クラウド基盤構築演習

第一部
クラウド基盤を支えるインフラ技術
～ 第1回 Linuxサーバ構築の基礎知識

ver1.1 2012/05/21

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

目次

- はじめに: クラウドの現状とインフラ技術の役割
- OSの役割とデバイスドライバ
- Linux起動プロセス
- ディスクパーティションとLVMの構成
- ディスクイメージファイルとループバックデバイスの利用
- iSCSIターゲットの構成
- 参考資料: RHEL6における機能変更

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

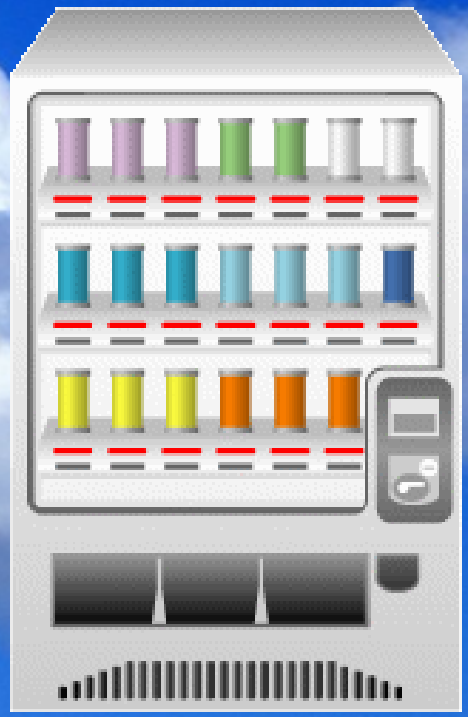
はじめに: クラウドの現状とインフラ技術の役割

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

これまでのクラウドが実現したもの

さまざまなITリソースを必要な時にすぐに利用できる環境

ITリソースの自動販売機



Copyright (C) 2012 National Institute of Informatics, All rights reserved.

利用者とサービス内容でクラウドを分類

企業内のユーザのみが利用

Privateクラウド

複数企業のユーザが共同利用

Publicクラウド

アプリケーション環境を提供 = SaaS
（エンドユーザーがすぐに使えるアプリケーション・サービス）

OS + Middleware 環境を提供 = PaaS
（プログラマがアプリケーションを開発できる環境）

Eucalyptus

OS 環境を提供 = IaaS
（管理者がシステムを構築・運用できる環境）

Amazon
EC2/S3

NII National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

企業向けクラウドに求められる5つの鍵

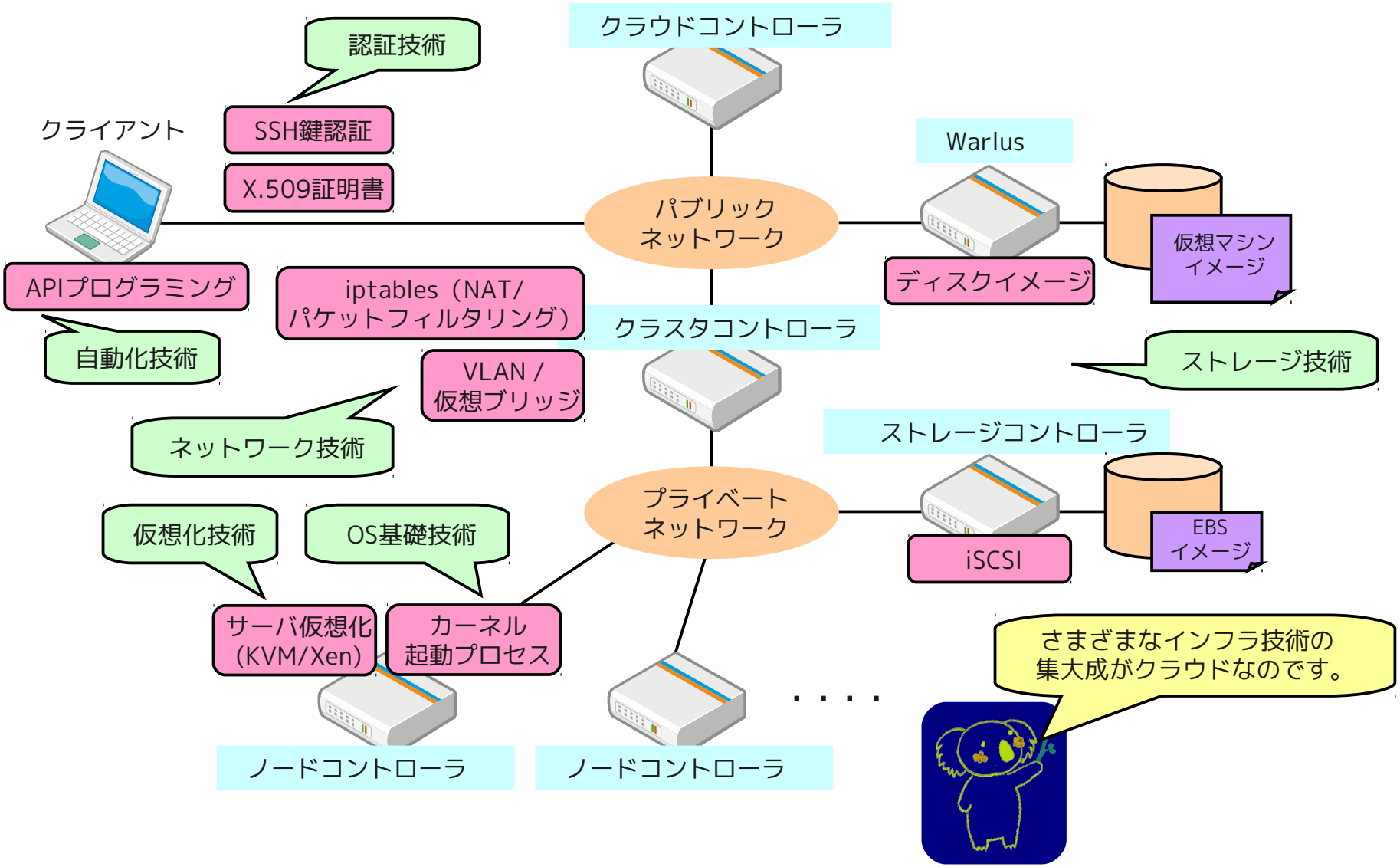
5つの鍵	内容	Linux/OSSの優位性
柔軟なサービス	お客様の必要に応じて自由にリソースの追加、変更ができる環境を提供	Linux/OSSは、ライセンスの制約に縛られずにリソースの追加が可能
適切なサービスレベル	お客様が必要とするサービスレベルとそれに見合った価格のサービスを提供	Linux/OSSベースのソリューションでサービス管理機能の実装が可能
可搬性のある環境	お客様のビジネス要件に応じて、様々なクラウドサービスの選択・移行が可能な環境を実現	Linux/OSSにより、特定ベンダに依存しないオープンスタンダード技術が活用可能
セキュリティとコンプライアンス	社内データセンターと同等の安全で管理された環境を実現	企業システムに求められるセキュリティ品質を満たすOSとしての実績を持つLinux
パートナーエコシステム	複数の企業が連携してクラウドサービスを提供する環境を確立	Linux/OSSは、クラウドサービス提供企業の共通基盤技術としてのデファクトスタンダード

さまざまな技術やサービスが連携するクラウド環境では、
標準技術としてのLinux/OSSの価値が発揮されます。

（出典）Linux and Commercial Software: Combining to Support the Cloud Environment
http://www.dbta.com/downloads/Hurwitz-IBM_Linux_Cloud.pdf

