KVM を用いたプライベートクラウド環境の構築

○藤原冨未治 A)、大下弘 A)、佐々木康俊 A) 、原祐一 A) 、雨宮尚範 A) 、松岡孝 B)

A) 工学系技術支援室　情報通信技術系

B) 共通基盤技術支援室　情報通信技術系

概要

近年、サーバーの運用形態はスタンドアローン型の物理サーバーからクラウド技術を用いた仮想化による集約化へと徐々に変移している。KVM(Kernel-based VirtualMachine)はカーネルに仮想化管理機能を統合したものであり完全仮想化による仮想環境を提供している。今回工学研究科技術部研修においてクラウド環境のベースとなる仮想化サーバーを KVM で構築し、運用及び動作検証を行ったので報告する。

1　実験環境

1.1　ハードウェア構成

実施環境としてのサーバー構成を表 1 に示す。使用サーバーは昨年度の研修で用いた PC サーバー２台（ホスト名 SV1,SV2）と新たなテストサーバー２台（ホスト名 SV3,SV4）を使用し合計４台、ネットワークストレージとして表 2 に示す Thecus 製の N7700pro、N8900 の 2 台を用い、２系統での実施環境を整えた。またネットワーク環境として YAMAHA ルータ RTX-810 を使用しプライベートネットワークの中で試験環境を構築し、グローバル環境である NICE 側には直接接続しないようにした。

表 1. 各仮想サーバーの構成.

	SV1、 SV2	SV3、 SV4
CPU	AMD PhenomII X4905eBOX	AMD Opteron 8-Core 6128BOX 2 基
Memory	DDR3 PC3-10600 8GBx2	DDR3 PC3-10600 4GBx8
HDD	Seagate (7200rpm 1TB)	Seagate (7200rpm 500GB)
Power Unit	SPKR5-550P(550W 80+B P)	770W 80PLUS Gold Single Power Supply
Motherboard	Biostar TA890GXB HD	AMD SR5690+SP5100

表 2.NAS の構成

	NAS1 (Thecus 製 N7700pro)	NAS2 (Thecus 製 N8900)
HDD	2T x 7	2T x 8
Raid	Raid 6	Raid 6

1.2　OS のインストール

ホスト OS には Red Hat Enterprise Linux をベースとして高い互換性を持ち、無償で配布されているものの中で研修時に最新のバージョンがリリースされていた CentOS6.3 をインストールした。インストール時に CentOS に同梱されている仮想化ホストとして KVM を合わせてインストールした。

