Heartbeat + Xenで仮想化クラスタリングしてみよう！

2010年2月26日

Linux-HA-Japanプロジェクト
田中崇幸

本日の話題

① Heartbeatって何？
② 仮想化クラスタリング構成
③ 仮想化クラスタリング応用構成編
④ Linux-HA-Japanプロジェクトについて
⑤ 参考情報

①

Heartbeatって何？

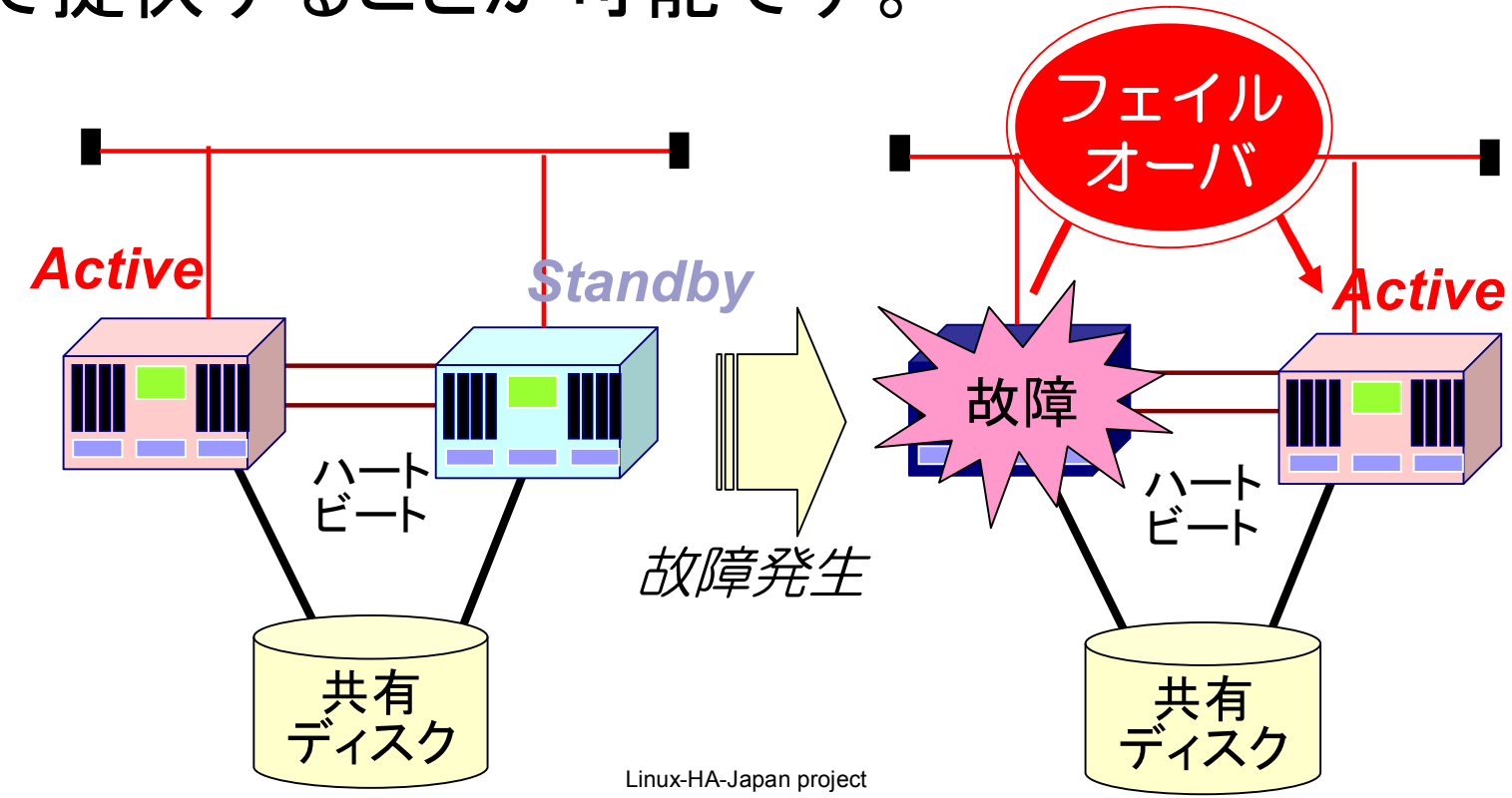
Heartbeatとは？

オープンソースで実現する
高可用性クラスタリングソフトウェアです

Heartbeat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。

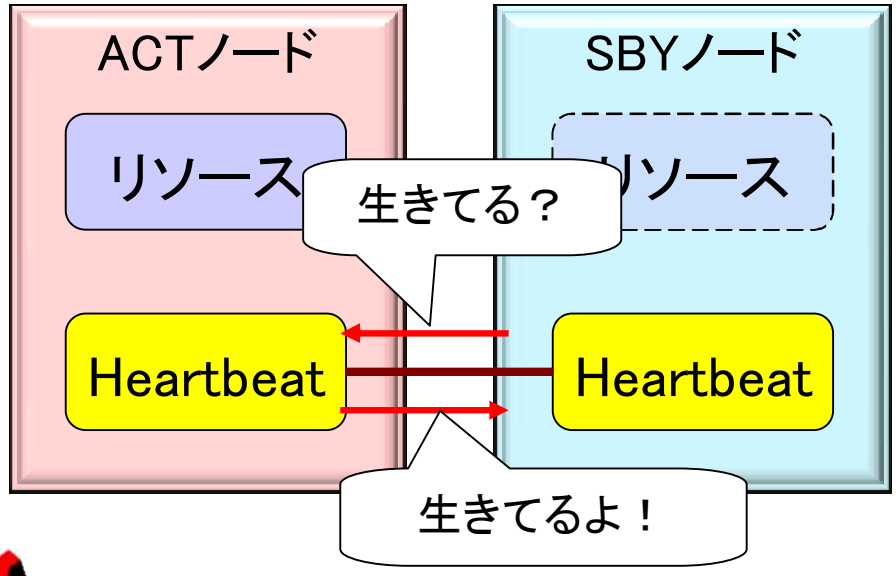
概要

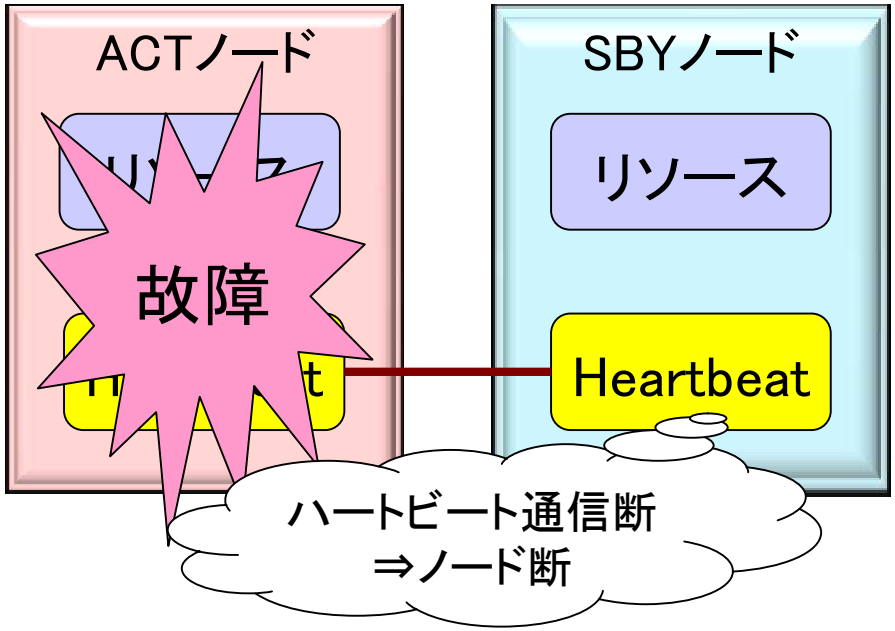
- Heartbeatは、故障発生を検知し、待機系サーバへフェイルオーバさせることが可能です。
- サービス利用者には故障を意識させずにサービスを継続して提供することが可能です。



基本的動作：ノード監視

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





「リソース」「リソースエージェント」とは？

Heartbeat ではよく出てくる
言葉なのでおぼえてください！

- リソース

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

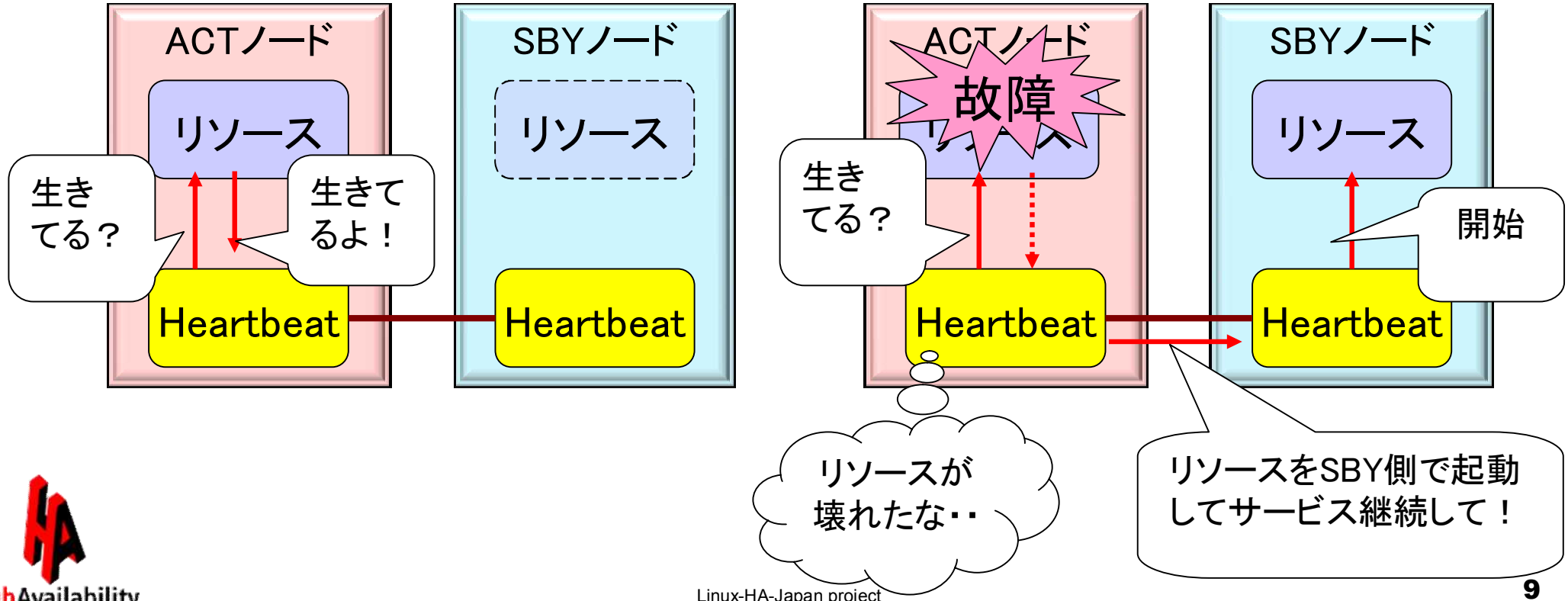
リソースエージェント（RA）

リソースエージェント（RA）とは、そのリソースとHeartbeat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。

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

基本的動作：リソース制御

- リソースの制御：起動(start)、停止(stop)、監視(monitor)
  - 起動後は一定間隔で監視。正しく動作していない場合にはフェイルオーバ等の処理を実施します。



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

MySQLや、Tomcat用のリソースエージェントなどもありますよ！

標準リソースエージェントの一例

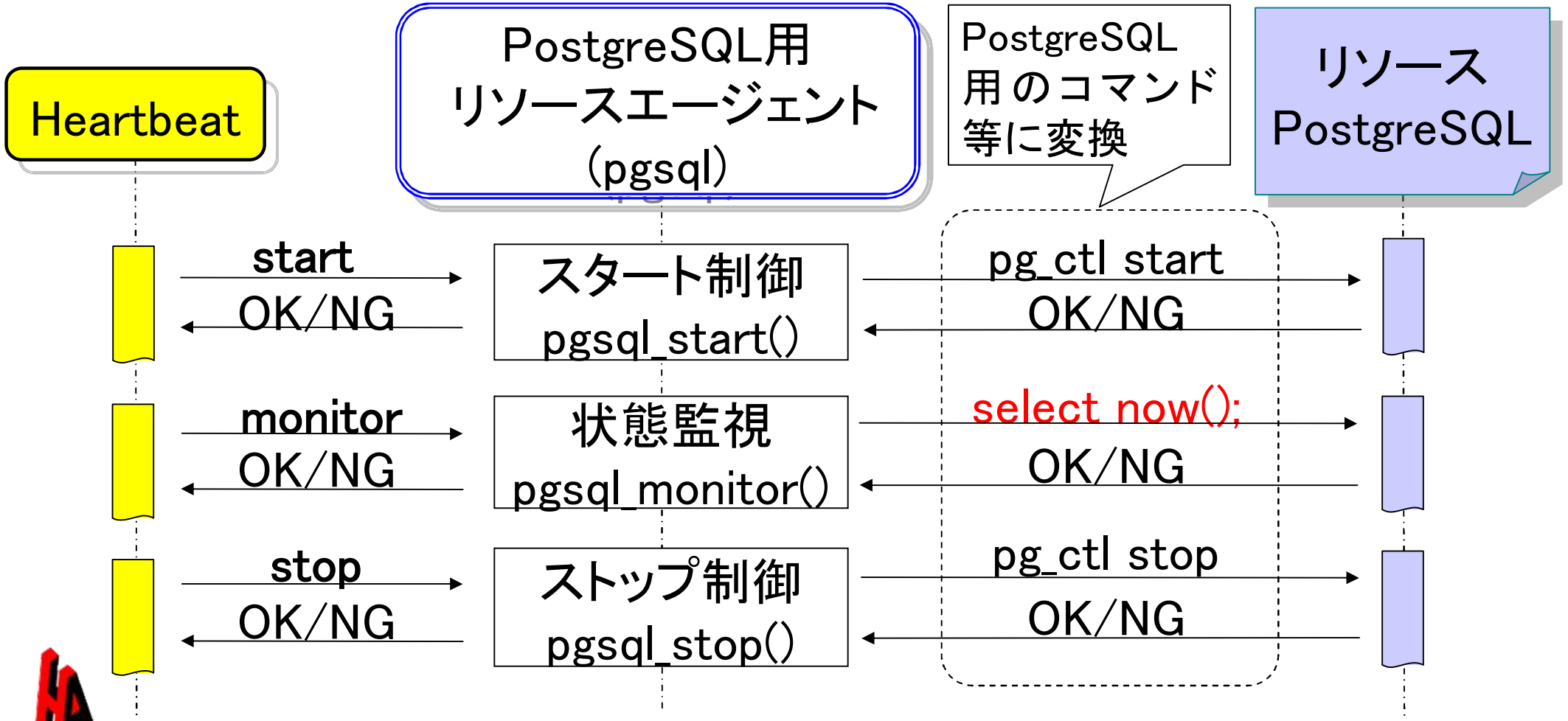
分類	リソース	リソースエージェント（/usr/lib/ocf/resource.d/heartbeat/）
ファイルシステム系	ディスクマウント	Filesystem
DB系	PostgreSQL	pgsql
Web系	Apache	apache
ネットワーク系	仮想IPアドレス	IPaddr

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）Heartbeat と PostgreSQLリソースエージェントの関係



