HAクラスタでできること！ Pacemakerの構築運用に役立つノウハウを紹介！

2016年 7月 30日
OSC2016 Kyoto

Linux-HA Japan
平田　和照

テーマ

- Pacemaker-1.1を味わうための“便利”な使い方
  ～保守運用に活用しよう～
- Pacemakerで対応する“故障”ケースの起こし方と復旧手順
  ～事前に動作検証しよう～
- 実際の構築運用シーンで起きる問題の“解決”方法
  ～よくある問題を理解しよう～

資料の構成内容

［イントロ］HAクラスタ、Pacemakerの概要

［テーマ１］保守運用の基本

１．Pacemakerの２つのツール紹介
２．故障発生ケースの一例（２つのツールの使い方）
３．復旧手順の流れ

［テーマ２］動作検証、復旧手順の基本

１．Pacemakerによる監視／制御と故障ケース（Pacemaker動作３パターン）
２．復旧手順の整理（３パターン）
３．各故障ケースの実例（６パターン）
① 発生手順イメージ
② 発生手順
③ 故障発生時の動作
④ pm_logconvのログ確認
⑤ 復旧手順

［テーマ３］よくある問題の実例

［付録］

【イントロ】

なぜHAクラスタが必要なのか？

Pacemakerは何ができるのか？

［イントロ］HAクラスタ、Pacemakerの概要

商用システムには何が必要か？

止められない システムの増加

インターネットを使用したビジネスの普及により、24時間365日、止まらないことを要求されるミッションクリティカルなシステムが増加している。

しかし、、、

障害はいつ 起きるか分からない

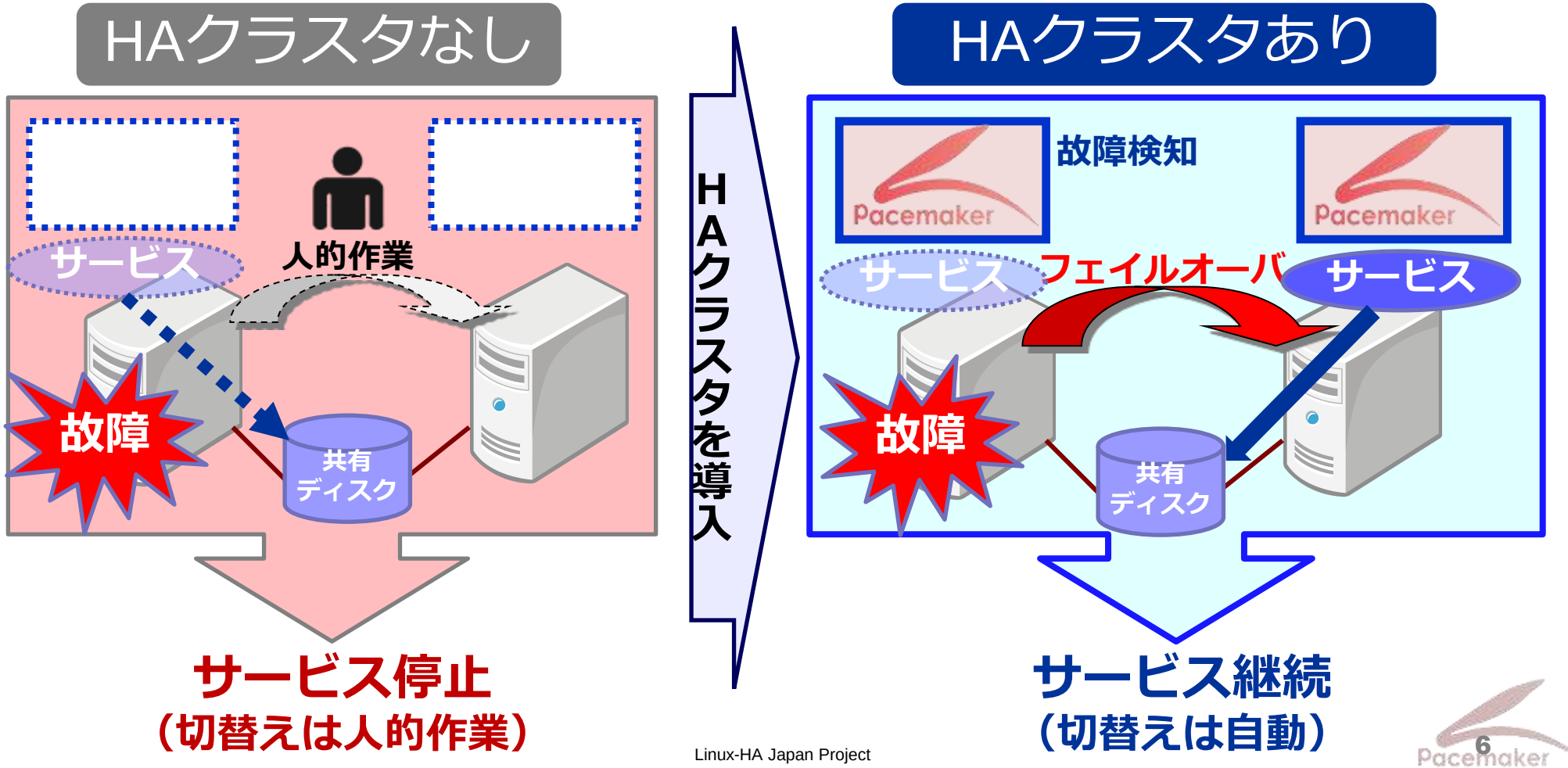
ネットワークやハードウェアの故障、ソフトウェア不具合により、システム停止に繋がる障害が発生。その結果、サービス中断に留まらず、収益の損失や信用の失墜を招く恐れがある。

サービス継続性 向上が必要

システム停止時間を最小限に抑えて、サービス継続性を向上する仕組みが必要。

HAクラスタはなぜ必要か？

HAクラスタを導入することで、システムに故障が発生した時に検知し、サービスを自動で切り替えて、継続することが可能になる。
この仕組みは「**フェイルオーバ (FO)**」と呼ばれる。



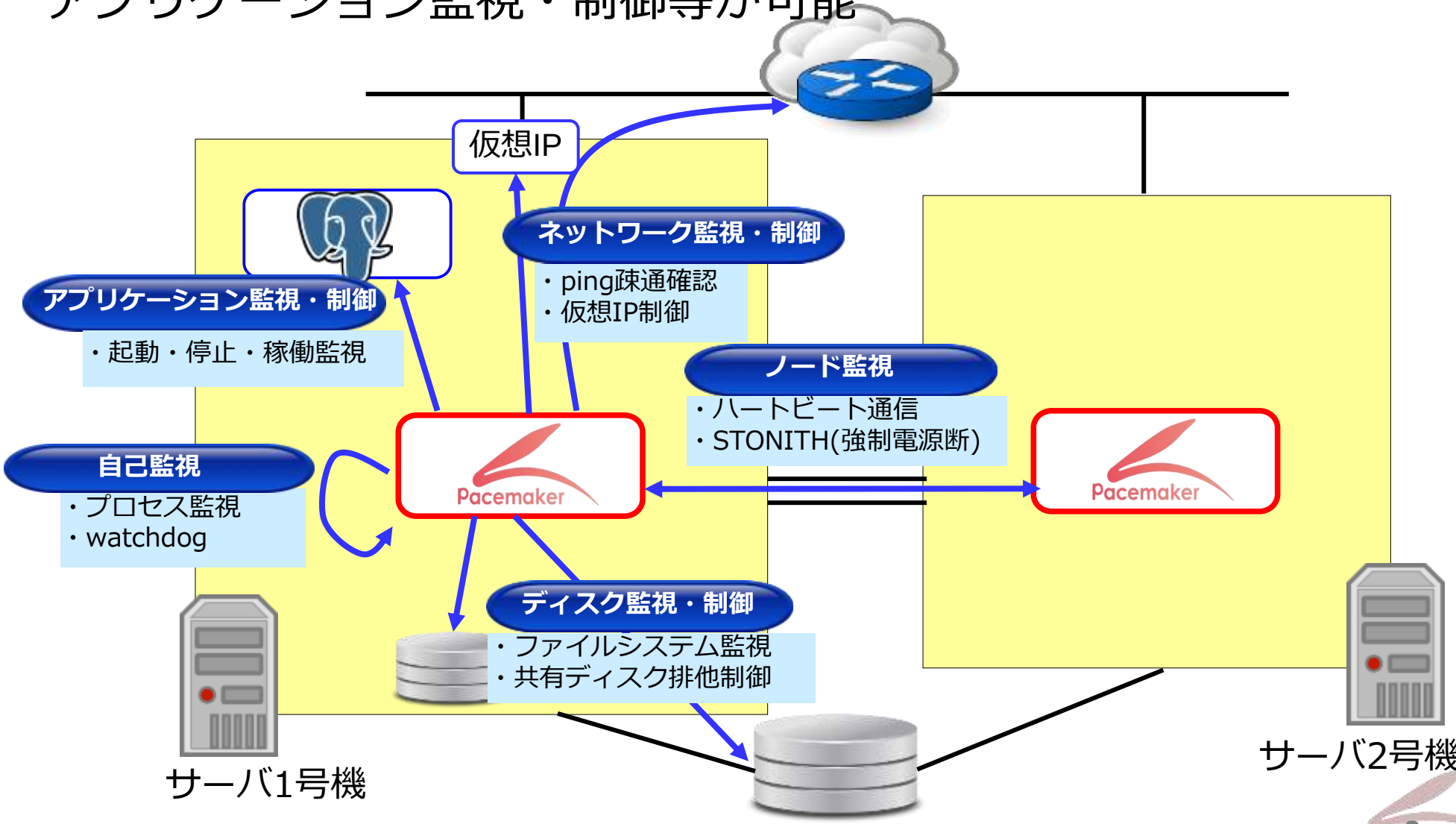
HAクラスタソフトといえば・・・・

は、

複数サーバで冗長構成されたシステム環境において、
故障時や保守時の切り替え制御を行い、
システムの可用性(システム稼働率)を向上させる
オープンソースのHAクラスタソフトである。

「Pacemaker」ができること

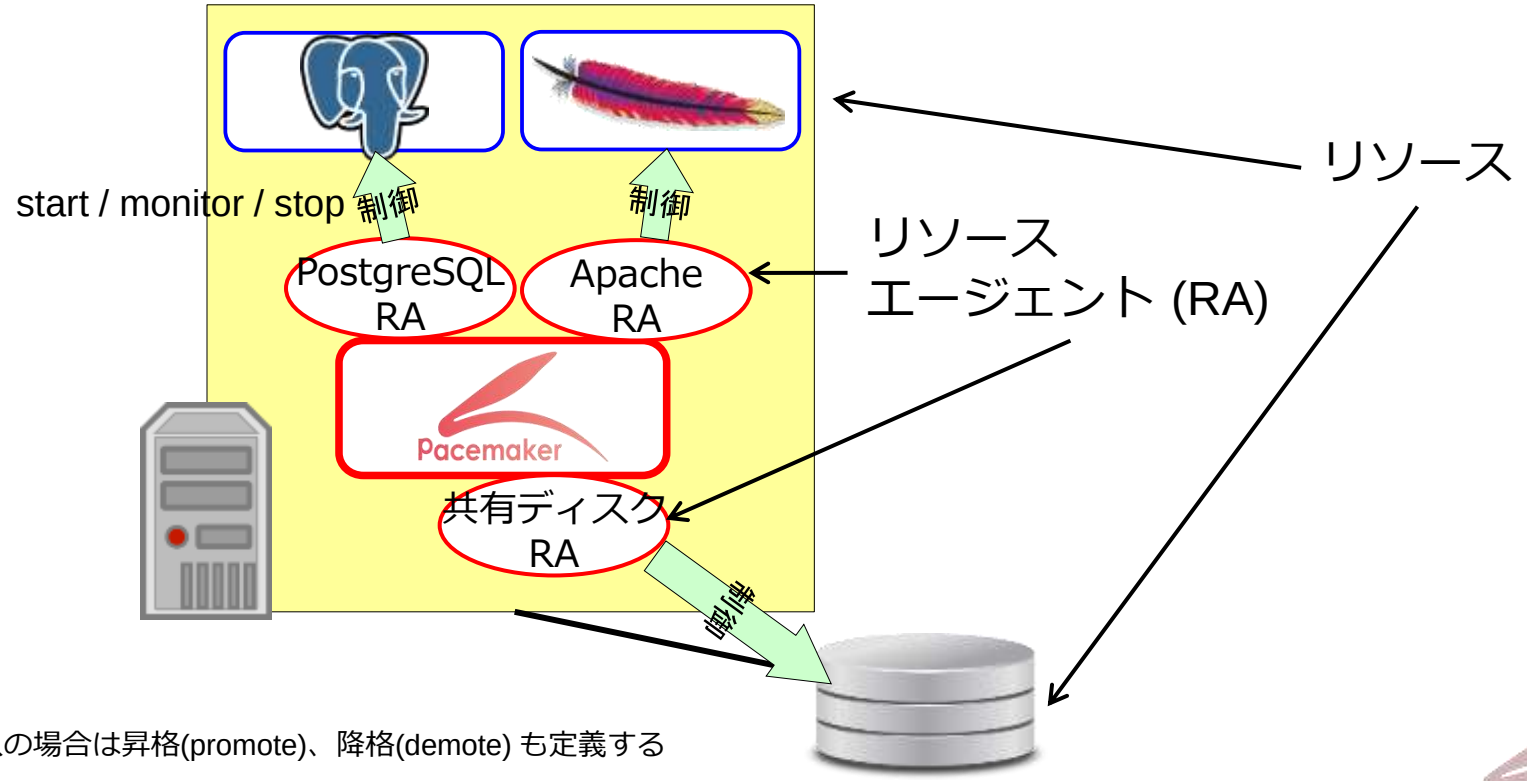
- ノード監視、ネットワーク監視・制御、ディスク監視・制御、アプリケーション監視・制御等が可能



「Pacemaker」の監視/制御の仕組み

- 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 等

- 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で制御可能としている
  - ✓ リソースの 起動(start)、監視(monitor)、停止(stop) を行うメソッドを定義する※



※ Master/Slaveリソースの場合は昇格(promote)、降格(demote) も定義する

【テーマ１】

Pacemaker-1.1を味わうための“便利”な使い方

～保守運用に活用しよう～

［テーマ１］保守運用の基本

１．Pacemakerの２つのツール紹介
２．故障発生ケースの一例（２つのツールの使い方）
３．復旧手順の流れ

１．Pacemakerの２つのツール紹介

Pacemakerを利用したシステムの保守運用に役立つツールを紹介します！

Pacemakerで保守運用を行う

Pacemakerを利用したクラスタシステムの保守運用に活用する２つのツールがあります。

クラスタの状態監視

リアルタイムに各リソースの起動状態などを確認

Pacemakerの監視コマンド
crm_mon

クラスタのログ確認

運用中のリソース起動停止やフェイルオーバの状況を確認

Pacemakerの動作ログ
pm_logconv

- pm_logconvは、Linux-HA Japanのリポジトリパッケージでのみ提供されています。

（その１）Pacemakerの監視コマンド「crm_mon」

- Pacemakerの「**crm_monコマンド**」を用いることで、リアルタイムでクラスタシステムの状態を確認できる。

srv01：サーバ1号機
srv02：サーバ2号機

```
Current DC: srv01 - partition with quorum  ①
    :  （省略）
Online: [ srv01 srv02 ]  ②
 Resource Group: grpPostgreSQLDB
     prmExPostgreSQLDB   (ocf::heartbeat:sfex):          Started srv01
     prmFsPostgreSQLDB   (ocf::heartbeat:Filesystem):    Started srv01
     prmIpPostgreSQLDB   (ocf::heartbeat:IPaddr2):       Started srv01
     prmApPostgreSQLDB   (ocf::heartbeat:pgsql):         Started srv01
    :  （省略）  ③
Node Attributes:
* Node srv01:
    + default_ping_set                 : 100
    + diskcheck_status                 : normal
* Node srv02:
    + default_ping_set                 : 100
    + diskcheck_status                 : normal
    :  （省略）  ④
Migration summary:
* Node srv01:
* Node srv02:  ⑤
Failed actions:  ⑥
Negative location constraints:  ⑦
rsc_location-grpStonith2-1-rule prevents grpStonith2 from running on srv02
rsc_location-grpStonith1-2-rule prevents grpStonith1 from running on srv01
```

両系ノードが正常起動

srv01でサービス起動

ネットワークやディスク監視は正常

リソース故障は発生していない

① **Quorum情報表示部**
QuorumやDCノード状態(*1)

② **ノード情報**
ノードのクラスタ参加状態(Online、OFFLINE)

③ **リソース情報**
リソースの各ノードでの稼働状態

④ **属性情報**
各ノードにおけるネットワーク経路監視、ディスク監視、ハートビートLANの状態

⑤ **故障回数**
故障したリソースID、故障許容回数(migration-threshold)、故障した回数

⑥ **制御エラー情報** (制御エラー発生時のみ表示)
リソースID、検知オペレーション(start/stop/monitor)、故障発生ノード、エラー内容("error"、"Timed Out"等)、リターンコード、エラー詳細内容

⑦ **実行不可制約**
設定されている実行不可制約の情報(対象ノードでリソース起動を行わない制約)

(*1) スプリットブレイン(ハートビートLAN故障等で他クラスタノードの認識不可)が発生した場合、孤立したノードのQuorum有無により動作を制御する。また、クラスタを統括するノードをDCノードと呼ぶ。

（その１）Pacemakerの監視コマンド「crm_mon」

```
# crm_mon -fA -L
```

-f ⑤
リソースの故障回数表示

-A ④
属性情報を表示

-L ⑦
実行不可制約を表示

オプション (簡易型)	内容
--help (-?)	オプションを表示
--verbose (-V)	デバック情報を表示
--group-by-node (-n)	ノード単位のリソースグループを表示
--simple-status (-s)	一行表示のクラスタ状態を表示
--inactive (-r)	停止状態中リソースを含む全てのリソースを表示
--one-shot (-1)	クラスタ状態を１回だけモニタに表示
--failcounts (-f)	リソースの故障回数を表示
--show-node-attributes (-A)	ノード毎のハートビートLAN状態、ディスク監視、ネットワーク監視の状態などを表示
--neg-locations (-L)	実行不可制約を表示 ※Pacemaker-1.1.13以降で使用可能

（その２）Pacemakerの動作ログ「pm_logconv」

- Pacemaker標準ログは出力が多く分かりにくいため、**pm_logconv** を使用して、**運用上必要なログだけを出力**することができる。
- Pacemaker本体のログ変更があった場合も、pm_logconv のログ変換で吸収することで影響を受けにくい。(監視ツール等の変更対応が不要)
- フェイルオーバ発生時には「Start to fail-over.」ログが出力される。

Pacemaker標準ログ

```
May 25 16:30:05 srv01 pgsql(prmApPostgreSQLDB)[19204]: INFO: PostgreSQL is down
May 25 16:30:05 srv01 crmd[15539]:  notice: Operation prmApPostgreSQLDB_monitor_10000: not running (node=srv01, call=77, rc=7, cib-update=76, confirmed=false)
    :
May 25 16:30:05 srv01 crmd[15539]:  notice: Operation prmApPostgreSQLDB_stop_0: ok (node=srv01, call=79, rc=0, cib-update=80, confirmed=true)
```

158行

ログ変換

運用上必要なログだけを出力

pm_logconv ログ (pm_logconv.out)

※出力ログ内容の詳細は【付録１】を参照

```
May 25 16:30:05 srv01   error: Resource prmApPostgreSQLDB does not work. (rc=7)
May 25 16:30:05 srv01   error: Start to fail-over.
May 25 16:30:05 srv01    info: Resource prmApPostgreSQLDB tries to stop.
May 25 16:30:05 srv01    info: Resource prmApPostgreSQLDB stopped. (rc=0)
```

4行

２．故障発生ケースの一例（２つのツールの使い方）

実際の故障発生ケースを例に、
「crm_monコマンド」と「pm_logconvログ」 の
確認手順を見てみよう！

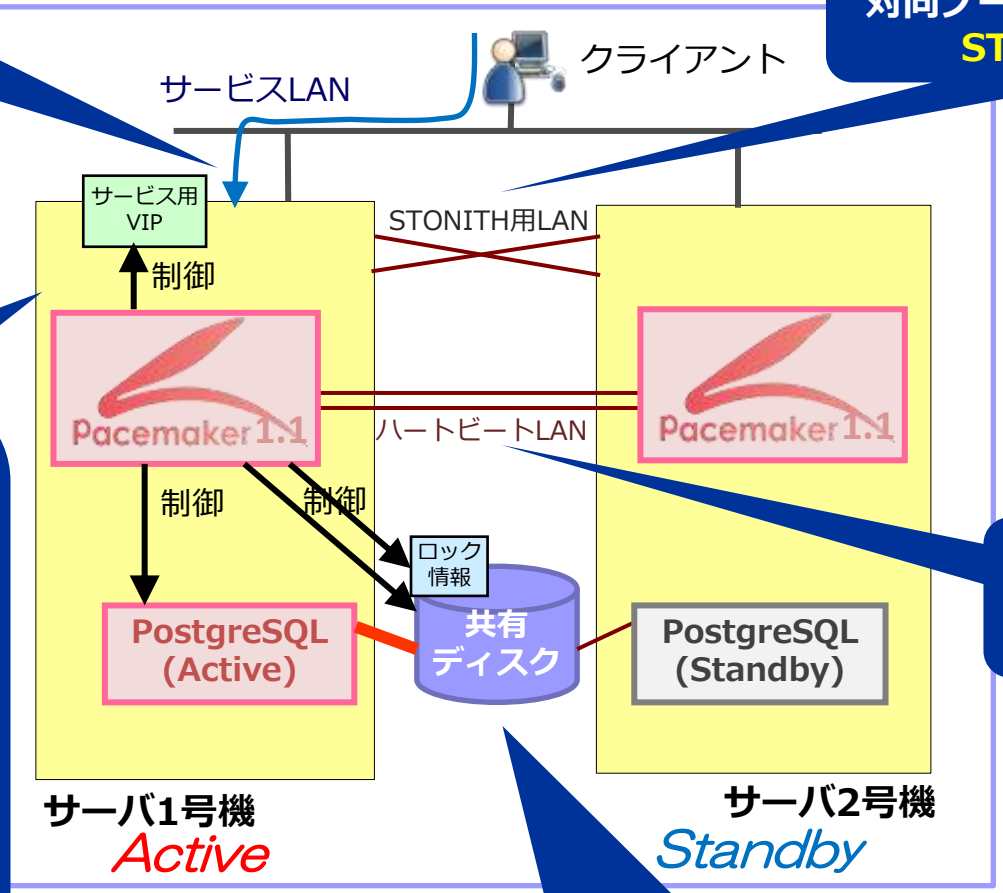
［ちょっと解説］故障発生イメージ図の見方

ポイント１

サービス用VIPの付与により、クライアントはサービス提供サーバにアクセス

ポイント４

クラスタ間に異常が発生した時に、対向ノードを強制的に電源断する STONITH機能を利用



ポイント５

サービス提供には４つのリソースが必要でPacemakerで監視/制御

① 共有ディスクのロック取得
② 共有ディスクのマウント
③ サービス用VIPの起動
④ PostgreSQLの起動

故障切り替え時には、故障サーバ側の4つのリソースを完全に停止してから、切り替え先サーバで4つのリソースを起動する

ポイント３

クラスタノード間でハートビート通信による稼働状態を確認し合う

ポイント２

両系からの共有ディスクマウントを防止するため、ロック情報をActiveサーバ側で取得

故障発生ケースの例

【サーバ1号機】
① PostgreSQLリソースの障害発生
② PacemakerがPostgreSQLの異常を検知

障害検知

フェイルオーバ開始

③ PacemakerがPostgreSQLを停止
④ 〃 サービス用VIPを停止
⑤ 〃 共有ディスクのアンマウント
⑥ 〃 共有ディスクのロック解除

PostgreSQL関連リソースの停止完了

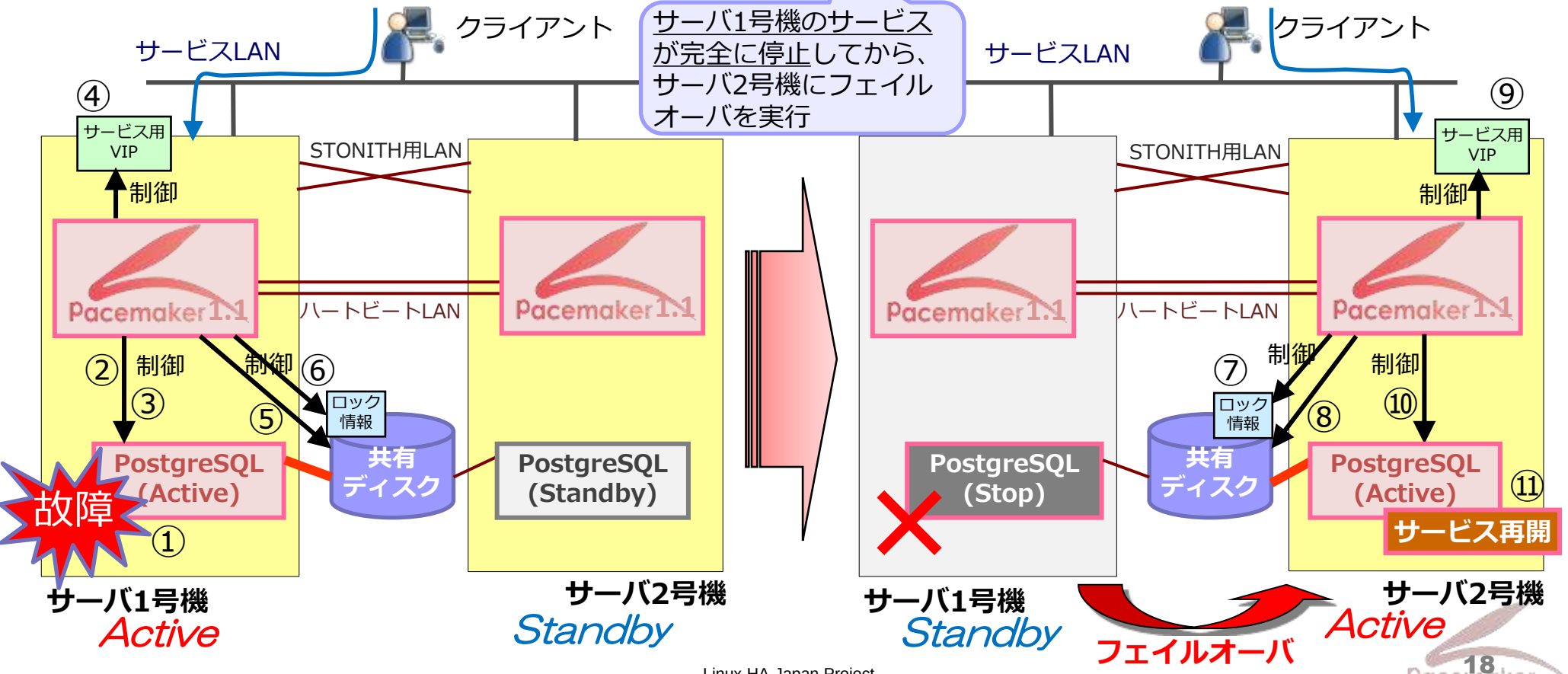
【サーバ2号機】
⑦ Pacemakerが共有ディスクのロック取得
⑧ 〃 共有ディスクのマウント
⑨ 〃 サービス用VIPを起動
⑩ 〃 PostgreSQLを起動

