Pacemakerで簡単・手軽にクラスタリングしてみよう！

2010年6月26日

Linux-HA Japan プロジェクト
田中崇幸

本日の話題

① Pacemakerって何？
② Pacemakerのコンポーネント構成
③ Pacemakerを動かそう！
④ Linux-HA Japanプロジェクトについて
⑤ 参考情報

①

Pacemakerって何？

簡単に言うと・・・

Pacemakerとは？

PacemakerとHeatbeatの関係は後でお話します

オープンソースで実現する
高可用性クラスタリングソフトウェアで
実績のある「Heartbeat」の後継ソフトウェアです

Pacemakerは、サービスの可用性向上ができるHAクラスタを可能とした、コストパフォーマンスに優れたオープンソースのクラスタリングソフトウェアです。

HighAvailability

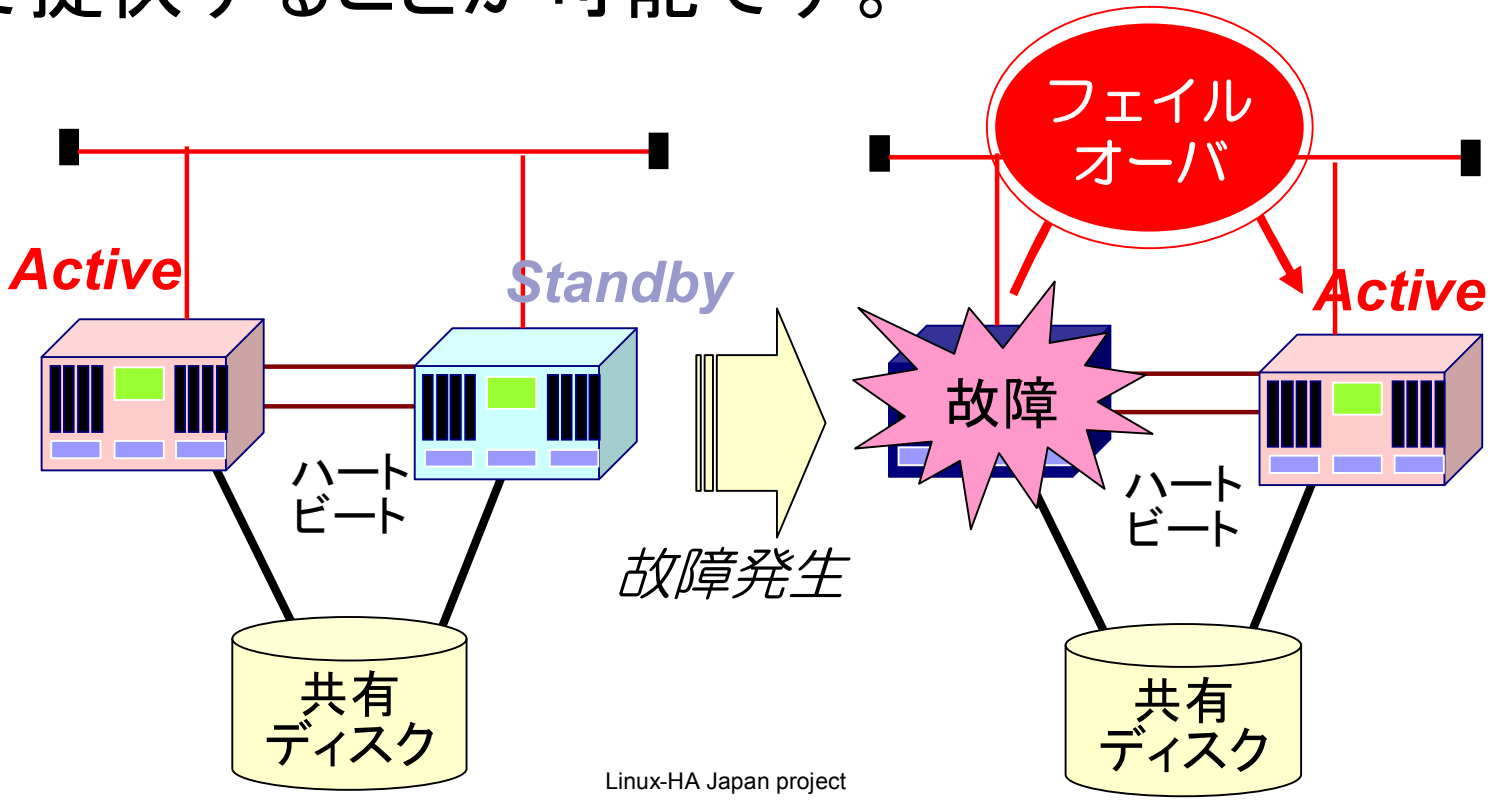
HAクラスタとは？

HAは「ハイ・アベイラビリティ」（High Availability）の略で、日本語では「高可用性」と訳されます。

あるサービスを提供するノードが落ちたときに、予備機がそのサービスを引き継ぐことにより、サービスのダウンタイムを減少させ、冗長性を持たせることが目的です。

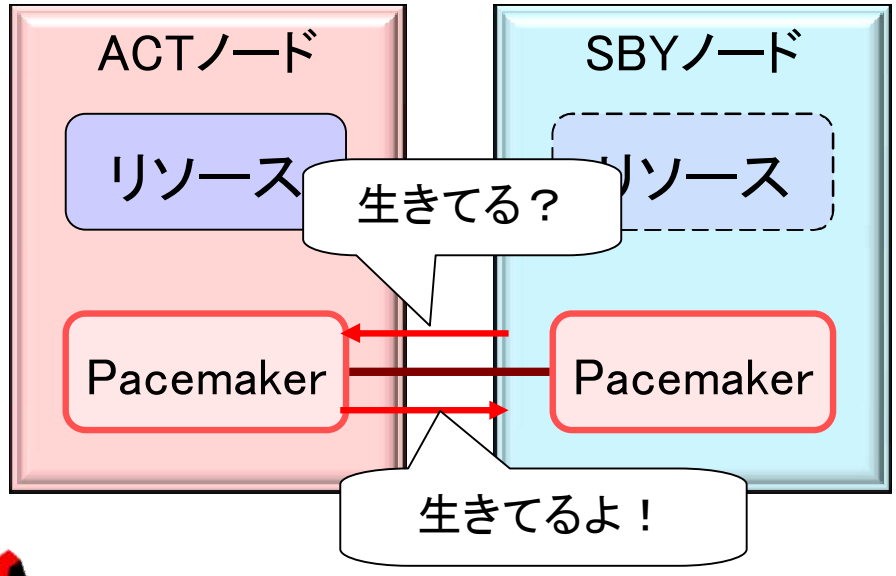
概要

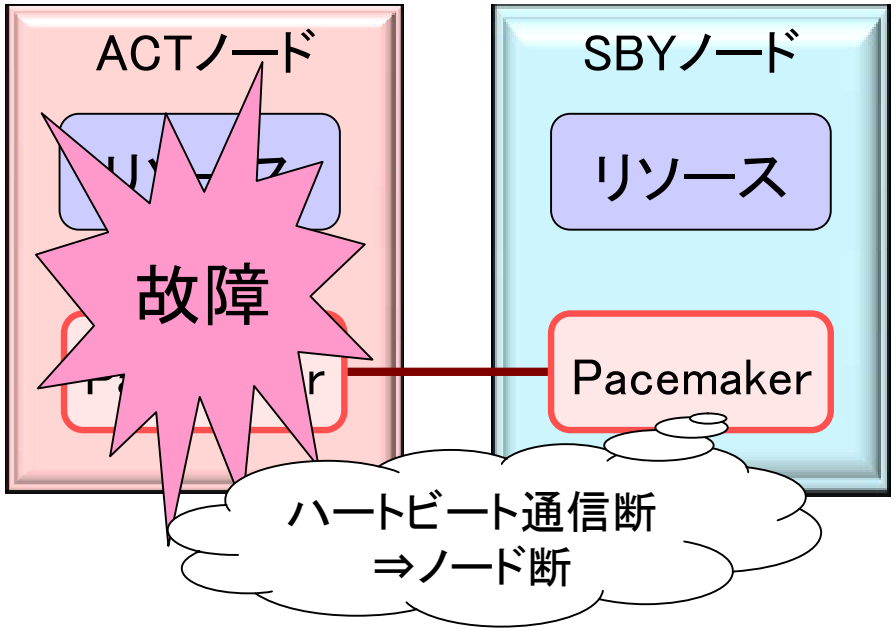
- Pacemakerは、故障発生を検知し、待機系サーバへフェイルオーバさせることが可能です。
- サービス利用者には故障を意識させずにサービスを継続して提供することが可能です。



基本的動作：ノード監視

- 相手ノードの監視
  - 一定間隔で相手ノードと通信し、相手ノードの生死を確認します。（ハートビート通信）
  - 相手ノードと通信できなくなった場合に、相手はダウンしたと判断し、フェイルオーバ処理を行います。





「リソース」「リソースエージェント」とは？

Pacemakerではよく出てくる
言葉なのでおぼえてください！

- リソース

HAクラスタにおけるリソースとは、サービスを提供するために必要な構成要素の事で、Pacemakerが起動、停止、監視等の制御対象とするアプリケーション、NIC、ディスク等を示します。