今回はプライベートネットワーク内でサーバーを構築するため、SELinux やファイアウォール、ネットワーク管理ツールを使用しない設定にした。

1.3　ネットワーク構成

構成したネットワークを図 1 に示す。ルータ直下の仮想サーバーのネットワーク SV1～SV4 にはそれぞれ

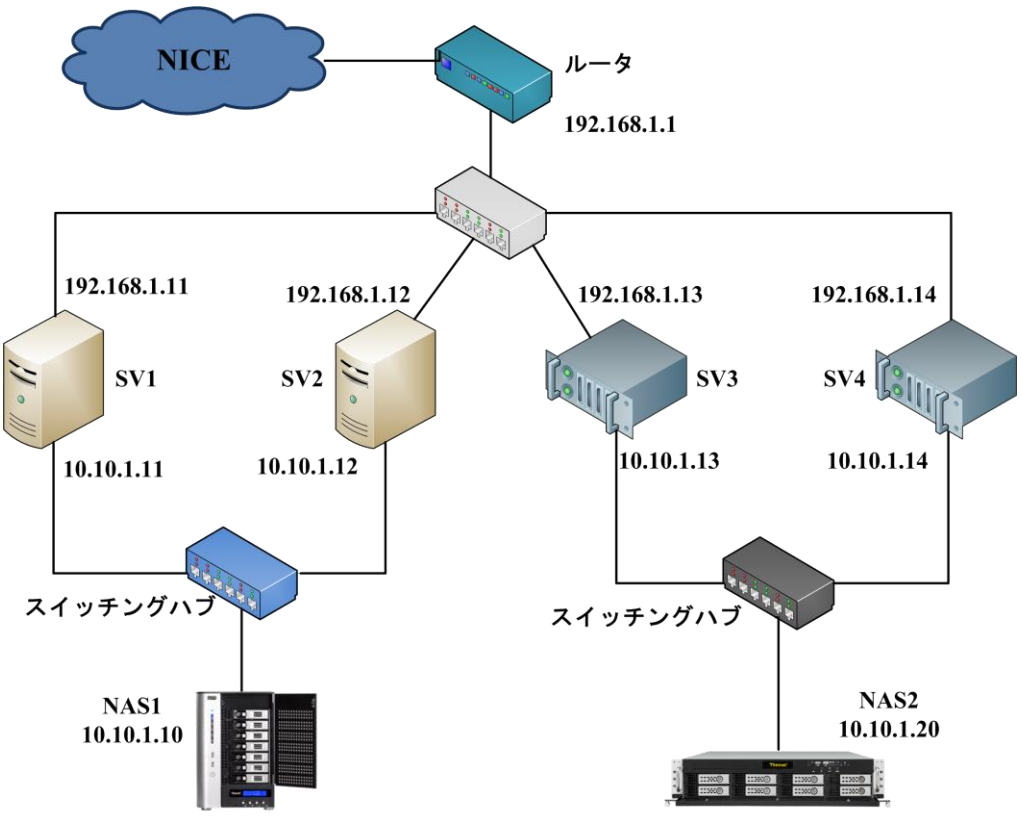

図 1.　実験環境構成図

IP アドレス 192.168.1.11～192.168.1.14 を割振り、各ストレージサーバーの IP アドレスは NAS1 に 10.10.1.10、NAS2 に 10.10.1.20 を設定した。ストレージサーバーとの接続には同様に 10.10.1.11～10.10.1.14 を設定した。仮想サーバーの OS とストレージサーバーのファームウエアは最新のバージョンにそれぞれアップデートし、ネットワークインターフェースカードを冗長化するためのボンディングの設定と仮想 LAN(VLAN)の設定を行い、KVM と仮想サーバーとの通信用に仮想的なブリッジ設定を施した。

2　ライブマイグレーション

ライブマイグレーションとは稼働中の仮想マシンを停止することなく別のハイパーバイザサーバーの仮想マシンに移動させる技術のことをいい、メンテナンス時に無停止でサービスの継続が可能となる。ライブマイグレーションを行うと移動元の仮想マシンのメモリイメージを丸ごと移動先の仮想マシンに移し替えるため稼働中の OS やアプリケーション、ネットワーク接続等を一切停止すること無く移動先の仮想マシン上で動作を継続することが出来る。

2.1　実行方法及び動作確認

ライブマイグレーションを実行するために GUI ツールの仮想マシンマネージャーを使用した（図 2)。動作中の移動元（SV3）仮想マシンでマイグレーションを選択起動し、図 3 の新しいホストに移動先（SV4）の仮想マシンを指定し、実行することでマイグレーションが開始される。