②

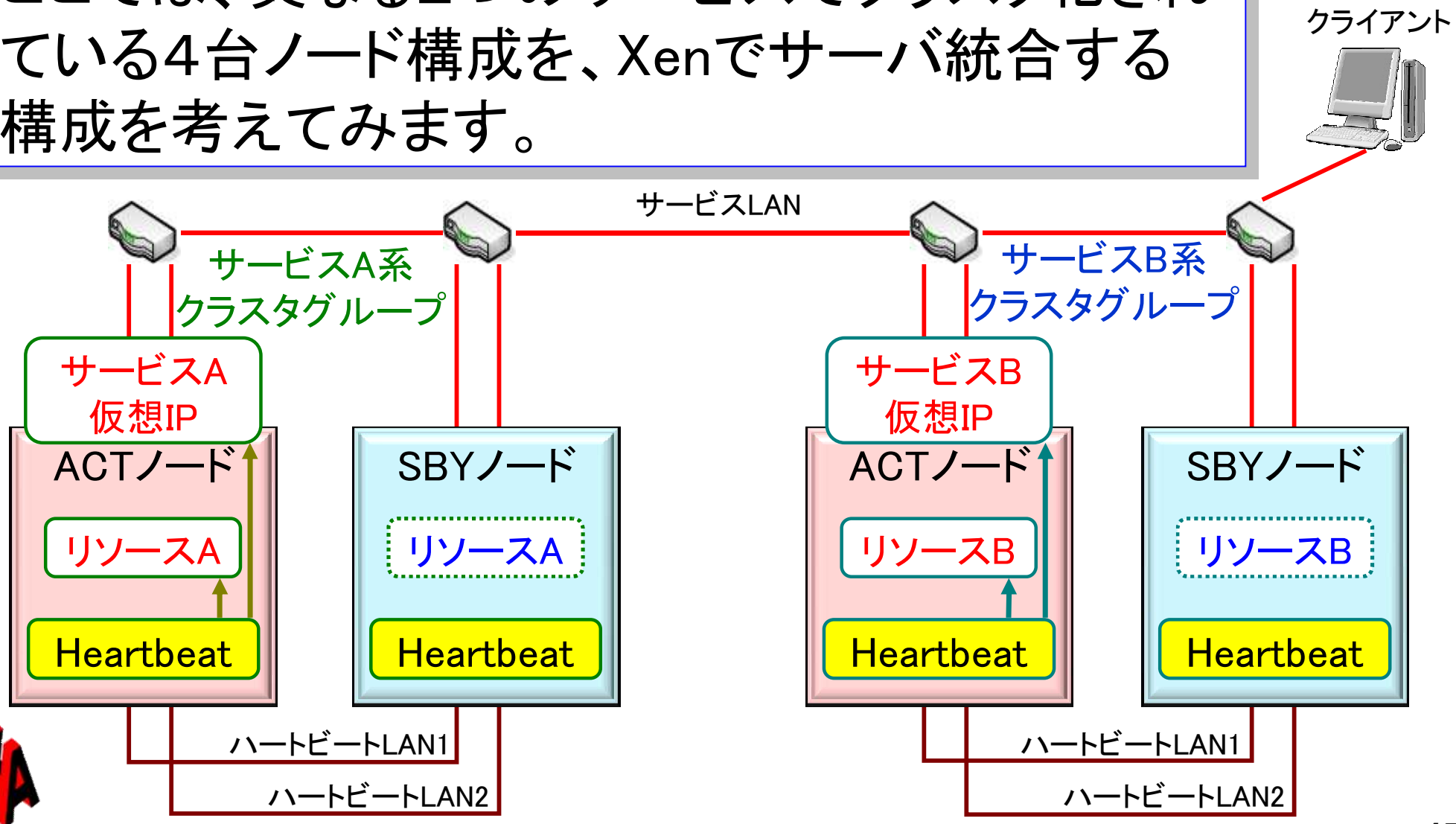
仮想化クラスタリング構成

ここからやっと本番？

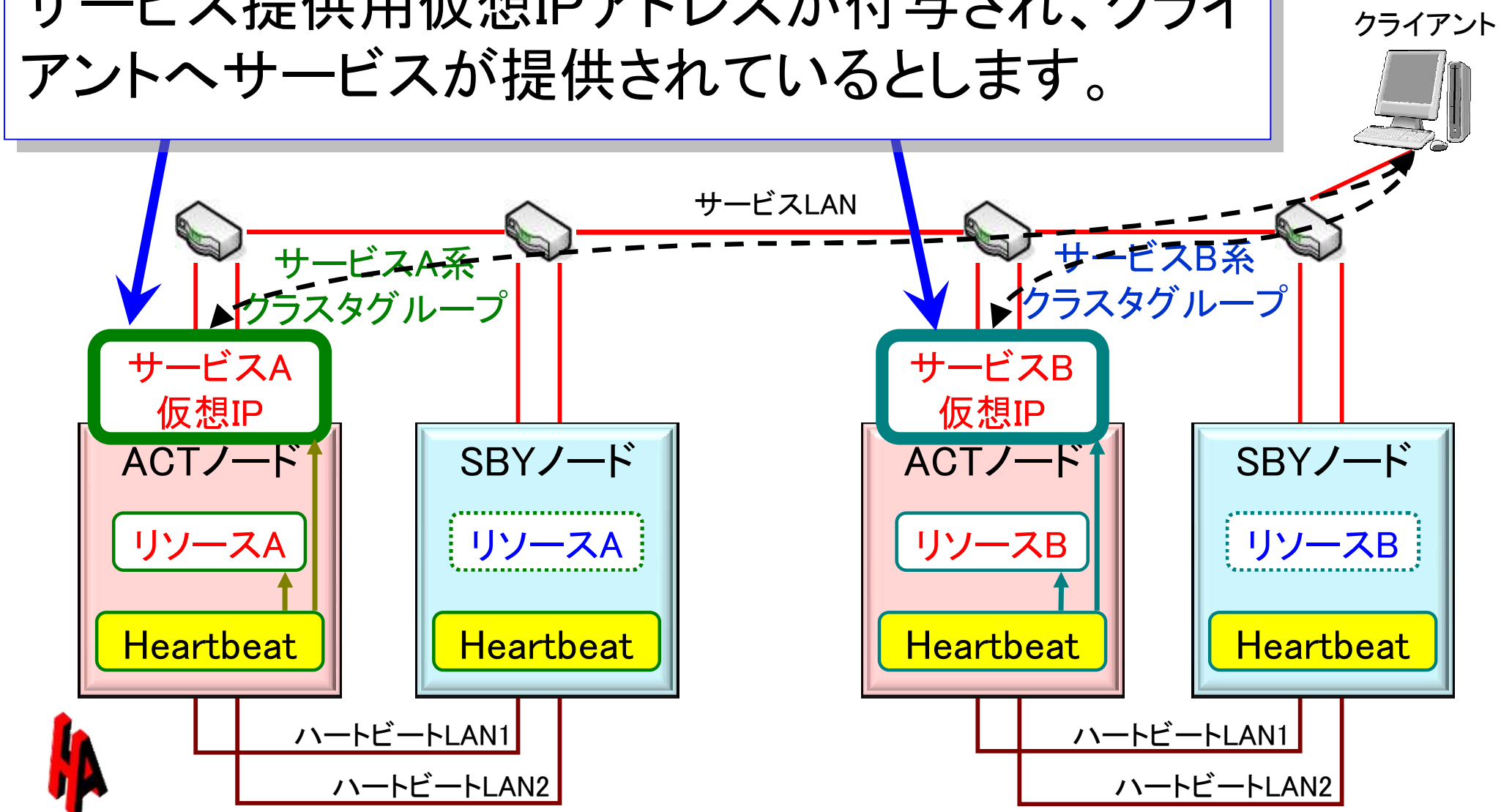
話を進めるにあたって
まずは
仮想化を行う
ターゲットの構成を決めます

仮想化のターゲットとするHA構成

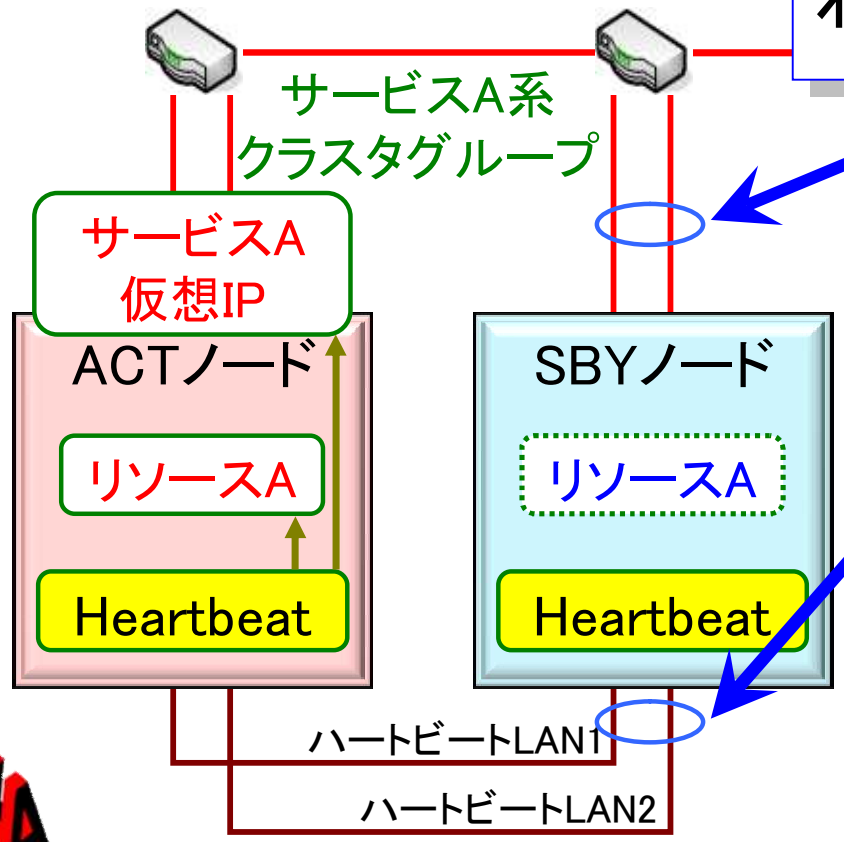
ここでは、異なる2つのサービスでクラスタ化されている4台ノード構成を、Xenでサーバ統合する構成を考えてみます。



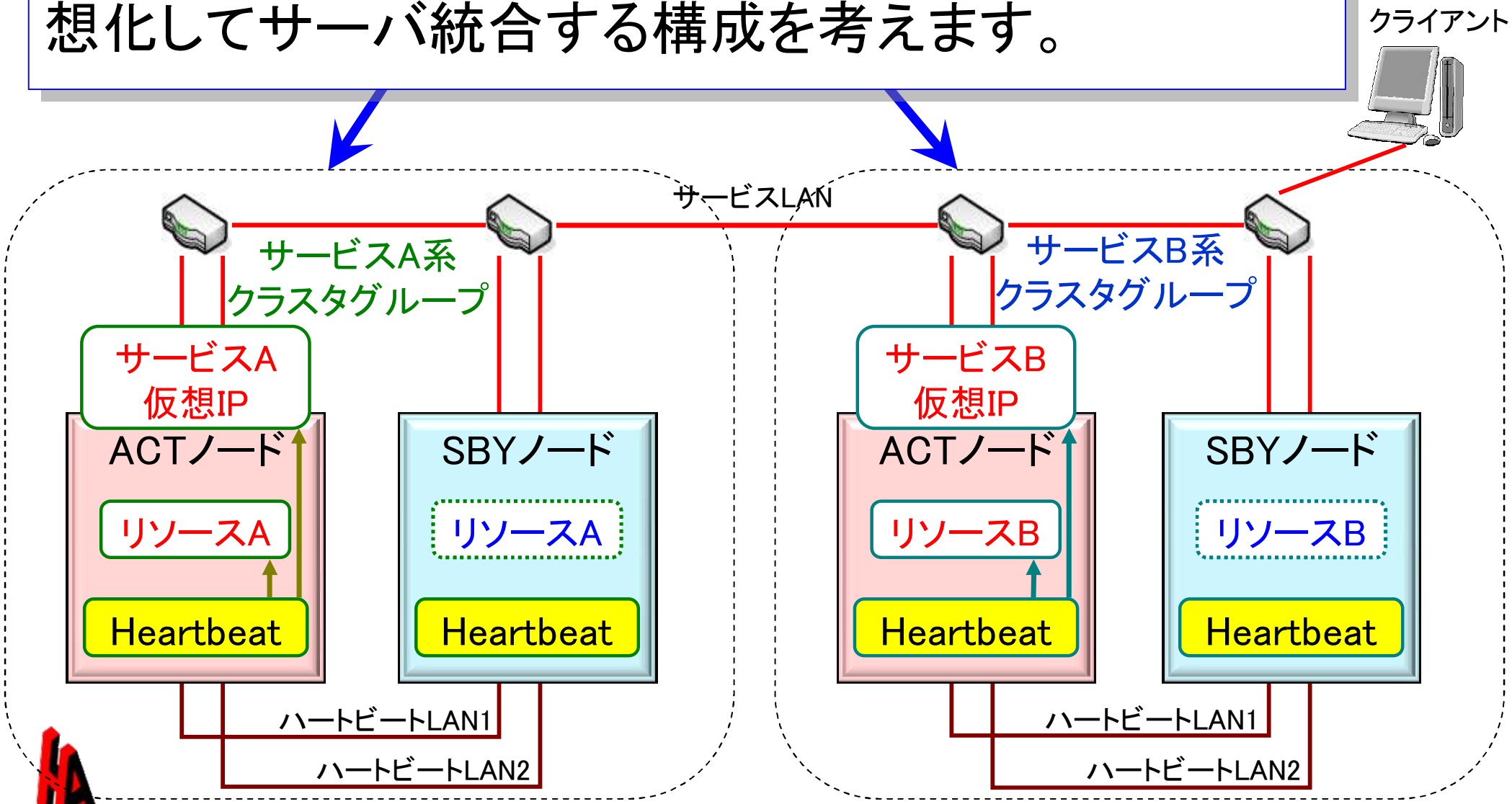
ACTノードには、サービスLAN側にHeartbeatからサービス提供用仮想IPアドレスが付与され、クライアントへサービスが提供されているとします。



