Technical Report UEC-IS-2004-3, Version 2004-05-17
Graduate School of Information Systems, The University of Electro-Communications

Clustermaticを用いたPCクラスタの構築

吉瀬 謙二　片桐 孝洋　本多 弘樹　弓場 敏嗣
電気通信大学 大学院情報システム学研究科

コスト性能比に優れた計算機システムとして，複数の汎用パーソナルコンピュータ (PC) を用いたPCクラスタが注目されている．PCクラスタを構築するためには，PCやネットワークスイッチなどのハードウェア構成，オペレーティングシステムや並列アプリケーションの支援環境を含むソフトウェア構成などにおいて選択すべき点が多い．本稿では，コスト性能比に優れていること，構築と管理が容易であること，設置面積が小さいこと，といった特徴を満たすPCクラスタの構築方法と利用方法をまとめる．

キーワード PCクラスタ，Linux, Clustermatic, 並列処理, MPI

Building a PC Cluster using the Clustermatic System Software

Kenji Kise, Takahiro Katagiri, Hiroki Honda, and Toshitsugu Yuba
Graduate School of Information Systems,
The University of Electro-Communications

As a cost-effective computer system, a PC cluster using two or more general purpose PCs has become popular. In order to build a PC cluster, there are many choices in hardware composition, such as the selection of PC components and a network switch, and in software composition including the operating system and the support environment of parallel applications. For this reason, experience of various fields is necessary to build the suitable PC cluster. We describe the way to build a PC cluster which is cost effective, easy to build and small area occupation.

Key-words PC cluster, Linux, Clustermatic, parallel processing, MPI

1 はじめに

コスト性能比に優れた計算機システムとして，多数の汎用パーソナルコンピュータ (PC) を用いたクラスタシステムが注目されるようになっている．このように，複数のPCを接続して利用する計算機システムのことを**PCクラスタ**と呼ぶことにする．

PCクラスタの利用法の一つとして，複数の独立したプログラムを処理するシステムを想定することができる．例えば，16台のPCを用いたPCクラスタに，16個の独立のプログラム (プロセス) を割り当てるといった使い方である．プロセスを処理するための時間が十分に長く，プロセスの起動や終了の手間が小さければ，個々のPCに，Linuxなどの汎用のオペレーティングシステムをインストールして，独立したPCの集合としてPCクラスタを構築することができる．PCの数が少ない場合には，この方法を検討する価値がある．しかしながら，本稿で構築するPCクラスタは，この方法を用いない．PCクラスタを構築するために開発されたシステムソフトウェアを利用する．数台のPCを利用する小規模なPCクラスタの場合であっても，システムの構築と管理が容易になるといった利点がある．このことから，本稿に示す方法の導入を検討してみるべきである．

複数のプロセスが通信しながら処理を進めていく形のアプリケーションを実行する場合には，PCクラスタを並列計算機として捉える必要がある．この場合に，PCクラスタは，個々のPCが独立した論理アドレスの空間を持つ分散メモリ型の並列計算機システムとなる．このようなシステムでは，MPIに代表されるメッセージパッシングインタフェースを用いてプログラムを記述することが一般的である．ただし，ソフトウェア分散共有メモリ[1]などのミドルウェアを利用することで，共有メモリのプログラミングモデルを利用するようにシステムを構築することも可能である．本稿で構築するPCクラスタは，MPIを用いたプログラミングの環境を提供する．しかし

ながら，今のところ，共有メモリのプログラミング環境は提供されていない．

PC クラスタを構築するためのシステムソフトウェアの選択肢は広い．国内においては，RWCP が開発し，現在は PC クラスタコンソーシアム [2] において管理されている SCore[3], が利用されることが多い．SCore は高性能で，比較的，導入も容易なシステムソフトウェアである．また，日本語のマニュアルやサポートも充実しているため，PC クラスタを構築する場合には，SCore の採用を検討すべきである．

海外のシステムソフトウェアとしては，ROCKS[4] や，Open Cluster Group が開発を進めている OSCAR[5] などが普及しているようである．本稿では，Los Alamos National Laboratory で開発が進められている Clustermatic を利用する．今のところ，これらのシステムソフトウェアに関する十分な比較はなされていない．

図 1: 第 2 世代の PC クラスタ．高さ 2U の筐体に 866MHz の Pentium III プロセッサを 2 個，ネットワークインタフェースとして Myrinet を装着．8 ノード，16 CPU 構成の PC クラスタ．オペレーティングシステムに Linux を採用．

我々は，これまでに，数種類の PC クラスタを構築してきた．第 1 世代の PC クラスタは Linux をインストールした複数の PC (Pentium II を 2 個搭載する SMP 型の計算機) をイーサネットと Myrinet で接続したものだった．

第 2 世代以降では，設置面積を削減するために，標準の 19 インチラックに PC とスイッチを格納する形態を採用している．第 2 世代の PC クラスタの写真を図 1 に示す．これは，Linux をインストールした PC をイーサネットと Myrinet で接続したものである．

その後，第 3 世代の PC クラスタとして，16 ノードで 32 CPU 構成の SCore のシステムを構築した．この写真を図 2 に示す．高さ 1U の筐体に 866MHz の Pentium III プロセッサを 2 個と Myrinet-2000 のネットワークカードを装着した PC を利用している．

