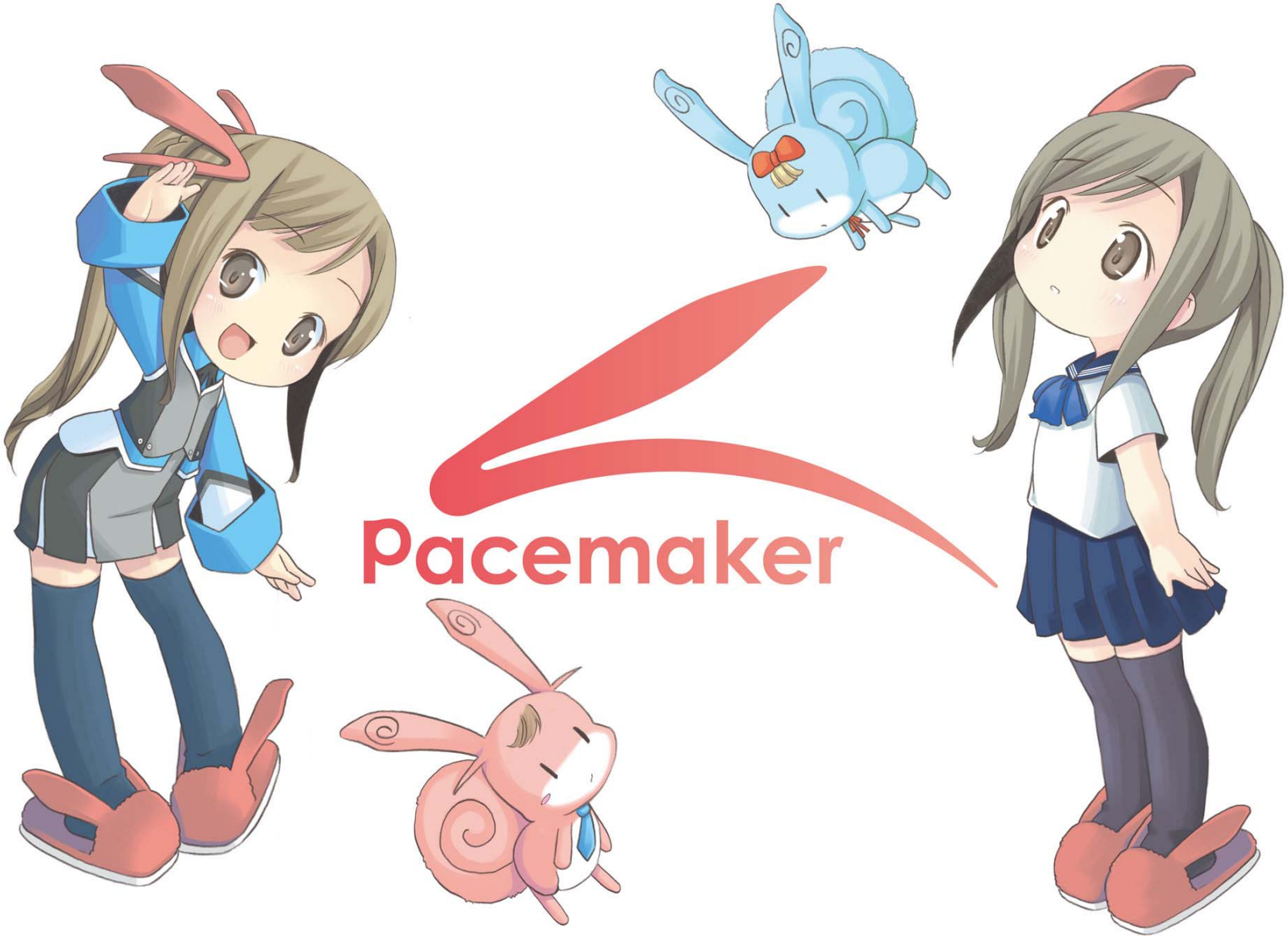
OSS HAクラスタPacemakerを活用したHAシステム構築の勘所

2011年4月28日

Linux-HA Japan
三井 一能

こんにちは



本日の内容

本日の内容
Pacemakerの歴史
コミュニティの動向
Pacemakerの概要
インストール、設定

講演を機会に

PacemakerでHAクラスタを組めそうだ

PacemakerでHAクラスタを組めそうだ
帰ってPacemakerでHAクラスタを組みたくなる

と思っていただきたいと思います

まず

まず自己紹介

名前

みいかずよし

漢字だと

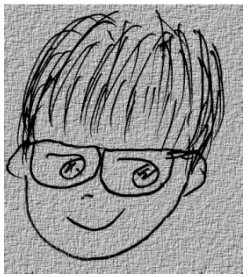
三井一能

三井一能
読み方が難しいですが、

「みいかずよし」と読みます

「みついさん」

でも返事します



id:kzmtw
よければfollowしてください

家族構成

妻
と
娘
の3人家族です

趣味

ありきたりですが、子育てですかね
イクメンの話題は大好きです

さて、Linux-HA Japanについて

Linux-HA Japanの経緯

HAクラスタ『Heartbeat』の日本における更なる普及展開を目的とし
2007年10月5日「Linux-HA (Heartbeat) 日本語サイト」を設立

Linux-HA Japanの経緯

Heartbeat２のrpmバイナリと、オリジナルのHeartbeat機能追加用パッケージを提供

Webサイト

http://linux-ha.sourceforge.jp/ （一般向け）
http://sourceforge.jp/projects/linux-ha/ (開発者向け

Pacemaker情報の公開用として
新しい一般向けウェブサイトが
2010/6/25にオープンしました。

本日の資料もこのサイトから
公開予定です！

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 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.sourceforge.jp/
  の「メーリングリスト」をクリック

- MLアドレス

  linux-ha-japan@lists.sourceforge.jp

※スパム防止のために、登録者以外の投稿は許可制です

gihyo.jp にて連載中！

『Pacemakerでかんたんクラスタリング体験してみよう！』

http://gihyo.jp/admin/serial/01/pacemaker

合計5回の連載で、Pacemakerの概要説明から構築方法、保守運用にいたるまで紹介しています。

勤務先

NTT
研究企画部門 OSSセンタ

疑問

NTTはなぜOSSに取り組んでいるのか

OSSの狙い

コスト削減

コスト削減
ベンダロックインの回避

コスト削減
ベンダロックインの回避
ホワイトボックスであるOSSを使った
技術力向上

こういった効果を狙い、社内システムのOSS適用支援を行っています

OSSセンタ

具体的な取り組みは？

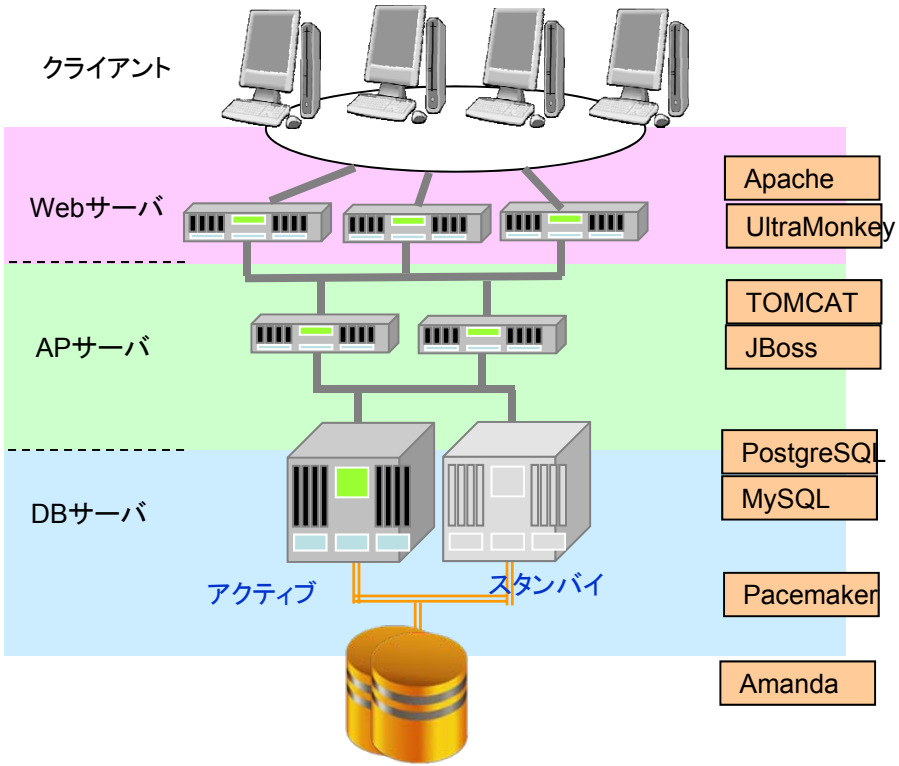
1つめ

OSSVERT

OSSVERT (オズバート)

OSs Suites VErified Technically

OSSVERT
安心して利用できるOSS製品の選定と技術検証の実施



2つめ

OSSVERTを構成するOSS製品の研究開発、コミュニティ活動

主に活動しているOSS製品
データベース: PostgreSQL
HAクラスタ: Pacemaker
APサーバ: JBoss

3つめ

グループ会社のOSS利用をトータルサポート

情報の一元提供
OSSの問題解決
個別パッチの提供

ここから本題

Pacemakerの話をします

Pacemakerってなに？

PacemakerはOSSのHAクラスタソフトウェアです

アンケートをとります

Pacemakerを知っていますか？

同じくOSSのHAクラスタであるHeartbeatを知っていますか？

Pacemakerは、Heartbeatの後継ソフトウェアです

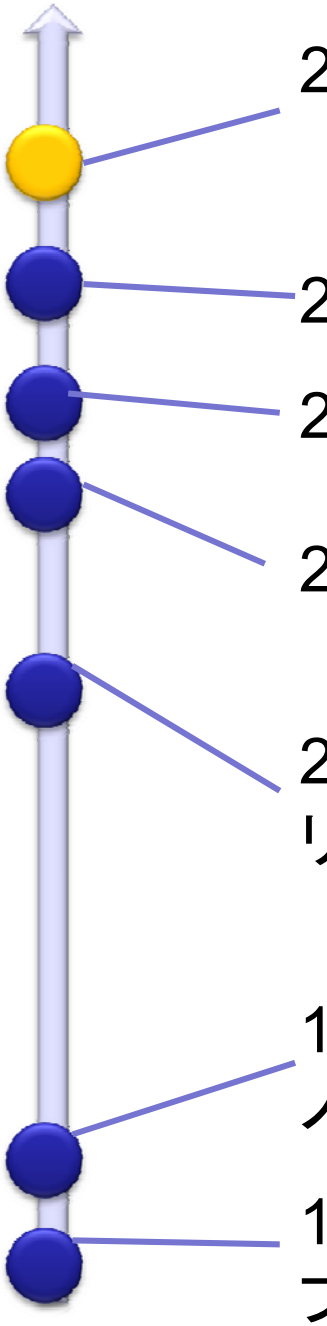
Pacemakerを導入すると

現用系で**故障**が発生しサービスができなくなったときに、待機系でサービスを**自動起動**し、サービス中断を最小限にすることができます

Pacemakerの歴史

Heartbeatの最初のバージョンから、12年の歴史があります

1998年 Linux-HAプロジェクト発足
Heartbeatのアルファ版



2010年 Pacemaker-1.0.10リリース

2009年 CorosyncをサポートするPacemaker-1.0.6

2008年 Pacemaker-1.0.0リリース

2007年 リソース管理機能がPacemakerとして独立

2005年 Heartbeatバージョン2リリース。多ノード構成、リソース監視が可能

1999年 Heartbeatバージョン1リリース。1+1構成、ノード監視可能

1998年 Linux-HAプロジェクト発足 Heartbeatのアルファ版

あれ、Pacemaker-1.1使っているけど

Pacemakerでのバージョンの考え方

Pacemakerでのバージョンの考え方

偶数バージョン: 1.0.x, 1.2.x

Pacemakerでのバージョンの考え方
偶数バージョン: 1.0.x, 1.2.x
長期安定リリース
バグフィックスのみ
3、4ヶ月周期でリリース
1.2系の安定版リリースは、2012-8ごろ

Pacemakerでのバージョンの考え方

奇数バージョン: 1.1.x

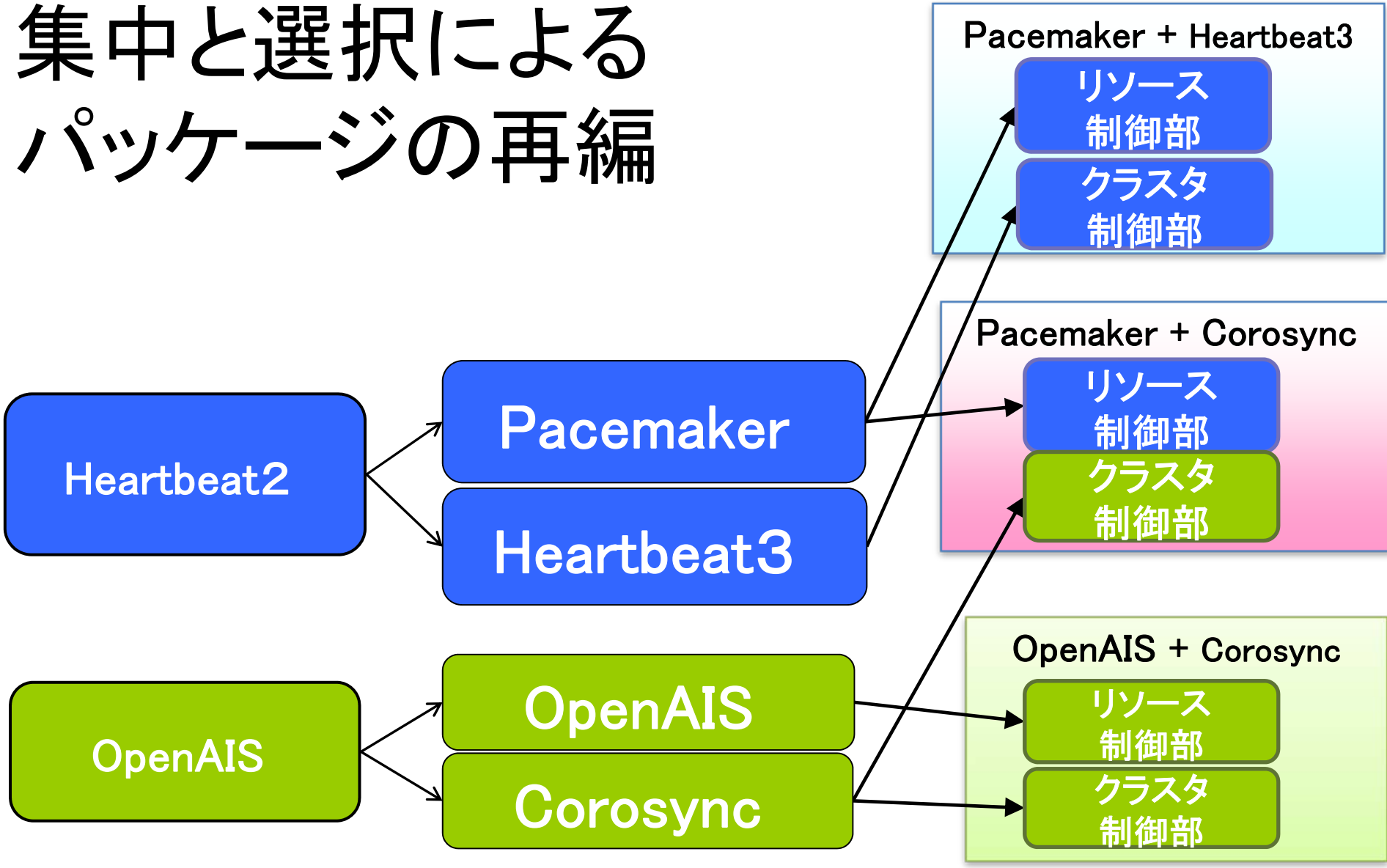
Pacemakerでのバージョンの考え方
奇数バージョン: 1.1.x
フィーチャーリリース
新規機能追加／削除
機能、設定の互換性は低め
3、4ヶ月周期でリリース
RHEL6にテクノロジープレビューとして同梱

現在の開発コミュニティの状況

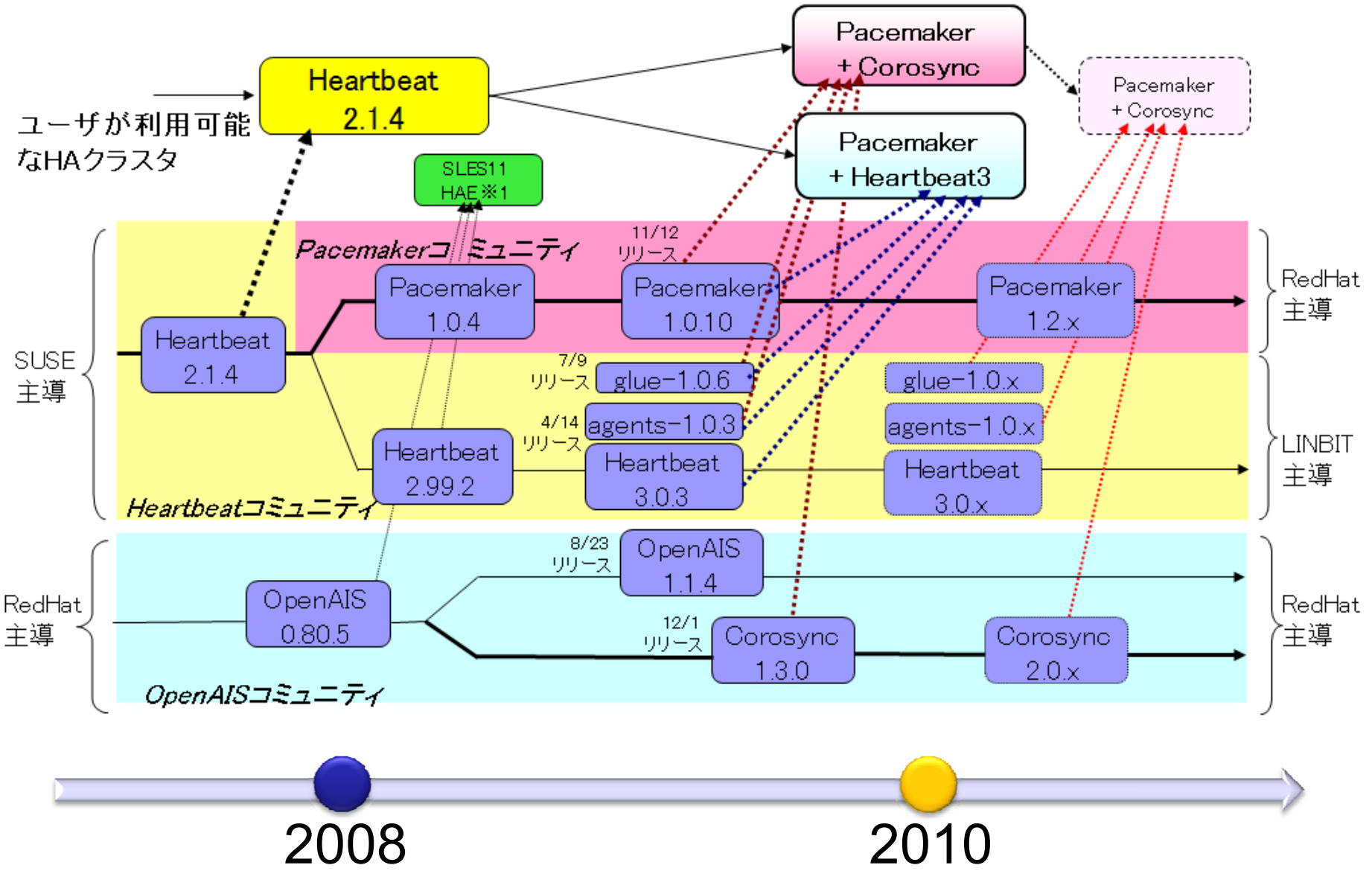
現在の開発コミュニティの状況

OSS HAクラスタ関連のプロジェクトが協力し合う関係が進んできている

現在の開発コミュニティの状況
集中と選択による
パッケージの再編



現在の開発コミュニティの状況



2011-4

Linux FoundationにHigh Availability Working Groupを設立

同じ傘の下でコミュニティ間の連携活動を強化する動き

- コミュニティ間でパッケージ統合
  - Resource agentsのマージ作業
- 2011-10 mini-summit開催(プラハ)