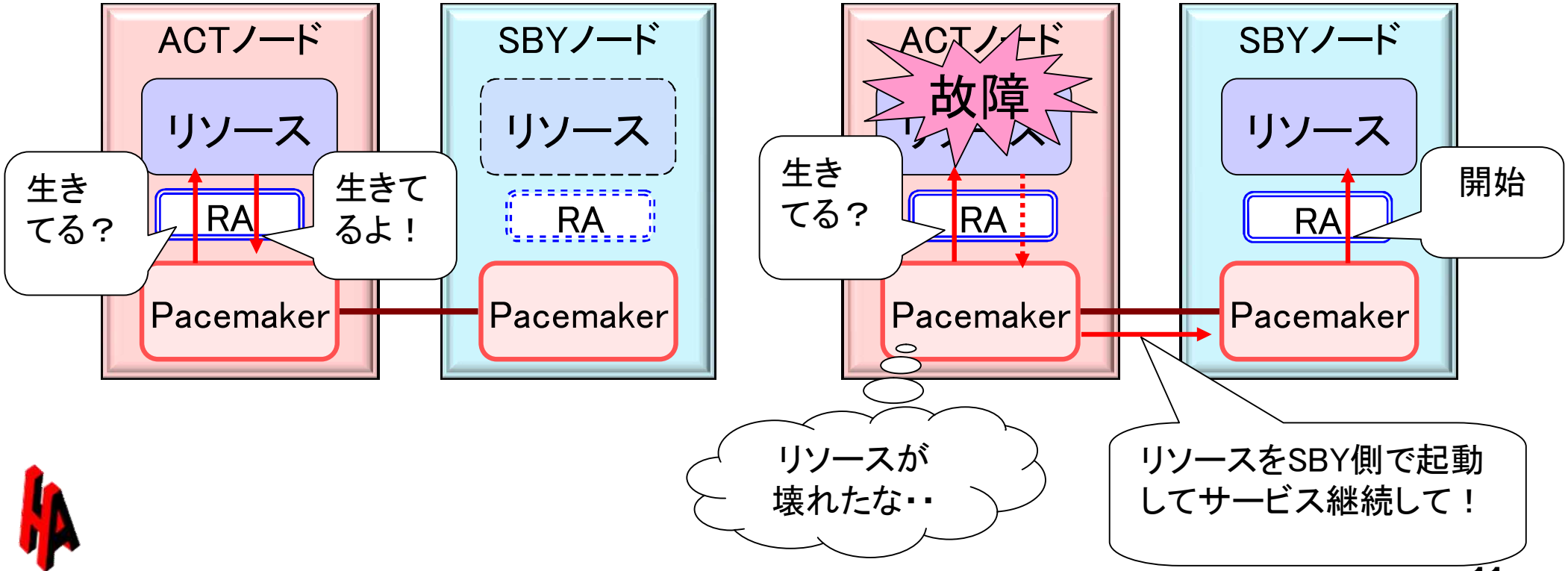
- リソースエージェント（RA）

  リソースエージェント（RA）とは、そのリソースとPacemakerを仲介するプログラムになり、主にシェルスクリプトで作成されています。

  Pacemakerは、リソースエージェントに対して指示を出し、リソースの起動(start)、停止(stop)、監視(monitor)の制御を行います。

基本的動作：リソース制御

- リソースの制御：起動(start)、停止(stop)、監視(monitor)
  - 起動後は一定間隔でRAを介してリソースを監視し、正しく動作していないと判断した場合にはフェイルオーバ等の処理を実施します。



Pacemakerでは、Web系、DB系、ネットワーク系、ファイルシステム系等のリソースエージェントがなど、標準で多数用意されています。

MySQLや、Tomcat用のリソースエージェントなどもありますよ！

標準リソースエージェントの一例

分類	リソース	リソースエージェント /usr/lib/ocf/resource.d/heartbeat/ /usr/lib/ocf/resource.d/pacemaker/
ファイルシステム系	ディスクマウント	Filesystem
DB系	PostgreSQL	pgsql
Web系	Apache	apache
ネットワーク系	仮想IPアドレス	IPaddr

pgsqlリソースエージェント

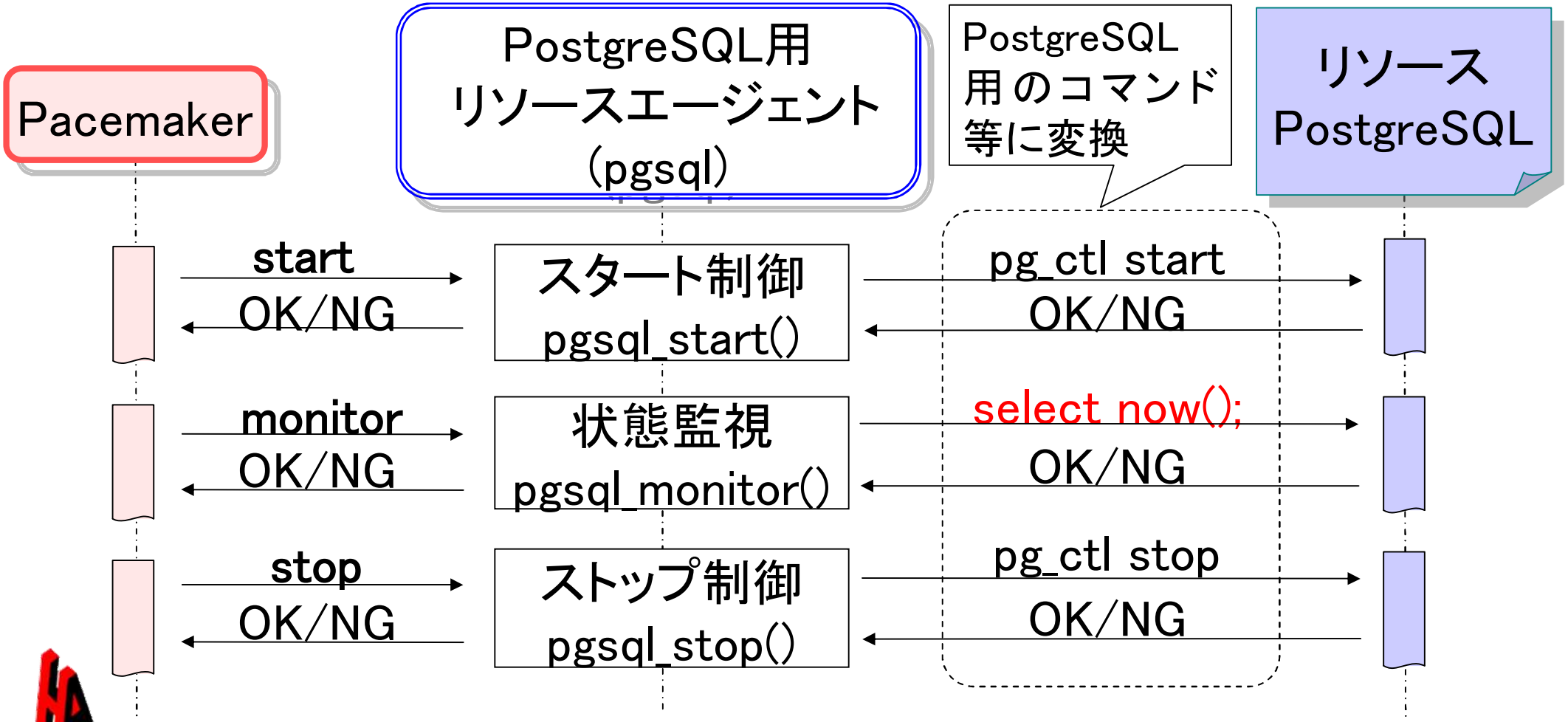
監視(monitor)処理の抜粋

```
#!/bin/sh

 (省略)
pgsql_monitor() {
    if ! pgsql_status
    then
        ocf_log info "PostgreSQL is down"
        return $OCF_NOT_RUNNING
    fi

    if [ "x" = "x$OCF_RESKEY_pghost" ]
    then
       runasowner "$OCF_RESKEY_psql -p $OCF_RESKEY_pgport -U
$OCF_RESKEY_pgdba $OCF_RESKEY_pgdb -c 'select now();' >/dev/null 2>&1"
    else
 (省略)
```