NII National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

Eucalyptusを実現するインフラ技術



NII National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

IaaSクラウド構築におけるインフラ技術の役割

- IaaSクラウドインフラは、

  『単純なものを組み合わせて、複雑なものを創り上げていく』

  というUnix/Linuxの思想にとてもよく適合する仕組みです。

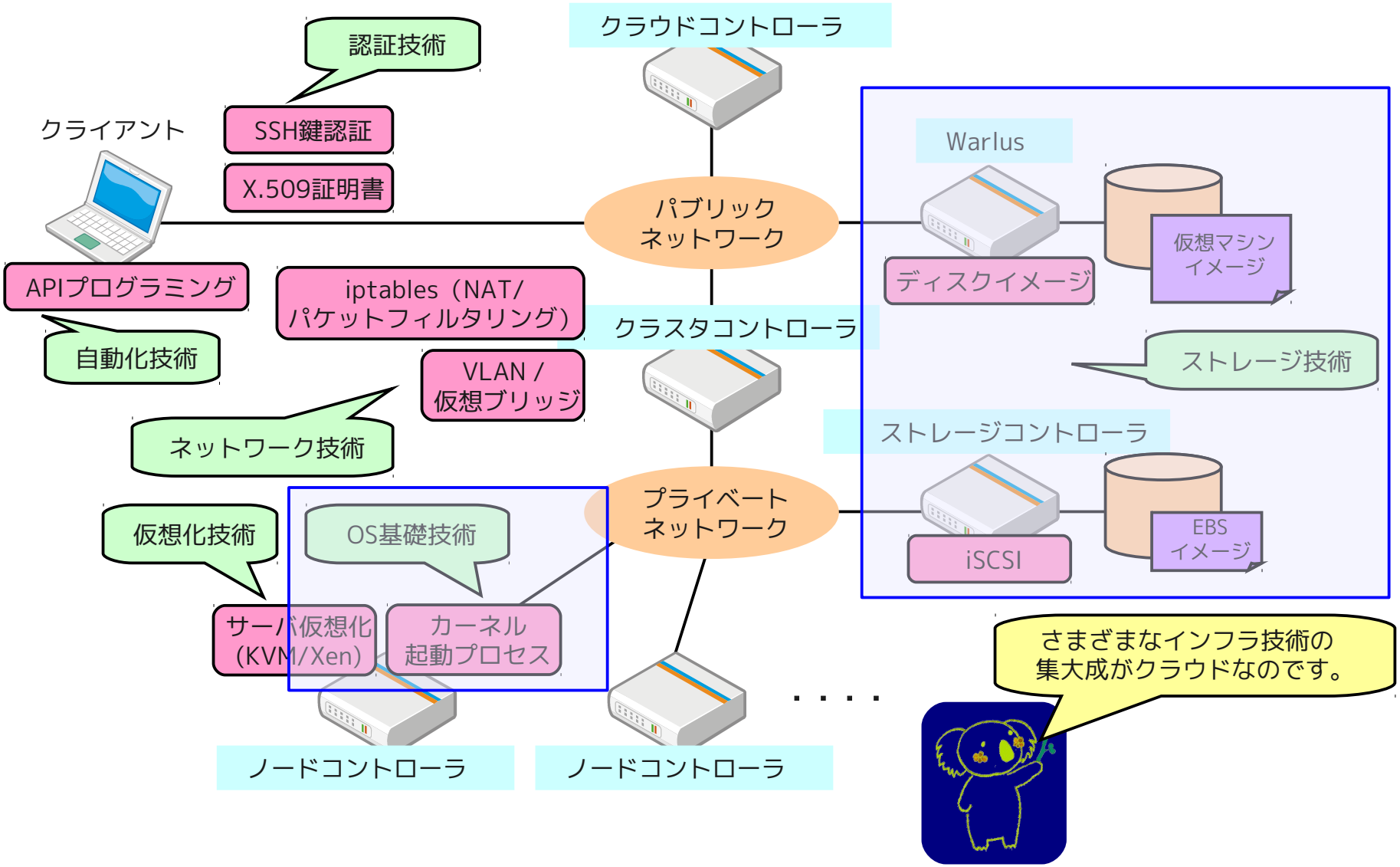
- まずは個々の基礎技術を根本から理解して、その上で

  『これらをどのように組み合わせれば最適なクラウドインフラが実現できるのか』

  というクラウドアーキテクチャの追求を行うことが大切です。

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

「第１回～第２回」の対象範囲



NII National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

メモとしてお使いください

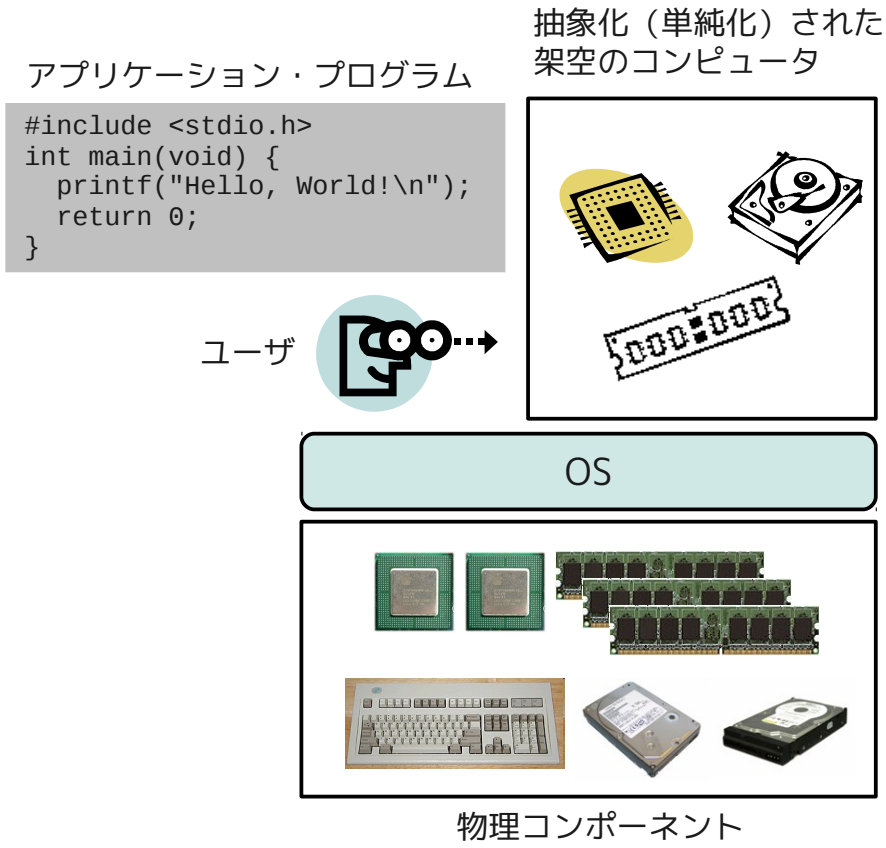
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

OSの役割とデバイスドライバ

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

OSの役割とは？

- OSは、物理コンポーネントの違いを吸収して、抽象化（単純化）された「架空のコンピュータ」をユーザ／アプリケーション・プログラムに提供します。
  - OSの「利用者」は、基本的にはハードウェアの違いを意識する必要がありません。
- OSを導入・設定する「システム管理者」は、ハードウェアの違いを理解した上で、適切に構成する必要があります。
  - 特にハードウェアに応じたデバイスドライバの選択、導入に注意が必要です。



OSによるハードウェアの抽象化

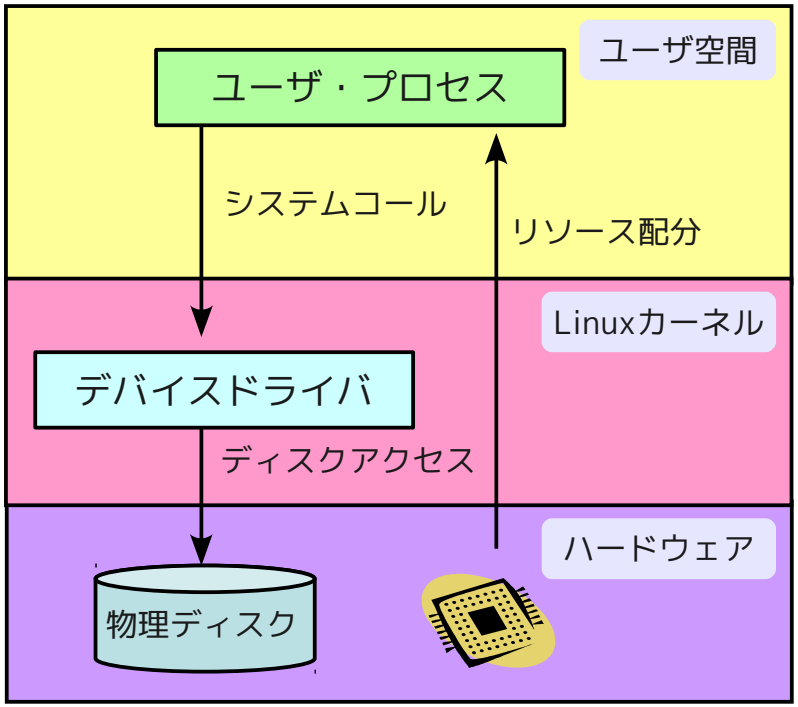
National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

デバイスドライバの役割

- ユーザ・プロセスがシステムコールにより物理ディスク、NICなどへのアクセスを要求すると、Linuxカーネルは適切なデバイスドライバを使用して、実際のアクセスを行います。

- Linuxカーネルのバージョンごとに、対応するデバイスドライバをコンパイルする必要があります。
  - Red Hat Enterprise Linuxでは、各カーネルのRPMパッケージ内に、カーネル本体とそれに合わせてコンパイルされたデバイスドライバー式が含まれています。
  - /lib/modules/<カーネルバージョン>/以下に該当カーネル用のデバイスドライバが導入されます。

- Linux起動時にルートファイルシステムをマウントするために必要なデバイスドライバは、初期ラムディスクにより提供します。
  - 初期ラムディスクは、この後で説明します。