mini-summitは主要開発者がface-to-faceで議論する場で、毎年開催

前回
2010-11 Linux Plumbers Conference(ケンブリッジ)で開催

Pacemaker
1.0パッチメンテナ

OpenAIS
Corosync

Linux-cluster

DRBD
Heartbeat3

Linux-HA

Linux-HA
Japan

Pacemaker

OCFS2

Pacemakerってなに？

PacemakerはHAクラスタソフトウェアです

多彩なクラスタ構成が可能です

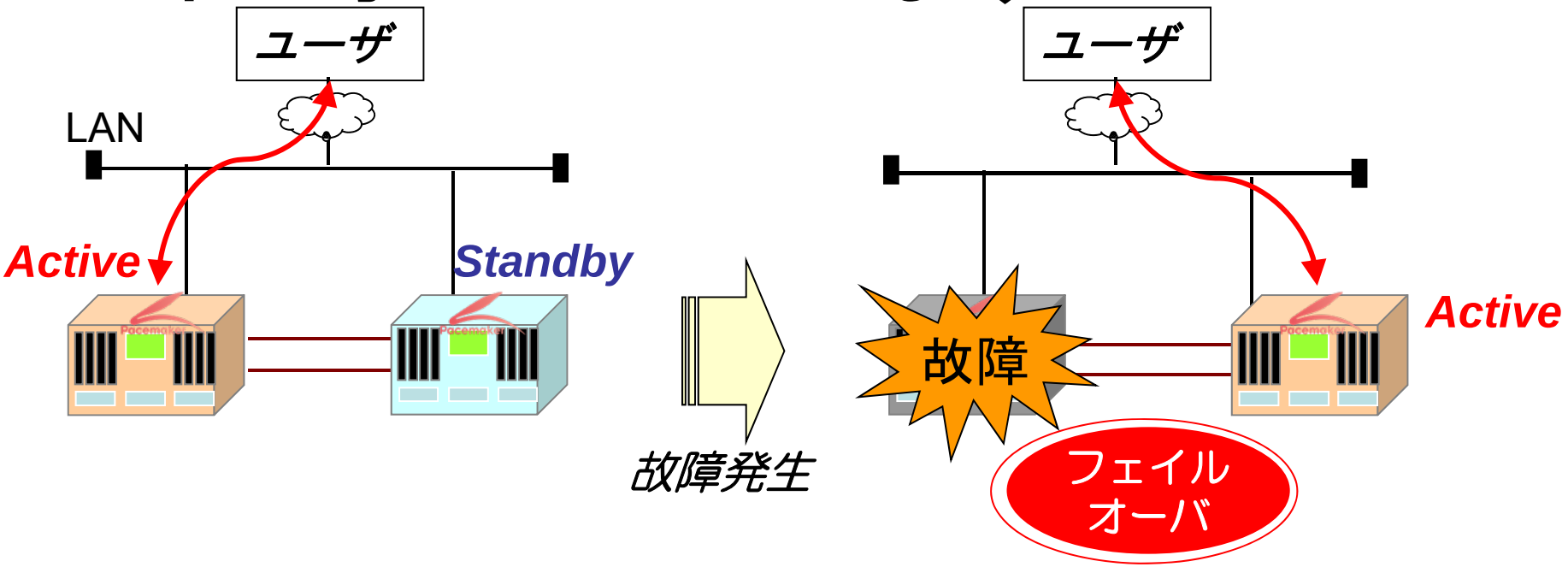
基本構成

1+1構成

1+1構成

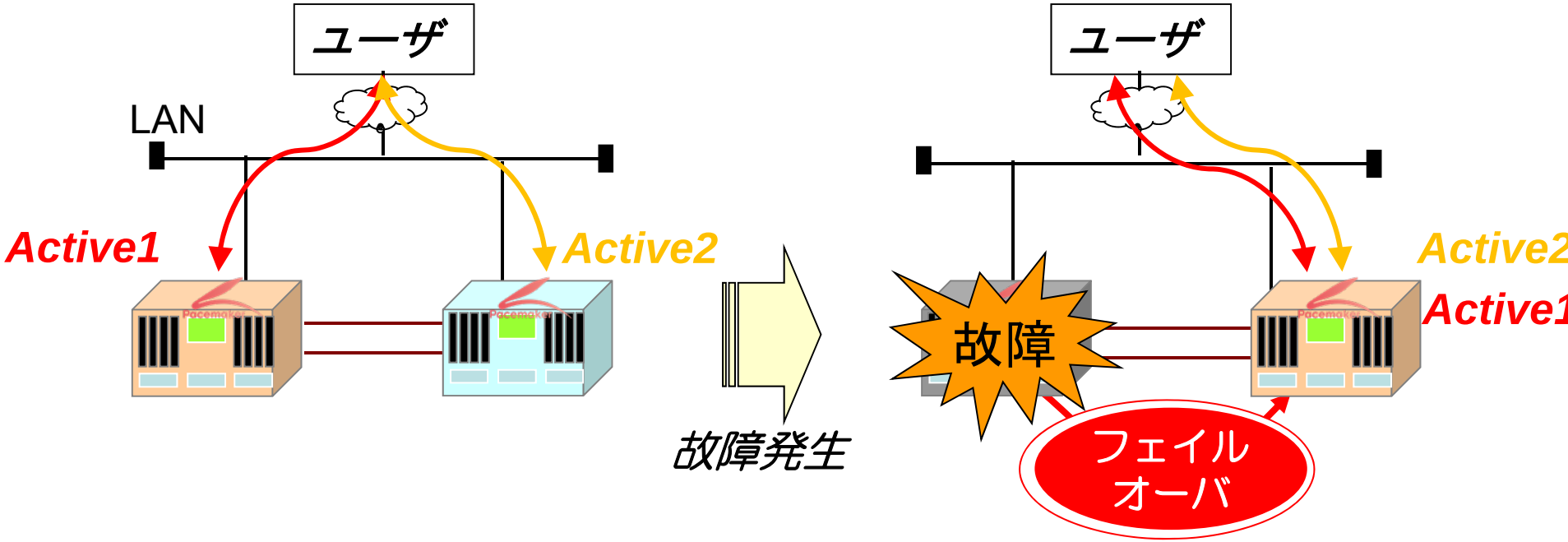
ACT-SBY構成

ACT-SBY構成
稼動系でサービスが動作
故障が発生すると待機系でサービスが起動
フェイルオーバといいます



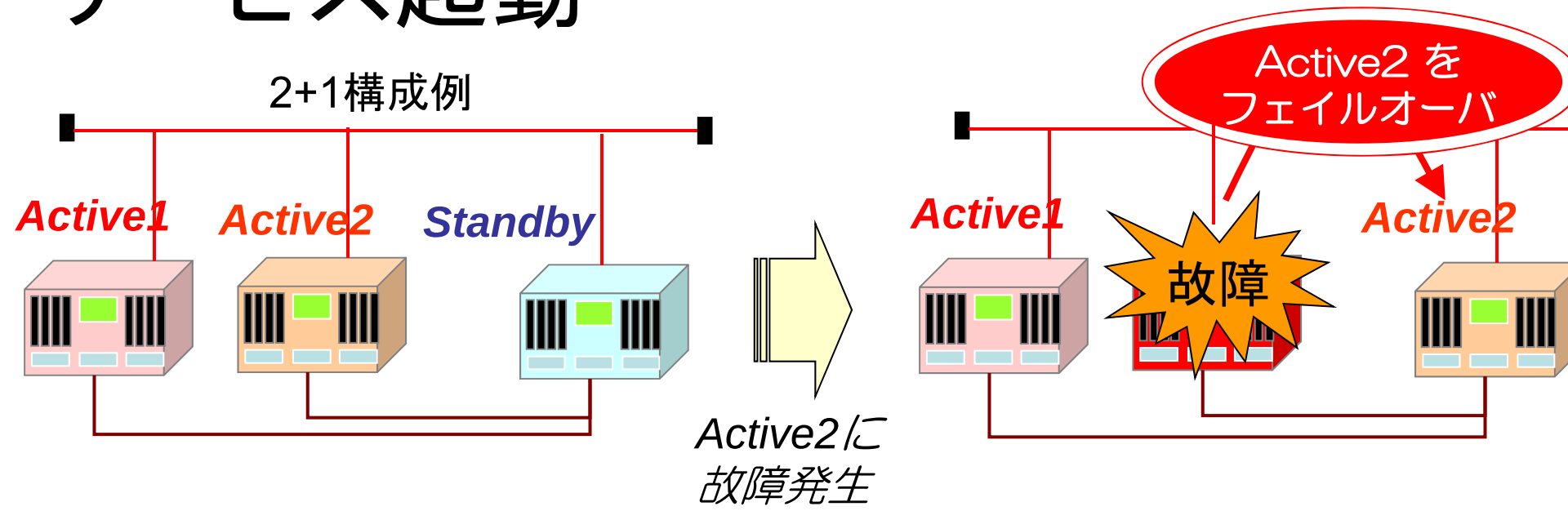
応用構成

ACT-ACT構成
両方のサーバでサービスが動作
故障が発生すると他方のサーバで
サービスを起動



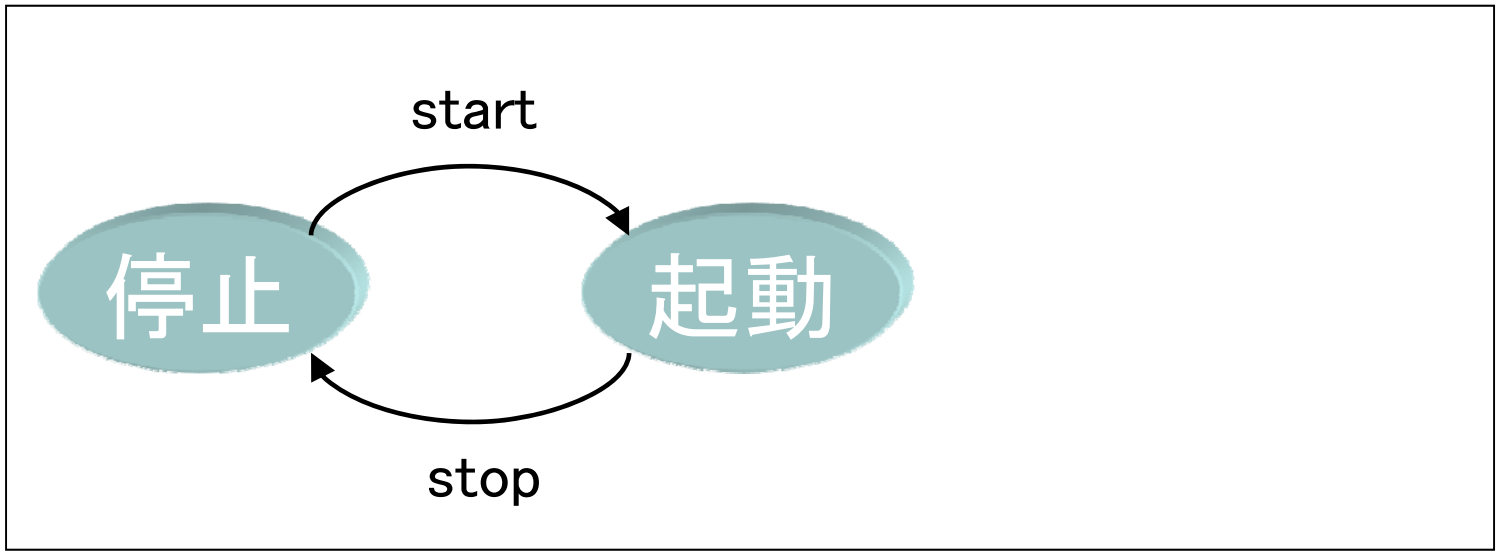
N+1構成
N+M構成

N+1構成
N+M構成
複数台のサーバでサービスが動作
故障が発生するとSBY(待機系)で
サービス起動



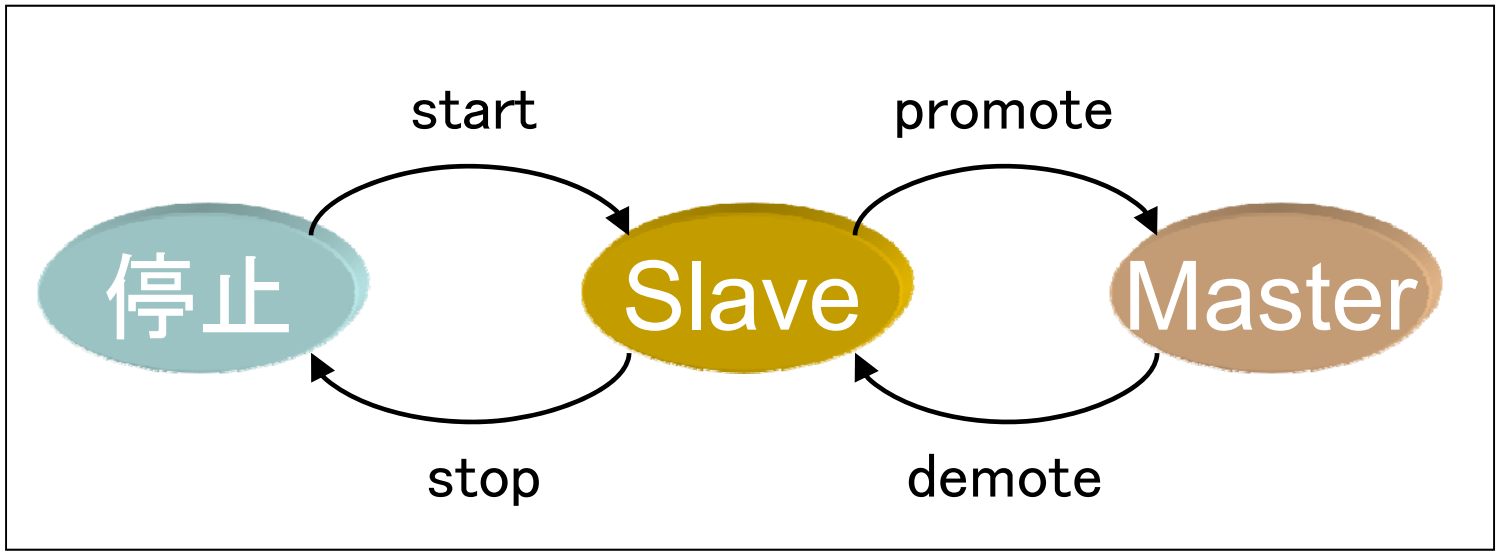
M/S構成
Multi State (Master/Slave)
稼動系と連携したサービスが待機系でも動作する構成

通常のリソースの状態遷移

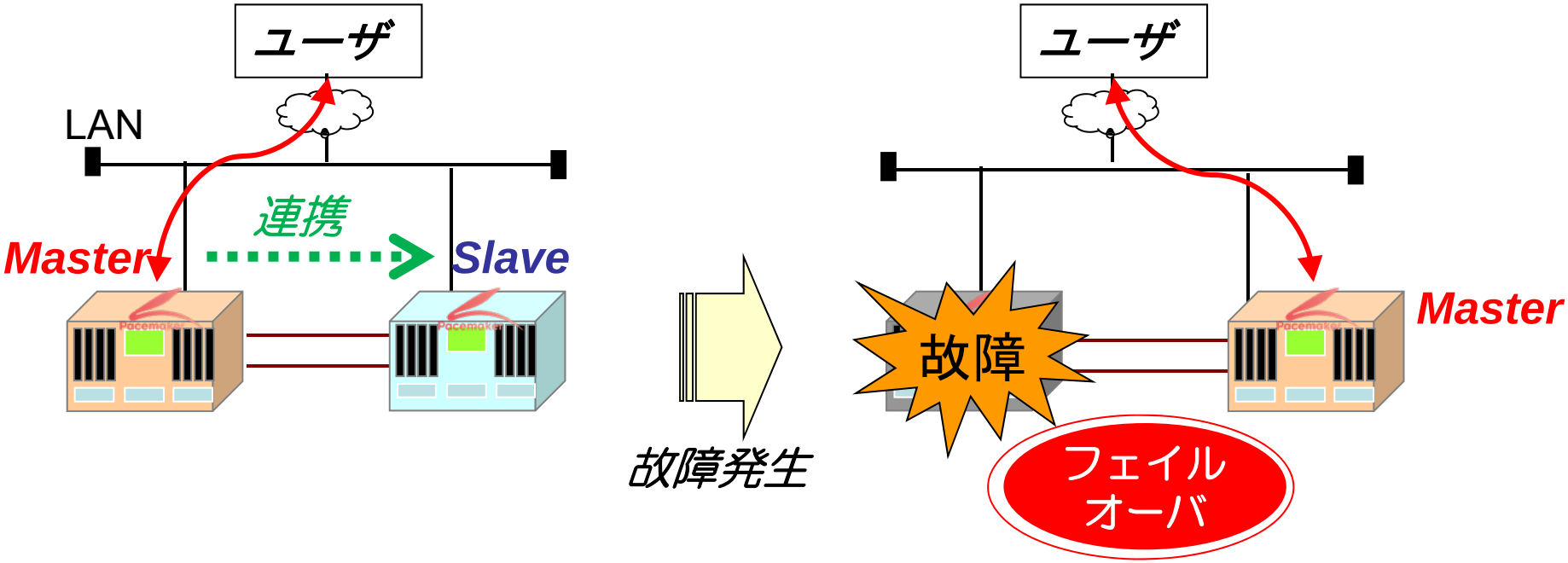


M/S構成
Multi State (Master/Slave)
稼動系と連携したサービスが待機系でも動作する構成

M/Sのリソースの状態遷移



M/S構成
M/S機能を持っているサービス
DRBD ディスク同期
mysql データベースレプリケーション



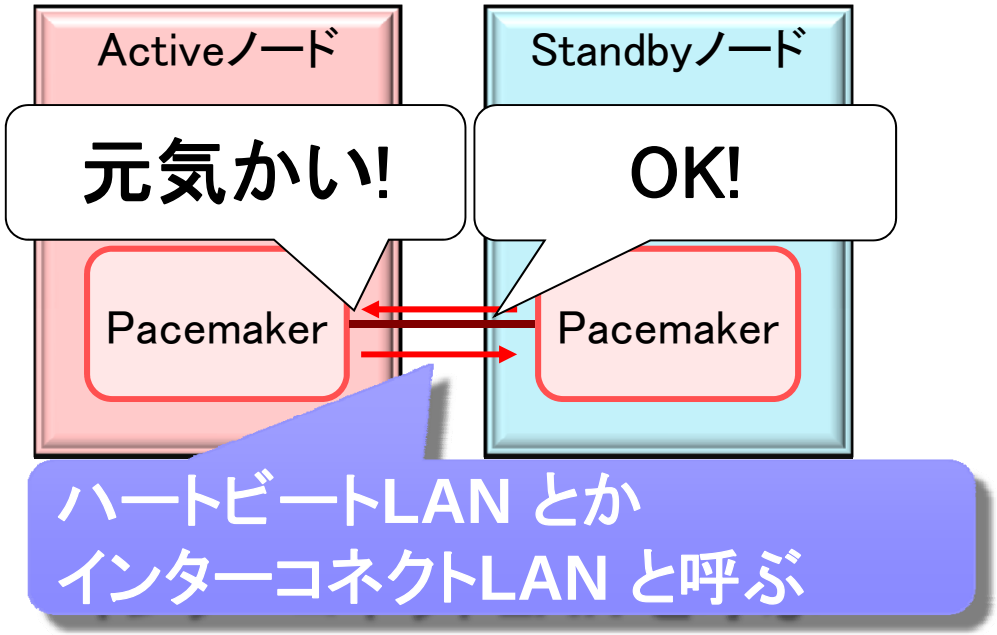
Pacemakerの基本動作

Pacemakerの基本動作

1. ノード監視
2. リソース監視
3. スプリットブレイン対策

基本動作1:ノード監視

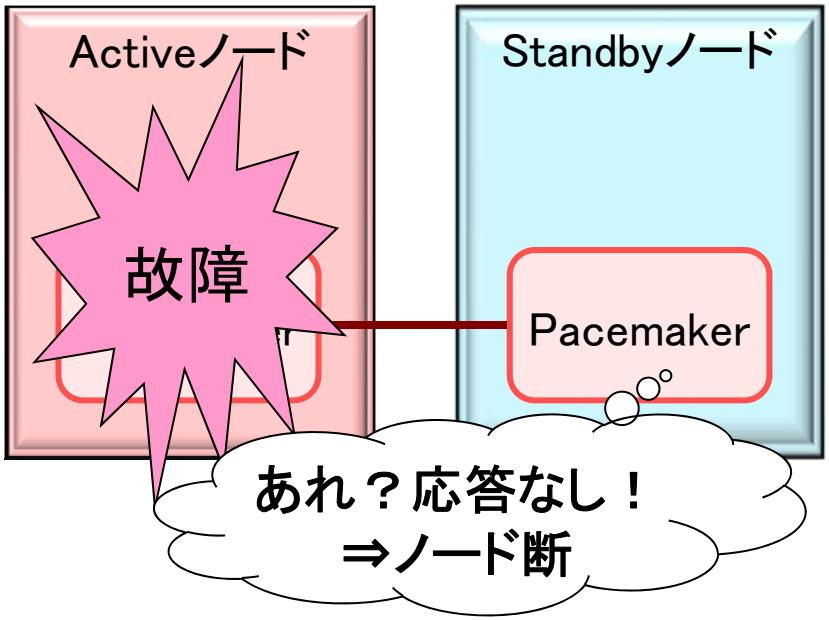
基本動作1:ノード監視
相手サーバの生死を確認するため、一定間隔で通信(ハートビート通信)



