ＩＴ特別講座
Linux 高可用性システム入門

～無停止システムを目指して～

Ver. 1.2

LA-Linux 専任講師 矢越昭仁

2011/07/09

インターネットの標準サーバ OS として発展してきた Linux には、多くの『システム停止を 回避する仕組み』(高可用性 HA: High Availability) が搭載されています。このコースでは HA の全般的な話から、Cent OS に搭載されている HA 機能を体験します。

目次

1.高可用性システムとは

1-1. はじめに

インターネットやモバイル、最近のクラウドといったICT基盤の普及とともに、コンピュータ・システムの停止は、即社会的な事件に発展してしまいます。

表1： 2011年システム障害ニュース

発生日（期間）	発生企業	概要
2011年5月6日(1週間)	ソニー	個人情報漏洩
2011年4月25日(1時間)	新生銀行	ATM・ネット取引停止
2011年4月9日(7時間)	ゆうちょ銀	一部ATM停止
2011年3月15日(1週間)	みずほ銀行	ATM・振込取引停止
2011年1月17日(1時間)	JR東日本	新幹線運航見合わせ

一言にシステム障害といっても、その原因は多岐にわたり、操作ミス、(設計者にとっての)想定外の使用、想定外の高負荷、ハードウエア故障、ソフトウエアバグ、悪意ある行為（ウイルス、クラッキング）などがあります。事件に発展するようなトラブルは複合的な原因によるものが多いようです。

システムがサービスを提供すべき総時間と、なんらかのトラブルで停止していた時間の比率を「可用率(稼働率)：Availability」といい、安定稼働の目安とされています。

可用率＝（総時間－停止時間）／総時間　(%)

つまり全く止まる事無くサービスを提供し続けることができれば 100%となります。24時間365日サービスを提供しているネット配信などのサービスでは、総時間が24h×365Day = 3,760h と非常に母数が大きくなり、いかに100%を目指す事が困難かが分かると思います。

表2： 可用率と停止時間

可用率(%)	年間停止時間	主なシステム・業界(目標値)
99	3.65日	初期のWebメール
99.8	17.5時間	基幹システム（平日換算5h[1]）
99.9	8.76時間	Google (有料サービス)
99.99	52.56分	銀行ATM、鉄道
99.999	5.3分	東証Arrowhead
99.9999	31.5秒	（局用）電話交換機

可用率はともかく、故障やトラブルによるシステム停止をできるだけ回避すべく、予備機を用意する、エラーを検知するなどいろいろな仕組みが開発されてきました。特に可用性を強化したシステムを高可用性システム（HA: High Availability System）と呼びます。

このコースでは高可用性システムとはなにか？またそれを実現する方法について、解説していきます。

[1] 稼働時間を24時間、365日ではなく10時間、週5日とした時の年間停止時間。

1-2. 冗長化

まず可用性を向上させる一番簡単な方法としては、予備を用意しておきトラブルが発生したら、本番機と置き換えるというアプローチです。システム構成に余裕を持たせることを「冗長化」と呼び、ハードディスクや CPU といったコンピュータ資源に余裕を持たせることや、通信時のエラーに備えデータにチェック用の項目を持たせることも広く「冗長化」「冗長性」と呼んでいます。

予備を用意する方法での冗長化は、システム構成要素ごとにいろいろな手法があります。

ホットスワップ

サーバ用途製品には「ホットスワップ（活線挿抜）」とよばれる機能があります。これはシステム稼働中に、停止させずに部品交換ができるというものでコンピュータメーカ各社から提供されています。

本体の電源が入った状態で抜き差ししてもノイズが発生しないよう設計された電子回路と、抜き差ししやすいようプラグ式になったインタフェースなどが必要です。サーバ用途コンピュータでは、この部品も 2 個ずつ用意しておき、故障の際は当該部品を差し替えることでシステム停止を防いでいます。

図 1 ホットプラグ電源（DELL）

ホットスワップに対応した部品は、電源、ファン、ハードディスク・DVD、NIC などがあります。

電源

電源装置（トランス）は先のホットプラグにより冗長化できますが、停電に備える冗長化もあります。大規模なデーターセンターでは自家発電装置（3 日間～2 週間の停電に備えられる）や、複数の電力会社からの電力供給（JR 東日本と東京電力など）を行い、センタ設備ごと停電対策を行っています。システム個別の対策としては、無停電電源装置(UPS: Uninterruptible Power Supply)を用いるのが一般的です。バッテリーを使ったものは量販店でも販売されており、個人でも購入できます。

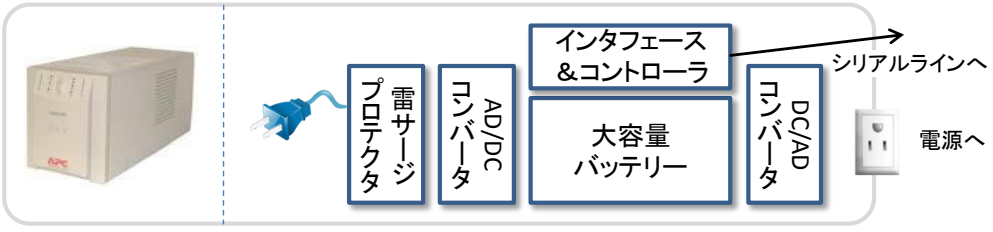

図 2 UPS 概念図

このタイプの UPS は平常時にはバッテリーを充電し、商用電源が停止した時にはバッテリーからシステムへ電源を供給しつづけます。商業電源停止（バッテリー切り替え）時、

電源復帰時、バッテリー容量低下時など状況が変化するたびに、インタフェースを通じコンピュータへ信号を発信する仕組みになっています。コンピュータ側はこれらの信号を受信すると必要に応じ、警告メッセージの表示や、計画停止（シャットダウン）を行うことができます。[2]
電源のバックアップ以外にも、電圧変動、周波数変動、周辺機器のノイズ、雷サージといった電源自体の揺らぎへの対応も組み込まれています。

ハードディスク

ハードディスクは物理的な可動部が多く、最も故障しやすいコンピュータ構成部品のひとつです。それだけに冗長性には多くの種類があり、RAID(Redundant Arrays of Inexpensive/Independent Disks)は、その代表例です。RAID は専用のハードウエアによる方法と、LVM などのソフトウエアを用いる 2 通りの方法があります。
冗長化という観点では、信頼性の高い高性能・大容量のディスクを複数のシステムで共有する事もあります。当然このような共有ディスクは RAID で構成されており、複数の OS と接続できるようになっています。明確な定義はありませんが、小規模構成には NAS(Network Attached Storage、ファイル単位で共有)、大規模構成には SAN(Storage Area Network、データブロック単位で共有)が用いられています。
コンピュータメーカ以外に、このような記憶装置の専業メーカとしては EMC、NetApp、3PAR(現 HP)などがあります。

1-3. 高可用性システムの考え方

先の冗長化は、システムの構成部品単位で冗長化するというものでした。それの対象範囲を更に拡大して、システム全体で予備を用意する方法をデュプレックス・システム(duplex system)と呼びます。システム規模で冗長化したシステムは「高可用性システム」と呼ばれ、保護の対象となる「必要最小限のシステム」をシンプレックス・システム（simplex system）と呼びます。このデュプレックス・システムが高可用性システムの考え方の基本で、どれだけの予備を用意するか、またトラブル発生時にどのように復旧させるのかによっていくつかの手法があります。