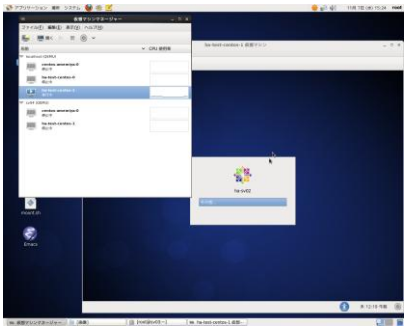

図 2.　仮想マシンマネージャー

図 3.　移動先の指定

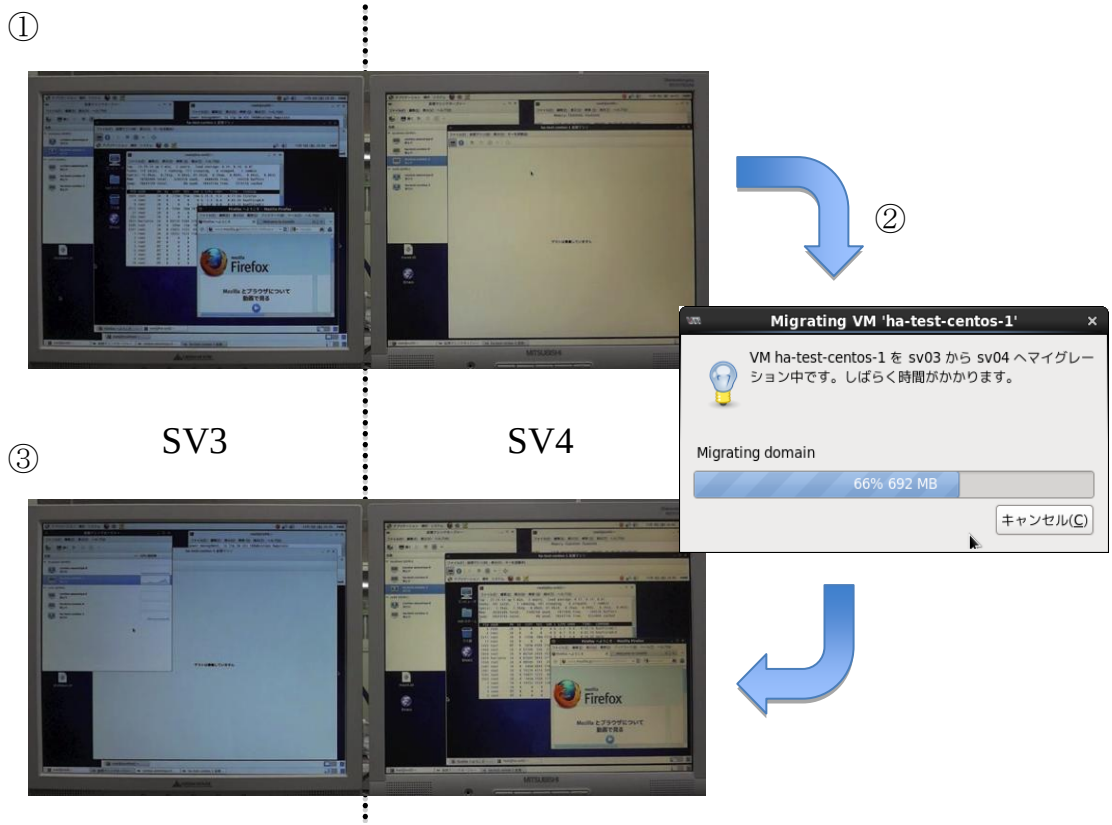

図 4.　仮想マシンのマイグレーション

図 4①がマイグレーション実行前の SV3 と SV4 の状態である。最初 SV3 の仮想マシンが稼働中であり SV4 は待機の状態である。マイグレーション操作を実行すると SV3 のメモリのコピーを開始（図 4②）しコピー終了後、図 4③のように移動元 SV3 から移動先 SV4 へ仮想マシンが切り替わり OS、実行中であったアプリケーション(Web ブラウザ)、Top コマンドが同じ状態で遷移したことがわかた。これによりハイパーバイザサーバー間での仮想マシンのライブマイグレーションが正しく行われることを確認した。

3　プロビジョニングサーバーの構築

プロビジョニングとは、仮想マシンを構築する際にネットワークやシステムなど必要なリソースなど事前に準備しておき、必要に応じてそれらを割り当て迅速に提供するこという。

プロビジョニングサーバーは「クローニング方式」と「自動インストール方式」の２つに分類されるが、今回は OS のインストール時、ソフトウェアの選択や仮想ディスクの構成等を用途に応じて変更できる自動インストール方式を用いた。