すべてのノードで、サービスLAN、ハートビートLANは別経路が用意され、それぞれのLANは冗長化されているとします。

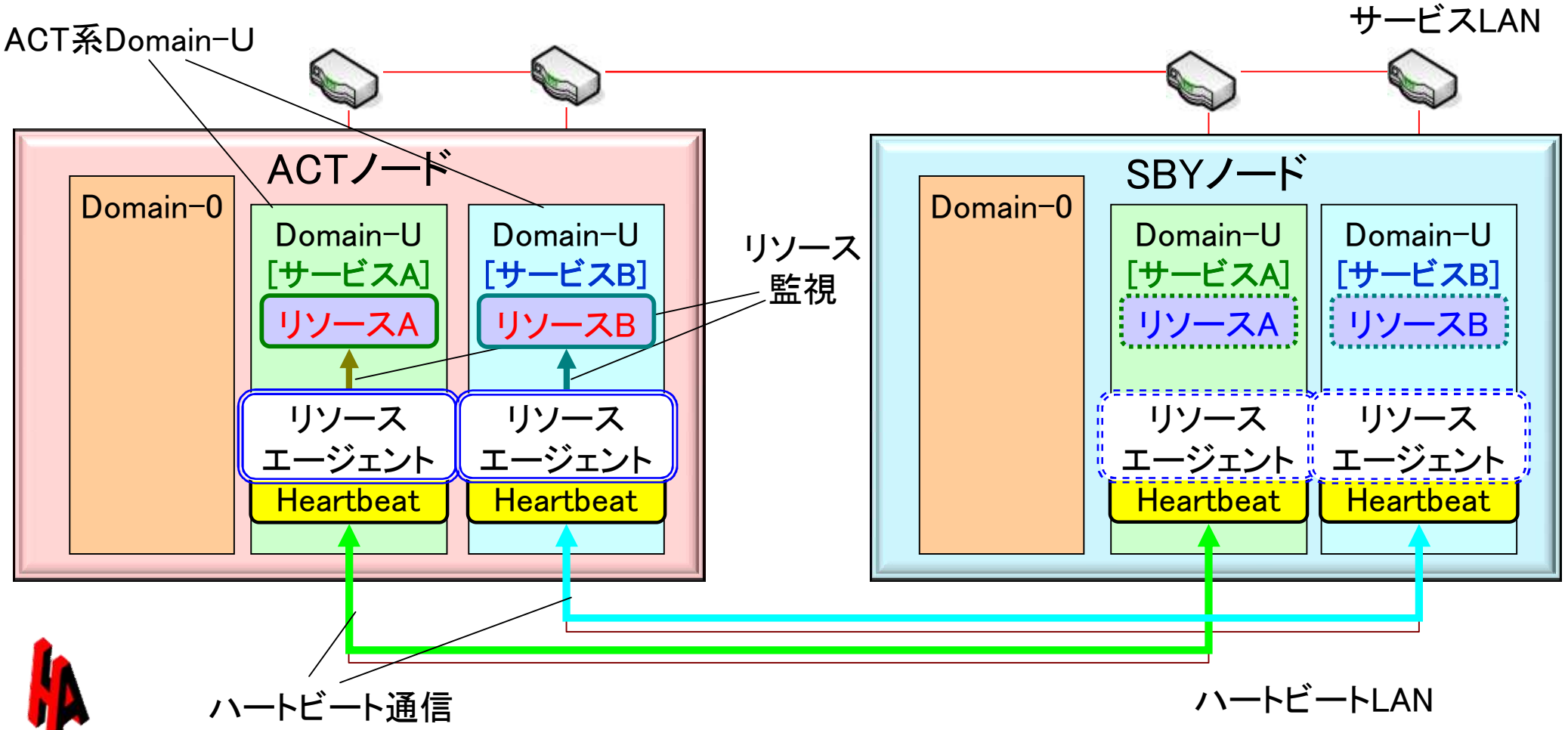


サービスA サービスB と、4ノードで別々のサービスが提供されているクラスタグループを、Xenにより仮想化してサーバ統合する構成を考えます。

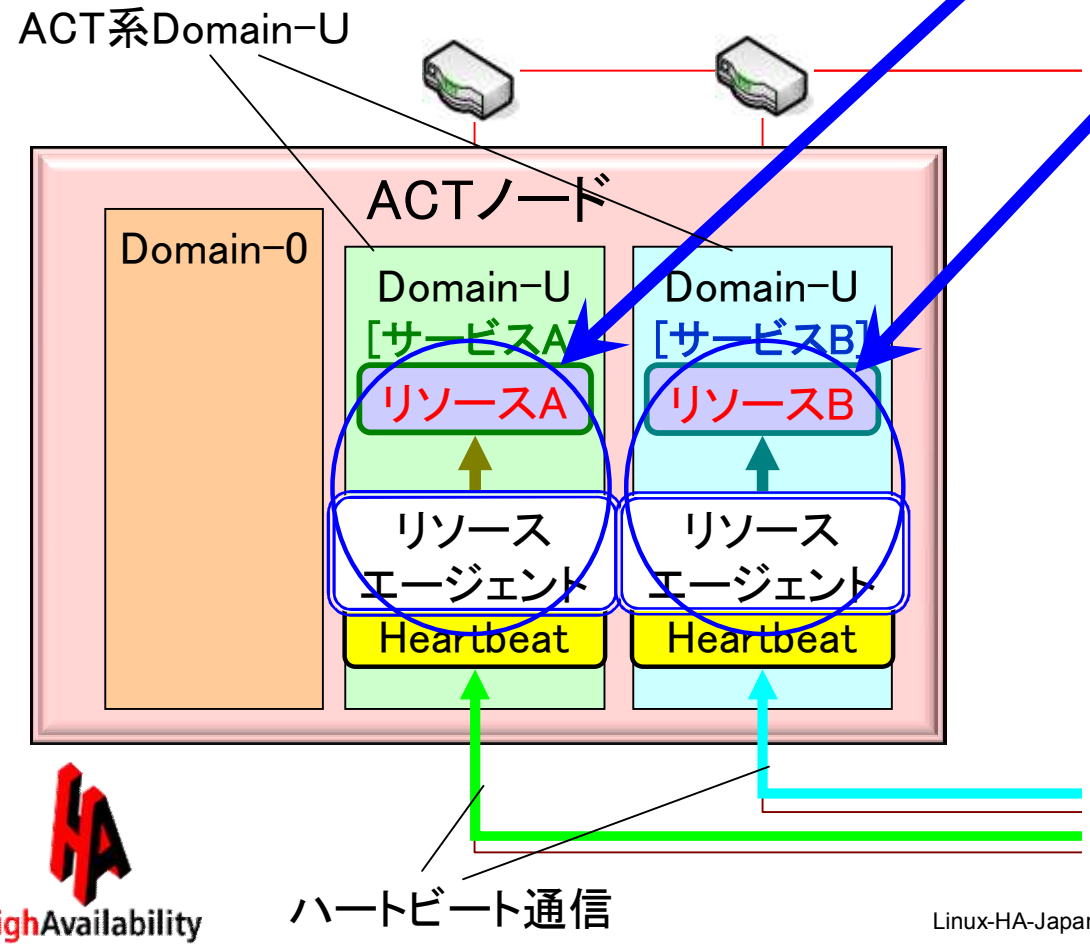


このターゲットで
仮想化クラスタリングの
HA構成を
考えてみました

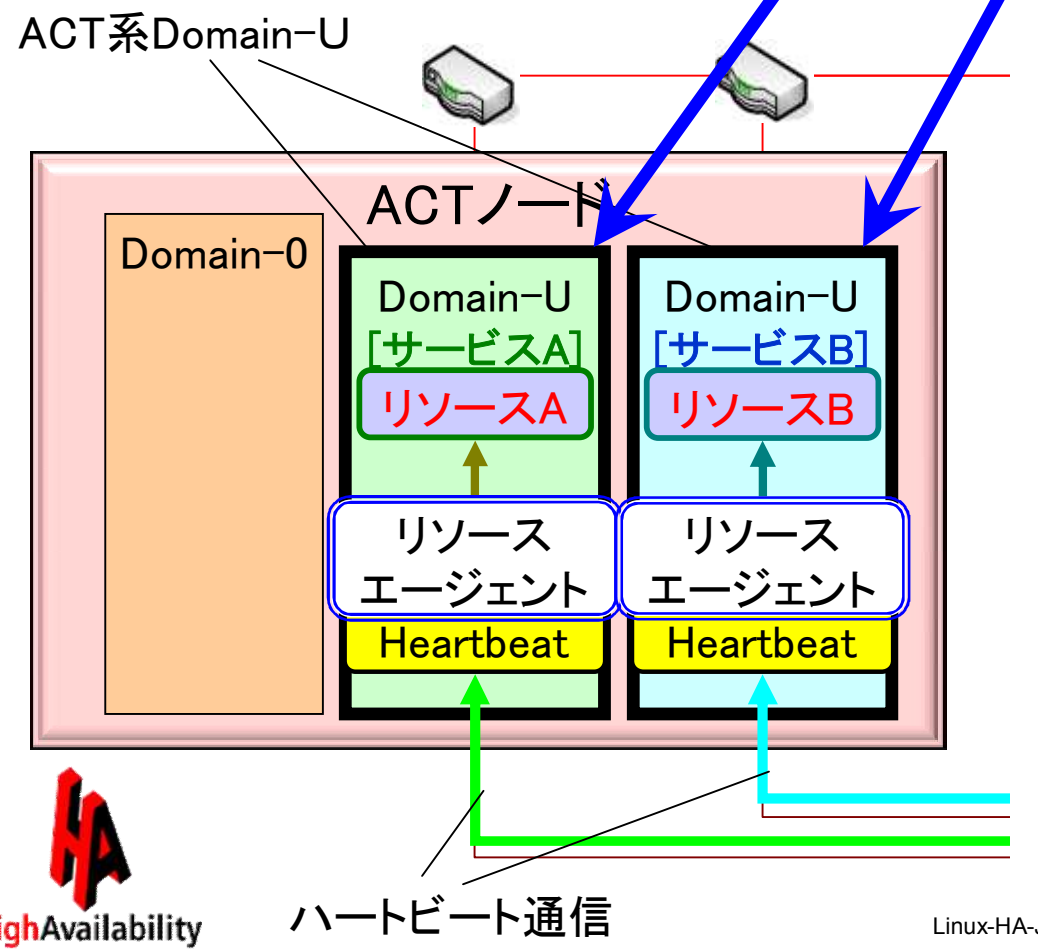
Heartbeatを各Domain-Uのゲスト OSにインストールし、ノード間のDomain-Uでハートビート通信を行い、HAクラスタを構成します。



リソース監視方法など、物理環境と同様に
リソースの制御が実現できます！

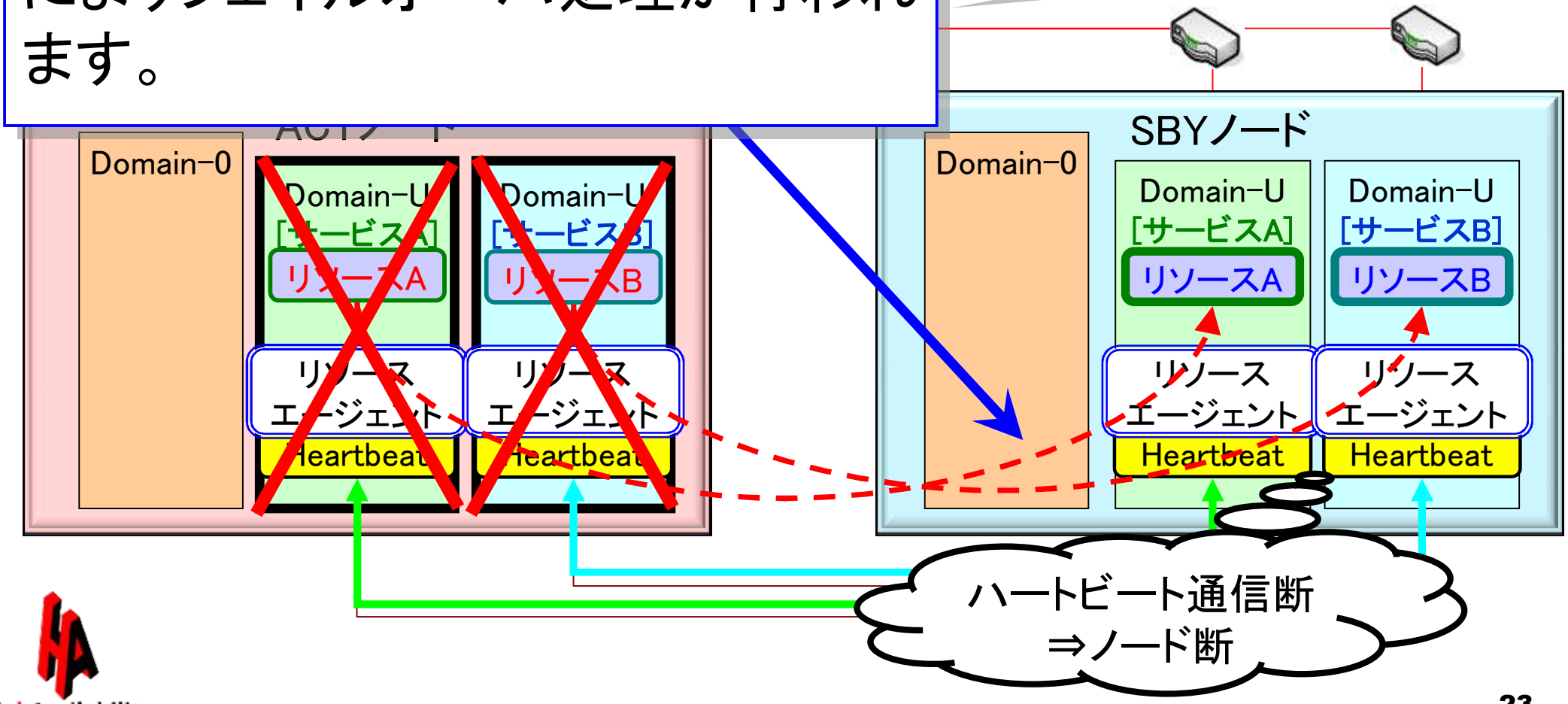


サービス層であるリソース監視のみで、
仮想マシン層であるDomain-Uの監視が
行われていないが…？



Domain-Uが停止している、またはホストOSであるDomain-0が停止している場合は、ハートビート通信断によりフェイルオーバ処理が行われます。

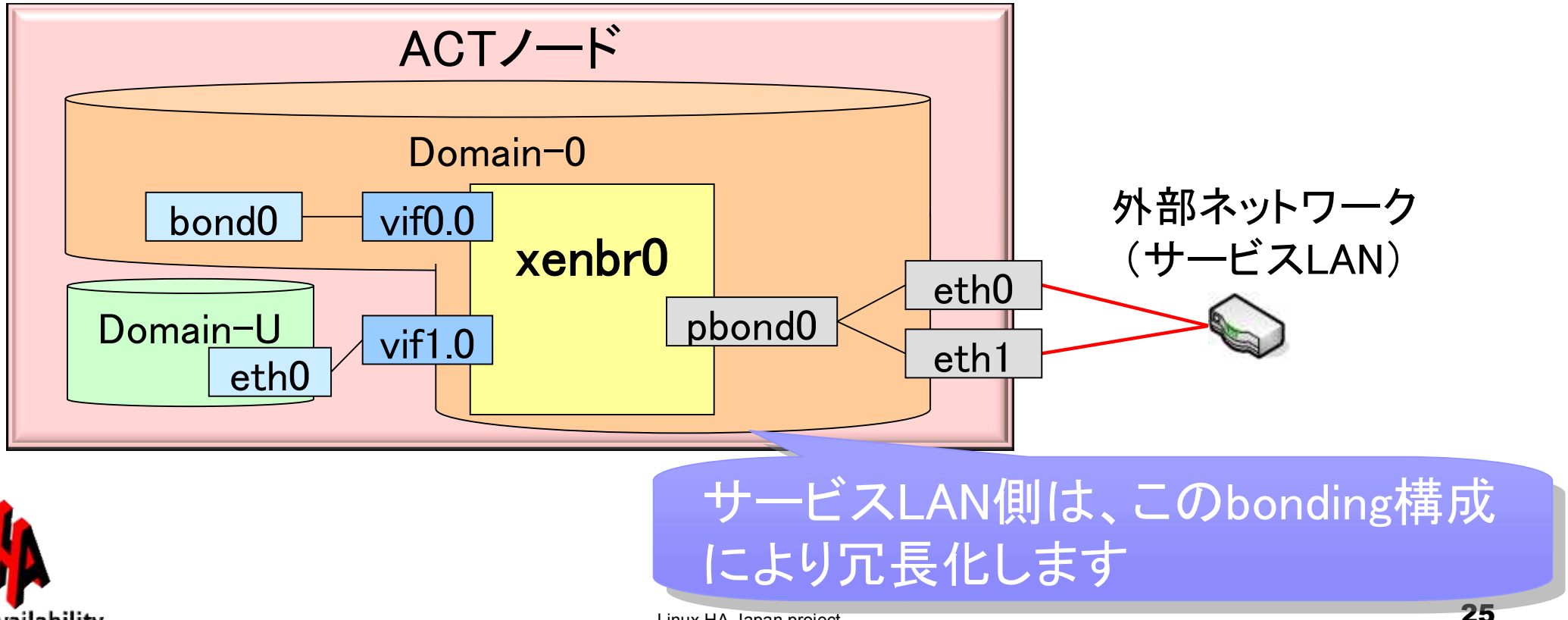
構成に面白味は無いけど、Domain-Uを物理環境（ハードウェア）とみなすシンプルな構成ですね！



