

Pacemaker-1.1で始める 高可用クラスタ入門 ～私が落ちても代わりはいるもの～

2015年5月23日 OSC2015
Nagoya

Linux-HA Japan
竹下 雄大



本日の内容

- Pacemakerってなに？
- 最新Pacemaker-1.1.12の特徴をご紹介
- Pacemakerクラスタを構築してみよう！
- 故障時の動きを体験してみよう！
 - リソース故障
- 今後のスケジュール

Pacemakerはオープンソースの
HAクラスタソフトです

Pacemakerってなに？

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

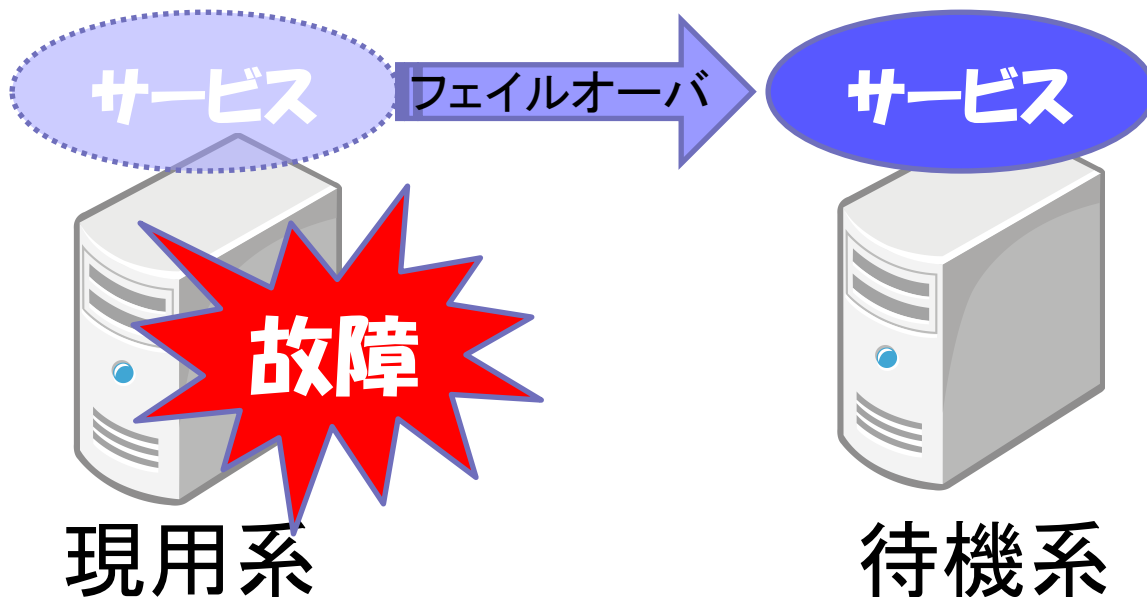
一台のコンピュータでは得られない高い信頼性を得るために、
複数のコンピュータを結合(クラスタ化)し、
ひとまとまりとする...