自動インストール方式のプロビジョニングサーバーは、PXE サーバーと KickStart サーバーから構成される。

表 3.　プロビジョニングサーバーの構成

PXE サーバー	DHCP によって、下記２点が行われる。 ・VM に対して PXE ブートするＩＰアドレスの付与 ・PXE ブートに必要なアクセス情報を提供 TFTP によって、ブートイメージなど必要な設定ファイルの提供し、インストーラの起動準備まで行う
KickStart サーバー	インストーラが質問する回答を記述した設定ファイルを参照することで、ＯＳの自動インストールを行う。 ※設定ファイルは、HTTP で提供することで、VM からインストールパッケージを参照できるようにする。

3.1　構築方法

次の順序でプロビジョニングサーバーの構築を行った。

(1)　ハイパーバイザサーバー上でプロビジョニングサーバー用ＶＭの構築

(2)　PXE サーバーの構築

(3)　KickStart サーバーの構築

(4)　リポジトリ（インストール OS）の準備

(5)　ＶＭの自動インストールの確認

KickStart による VM の自動インストールには、下記２つのＯＳを用いた。

- Red Hat Enterprise Linux 30 日間の試用版
- CentOS 6.3 (32bit)

3.2　実行処理の流れ

プロビジョニングサーバーの処理の流れは次の通りである。

(1)　VM 作成用起動シェルスクリプトの準備と実行（ディスク、メモリ、CPU 等の割当）

(2)　DHCP による IP アドレス取得

(3)　ブートイメージのファイル名の取得

(4)　ブートイメージ取得

(5)　KickStart の実行によるパッケージのインストール

(6)　インストール完了後の設定

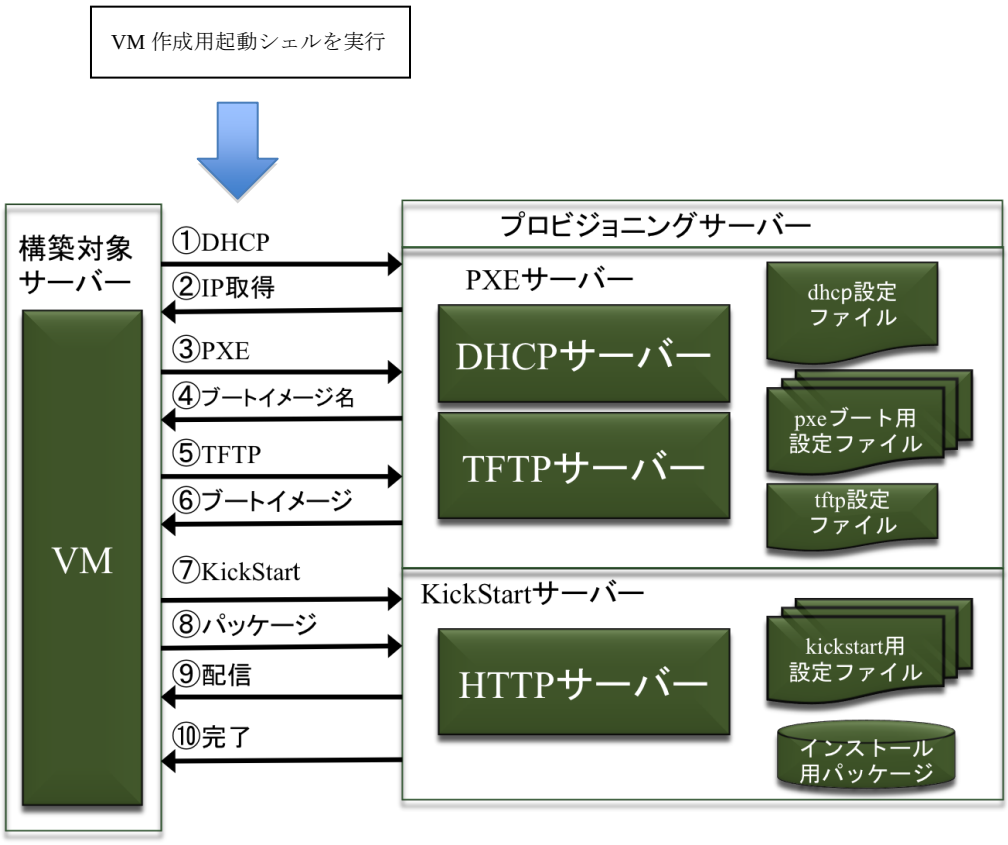

図 5.　プロビジョニングサーバーの流れ

3.3　PXE ブート

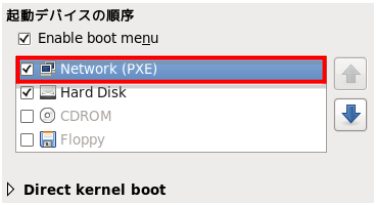

図 6.　PXE アクティブ設定方法