図 3　デュプレックス・システム概念図

実務で使っているシステム（シンプレックス）を本番機(Production System)と呼び、予備側をスタンバイ(Standby)と呼びます。
予備機はその待機方法によりさらに 3 つに分類されます。

1. ホット・スタンバイ
   本番機と全く同じ構成のシステムを稼働させた状態で待機させる方法。トラブル発生時

[2] Linux では/etc/initab の pf(power fail)エントリーで対応します。

はネットワークを切り替える程度で素早く切り替えることができるが、運用コストが高くなる。数時間以内の切り替え時間を想定している。

2. コールド・スタンバイ
   完全に停止させた状態で待機させる方法。多くはフルバックアップを回復するところから始めるため、復旧には数日かかる場合があるが、運用コストは低く抑えられる（場合によっては倉庫に梱包して保存）
3. ウォーム・スタンバイ
   本番機と同じ構成のハードウエアに OS を載せた状態で待機。トラブル発生時にはデータやアプリケーションをバックアップから戻し、本番機と同じ構成にしてから切り替える。切り替え時間は２４時間～数日程度を想定している。
   複数のソフトウエア構成の違う本番機に対応できるが、設定までに時間がかかる。

最近はコンピュータのコストも低くなってきているので、ホット・スタンバイを応用し本番機を二重構成とし正常時は分散処理を行い、トラブル時には処理能力が半減するものの停止をしない縮退運用(Degration)を行うものや、ホット・スタンバイで不要不急処理を行いトラブル時には当該サービスを停止させる、といった運用も見られます。
ホット・スタンバイをさらに発展させ、両方を本番機として利用する方式をデュアル・システム(dual system)と呼びます。これは予備機ではなく、全く同じ構成のシステムを複数用意しておき、普段からクロスチェックを行うシステムです。性能的には複数台あってもシンプレックスと同で、障害発生時には故障したシステムを切り離します。
極端な例としては、スペースシャトルに搭載されていた DPS や、ボーイング 777 の Fly By Wire があります。

- スペースシャトルの多数決システム
  ４重の冗長性をもち、多数決により処理を決定。さらに予備機があり全部で５系統の冗長化を実現。放射線・電磁波対策のため特殊なコンピュータを使用(IBM AP-101[3])
- ボーイング 777 の Fly By Wire システム
  異なるバージョン・部品構成のシンプレックスを３種類[4]用意し、それを３系統で冗長化（3x3)。ハードウエア故障だけでなく、バグに対する耐性も有する。

1-4. フォールト・トレラント・システム

完全なデュアル・システムを発展させ、１筐体に２つのシステムを収め、外見上は１つのシステムに見えるようなシステムがあります。システムの構成部品すべて（CPU、メモリ、HDD、電源など）を二重化し、密なクロスチェック、障害検知、障害部分の切り離しなどをハードウエアよって行うシステムをフォールト・トレラント・システム（Fault tolerant system：故障に耐性がある）と呼びます。
旧タンデム社(現 HP)の NonStop シリーズ、ストラタス・テクノロジー社の ftServer が専業メーカとして有名です。また大手コンピュータメーカも FT を冠する製品をいくつか販

[3] System360(汎用機)の流れを持つ組込機。F-15,B-52 などにも採用、磁気コアメモリ。
[4] 32bitCPU, Intel 80486, Motorola 68040, AMD 29050 の 3 種類

売しています。

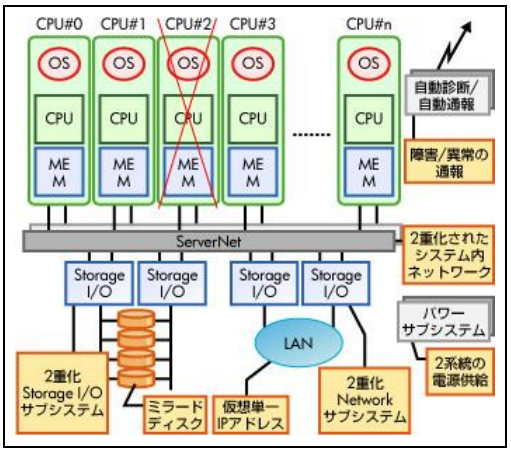

図 4 HP Integrity NonStop サーバーアーキテクチャ

フォールト・トレラント・システムでは構成要素の全てが二重化されており、故障するとシステム全体が停止する箇所（単一障害点：Single Point of Failure/Repair）がないよう設計されています。

フォールト・トレラントは、「故障してもそのまま処理を続ける」ことができる事をいいますが、似た用語にフェール・セーフ(fail-safe)があります。フェール・セーフは「故障したら安全に停止（縮退）する」という事で、普段の生活ではコンピュータよりも自動車や鉄道といった装置に組み込まれている工夫です。

1-5. コンピュータ・クラスタ

FT システムは専用のシステムを用意する必要がありますが、既存のシステムをネットワークで接続し冗長性を高めることも可能です。1980 年代に DEC 社が発表した VAX Cluster 以降、各コンピュータメーカも同様の製品を提供しています。

クラスタとは「房」のことで、コンピュータ１台１台をブドウの粒に例えると、それらが連携して一つの「ブドウの房」のように振る舞うところから命名されています。

クラスタには個々のコンピュータをどう接続するかで大きく２つに分類されます。

１． 疎結合クラスタ

ネットワークで接続されたコンピュータが、それぞれの資源をネットワーク経由で共有し共同作業を行う。各コンピュータもそれぞれ独立して動作することが可能。現在の Web サーバ、ロードバランサー構成が代表例。

２． 密結合クラスタ

周辺機器インターフェース(iSCSI など)や、専用バスを使ってコンピュータ相互を接続し共同作業を行う。共有メモリ、共有ディスクなどの共有リソースを使う。

また、最近ではクラスタよりも大きな概念で有機的に接続された多数のコンピュータによって仕事をこなす「グリッド・コンピューティング」もクラウドコンピュータの要素として普及しています。最近話題の「スマート・グリッド」は町中にある PC や PC が搭載された自動車、店舗などを一つの大きなコンピュータの集合としてとらえ、効率的なエネル

ギー利用を目指しています。

疎結合クラスタ

疎結合クラスタはもともと稼働しているシステムを拡張し、比較的手軽に導入することができます。ネットワークで接続するため、密結合ほどのパフォーマンスは発揮できませんが離れた場所のサーバを繋ぐことも可能です。DEC VAX Cluster では、ノード間が 800km[5] 離れた場所でのクラスタリング[6]に成功した事例もあります。

クラスタを構成する個々のコンピュータはノードと呼ばれ、疎結合の場合は各ノードも普通に利用する事が出来ます。たとえば cluster-1 というサーバに処理を要求すると、内部の node-1～3 の何れかに割り振られて処理が行われます。全てのノードが同じ性能であれば、理論上 cluster-1 は構成する各ノードの約 3 倍の処理を行うことができます。同時に個別にノードを指定し処理を行うことも可能です。