そして，本稿で構築する PC クラスタは，34 ノードで 68 CPU 構成の第 4 世代の PC クラスタである．

図 2: 第 3 世代の PC クラスタ．高さ 1U の筐体に 866MHz の Pentium III プロセッサを 2 個，ネットワークインタフェースとして Myrinet-2000 を装着．16 ノード，32 CPU 構成の PC クラスタ．システムソフトウェアとして SCore を採用．

本稿では，PC クラスタ全体を管理するために利用する 1 台の計算機のことを**サーバ計算機**と呼ぶことにする．また，サーバ計算機を除く，計算を担当する計算機のことを**ノード計算機**と呼ぶことにする．

本稿で構築方法をまとめる計算機システムは，PC クラスタの構築のために開発されているシステムソフトウェア **Clustermatic 4.0 (Fall 2003)** を利用する．Clustermatic は Los Alamos National Laboratory で開発が進められている．Clustermatic は，2,816 個の Opteron プロセッサから構成される PC クラスタ ASCI Lightning に採用されており，大規模な PC クラスタにおける稼働実績を持つ．以下に，Clustermatic 4.0 の特徴を列挙する．

- GPL のオープンソースとして開発されており，無

料で入手できる．また，利用するために契約書にサインするといった処理が不要なので，手軽に試すことができる．

- シングルシステムイメージの利用環境を提供することで，利用効率を向上させている．PC クラスタ全体のプロセスの状態をサーバ計算機から把握し，制御できる．個々のノード計算機にログインして作業する必要はない．(個々のノード計算機にログインすることはできない．)
- ノード計算機として，ハードディスクを利用しないディスクレスの構成を採用できる．これにより，費用を低く抑えることができる．また，ノード計算機にオペレーティングシステムをインストールする必要がないので，構築に必要となる時間を短縮できる．
- CD-ROM などから起動すれば，既にインストール済みのオペレーティングシステムの内容を維持したまま，PC クラスタとして運用することが可能である．試験的に運用する場合などに適している．
- ノード計算機はハードディスクを利用しないディスクレスの構成なので，故障時に，ノード計算機の交換の処理が単純になる．
- ノード計算機では，1 つのシステムプロセスのみが起動している．標準的な Linux のシステムでは，10 から 40 個程度のプロセスが動作している．これらのシステムと比較して，安定した計算結果を得ることができる．
- サーバ計算機のほとんどのファイル構成は Linux と同じであり，使い慣れた Linux とほぼ同様のコマンドを用いて PC クラスタを管理できる．専用のオペレーティングシステムを採用する場合と比較して，管理の作業が複雑になることを避けることができる．

本稿の構成を示す．2 章では，3 台の PC を用いる PC クラスタの構築方法をまとめる．この章は，PC クラスタの本格導入を検討する場合や，小規模な並列計算環境を構築するための参考となる．3 章では，PC クラスタの利用方法をまとめる．4 章では，34 台のノード計算機を用いた PC クラスタの構築の様子と，その構築時間をまとめる．

2 PC クラスタの構築

本章では，1 台のサーバ計算機と，2 台のノード計算機の合計 3 台の PC を用いた PC クラスタの構築方法をまとめる．構築する PC クラスタの構成を図 3 に示す．

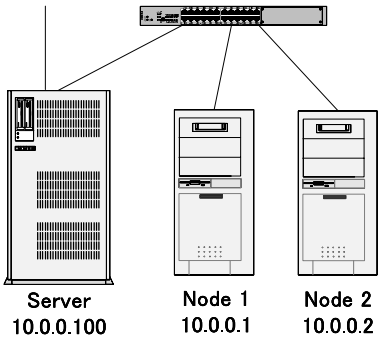

図 3: 1 台のサーバ計算機と，2 台のノード計算機，ネットワークスイッチを用いた PC クラスタの構成．

2.1 計算機のハードウェア仕様

2.1.1 サーバ計算機のハードウェア仕様

サーバ計算機のハードウェア仕様をまとめる．

- RedHat Linxu 9.0 の全てのパッケージをインストールできる PC であること．
- 6 GB 以上の容量のハードディスクを搭載すること．RedHat Linxu 9.0 の全てのパッケージをインストールするために 5GB 程度の容量が必要となる．また，ユーザのホーム領域などの利用のために 20GB 以上の容量を持つハードディスクを搭載したい．
- CD-ROM ドライブを搭載すること．Clustermatic の CD-ROM を用いてインストールするために，CD-ROM ドライブを利用する．ただし，ネットワーク経由でファイルをコピーすれば，必ずしも CD-ROM ドライブが必要という訳ではない．
- 2 つのネットワークインタフェース (ポート) を搭載すること．100Mbps, 1000Mbps のどちらでも問題ない．(本稿では取り上げないが，Myrinet を利用することもできるらしい．) ポートの 1 つは，ネットワークスイッチを介して，PC クラスタを構築するノード計算機に接続される．もう 1 つのポートは，構築した PC クラスタを他の計算機と接続するために利用される．構築する PC クラスタをスタンドアロンで利用する場合には，1 つのネットワークインタフェースのみで問題ない．