しかし図 5 の流れ③において、PXE ブートが自動で起動出来なかった。PXE ブートを起動するために、

(1)　起動ウインドを停止

(2)　起動デバイスの設定で「Network(PXE)」をアクティブにする（図 6）

(3)　起動ウインドを開始する

という手順が必要となり、完全な自動化まで、実現できなかった。

(4)　PXE ブートし、インストールイメージの選択後はＶMのインストールは自動化され、OS（RedHat Linux 及び CentOS）のインストールに成功した。

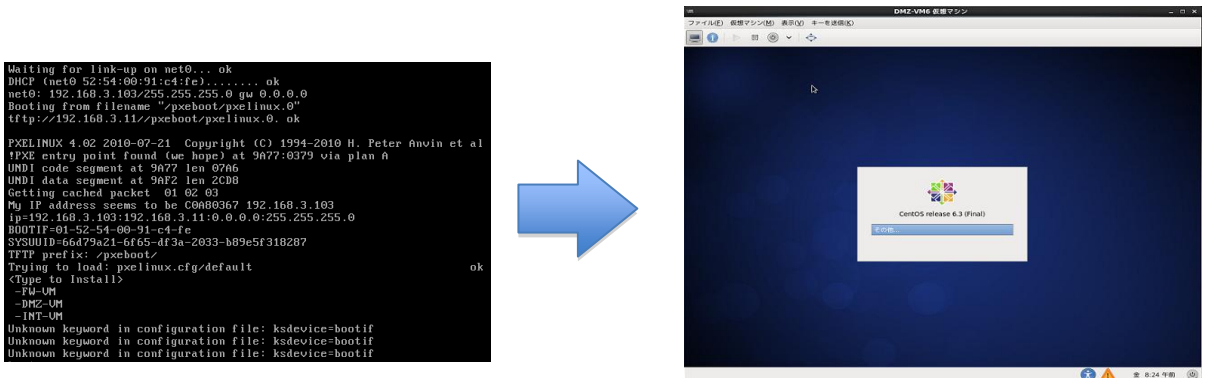

図 7.　PXE ブートから OS インストール完了まで

4　ハイパーバイザサーバーの冗長化

仮想化によりサーバーを集約した場合、故障発生によりハイパーバイザサーバーが利用できなくなると集約したサーバー全体に影響が及んでしまう。システムの信頼性を高める手段としてはシステムの冗長化がある。ハイパーバイザサーバーの冗長化はサーバーの高可用性クラスタ構成により実現されている。本研修では冗長化構成を行い、動作を検証した。

4.1　フェイルオーバー

冗長化により実現できる働きは大きく分けて 2 種類ある。1 つ目はハイパーバイザサーバーが故障して仮想マシンが動作しなくなってしまった場合に、別のハイパーバイザサーバーで仮想マシンを再起動するというものである。2 つ目は運用中のハイパーバイザサーバーに関するネットワーク経路などが故障して仮想マシンを利用できなくなってしまった場合に、別の健全なハイパーバイザサーバーに仮想マシンをマイグレーションするというものである。こうしたフェイルオーバー機能により、故障からの復旧を速やかに行うことができる。

4.2　クラスタ構成の設定

本研修のシステムは、図 1 のようなハードウェア構成である。システムには Pacemaker と Corosync というソフトウェアを利用した。Pacemaker はリソース（ネットワーク経路、仮想マシンなど）の監視と故障発生時の制御を行う。また、Corosync はノード（ハイパーバイザ）間の通信を行う。これらを組み合わせることでクラスタを構築することができる（図 8）。

