

Diskless Cluster 構築入門

同志社大学 大学院 工学研究科

谷村 勇輔

tanisuke@mikilab.doshisha.ac.jp

はじめに

PC Cluster は、汎用的な PC を多数集めて構築される並列計算機のことです。PC Cluster はコストパフォーマンスが高いことから、最近では数多く作られるようになりました。それに伴い、Cluster を構築・運用する際の技術も充実しつつあり、用途に合わせた Cluster を構築することが可能になりました。ここでは PC Cluster の中でも、プロセッサ数が比較的小規模（8~32）な場合においてしばしば用いられる Cluster のモデルの 1 つ、Diskless Cluster について述べたいと思います。

Diskless Cluster とは

PC Cluster は一般に、サーバマシンと呼ばれる Cluster 全体を管理するためのマシンと 計算マシンと呼ばれる実際に計算を行うマシンの 2 種類からなります。およそサーバマシンは 1 台であり、計算マシンは複数台用意されることとなります。図 3.1 に示すような Cluster を想像して下さい。Diskless Cluster では、この計算マシンのディスク（主に HDD）がありません。必要なファイルは NFS（Network File System）などを利用して、サーバマシン上のファイルをネットワーク経由で参照します。これらの特徴より Diskless Cluster の利点は、

- HDD がいないために障害が発生しにくく、かつ低価格で構築できる。あるいは、他の OS が動作しているマシンを Cluster の計算マシンとして使用できる。
- カーネルやファイルシステムをサーバマシンが持つことにより、一元管理しやすい。

であるといえます。逆に欠点としては次のことが考えられます。

- ネットワークの負荷が高くなるために、性能の低下が起きる。
- ローカルに HDD をもたないので swap 領域をとれない。
- 実行時にファイル入出力を頻繁に行うアプリケーションでは、性能が著しく低下する。

実際に大幅な性能低下が起きる可能性があるのは、全計算マシンからネットワーク経由でファイルアクセスがある時です。実行する並列アプリケーションが、頻繁にファイルアクセスしないのであれば、それほど性能の低下を招くことはないと思います。ただし並列アプリケーション実行時には、各計算マシンにおいてネットワーク経由で実行モジュールを読み出さなければなりません。これは、アプリケーションが立ち上がるまでの遅延を引き起こします。また、計算マシンを一斉起動する場合に

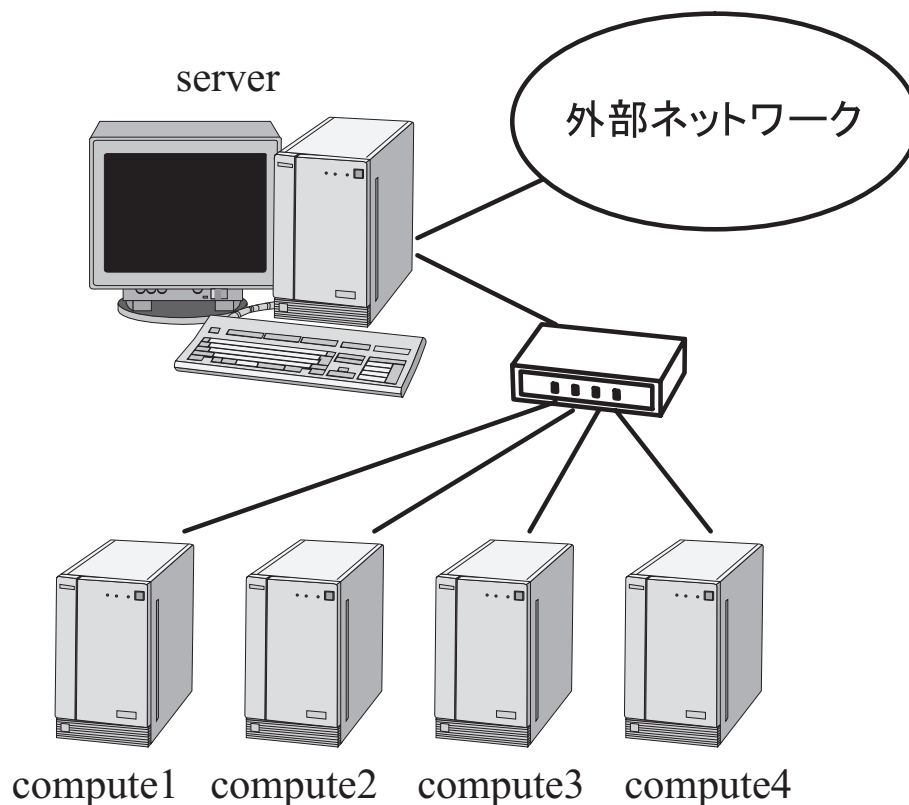


図 3.1: Cluster の概観

もネットワークやサーバマシンに負荷が集中する可能性があります。こうした影響は、Cluster の規模が比較的大きくなった場合に問題になってきます。

Diskless Cluster のブートの仕組み

Diskless Cluster は、計算マシンのブートをどのように行うかによって次の 3 つに分類できます。

- FD を用いる
 - (1) FD にカーネルを書き込んでおく
 - (2) サーバマシン上にあるカーネルを読み出す
- ブート ROM を用いる
 - (3) サーバマシン上にあるカーネルを読み出す

FD を用いる方法では、カーネルを FD に書き込むかサーバマシン上に置いておくかの 2 通りの選択肢があります。いずれの場合もブート後は FD を使用しないため、ブートの手順だけが異なります。(1) の場合には次のような手順となります。

電源 ON (FD より) カーネルのロード ネットワークの設定 ルートファイルシステムのマウント (ネットワーク経由) 各種サービス開始

これを見れば分かるように、(1) は FD で動く Linux (<http://saturn.fokus.gmd.de/linux/linux-distrib-small.html>) を発展させた形であるといえます。FD で動く Linux との違いは、ルー

トファイルシステムをサーバマシン上に持つことです。(2)では、サーバマシン上にあるカーネルをネットワーク経由で読み出します。FDからロードされて最初に動くプログラム(起動ROMバイナリ)が、ネットワークの設定を行います。この種のプログラムとしてはNetbootやEtherboot、NILOなどが挙げられます。これらについての詳細は後述します。以下に(2)のブート手順を示します。

電源 ON ネットワーク設定 カーネルのロード(ネットワーク経由) ルートファイルシステムのマウント(ネットワーク経由) (改めて)ネットワークの設定 各種サービス開始