PostgreSQL関連リソースの起動完了

⑪ サービス再開

フェイルオーバ完了

サーバ1号機のサービスが完全に停止してから、サーバ2号機にフェイルオーバを実行



（その１）「crm_mon」の表示確認

フェイルオーバ完了

故障前

PostgreSQL関連リソースのグループが **srv01** で起動

```
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
     prmExPostgreSQLDB       (ocf::heartbeat:sfex):          Started srv01
     prmFsPostgreSQLDB       (ocf::heartbeat:Filesystem):    Started srv01
     prmIpPostgreSQLDB       (ocf::heartbeat:IPaddr2):       Started srv01
     prmApPostgreSQLDB       (ocf::heartbeat:pgsql):         Started srv01
 Resource Group: grpStonith1
     prmStonith1-1           (stonith:external/stonith-helper):  Started srv02
     prmStonith1-2           (stonith:external/ipmi):        Started srv02
 Resource Group: grpStonith2
     prmStonith2-1           (stonith:external/stonith-helper):  Started srv01
     prmStonith2-2           (stonith:external/ipmi):        Started srv01
 Clone Set: clnPing [prmPing]
     Started: [ srv01 srv02 ]
 Clone Set: clnDiskd [prmDiskd]
     Started: [ srv01 srv02 ]

Node Attributes:
* Node srv01:
    + default_ping_set          : 100
    + diskcheck_status          : normal
* Node srv02:
    + default_ping_set          : 100
    + diskcheck_status          : normal

Migration summary:
* Node srv01:
* Node srv02:
```

故障後

PostgreSQL関連リソースのグループが **srv02** で起動

PostgreSQL関連リソースの起動完了

```
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
     prmExPostgreSQLDB       (ocf::heartbeat:sfex):          Started srv02
     prmFsPostgreSQLDB       (ocf::heartbeat:Filesystem):    Started srv02
     prmIpPostgreSQLDB       (ocf::heartbeat:IPaddr2):       Started srv02
     prmApPostgreSQLDB       (ocf::heartbeat:pgsql):         Started srv02
 Resource Group: grpStonith1
     prmStonith1-1           (stonith:external/stonith-helper):  Started srv02
     prmStonith1-2           (stonith:external/ipmi):        Started srv02
 Resource Group: grpStonith2
     prmStonith2-1           (stonith:external/stonith-helper):  Started srv01
     prmStonith2-2           (stonith:external/ipmi):        Started srv01
 Clone Set: clnPing [prmPing]
     Started: [ srv01 srv02 ]
 Clone Set: clnDiskd [prmDiskd]
     Started: [ srv01 srv02 ]

Node Attributes:
* Node srv01:
    + default_ping_set          : 100
    + diskcheck_status          : normal
* Node srv02:
    + default_ping_set          : 100
    + diskcheck_status          : normal

Migration summary:
* Node srv01:
   prmApPostgreSQLDB: migration-threshold=1 fail-count=1 last-failure='Wed May 25 16:30:05 2016'
* Node srv02:

Failed actions:
    prmApPostgreSQLDB_monitor_10000 on srv01 'not running' (7): call=77, status=complete, exit-reason='none', last-rc-change='Wed May 25 16:30:05 2016', queued=0ms, exec=0ms
```

リソース故障情報の表示

RAの故障理由がcrm_monに表示されるように改善

障害検知

※基本情報、ringnumber属性値の表示を省略

［ちょっと解説］RAの故障理由がcrm_monに表示される

crm_monの表示結果の Failed actions (制御エラー情報表示部)に
RA動作における故障理由が出力されるようになりました。

従来、Pacemaker標準出力ログを確認しないとエラー理由が分からなかった。

Pacemaker-1.1系(*1)では、エラー理由がcrm_monの監視画面で分かるので、
運用の利便性が向上！

```
# crm_mon -fA
    :
Failed actions:
prmApPostgreSQLDB_start_0 on srv01 'unknown error' (1): call=76, status=complete,
exit-reason='Can't start PostgreSQL.', last-rc-change='Thu Jun 16 16:29:11
2016',queued=0ms, exec=118ms
```

従来はログのみに出力されていたRAのエラー詳細内容が表示されます！

例えば、以下のような**運用エラー**も crm_mon 監視画面で分かるようになります。

- ✓ PostgreSQL can‘t write to the log file: /var/log/pg_log　（**ログファイルが存在しないよ！**）
- ✓ My data may be inconsistent. You have to remove /var/lib/pgsql/tmp/PGSQL.lock file to force start.
（**ロックファイルを削除して！**）

(*1) Pacemaker-1.1.13以降でcrm_mon表示に対応。

（その２）「pm_logconv」のログ確認

故障後

srv01 で prmApPostgreSQLDB リソースの monitor 故障が発生

【サーバ1号機】

```
May 25 16:30:05 srv01   error: Resource prmApPostgreSQLDB does not work. (rc=7)
May 25 16:30:05 srv01   error: Start to fail-over.
May 25 16:30:05 srv01    info: Resource prmApPostgreSQLDB tries to stop.
May 25 16:30:05 srv01    info: Resource prmApPostgreSQLDB stopped. (rc=0)
May 25 16:30:05 srv01    info: Resource prmIpPostgreSQLDB tries to stop.
May 25 16:30:05 srv01    info: Resource prmIpPostgreSQLDB stopped. (rc=0)
May 25 16:30:05 srv01    info: Resource prmFsPostgreSQLDB tries to stop.
May 25 16:30:05 srv01    info: Resource prmFsPostgreSQLDB stopped. (rc=0)
May 25 16:30:05 srv01    info: Resource prmExPostgreSQLDB tries to stop.
May 25 16:30:05 srv01    info: Resource prmExPostgreSQLDB stopped. (rc=0)
```

① PostgreSQLリソースの障害発生

② PacemakerがPostgreSQLの異常を検知

障害検知

フェイルオーバ開始

③ PacemakerがPostgreSQLを停止

④ 〃 サービス用VIPを停止

⑤ 〃 共有ディスクのアンマウント

⑥ 〃 共有ディスクのロック解除

PostgreSQL関連リソースの停止完了

【サーバ2号機】

```
May 25 16:30:05 srv02    info: Resource prmExPostgreSQLDB tries to start.
May 25 16:30:06 srv02    info: Resource prmExPostgreSQLDB started. (rc=0)
May 25 16:30:06 srv02    info: Resource prmFsPostgreSQLDB tries to start.
May 25 16:30:06 srv02    info: Resource prmFsPostgreSQLDB started. (rc=0)
May 25 16:30:07 srv02    info: Resource prmIpPostgreSQLDB tries to start.
May 25 16:30:07 srv02    info: Resource prmIpPostgreSQLDB started. (rc=0)
May 25 16:30:07 srv02    info: Resource prmApPostgreSQLDB tries to start.
May 25 16:30:08 srv02    info: Resource prmApPostgreSQLDB started. (rc=0)
```

⑦ Pacemakerが共有ディスクのロック取得

⑧ 〃 共有ディスクのマウント

⑨ 〃 サービス用VIPを起動

⑩ 〃 PostgreSQLを起動

PostgreSQL関連リソースの起動完了

※DCノード(*1)で出力

```
May 25 16:30:08 srv01    info: Resource prmExPostgreSQLDB : Move srv01 -> srv02
May 25 16:30:08 srv01    info: Resource prmApPostgreSQLDB : Move srv01 -> srv02
May 25 16:30:08 srv01    info: fail-over succeeded.
```

⑪ サービス再開

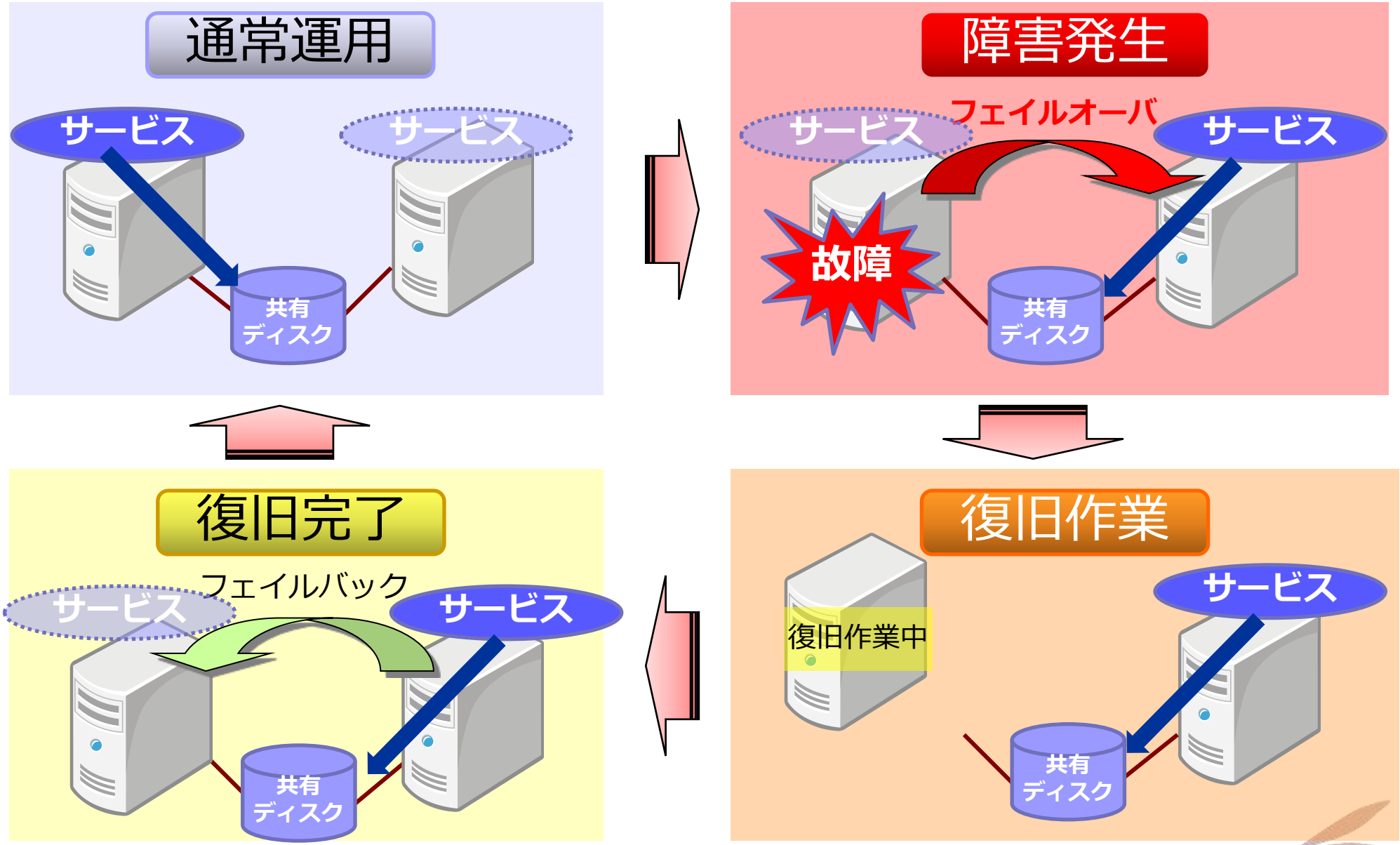
フェイルオーバ完了

(*1) クラスタを統括するノードをDCノードと呼ぶ。

３．復旧手順の流れ

故障発生時の
復旧手順の流れをつかんでみよう！

障害発生～復旧までの大きな流れ



復旧手順の流れ

復旧手順の一例

手順１ ノード状態確認
手順２ ACT化抑止
手順３ ノード状態確認

故障復旧

手順４ 故障回数のクリア
手順５ ACT化抑止の解除
手順６ ノード状態・故障回数の確認

手順７ リソースグループの切り戻し(1/2)
手順８ リソース状態の確認
手順９ リソースグループの切り戻し(2/2)
手順10 リソース状態の確認

障害発生 片系でサービス中

［1］安全に復旧作業を行うための準備

復旧作業 片系でサービス中

［2］復旧作業前の状態に戻す

復旧完了 両系でサービス再開

［3］故障発生前の状態に戻す

※切り戻しを行う場合

通常運用

復旧手順の一例　［1］安全に復旧作業を行うための準備

手順１ ノード状態確認

手順２ ACT化抑止

手順３ ノード状態確認

故障復旧

手順４ 故障回数のクリア

手順５ ACT化抑止の解除

手順６ ノード状態・故障回数の確認

手順７ リソースグループの切り戻し(1/2)

手順８ リソース状態の確認

手順９ リソースグループの切り戻し(2/2)

手順10 リソース状態の確認

- **サーバ2号機で、サービスリソースの起動を確認**
  ⇒ 2号機でサービス継続中

- **故障の復旧作業中に、サーバ1号機が ACT状態へ遷移しないように抑止**

- **サーバ1号機の状態が “standby” であることを確認**

安全に復旧作業を行う準備完了！

```
# crm_mon -fA
   :
Node srv01: standby
 Online: [ srv02 ]
   :
Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB       (ocf::heartbeat:sfex):      Started srv02
    prmFsPostgreSQLDB       (ocf::heartbeat:Filesystem): Started srv02
    prmIpPostgreSQLDB       (ocf::heartbeat:IPaddr2):   Started srv02
    prmApPostgreSQLDB       (ocf::heartbeat:pgsql):     Started srv02
   :
```

フェイルオーバにより サーバ2号機で起動

※crm_mon表示は一部省略

復旧手順の一例　［2］復旧作業前の状態に戻す

手順１　ノード状態確認
手順２　ACT化抑止
手順３　ノード状態確認

故障復旧

手順４　故障回数のクリア
手順５　ACT化抑止の解除
手順６　ノード状態・故障回数の確認

手順７　リソースグループの切り戻し(1/2)
手順８　リソース状態の確認
手順９　リソースグループの切り戻し(2/2)
手順10　リソース状態の確認

※crm_mon表示は一部省略

- **故障リソースの故障回数とエラーステータスをクリア**
  ※リソース監視を初期状態に戻すのに必要な手順です
- **サーバ1号機が、ACT状態へ遷移できるように抑止を解除**
- **サーバ1号機の状態が “Online” であることを確認**

```
# crm_mon -fA
  :
Online: [ srv01 srv02 ]
  :
```

復旧作業前の状態戻し完了！

復旧手順の一例　［3］故障発生前の状態に戻す

手順１　ノード状態確認
手順２　ACT化抑止
手順３　ノード状態確認

故障復旧

手順４　故障回数のクリア
手順５　ACT化抑止の解除
手順６　ノード状態・故障回数の確認

手順７　リソースグループの切り戻し(1/2)
手順８　リソース状態の確認
手順９　リソースグループの切り戻し(2/2)
手順10　リソース状態の確認

※切り戻しを実施しなくても、サービス継続は可能です。
　次に故障が起きた場合にも、2号機⇒1号機に自動で切り替わります。

- **リソースグループをサーバ1号機に切り戻す**
  ⇒ 1号機に切り戻して、サービスを継続

- **サーバ1号機で、サービスリソースの起動を確認**

```
# crm_mon -fA -L
   :
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB        (ocf::heartbeat:sfex):          Started srv01
    prmFsPostgreSQLDB        (ocf::heartbeat:Filesystem):    Started srv01
    prmIpPostgreSQLDB        (ocf::heartbeat:IPaddr2):       Started srv01
    prmApPostgreSQLDB        (ocf::heartbeat:pgsql):         Started srv01
    :
Negative location constraints:
  cli-ban-grpPostgreSQLDB-on-srv02 prevents grpPostgreSQLDB from running on srv02
```

フェイルバックにより
サーバ1号機で起動

- **サーバ2号機の実行不可制約を解除**
  ※実行不可制約については後程ご説明します

※crm_mon表示は一部省略

切り戻しの流れ

【サーバ2号機】

① PacemakerがPostgreSQLを停止
② 〃 サービス用VIPを停止
③ 〃 共有ディスクのアンマウント
④ 〃 共有ディスクのロック解除

PostgreSQL関連リソースの停止完了

【サーバ1号機】

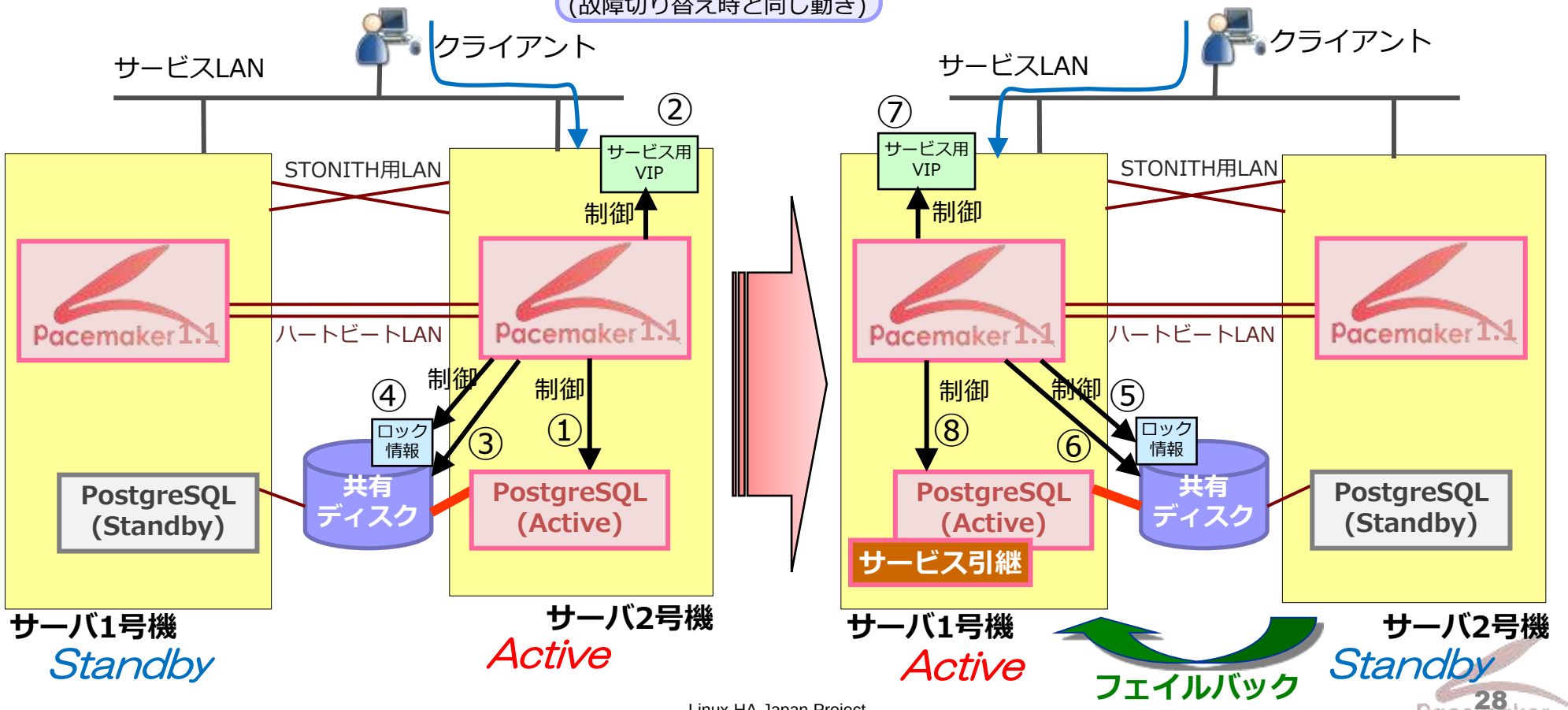
⑤ Pacemakerが共有ディスクのロック取得
⑥ 〃 共有ディスクのマウント
⑦ 〃 サービス用VIPを起動
⑧ 〃 PostgreSQLを起動

PostgreSQL関連リソースの起動完了

⇒ サービス切り戻し完了

フェイルバック完了

サーバ1号機に
フェイルバックを実行
(故障切り替え時と同じ動き)



【テーマ２】

Pacemakerで対応する “故障”ケースの起こし方と復旧手順

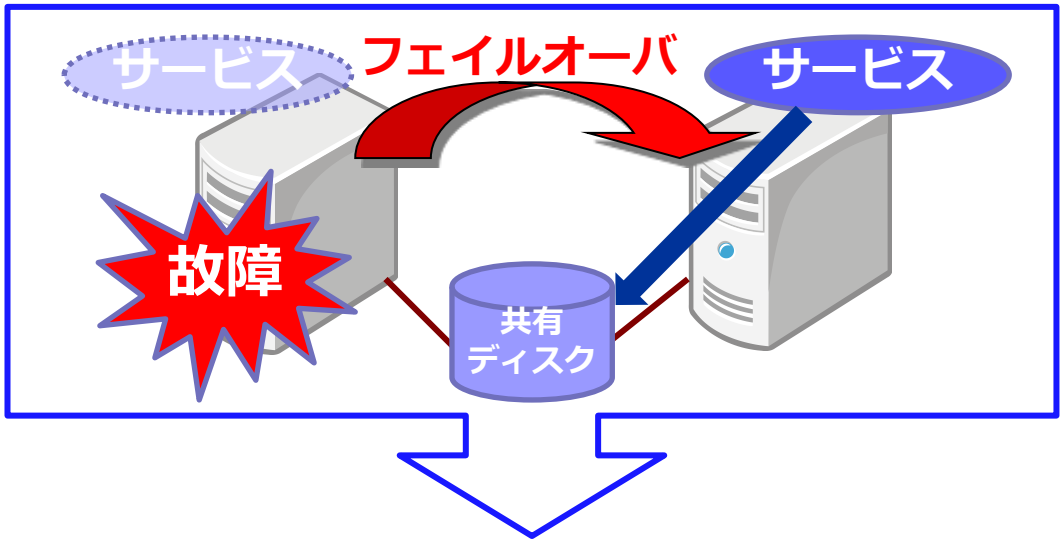
～事前に動作検証しよう～

[テーマ２] 動作検証、復旧手順の基本

１．Pacemakerによる監視／制御と故障ケース（Pacemaker動作３パターン）
２．復旧手順の整理（３パターン）
３．各故障ケースの実例（６パターン）
　① 発生手順イメージ　② 発生手順　③ 故障発生時の動作
　④ pm_logconvのログ確認　⑤ 復旧手順