National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

メモとしてお使いください

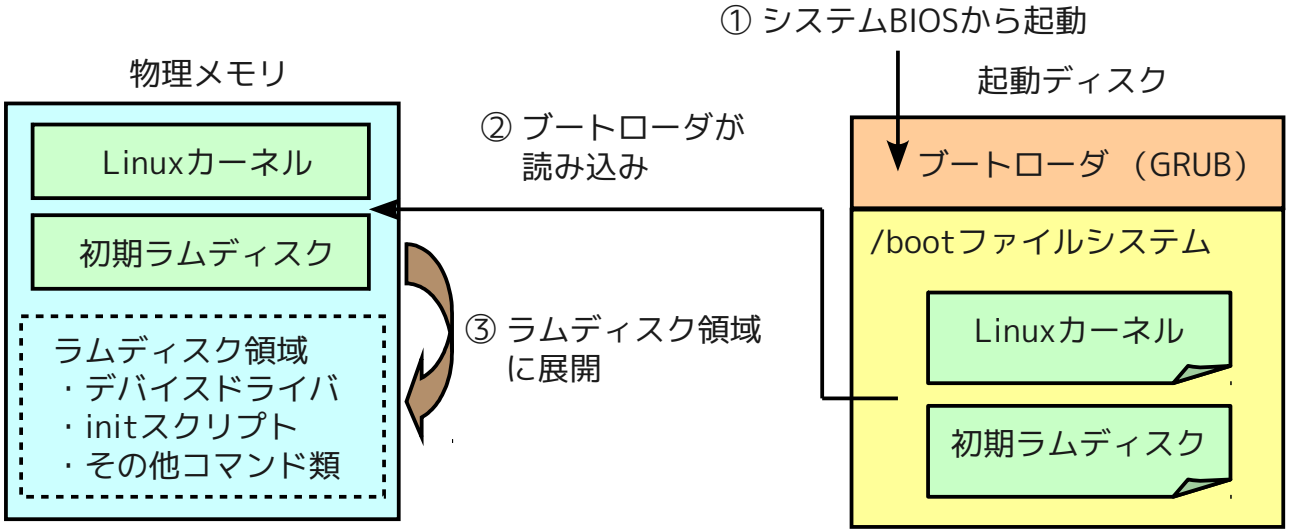
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

Linux起動プロセス

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

Linuxの起動プロセスと初期ラムディスク (1)

- サーバの電源を入れると、ハードウェアに搭載された「システムBIOS」が起動ディスクからブートローダ（GRUB）をメモリに読み込み、実行を開始します。
  - システムBIOSはデバイスドライバを使用せずに独自機能で起動ディスクにアクセスします。
- GRUBは起動カーネル選択画面を表示して、指定されたカーネルと初期ラムディスクをメモリに読み込んだ後に、カーネルの実行を開始します。
  - 初期ラムディスクには、ルートファイルシステムへのアクセスに必要なデバイスドライバと「initスクリプト」が含まれます。
- カーネルは、初期ラムディスクをメモリ上のラムディスク領域に展開して、「initスクリプト」を実行します。



National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

Linuxの起動プロセスと初期ラムディスク (2)

- 「initスクリプト」は、必要なデバイスドライバを読み込んで、ルートファイルシステムをマウントした後に、最初のプロセスとなる「/sbin/init」を実行します。
  - この後、/sbin/initは、ランレベルに応じたサービスを起動していきます。

- RHEL6より初期ラムディスクの形式が変更されています。
  - RHEL5までは、サーバの構成（ルートファイルシステムを配置する物理ディスク）に合わせて、適切なデバイスドライバを含む初期ラムディスク（/boot/initrd-*）を作成する必要がありました。
    - /etc/modprobe.confに必要なデバイスドライバを記載して、mkinitrdコマンドで作成します。
  - RHEL6からは、一般的なデバイスドライバを全て含んだ汎用的な初期ラムディスク（/boot/initramfs-*）が用意されるように変わりました。
    - この初期ラムディスクは、dracutコマンドで作成します。（mkinitrdコマンドは、dracutコマンドのラッパースクリプトに変更されています。）
    - 初期ラムディスクに含まれるファイルは、lsinitramfsコマンドで表示できます。

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

（参考）RHEL5とRHEL6のinitスクリプト比較

- 初期RAMディスクはcpio+gzipアーカイブファイルなのでpaxコマンドで展開可能です。
  - RHEL5： # pax -rzf /boot/initrd-2.6.18-238.el5.img
  - RHEL6： # pax -rzf /boot/initramfs-2.6.32-131.0.15.el6.x86_64.img

```
#!/bin/nash
（中略）
echo "Loading ehci-hcd.ko module"
insmod /lib/ehci-hcd.ko
echo "Loading ohci-hcd.ko module"
insmod /lib/ohci-hcd.ko
echo "Loading uhci-hcd.ko module"
insmod /lib/uhci-hcd.ko
mount -t usbfs /proc/bus/usb /proc/bus/usb
echo "Loading jbd.ko module"
insmod /lib/jbd.ko
echo "Loading ext3.ko module"
insmod /lib/ext3.ko
（中略）
echo Mounting root filesystem.
mount /sysroot
echo Setting up other filesystems.
setuproot
echo Switching to new root and running init.
switchroot
```

RHEL5の場合
→ insmodコマンドで明示的にドライバをロード

```
#!/bin/sh
#
# Licensed under the GPLv2
#
# Copyright 2008-2009, Red Hat, Inc.
# Harald Hoyer <harald@redhat.com>
# Jeremy Katz <katzj@redhat.com>
（中略）
# start up udev and trigger cold plugs
udevd --daemon –resolve-names=never
（中略）
    exec switch_root "$NEWROOT" "$INIT" $initargs || {
        warn "Something went very badly wrong in the init
ramfs.  Please "
        warn "file a bug against dracut."
        emergency_shell
    }
fi
```

RHEL6の場合
→ udevdで必要なドライバを判別してロード(*)

(*) udevdによるデバイスドライバの判別については、「参考資料: RHEL6における機能変更 – udevによるデバイスドライバの自動ロード」を参照。

National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

ディスクパーティションとLVMの構成

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

システム領域のディスクパーティション構成

- システム領域のディスクパーティションは、特別な要件が無い限り、次の単純な構成をお勧めします。

パーティション	フォーマット	説明
/boot	ext3 / ext4	通常は、200MB程度で十分です。
/	ext3 / ext4	16GB程度あれば、全てのパッケージを導入することが可能です。 内蔵ディスクの総容量に応じて決定します。
スワップ領域	Swap	Red Hat社のマニュアルでは、下記サイズが推奨されています。(*) ・実メモリが 2GB 以下の時：実メモリの 2 倍 ・実メモリが 2GB 以上の時：実メモリ + 2GB

(*) 本来はスワップ領域は不要な程度に、十分な実メモリを搭載することが望まれます。その場合は、最大でも2GB程度のスワップ領域を用意すれば十分です。

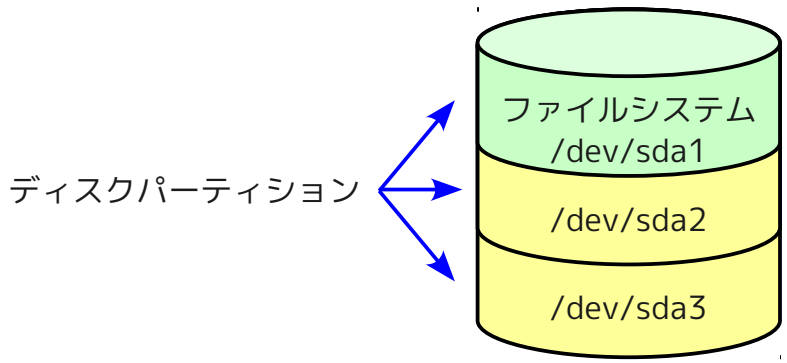
- インストール時のパーティション設定画面で『カスタムレイアウトを作成します』を選択後、上記のパーティションを指定します。
  - デフォルトの自動パーティショニングを選択すると、LVM (論理ボリュームマネージャ) を利用したパーティション構成になります。

NII National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

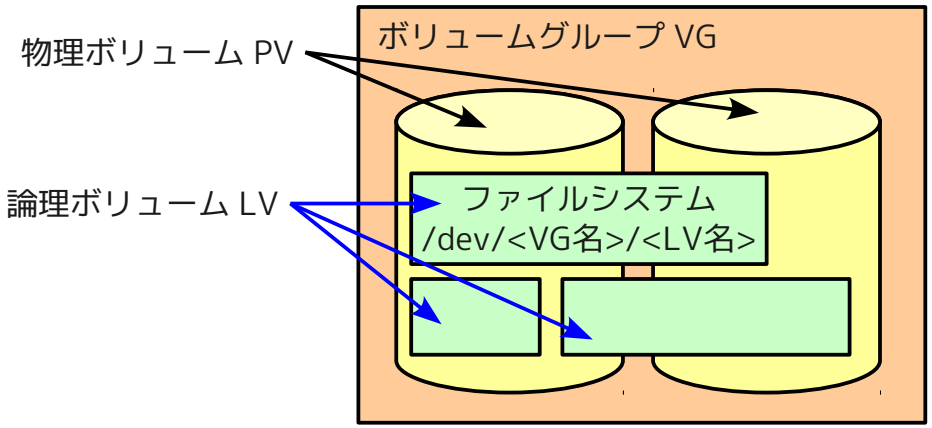
LVMの概要

- 論理ボリューム・マネージャ（LVM）では、物理ボリューム（PV）、ボリュームグループ（VG）、論理ボリューム（LV）の概念でディスクを管理します。
  - 物理ボリューム（PV）：個々の物理ディスク（/dev/sdX）のこと
  - ボリュームグループ（VG）：複数のPVをまとめたグループ
  - 論理ボリューム（LV）：VG内に作成した「仮想的なデバイス」
- LVの上にファイルシステムを作成して使用します。
  - 複数PVにまたがったLVが作成可能で、後からLVサイズの拡張も可能です。
  - 新たなPVを追加することで、VG全体のサイズも拡張可能です。