○システムの設定

- 簡単のため iptables、NetworkManager を停止させ、SELinux を無効にした。
- 共有ストレージを/guest にマウントし、仮想マシンのストレージプールに指定した。
- 相互に名前解決できるようにした。
- root にパスワードなしで ssh 接続できるようにした。

○Corosync の設定

corosync.conf.example の bindnetaddr を 10.10.1.0 に変更して設定に使用した。

○Pacemaker の設定

ネットワーク経路の死活監視をルータ（外側）と NAS（内側）への ping アクセスで行うようにした。

仮想マシン vm0 を作成してリソースとして登録した。

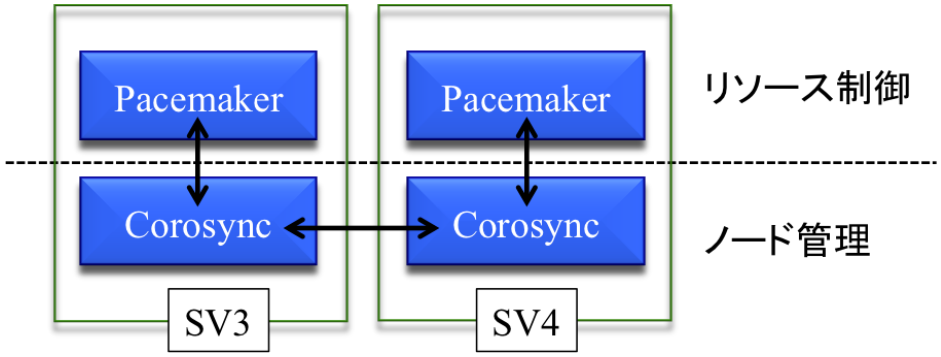

図 8.　Pacemaker と Corosync

4.3　動作検証 1（ハイパーバイザサーバーの故障）

設定を行った後、実際に故障を再現してクラスタの動作を確認した。1 つ目の検証（図 9）では SV3 で vm0 を動作させた状態で SV3 を強制的に停止させた。すると、SV3 の停止が認識された後、SV4 で vm0 が自動的に起動した。SV3 を再起動した後に vm0 をマイグレーションさせて復旧を完了することができた。これにより、ハイパーバイザが故障しても自動的にサービスを再開できることが確認できた。

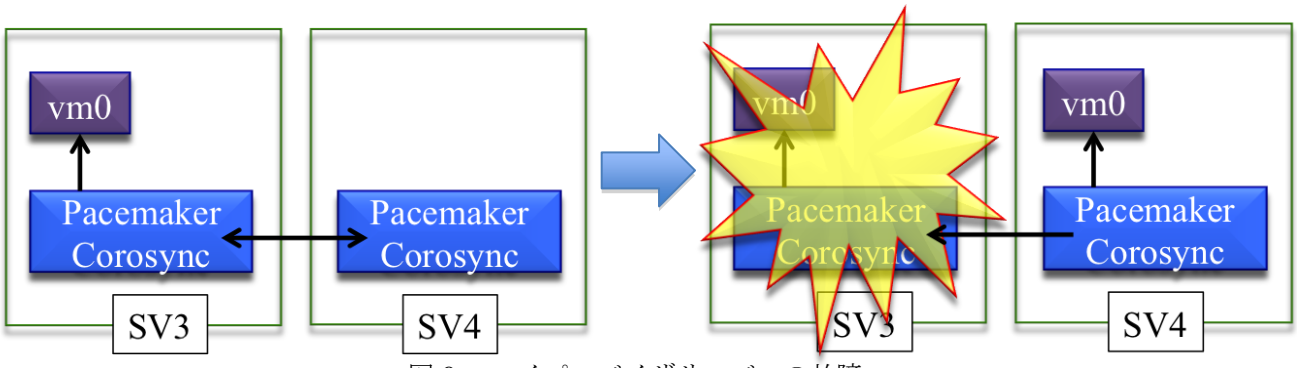

図 9.　ハイパーバイザサーバーの故障

4.4　動作検証 2（リソースの故障）

2 つ目の検証(図 10)では SV3 で vm0 を動作させた状態で、SV3 の LAN ケーブルを引き抜いて外側ネットワーク経路を断線させた。故障の認識後、vm0 が SV4 に自動的にマイグレーションされた。LAN ケーブルを挿し直し、vm0 を SV3 に手動でマイグレーションすることで復旧を行うことができた。これにより、リソースの故障時にも、サービスを継続できることが確認できた。