- ルート権限でコマンドを実行する環境が整っていること．特にマウスは必要ない．キーボードとディスプレイが接続されている，あるいは，リモートの計算機からログインして作業できること．

2.1.2 ノード計算機のハードウェア仕様

ノード計算機のハードウェア仕様をまとめる．サーバ計算機と比較して，ノード計算機の満たすべき項目は少ない．

- サーバ計算機と同じ Linux のカーネルが動作する PC であること．
- CD-ROM ドライブから起動できること．CD-ROM ドライブが搭載されている PC，あるいは USB で接続された CD-ROM ドライブから起動できる環境が整っていること．
- 1 つ以上のネットワークインタフェースを搭載すること．100Mbps, 1000Mbps のどちらでも問題ない．

図 4: ノード計算機として用いる Proside SV t2150-mini の内部．高さ 1U の筐体に 2.8GHz の Xeon プロセッサを 2 個，DDR266 デュアルチャネルのメモリを 1GB 搭載する．マザーボードには 1000Mbps のネットワークポート 1 つと，100Mbps のネットワークポート 1 つを持つ．今回の構成では，右上のハードディスクは使わない．

2.2 サーバ計算機のセットアップ

2.2.1 オペレーティングシステムのインストール

サーバ計算機にオペレーティングシステムとして Red Hat Linux 9 をインストールする．Red Hat Linux 9 のパッケージはフルインストールする．インストールの種類で “CUSTOM” を選択し，パッケージグループの選択では “Everything” を選択する．

2.2.2 Clustermatic パッケージのインストール

カーネルを通常の Linux カーネルから，変更を加えた Clustermatic のカーネルに変更する前に，依存関係のあるパッケージを削除する．このためのコマンドを次に示す．

```
# rpm -e oprofile
# rpm -e lam pvm pvm-gui
```

Clustermatic のホームページ [6] から，CD-ROM の iso イメージのファイル CM4.iso をダウンロードして，CD-R に書き込む．作成した Clustermatic の CD-ROM をマウントして Clustermatic のパッケージをインストールする．

Pentium 4 プロセッサを搭載するシステムを想定したコマンドを次に示す．

```
# mount /mnt/cdrom
# cd /mnt/cdrom
# cd RPM/i686
# rpm -Uvh *
```

CD-ROM の RPM 以下のディレクトリには，アーキテクチャに対応したサブディレクトリが存在する．例えば Pentium 4 のシステムのために i686 というサブディレクトリが存在する．利用するアーキテクチャに応じて，適切なサブディレクトリを選択すること．

2.2.3 config ファイルの編集

/etc/clustermatic/config ファイルを編集する．**ここでミスが発生しやすいので注意すること．**

/etc/clustermatic/config ファイルの interface の行を設定する．この設定をおこなったネットワークインタフェースはオペレーティングシステムの起動時に強制的に指定したアドレスに再設定される．ここでは，サーバ計算機のネットワークインタフェース eth1 を，アドレスを 10.0.0.100，ネットマスク 255.255.255.0 に設定する記述例を示す．

```
interface eth1   10.0.0.100 255.255.255.0
```

次に，ノード計算機の IP アドレスの範囲を指定する．ノード計算機の数にも依存するが，プライベートアドレスを利用するので，クラス A，クラス B，クラス C のどのアドレスを用いても問題はない．iprange に続く数字は，ノード計算機の論理番号 (識別子) として利用する数字を指定する．この番号のことを**ノード番号**と呼ぶこと

にする．また，ノード番号として 1 を持つノード計算機のことをノード 1 と呼ぶことにする．下の設定では，ノード番号の開始値として 1 を指定する．これにより，最初のノード計算機はノード 1，次のノード計算機はノード 2 となる．続く 2 つのアドレスを用いて，IP アドレスの範囲を指定する．ここでは，2 台のノード計算機を使うので，10.0.0.1 から 10.0.0.2 の範囲を指定する．設定例を下に示す．

```
iprange 1 10.0.0.1  10.0.0.2
```

ノード番号の開始値には，任意の数字を指定できる．特に，ノード番号の開始値を 0 とするか 1 とするかは難しい判断である．ここでは，IP アドレスのホスト部の番号とノード番号が一致することを優先して，開始値を 1 に設定した．

続いて，ノード計算機のノード番号と MAC アドレスの組を列挙する．設定ファイルの 1 行に，ノード計算機 1 台の構成を記述する．node という文字に続いて，ノード番号，MAC アドレスを記述する．下の設定では，MAC アドレス 00:e0:81:27:5e:f9 の計算機がノード 1 に，MAC アドレス 00:e0:81:27:5e:f1 の計算機がノード 2 に設定される．

```
node 1 00:e0:81:27:5e:f9
node 2 00:e0:81:27:5e:f1
```

外見から，計算機の MAC アドレスを知ることは困難である．MAC アドレスを知る一つの方法は，Clustermatic の CD-ROM を用いて計算機を起動して，表示された MAC アドレスを書き写すことである．

特に，大規模なクラスタシステムを構築する場合に，MAC アドレスを記述していく方法には無駄が大きい．/usr/lib/beoboot/bin/ 以下の nodeadd コマンドを用いてノード計算機の MAC アドレスを自動登録することもできる．サーバ計算機では，以下のコマンドを実行する．

```
# /usr/lib/beoboot/bin/nodeadd -a -e -n 1 eth1
```

