

# Pacemakerで簡単・手軽に クラスタリングしてみよう！

2010年7月10日 OSC2010 Kansai@Kyoto

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

①

Pacemakerって何？

簡単に言うと・・・

# Pacemakerとは？



オープンソースの  
HA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。

決して次の物ではありません・・・

# 心臓ペースメーカー

※ ウィキペディアより

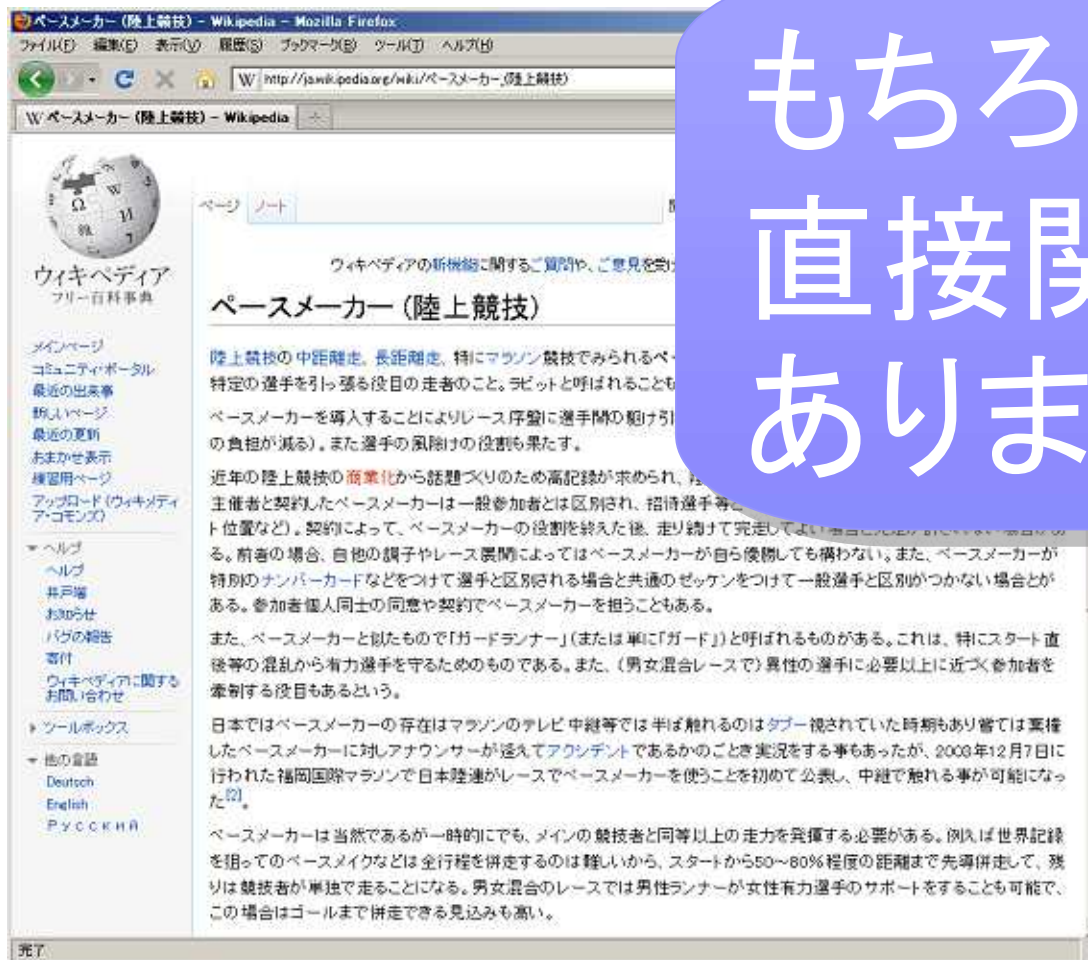


医療機器とは  
関連ありません...



# ペースメーカー(陸上競技)

※ ウィキペディアより



もちろんマラソンも  
直接関係  
ありません...







# Pacemaker は

# HAクラスタソフトウェアです！

冗談はさておき  
ここで質問ですが、

そのHAクラスタソフトである



Pacemaker は

知っていましたか？

同じくHAクラスタソフトウェアである

Heartbeatバージョン1 は


知っていましたか？

さらに  
同じくHAクラスタソフトウェアである  
Heartbeatバージョン2 は  
知っていましたか？

「Pacemaker」と「Heartbeat」の  
詳しい関係は後ほどお話します。

ここで、そもそもですが・・・

# HAクラスタって？

 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、  
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。

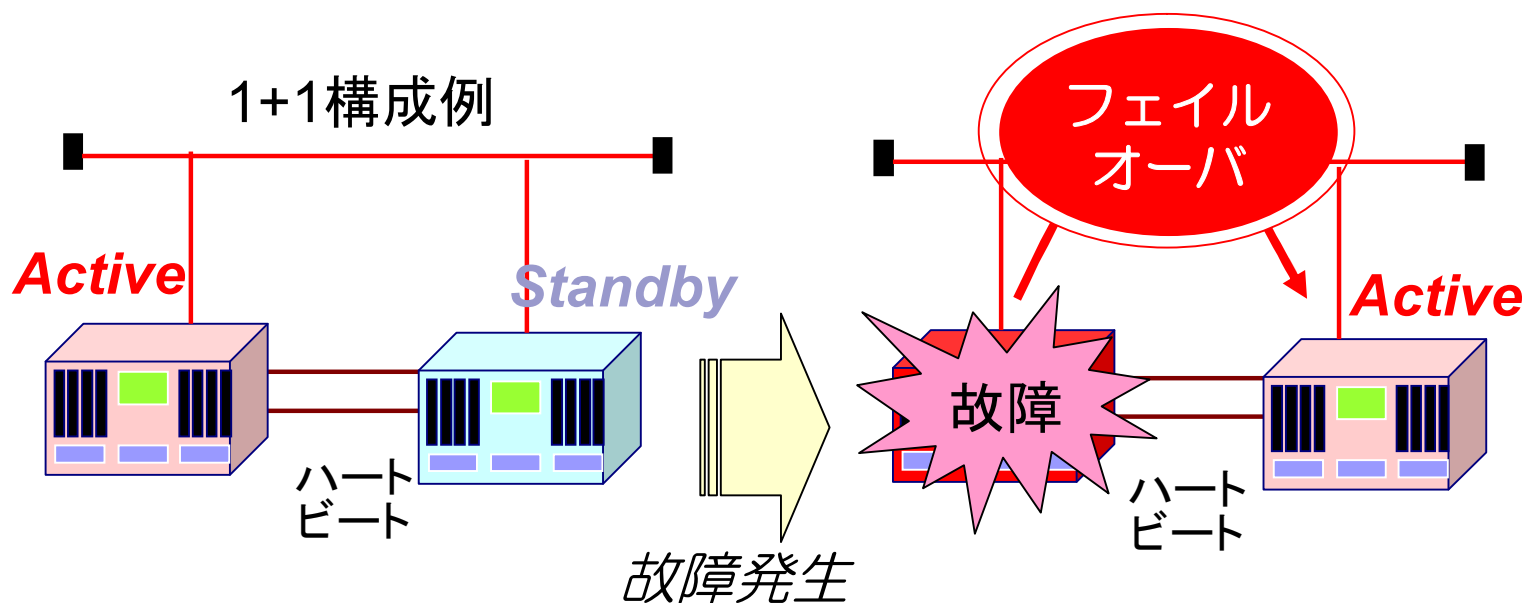
サービスを引き継ぐ予備機を  
用意する必要性があるので、  
HAクラスタには  
最低でも2台のノードが必要です。