(1)と(2)の比較では、カーネルの変更があっても管理しやすいこと、カーネルイメージのサイズをあまり気にしなくて良いことなどの理由から、(1)よりも(2)の方が扱い易いと思います。(3)では、(2)で用いた起動ROMバイナリをBIOSに組み込んでおきます。これにより計算マシンではFDDすらも必要でなくなります。ブートROMへの書き込みにはROMライターが必要ですが、On Board書き込みをすることも可能です。ただし、これらの作業には常にBIOS破壊の危険が伴います。ROMライターを用いてあらかじめバックアップをとっておくこと、それなりに覚悟を決めて各自の責任の元で作業を行うようお願いいたします。最近の高機能のNICには、こうした起動ROMバイナリと同じような機能(BOOTPやDHCPなどの機能)を付加した製品もあります。そのようなNICを利用できる場合には、さらに構築の手間を省くことができると思います。ここで紹介した中から、今回は(2)を用いてDiskless Clusterを構築する方法について説明します。また(3)の手順を用いる場合にも、(2)で十分に動作確認をした後にROMへの書き込みを行うのが良いかと思えます。

構築手順

流れ

本章ではDiskless Clusterの構築の手順をおおまかに示します。OSにはDebian/GNU Linuxを用いるため、各種サービスの設定はDebianのものを記述します。その他のディストリビューションやOSを用いる場合には、適宜読みかえて下さい。以降の節では、次のような手順でDiskless Clusterを構築していきます。

- サーバマシンの構築

- Linuxのインストール

- カーネルの再構築

- BOOTPサーバの設定

- TFTPサーバの設定

- NFSサーバの設定

- 計算マシンの構築

- 計算マシン用ディレクトリの作成

- 計算マシン用の設定ファイル、起動・終了スクリプトの作成

- 計算マシン用カーネルの作成

- 計算マシン用起動ディスクの作成

- Diskless Clusterの動作確認

Cluster の構成としてはサーバマシンが 1 台，計算マシンが複数台あるものとします．図 3.1 に示すようにサーバマシンの名称を server，計算マシンの名称を compute1，compute2 のように付けていきます．各マシンの IP アドレスは以下に示す通りです．参考までに，表 3.1 に使用するマシンスペックを示しておきます．

```
server   : 202.23.xx.xx (グローバル)
          192.168.1.1  (Cluster 内)

compute1 : 192.168.1.11
compute1 : 192.168.1.12
```

表 3.1: 各マシンスペック

CPU	Celeron (Mendocino)
Clock	466MHz
Memory	64MB
NIC	Intel EtherExpress Pro/100+
OS	Debian GNU / Linux 2.2 (potato)
kernel version	2.2.15

サーバマシンの準備

まず最初に，Diskless Cluster のサーバマシンを構築します．このサーバマシンは通常の PC Cluster のサーバマシンに，ディスクレスである計算マシンのためのサービスを付加したものになります．以下の説明では，既にサーバマシンに Linux がインストールされ，コンパイラやエディタなどの開発環境が整っているものとします．

カーネルの構築

計算マシンのネットワーク設定を一元管理するために BOOTP を用います．計算マシンのファイルシステムをネットワーク経由で提供するためには NFS を用います．これらを利用するために，以下に示す項目にチェックをしてカーネルを構築する必要があります．

```
CONFIG_INET_RARP=y # RARP をサポートする
CONFIG_NFS_FS=y   # NFS をサポートする
CONFIG_NFSD=y     # NFS サーバをサポートする
```

BOOTP サーバの設定

BOOTP は「自ら OS を持たない端末がサーバに接続するためのプロトコル」です．サーバマシン上の/etc/bootptab に MAC アドレスとそれに対応する IP アドレスを記述しておくことで，静的に IP アドレスを割り当てることができます．DHCP を利用することも可能ですが，今回は BOOTP を利用します．Debian において BOOTP サーバは bootpd パッケージに含まれています．BOOTP の設定は，/etc/bootptab に計算マシンの NIC の MAC アドレスとそれに割り当てる IP アドレスを記述します．以下に設定例を示します．

/etc/bootptab の記述

```
.default:\
    :sm=255.255.255.0:\
    :gw=192.168.1.1:\
    :ds=202.23.xx.xx:\
    :ht=ethernet:\
    :bf=bootImage:\
    :hn:to=-18000:

compute1:ha=00A0A0A0A0A1:ip=192.168.1.11:tc=.default:hd=/tftpboot/192.168.1.1
compute2:ha=00A0A0A0A0A2:ip=192.168.1.12:tc=.default:hd=/tftpboot/192.168.1.1
```

記述の主な意味

sm: サブネットマスク
 gw: ゲートウェイ
 ds: DNS サーバ
 bf: ブートファイル
 ha: IP アドレスを割り当てる NIC の MAC アドレス
 ip: 割り当てる IP アドレス
 tc: グローバル・テンプレートの読み込み
 hd: ブートファイルの置かれているディレクトリ

設定後に bootpd を起動する必要がありますが、一般的に bootpd は inetd 経由で起動されるようになっています。/etc/inetd.conf に以下の記述があるかどうかを確認し、なければ追加します。i オプションは inetd モードであることの指定、t オプションは BOOTP パケットを待つタイムアウト値を設定しています。

/etc/inetd.conf の記述

```
bootps dgram udp wait root /usr/sbin/tcpd /usr/sbin/bootpd -i -t 120
```

TFTP サーバの設定

TFTP (Trivial File Transfer Protocol) は、計算マシンが BOOTP によりネットワーク設定を行った後に、サーバマシン上にあるカーネルをダウンロードするために用います。TFTP は FTP と同様なファイル転送プロトコルですが、非常に単純化されています。Debian において TFTP サーバは tftpd パッケージに含まれています。一般的に tftpd も bootpd と同様、inetd 経由で起動されるようになっています。/etc/inetd.conf に以下の記述があるかどうかを確認し、なければ追加します。引数には TFTP によるアクセスを許可するディレクトリを指定します。

/etc/inetd.conf の記述

```
tftp dgram udp wait nobody /usr/sbin/tcpd /usr/sbin/in.tftpd /tftpboot
```

NFS サーバの設定

Diskless Cluster では、各計算マシンが NFS を利用してサーバ上のファイルシステムをマウントします。Debian において NFS サーバは nfs-server パッケージに含まれています。インストール後、NFS サーバの設定ファイル (/etc/exports) の中に、各計算マシンにどのディレクトリをマウントさせるかを記述しておきます。これは、計算マシン間でどの程度ファイルを共有するかにも依存します。今回は、以下のように設定ファイルを記述します。ファイル中の no_root_squash は、root ユーザが NFS のク

クライアントマシン上においても root 権限を持つことができることを指定しています。さらに、root、home ディレクトリについては NFS 経由の書き込み権限を与えている点に注意して下さい。また、書き込み権限は必要なディレクトリに対してのみ与えるべきです。

/etc/export の記述

```
/usr          192.168.1.0/255.255.255.0(ro)
/bin          192.168.1.0/255.255.255.0(ro)
/lib          192.168.1.0/255.255.255.0(ro)
/sbin        192.168.1.0/255.255.255.0(ro)
/root         192.168.1.0/255.255.255.0(rw,no_root_squash)
/home         192.168.1.0/255.255.255.0(rw,no_root_squash)
/tftpboot/template 192.168.1.0/255.255.255.0(ro)
/tftpboot/192.168.1.11 192.168.1.11/255.255.255.0(rw,no_root_squash)
/tftpboot/192.168.1.12 192.168.1.12/255.255.255.0(rw,no_root_squash)
```