その後，ノード計算機を起動することで，見つかったノード計算機から順番にノード 1, ノード 2, という様に config ファイルに追加されていく．オプション -n で指定する数字が，ノード番号の開始値である．

config ファイルを変更した場合には，デーモンプロセスを再起動する．このためのコマンドを示す．

```
# /sbin/service clustermatic restart
```

2.2.4 Phese 2 boot image の作成

config ファイルの編集が終わったところで，サーバ計算機を再起動する．この時点で，標準の Linux カーネルから， Clustermatic のカーネルに変わっているはずである．このことを，次のコマンドで確認する．

```
# uname -a
Linux server.cluster 2.4.22-cm36smp
```

特に，SMP 構成のノード計算機を用いる場合には，最後の表示に smp という文字があることを確認する．オペレーティングシステムの起動時に，複数のカーネルから選択できる場合には，適切なものを選択するように注意する．

続いて，ノード計算機が起動する際に必要となるカーネルを構築するために，次のコマンドを実行する．このコマンドにより，/var/clustermatic/boot.img というファイルが生成される．

```
# beoboot -2 -n
```

2.2.5 再起動と起動確認

これまでの作業が終了したら，サーバ計算機を再起動する．起動時に，適切なデーモンが起動し，下のメッセージが表示されることを確認する．

```
Starting Clustermatic...
  Configuring network interface (vmnet1): [  OK  ]
  Loading modules:                        [  OK  ]
  Setting up libraries:                   [  OK  ]
  Mounting bpfs:                          [  OK  ]
  Starting bpmaster:                      [  OK  ]
  Starting beoserv:                       [  OK  ]
```

2.3 ノード計算機のセットアップ

ノード計算機に Clustermatic の CD-ROM をセットして起動する．数分後に，ノード計算機の設定が終了する．起動時にしか CD-ROM ドライブを利用しないので，図 5 に示す様に，USB 接続の CD-ROM ドライブを利用することで，ノード計算機のコストを抑えることができる．

この作業 (といっても CD-ROM を入れて起動するだけだが) を，それぞれのノード計算機で繰り返す．

3 PC クラスタの動作テストと利用方法

章の設定により，次に示す IP アドレスを持ち，3 台で構成される PC クラスタを構築方法することができた．本章では，小規模 PC クラスタの動作テストと利用方法をまとめる．

図 5: ノード計算機のセットアップ．USB 接続の CD-ROM ドライブを用いて起動することで，ノード計算機のコストを削減できる．必要に応じて，BIOS 設定を変更し，起動デバイスを USB 接続の CD-ROM ドライブに設定する．

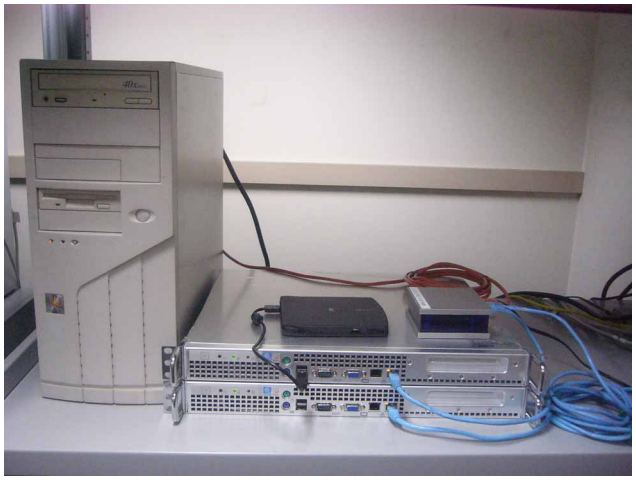

図 6: 構築したクラスタシステム．左に見えるのが Pentium 4 を搭載したサーバ計算機．Hyper Threading を有効にしているので，動作するカーネルはノード計算機と同じく SMP 用のものである．

```
server host : 10.0.0.100
node 1      : 10.0.0.1
node 2      : 10.0.0.2
```

3.1 ノード計算機の属性の変更

ルートでログインして，クラスタの状態を表示するための bpstat というコマンドを実行する．全ての計算機が動作している場合の表示例を示す．

```
# bpstat
  1-2    up   ---x------ root   root
```

up という文字から，ノード 1 とノード 2 が起動していることを確認できる．計算機が利用できない場合には down という文字が表示される．

up という文字の右に出ているマイナスと x の部分は，ノード計算機の属性を示している．上の表示例は，ルートのみがノード計算機でプログラムを実行する権利を持つことを示している．

一般のユーザ権限でプログラムの実行を許可するためには，bpctl というコマンドを利用する．ノード 1 とノード 2 の属性を変更する例を示す．

```
# /usr/sbin/bpctl -S 1 -m 111
# /usr/sbin/bpctl -S 2 -m 111
```

bpstat コマンドを用いて，クラスタの状態が変わったことを確認する．

```
# bpstat
  1-2    up   ---x--x--x root   root
```

複数のノード計算機の属性を変更するために，ノード番号の範囲を指定することもできる．ノード 1 とノード 2 の属性を変更するコマンドを示す．

```
# /usr/sbin/bpctl -S 1-2 -m 111
```

