HAクラスタサポートの日々
～Pacemaker導入･運用の勘所～

2012年3月17日　OSC2012 Tokyo

Linux-HA Japan
赤松　洋

HighAvailability Pacemaker

自己紹介

赤松　洋

です！

本日のお話

① 「Pacemaker」を導入するということは？
～ トラブル事例１：RA起因 ～

② ある日の保守
～ トラブル事例２：フェイルオーバー ～

③ こんな事象が発生！
～ トラブル事例３：タイムアウト値 ～
～ トラブル事例４：共有ディスクの話 ～

④ こんな要望にも答えます！
～ 解析・要望実現方法 ～

ちなみに…

インストール・環境構築の話はありません

Corosync の話もありません

下回りはHeartbeat に特化してます。

仮想化の話もありません（擬人化もないです）

DRBD の話もありません

ちなみに…

実際にお使いになっているお客様や運用者様にとって
有用な情報を提供していこうと考えています。

環境は Pacemaker＋Heartbeat (1.0.11)に特化してます。

かなり個人的な見解に偏っている所もあるので、必ずしも
コミュニティの考えに沿っていない発言があるかもしれません。

①

「Pacemaker」を導入するということは？

～ トラブル事例1：RA起因 ～

なぜ、Pacemaker を導入するんですか？

アプリケーションをクラスタで管理
出来るからです！

どうやってアプリケーションを管理する
んですか？

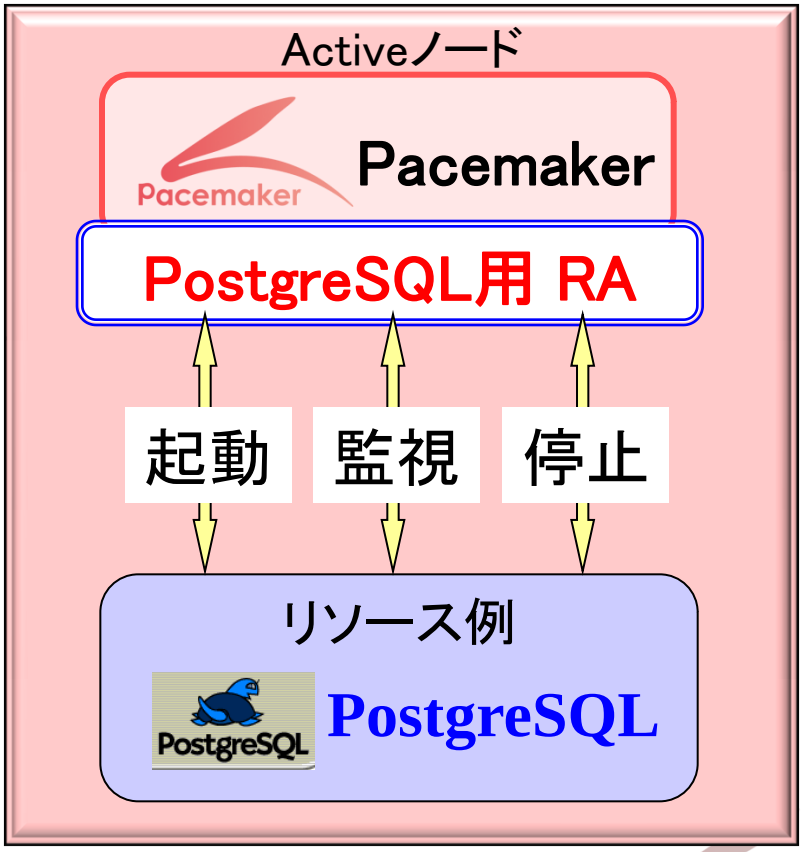
RA と呼ばれる、処理機構で実現します！

RAってなんですか？なにができるんですか？

「リソースエージェント（RA）」とは？

リソースとPacemakerを仲介するプログラム
主にシェルスクリプトで記述されています

Pacemakerは、
リソースエージェントに対し
リソースの
　起動(start)
　停止(stop)
　監視(monitor)
を指示します。



業務にかかわる所で、RAの何を知っていれば良いのですか？

起動・監視・停止時に、具体的にどんな動作をするのか、望むような処理を行っているのか、押さえておきましょう。

たとえば PostgreSQL 用RA であれば…

● 各 OCF パラメータ:

```
pgctl="/usr/pgsql-9.0/bin/pg_ctl"      start_opt="-p 5432 -h 127.0.0.1"
psql="/usr/pgsql-9.0/bin/psql"         pgdata="/dbfp/pgdata/data"
pgdba="postgres"                       pgport="5432"
pgdb="template1"
```

● 起動:

```
su postgres -c 'cd /dbfp/pgdata/data; /usr/pgsql-9.0/bin/pg_ctl  -D /dbfp/pgdata/data ¥
  -l /dev/null -o '¥'' -p 5432 -h 127.0.0.1'¥'' start'
```

● 監視:

```
su postgres -c 'cd /dbfp/pgdata/data; kill -s 0 `head -n 1 /dbfp/pgdata/data/postmaster.pid` ¥
  >/dev/null 2>&1'

su postgres -c 'cd /dbfp/pgdata/data; /usr/pgsql-9.0/bin/psql -p 5432 -U postgres template1 ¥
  -c '¥''select now();'¥''' 
```

● 停止:

```
su postgres -c 'cd /dbfp/pgdata/data; /usr/pgsql-9.0/bin/pg_ctl -D /dbfp/pgdata/data stop ¥
  -m fast'
su postgres -c 'cd /dbfp/pgdata/data; /usr/pgsql-9.0/bin/pg_ctl -D /dbfp/pgdata/data stop ¥
  -m immidiate'
```


ここからわかる事は…

- 127.0.0.1(::1) からのアクセスは認証なしでいけるよう設定する必要があります。
  具体的には /dbfp/pgdata/data/pg_hba.conf にて下記のとおり設定が必須です。

```
host    all     all     127.0.0.1/32    trust
host    all     all     ::1/128         trust
```

- pg_ctl コマンドはタイムアウトを設定できるらしいですが RA にその仕様がない(デフォルト60sが適用される)。
  よって各タイムアウト値は最低でも 60秒以上必要になります。

- “select now()”を監視処理にて定期的に実行するため PostgreSQL 側のアクセスログは業務上のアクセス以上に膨れる可能性があります。

- 事例1：PostgreSQL が起動できない
  ⇒ 認証設定を施していた

- 事例2：IPv4でApache起動出来たのにIPv6で起動できない
  ⇒ RA にて IPv4固定の bind が埋め込まれている

- 事例3：Tomcat/Jboss 等で日本語のログが文字化け（英語出力）
  ⇒ RA の親玉的なファイルで LC_ALL=C としている

- 事例4：LVM を使うとログが鬼のように出る
  ⇒ RA内で vgdisplay, vgck を行ってて、これがガシガシ出力してる

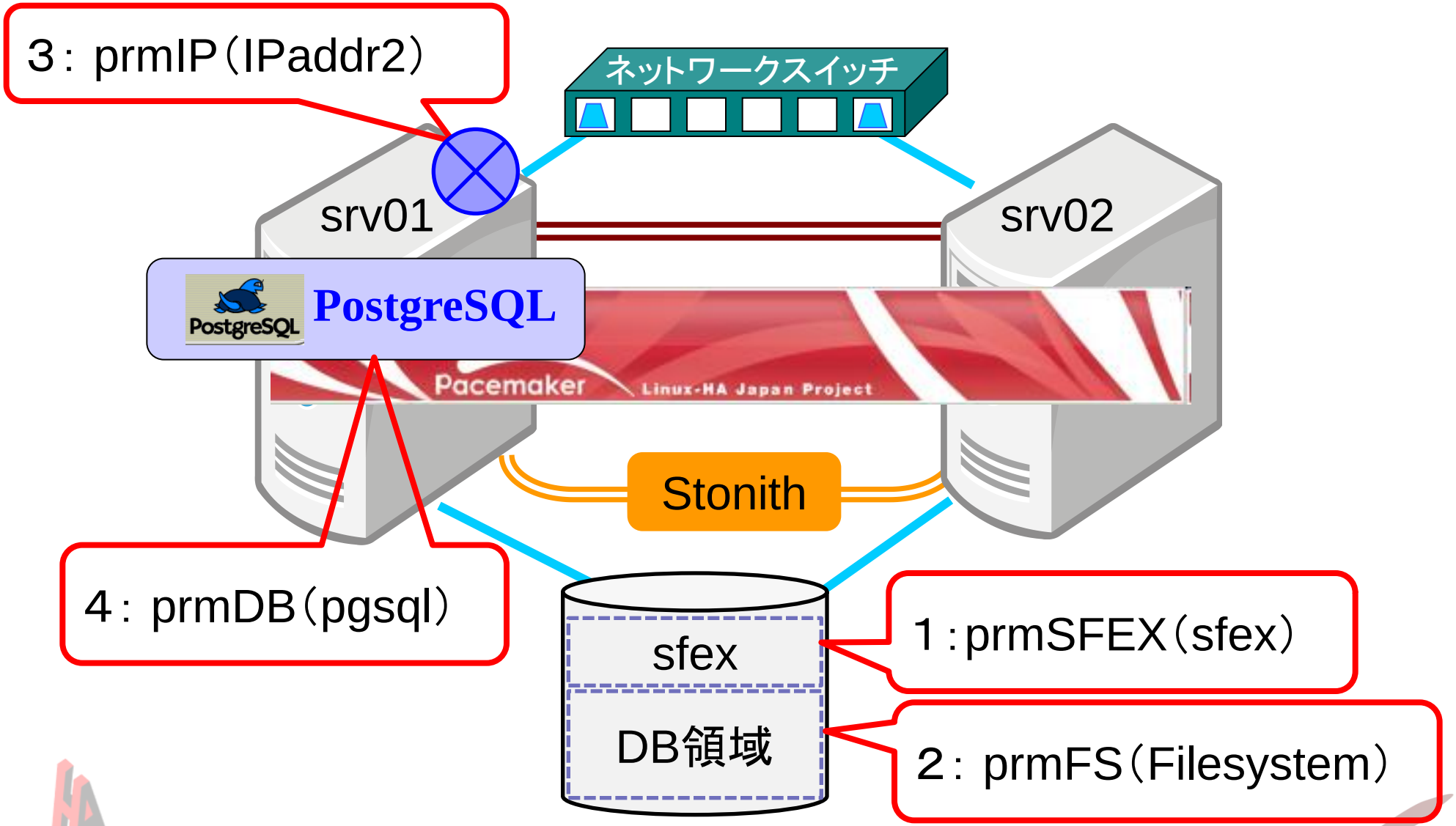
RA を読むことで、トラブルを事前に回避（？）できます

②

ある日の保守

～ トラブル事例2：フェイルオーバー ～

某社のシステム構成


3：prmIP（IPaddr2）
ネットワークスイッチ
srv01
srv02
PostgreSQL
PostgreSQL
Pacemaker
Linux-HA Japan Project
Stonith
4：prmDB（pgsql）
sfex
DB領域
1：prmSFEX（sfex）
2：prmFS（Filesystem）