例）Pacemaker と PostgreSQLリソースエージェントの関係

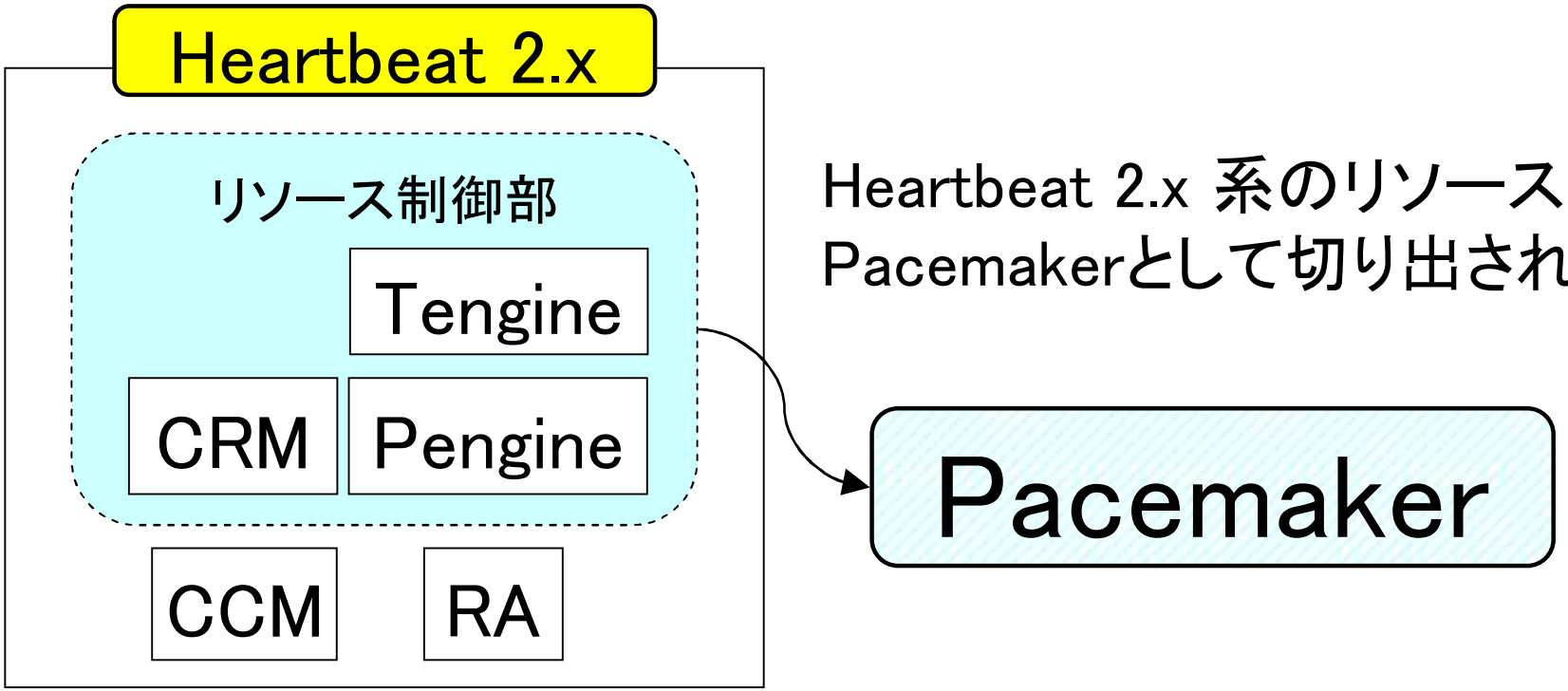


②

Pacemakerの
コンポーネント構成

Pacemaker のコンポーネント構成は
複数に分かれていて
単純ではないのです…

Pacemaker



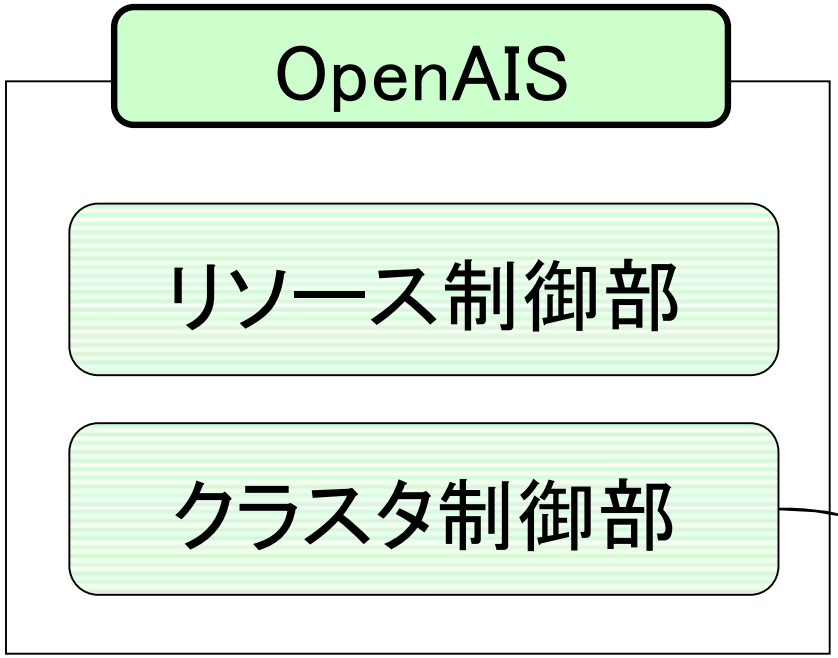
Heartbeat 2.x 系のリソース制御部が
Pacemakerとして切り出されました。

CRM: Cluster Resource Manager
Tengine: Transition Engine
Pengine: Policy engine
CCM: Cluster Consensus Membership
RA: Resource Agent

つまり Pacemaker 単独では
HAクラスタソフトとして
動作しないのです

ノード監視等を行う
「クラスタ制御部」が必要..