LVMによらないディスク管理



LVMによるディスク管理



NII National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

LVMの基本操作

- 新規PVからVGを構成して、LVを作成する手順の例です。(1)
  - pvcreate: /dev/sdbと/dev/sdcをPVとして登録
  - vgcreate: /dev/sdbと/dev/sdcからなるVG「vg_data00」を作成
  - lvcreate:「vg_data00」内に512MBのLV「lv_data00」を作成
    - -iオプションでは、複数PV（この例では2個）へのストライピングを指定しています。
- LVの上にファイルシステムを作成して使用します。(2)
  - 複数PVにまたがったLVが作成可能で、後からLVサイズの拡張も可能です。(3)
  - 新たなPVを追加することで、VG全体のサイズも拡張可能です。(4)

(1) PV/VGの構成とLVの作成

```
# pvcreate /dev/sdb
# pvcreate /dev/sdc
# vgcreate vg_data00 /dev/sdb /dev/sdc
# lvcreate -i 2 -L 512M -n lv_data00 vg_data00
```

(2) LV上へのファイルシステム作成とマウント

```
# mkfs.ext4 /dev/vg_data00/lv_data00
# mkdir /data00
# mount /dev/vg_data00/lv_data00 /data00
```

(3) LVとファイルシステムの拡張

```
# lvextend -L+128M /dev/vg_data00/lv_data00
# resize2fs /dev/vg_data00/lv_data00
```

(4) VGへの新規PVの追加

```
# pvcreate /dev/sdd
# vgextend vg_data00 /dev/sdd
```

構成情報の確認

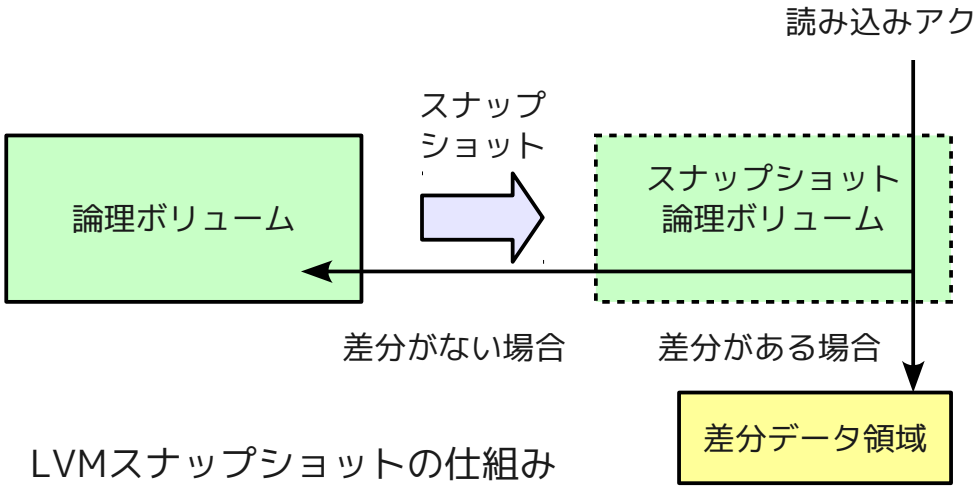
```
# pvdisplay
# vgdisplay
# lvdisplay
```


National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

LVMスナップショットについて

- 既存の論理ボリュームの「スナップショット論理ボリューム」が作成できます。
  - これは、元の論理ボリュームとの差分データを保存する領域だけを持ちます。元の論理ボリュームかスナップショットのどちらかに書き込みが発生すると、差分データが書きこまれます。
  - 読み込みアクセスの際は、差分がある場合は「差分データ」を読み、差分がない場合は元の論理ボリュームのデータを読みます。

- 差分データ領域が不足すると、スナップショットは使用不能になります。
  - 通常は、スナップショットからさらに物理コピーを取得した後に、スナップショットを破棄します。



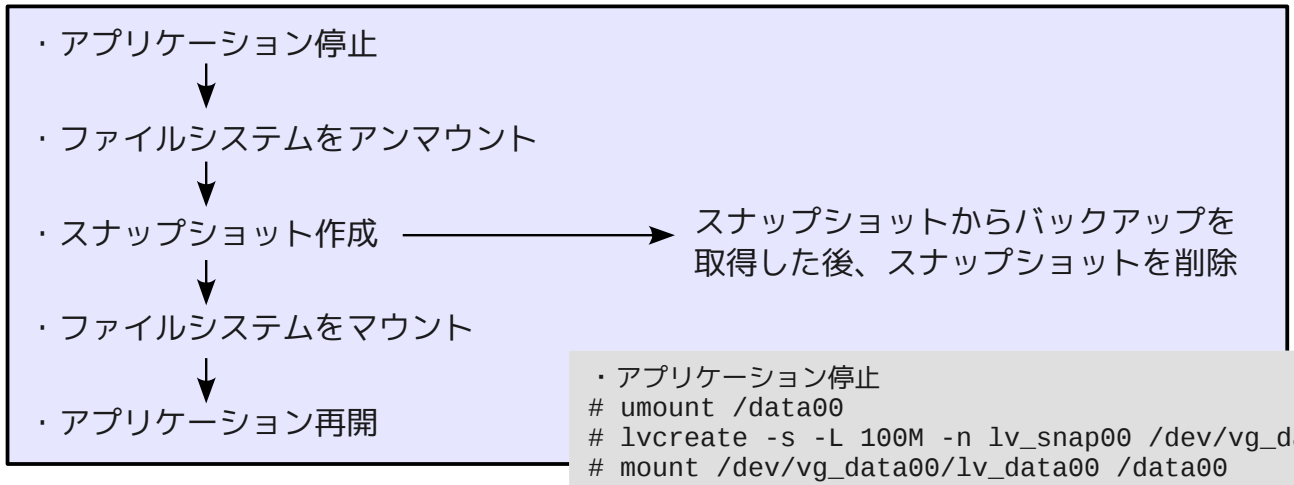
LVMスナップショットの仕組み

National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

LVMスナップショットによるバックアップ

- スナップショットの作成はほぼ一瞬で終わるので、バックアップ取得にともなうアプリケーション停止時間を短縮したい場合などに有用です。
  - スナップショットの作成と削除は次のコマンドを用います。
    - lvcreate -s -L <差分領域の容量> -n <スナップショットLV名> /dev/<VG名>/<LV名>
    - lvremove -f /dev/<VG名>/<スナップショットLV名>

LVMスナップショットによるバックアップ取得の流れとコマンド例




```
・アプリケーション停止
# umount /data00
# lvcreate -s -L 100M -n lv_snap00 /dev/vg_data00/lv_data00
# mount /dev/vg_data00/lv_data00 /data00
・アプリケーション再開

# mount /dev/vg_data00/lv_snap00 /mnt/snapshot
・/mnt/snapshotのファイルをバックアップ
# umount /mnt/snapshot
# lvremove -f /dev/vg_data00/lv_snap00
```


National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

メモとしてお使いください

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

メモとしてお使いください

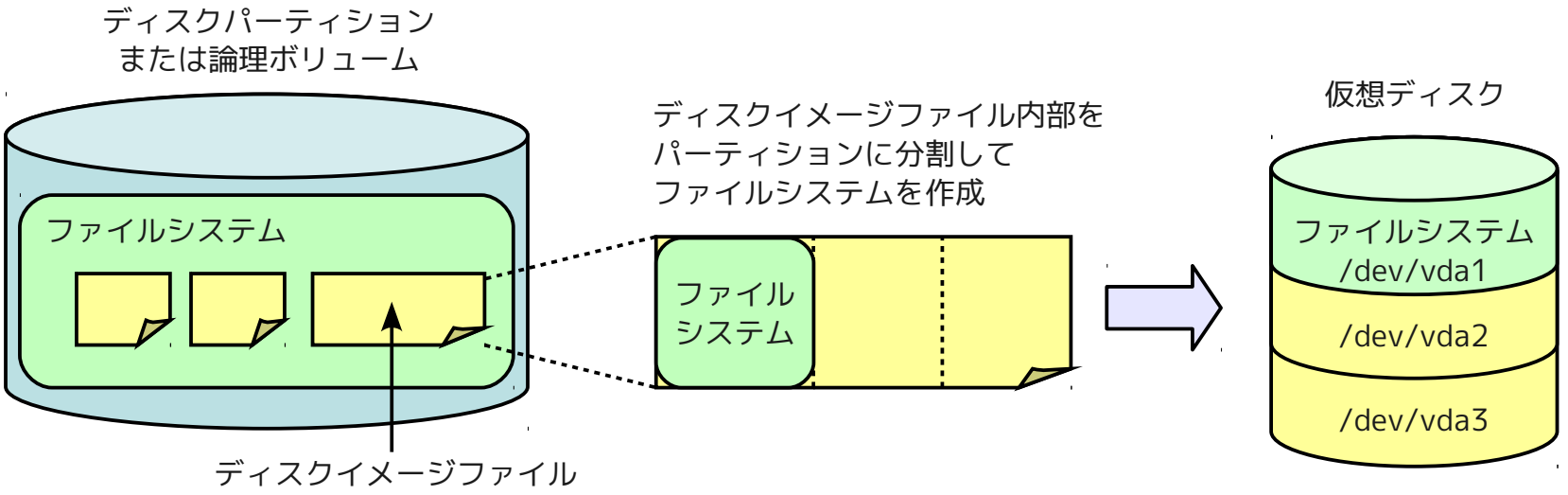
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