動作検証の必要性

➢実際に、HAクラスタシステムを運用するユーザからの問合せで多いのは？



ユーザ

フェイルオーバが発生した理由を調べてほしい。

故障発生後の復旧方法を教えてほしい。

事前に「故障発生時の動き」や「復旧手順」を確認しておくことで、安心して保守運用ができます

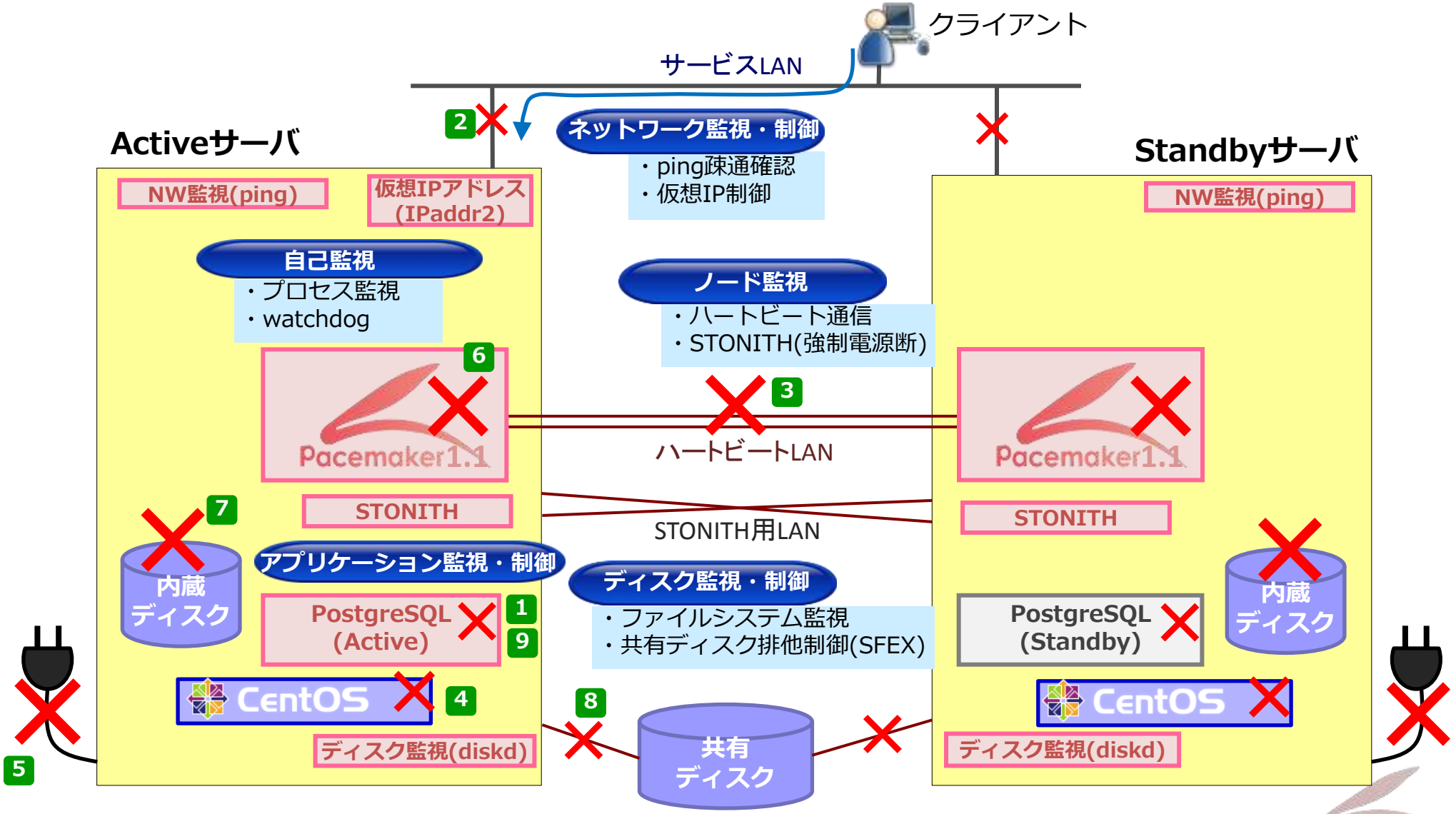
サポート担当者

１．Pacemakerによる監視／制御と故障ケース（Pacemaker動作３パターン）

Pacemakerで監視・制御できる故障ケースと
その動作パターンを詳しく見てみよう！

Pacemakerによる監視/制御と故障ケース

Pacemakerでは様々な故障を検知して、サービスの継続性を高めることができる。



故障ケース毎のPacemakerの動作

故障ケースとPacemakerの動作を整理すると、以下のようになる。
（Active側のみ記載）

	故障項目	故障内容	Pacemakerの動作	
1	リソース故障	1 PostgreSQL故障	[1] リソース／プロセス再起動 or [2] 通常フェイルオーバ ※1	アプリケーション監視・制御
2	ネットワーク故障	2 サービスLAN故障	[2] 通常フェイルオーバ	ネットワーク監視・制御
		3 ハートビートLAN故障	[3] STONITH後フェイルオーバ	
3	ノード故障	4 カーネルハング	[3] STONITH後フェイルオーバ	ノード監視
		5 サーバ電源停止	[3] STONITH後フェイルオーバ	
4	Pacemakerプロセス故障	6 corosyncプロセス故障	[3] STONITH後フェイルオーバ	自己監視
5	ディスク故障	7 内蔵ディスク故障	[2] 通常フェイルオーバ or [3] STONITH後フェイルオーバ ※2	ディスク監視・制御
		8 共有ディスクケーブル故障	[2] 通常フェイルオーバ	
6	リソース停止失敗	9 PostgreSQL stop失敗	[3] STONITH後フェイルオーバ	アプリケーション監視・制御

※1 設定により変更可能
※2 ディスク故障範囲により動作が異なる

故障時のPacemakerの動作（３パターン）

故障時のPacemakerの動作は、サービス影響や故障サーバ状態により、３パターンに分かれる。

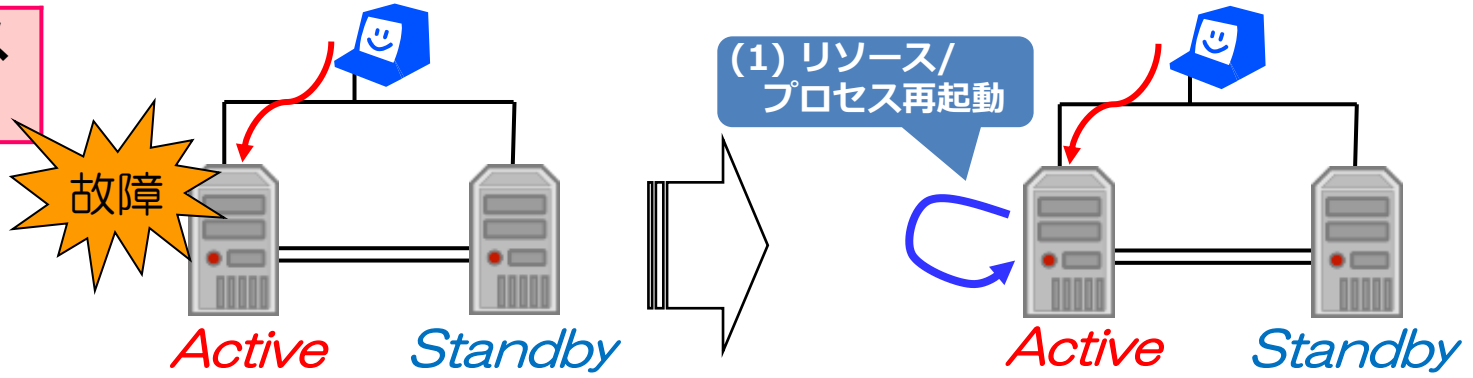
	Pacemakerの動作		
	[1] リソース/プロセス再起動	**[2] 通常フェイルオーバ**	**[3] STONITH後フェイルオーバ**
動作概要	同じサーバ上でリソース/プロセスをもう一度起動、または設定変更する。 ※フェイルオーバはしない。	故障サーバの**関連リソースを停止**後、Standbyサーバでリソースを起動する。	故障サーバの**電源を強制的に断(STONITH)**後、Standbyサーバでリソースを起動する。
対処条件	**サービス継続に直接関係ない**リソース故障時の対処。	**サービス継続に影響がある**故障時の対処。	**故障サーバの状態が確認できない**場合に二重起動を防ぐ対処。
故障例	・レプリケーションLAN故障 (共有ディスク無し構成)	・DBプロセス停止 ・サービスLAN故障 ・共有ディスクケーブル故障	・サーバ電源停止 ・Pacemakerプロセス故障 ・ハートビートLAN故障 ・リソース停止失敗

短い（数秒程度） ← サービス中断時間 → 長い（数十秒～数分程度）

故障時のPacemakerの動作（３パターン）

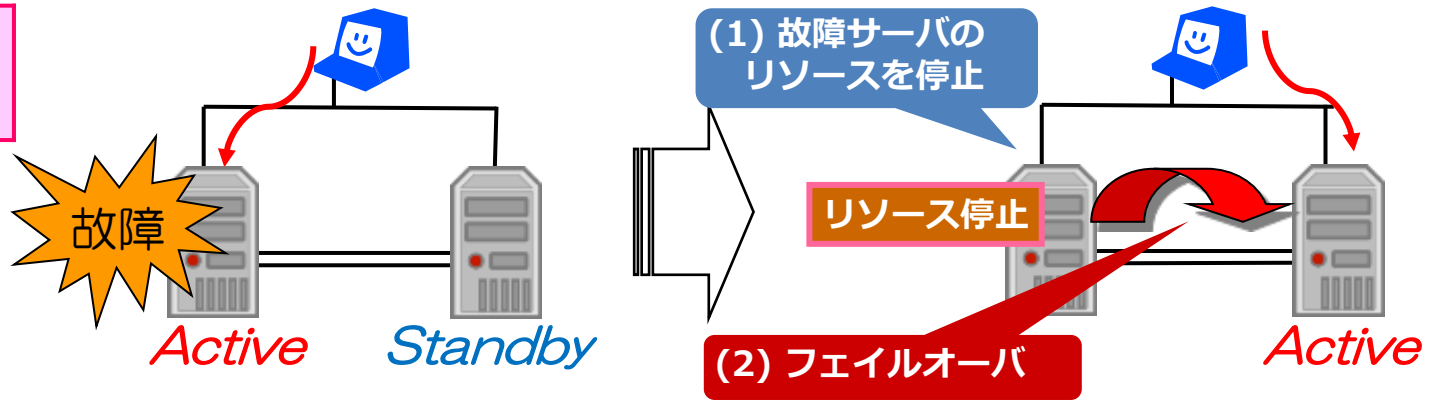
[1] リソース/プロセス再起動

※フェイルオーバはせずに、故障リソースのみ再起動する。



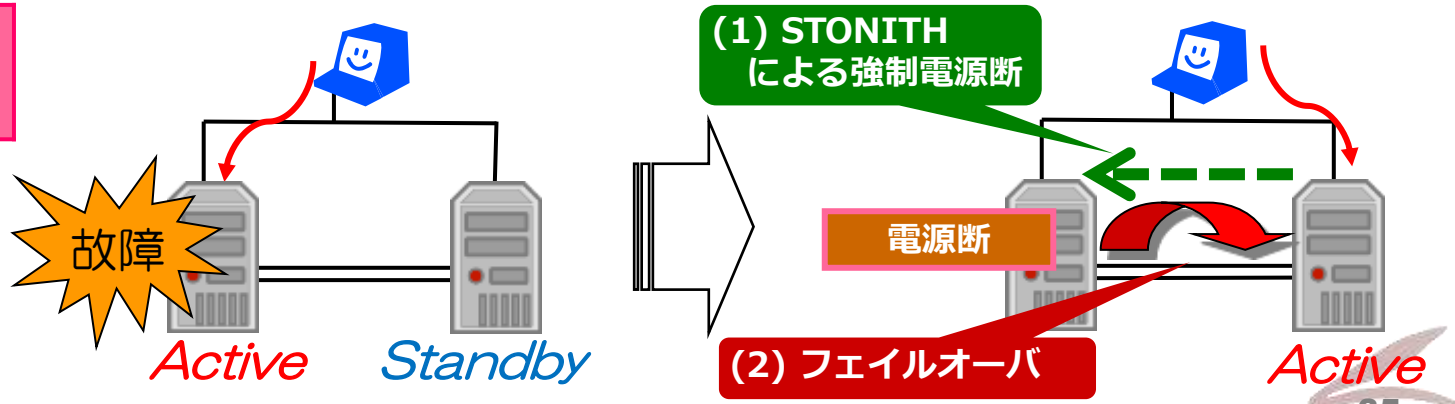
[2] 通常フェイルオーバ

※故障サーバのリソースを停止後に、フェイルオーバを行う。（通常の切り替え動作）



[3] STONITH後フェイルオーバ

※故障サーバのリソース停止不可や、故障サーバの状態確認不可の場合に、二重起動を防ぐため、強制電源断後にフェイルオーバを行う。



２．復旧手順の整理（３パターン）

復旧手順が必要な理由を知ることで、
クラスタ復旧の理解を深めてみよう！

復旧手順の違いは？

復旧手順の違いが
よく分からない・・・・

	復旧パターン１	復旧パターン２	復旧パターン３
復旧前の準備	手順１ ノード状態確認	手順１ ノード状態確認	手順１ ノード状態確認
	手順２ ACT化抑止	手順２ ノード起動	手順２ 強制電源断
	手順３ ノード状態確認		手順３ ノード状態確認
故障状態により手順が異なる	故障復旧	故障復旧	故障復旧
	手順４ 故障回数のクリア		手順４ ノード起動
	手順５ ACT化抑止の解除	手順３ Pacemaker起動	手順５ Pacemaker起動
	手順６ ノード状態・故障回数の確認	手順４ ノード状態確認	手順６ ノード状態確認
2号機⇒1号機の切り戻し	手順７ リソースグループの切り戻し(1/2)	手順５ リソースグループの切り戻し(1/2)	手順７ リソースグループの切り戻し(1/2)
	手順８ リソース状態の確認	手順６ リソース状態の確認	手順８ リソース状態の確認
	手順９ リソースグループの切り戻し(2/2)	手順７ リソースグループの切り戻し(2/2)	手順９ リソースグループの切り戻し(2/2)
	手順10 リソース状態の確認	手順８ リソース状態の確認	手順10 リソース状態の確認

故障発生後の状態から「復旧に必要な手順」を確認

	復旧パターン１	復旧パターン２	復旧パターン３
故障内容	**リソース故障** ネットワーク故障	ハートビート通信断　ノード故障 プロセス故障　リソース停止失敗	ディスク故障
必要な対応	復旧後に故障回数をクリアしないと、リソース監視が初期状態に戻らない ② **リソース故障の場合、故障回数クリアが必要**		
ノード状態	両系起動	**片系起動** or 両系起動	**片系起動／片系異常**
必要な対応		ノード停止状態の場合、起動が必要 ④ **ノード起動が必要**	ディスク故障で正常停止不可 ③ **強制電源断が必要** ④ **ノード起動が必要**
Pacemaker状態	**両系起動**	**片系起動**	**片系起動**
必要な対応	復旧前に再切替えが発生するとサービス停止してしまう ① **復旧前に、SBY側故障による再ACT化の防止が必要**	STONITHによりPacemaker停止 ⑤ **Pacemaker起動が必要**	STONITHによりPacemaker停止 ⑤ **Pacemaker起動が必要**
リソース移動	**あり**	**あり** or なし	**あり**
必要な対応	⑥ **リソース切り戻しが必要**	⑥ **リソース切り戻しが必要**	⑥ **リソース切り戻しが必要**

故障発生後の状態から
復旧に必要な手順が決まります

復旧手順の流れの整理（３パターン）

	復旧パターン１	復旧パターン２	復旧パターン３
故障内容	リソース故障 ネットワーク故障	ハートビート通信断　ノード故障 プロセス故障　リソース停止失敗	ディスク故障
必要な対応	②リソース故障の場合、故障回数クリアが必要		
ノード状態	両系起動	片系起動 or 両系起動	片系起動／片系異常
必要な対応		④ノード起動が必要	③強制電源断が必要　④ノード起動
Pacemaker状態	両系起動	片系起動	片系起動
必要な対応	①復旧前に、SBY側故障による再ACT化の防止が必要	⑤Pacemaker起動が必要	⑤Pacemaker起動が必要
リソース移動	あり	あり or なし	あり
必要な対応	⑥リソース切り戻しが必要	⑥リソース切り戻しが必要	⑥リソース切り戻しが必要

手順にすると

	復旧パターン１	復旧パターン２	復旧パターン３
復旧前	ACT化抑止 ①	ノード起動 ④	強制電源断 ③
故障状態による	故障復旧 故障回数のクリア ② ACT化抑止の解除 ①	故障復旧 Pacemaker起動 ⑤	故障復旧 ノード起動 ④ Pacemaker起動 ⑤
切り戻し	リソースグループの切り戻し ⑥	リソースグループの切り戻し ⑥	リソースグループの切り戻し ⑥

［テーマ２］動作検証、復旧手順の基本

３．各故障ケースの実例（６パターン）
① 発生手順イメージ
② 発生手順
③ 故障発生時の動作
④ pm_logconvのログ確認
⑤ 復旧手順

故障ケースを起こして動作検証することで、
クラスタ動作の理解を深めてみよう！

※本資料上の故障時の動作は一例であり、個々の故障内容に応じて異なる動作の場合もあります。

故障項目毎の故障発生手順・復旧手順

凡例 [1] リソース／プロセス再起動
[2] 通常フェイルオーバ
[3] STONITH後フェイルオーバ

	故障項目	故障内容	Pacemakerの動作	故障発生手順	復旧手順
1	**リソース故障**	PostgreSQL故障	[1] or [2]	$ pg_ctl -m i stop （または、# kill -9 PID[PostgreSQL]）	[パターン１’] (フェイルバック)
2	**ネットワーク故障**	サービスLAN故障	[2]	# iptables -A INPUT -i [S-LAN_IF] -j DROP; iptables -A OUTPUT -o [S-LAN_IF] -j DROP （または、ネットワークケーブルの抜線）	[パターン１] (フェイルバック)
		ハートビートLAN故障	[3]	# iptables -A INPUT -i [HB-LAN1_IF] -j DROP; iptables -A OUTPUT -o [HB-LAN1_IF] -j DROP # iptables -A INPUT -i [HB-LAN2_IF] -j DROP; iptables -A OUTPUT -o [HB-LAN2_IF] -j DROP	[パターン２’] Pacemaker再起動
3	**ノード故障**	カーネルパニック	[3]	# echo c > /proc/sysrq-trigger	[パターン２] Pacemaker再起動 (+フェイルバック)
		サーバ電源停止	[3]	# poweroff -nf	
4	**Pacemakerプロセス故障**	corosyncプロセス故障	[3]	# pkill -9 corosync	
5	**ディスク故障**	内蔵ディスク故障	[2] or [3]	内蔵ディスク引き抜き	[パターン３] 強制電源断 ＋Pacemaker再起動 (＋フェイルバック)
		共有ディスクケーブル故障	[2]	ディスクケーブル引き抜き	
6	**リソース停止失敗**	PostgreSQL stop失敗	[3]	pgsql RAのstopメソッドを return $OCF_ERR_GENERICに書き換え	[パターン２] Pacemaker再起動 (＋フェイルバック)

✓本編では「ネットワーク故障(サービスLAN)」「リソース停止失敗」の２ケースを取り上げます。
✓それ以外のケースは【付録２】を参照してください。

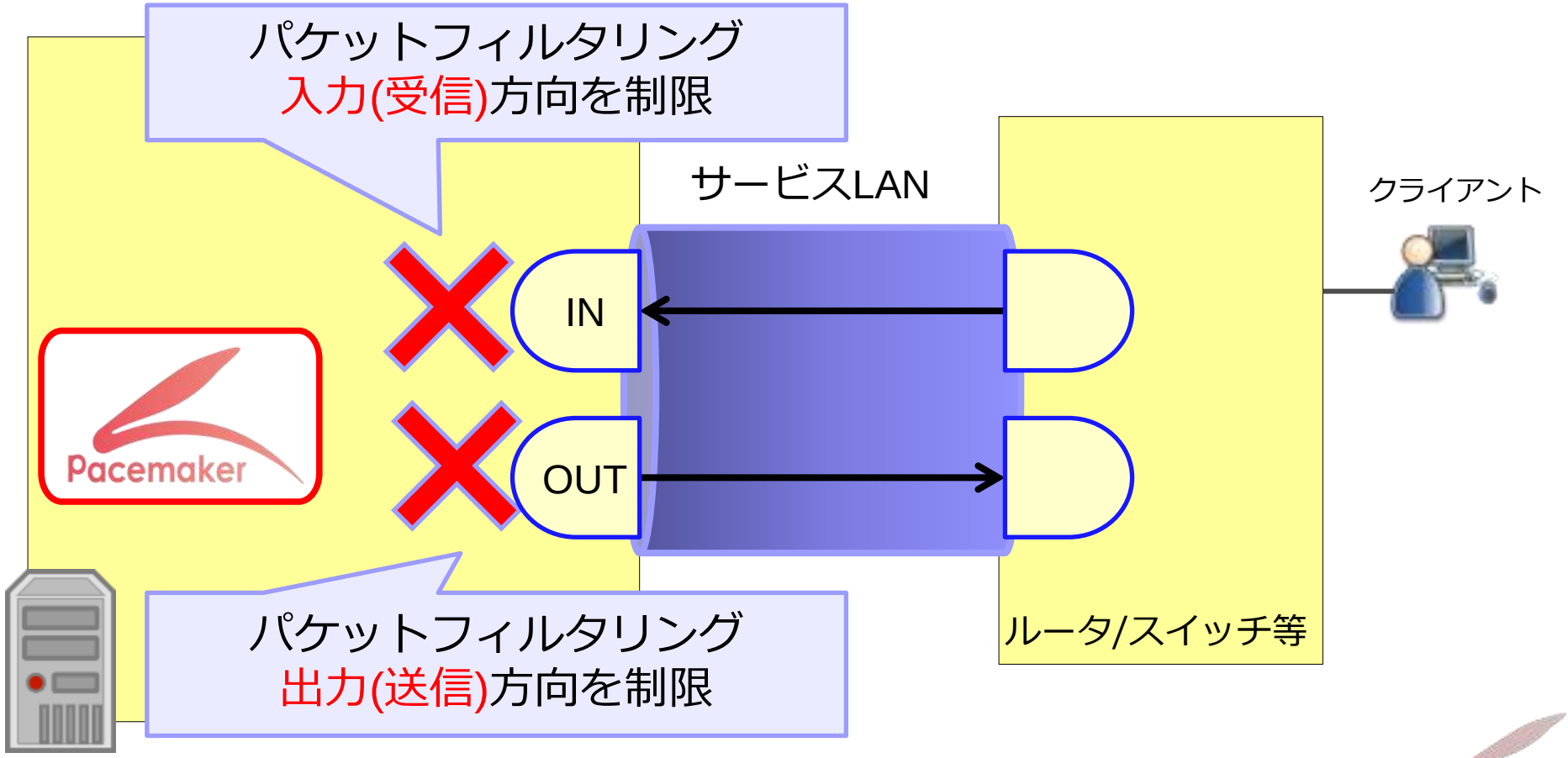
【2.ネットワーク故障】

サービスLAN故障

【2.ネットワーク故障-1】①発生手順イメージ

凡例[1] リソース／プロセス再起動
[2] 通常フェイルオーバ
[3] STONITH後フェイルオーバ

故障項目	故障内容	Pacemakerの動作	故障発生手順	復旧手順
ネットワーク故障	サービスLAN故障	[2]	# iptables -A INPUT -i [S-LAN_IF] -j DROP; iptables -A OUTPUT -o [S-LAN_IF] -j DROP （または、ネットワークケーブルの抜線）	[パターン１] (フェイルバック)