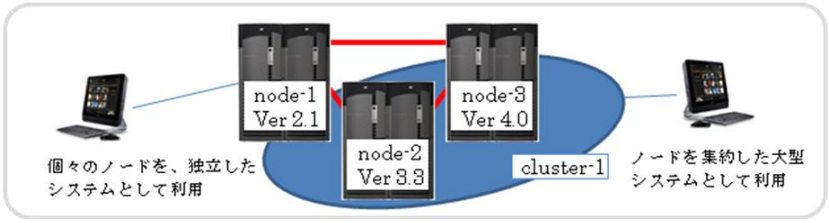

図 5　疎結合クラスタ概念図

クラスタに参加するメンバーはハードウエア構成やソフトウエアのバージョンが異なっていても問題なく接続できます。この特徴を生かして、ゆるやかな OS のバージョンアップに使うこともできます。node-1 が元々のシステムで、node-3 の環境へ移行する場合、node-1 を稼働させたまま、修正・テストが済んだプログラムを随時 node-3 へ移すことができます。cluster-1 に処理要求をすると、node-1 にしかない（移行が済んでいない）処理は node-1 で動作し、移行が済んだ処理は node-3 でも動作する。といった段階的な移行が可能です。疎結合クラスタは OSS でも数多く製品が出ており、この講座でも紹介します。

密結合クラスタ

密結合クラスタはノード間でネットワーク以外の資源も共有する形式です。共有する資源としてはハードディスク装置が一般的で、専用の高速インタフェースで共有するタイプが一般的です。さらに高速な処理を行うためメモリを共有するタイプのものもあります。高速なインターフェース（iSCSI や PCI バス）を用いるため、接続できるノード間の距離に物理的な制約が発生します。また各ノードは原則として同じ構成である必要があります。

図 6 密結合クラスタ概念図

[5] 800km はおよそ、東京～広島、東京～青森ほどの距離となります。
[6] この様な構成はフォールト・トレンラント（Fault tolerant：故障耐性）を超えディザスター・トレラント（Disaster tolerant：災害耐性）と呼ばれる事もあります。

多くの密結合クラスタはスーパーコンピュータとして用いられる事が多く、2011 年 6 月に世界最速となった「京(K)」も密結合型クラスタのひとつです。

(http://www.nsc.riken.jp/K/diary.html)

スーパーコンピュータとして利用される場合は HPC(High Performance Computing / Clustering)と呼ばれることもあり、最近のスーパーコンピュータの主流となっています。

表3： 世界のスーパーコンピュータ TOP10 (2011 年 6 月)

#	システム名	運営者	OS	core	Rmax(GFlops)	Rpeak(GFlops)
1	京(K)	理科科学研究所 計算科学研究機構(日)	Linux	548,352	8,162,000	8,773,630
2	Tianhe-1A	国立スーパーコンピュータセンター天津(中)	Linux	186,368	2,566,000	4,701,000
3	Jagur	オークリッジ国立研究所(米)	Linux	224,162	1,759,000	2,331,000
4	Nebulae	国立スーパーコンピュータセンター深圳(中)	Linux	120,640	1,271,000	2,984,300
5	TSUBAME 2.0	東京工業大学 GSIC センター	Linux	7,327	1,192,000	2,287,630
6	Hopper	ローレンス・バークレー国立研究所*1(米)	Linux	153,408	1,054,000	1,288,627
7	Tera-100	原子力庁(仏)	Linux	138,368	1,050,000	1,254,550
8	Roadrunner	ロス・アラモス国立研究所*2(米)	Linux	122,400	1,042,000	1,375,776
9	Kraken XT5	テネシー大学(米)	Linux	98,928	831,700	1,028,851
10	JUGENE	ユーリヒ研究センター(独)	Linux	294,912	825,500	1,002,701

*1) DOE/SC/LBNL/NERSC Department of Energy / Office of Since / Lawrence Berkeley National Laboratory /National Energy Research Scientific Computing Center
*2)DOE/NNSA/LANL National Nuclear Security Administration(エネルギー省国家核安全保障局) / LOS AlamOS National Lab

（出典： http://www.top500.org/）

2.仮想化

前出のように可用性を実現するには、非常に多くの構成要素があり、要件検討が重要だというになります。そんな中、最近は可用性を向上させる手段として「仮想化」が注目されています。

仮想化とは1つのハードウエアをソフトウエアによって、複数のハードウエアがあるように見せかける技術で、古くは1970年代に汎用機で実装されVMモニタと呼ばれていました。当時は高価な大型計算（ハードウエア）を複数用意する事が難しく、少ない資源を有効に活用するために編み出されました。現在の仮想化ブームの火付け役ともいえるVMwareはx86(_64)用仮想化ソフトとして1998年に誕生しました。

表4： 主な仮想化ソフト

	VMware vSphere	Xen Server	Hyper-V	KVM
ベンダー	EMC(VMware)	Citrix(Xen)	MicroSoft	Red Hat
特徴	最古参で安定しておりサポートが充実。高スケーラビリティ	オープンソースの高い拡張性。準仮想化[7]による高パフォーマンス。	Win Server 2008 添付。高コストパフォーマンス	カーネル組込で高パフォーマンスが期待できる。

最近ではサーバ側だけでなく、クライアントの環境を仮想化しサーバに集約する「仮想化ディスクトップ」と「シン・クライアント(Thin Client)」の組み合わせも普及しています。

2-1. ハイパーバイザ

仮想化ソフトは大きくホストOS型とハイパーバイザ型の2種類に分かれます。

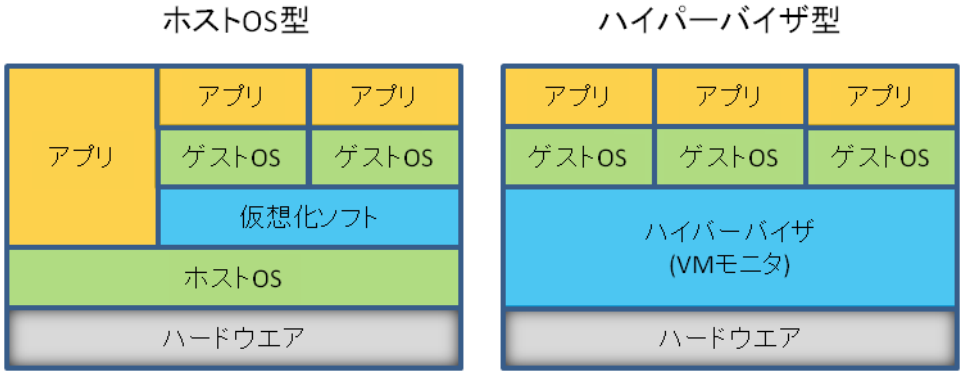

図 7 仮想化ソフトの種類

ホストOS型は、既に動作しているOS上でアプリケーションとして仮想化ソフトを動作させ、さらに、その上にゲストOSをインストールします。VMware Workstation / Server や、MS Virtual PC などがこの方式です。ハイパーバイザ型は仮想化ソフトウエアを OS としてインストールし、その上にゲスト OS をインストールして動作させます。Xen や KVM、VMware ESX、MS Hyper-V はこのタイプになります。ハイパーバイザはホストOS型に比べ間接的な処理が少なく処理効率が高いという利点があります。

なおハイパーバイザ型は仮想ソフトが直接ハードウエアを駆動することから「ベアメタル(Bare Metal、金属むきだしの意)」や、「VM(仮想マシン)モニタ」とも呼ばれます。OSの別名がスーパーバイザだったことから、その上位に位置づくためハイパーバイザと呼ばれています。