ためのソフトウェアです

Pacemakerってなに？

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

→このことを「**フェイルオーバー**」と言います



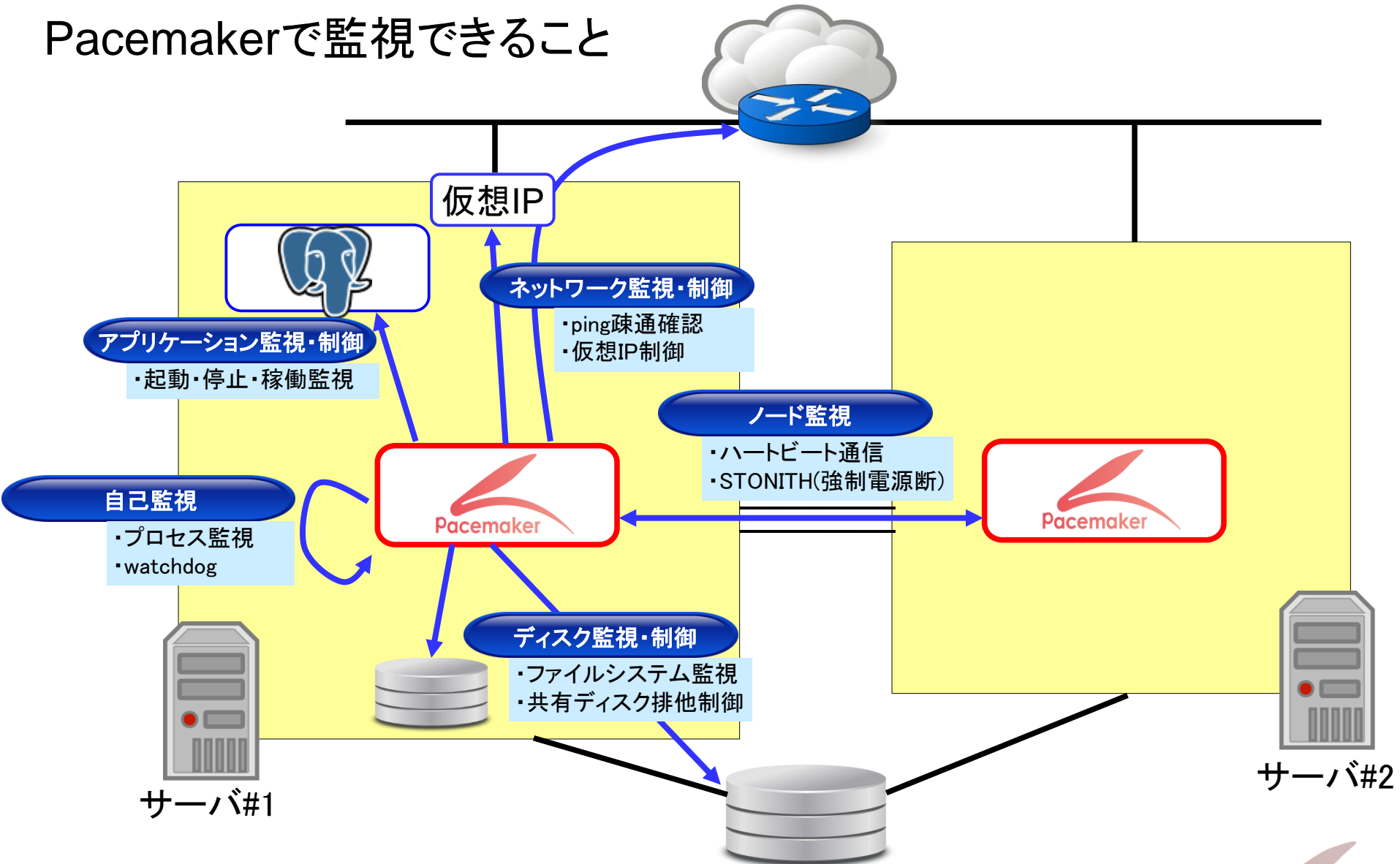
Pacemakerってなに？



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

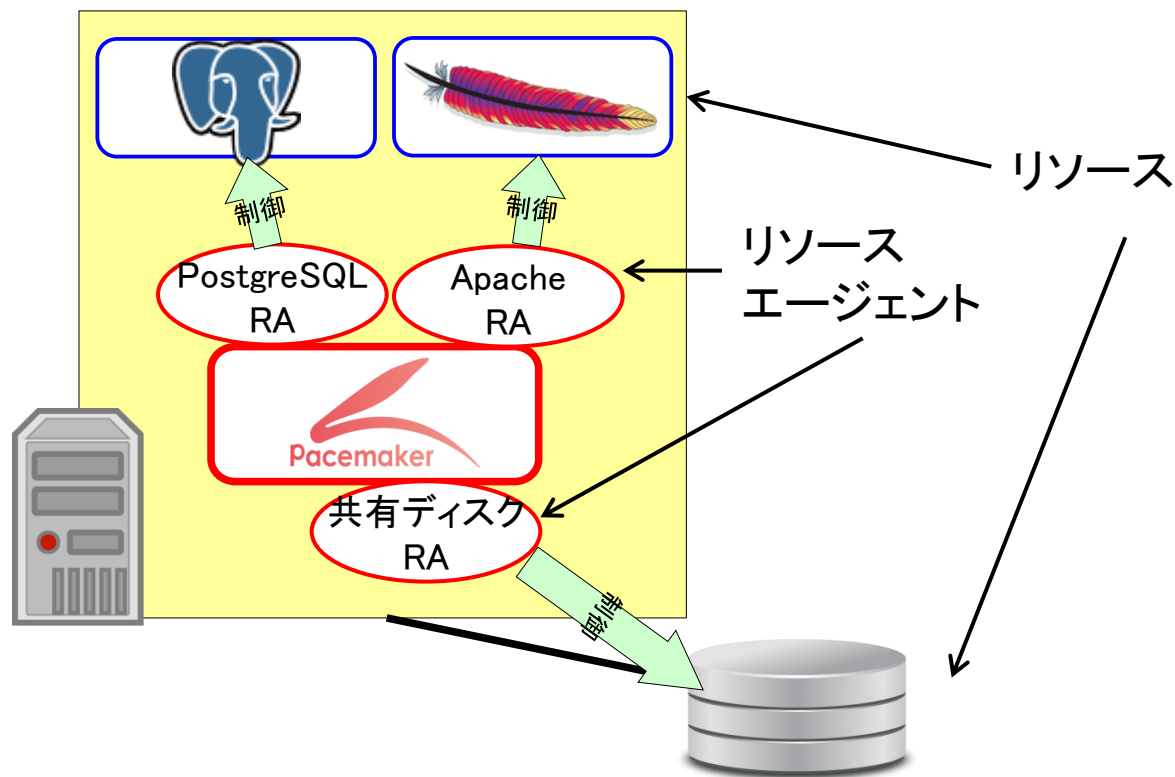
Pacemakerってなに？

Pacemakerで監視できること



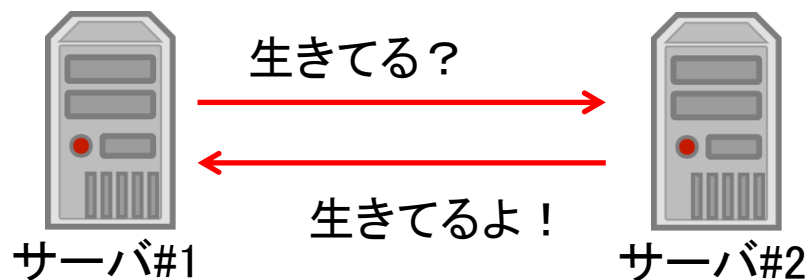
Pacemakerってなに？

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



スプリットブレインとSTONITH (1)

- ✓ Pacemakerで管理されるノードはインターコネクトLANを通して、ハートビート通信によってお互いの状態を把握しています



- ✓ ハートビート通信が途切れると・・・?
 - ✓ ACT系: SBY系がダウンしたかも? でもサービスは稼働しているからこのままでいいか
 - ✓ SBY系: ACT系がダウンしたかも! ? サービス停止するかもしれないから、サービスを起動しないと!



両ACT状態になる(**スプリットブレイン**)



サービスの両系起動
仮想IPアドレスの競合

共有ディスクの2重マウント(最悪の場合、**データ破壊**が発生)

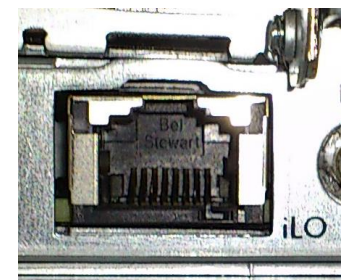
スプリットブレインとSTONITH (2)

- ✓ スプリットブレインを阻止するには？
- ✓ 確実(かつ最終的)な手段はこれ



スプリットブレインとSTONITH (3)

- ✓ STONITH (**S**hoot **T**he **O**ther **N**ode **I**n **T**he **H**ead)
 - ✓ 両ACT状態になる前に、対向ノードの**強制電源断**を実行する機能
 - ✓ サーバ付属のリモートHW制御ボード(iLOなど)を利用
 - ✓ OSと別系統のネットワークなので信頼性が高い



主なSTONITHプラグイン	用途
ipmi	物理環境で利用 OSと別系統のネットワークを利用するため信頼性が高い ただし、iLOなどのHW制御ボードが必要
libvirt	libvirtで制御される仮想環境で利用 ホストマシンからゲストマシンを強制停止する
stonith-helper※	スプリットブレイン発生時に、下記の事象を防止する補助プラグイン (電源断は実行しない) <ul style="list-style-type: none">・正常稼働中のACTがSTONITHされること (不必要なフェイルオーバーの発生)・お互いが同時にSTONITHを実行し、両系ダウン(相撃ち)

※ Linux-HA Japan製のプラグイン

Linux-HA Japan公式サイト(<http://osdn.jp/projects/linux-ha/>)から取得可能

【参考】STONITHプラグインが使えない場合のスプリットブレイン対策

✓ sfex

- ✓ 共有ディスクのsfex専用パーティションに、ディスクのロック情報を定期的に書き込む
- ✓ ACT系によりロック情報が更新されていれば、ACT系が生存していると判断し、SBY系でのリソース起動を抑止

✓ VIPcheck

- ✓ SBY系からACT系のVIPに対してpingを送信
- ✓ ping応答があれば、ACT系が生存していると判断し、SBY系でのリソース起動を抑止

最新Pacemaker-1.1.12 の特徴をご紹介します

2つのPacemaker

- ✓ Pacemakerには1.0系と1.1系の2種類が存在します
 - ✓ 基本機能や管理できるリソースに違いはありません
- ✓ 何が違うの？
 - ✓ コンポーネント
 - ✓ 新機能
 - ✓ 動作速度の向上
 - ✓ Pacemaker-1.0は開発・メンテナンスが終了しました
- ✓ どちらを使えばいいの??