【2.ネットワーク故障-1】②発生手順(1/2)

発生手順

サービスLAN故障

➢ **サービスLAN不通を起こすため、パケットフィルタリングを設定**
- ✓ サブコマンド： -A(ルールを追加)
- ✓ オプション： -i/-o [入力/出力ネットワークインタフェースを指定] –j [ルールにマッチした場合の動作を指定]

```
# iptables -A INPUT -i [S-LAN_IF] -j DROP;  iptables -A OUTPUT -o [S-LAN_IF] -j DROP
```

⚠ IN/OUT双方向の通信を切断すること

ネットワーク不通の方法として**「ifdownコマンド」の手順は選択しないこと**。
ifdownコマンドによりネットワーク不通とした場合、実環境のネットワーク断とは異なる動作となり、復旧手順も異なる。
つまり、ifdownコマンドでは運用時の障害を想定した動作検証が十分に行えないため、iptablesコマンド、またはケーブル抜線を行ってください。

確認手順

NW状態確認

➢ **パケットフィルタリングの設定状況を確認**
- ✓ サブコマンド： -L(ルールを表示)

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
```

IN/OUT方向共にDROPが設定されている

【2.ネットワーク故障-1】②発生手順(2/2)

確認手順

ノード状態確認

➢ **1号機のping確認不可となり、PostgreSQLリソースが2号機で起動を確認**

```
# crm_mon -fA
   :
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB        (ocf::heartbeat:sfex):        Started srv02
    prmFsPostgreSQLDB        (ocf::heartbeat:Filesystem):  Started srv02
    prmIpPostgreSQLDB        (ocf::heartbeat:IPaddr2):     Started srv02
    prmApPostgreSQLDB        (ocf::heartbeat:pgsql):       Started srv02
    :
Node Attributes:
* Node srv01:
    + default_ping_set              : 0             : Connectivity is lost
    :
```

フェイルオーバにより
サーバ2号機で起動

ネットワーク監視
エラーが発生

回復手順

サービスLAN故障
回復

➢ **サービスLAN不通のパケットフィルタリングを解除**

✓ サブコマンド： -F(ルールを解除)、 -L(ルールを表示)

```
# iptables -F
```

確認手順

NW状態確認

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

IN/OUT方向共に
DROPが解除されている

ノード状態確認

```
# crm_mon -fA
   :
Node Attributes:
* Node srv01:
    + default_ping_set              : 100
```

ネットワーク監視が
正常に回復

※crm_mon表示は一部省略

【2.ネットワーク故障-1】③故障発生時の動作

【サーバ1号機】

① サービスLANの障害発生
② Pacemakerがping監視の異常を検知
③ PacemakerがPostgreSQLを停止
④ 〃 サービス用VIPを停止
⑤ 〃 共有ディスクのアンマウント
⑥ 〃 共有ディスクのロック解除

障害検知

PostgreSQL関連リソースの停止完了

【サーバ2号機】

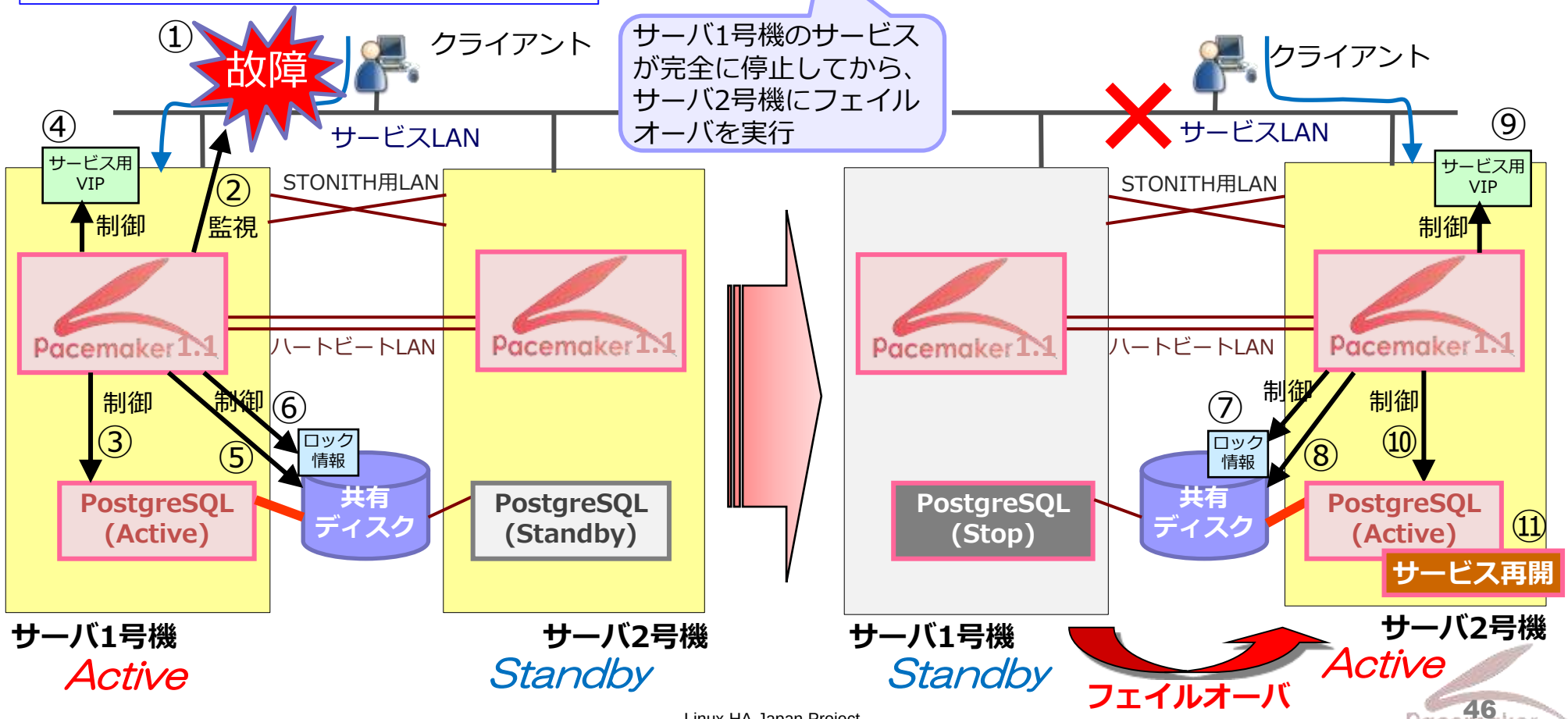
⑦ Pacemakerが共有ディスクのロック取得
⑧ 〃 共有ディスクのマウント
⑨ 〃 サービス用VIPを起動
⑩ 〃 PostgreSQLを起動

PostgreSQL関連リソースの起動完了

⑪ サービス再開

フェイルオーバ完了

サーバ1号機のサービスが完全に停止してから、サーバ2号機にフェイルオーバを実行



【2.ネットワーク故障-1】④pm_logconvのログ確認

故障後

srv01でサービスLANの ping監視NG が発生し、属性値(default_ping_set)を 100→0 に変更

① サービスLANの障害発生

【サーバ1号機】

```
May 25 17:32:18 srv01   error: Network to 192.168.101.1 is unreachable.
May 25 17:32:18 srv01    info: Attribute "default_ping_set" is updated to "0" at “srv01".
May 25 17:32:23 srv01   error: Start to fail-over.
May 25 17:32:23 srv01    info: Resource prmApPostgreSQLDB tries to stop.
May 25 17:32:25 srv01    info: Resource prmApPostgreSQLDB stopped. (rc=0)
May 25 17:32:25 srv01    info: Resource prmIpPostgreSQLDB tries to stop.
May 25 17:32:25 srv01    info: Resource prmIpPostgreSQLDB stopped. (rc=0)
May 25 17:32:25 srv01    info: Resource prmFsPostgreSQLDB tries to stop.
May 25 17:32:25 srv01    info: Resource prmFsPostgreSQLDB stopped. (rc=0)
May 25 17:32:25 srv01    info: Resource prmExPostgreSQLDB tries to stop.
May 25 17:32:25 srv01    info: Resource prmExPostgreSQLDB stopped. (rc=0)
```

② Pacemakerがping監視の異常を検知

障害検知

フェイルオーバ開始

③ PacemakerがPostgreSQLを停止

④ 〃 サービス用VIPを停止

⑤ 〃 共有ディスクのアンマウント

⑥ 〃 共有ディスクのロック解除

PostgreSQL関連リソースの停止完了

【サーバ2号機】

```
May 25 17:32:18 srv02    info: Attribute "default_ping_set" is updated to "0" at "srv01".
May 25 17:32:25 srv02    info: Resource prmExPostgreSQLDB tries to start.
May 25 17:32:26 srv02    info: Resource prmExPostgreSQLDB started. (rc=0)
May 25 17:32:26 srv02    info: Resource prmFsPostgreSQLDB tries to start.
May 25 17:32:26 srv02    info: Resource prmFsPostgreSQLDB started. (rc=0)
May 25 17:32:26 srv02    info: Resource prmIpPostgreSQLDB tries to start.
May 25 17:32:26 srv02    info: Resource prmIpPostgreSQLDB started. (rc=0)
May 25 17:32:26 srv02    info: Resource prmApPostgreSQLDB tries to start.
May 25 17:32:28 srv02    info: Resource prmApPostgreSQLDB started. (rc=0)
```

⑦ Pacemakerが共有ディスクのロック取得

⑧ 〃 共有ディスクのマウント

⑨ 〃 サービス用VIPを起動

⑩ 〃 PostgreSQLを起動

PostgreSQL関連リソースの起動完了

※DCノード(*1)で出力

```
May 25 17:32:28 srv01    info: Resource prmExPostgreSQLDB : Move srv01 -> srv02
May 25 17:32:28 srv01    info: Resource prmApPostgreSQLDB : Move srv01 -> srv02
May 25 17:32:28 srv01    info: fail-over succeeded.
```

⑪ サービス再開

フェイルオーバ完了

(*1) クラスタを統括するノードをDCノードと呼ぶ。

【2.ネットワーク故障-1】⑤復旧手順(1/3) 復旧手順パターン1

手順1 ノード状態確認

➢ **リソース状態が “Started サーバ2号機” となっていることを確認**

```
# crm_mon -fA
   :
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB        (ocf::heartbeat:sfex):        Started srv02
    prmFsPostgreSQLDB        (ocf::heartbeat:Filesystem):  Started srv02
    prmIpPostgreSQLDB        (ocf::heartbeat:IPaddr2):     Started srv02
    prmApPostgreSQLDB        (ocf::heartbeat:pgsql):       Started srv02
    :
Node Attributes:
* Node srv01:
    + default_ping_set        : 0          : Connectivity is lost
    + diskcheck_status        : normal
    :
* Node srv02:
    + default_ping_set        : 100
    + diskcheck_status        : normal
```

フェイルオーバにより サーバ2号機で起動

サーバ1号機で ネットワーク監視 エラーが発生

手順2 ACT化抑止

➢ **故障復旧作業中に、サーバ1号機がACT状態へ遷移しないよう抑止**

✓ crm_standbyコマンドは、ノードのステータス(Online/OFFLINE/standby)制御を行う
✓ オプション： -U [ノードのホスト名] -v [ステータスをstandbyにするか否かを指定]

```
# crm_standby -U srv01 -v on
```

手順3 ノード状態確認

➢ **サーバ1号機の状態が “standby” となっていることを確認**

```
# crm_mon -fA
   :
Node srv01: standby
 Online: [ srv02 ]
   :
```

安全に復旧作業を行う準備完了！

※crm_mon表示は一部省略

【2.ネットワーク故障-1】⑤復旧手順(2/3) 復旧手順パターン1

故障復旧

~~手順4 故障回数のクリア~~

【1.リソース故障】ではないため、故障回数のクリア手順は不要です。

手順5 ACT化抑止の解除

➢ **サーバ1号機が ACT状態へ遷移できるように抑止を解除**

✓ crm_standbyコマンドは、ノードのステータス(Online/OFFLINE/standby)制御を行う
✓ オプション： -U [ノードのホスト名] -v [ステータスをstandbyにするか否かを指定]

```
# crm_standby -U srv01 -v off
```

手順6 ノード状態~~・故障回数~~の確認

➢ **サーバ1号機の状態が “Online” となっていることを確認**

✓ 現用機の“Migration summary”に何も表示されていないことを確認

```
# crm_mon -fA
   :
Online: [ srv01 srv02 ]
   :
Migration summary:
* Node srv02:
* Node srv01:
```

復旧作業前の状態戻し完了！

※crm_mon表示は一部省略

【2.ネットワーク故障-1】⑤復旧手順(3/3) 復旧手順パターン1

手順7 リソースグループの切り戻し(1/2)

➢ **リソースグループをサーバ1号機に切り戻す**

✓ crm_resourceコマンドは、リソースを動的に操作(表示/設定/削除)する
✓ オプション： -M(リソースを指定ノードで起動するように切り替える制約追加) -r [リソースIDを指定] –N [ホスト名] –f(リソースを強制的に再配置) –Q(値のみ表示)

```
# crm_resource -M -r grpPostgreSQLDB -N srv01 -f -Q
```

手順8 リソース状態の確認

➢ **リソース状態が “Started サーバ1号機” となっていることを確認**

✓ リソースの実行不可制約がサーバ2号機に設定されていること

手順7でサーバ1号機にリソースを切り戻すため、サーバ2号機でリソース起動を行わない制約が設定されます。
切り戻し完了後に、その制約を解除しておく必要があります。

```
# crm_mon -fA -L
  :
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB       (ocf::heartbeat:sfex):          Started srv01
    prmFsPostgreSQLDB       (ocf::heartbeat:Filesystem):    Started srv01
    prmIpPostgreSQLDB       (ocf::heartbeat:IPaddr2):       Started srv01
    prmApPostgreSQLDB       (ocf::heartbeat:pgsql):         Started srv01
  :
Negative location constraints:
  cli-ban-grpPostgreSQLDB-on-srv02 prevents grpPostgreSQLDB from running on srv02
```

-L(実行不可制約表示)を付ける

手順9 リソースグループの切り戻し(2/2)

➢ **サーバ2号機の実行不可制約を解除**

✓ オプション： -U(切り替えによる制約を解除) -r [リソースIDを指定]

よく解除忘れが起こるので注意

```
# crm_resource -U -r grpPostgreSQLDB
```

手順10 リソース状態の確認

➢ **実行不可制約の解除を確認**

```
# crm_mon -fA -L
  :
Negative location constraints:
```

リソース切り戻し時の実行不可制約の解除漏れを防止

※crm_mon表示は一部省略

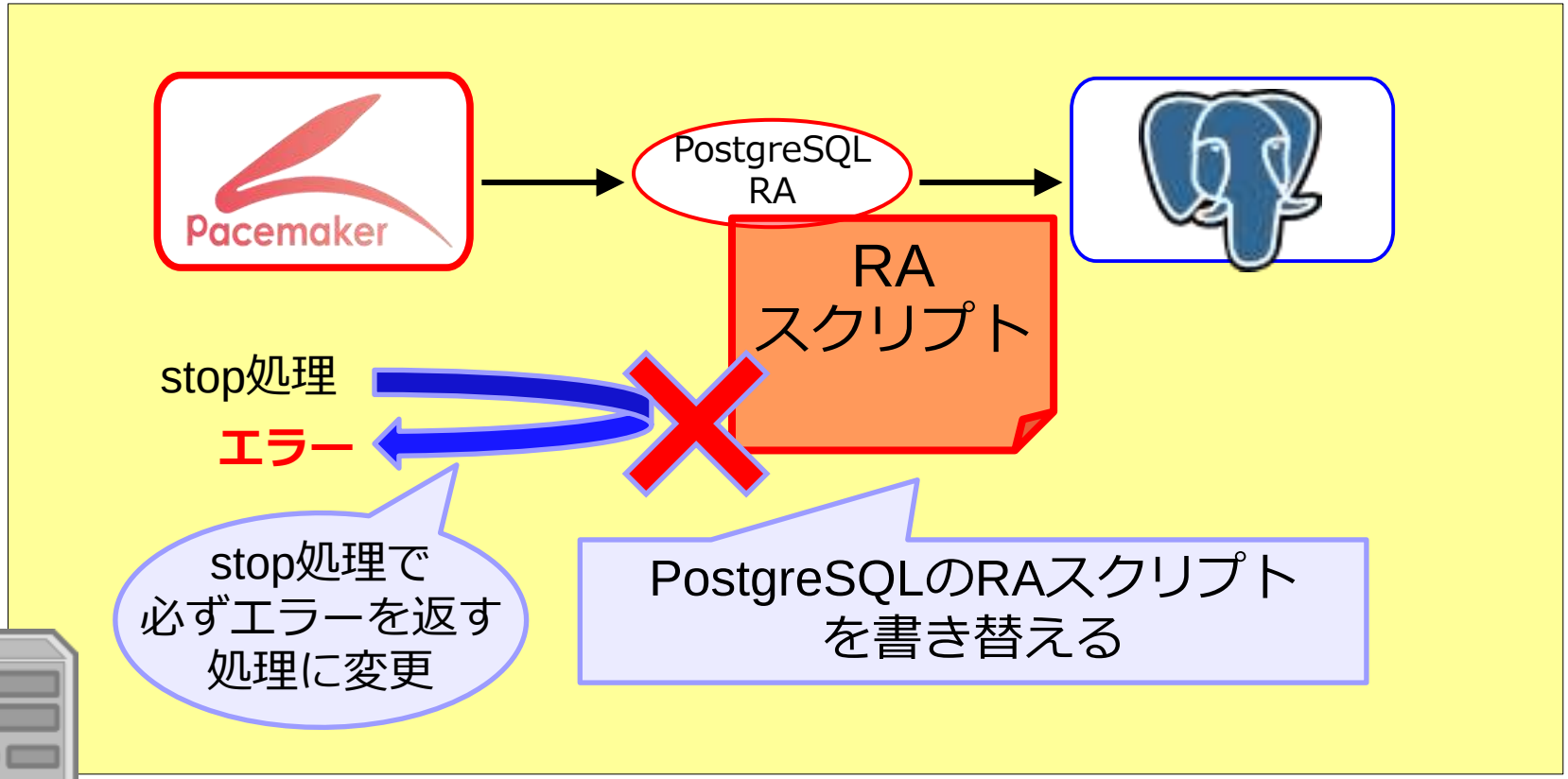
【6.リソース停止失敗】

PostgreSQL stop失敗

【6.リソース停止失敗】①発生手順イメージ

凡例 [1] リソース／プロセス再起動
[2] 通常フェイルオーバ
[3] STONITH後フェイルオーバ

故障項目	故障内容	Pacemakerの動作	発生手順	復旧手順
リソース停止失敗	PostgreSQL stop失敗	[3]	pgsql RAのstopメソッドを return $OCF_ERR_GENERICに書き換え	[パターン2] Pacemaker再起動 (+フェイルバック)



【6.リソース停止失敗】②発生手順(1/3)

発生手順

疑似RAの作成

- **pgsql RA原本のバックアップを作成する**

pgsql_bak：原本のバックアップ

```
# cp /usr/lib/ocf/resource.d/heartbeat/pgsql
/usr/lib/ocf/resource.d/heartbeat/pgsql_bak
```

pgsql RA の格納場所
/usr/lib/ocf/resource.d/
heartbeat/pgsql

- **pgsql RAの stopメソッドをエラーで終了するように書き換える**

```
# vi /usr/lib/ocf/resource.d/heartbeat/pgsql

 :
#pgsql_stop: pgsql_real_stop() wrapper for replication
pgsql_stop() {

#stopNG
return $OCF_ERR_GENERIC

    if ! is_replication; then
        pgsql_real_stop
        return $?
    else
        pgsql_replication_stop
        return $?
    fi
}
 :
```

追記

stop処理で
必ずエラーを返す
処理に変更

【6.リソース停止失敗】②発生手順(2/3)

発生手順

Pacemaker起動

➢ **Pacemakerを起動**

```
# systemctl start pacemaker
```

RA差替えタイミングは、
・stop故障の場合は、Pacemaker起動前後いずれでも問題ない
・monitor故障の場合は、必ずPacemaker起動後にRA差替えを行う
・start故障の場合は、必ずPacemaker起動前にRA差替えを行う

リソースグループの移動

➢ **リソースグループをサーバ2号機に移動させる**

✓ crm_resourceコマンドは、リソースを動的に操作(表示/設定/削除)する
✓ オプション： -M(リソースを指定ノードで起動するように切り替える制約追加) -r [リソースIDを指定] -N [ホスト名] -f(リソースを強制的に再配置) -Q(値のみ表示)

```
# crm_resource -M -r grpPostgreSQLDB -N srv02 -f -Q
```

リソースグループの移動手順の代わりに、PostgreSQLの故障を発生させても問題ない
⇒ 手順は【1.リソース故障】の発生手順を参照

リソース停止失敗の故障後動作

サーバ1号機のPostgreSQL「**リソースの停止処理失敗**」
↓
サーバ2号機からサーバ1号機への「**STONITH実行**」
↓
サーバ1号機停止後にサーバ2号機で「**リソース起動**」

【6.リソース停止失敗】②発生手順(3/3)

確認手順

リソース状態の確認 ➢ **リソース状態が “Started サーバ2号機” となっていることを確認**

```
# crm_mon -fA -L
   :
Online: [ srv02 ]
OFFLINE: [ srv01 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB        (ocf::heartbeat:sfex):          Started srv02
    prmFsPostgreSQLDB        (ocf::heartbeat:Filesystem):    Started srv02
    prmIpPostgreSQLDB        (ocf::heartbeat:IPaddr2):       Started srv02
    prmApPostgreSQLDB        (ocf::heartbeat:pgsql):         Started srv02
    :
Negative location constraints:
  cli-ban-grpPostgreSQLDB-on-srv01 prevents grpPostgreSQLDB from running on srv01
```

切戻し作業

リソースグループの切り戻し ➢ **サーバ1号機の実行不可制約を解除**

✓ オプション： -U(切り替えによる制約を解除) -r [リソースIDを指定]

```
# crm_resource -U -r grpPostgreSQLDB
```

リソース状態の確認 ➢ **実行不可制約解除を確認**

```
# crm_mon -fA -L
   :
Negative location constraints:
```

RA原本を戻す ➢ **pgsql RA原本に戻す**