ここまで文字ばかりで  
飽きちゃいそうですから、  
図で説明します。

# PacemakerによるHAクラスタの基本構成

## Active/Standby (1+1) 構成

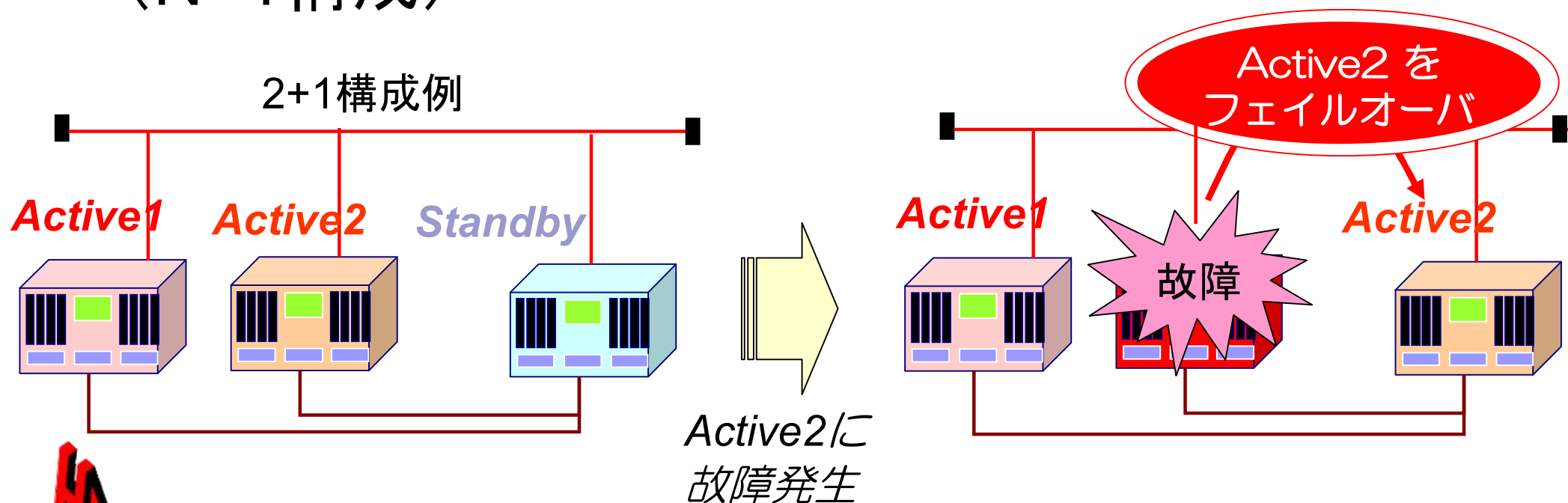
- Pacemakerは、故障発生を検知した場合、待機系へフェイルオーバさせることによってサービスの継続が可能になります。



# Pacemakerでは複数台構成も可能です

※ Heartbeatバージョン1では実現できませんでした

- Pacemakerでは、2台など複数台の運用系ノードに対し、待機系ノードを1台にする事も可能です。  
(N+1構成)



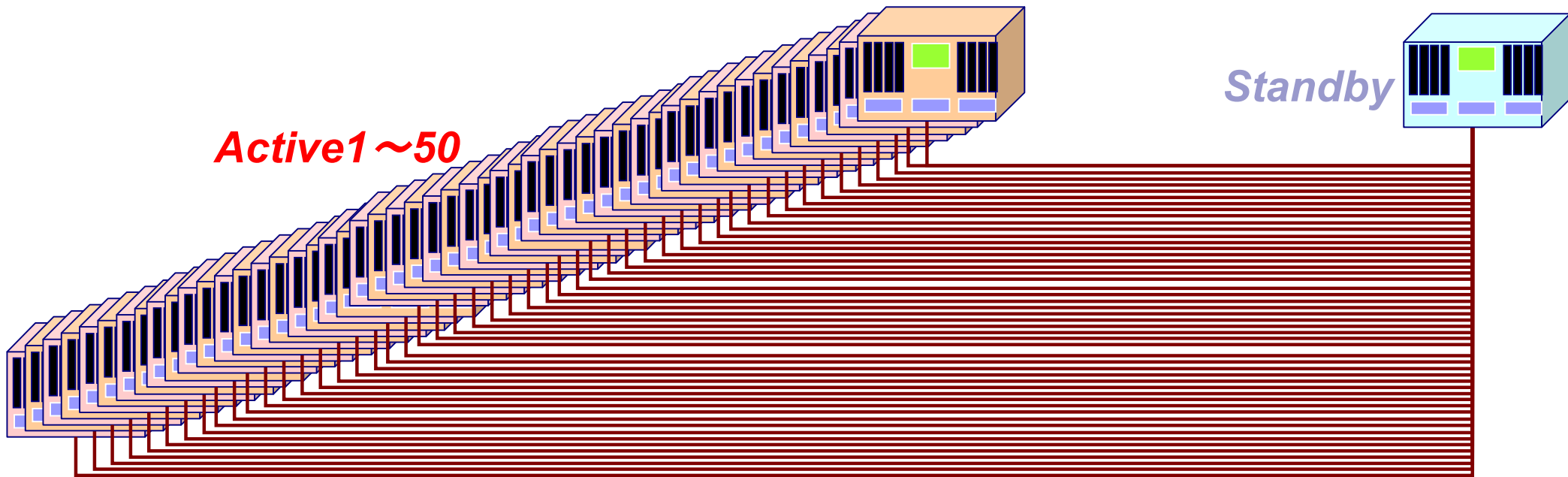
複数台 (N+1) 構成が  
可能ということは、  
こんな構成も可能か??

# 50+1構成は...

適切なHAクラスタ設計は  
十分検討しましょう！！

- これはムチャ構成です..(泣)