新規導入の場合、Pacemaker-1.1の利用をお勧めします！

Pacemaker-1.0と1.1の比較

	Pacemaker-1.0系	Pacemaker-1.1系
対応OS(※1)	RHEL 5/6, CentOS 5/6	RHEL 6/7, CentOS 6/7
クラスタ上限(※2)	6ノード	16ノード
クラスタ起動速度	比較的遅い	早い (1.0系から7、8割程度短縮)
フェイルオーバー速度	比較的遅い	早い (1.0系から7割程度短縮)
機能		Pacemaker-1.0の機能は踏襲 Pacemaker-1.1系でのみ利用可能な新機能あり(※3)
実績	たくさん	これから

※1 Linux-HA Japan提供のリポジトリパッケージを利用する場合

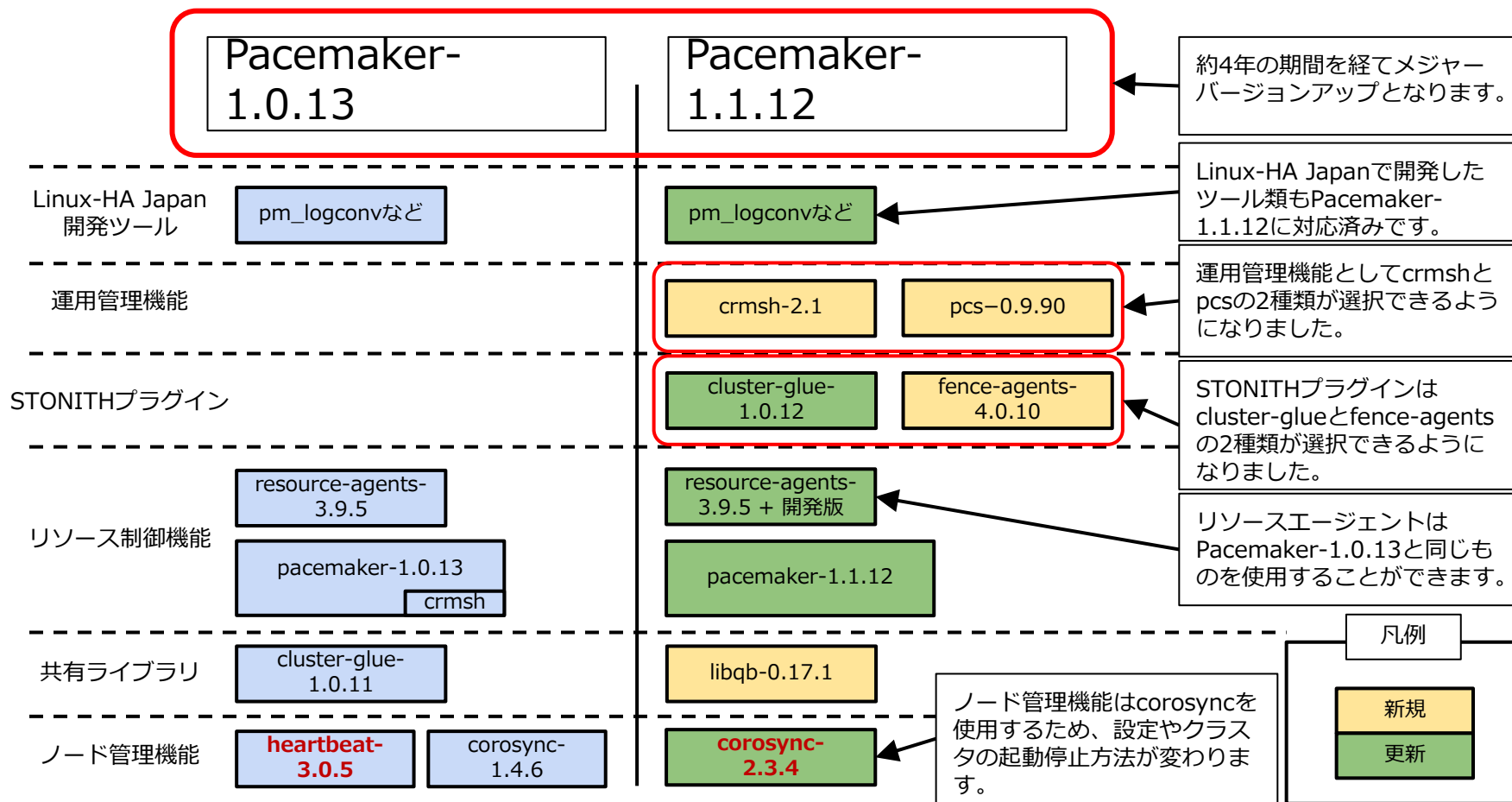
※2 上限値や速度はLinux-HA Japanでの検証によるもの(環境により異なります)

※3 Pacemaker-1.1の性能、新機能の詳細については下記参照

<http://linux-ha.osdn.jp/wp/archives/4075>

コンポーネントの比較

- ✓ Pacemakerは、様々なコンポーネントが組み合わさって動作します
- ✓ Pacemaker-1.1でコンポーネントが刷新されました



- ✓ 運用管理機能にはcrmshを利用する前提でお話します
 - ✓ crmshとpcsで管理コマンドなどが異なるため

ノード管理機能にCorosyncを採用しました

- ✓ Pacemaker-1.1.12からLinux-HA Japanではノード管理機能にCorosyncを採用しました！
- ✓ Corosyncを使用することによるメリット
 - ✓ Corosyncを使うとノード故障の検知速度が向上し、フェイルオーバー完了に要する時間を短縮することができます
 - ✓ Heartbeatでは6ノード、80リソース構成程度が限界だったが、Corosyncでは16ノード、200リソース程度までの動作実績があります

Pacemaker-1.1系の動作速度向上の主要因

Pacemakerクラスタを 構築してみよう！

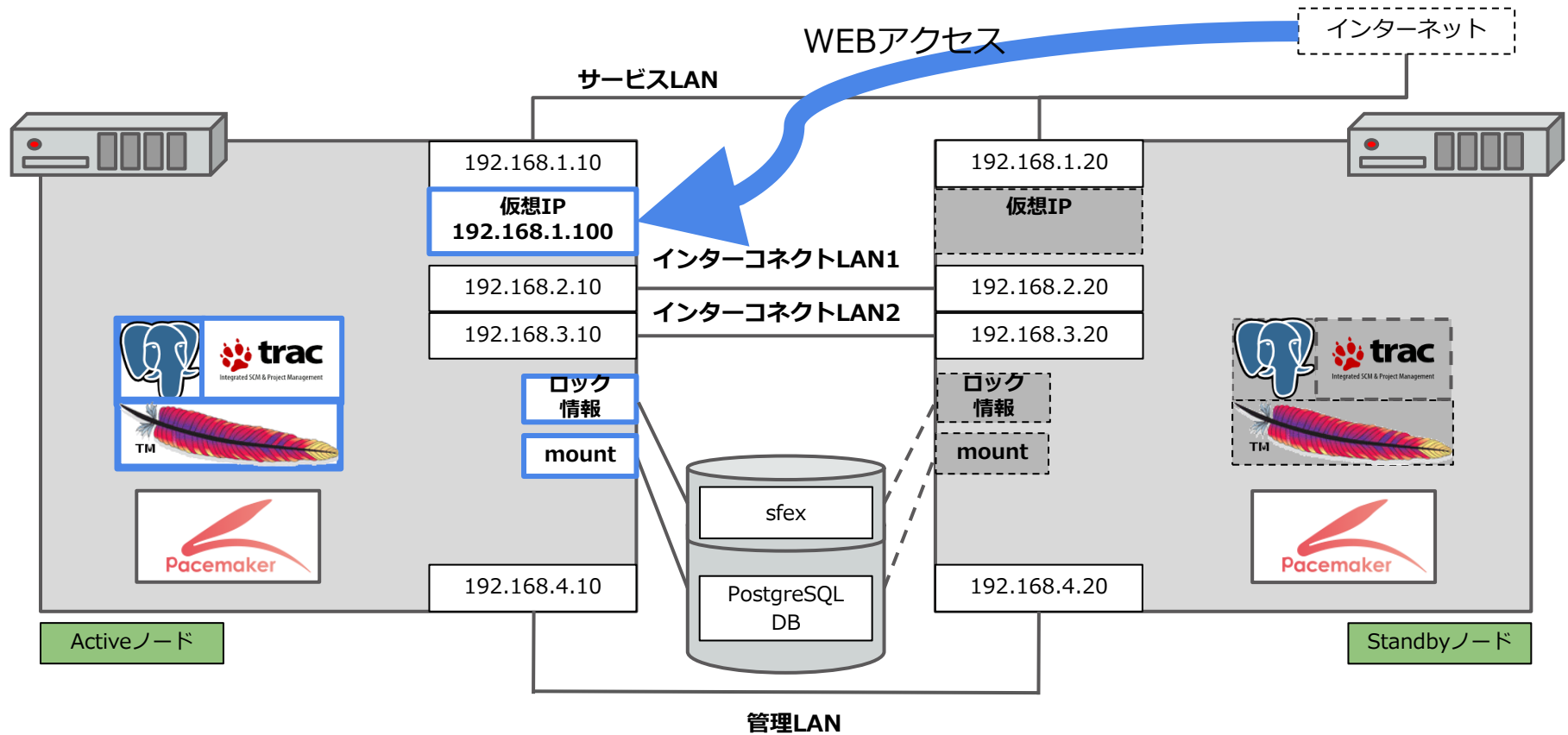
デモ環境について (1)