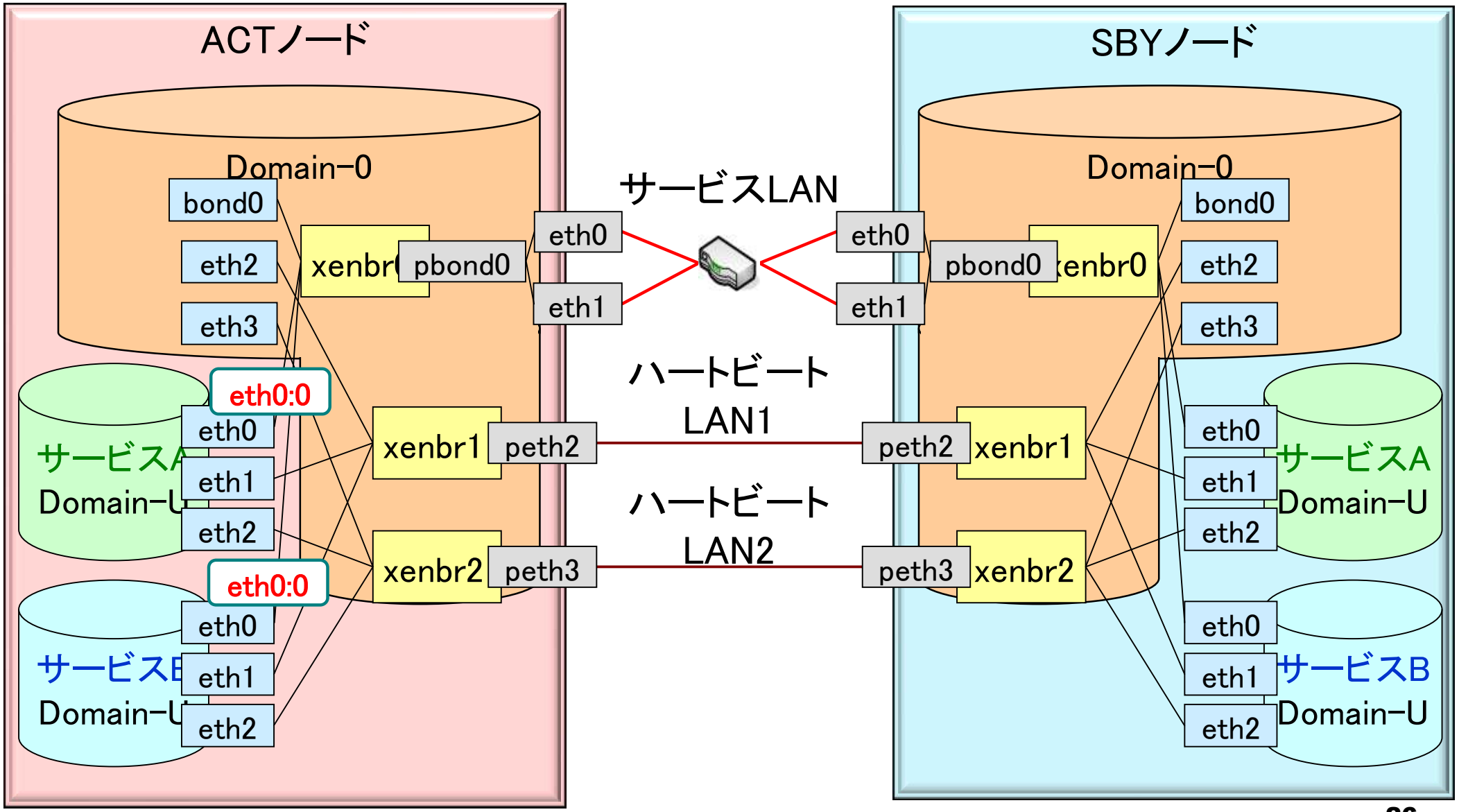
仮想化クラスタリングに適した
ネットワークの構成を
考えてみました

仮想ブリッジとbonding構成

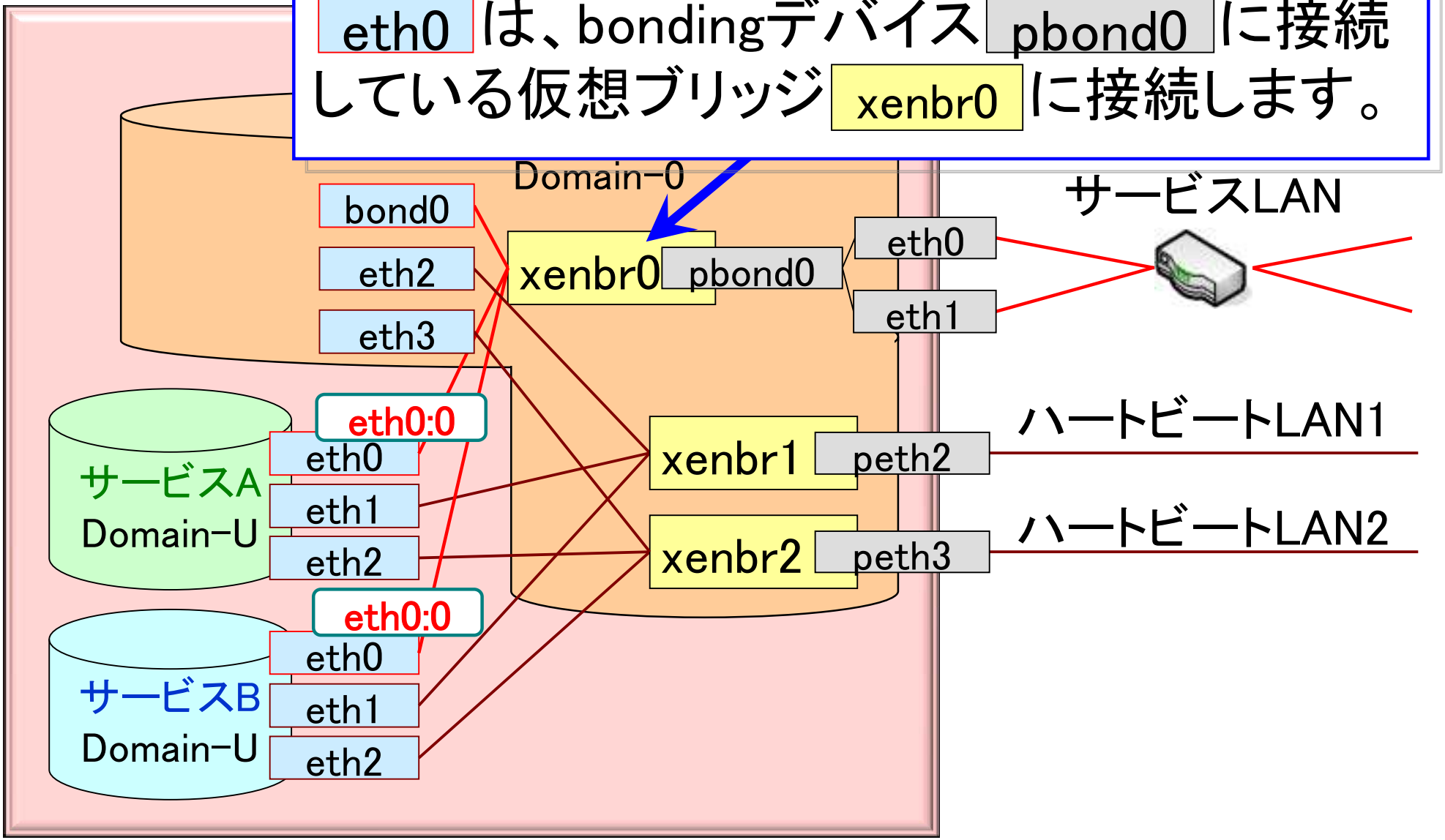
ネットワークの冗長化は、Domain-0で bondingを構成し、仮想ブリッジと接続することで実現可能です。
Domain-U は bonding と結び付けられている仮想ブリッジに接続させるだけで bonding による冗長化の効果を得ることができます。



仮想化クラスタリングのネットワーク構成

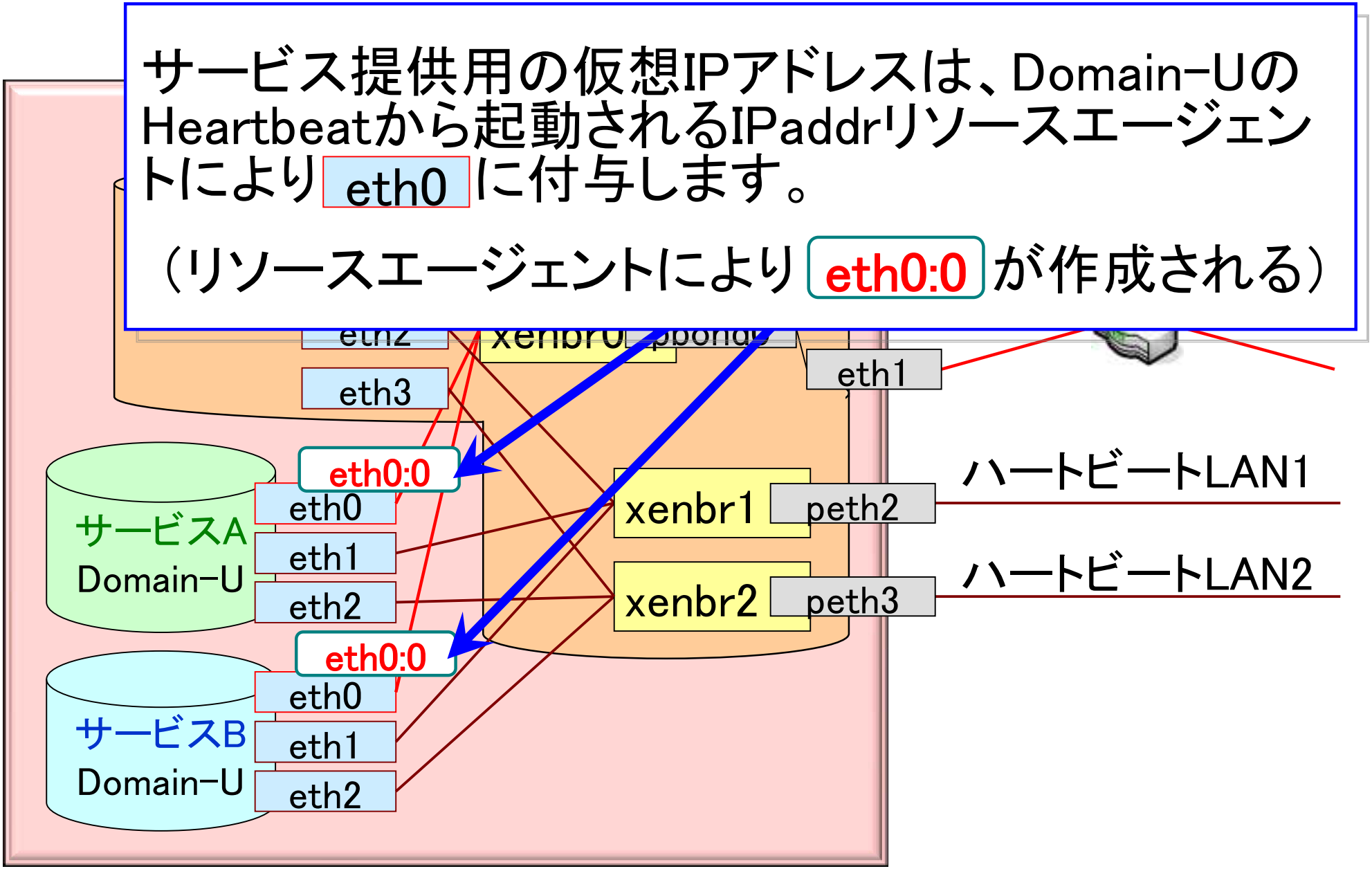


Domain-Uのサービス LAN側インターフェース eth0 は、bondingデバイス pbond0 に接続している仮想ブリッジ xenbr0 に接続します。