[7] 仮想化用にチューニングされたゲストOSを用いるタイプ。ゲストOSに制限がある。

2-2. 仮想化のメリット

仮想化によるメリットとしては以下のものがあります。

可用性の向上（システムのカプセル化）

ハードウエアに障害が発生し縮退運転となった場合でも、仮想マシンを他のハードウエアへコピーして継続する「マイグレーション」機能、システムの共通部分を「ひな型」として用意し、サーバ追加時は固有部分を追加・変更しすばやく立ち上げる「テンプレート」機能。同様に仮想マシンを丸ごとコピーやバックアップし、ハードウエア固有部分を書き換えて起動する「クローン」機能があります。

リソース有効利用

もともとハードウエアは１セットなのでコンピュータ資源の一元管理が可能で、資源をハイパーバイザやホスト OS を通じ、ゲスト OS へ動的に割り当てる事ができます。余裕のあるサーバから負荷の高いサーバへ資源を移動することができます。

レガシーマイグレーション（過去資産の移行）

ゲスト OS が必要とするハードウエアではなく、それに準ずる環境をソフトウエアで提供（エミュレーション）するため、販売終了・保守期限切れのハードウエアを前提とするソフトウエアでも稼働させることが可能です。Win XP SP-1 以前の OS を最新のハードウエアで動作させるためには固有のハードウエアやドライバなどが対応しないなど大変な苦労を伴いますが、仮想化であれば可能となります。

3. 実習手順

冗長構成を実現するには複数のコンピュータシステム（ハードウエア）が必要となりますが、LA の実習環境を鑑み、今回は仮想化を使って 1 台の PC 上にクラスタリングを実装することにします。

手順としては以下のようになります。

1. 仮想化
   PC 上に最小構成の Linux 仮想マシンを構築、以下仮想マシンで作業
2. LVM
   ハードディスクの冗長化例として LVM によるデバイス追加、ファイルシステムの拡張の実施
3. クローン
   作成した環境のコピーを作成し 2 つ目の仮想マシンを構築（作業効率向上のため、ソフトウエアのインストールは先行）
4. クラスタリング
   ２つの仮想マシンでクラスタを実現

3-1. 仮想化

実習では OSS でパフォーマンスがよく、動作環境の制限が少ない Xen を用いる事にしました。以下の手順に沿って、仮想マシンを構築します。

BIOS の設定

（BIOS は PC メーカ、機種によって内容が異なります。以下は DELL Optiplex 745）

HDD の設定を高速(SATA モード)、CPU の仮想化オプションを有効にしておきます。

System – Drivers

SATA Operation ［**Normal**］ Legacy

Performance

Virtualization ［**On**］ Off

ゲスト OS キットの準備

仮想マシンにインストールするための DVD を準備します。今回はホスト OS の DVD を HTTP 経由で読み込むため、Web サーバを立ち上げます。以下の手順を参考にしてください。

・DVD をトレーにセット、自動的にマウントされた場合はアンマウントする。

```
# umount /dev/scd0
```

・以下、Web による DVD 公開手順

1. Apache Web Server (以下、Web サーバ)のデータディレクトリ(DocumentRoot)下にマウントポイント作成

```
# mkdir /var/www/html/CentOS
```

2. DVD をマウント

```
# mount /dev/scd0 /var/www/html/CentOS
```

3. Web サーバの起動

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

4. 確認

http://localhOSt/CentOS にアクセスし DVD の内容を確認します。

パーティションの追加(オプション)

実習環境によってはディスクが不足する場合があります。df を実行し空き容量が 10GB 未満の場合はパーティションを追加します。

パーティションの追加
fdiskを使い、未割当領域を新たにパーティションとして追加します。この例では sda8 まで割り当てられている状態から、新しくsda9を作る手順を示します。パーティションのサイズは開始・終了シリンダとも既定値（空[Enter]）

```
# fdisk /dev/sda

Command (m for help): p            （現行パーティションの確認）

Disk /dev/sda: 106 MB, 106954752 bytes
   Device Boot    Start     End      Blocks   Id   System
/dev/sda1         1         6        6128   83   Linux
                  :
/dev/sda8        37         42       6128   83   Linux

Command (m for help): n           （パーティションの追加）
First cylinder (43-102, default 43):[Enter]
Using default value 43
Last cylinder or +size or +sizeM or +sizeK (43-102, default 102):[Enter]
Using default value 102


Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or
resource busy.
The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
```

1. パーティションテーブルの再読み込み
   fdisk　で警告が発生すると、OSのリブートが必要ですがこれはパーティションテーブル情報が同期できないためです。
   強制的に同期させるためには、partprobeを使います
   ```
   # partprobe /dev/sda
   ```

2. ファイルシステムの作成
   新しく作成されたパーティションを初期化し、利用可能な状態にします。
   ```
   # mkfs -t ext3 /dev/sda9
   # mkdir /images
   # mount /dev/sda9 /images
   ```

※ この操作を行った場合は､以下に登場するファイル名 /var/lib/xen/images を /images に読み替えて実行します。

仮想マシンの作成

CentOS の仮想マシンツールを使って、仮想マシンを作成します。このツールはパッケージグループ「仮想化」に含まれます。インストールが済んでいない場合は yum などで事前にインストールしておきます。

```
# yum  groupinstall "Viertualization"
```

1. 「仮想マシンマネージャ」起動
   最上行を選択し、[+新規(N)]ボタンをクリックし「新規の仮想マシンを作成」を起動
2. 「仮想システムの名前を指定」
   「システム名（N)」は任意。今回は vm01 とする。
3. 仮想化方式
   「準標準化」を選択
4. インストール方法
   「ネットワークのインストールツリー」を選択、OS タイプは Linux、OS 種別は Generic 2.6.25 or later kernel with virtualization を選択
5. ＵＲＬの指定
   installation media URL にて、先の CentOS をセットした場所を指定
   ・サーバはホスト自ＩＰ（192.168.xx.xx など。不明な場合は /sbin/ifconfig で確認）
   ・URL は(1)で作成したディレクトリ「http://192.168.xx.xx/CentOS」
6. ストレージ
   規定値を採用（File, Location = /var/lib/xen/images/vm01.img, 4000MB）
7. ネットワーク
   仮想ネットワーク(default)を採用
8. メモリとＣＰＵ割り当て
   メモリは最大を 512MB、起動時 256MB に、CPU は仮想 CPU 数を 1 とする。
9. 終了
   [終了(F)]を押すと、ディスクイメージが作成され初期設定が開始される。

※ディスク空き容量不足で仮想システム作成に失敗すると、巨大なゴミファイルが残ってしまうため、以下の手順で再作成します。

・「仮想マシンマネージャ」から、仮想システムを選び、[削除]を実行
・作成途中のファイルを削除、この例では /var/lib/xen/images/vm01.img
・再度上記の手順 1 から実行、手順 6 で指定するイメージファイルの出力先は十分に余裕のあるディレクトリを選択