計算マシン用のディレクトリの作成，およびシステムの設定

各計算マシン用のディレクトリをサーバマシン上に構築します。Cluster の管理を容易にするために、共有できるファイルはできる限り共有し、特定マシンのファイルを少なくするようにします。Cluster Computing の分野ではこうした Cluster の一元管理のように、相互接続されたコンピュータの集合を単一のリソースとして扱うことを SSI (Single System Image) と呼んでいます。管理者だけでなく、一般ユーザにとっても SSI が提供されることは、喜ばしいことだと思います。そこで今回構築する Diskless Cluster が少しでも SSI を実現できるように、ここでの作業を進めていきたいと思います。しかし、実際にはこの作業を徹底的に行おうとすると、Linux のファイル構成を大きく変更する必要が出てきてしまいます。構築の手間を考えると、ある程度のファイルは共有し、いくつかのファイルは重複させざるを得ないかもしれません。

主なディレクトリの作成

今、Debian GNU/Linux で構築されたシステムを見ると次のようになっています。

```
$ ls -l /
drwxr-xr-x  2 root   root     2048 Jul 25 21:41 bin
drwxr-xr-x  2 root   root     1024 Aug 11 15:55 boot
drwxr-xr-x  2 root   root     1024 May 26 21:11 cdrom
drwxr-xr-x  3 root   root    18432 Aug 25 14:57 dev
drwxr-xr-x 51 root   root     3072 Aug 25 14:57 etc
drwxr-xr-x  2 root   root     1024 May 26 21:11 floppy
drwxrwsr-x 11 root   staff    1024 Jun 29 15:45 home
drwxr-xr-x  2 root   root     1024 May 26 21:11 initrd
drwxr-xr-x  4 root   root     4096 Jul 25 21:41 lib
drwxr-xr-x  2 root   root    12288 May 27 05:06 lost+found
drwxr-xr-x  2 root   root     1024 Jun 29 16:01 mnt
dr-xr-xr-x 38 root   root         0 Aug 25  2000 proc
drwxr--r--  5 root   root     1024 Aug 23 11:40 root
drwxr-xr-x  2 root   root     2048 Jul 25 21:43 sbin
drwxrwxrwt  3 root   root     1024 Aug 25 15:27 tmp
drwxr-xr-x 14 root   root     1024 May 26 21:11 usr
drwxr-xr-x 14 root   root     1024 May 29 09:56 var
lrwxrwxrwx  1 root   root         20 May 26 21:53 vmlinuz -> /boot/vmlinuz-2.2.15
```

これらの中で、Diskless Cluster の計算マシンに必要なもの、またマシン間で共有すべきもの、そうでないものと分類を行います。分類を行った結果は以下に示す通りです。

```
計算マシンで必要ない      : cdrom floppy initrd lost+found mnt vmlinuz
Cluster 全体で共有        : bin lib sbin usr root home
個々に用意                : boot dev etc var
個々にディレクトリだけ用意 : proc tmp
```

Diskless Cluster ではサーバマシン上に、各計算マシン用のディレクトリを作成しますが、ここでは /tftpboot 以下にディレクトリを作成することにします。前節で示した NFS の設定ファイルからも分かるように、計算マシンは /tftpboot/IP_ Address をルートファイルシステムとしてマウントします。Cluster 全体で共有するファイルも NFS を利用して共有するため、マウント・ポイントとなるディレクトリをあらかじめ準備しておく必要があります。

```
# mkdir -p /tftpboot/192.168.1.11
# cd /tftpboot/192.168.1.11
# mkdir bin lib sbin usr root home
```

このように非常に多くのファイルを、サーバマシンから参照します。ただし、各計算マシンを起動するために、最小限コピーしておかなければならないファイルがあります。これらは Linux の起動の順序に依存しますが、/sbin/init は最初に実行されるプログラムですので必ず必要になります。以下に、私が用意したバイナリの一覧を示します。これらを /tftpboot/IP_ Address/sbin、/tftpboot/IP_ Address/bin にコピーしておきます。

```
/sbin/init
/sbin/update
/sbin/getty

/bin/sh
/bin/mount
/bin/umount
/bin/uname
/bin/grep
/bin/rm
```

また、上記バイナリファイルの多くは、デフォルトでダイナミックリンクされています。/lib のいくつかのファイルを、同じように /tftpboot/IP_ Address/lib にコピーしておきます。この時、ldd コマンドを使用することで、必要なライブラリを調べることができます。例えば、下記のようにコマンドを実行します。これより mount には libc.so.6 と ld-linux.so.2 が必要であることが分かります。

```
$ ldd mount
libc.so.6 => /lib/libc.so.6 (0x40016000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

私の場合には、以下に示すライブラリが必要になりました。

```

ld-2.1.3.so
ld-linux.so.2 -> ld-2.1.3.so
libc-2.1.3.so
libc.so.6 -> libc-2.1.3.so
libdl-2.1.3.so
libdl.so.1 -> libdl.so.1.9.11
libdl.so.1.9.11
libdl.so.2 -> libdl-2.1.3.so
libncurses.so.5 -> libncurses.so.5.0
libncurses.so.5.0

```

次に個々に用意するディレクトリを作成します。dev は各マシンで別々に使われるので、サーバマシンの dev を丸ごとコピーします。ただし計算マシンの Root device を dev/nfsroot にするため、そのファイルがあるかどうかを確認します。なければ作成し、その後にコピーするのが良いでしょう。etc や var については個々に用意すべきファイルはあるものの、その多くを共有することができます。そこで、いったん/tftpboot/template 以下にコピーしておきます。boot ディレクトリはカーネルの System.map を置くだけです。System.map ファイルは、計算マシン用のカーネルを構築する時に作られます。ここでは、とりあえずディレクトリだけ作成しておきます。tmp や proc はディレクトリさえ提供しておけば問題ありません。

```

# cd /dev
# mknod nfsroot b 0 255
# cd /tftpboot/192.168.1.11
# tar cCf / - dev | tar xCfp /tftpboot/192.168.1.11 -
# mkdit template
# tar cCf / - etc var | tar xCfp /tftpboot/template -
# mkdir boot tmp proc

```

etc ディレクトリの作成と設定

現時点で etc ディレクトリは、/tftpboot/template/etc にあります。各計算マシン用の etc ディレクトリを/tftpboot/IP_ Address 以下に作成し、共有するファイルは/tftpboot/template/etc からのシンボリックリンクにします。注意すべきことは、/ディレクトリが/tftpboot/IP_ Address であると想定しなければならぬことです。すなわち、以下のようにコマンドを実行します。