- ✓ Pacemaker、Apache、PostgreSQL、Tracを使用したWEBサービスのActive/Standby構成を作ります
 - ✓ 一般的なWEBサービスに必要なリソースはすべて組み込んでいます(仮想IPや共有ディスクも含め)
- ✓ この環境では次に挙げる故障に対応できます
 - ✓ リソース故障
 - ✓ ノード故障
 - ✓ ディスク故障(内蔵・共有ディスク)
 - ✓ ネットワーク故障(サービスLAN故障)
 - ✓ インターコネクトLAN故障(スプリットブレイン)
 - ✓ ただし、STONITHではなく、sfexによる対応

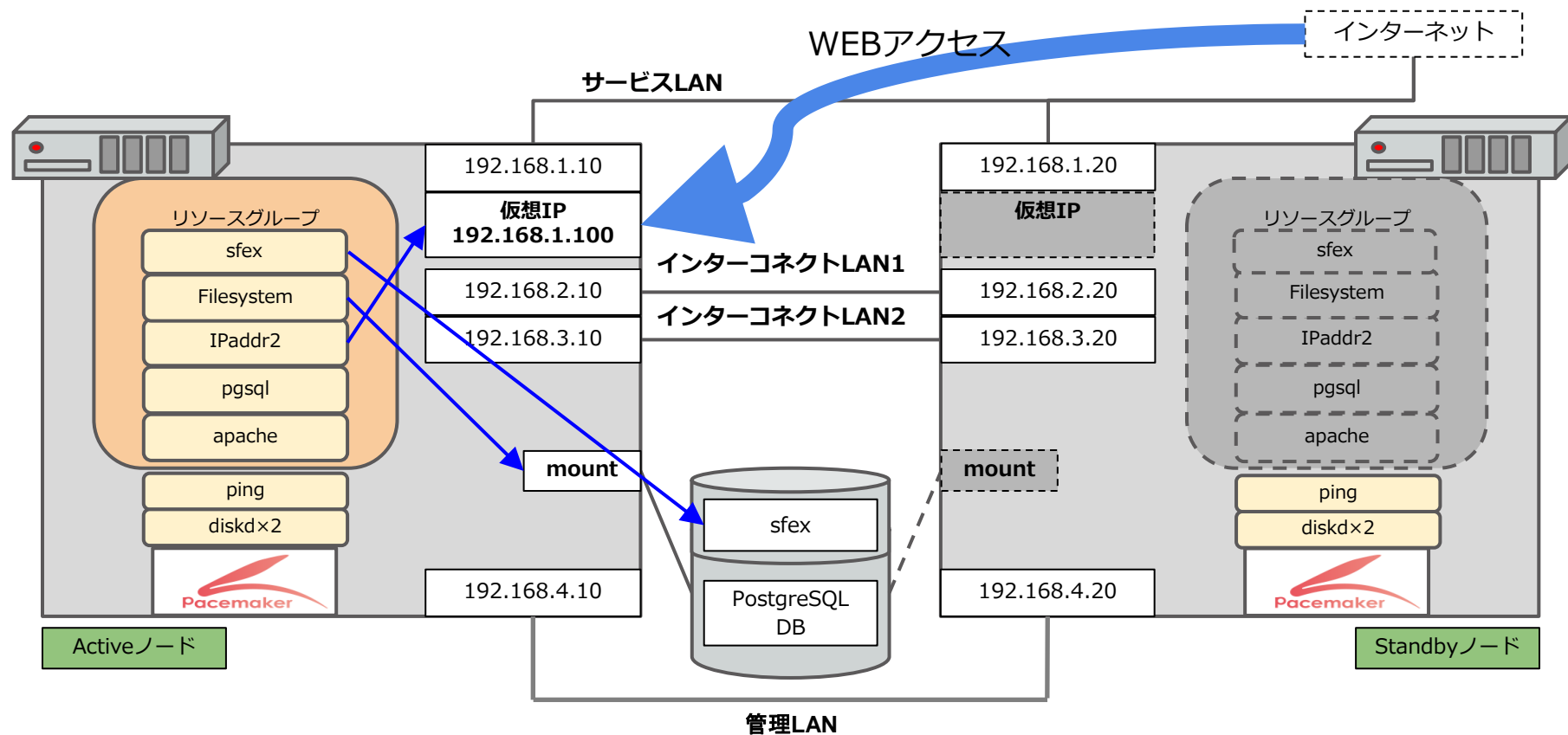
デモ環境について (2)

- ✓ デモ環境は仮想マシン2台(VMware Player)を利用します
 - ✓ ハードウェア
 - ✓ CPU: 1コア、メモリ: 1GB、ディスク: 10GB
 - ✓ ソフトウェア
 - ✓ OS: CentOS-6.6-x86_64
 - ✓ PostgreSQL-9.4.1 (公式サイトから取得)
 - ✓ Trac-1.0.1 (easy_installコマンドを使ってインストール)
 - ✓ httpd-2.2.15-39 (OS同梱版を使用)

こんな環境を作ります



Pacemakerのリソースに表すと、このようになります



前提条件

- ✓ 説明を簡略化するため、以下のセットアップを予め行っています
 - ✓ Apache、PostgreSQL、Tracはインストール済みで動作する状態
 - ✓ Tracは下記のアドレスでアクセスできるように構築済み
 - ✓ <http://192.168.1.100/osc2015nagoya>
 - ✓ **selinuxとiptablesは無効化**しています
- ✓ 作業はrootユーザで行います

