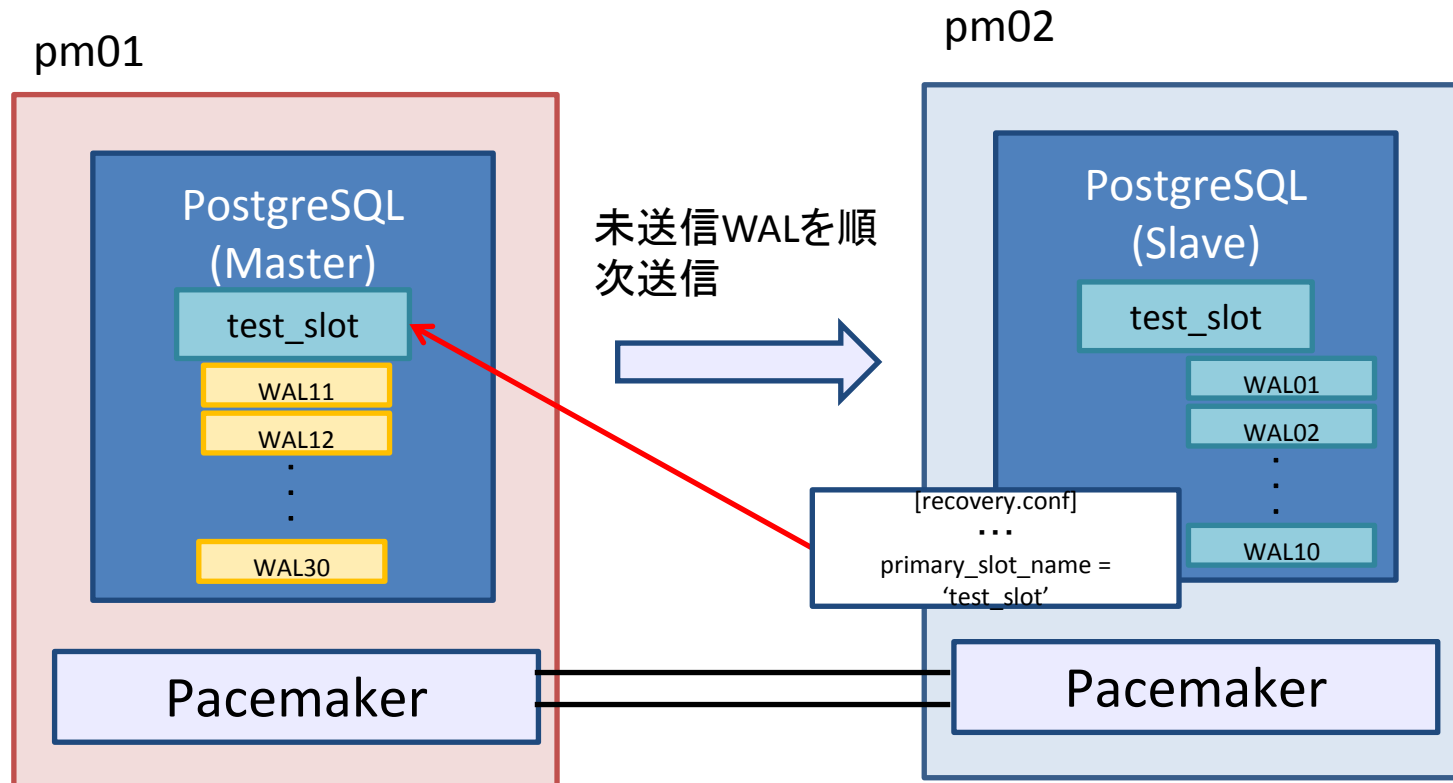


- レプリケーションが中断されることで、MasterとSlaveに差分が発生
 - スロットの効果により以降の差分WALが蓄積される(Slaveが再接続されるまで)

スロット利用イメージ(3/3)

Slave復旧時

- ❑ Slaveの故障原因を取り除く
- ❑ Slaveを起動
 - ❑ primary_slot_nameに設定されたMasterのスロットへ接続
 - ❑ Masterはスロットに蓄積された未送信WALをSlaveへ順次送信
 - ❑ Masterのスロットへ蓄積された(wal_keep_segmentsを超過した)WALは、Slaveへ送信後、CHECKPOINTのタイミングで削除される