```

# cd /tftpboot/192.168.1.11/etc
# ln -s ../template/etc/hosts.allow .

あるいは

# cd /tftpboot/192.168.1.11/etc
# ln -s /template/etc/hosts.allow .

```

私の場合は、以下に示すファイル、ディレクトリをシンボリックリンクとして用意しました。

```
alternatives cron.d cron.daily cron.weekly group hostname hosts.allow
hosts.deny hosts.equiv inetd.conf ld.so.conf login.defs modules
networks nsswitch.conf pam.conf pam.d passwd protocols rc0.d rc1.d
rc2.d rc3.d rc4.d rc5.d rc6.d securetty serial.conf services
syslog.conf terminfo
```

また次のファイル、ディレクトリを各計算マシンごとに用意します。これらはカーネルが読み込まれる時に必要なファイルであったり、マシンごとに設定が異なるファイルであったりします。

```
default fstab hosts init.d inittab network rcS.d resolv.conf
```

計算マシンの設定に関して、最も注意すべき部分は起動と終了のスキプトの設定です。これは、NFSによるマウントのタイミングを考えて設定を行って下さい。設定方法は各システムに依存していますので、以降で説明する事項を適宜読みかえて下さい。

起動の設定

起動では、`etc/rcS.d`ディレクトリ以下のスキプトのうち、ファイル名の先頭の番号が小さいものから順に実行されていきます。今はサーバマシンのものをコピーしただけの状態ですから、計算マシンに必要なでないスキプトを実行しないように修正を加えます。ただし、`etc/rcS.d`以下のスキプトは全て `etc/init.d`以下にあるスキプトのシンボリックリンクとなっていることに注意して下さい。

`S30checkfs.sh` は `fsck` を実行するスキプトです。すなわち、ディスクレスのマシンには必要ありません。ファイル自体を消去しても構いませんが、とりあえずファイルの先頭に `exit` を記述して、すぐにスキプトを抜けるように設定しておきます。

`S30checkfs.sh` の記述

```
#
# checkfs.sh    Check all filesystems.
#
# Version:      @(#)checkfs  2.78  13-Nov-1999  miquels@cistron.nl
#

exit 0    # この1行を追加

. /etc/default/rcS
```

`S35mountall.sh` は、元々ローカルに存在するファイルシステムをマウントするためのスキプトです。しかし、ディスクレスマシンの場合には早い段階でファイルシステムをマウントしなければ、以下の起動スキプトがうまく動作してくれません。そこで、ファイル中の以下の部分を修正します。mount コマンドの `a` オプションは、`etc/fstab` に指定されているファイルシステムを全てマウントするという意味です。

S35mountall.sh の記述

```
- mount -avt nonfs,noproc,nosmbfs
+ # mount -avt nonfs,noproc,nosmbfs
+ mount -av
```

この時に参照する各計算マシンの `etc/fstab` は次のように記述します。これは IP アドレスが 192.168.1.11 となる計算マシンの `fstab` です。

fstab の記述

```
# <file system> <mount point> <type> <options> <dump> <pass>
192.168.1.1:/tftpboot/192.168.1.11 /      nfs  rw  0  0
192.168.1.1:/usr                /usr   nfs  ro  0  0
192.168.1.1:/lib                /lib   nfs  ro  0  0
192.168.1.1:/bin                /bin   nfs  ro  0  0
192.168.1.1:/sbin               /sbin  nfs  ro  0  0
192.168.1.1:/root               /root  nfs  rw  0  0
192.168.1.1:/home               /home  nfs  rw  0  0
192.168.1.1:/tftpboot/template /template nfs  ro  0  0
none                            /proc  proc defaults
```

S40hostname.sh はホスト名を設定するスクリプトです。しかし、ここではホスト名を BOOTP から与えたいので、以下のように `hostname` コマンドをコメントアウトしておきます。また、`/tftpboot/IP-Address/etc/hostname` はテンプレートへのシンボリックリンクですが、中身を空にしておきます。

S40hostname.sh の記述

```
# hostname --file /etc/hostname
```

S40network はネットワークを設定するスクリプトです。今回、これら起動スクリプトは各マシンごとに用意するため、ここに各計算マシンの IP アドレスなどを記述できます。しかし、せっかく BOOTP サーバの設定をしているのですから、BOOTP を使って IP アドレスを取得しましょう。もちろん、こちらの方が多少管理の手間を省けます。S40network を以下のように記述します。bootpc コマンドは bootpc パッケージに入っています。

/etc/init.d/network の記述

```
#!/bin/sh

/sbin/bootpc --dev eth0 --server 192.168.1.1 --timeoutwait 20
ifconfig lo 127.0.0.1
route add -net 127.0.0.0 netmask 255.0.0.0 lo

exit 0
```

以上を設定すれば、なんとか計算マシンを起動することができるでしょう。もし何らかのエラーが出る場合は、スクリプトの実行順序を変えてみたり、いくつかのコマンドをコメントアウトしたりして解決を図って下さい。

終了の設定

既に何度も繰り返してきたように、各計算マシンは HDD をもっていません。マシンを終了させるためにいきなり電源を落としたり、リセットをかけたりしてもそれほど問題がないかもしれませんが。しかし、リモートから Cluster を管理するという観点では、手順を踏んだ終了を行いたいものです。一般にマシンを終了させるコマンドには、halt と reboot の 2 種類があります。etc/inittab の記述を見ると、Debian では halt のランレベルは 0、reboot のランレベルは 6 であることが分かります。つまり halt に関しては etc/rc0.d、reboot に関しては etc/rc6.d 以下のスクリプトが重要であることが分かります。

/etc/inittab の記述

```
# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.
```

終了に関して最も注意すべき点は、先にファイルシステムをアンマウントしてしまうことにより、halt や reboot などのコマンドが見つからず、途中でシステムがハングしてしまうことです。これを回避するために、ファイルシステムのアンマウントに関するスクリプトを最後尾にもってきます。S90halt スクリプトの後ろにもってきたスクリプトは、実際に実行されることはありません。

/etc/rc0.d 以下のスクリプトの変更

```
[ 変更前 ]
S31umountnfs.sh
S40umountfs
S90halt

[ 変更後 ]
S90halt
S93umountnfs.sh
S95umountfs
```

/etc/rc6.d 以下のスクリプトの変更

```
[ 変更前 ]
S31umountnfs.sh
S40umountfs
S90reboot

[ 変更後 ]
S90reboot
S93umountnfs.sh
S95umountfs
```

上記のように変更を加えると、ルートファイルシステムがアンマウントされたままシステムが終了してしまいます。さらに、終了時において NFS の同期がしっかり取れているかどうかといった問題も

あります。これらに対処してもっとうまくやるには、上記のスクリプトの中身を修正する必要があると思います。私の方では試していませんが、余裕のある方は試してみてください。

var ディレクトリの作成

var ディレクトリも etc ディレクトリと同様に、共有するファイルは/tftpboot/template/var へのシンボリックリンクとして作成します。私の場合には、以下に示すディレクトリを各計算マシンごとに用意し、その他を共有することにしました。

```
var/run var/lock var/spool var/log
```

以上で、計算マシン用のディレクトリの作成はおおよそ終了です。2 台目以降の計算マシンの構築は、次のようにサーバマシン上で 1 代目のマシンのディレクトリを丸ごとコピーし、各マシンに依存するファイルだけを修正すれば良いわけです。