Pacemakerのインストール

1. Pacemakerリポジトリパッケージの取得
2. Pacemakerのインストール

Pacemakerリポジトリパッケージの取得

- ✓ Linux-HA JapanのHPからPacemakerリポジトリパッケージを取得します

http://osdn.jp/projects/linux-ha/downloads/62369/pacemaker-repo-1.1.12-1.1.el6.x86_64.rpm/

Pacemakerのインストール

- ✓ 先ほど取得したRPMをインストールします
 - ✓ Pacemakerのパッケージ群(RPM)とyumリポジトリの定義ファイル(repoファイル)がローカルディスクに配置されます

```
# rpm -ivh pacemaker-repo-1.1.12-1.1.el6.x86_64.rpm
```

```
# ls /opt/linux-ha/pacemaker/rpm/  
pacemaker-1.1.12-1.el6.x86_64.rpm  corosync-2.3.4-  
1.el6.x86_64.rpm  
resource-agents-3.9.5-1.589.b6443.el6.x86_64.rpm  
crmsh-2.1-1.el6.x86_64.rpm  
(snip)
```

```
# ls /etc/yum.repos.d/pacemaker.repo  
/etc/yum.repos.d/pacemaker.repo
```

Pacemakerのインストール

- ✓ yumコマンドでPacemakerをインストールします
 - ✓ 依存関係のあるパッケージは自動的にインストールされます
 - ✓ インターネットに接続できない(外部のyumリポジトリを利用できない)場合は、インストールメディアを利用してローカルリポジトリを作成してください※1

```
# yum -y install pacemaker-all
(snip)
Complete!
```

(※1) CentOS等では、PacemakerがOSに同梱されているため、yumによりOS同梱のPacemakerがインストールされる場合があります
その場合、対象のrepoファイルに下記を追記してください

`exclude=pacemaker pacemaker-libs corosync cluster-glue heartbeat resource-agents`

Pacemakerクラスタを動かすための設定

1. corosync.confの作成
2. 認証鍵ファイルの作成（corosync通信用）
3. /etc/sysconfig/pacemakerの設定
4. クラスタ起動スクリプトの修正

corosync.confの作成

- ✓ /etc/corosync/corosync.confを以下のように作成します
 - ✓ クラスタを組む全てのマシンに同じファイルを配置してください
 - ✓ 赤字の設定については自身の環境に合わせて適宜変更してください

```
totem {  
  version: 2  
  rrp_mode: active  
  token: 1000  
  interface {  
    ringnumber: 0  
    bindnetaddr: 192.168.2.0  
    mcastaddr: 239.255.1.1  
    mcastport: 5405  
  }  
  interface {  
    ringnumber: 1  
    bindnetaddr: 192.168.3.0  
    mcastaddr: 239.255.1.1  
    mcastport: 5405  
  }  
}
```

クラスタ間通信に使用するネットワークアドレスを設定してください

マルチキャスト通信のアドレスを設定してください。
239.0.0.0～239.255.255.255の範囲が推奨です

クラスタ間通信で利用する受信ポートです

クラスタ間通信に使用するネットワーク全てについて定義してください

```
# ★ (続き)  
logging {  
  syslog_facility: daemon  
  debug: off  
}  
  
quorum {  
  provider: corosync_votequorum  
  expected_votes: 2  
}
```

syslogのファシリティを設定します
デフォルトの設定では「daemon」が設定されます

クラスタに参加するノード数を設定してください

認証鍵ファイルの作成

- ✓ 以下のコマンドを実行してクラスタ間通信に使用する認証鍵ファイルを作成します
 - ✓ 生成された認証鍵ファイルをクラスタを組む全てのマシンにコピーしてください

```
# corosync-keygen -l  
# ls -la /etc/corosync/authkey  
-rw-r--r-- 1 root root 128  8月 20 16:56 14  
/etc/corosync/authkey  
# scp -p /etc/corosync/authkey server02:/etc/corosync/authkey
```

/etc/sysconfig/pacemakerの設定

- ✓ 本設定でPacemakerのプロセスが故障した時の振る舞いを指定できます。
 - ✓ 本設定を追加すると、Pacemakerのプロセスが故障したノードはhalt状態となり、他のノードからはノードに故障が発生したと判断されるようになります。

```
# vi /etc/sysconfig/pacemaker
```

```
(snip)
```

```
67 # Enable this for rebooting this machine at the time of process  
(subsystem) failure
```

```
68 export PCMK_fail_fast=yes
```

```
69
```

```
(snip)
```

コメントアウトを外し、設定値を「yes」に

クラスタ起動スクリプトの修正(1)

- ✓ corosyncプロセスが故障した場合にcorosyncのwatchdogを動作させるため、起動スクリプトの52行目を有効にします。

```
# vi /etc/init/pacemaker.combined.conf
(snip)
50
51  # if you use watchdog of corosync, uncomment the line below.
52  pidof corosync || false
53
54  pidof crmd || stop corosync
(snip)
```

コメントアウトを外す

クラスタ起動スクリプトの修正(2)

- ✓ クラスタ起動中にOSをshutdownした場合にクラスタを正常に停止させるため、起動スクリプトの5行目に設定を追加します

```
# vi /etc/init/pacemaker.combined.conf
```

```
(snip)
```

```
3 # Starts Corosync cluster engine and Pacemaker cluster manager.
```

```
4
```

```
5 stop on runlevel [0123456]
```

この1行を追加

```
6 kill timeout 3600
```

```
7 respawn
```

```
(snip)
```

クラスタを起動する

1. クラスタを起動する
2. クラスタの状態を確認する

クラスタを起動する

- ✓ 以下のコマンドを実行してクラスタを起動します
 - ✓ Pacemaker-1.1.12からはUpstart経由(CentOS6)で起動します

```
# initctl start pacemaker.combined  
pacemaker.combined start/running, process 25490
```

- ✓ クラスタ停止コマンドはこちら

```
# initctl stop pacemaker.combined  
pacemaker.combined stop/waiting
```

クラスタの状態を確認する

- ✓ `crm_mon`※を実行してノードの状態が「Online」になっていることを確認します

```
# crm_mon -fAD1  
Online: [ server01 server02 ]  
(snip)
```

2台のマシンの状態が
Onlineになっているこ
とを確認します。

※`crm_mon`はクラスタの状態を確認するためのコマンドです

クラスタにリソースを管理させる

1. リソース定義ファイルを作成する
2. リソース定義ファイルをクラスタに読み込ませる
3. クラスタの状態を確認する
4. サービスが起動したことを確認してみよう

リソース定義ファイルを作成する

- ✓ 今回のデモ構成では以下のものをリソース化します
 - ✓ サービスリソース
 - ✓ apache
 - ✓ pgsql
 - ✓ IPaddr2(仮想IPの管理)
 - ✓ Filesystem(mountの管理)
 - ✓ sfex(共有ディスクロック情報の管理)
 - ✓ 監視リソース
 - ✓ ping(ネットワークを監視するリソース)
 - ✓ diskd(ディスクを監視するリソース)

リソース定義ファイルをクラスタに読み込ませる

- ✓ crmコマンド※1を実行してクラスタにリソース定義ファイル※2を読み込ませます。

```
# crm configure load update osc2015nagoya.crm
```