基本動作1:ノード監視

ハートビート通信に失敗すると、相手はダウンしたと判断

フェイルオーバなどのクラスタ制御を行う



基本動作2:リソース制御

基本動作2:リソース制御
リソースって？

基本動作2:リソース制御

リソース = クラスタが管理するものすべて

基本動作2:リソース制御

リソース = クラスタが管理するものすべて

大きくは2つ

1つめ
サービス継続するのに必要なもの

1つめ
サービス継続するのに必要なもの

- サーバプログラム
- コンピュータ資源
    * 仮想IPアドレス
    * ファイルシステム

2つめ
故障を検知するために監視が必要なもの

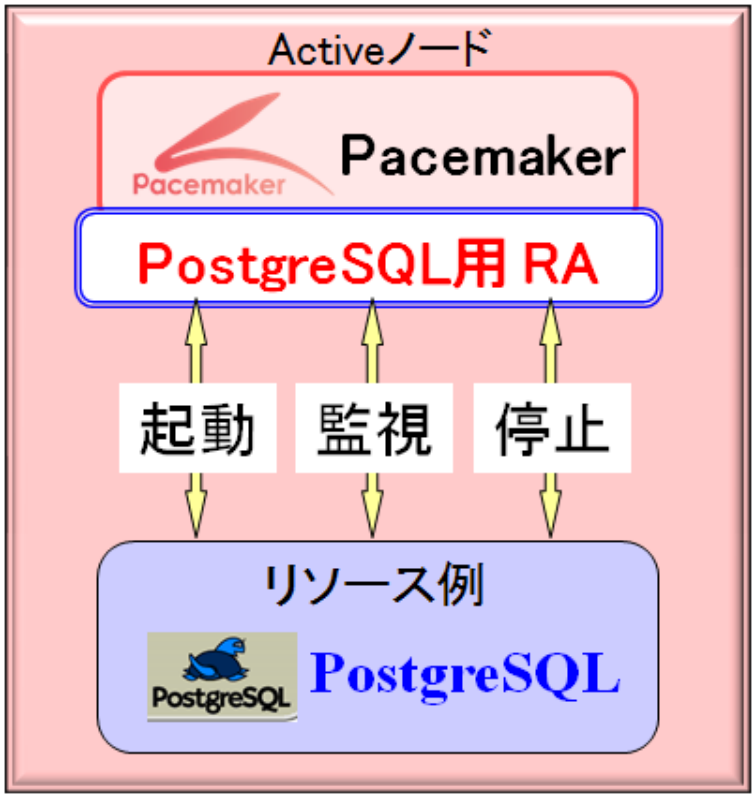
2つめ
故障を検知するために監視が必要なもの

- ネットワーク経路監視
- ディスク監視

リソース制御

リソース制御

サービスを提供するために、
Pacemakerがリソースを
起動(start)、停止(stop)、監視
(monitor)すること



リソースエージェント(RA)

リソースエージェント(RA)
Pacemakerがリソース制御するために利用するスクリプト

あらかじめ含まれる標準RA

74個

```
# crm ra list ocf
AoEtarget       AudibleAlarm    CTDB            ClusterMon      Delay
Dummy           EvmsSCC         Evmsd           Filesystem      HealthCPU
HealthSMART     ICP             IPaddr          IPaddr2         IPsrcaddr
IPv6addr        LVM             LinuxSCSI       MailTo          ManageRAID
ManageVE        NVclient        Pure-FTPd       Raid1           Route
SAPDatabase     SAPInstance     SendArp         ServeRAID       SphinxSearchDaemon
Squid           Stateful        SysInfo         SystemHealth    VIPArip
VIPcheck        VirtualDomain   WAS             WAS6            WinPopup
Xen             Xinetd          anything        apache          controld
db2             diskd           drbd            eDir88          exportfs
fio             iSCSILogicalUnit iSCSITarget    ids             iscsi
ldirectord      mysql           mysql-proxy     nfsserver       o2cb
oracle          oralsnr         pgsql           ping            pingd
portblock       postfix         proftpd         rsyncd          scsi2reservation
sfex            syslog-ng       tomcat          vmware
```

あらかじめ含まれる標準RA

目的	リソース	リソースエージェント名 (/usr/lib/ocf/resource.d/ に存在)
サーバプログラム	データベース インターネットサーバ	pgsql, oracle, oralsnr, mysql apache, tomcat, jboss, postfix
コンピュータ資源	ファイルシステム	Filesystem (複数のファイルシステムに対応)
	仮想IPアドレス	IPaddr2, IPv6addr
異常監視	ネットワーク経路監視 ディスク監視 共有ディスク排他 仮想IPアドレス排他	pingd diskd (Linux-HA Japan提供) sfex VIPcheck (Linux-HA Japan提供)

リソースエージェント(RA)
クラスとプロバイダで分類

リソースエージェント(RA)
クラス = RAの準拠している仕様

lsb と ocf の2つ

リソースエージェント(RA)
クラス = RAの準拠している仕様

lsb: LSB仕様のinitscript形式
/etc/init.d/* にあるスクリプトを利用

ただし、リターンコードを正しく返却されていることが条件

リソースエージェント(RA)
クラス = RAの準拠している仕様

ocf: Open Clustering Framework
　lsbを拡張し、RAへの引数やコマンドを追加
　Pacemakerの機能をフルに使うのはこっち

リソースエージェント(RA)
プロバイダ = RAの提供元

heartbeat: Linux-HAプロジェクトが提供
pacemaker: Pacemakerが提供

独自のRAを作るときは、専用のプロバイダを作るとよい

リソースエージェント実装例

PostgreSQL(pgsql RA)監視(monitor)処理の抜粋

```
pgsql_monitor() {                                    ←PostgreSQLの監視のメイン関数
    ......
    if ! pgsql_status                                ←PostgreSQLプロセスの存在を確認
    then                                             ←PostgreSQLプロセスがいなければ
        ocf_log info "PostgreSQL is down"
        return $OCF_NOT_RUNNING                      ←PostgreSQLは停止していると判断
    fi
    ......
   runasowner -q $loglevel  "$OCF_RESKEY_psql $psql_options -c
 '$OCF_RESKEY_monitor_sql' "
            ↑実際にSQL(select now())を実行してPostgreSQLの正常性を確認
    ......
    return $OCF_SUCCESS                              ←PostgreSQLは動作していると判断
}
```