ディスクイメージファイルと
ループバックデバイスの利用

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

ディスクイメージファイルの仕組み

- 「ディスクイメージファイル」は、通常のファイルをディスクイメージとして使用する仕組みです。
  - DVDメディアをイメージ形式でファイルに書きだした「ISOイメージファイル」もディスクイメージの一種です。
- ディスクイメージファイルの内部をさらにパーティションに分割して、各パーティションにファイルシステムを作成することができます。
  - サーバ仮想化環境では、「ディスクイメージファイル」を仮想マシンに接続する仮想ディスクとして利用します。



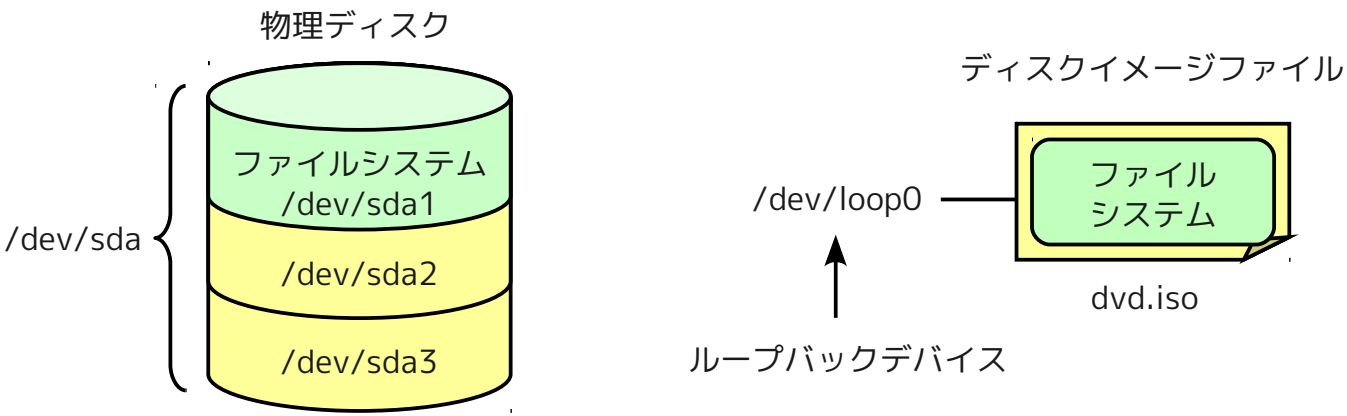
National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

ループバックデバイスの利用 (1)

- ディスクパーティションや論理ボリュームには、対応するデバイスファイル（「/dev/sda1」「/dev/<VG名>/<LV名>」など）があります。
  - ファイルシステムをマウントするときは、これらのデバイスファイルを指定します。

- ディスクイメージファイルを「ループバックデバイス」に紐づけることで、イメージ内のファイルシステムをマウントして編集することができます。
  - 内部にディスクパーティションを持たないISOファイルなどは、「-o loop」オプションをつけてマウントすると、自動的にループバックデバイスへの紐づけが行われます。
    - 次の例では、「dvd.iso」がループバックデバイス「/dev/loop0」に紐づけられています。

```
# mount -o loop dvd.iso /mnt/iso
# df | grep loop
/dev/loop0              3351190   3351190         0 100% /mnt/iso
```

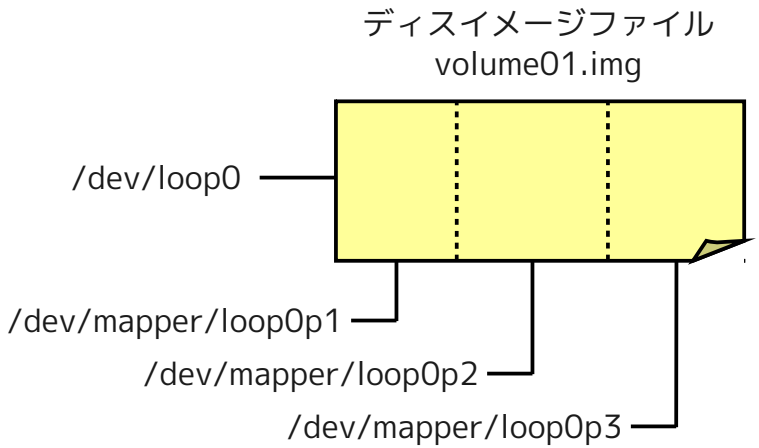


National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

ループバックデバイスの利用 (2)

- ディスクイメージファイル内部にパーティションを作成する場合は、それぞれのパーティションを個別にマッピングする操作が必要になります。
  - 次のコマンドで各パーティションを「/dev/mapper/loopXpY」にマッピングします。
    - losetup -fv <ディスクイメージファイル> : イメージファイル全体を紐付けます。
    - kpartx -av <ループバックデバイス> : イメージファイルに含まれるパーティションをマッピングします。
    - kpartx -d <ループバックデバイス> : マッピングを解除します。
    - losetup -d <ループバックデバイス> : イメージファイル全体の紐づけを解除します。
  - 次は3つのパーティションを持つ「volume01.img」の第2パーティションをマウントする例です。
    - 「/bootパーティション」「ルートパーティション」「Swap領域」を持つ仮想ディスクのイメージです。

```
# losetup -fv volume01.img
Loopデバイス は /dev/loop0 です
# kpartx -av /dev/loop0
add map loop0p1 (253:0): 0 208782 linear /dev/loop0 63
add map loop0p2 (253:1): 0 8193150 linear /dev/loop0 208845
add map loop0p3 (253:2): 0 1044225 linear /dev/loop0 8401995
# mount /dev/mapper/loop0p2 /mnt/image
# ls -l /mnt/image/
合計 184
drwxr-xr-x.  2 root root  4096  2月 14 11:56 2011 bin
drwxr-xr-x.  2 root root  4096  2月 14 09:57 2011 boot
drwxr-xr-x.  2 root root  4096  2月 14 09:57 2011 dev
drwxr-xr-x. 74 root root  4096  3月  4 08:56 2011 etc
（以下略）
# umount /mnt/image
# kpartx -d /dev/loop0
# losetup -d /dev/loop0
```



National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

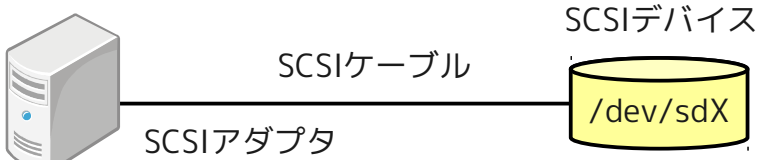
iSCSIターゲットの構成

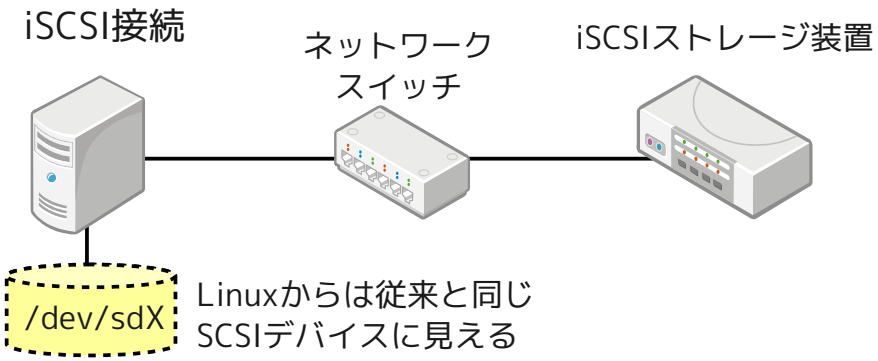
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