大変です！

どうされましたか？

リソースがフェイルオーバーしてます！

Pacemaker が、何か異常を検知したんですね。

何をしたらよいでしょうか！？

1

srv02 でリソースは正常に
稼働していますか？

DB であれば正常にデータが取れるか等の確認になります。
それぞれのプロダクトや保守担当者に行ってもらうのが良いですが
この時点では最低限でもOKでしょう。

具体的には下記コマンドで 確認 します。

srv02

下記の場合 ② や ③ に行きます。
crm_mon -n -1 | grep -w Node
Node srv01 (xxxx): OFFLINE
Node srv02 (xxxx): online

下記の場合 ④ に行きます。
crm_mon -n -1 | grep -w Node
Node srv01 (xxxx): online
Node srv02 (xxxx): online

2 （あたりまえですが）電源の確認してください

srv01 のサーバそのものに電源が入っているか確認しましょう。
OS もきちんと動作してるか確認します。
確認後は Pacemaker を起動します。

3 Pacemaker が起動してるか確認してください

srv01 の Pacemaker が起動していない場合 srv02 から Stonith で落とされた可能性が高いです。原因は以下の2点に絞られます。
どちらであっても srv01 のログファイル等を取得後、起動します。

srv01

リソースの監視異常と停止異常が発生していた

（経験ないですが）Pacemaker プロセスが暴走等

※ インターコネクトLAN抜線でも STONITH が発動しますが、その場合は srv02 に対して行われるため、現状と異なるのでこの事象はありえません。

4 リソースなどの確認

まずは下記コマンドを実行し、srv01 を standby 化しておきます。

srv01

 # crm node standby srv01
 # crm_mon -n -1 | grep -w Node
Node srv01 (xxxx): standby
Node srv02 (xxxx): online

その後 *crm_mon -fA -1* コマンドで状況を確認します。

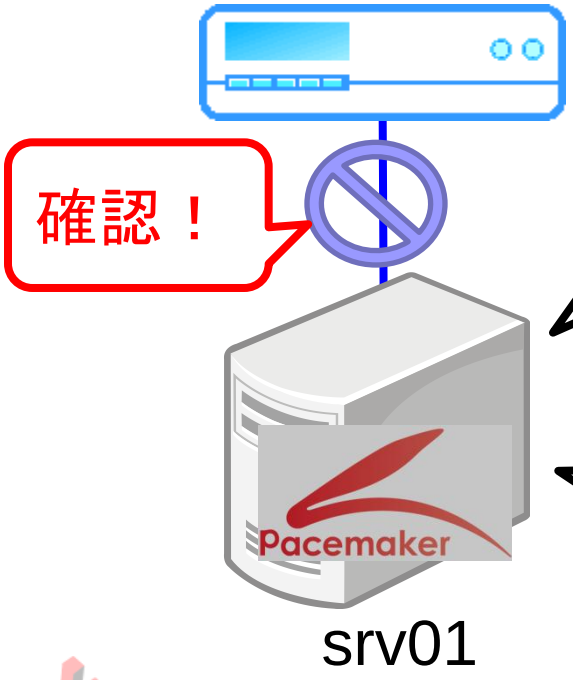
出力結果から、主に3パターンの運用に分かれます。

- NW経路監視に異常が発生
- Disk監視に異常が発生
- リソース異常が発生

４－１　NW経路監視 に異常が発生

srv01 から経路監視先へ ping が通らないことを確認。
ケーブル抜線か NIC 故障、ifconfig down による NICダウン、それ以外に iptables コマンドによる制約付与が疑われます。

ネットワーク関係の確認を行いましょう！



 # crm_mon -f A-1
…
* Node srv01:
　+ default_ping_set : 0 : Connectivity is lost

ネットワーク関係の修復、確認等…
 # crm node online srv01

４－２　Disk監視 に異常が発生

srv01 から共有Diskへの監視異常と出ています。リソース異常も併発しています。
FCケーブル断線等の事象が疑われます。

サーバと共有ディスク間の導通を確認しましょう。

 # crm_mon -f A-1

…

Failed actions:

prmSFEX_monitor_10000 ¥
(node=srv01, call=XXX, rc=7, status=complete): ¥
not running
prmDB_monitor_10000 ¥
(node=srv01, call=XXX, rc=-2, status=Timed Out): ¥
unknown exec error

…

+ diskcheck_status : ERROR

４－２　Disk監視 に異常が発生

サーバと共有ディスク間の導通が確認できましたら、リソース異常における対処を行います。
尚、Disk監視の正常性は自動で検知します。

 # crm resource cleanup prmDB srv01
 # crm resource cleanup prmSFEX srv01
 # crm node online srv01
 # crm_mon –fA -1

４－３ リソース異常が発生

リソース異常が発生したためにフェイルオーバーが発生したことを意味してます。
fail-count を落とし、クラスタメンバに復帰させた後、ha-log の解析処理に入ります。

srv01

 # crm mon -f A-1

…

* Node srv01:
prmDB: migration-threshold=1 fail-count=1

Failed actions:
prmDB_monitor_10000 ¥
(node=srv01, call=XXX, rc=-2, status=Timed Out): ¥
unknown exec error