```
# cp -a /tftpboot/192.168.11 /tftpboot/192.168.1.12
```

du コマンドを使って、各計算マシン用のディレクトリの合計サイズ (KB 単位) を表示させてみます。多くのファイルを共有したことにより、非常に小さなサイズとなっているのが分かります。もちろん、工夫を加えることでさらにサイズを小さくできると思います。

```
# du -sk /tftpboot/*
4058  /tftpboot/192.168.1.11
3267  /tftpboot/192.168.1.12
533   /tftpboot/bootImage
187048 /tftpboot/template
```

ユーザの共有

通常の PC Cluster では、NIS (Network Information Service) を利用してユーザの一元管理を行います。しかし、Diskless Cluster では計算マシンのファイルがサーバ上に存在するため、わざわざ NIS を使う必要がありません。今、/tftpboot/template/etc/passwd にパスワードファイルがコピーされています。そこで/etc/passwd をそれに対するシンボリックリンクとします。/etc/group も同様にシンボリックリンクとします。これより Cluster 全体でユーザ情報を共有できます。

```
# cd /etc
# rm passwd group
# ln -s /tftpboot/template/etc/passwd .
# ln -s /tftpboot/template/etc/group .
```

計算マシン用のカーネルの作成

サーバマシン上で、計算マシン用のカーネルを作成します。カーネルのコンフィギュレーションでは最低限、以下の項目をチェックする必要があります。またこの際にモジュールではなく、カーネルに組み込まれるようにして下さい。

```
CONFIG_IP_PNP=y          # IP kernel level autoconfiguration をサポートする
CONFIG_IP_PNP_BOOTP=y    # BOOTP をサポートする
CONFIG_NFS_FS=y          # NFS をサポートする
CONFIG_ROOT_NFS=y        # Root on NFS を有効にする
CONFIG_EEEXRESS_PRO100=y # 使用する NIC ドライバをサポートする
```

以下のようにコマンドを実行し、カーネルを構築します。

```
# cd /usr/src/linux
# make mrproper
# make menuconfig
# make dep clean
# make
# make bzImage
# make modules
# make modules_install
```

構築後、Netboot に付属している `mknbi-linux` コマンドを使ってカーネルイメージからブートファイルを作成します。作成したブートファイルは、`/etc/bootptab` ファイル中で指定したディレクトリにコピーしておきます。また、前節で作成した計算マシン用の `boot` ディレクトリに、作成された `System.map` をコピーするのを忘れないで下さい。

```
# cd /usr/src/linux/boot/
# mknbi-linux -d rom --i rom -k bzImage -o bootImage
# cp bootImage /tftpboot/
# cp /usr/src/linux/System.map /tftpboot/192.168.1.1/boot/
```

ブート

前章で述べたように、計算マシンのブートには Linux のカーネルがロードされる前にネットワーク設定を行うプログラム（起動 ROM バイナリ）を用います。起動 ROM バイナリを提供するソフトウェアとして、Netboot や Etherboot、NILO が代表的です。Netboot は主に Linux、MS-DOS をサポートし、Crynwr Packet Driver Collection によって提供されている NIC のドライバを多数利用しています。Netboot はサポートする NIC 数は多いのですが、ROM イメージのサイズが大きくなってしまふこと、`autoprobe` の機能がないことなどの欠点があります。Etherboot はそれらの欠点を解消したものといたえますが、サポートする NIC 数は Netboot ほどではありません。Etherboot の欠点は NIC の各チップセットに対して、1 つ 1 つドライバを書かなければいけないこと、さらにこれらのドライバは Linux のドライバを流用できないことです。NILO (Network Interface LOader) は Netboot、Etherboot を引き継ぎ、これらの欠点を解消した新しいネットワーク・ブータであるといえます。

今回は、これらの中で Netboot を利用して起動 ROM バイナリを作成する方法を紹介します。Netboot を選択した理由は、現在の Debian のリリースにおいて Netboot だけが `.deb` ファイルとして提供されており、インストールの手間が省けるからです。しかし、もし余裕があるのならば NILO も試してみてください。

インストール手順について説明します。まず `netboot` パッケージをインストールします。次に `makerom` コマンドで NIC のドライバを組み込んだ起動 ROM バイナリを作成します。`makerom` コマンドを実行するといくつかの質問がなされるので、どの NIC のドライバを組み込むかはそこで指定してやり

ます．するとコマンドを実行したディレクトリに，image.flo と image.rom の 2 つのファイルができます．image.flo は FD 用のバイナリ，image.rom はブート ROM 用のバイナリです．後は，あらかじめフォーマットした FD に image.flo を書き込んで起動ディスクを作成します．

```
# apt-get install netboot
# makerom
# superformat /dev/fd0
# dd if=image.flo of=/dev/fd0
```

動作確認

それでは実際に FDD に起動ディスクを挿し込み，計算マシンが正常に起動するか確認してみましょう．正常に起動した時，サーバマシン上の /var/log/syslog に記録されたログを以下に示しておきます．

サーバマシンに出力されたログ