Slaveを起動するだけでクラスタへの復旧完了！

Slaveのノード数による動作の違い

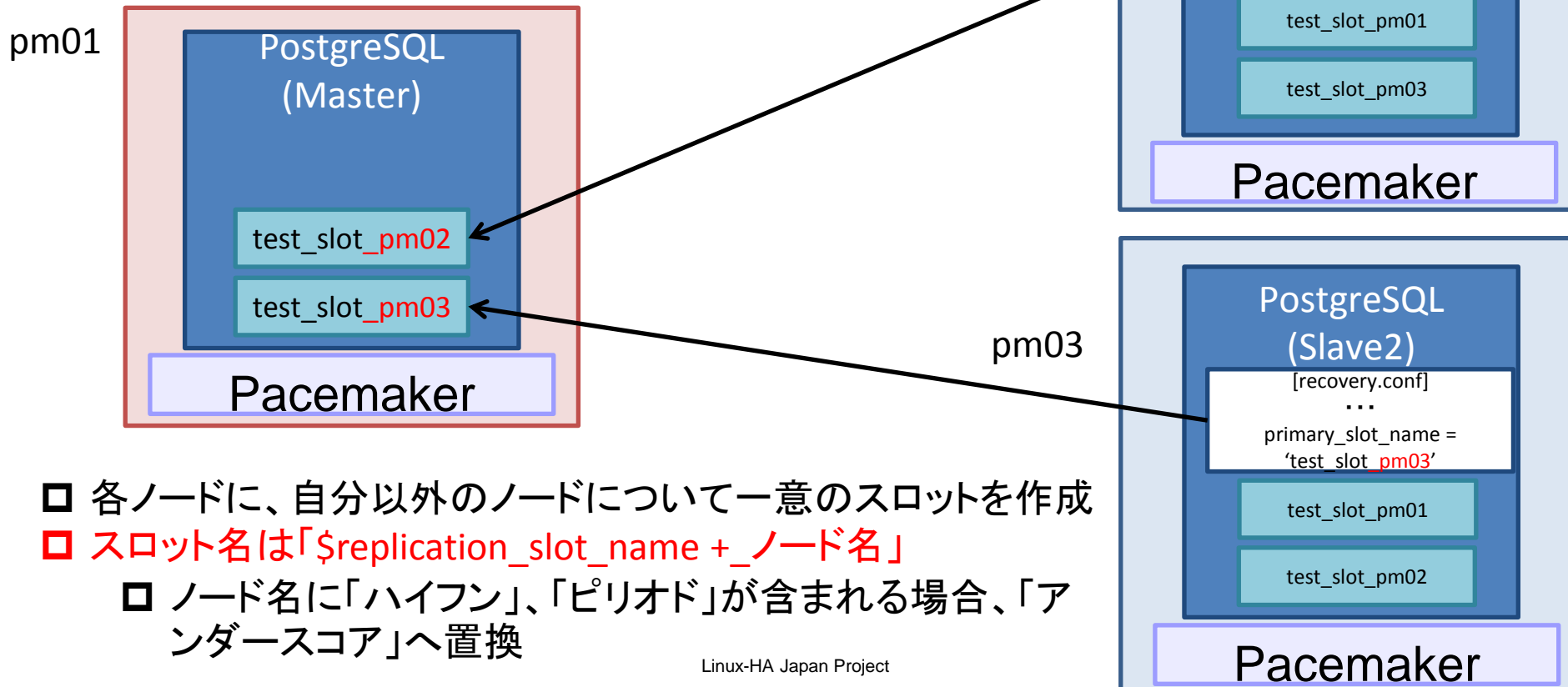
❑ Slaveが1つの場合と複数の場合で動作が異なる

❑ Slaveが1つ(node_listが2つ)の場合

- ❑ 前述の通り

❑ Slaveが2つ以上(node_listが3つ以上)の場合

- ❑ node_list="pm01 pm02 pm03"