 # crm resource cleanup prmDB srv01
 # crm node online srv01

大変です！

どうされましたか？

リソースが両系共にいません！

両系とも電源・Pacemaker の稼働は
確認済ですね（①・②・③は確認済）？

引き続き、何をしたらよいでしょうか！？

4 リソースなどの確認

まずは下記コマンドを実行し（最終的に srv01でリソースを起動させるため）
srv02 を standby 化しておきます。

srv02

 # crm node standby srv02
 # crm_mon -n -1 | grep -w Node
Node srv01 (xxxx): online
Node srv02 (xxxx): standby

その後 *crm_mon -fA -1* コマンドで状況を確認します。

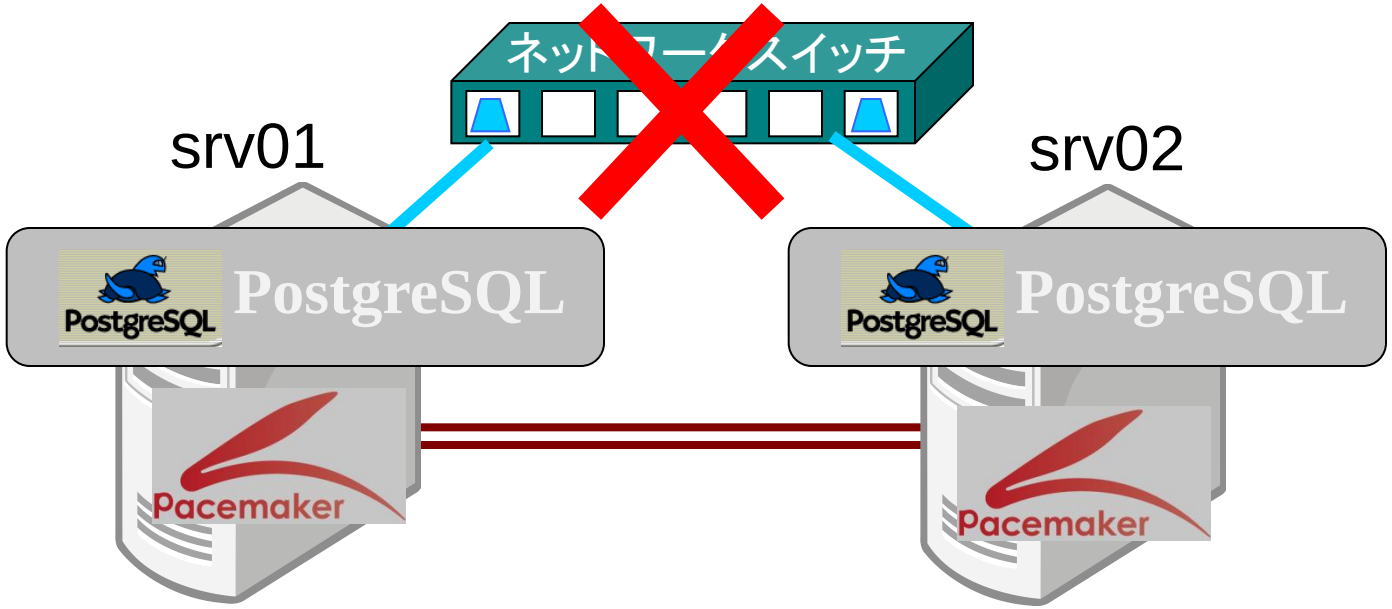
出力結果から、主に4パターンの運用に分かれます。

NW経路監視が両系ともに異常	Disk監視が両系ともに異常
両系でリソース異常	リソース移動制約が効いてる

４－１　NW経路監視が両系ともに異常

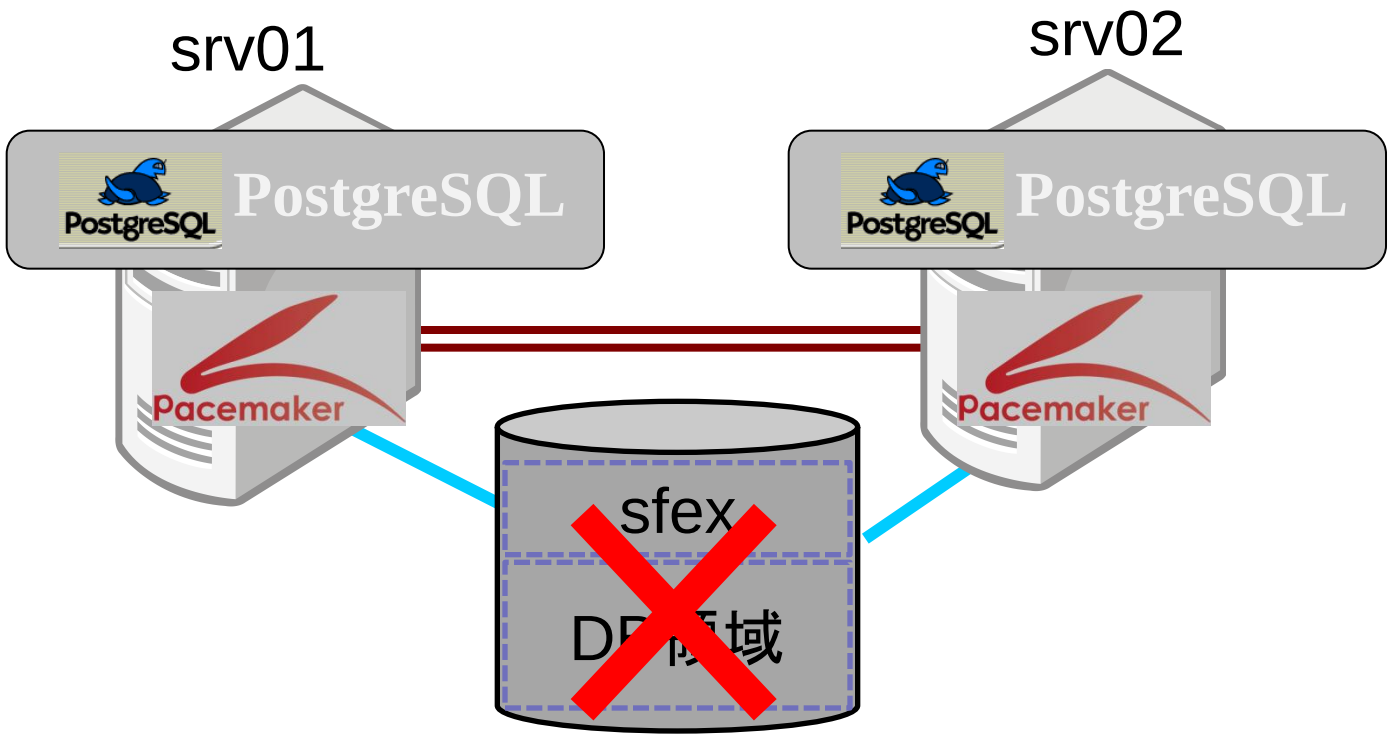
経路監視先そのものが落ちている可能性があります。
ping が通らないことを確認、スイッチ（ルータ）等のNW関係の機材確認等を行ってください。

尚、Pacemaker は両系共にとめる必要はありません。
経路監視が復帰すれば、自動的に srv01 でリソースが起動します。



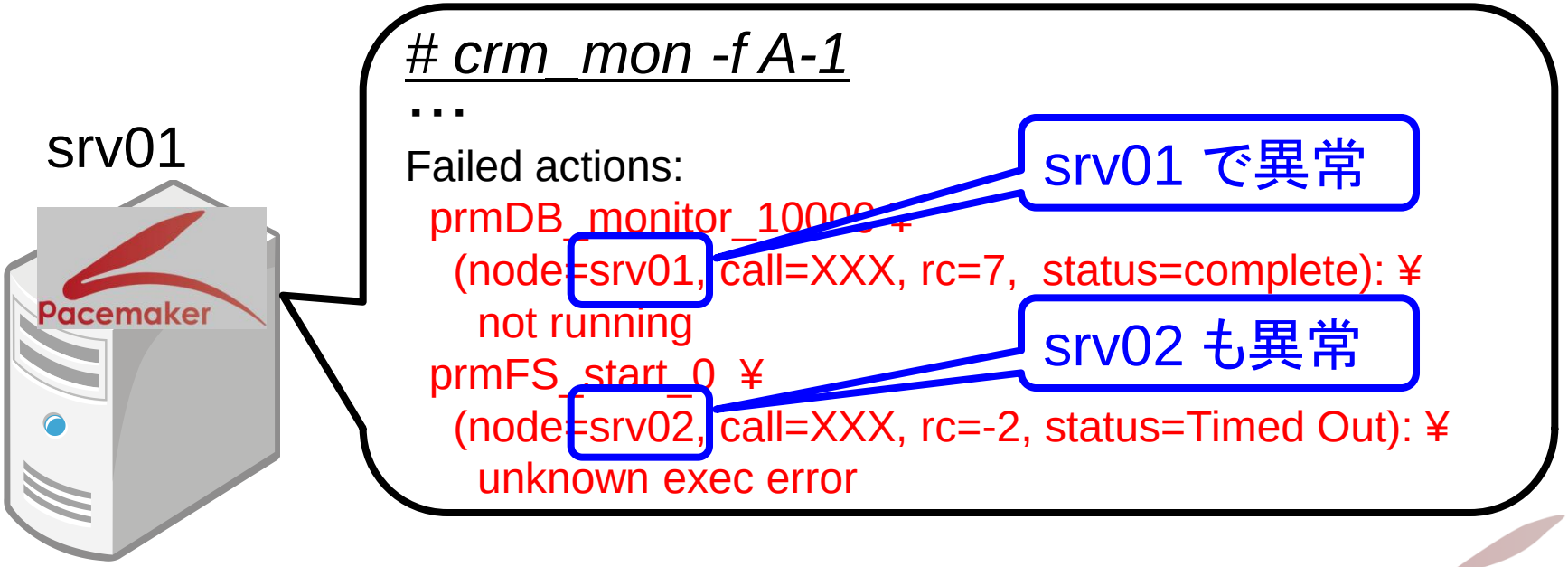
４－２　Disk監視が両系ともに異常

共有Diskに異常が発生している可能性があります。
FCケーブル等に異常が無い場合、Pacemaker を直ちに停止し
データの調査・復旧等を行ってください。



４－３　両系でリソース異常

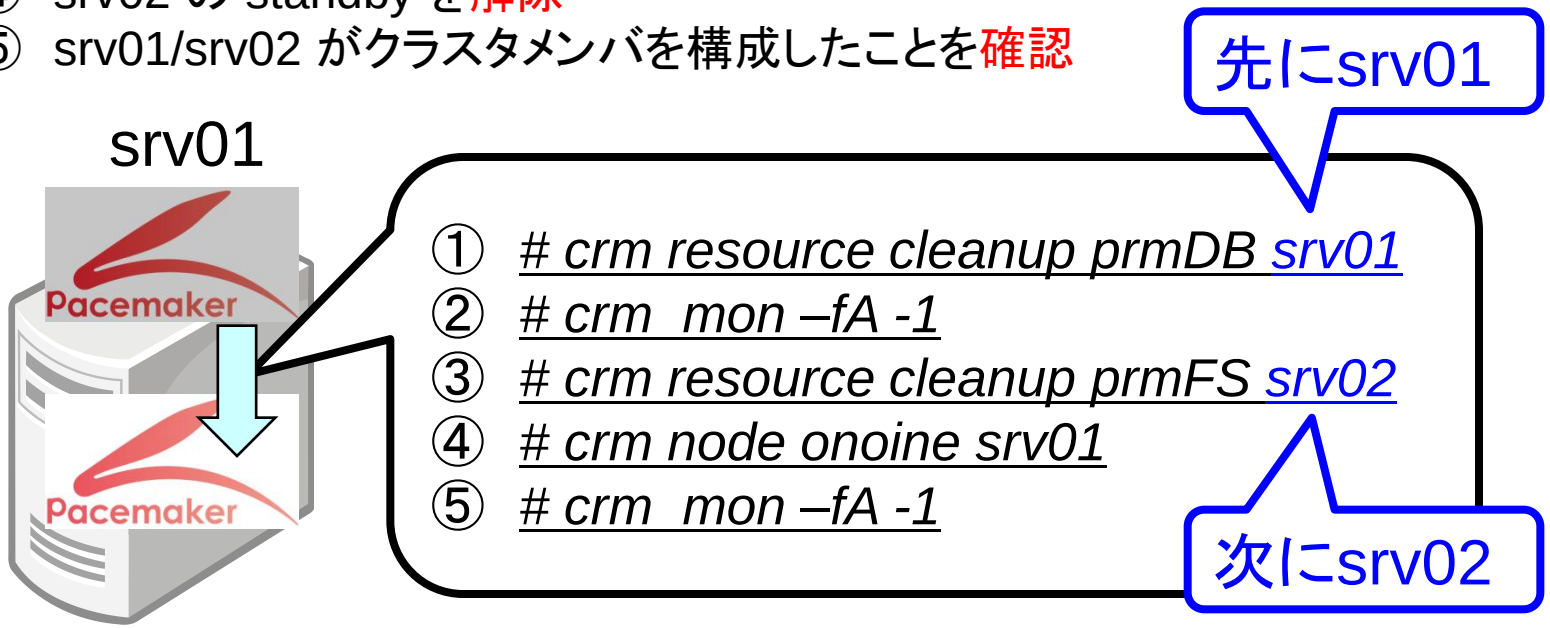
srv01 でリソース監視異常、srv02 でリソース起動異常 というパターンです。
このパターンは割りと多く出ており、その一つに共有ディスクの設定が関わっている場合もあります。（詳細は ③－２: を参照）



４－３　両系でリソース異常

具体的には下記のとおりです
（ただし 共有ディスクの設定を行う場合、その設定は①よりも先に実施します）

① srv01 のリソース異常を解除
② srv01 にてリソースが稼働したことを確認
③ srv02 のリソース異常を解除
④ srv02 の standby を解除
⑤ srv01/srv02 がクラスタメンバを構成したことを確認