```
# mv /usr/lib/ocf/resource.d/heartbeat/pgsql_bak
/usr/lib/ocf/resource.d/heartbeat/pgsql
```

※crm_mon表示は一部省略

【6.リソース停止失敗】③故障発生時の動作

【サーバ1号機】
① PostgreSQLリソースの障害発生
② PacemakerがPostgreSQLの異常を検知
③ PacemakerがPostgreSQLリソース停止に失敗

【サーバ2号機】
③′ Pacemakerがリソース異常を確認
④ PacemakerがSTONITHを実行

障害検知

PostgreSQL関連リソースの停止失敗

STONITH実行

【サーバ1号機】
⑤ サーバ停止(*1)

【サーバ2号機】
⑥ Pacemakerが共有ディスクのロック取得
⑦ 〃 共有ディスクのマウント
⑧ 〃 サービス用VIPを起動
⑨ 〃 PostgreSQLを起動

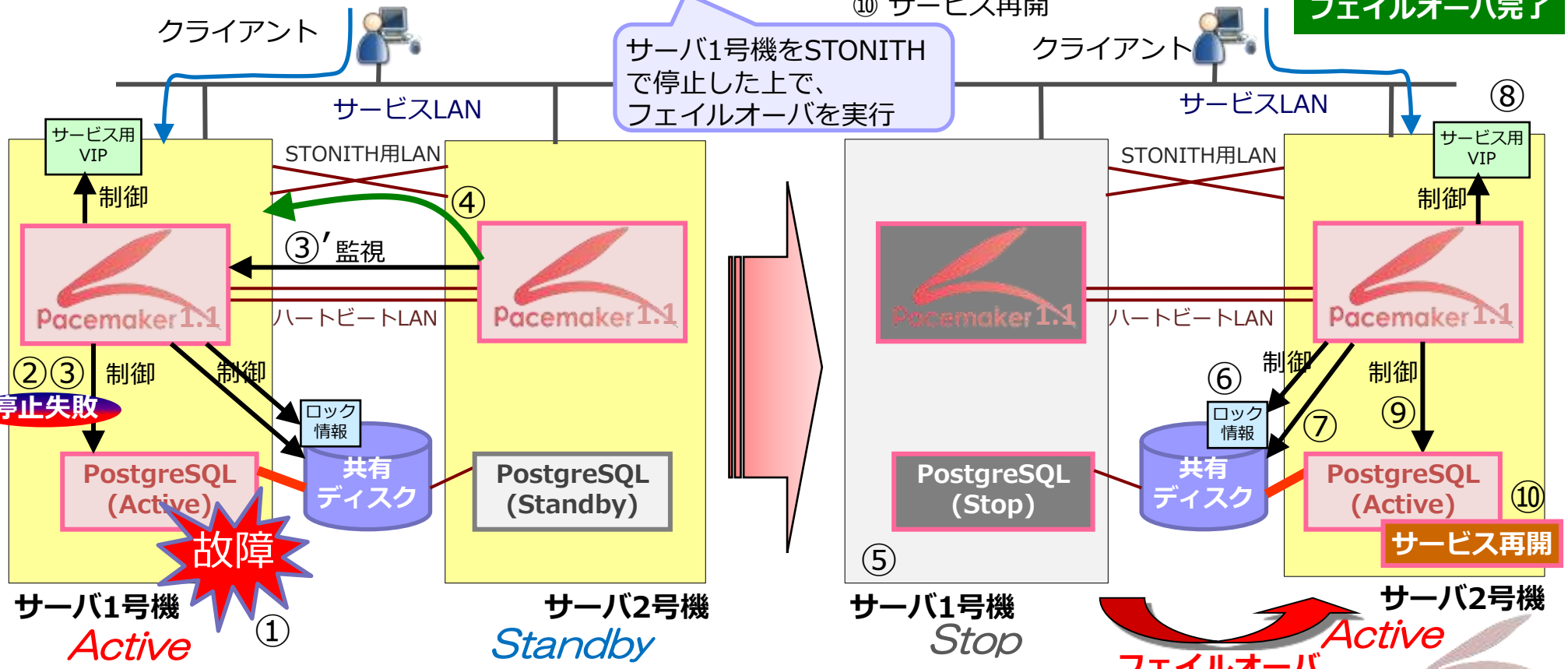
PostgreSQL関連リソースの起動完了

⑩ サービス再開

STONITH完了

フェイルオーバ完了

サーバ1号機をSTONITHで停止した上で、フェイルオーバを実行



(*1) STONITH動作を reboot に設定している場合は停止後に再起動される。

【6.リソース停止失敗】④pm_logconvのログ確認

故障後

【サーバ1号機】

srv01のリソース停止失敗

```
May 25 17:41:33 srv01    info: Resource prmApPostgreSQLDB tries to stop.
May 25 17:41:33 srv01    error: Resource prmApPostgreSQLDB failed to stop. (rc=1)
```

① サーバ1号機のPostgreSQLリソースの故障発生

② PacemakerがPostgreSQLの異常を検知

障害検知

③ PacemakerがPostgreSQLリソース停止に失敗

【サーバ2号機】

```
May 25 17:42:34 srv02    info: Try to execute STONITH device prmStonith1-1 on srv02 for reboot srv01.
May 25 17:42:38 srv02 warning: Failed to execute STONITH device prmStonith1-1 for srv01.
May 25 17:42:38 srv02    info: Try to execute STONITH device prmStonith1-2 on srv02 for reboot srv01.
May 25 17:42:41 srv02    info: Succeeded to execute STONITH device prmStonith1-2 for srv01.
May 25 17:42:41 srv02    info: Unset DC node srv01.
May 25 17:42:41 srv02 warning: Node srv01 is lost
May 25 17:42:41 srv02    info: Succeeded to STONITH (reboot) srv01 by srv02.
May 25 17:42:41 srv02    info: Set DC node to srv02.
May 25 17:42:42 srv02    error: Start to fail-over.
May 25 17:42:42 srv02    info: Resource prmExPostgreSQLDB tries to start.
May 25 17:43:53 srv02    info: Resource prmExPostgreSQLDB started. (rc=0)
May 25 17:43:53 srv02    info: Resource prmFsPostgreSQLDB tries to start.
May 25 17:43:53 srv02    info: Resource prmFsPostgreSQLDB started. (rc=0)
May 25 17:43:53 srv02    info: Resource prmIpPostgreSQLDB tries to start.
May 25 17:43:53 srv02    info: Resource prmIpPostgreSQLDB started. (rc=0)
May 25 17:43:53 srv02    info: Resource prmApPostgreSQLDB tries to start.
May 25 17:43:55 srv02    info: Resource prmApPostgreSQLDB started. (rc=0)
May 25 17:43:55 srv02    info: Resource prmExPostgreSQLDB : Started on srv02
May 25 17:43:55 srv02    info: Resource prmApPostgreSQLDB : Started on srv02
May 25 17:43:55 srv02    info: fail-over succeeded.
```

③′ Pacemakerがリソース異常を確認

④ PacemakerがSTONITHを実行

STONITH完了

⑤ サーバ停止

フェイルオーバ開始

⑥ Pacemakerが共有ディスクのロック取得

⑦ 〃 共有ディスクのマウント

⑧ 〃 サービス用VIPを起動

⑨ 〃 PostgreSQLを起動

2号機のPostgreSQL関連リソース起動

⑩ サービス再開

フェイルオーバ完了

【6.リソース停止失敗】⑤復旧手順(1/2)

復旧手順パターン2

【サーバ2号機】

手順1 ノード状態確認 ➢ **サーバ2号機で、リソース状態が“Started サーバ2号機”であることを確認**

```
# crm_mon -fA
   :
Online: [ srv02 ]
OFFLINE: [ srv01 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB        (ocf::heartbeat:sfex):        Started srv02
    prmFsPostgreSQLDB        (ocf::heartbeat:Filesystem):  Started srv02
    prmIpPostgreSQLDB        (ocf::heartbeat:IPaddr2):     Started srv02
    prmApPostgreSQLDB        (ocf::heartbeat:pgsql):       Started srv02
   :
```

フェイルオーバにより
サーバ2号機で起動

【サーバ1号機】

手順2 ノード起動 ➢ **サーバ1号機の電源が停止している場合は起動**

故障復旧

【サーバ1号機】

手順3 Pacemaker起動 ➢ **サーバ1号機のPacemakerを起動**

```
# systemctl start pacemaker
```

手順4 ノード状態確認 ➢ **サーバ1号機の状態が “Online” となっていることを確認**

```
# crm_mon -fA
   :
Online: [ srv01 srv02 ]
   :
```

※crm_mon表示は一部省略

【6.リソース停止失敗】⑤復旧手順(2/2)

復旧手順パターン2

【サーバ1号機】

手順5 リソースグループの切り戻し(1/2)

➢ **リソースグループをサーバ1号機に切り戻す**

✓ crm_resourceコマンドは、リソースを動的に操作(表示/設定/削除)する
✓ オプション： -M(リソースを指定ノードで起動するように切り替える制約追加) -r [リソースIDを指定] –N [ホスト名] –f(リソースを強制的に再配置) –Q(値のみ表示)

```
# crm_resource -M -r grpPostgreSQLDB -N srv01 -f -Q
```

手順6 リソース状態の確認

➢ **リソース状態が “Started サーバ1号機” となっていることを確認**

✓ リソースの実行不可制約がサーバ2号機に設定されていること

手順5でサーバ1号機にリソースを切り戻すため、サーバ2号機でリソース起動を行わない制約が設定されます。
切り戻し完了後に、その制約を解除しておく必要があります。

```
# crm_mon -fA -L
  :
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB        (ocf::heartbeat:sfex):        Started srv01
    prmFsPostgreSQLDB        (ocf::heartbeat:Filesystem):  Started srv01
    prmIpPostgreSQLDB        (ocf::heartbeat:IPaddr2):     Started srv01
    prmApPostgreSQLDB        (ocf::heartbeat:pgsql):       Started srv01
  :
Negative location constraints:
  cli-ban-grpPostgreSQLDB-on-srv02 prevents grpPostgreSQLDB from running on srv02
```

-L(実行不可制約表示)を付ける

手順7 リソースグループの切り戻し(2/2)

➢ **サーバ2号機の実行不可制約を解除**

✓ オプション： -U(切り替えによる制約を解除) -r [リソースIDを指定]

よく解除忘れが起こるので注意

```
# crm_resource -U -r grpPostgreSQLDB
```

手順8 リソース状態の確認

➢ **実行不可制約解除を確認**

```
# crm_mon -fA -L
  :
Negative location constraints:
```

リソース切り戻し時の実行不可制約の解除漏れを防止

※crm_mon表示は一部省略

【テーマ３】

実際の構築運用シーンで起きる
問題の“解決”方法

～よくある問題を理解しよう～

［テーマ３］よくある問題の実例

よくある問題はいつ起きるのか？

サポート担当者

初期構築時も、保守運用時も、どちらも問合せは多いですね。
保守運用段階では、緊急度の高い問合せも増える傾向があります。

システム構築運用の流れ

- システム設計
  - ・クラスタ構成の検討
  - ・故障対応の検討
  - ・クラスタ設計、設定検討
- システム構築
  - ・検証/商用環境の構築
- 動作検証
  - ・環境構築後の設定確認
  - ・クラスタ動作検証の実施
- システムリリース
- 保守監視
  - ・故障発生時の対応
- メンテナンス
  - ・定期メンテナンス時の対応

初期構築

- **設定**に関する問合せ
- **動作検証内容**に関する問合せ

保守運用

- 発生した**故障理由**の問合せ
- **復旧手順**の問合せ

[ケース1] 古い設定ファイルの削除漏れについて

ユーザ

サポート担当者

crmファイル（リソース定義ファイル）の設定を変更したが、正常に起動しない。

crmファイルを変更した場合は、**変更前の古い設定ファイル（ /var/lib/pacemaker/cib/ 配下）も削除**が必要です。
よく忘れるので注意が必要です！

【手順】 crmファイルの内容を変更した場合は以下の手順が必要になる。

(1) 「pm_crmgen環境定義書」を再修正し、新たなcsvファイルを生成する。
(2) 生成したcsvファイルから、pm_crmgenコマンドを使用して新たなcrmファイルを生成する。
(3) Pacemakerが停止している状態で、**/var/lib/pacemaker/cib/ 配下のファイルを全て削除**する。

```
# rm -f /var/lib/pacemaker/cib/*
```

Active/Standbyの両サーバで削除が必要

(4) Pacemakerを両サーバで続けて起動する。

```
# systemctl start pacemaker
```

(5) crmコマンドで新たなcrmファイルを反映する。

```
# cd
# crm options sort-elements no
# crm configure load update sample.crm
```

[ケース2] リソース切り戻し時の手順漏れについて

ユーザ

故障時に、フェイルオーバが発生せずに、サービス停止した。
そういえば、故障復旧作業を行ったばかりだけど‥‥。

サポート担当者

故障したサーバが復旧した時に、切り戻しを行う手順の一部が足りなかった可能性があります。
リソース切り戻しを行った場合は、**切り戻し元のサーバに設定される「リソース起動を行わない制約」を解除**し忘れると、再び、故障が起きた時にフェイルオーバできなくなってしまいます。

【手順】

※詳細はP50参照

(1) リソースグループをサーバ1号機に切り戻す

```
# crm_resource -M -r grpPostgreSQLDB -N srv01 -f -Q
```

```
# crm_mon -fA -L
   :
Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB          (ocf::heartbeat:sfex):          Started srv01
   :
Negative location constraints:
  cli-ban-grpPostgreSQLDB-on-srv02 prevents grpPostgreSQLDB from running on srv02
```

1号機に切り戻し完了

2号機に実行不可制約が設定された

(2) サーバ2号機の実行不可制約(リソース起動を行わない制約)を解除

実行不可制約が削除される

```
# crm_resource -U -r grpPostgreSQLDB
```

```
# crm_mon -fA -L
   :
Negative location constraints:
```

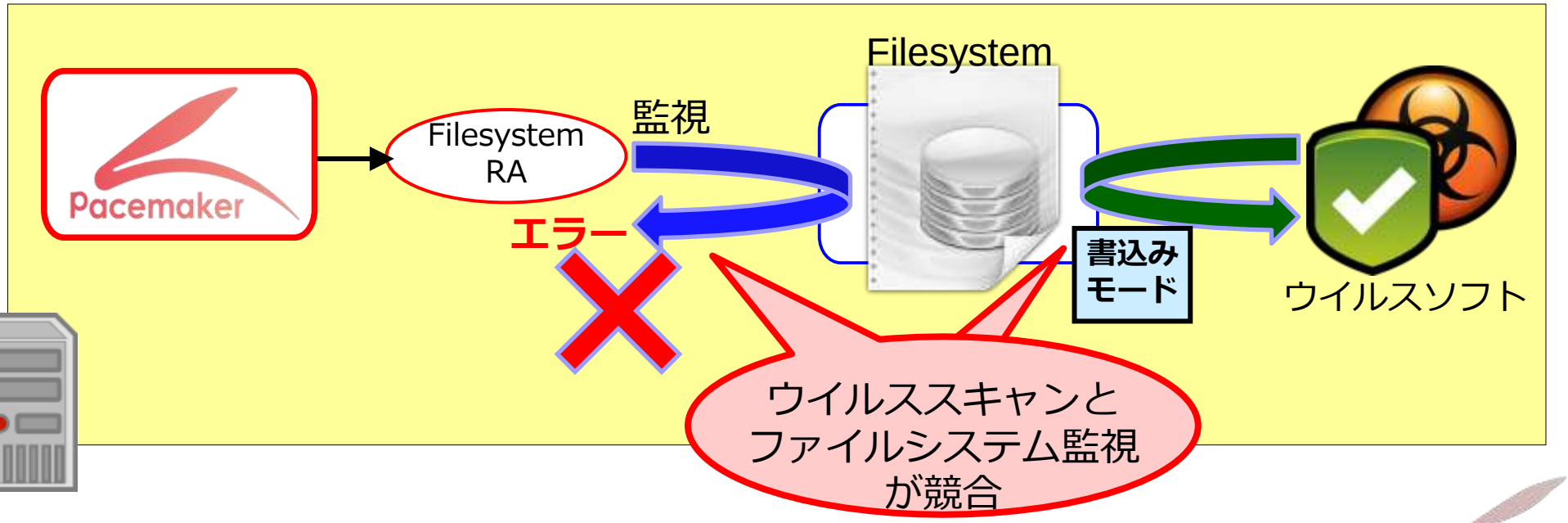
[ケース3] ウイルスソフト利用時のスキャン設定について

ユーザ

Pacemakerによるファイルシステム定期監視に失敗して、フェイルオーバが実行された。

サポート担当者

システム的な障害が確認されない場合は、**ウイルススキャンの処理が競合**していることも原因として考えられます。
対策として、**ウイルスソフトの書き込みモードでのスキャンを無効化**する設定が有効の場合もあります。



［お知らせ］Pacemaker-1.1.14の紹介

7/22に、Pacemaker-1.1.14-1.1 リポジトリパッケージをリリースしました。

■ Pacemaker-1.1.14-1.1の主な変更点

➢ 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、影響有無の確認が必要

  ※pm_logconvは本変更に対応済のため、pm_logconvログを監視している場合は影響なし

【Pacemaker-1.1.13】

```
Mar 22 16:10:05 srv01 crmd[15529]:  notice: process_lrm_event: Operation
prmApPostgreSQLDB_monitor_10000: not running (node=srv01, call=77, rc=7, cib-update=76, confirmed=false)
```

【Pacemaker-1.1.14】

関数名の削除

```
May 25 16:30:05 srv01 crmd[15539]:  notice: Operation prmApPostgreSQLDB_monitor_10000: not running
(node=srv01, call=77, rc=7, cib-update=76, confirmed=false)
```

➢ ネットワーク冗長化設定の変更

- ✓ 1.1.14-1.1 以降では、複数の interface を設定する場合は mcastaddr もしくは mcastport のいずれかを異なる値に設定することが必須 (1.1.13-1.1までの設定はそのまま使用不可)
- ✓ 設定例は、http://linux-ha.osdn.jp/wp/archives/4490 を参照

※その他の変更点は、Linux-HA Japanサイトのリリース情報をご覧ください。

［お知らせ］Linux-HA Japan のご紹介

- Pacemakerの日本公式コミュニティとして「**Linux-HA Japan**」を運営しています。
- Pacemaker関連の最新情報を日本語で発信しています。
  - ✓ 過去のOSC講演資料も公開中！
- Pacemakerのrpmパッケージ(*)の配布も行っています。

(*)関連パッケージをまとめ、インストールが楽なリポジトリパッケージを作成・公開しています。

http://linux-ha.osdn.jp/wp/

・・・最新情報発信、ML登録はこちらから

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

・・・rpmパッケージダウンロードはこちらから

［お知らせ］Linux-HA Japan のご紹介

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。

Linux-HA Japan

検索

付録

【付録１】主な pm_logconv出力ログ内容

【付録２】故障項目毎の故障発生手順・復旧手順

※「ネットワーク故障(サービスLAN)」「リソース停止失敗」は本編を参照してください。

	故障項目	故障内容	参照先
1	リソース故障	PostgreSQL故障	付録
2	ネットワーク故障	サービスLAN故障	(本編)
		ハートビートLAN故障	付録
3	ノード故障	カーネルパニック	付録
		サーバ電源停止	付録
4	Pacemaker プロセス故障	corosyncプロセス故障	付録
5	ディスク故障	内蔵ディスク故障	付録
		共有ディスクケーブル故障	
6	リソース停止失敗	PostgreSQL stop失敗	(本編)

※本資料上の故障時の動作は一例であり、個々の故障内容に応じて異なる動作の場合もあります。

【付録１】主なpm_logconv出力ログ内容(1/3)

➢ リソース起動・監視・停止

分類	状態	ログ出力内容	意味
リソース起動	成功	info: Resource prmApPostgreSQLDB started. (rc=0)	リソースID“prmApPostgreSQLDB”の起動(start)が正常に終了(rc=0)
	失敗	error: Resource prmApPostgreSQLDB failed to start. (rc=1)	リソースID“prmApPostgreSQLDB”の起動(start)でエラー発生(rc=1)
リソース監視	失敗	error: Resource prmApPostgreSQLDB failed to monitor. (rc=1)	リソースID“prmApPostgreSQLDB”の監視(monitor)でエラー発生(rc=1)
	失敗	error: Resource prmApPostgreSQLDB does not work. (rc=7)	リソースID“prmApPostgreSQLDB”の監視(monitor)でリソース停止検知(rc=7)
リソース停止	成功	info: Resource prmApPostgreSQLDB stopped. (rc=0)	リソースID“prmApPostgreSQLDB”の停止(stop)が正常に終了(rc=0)
	失敗	error: Resource prmApPostgreSQLDB failed to stop. (rc=1)	リソースID“prmApPostgreSQLDB”の停止(stop)でエラー発生(rc=1)

➢ ハートビートLAN状態

分類	状態	ログ出力内容	意味
ハートビートLAN	故障	warning: Ring number 0 is FAULTY (interface 192.168.103.1).	ハートビートLAN“ringnumber 0”の故障検知
	回復	info: Ring number 0 is recovered.	ハートビートLAN“ringnumber 0”の回復検知

主なpm_logconv出力ログ内容(2/3)

➢ ノード状態

分類	状態	ログ出力内容	意味
ノード状態	停止	warning: Node srv01 is lost	ノード“srv01”が故障／停止
	回復	info: Node srv01 is member	ノード“srv02”が起動／回復

➢ ネットワーク監視

分類	状態	ログ出力内容	意味
ネットワーク監視	故障	error: Network to 192.168.101.1 is unreachable.	監視先IPアドレス“192.168.101.1”に通信不可

➢ ディスク監視

分類	状態	ログ出力内容	意味
ディスク監視	停止	error: Disk connection to /dev/mapper/mpatha is ERROR. (attr_name=diskcheck_status)	/dev/mapper/mpatha に対するディスク監視(属性値=diskcheck_status)で故障(ERROR)検知

主なpm_logconv出力ログ内容(3/3)

➢ フェイルオーバ動作

分類	状態	ログ出力内容	意味
フェイルオーバ	開始	error: Start to fail-over.	フェイルオーバ開始
	成功	info: fail-over succeeded.	フェイルオーバ成功
	失敗	error: fail-over failed.	フェイルオーバ失敗

➢ STONITH動作

分類	状態	ログ出力内容	意味
STONITH処理	開始	info: Try to STONITH (reboot) srv02.	ノード“srv02”に対するSTONITH処理実行
	開始	info: Try to execute STONITH device prmStonithN2-1 on srv01 for reboot srv02.	ノード“srv01”上のSTONITHデバイス“prmStonithN2-1”からノード“srv02”に対する実行
	成功	info: Succeeded to STONITH (reboot) srv02 by srv01.	ノード“srv01”からノード“srv02”に対するSTONITH処理成功
	成功	info: Succeeded to execute STONITH device prmStonithN2-2 for srv02.	STONITHデバイス“prmStonithN2-2”からノード“srv02”に対する実行成功
	失敗	error: Failed to STONITH (reboot) srv02 by srv01.	ノード“srv01”からノード“srv02”に対するSTONITH処理失敗
	失敗	warning: Failed to execute STONITH device prmStonithN2-1 for srv02.	STONITHデバイス“prmStonithN2-1”からノード“srv02”に対する実行失敗

【付録２】故障項目毎の故障発生手順・復旧手順

凡例
[1] リソース／プロセス再起動
[2] 通常フェイルオーバ
[3] STONITH後フェイルオーバ

	故障項目	故障内容	Pacemakerの動作	故障発生手順	復旧手順
1	**リソース故障**	PostgreSQL故障	[1] or [2]	$ pg_ctl -m i stop （または、# kill -9 PID[PostgreSQL]）	[パターン１’] (フェイルバック)
2	**ネットワーク故障**	サービスLAN故障	[2]	# iptables -A INPUT -i [S-LAN_IF] -j DROP; iptables -A OUTPUT -o [S-LAN_IF] -j DROP （または、ネットワークケーブルの抜線）	[パターン１] (フェイルバック)
		ハートビートLAN故障	[3]	# iptables -A INPUT -i [HB-LAN1_IF] -j DROP; iptables -A OUTPUT -o [HB-LAN1_IF] -j DROP # iptables -A INPUT -i [HB-LAN2_IF] -j DROP; iptables -A OUTPUT -o [HB-LAN2_IF] -j DROP	[パターン２’] Pacemaker再起動
3	**ノード故障**	カーネルパニック	[3]	# echo c > /proc/sysrq-trigger	[パターン２] Pacemaker再起動 (＋フェイルバック)
		サーバ電源停止	[3]	# poweroff -nf	
4	**Pacemakerプロセス故障**	corosyncプロセス故障	[3]	# pkill -9 corosync	
5	**ディスク故障**	内蔵ディスク故障	[2] or [3]	内蔵ディスク引き抜き	[パターン３] 強制電源断 ＋Pacemaker再起動 (＋フェイルバック)
		共有ディスクケーブル故障	[2]	ディスクケーブル引き抜き	
6	**リソース停止失敗**	PostgreSQL stop失敗	[3]	pgsql RAのstopメソッドを return $OCF_ERR_GENERICに書き換え	[パターン２] Pacemaker再起動 (＋フェイルバック)

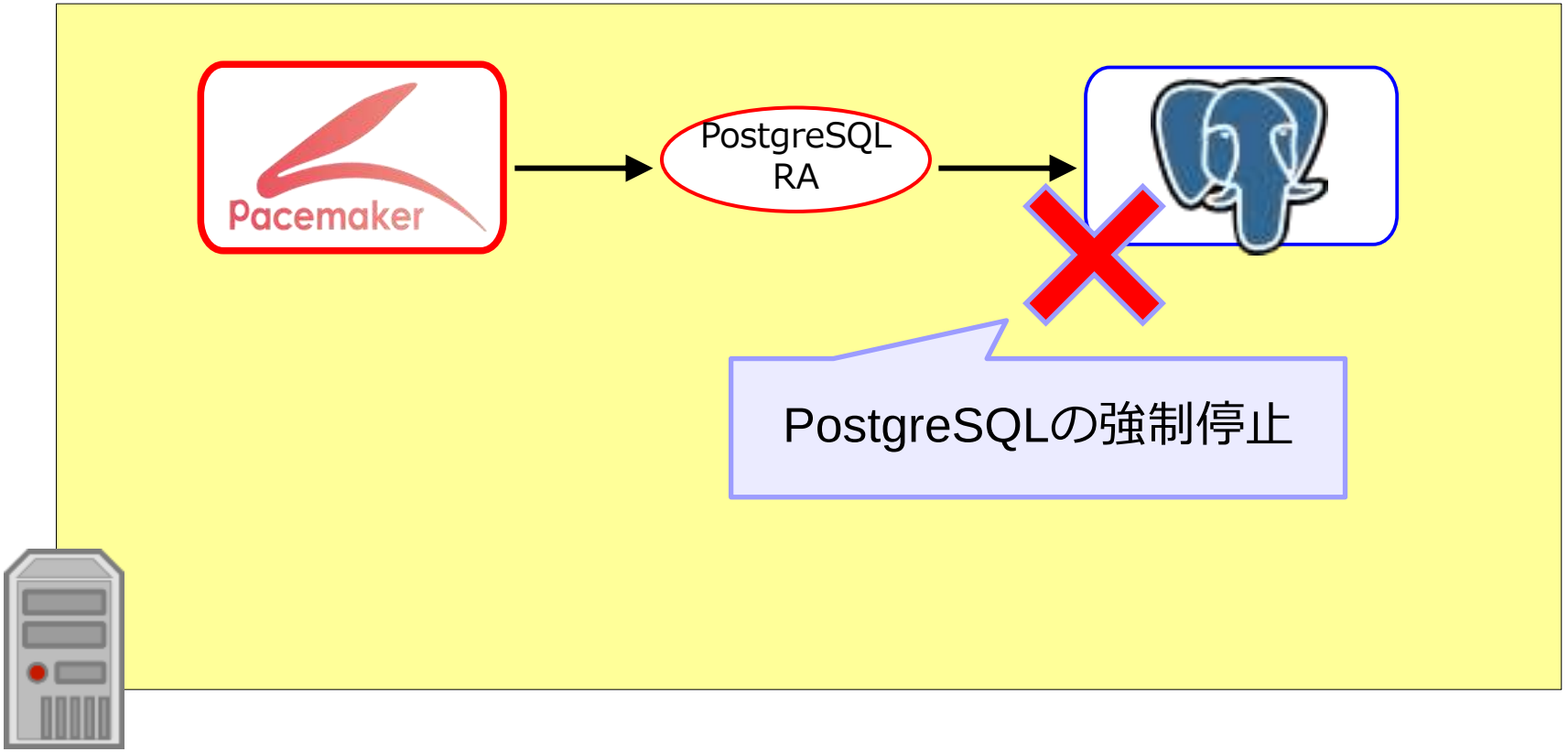
✓【付録２】では青枠のケースを取り上げています。
✓「ネットワーク故障(サービスLAN)」「リソース停止失敗」は本編を参照してください。

【１.リソース故障】

【1.リソース故障】①発生手順イメージ

凡例 [1] リソース／プロセス再起動
[2] 通常フェイルオーバ
[3] STONITH後フェイルオーバ

故障項目	故障内容	Pacemakerの動作	故障発生手順	復旧手順
リソース故障	PostgreSQL故障	[1] or [2]	$ pg_ctl -m i stop (または、# kill -9 PID[PostgreSQL])	[パターン1’] (フェイルバック)



【1.リソース故障】②発生手順

発生手順

PostgreSQL故障

- **PostgreSQLの起動を確認**