リソース定義ファイル

※1 crmコマンドはPacemakerクラスタを操作する運用管理コマンドです。
運用管理機能にcrmshを用いた場合に利用できます。

※2 リソース定義ファイルの詳細については下記参照
<http://linux-ha.osdn.jp/wp/archives/3786>

クラスタの状態を確認する

- ✓ crm_monを実行して、リソースがActiveノード※上で「Started」状態になったことを確認します

```
# crm_mon -fAD1
```

```
Online: [ server01 server02 ]
```

```
Full list of resources:
```

```
(snip)
```

```
Resource Group: grpTrac
```

```
    prmSFEX      (ocf::heartbeat:sfex):Started server01
```

```
    prmFS        (ocf::heartbeat:Filesystem):  Started server01
```

```
    prmVIP       (ocf::heartbeat:IPaddr2):    Started server01
```

```
    prmDB        (ocf::heartbeat:pgsql):      Started server01
```

```
    prmWEB       (ocf::heartbeat:apache):      Started server01
```

```
Clone Set: clnDiskd1 [prmDiskd1]
```

```
    Started: [ server01 server02 ]
```

```
Clone Set: clnDiskd2 [prmDiskd2]
```

```
    Started: [ server01 server02 ]
```

```
Clone Set: clnPing [prmPing]
```

```
    Started: [ server01 server02 ]
```

```
(snip)
```

リソースがActiveノード上で「Started」状態になっていることを確認します。

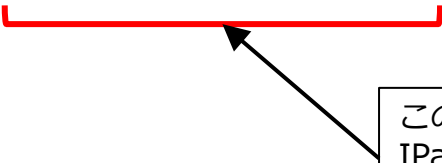
※ どちらのノードがActiveノードとなるかは、リソース定義ファイルの「制約」で記述します。制約については下記参照

<http://linux-ha.osdn.jp/wp/archives/3882>

サービスが起動したことを確認してみよう

- ✓ WEBブラウザを起動して、下記アドレスにアクセスします。Tracに接続できたら無事構築完了です

- ✓ <http://192.168.1.100/osc2015nagoya>



このIPはリソース定義の
IPaddr2で設定した仮想IPで
す。

故障時の動きを体験してみよう！

Pacemakerはどんな時にフェイルオーバーしてくれるの？

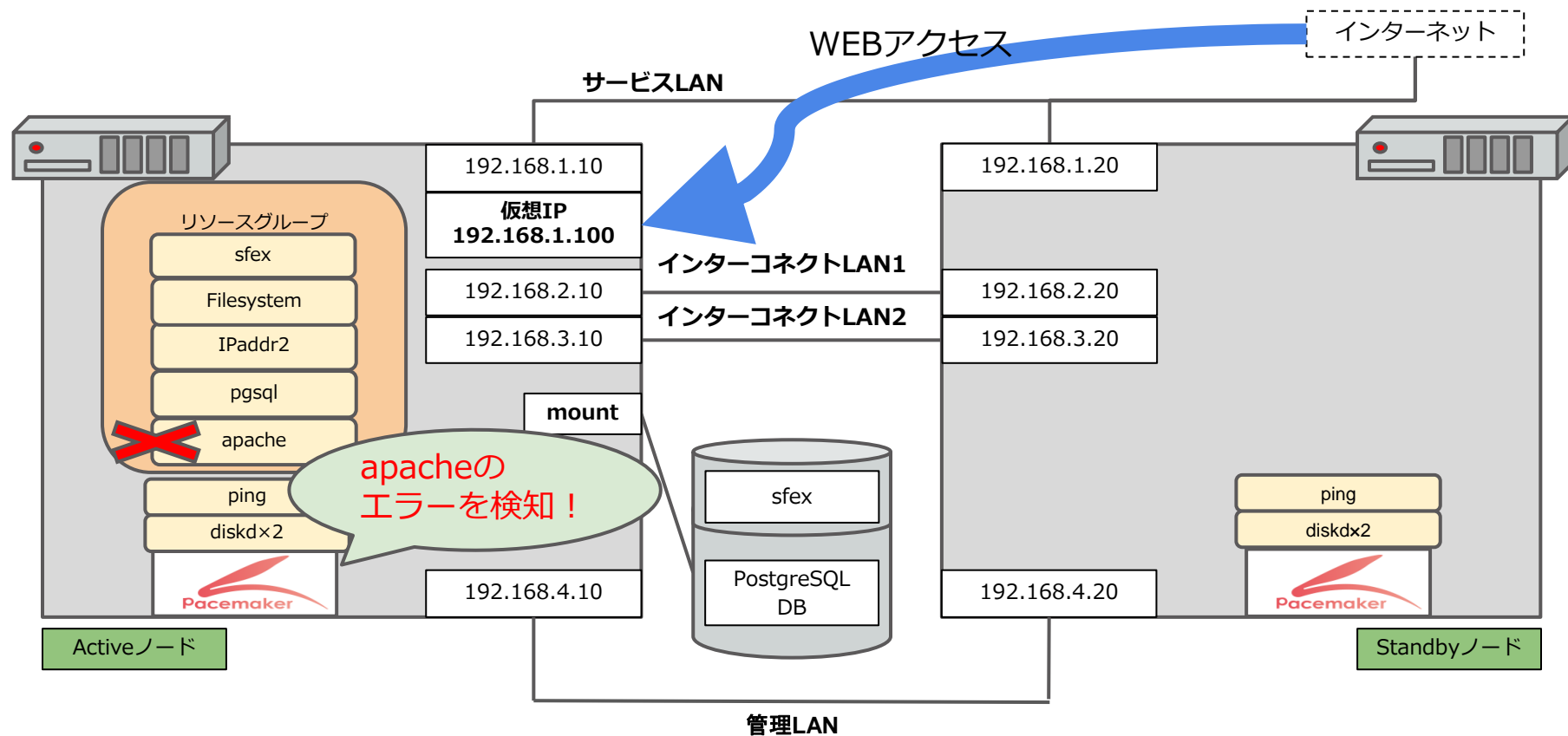
- ✓ 例えば、次に挙げるような状況になった時、リソースをフェイルオーバーしてくれます
 - ✓ リソース故障
 - ✓ 例) httpdプロセスが故障により停止してしまった時
 - ✓ ノード故障
 - ✓ 例) 電源故障によりノードが停止してしまった時
 - ✓ ディスクやネットワークの故障