❑ 各ノードに、自分以外のノードについて一意のスロットを作成

❑ スロット名は「\$replication_slot_name + ノード名」

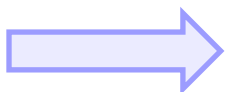
- ❑ ノード名に「ハイフン」、「ピリオド」が含まれる場合、「アンダースコア」へ置換

スロット利用時の注意点(1/2)

1. ディスク容量に注意すること

- ❑ Pacemaker(pgsql RA)はスロットの監視、削除を行わない
- ❑ Slave故障後、Slaveが復旧するまでMasterに未反映WALがたまり続ける
 - ❑ 上限はディスク容量！
 - ❑ **Slave復旧の遅れによりMasterのディスク溢れが発生する可能性あり**
 - ❑ 以下の場合にスロットのWALが削除される

1. 当該スロットにSlaveノードが接続した後のCHECKPOINT
2. 当該スロットを削除した後のCHECKPOINT



運用者は速やかにSlaveの復旧 or スロットの削除を！

RAが監視を行っていないため、オンラインでスロット削除可能

スロット削除コマンド

```
postgres=# select pg_drop_replication_slot('スロット名');
```

2. RAが作成するSlaveのスロットは使わないでください

- ❑ SlaveのスロットにPostgreSQLを一度接続し切断すると、以降WALがたまり続ける
 - ❑ 以降そのスロットには誰も接続しないため、**Slaveでディスク溢れ発生**
 - ❑ 接続した場合は上記「1」と同様の対処を！

スロット利用時の注意点(2/2)