3.2 コマンドの実行

ノード計算機でプログラムを実行する bpsh というコマンドの利用例を示す．現在の時刻を表示する date というコマンドを実行するコマンドと実行例を示す．

```
# bpsh -a -p date
  2: Thu Feb 19 17:55:08 JST 2004
  1: Thu Feb 19 17:55:08 JST 2004
```

bpsh コマンドのオプション -a は，起動している全てのノード計算機でコマンドを実行することを指示する．オプション -p は，出力結果の先頭にノード番号を表示することを指定する．

オプション -p を指定しない場合の実行結果を示す．

```
# bpsh -a date
Thu Feb 19 17:57:03 JST 2004
Thu Feb 19 17:57:02 JST 2004
```

計算ノードを指定する場合の実行例を示す．オプション -a を用いる代わりに，ノード番号を利用する．コマンドの後ろにコメントを付加した．

```
# bpsh 1 date   // 計算ノード 1 で実行
# bpsh 2 date   // 計算ノード 2 で実行
# bpsh 1-2 date // 計算ノード 1 と計算ノード 2 で実行
```

ノード計算機で動作しているカーネルのバージョンを確認してみる．見やすいように，出力結果に改行を入れている．

```
# bpsh -a -p uname -a
  1: Linux n1 2.4.22-cm36smp #1 SMP Fri Nov 7
     12:03:23 MST 2003 i686 i686 i386 GNU/Linux
  2: Linux n2 2.4.22-cm36smp #1 SMP Fri Nov 7
     12:03:23 MST 2003 i686 i686 i386 GNU/Linux
```

3.3 ネットワークファイルシステムの設定

ファイルの入出力が必要となるアプリケーションのために，サーバ計算機のディレクトリを，NFS を用いて，ノード計算機にエキスポートする方法をまとめる．ここでは，サーバ計算機の上で /work というディレクトリを作成し，ノード計算機では，これを /work というディレクトリにマウントする．/home ディレクトリをマウントする方が都合がよい場合がある．その場合には，/work を /home に変更して設定すればよい．

まず，サーバ計算機が NFS サーバとして動作するように設定する．

```
# mkdir /work
# chmod 777 /work
# chkconfig --list nfs
# chkconfig nfs on
```

ファイル /etc/exports に次の行を追加し，ノード計算機へのファイルのエキスポートを許可する．

```
/work 10.0.0.0/24(rw,sync)
```

ディレクトリのエキスポート設定を反映するために，次のコマンドを実行する．

```
# exportfs -a
```

ノード計算機が，サーバ計算機のディレクトリをマウントするように，/etc/clustermatic/fstab を編集し，次の行を追加する．

```
MASTER:/work  /work  nfs  nolock  0 0
```

ノード計算機を再起動し，ディレクトリがマウントされていることを確認する．以下のコマンドを実行し，結果に /work というディレクトリ名が表示されることを確認する．

```
# bpsh -a -p df
```

3.4 逐次プログラムの実行

本節では，逐次プログラムを記述，コンパイルし，実行する方法をまとめる．

3.4.1 標準出力の利用 printf

標準出力に文字列を出力する初歩的なプログラムの実行方法を示す．ファイル名は main.c である．ファイルの内容を下に示す．

```
#include <stdio.h>
int main(){
  printf("hello, world\n");
  return 0;
}
```

クラスタのノード計算機には，最低限のファイルしか存在しない．動的にライブラリをリンクする場合にはエラーが発生することがある．この場合には，コンパイル時に静的にライブラリをリンクする．gcc を用いる場合には，次に示すように -static オプションを用いる．

```
$ gcc -static main.c
```

2 台のノード計算機でプログラムを実行するコマンドと実行結果を示す．それぞれのノードに標準出力に文字列が表示されている様子を確認できる．

```
$ bpsh 1-2 -p a.out
  1: hello, world
  2: hello, world
```

3.4.2 ファイルへの出力 fopen

ファイルに文字列を出力するプログラムを用いてテストする．ファイル名は main.c である．出力先のファイル名は test.log とした．ファイルの内容を下に示す．

```
#include <stdio.h>
#include <stdlib.h>
int main(){
  FILE *f = fopen("test.log", "w");
  fprintf(f, "hello, world\n");
  fclose(f);
  return 0;
}
```

gcc を用いる場合には，次に示すように -static オプションを用いてコンパイルする．

```
$ gcc -static main.c
```

1 台のノード計算機でプログラムを実行するコマンドと結果を示す．プログラムを起動したディレクトリが，ノード計算機で利用できない場合には，ファイルにアクセスができないので，エラーを表示してプロセスが終了する．

```
$ bpsh 1 -p a.out
bpsh: Child process exited abnormally.
```

サーバ計算機の /work というディレクトリをエキスポートする設定をおこなっていれば，そのディレクトリに移動して，コマンドを実行すればよい．

```
$ cp a.out /work
$ cd /work
$ bpsh 1 -p a.out
```

3.4.3 複数のプロセスの起動とファイル操作

前節でコンパイルしたファイルへの出力をおこなうプログラムを複数のノード計算機で実行すると，複数のプロセスが同じ名前 test.log のファイルに結果を出力するために，どちらかの結果が上書きされてしまう．