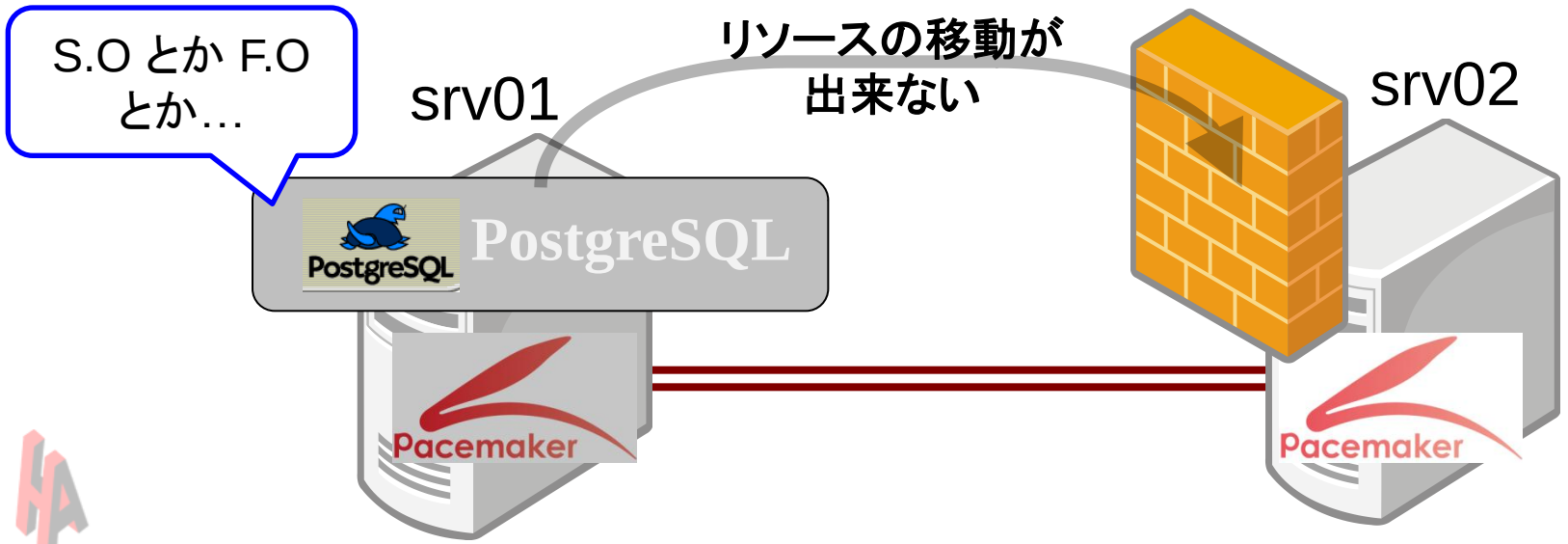


４－４　リソース移動制約が効いている

srv01 へリソース移動制約を付与した後、リソース移動制約解除コマンドの実行を行っていない、且つ srv01 にリソース異常があった場合に発生する事象です。

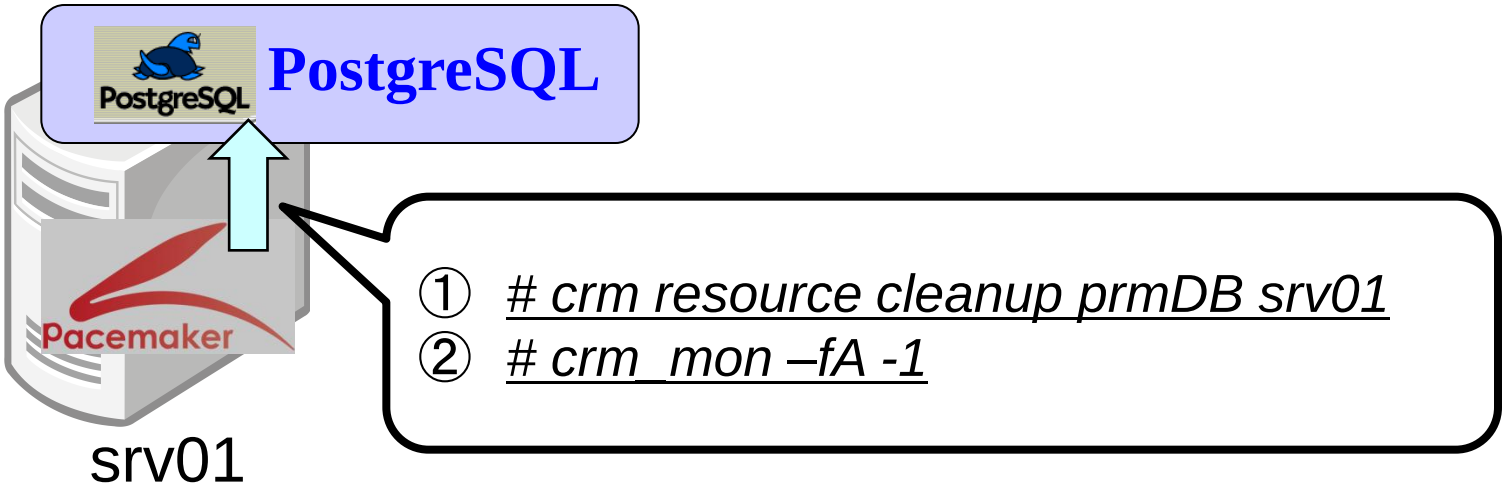
リソース移動制約が付与されていることを crm_mon コマンドでは判定できません。

また制約を付与するコマンドの実行はログに出力されますが、ファイルがローテートされ続けて消えている可能性もあり、判定が容易ではないという側面があります。



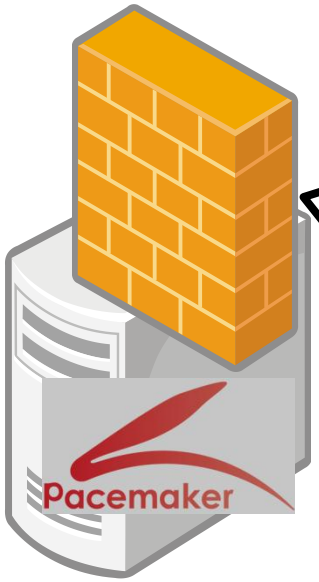
４－４　リソース移動制約が効いている

まずsrv01 に対してリソース異常を解除し、srv01 でリソースを起動させます。
（後に srv01 の ha-log は別途解析が必要です）



４－４　リソース移動制約が効いている

次にリソース移動制約が付与されていることを**確認**します。



srv02

 # cibadmin -Q | grep cli | grep srv01

```
<expression id="cli-prefer-expr-grpDB" attribute="#uname“ ¥
operation="eq“ value=“srv01" type="string"/>
```

 # cibadmin -Q | grep cli | grep srv02

```
<expression id=“cli-standby-expr-grpDB" attribute="#uname" ¥
operation="eq“ value=“srv02“ type="string"/>
```

リソース移動制約を**解除**します。

 # crm resource unmove grpDB

再度**確認** 何も出力されなければ OKです。
srv02 の standby を解除してください。

③

こんな事象が発生！

～ トラブル事例3：タイムアウト値 ～

～ トラブル事例4：共有ディスクの話 ～

1. タイムアウト値について

Pacemaker は crm コマンドでリソースを制御する為の情報を設定します。

RA に対してのパラメータも記述しますが RA毎に各稼働（起動・監視・停止）のタイムアウト値も設定します。

Filesystemに渡すパラメータ

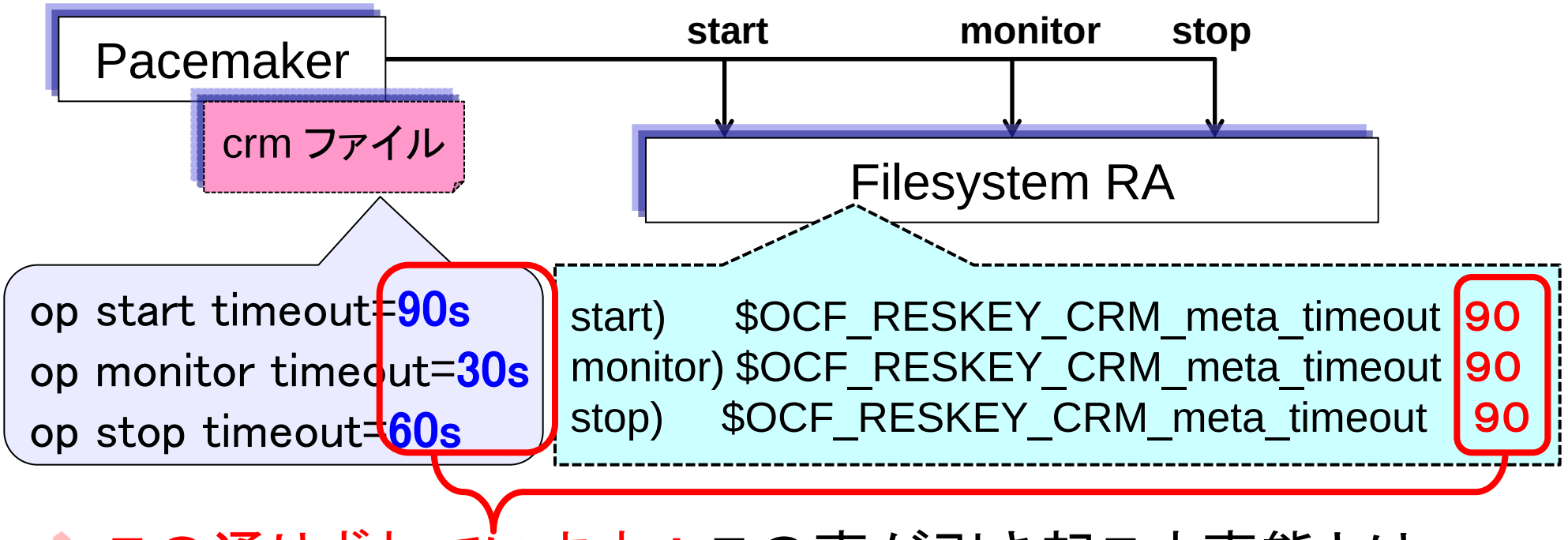
監視間隔やタイムアウト故障時の動作

```
crm(live)configure# primitive MyFS ocf:heartbeat:Filesystem ¥
        params ¥
                fstype="ext3" ¥
                run_fsck="force" ¥
                device="/dev/mapper/mpath1p1" ¥
                directory="/media" ¥
        op start interval="0s" timeout="90s" on-fail="restart" ¥
        op monitor interval="10s" timeout="30s" on-fail="restart" ¥
        op stop interval="0s" timeout="60s" on-fail="fence"
```