iSCSIの概要

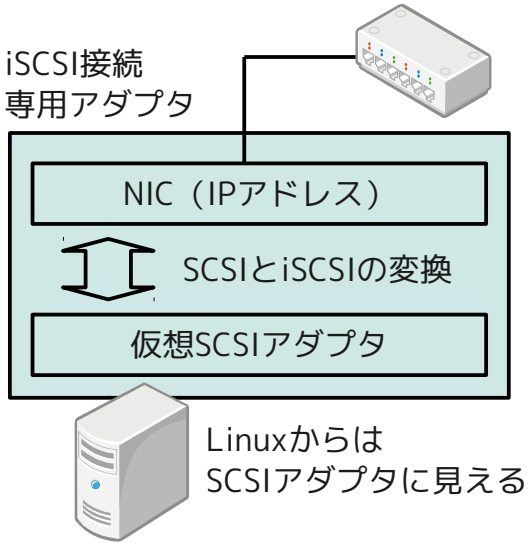
- iSCSIはストレージ接続の「SCSIプロトコル」をIPネットワークで転送する技術です。
  - サーバからは、従来と同じSCSIデバイスとして認識されます。
- iSCSI接続専用のアダプタを使用する「ハードウェア・イニシエータ」、もしくは通常のNICを使用する「ソフトウェア・イニシエータ」のどちらかを利用します。
  - ハードウェア・イニシエータでは、iSCSIの処理はアダプタ上で行われるので、Linuxからは通常のSCSIアダプタとして認識されます。
  - ソフトウェア・イニシエータでは、Linuxの機能でiSCSIの処理を行います。

従来のSCSI接続





ハードウェア・イニシエータの仕組み

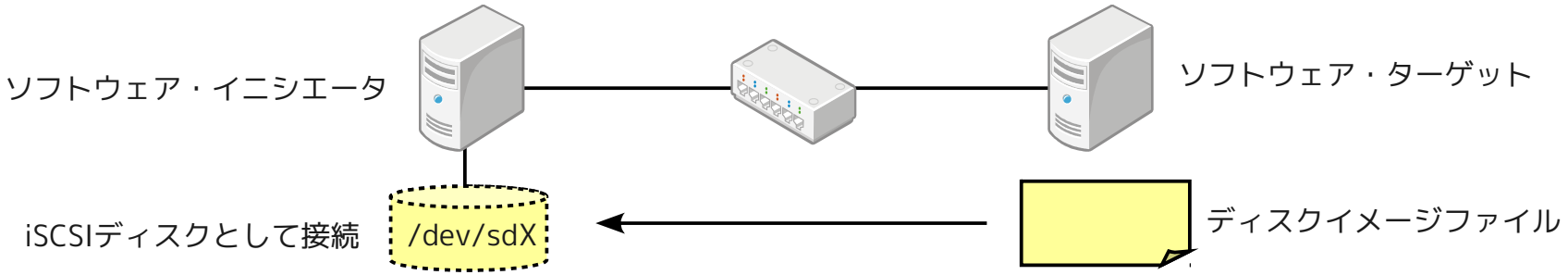


National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

ソフトウェア・ターゲットの利用

- Linuxの機能によりiSCSIストレージ装置をエミュレーションすることができます。これを「ソフトウェア・ターゲット」と呼びます。
  - Linuxサーバ上の論理ボリュームやディスクイメージファイルをiSCSIディスクとして、他のLinuxサーバに見せることができます。
  - 次は、それぞれ512MBのディスクイメージファイルをiSCSIディスクとして公開する例と、公開されたiSCSIディスクをソフトウェア・イニシエータで接続する手順の例です。

```
# yum install scsi-target-utils
# dd if=/dev/zero of=/var/lib/tgtd/volume01.img bs=1M count=512
# cat <<EOF >>/etc/tgt/targets.conf
<target iqn.2011-01.com.example.server01:tgt01>
    backing-store /var/lib/tgtd/volume01.img
</target>
EOF
# chkconfig tgtd on
# service tgtd start
```



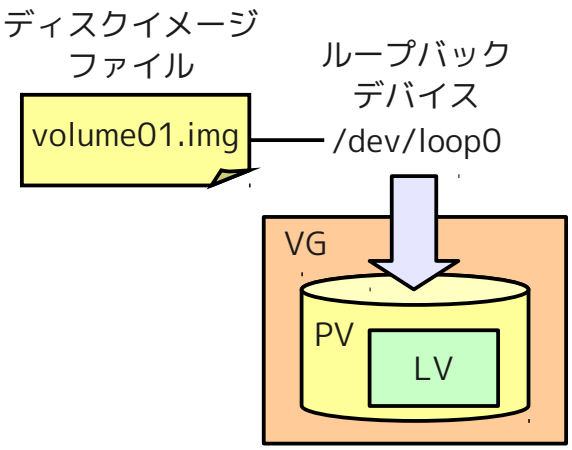

```
# yum install iscsi-initiator-utils
# iscsiadm -m discovery --type sendtargets --portal <ターゲットのIPアドレス>
# chkconfig iscsi on
# service iscsi start
```


National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

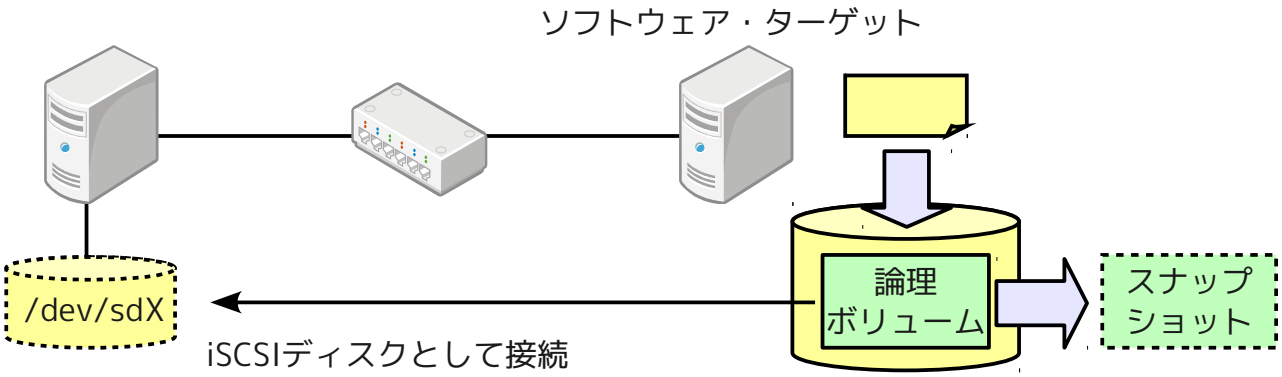
（参考）ディスクイメージとLVMの組み合わせ

- ループバックデバイスを利用すると、ディスクイメージファイルを物理ボリューム（PV）とみなしてLVMを構成することができます。
  - 次はディスクイメージファイルvolume01.imgをPVとして、論理ボリューム（LV）を作成する手順の例です。

```
# dd if=/dev/zero of=/var/lib/tgtd/volume01.img bs=1M count=576
# losetup -fv /var/lib/tgtd/volume01.img
Loopデバイス は /dev/loop0 です
# pvcreate /dev/loop0
# vgcreate vg_disk01 /dev/loop0
# lvcreate -L 512M -n lv_disk01 vg_disk01
```



- この論理ボリュームをソフトウェア・ターゲットでiSCSIディスクとして公開すると、iSCSIディスクのスナップショットコピーが取得できるようになります。



National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

参考資料: RHEL6における機能変更

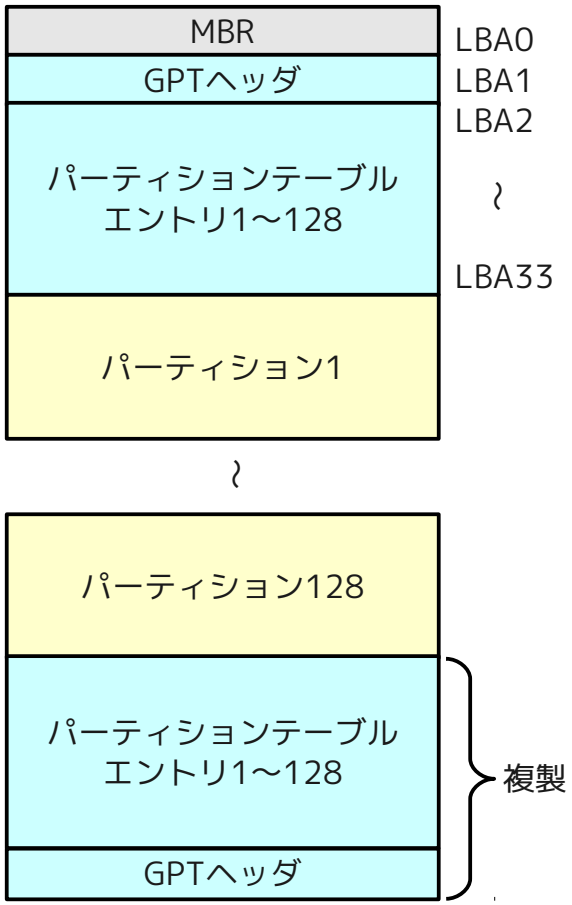
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