この問題を解決する方法の一つは，ノード計算機毎にディレクトリを作成し，そのディレクトリにおいてプログラムを実行することである．

まず，次のコマンドを用いて，ディレクトリを作成し，実行ファイルをコピーする．

```
$ cd /work
$ mkdir node1
$ mkdir node2
$ cp a.out node1
$ cp a.out node2
```

それぞれのディレクトリに移動して，プログラムを実行する．

```
$ cd /work/node1
$ bpsh 1 a.out &
$ cd /work/node2
$ bpsh 2 a.out &
```

3.5 MPI プログラムの実行

本節では，MPI を用いたアプリケーションの実行方法をまとめる．

MPI を利用するため，実行パスに /usr/mpich-p4/bin を追加する．実行パスが設定されているか，次のコマンドで確認できる．

```
$ which mpicc
/usr/mpich-p4/bin/mpicc
```

実行パスが正しく設定されていない場合には，次の様なエラーメッセージが出力される．

```
$ which mpicc
mpicc: Command not found.
```

文献 [7] の 3 章に記述されている greetings という C のプログラムをコンパイルするコマンドを示す．このプログラムは，ランク 0 以外のプロセスがランク 0 にメッセージを送信して，それを表示するプログラムである．このプログラムは書籍のホームページ
www.cs.usfca.edu/mpi からダウンロードできる．

```
$ mpicc -O2 greetings.c -o a.out
```

myprog.cc というファイル名の C++ のコードをコンパイルする場合には，**mpiCC** というコマンドを利用する．

```
$ mpiCC myprog.cc -o a.out
```

myprog77.f というファイル名の Fortran 77 のコードをコンパイルする場合には，**mpif77** というコマンドを利用する．

```
$ mpif77 myprog77.f -o a.out
```

myprog90.f というファイル名の Fortran 90 のコードをコンパイルする場合には，**mpif90** というコマンドを利用する．

```
$ mpif90 myprog90.f -o a.out
```

プロセス数を 1,2,3 個のそれぞれに設定して実行するコマンドと実行結果を示す．計算ノードの数が 2 台なので，プロセス数として 3 を指定するとエラーが表示される．

```
$ mpirun -P --np 1 a.out

$ mpirun -P --np 2 a.out
Greetings from process 1!

$ mpirun -P --np 3 a.out
Not enough nodes to allocate all processes
```

今回利用している計算機は各ノードが 2 個の CPU を搭載する SMP 型の計算機なので，各ノードに 2 つのプロセスを作成することができる．このためのコマンドと実行結果を示す．

```
$ mpirun -P --np 4 --nper 2 a.out
Greetings from process 1!
Greetings from process 2!
Greetings from process 3!
```

オプション - - nper ではなく -N を使うと Segmentation fault になるので注意すること．

3.6　バッチ処理を用いたプログラムの実行

Clustermatic には，BJS と呼ばれるバッチ処理システムが組み込まれている．

bjsstat コマンドにより，バッチ処理システムの状態を確認できる．

bjssub コマンドで，ジョブを投入することができる．

4　PC クラスタの構築時間

PC クラスタを構築するための時間はできるだけ短く抑えたい．構築時間の目安となるように，ソフトウェアの構築のために必要となる時間を計測した結果をまとめる．ハードウェアのセットアップが全て終了していることを前提としており，このための時間は含んでいないことに注意すること．

2 台のノード計算機を持つ PC クラスタの構築時間を表 1 にまとめる．Linux のインストールと設定に 2 時間，Clustermatic のインストールと設定に 1 時間の合計 3 を必要とした．Linux のインストールに慣れていない場合とサーバ計算機として低い性能の PC を利用する場合には，より多くの時間が必要となる可能性がある．

Linux のインストールと設定	2 時間
Clustermatic のインストールと設定	1 時間

表 1: **2 台のノード計算機**を持つ PC クラスタの構築時間．合計で 3 時間を必要とした．

同様の構成で，ノード計算機の台数を 34 台に増やした場合の構築時間を表 2 にまとめる．この場合には，全体で 4 時間を必要とした．CD-ROM を取り付けて起動するだけの作業でノード計算機の設定が完了するので，ノード計算機の数を増やしても，構築時間はそれほど変わらない．

Linux のインストールと設定	2 時間
Clustermatic のインストールと設定	2 時間

表 2: **34 台のノード計算機**を持つ PC クラスタの構築時間．合計で約 4 時間を必要とした．

PC クラスタの構築作業の様子を図 7 から図 10 に示す．34 台のノード計算機の中で，2 台にトラブルが発生した．1 台は電源が入らないというトラブル，もう 1 台は，ハードディスクのアクセスランプが常に点灯するという異常

図 7: ノード計算機が梱包されている箱が 34 個届いたところ．時刻は 11 時 26 分．

図 8: 30 分が経過し，11 台のノード計算機をラックに格納したところ．高さ 40U のラック 1 つに格納することも可能だが，20U のラックを 2 つ利用することにした．作業は一人でおこなっている．時刻は 12 時 2 分．