1. タイムアウト値について

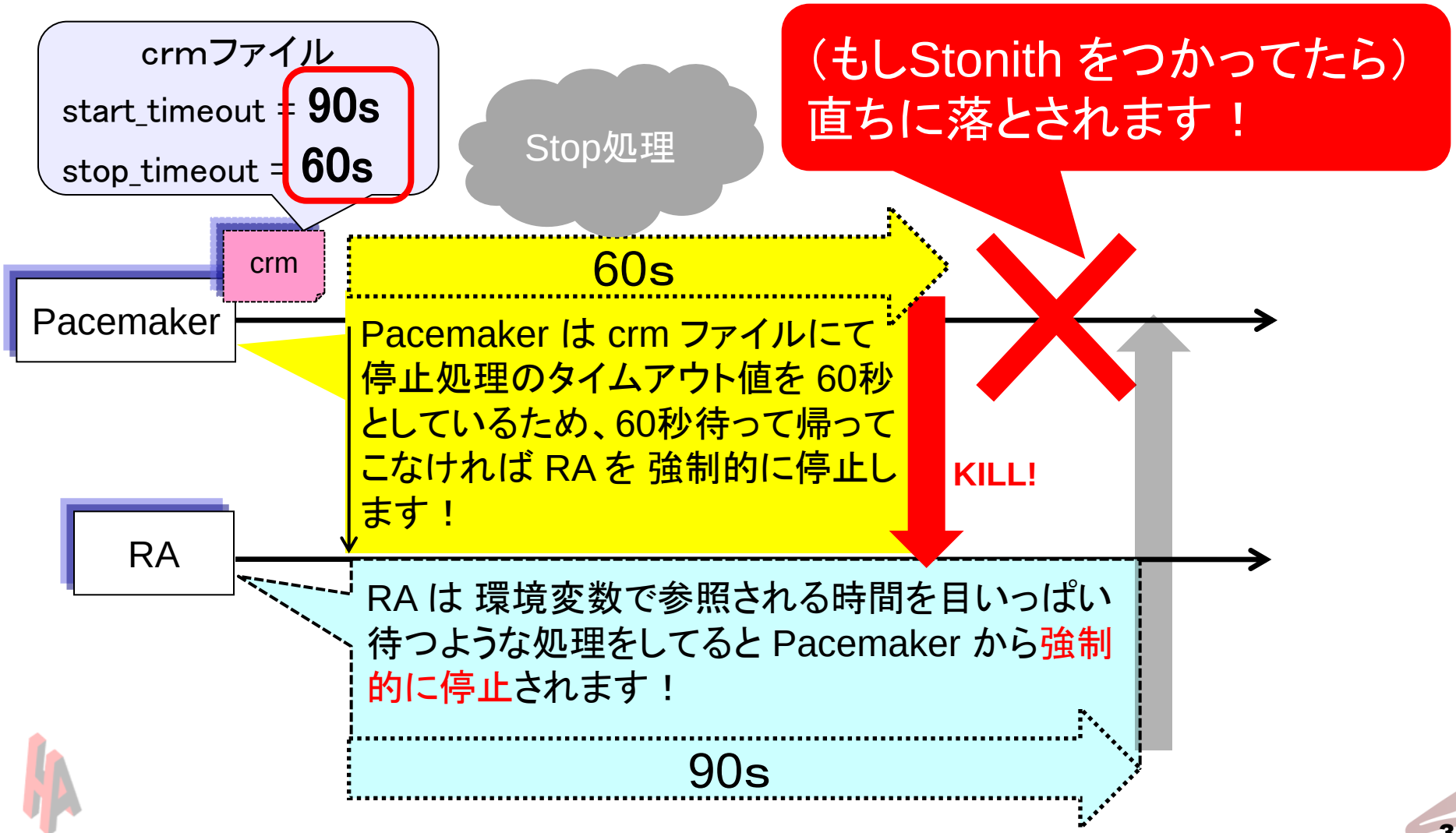
一方、RA は環境変数（※）を参照することで、各稼働における
タイムアウト値をRA 内で参照することが出来ます…が！

※：OCF_RESKEY_CRM_meta_timeout



この通りずれています！この事が引き起こす事態とは…

1. タイムアウト値について



1. タイムアウト値について

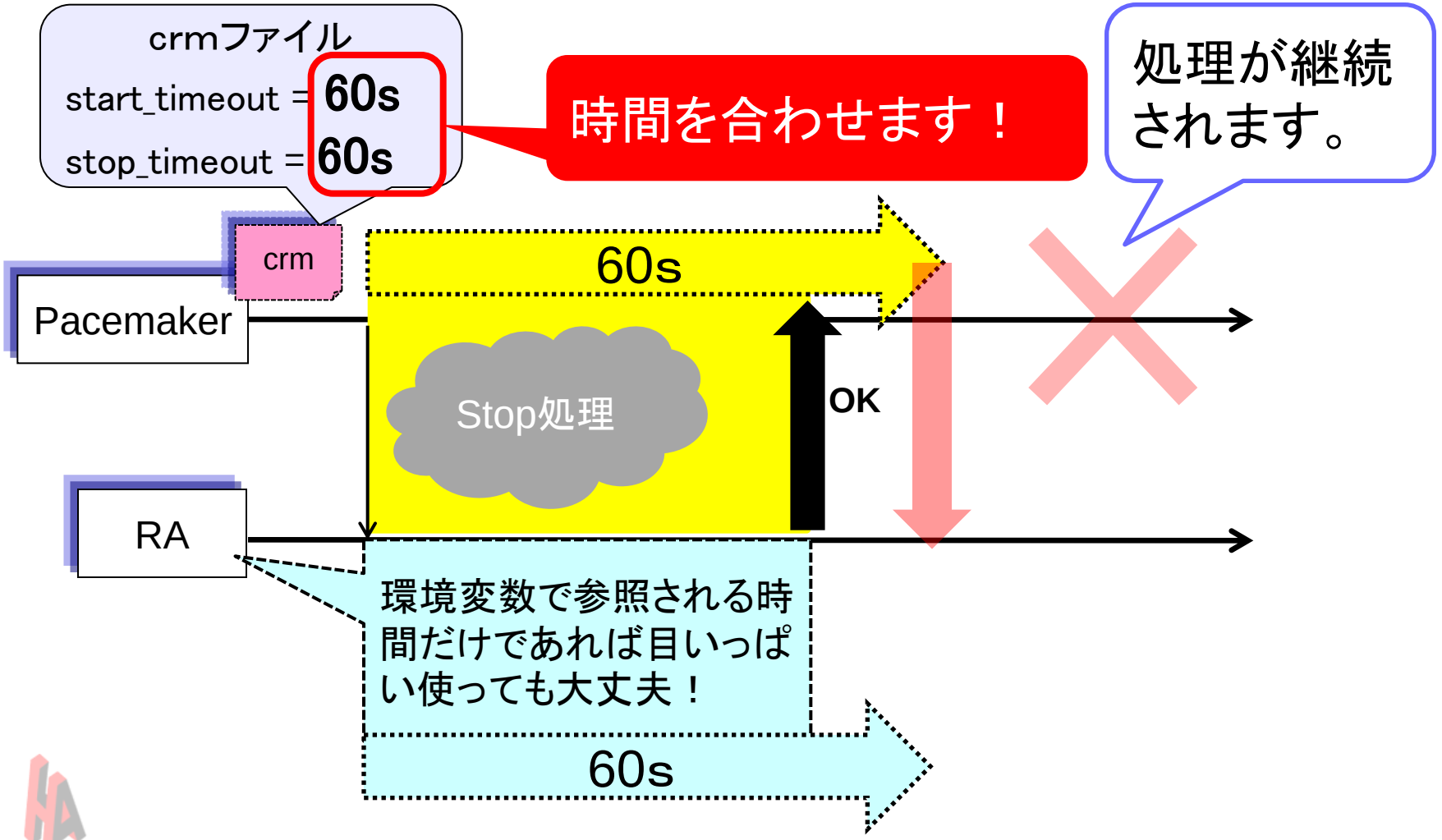
当事象は RA 内で環境変数(※)を使ってループして待つような
処理に影響をあたえます！　対象となる代表的なRA は…