GPT（GUID Partition Table）ディスク・ブート

- 従来のパーティションは、2TB以上のディスクでは使用できません。
  - MBR内にパーティションテーブルがあり、記録できるセクタ番号が4バイトに限られているためです。
- 2TB以上のディスクでは、GPTを使用してパーティションを作成します。
  - ただし、ブートローダがGPTに対応している必要があります。

	従来のパーティション	GPT
パーティション テーブルの位置	MBR	MBRの直後（ディスクの末尾にも複製）
パーティション サイズの単位	シリンダ	LBA （通常は512バイト）
最大ディスク容量	2TB	8ZB（実質的に無限）
最大パーティション数	15（SCSIディスクの場合）	128
パーティションラベル	パーティションID （用途を表すID）	GUID（用途を表すID +ユニークID）
パーティション 作成ツール	fdisk	parted

GPTの構造



National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

ext4ファイルシステムの採用

- RHEL6では、ext4ファイルシステムがデフォルトのファイルシステムになります。

- ext4ファイルシステムの主な特徴は次のとおりです。
  - 最大ファイルシステムサイズの理論値は1EBです。ただし、RHEL6.0/6.1のサポート上限は16TBまでです。フィルサイズの上限は16TBです。
  - サブディレクトリ数の上限がありません。（ext3は32,000個が上限。）
  - ディレクトリ検索、およびファイルアクセスのパフォーマンスが向上しています。
    - ディレクトリインデックスにハッシュBツリーを採用しています。
    - 「エクステント」データ構造と事前アロケーションにより連続ブロックを効果的に使用します。
    - 特にKVMゲストのディスクイメージの作成時間が短縮されます。
  - ext3との前方互換性を持ちます。
    - ext3をext4に変換可能です。ext3をext4としてマウントすることもできます。ただし、これらは、RHEL6ではサポート対象外となります。ext3ファイルシステムからの移行は、新規作成したext4へのリストアが推奨されます。
  - タイムスタンプをナノ秒単位で管理します。

National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

Transparent huge pagesの設定方法

- RHEL6の「Transparent huge pages」は物理メモリの割り当て単位となるページサイズを必要に応じて、標準の4KBから2MBに自動拡張する機能です。
  - RHEL5以前で2MBページを使用する際は、アプリケーションが明示的にHugepageに対応している必要がありました。RHEL6ではアプリケーションに対して透過的にHugepageが割り当てられます。
  - KVM仮想マシンなど大容量メモリを使用するアプリケーションのパフォーマンスが向上します。
  - デフォルトで有効化されます。起動時に無効化する際は、/etc/grub.confのカーネル起動オプションに「transparent_hugepage=0」を指定します。

- コマンドによる設定変更方法
  - 設定の確認と変更（always:有効 / never:無効）
    - # cat /sys/kernel/mm/redhat_transparent_hugepage/enabled
    - # echo "never" > /sys/kernel/mm/redhat_transparent_hugepage/enabled
    - # echo "always" > /sys/kernel/mm/redhat_transparent_hugepage/enabled
  - 現在のHugepageの利用量を確認
    - # grep AnonHugePages /proc/meminfo

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

NetworkManagerサービスについて

- 「NetworkManagerサービス」は、ノートPCなど、ネットワーク接続が頻繁に変化する環境において、ネットワーク設定を自動変更するサービスです。
  - NetworkManagerサービスが管理するインターフェースは、設定ファイル「ifcfg-ethX」の書式が従来とは異なります。
  - 「ifcfg-ethX」内で「NM_CONTROLLED=no」を指定すると、NetworkManagerサービスの管理対象外となり、従来の書式での設定が可能です。

- サーバ環境では、NetworkManagerサービスの起動を停止することをお勧めします。
  - とくに、NetworkManagerサービスでは、KVMのホスト環境における仮想ブリッジを作成することができません。
  - 次は、NetworkManagerサービスを停止するコマンドの例です。

```
# chkconfig NetworkManager off
# service NetworkManager stop
```


National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

NICデバイスの命名規則

- DELL社など特定ベンダのサーバでは、NICデバイスの命名規則が変わります。
  - オンボードNICは「em[1-N]」、PCIカードのNICは「p<slot>p<port>」などになります。
  - これは、DELL社からのリクエストによる変更のためDELL社などのサーバのみで変更されます。

- それ以外の一般のサーバでは、MACアドレスに対して固定のデバイス名「ethX」が割りあてられます。
  - 新しいNICを接続してサーバを起動すると、連番でデバイス名「ethX」が割り当てられて、MACアドレスとデバイス名の関係が「/etc/udev/rules.d/70-persistent-net.rules」に記録されます。
    - 次は「/etc/udev/rules.d/70-persistent-net.rules」に記録された設定の例です。

```
# PCI device 0x14e4:0x164c (bnx2) (custom name provided by external tool)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:14:5e:fa:98:a2", ATTR{type}
=="1", KERNEL=="eth*", NAME="eth0"

# PCI device 0x14e4:0x164c (bnx2) (custom name provided by external tool)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:14:5e:fa:98:a0", ATTR{type}
=="1", KERNEL=="eth*", NAME="eth1"
```

  - NICを交換した場合などは、デバイス名の番号に抜けが発生します。このような際は、上記の設定ファイルを修正した後にサーバを再起動します。

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

Dracutによる初期ラムディスクの作成

- 初期ラムディスクの作成ツールがdracutに変更されています。
  - 初期ラムディスクのファイル名がinitrd-<version>.imgからinitramfs-<version>.imgに変更されています。
  - RHEL5までのmkinitrdは、/etc/modprobe.confなどからサーバ個別に必要なドライバを選択していましたが、dracutではサーバの構成に依存しない汎用的な初期ラムディスクを作成します。
    - 初期ラムディスクに含まれるドライバの読み込みにはudevが使用されます。udevはサーバのハードウェア構成を自動認識して必要なドライバをロードする仕組みです（次ページを参照）。
  - mkinitrdコマンドは、dracutコマンドのラッパーに変更されています。

- dracutコマンドの基本的な使い方は次のとおりです。
  - # dracut /boot/initramfs-2.6.32-71.el6.x86_64.img 2.6.32-71.el6.x86_64
    - -f: 既存ファイルを上書き
    - --add-drivers "foo bar": 特定のドライバを追加する
    - -H: そのサーバに必要なドライバのみを含める

- 初期ラムディスク(initramfs-*)に含まれるファイルは次のコマンドで確認できます。
  - # lsinitrd /boot/initramfs-2.6.32-71.el6.x86_64.img

National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

udevによるデバイスドライバの自動ロード

- udevが個々のデバイスに対して、必要なデバイスドライバを判別する方法は次のとおりです。
  - PCIデバイスなどは、/sysファイルシステムに登録されるとデバイスに固有の「modalias」情報が登録されます。
    - 次は、modalias情報の参照例です。

```
# cat /sys/devices/pci0000\:00/0000\:00\:06.0/0000\:03\:00.0/0000\:04\:00.0/modalias
pci:v000014E4d0000164Csv00001014sd00000342bc02sc00i00
```

  - 一方、各デバイスドライバは、サポートするデバイスのmodalias情報を保持しています。
    - デバイスドライバのmodalias情報は、modinfoコマンドで確認できます。

```
# modinfo bnx2
filename:       /lib/modules/2.6.32-71.el6.x86_64/kernel/drivers/net/bnx2.ko
...
alias:          pci:v000014E4d0000163Csv*sd*bc*sc*i*
alias:          pci:v000014E4d0000163Bsv*sd*bc*sc*i*
alias:          pci:v000014E4d0000163Asv*sd*bc*sc*i*
alias:          pci:v000014E4d00001639sv*sd*bc*sc*i*
...
```

    - これらの情報は、depmodコマンドにより、「/lib/modules/<カーネルバージョン>/modules.alias」にまとめて記録されています。
  - Linuxカーネルは新しいデバイスを認識すると、そのmodalias情報をudevに通知します。udevが次のmodprobeコマンドを実行すると、上記の情報を参照して、必要なデバイスドライバをロードします。
    - # modprove <modalise>

National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

ループバックデバイスの追加方法

- デフォルトで用意されるループバックデバイスは「loop0」～「loop7」の8個です。9個以上のループバックデバイスを同時に使用する場合は、事前に追加します。
  - RHEL5では、次を実行してサーバを再起動します。最大で「loop255」まで作成できます。

```
# echo 'options loop max_loop=256' > /etc/modprobe.d/loop
# for i in `seq 8 255`; do echo loop${i} >> /etc/udev/makedev.d/50-udev.nodes; done
```

  - RHEL6では、次を実行してサーバを再起動します。

```
# for i in $(seq 8 511); do echo "loop$i" >> /etc/udev/makedev.d/50-udev.nodes
```

    - 256 個以上の loop デバイスを作成する場合は、/etc/makedev.d/01linux-2.6.x で指定される最大値も修正しておきます。

```
#b $STORAGE               7   0  1 256 loop%d   # デフォルトは最大 256 個
b $STORAGE               7   0  1 2046 loop%d   # 最大 2046 個に変更
```