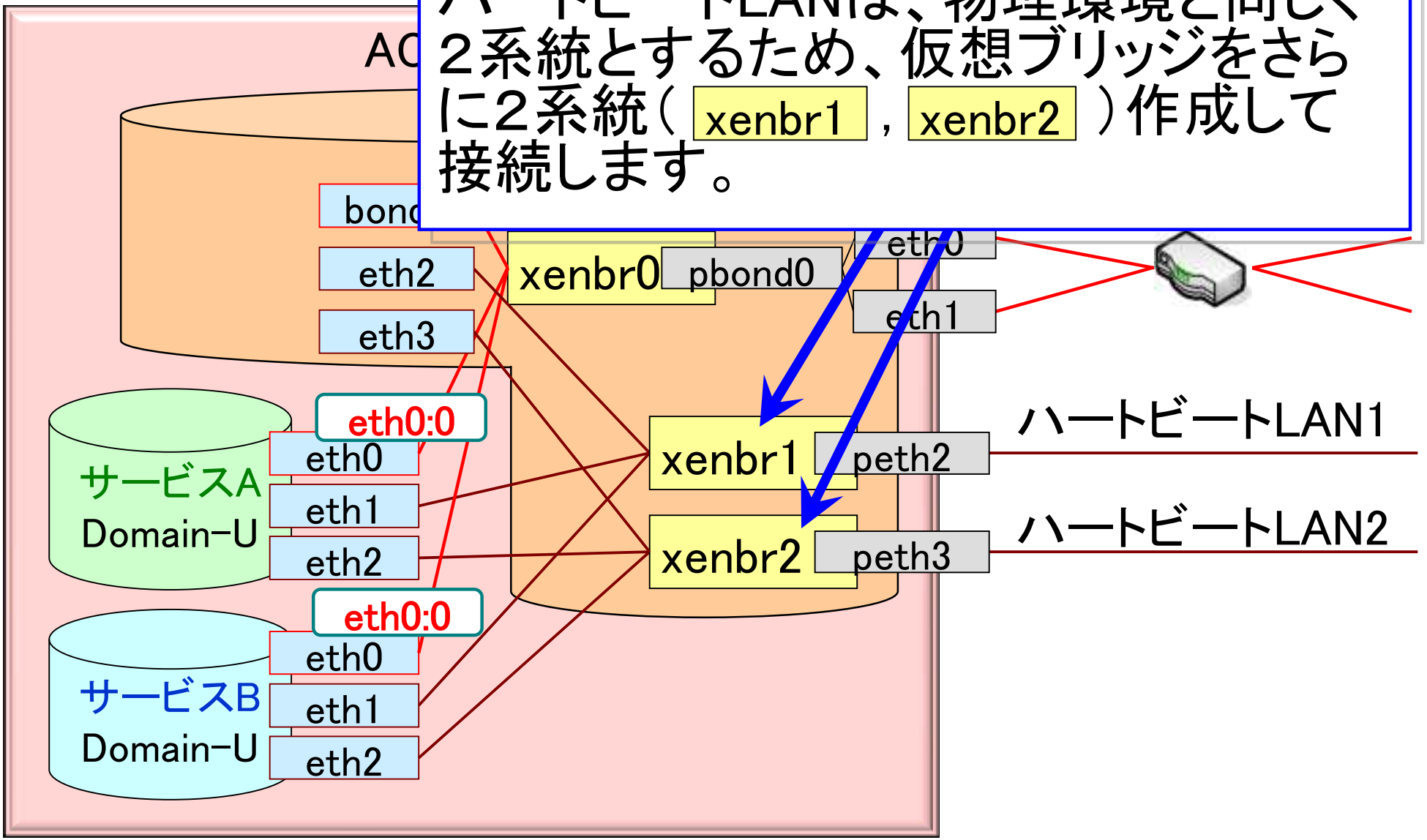


サービス提供用の仮想IPアドレスは、Domain-Uの Heartbeatから起動されるIPaddrリソースエージェントにより eth0 に付与します。

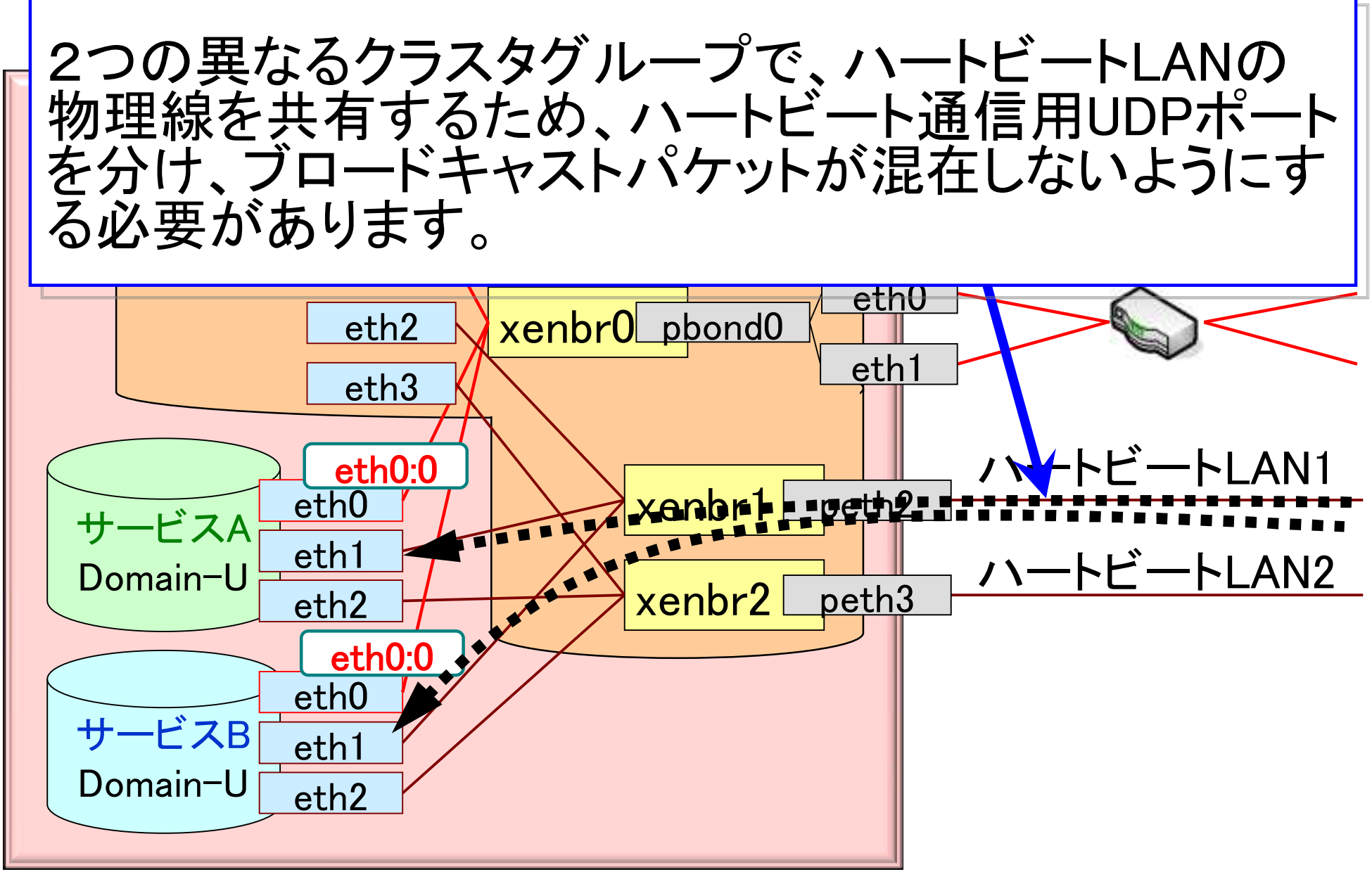
（リソースエージェントにより eth0:0 が作成される）



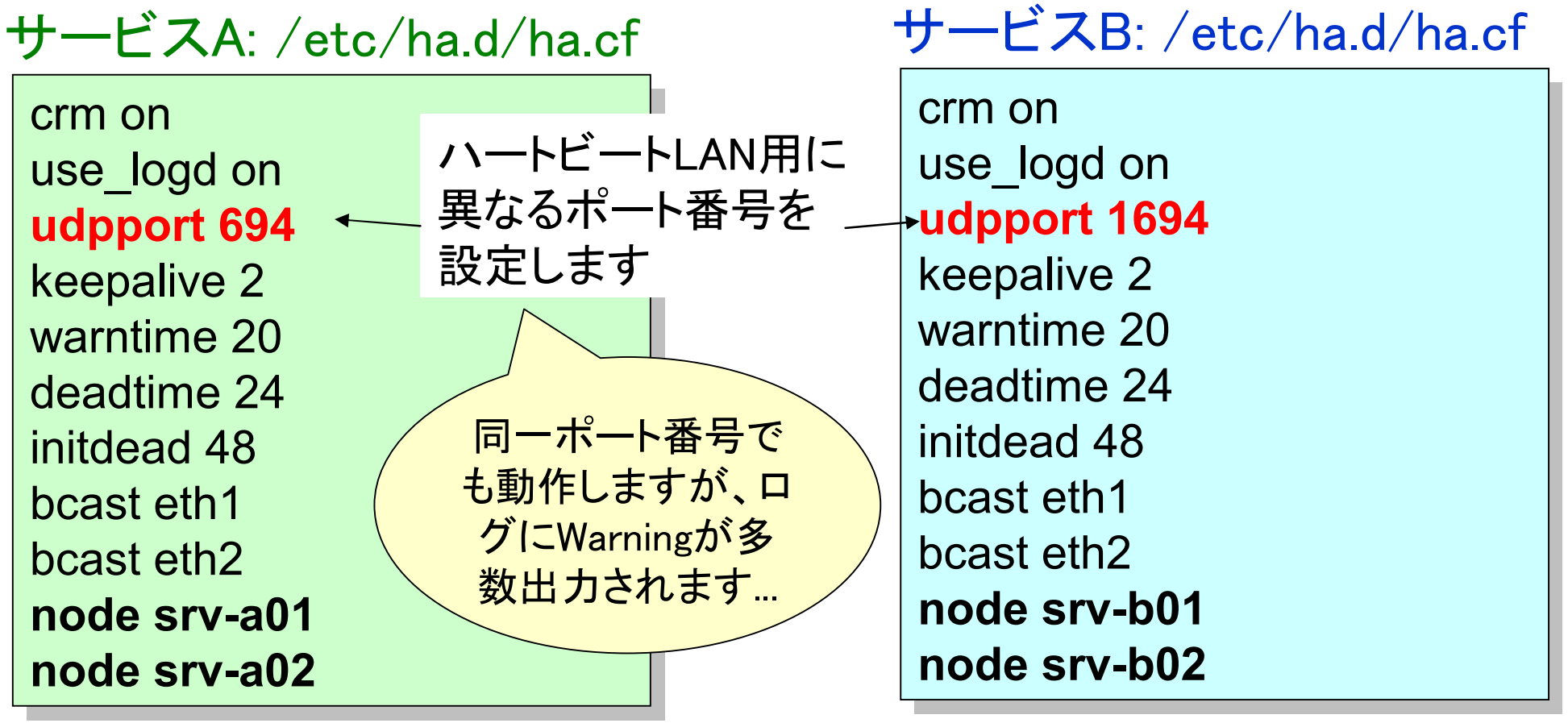
ハートビートLANは、物理環境と同じく２系統とするため、仮想ブリッジをさらに２系統（ xenbr1 ， xenbr2 ）作成して接続します。



2つの異なるクラスタグループで、ハートビートLANの物理線を共有するため、ハートビート通信用UDPポートを分け、ブロードキャストパケットが混在しないようにする必要があります。



- ハートビート通信用ポートを変更するためには、Heartbeatの基本設定ファイル(/etc/ha.d/ha.cf)に、異なるポート番号の設定が必要です。



仮想ネットワークの設定方法

- ブリッジ作成用カスタムスクリプト(network-custom-hoge)を新規に作成します。
  - /etc/xen/scripts/network-custom-hoge の例

```
#!/bin/bash

script=/etc/xen/scripts/network-bridge
$script start vifnum=0 bridge=xenbr0 netdev=bond0
$script start vifnum=1 bridge=xenbr1 netdev=eth2
$script start vifnum=2 bridge=xenbr2 netdev=eth3
```

仮想ブリッジxenbr0 に物理デバイスbond0を関連付ける

- Xenデーモンの構成ファイル(xend-config.sxp)を修正して、作成したカスタムスクリプト(network-custom-hoge)を読み込ませるように設定します。
- 設定後、Domain-0を再起動させると新しい仮想ブリッジが作成されます。

  - /etc/xen/xend-config.sxp の修正箇所

```
#(network-script network-bridge)
(network-script network-custom-hoge)
```

新規に作成したスクリプト名を記載する

※ Xen仮想ブリッジの作成方法は、HAに特化した設定ではなく、インターネット上では多数紹介されています！

あとは、物理環境と同様に Heartbeatを設定して 起動するだけ！

```
# service heartbeat start
Starting High-Availability service:    [OK]
```

③

仮想化クラスタリング
応用構成編

仮想化クラスタリングでの STONITH構成を 考えてみました

STONITHとは？

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

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

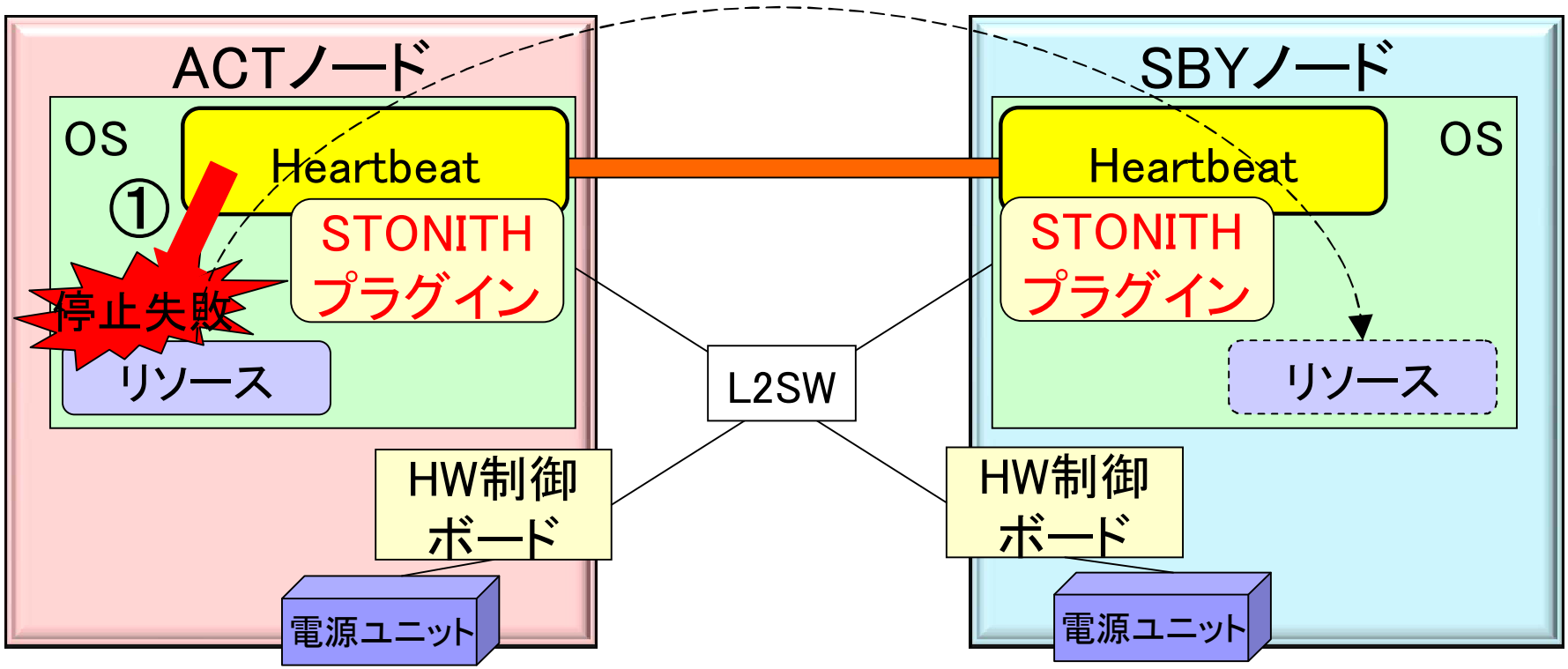
HW制御ボードの例

- IBM社
  - リモート管理アダプターⅡ SlimLine【RSAⅡ】
    - System x® 3650（旧モデル）等にオプションで搭載が可能
  - Integrated Management Module【IMM】
    - System x® 3650 M2（新モデル）等に標準搭載
- HP社
  - Integrated Lights-Out 2【iLO2】
    - ProLiant DL380 G6 等に標準搭載