※: OCF_RESKEY_CRM_meta_timeout

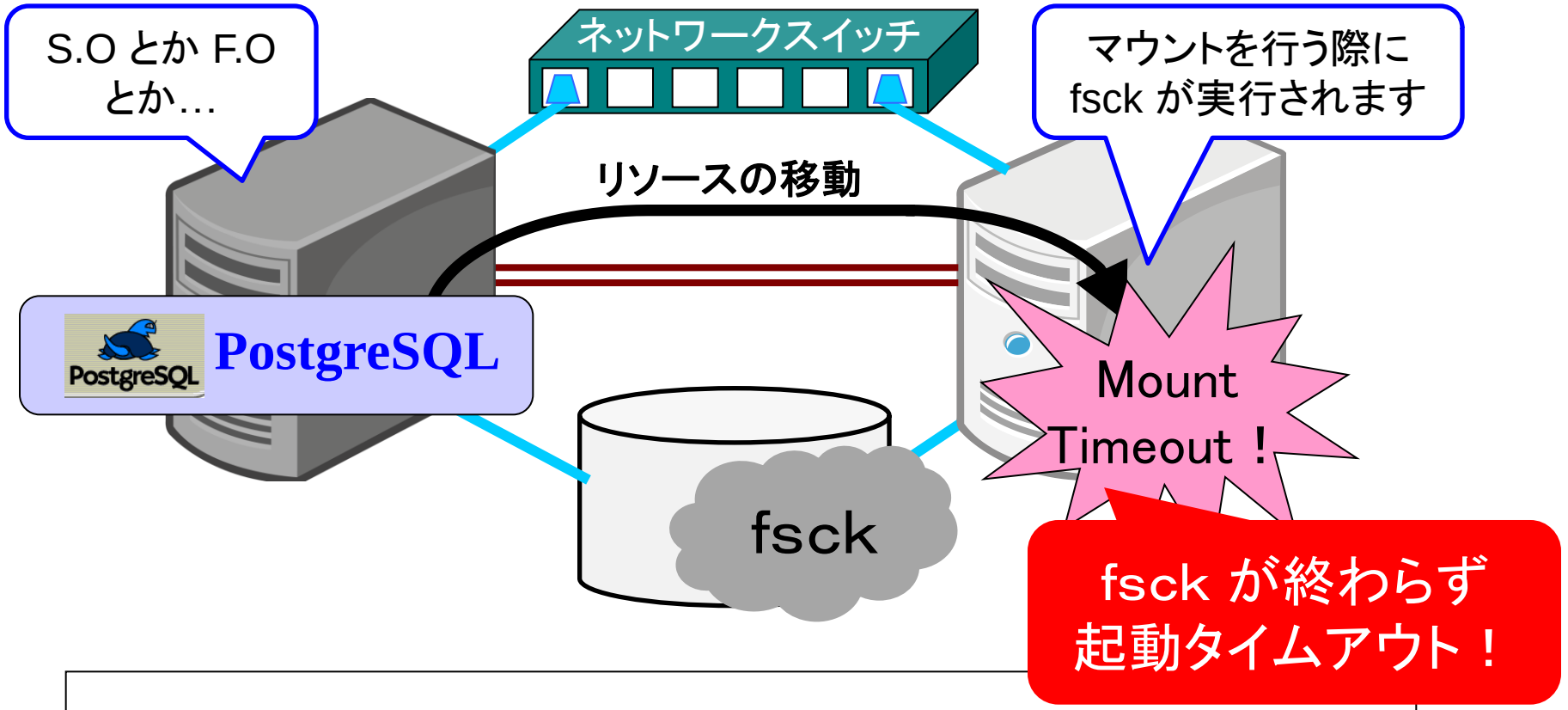
Filesystem RAの停止処理	sfex／MySQL RAの停止処理
アンマウント失敗時、環境変数を2で割り SIGTERM 及び SIGKILL を送信する期間をきめています。	停止処理後、環境変数の5秒前まで待ち、それでもプロセスが存在していたら SIGKILL を送信する等強制終了を試みています。

以上より、リソースの timeout に設定する値は当面
op start timeout = op stop timeout
と設定する事を覚えておいて下さい。

1. タイムアウト値について



2. 共有ディスクに対する事前対処



上記のような状況を避けるには…

2. 共有ディスクに対する事前対処

マウント対象デバイスには事前に しきい値 の
無効化を行っておきましょう！

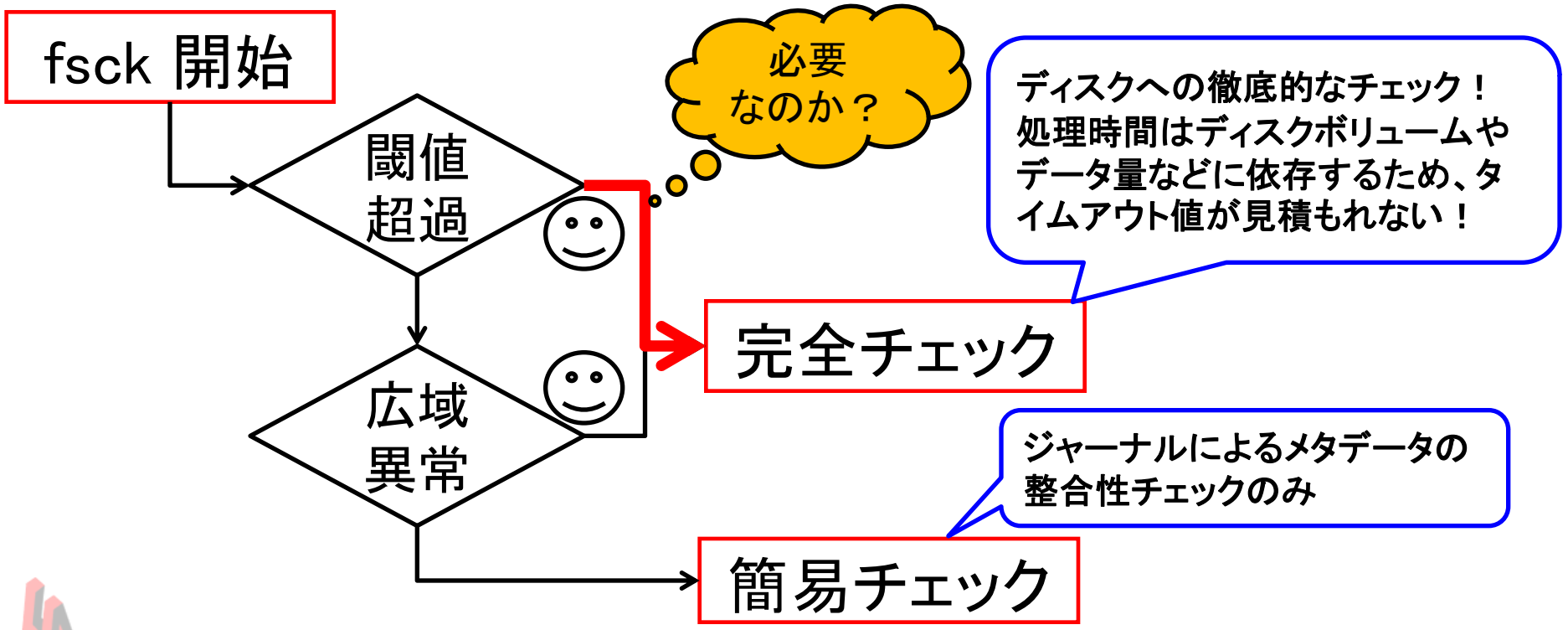
共有ディスクに対して tune2fs コマンドを
実行すると、下記”しきい値”が表示されます。

- マウント回数
- 最終チェック日
- チェック間隔

```
# tune2fs -l <デバイス>  | grep -E Maximum¥|Mount
Mount count:                  100
Maximum mount count:      25
# tune2fs -l <デバイス>  | grep -i check
Last checked:              Tue Jan 24 17:16:10 2012
Check interval:           15552000 (6 months)
Next check after:         Sun Jul 22 17:16:10 2012
```