```
# ps -ef | grep postgres
   :
postgres   627     1  0 17:08 ?  00:00:00 /usr/pgsql-9.5/bin/postgres
postgres   666   627  0 17:08 ?  00:00:00 postgres: logger process
```

- **PostgreSQLの強制停止を実行（postgreユーザで実行）**

```
$ pg_ctl -m i stop
```

- **PostgreSQLが起動していないことを確認**

```
# ps -ef | grep postgres
```

確認手順

ノード状態確認

- **PostgreSQLリソースがサーバ2号機で起動していることを確認**

```
# crm_mon -fA
   :
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
     prmExPostgreSQLDB          (ocf::heartbeat:sfex):          Started srv02
     prmFsPostgreSQLDB          (ocf::heartbeat:Filesystem):    Started srv02
     prmIpPostgreSQLDB          (ocf::heartbeat:IPaddr2):       Started srv02
     prmApPostgreSQLDB          (ocf::heartbeat:pgsql):         Started srv02
     :
Migration summary:
* Node srv01:
   prmApPostgreSQLDB: migration-threshold=1 fail-count=1 last-failure='Wed May 25 16:30:05 2016'
* Node srv02:

Failed actions:
    prmApPostgreSQLDB_monitor_10000 on srv01 'not running' (7): call=77, status=complete, exit-reason='none',
last-rc-change='Wed May 25 16:30:05 2016', queued=0ms, exec=0ms
```

フェイルオーバにより
サーバ2号機で起動

※crm_mon表示は一部省略

【１.リソース故障】③故障発生時の動作

【サーバ1号機】

① PostgreSQLリソースの障害発生
② PacemakerがPostgreSQLの異常を検知
③ PacemakerがPostgreSQLを停止
④ 〃 サービス用VIPを停止
⑤ 〃 共有ディスクのアンマウント
⑥ 〃 共有ディスクのロック解除

障害検知

PostgreSQL関連リソースの停止完了

【サーバ2号機】

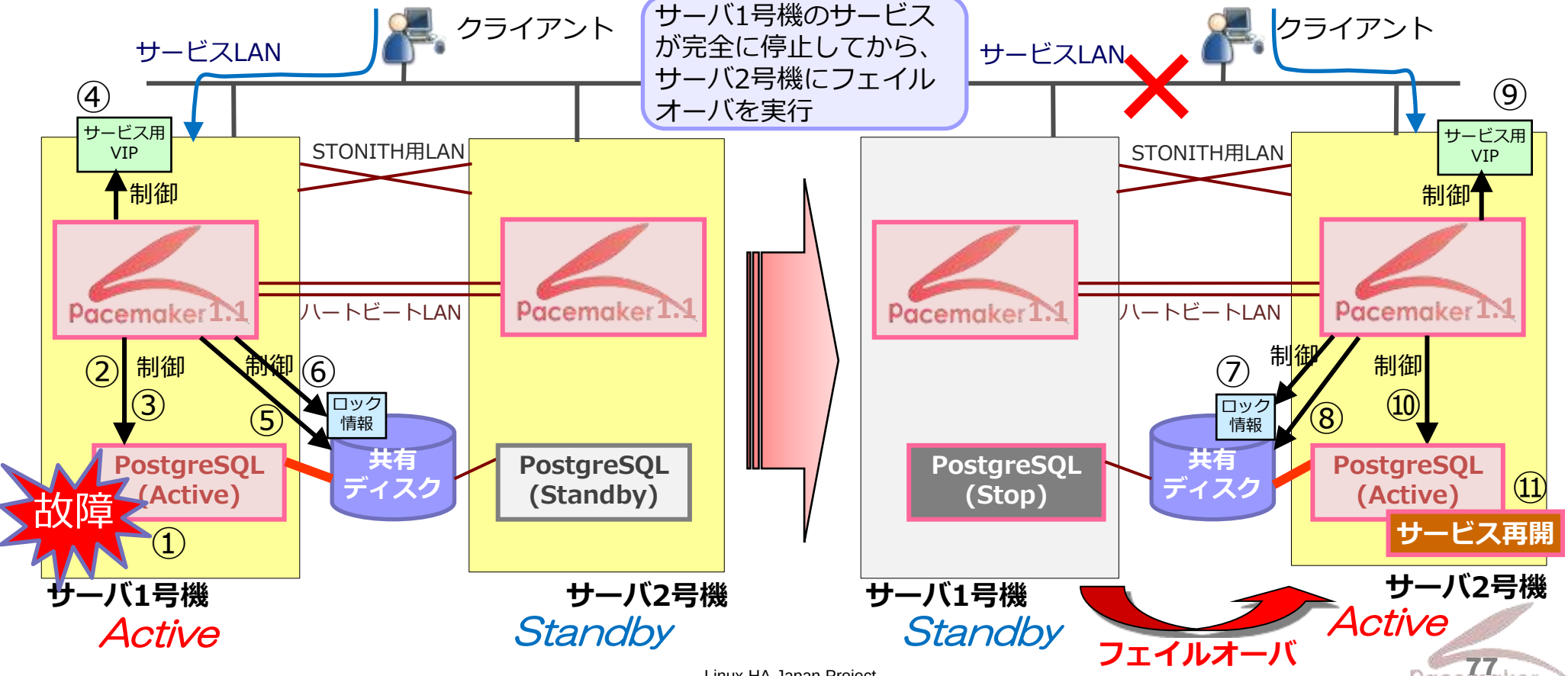
⑦ Pacemakerが共有ディスクのロック取得
⑧ 〃 共有ディスクのマウント
⑨ 〃 サービス用VIPを起動
⑩ 〃 PostgreSQLを起動

PostgreSQL関連リソースの起動完了

⑪ サービス再開

フェイルオーバ完了

サーバ1号機のサービスが完全に停止してから、サーバ2号機にフェイルオーバを実行



【１.リソース故障】④pm_logconvのログ確認

故障後

srv01でprmApPostgreSQLDBリソースのmonitor故障が発生

【サーバ1号機】

```
May 25 16:30:05 srv01   error: Resource prmApPostgreSQLDB does not work. (rc=7)
May 25 16:30:05 srv01   error: Start to fail-over.
May 25 16:30:05 srv01    info: Resource prmApPostgreSQLDB tries to stop.
May 25 16:30:05 srv01    info: Resource prmApPostgreSQLDB stopped. (rc=0)
May 25 16:30:05 srv01    info: Resource prmIpPostgreSQLDB tries to stop.
May 25 16:30:05 srv01    info: Resource prmIpPostgreSQLDB stopped. (rc=0)
May 25 16:30:05 srv01    info: Resource prmFsPostgreSQLDB tries to stop.
May 25 16:30:05 srv01    info: Resource prmFsPostgreSQLDB stopped. (rc=0)
May 25 16:30:05 srv01    info: Resource prmExPostgreSQLDB tries to stop.
May 25 16:30:05 srv01    info: Resource prmExPostgreSQLDB stopped. (rc=0)
```

① PostgreSQLリソースの障害発生

② PacemakerがPostgreSQLの異常を検知

障害検知

フェイルオーバ開始

③ PacemakerがPostgreSQLを停止

④ 〃 サービス用VIPを停止

⑤ 〃 共有ディスクのアンマウント

⑥ 〃 共有ディスクのロック解除

PostgreSQL関連リソースの停止完了

【サーバ2号機】

```
May 25 16:30:05 srv02    info: Resource prmExPostgreSQLDB tries to start.
May 25 16:30:06 srv02    info: Resource prmExPostgreSQLDB started. (rc=0)
May 25 16:30:06 srv02    info: Resource prmFsPostgreSQLDB tries to start.
May 25 16:30:06 srv02    info: Resource prmFsPostgreSQLDB started. (rc=0)
May 25 16:30:07 srv02    info: Resource prmIpPostgreSQLDB tries to start.
May 25 16:30:07 srv02    info: Resource prmIpPostgreSQLDB started. (rc=0)
May 25 16:30:07 srv02    info: Resource prmApPostgreSQLDB tries to start.
May 25 16:30:08 srv02    info: Resource prmApPostgreSQLDB started. (rc=0)
```

⑦ Pacemakerが共有ディスクのロック取得

⑧ 〃 共有ディスクのマウント

⑨ 〃 サービス用VIPを起動

⑩ 〃 PostgreSQLを起動

PostgreSQL関連リソースの起動完了

※DCノード(*1)で出力

```
May 25 16:30:08 srv01    info: Resource prmExPostgreSQLDB : Move srv01 -> srv02
May 25 16:30:08 srv01    info: Resource prmApPostgreSQLDB : Move srv01 -> srv02
May 25 16:30:08 srv01    info: fail-over succeeded.
```

⑪ サービス再開

フェイルオーバ完了

(*1) クラスタを統括するノードをDCノードと呼ぶ。

【1.リソース故障】⑤復旧手順(1/3)

復旧手順パターン1'

手順1 ノード状態確認

➢ **リソース状態が "Started サーバ2号機" となっていることを確認**

```
# crm_mon -fA
   :
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB        (ocf::heartbeat:sfex):
    prmFsPostgreSQLDB        (ocf::heartbeat:Filesystem):
    prmIpPostgreSQLDB        (ocf::heartbeat:IPaddr2):
    prmApPostgreSQLDB        (ocf::heartbeat:pgsql):
   :
Migration summary:
* Node srv01:
   prmApPostgreSQLDB: migration-threshold=1 fail-count=1 last-failure='Wed May 25 16:30:05 2016'
* Node srv02:

Failed actions:
    prmApPostgreSQLDB_monitor_10000 on srv01 'not running' (7): call=77, status=complete, exit-reason='none',
last-rc-change='Wed May 25 16:30:05 2016', queued=0ms, exec=0ms
```

フェイルオーバにより
サーバ2号機で起動

Started srv02
Started srv02
Started srv02
Started srv02

手順2 ACT化抑止

➢ **故障復旧作業中に、サーバ1号機がACT状態へ遷移しないよう抑止**

✓ crm_standbyコマンドは、ノードのステータス(Online/OFFLINE/standby)制御を行う
✓ オプション： -U [ノードのホスト名] -v [ステータスをstandbyにするか否かを指定]

```
# crm_standby -U srv01 -v on
```

手順3 ノード状態確認

➢ **サーバ1号機の状態が "standby" となっていることを確認**

```
# crm_mon -fA
   :
Node srv01: standby
 Online: [ srv02 ]
   :
```

安全に復旧作業を行う準備完了！

※crm_mon表示は一部省略

【1.リソース故障】⑤復旧手順(2/3)

復旧手順パターン1'

故障復旧

手順4 故障回数のクリア ➢ **故障リソースの故障回数とエラーステータスをクリア**

✓ crm_resourceコマンドは、リソースを動的に操作(表示/設定/削除)する
✓ オプション： -C(エラーステータスクリア) -r [リソースIDを指定] –N [ホスト名]

故障回数をクリアして、リソース監視を初期状態に戻します。

```
# crm_resource -C -r prmApPostgreSQLDB -N srv01
```

手順5 ACT化抑止の解除 ➢ **サーバ1号機がACT状態へ遷移できるように抑止を解除**

✓ crm_standbyコマンドは、ノードのステータス(Online/OFFLINE/standby)制御を行う
✓ オプション： -U [ノードのホスト名] -v [ステータスをstandbyにするか否かを指定]

```
# crm_standby -U srv01 -v off
```

手順6 ノード状態・故障回数の確認 ➢ **サーバ1号機の状態が"Online"となっていることを確認**

✓ 現用機の"Migration summary"に何も表示されていないことを確認

```
# crm_mon -fA
   :
Online: [ srv01 srv02 ]
   :
Migration summary:
* Node srv02:
* Node srv01:
```

復旧作業前の状態戻し完了！

【1.リソース故障】⑤復旧手順(3/3)

復旧手順パターン1'

手順7 リソースグループの切り戻し(1/2)

➢ **リソースグループをサーバ1号機に切り戻す**

✓ crm_resourceコマンドは、リソースを動的に操作(表示/設定/削除)する
✓ オプション： -M(リソースを指定ノードで起動するように切り替える制約追加) -r [リソースIDを指定] –N [ホスト名] –f(リソースを強制的に再配置) –Q(値のみ表示)

```
# crm_resource -M -r grpPostgreSQLDB -N srv01 -f -Q
```

手順8 リソース状態の確認

➢ **リソース状態が “Started サーバ1号機” となっていることを確認**

✓ リソースの実行不可制約がサーバ2号機に設定されていること

手順7でサーバ1号機にリソースを切り戻すため、サーバ2号機でリソース起動を行わない制約が設定されます。
切り戻し完了後に、その制約を解除しておく必要があります。

```
# crm_mon -fA -L
  :
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB       (ocf::heartbeat:sfex):          Started srv01
    prmFsPostgreSQLDB       (ocf::heartbeat:Filesystem):    Started srv01
    prmIpPostgreSQLDB       (ocf::heartbeat:IPaddr2):       Started srv01
    prmApPostgreSQLDB       (ocf::heartbeat:pgsql):         Started srv01
  :
Negative location constraints:
  cli-ban-grpPostgreSQLDB-on-srv02 prevents grpPostgreSQLDB from running on srv02
```

-L(実行不可制約表示)を付ける

手順9 リソースグループの切り戻し(2/2)

➢ **サーバ2号機の実行不可制約を解除**

✓ オプション： -U(切り替えによる制約を解除) -r [リソースIDを指定]

よく解除忘れが起こるので注意

```
# crm_resource -U -r grpPostgreSQLDB
```

手順10 リソース状態の確認

➢ **実行不可制約の解除を確認**

```
# crm_mon -fA -L
  :
Negative location constraints:
```

リソース切り戻し時の実行不可制約の解除漏れを防止

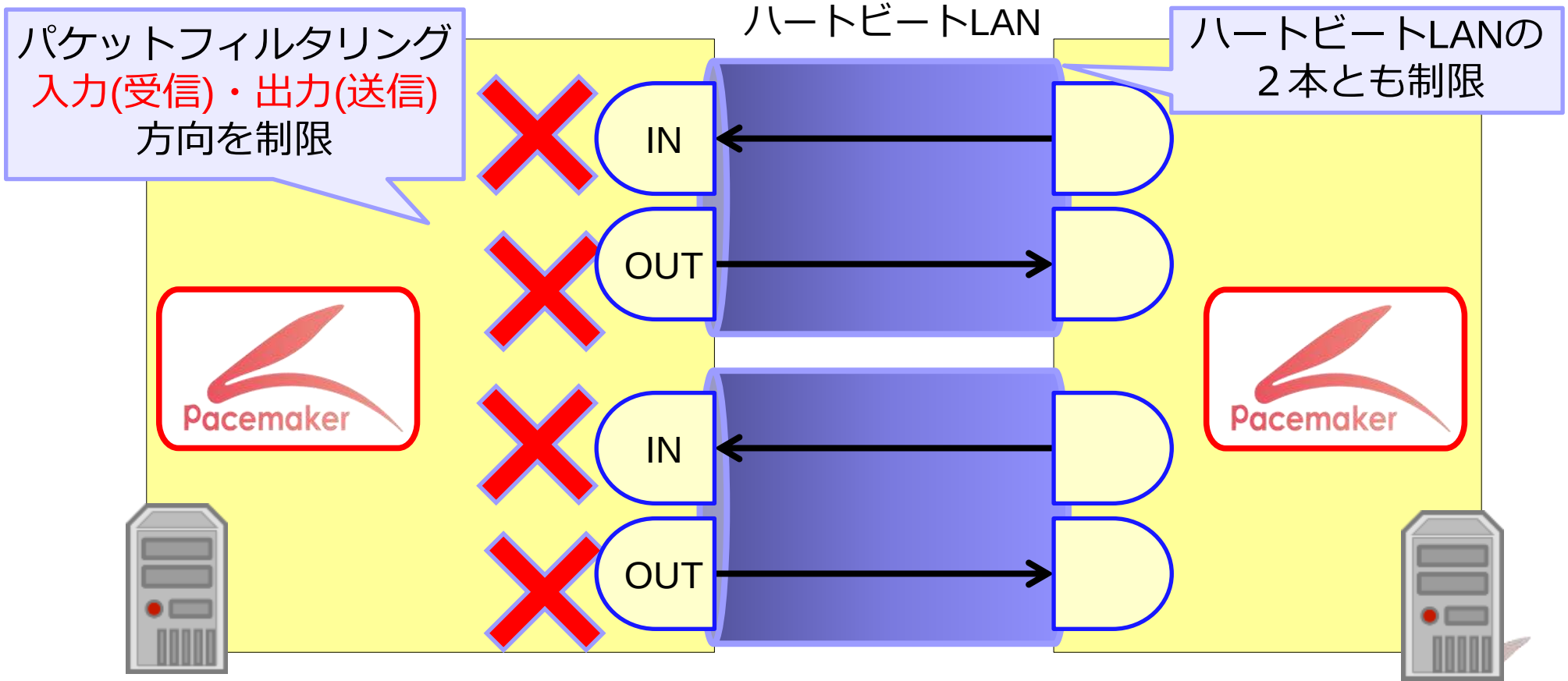
【2.ネットワーク故障】

ハートビートLAN故障

【2.ネットワーク故障-2】①発生手順イメージ

凡例[1] リソース／プロセス再起動
[2] 通常フェイルオーバ
[3] STONITH後フェイルオーバ

故障項目	故障内容	Pacemakerの動作	故障発生手順	復旧手順
ネットワーク故障	ハートビートLAN故障	[3]	# iptables -A INPUT -i [HB-LAN1_IF] -j DROP; iptables -A OUTPUT -o [HB-LAN1_IF] -j DROP # iptables -A INPUT -i [HB-LAN2_IF] -j DROP; iptables -A OUTPUT -o [HB-LAN2_IF] -j DROP （または、ネットワークケーブルの抜線）	[パターン2’] Pacemaker再起動



【2.ネットワーク故障-2】②発生手順(1/2)

発生手順

ハートビートLAN故障

➢ **ハートビートLAN不通を起こすため、パケットフィルタリングを設定**
- ✓ サブコマンド： -A(ルールを追加)
- ✓ オプション： -i/-o [入力/出力ネットワークインタフェースを指定]
  -j [ルールにマッチした場合の動作を指定]

IN/OUT双方向の通信を切断すること

```
# iptables -A INPUT -i [HB-LAN1_IF] -j DROP; iptables -A OUTPUT -o
[HB-LAN1_IF] -j DROP
# iptables -A INPUT -i [HB-LAN2_IF] -j DROP; iptables -A OUTPUT -o
[HB-LAN2_IF] -j DROP
```

ネットワーク不通の方法として**「ifdownコマンド」の手順は選択しないこと**。
ifdownコマンドによりネットワーク不通とした場合、実環境のネットワーク断とは異なる動作となり、復旧手順も異なリます。
つまり、ifdownコマンドでは運用時の障害を想定した動作検証が十分に行えないため、iptablesコマンド、またはケーブル抜線を行ってください。

確認手順

NW状態確認

➢ **パケットフィルタリングの設定状況を確認**
- ✓ サブコマンド： -L(ルールを表示)

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
DROP       all  --  anywhere             anywhere
DROP       all  --  anywhere             anywhere
```

IN/OUT方向共にDROPが設定されている

【2.ネットワーク故障-2】②発生手順(2/2)

確認手順

ノード状態確認

➢ **スプリットブレイン(*1)が発生するため、STONITHによりサーバ2号機が強制電源断となり、サーバ1号機のみで起動していることを確認**

STONITHにより
サーバ2号機が停止

```
# crm_mon -fA
   :
Online: [ srv01 ]
OFFLINE: [ srv02 ]

Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB       (ocf::heartbeat:sfex):          Started srv01
    prmFsPostgreSQLDB       (ocf::heartbeat:Filesystem):    Started srv01
    prmIpPostgreSQLDB       (ocf::heartbeat:IPaddr2):       Started srv01
    prmApPostgreSQLDB       (ocf::heartbeat:pgsql):         Started srv01
Resource Group: grpStonith2
    prmStonith2-1           (stonith:external/stonith-helper):  Started srv01
    prmStonith2-2           (stonith:external/ipmi):            Started srv01
Clone Set: clnPing [prmPing]
    Started: [ srv01 ]
Clone Set: clnDiskd [prmDiskd]
    Started: [ srv01 ]
```