$OCF_SUCCESS, $OCF_NOT_RUNNINGはPacemakerで定義済みの変数

リソースエージェントは自作可能

```
#!/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で実装できます。

いくつか必須のパラメータ呼び出しに対する処理と、定義済みの戻り値を返すように実装する必要があります。

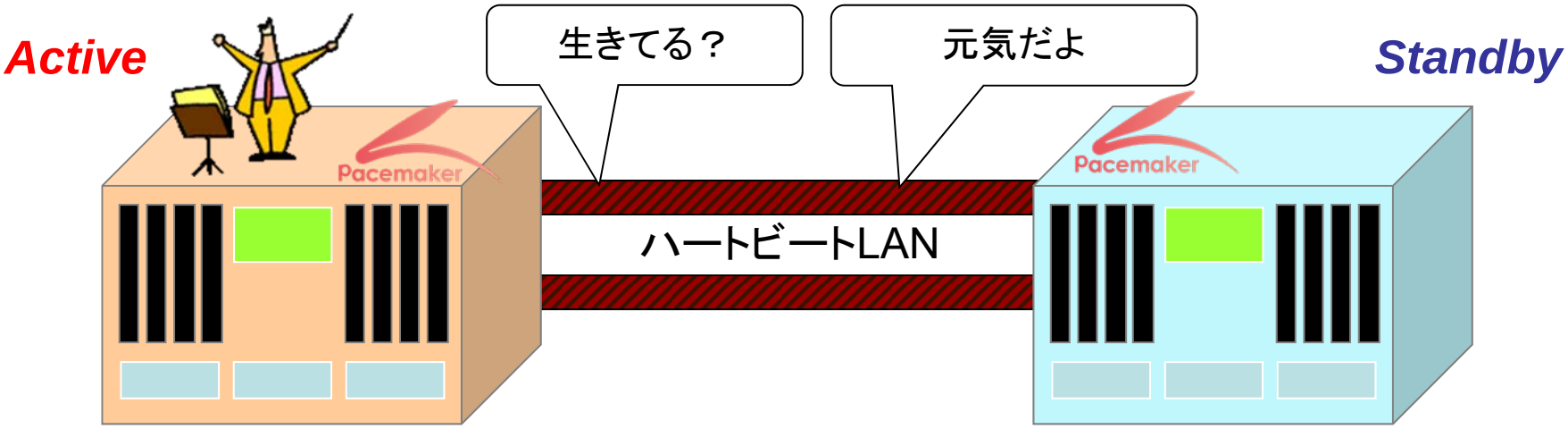
リソース開始・監視・停止の処理

シェルに渡されるパラメータを元にRA処理を振り分け

さらに、HAクラスタとして重要な機能

基本機能3:スプリットブレイン対策

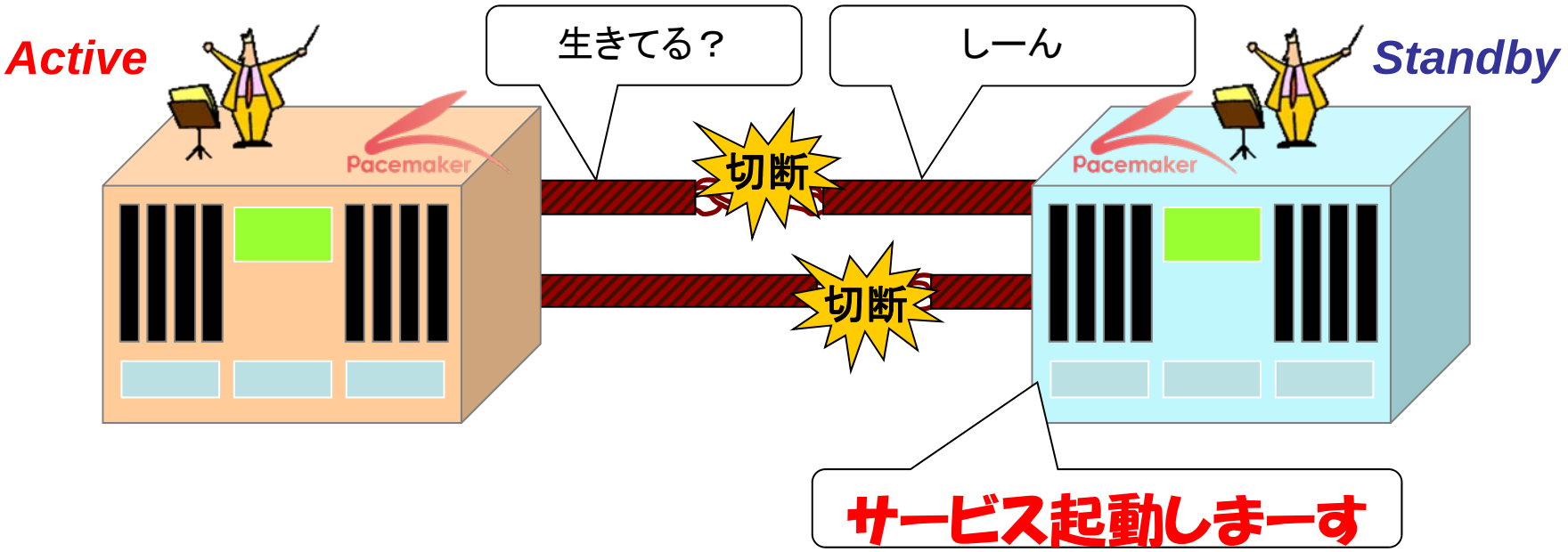
全てのハートビートLANが切れてしまった場合



基本機能3：スプリットブレイン対策

全てのハートビートLANが切れてしまった場合

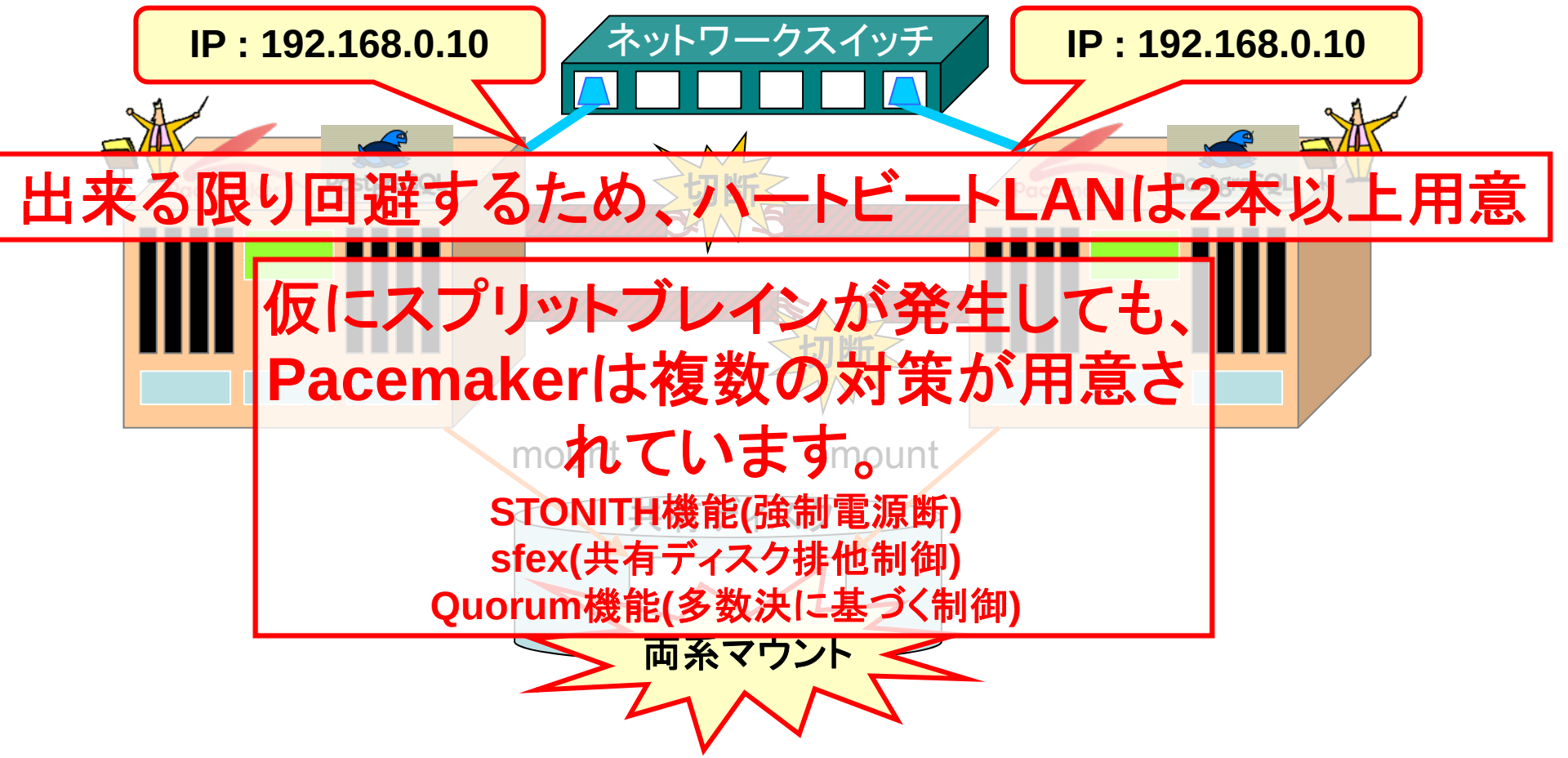
お互いが相手が故障したと判断し、サービスを引き継ごうとします。これを**スプリットブレイン**と呼びます。



両サーバが勝手に動き始めると

(例えば) データを共有していると → **データ破壊発生**

IPを共有していると → **IP競合発生**



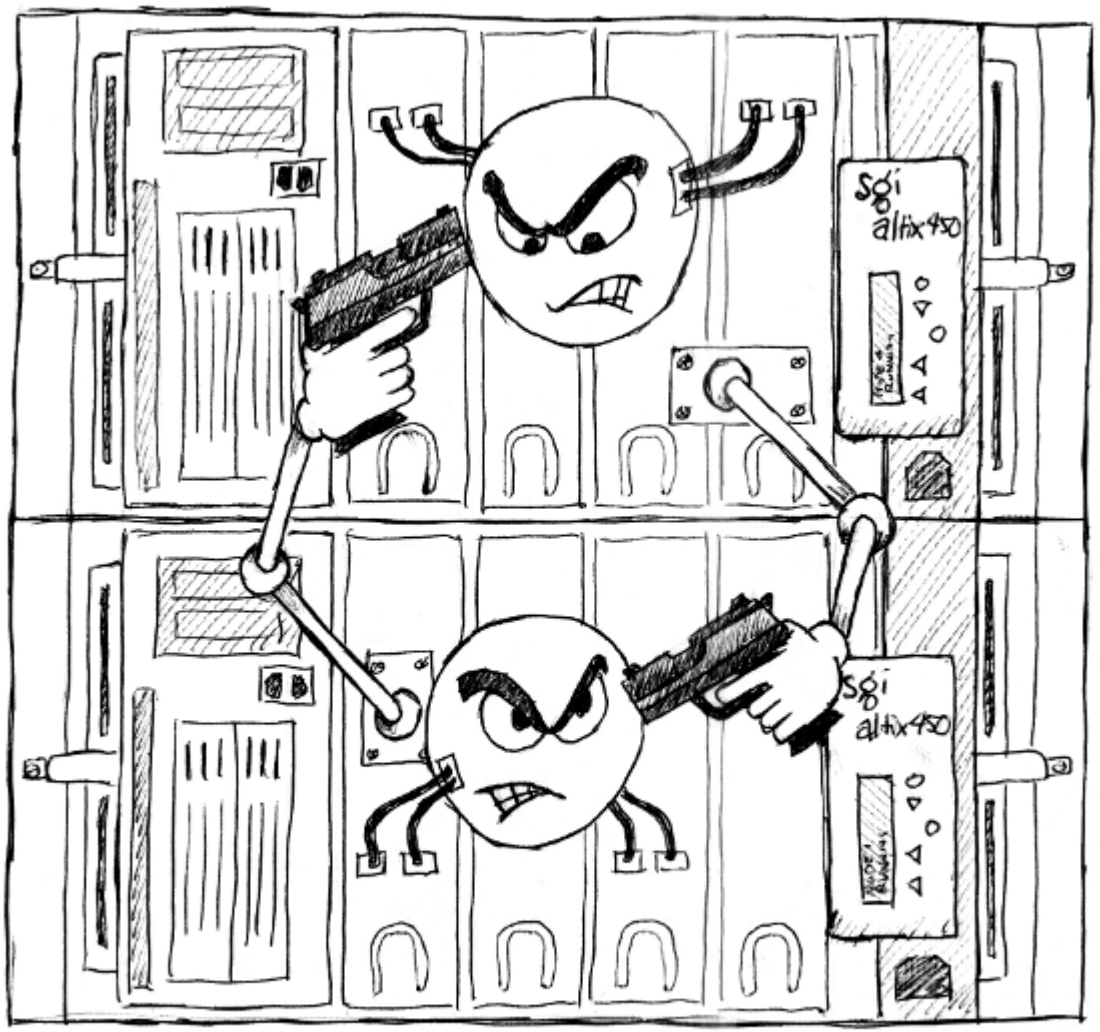
対策1:STONITH

STONITH

Shoot-The-Other-Node-In-The-Head

絵で表現すると

STONITH



http://ourobengr.com/ha

STONITH
サービス継続を邪魔するサーバをクラスタから「強制的に離脱」させる機能
用語的には、ノードフェンシング

STONITH
いつ発動される？

STONITH
発動タイミング

- スプリットブレイン発生時
- リソースの停止処理に失敗したとき

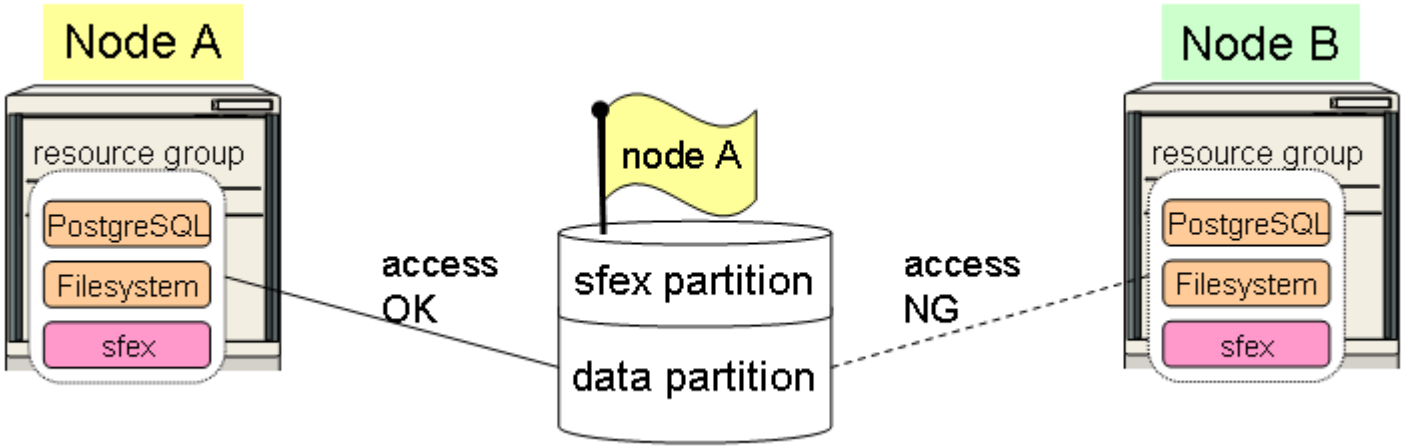
STONITHデバイス
実現方法によりさまざまなSTONITHプラグインが用意されている

- サーバ搭載のHW制御ボード
- リモートパワースイッチ
- UPS
- poweroffコマンド
- 保守者による手動リセット

対策2：sfex

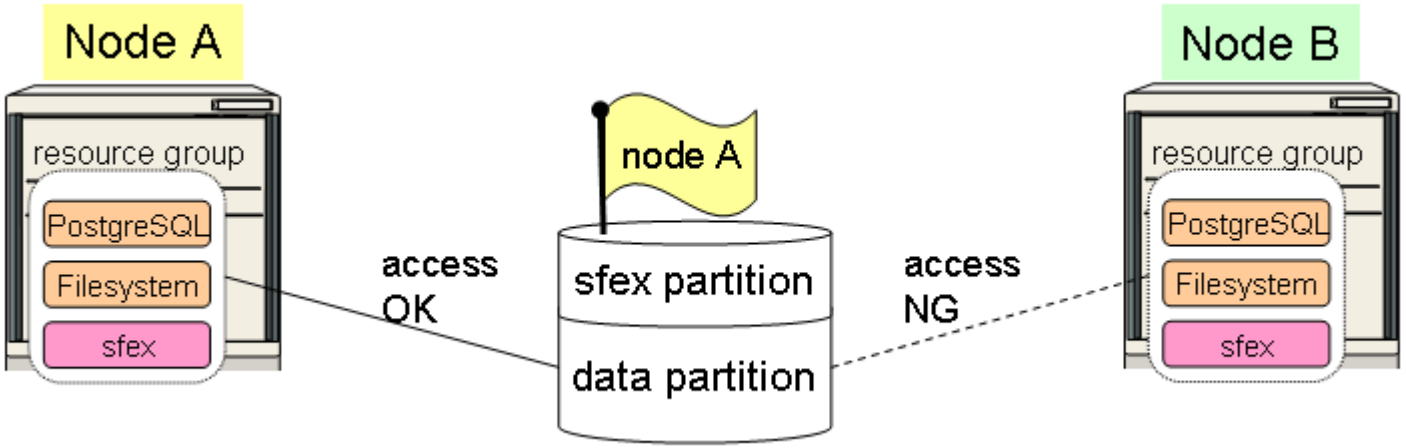
sfex
共有ディスク排他制御機能
絵で表現すると

sfex
共有ディスク排他制御機能



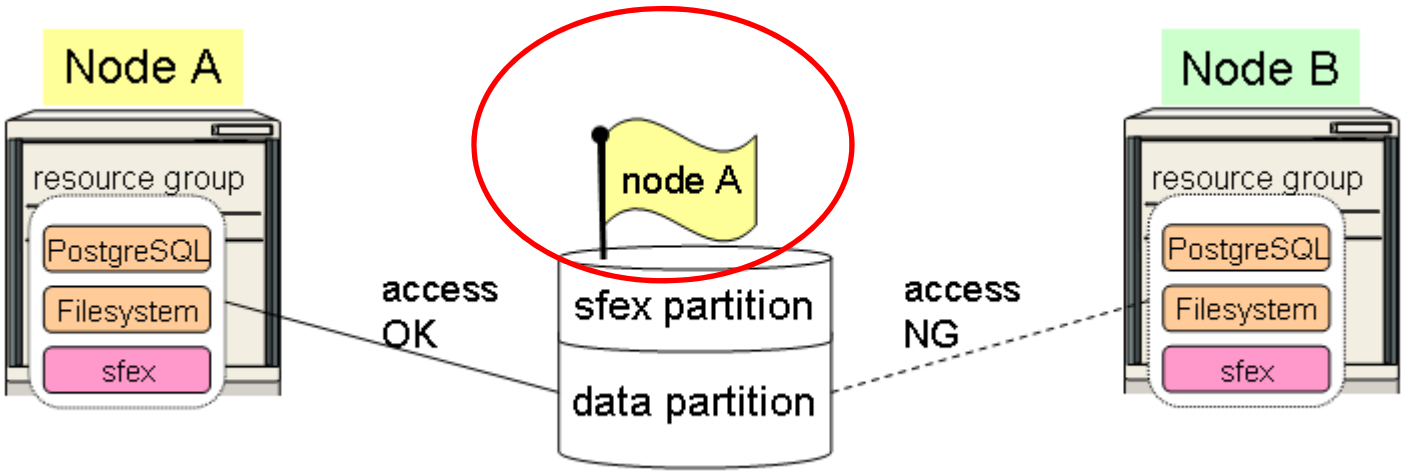
http://sourceforge.jp/projects/linux-ha/wiki/hb-sfex

sfex
共有ディスク排他制御機能
意図しない両系マウントによるファイルシステム破壊を防ぐ

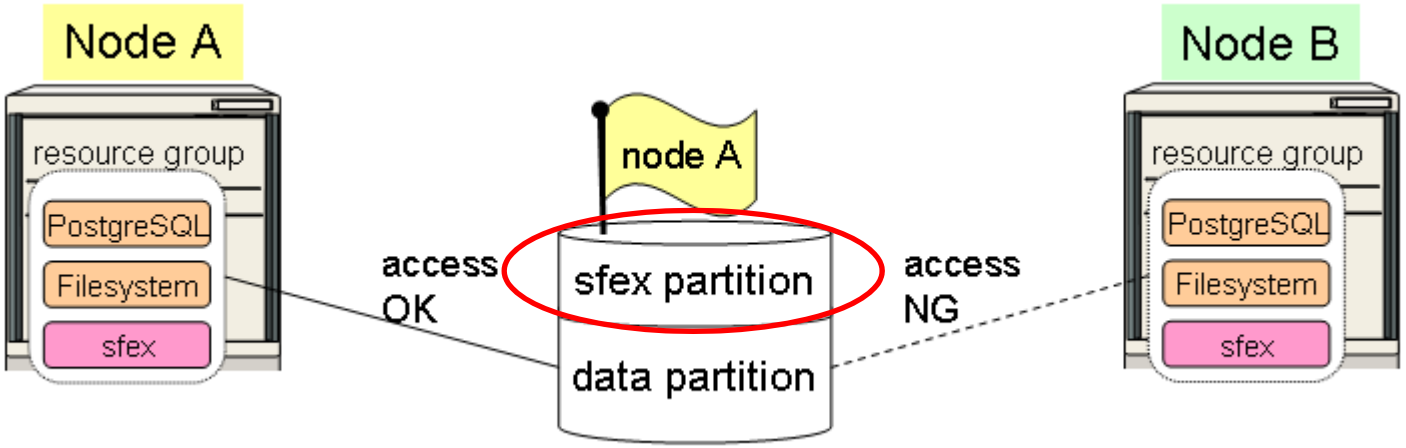


http://sourceforge.jp/projects/linux-ha/wiki/hb-sfex

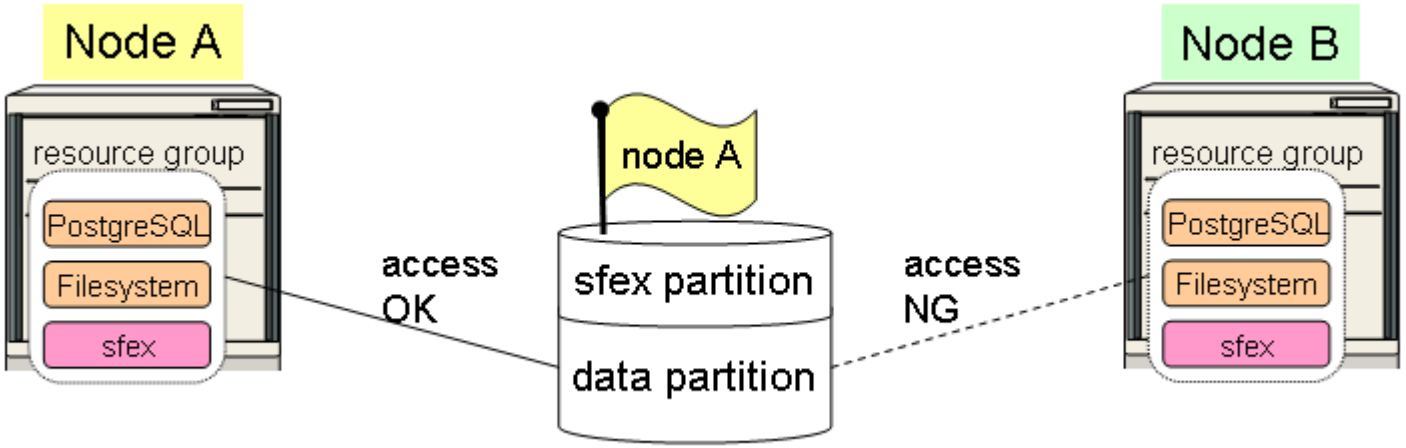
sfex
共有ディスクの所有権を制御するリソース



sfex
共有ディスク上に専用パーティションを用意し、所有者を管理



sfex
ハードウェア依存性が小さいことが特徴



その他の対策

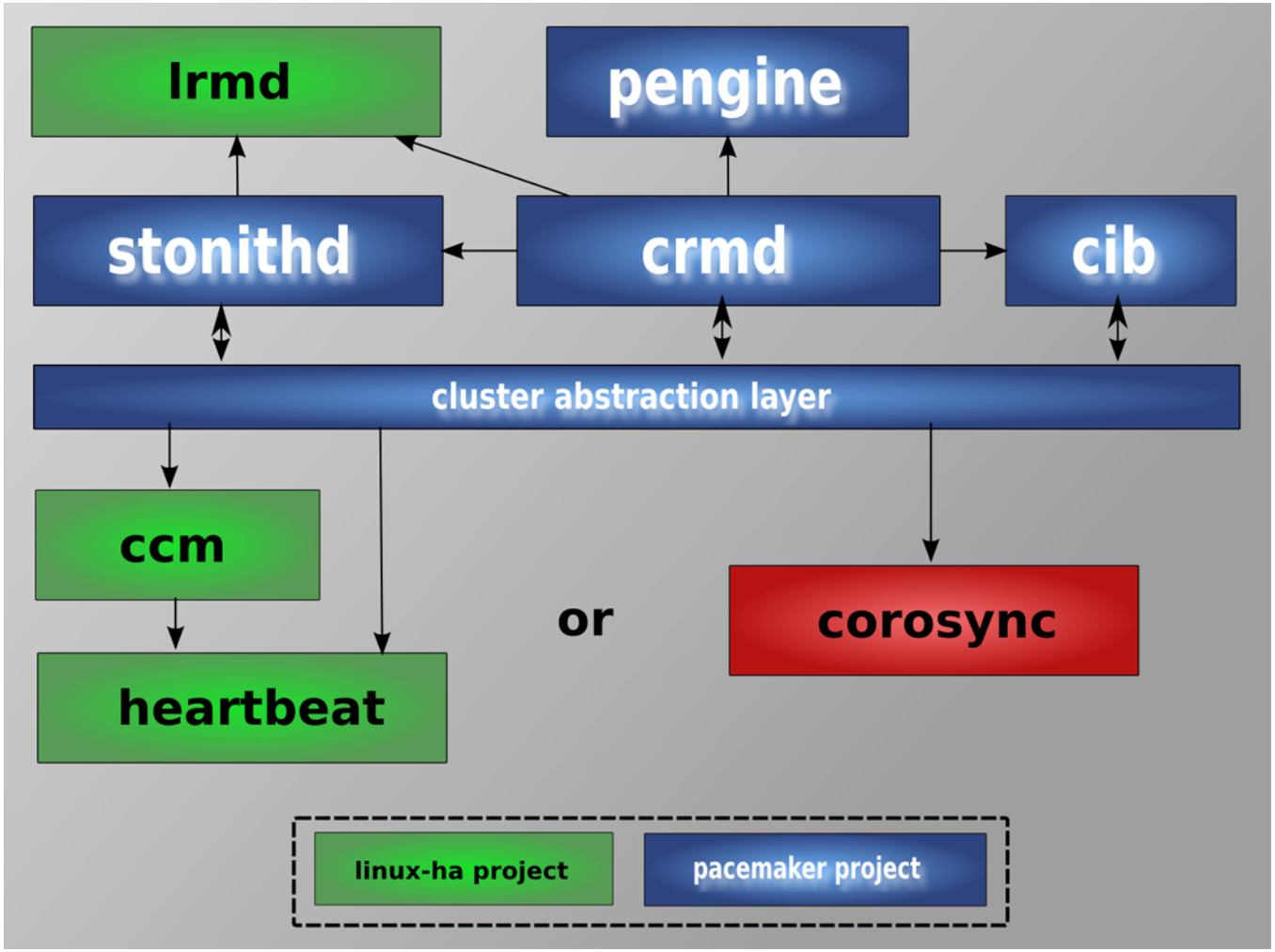
Quorum：ノード数に基づくリソース制御(3ノード以上)

VIPcheck: サービス用仮想IPアドレスに基づくリソース制御

などがあります

Pacemakerを構成するコンポーネントを見ていきます

Pacemakerを構成するコンポーネント



Pacemakerを構成するコンポーネント
役割分担
Pacemaker：リソース制御
Heartbeat or Corosync：クラスタ制御

Pacemakerプロセス

- crmd: Pacemakerのメインプロセス
- cib: クラスタに関する情報を一元管理
- pengine: クラスタ状態に基づきリソース配置を決定し状態遷移を計算
- stonithd: STONITHプラグインの管理

Heartbeatプロセス

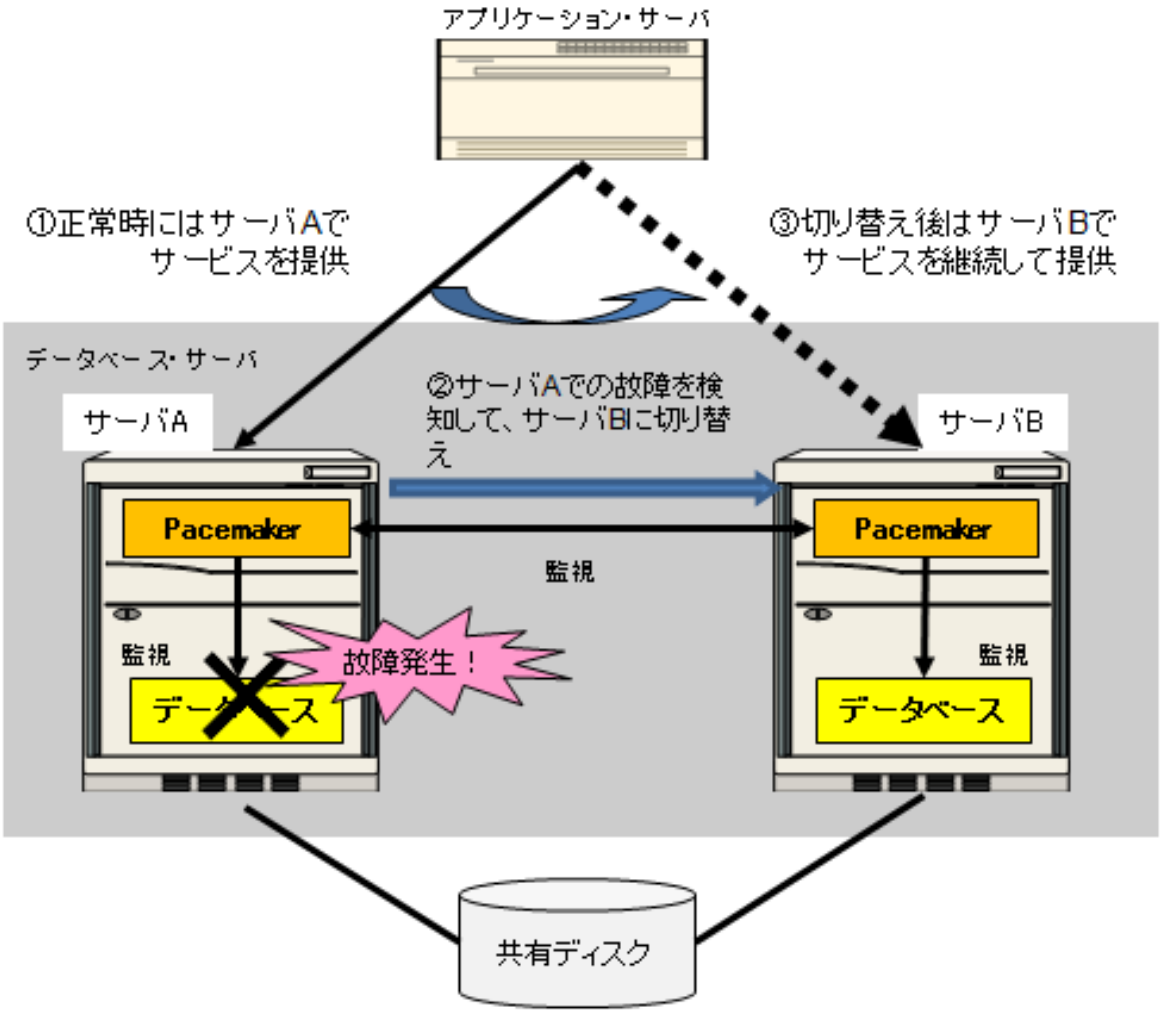
- ccm: メンバーシップ管理
- heartbeat: プロセス間通信、サブプロセス管理

Corosyncプロセス

- corosync: メンバーシップ管理、ノード間通信、サブプロセス管理

いよいよ、HAクラスタをどう構築するかを見ていきます

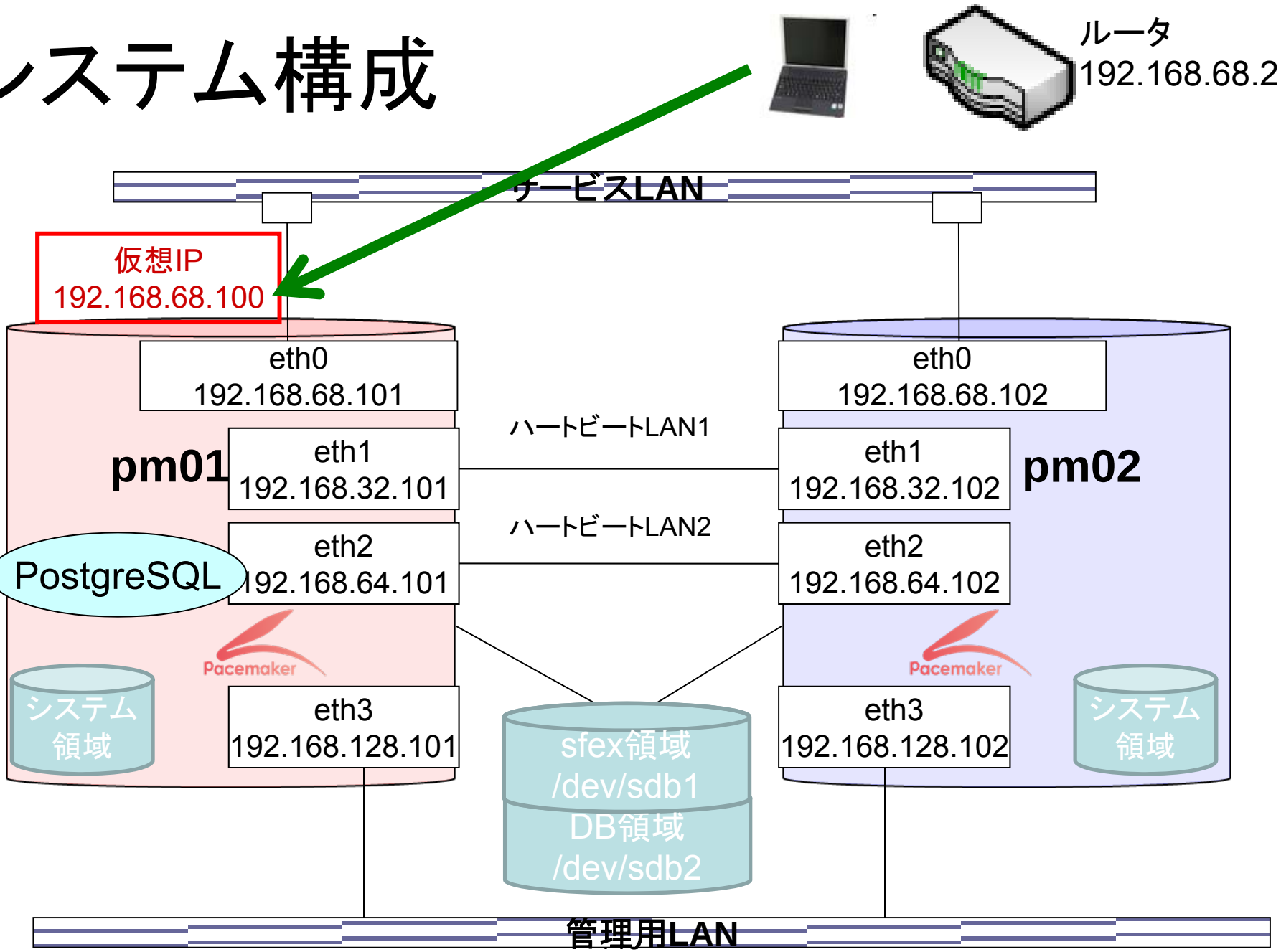
PostgreSQLのHAクラスタ構成



PostgreSQLのHAクラスタ構成

- ハードウェア
  - VMware VM (1CPU, 1GB memory, 8GB HDD) 2個
- OS
  - CentOS 5.5 x86_64
- HAクラスタ
  - Pacemaker-1.0.10
- クラスタ化するアプリケーション
  - PostgreSQL 9.0.3
  - ACT-SBY構成
- 共有ディスク
  - iSCSI
  - データベース領域として利用
- NIC
  - 4つ使用(サービスLAN、ハートビートLAN×2、管理LAN)

システム構成


ルータ
192.168.68.2
サービスLAN
仮想IP
192.168.68.100
eth0
192.168.68.101
eth0
192.168.68.102
ハートビートLAN1
pm01
eth1
192.168.32.101
eth1
192.168.32.102
pm02
ハートビートLAN2
PostgreSQL
eth2
192.168.64.101
eth2
192.168.64.102
Pacemaker
Pacemaker
システム
領域
eth3
192.168.128.101
sfex領域
/dev/sdb1
DB領域
/dev/sdb2
eth3
192.168.128.102
システム
領域
管理用LAN

リソースの洗い出し

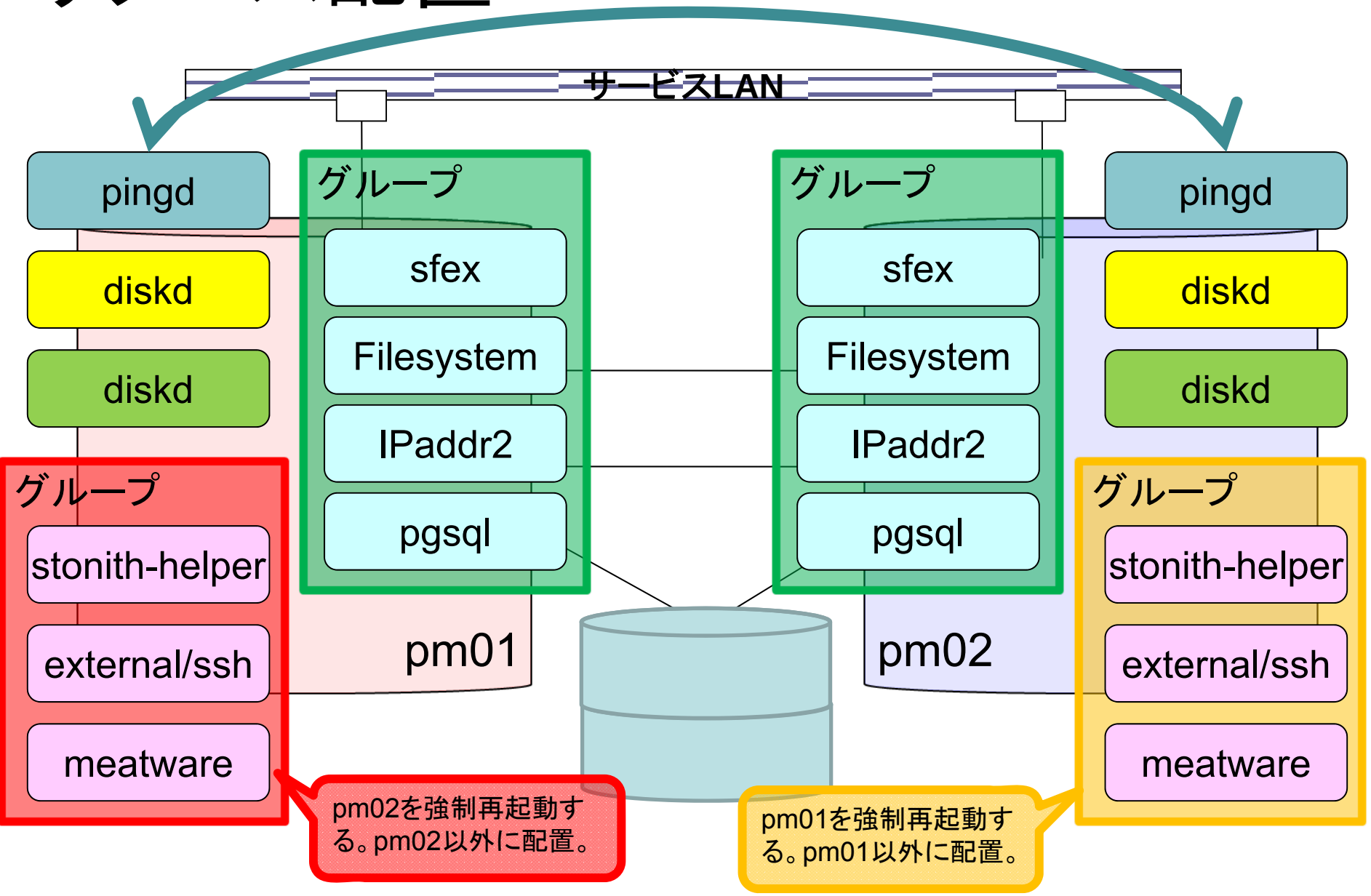
種別	リソース	RA / STONITH Plugin
フェイルオーバ対象 (稼動系で起動)	共有ディスク排他制御 共有ディスクのマウント 仮想IPアドレス PostgreSQL	sfex Filesystem IPaddr2 pgsql
故障検知 (すべてのノードで起動)	ネットワーク経路監視 内蔵ディスク監視 共有ディスク監視	pingd diskd diskd