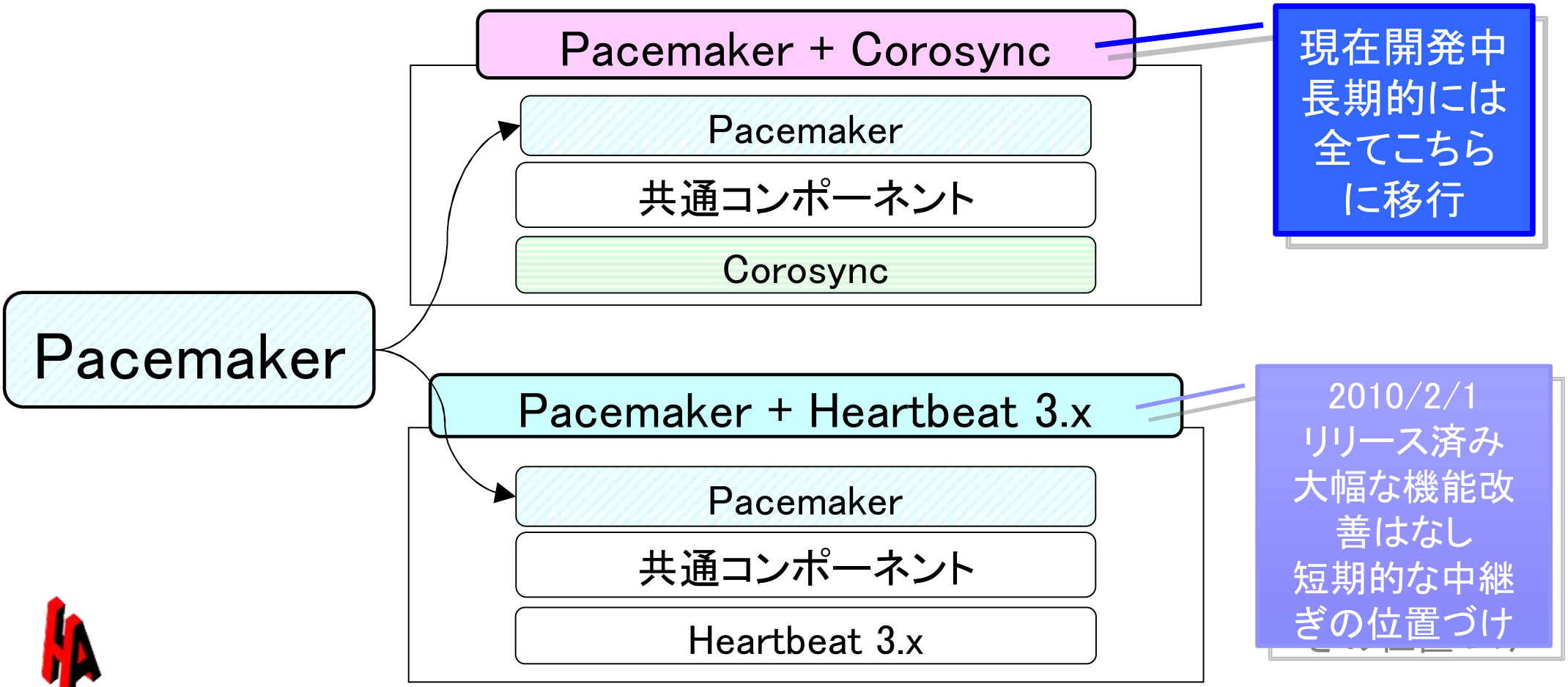
Corosync



OpenAISコミュニティによって開発されたクラスタソフトである【OpenAIS】のクラスタ制御部はCorosyncとして分離されました。

Corosync

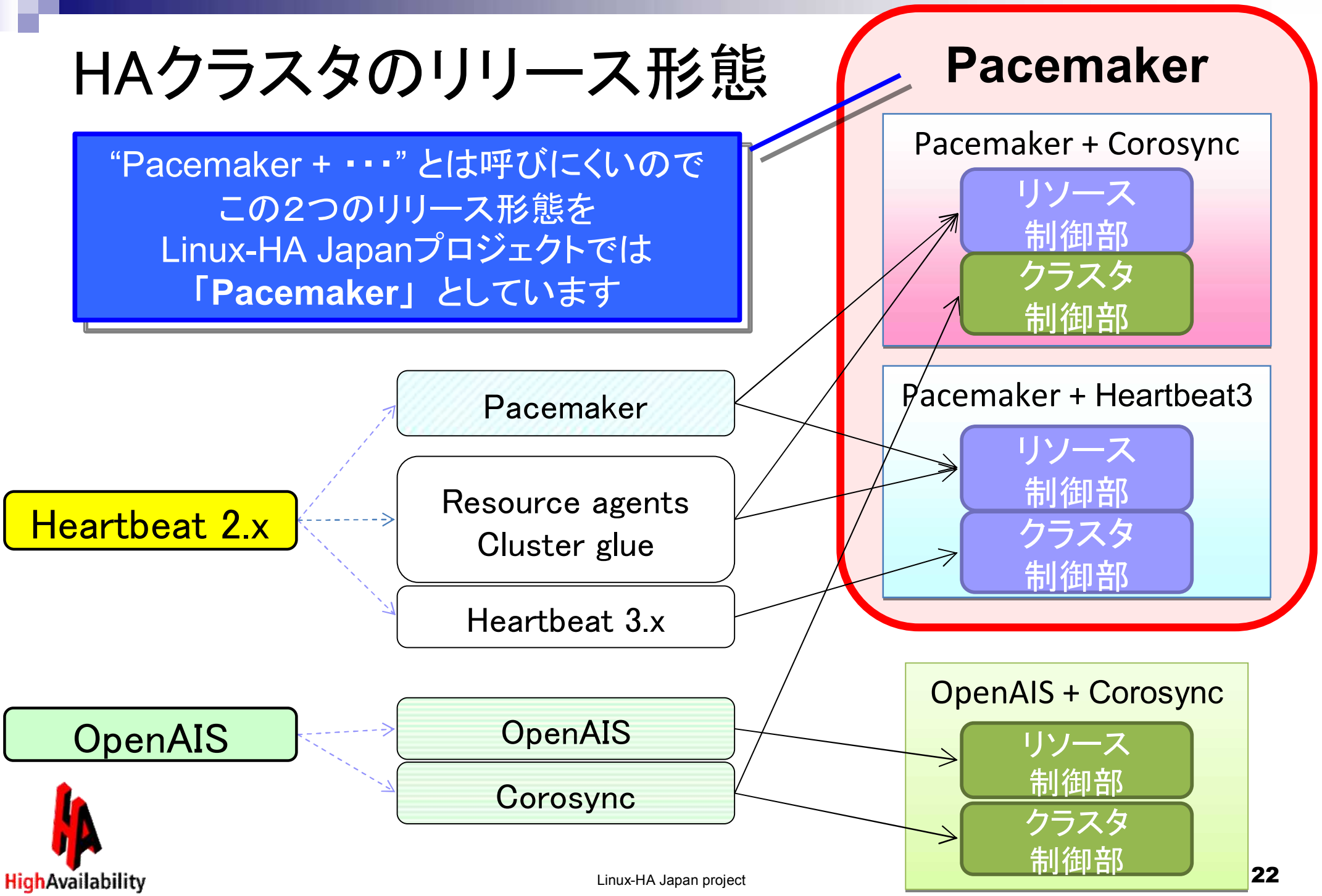
Pacemakerは単独で動作させるのではなく、複数のコンポーネントの組み合わせとして提供されます。
開発コミュニティでは、クラスタソフトウェア間でのコンポーネントの共通化を行い、コミュニティを統合していくという流れになっています。



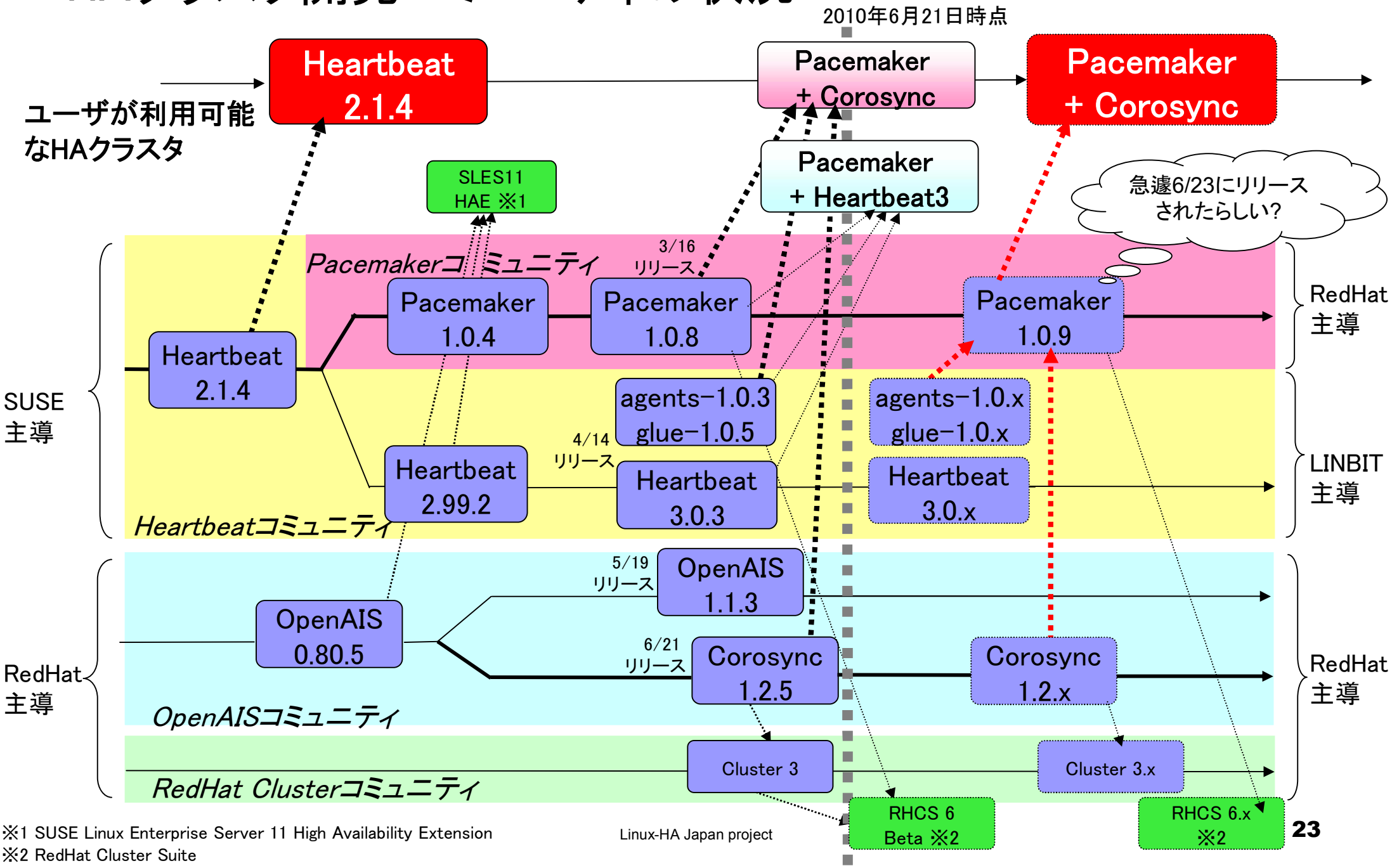
この複数のコンポーネント構成が
少々フクザツなのです…

HAクラスタのリリース形態

“Pacemaker + ・・・” とは呼びにくいので
この2つのリリース形態を
Linux-HA Japanプロジェクトでは
「**Pacemaker**」としています



HAクラスタ開発コミュニティの状況



※1 SUSE Linux Enterprise Server 11 High Availability Extension
※2 RedHat Cluster Suite

③

Pacemakerを動かそう！

Pacemakerのインストール方法は？

Pacemaker rpmパッケージ一覧

CentOS5.5(x86_64)に、「Pacemaker + Corosync」によるHAクラスタを構築する場合の、rpmパッケージ一覧です。
2010年6月26日現在で公開されている最新rpmのバージョンです。

- pacemaker-1.0.9.1-1.el5.x86_64.rpm
- pacemaker-libs-1.0.9.1-1.el5.x86_64.rpm
- corosync-1.2.5-1.3.el5.x86_64.rpm
- corosynclib-1.2.5-1.3.el5.x86_64.rpm
- cluster-glue-1.0.5-1.el5.x86_64.rpm
- cluster-glue-libs-1.0.5-1.el5.x86_64.rpm
- resource-agents-1.0.3-2.el5.x86_64.rpm
- heartbeat-3.0.3-2.el5.x86_64.rpm
- heartbeat-libs-3.0.3-2.el5.x86_64.rpm

こんなに沢山のrpmを
ダウンロードしてくるのは大変…

しかし
CentOS5系（RHEL5系）ならば、
yumを使えば
インストールは簡単！

CentOS5.5(x86_64)の場合のPacemakerインストール方法

※ Pacemaker + Corosync の場合の例です。

- 足りないライブラリのインストール

  Pacemakerのインストールには、rpmパッケージ依存の関係上、libesmtpのインストールが必要です。
  CentOS5.5にはlibesmtpは同梱されて無いため、
  download.fedora.redhat.comからダウンロードします。

```
# wget http://download.fedora.redhat.com/pub/epel/5/x86_64/libesmtp-1.0.4-5.el5.x86_64.rpm

# rpm –ivh libesmtp-1.0.4-5.el5.x86_64.rpm
```

yumリポジトリを設定

clusterlabs.org からrepoファイルをダウンロードして、yumリポジトリを設定します。

```
# cd /etc/yum.repo.d
# wget http://clusterlabs.org/rpm/epel-5/clusterlabs.repo
```

※ clusterlabs.repoの内容

```
name=High Availability/Clustering server technologies (epel-5)
baseurl=http://www.clusterlabs.org/rpm/epel-5
type=rpm-md
gpgcheck=0
enabled=1
```

yumで簡単インストール！

これだけでインストールは完成！

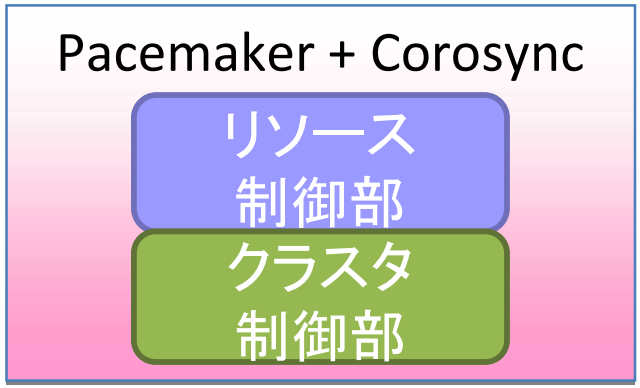
```
# yum install corosync.x86_64 heartbeat.x86_64 pacemaker.x86_64
```

rpmの依存関係で以下のパッケージも自動的にインストールされます。

- pacemaker-libs
- corosynclib
- cluster-glue
- cluster-glue-libs
- resource-agents
- heartbeat-libs