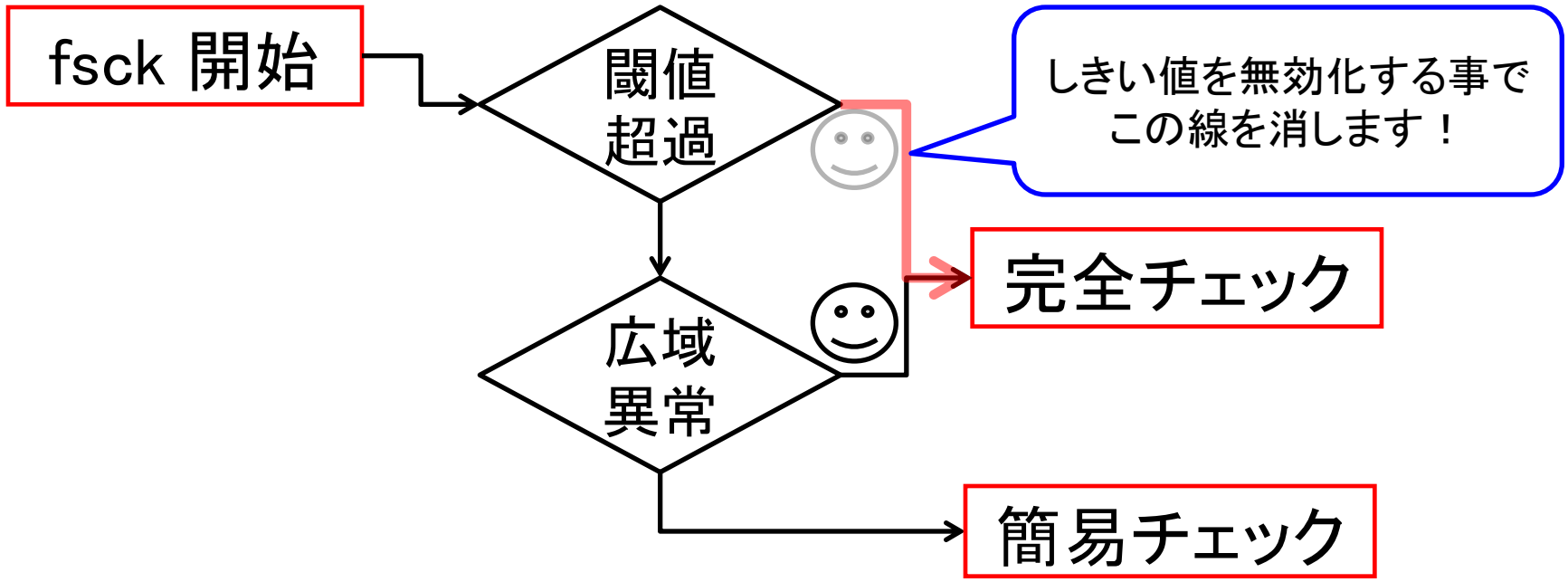
2. 共有ディスクに対する事前対処

これらしきい値を超えて fsck を行いますと fsck の完全チェックが行われ、開始処理にてタイムアウトが発生します。



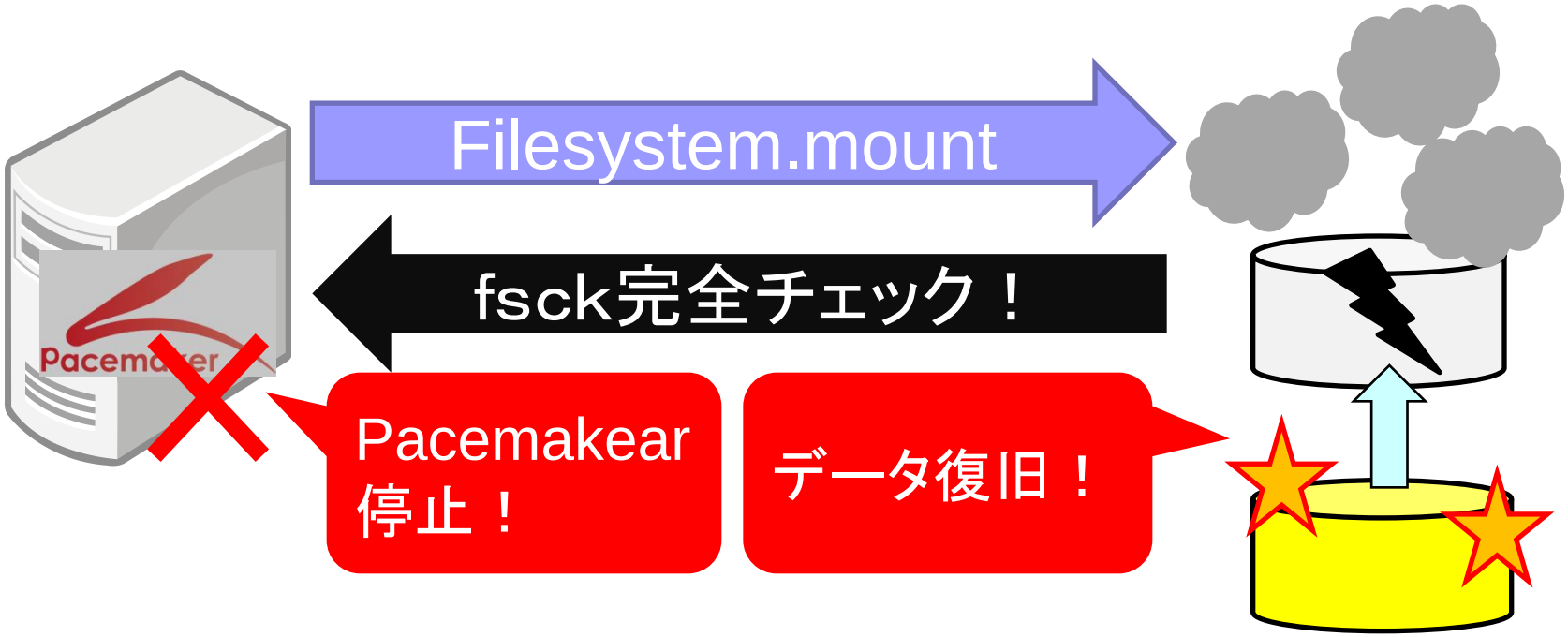
2. 共有ディスクに対する事前対処

このような事象を回避するため、しきい値を無効化します。
その結果、しきい値超えによる fsck 完全チェックは行われなくなり
本当に必要な時（ディスクが広域でヤラれてる時）のみ実行 します。



2. 共有ディスクに対する事前対処

仮に本当にfsck が実行された時、ハードウェア故障が疑われます！
チェックの正常終了を待たず、速やかにデータの確認処理やリストア等の保守運用を行い、サービスの早期再開を目指しましょう。



2. 共有ディスクに対する事前対処

下記はしきい値を無効化する為の手順になります。
管理対象の共有ディスク上の全デバイスが対象です。

```
# tune2fs -c -1 -i 0 <デバイス>
tune2fs 1.39 (29-May-2006)
Setting maximal mount count to -1
Setting interval between checks to 0 seconds

# tune2fs -l <デバイス>  | grep -E Maximum¥|Mount
Mount count:                  100
Maximum mount count:      -1

# tune2fs -l <デバイス>  | grep -i check
Last checked:             Tue Jan 24 17:16:10 2012
Check interval:           0 (<none>)
```

対象デバイスが両系ともにマウントされていないことを事前確認！

④

こんな要望にも答えられます！

～ 解析・要望実現方法 ～

3. 解析について

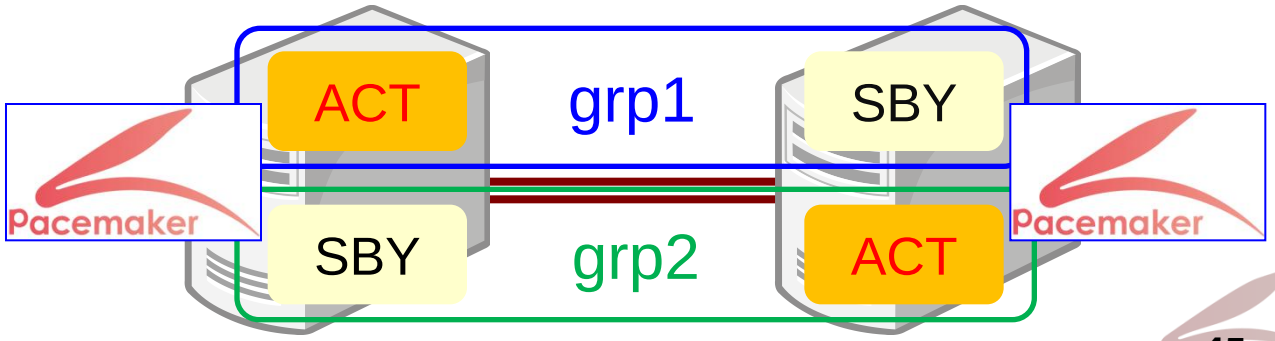
解析作業ではログ（ha-log）は本当に沢山出力されます。
よって pm_logconv のご利用は一時切り分けとして有用です。

ただし…

たすきがけ構成（Act-Act ※）では無理です。
Master/Slave リソースは検知できません。

よって現状では事象の確認は ha-log と messages の参照も必要。
DCノードと 非DCノードは出力されてるログが違うので両系分必要。

※たすきがけ構成例



3. 解析について

個人的には解析の依頼があった場合は最低でも下記の資材の提供を「両系分」お願いしています。

/var/lib/heartbeat/crm/cib.xml
/etc/ha.d/ha.cf
/var/log/ha-log
/var/log/messages

さらに crm ファイルを取得して頂くには…

crm configure save <filename>
でファイルに落としてもらいます。

3. 解析について

基本的には、プライオリティが” ERROR”のログメッセージを中心に探っていきます。

起動・停止・スイッチオーバー・フェイルオーバー等の処理ではどのようなログが出力するのかを事前に抑えておくことも重要です。

尚、個人的にはプライオリティが” WARN”のログメッセージは、基本的には無視してよいと思ってます。

次ページでは、過去調査した（一部の）WARNメッセージと、その意味を説明しています。

問合せのあったWARN の回答の一部です

- lrmd[…]: <日付> WARN: G_SIG_dispatch: Dispatch function for SIGCHLD was delayed 1000 ms (> 100 ms) before being called (GSource:0xc4ef7f8)
  SIGCHLD シグナルを受信して処理を開始するまでの時間が、予め設定された制限時間(100ms)を超えた為に出力しています。短期間に頻出し続けない限り問題ないです。

- heartbeat[…]: <日付> WARN: 1 lost packet(s) for [vnk･･･] [･･:･･]
  Heartbeat 内の状態確認などの通信（UDP）にてパケットロストが発生（検知）。
  頻発しない限り問題・対処の必要はありません。

- attrd[…]: <日付> WARN: add_cib_op_callback: CIB call failed: ¥
  The object/attribute does not exist
  （詳細省きますが）起動時にタイミングにより出力されるメッセージで、問題ありません。

- diskd[…]: <日付> WARN: check_old_status: disk status is changed, new_status = normal
  diskd による内部状態変更メッセージ、起動時且つ new_status = normal であれば問題ありません。

- crmd[…]: <日付> WARN: do_log: [[FSA]] Input I_DC_TIMEOUT from crm_timer_popped() ¥
  received in state (S_PENDING)
  起動時に参加ノードの締め切りを表すメッセージ、問題ありません。

- cib[…]: <日付> WARN: ccm_connect: CCM Activation failed
  cib[…]: <日付> WARN: ccm_connect: CCM Connection failed 1 times (30 max)
  Heartbeat の内部プロセスである CCM に対する接続を行う際、まだ CCM が起動中であった為に出力されている、問題ありません。

  などなど･･･

4. SNMPについて

実は Trap が飛ばせます。 snmpwalk で情報取得もできます。
でもちょっと怪しい動作（特に Trap）も…

OID（enterprises）	Trap の内容
.4682.900.1	ノード状態の変化
.4682.900.3	ノードのリンク状態の変化
.4682.900.5	ノードのメンバシップ状態の変化
.4682.900.7	snmp エージェントの起動
.4682.900.9	snmp エージェントの停止
.4682.900.11	リソース状態の変化

Trap は UDP という事もあるので、Trap に頼らない運用が望ましいです（定期的に snmpwalk をするとか）。

● ha.cf の設定　以下を追記

```
respawn root /usr/lib64/heartbeat/hbagent -r 0
```

● snmpd.conf の設定　以下を追記

```
master agentx
trapcommunity public
trap2sink <Trapを飛ばしたい先の IPアドレス>
```