実現しても、50台いっぺんにフェイルオーバーしてきたら  
大変なことになります。



ここからPacemakerの説明は、  
Active / Standby の  
単純構成を例としてお話しします。

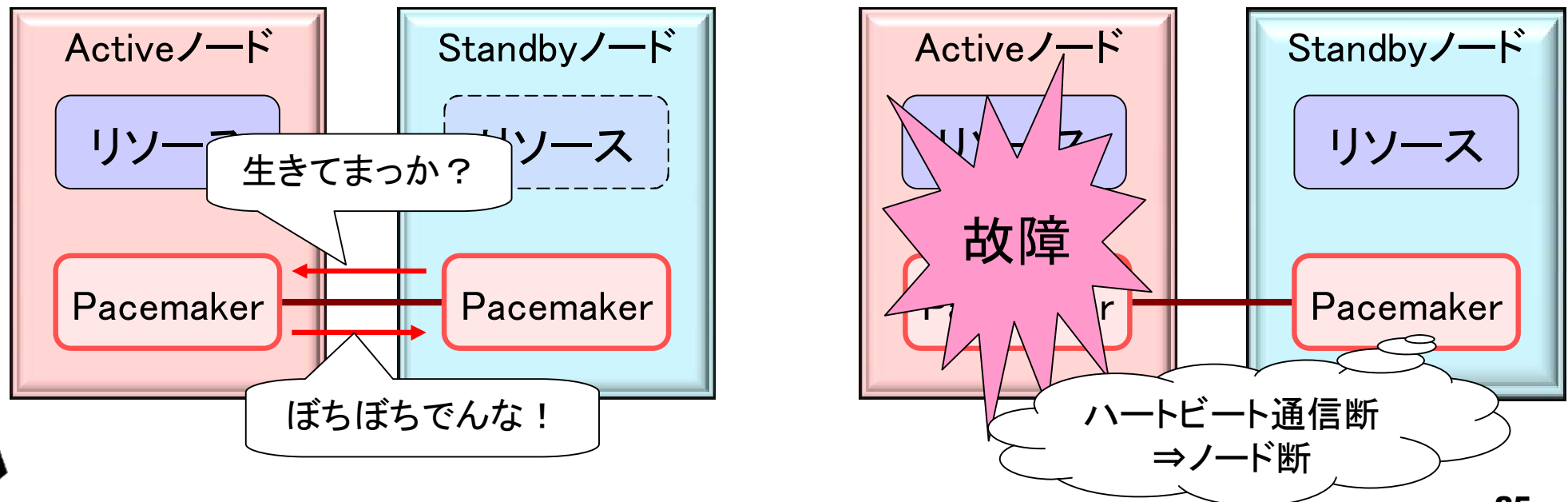


# 基本的動作：ノード監視

クラスタ  
制御部

## □ 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、フェイルオーバなどのクラスタ制御の処理を行います。

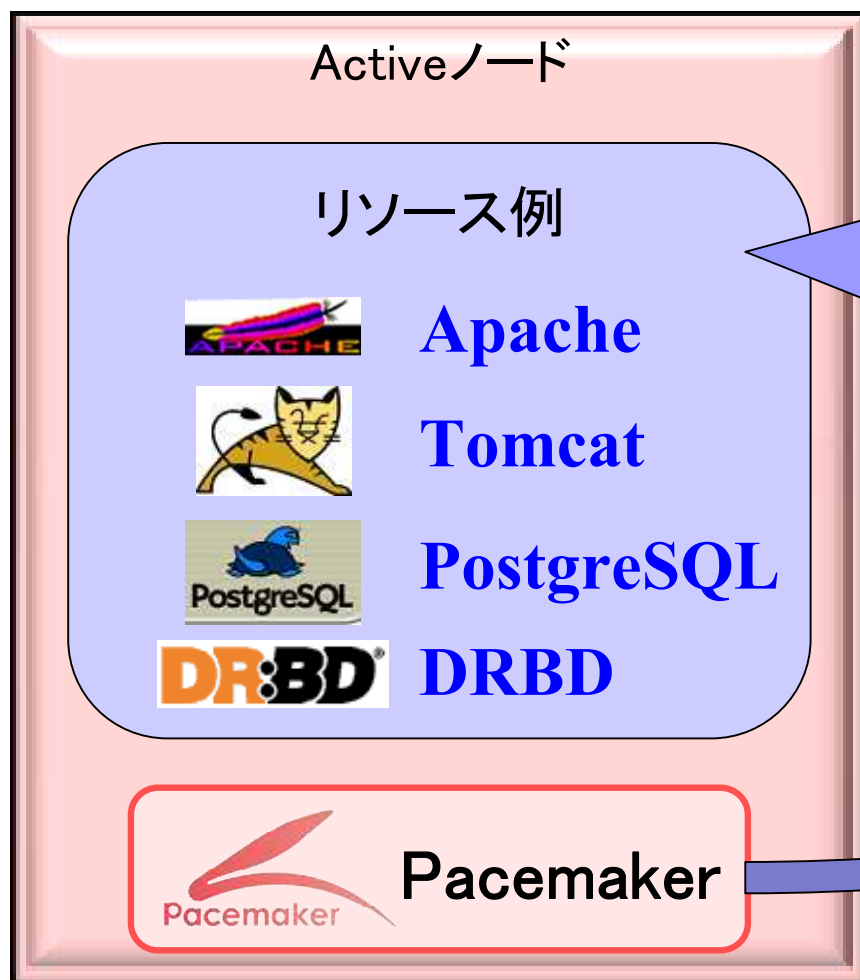


# 「リソース」とは？

Pacemakerではよく出てくる言葉なのでおぼえてください！

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。

# 例えばこんなのが Pacemaker から見た「リソース」になります



Pacemaker から見ると、PostgreSQL などのアプリケーションは、「リソース」となります。

# 「リソースエージェント」とは？

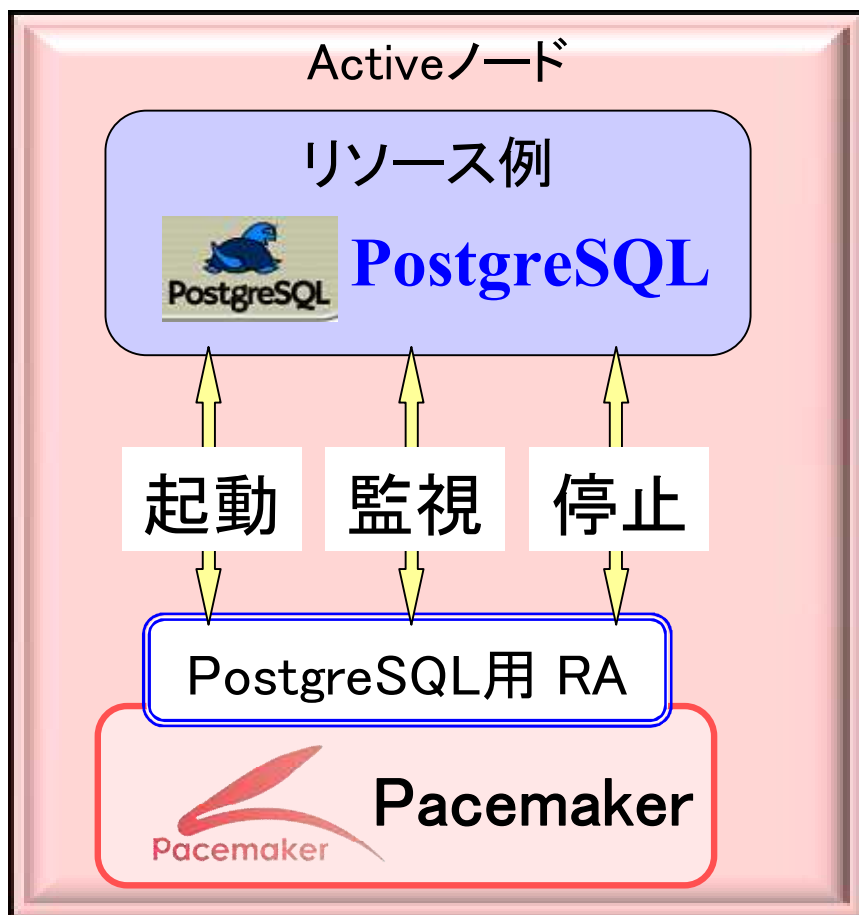
これまたPacemakerではよく出てくる言葉なのでおぼえてください！

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。



# 「リソース」と「リソースエージェント」は こんな関係 になります



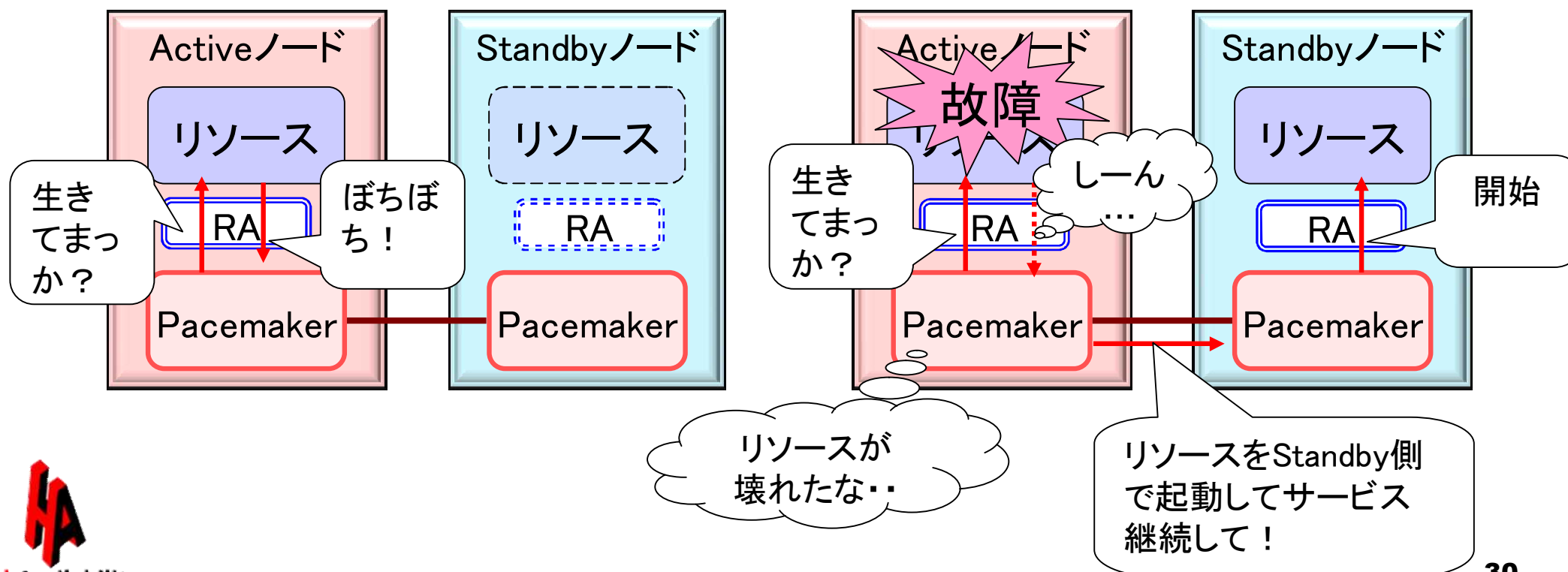
Pacemaker は、PostgreSQL などのリソースを、リソース エージェントを介して起動、停止、監視等の制御をおこなうことができます。

※ Heartbeatバージョン1ではリソース監視の機能はありませんでした

# 基本的動作：リソース制御

リソース  
制御部

- 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リソースエージェント(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標準で多数用意されています。