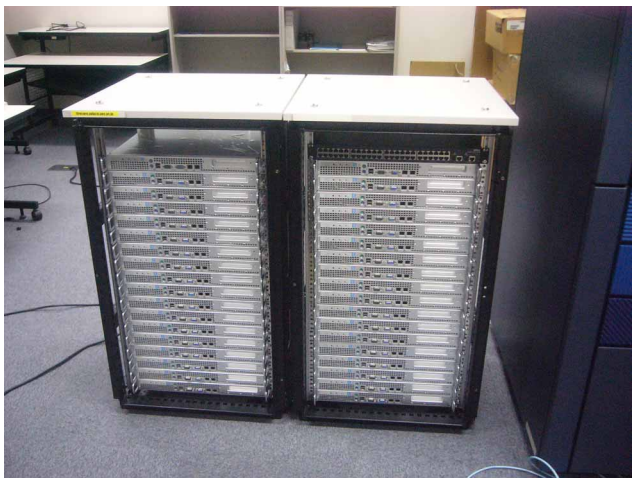

図 9: 全てのノード計算機をラックに格納したところ．時刻は 12 時 49 分.

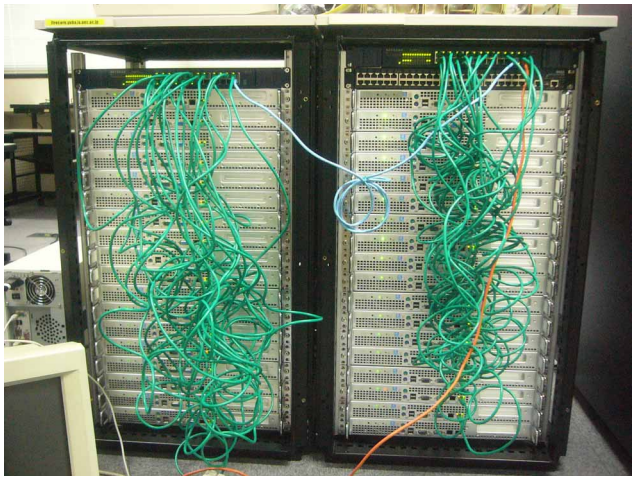

図 10: 配線とソフトウェアの設定が完了したところ．時刻は 15 時 33 分．作業を開始する前にサーバ計算機の構築が終了していた．このため，ノード計算機のラックへの取り付けと，ソフトウェア設定の作業は 4 時間程度で終了した.

が発生した．事前に準備していた予備の 2 台の計算機を用いて，合計 34 ノードとして起動することに成功した.

特に，ノード計算機の数が多くなる場合には，2 から 4 台程度，予備のための計算機を準備しておくとよい.

5　おわりに

コスト性能比に優れた計算機システムとして，汎用のパーソナルコンピュータを利用する PC クラスタが注目されている．PC クラスタを構築するためには，PC を構成する部品やネットワークスイッチなどのハードウェア構成，オペレーティングシステムや並列アプリケーションの支援環境を含むソフトウェア構成などにおいて選択すべき点が多い.

Los Alamos National Laboratory で開発が進められているシステムソフトウェア Clustermatic を利用して，コスト性能比に優れていること，構築と管理が容易であること，設置面積が小さいこと，といった特徴を満たす PC クラスタの構築方法と利用方法をまとめた.

Clustermatic は，優れたシステムソフトウェアであるにもかかわらず，その認知度は高くない．本稿が Clustermatic を用いた PC クラスタの構築と利用の際の参考となれば幸いである.

A　起動用のフロッピーディスクまたは USB フラッシュメモリの作成

ノード計算機を起動するためのフロッピーディスクの作成方法をまとめる.

/etc/clustermatic/config.boot を編集して必要なデバイスのみを選択する．FireCore クラスタのノード計算機は，Intel 82545EM Gigabit Ethernet と，Intel 82551QM Fast Ethernet を搭載している．PCI デバイスは，e1000, e100 に関するのものを残して，その他の行を削除する.

下のコマンドで，起動用のイメージファイル cm.img を構築する.

```
# beoboot -1 -f -o cm.img
  -k /boot/vmlinuz-2.4.22-cm36beoboot
```

A.1　起動用のフロッピーディスクの作成

下のコマンドで，フロッピーディスクを作成できる.

```
# dd if=cm.img of=/dev/fd0 bs=1024
```

A.2　起動用の USB フラッシュメモリの作成

下のコマンドで，起動用の USB フラッシュメモリを作成できる．ただし，起動可能な USB フラッシュメモリと計算機には相性があるので注意する必要がある.

```
# dd if=cm.img of=/dev/sda1 bs=1024
```

fdisk コマンドで，USB フラッシュメモリの状態を確認できる．コマンドと出力の例を下に示す．

```
# fdisk -l /dev/sda
Disk /dev/sda: 2 heads, 32 sectors, 1000 cylinders
Units = cylinders of 64 * 512 bytes

   Device Boot  Start  End  Blocks Id  System
/dev/sda1   *       1  999   31952  4  FAT16 <32M
```

幾つかの USB フラッシュメモリを試したところ，下の型番のもので起動に成功した．

- BUFFALO RUF-C32ML Clip Drive
- ELEVE HandyBit E-H2-512MB RV03BMP1
- Princeton XiaoJr. PFU-J32