Pacemakerの設定方法は？

Pacemaker では
「クラスタ制御部」「リソース制御部」
それぞれの設定が必要です



クラスタ制御部の設定（Corosync）

クラスタ
制御部

- /etc/corosync/corosync.conf
  - クラスタの基本的な動作情報
  - クラスタ内の全ノードに同じ内容のファイルを配置

```
compatibility: whitetank
aisexec {
        :
}
service {
        :
}
totem{
        :
}
logging{
        :
}
```

corosync.confに
4つのディレクティブ
の設定が必要です。

aisexec

aisexecディレクティブにはクラスタを実行するユーザとグループを指定します。
クラスタの子プロセスは RA を実行するのに十分な権限を所有している必要があるため、rootユーザで実行するように指定します。

```
aisexec {
        user: root
        group: root
}
```

実行ユーザ・グループ名

service

使用するクラスタに関する情報を指定します。

```
service {
    name: pacemaker
    ver: 0
}
```

使用するクラスタ(pacemaker)を指定

totem

ノードがクラスタ内で使用するプロトコルのバージョンやオプション、暗号化などハートビート通信方法を指定します。

```
totem {
        version: 2
        secauth: off
        threads: 0
        rrp_mode: none
        clear_node_high_bit: yes
        token: 4000
        consensus: 10000
        rrp_problem_count_timeout: 3000
        interface {
                ringnumber: 0
                bindnetaddr: 192.168.1.0
                mcastaddr: 226.94.1.1
                mcastport: 5405
        }
}
```

暗号化無し設定

TOKEN受信のタイムアウト値
→ 4秒応答がなければフェイルオーバ処理を行う

リングナンバー
バインドするネットワークアドレス
マルチキャスト通信アドレス
受信ポート番号

logging

Pacemakerのログ出力に関する情報を指定します。

```
logging {
    fileline: on
    to_syslog: yes
    syslog_facility: local1
    syslog_priority: info
    debug: off
    timestamp: on
}
```

syslogを使用し、syslogのファシリティを「local1」に指定

/etc/syslog.conf

- /etc/corosync.conf で指定したファシリティの設定が必要

/var/log/ha-log にログを出力するように設定します。
また、同内容のログを /var/log/messages に2重出力しないように、「local1.none」の追記も行います。

```
*.info;mail.none;authpriv.none;cron.none;local1.none      /var/log/messages
                :
            （省略）
                :
local1.*                                                   /var/log/ha-log
```

これでとりあえずは Pacemakerが起動します！

```
# service corosync start
Starting Corosync Cluster Engine (corosync):       [  OK  ]
```

起動はクラスタ制御部である
corosyncを各サーバで起動します

起動状態の確認

Pacemakerのコマンド /usr/sbin/crm_mon を利用して起動状態が確認できます。

```
============
Last updated: Tue Jun 15 06:31:16 2010
Stack: openais
Current DC: pm01 - partition with quorum
Version: 1.0.8-9881a7350d6182bae9e8e557cf20a3cc5dac3ee7
2 Nodes configured, 2 expected votes
0 Resources configured.
============

Online: [ pm02 pm01 ]
```

クラスタに組み込まれている
ノード名が表示されます

しかしこれだけでは、
リソース制御部の設定が無いので
なーんにも
リソースは
起動していません…

リソース制御部の設定

リソース
制御部

- リソース制御部には次のような設定が必要です。
  - どのようなリソースをどのように扱うか
    → Apache、PostgreSQLなど、どのリソース（アプリケーション）を起動するか？
  - 起動、監視、停止時に関連する時間
    → リソースの監視（monitor）間隔は何秒にするか？？
  - リソースの配置 などを指定
    → リソースをどのノードで起動するか？？？
- 設定方法には主に2通りあります。
  - cib.xml にXML形式で設定を記述（従来のHeartbeat 2.x での方法）
  - crmコマンドで設定（Pacemakerからの新機能）

cib.xml

- /var/lib/heartbeat/crm/cib.xml

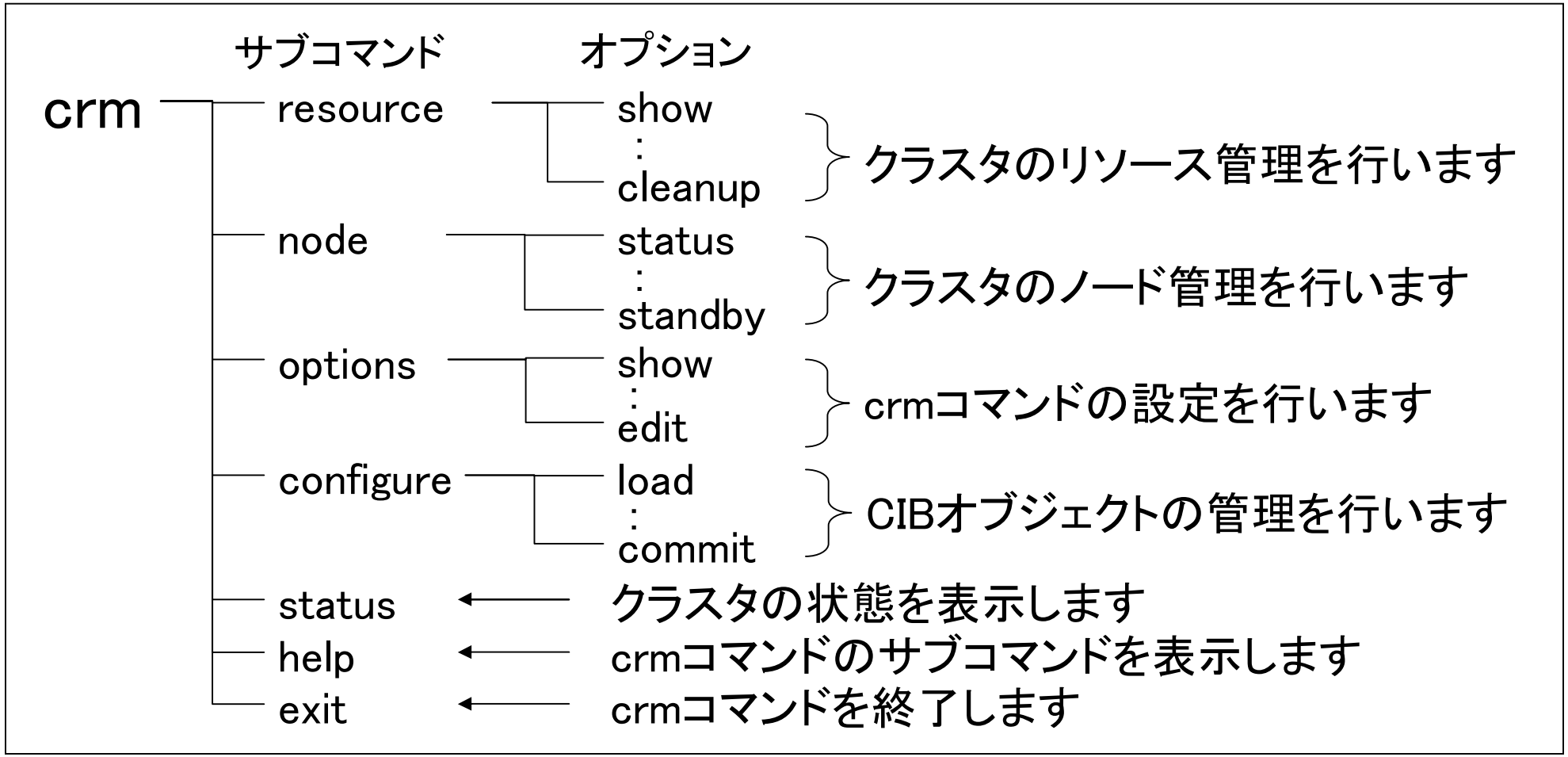
主に、リソースの定義を設定するXMLファイルです。

```
 (..略..)
<resources>
   <primitive class="ocf" id="prmIpWWW" provider="heartbeat" type="IPaddr">
     <instance_attributes id="prmIpWWW-instance_attributes">
       <nvpair id="prmIpWWW-instance_attributes-ip" name="ip" value="192.168.0.108"/>
       <nvpair id="prmIpWWW-instance_attributes-nic" name="nic" value="eth1"/>
       <nvpair id="prmIpWWW-instance_attributes-cidr_netmask" name="cidr_netmask"
value="255.255.255.0"/>
     </instance_attributes>
     <operations>
       <op id="prmIpWWW-start-0s" interval="0s" name="start" on-fail="restart" timeout="60s"/>
       <op id="prmIpWWW-monitor-10s" interval="10s" name="monitor" on-fail="restart"
timeout="60s"/>
       <op id="prmIpWWW-stop-0s" interval="0s" name="stop" on-fail="fence" timeout="60s"/>
     </operations>
   </primitive>
</resources>
 (..略..)
```

XMLの記法を知る
必要があり難しい…

crmコマンド

crmコマンドは、クラスタ状態を管理するためのコマンドラインインターフェイスです。



crmコマンド実行例

「IPaddr」リソースエージェントを使用して仮想IPを設定をするcrmコマンド例です

```
# crm
crm(live)# configure
crm(live)configure# primitive prmIpWWW ocf:heartbeat:IPaddr ¥
 params ip="192.168.0.108" nic="eth1“ ¥
 cidr_netmask="255.255.255.0" ¥
 op start interval="0s“ timeout="60s" on-fail="restart" ¥
 op monitor interval="10s“ timeout="60s" on-fail="restart" ¥
 op stop interval="0s" timeout="60s" on-fail="fence“
                :
crm(live)configure# commit
```