STONITHにより
サーバ1号機で継続起動

回復手順

ハートビートLAN
故障回復

➢ **ハートビートLAN不通のパケットフィルタリングを解除**

✓ サブコマンド： -F(ルールを解除)、 -L(ルールを表示)

```
# iptables -F
```

確認手順

NW状態確認

```
# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

IN/OUT方向共に
DROPが解除されている

(*1) スプリットブレインとは、ハートビートLAN故障等で他クラスタノードの認識ができなくなる状態のこと。両系起動を防ぐため、Active側のサーバから優先的にSTONITH(強制電源断)を行うことで、Standby側のサーバを停止します。

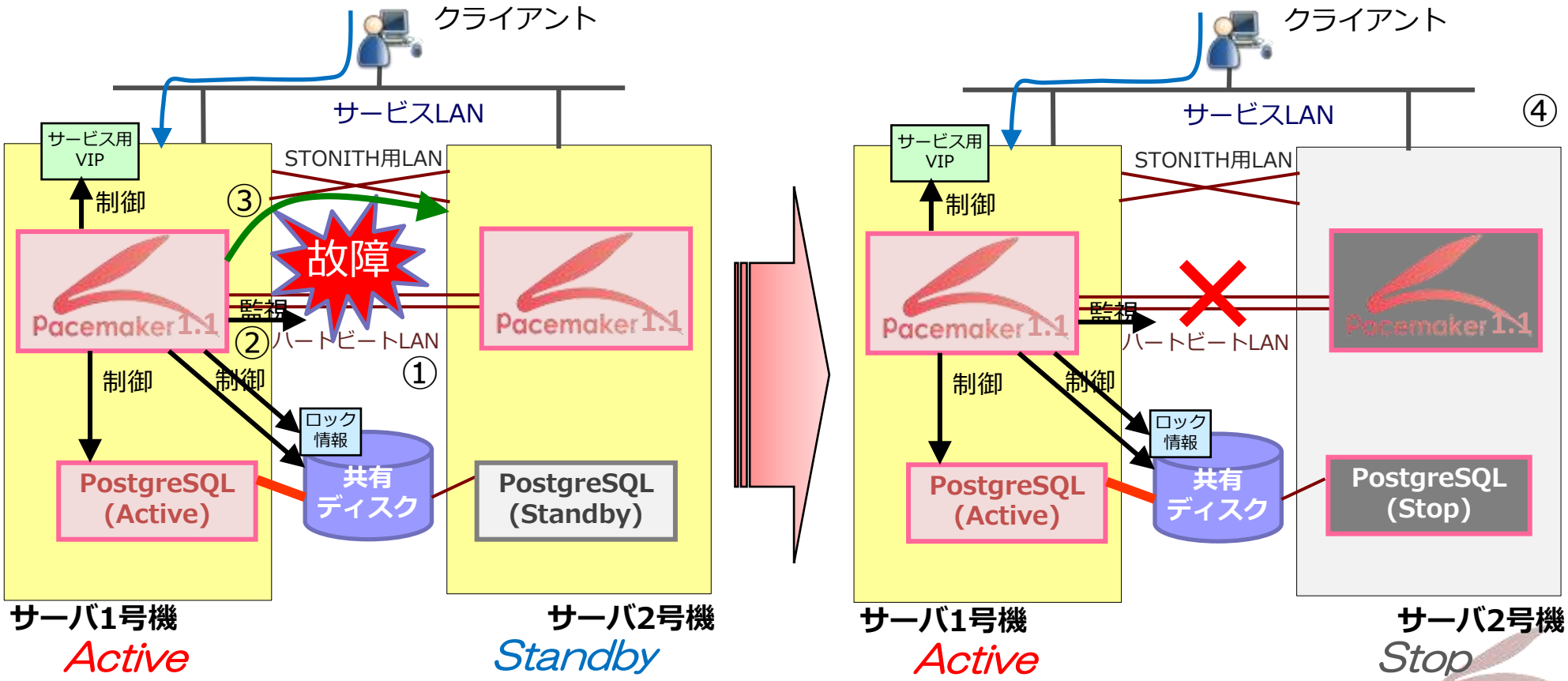
※crm_mon表示は一部省略

【2.ネットワーク故障-2】③故障発生時の動作

【サーバ1号機】
① ハートビートLANの障害発生
② Pacemakerがサーバ2号機の異常を検知
障害検知
③ Pacemakerがサーバ2号機のSTONITHを実行
STONITH実行

【サーバ2号機】
④ サーバ停止(*1)
STONITH完了

ハートビートLAN故障時は、両系起動を抑止するため、
サーバ2号機をSTONITHで強制停止する



(*1) STONITH動作を reboot に設定している場合は停止後に再起動される。

【2.ネットワーク故障-2】④pm_logconvのログ確認

故障後

srv02の故障を検知

【サーバ1号機】

```
May 25 17:54:56 srv01    info: Unset DC node srv02.
May 25 17:54:56 srv01 warning: Node srv02 is lost
May 25 17:54:56 srv01    info: Set DC node to srv01.
May 25 17:54:57 srv01    info: Try to STONITH (reboot) srv02.
May 25 17:54:58 srv01    info: Try to execute STONITH device prmStonith2-1 on srv01 for
reboot srv02.
May 25 17:55:02 srv01 warning: Failed to execute STONITH device prmStonith2-1 for srv02.
May 25 17:55:02 srv01    info: Try to execute STONITH device prmStonith2-2 on srv01 for
reboot srv02.
May 25 17:55:04 srv01    info: Succeeded to execute STONITH device prmStonith2-
2 for srv02.
May 25 17:55:04 srv01    info: Succeeded to STONITH (reboot) srv02 by srv01.
```

① ハートビートLANの障害発生

② Pacemakerがハートビート LANの異常を検知(*1)

障害検知

③ PacemakerがSTONITHを実行

STONITH実行

④ 2号機のサーバ停止

STONITH完了

【サーバ2号機】

（ログ出力なし）

(*1) ハートビートLAN故障を表す以下のログは、冗長化している ハートビートLAN の片方のインタフェースが故障した場合のみ出力されます。

```
warning: Ring number 0 is FAULTY (interface 192.168.XXX.XXX).
```

本手順のように、ハートビートLAN が全断する障害の場合は、対向ノード確認ができないログ出力で確認してください。

```
warning: Node srv02 is lost
```

【2.ネットワーク故障-2】⑤復旧手順(1/2)

復旧手順パターン2’

【サーバ1号機】

手順1 ノード状態確認

- **サーバ2号機の状態が “OFFLINE” であることを確認**
- **リソース状態が “Started サーバ1号機” となっていることを確認**

STONITHにより
サーバ2号機が停止

STONITHにより
サーバ1号機で継続起動

```
# crm_mon -fA
    :
Online: [ srv01 ]
OFFLINE: [ srv02 ]

 Resource Group: grpPostgreSQLDB
     prmExPostgreSQLDB          (ocf::heartbeat:sfex):          Started srv01
     prmFsPostgreSQLDB          (ocf::heartbeat:Filesystem):    Started srv01
     prmIpPostgreSQLDB          (ocf::heartbeat:IPaddr2):       Started srv01
     prmApPostgreSQLDB          (ocf::heartbeat:pgsql):         Started srv01
 Resource Group: grpStonith2
     prmStonith2-1              (stonith:external/stonith-helper):  Started srv01
     prmStonith2-2              (stonith:external/ipmi):            Started srv01
 Clone Set: clnPing [prmPing]
     Started: [ srv01 ]
 Clone Set: clnDiskd [prmDiskd]
     Started: [ srv01 ]

Node Attributes:
* Node srv01:
    + default_ping_set                : 100
    + diskcheck_status                : normal

Migration summary:
* Node srv01:
```

【サーバ2号機】

手順2 ノード起動

- **サーバ2号機の電源が停止している場合は起動**

※crm_mon表示は一部省略

【2.ネットワーク故障-2】⑤復旧手順(2/2)

復旧手順パターン2'

故障復旧

【サーバ2号機】

手順3 Pacemaker起動 ➢ **サーバ2号機のPacemakerを起動**

```
# systemctl start pacemaker
```

手順4 ノード状態確認 ➢ **ノード状態を確認し、2号機の状態が "Online" であることを確認**

```
# crm_mon -fA
  :
Online: [ srv01 srv02 ]

Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB       (ocf::heartbeat:sfex):          Started srv01
    prmFsPostgreSQLDB       (ocf::heartbeat:Filesystem):    Started srv01
    prmIpPostgreSQLDB       (ocf::heartbeat:IPaddr2):       Started srv01
    prmApPostgreSQLDB       (ocf::heartbeat:pgsql):         Started srv01
     :
Clone Set: clnPing [prmPing]
    Started: [ srv01 srv02 ]
Clone Set: clnDiskd [prmDiskd]
    Started: [ srv01 srv02 ]
     :
```

ハートビートLAN故障時は、サーバ2号機をSTONITHで停止するため、切り戻し手順は不要です。

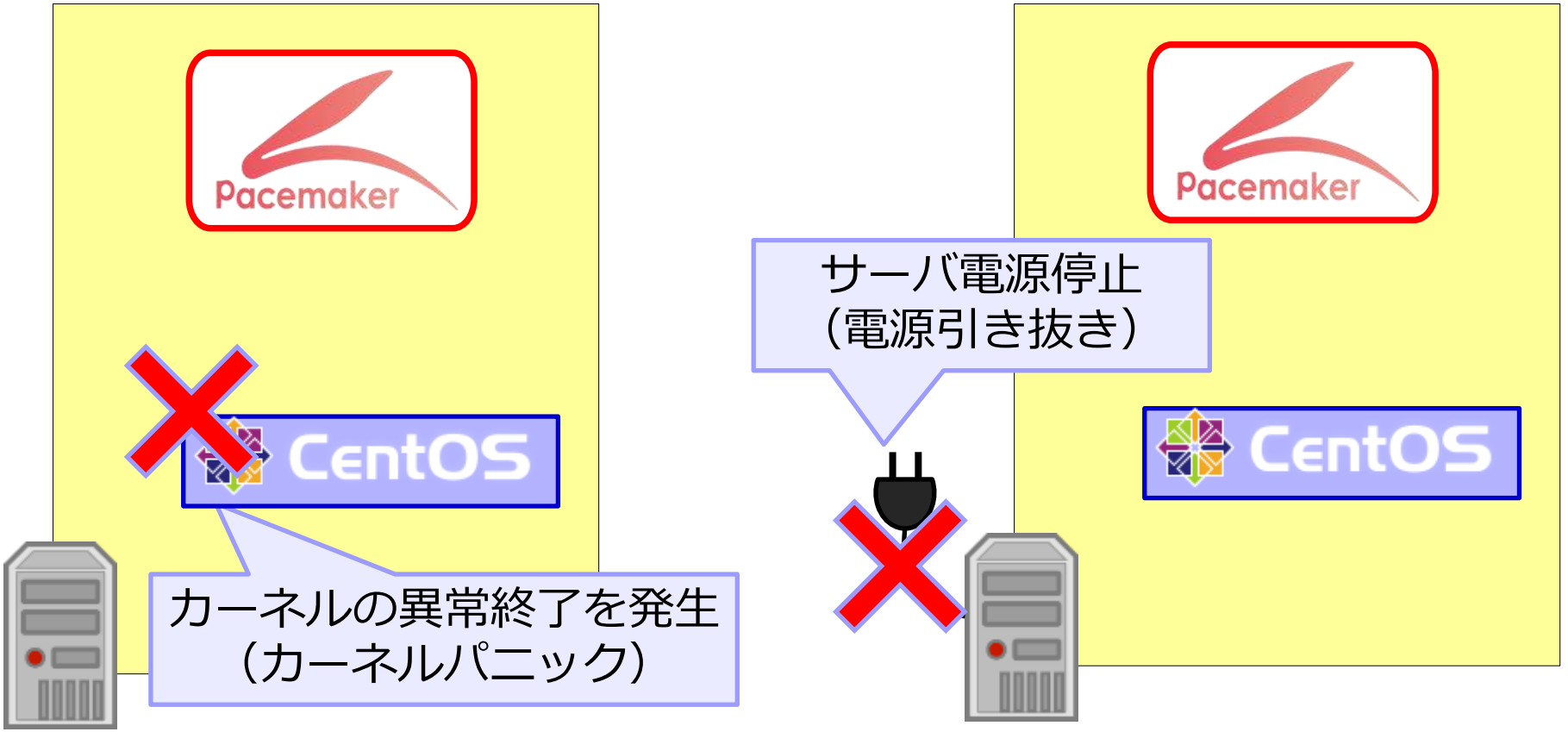
※crm_mon表示は一部省略

【3.ノード故障】

【3.ノード故障】①発生手順イメージ

凡例 [1] リソース／プロセス再起動
[2] 通常フェイルオーバ
[3] STONITH後フェイルオーバ

故障項目	故障内容	Pacemakerの動作	故障発生手順	復旧手順
ノード故障	カーネルパニック	[3]	# echo c > /proc/sysrq-trigger	[パターン2] Pacemaker再起動 (+フェイルバック)
	サーバ電源停止	[3]	# poweroff -nf	



【3.ノード故障】②発生手順

発生手順

ノード故障

➢ **ノード故障を起こすため、カーネルパニックを発生させる**

```
# echo c > /proc/sysrq-trigger
```

確認手順

ノード状態確認

➢ **サーバ1号機が接続不可となり、PostgreSQL リソースがサーバ2号機で起動していることを確認**

```
# crm_mon -fA
    :
Online: [ srv02 ]
OFFLINE: [ srv01 ]

 Resource Group: grpPostgreSQLDB
     prmExPostgreSQLDB          (ocf::heartbeat:sfex):          Started srv02
     prmFsPostgreSQLDB          (ocf::heartbeat:Filesystem):    Started srv02
     prmIpPostgreSQLDB          (ocf::heartbeat:IPaddr2):       Started srv02
     prmApPostgreSQLDB          (ocf::heartbeat:pgsql):         Started srv02
    :
Clone Set: clnPing [prmPing]
     Started: [ srv02 ]
 Clone Set: clnDiskd [prmDiskd]
     Started: [ srv02 ]

Node Attributes:
* Node srv02:
    + default_ping_set                 : 100
    + diskcheck_status                 : normal

Migration summary:
* Node srv02:
```

※crm_mon表示は一部省略

【3.ノード故障】③故障発生時の動作

【サーバ1号機】
① ノード故障発生

【サーバ2号機】
② Pacemakerがノード故障を検知
③ PacemakerがSTONITHを実行

障害検知

STONITH完了

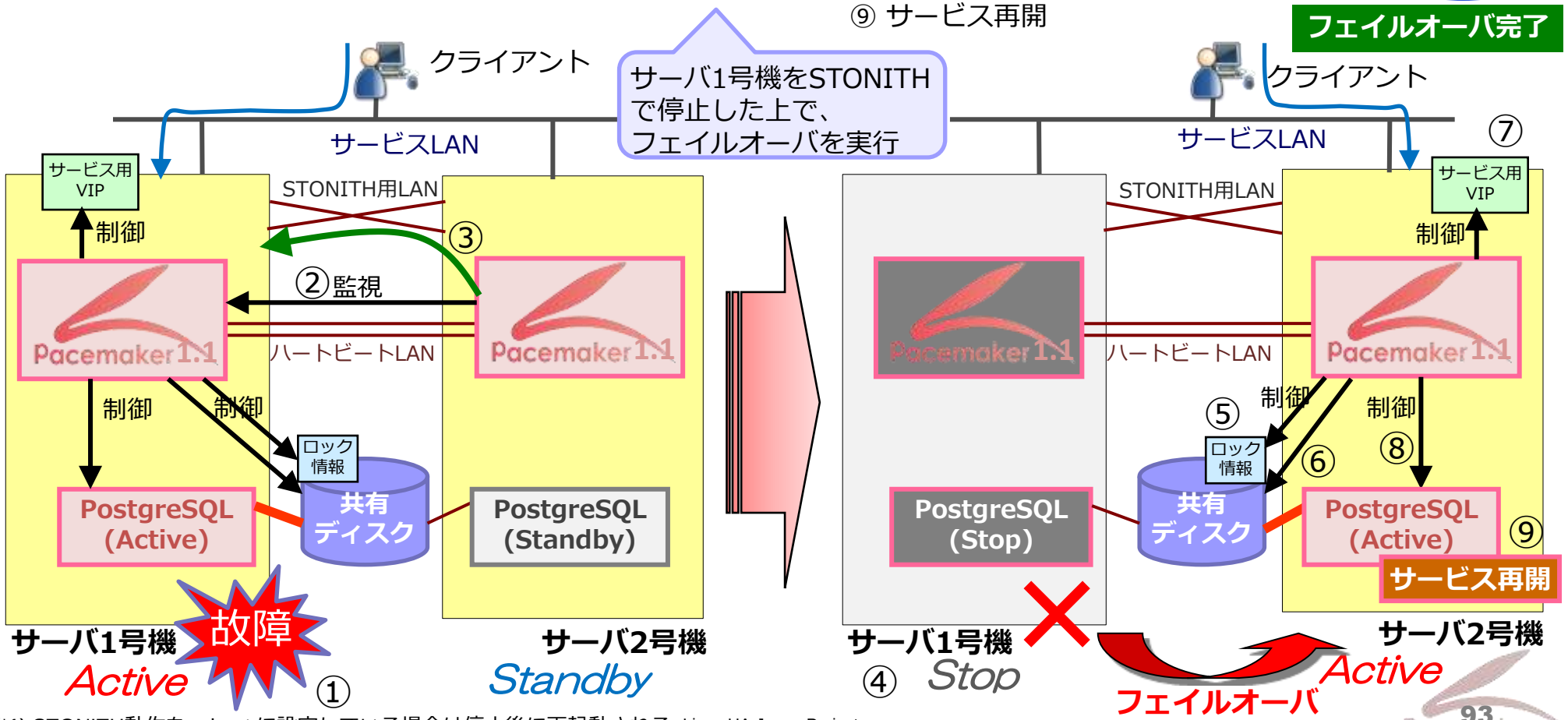
【サーバ1号機】
④ サーバ停止(*1)

【サーバ2号機】
⑤ Pacemakerが共有ディスクのロック取得
⑥ 〃 共有ディスクのマウント
⑦ 〃 サービス用VIPを起動
⑧ 〃 PostgreSQLを起動

PostgreSQL関連リソースの起動完了

⑨ サービス再開

フェイルオーバ完了



(*1) STONITH動作を reboot に設定している場合は停止後に再起動される。

【3.ノード故障】④pm_logconvのログ確認

故障後

srv01のノード故障を検知

【サーバ2号機】

```
May 25 18:11:58 srv02 warning: Node srv01 is lost
May 25 18:11:59 srv02   error: Start to fail-over.
May 25 18:11:59 srv02    info: Try to STONITH (reboot) srv01.
May 25 18:12:01 srv02    info: Try to execute STONITH device prmStonith1-1 on srv02 for
reboot srv01.
May 25 18:12:30 srv02 warning: Failed to execute STONITH device prmStonith1-1 for srv01.
May 25 18:12:30 srv02    info: Try to execute STONITH device prmStonith1-2 on srv02 for
reboot srv01.
May 25 18:12:32 srv02    info: Succeeded to execute STONITH device
prmStonith1-2 for srv01.
May 25 18:12:32 srv02    info: Succeeded to STONITH (reboot) srv01 by srv02.
May 25 18:12:32 srv02    info: Resource prmExPostgreSQLDB tries to start.
May 25 18:12:32 srv02    info: Resource prmExPostgreSQLDB started. (rc=0)
May 25 18:12:32 srv02    info: Resource prmFsPostgreSQLDB tries to start.
May 25 18:12:32 srv02    info: Resource prmFsPostgreSQLDB started. (rc=0)
May 25 18:12:32 srv02    info: Resource prmIpPostgreSQLDB tries to start.
May 25 18:12:32 srv02    info: Resource prmIpPostgreSQLDB started. (rc=0)
May 25 18:12:32 srv02    info: Resource prmApPostgreSQLDB tries to start.
May 25 18:12:32 srv02    info: Resource prmApPostgreSQLDB started. (rc=0)
May 25 18:12:32 srv02    info: Resource prmExPostgreSQLDB : Started on srv02
May 25 18:12:32 srv02    info: Resource prmApPostgreSQLDB : Started on srv02
May 25 18:12:32 srv02   info: fail-over succeeded.
```

【サーバ1号機】

（ログ出力なし）

① サーバ1号機のノード故障発生

② Pacemakerがノード故障を検知

フェイルオーバ開始　**障害検知**

③ PacemakerがSTONITHを実行

④ サーバ停止　**STONITH完了**

⑤ Pacemakerが共有ディスクのロック取得

⑥　〃　共有ディスクのマウント

⑦　〃　サービス用VIPを起動

⑧　〃　PostgreSQLを起動

2号機のPostgreSQL関連リソース起動

フェイルオーバ完了

【3.ノード故障】⑤復旧手順

手順1 ノード状態確認

手順2 ノード起動

故障復旧

手順3 Pacemaker起動

手順4 ノード状態確認

手順5 リソースグループの切り戻し(1/2)

手順6 リソース状態の確認

手順7 リソースグループの切り戻し(2/2)

手順8 リソース状態の確認

復旧手順パターン2

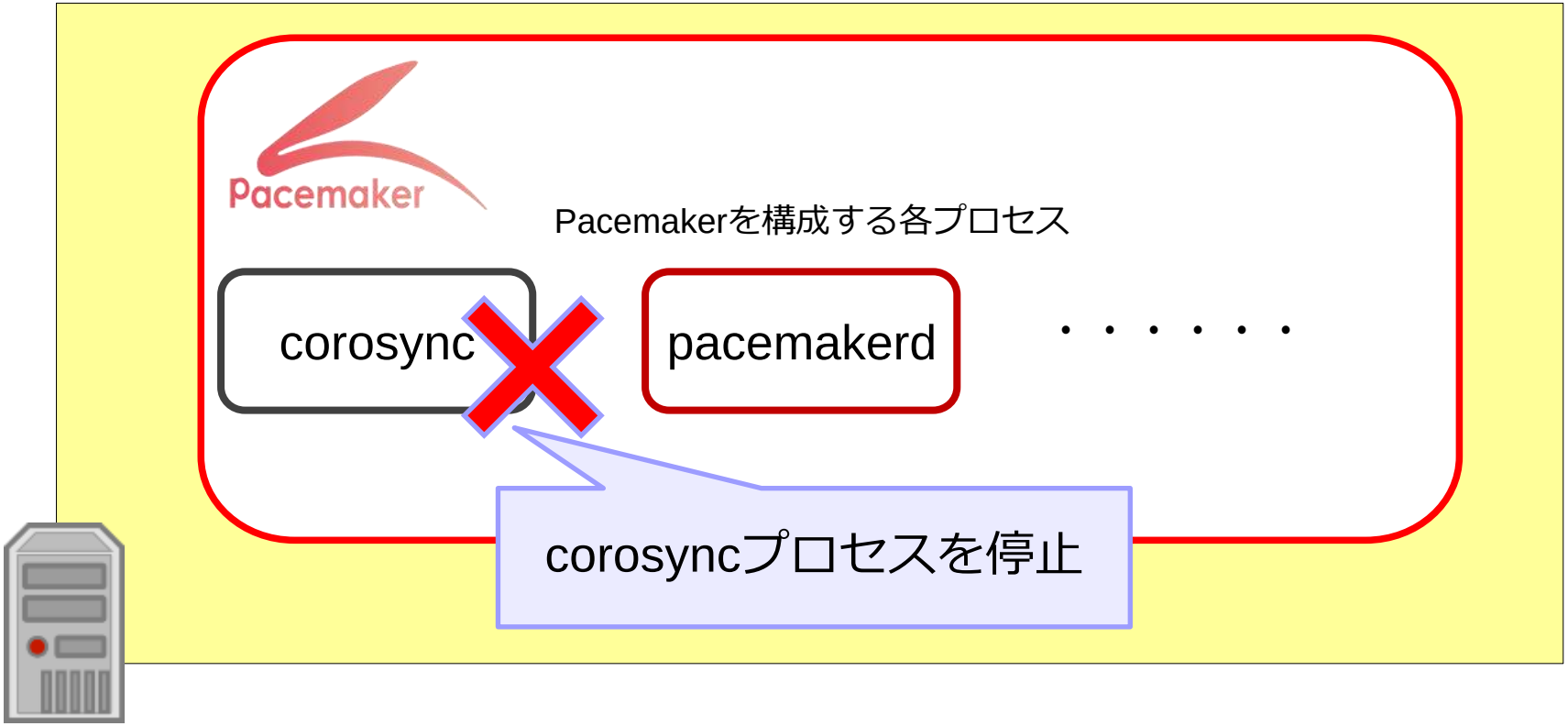
（P58～P59を参照）

【４.Pacemakerプロセス故障】

【4.Pacemakerプロセス故障】①発生手順イメージ

故障項目	故障内容	Pacemakerの動作	故障発生手順	復旧手順
Pacemaker プロセス故障	corosync プロセス故障	[3]	# pkill -9 corosync	[パターン2] Pacemaker再起動 (+フェイルバック)

凡例 [1] リソース／プロセス再起動
[2] 通常フェイルオーバ
[3] STONITH後フェイルオーバ



【4.Pacemakerプロセス故障】②発生手順

発生手順

プロセス故障

- **Corosync プロセスの起動を確認**

```
# ps -ef | grep corosync
15491 corosync
```

- **Corosync のプロセスKILLを実行**

```
# pkill -9 corosync
```

確認手順

ノード状態確認

- **サーバ1号機が接続不可となり、PostgreSQL リソースがサーバ2号機で起動していることを確認**