```
server bootpd[749]: connect from 0.0.0.0
server bootpd[749]: version 2.4.3
server in.tftpd[750]: connect from 192.168.1.11
server tftpd[751]: tftpd: trying to get file: /tftpboot/192.168.1.11/bootImage
server mountd[184]: NFS mount of /tftpboot/192.168.1.11 attempted from 192.168.1.11
server mountd[184]: /tftpboot/192.168.1.11 has been mounted by 192.168.1.11
server mountd[184]: NFS mount of /usr attempted from 192.168.1.11
server mountd[184]: /usr has been mounted by 192.168.1.11
server mountd[184]: NFS mount of /lib attempted from 192.168.1.11
server mountd[184]: /lib has been mounted by 192.168.1.11
server mountd[184]: NFS mount of /bin attempted from 192.168.1.11
server mountd[184]: /bin has been mounted by 192.168.1.11
server mountd[184]: NFS mount of /sbin attempted from 192.168.1.11
server mountd[184]: /sbin has been mounted by 192.168.1.11
server mountd[184]: NFS mount of /root attempted from 192.168.1.11
server mountd[184]: /root has been mounted by 192.168.1.11
server mountd[184]: NFS mount of /home attempted from 192.168.1.11
server mountd[184]: /home has been mounted by 192.168.1.11
server mountd[184]: NFS mount of /tftpboot/template attempted from 192.168.1.11
server mountd[184]: /tftpboot/template has been mounted by 192.168.1.11
```

もし計算マシンがうまく起動しない場合には，次のことを順に確認してください．

(1) BOOTP サーバが応答しない

- /usr/sbin/bootptest コマンドを利用して，サーバの応答性を確認する．
- bootpd デーモンを -d4 などのデバッグオプションを付けて起動し，ログを確認する．

(2) TFTP を利用してカーネルイメージがダウンロードできない

- /var/log/syslog に出力されるログを見る．多くは /etc/bootptab に記述したブートファイルの置き場所を間違えて指定しているか，TFTP アクセスを可能にするディレクトリを間違えているか指定しているかが原因である．

(3) 途中でカーネルパニックを引き起こす

- カーネル，そしてブートファイルが適切に作られているか確認する．
- 計算マシン用のディレクトリが正しく作成されているか確認する．

(4) ルートファイルシステムのマウントに失敗する

- 適切にカーネルが作られているか確認する。
- /etc/exports の記述が適切であるか確認する。
- /tftpboot/IP-Address/etc/fstab の記述が適切であるか確認する。
- /var/log/syslog に出力されるログを見る。

(5) (4) 以降でエラーが起こる

- 起動スクリプトが適切であるか確認する。
- 起動に必要な最小限のファイルが計算マシン上に準備してあるか確認する。

本章では Diskless Cluster の構築について説明してきました。繰り返しますが、設定ファイルに関する部分はシステムによって異なる上、今後も大きく変更される可能性があります。その OS についてある程度の知識や情報がなければ、なかなか難しいかもしれません。しかし、後述する参考文献に示しているように、ディスクレスマシンや Linux についてのドキュメントは Web 上に多数あります。さらに次章では、より優れた Cluster を構築するためのヒントを示します。これらを利用して、私が紹介したものよりも「管理がし易く、安定して動き、性能の良い Cluster」を皆様が構築できることを願っています。

より高機能・高性能に

本章では、構築した Diskless Cluster に対して「管理の容易さ」「性能向上」を図るためのヒントを紹介します。構築される Cluster の数が増加するにしたがって、こうした情報が数多く公開されるようになっていきます。

リモートの電源管理

リモートの電源管理は、特に Diskless Cluster に限った話ではありません。しかしこれまで説明してきた Diskless Cluster では、一元管理を 1 つの目標にしてきました。そこで、計算マシンの電源管理もサーバマシンから行いたいと思います。電源管理には ON/OFF がありますが、それぞれについて以下で説明を行います。

電源 ON

リモートから電源を ON するには Wake On LAN という仕組みを用います。Wake On LAN とはネットワーク経由でマシンを起動する仕組みであり、これを用いることで Cluster 内の計算マシンを一斉に起動することが可能になります。Wake On LAN を行うには、マザーボードと NIC、電源が Wake On LAN に対応している必要があります。そして BIOS で Wake On LAN を有効にしておく必要があります。

Wake On LAN では、リモートから MagicPacket と呼ばれる起動用パケットを NIC に送りつけます。NIC が MagicPacket を受け取ると、電源が ON になりマシンが起動されるわけです。詳しくは MagicPacket のドキュメントなどを参照して下さい。MagicPacket は、FF を 6 バイト繰り返した後に、MAC アドレスを 16 回繰り返す計 102 バイトが含まれるパケットです。以下に神戸大学の児玉さんが作成された MagicPacket を送信するスクリプトを示します。児玉さんのページ (<http://www.math.kobe-u.ac.jp/~kodama/index.html>) には、この他に ping プログラムに修正を加えて MagicPacket を送信する方法などが紹介されています。

ruby を利用したスクリプト

```

#!/bin/bash
# Wake On LAN. K.Kodama 1999-11-29 2000-05-13
# line format of hw-addresses: ip hostname hw-address
#   e.g. 192.168.1.10 host10 0A:0A:0A:0A:0A:0A

PROG='basename $0'

function HELP(){ cat<< EOF
Wake up on LAN
Usage1: $PROG [-b broadcast] -a          wake up all
Usage2: $PROG [-b broadcast] host       wake up the host
Usage3: $PROG [-b broadcast] -s        read MAC list from stdin
EOF
}

function SendMagic(){
/usr/local/bin/ruby -e 'require "socket"
if ARGV.size>=1; adr=ARGV[0]; ARGV.clear; else adr=""; end
s=UDPSocket.open(); s.setsockopt(Socket::SOL_SOCKET,Socket::SO_BROADCAST,1)
while gets
  l=$_sub(/^s+/, "").split(/\s+/)
  ip=l[0]; name=l[1]; hw=l[2]; hs=hw.split(/:/).pack("H*H*H*H*H*H*")
  if adr==""; # Set adr as broadcast. A:1--127, B:128--191, C:192--223
    ips=ip.split(/\./);c=ips[0].to_i
    if c<=127; ips[1]="255";end
    if c<=191; ips[2]="255";end
    if c<=223; ips[3]="255";adr=ips.join(".");else adr="";end
  end
  2.times{ s.send((0xff.chr)*6+hs*16,0,adr,"discard") }
end' @$
}

if [ "$PROG" = "wakeup1" ];then macfile=~mac-list1;broadcast=""
elif [ "$PROG" = "wakeup2" ];then macfile=~mac-list2;broadcast=""
else macfile=~arp-sample;broadcast=""
fi
if [ "$1" = "-b" ];then broadcast=$2; shift 2;fi
if [ "$1" = "-h" -o "$1" = "" ];then HELP
elif [ "$1" = "-a" ];then SendMagic $broadcast < $macfile
elif [ "$1" = "-s" ];then SendMagic $broadcast
elif grep $1 $macfile>/dev/null;then grep $1 $macfile|SendMagic $broadcast
else HELP
fi

```

上記のスクリプトは、次に示す MAC アドレスが記述されているファイル (~/arp-sample) を読み込み、MagicPacket をブロードキャストします。このスクリプトを用いることにより、計算マシンの一斉起動を実現することができます。

~/arp-sample の記述

```

192.168.1.11 compute1 00:A0:A0:A0:A0:A1
192.168.1.12 compute2 00:A0:A0:A0:A0:A2

```

電源 OFF

リモートから電源 OFF を行うには、自動電源 OFF の機能とリモートから電源 OFF の命令を実行できる機能が必要です。自動電源 OFF を行うには、まずマザーボードと電源が APM (Advanced Power Management) に対応している必要があります。そして BIOS で APM を有効にします。Linux のカーネルは APM BIOS 対応を有効にし、該当する項目をチェックして構築する必要があります。カーネル 2.2.15 においては、以下の設定で終了時に電源を OFF にできたという報告があります。