※　enterprises(.1.3.6.1.4.1)を読める view である事が前提になります

● mibファイルの場所（自動的に置かれます）

```
/usr/share/snmp/mibs/LINUX-HA-MIB.txt
```

5. 特定のリソースを止めたい！

F.O とかはイヤ！障害通知がエラい人のところに上がってしまうのは避けたい（調整めんどう）という人はそれなりにいらっしゃいます。

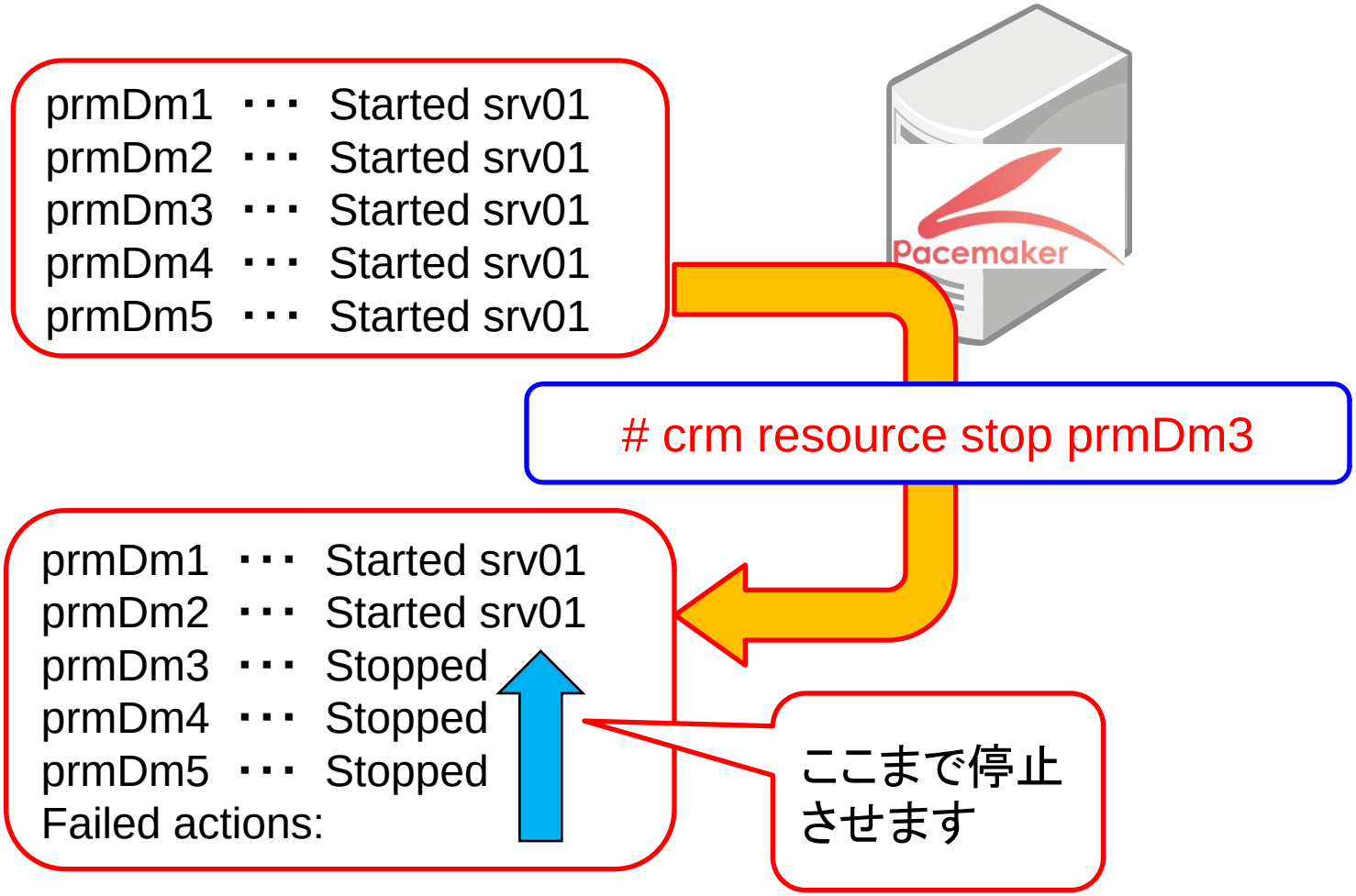
そんな時は crm resouce stop/start を
使用します！

```
# crm resource stop <リソース名／グループ名>
  ・・・ ゴニョゴニョ ・・・
# crm resource start <リソース名／グループ名>
```

アプリケーションや設定ファイルの更新に便利です。
対象リソースよりも上位リソースも停止されます、気をつけて下さい。

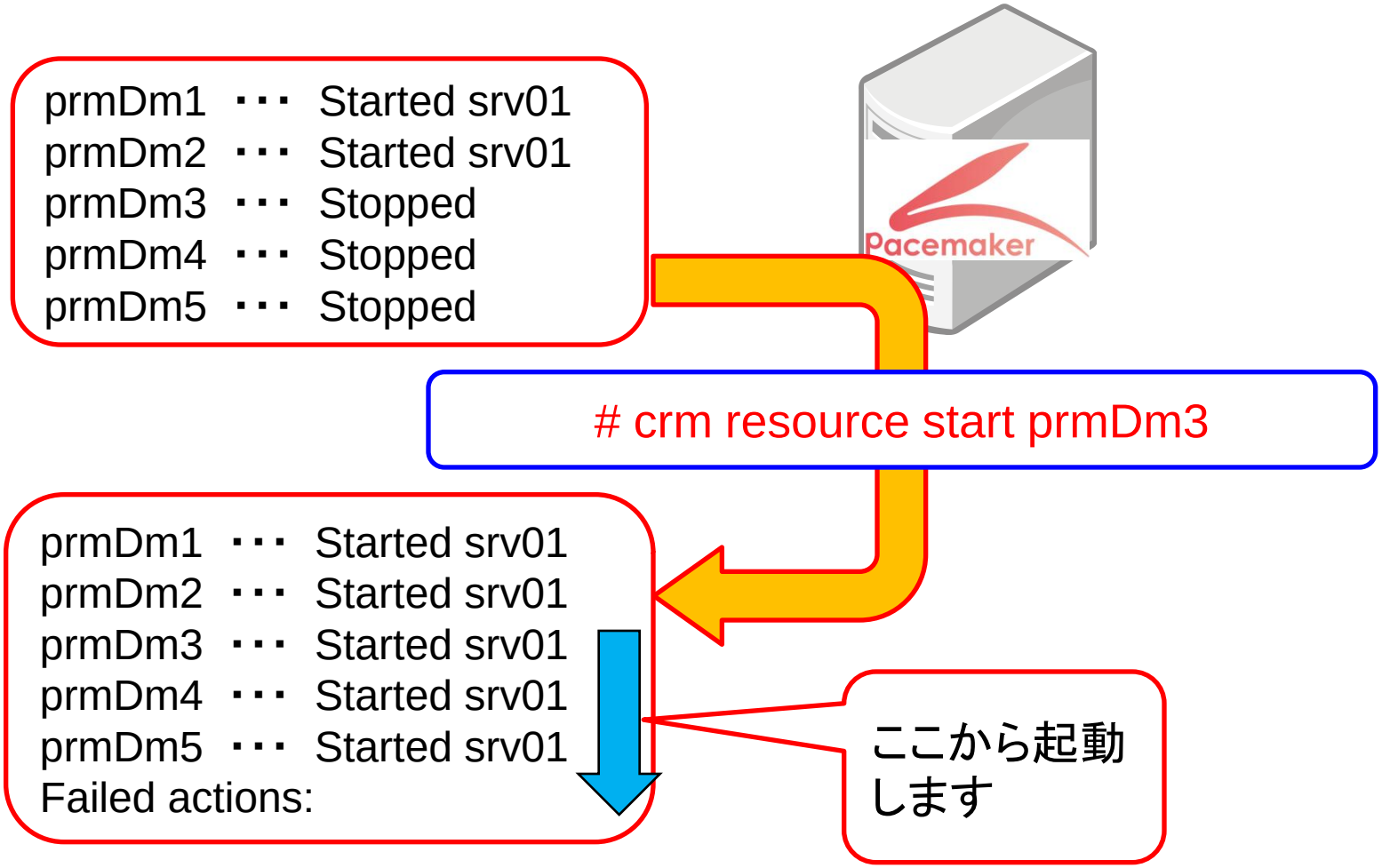
●target-role の動作について

以下、prmDm3 に crm resource stop を実行すると…



●target-role の動作について

prmDm3 の crm resource start を実行すると…



6. 特定のPacemakerを止めたい！

F.O とかは絶対イヤ！という人はそれなりにいらっしゃるかもしれません（ha.cf の編集だけしたいとか）…。

そんな時は メンテナンスモード を使用します！

```
# crm configure property maintenance-mode=true
crm_verify[XXX]: <日付> WARN: unpack_nodes: Blind faith: not fencing unseen nodes
# service heartbeat stop
・・・ ゴニョゴニョ ・・・
# service heartbeat start
# crm configure property maintenance-mode=false
```

この WARN は無視で！

リソースを一切止めることなく、Pacemaker だけ（両系ともに）落とす事ができます。

ただし！その間、リソースの異常を検知する事は出来ません。

●メンテナンスモード の動作について

以下、メンテナンスモードに移行すると…

prmDm1 ・・・ Started srv01
prmDm2 ・・・ Started srv01
prmDm3 ・・・ Started srv01
prmDm4 ・・・ Started srv01
prmDm5 ・・・ Started srv01

crm configure property maintenance-mode=true

prmDm1 ・・・ Started srv01 (unmanaged)
prmDm2 ・・・ Started srv01 (unmanaged)
prmDm3 ・・・ Started srv01 (unmanaged)
prmDm4 ・・・ Started srv01 (unmanaged)
prmDm5 ・・・ Started srv01 (unmanaged)

Pacemaker だけ再起動してもリソースは稼働したまま

clone, stonith もすべて unmanaged ！

7. 特定のNW機器を止めたい！

サービスLAN監視先のNW機器を止めるとPacemakerが異常と検知しリソースが縮退してしまいます。そういうのは面倒という人はそれなりにいらっしゃるかもしれません…。

そんな時は（ちょっと面倒ですが）crmコマンドでファイルを編集します！

```
# crm configure edit
（“default_ping_set” を検索 “/ default_ping_set”）
default_ping_set lt 100
↓
default_ping_set lt 0
（保存して終了 “:wq”）
```

```
# crm_mon -A -1 | grep default_ping_set
   + default_ping_set              : 100
   + default_ping_set              : 100

# iptables -A INPUT -s <監視先IPアドレス> -j DROP
# crm_mon -A -1
・・・
   prmDummy (ocf::pacemaker:Dummy)     :   Started node1
・・・
   + default_ping_set              : 0          : Connectivity is lost
   + default_ping_set              : 100
```

見た目では解りづらいですが、NW機器メンテ等の運用では利用できる機能と思います。

ちゃんと戻すことを忘れないで下さい！

ご清聴ありがとうございました