National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

Upstartによる起動処理 (1)

- Linuxカーネルが最初に起動するプロセス「/sbin/init」は、ユーザレベルのサーバ設定やサービス起動などの処理を行います。

- RHEL5の「/sbin/init」は、設定ファイル「/etc/inittab」にしたがって次の処理を行います。
  - 初期設定スクリプト「/etc/rc.d/rc.sysinit」を実行して、ファイルシステムのfsckチェックとマウント処理を実施します。
  - スクリプト「/etc/rc.d/rc」を実行して、起動時のランレベルに応じたサービスを起動します。
  - コンソールログインの受付プロセス(mingetty)を起動します。
  - ランレベル5の場合は、GUIのログイン画面を起動します。

- RHEL6の「/sbin/init」はUpstartと呼ばれるイベントベースのジョブ管理システムに変更されています。
  - 「/etc/inittab」はデフォルトのランレベル設定のみが記載されます。

RHEL5の/etc/inittabの例

```
id:5:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
（中略）

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm -nodaemon
```


National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

Upstartによる起動処理 (2)

- Upstartでは、「/etc/init/<Job>.conf」に個々のジョブの定義を記載します。
  - 基本的には、1つのジョブは1つのプロセスに対応します。
    - インスタンス指定により、同じジョブに属するプロセスを複数起動することも可能です。
  - 次のような内容を定義します。
    - ジョブを起動するイベント、停止するイベント
    - ジョブの起動コマンドなどの指定
    - ジョブに対応するプロセスが終了した際に再起動するかどうか(task/respawn)
  - initctlコマンドでイベントの発生、ジョブ管理などが可能です。
    - # initctl emit <イベント>
    - # initctl start <ジョブ>
    - # initctl stop <ジョブ>
    - # initctl restart <ジョブ>
    - # initctl reload <ジョブ>
    - # initctl status <ジョブ>
    - # initctl list

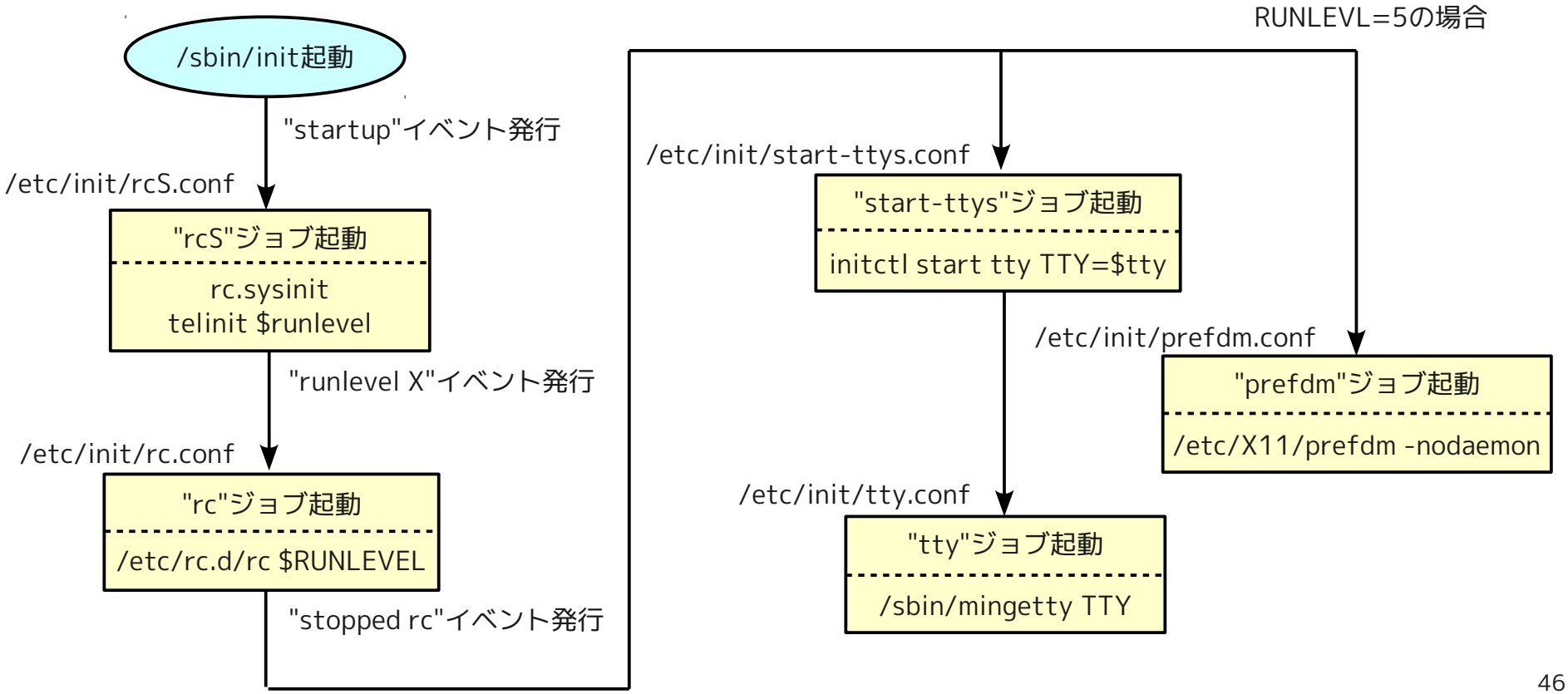
Upstartの主なイベント

イベント	説明
startup	/sbin/initの起動時に発生
starting <Job>	<Job>の起動処理の開始時に発生
started <Job>	<Job>の起動処理の完了時に発生
stopping <Job>	<Job>の停止処理の開始時に発生
stopped <Job>	<Job>の停止処理の完了時に発生
runlevel X	telinitコマンドでランレベル変更時に発生

National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

Upstartによる起動処理 (3)

- RHEL6では、RHEL5までの/etc/inittabと同等のプロセス起動をUpstartのジョブとして定義しています。
  - 下図は代表的なジョブの起動順序を示します。とくに「“rc”ジョブ」において、chkconfigコマンドで設定された/etc/init.d/以下のサービスを起動します。



National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

anacronによる定期ジョブの実行 (1)

- RHEL5では、日次・週次・月次の定期ジョブはcronデーモンが実行します。
  - サーバが停止していたなどで実行されなかったジョブは、サーバ起動時にanacronが再実行するように設定されています。
- RHEL6では、日次・週次・月次の定期ジョブは、anacronが実行します。
  - 実行されなかったジョブの再実行も同じanacronの機能で行われます。
  - その他の定期ジョブは、RHEL5までと同様にcronデーモンが実行します。
- anacronは次のような特徴を持ちます。
  - ジョブの実行間隔を1日単位で指定します。したがって、ジョブの実行頻度は、1日1回以上には設定できません。ジョブの実行時刻を指定するのではなく、実行時間間隔のみを指定します。
  - ジョブの設定は「/etc/anacrontab」に指定します。rootユーザのみが設定を行えます。
  - ジョブの実行タイミングにランダムな遅延を挿入することができます。これにより、同じ時刻に多数のジョブが同時起動する問題を回避します。
  - デーモンを持たないため、cronジョブから定期的にanacronコマンドを実行します。anacronコマンドを実行したタイミングで、実行するべきジョブがあれば、それが実行されます。

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

anacronによる定期ジョブの実行 (2)

- 下図は、RHEL6の「/etc/anacrontab」のデフォルト設定です。
  - anacronコマンドが実行されると、現在時刻が「START_HOURS_RANGE」に含まれている場合に、前回のジョブ実行日から"period in days"以上の日数が経過しているジョブを
    - 各ジョブの実行日は「/var/spool/anacron/<job-identifier>」に記録されています。（日付までで、時刻は記録されません。）
    - ジョブの開始から「"delay in minutes" +（0～RANDOM_DELAYの乱数）」分の遅延の後に、実際にジョブが実行されます。
  - anacronコマンドは、cronジョブにより毎時01分に実行されるので、この設定の場合、日時、週次、月次のrun-partsコマンドが「03:01（+遅延時間）」に実行されます。
  - サーバを長期間停止した後に再度起動すると（起動時刻が「START_HOURS_RANGE」に含まれている場合）、起動直後の01分のタイミングで必要なジョブが実行されます。その次の実行は、翌日以降の03:01に戻ります。

```
SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
RANDOM_DELAY=45
# the jobs will be started during the following hours only
START_HOURS_RANGE=3-22

#period in days   delay in minutes   job-identifier   command
1                 5                  cron.daily      nice run-parts /etc/cron.daily
7                 25                 cron.weekly      nice run-parts /etc/cron.weekly
@monthly          45                 cron.monthly     nice run-parts /etc/cron.monthly
```


National Institute of Informatics
Copyright (C) 2012 National Institute of Informatics, All rights reserved.

メモとしてお使いください

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

メモとしてお使いください

Copyright (C) 2012 National Institute of Informatics, All rights reserved.

Copyright (C) 2012 National Institute of Informatics, All rights reserved.