```
CONFIG_APM=y
# CONFIG_APM_DISABLE_BY_DEFAULT is not set
# CONFIG_APM_IGNORE_USER_SUSPEND is not set
# CONFIG_APM_DO_ENABLE is not set
# CONFIG_APM_CPU_IDLE is not set
# CONFIG_APM_DISPLAY_BLANK is not set
# CONFIG_APM_IGNORE_MULTIPLE_SUSPEND is not set
# CONFIG_APM_IGNORE_SUSPEND_BOUNCE is not set
# CONFIG_APM_RTC_IS_GMT is not set
# CONFIG_APM_ALLOW_INTS is not set
CONFIG_APM_REAL_MODE_POWER_OFF=y
```

しかしながら APM の自動電源 OFF については、これ以上正確なことを述べることはできません。上記の設定ではなく、lilo.conf でカーネルに apm のパラメータを与えたり、CONFIG_APM_REAL_MODE_POWER_OFF をセットせずにカーネルを作成した場合に、自動電源 OFF が動作したという例も報告されています。その他のドキュメントを参照しながら、各自の環境に合わせた設定を行うようにして下さい。APM ではなく、ACPI (Advanced Configuration and Power Interface) を利用するという方法もありますが、その時には開発バージョンのカーネルを使う必要があります。

電源 OFF の命令は、shutdown コマンドを rsh を利用して発行します。一般的な Cluster では rsh (あるいは ssh) が必須であるため、rsh の設定は既になされているはずで、これを利用するのが良いと思います。ただし、Debian では root 権限での rsh を許可していません。root 権限での rsh を有効にするには、計算マシンの rshd を以下のオプションを付けて起動するようにします。rshd も通常は inetd 経由で起動されるので、etc/inetd.conf を次のように修正します。

/etc/inetd.conf の中身

```
shell stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rshd -h
```

rsh を利用したリモートからのシャットダウンの例

```
# rsh compute1 shutdown -h now
```

(注意)

上記は古い rsh-server の設定例です。少なくとも Debian の rsh-server のバージョンが 0.10-7 以降で、上記の設定を試すと次のようなエラーが syslog に出力されます。すなわち、rshd には h オプションがなくなり、代わりに PAM で設定を行わなければならないということです。

```
server in.rshd[138]: connect from 192.168.1.11
server rshd[138]: -l and -h functionality has been moved to pam_rhosts_auth
                  in /etc/pam.conf
```

pam.conf および pam.d は共有されるので、/tftpboot/template/etc/pam.d/rsh ファイルに次の記述を加えてやります。これで root での rsh が使えるようになるはずですが、PAM についての詳細は、<http://riksun.riken.go.jp/archives/Linux/kernel.org/libs/pam/> などのドキュメントを参照して下さい。

```
##PAM-1.0
auth    sufficient    pam_rootok.so        # この 1 行を追加
auth    required      pam_rhosts_auth.so
auth    required      pam_nologin.so
auth    required      pam_env.so
account required      pam_unix_acct.so
session required      pam_unix_session.so
```

またこれらを利用するにあたり、root 権限での rsh がセキュリティ上、非常に危険であることは理解しておいて下さい。

Cluster NFS

Cluster NFS は Universal NFS Daemon サーバにパッチを当てて、同じルートファイルシステムを複数のディスクレス・クライアントで共有可能にするソフトウェアです。Cluster NFS では同じファイル名をもつファイルに対して、共有されるべきファイル、特定マシンのファイルを次のように区別することができます。

```
共有されるファイル：sample.txt
特定マシンのファイル：sample.txt$$IP=192.168.1.11$$
```

Cluster NFS は、通常の NFS とほとんど設定が同じであるため、容易に導入することができます。Cluster NFS をうまく使うことで、重複ファイルを減らし、スムーズにファイルの共有ができるのではないかと思います。

NFS Performance Benchmarks

Diskless Cluster では NFS が非常に重要な役割を果たしています。並列プログラムの性能に直接関わることはありませんが、実行モジュールのロードや各計算マシンの起動時など、ユーザが体感しやすい部分に NFS が関わってきます。小規模の Cluster の場合には気にならなくても、大規模になると NFS の性能の影響を大きく受けるようになるかもしれません。これを解決するために、Cluster 内の NFS の性能を測定し、NFS のパラメータを適切に設定する必要があるかもしれません。計算マシンの etc/fstab の記述において

```
192.168.1.1:/tftpboot/192.168.1.11 / nfs rsize=8192,wsiz=8192 0 0
```

のように rsize と wsize を指定してみましょう。値は 1024 の倍数で 16384 を越えない範囲で設定します。値を変更すると、Linux のカーネルと NIC の組み合わせによって NFS の性能が変わってきます。

す。以下に述べる簡単なベンチマークを行うことで、この値を適切に設定します。

NFS のベンチマークプログラムとしては、SPEC が開発した LADDIS ベンチマーク (Keith93) があります。これはシミュレートされたクライアントの負荷に対する NFS サーバの性能を測定します。しかし LADDIS は商用なので、実際にはなかなか利用できないでしょう。Debian では、nhfsstone という NFS のベンチマーク・ソフトウェアが用意されています。nhfsstone は nhfsstone パッケージに含まれています。nhfsstone プログラムを計算マシン上で実行すると、次のような結果が得られます。

```
# nhfsstone
nhfsstone: INVALID RUN, mix generated is off by 43.64%
op      want      got      calls     secs  msec/call  time %
null    0%      0.00%    0         0.00   0.00      0.00%
getattr 13%     14.27%   715       0.04   0.06      1.43%
setattr 1%      10.68%   535       0.00   0.00      0.09%
root    0%      0.00%    0         0.00   0.00      0.00%
lookup  34%     42.65%   2136      0.32   0.15      10.26%
readlink 8%      8.98%    450       0.40   0.90      13.03%
read    22%     2.57%    129       1.30   10.13     41.69%
wrcache 0%      0.00%    0         0.00   0.00      0.00%
write   15%     16.13%   808       0.42   0.52      13.42%
create  2%      1.05%    53        0.37   7.11     12.01%
remove  1%      1.05%    53        0.03   0.73      1.24%
rename  0%      0.00%    0         0.00   0.00      0.00%
link    0%      0.00%    0         0.00   0.00      0.00%
symlink 0%      0.00%    0         0.00   0.00      0.00%
mkdir   0%      0.00%    0         0.00   0.00      0.00%
rmdir   0%      0.00%    0         0.00   0.00      0.00%
readdir 3%      1.53%    77        0.20   2.59      6.37%
fsstat  1%      1.03%    52        0.01   0.25      0.41%
169 sec 5008 calls 29.63 calls/sec 0.62 msec/call
```

特別なベンチマーク・プログラムを用いなくても、簡単なコマンドを実行しておおまかな性能を計測することも可能です。次に示すのは、中身が 0 のバイト列で埋めつくされた 64MB のファイルの書き込みの性能のテストです。

```
$ time dd if=/dev/zero of=/tmp/testfile bs=16k count=4096
```

次に、このファイルを再度読み込みすることで、読み込み性能のテストを行います。