### 標準リソースエージェントの一例

MySQLや、Tomcat用のリソースエージェントなどもありますよ！

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

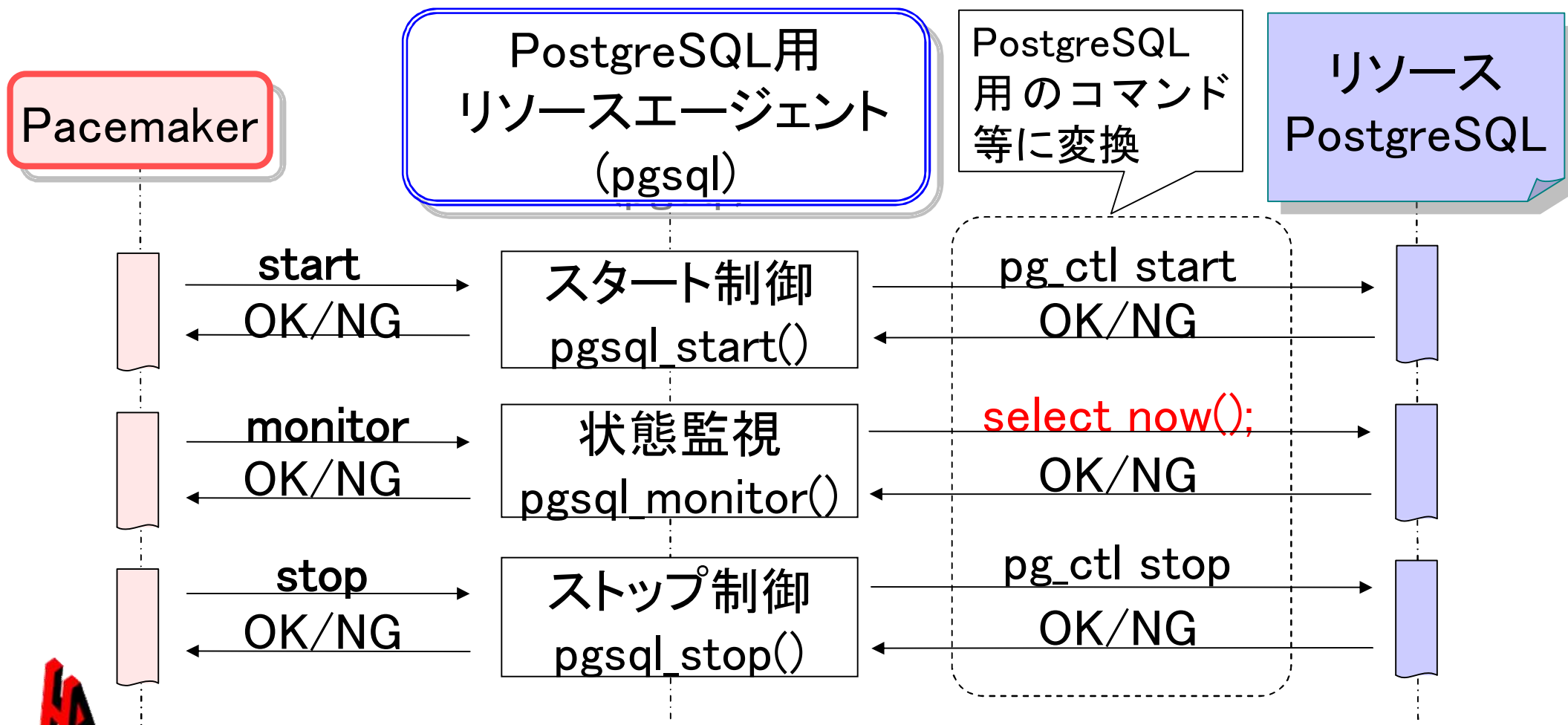
```
#!/bin/sh
```

(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



# リソースエージェントは自分でも作れます！

```
#!/bin/sh
. ${OCF_ROOT}/resource.d/heartbeat/.ocf-shellfuncs
```

```
start処理() {
}
stop処理() {
}
monitor処理 {
}
meta-data処理(){
}
validate-all処理(){
}
```

```
case $1 in
start)    start処理();;
stop)    stop処理();;
monitor)  monitor処理();;
...
esac
```

通常のシェルスクリプトの記述方法ですが、いくつか必須のパラメータ呼び出しに対する処理を行う必要があります。

リソース開始・監視・停止の処理

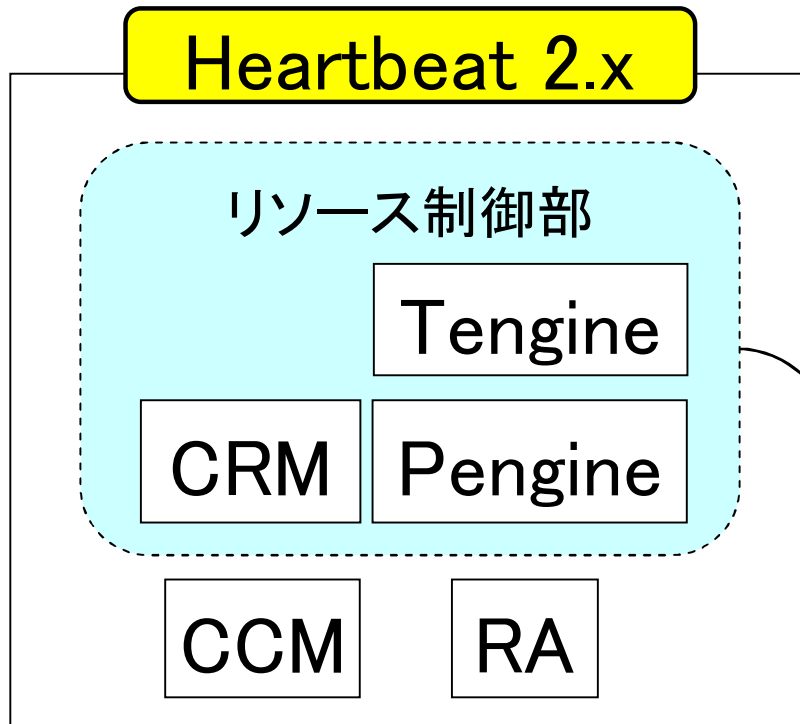
RA処理の振り分け

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バージョン2系のリソース制御部が Pacemakerとして切り出されてリリースされました。

Pacemaker

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

切り出された！？

ということは・・・

Pacemaker 単独では  
HAクラスタソフトとして  
動作しない？

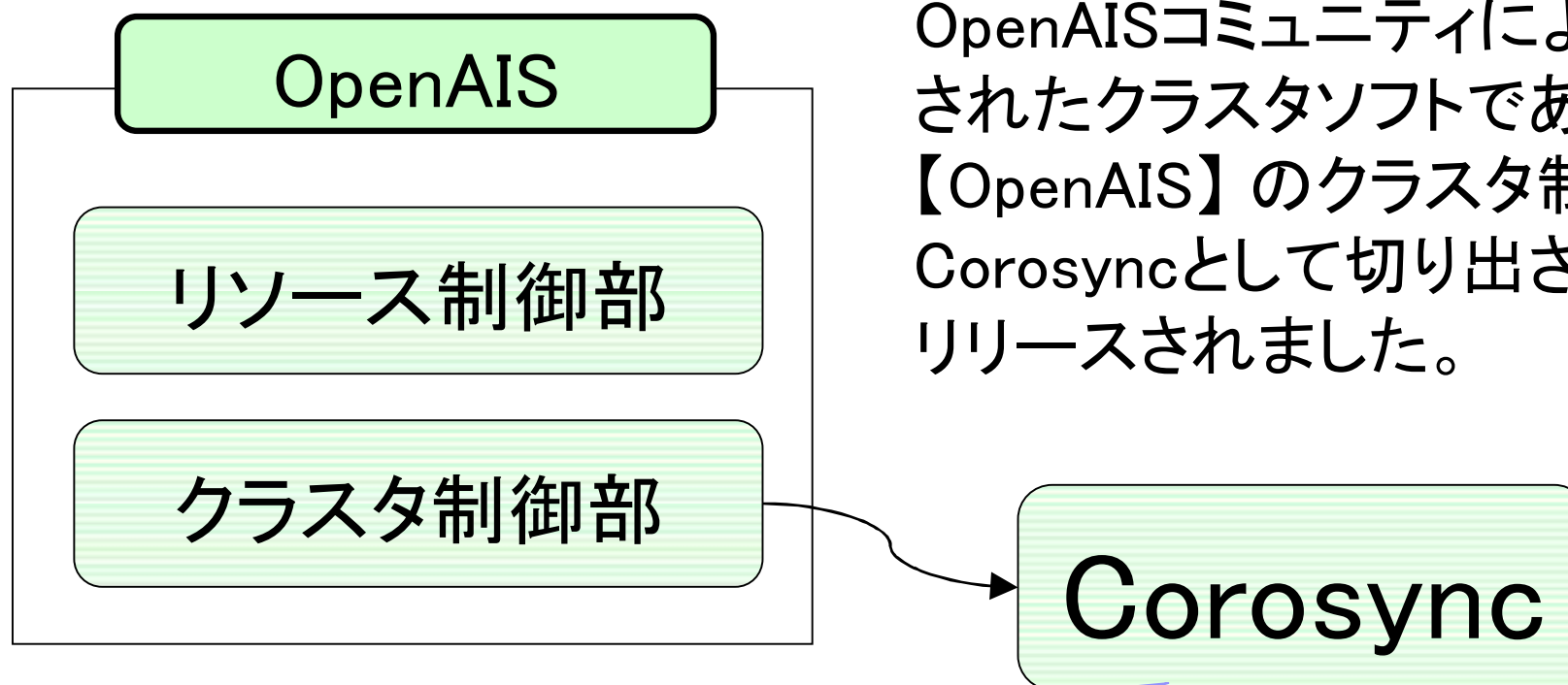


そのとおりです..