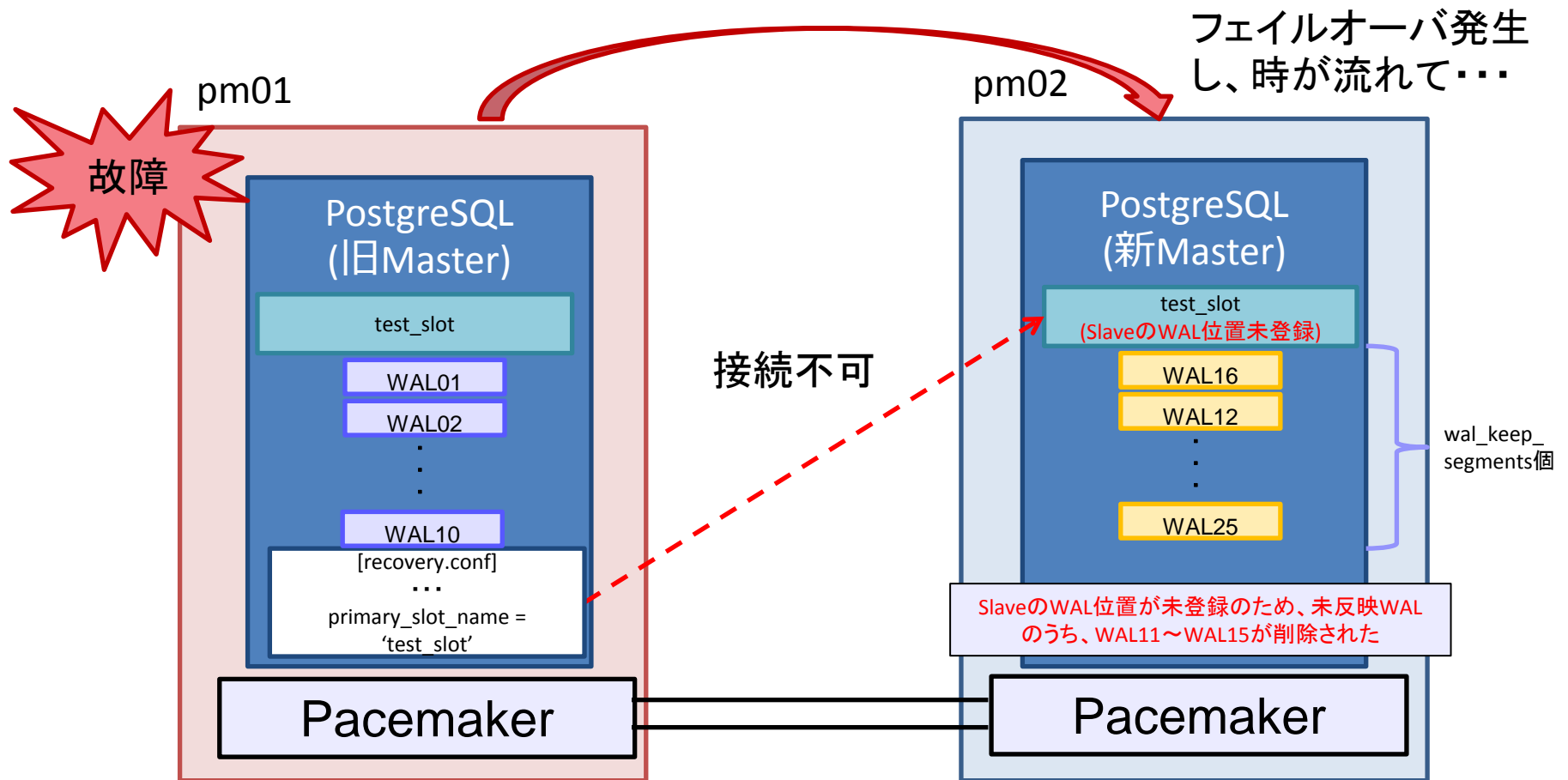
3. Slave停止後、続けてMasterを停止した場合は全コピーが必要

- 次回Master起動時に、既存のスロットが初期化される
 - Slaveの最新WAL位置の情報が失われる
- もし、起動時にスロットを初期化しなければ・・・
 - Masterがフェイルオーバーし、次回Slaveとして起動した際に、スロットにMaster停止時点のSlaveのWAL情報が残ったままとなる
 - SlaveとなったMasterIには誰も接続しないため、以降のWALが蓄積され続ける



旧Masterのディスク溢れが発生！

Master故障(フェイルオーバー)が発生した場合は・・・？



- ❑ Master故障後は、従来通り全コピーが必要
- ❑ 新Masterのスロットに、Slave(旧Master)のWAL位置の情報が存在しないため、新MasterはSlave未反映のWALを蓄積しない

➡ PostgreSQL 9.5の新機能「**pg_rewind**」が使えるかも??

レプリケーションスロット対応PG-REXのまとめ

□ メリット

□ Slave故障時の復旧手順、復旧時間の短縮

- データの全コピー不要！
- Pacemakerを起動するだけ！

□ wal_keep_segmentの見積が不要に

- ディスク容量の見積に代わる

□ デメリット

□ 復旧遅れによるディスク溢れが発生しうる

- 運用者による速やかなSlaveの復旧、または手動でのスロット削除を行う
- スロット削除した場合には従来通りの復旧手順

□ 使い方

□ PostgreSQLのリソース定義にreplication_slot_nameを追加

- 使える文字は英小文字、数字、アンダースコア
- ノード数により、スロット作成の動作が異なる

□ その他

□ Master故障は従来通り

- PostgreSQL 9.5(pg_rewind)に期待！

リポジトリパッケージPacemaker-1.1.14 と今後のスケジュール

リポジトリパッケージPacemaker-1.1.14-1.1と今後のスケジュール(1/3)

❑ Pacemaker-1.1.14はリリース済み(2016/1/15)