```
# crm_mon -fA
    :
Online: [ srv02 ]
OFFLINE: [ srv01 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB       (ocf::heartbeat:sfex):          Started srv02
    prmFsPostgreSQLDB       (ocf::heartbeat:Filesystem):    Started srv02
    prmIpPostgreSQLDB       (ocf::heartbeat:IPaddr2):       Started srv02
    prmApPostgreSQLDB       (ocf::heartbeat:pgsql):         Started srv02
    :
Node Attributes:
* Node srv02:
   + default_ping_set              : 100
   + diskcheck_status              : normal
    :
Migration summary:
* Node srv02:
```

※crm_mon表示は一部省略

【4.Pacemakerプロセス故障】③故障発生時の動作

【サーバ1号機】
① Corosyncプロセス故障発生
② watchdog機能が障害を検知
③ watchdogによりサーバ再起動(*1)

【サーバ2号機】
②′ Pacemakerが対向ノード不明を検知
③′ PacemakerがSTONITHを実行(*1)

障害検知

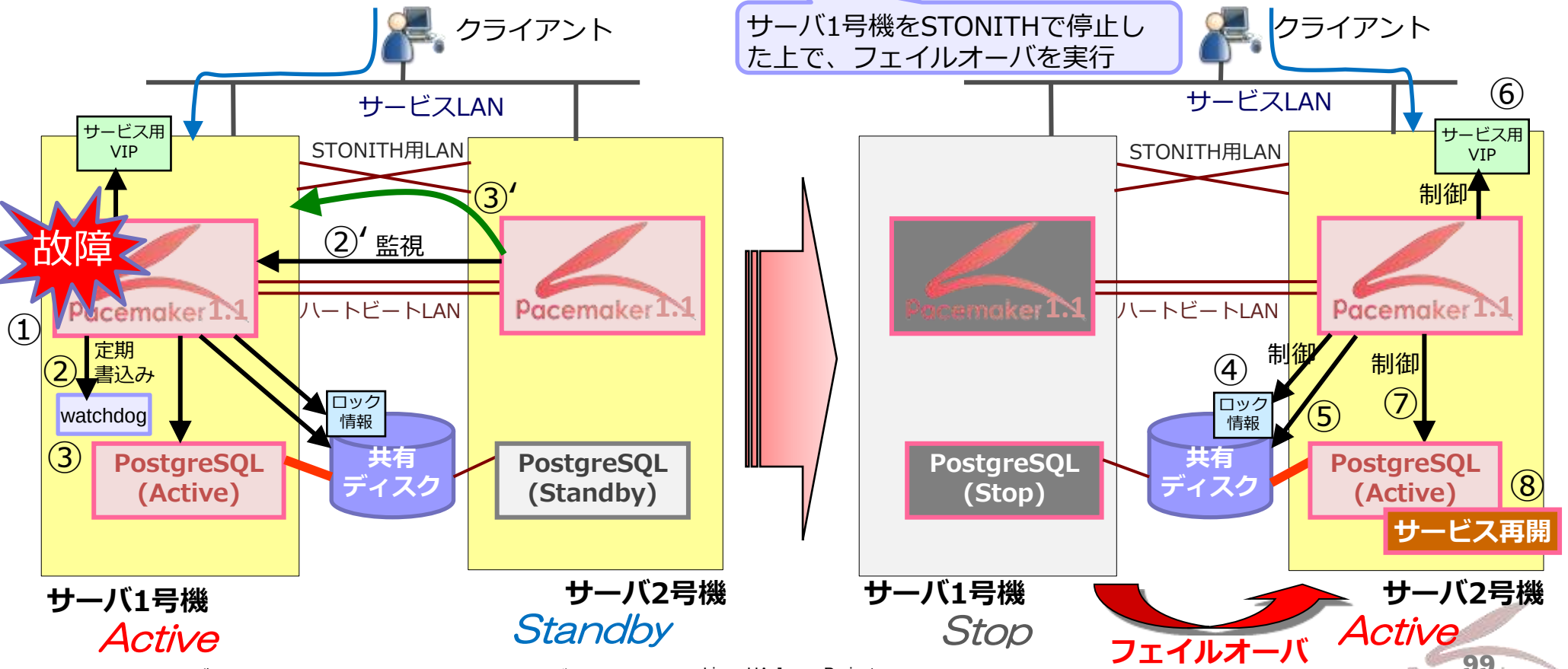
STONITH完了

【サーバ2号機】
④ Pacemakerが共有ディスクのロック取得
⑤ 〃 共有ディスクのマウント
⑥ 〃 サービス用VIPを起動
⑦ 〃 PostgreSQLを起動

PostgreSQL関連リソースの起動完了

⑧ サービス再開

フェイルオーバ完了



(*1) 設定タイミングでwatchdog、STONITHによる再起動が実行される。

【４.Pacemakerプロセス故障】④pm_logconvのログ確認

故障後

srv01のノード故障を検知

【サーバ2号機】

```
May 28 10:23:13 srv02 warning: Node srv01 is lost
May 28 10:23:14 srv02   error: Start to fail-over.
May 28 10:23:14 srv02    info: Try to STONITH (reboot) srv01.
May 28 10:23:15 srv02    info: Try to execute STONITH device prmStonith1-1 on srv02 for
reboot srv01.
May 28 10:23:44 srv02 warning: Failed to execute STONITH device prmStonith1-1 for srv01.
May 28 10:23:44 srv02    info: Try to execute STONITH device prmStonith1-2 on srv02 for
reboot srv01.
May 28 10:23:45 srv02    info: Succeeded to execute STONITH device
prmStonith1-2 for srv01.
May 28 10:23:45 srv02    info: Succeeded to STONITH (reboot) srv01 by srv02.
May 28 10:23:45 srv02    info: Resource prmExPostgreSQLDB tries to start.
May 28 10:24:56 srv02    info: Resource prmExPostgreSQLDB started. (rc=0)
May 28 10:24:56 srv02    info: Resource prmFsPostgreSQLDB tries to start.
May 28 10:24:57 srv02    info: Resource prmFsPostgreSQLDB started. (rc=0)
May 28 10:24:57 srv02    info: Resource prmIpPostgreSQLDB tries to start.
May 28 10:24:57 srv02    info: Resource prmIpPostgreSQLDB started. (rc=0)
May 28 10:24:57 srv02    info: Resource prmApPostgreSQLDB tries to start.
May 28 10:24:58 srv02    info: Resource prmApPostgreSQLDB started. (rc=0)
May 28 10:24:59 srv02    info: Resource prmExPostgreSQLDB : Started on srv02
May 28 10:24:59 srv02    info: Resource prmApPostgreSQLDB : Started on srv02
May 28 10:24:59 srv02    info: fail-over succeeded.
```

【サーバ1号機】

（ログ出力なし）

① サーバ1号機のcorosyncプロセス故障発生

② Pacemakerが対向ノード不明を検知

障害検知

フェイルオーバ開始

③ PacemakerがSTONITHを実行

STONITH完了

④ Pacemakerが共有ディスクのロック取得

⑤ 〃 共有ディスクのマウント

⑥ 〃 サービス用VIPを起動

⑦ 〃 PostgreSQLを起動

2号機のPostgreSQL関連リソース起動

フェイルオーバ完了

【4.Pacemakerプロセス故障】⑤復旧手順

手順1 ノード状態確認

手順2 ノード起動

故障復旧

手順3 Pacemaker起動

手順4 ノード状態確認

手順5 リソースグループの切り戻し(1/2)

手順6 リソース状態の確認

手順7 リソースグループの切り戻し(2/2)

手順8 リソース状態の確認

復旧手順パターン2

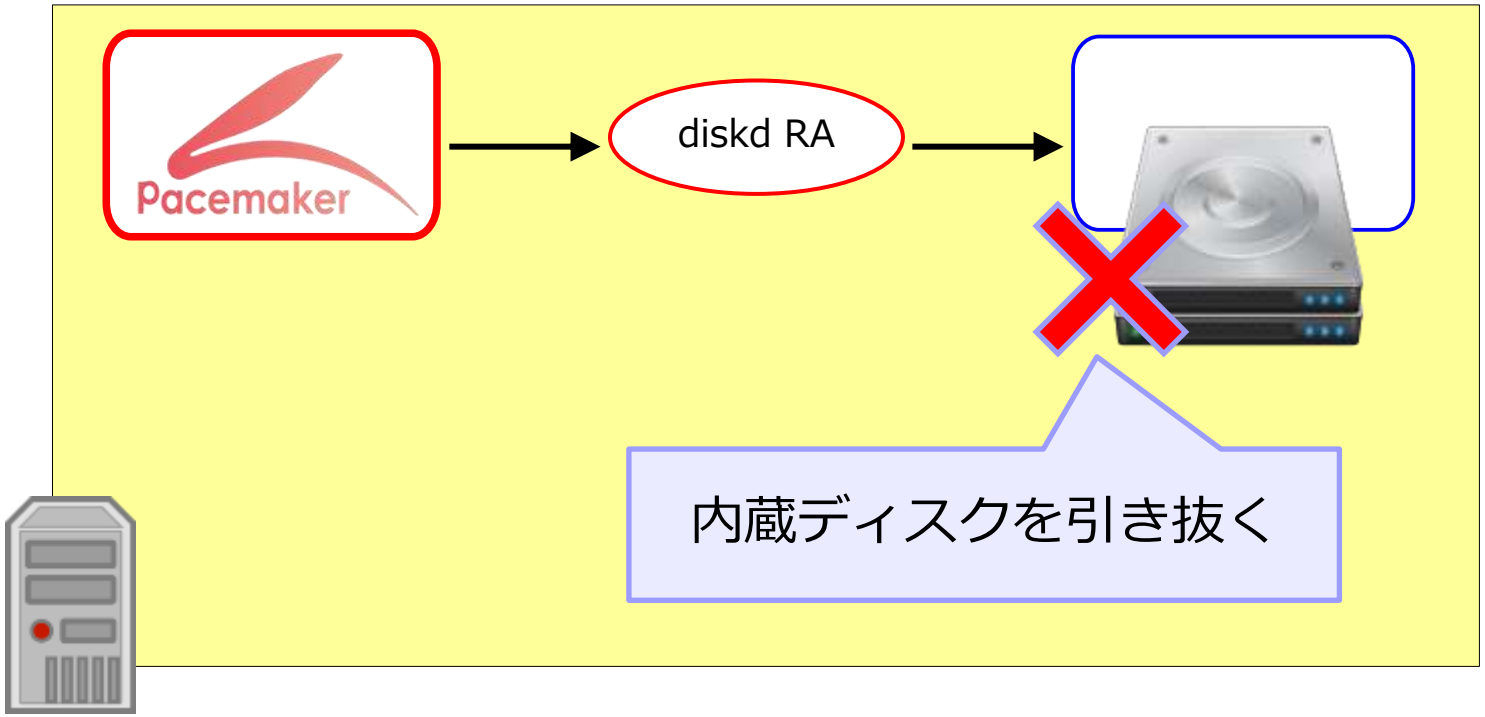
(P58～P59を参照)

【５.ディスク故障】

【5.ディスク故障】①発生手順イメージ

凡例 [1] リソース／プロセス再起動
[2] 通常フェイルオーバ
[3] STONITH後フェイルオーバ

故障項目	故障内容	Pacemakerの動作	発生手順	復旧手順
ディスク故障	内蔵ディスク故障	[2] or [3]	内蔵ディスク引き抜き	[パターン３] 強制電源断 ＋Pacemaker再起動 (＋フェイルバック)
	共有ディスクケーブル故障	[2]	ディスクケーブル引き抜き	



【5.ディスク故障】②発生手順

発生手順

内蔵ディスク故障

➢ **内蔵ディスクを引き抜く**

ディスクが本当に壊れてしまう場合もあり得るため、検証の順番として一番最後に実施することをお奨めします

確認手順

ノード状態確認

➢ **サーバ1号機が接続不可となり、PostgreSQLリソースがサーバ2号機で起動していることを確認**

```
# crm_mon -fA
    :
Online: [ srv02 ]
OFFLINE: [ srv01 ]

 Resource Group: grpPostgreSQLDB
     prmExPostgreSQLDB          (ocf::heartbeat:sfex):          Started srv02
     prmFsPostgreSQLDB          (ocf::heartbeat:Filesystem):    Started srv02
     prmIpPostgreSQLDB          (ocf::heartbeat:IPaddr2):       Started srv02
     prmApPostgreSQLDB          (ocf::heartbeat:pgsql):         Started srv02
     :
 Clone Set: clnPing [prmPing]
     Started: [ srv02 ]
 Clone Set: clnDiskd1 [prmDiskd1]
     Started: [ srv02 ]

Node Attributes:
* Node srv02:
    + default_ping_set                : 100
    + diskcheck_status                : normal
    :
Migration summary:
* Node srv02:
```

※crm_mon表示は一部省略

【5.ディスク故障】③故障発生時の動作

【サーバ1号機】
① サーバ1号機の内蔵ディスク故障発生
② Pacemakerがディスク故障エラーを検知
③ PacemakerがPostgreSQLリソース停止失敗

【サーバ2号機】
②' Pacemakerがノード故障を検知
④ PacemakerがSTONITHを実行

障害検知

PostgreSQL関連リソースの停止失敗

STONITH成功

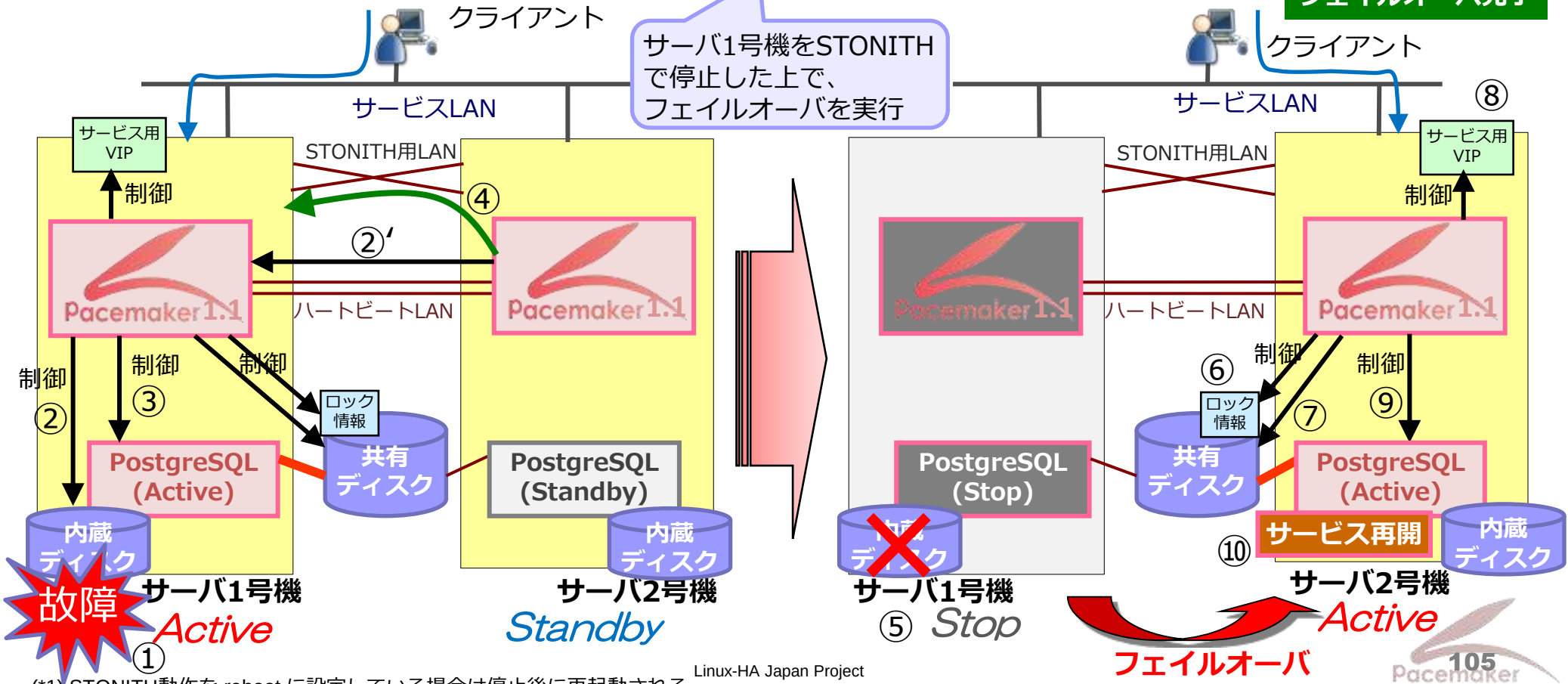
【サーバ1号機】
⑤ サーバ停止(*1)

【サーバ2号機】
⑥ Pacemakerが共有ディスクのロック取得
⑦ 〃 共有ディスクのマウント
⑧ 〃 サービス用VIPを起動
⑨ 〃 PostgreSQLを起動
⑩ サービス再開

PostgreSQL関連リソースの起動完了

フェイルオーバ完了

サーバ1号機をSTONITHで停止した上で、フェイルオーバを実行



(*1) STONITH動作を reboot に設定している場合は停止後に再起動される。

【5.ディスク故障】④pm_logconvのログ確認

故障後

srv01のディスク故障を検知

ディスク故障の検知の前に、ping監視エラー等の他の障害検知が発生する場合もあります

① サーバ1号機の内蔵ディスク故障発生

【サーバ1号機】

```
May 11 10:33:18 srv01   error: Disk connection to /dev/internal_a is ERROR.
(attr_name=diskcheck_status_internal)
May 11 10:33:18 srv01    info: Attribute "diskcheck_status_internal" is updated to "ERROR" at "srv01".
May 11 10:33:20 srv01    info: Resource prmApPostgreSQLDB tries to stop.
May 11 10:33:20 srv01   error: Resource prmApPostgreSQLDB failed to stop. (status=4)
May 11 10:33:20 srv01   error: Resource prmExPostgreSQLDB failed to monitor. (status=4)
May 11 10:33:20 srv01   error: Resource prmFsPostgreSQLDB failed to monitor. (status=4)
May 11 10:33:21 srv01   error: Resource prmIpPostgreSQLDB failed to monitor. (status=4)
```

② Pacemakerがディスク故障エラーを検知

障害検知

③ PacemakerがPostgreSQLリソース停止に失敗(その他の監視もエラー)

【サーバ2号機】

```
May 11 10:33:18 srv02    info: Attribute "diskcheck_status_internal" is updated to "ERROR" at "srv01".
May 11 10:33:20 srv02   error: Start to fail-over.
May 11 10:33:20 srv02    info: Try to STONITH (reboot) srv01.
May 11 10:33:21 srv02    info: Try to execute STONITH device prmStonith1-1 on srv02 for reboot srv01.
May 11 10:33:25 srv02 warning: Failed to execute STONITH device prmStonith1-1 for srv01.
May 11 10:33:25 srv02    info: Try to execute STONITH device prmStonith1-2 on srv02 for reboot srv01.
May 11 10:33:28 srv02 warning: Node srv01 is lost
May 11 10:33:28 srv02    info: Succeeded to execute STONITH device prmStonith1-2 for srv01.
May 11 10:33:28 srv02    info: Succeeded to STONITH (reboot) srv01 by srv02.
May 11 10:33:29 srv02    info: Resource prmExPostgreSQLDB tries to start.
May 11 10:34:40 srv02    info: Resource prmExPostgreSQLDB started. (rc=0)
May 11 10:34:40 srv02    info: Resource prmFsPostgreSQLDB tries to start.
May 11 10:34:40 srv02    info: Resource prmFsPostgreSQLDB started. (rc=0)
May 11 10:34:40 srv02    info: Resource prmIpPostgreSQLDB tries to start.
May 11 10:34:40 srv02    info: Resource prmIpPostgreSQLDB started. (rc=0)
May 11 10:34:40 srv02    info: Resource prmApPostgreSQLDB tries to start.
May 11 10:34:42 srv02    info: Resource prmApPostgreSQLDB started. (rc=0)
May 11 10:34:42 srv02    info: Resource prmExPostgreSQLDB : Started on srv02
May 11 10:34:42 srv02    info: Resource prmApPostgreSQLDB : Started on srv02
May 11 10:34:42 srv02   info: fail-over succeeded.
```

フェイルオーバ開始

④ PacemakerがSTONITHを実行

STONITH完了

⑤ サーバ停止

2号機のPostgreSQL関連リソース起動

⑥ Pacemakerが共有ディスクのロック取得

⑦ 〃 共有ディスクのマウント

⑧ 〃 サービス用VIPを起動

⑨ 〃 PostgreSQLを起動

フェイルオーバ完了

【5.ディスク故障】⑤復旧手順(1/3)

復旧手順パターン3

手順1 ノード状態確認 ➢ **サーバ2号機で、リソース状態が“Started サーバ2号機”であることを確認**

```
# crm_mon -fA
   :
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB        (ocf::heartbeat:sfex):        Started srv02
    prmFsPostgreSQLDB        (ocf::heartbeat:Filesystem):  Started srv02
    prmIpPostgreSQLDB        (ocf::heartbeat:IPaddr2):     Started srv02
    prmApPostgreSQLDB        (ocf::heartbeat:pgsql):       Started srv02
    :
```

フェイルオーバにより
サーバ2号機で起動

手順2 強制電源断 ➢ **サーバ1号機の電源を強制的に停止**

手順3 ノード状態確認 ➢ **サーバ2号機で、サーバ1号機の状態が “OFFLINE” であることを確認**

```
# crm_mon -fA
   :
Online: [ srv02 ]
OFFLINE: [ srv01 ]
```

サーバ1号機の状態が**“UNCLEAN(offline)”**となっている場合は、手動でSTONITHを終了させたことをクラスタに通知するために、stonith_adminコマンドによる保守者介在処理を行います。
※サーバ2号機の pm_logconv.out に以下のログが出力されています。
「May 11 10:33:28 srv02　error: Failed to STONITH (reboot) srv01 by srv02.」
サーバ2号機で stonith_adminコマンドを以下の通り実施後、再度ノード状態を確認してください。

```
# stonith_admin -C srv01
```

※crm_mon表示は一部省略

【5.ディスク故障】⑤復旧手順(2/3)

復旧手順パターン3

故障復旧

手順4 ノード起動 ➢ **サーバ1号機の電源を起動**

手順5 Pacemaker起動 ➢ **サーバ1号機の Pacemakerを起動**

```
# systemctl start pacemaker
```

手順6 ノード状態確認 ➢ **ノード状態を確認し、１号機の状態が “Online” であることを確認**

```
# crm_mon -fA
  :
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB       (ocf::heartbeat:sfex):          Started srv02
    prmFsPostgreSQLDB       (ocf::heartbeat:Filesystem):    Started srv02
    prmIpPostgreSQLDB       (ocf::heartbeat:IPaddr2):       Started srv02
    prmApPostgreSQLDB       (ocf::heartbeat:pgsql):         Started srv02
    :
```

※crm_mon表示は一部省略

【5.ディスク故障】⑤復旧手順(3/3)

復旧手順パターン3

手順7 リソースグループの切り戻し(1/2)

➢ **リソースグループをサーバ1号機に切り戻す**

✓ crm_resourceコマンドは、リソースを動的に操作(表示/設定/削除)する
✓ オプション： -M(リソースを指定ノードで起動するように切り替える制約追加) -r [リソースIDを指定] –N [ホスト名] –f(リソースを強制的に再配置) –Q(値のみ表示)

```
# crm_resource -M -r grpPostgreSQLDB -N srv01 -f -Q
```

手順8 リソース状態の確認

➢ **リソース状態が “Started サーバ1号機” となっていることを確認**

✓ リソースの実行不可制約がサーバ2号機に設定されていること

手順7でサーバ1号機にリソースを切り戻すため、サーバ2号機でリソース起動を行わない制約が設定されます。
切り戻し完了後に、その制約を解除しておく必要があります。

```
# crm_mon -fA -L
  :
Online: [ srv01 srv02 ]

 Resource Group: grpPostgreSQLDB
    prmExPostgreSQLDB        (ocf::heartbeat:sfex):        Started srv01
    prmFsPostgreSQLDB        (ocf::heartbeat:Filesystem):  Started srv01
    prmIpPostgreSQLDB        (ocf::heartbeat:IPaddr2):     Started srv01
    prmApPostgreSQLDB        (ocf::heartbeat:pgsql):       Started srv01
  :
Negative location constraints:
  cli-ban-grpPostgreSQLDB-on-srv02 prevents grpPostgreSQLDB from running on srv02
```

-L(実行不可制約表示)を付ける

手順9 リソースグループの切り戻し(2/2)

➢ **サーバ2号機の実行不可制約を解除**

✓ オプション： -U(切り替えによる制約を解除) -r [リソースIDを指定]

よく解除忘れが起こるので注意

```
# crm_resource -U -r grpPostgreSQLDB
```

手順10 リソース状態の確認

➢ **実行不可制約解除を確認**

```
# crm_mon -fA -L
  :
Negative location constraints:
```

リソース切り戻し時の実行不可制約の解除漏れを防止

※crm_mon表示は一部省略