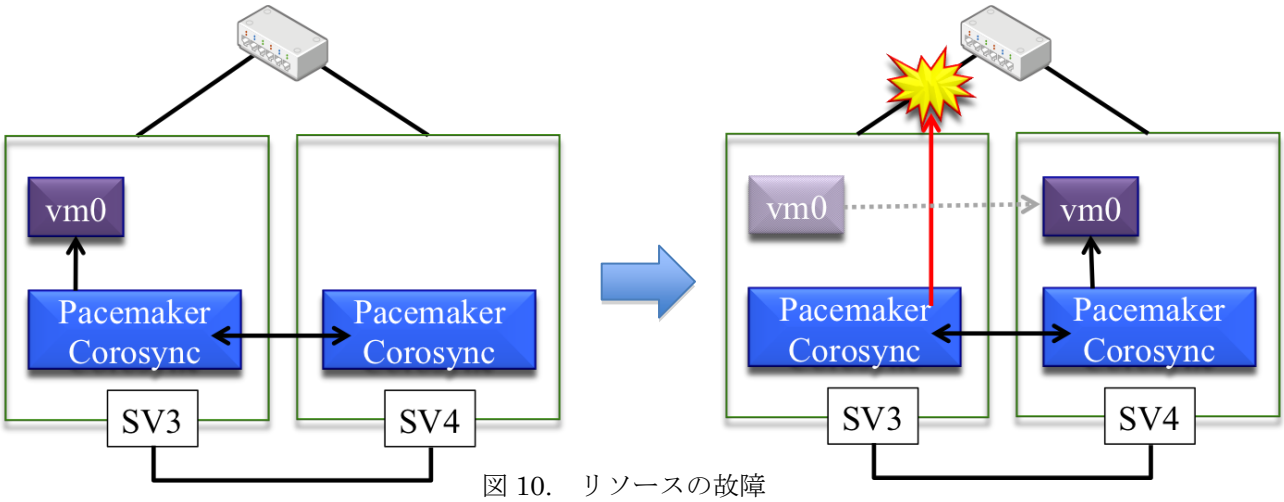

図 10.　リソースの故障

5　まとめ

今回の構築実験では、ScientificLinux6.3 と CentOS6.3 という無償版のソフトウェアを用いて２系統のテストクラウド環境を構築し、ライブマイグレーションの実行環境を整え、その動作を確認した。次に仮想マシン作成簡素化のためのプロビジョニングサーバーを構築し、２種類のゲスト OS のインストールを行えることを確認した。最後に、安定運用のための技術である HA クラスタを実現するため、ハイパーバイザサーバーの冗長化を Pacemaker と Corosync を組み合わせて構築した。そして、片方のサーバーが故障した場合にもう一方のサーバーで、ゲスト OS を自動起動しサービスを再開できることを確認した。

この研修を行うことにより、研修参加者の仮想化技術及びクラウド技術に関する知識が深まった。特に、iSCSI 装置を用いて排他制御のためにクラスタファイルシステムを構成することが困難であること、また、フリーソフトで構成する場合に管理を自動化することが非常に困難であることが認識できた

参考文献

[1]　知識ゼロから始める Linux サーバーの作り方，日経 Linux, pp162-209(2012.1.10)

[2]　プライベートクラウド用サーバの構築，名古屋大学工学研究科・工学部技術部技報 vol.14, ppl-8

[3]　Linux KVM による仮想化環境の構築，名古屋大学全学技術センター技術報告第 7 回, PJOU-2