リソース故障によるフェイルオーバーのデモ

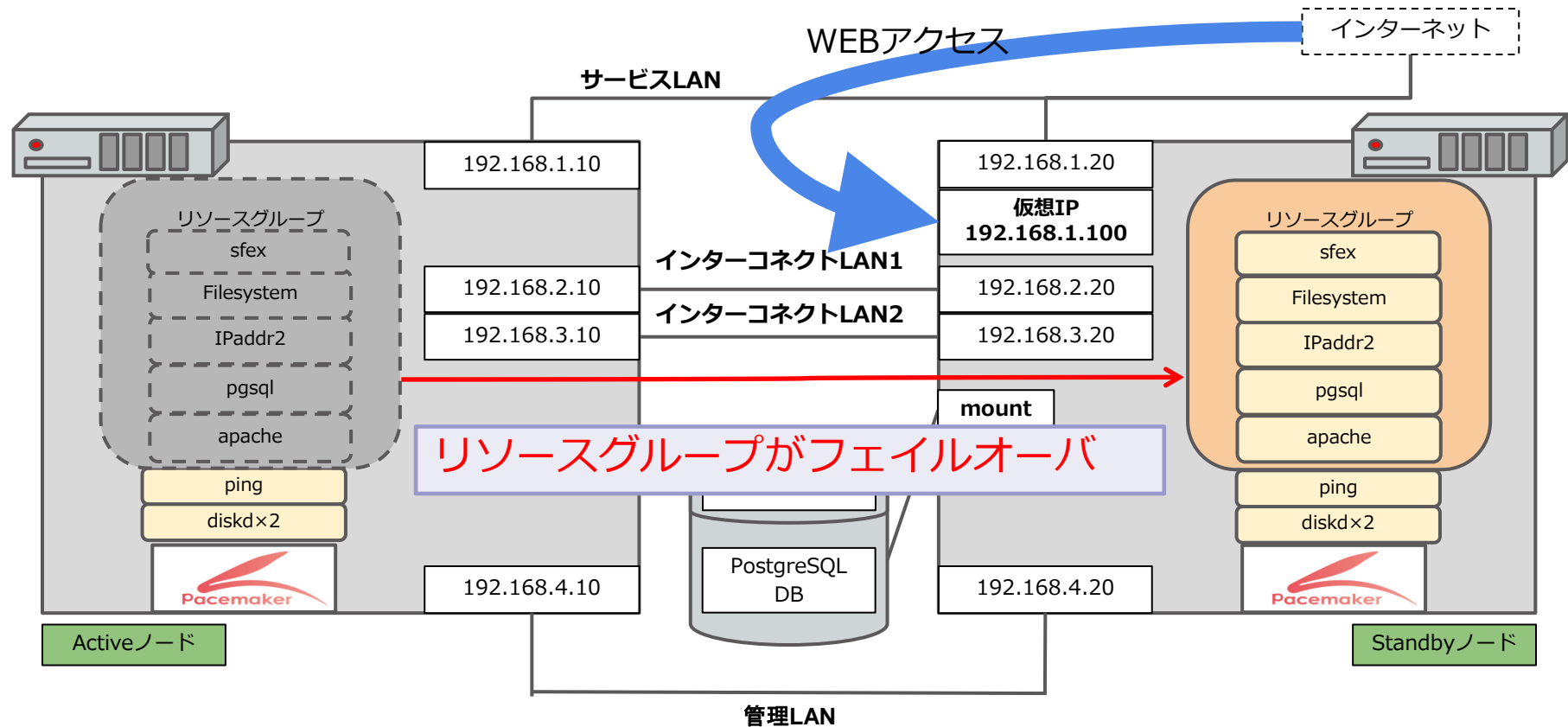
- ✓ 今回のデモではActiveノードでApache(httpd)プロセスをkillコマンドで強制停止させることで、フェイルオーバーを発生させます

```
# kill -9 <httpdの親プロセス>
```

リソース故障によるフェイルオーバー(故障発生時)



リソース故障によるフェイルオーバ(故障発生後)



- ✓ リソース故障時は、故障したリソースだけでなく、そのリソースが属するリソースグループ全体がフェイルオーバします

リソース故障発生後のクラスタ状態

```
# crm_mon -fAD1
```

```
(snip)
```

```
Resource Group: grpTrac
```

```
  prmSFEX    (ocf::heartbeat:sfex): Started server02  
  prmFS      (ocf::heartbeat:Filesystem): Started server02  
  prmVIP     (ocf::heartbeat:IPaddr2): Started server02  
  prmDB      (ocf::heartbeat:pgsql): Started server02  
  prmWEB     (ocf::heartbeat:apache): Started server02
```

```
Clone Set: clnDiskd1 [prmDiskd1]
```

```
  Started: [ server01 server02 ]
```

```
Clone Set: clnDiskd2 [prmDiskd2]
```

```
  Started: [ server01 server02 ]
```

```
Clone Set: clnPing [prmPing]
```

```
  Started: [ server01 server02 ]
```

```
(snip)
```

```
Migration summary:
```

```
* Node server01:
```

```
  prmWEB: migration-threshold=1 fail-count=1 last-failure='Mon May 18 14:04:52 2015'
```

```
* Node server02:
```

```
Failed actions:
```

```
  prmWEB_monitor_10000 on server01 'not running' (7): call=66, status=complete, last-rc-change='Mon May 18 14:04:52 2015', queued=0ms, exec=0ms
```

リソースはフェイルオーバーされ、Standbyノード上で起動されます

「Migration summary」に故障リソースの情報が表示されます

「Failed actions」に故障発生時のオペレーション情報が表示されます

(注)本来の運用では故障原因を取り除き、fail-countをクリアするなどして故障発生前の状態に戻しますが、今回のデモでは時間の都合上復旧の説明・手順は省き、一旦クラスタを再起動させる手順を取ります

今後のスケジュール

- ✓ 2015/5月現在のコミュニティ動向
 - ✓ ClusterLabs(本家コミュニティ)
 - ✓ 現在Pacemaker-1.1.13のリリースへ向け、作業中
 - ✓ Linux-HA Japan
 - ✓ Pacemaker-1.1.12のRHEL7対応
 - ✓ Pacemaker-1.1.13の確認、フィードバック

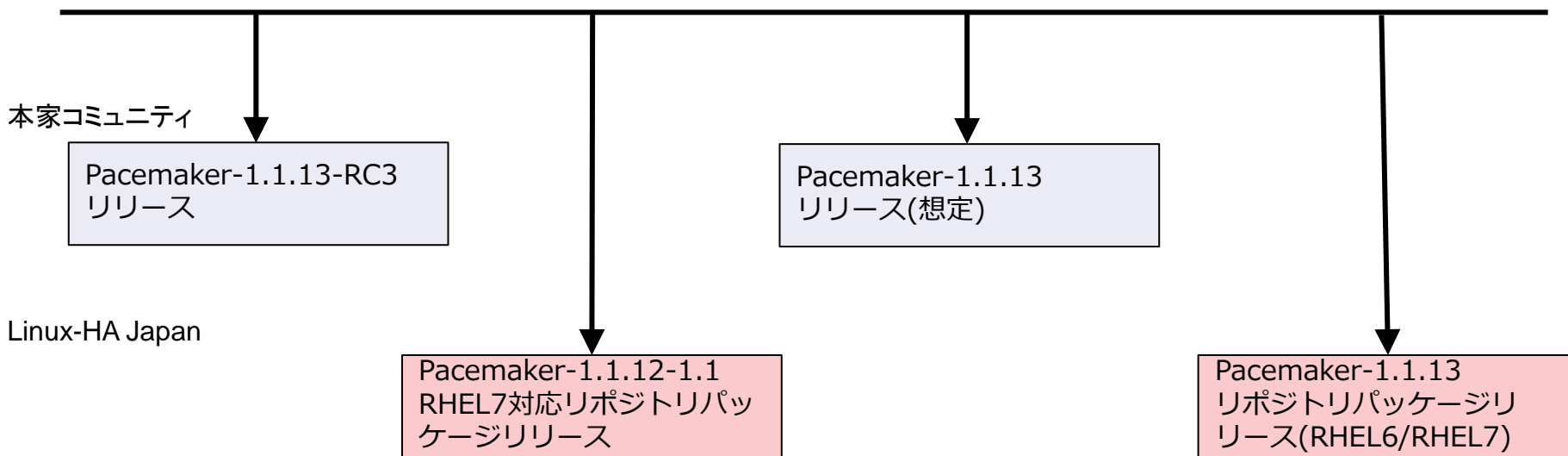
✓ 今後のリリーススケジュール(予定)