STONITH (STONITH対象以外のノードで起動)	相打ち防止 sshプラグイン(※) 保守者介在	stonith-helper external/ssh meatware

日本コミュニティ提供ツールで出現します

※テスト用のプラグイン。cluster-glue-libs-develパッケージに含まれるので、別途インストールしてください

ただし、実運用では使用しないこと

リソース配置


クローン
サービスLAN
pingd
diskd
diskd
グループ
stonith-helper
external/ssh
meatware
pm01
グループ
sfex
Filesystem
IPaddr2
pgsql
グループ
sfex
Filesystem
IPaddr2
pgsql
pm02
pingd
diskd
diskd
グループ
stonith-helper
external/ssh
meatware
pm02を強制再起動する。pm02以外に配置。
pm01を強制再起動する。pm01以外に配置。

リソース設定の種類

リソース設定の種類

primitive

clone

group

リソース設定の種類

primitive

リソース設定の種類

primitive

すべてのリソース設定の基本

RAはまずprimitive設定することからはじまる

リソース設定の種類

clone

リソース設定の種類

clone

同じ設定のリソースを複数のノードで動かしたい場合に使用

ネットワーク経路監視やディスク監視で使用

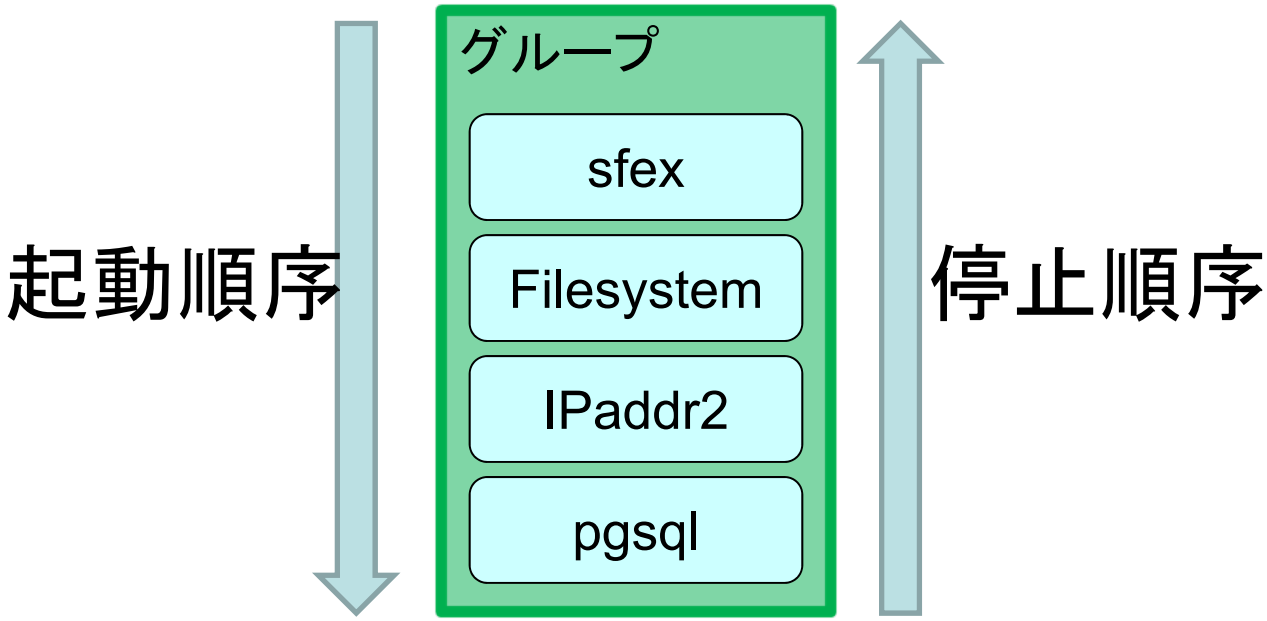
リソース設定の種類

group

リソース設定の種類

group

複数のリソースをまとめてフェイルオーバさせるために使用

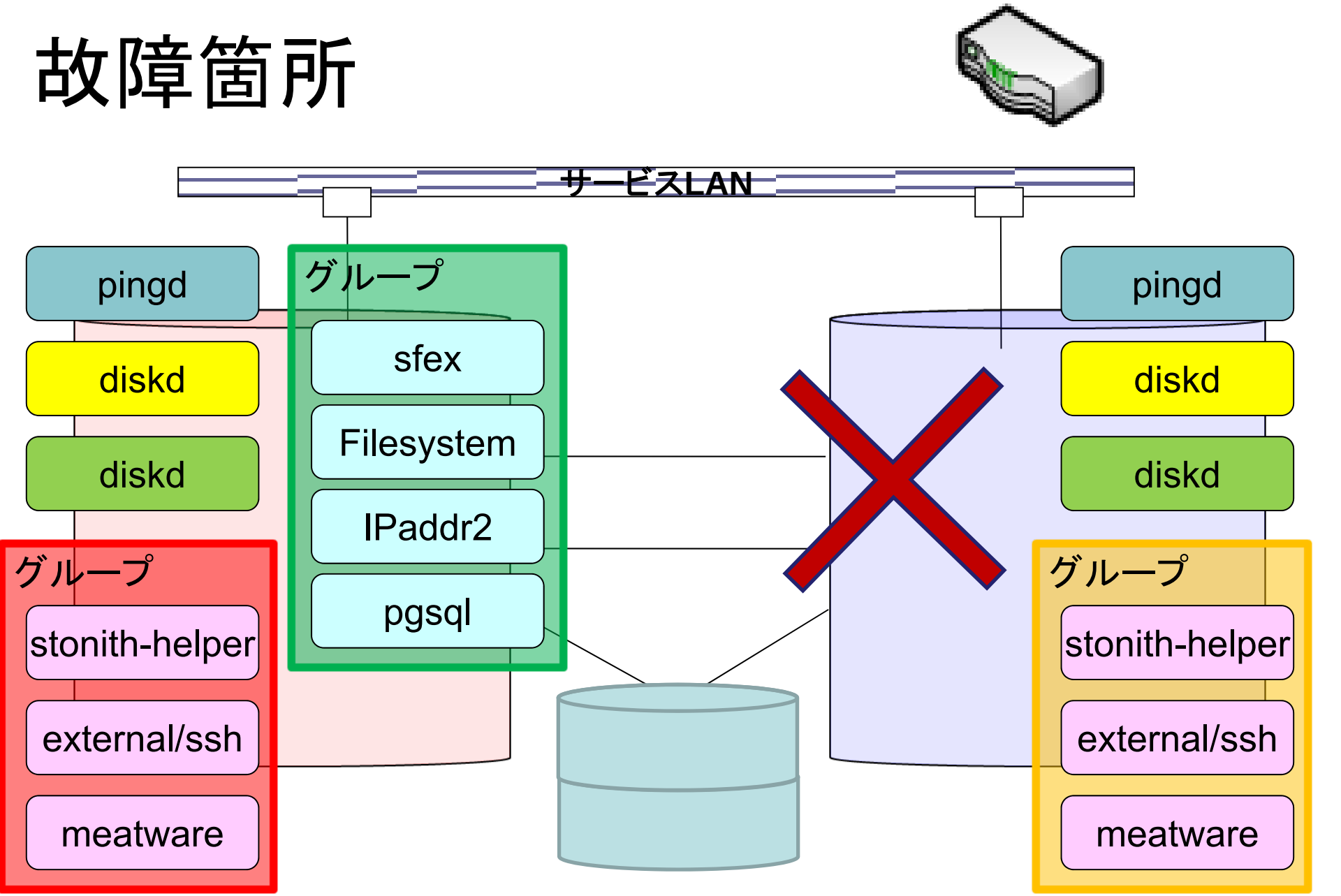


フェイルオーバ条件

フェイルオーバ条件
ノード監視

フェイルオーバ条件
ノード監視
ハートビート通信が不通

故障箇所

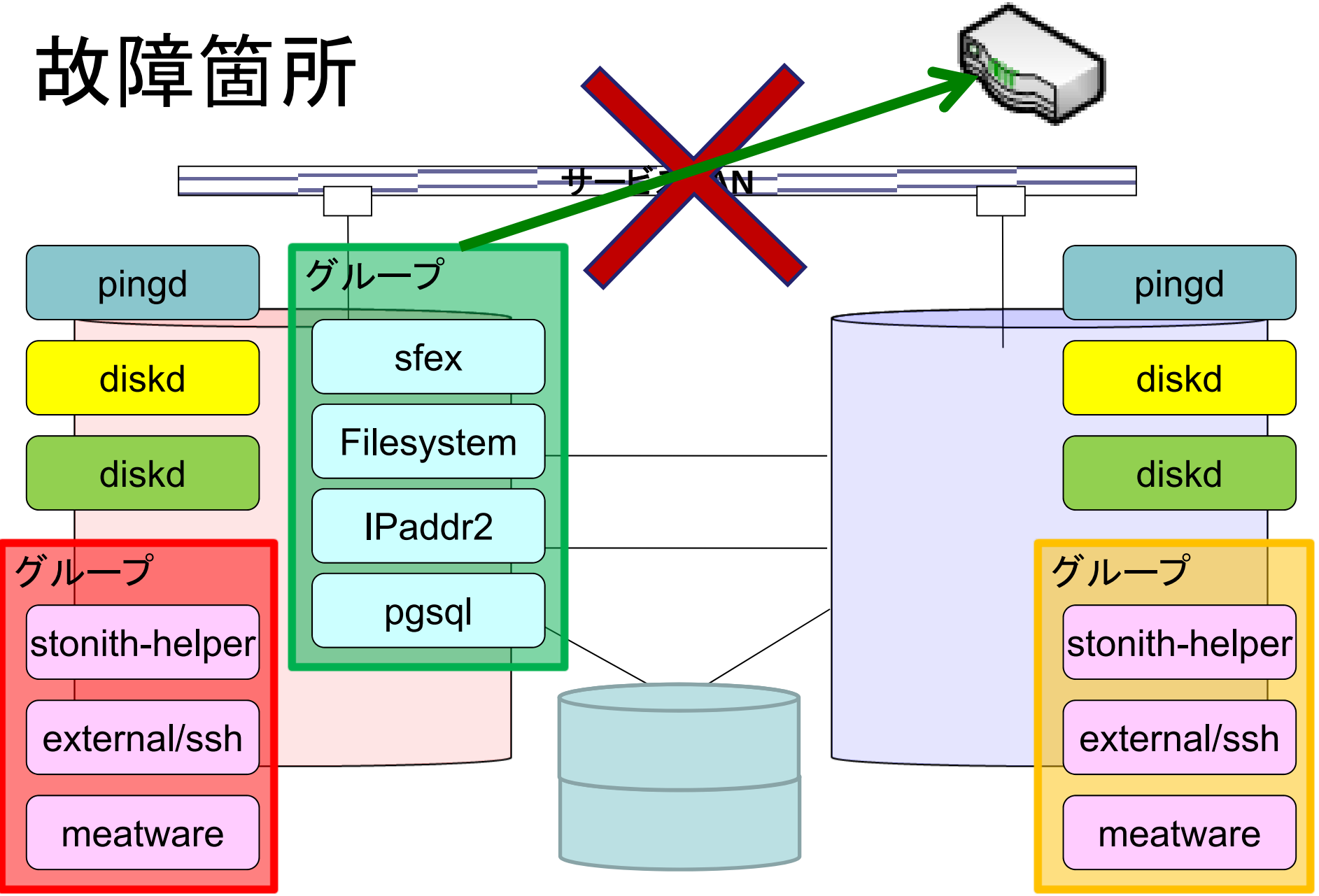

サービスLAN
pingd
diskd
diskd
グループ
stonith-helper
external/ssh
meatware
グループ
sfex
Filesystem
IPaddr2
pgsql
pingd
diskd
diskd
グループ
stonith-helper
external/ssh
meatware

フェイルオーバ条件
ノード監視
ハートビート通信が不通
→ノード故障
→ノードフェンシング

フェイルオーバ条件
ネットワーク経路監視

フェイルオーバ条件
ネットワーク経路監視
指定されたIPアドレスまでネットワーク通信ができない

故障箇所

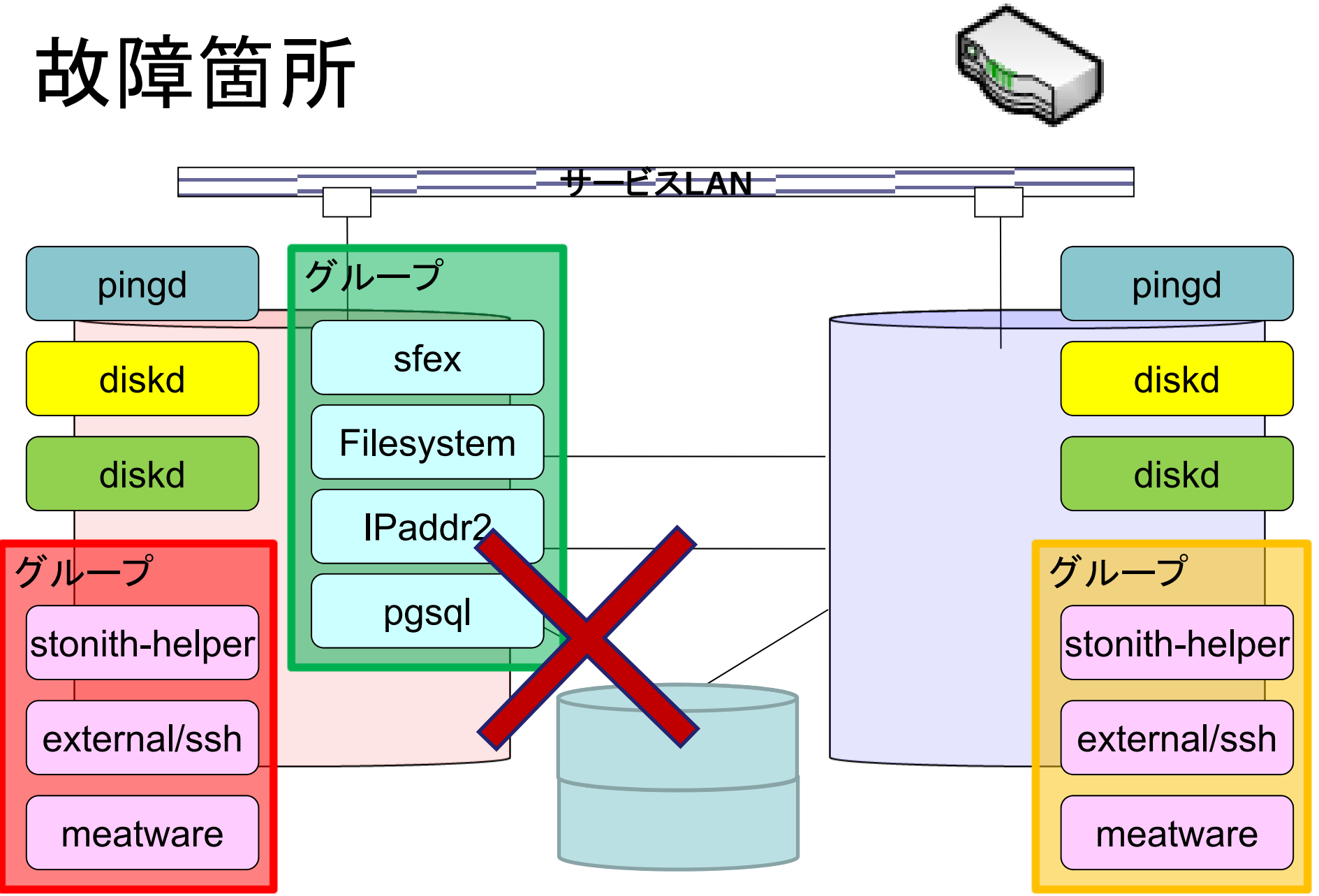

pingd
diskd
diskd
グループ
stonith-helper
external/ssh
meatware
グループ
sfex
Filesystem
IPaddr2
pgsql
pingd
diskd
diskd
グループ
stonith-helper
external/ssh
meatware

フェイルオーバ条件
ネットワーク経路監視
指定されたIPアドレスまでネットワーク通信ができない
→故障が稼動系ならばリソースをフェイルオーバ

フェイルオーバ条件
ディスク監視(sfex)

フェイルオーバ条件
ディスク監視(sfex)
共有ディスクにアクセスできない

故障箇所

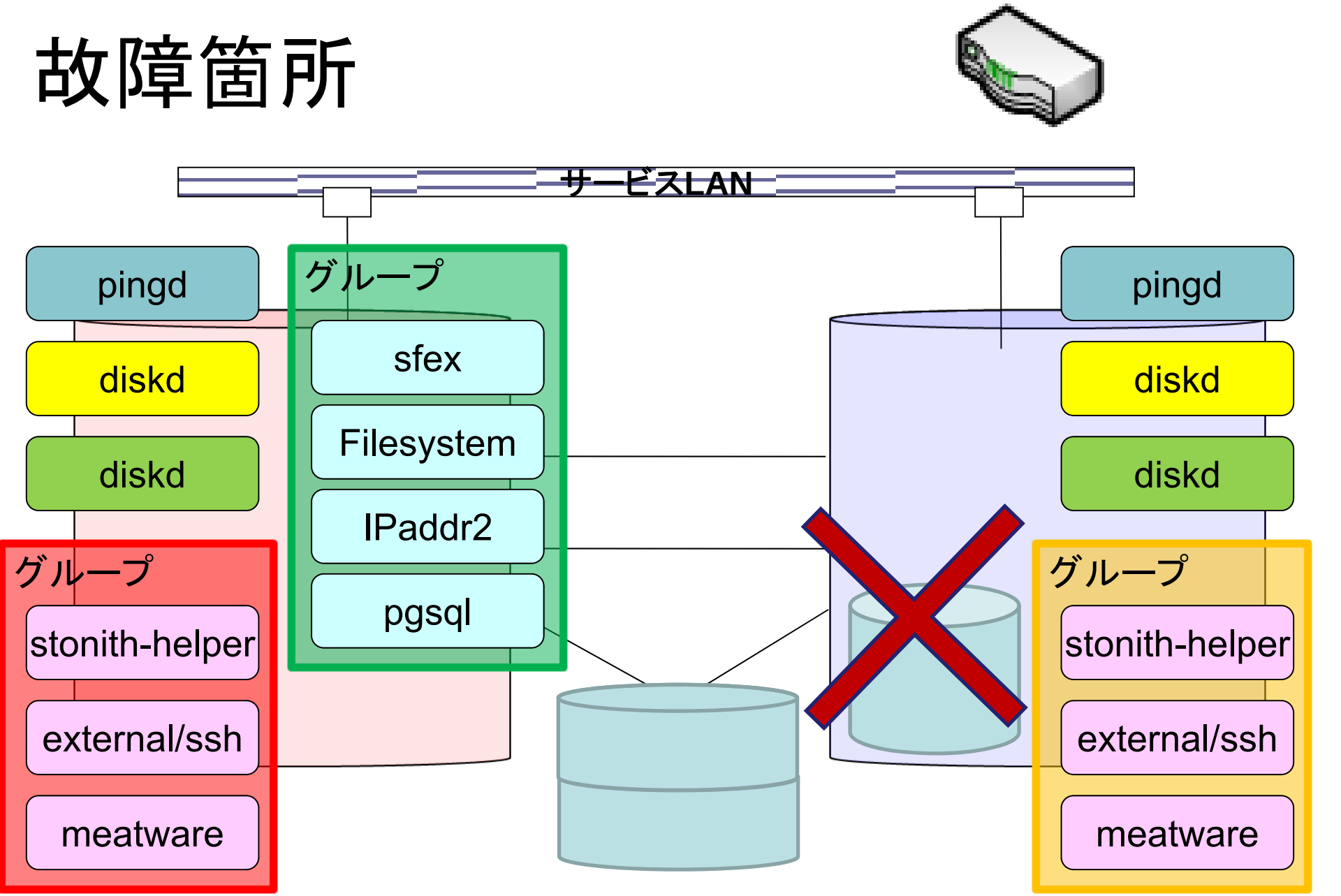

サービスLAN
pingd
diskd
diskd
グループ
stonith-helper
external/ssh
meatware
グループ
sfex
Filesystem
IPaddr2
pgsql
pingd
diskd
diskd
グループ
stonith-helper
external/ssh
meatware

フェイルオーバ条件
ディスク監視(sfex)
共有ディスクにアクセスできない
→リソースをフェイルオーバ

フェイルオーバ条件
ディスク監視(diskd)

フェイルオーバ条件
ディスク監視(diskd)
内蔵ディスクまたは共有ディスクにアクセスできない

故障箇所

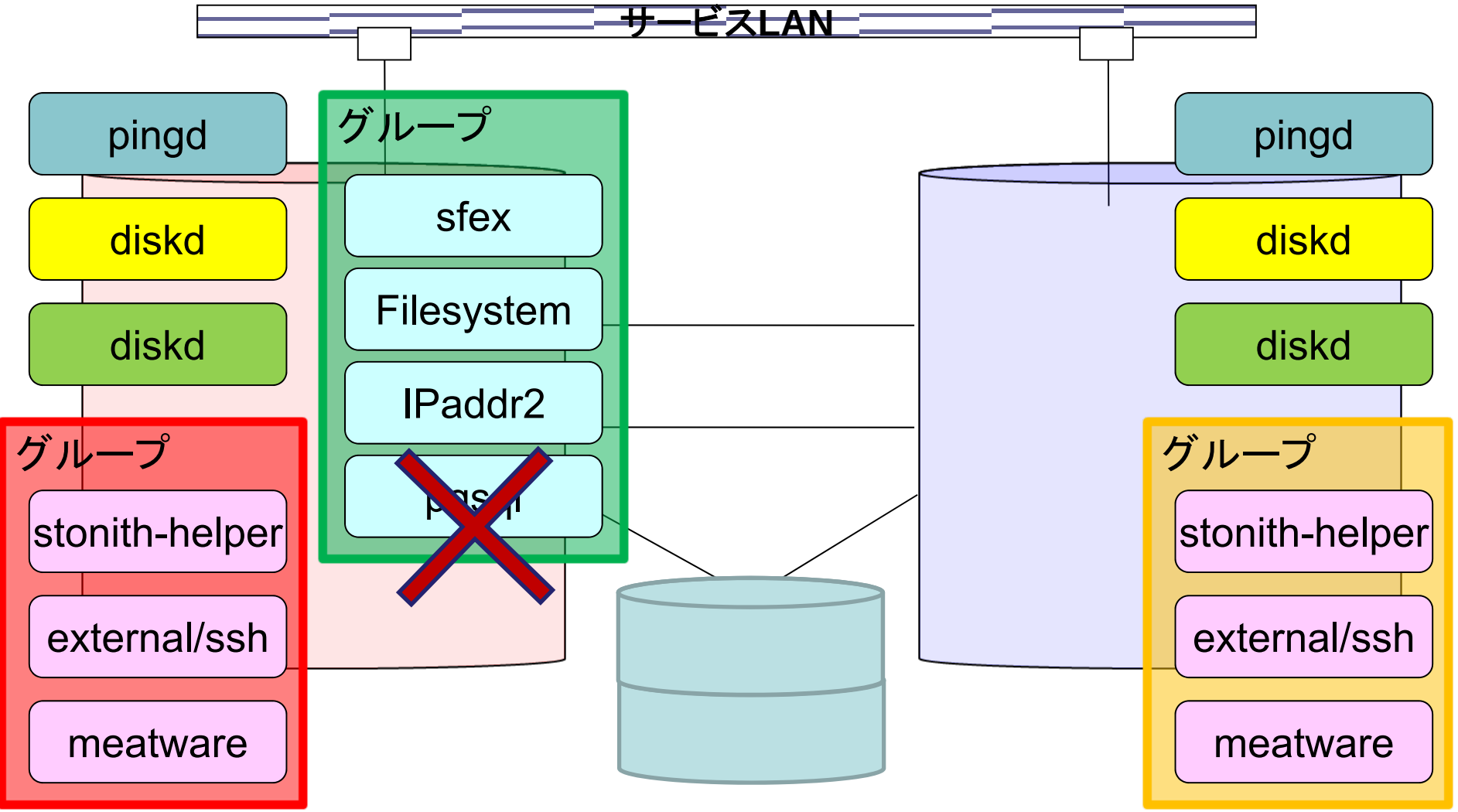

サービスLAN
pingd
diskd
diskd
グループ
stonith-helper
external/ssh
meatware
グループ
sfex
Filesystem
IPaddr2
pgsql
pingd
diskd
diskd
グループ
stonith-helper
external/ssh
meatware

フェイルオーバ条件
ディスク監視(diskd)
内蔵ディスクまたは共有ディスクにアクセスできない
→故障が稼動系ならばリソースをフェイルオーバ

フェイルオーバ条件
リソース故障

フェイルオーバ条件
リソース故障
PostgreSQLのmonitor処理に失敗

故障箇所


サービスLAN
pingd
グループ
sfex
Filesystem
IPaddr2
pgsql
diskd
diskd
pingd
diskd
diskd
グループ
stonith-helper
external/ssh
meatware
グループ
stonith-helper
external/ssh
meatware

フェイルオーバ条件
リソース故障
PostgreSQLのmonitor処理に失敗
→リソースをフェイルオーバ

まずインストール

インストールの方法
方法1. 本家(clusterlabs)のyumリポジトリを使用
    * 別のyumリポジトリ(EPEL)も必要

インストールの方法
方法2. 日本コミュニティのリポジトリパッケージ

- yumリポジトリのアーカイブ
- ローカルリポジトリとして使用
- 検証をパスした組み合わせ
- Linux-HA Japanオリジナルパッケージも含む

こちらを推奨

リポジトリパッケージでインストール

http://sourceforge.jp/projects/linux-ha/ からダウンロード

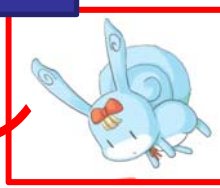
pacemaker-1.0.10-1.4.1.el5.x86_64.repo.tar.gz
をダウンロード
(32bit環境の場合は、pacemaker-1.0.10-1.4.1.el5.i386.repo.tar.gz を選んでください。)

リポジトリパッケージでインストール

/tmp にアーカイブファイルを展開し、yumでインストールします