コミットされると、cib.xmlに反映されてリソースが起動されます。
→ つまりリソース制御部の根っこは cib.xmlなのです。

やっぱり設定方法は
わかりにくいですよね...

crmは恐くない！

- 複雑なリソース制御の設定もcrmファイル編集ツール pm-crmgenで解決！

pm-crmgenを使用すれば、テンプレートExcelファイルから簡単にリソース制御部を設定する事が可能です。

Linux-HA Japanプロジェクトで
crmファイル編集ツールを開発中！

開発版は、Linux-HA Japanプロジェクトのリポジトリよりダウンロード可能です。
http://hg.sourceforge.jp/view/linux-ha/pm-crmgen/

crmファイル編集ツールで簡単設定！

※ 6/26 時点での開発版での状況です

① テンプレートExcelファイルにリソース定義を記載

赤枠線の中に値を記入します。
仮想IPをActiveノードに付与する場合の例です。

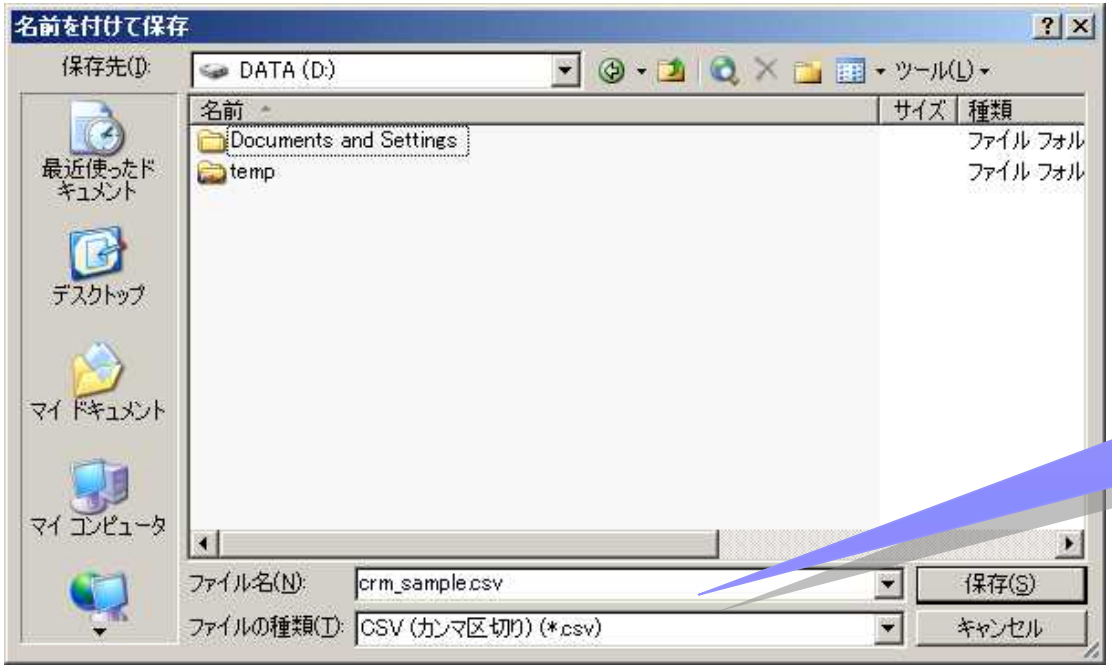
39	#表 2-2 クラスタ設定 … Primitiveリソース (id=prmIpWWW)					
40	PRIMITIVE					
41	P	id	class	provider	type	
42	#	リソースID	class	provider	type	
43		prmIpWWW	ocf	heartbeat	IPaddr	
44	A	type	name	value		
45	#	パラメータ種別	項目	設定内容		
46		params	ip	192.168.0.108		
47			nic	eth1		
48			cidr_netmask	255.255.255.0		
49	O	type	timeout	interval	on-fail	
50	#	オペレーション	タイムアウト値	監視間隔	on_fail(障害時の動作)	備考
51		start	60s	0s	restart	
52		monitor	60s	10s	restart	
53		stop	60s	0s		

「IPaddr」のリソースエージェントを使用

付与する仮想IPのIPアドレス等を入力

監視間隔などを入力

② CSV形式でファイルを保存



「crm_sample.csv」としてCSV形式で保存

③ CSVファイルをノードへ転送

CSVファイル保存後、SCPコマンド等でACT系ノードへ転送
→ ACT系、SBY系どちらか片方のノードに転送すればOK！

④ pm-crmgenをインストール

rpmパッケージ名は予定名です。
プログラム自体は pythonで作成されています。

```
# rpm –ivh pm-crmgen-XXX.noarch.rpm
```

⑤ pm_crmgenコマンドでcrmファイルを生成

```
# pm_crmgen –o crm_sample.crm  crm_sample.csv
```

③で転送したCSVファイル

生成するcrmファイル名

出来上がった crmファイル例

```
(..略..)

### Primitive Configuration ###
primitive prmIpWWW ocf:heartbeat:IPaddr ¥
        params ¥
                ip="192.168.0.108" ¥
                nic="eth1" ¥
                cidr_netmask="255.255.255.0" ¥
        op start interval="0s" timeout="60s" on-fail="restart" ¥
        op monitor interval="10s" timeout="60s" on-fail="restart" ¥
        op stop interval="0s" timeout="60s" on-fail="fence“

(..略..)
```

Excelファイルで記述した
仮想IPを設定する
crmサブコマンドが
ファイルに記述されます

⑥ crmコマンドを実行してリソース設定を反映

```
# crm
crm(live)# configure
crm(live)configure# load update crm_sample.crm
crm(live)configure# commit
```

⑤で生成したcrmファイル名

commitで設定が反映される

または以下のようにcrmコマンド一発で反映も可能です。（即コミットされますが…）

```
# crm configure load update crm_sample.crm
```

これでリソースも起動しました！

/usr/sbin/crm_mon を利用して起動したリソースが確認できます。

```
============
Last updated: Tue Jun 15 21:54:14 2010
Stack: openais
Current DC: pm01 - partition with quorum
Version: 1.0.8-9881a7350d6182bae9e8e557cf20a3cc5dac3ee7
2 Nodes configured, 2 expected votes
3 Resources configured.
============

Online: [ pm02 pm01 ]

 Resource Group: grpStonithN1
     prmStonithN1       (stonith:external/riloe):       Started pm02
 Resource Group: grpStonithN2
     prmStonithN2       (stonith:external/riloe):       Started pm01
 Resource Group: grpWWW
     prmIpWWW    (ocf::heartbeat:IPaddr):        Started pm01
```

ノード1で
仮想IPが
付与されました

④

Linux-HA Japan プロジェクトについて

Linux-HA Japan プロジェクトの経緯

『Heartbeat(ハートビート)』の日本における更なる普及展開を目的として、2007年10月5日「Linux-HA (Heartbeat) 日本語サイト」を設立しました。

その後、日本でのLinux-HAコミュニティ活動として、Heartbeat-2.x のrpmバイナリと、Heartbeat機能追加パッケージを提供しています。

そしてこれからは
Linux-HA Japanプロジェクトから
Pacemaker関連の
情報やパッケージも提供します！

Linux-HA JapanプロジェクトURL

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

Pacemaker関連情報の
公開用として SourceForge.jp に
新しいウェブサイトが 6/25に
オープンしました。

これから随時情報を更新していきます！

Linux-HA Japan開発者向けサイト
Heartbeat-2.x 用の情報も公開中

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

RHEL/CentOS用 Heartbeat-2.x rpmバイナリの提供や、機能追加パッケージ類を、GPLライセンスにて公開しています。
共有ディスク排他制御機能（SFEX）や、ディスク監視デーモン 等が提供されています。

Pacemaker関連の開発ソースコードもこのサイトから参照可能です。

linux-ha.org
本家Linux-HAサイト

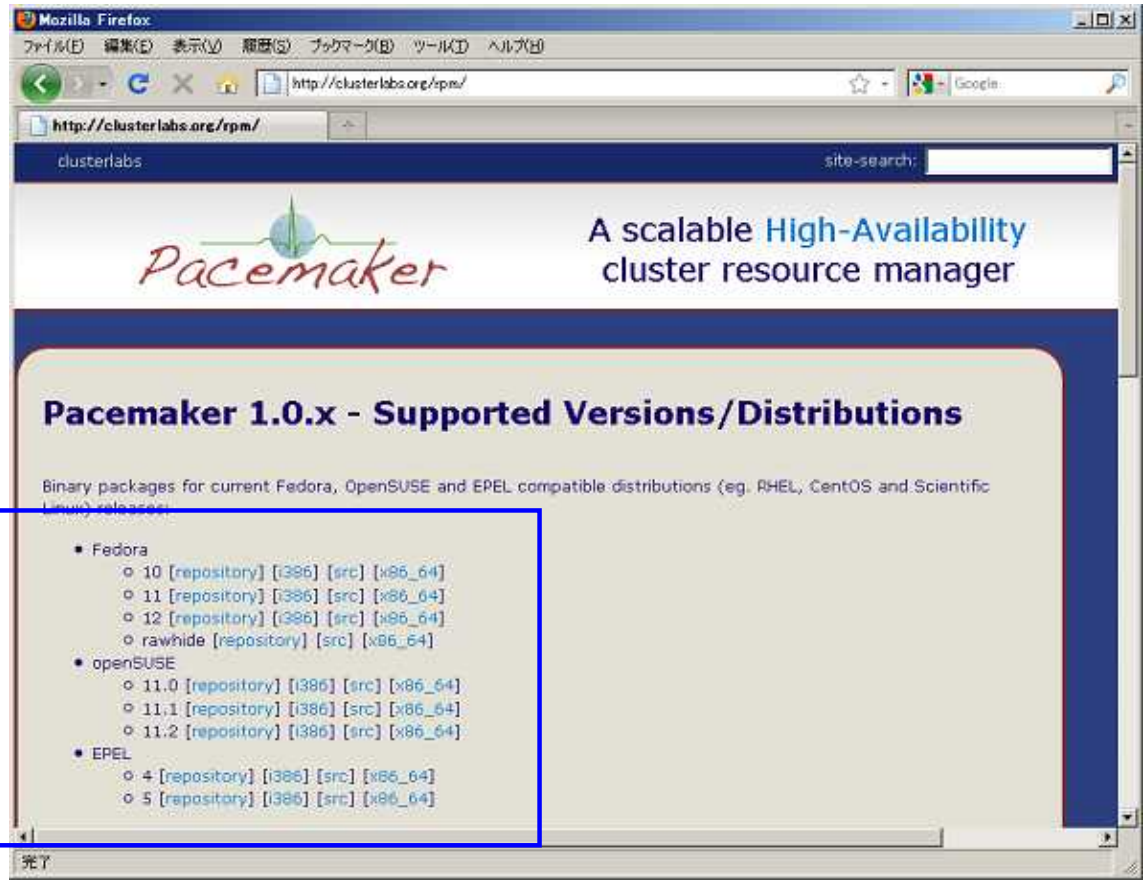
http://www.linux-ha.org/wiki/Main_Page/ja