▲ iLO2

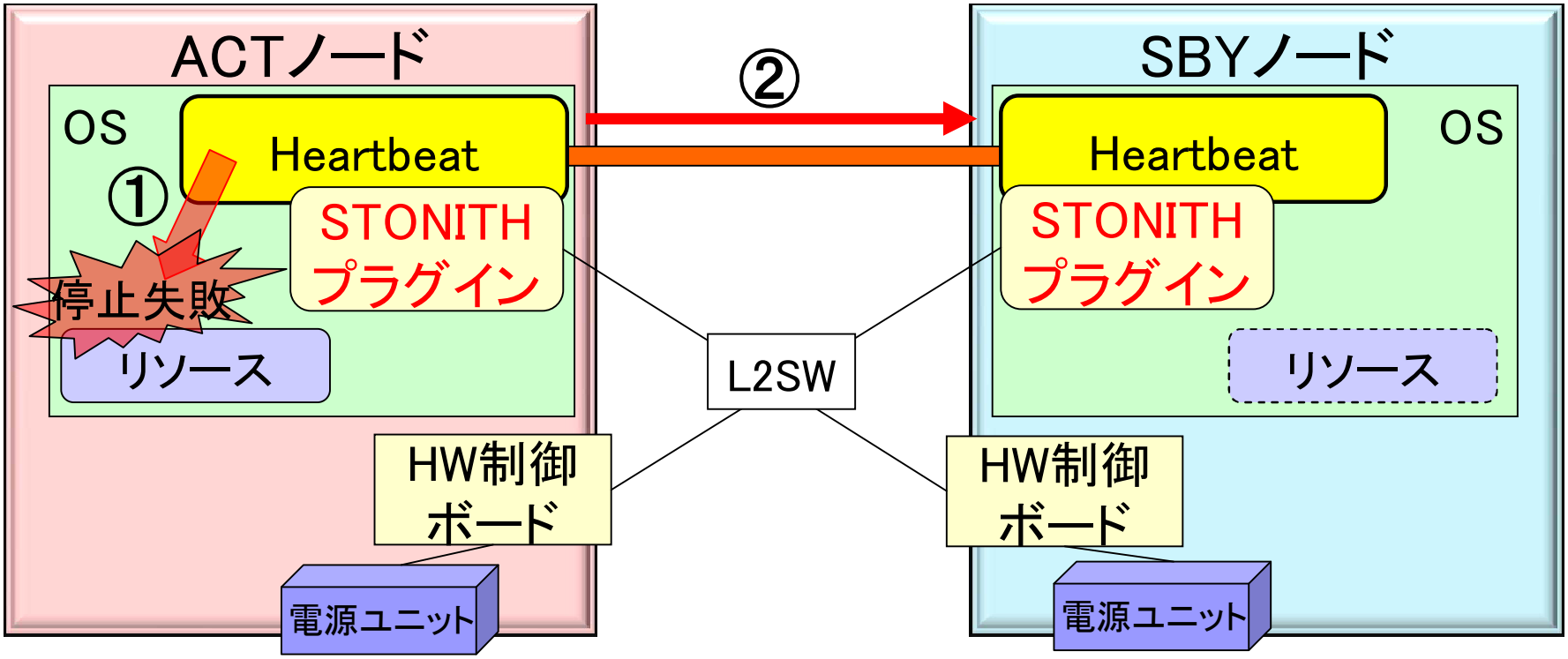
フェイルオーバ時にSTONITH機能が発動されるまでの流れ①

① フェイルオーバ時にACTノードのリソース停止処理がNGとなる事象が発生



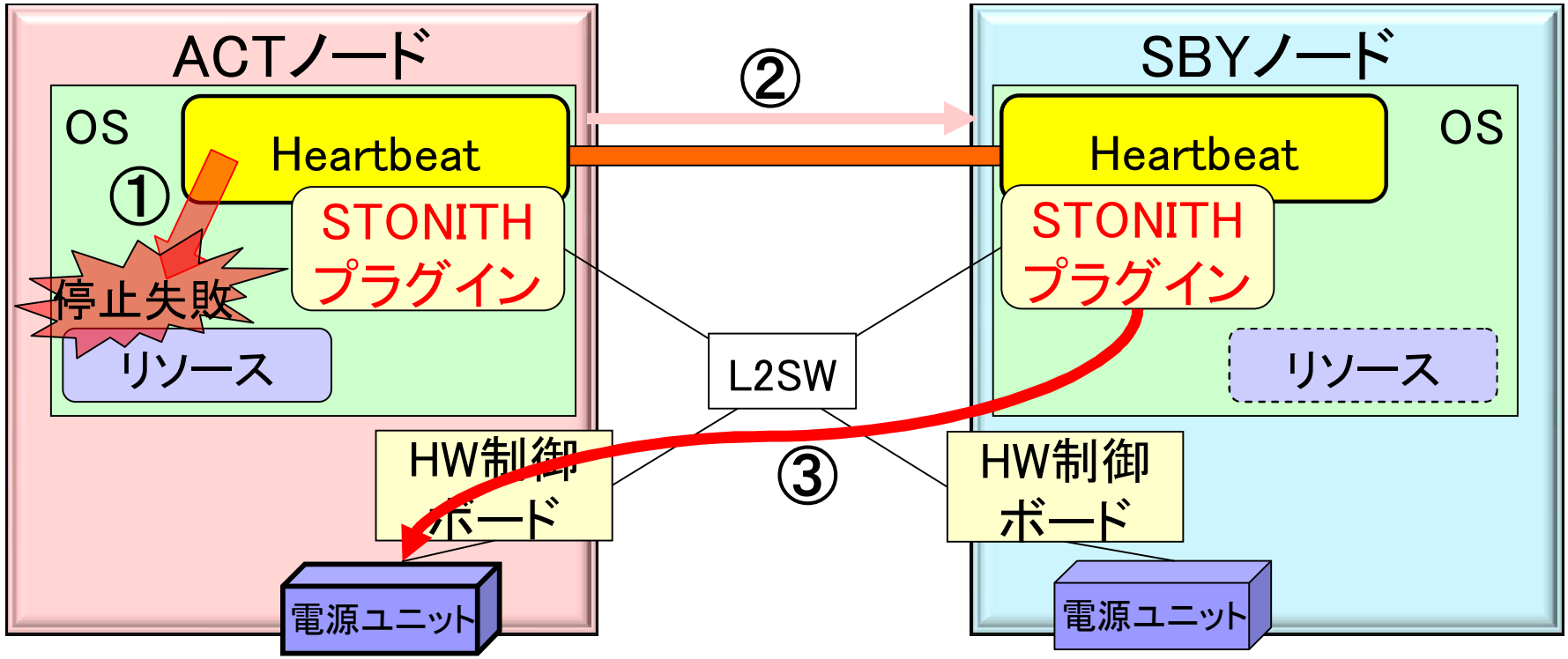
フェイルオーバ時にSTONITH機能が発動されるまでの流れ②

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



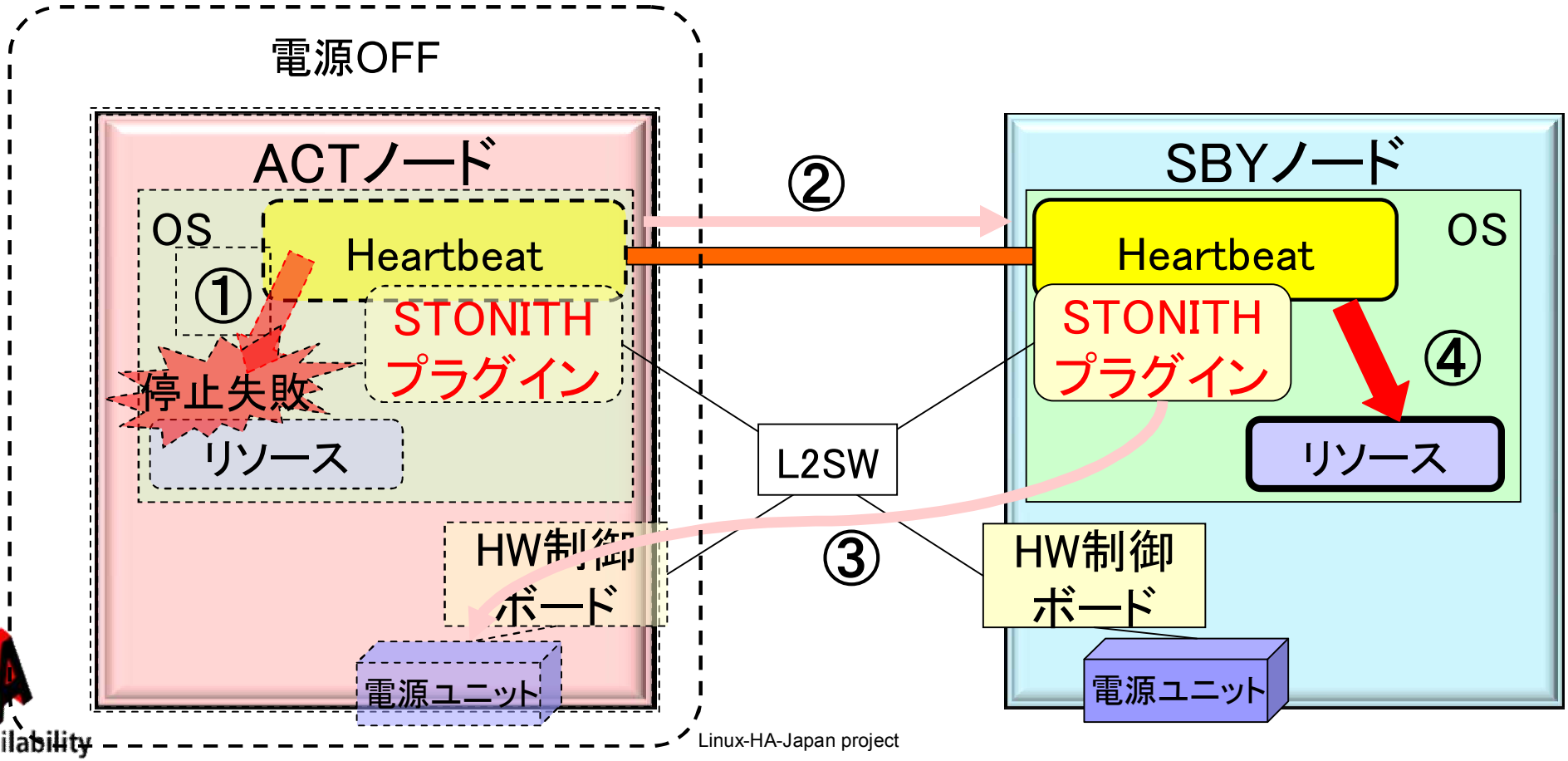
フェイルオーバ時にSTONITH機能が発動されるまでの流れ③

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



フェイルオーバ時にSTONITH機能が発動されるまでの流れ④

④ 強制電源断成功後、リソースが起動



STONITHプラグイン

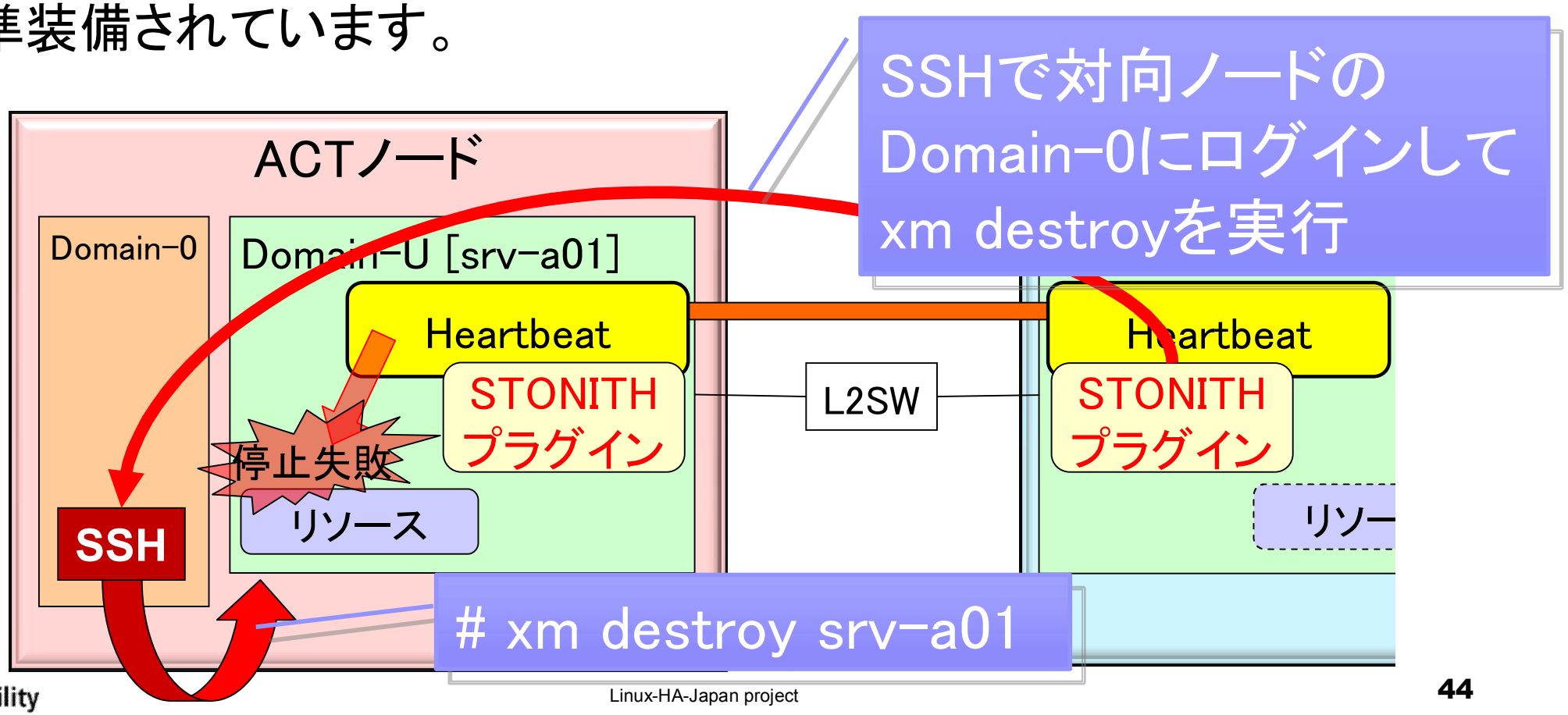
Heartbeatには、様々なHW制御ボードに対応したSTONITHプラグインが標準装備されています。
プラグインは、シェルスクリプト、Perl、Python等で作成されています。

- HP iLO2用プラグイン
  (/usr/lib64/stonith/plugins/external/riloe)
- IBM RSA Ⅱ 用プラグイン
  (/usr/lib64/stonith/plugins/external/ibmrsa-telnet)
- IPMI用プラグイン
  (/usr/lib64/stonith/plugins/external/ipmi)

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

「xen0」STONITHプラグイン
(/usr/lib64/stonith/plugins/external/xen0)

HW制御ボード操作の代わりに SSH経由で xm destroyコマンドを Domain-0からDomain-Uの強制断を行うSTONITHプラグインが標準装備されています。



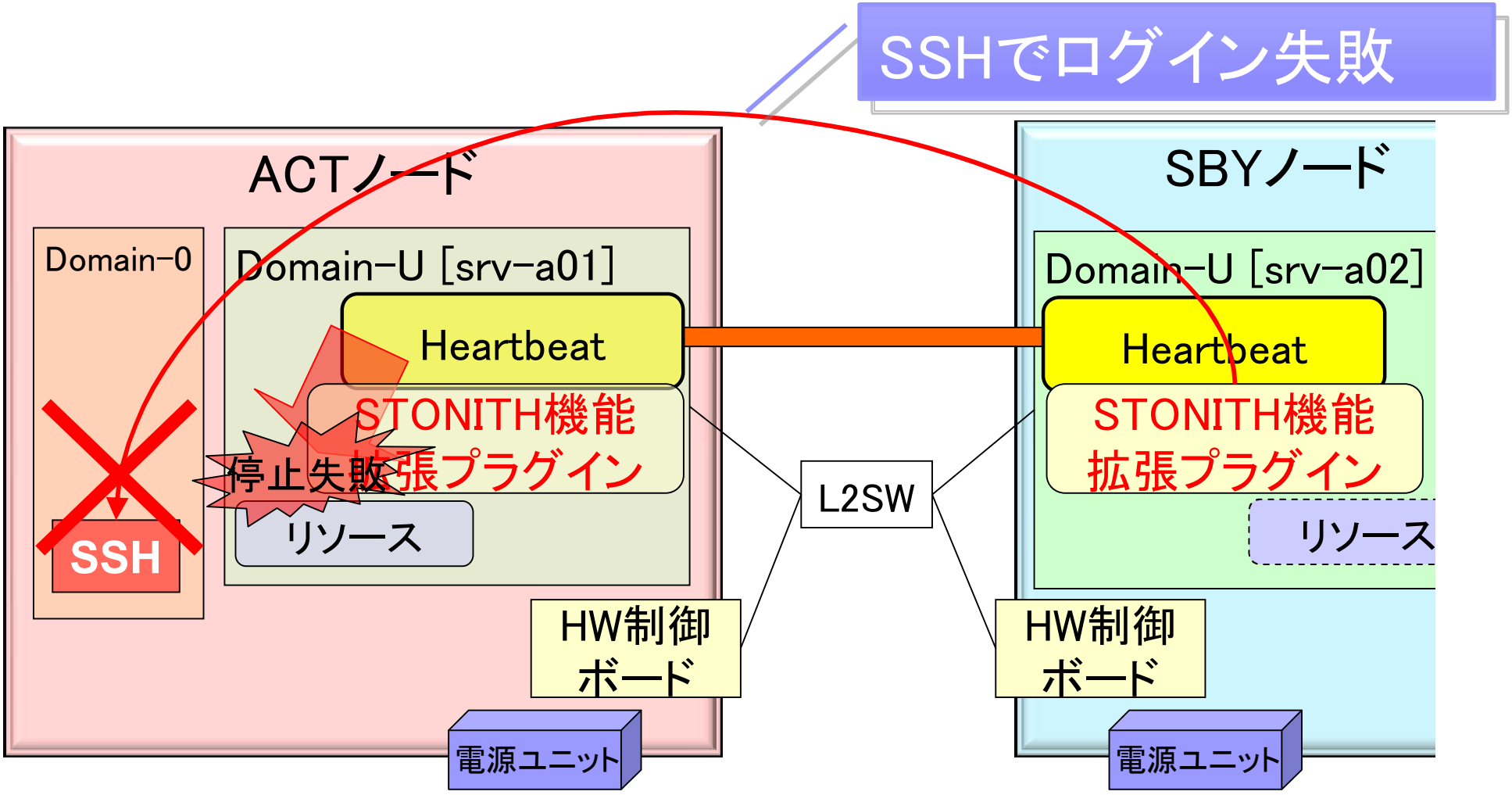
しかし…
Domain-0にSSHログイン
不可な故障状態ならば
STONITH機能の発動が
できないのでは…？

STONITHエスカレーション機能

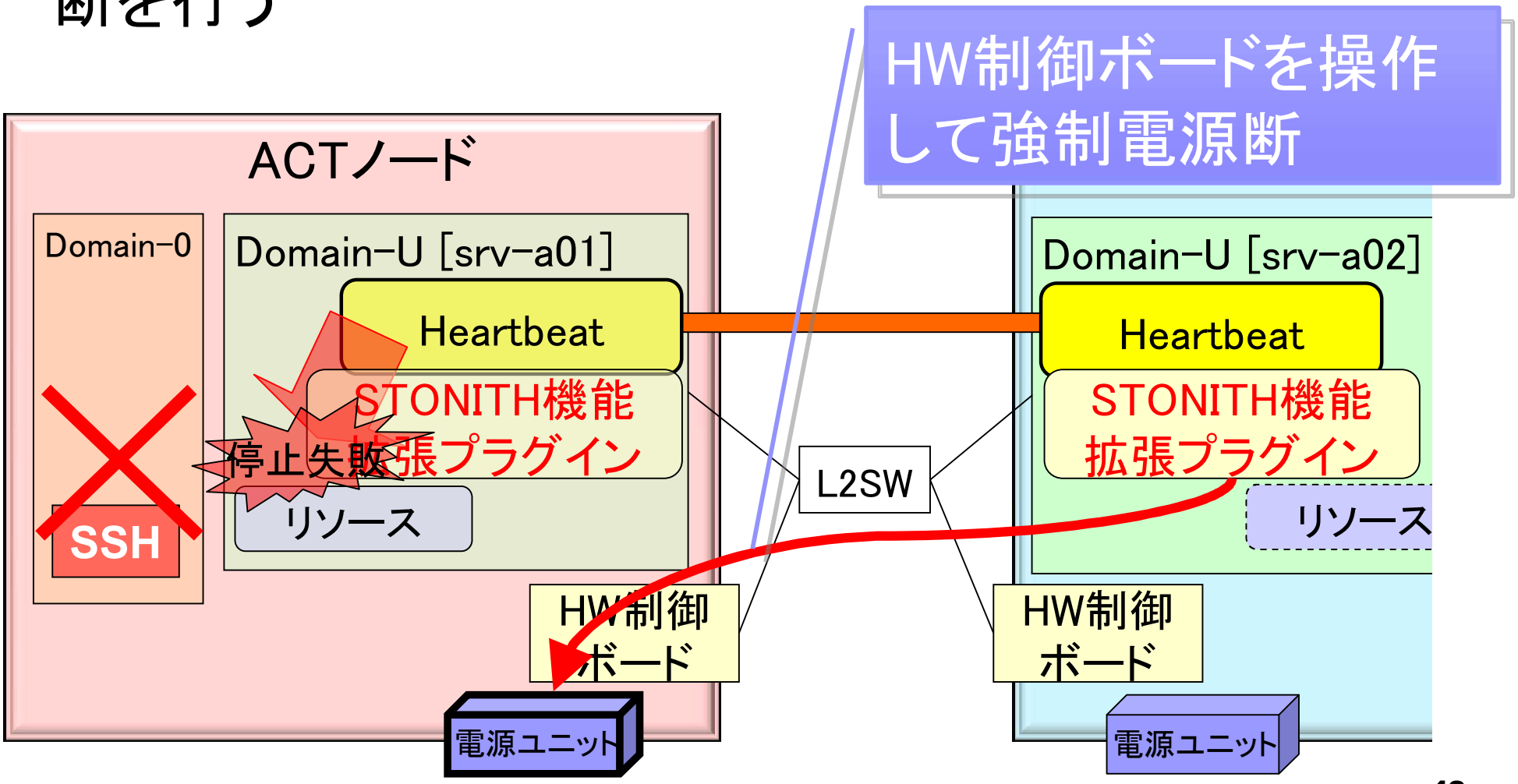
SSHでログイン出来ない場合でも、STONITH機能拡張プラグインによるエスカレーション機能により、HW制御ボードを利用して強制離脱させる事が可能になります。