ゲスト OS インストール

ここからは、仮想マシンの中で CentOS をインストールします。DVD をホスト OS で動く Web サーバ経由でインストールを行います。

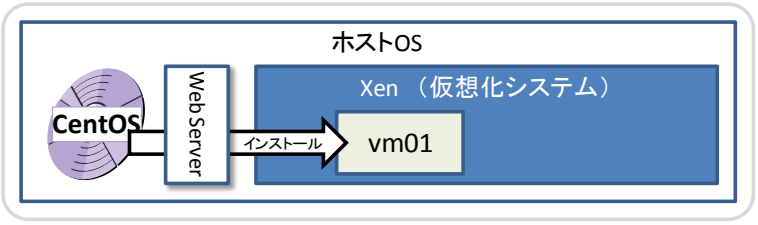

図 8 仮想マシンへのインストール

1. 言語選択：日本語(Japanese)
2. キーボード：日本(JP106)
3. ネットワーク設定：IPv4 を選択
   IP アドレス、ホスト名などは自動設定（DHCP）とします。
   直後に Language Unavailable(指定言語でメッセージが表示できない)、Welcome が表示されますが、それらは[OK]で進みます。
4. パーティションの指定
   パーティションテーブルが見当たらない旨のエラーメッセージが表示されますが、それは無視します。
   パーティションは仮想化ディスク(xvda)を選択し全て削除、デフォルトパーティションを採用します。(Remove all partition on selected disk…)
   削除確認(You have chosen remove all partition …)には[YES]、レイアウト確認(Review partition Layout..)は任意。
   またメモリ不足が指摘(Low memory)される場合がありますが、無視して先に進みます。
5. ネットワーク設定
   ・ネットワーク設定(Configure Network Interface..) は[YES]
   ・アダプタ毎の設定(Network Configuration for eth0)では、ブート時に起動(Activation on boot)と IPv4 使用(Enable IPv4 Support..)を有効に
   ・IPv4 設定(IPv4 Configuration for eth0..)では、192.168.122.101/24 を指定
   ・ホスト名設定(Hostname Configuration)は server1.localdomain を指定
   ・GATEWAY, DNS とも 192.168.122.1 を指定
6. タイムゾーン設定：システムクロックは未選択、地域は Asia/Tokyo を選択
7. ルートパスワード：忘れずに覚えておいてください。
8. パッケージ選定
   詳細を選択し、必要最小限のものを選択します。CUI だけで問題ありません、この場合は Base と Editor(vim)を選択します。この時に必要となるディスク容量は 1.5 BG+スワップ領域となります(vm01 の設定であれば合計 2GB)。

GUI を使う場合は、X Window System が必須、GNOME や KDE ディスクトップも必要でしょう。

表5： 必要最小限パッケージ(例)

パッケージ名		内容
Base	ベース	コマンド類(ユーザーコマンド、管理者コマンド)
Editors	エディター	テキストエディタ vim

3-2. LVM

LVM(Logical Volume Manager)は、複数の HDD（またはパーティション）を一つの大きなディスクにみたて、そこから必要な容量を切り出してディスクのように扱う機能です。RAID のようにミラーリングやストライピングが可能なほか、割り当てたディスクの拡大・縮小ができ柔軟なシステム管理を可能とします。多くのディストリビュータで標準機能として提供されています。

以下が LVM の概念図で、材料となる HDD を物理ボリューム(PV: Physical Volume)、その集合をボリューム・グループ(Volume Group)、切り出した領域を論理ボリューム(LV: Logical Volume)と呼びます。

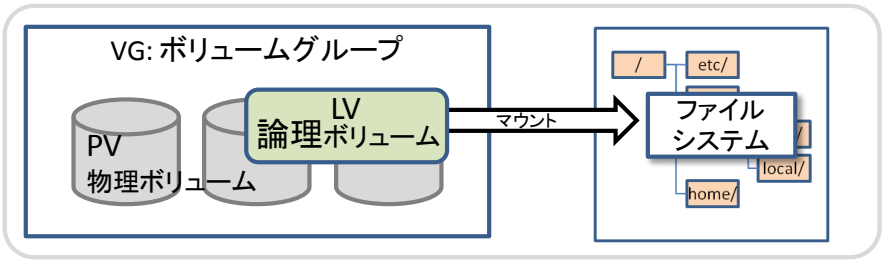

図 9 LVM 概念図

手順は大きく 3 つに分かれます、追加する HDD の準備、LVM の設定と LVM を用いたファイルシステムの作成です。

1. HDD の準備

i. 仮想ディスクの準備

実際のマシンを用いる場合は、本当に外付け HDD や拡張パーツを購入し PC 本体へ接続する必要がありますが、仮想化システムなのでソフトウエア的にディスクを追加します。ディスクはホスト OS 上に大きな 1 つのファイルとして作成され、ゲスト OS はそれを 1 つのディスクまたはパーティションに見立てて入出力します。

・仮想マシンの「ハードウエアタブ」の［+Add Hardware］ボンタンをクリック

・ハードウエアタイプ：Storage、ファイルはディスクイメージ

・ファイル名はフルパスで指定(/var/lib/xen/images/disk1)、サイズは 100MB

ii. パーティション作成(fdisk)

仮想ディスクにパーティションを割り当てます。今回は 1 ディスクまるごと利用するので、パーティションはひとつです。

```
# fdisk /dev/xvdb
```

2.LVM 設定

i. 物理ボリュームの作成

パーティション作成が済んだデバイスファイル（末尾が数字の１）を物理ボリュームとして設定します。このとき、他で使用されてない事を df, pvdisplay, swapon で確認します。

```
# pvcreate /dev/xvdb1
```

ii. ボリューム・グループの作成

物理ボリューム(PV)をボリュームグループ(VG)に参加させます。複数の PV をまとめて一つの VG に参加させる事もできます。今回は PV が 1 つの VG を作成します。

```
# vgcreate vgwk /dev/xvdb1
```

iii. 論理ボリュームボリュームの作成

作成した VG から、論理ボリュームを切り出します。オプション設定によりストライピングやミラーリングを行う事もできます。

（割り当てたディスクの容量は 100M ですが、管理用情報があるため、少し少なめの 80M を割り当てています。）

```
# lvcreate -L 80M -n lvwk vgwk
```

3-3. クローンによるバックアップ環境の構築

この実習ではクラスタリングを行います。つまりほぼ同じ設定内容のシステムを 2 つ構築しなければならず、インストールや設定を 2 回繰り返すことになり、とても面倒です。そこで仮想化のメリットであるクローン機能を使って、既存の仮想マシンを別名でコピーします。

クラスタリングでの作業を減らすため、必要なソフトウエアのインストールと共通の設定については先行する事にします。

1.ネットワークの修正

ゲスト OS インストール時の設定が正しいか確認します。IP アドレスは ifconfig eth0 や、ip addr show eth0 などを用います。ホスト名は hostname、ルータ（デフォルトゲートウェイ）は retstat -r や route を用い最下部に表示される “default” で表示される行の値です。

・/etc/sysconfig/network-scripts/ifcfg-eth0

現在の IP アドレス、ネットワークアドレスを確認し、パラメータを修正します。

```
BOOTPROT=static
IPADDR=192.168.122.101
~~HWADDR=xx:xx:xx:xx:xx:xx~~ （削除、又はコメントアウト）
NETMASK=255.255.255.0
```

・/etc/hosts

自分自身のホスト名、ＩＰアドレスとクラスタを組む相方、クラスタ全体の代表名を追加します。また localhost(127.0.0.1)の正式名称として登録された server1 の記述は削除します。

```
127.0.0.1   ~~server1.localdomain server1~~ localhost.localdomain localhost
          （前半の server1 に係る部分を削除）
192.168.122.101   server1.localdomain  server1    （追加）
192.168.122.102   server2.localdomain  server2    （追加）
192.168.122.100   server.localdomain  server      （追加）
```