B　ノード計算機のハードディスクからの起動

USB 接続のフラッシュメモリ，CD-ROM, フロッピーディスクドライブに加えて，内蔵のハードディスクからノード計算機を起動するように設定することもできる．このための手順を簡潔にまとめる．

1. ハードディスクのパーティションを作成する．例えば，Windows 98 の CD-ROM を用いて計算機を起動して，fdisk コマンドによりパーティションを作成する．
2. MBR を作成する．例えば，Windows 98 の CD-ROM を用いて計算機を起動して，fdisk /mbr というコマンドを実行する．
3. 作成したパーティションに起動用のイメージを書き込む．例えば，KNOPPIX 3.3 の CD-ROM を用いて計算機を起動し，USB フラッシュメモリなどで，書き込むイメージファイル cm.img をコピーし，次のコマンドを実行する．
   dd if=cm.img of=/dev/hda1 bs=1024

これらの手順により，ハードディスクから起動することができる．詳細は SYSLINUX のホームページを参照のこと．

C　標準のコンパイラの設定

MPI のプログラムをコンパイルする際に利用される標準のコンパイラを設定するには，適切にシェルスクリプトの内容を変更する．

例えば，mpicc のコンパイラを変更するためには，/usr/mpich-p4/bin/mpicc の CCBASE=”gcc” , CLINKERBASE=”gcc” という行のコマンド名を変更する．

D　バッチ処理システム BJS

Clustermatic に組み込まれている BJS と呼ばれるバッチ処理システムの設定と利用方法に関して述べる．

D.1　BJS の設定

ファイル /etc/clustermatic/bjs.conf の内容を適切に編集する．FireCore クラスタの設定内容を示す．

```
spooldir   /var/spool/bjs
policypath /usr/lib64/bjs:/usr/lib/bjs
socketpath /tmp/.bjs
#acctlog   /tmp/acct.log

pool default
        policy filler
        nodes  1-34
```

FireCore クラスタは 34 台の全てのノードをバッチシステムの対象とする．

設定ファイルを更新した後に，下のコマンドを用いてデーモンを再起動する．

```
# /etc/rc.d/init.d/bjs restart
```

D.2　ジョブの管理

ジョブの投入状況を確認するには/usr/bin/bjsstat コマンドを利用する．待機しているジョブがない場合の実行例を下に示す．

```
# bjsstat
Pool: default   Nodes (total/up/free): 34/34/34
ID      User     Command            Requirements
```

ジョブを投入するには，/usr/bin/bjssub コマンドを利用する．オプション n を用いて利用するノード数を指定し，オプション s を用いて実行時間（秒）を指定する．オプション O を用いて標準出力を記録するためのファイル名を指定する．実行時間は必要となる時間を超える様に十分な時間を指定すること．

3600 秒の実行時間，標準出力のファイル名 test.log，1 台のノードを指定してプログラム a.out のジョブを投入するコマンドを下に示す．

```
# bjssub -O test.log -n 1 -s 3600 a.out
```

一つのジョブを投入した直後の状態を表示させた結果を示す.

```
#bjsstat
Pool: default   Nodes (total/up/free): 34/34/33
ID      User     Command         Requirements
   27 R kis      a.out           nodes=1 secs=3600
```

同様のジョブを 3 回投入した場合の状態を表示させた結果を示す.

```
#bjsstat
Pool: default   Nodes (total/up/free): 34/34/31
ID      User     Command         Requirements
   28 R kis      a.out           nodes=1 secs=3600
   29 R kis      a.out           nodes=1 secs=3600
   30 R kis      a.out           nodes=1 secs=3600
```

実行中のジョブを削除するためには，/usr/bin/bjsctl コマンドを利用する．ジョブ番号が 30 番のジョブを削除するためのコマンドを示す.

```
#bjsctl -r 30
```

E　計算機の管理に関する補足

E.1　システムの起動

まず，サーバ計算機を起動する．起動を確認した後に，ノード計算機を起動する.

E.2　システムの停止

まず，次のコマンドを利用して，全てのノード計算機 (ノード 0 とノード 1) を停止する.

```
# bpctl -S 1-2 --halt
```

ノード計算機の停止を確認した後に，サーバ計算機を停止する.

```
# halt
```

E.3　ノード計算機の再起動

特定のノード計算機を再起動するには，bpctl コマンドを利用する．ノード 1 を再起動するコマンドを示す.

```
# bpctl -S 1 --reboot
```

参考文献

[1] L. Kai. IVY: A shared virtual memory system for parallel computing. *International Conference on Parallel Processing*, pages 94–101, 1988.

[2] *PC Cluster Consortium, pdswww.rwcp.or.jp.*

[3] 石川裕, 佐藤三久, 堀敦史, 住元真司, 原田浩, and 高橋俊行. *Linux* **で並列処理をしよう**. 共立出版, 2002.

[4] ROCKS, Rocks Cluster Distribution, An Open Source High Performance Linux Cluster Solution, www.rocksclusters.org.

[5] OSCAR, Open Source Cluster Application Resources, oscar.openclustergroup.org.

[6] CLUSTERMATIC, Redesigning the Cluster Architecture, A project of the Cluster Research Lab in the Advanced Computing Laboratory at Los Alamos National Laboratory, www.clustermatic.org.

[7] P. パチェコ. *MPI* **並列プログラミング**. 倍風館, 2001.