2015/5/15

2015/5月末

2015/6月

2015/10月



さいごに

Linux-HA Japan URL

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

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



The screenshot shows the Linux-HA Japan Project website. At the top is the logo with the text "Linux-HA JAPAN High-Availability Clustering on Linux". Below the logo is a navigation bar with links: HOME, メーリングリスト, ダウンロード&インストール, マニュアル, デスクトップテーマ・壁纸等, コミュニティ概要. Below the navigation bar is a section titled "Linux-HA Japan プロジェクト" with a description in Japanese. It mentions that the project is for building HA clusters on Linux and provides resources like RPM packages, manuals, and mailing lists. There is also a table with links to various resources like manuals, mailing lists, event information, and developer sites. At the bottom, there is a Twitter link and a note about the site's maintenance.

Linux-HA JAPAN
High-Availability Clustering on Linux

HOME メーリングリスト ダウンロード&インストール マニュアル デスクトップテーマ・壁纸等 コミュニティ概要

その他 ニュース イベント情報 読み物 WEBラジオ

Linux-HA Japan プロジェクト

Linux上で高可用(HA)クラスターシステムを構築するための 部品として、オープンソースの、クラスタリソースマネージャ、ク ラスタ通信レイヤ、ブロックデバイス複製、その他、さまざまなアプリケーションに対応するための数多くのソースエージェンツ等を、日本国内向けに維持管理、支援等を行っているプロジェクトです。

今は主に Pacemaker, Heartbeat, Corosync, DRBD等を扱っています。

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

Twitter 公式ハッシュタグ #linux_ha_jp

本サイトに関するお問い合わせは、Linux-HA Japan メーリングリスト管理者
(linux-ha-japan-owner アット lists.sourceforge.jp) までお願いいたします。

Pacemaker関連の最新情報を 日本語で発信

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

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

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

- ・ ML登録用URL

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



- ・ MLアドレス

linux-ha-japan@lists.osdn.me

※スパム防止のために、登録者以外の投稿は許可制です

ご清聴ありがとうございました。



Linux-HA Japan

検索



【参考】osc2015nagoya.crm

```
### Cluster Option ###
property no-quorum-policy="ignore" ¥
stonith-enabled="false" ¥
startup-fencing="false"

### Resource Defaults ###
rsc_defaults resource-stickiness="INFINITY" ¥
migration-threshold="1"

### Group Configuration ###
group grpTrac ¥
  prmSFEX ¥
  prmFS ¥
  prmVIP ¥
  prmDB ¥
  prmWEB

### Clone Configuration ###
clone clnPing ¥
  prmPing

clone clnDiskd1 ¥
  prmDiskd1

clone clnDiskd2 ¥
  prmDiskd2

### Master/Slave Configuration ###

### Fencing Topology ###

### Primitive Configuration ###
primitive prmSFEX ocf:heartbeat:sfex ¥
  params ¥
    device="/dev/sdb1" ¥
    index="1" ¥
    lock_timeout="70" ¥
    monitor_interval="10" ¥
  op start interval="0s" timeout="90s" on-fail="restart" ¥
  op monitor interval="10s" timeout="60s" on-fail="restart" ¥
  op stop interval="0s" timeout="60s" on-fail="block"
```

```
primitive prmFS ocf:heartbeat:Filesystem ¥
  params ¥
    fstype="ext4" ¥
    run_fsck="force" ¥
    device="/dev/sdb2" ¥
    options="barrier=0" ¥
    directory="/pgsqldb" ¥
  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="block"

primitive prmVIP ocf:heartbeat:IPaddr2 ¥
  params ¥
    ip="192.168.1.100" ¥
    nic="eth0" ¥
    cidr_netmask="24" ¥
  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="block"

primitive prmDB ocf:heartbeat:pgsql ¥
  params ¥
    pgctl="/usr/pgsql-9.4/bin/pg_ctl" ¥
    psql="/usr/pgsql-9.4/bin/psql" ¥
    pgdata="/pgsqldb/pgdata/data" ¥
    start_opt="-p 5432" ¥
    pgdba="postgres" ¥
    pgport="5432" ¥
    pgdb="template1" ¥
  op start interval="0s" timeout="300s" on-fail="restart" ¥
  op monitor interval="10s" timeout="60s" on-fail="restart" ¥
  op stop interval="0s" timeout="300s" on-fail="block"

primitive prmWEB ocf:heartbeat:apache ¥
  op start interval="0s" timeout="300s" on-fail="restart" ¥
  op monitor interval="10s" timeout="60s" on-fail="restart" ¥
  op stop interval="0s" timeout="300s" on-fail="block"
```

【参考】osc2015nagoya.crm

```
primitive prmping ocf:pacemaker:ping ¥
  params ¥
    name="default_ping_set" ¥
    host_list="192.168.1.5" ¥
    multiplier="100" ¥
    attempts="2" ¥
    timeout="2" ¥
    debug="true" ¥
  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="ignore"

primitive prmdisk1 ocf:pacemaker:diskd ¥
  params ¥
    name="diskcheck_status" ¥
    device="/dev/sdb" ¥
    options="-e -t 70" ¥
    interval="10" ¥
    dampen="2" ¥
  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="ignore"

primitive prmdisk2 ocf:pacemaker:diskd ¥
  params ¥
    name="diskcheck_status_internal" ¥
    device="/dev/sda" ¥
    options="-e" ¥
    interval="10" ¥
    dampen="2" ¥
  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="ignore"

### Resource Location ###
location rsc_location-grpTrac-1 grpTrac ¥
  rule 200: #uname eq server01 ¥
  rule 100: #uname eq server02 ¥
  rule -INFINITY: not_defined default_ping_set or default_ping_set lt 100 ¥
  rule -INFINITY: not_defined diskcheck_status or diskcheck_status eq ERROR ¥
  rule -INFINITY: not_defined diskcheck_status_internal or diskcheck_status_internal eq
ERROR
```

```
### Resource Colocation ###
colocation rsc_colocation-grpTrac-clnPing-1 INFINITY: grpTrac clnPing
colocation rsc_colocation-grpTrac-clnDiskd1-2 INFINITY: grpTrac clnDiskd1
colocation rsc_colocation-grpTrac-clnDiskd2-3 INFINITY: grpTrac clnDiskd2

### Resource Order ###
order rsc_order-clnPing-grpTrac-1 0: clnPing grpTrac symmetrical=false
order rsc_order-clnDiskd1-grpTrac-2 0: clnDiskd1 grpTrac symmetrical=false
order rsc_order-clnDiskd2-grpTrac-3 0: clnDiskd2 grpTrac symmetrical=false
```