2.HA 関連ソフトウエアのインストール

今回の実習では 2 つの HA 関連ソフトを用います、ハードディスクをデュプレックス構成にする DRBD(Distributed Replicated Block Device)と、サーバ同士の死活監視と切り替えを行う Hearbeat です。

DRBD はカーネルに組み込まれるため、動作するカーネルのバージョンごとに異なる種類が用意されています。2011 年 5 月現在、CentOS のディストリビューションからは、Ver 8.0 系、8.2 系、8.3 系の 3 種類が提供されています。対応する OS のバージョンに注意が必要です。また DRBD 本体だけでなく、カーネルモジュールも必要で、これはさらにカーネルの種類により 3 系統[8]提供されています（計 9 種類）。カーネルの種類は uname –a コマンドで確認します。

```
[root@server1 ~]# yum list "*drbd*"
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * addons: ftp.oss.eznetsols.org
 * base: ftp.oss.eznetsols.org
 * extras: ftp.oss.eznetsols.org
 * updates: ftp.oss.eznetsols.org
Installed Packages
drbd83.i386                  8.3.8-1.el5.centos        installed
kmod-drbd83.i686             8.3.8-1.el5.centos        installed
Available Packages
drbd.i386                    8.0.16-5.el5.centos       extras
drbd82.i386                  8.2.6-1.el5.centos        extras
kmod-drbd.i686               8.0.16-5.el5_3            extras
kmod-drbd-PAE.i686           8.0.16-5.el5_3            extras
kmod-drbd-xen.i686           8.0.16-5.el5_3            extras
kmod-drbd82.i686             8.2.6-2                   extras
kmod-drbd82-PAE.i686         8.2.6-2                   extras
kmod-drbd82-xen.i686         8.2.6-2                   extras
kmod-drbd83-PAE.i686         8.3.8-1.el5.centos        extras
kmod-drbd83-xen.i686         8.3.8-1.el5.centos        extras
```

今回は 8.3 系を用います。

```
# yum install kmod-drbd83-xen drbd83
```

つづいて、heartbeat をインストールします。Hearbeat は heartbeat, heartbeat-pils,

[8] ノーマル、Xen 用拡張(-xen)、物理アドレス拡張版（PAE:Physical Address Extension）

heartbeat-stonith の 3 つのソフトで構成されています。これも yum でインストールします。

```
# yum install heartbeat
(heartbeat 以外の 2 つは前提ソフトなので一緒にインストールします)
```

現在のバージョン Ver 2.1.3 にはインストール準備スクリプト(preinstall)にバグがあり、初回は失敗します。

```
error: %pre(heatbeat-2.1.3-xxx) scriptlet failed, exit status 9 ～
```

上記のようなエラーが表示されますので、再度 heartbeat をインストールしてください（前提ソフトは正常にインストールされるので 2 回目は heartbeat のみ、初回のインストールでエラー原因が取り除かれているため、そのままインストールできます）

```
# yum install heartbeat
```

2.クローンの実行

Xen でクローンを作成するには、いったん仮想マシンを停止する必要があります。仮想コンソールから vm01 を一時停止させてから、virt-clone コマンドを用います。

```
# virt-clone --original vm01 --name vm02 ¥
  --file /var/lib/xen/images/vm02.img ¥
  --file /var/lib/xen/images/disk2.img
```

virt-clone コマンドはイメージファイルのコピー以外に、仮想マシン名(--name)、UUID、MAC アドレスの設定変更を行う事もできます。今回は仮想マシン名を変えてコピーしています。

またコピー先ファイル(--file)をフルパスで指定していますが、vm02.img は仮想マシン本体、disk2.img は LVM 実習で追加した仮想ディスクです。

virt-clone は引数が非常に多く複雑なため、対話モードも用意されています。オプション –prompt により、解説を受けながらパラメータを指定することができます。

クローンが作成されたら、vm02 を起動しネットワーク関連ファイル(network の HOSTNAME, ifcfg-eth0 の IP アドレス)を修正し、再起動しておきます。

vm01, vm02 両方が起動したらそれぞれから ping や ssh などでネットワークが正しく動作しているか確認を行います。

3-4. DRBD の設定

DRBS(Distributed Remote Block Device)は、論理的なデバイスファイル(Block Device)を作成し、それを複数のマシンで同期する仕組みです。

共有するのではなく、正副(Primary / Secondary)2 系統のサーバとディスクを用意し、Primary に加えられた変更を、Secondary に随時反映するミラーリング方式をとっています。ディスクを直接更新できるのは Primary だけで、Secondary はバックアップ専用として直接ディスクをアクセスすることはできません（mount もできません)。

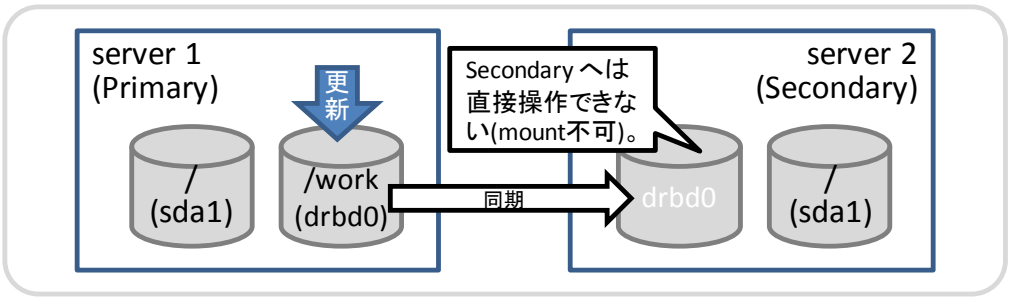

図 10 DRBD 概念図

DRBD の設定ファイルは /etc/drbd.conf で、大きくセクションと呼ばれるかたまりに分かれます。セクションは中カッコ({,})でかこまれ、その中にパラメータと値を記述しセミコロン(;)で終わります。コメントは#で始まり行末までとなります。

設定ファイルは正副共通なので、server1 側で作成し内容を確認後、scp などで server2 へコピーするとよいでしょう。

・/etc/drbd.conf

```
# Create new 2011/06/01, yakoshi
# DRBD configuration file
global { usage-count yes; }
common {
  protocol C;
  syncer { rate 10M; }
}
resource disk0 {
  net {
          cram-hmac-alg   sha1;
          shared-secret   "himitu";
  }
  on server1.localdomain {
          address         192.168.78.129:7788;
          device          /dev/drbd0;
          disk            /dev/vgwk/lvwk;
          meta-disk       internal;
  }
  on server2.localdomain {
          address         192.168.78.130:7788;
          device          /dev/drbd0;
          disk            /dev/vgwk/lvwk;
          meta-disk       internal;
  }
}
```