```
# tar zxvf pacemaker-1.0.10-1.4.1.el5.x86_64.repo.tar.gz -C /tmp
# cd /tmp/pacemaker-1.0.10-1.4.1.el5.x86_64.repo
# yum -c pacemaker.repo install pacemaker heartbeat
```

日本コミュニティ提供ツールをインストール



```
# yum –c pacemaker.repo  pm_extra pm_crmgen pm_diskd pm_logconv-hb
```

テスト用STONITHプラグインをインストール

```
# yum –c pacemaker.repo  cluster-glue-libs-devel
```

Pacemakerの自動起動をオフにする

```
# chkconfig --level 2345 heartbeat off
```

オンの場合、サーバ起動後に自動的にクラスタに組み込まれる

オフの場合、手動でPacemakerを起動

※運用で選択

設定ファイルのたぐい

設定ファイルのたぐい

/etc/ha.d/ha.cf

　Heartbeatの設定

/etc/ha.d/authkeys

　ハートビート通信の認証用キー

/etc/syslog.conf

　ログファイルの分離

/etc/pm_logconv.conf

　ログメッセージ変換

/etc/ha.d/ha.cf

```
pacemaker on
debug 0
udpport 694
keepalive 2
warntime 7
deadtime 10
initdead 10
logfacility local1
bcast eth1
bcast eth2
node pm01
node pm02
watchdog /dev/watchdog
respawn root /usr/lib64/heartbeat/ifcheckd
```

Pacemakerを使う
デバッグログ出力フラグ
HB通信のUDPポート
HB通信送信間隔
HB通信断線時の警告までの時間
HB通信断線判断までの時間
初期起動時の待ち合わせ時間
syslog出力時のファシリティ指定
HB通信方法の指定

クラスタに参加するノード名

kernel提供のsoftdogデバイス名
サブプロセスの起動

/etc/ha.d/authkeys
ハートビート通信の認証用キー

```
auth 1
1 sha1 secret
```

secretを任意のパスフレーズで書き換える

/etc/ha.d/authkeys

rootのみにread権限を与えてください

```
# chmod 0600 /etc/ha.d/authkeys
```

/etc/syslog.conf
Pacemakerのログ量が多いため、ログファイルを分けることが望ましい

慣習として出力先: /var/log/ha-log

/etc/syslog.conf
変更点

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

設定反映

```
# /etc/init.d/syslog restart
```

pm_logconv

日本コミュニティ提供ツール

Pacemakerのログを見やすくする

pm_logconv

/etc/pm_logconv.conf を編集

```
[Settings]
#ha_log_path = /var/log/ha-log
#output_path = /var/log/pm_logconv.out
#hostcache_path = /var/lib/heartbeat/hostcache
#syslogformat = True
#reset_interval = 60
attribute_pingd = default_ping_set, lt, 100
attribute_diskd = diskcheck_status, eq, ERROR
attribute_diskd_inner = diskcheck_status_internal, eq, ERROR
#logconv_logfacility = daemon
act_rsc = prmEx prmPg
```

pm_logconv

/etc/inittabに起動設定

```
logc:2345:respawn:/usr/share/pacemaker/pm_logconv/pm_logconv.py
```

initを再起動

```
# telinit q
```

pm_logconv

ログローテーションの設定
/etc/logrotate.d/heartbeat に下記を追加

```
/var/log/pm_logconv.out {
        missingok
}
```

以上で1台めのセットアップが完了

もう1台のサーバも同じようにセットアップしてください

Pacemakerの起動準備ができました

Pacemakerの起動
2台のサーバでコマンドを実行

```
# /etc/init.d/heartbeat start
```

Pacemakerの起動確認
いづれかのサーバで状態表示コマンド実行

```
# crm_mon -A
```

Pacemakerの起動確認
状態表示コマンド実行結果

```
============
Last updated: Fri Mar 18 22:31:07 2011
Stack: Heartbeat
Current DC: pm01 (755595f2-7905-4ba3-909e-68c4e74067bf) - partition with quorum
Version: 1.0.10-da7075976b5ff0bee71074385f8fd02f296ec8a3
2 Nodes configured, unknown expected votes
0 Resources configured.
============

Online: [ pm02 pm01 ]

Node Attributes:
* Node pm02:
    + pm01-eth1                         : up
    + pm01-eth2                         : up
* Node pm01:
    + pm02-eth1                         : up
    + pm02-eth2                         : up
```

2台のサーバがクラスタに組み込まれている

ハートビート通信の状態を表示。ifcheckdによる機能

起動時のログの比較

/var/log/ha-log

v.s.

/var/log/pm_logconv.out

起動時のログの比較
/var/log/ha-log (353行)