```
$ time dd if=/tmp/testfile of=/dev/null bs=16k
```

何度か平均をとって測定し、最も性能の良かった `rsize`、`wsize` を設定すれば良いわけです。

ローカルディスクの使用

これまで説明してきたように、Diskless Cluster ではファイルアクセスの多いジョブを実行する場合に性能が低下してしまいます。この問題を改善するためには、各計算マシンにローカルディスクを取り付けるという対処が考えられます。Diskless Cluster とは言えなくなってしまうのですが、`tmp` や `scratch` などのワーキングディレクトリ、そして `swap` 領域を個々のローカルディスク (一般に HDD) 上に作成します。ジョブの実行前にデータファイルをローカルディスク上のディレクトリにコピーしておき、実行後に回収することで高速にジョブを実行することが可能になります。また、ローカルディ

スクは作業用としてのみ使用するため、計算マシンのシステム管理はサーバマシン上で行うことが可能です。

おわりに

実際に Diskless Cluster を構築してみると、計算マシン内で作業することはほとんどなく、サーバマシン内で多くの作業を行うのが分かるかと思います。そうすると Cluster での SSI が、思いのほか容易なのではないかと思えるかもしれません。今後、Cluster をより有効に利用していくには構築・管理の容易さが大切になってきます。私はこうした Diskless Cluster の一元管理の仕組みが、より簡単に利用できるようなれば良いと思います。現在利用されている Linux のファイルシステムや起動の仕組み、NFS といったソフトウェアなどは汎用的に作られています。そのために、Cluster で利用するとどうしても不便な場合があります。Cluster 用の Linux ディストリビューションや Cluster NFS などのプロジェクトに期待したいと思います。また、次章では主な Cluster の構築・管理ソフトウェアを紹介します。これらはディスクレスであるなしに関わらず、今回紹介したような Cluster を構築する一連の作業を自動的に行ってくれます。こうしたソフトウェアは非常に有効であり、Cluster を構築するための手間を減らすだけでなく、性能の良い Cluster を構築することができます。今後、これらの開発がさらに進み、今以上に簡単に Cluster を構築できるようになれば良いと思います。

Cluster の構築・運用に関するリソース

- FAI (Fully Automatic Installation)

FAI は Linux Cluster を構築するために、各計算マシンに Debian GNU/Linux を自動インストールするためのツールです。まず Diskless Cluster のサーバマシンに似た FAI のサーバマシンを用意します。次に、計算マシンのシステムを構築するための configure ファイルと起動ディスクを作成します。そして作成した起動ディスク (FD) を用いて計算マシンを起動します。ここまでは Diskless Cluster とほぼ同じですが、FAI においてこれはインストールの 1 ステップにすぎません。FAI では、この後に計算マシンのローカルディスクに対して、configure ファイルで指定したパッケージをインストールします。インストール後、計算マシンはローカルディスクより起動されるようになります。FAI ではユーザが行う主な作業は、最初の configure ファイルの作成です。それ以降は、ほぼ自動で個々の計算マシンのインストール作業が行われます。

<http://www.informatik.uni-koeln.de/fai/>

- ALINKA LCM (Linux Cluster Manager)

ALINKA LCM は Linux Cluster を構築・管理するためのツールです。ALINKA LCM は ALINKA RAISIN と呼ばれる商用ソフトウェアの GUI 部分を除いたものであり、オープンソースで提供されています。Cluster を構築するには、まずサーバマシンに ALINKA LCM をインストールし、設定を行います。計算マシンへの OS のインストール、および管理は、サーバマシン上の ALINKA LCM を通して行います。ALINKA LCM で構築される計算マシンは、サーバマシンのクローンとなります。また現在、ALINKA LCM と Linux のインストーラを組み合わせたオリジナルのディストリビューション (Debian GNU/Linux VT-PC Cluster) を VT の中田さんが作成されています。

<http://www.alinka.com/>

- SCore Cluster System Software

SCore は Workstation , および PC Cluster において , 高度な並列計算を行うための環境を提供します . SCore による Cluster の構築はほぼ自動化されています . インポートホストとなる通常の Linux BOX に SCore をインストールし , GUI 上で Cluster の設定を行います . そして , サーバマシンと計算マシン用の起動ディスクを作成します . インストール対象となるマシンを , 作成した起動ディスクを用いて起動すれば , 後はネットワーク経由で必要なパッケージがインストールされる仕組みです . SCore を用いた Cluster の構築は非常に便利ですが , SCore により提供されている Cluster の運用ツールはそれ以上に充実しています . Myrinet などの高性能ネットワークのサポート , Cluster 内のジョブスケジューリング , フォールトトレランス , 容易に並列プログラミングを行うことのできる環境などが提供されています . SCore を利用することで , 本格的な PC Cluster を容易に構築することができます .

<http://pdswww.rwcp.or.jp/>

謝辞 , およびフィードバック

本ドキュメントの作成 , および講習会での講演にあたり , 多大なご協力を頂いた三木・廣安先生に心より感謝いたします . そして神戸大学の児玉様 , Linux の開発・ドキュメントの作成に関わっていらっしゃる多くの方々に深く感謝いたします . 最後に , 様々な作業を手伝って頂いた三木研究室の学生の皆様に感謝いたします .

本ドキュメントの作成については細心の注意をはらいましたが , 私力の不足により間違いや分かりにくいところなどがあるかもしれません . このドキュメントに関して訂正・指摘・質問などがございましたら , 谷村 (tanisuke@mikilab.doshisha.ac.jp) 宛にメールを頂ければ幸いです (2000 年 8 月 30 日版)

参考文献

- 1) Netboot. <http://www.han.de/~gero/netboot/>.
- 2) Etherboot. <http://etherboot.sourceforge.net/>.
- 3) NILO. <http://nilo.sourceforge.net/>.
- 4) Linux Start. <http://www.linuxstart.com/applications/boot.html>.
- 5) Cluster NFS. <http://clusternfs.sourceforge.net/>.
- 6) Diskless Linux Mini Howto. <http://www.linux.or.jp/JF/JFdocs/Diskless.txt>.
- 7) NFS Howto. <http://www.linux.or.jp/JF/JFdocs/NFS-HOWTO.txt>.
- 8) NFS Root Client Mini Howto.
<http://www.linux.or.jp/JF/JFdocs/NFS-Root-Client.html>.
- 9) NFS Root Mini Howto. <http://www.linux.or.jp/JF/JFdocs/NFS-Root.txt>
- 10) 児玉宏児 (神戸大学理学部). リモートで電源 On/Off.
<http://www.math.kobe-u.ac.jp/~kodama/tips-WakeOnLAN.html>.
- 11) Aoyama Plus Project. <http://www.phys.aoyama.ac.jp/~naoya/>.
- 12) 清水尚彦. Linux/Alpha 活用講座 第 21 回 完全ディスクレスマシンの構築.
Linux Japan 2000 年 8 月号.
- 13) 野澤恵 (茨城大学理学部).
reno project. <http://www.env.sci.ibaraki.ac.jp/~snozawa/reno/>.
- 14) Linux Biscuit のおいしい食べ方. <http://www.research.co.jp/biscuit/index.shtml>.
- 15) 増田景一, 坂上仁志, 高橋豊 (姫路工業大学情報工学科).
ローエンド PC を用いた PC クラスタの構築. 第 18 回超並列計算研究会.
- 16) 小坂隆浩 (大阪産業大学). 実際の応用. PC クラスタ超入門, 1999.
- 17) Standard Performance Evaluation Corporation. <http://www.spec.org/>.