Linux-HA-Japanプロジェクトより
機能拡張プラグインを3月公開予定！！

① 仮想環境でのSTONITH発動時、SSHで対向ノードへのログインが失敗



② SSHログインが失敗した場合、エスカレーション機能で物理環境と同様にHW制御ボードを操作して強制電源断を行う



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.* 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。

Linux-HA-JapanプロジェクトURL

SourceForge.JP に開設中

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

RHEL用 Heartbeat-2.* 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 等が提供中で、これからも多数の追加パッケージ公開を予定しています。

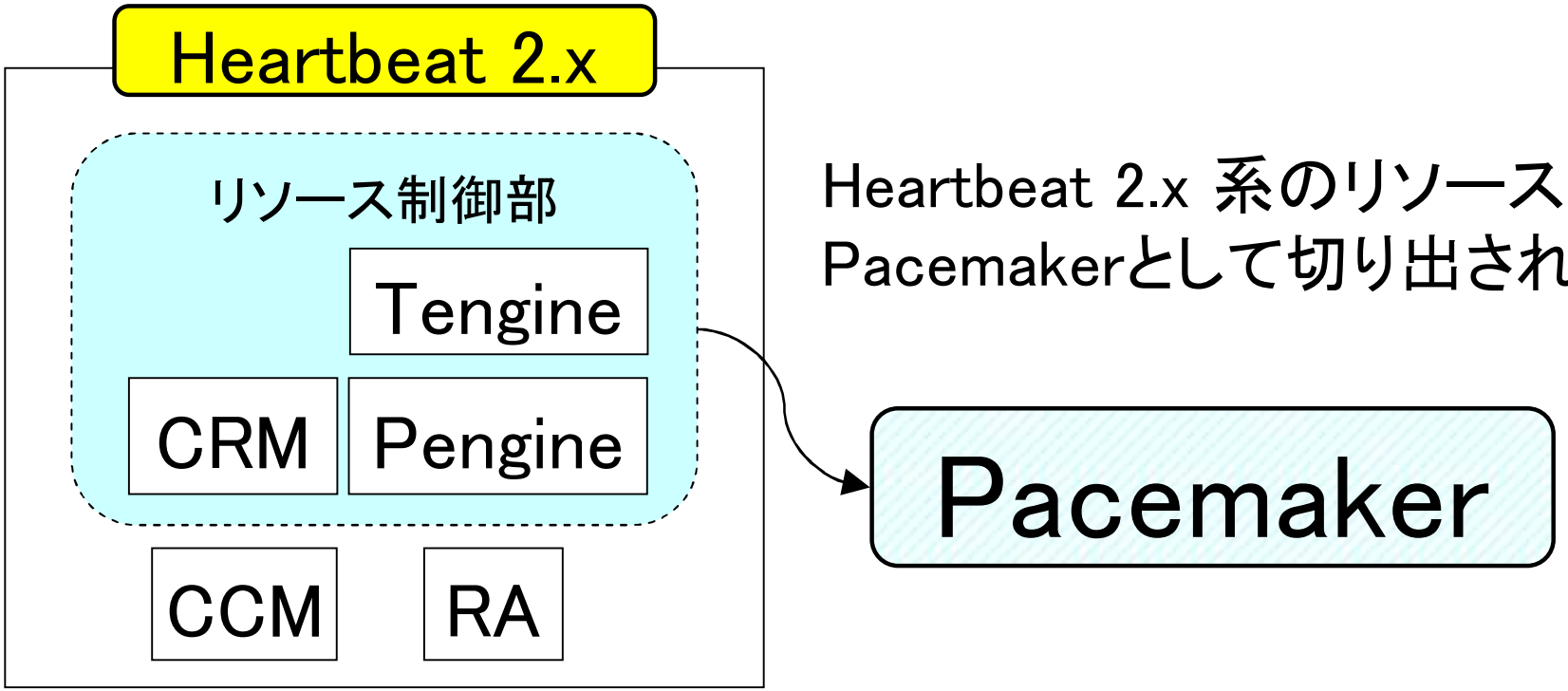
本家Linux-HAサイト

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

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

Heartbeatの次期バージョンってどうなっているの？

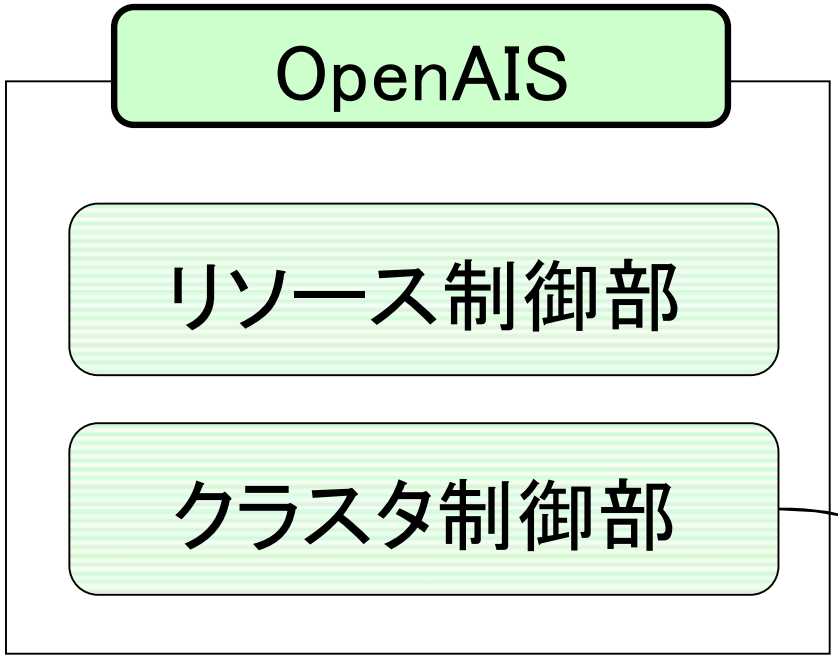
Pacemaker



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

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

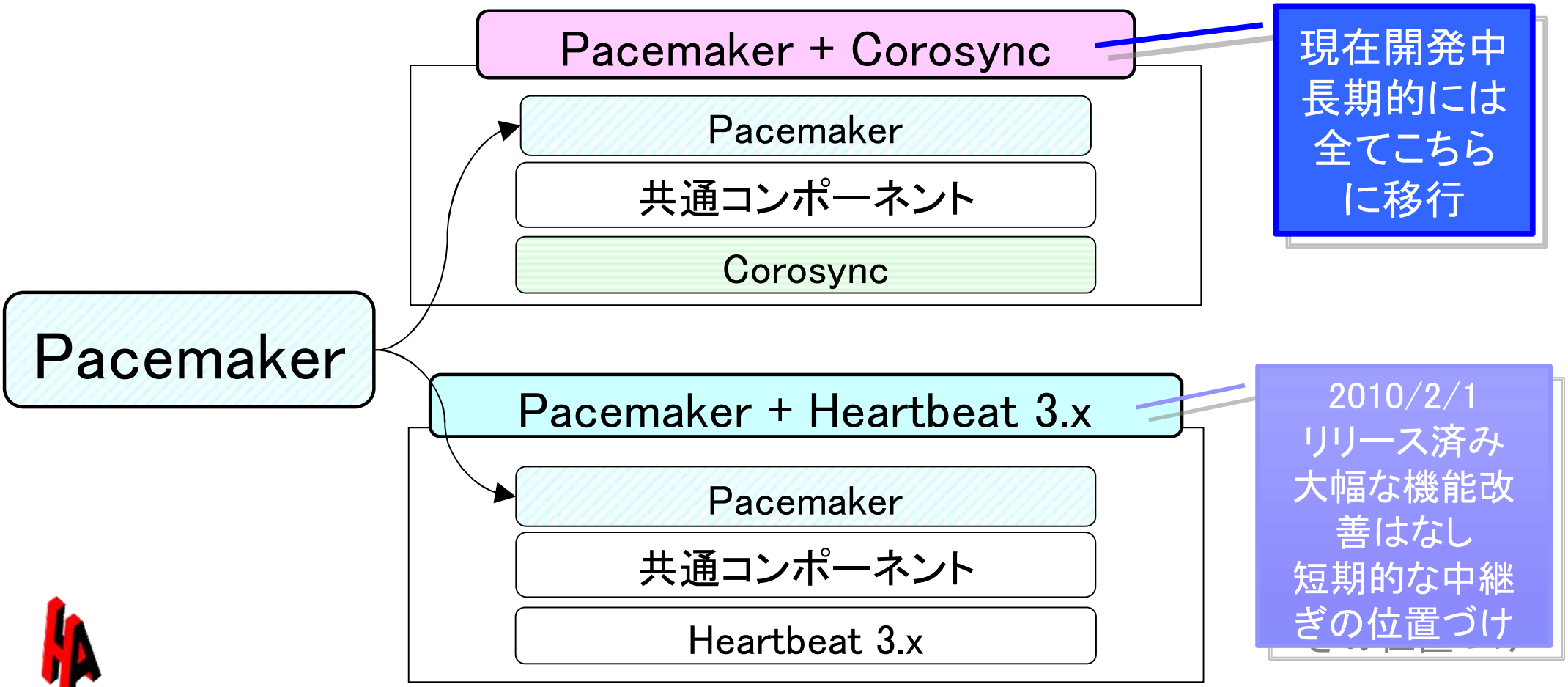
Corosync



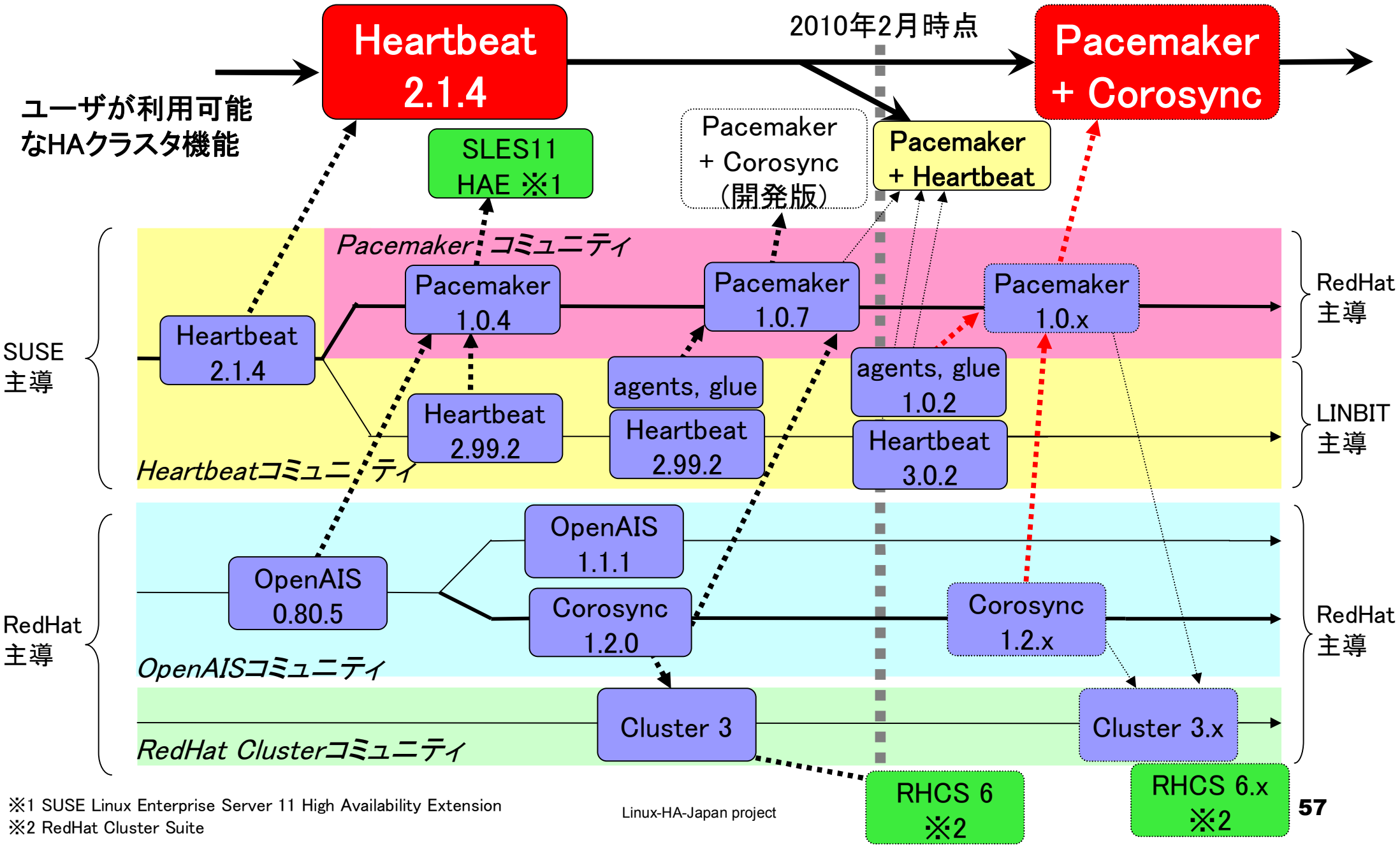
OpenAISコミュニティによって開発されたクラスタソフトである【OpenAIS】のクラスタ制御部はCorosyncとして分離されました。

Corosync

Pacemaker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。



HAクラスタ開発コミュニティの状況



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

Pacemakerロゴアンケート実施中

Linux-HA-Japanプロジェクトでは、
次期クラスタソフトウェアPacemakerの
ロゴを作成中です。

投票サイトに5つ案を載せましたので、
どれが良いか皆さんの投票をお待ちし
ています！

アンケート実施期間
2010年3月1日まで

ロゴの案、紹介ページ
http://sourceforge.jp/projects/linux-ha/wiki/PM_logo

アンケートページ
http://www.efeel.to/survey/linux-ha-japan-logo/

Linux-HA-Japanメーリングリスト

日本におけるHeartbeat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、Heartbeat 3.0、Corosync
その他DRBDなど、HAクラスタに関連する話題は全て歓迎します！

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

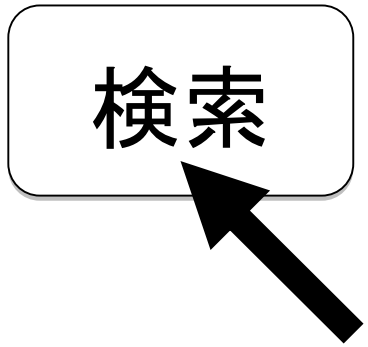
さいごに…

Linux-HA-Japanプロジェクトでは
様々なHeartbeat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

Heartbeatや
次期クラスタソフトPacemaker、
Corosyncの
普及展開を推し進めます

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

Linux-HA-Japan



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

⑤

参考情報

Heartbeatのインストール方法は？

- CentOS5系（RHEL5系）ならば、インストールは簡単！
  - Heartbeat-2.1.4-1 の rpmパッケージを、Linux-HA-Japanプロジェクトからダウンロード
    - heartbeat-2.1.4-1.rhel5.x86_64.RPMS.tar.gz
      - heartbeat-2.1.4-1.x86_64.rpm
      - stonith-2.1.4-1.x86_64.rpm
      - pils-2.1.4-1.x86_64.rpm