Pacemaker は  
他のクラスタ制御部の  
アプリケーションと組み合わせて  
使用しなければなりません..

前向きなことを言うと  
クラスタ制御部の  
選択肢が広がったのです！

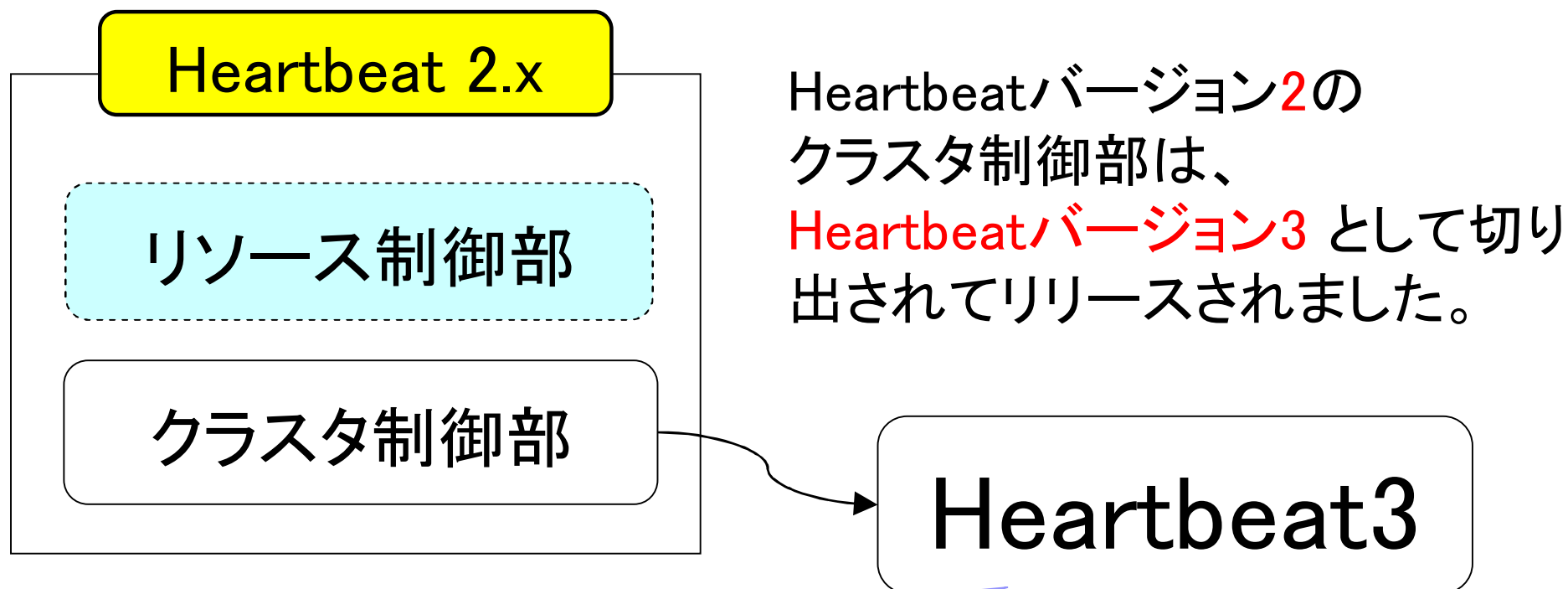
# Corosync



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として切り出されてリリースされました。

つまり Corosyncも単独ではHAクラスタとしては動作しない！？

# Heartbeat3

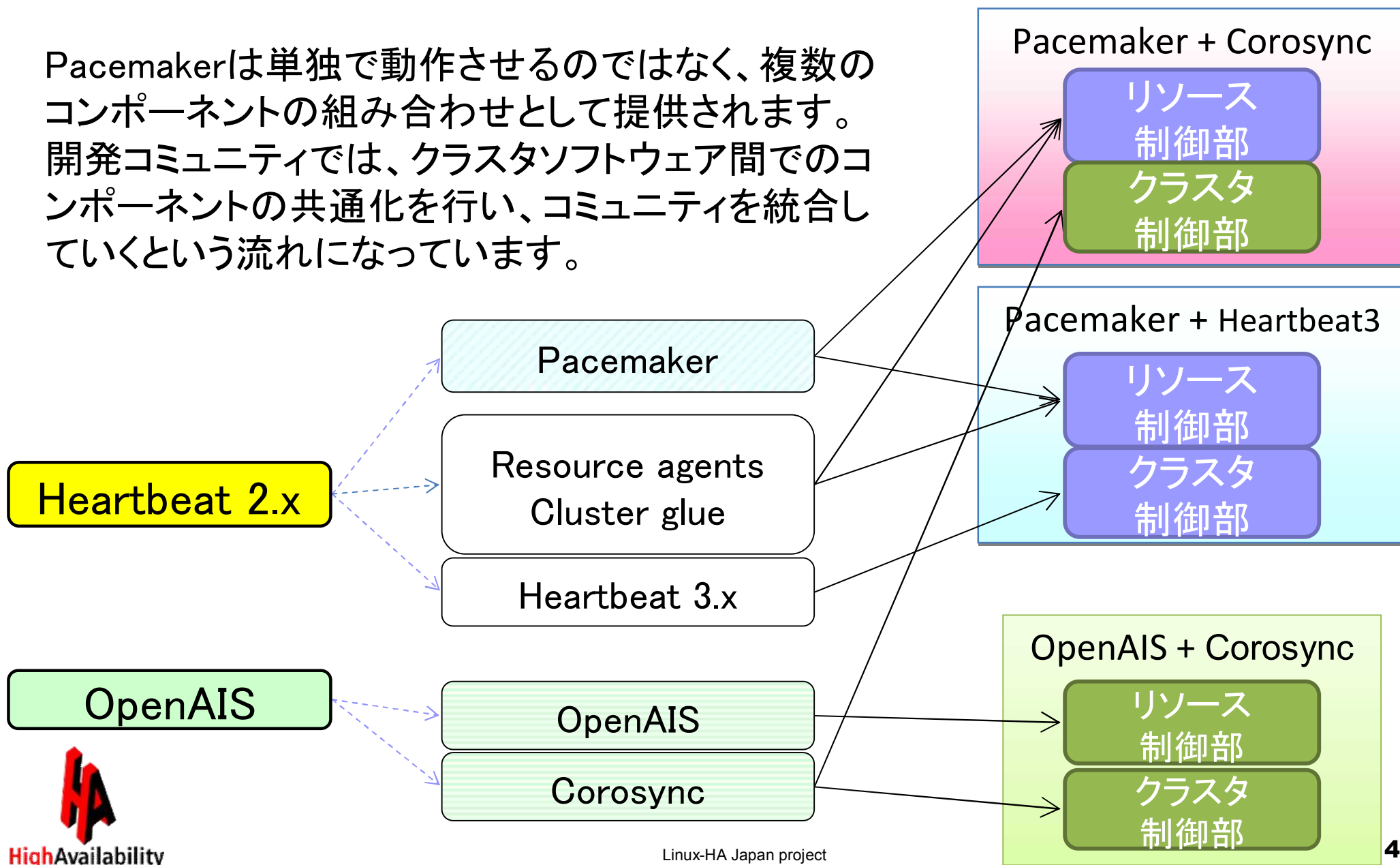


なので“2”から“3”と数字が上がったのに、機能的にはデグレ！？

Pacemaker は  
この「Corosync」と「Heartbeat3」  
のクラスター制御部が  
選択可能です。

# HAクラスタのリリース形態

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。



では、プロダクト名は  
「Pacemaker ぶらす ……」  
って呼ぶの??