DRBD の起動と動作確認

まず server2(vm02)側を Primary として起動、設定を行います。

```
Server2# /etc/init.d/drbd start
Server2# drbdadm create-md disk0
Server2# drbdadm -- --overwrite-data-of-peer primary all
Server2# mkfs -t ext3 /dev/drbd0
Server2# mound /dev/drbd0 /work
```

```
Server2# /etc/initd/drbd status
drbd driver loaded OK; device status:
version: 8.3.8 (api:88/proto:86-94)
GIT-hash: d78846e ～ 2d422321d build by mockbuild@builder10.centos.org,
2010-06-04 08:04:16
m:res    cs         ro                 ds                 p  mounted  fstype
0:disk0  Connected Primary/Secondary Diskless/UpToDate C  /work    ext3
Server1#  (work ディレクトリ操作、ファイル作成など)
```

続いて server1(vm01)側を Secondary として設定します。

```
Server1# /etc/init.d/drbd start
Server1# drbdadm secondary all
Server1# /etc/init.d/drbd status
drbd driver loaded OK; device status:
version: 8.3.8 (api:88/proto:86-94)
GIT-hash: d78846e ～ 2d422321d build by mockbuild@builder10.centos.org,
2010-06-04 08:04:16
m:res    cs         ro                 ds                 p  mounted  fstype
0:disk0  Connected Secondary/Primary Diskless/UpToDate C
```

最後の primary/secondary を切り替え、動作を確認します。マウントしたままでは DRBD を操作できないので事前の umount が必要です。

```
Server2# umount /work
Server2# drbdadm secondary all

Server1# drbdadm primary all
Server1# mount /dev/drbd0 /work
```

表6： 主な DRBD のパラメータ

#	セクション/パラメータ	意味
1	global {}	定義ファイル全体に共通なセクション。必ず先頭で宣言する必要がある。
2	usage-count　yes/no	統計情報利用の有効化(yes)または、無効(no)
3	common {}	リソースに共通な情報を記載するセクション
4	syncer rate 速度	同期する最大転送速度。省略時は 250KiB/sec
5	protocol A/B/C	レプリケーション（同期）方法の指定。A: 非同期（自ディスクへ書き込んだ時点で完了)、B:メモリ同期（データパケットが他機に届いた時点で完了)、C:同期（両者のディスクに書き込んだ時点で完了）
6	resource リソース名 {}	リソースを定義するセクション
7	net {}	通信に関するセクション
8	cram-hmac-alg 方式	ノード認証用暗号化方式（アルゴリズム）の指定。/proc/crypt のものが利用可能。crc32c, sha1 など。
9	shared-secret キー	ノード認証で使う秘密鍵(最大 64bytes)
10	on ノード名 {}	ノード固有情報の指定セクション
11	address IP:PORT	IP アドレスとポート番号を指定。一般に DRBD のポートは 7788(省略不可)
12	device	DRBD が提供するデバイス名
13	disk	元となるデバイス名、ここに実データを記録する
14	meta-disk 配置場所	DRBD の管理情報(メタデータ)の配置場所の指定。internal は disk 上の末尾。他のデバイスファイルを指定することもできる。

3-5. Heartbeat の設定

Heartbeat はクラスタ間で相互に相手の稼働状況を監視し（死活監視）、異常を発見すると予め定義されたサービスを引き継ぎます。

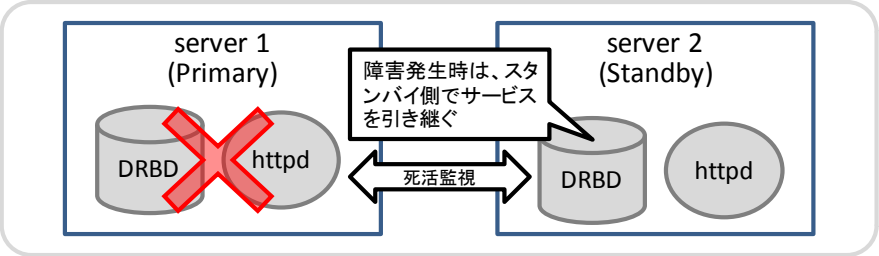

図 11 Heartbeat の概念図

サービスの継続にあたっては、OCF(Open Cluster Framework)と呼ばれるクラスタ専用の起動スクリプトがあり、これらを用いてサービスの引き継ぎ（再起動）を行います。具体的には/usr/lib/ocf/resource.d/heartbeat の下にあります。

Heartbeat の設定ファイルは、その動作を定義する ha.cf、クラスタメンバーの認証に用いる authkeys、ログ出力デーモンの logd.cf と起動用スクリプトを呼び出す haresources (cib.xml)があります。以降、それぞれの設定方法について解説します。

/etc/ha.d/ha.cf

/usr/share/doc/heartbeat-2.1.x/ha.cf を/etc/ha.d/にコピーし、ha.cf 修正。

ha.cf は heartbeat 全体の設定を行います。この内容は server1,server2 とも共通です。

```
keepalive         2                 (有効化)
initdead          120               (有効化)
udpport           694               (有効化)
bcast             eth0              (有効化)
watchdog  /dev/watchdog             (有効化)
node      server1.localdomain       (追加)
node      server2.localdomain       (追加)
use_logd  yes                       (有効化、yes にする)
crm               yes               (末尾に追加)
```

/etc/ha.d/authkeys

/usr/share/doc/heartbeat-2.1.x/authkeys を/etc/ha.d/authkeys にコピーし、修正。モードは 600 に変更。

ノード認証用のアルゴリズムと秘密鍵、server1/2 とも共通

```
auth  2             (有効化、修正)
2 sha1 himitu       (有効化、修正)
```

アルゴリズムは sha1, md5, crc の 3 種類がありますが、表記の順にセキュリティ強度が低くなります。CRC は簡単すぎる(厳密には暗号ではない)ので推奨されていません。

この２つのファイルは専用スクリプトによって、他のクラスタメンバーへコピーします。

```
Server1# /usr/lib/heartbeat/ha_propagate
Propagating HA configuration files to node server2.localdomain.
root@server2.localdomain's password: (server2 のパスワード)
ha.cf                         100%   11KB  10.7KB/s   00:00 ETA
authkeys                      100%  664     0.7KB/s   00:00 ETA
Setting HA startup configuration on node server2.localdomain.
```

/etc/logd.cf

/usr/share/doc/heartbeat-2.1.x/logd.cf を/etc/logd.cf にコピーし、修正。
Heartbeat のログ出力に関する定義ファイル、server1/2 とも共通。

```
debugfile /var/log/ha-debug              (有効化)
logfile          /var/log/ha-log         (有効化)
logfacility      none                    (変更、daemon → none)
```

haresources

サービス起動用のスクリプトファイルは、heartbeat 2.x から XML 形式に移行していますが、記述が面倒なため古い haresource 形式のファイルを作成し、それをコンバータで変換する方法がよく用いられています。このファイルはサーバごとに異なります。
まず root のホームディレクトリで（場所は/etc/ha.d *以外* であればどこでもよい）haresource を作成します。
～/haresources（１行のファイル）

```
server1.localdomain IPaddr2::192.168.78.100/24/eth0/192.168.78.255
        drbddisk::disk0 Filesystem::/dev/drbd0::/work
```

専用のスクリプトにより XML 形式に変換します。

```
Server1# pwd
/root
Server1# /usr/lib/heartbeat/haresources2cib.py haresources
Server1# ls -l /var/lib/heartbeat/crm/
total 4
-rw-r--r-- 1 hacluster haclient 3975 Jun 20 12:51 cib.xml
```