    上記Tarballには他rpmファイルもありますが、最低限これだけを入れればOK!
  - Heartbeat用ユーザ/グループの設定
    - グループ： haclient / gid:90
    - ユーザ名： hacluster / uid:90
  - ダウンロードしたrpmファイルをrpmコマンドでインストール

Linux-HA-Japanプロジェクト提供の Heartbeat　rpmダウンロードサイト

http://sourceforge.jp/projects/linux-ha/releases/?package_id=10606

Heartbeatの設定方法は？

- /etc/ha.d/ha.cf
  - Heartbeat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

- crm on — Heartbeat Ver.2モードで動作させる
- use_logd on — logデーモンを使ってログを出力
- udpport 694 — ハートビート通信に ポート694 を使用
- keepalive 2 — ハートビート間隔を2秒に設定
- warntime 20 — 警告発信用ハートビート不通時間を20秒に設定
- deadtime 24 — ノードが死んだと判断する時間を 24秒に設定
- initdead 48 — nodeで指定された全ノードの起動を待つ時間を 48秒に設定
- bcast eth1
- bcast eth2

} ハートビート通信に bcast を使用し、I/Fに eth1,eth2 を使用

- node srv01
- node srv02

} クラスタを構成するノード名が srv01、srv02

- /etc/logd.cf
  - Heartbeatのログ出力先を指定
  - クラスタ内の全ノードに、同じ内容のファイルを配置

- /etc/ha.d/authkeys
  - クラスタを構成する認証キーを保持するファイル
  - 認証キーが同じノード群でクラスタを構成
  - クラスタ内の全ノードに、同じ内容のファイルを配置

- /var/lib/heartbeat/crm/cib.xml
  - cib.xml = 主に、リソースの定義を設定するxmlファイル
    - どのようなリソースをどのように扱うか
    - 起動、監視、停止時に関連する時間
    - リソースの配置などを指定

cib.xmlの記法(DTD)を知る必要があり難しい…

```
<constraints>
  (..略..)
   <rsc_location id="rul_DK_dsc" rsc="grp_pgsql">
     <rule id="prefered_rul_DK_dsc" score="-INFINITY" boolean_op="and">
        <expression attribute="diskcheck_status" id="dkstatus1"
operation="defined"/>
        <expression attribute="diskcheck_status" id="dkstatus2" operation="eq"
value="ERROR"/>
     </rule>
    </rsc_location>
  (..略..)
</constraints>
```

cib.xmlはもう恐くない！

- 複雑なXML形式の設定も hb-cibgenで解決！

Heartbeat Ver.2ではDBMS等のリソース構成や動作は、cib.xmlで記述します。
このcib.xmlが複雑なので、なかなか手が出せないという声がありました。

hb-cibgenを使用すれば、Excelファイルからcib.xmlを簡単に生成する事が可能です。

cib.xml編集ツールを使おう！

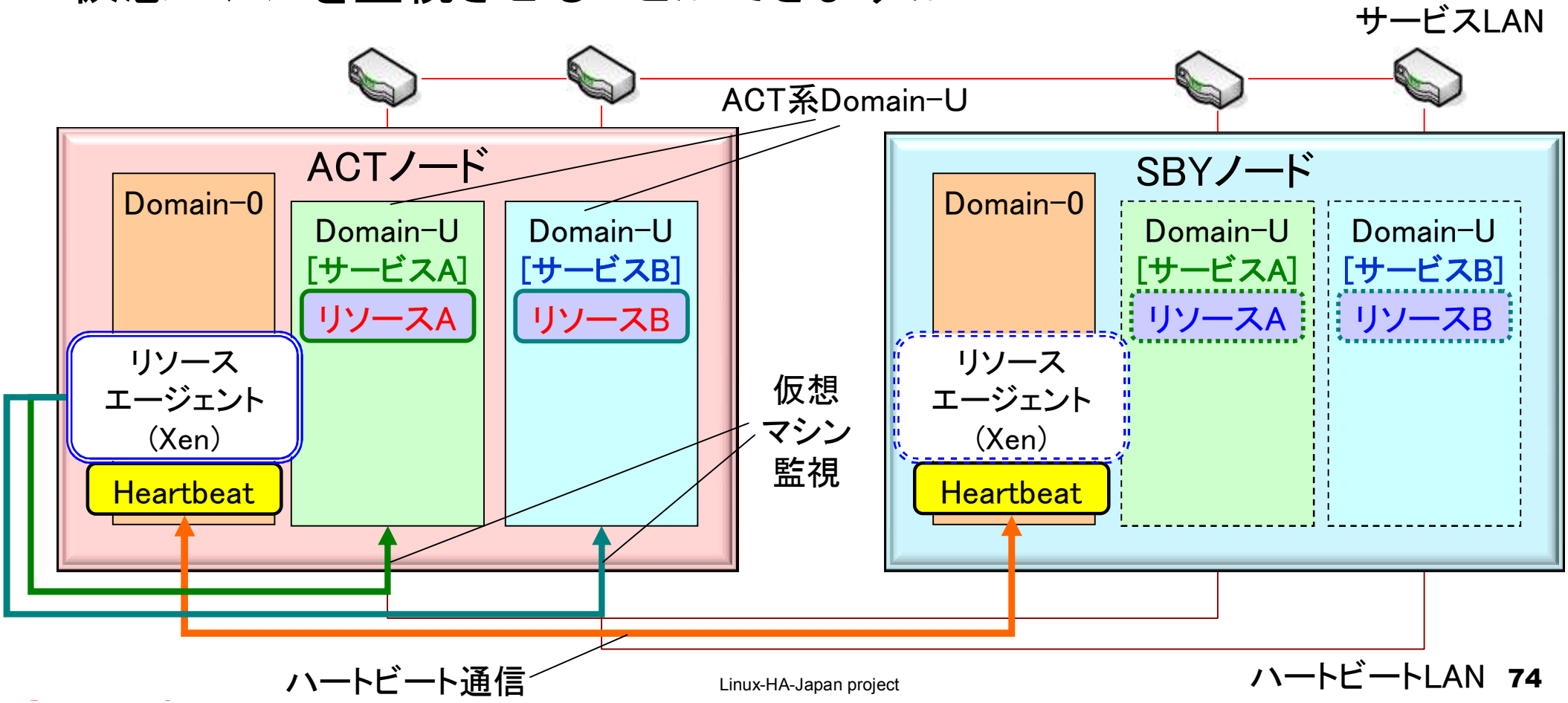
- hb-cibgen のパッケージ(zipファイル)を、Linux-HA-Japanプロジェクトのサイトからダウンロード
  - hb-cibgen-1.03-1_1-noarch.zip
    - hb-cibgen-1.03-1.noarch.rpm …… hb_cibgen本体
    - hb_cibgen_Env_1.03-1_1.xls …… テンプレートファイル(Excel形式)
- hb-cibgen本体のrpmファイルをrpmコマンドでインストール
- ①テンプレート作成 → ②csvファイル作成 → ③hb_cibgenコマンド実行 により、簡単にxmlファイルが生成可能！

詳しくは、cib.xml編集ツールの
公開wikiを参照しましょう！

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

「Xen」リソースエージェント
（/usr/lib/ocf/resource.d/heartbeat/Xen）

Domain-0からHeartbeat標準の「Xen」リソースエージェントを使用して仮想マシンを監視させることができますが…



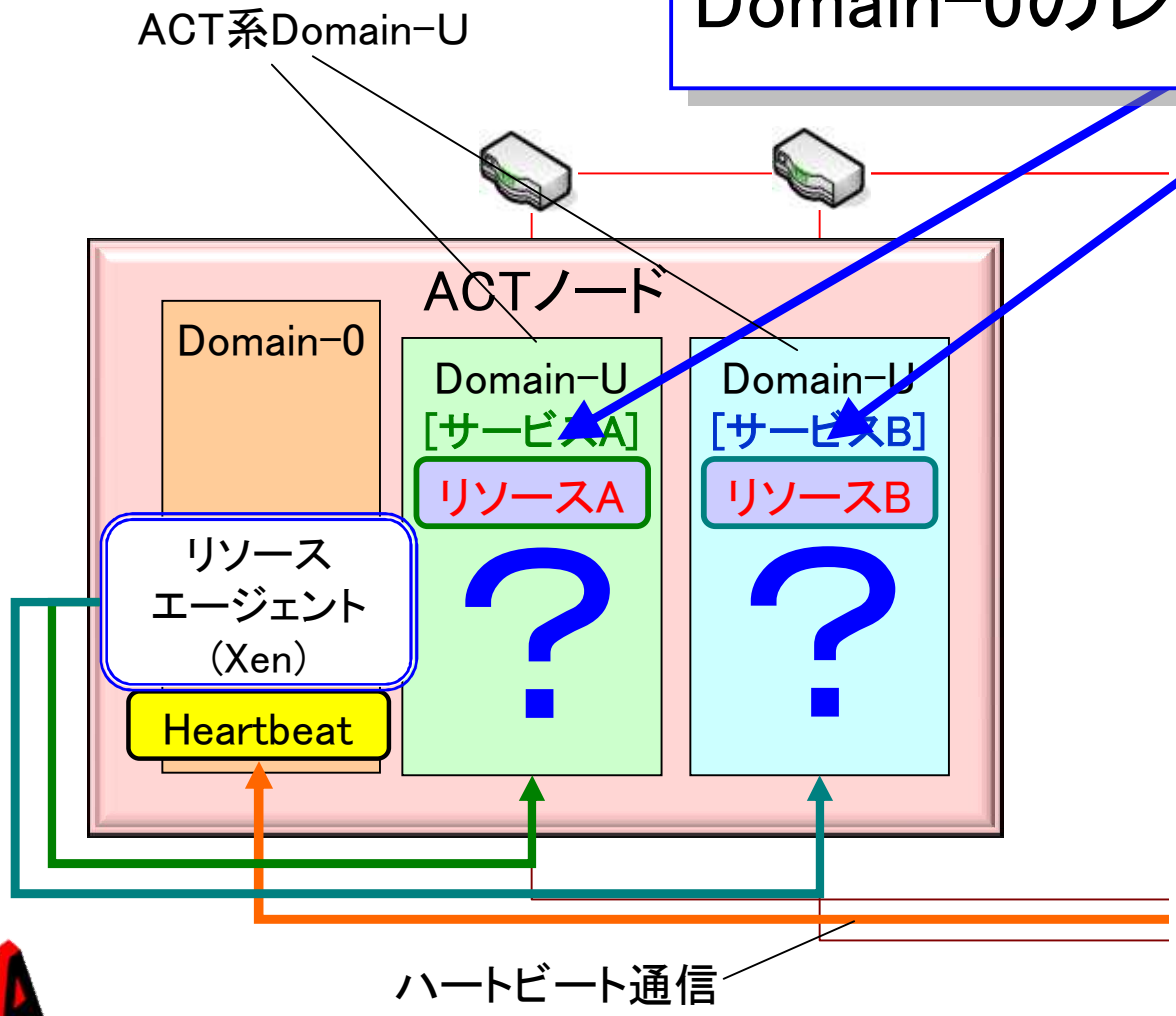
- このリソースエージェントによる監視では、Domain-0から xm list コマンドの結果を参照するのみなので、Domain-U内の詳細な稼動状態がわからない…

Xenリソースエージェント 監視(monitor)処理の抜粋

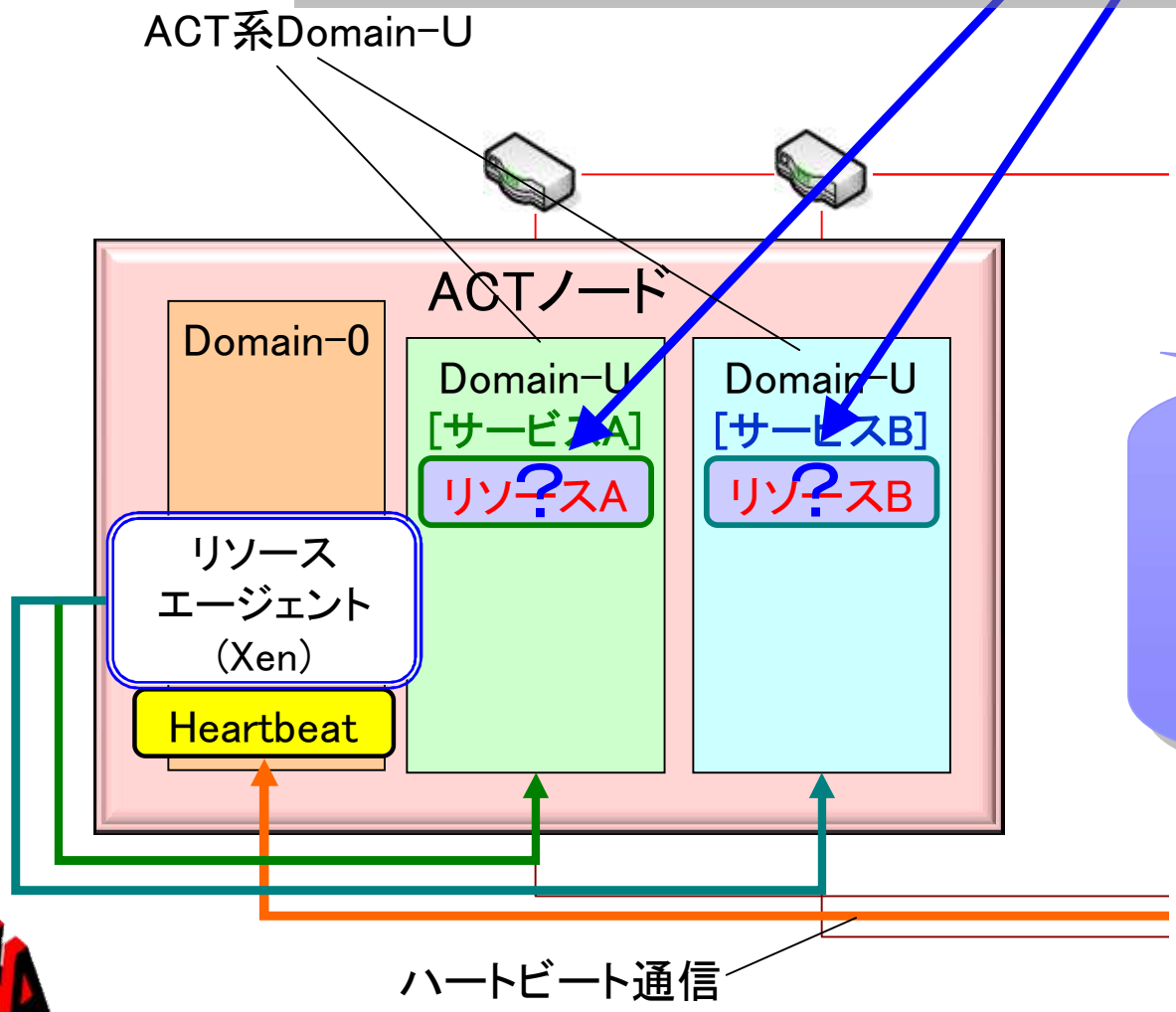
```
#!/bin/sh

 (省略)
    # we have Xen 3.0.3 or lower
    STATUS=`xm list --long $1 2>/dev/null | grep state 2>/dev/null`
    echo "${STATUS}" | grep -qs "[-r][-b][-p]---"
    if [ $? -ne 0 ]; then
      return $OCF_NOT_RUNNING
    else
      return $OCF_SUCCESS
    fi
 (省略)
```

xm listの結果からDomain-Uが、
r (running), b (blocked), p (paused)
かどうかの状態しか判断していない

仮想マシン内が異常であるかどうかは、
Domain-0のレイヤからは検出できない



監視対象は、ACT系Domain-Uの仮想マシンであり、
Domain-U内のリソース監視を行うことができない



仮想マシンを
いちリソースとみなす構成。
これでは高可用性のシステムは構築できませんね…

デバイス一括監視機能

Domain-U のディスクI/O処理は、Xen Hyperviosr と Domain-0経由のため、Domain-UのゲストOSからではディスク故障検出ができない場合がありました。
デバイス一括監視機能は、ホストOSのDomain-0 にディスクやping監視機能を搭載する事が可能なため、故障検出をより確実に行うことが可能になります。

Linux-HA-Japanプロジェクトより
３月公開予定！！

- デバイス一括監視機能による、内蔵ディスク監視の例