それは呼びにくいですね・・・

なので、

# Pacemaker + Corosync も



# Pacemaker + Heartbeat3 も

Pacemaker + Heartbeat3

リソース  
制御部

クラスタ  
制御部

Linux-HA Japan プロジェクトでは  
プロダクト名を



**Pacemaker**

としています。

この2つのリリース形態を「Pacemaker」としています

Heartbeat 2.x

OpenAIS

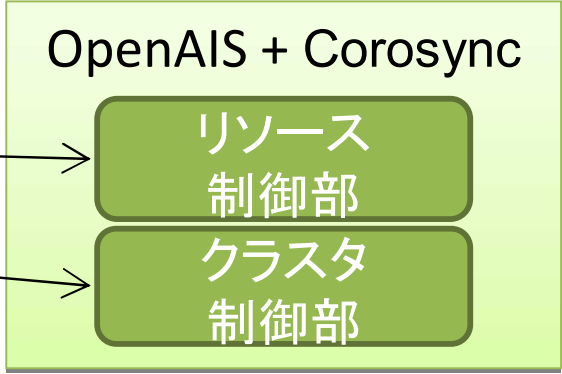
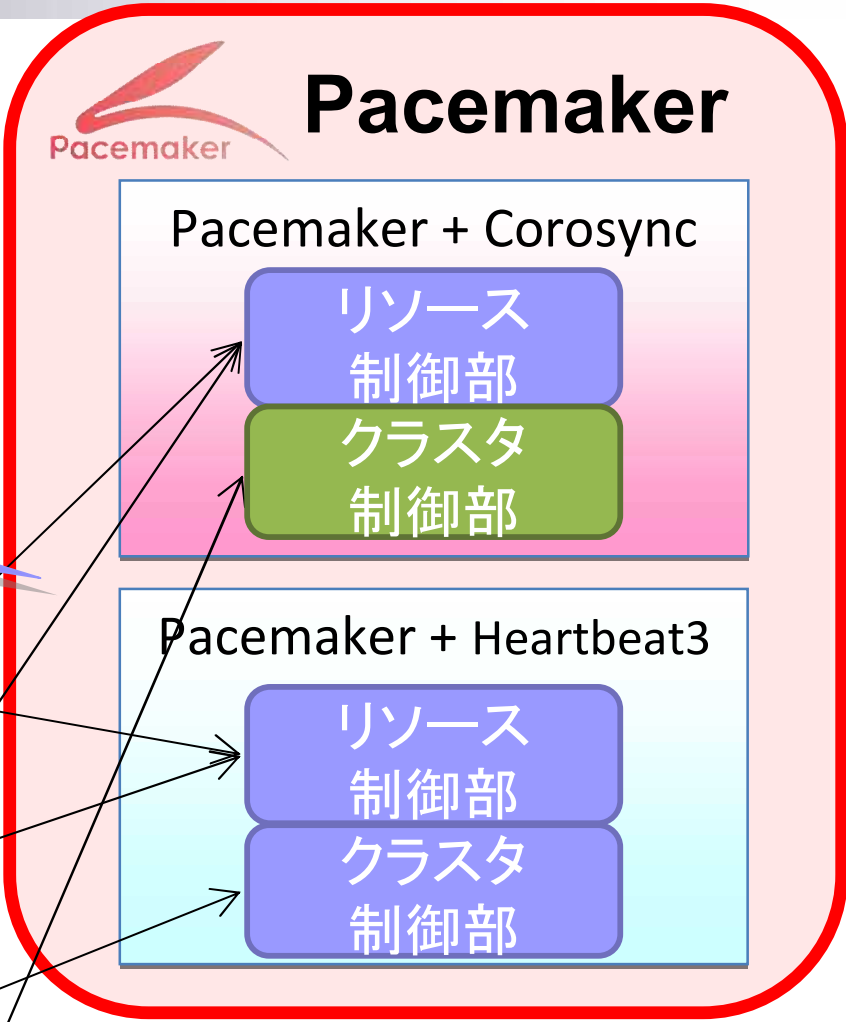
Pacemaker

Resource agents  
Cluster glue

Heartbeat 3.x

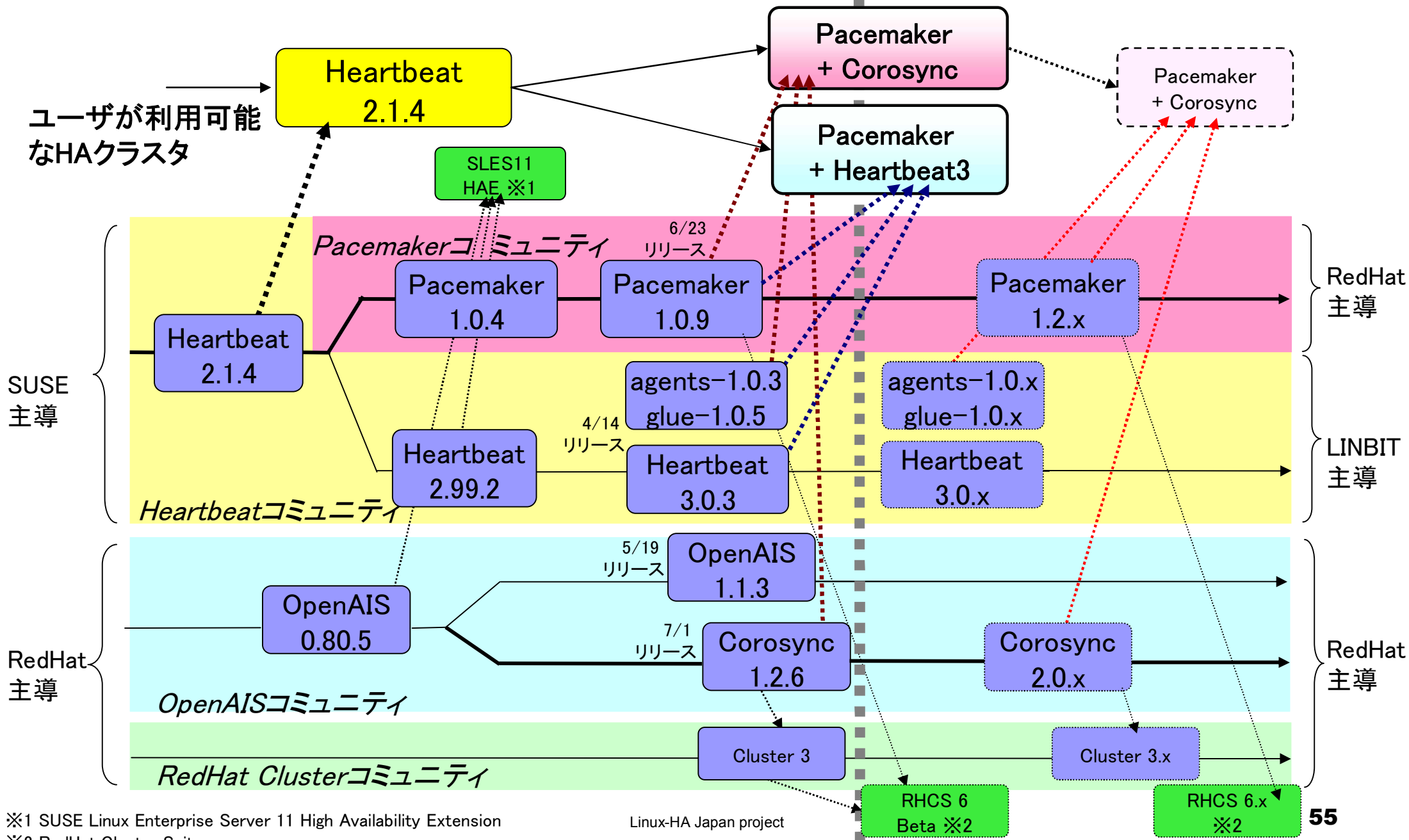
OpenAIS

Corosync



# HAクラスタ開発コミュニティの状況

2010年7月10日時点



※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7月10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。

## ■ epel の yumリポジトリを設定

download.fedora.redhat.com から epel-release の rpmファイルをダウンロードしてインストールします。

```
# wget http://download.fedora.redhat.com/pub/epel/5/x86_64/epel-  
release-5-3.noarch.rpm  
  
# rpm -ivh epel-release-5-3.noarch.rpm
```

## ■ clusterlabs.org の 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 (clusterlabs)
- corosynclib (clusterlabs)
- cluster-glue (clusterlabs)
- cluster-glue-libs (clusterlabs)
- resource-agents (clusterlabs)
- heartbeat-libs (clusterlabs)
- libesmtp (epel)