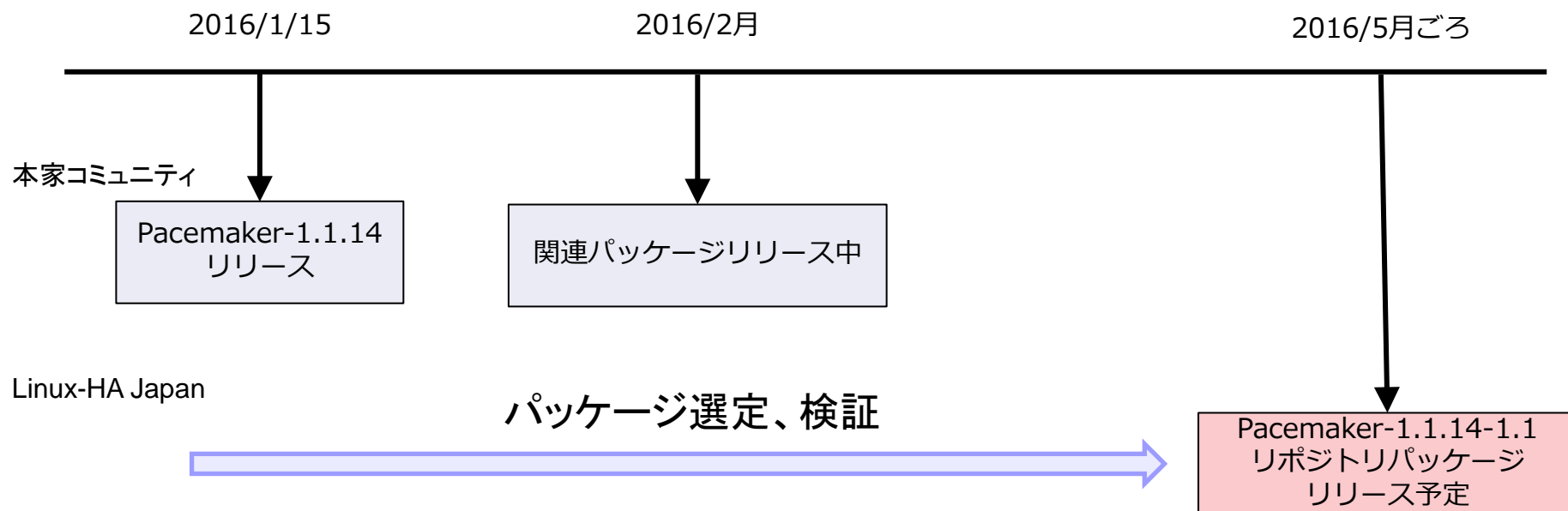
❑ 関連コンポーネントも続々リリース

❑ corosync 2.3.5

❑ resource-agents 3.9.7

❑ crmsh 2.1.5/2.2.0

など



□ Pacemaker-1.1.14-1.1の主な変更点



1. ログメッセージの簡易化

- syslog経由でログを出力する場合に、ログに関数名が含まれなくなった
- **syslog経由で出力したログを運用管理ツール等で監視している場合は影響有無をご確認ください！**
- pm_logconvは本変更に対応予定のため、pm_logconvのログを監視している方は影響なし

★Pacemaker-1.1.13

Jul 31 10:45:44 rhel66-1 crmd[3960]: info: **do_lrm_rsc_op**: Performing key=5:1:7:747cb3f1-2c48-44cb-9453-f7234bdd7a07
op=prmExPostgreSQLDB_monitor_0

★Pacemaker-1.1.14

Dec 15 10:19:03 rhel72-1 crmd[7593]: info: Performing key=3:0:7:6b2fe23a-8c9f-4846-95db-577921ec7125 op=prmExPostgreSQLDB_monitor_0