```
Apr 22 08:42:45 pm01 heartbeat: [4978]: WARN: Logging daemon is disabled --enabling logging daemon is recommended
Apr 22 08:42:45 pm01 heartbeat: [4978]: info: **************************
Apr 22 08:42:45 pm01 heartbeat: [4978]: info: Configuration validated. Starting heartbeat 3.0.4
Apr 22 08:42:45 pm01 heartbeat: [4979]: info: heartbeat: version 3.0.4
Apr 22 08:42:45 pm01 heartbeat: [4979]: info: Heartbeat generation: 1300455432
Apr 22 08:42:45 pm01 heartbeat: [4979]: info: glib: UDP Broadcast heartbeat started on port 694 (694) interface eth1
Apr 22 08:42:45 pm01 heartbeat: [4979]: info: glib: UDP Broadcast heartbeat closed on port 694 interface eth1 - Status: 1
Apr 22 08:42:45 pm01 heartbeat: [4979]: info: glib: UDP Broadcast heartbeat started on port 694 (694) interface eth2
Apr 22 08:42:45 pm01 heartbeat: [4979]: info: glib: UDP Broadcast heartbeat closed on port 694 interface eth2 - Status: 1
Apr 22 08:42:45 pm01 heartbeat: [4979]: info: G_main_add_TriggerHandler: Added signal manual handler
Apr 22 08:42:45 pm01 heartbeat: [4979]: info: G_main_add_TriggerHandler: Added signal manual handler
Apr 22 08:42:45 pm01 heartbeat: [4979]: notice: Using watchdog device: /dev/watchdog
Apr 22 08:42:45 pm01 heartbeat: [4979]: info: G_main_add_SignalHandler: Added signal handler for signal 17
Apr 22 08:42:45 pm01 heartbeat: [4979]: info: Local status now set to: 'up'
Apr 22 08:42:46 pm01 heartbeat: [4979]: info: Link pm01:eth1 up.
Apr 22 08:42:46 pm01 heartbeat: [4979]: info: Link pm01:eth2 up.
Apr 22 08:42:49 pm01 heartbeat: [4979]: info: Link pm02:eth1 up.
Apr 22 08:42:49 pm01 heartbeat: [4979]: info: Status update for node pm02: status up
Apr 22 08:42:49 pm01 heartbeat: [4979]: info: Link pm02:eth2 up.
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: Comm_now_up(): updating status to active
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: Local status now set to: 'active'
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: Starting child client "/usr/lib64/heartbeat/ccm" (101,105)
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: Starting child client "/usr/lib64/heartbeat/cib" (101,105)
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: Starting child client "/usr/lib64/heartbeat/lrmd -r" (0,0)
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: Starting child client "/usr/lib64/heartbeat/stonithd" (0,0)
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: Starting child client "/usr/lib64/heartbeat/attrd" (101,105)
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: Starting child client "/usr/lib64/heartbeat/crmd" (101,105)
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: Starting child client "/usr/lib64/heartbeat/ifcheckd" (0,0)
Apr 22 08:42:50 pm01 heartbeat: [4989]: info: Starting "/usr/lib64/heartbeat/ccm" as uid 101  gid 105 (pid 4989)
Apr 22 08:42:50 pm01 heartbeat: [4990]: info: Starting "/usr/lib64/heartbeat/cib" as uid 101  gid 105 (pid 4990)
Apr 22 08:42:50 pm01 heartbeat: [4991]: info: Starting "/usr/lib64/heartbeat/lrmd -r" as uid 0  gid 0 (pid 4991)
Apr 22 08:42:50 pm01 heartbeat: [4992]: info: Starting "/usr/lib64/heartbeat/stonithd" as uid 0  gid 0 (pid 4992)
Apr 22 08:42:50 pm01 heartbeat: [4993]: info: Starting "/usr/lib64/heartbeat/attrd" as uid 101  gid 105 (pid 4993)
Apr 22 08:42:50 pm01 heartbeat: [4994]: info: Starting "/usr/lib64/heartbeat/crmd" as uid 101  gid 105 (pid 4994)
Apr 22 08:42:50 pm01 attrd: [4993]: info: Invoked: /usr/lib64/heartbeat/attrd
Apr 22 08:42:50 pm01 lrmd: [4991]: info: G_main_add_SignalHandler: Added signal handler for signal 15
Apr 22 08:42:50 pm01 attrd: [4993]: info: main: Starting up
Apr 22 08:42:50 pm01 heartbeat: [4995]: info: Starting "/usr/lib64/heartbeat/ifcheckd" as uid 0  gid 0 (pid 4995)
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: Status update for node pm02: status active
Apr 22 08:42:50 pm01 cib: [4990]: info: Invoked: /usr/lib64/heartbeat/cib
Apr 22 08:42:50 pm01 stonithd: [4992]: info: G_main_add_SignalHandler: Added signal handler for signal 10
Apr 22 08:42:50 pm01 ifcheckd: [4995]: info: Invoked: /usr/lib64/heartbeat/ifcheckd
Apr 22 08:42:50 pm01 ccm: [4989]: info: Hostname: pm01
Apr 22 08:42:50 pm01 cib: [4990]: info: G_main_add_TriggerHandler: Added signal manual handler
Apr 22 08:42:50 pm01 stonithd: [4992]: info: G_main_add_SignalHandler: Added signal handler for signal 12
Apr 22 08:42:50 pm01 cib: [4990]: info: G_main_add_SignalHandler: Added signal handler for signal 17
Apr 22 08:42:50 pm01 lrmd: [4991]: info: G_main_add_SignalHandler: Added signal handler for signal 17
Apr 22 08:42:50 pm01 lrmd: [4991]: info: enabling coredumps
Apr 22 08:42:50 pm01 lrmd: [4991]: info: G_main_add_SignalHandler: Added signal handler for signal 10
Apr 22 08:42:50 pm01 lrmd: [4991]: info: G_main_add_SignalHandler: Added signal handler for signal 12
Apr 22 08:42:50 pm01 lrmd: [4991]: info: Started.
Apr 22 08:42:50 pm01 cib: [4990]: info: retrieveCib: Reading cluster configuration from: /var/lib/heartbeat/crm/cib.xml (digest: /var/lib/heartbeat/crm/cib.xml.sig)
Apr 22 08:42:50 pm01 crmd: [4994]: info: Invoked: /usr/lib64/heartbeat/crmd
Apr 22 08:42:50 pm01 crmd: [4994]: info: main: CRM Hg Version: da7075976b5ff0bee71074385f8fd02f296ec8a3
Apr 22 08:42:50 pm01 crmd: [4994]: info: crmd_init: Starting crmd
Apr 22 08:42:50 pm01 crmd: [4994]: info: G_main_add_SignalHandler: Added signal handler for signal 17
Apr 22 08:42:50 pm01 attrd: [4993]: info: register_heartbeat_conn: Hostname: pm01
Apr 22 08:42:50 pm01 attrd: [4993]: info: register_heartbeat_conn: UUID: 8559a0d8-a5c9-4a84-aa7e-1961cad9658c
Apr 22 08:42:50 pm01 attrd: [4993]: info: crm_cluster_connect: Connecting to Heartbeat
Apr 22 08:42:50 pm01 attrd: [4993]: info: main: Cluster connection active
Apr 22 08:42:50 pm01 attrd: [4993]: info: main: Accepting attribute updates
Apr 22 08:42:50 pm01 attrd: [4993]: info: main: Starting mainloop...
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: the send queue length from heartbeat to client ccm is set to 1024
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: the send queue length from heartbeat to client attrd is set to 1024
Apr 22 08:42:50 pm01 cib: [4990]: info: startCib: CIB Initialization completed successfully
Apr 22 08:42:50 pm01 stonithd: [4992]: info: register_heartbeat_conn: Hostname: pm01
Apr 22 08:42:50 pm01 stonithd: [4992]: info: register_heartbeat_conn: UUID: 8559a0d8-a5c9-4a84-aa7e-1961cad9658c
Apr 22 08:42:50 pm01 stonithd: [4992]: info: crm_cluster_connect: Connecting to Heartbeat
Apr 22 08:42:50 pm01 heartbeat: [4979]: info: the send queue length from heartbeat to client stonithd is set to 1024
Apr 22 08:42:50 pm01 stonithd: [4992]: notice: /usr/lib64/heartbeat/stonithd start up successfully.
Apr 22 08:42:50 pm01 stonithd: [4992]: info: G_main_add_SignalHandler: Added signal handler for signal 17
Apr 22 08:42:51 pm01 cib: [4990]: info: register_heartbeat_conn: Hostname: pm01
Apr 22 08:42:51 pm01 cib: [4990]: info: register_heartbeat_conn: UUID: 8559a0d8-a5c9-4a84-aa7e-1961cad9658c
Apr 22 08:42:51 pm01 cib: [4990]: info: crm_cluster_connect: Connecting to Heartbeat
Apr 22 08:42:51 pm01 cib: [4990]: info: ccm_connect: Registering with CCM...
Apr 22 08:42:51 pm01 cib: [4990]: WARN: ccm_connect: CCM Activation failed
Apr 22 08:42:51 pm01 cib: [4990]: WARN: ccm_connect: CCM Connection failed 1 times (30 max)
Apr 22 08:42:51 pm01 heartbeat: [4979]: info: the send queue length from heartbeat to client cib is set to 1024
Apr 22 08:42:51 pm01 crmd: [4994]: info: do_cib_control: Could not connect to the CIB service: connection failed
Apr 22 08:42:51 pm01 crmd: [4994]: WARN: do_cib_control: Couldn't complete CIB registration 1 times... pause and retry
Apr 22 08:42:51 pm01 crmd: [4994]: info: crmd_init: Starting crmd's mainloop
```

```
Apr 22 08:42:53 pm01 ccm: [4989]: info: G_main_add_SignalHandler: Added signal handler for signal 15
Apr 22 08:42:53 pm01 crmd: [4994]: info: crm_timer_popped: Wait Timer (I_NULL) just popped!
Apr 22 08:42:54 pm01 cib: [4990]: info: ccm_connect: Registering with CCM...
Apr 22 08:42:54 pm01 cib: [4990]: info: cib_init: Requesting the list of configured nodes
Apr 22 08:42:54 pm01 cib: [4990]: info: cib_init: Starting cib mainloop
Apr 22 08:42:54 pm01 cib: [4990]: info: cib_client_status_callback: Status update: Client pm01/cib now has status [join]
Apr 22 08:42:54 pm01 cib: [4990]: info: crm_new_peer: Node 0 is now known as pm01
Apr 22 08:42:54 pm01 cib: [4990]: info: crm_update_peer_proc: pm01.cib is now online
Apr 22 08:42:54 pm01 cib: [4990]: info: cib_client_status_callback: Status update: Client pm02/cib now has status [join]
Apr 22 08:42:54 pm01 cib: [4990]: info: crm_new_peer: Node 0 is now known as pm02
Apr 22 08:42:54 pm01 cib: [4990]: info: crm_update_peer_proc: pm02.cib is now online
Apr 22 08:42:54 pm01 cib: [4990]: info: cib_client_status_callback: Status update: Client pm01/cib now has status [online]
Apr 22 08:42:54 pm01 crmd: [4994]: info: do_cib_control: CIB connection established
Apr 22 08:42:54 pm01 cib: [5002]: info: write_cib_contents: Archived previous version as /var/lib/heartbeat/crm/cib-90.raw
Apr 22 08:42:54 pm01 cib: [5002]: info: write_cib_contents: Wrote version 0.244.0 of the CIB to disk (digest: d0cf804b2a297ae9841f7b172f2a18fe)
Apr 22 08:42:54 pm01 cib: [5002]: info: retrieveCib: Reading cluster configuration from: /var/lib/heartbeat/crm/cib.cld1zN (digest:
/var/lib/heartbeat/crm/cib.6Fjral)
Apr 22 08:42:55 pm01 heartbeat: [4979]: WARN: 1 lost packet(s) for [pm02] [16:18]
Apr 22 08:42:55 pm01 heartbeat: [4979]: info: No pkts missing from pm02!
Apr 22 08:42:55 pm01 cib: [4990]: info: cib_client_status_callback: Status update: Client pm02/cib now has status [online]
Apr 22 08:42:55 pm01 crmd: [4994]: info: register_heartbeat_conn: Hostname: pm01
Apr 22 08:42:55 pm01 crmd: [4994]: info: register_heartbeat_conn: UUID: 8559a0d8-a5c9-4a84-aa7e-1961cad9658c
Apr 22 08:42:55 pm01 crmd: [4994]: info: crm_cluster_connect: Connecting to Heartbeat
Apr 22 08:42:55 pm01 heartbeat: [4979]: info: the send queue length from heartbeat to client crmd is set to 1024
Apr 22 08:42:55 pm01 crmd: [4994]: info: do_ha_control: Connected to the cluster
Apr 22 08:42:55 pm01 crmd: [4994]: info: do_ccm_control: CCM connection established... waiting for first callback
Apr 22 08:42:55 pm01 crmd: [4994]: info: do_started: Delaying start, CCM (0000000000100000) not connected
Apr 22 08:42:55 pm01 crmd: [4994]: notice: crmd_client_status_callback: Status update: Client pm01/crmd now has status [online] (DC=false)
Apr 22 08:42:55 pm01 attrd: [4993]: info: cib_connect: Connected to the CIB after 1 signon attempts
Apr 22 08:42:55 pm01 attrd: [4993]: info: cib_connect: Sending full refresh
Apr 22 08:42:55 pm01 crmd: [4994]: info: crm_new_peer: Node 0 is now known as pm01
Apr 22 08:42:55 pm01 crmd: [4994]: info: crm_update_peer_proc: pm01.crmd is now online
Apr 22 08:42:55 pm01 crmd: [4994]: info: crmd_client_status_callback: Not the DC
Apr 22 08:42:55 pm01 crmd: [4994]: notice: crmd_client_status_callback: Status update: Client pm01/crmd now has status [online] (DC=false)
Apr 22 08:42:56 pm01 crmd: [4994]: info: crmd_client_status_callback: Not the DC
Apr 22 08:42:56 pm01 crmd: [4994]: notice: crmd_client_status_callback: Status update: Client pm02/crmd now has status [offline] (DC=false)
Apr 22 08:42:56 pm01 crmd: [4994]: info: crm_new_peer: Node 0 is now known as pm02
Apr 22 08:42:56 pm01 crmd: [4994]: info: crmd_client_status_callback: Not the DC
Apr 22 08:42:56 pm01 crmd: [4994]: info: do_started: Delaying start, CCM (0000000000100000) not connected
Apr 22 08:42:56 pm01 crmd: [4994]: info: config_query_callback: Checking for expired actions every 900000ms
Apr 22 08:42:56 pm01 crmd: [4994]: info: do_started: Delaying start, CCM (0000000000100000) not connected
Apr 22 08:42:57 pm01 crmd: [4994]: notice: crmd_client_status_callback: Status update: Client pm02/crmd now has status [online] (DC=false)
Apr 22 08:42:57 pm01 cib: [4990]: info: mem_handle_event: Got an event OC_EV_MS_NEW_MEMBERSHIP from ccm
Apr 22 08:42:57 pm01 cib: [4990]: info: mem_handle_event: instance=2, nodes=2, new=2, lost=0, n_idx=0, new_idx=0, old_idx=4
Apr 22 08:42:57 pm01 cib: [4990]: info: cib_ccm_msg_callback: Processing CCM event=NEW MEMBERSHIP (id=2)
Apr 22 08:42:57 pm01 cib: [4990]: info: crm_get_peer: Node pm02 now has id: 1
Apr 22 08:42:57 pm01 cib: [4990]: info: crm_update_peer: Node pm02: id=1 state=member (new) addr=(null) votes=-1 born=1 seen=2
proc=00000000000000000000000000000100
Apr 22 08:42:57 pm01 cib: [4990]: info: crm_update_peer_proc: pm02.ais is now online
Apr 22 08:42:57 pm01 cib: [4990]: info: crm_update_peer_proc: pm02.crmd is now online
Apr 22 08:42:57 pm01 cib: [4990]: info: crm_update_peer: Node pm01: id=0 state=member (new) addr=(null) votes=-1 born=2 seen=2
proc=00000000000000000000000000000100
Apr 22 08:42:57 pm01 cib: [4990]: info: crm_update_peer_proc: pm01.ais is now online
Apr 22 08:42:57 pm01 cib: [4990]: info: crm_update_peer_proc: pm01.crmd is now online
Apr 22 08:42:57 pm01 crmd: [4994]: info: crm_update_peer_proc: pm02.crmd is now online
Apr 22 08:42:57 pm01 crmd: [4994]: info: crmd_client_status_callback: Not the DC
Apr 22 08:42:57 pm01 crmd: [4994]: info: mem_handle_event: Got an event OC_EV_MS_NEW_MEMBERSHIP from ccm
Apr 22 08:42:57 pm01 crmd: [4994]: info: mem_handle_event: instance=2, nodes=2, new=2, lost=0, n_idx=0, new_idx=0, old_idx=4
Apr 22 08:42:57 pm01 crmd: [4994]: info: crmd_ccm_msg_callback: Quorum (re)attained after event=NEW MEMBERSHIP (id=2)
Apr 22 08:42:57 pm01 crmd: [4994]: info: ccm_event_detail: NEW MEMBERSHIP: trans=2, nodes=2, new=2, lost=0 n_idx=0, new_idx=0, old_idx=4
Apr 22 08:42:57 pm01 crmd: [4994]: info: ccm_event_detail:      CURRENT: pm02 [nodeid=1, born=1]
Apr 22 08:42:57 pm01 crmd: [4994]: info: ccm_event_detail:      CURRENT: pm01 [nodeid=0, born=2]
Apr 22 08:42:57 pm01 crmd: [4994]: info: ccm_event_detail:      NEW:     pm02 [nodeid=1, born=1]
Apr 22 08:42:57 pm01 crmd: [4994]: info: ccm_event_detail:      NEW:     pm01 [nodeid=0, born=2]
Apr 22 08:42:57 pm01 crmd: [4994]: info: crm_get_peer: Node pm02 now has id: 1
Apr 22 08:42:57 pm01 crmd: [4994]: info: crm_update_peer: Node pm02: id=1 state=member (new) addr=(null) votes=-1 born=1 seen=2
proc=00000000000000000000000000000200
Apr 22 08:42:57 pm01 crmd: [4994]: info: crm_update_peer_proc: pm02.ais is now online
Apr 22 08:42:57 pm01 crmd: [4994]: info: crm_update_peer: Node pm01: id=0 state=member (new) addr=(null) votes=-1 born=2 seen=2
proc=00000000000000000000000000000200
Apr 22 08:42:57 pm01 crmd: [4994]: info: crm_update_peer_proc: pm01.ais is now online
Apr 22 08:42:57 pm01 crmd: [4994]: info: do_started: The local CRM is operational
Apr 22 08:42:57 pm01 crmd: [4994]: info: do_state_transition: State transition S_STARTING -> S_PENDING [ input=I_PENDING cause=C_FSA_INTERNAL
origin=do_started ]
Apr 22 08:42:58 pm01 heartbeat: [4979]: WARN: 1 lost packet(s) for [pm02] [23:25]
Apr 22 08:42:58 pm01 heartbeat: [4979]: info: No pkts missing from pm02!
Apr 22 08:43:58 pm01 crmd: [4994]: info: crm_timer_popped: Election Trigger (I_DC_TIMEOUT) just popped!
Apr 22 08:43:58 pm01 crmd: [4994]: WARN: do_log: FSA: Input I_DC_TIMEOUT from crm_timer_popped() received in state S_PENDING
Apr 22 08:43:58 pm01 crmd: [4994]: info: do_state_transition: State transition S_PENDING -> S_ELECTION [ input=I_DC_TIMEOUT
cause=C_TIMER_POPPED origin=crm_timer_popped ]
Apr 22 08:43:59 pm01 crmd: [4994]: info: do_state_transition: State transition S_ELECTION -> S_PENDING [ input=I_PENDING cause=C_FSA_INTERNAL
origin=do_election_count_vote ]
```

まだまだ続く

起動時のログの比較

/var/log/pm_logconv.out (41行)

```
Apr 22 08:42:45 pm01 info: Starting Heartbeat 3.0.4.
Apr 22 08:42:46 pm01 info: Link pm01:eth1 is up.
Apr 22 08:42:46 pm01 info: Link pm01:eth2 is up.
Apr 22 08:42:49 pm01 info: Link pm02:eth1 is up.
Apr 22 08:42:49 pm01 info: Link pm02:eth2 is up.
Apr 22 08:42:50 pm01 info: Start "ccm" process. (pid=4989)
Apr 22 08:42:50 pm01 info: Start "cib" process. (pid=4990)
Apr 22 08:42:50 pm01 info: Start "lrmd" process. (pid=4991)
Apr 22 08:42:50 pm01 info: Start "stonithd" process. (pid=4992)
Apr 22 08:42:50 pm01 info: Start "attrd" process. (pid=4993)
Apr 22 08:42:50 pm01 info: Start "crmd" process. (pid=4994)
Apr 22 08:42:50 pm01 info: Start "ifcheckd" process. (pid=4995)
Apr 22 08:44:04 pm01 info: Set DC node to pm02.
Apr 22 08:44:10 pm01 info: Resource prmStonith2-1 tries to start.
Apr 22 08:44:10 pm01 info: Resource prmStonith2-1 started. (rc=0)
Apr 22 08:44:11 pm01 info: Attribute "pm02-eth1" is updated to "up".
Apr 22 08:44:11 pm01 info: Attribute "pm02-eth2" is updated to "up".
Apr 22 08:44:11 pm01 info: Resource prmStonith2-2 tries to start.
Apr 22 08:44:11 pm01 info: Resource prmDiskd1:1 tries to start.
Apr 22 08:44:11 pm01 info: Resource prmDiskd2:1 tries to start.
Apr 22 08:44:11 pm01 info: Resource prmPingd:1 tries to start.
Apr 22 08:44:12 pm01 info: Attribute "diskcheck_status" is updated to "normal".
Apr 22 08:44:12 pm01 info: Resource prmDiskd2:1 started. (rc=0)
Apr 22 08:44:12 pm01 info: Attribute "diskcheck_status_internal" is updated to "normal".
Apr 22 08:44:12 pm01 info: Resource prmStonith2-2 started. (rc=0)
Apr 22 08:44:12 pm01 info: Resource prmDiskd1:1 started. (rc=0)
Apr 22 08:44:12 pm01 info: Attribute "default_ping_set" is updated to "100".
Apr 22 08:44:12 pm01 info: Resource prmPingd:1 started. (rc=0)
Apr 22 08:44:12 pm01 info: Attribute "diskcheck_status" is updated to "normal".
Apr 22 08:44:12 pm01 info: Attribute "diskcheck_status_internal" is updated to "normal".
Apr 22 08:44:12 pm01 info: Attribute "default_ping_set" is updated to "100".
Apr 22 08:44:12 pm01 info: Resource prmEx tries to start.
Apr 22 08:44:12 pm01 info: Resource prmStonith2-3 tries to start.
Apr 22 08:44:13 pm01 info: Resource prmStonith2-3 started. (rc=0)
Apr 22 08:44:14 pm01 info: Resource prmEx started. (rc=0)
Apr 22 08:44:14 pm01 info: Resource prmFs tries to start.
Apr 22 08:44:15 pm01 info: Resource prmFs started. (rc=0)
Apr 22 08:44:16 pm01 info: Resource prmIp tries to start.
Apr 22 08:44:16 pm01 info: Resource prmIp started. (rc=0)
Apr 22 08:44:18 pm01 info: Resource prmPg tries to start.
Apr 22 08:44:21 pm01 info: Resource prmPg started. (rc=0)
```

Pacemakerのログから、特に運用に関係のあるものを抽出、メッセージ変換するツール

リソース設定

Crmコマンド (統合シェル)を使用

Crmコマンド (統合シェル)を使用

Pacemakerで導入され

リソース設定からノード・リソース管理なども行える

Crmコマンド (統合シェル)を使用

リソース設定をファイル(例：pg.crm)で作成して、コマンドを実行

```
# crm configure load update pg.crm
```

リソース設定

リソース設定

クラスタオプション

リソースデフォルトオプション

個別のリソース設定

リソース制約条件

リソース設定

クラスタオプション

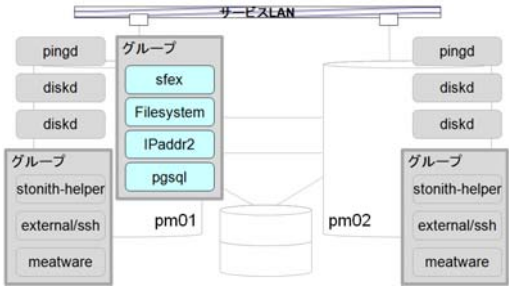
```
property no-quorum-policy="ignore" ¥
   stonith-enabled="true" ¥
   startup-fencing="false" ¥
   stonith-timeout="740s"
```

リソース設定

リソースデフォルトオプション