# 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  
    interface {  
        ringnumber: 0  
        bindnetaddr: 192.168.1.0  
        mcastaddr: 226.94.1.1  
        mcastport: 5405  
    }  
}
```

token:  
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ー  
処理を行う

interface:  
インターコネクト通信を行う  
TOKEN通信の情報を設定



## □ 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を各ノードで起動します

# 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: 1.0.9-89bd754939df5150de7cd76835f98fe90851b677  
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="prmlpWWW" provider="heartbeat" type="IPaddr">
    <instance_attributes id="prmlpWWW-instance_attributes">
      <nvpair id="prmlpWWW-instance_attributes-ip" name="ip" value="192.168.0.108"/>
      <nvpair id="prmlpWWW-instance_attributes-nic" name="nic" value="eth1"/>
      <nvpair id="prmlpWWW-instance_attributes-cidr_netmask" name="cidr_netmask"
value="255.255.255.0"/>
    </instance_attributes>
    <operations>
      <op id="prmlpWWW-start-0s" interval="0s" name="start" on-fail="restart" timeout="60s"/>
      <op id="prmlpWWW-monitor-10s" interval="10s" name="monitor" on-fail="restart"
timeout="60s"/>
      <op id="prmlpWWW-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...



Heartbeatバージョン2を  
使おうとして、  
このXMLで挫折した人は  
多いはずですよ...

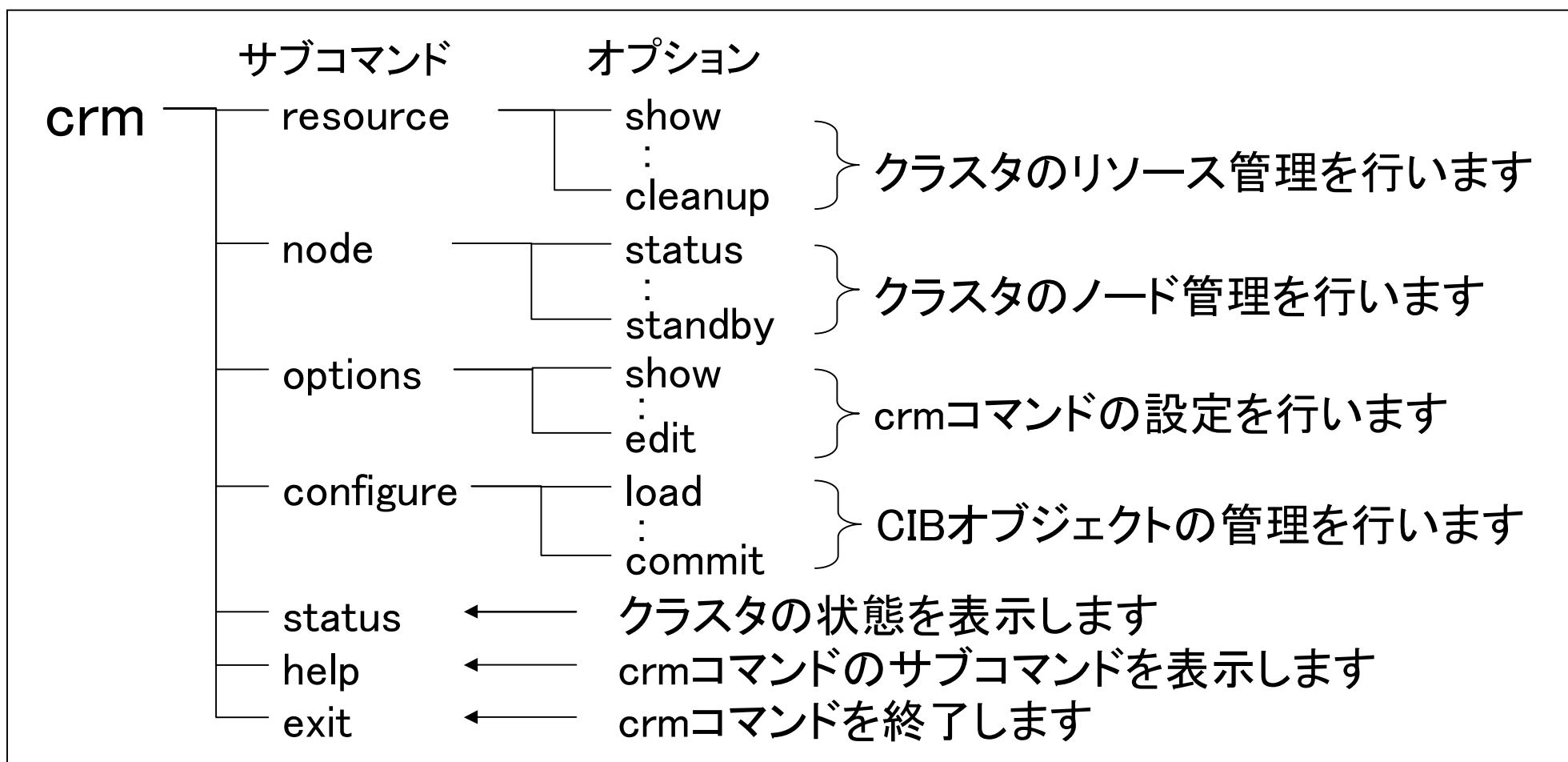
そこで、



# Pacemaker での新機能 crmコマンドを 使ってみよう！

# 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 prmlpWWW 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。  
(つまりリソース設定の根っこは cib.xml なのです)

これでも設定方法が  
わかりにくいって人には、

# 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！

※ 7/10 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。

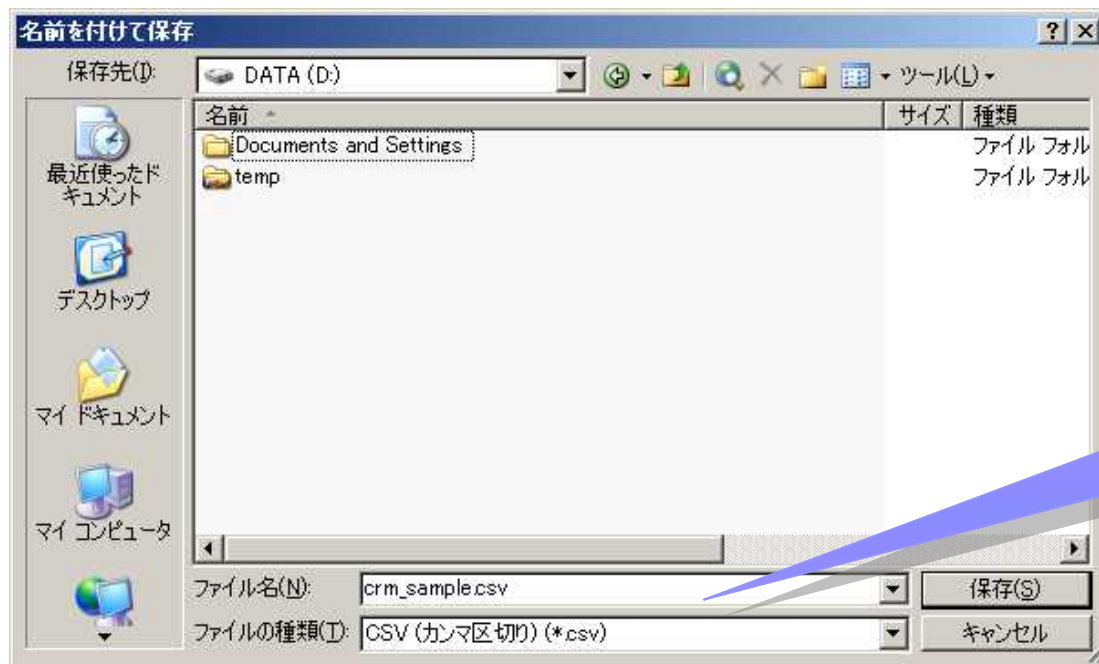
39	#表 2-2 クラスタ設定 ... Primitiveリソース (id=prmlpWWW)				
40	PRIMITIVE				
41	P	id	class	provider	type
42	#	リソースID	class	provider	type
43		prmlpWWW	ocf	heartbeat	IPaddr
44	A	type	name	value	
45	#	パラメータ種別	項目	設定内容	
46		params	ip	192.168.0.108	
47			nic	eth1	
48			cidr_netmask	255.255.255.0	
49	O	type	timeout	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>
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



## 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 prmlpWWW 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 Jul 6 12:42:14 2010

Stack: openais

Current DC: pm01 - partition with quorum

Version: 1.0.9-89bd754939df5150de7cd76835f98fe90851b677

2 Nodes configured, 2 expected votes

3 Resources configured.

=====

Online: [ pm02 pm01 ]

Resource Group: grpStonithN1

prnStonithN1 (stonith:external/riloe): Started pm02

Resource Group: grpStonithN2