関数名が削除

- 関数名を出力したい場合は、syslog経由ではなく、ファイルに直接出力する
- デフォルトは/var/log/pacemaker.log

2. AWS EC2用 STONITH(強制電源断)プラグイン

- ec2プラグインによって、AWSインスタンスの強制電源断が可能に！

すぐに使い方は下記から入手可能

<http://hg.linux-ha.org/glue/file/56f40ec5d37e/lib/plugins/stonith/external/ec2>

Linux-HA Japan Project

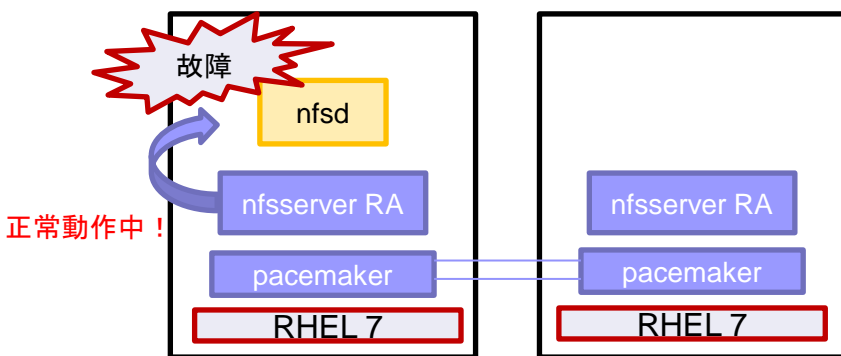
□ Pacemaker-1.1.14-1.1の主な変更点

3. nfsserver RAの改善

- RHEL 7上でNFSサーバをHA構成で運用する際にnfsdプロセス故障を検知できない課題を改善

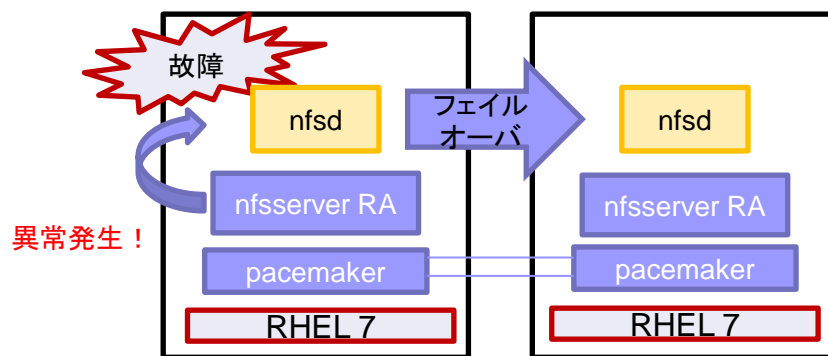
- RHEL 6では問題なし

Pacemaker-1.1.13-1.1



- サービス停止
- ログも出ないのでサイレント故障

Pacemaker-1.1.14-1.1



- フェイルオーバーによりサービス継続!

すぐに使いたい方は下記から入手可能

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

さいごに

Linux-HA Japan URL

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

<http://osdn.jp/projects/linux-ha/>



The screenshot shows the Linux-HA Japan Project website. At the top is the logo with the text "LINUX-HA JAPAN High-Availability Clustering on Linux". Below the logo is a navigation bar with links: HOME, メーリングリスト, ダウンロード&インストール, マニュアル, デスクトップテーマ・壁纸等, コミュニティ概要. Below the navigation bar is a section titled "Linux-HA Japan プロジェクト" with social media links and a "Check" button. The main content area has a table with links to various resources:

Linux-HA Japan 成果物ダウンロード	RHEL/CentOS向けPacemaker RPM/パッケージ(yumのリポジトリ形式)や設定ファイル(crm)作成支援ツール、ディスク監視機能などをダウンロードできます。とりえずRHELもしくはCentOS等のRHEL互換OSにインストールしてみたい場合はこちら。インストール後にとりえず何か動かしてみたい場合はこちらを参考してみてください。
マニュアル	本家コミュニティ提供の公式マニュアルやLinux-HA Japan提供の翻訳マニュアル。マニュアル読んでもよくわからない場合は、過去のカンファレンスや勉強会等の発表資料も参考に。
メーリングリスト	インストール方法や設定方法等の質問はMLまで。 ※投稿するにはメールアドレスの登録が必要です。
イベント情報	カンファレンスへの出席や講演、勉強会開催情報、講演時のスライド公開など。
開発者向けサイト	Linux-HA Japan開発者向けサイトです。Linux-HA Japan独自開発機能のソースコードやバイナリのダウンロード等。

At the bottom, there is a Twitter link: Twitter 公式ハッシュタグ #linux_ha_jp. A footer note says: "本サイトに関するお問い合わせは、Linux-HA Japan メーリングリスト管理者 (linux-ha-japan-owner アット lists.sourceforge.jp) までお願いいたします。"

Pacemaker関連の最新情報を 日本語で発信

Pacemakerのダウンロードもこ ちらからどうぞ (インストールが楽なリポジトリパッケージ を公開しています)

日本におけるHAクラスタについての活発な意見交換の場として「Linux-HA Japan 日本語メーリングリスト」も開設しています。

Linux-HA-Japan MLでは、Pacemaker、Heartbeat3、Corosync DRBDなど、HAクラスタに関連する話題は歓迎！

- ・ ML登録用URL

<http://linux-ha.osdn.jp/>
の「メーリングリスト」をクリック



- ・ MLアドレス

linux-ha-japan@lists.osdn.me

※スパム防止のために、登録者以外の投稿は許可制です

ご清聴ありがとうございました。
May the Pacemaker be with you !



Linux-HA Japan

検索