Linux-HA Japanプロジェクトのサイトとは、相互リンクを張っていきます

clusterlabs.org
本家Pacemakerサイト

http://clusterlabs.org/

Fedora, openSUSE, EPEL(RHEL/CentOS)のrpmがダウンロード可能です。



Pacemakerロゴ

Linux-HA Japan プロジェクトでは、
Pacemakerのロゴを作成しました。

Pacemaker 動画CM

Linux-HA Japan プロジェクトでは、
Pacemakerの動画CMも作成しちゃいました。

Linux-HA Japanメーリングリスト

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
その他DRBDなど、HAクラスタに関連する話題は全て歓迎します！

- ML登録用URL
  http://lists.sourceforge.jp/mailman/listinfo/linux-ha-japan
- MLアドレス
  linux-ha-japan@lists.sourceforge.jp

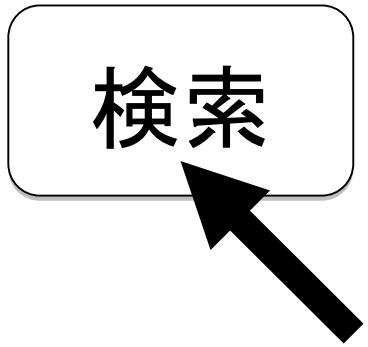
さいごに…

Linux-HA Japanプロジェクトでは
Pacemakerの
様々な設定例や
追加パッケージなどの
コンテンツを載せていき

Pacemakerの
普及展開を推し進めます

ぜひ
メーリングリストに登録して
HAクラスタの
活発な意見交換を
交わしましょう！

Linux-HA Japan



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

⑤

参考情報

STONITHとは？

STONITHとは「Shoot The Other Node In The Head」の略であり、不具合のあるノードを強制的にそのノードをダウンさせる機能です。
コントロールが利かないノードをPacemakerからSTONITHプラグインを通じて「強制的に離脱」させることにより、リソースの２重起動を防ぎます。

確実にノードを強制離脱させるために、サービスを提供するOSとは別経路の「HW制御ボード」を用いた電源操作を推奨します。

HW制御ボードの例

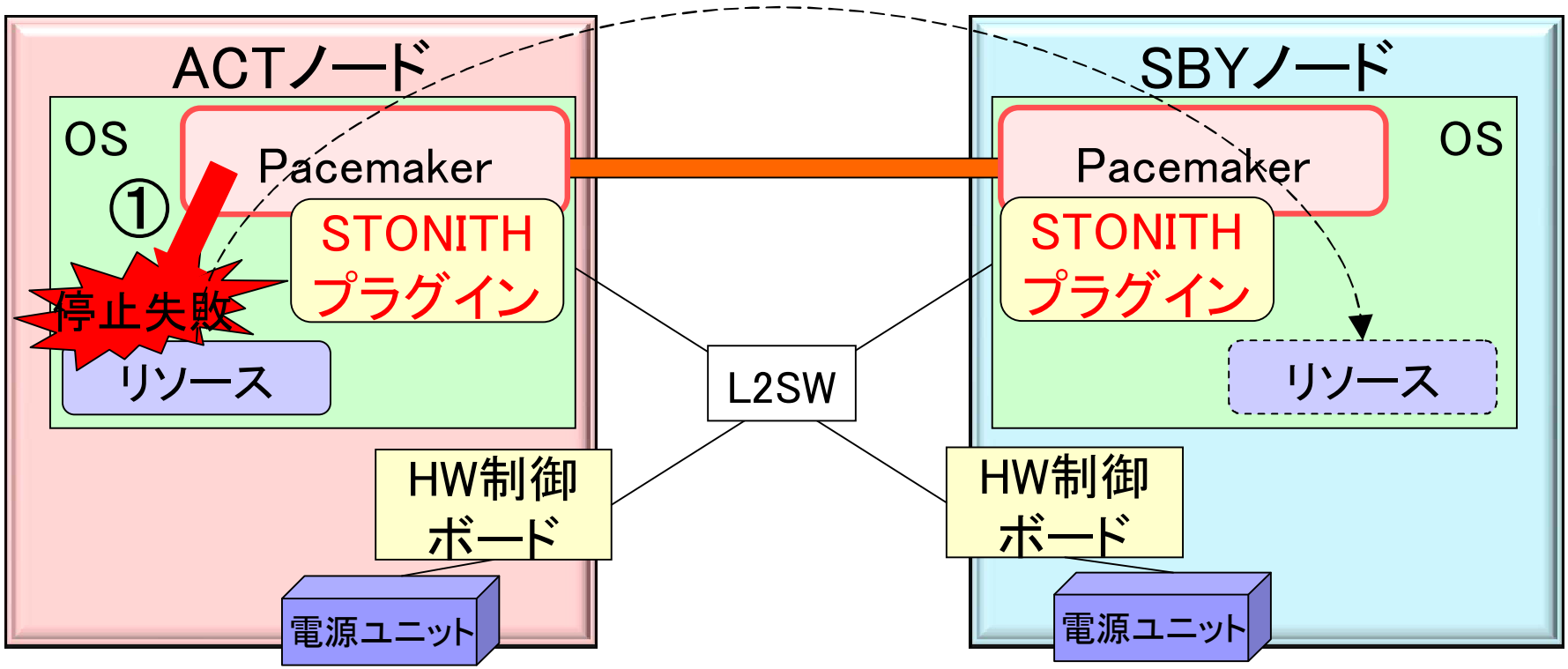
- IBM社
  - リモート管理アダプターⅡ SlimLine【RSAⅡ】
    - System x® 3650（旧モデル）等にオプションで搭載が可能
  - Integrated Management Module【IMM】
    - System x® 3650 M2 等に標準搭載

- HP社
  - Integrated Lights-Out 2【iLO2】
    - ProLiant DL380 G6 等に標準搭載
  - Integrated Lights-Out 3【iLO3】
    - ProLiant DL380 G7（新モデル）等に標準搭載