このスクリプトにはバグがあり、IPaddr2（コマンド）の NIC 名と Netmask が入れ替わっていますので、これを修正しておきます。

```
# vi /var/lib/eartbeat/crm/cib.xml
誤)       中略～ name="nic" value="24"/>
          中略～ name="cidr_netmask="eth0"/>
正)       中略～ name="nic" value="eth0"/>
          中略～ name="cidr_netmask="24"/>
```

この操作は server1/2 両方で行う必要があります。

Heartbeat の起動と確認

/etc/init.d/heartbeat スクリプトにより起動、crm_mon でクラスタの状況を確認することができます。また、ログファイルは /var/log/ha-log に保存されます。

```
# crm_mon
Defaulting to one-shot mode
You need to have curses available at compile time to enable console mode


============
Last updated: Mon Jun 20 14:11:23 2011
Current DC: server2.localdomain (ed7e8170-e673-4214-98da-7ceb1bc8b0a2)
2 Nodes configured.
1 Resources configured.
============

Node: server2.localdomain (ed7e8170-e673-4214-98da-7ceb1bc8b0a2): standby
Node: server1.localdomain (17e31103-31f8-41ef-b417-c617d6dfb5ba): online

Resource Group: group_1
    IPaddr2_1  (heartbeat::ocf:IPaddr2):     Started server1.localdomain
    drbddisk_2 (heartbeat:drbddisk):   Started server1.localdomain
    Filesystem_3    (heartbeat::ocf:Filesystem): Started
server1.localdomain
```

この例では、server1 で、IPaddr2(仮想化 IP)、drbddisk、ファイルシステムへのマウントが動作し、server2 がスタンバイとなっている事を表しています。
正副の手動による切り替えは、crm_standby コマンドを使います。

```
#crm_standby -U `hostname` -v on          (副に切換え)
#crm_standby -U `hostname` -v off         (正に切換え)
```

3-6. 動作実験

仮想マシン側では、ip コマンドや、crm_mon を使って本番機がどちらかを確認します。本番機では、/work に DRBD ディスクが割り当てられているので、適当なファイルを作成またはコピーできることを確認します。

続いてホスト OS 側から、代表 IP へ ping を飛ばしながら、上記で確認した本番機をシャットダウンします。

自動的にスタンバイ側にサービスが切り替わるのを確認し（その間、ping は停止しない、もしくは PC の性能によっては少しの中段）ます。

シャットダウンする直前に変更したファイルが、/work にあることを確認します。

実行例

```
Script started on Wed 29 Jun 2011 10:26:14 AM JST
[root@server2 ~]# crm_mon
Defaulting to one-shot mode
You need to have curses available at compile time to enable console mode

============
Last updated: Wed Jun 29 10:26:17 2011
Current DC: server1.localdomain (17e31103-31f8-41ef-b417-c617d6dfb5ba)
2 Nodes configured.
1 Resources configured.
============

Node: server2.localdomain (ed7e8170-e673-4214-98da-7ceb1bc8b0a2): online
Node: server1.localdomain (17e31103-31f8-41ef-b417-c617d6dfb5ba): online

Resource Group: group_1
    IPaddr2_1      (heartbeat::ocf:IPaddr2):       Started
server2.localdomain
    drbddisk_2     (heartbeat:drbddisk):   Started server2.localdomain
    Filesystem_3   (heartbeat::ocf:Filesystem):    Started
server2.localdomain
```

（Server2 が本番環境である）

本番機(Server2)にて、環境の確認

```
[root@server2 ~]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen
1000
    link/ether 00:0c:29:9a:65:40 brd ff:ff:ff:ff:ff:ff
    inet 192.168.78.130/24 brd 192.168.78.255 scope global eth0
    inet 192.168.78.100/24 brd 192.168.78.255 scope global secondary eth0
    inet6 fe80::20c:29ff:fe9a:6540/64 scope link
       valid_lft forever preferred_lft forever
```

（仮想 IP 192.168.78.100 が割り当てられていることがわかる）

/work(DRDB ディスク)にファイルを作成

```
[root@server2 ~]# date > /work/date20110629.txt
```

ホスト OS、またはスタンバイ環境から仮想 IP へ ping を行う

```
[root@server1 ~]# ping server
PING server.localdomain (192.168.78.100) 56(84) bytes of data.
64 bytes from server.localdomain (192.168.78.100): icmp_seq=1 ttl=64 time=1.97 ms
64 bytes from server.localdomain (192.168.78.100): icmp_seq=2 ttl=64 time=0.167 ms
                              （中略）
64 bytes from server.localdomain (192.168.78.100): icmp_seq=20 ttl=64 time=0.140 ms
64 bytes from server.localdomain (192.168.78.100): icmp_seq=21 ttl=64 time=0.150 ms
                              （この時点で、本番機をシャットダウン）
64 bytes from server.localdomain (192.168.78.100): icmp_seq=24 ttl=64 time=0.139 ms
64 bytes from server.localdomain (192.168.78.100): icmp_seq=25 ttl=64 time=0.146 ms
64 bytes from server.localdomain (192.168.78.100): icmp_seq=26 ttl=64 time=0.051 ms
64 bytes from server.localdomain (192.168.78.100): icmp_seq=27 ttl=64 time=0.047 ms
                              （中略）
64 bytes from server.localdomain (192.168.78.100): icmp_seq=45 ttl=64 time=0.059 ms

--- server.localdomain ping statistics ---
45 packets transmitted, 45 received, 0% packet loss, time 59718ms
rtt min/avg/max/mdev = 0.040/0.148/1.973/0.279 ms
```

環境が切り替わっている事を確認

```
[root@server1 ~]# crm_mon
Defaulting to one-shot mode
You need to have curses available at compile time to enable console mode


============
Last updated: Wed Jun 29 10:26:00 2011
Current DC: server1.localdomain (17e31103-31f8-41ef-b417-c617d6dfb5ba)
2 Nodes configured.
1 Resources configured.
============

Node: server2.localdomain (ed7e8170-e673-4214-98da-7ceb1bc8b0a2): OFFLINE
Node: server1.localdomain (17e31103-31f8-41ef-b417-c617d6dfb5ba): online

Resource Group: group_1
    IPaddr2_1       (heartbeat::ocf:IPaddr2): Started server1.localdomain
    drbddisk_2      (heartbeat:drbddisk):     Started server1.localdomain
    Filesystem_3    (heartbeat::ocf:Filesystem):       Started server1.localdomain
[root@server1 ~]# df
Filesystem     1K-blocks        Used Available Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                18156292     1774860  15444268  11% /
/dev/sda1          101086     20090    75777  21% /boot
tmpfs                      517552         0    517552   0% /dev/shm
/dev/drbd0           79287       5677    69516   8% /work


[root@server1 ~]# ls -l /work/date20110629.txt
-rw-r--r-- 1 root    root    29 Jun 29  2011 date20110629.txt
m[root@server1 ~]# exit
Script done on Wed 29 Jun 2011 10:26:22 AM JST
```

4.参考資料

- ✓ 高信頼システム構築標準教科書（LPI-JAPAN）
  http://www.lpi.or.jp/linuxtext/ha

- ✓ @IT Linux フォーラム
  http://www.atmarkit.co.jp/flinux/

- ✓ DRBD 日本サイト
  http://www.drbd.jp

- ✓ Linux Academy 関連資料
  http://ycos.sakura.ne.jp/LA