```
rsc_defaults resource-stickiness="INFINITY" ¥
  migration-threshold="1"
```

リソース設定

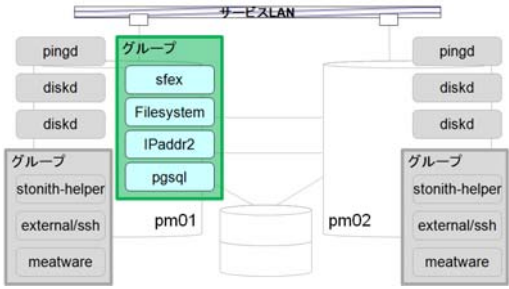


PostgreSQLのフェイルオーバグループのprimitive宣言

```
primitive prmEx ocf:heartbeat:sfex ¥
  params ¥
    device="/dev/sdb1" ¥
    index="1" ¥
    collision_timeout="1" ¥
    lock_timeout="10" ¥
    monitor_interval="10" ¥
  op start interval="0s" timeout="300s" on-fail="restart" ¥
  op monitor interval="10s" timeout="90s" on-fail="restart" ¥
  op stop interval="0s" timeout="60s" on-fail="fence"

primitive prmFs ocf:heartbeat:Filesystem ¥
  params ¥
    fstype="ext3" ¥
    device="/dev/sdb2" ¥
    directory="/var/lib/pgsql/9.0/data" ¥
  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"
```

```
primitive prmIp ocf:heartbeat:IPaddr2 ¥
  params ¥
    ip="192.168.68.100" ¥
    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="fence"

primitive prmPg ocf:heartbeat:pgsql ¥
  params ¥
    pgctl="/usr/pgsql-9.0/bin/pg_ctl" ¥
    start_opt="-p 5432 -h 192.168.68.100" ¥
    psql="/usr/pgsql-9.0/bin/psql" ¥
    pgdata="/var/lib/pgsql/9.0/data" ¥
    pgdba="postgres" ¥
    pgport="5432" ¥
    pgdb="template1" ¥
  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"
```

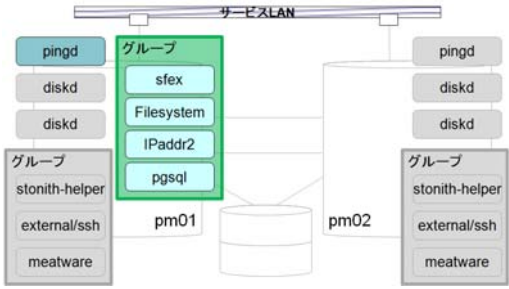
リソース設定



PostgreSQLのフェイルオーバグループのグループ宣言

```
group grpPg ¥
    prmEx ¥
    prmFs ¥
    prmIp ¥
    prmPg
```

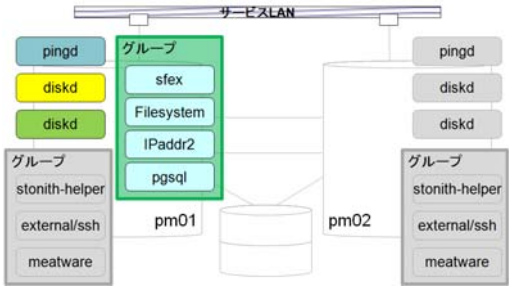
リソース設定



ネットワーク経路監視のprimitive宣言

```
primitive prmPingd ocf:pacemaker:pingd ¥
   params ¥
      name="default_ping_set" ¥
      host_list="192.168.68.2" ¥
      multiplier="100" ¥
      dampen="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="ignore"
```

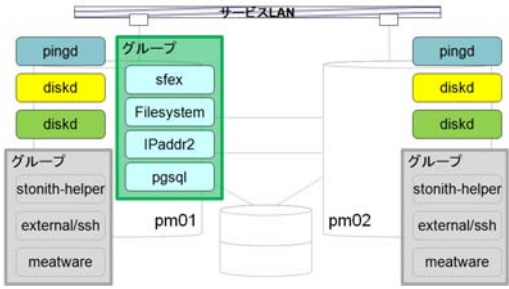
リソース設定



ディスク監視のprimitive宣言

```
primitive prmPingd ocf:pacemaker:pingd ¥
   params ¥
      name="default_ping_set" ¥
      host_list="192.168.68.2" ¥
      multiplier="100" ¥
      dampen="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="ignore"

primitive prmDiskd1 ocf:pacemaker:diskd ¥
   params ¥
      name="diskcheck_status" ¥
      device="/dev/sdb" ¥
      interval="10" ¥
   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"
```

リソース設定



ネットワーク経路監視、ディスク監視のclone化

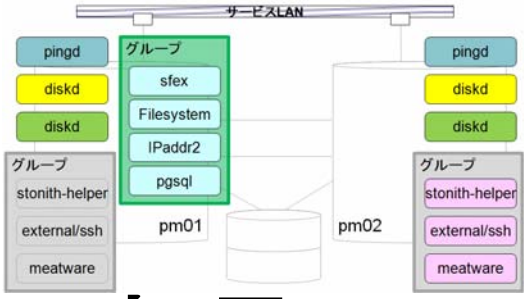
```
clone clnPingd ¥
   prmPingd

clone clnDiskd1 ¥
   prmDiskd1

clone clnDiskd2 ¥
   prmDiskd2
```

リソース設定

PostgreSQLリソースグループの配置制約

```
location rsc_location-grpPg-1 grpPg ¥
   rule 200: #uname eq pm01 ¥
   rule 100: #uname eq pm02 ¥
   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
```

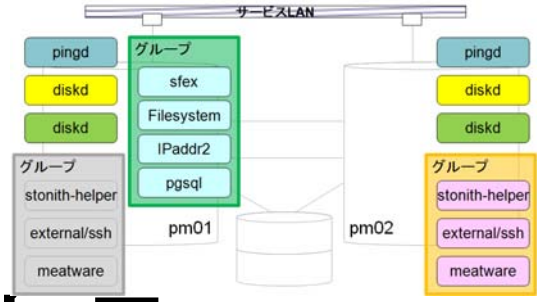
STONITH設定



STONITH Pluginのprimitive宣言

```
primitive prmStonith1-1 stonith:external/stonith-helper ¥
  params ¥
    priority="1" ¥
    stonith-timeout="70" ¥
    hostlist="pm01" ¥
    dead_check_target="192.168.68.101 192.168.32.101 192.168.64.101 192.168.128.101" ¥
    standby_wait_time="15" ¥
    standby_check_command="/usr/sbin/crm_resource -r prmEx -W | grep -q `hostname`" ¥
  op start interval="0s" timeout="60s" on-fail="restart" ¥
  op monitor interval="3600s" timeout="60s" on-fail="restart" ¥
  op stop interval="0s" timeout="60s" on-fail="ignore"

primitive prmStonith1-2 stonith:external/ssh ¥
  params ¥
    priority="2" ¥
    stonith-timeout="300" ¥
    hostlist="pm01" ¥
  op start interval="0s" timeout="60s" on-fail="restart" ¥
  op monitor interval="3600s" timeout="60s" on-fail="restart" ¥
  op stop interval="0s" timeout="60s" on-fail="ignore"

primitive prmStonith1-3 stonith:meatware ¥
  params ¥
    priority="3" ¥
    stonith-timeout="600" ¥
    hostlist="pm01" ¥
  op start interval="0s" timeout="60s" ¥
  op monitor interval="3600s" timeout="60s" ¥
  op stop interval="0s" timeout="60s"
```

STONITH設定

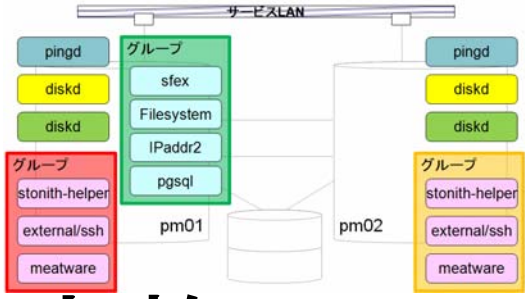


STONITH Pluginのgroup宣言

```
group grpStonith1 ¥
   prmStonith1-1 ¥
   prmStonith1-2 ¥
   prmStonith1-3
```

STONITH設定

pm02用の設定も同じように実施



STONITH設定

STONITH Pluginの配置制約

```
location rsc_location-grpStonith1-2 grpStonith1 ¥
   rule -INFINITY: #uname eq pm01
location rsc_location-grpStonith2-3 grpStonith2 ¥
   rule -INFINITY: #uname eq pm02
```

リソース設定の反映

リソース設定をエディタ(例：pg.crm)で作成して、次のコマンドを実行

```
# crm configure load update pg.crm
```

Pacemakerのクラスタ状態表示

```
# crm_mon -A
```

Pacemakerのクラスタ状態表示

```
============
(省略)
============
Online: [ pm02 pm01 ]
Resource Group: grpPg
    prmEx      (ocf::heartbeat:sfex):  Started pm01
    prmFs      (ocf::heartbeat:Filesystem):    Started pm01
    prmIp      (ocf::heartbeat:IPaddr2):       Started pm01
    prmPg      (ocf::heartbeat:pgsql): Started pm01
Resource Group: grpStonith1
    prmStonith1-1      (stonith:external/stonith-helper):      Started pm02
    prmStonith1-2      (stonith:external/ssh): Started pm02
    prmStonith1-3      (stonith:meatware):     Started pm02
Resource Group: grpStonith2
    prmStonith2-1      (stonith:external/stonith-helper):      Started pm01
    prmStonith2-2      (stonith:external/ssh): Started pm01
    prmStonith2-3      (stonith:meatware):     Started pm01
Clone Set: clnDiskd1
    Started: [ pm02 pm01 ]
Clone Set: clnDiskd2
    Started: [ pm02 pm01 ]
Clone Set: clnPingd
    Started: [ pm02 pm01 ]
Node Attributes:
* Node pm02:
   + default_ping_set                  : 100
   + diskcheck_status                  : normal
   + diskcheck_status_internal         : normal
   + pm01-eth1                         : up
   + pm01-eth2                         : up
```

PostgreSQL関連の設定

STONITHの設定

ネットワーク経路監視、ディスク監視の設定

pm02における各種監視情報の値(ネットワーク経路監視、ディスク監視、ハートビートLAN監視)

デモビデオ

デモビデオ1

リソースがなにも設定されていない状態から、PostgreSQLのHAクラスタ設定をロードします

```
# crm configure load update pg.crm
```

デモビデオ2

リソース故障(PostgreSQL)を発生させます

Pacemakerが故障を検知し、リソースをフェイルオーバさせます

故障復旧後、次のコマンドで故障状態をクリアします

```
# crm resource cleanup prmPg
```

デモビデオ3

ネットワーク経路監視でエラーを発生させます

Pacemakerが故障を検知し、リソースをフェイルオーバさせます

デモビデオ4

ハートビートLANをすべて断線し、スプリットブレインを発生させます

STONITHが発動されます。stonith-helperにより、待機ノードがリブートされます

pm_crmgen

ExcelまたはOpenOfficeで編集したテンプレートファイルからcrmコマンドの入力ファイルを生成するツールです

pm_crmgen

1） Excelのテンプレートファイルにリソース定義を記載

/usr/share/pacemaker/pm_crmgen/pm_crmgen_env.xls　ファイルを

Excel が使用できるPCにコピーします。
テンプレートは青枠の中に値を
記入していきます。

83	#表 5-3 クラスタ設定　...　Primitiveリソース (id=prmIp)					
84	PRIMITIVE					
85	P	id	class	provider	type	
86	#	リソースID	class	provider	type	
87		prmIp	ocf	heartbeat	IPaddr2	
88	A	type	name	value		
89	#	パラメータ種別	項目	設定内容		
90		params	ip	192.168.0.100		
91			nic	eth0		
92			cidr_netmask	24		
93	O	type	timeout	interval	on-fail	start-delay
94	#	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）	起動前処理
95		start	60s	0s	restart	
96		monitor	60s	10s	restart	
97		stop	60s	0s	fence	

デモビデオ環境は、このExcelの設定例シートのみで作成

監視間隔やタイムアウト値、故障時の動作などを入力

pm_crmgen

どのサーバを稼動系にするかといった
リソース配置制約の設定も、サーバ名を
記述するだけで可能です。

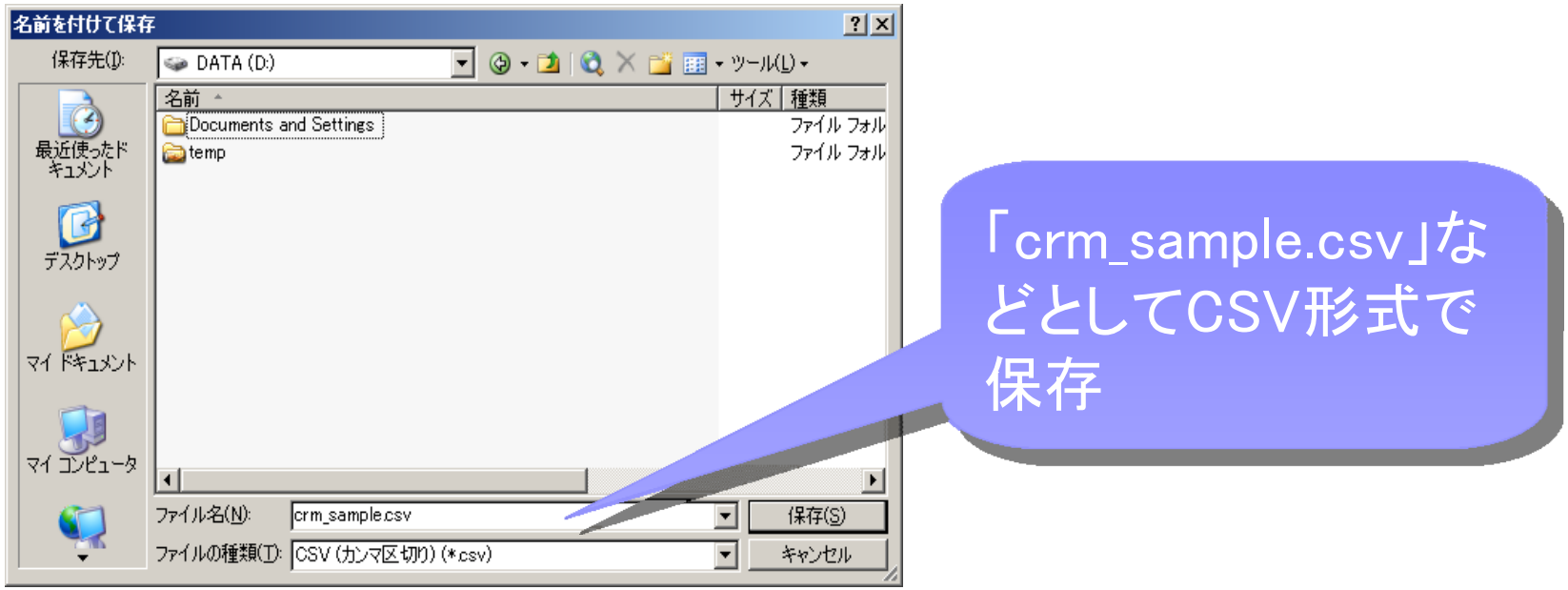
```
270 #表 6-1 クラスタ設定　...　リソース配置制約
271   LOCATION
272     rsc          score:200    score:100    score:-inf
273 #   リソースID    Activeノード  Standbyノー  非稼働ノード
274     grpPg        pm1          pm2
275     grpStonith1                            pm1
276     grpSt                                  pm2
```

リソース ID

稼動系と待機系サーバを指定

pm_crmgen
crm用設定ファイルに変換

2）CSV形式でファイルを保存



3）CSVファイルをサーバへ転送

CSVファイル保存後、SCPやFTP等でpm_crmgenがインストールされたサーバへ転送

pm_crmgen
crm用設定ファイルに変換

4）pm_crmgenコマンドでcrmファイルを生成

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

生成する設定ファイル名

3）で転送した
CSVファイル

5）crmコマンドを実行してリソース設定を反映

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

最後に皆さんへのメッセージ

Pacemakerは商用製品にも負けない機能と信頼性を持ったHAクラスタソフトウェアです

まずは、Pacemakerを使ってみてください

皆さんが使うことが、OSSの力になります

できることからOSSへの貢献を始めてみませんか？

できることからOSSへの貢献を始めてみませんか？

質問

バグ報告

新規RAの提案

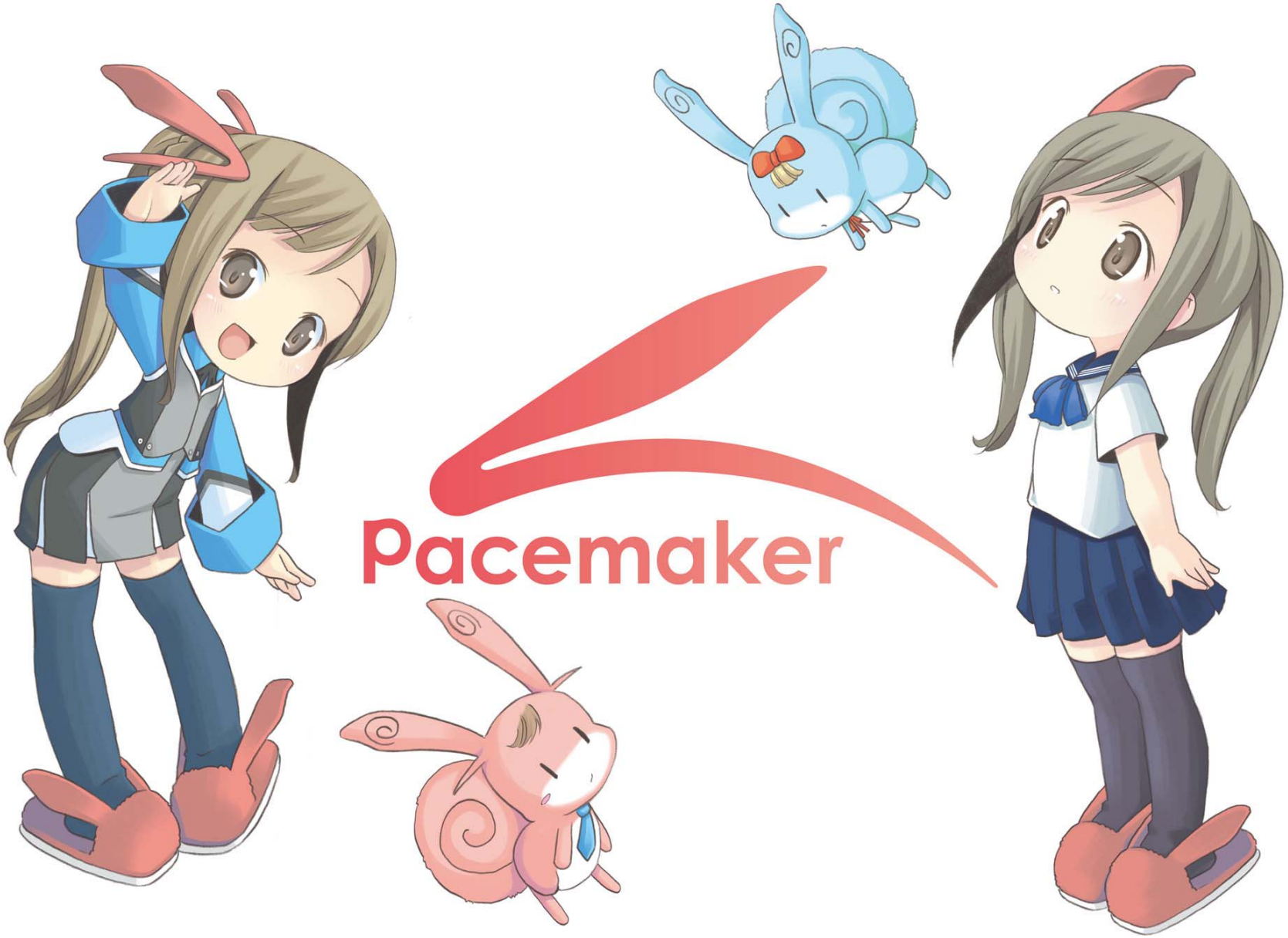
エンハンスメントの提案

自己のスキルレベルアップを確認したくなったら、

自己のスキルレベルアップを確認したくなったら、

LPIC304を受けてみましょう!

ご静聴ありがとうございました！



新ロゴ誕生の裏話

昨年、Pacemakerをプロモーションしていこうとなったとき、どうしても気になったこと

Pacemakerのロゴ

医療用のPacemakerっぽいのをどうにかしたい!

Linux-HA Japanで投票で新しいロゴを作りました

Linux-HA Japanで投票で新しいロゴを作りました

これって、うさぎ？

Linux-HA Japanで投票で新しいロゴを作りました

はいそうです。続きは、Web(↓)で

http://gihyo.jp/admin/serial/01/pacemaker/0001

コラム：Pacemakerロゴあれこれ

調子にのって、本家の開発者にも新ロゴを提案

Linux Plumbers Conference 2010

意外にも好印象!

公開投票でどっちがいいか決めようということに

公開投票の結果は？

公開投票の結果は？

というわけで、本家のロゴにも採用

でも、カラーにこだわりが