▲ iLO2

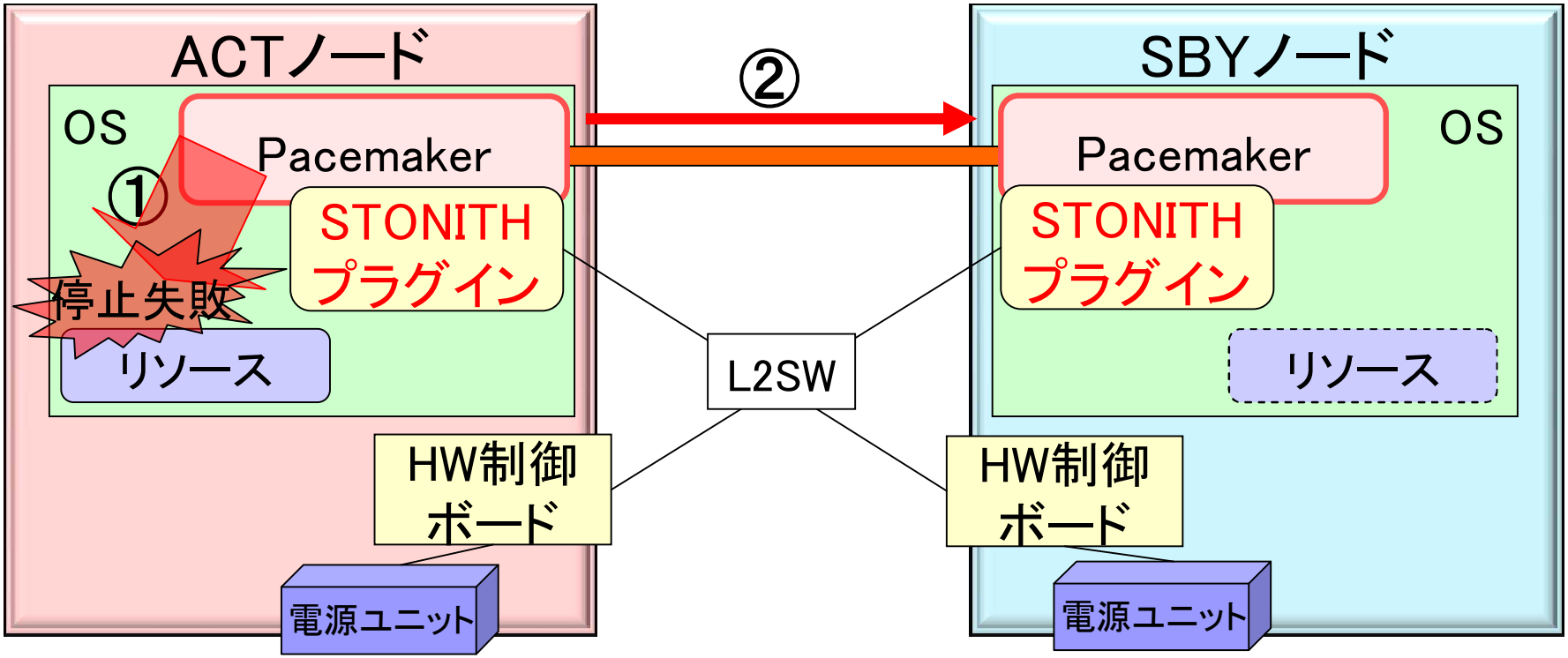
フェイルオーバ時にSTONITH機能が発動されるまでの流れ①

① フェイルオーバ時にACTノードのリソース停止処理がNGとなる事象が発生



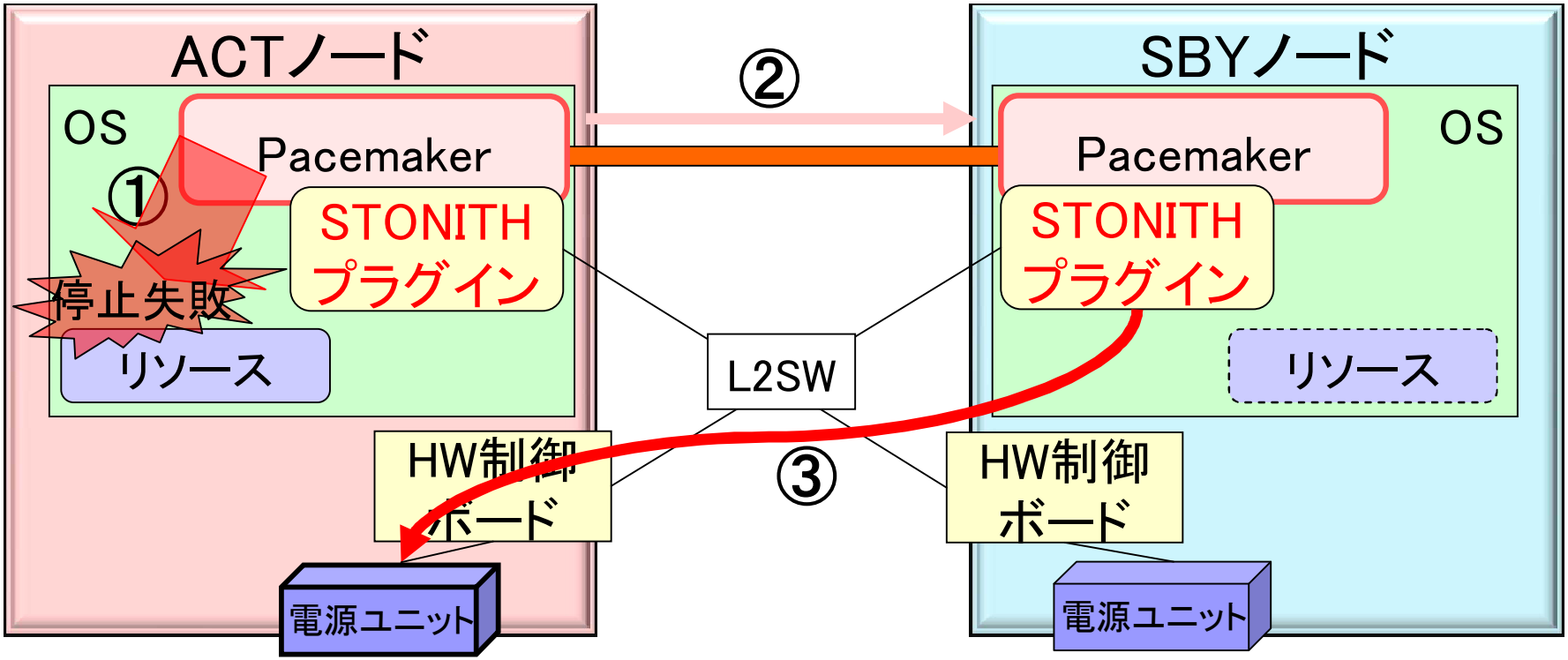
フェイルオーバ時にSTONITH機能が発動されるまでの流れ②

② 検知したPacemakerが、SBYノードのPacemakerに状態を伝達



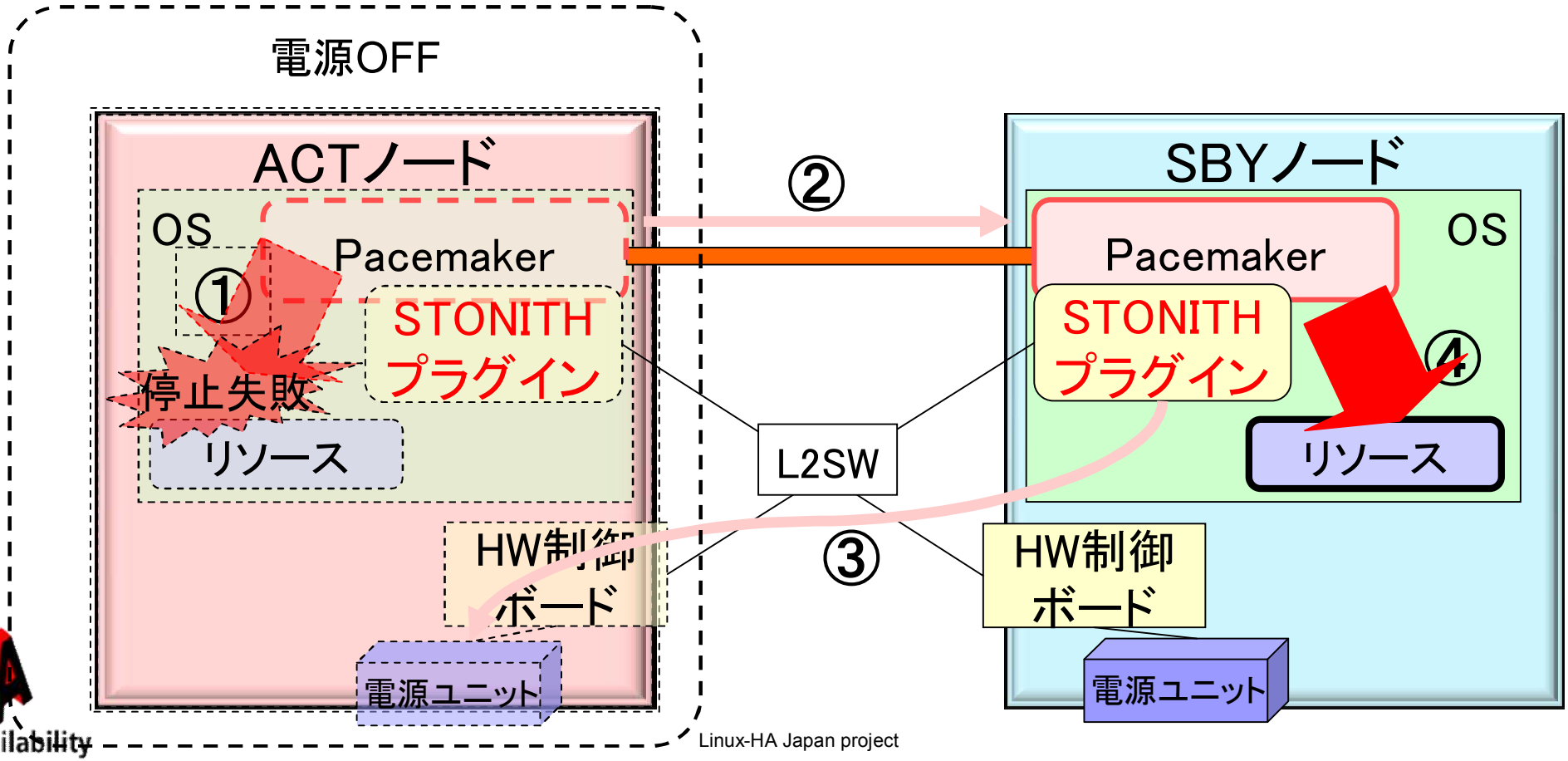
フェイルオーバ時にSTONITH機能が発動されるまでの流れ③

③ SBYノードのPacemakerがSTONITHプラグインを通じ、故障発生サーバのHW制御ボードを操作して強制電源断



フェイルオーバ時にSTONITH機能が発動されるまでの流れ④

④ 強制電源断成功後、リソースが起動



STONITHプラグイン

Pacemakerには、様々なHW制御ボードに対応したSTONITHプラグインが標準装備されています。
プラグインは、シェルスクリプト、Perl、Python等で作成されています。

- HP iLO2用プラグイン

  (/usr/lib64/stonith/plugins/external/riloe)
- IBM RSAⅡ用プラグイン

  (/usr/lib64/stonith/plugins/external/ibmrsa-telnet)
- IPMI用プラグイン

  (/usr/lib64/stonith/plugins/external/ipmi)

IPMIとは…
サーバー・プラットフォームの状態（温度、電圧、ファン、バスなど）監視や復旧、リモート制御を行うための標準インターフェイス仕様。
IBM IMM、HP iLO3等で使用が可能。