prnStonithN2 (stonith:external/riloe): Started pm01

Resource Group: grpWWW

prmlpWWW (ocf::heartbeat:IPaddr): Started pm01

ノード1に仮想IPが付与されました



もしノード故障が発生すると・・・

=====

Last updated: Tue Jul 6 21:59:47 2010

Stack: openais

Current DC: pm02 - partition WITHOUT quorum

Version: 1.0.9-89bd754939df5150de7cd76835f98fe90851b677

2 Nodes configured, 2 expected votes

3 Resources configured.

=====

Online: [ pm02 ]

OFFLINE: [ pm01 ]

ノード2からはノード1が見えなくなったので「OFFLINE」と表示されます

Resource Group: grpStonithN1

prnStonithN1 (stonith:external/riloe): Started pm02

Resource Group: grpWWW

prmlpWWW (ocf::heartbeat:IPaddr): Started pm02

フェイルオーバーしてノード2に仮想IPが付与されます



もしリソース故障が発生すると・・・

=====

Last updated: Tue Jul 6 12:43:22 2010

Stack: openais

Current DC: pm01 - partition with quorum

Version: 1.0.9-89bd754939df5150de7cd76835f98fe90851b677

2 Nodes configured, 2 expected votes

3 Resources configured.

=====

Online: [ pm02 pm01 ]

Resource Group: grpStonithN1

prnStonithN1 (stonith:external/riloe): Started pm02

Resource Group: grpStonithN2

prnStonithN2 (stonith:external/riloe): Started pm01

Resource Group: grpWWW

prmlpWWW (ocf::heartbeat:IPaddr): Started pm02

Failed actions:

prmlpWWW\_monitor\_10000 (node=pm01, call=8, rc=7, status=complete): not running

フェイルオーバーしてノード2に  
仮想IPが付与されます

リソース故障状況  
が表示されます

HAクラスタの動作画面が  
地味なのは  
かんべんしてくださいね....。



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！

「PacemakerとDRBDでサーバー構築してみよう」の動画デモを公開中

# 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](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。

clusterlabs site-search:

*Pacemaker* A scalable High-Availability cluster resource manager

### Pacemaker 1.0.x - Supported Versions/Distributions

Binary packages for current Fedora, OpenSUSE and EPEL compatible distributions (eg. RHEL, CentOS and Scientific Linux) releases:

- Fedora
  - 10 [repository] [i386] [src] [x86\_64]
  - 11 [repository] [i386] [src] [x86\_64]
  - 12 [repository] [i386] [src] [x86\_64]
  - rawhide [repository] [src] [x86\_64]
- openSUSE
  - 11.0 [repository] [i386] [src] [x86\_64]
  - 11.1 [repository] [i386] [src] [x86\_64]
  - 11.2 [repository] [i386] [src] [x86\_64]
- EPEL
  - 4 [repository] [i386] [src] [x86\_64]
  - 5 [repository] [i386] [src] [x86\_64]

実は  
本家Pacemakerのロゴは

これ  です。



しかし

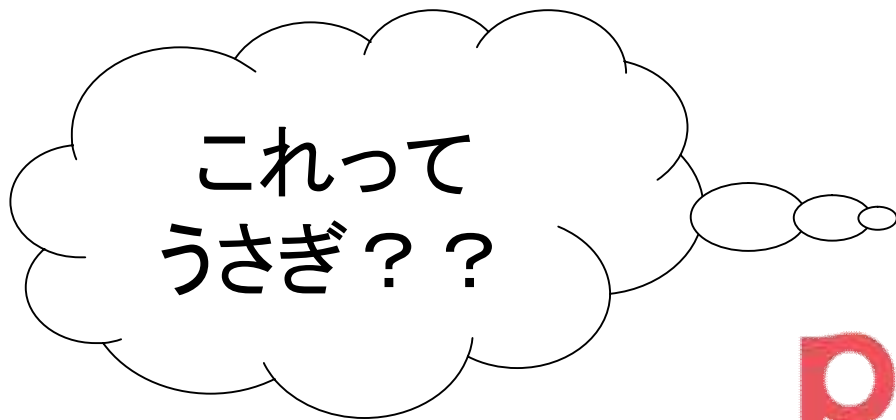
これ  では、

いかにも医療機器ですよね...

なので、

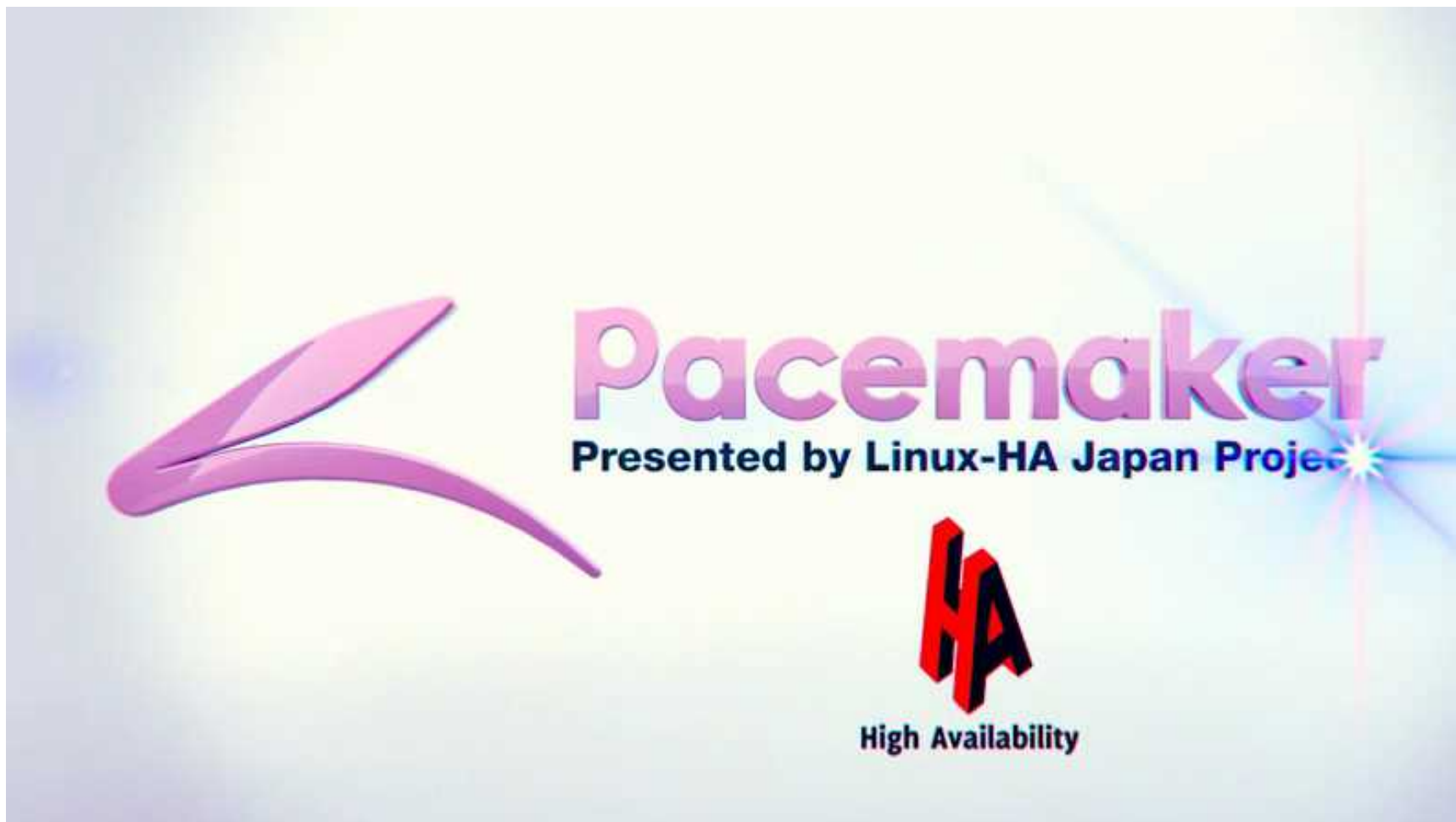
# Pacemakerロゴ

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](mailto: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

検索

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

# この娘たちもPacemakerを応援しています！



ご清聴ありがとうございました。