Beowulf型 PC クラスタの構築

三宅川 弘明・水谷 由宏

上智大学理工学部物理学科　東京都千代田区紀尾井町 7-1（〒102-8554）
キーワード:ベオウルフ，PC クラスタ，MPICH，姫野ベンチ

Construction of a Beowulf Type PC Cluster

Hiroaki MIYAGAWA and Yoshihiro MIZUGAI

Department of Physics, Faculty of Science and Technology, Sophia University
7-1 Kioicho, Chiyoda-ku Tokyo, 102-8554

Keywords: Beowulf, PC cluster, MPICH, Himeno benchmark

1.はじめに

今日，科学の諸分野において計算機によるシミュレーションが「計算機実験」という有力な実験手法の一つとして確立し始めている．特に分子構造や分子集団の挙動など，広い意味での分光学的および分子動力学的見地からのアプローチが必要となるテーマは数多い．我々は前に希ガスのクーロン爆発や TOF スペクトルメーターにおける空間電荷効果について計算機シミュレーションの報告をした[1－3]．そこでは計算機実験を行なうことで理論的解析や実験観測が不可能な物理量，例えば多粒子系における相互作用の時間変化や極微小時間での粒子分布の時間変化などを議論することが可能となる．

このように計算機実験は非常に有効な手段であるが，計算機実験を行うために必要なスーパーコンピュータは非常に高価かつ巨大な設備であり，大学等の研究室レベルでの導入はほぼ不可能であると言える．

1994 年に NASA の Goddard Space Flight Center (GSFC)の Thomas Sterling と Donald Becker は 16 個の DX4 プロセッサを 10Mbps の Ethernet でつないで，Linux マシンそれぞれが協調して一つのシステムとして動作する分散メモリ型のクラスタシステムを構築し「Beowulf」と名づけた．[4]

以来，一般に入手可能なコンピュータに Linux をインストールし，Ethernet などのネットワークで接続し，NFS を介して/home を共有し，Remote Shell (rsh)によって互いに信用した通信を行い，Parallel Virtual Machine (PVM)[5] や Message Passing Interface (MPI)[6] などのフリーな通信ライブラリを組み込んでスーパーコンピュータ並の計算性能を出せるように最適化したシステムを「Beowulf 型 PC クラスタ」と呼ぶようになった．

特に近年のコモディティハードウェアの飛躍的な性能向上によって 1CPU あたりの性能でもクロック数が数 GHz は当たり前となり，また大幅な価格の下落によってネットワークの分野でも数万円程度で非常に高速な Gigabit Ethernet のネットワーク環境が手に入れられるようになった．

その結果，TOP500 Supercomputer Sites[7] で世界のスーパーコンピュータの上位に PC ク

ラスタがランキングされていることからもわかるように，非常に安価にスーパーコンピュータと同等の演算能力を持つ Beowulf 型 PC クラスタの構築が可能となった.

ところが最近では各種ベンダーによって技術料や人件費を上乗せされたクラスタ製品が発売され, 折角非常に優れたコストパフォーマンスを持った Beowulf 型 PC クラスタの意義が翳ってしまっている.

そこで本稿では実際に 8 台の PC を Gigabit Ethernet と Fast Ethernet で接続し，Beowulf 型 PC クラスタ(Fig.1)を構築した際の手順をできるだけ具体的に，各種設定ファイルの設定値なども含め紹介することとする.

Fig. 1. The Beowulf type PC Cluster at Molecular Physics Lab., Sophia University.

2.ハードウェア

Beowulf 型 PC クラスタを構築するためにまずはハードウェアを揃えなければならない. 主に必要となる機材はファイルサーバと演算ノードという PC 群とスイッチングハブ, ネットワークケーブルである.

2.1 ファイルサーバ

外部からクラスタシステムへの接続はファイルサーバを通して行われ，演算ノードは外部ネットワークと直接接することはない．

そのためファイルサーバについては外部と内部のネットワークを結ぶために 2 枚の Fast Ethernet の Network Interface Card (NIC)が必要となる．

またファイルサーバは各演算ノードの NFS サーバとして稼働するためチップセットとして ServerWorks GC-SL を搭載した Linux 動作確認済みの DELL PowerEdge 600SE を Table Ⅰのようにカスタマイズして購入した．

File server	
CPU	Intel Pentium4 processor 2.80GHz
FSB	533MHz
2nd cash	512KB L2 cash
Chipset	ServerWorks GC-SL
Memory	DDR266 ECC 256MB×2
HDD	40GB 7200rpm
Drive	CD-ROM / 2mode Floppy
Video	onboard ATI RAGE XLTM 8MB
Network	Onboard Intel PRO/1000MT
	Intel PRO/100S Server

Table Ⅰ Machine spec for File server.

2.2 演算ノード

演算ノードは実際に並列演算を行うマシン群となる．

MPI ではプログラムを実行したノード上にマスタープロセスが起動され，その他のノードのプロセスはマスタープロセスによって rsh によって起動される．

その為，特にプログラムをコンパイル・実行する演算ノードのことをマスター演算ノード，その他の演算ノードのことをスレーブ演算ノードと呼ぶ．

演算ノードはマスターとスレーブの合わせて最低 2 台以上であれば何台でもかまわないが，プログラムの並列化の都合上 2 の階乗の台数にすることが多い．

また演算ノードについてはコマンド/NFS 用の Fast Ethernet の NIC と MPI 用の Gigabit Ethernet の NIC がそれぞれ 1 枚ずつ搭載されている必要がある．これらのいずれも Linux をサポートしているものでなくてはならない．

本稿ではクラスタ用に最適な構成とするために，演算ノードでは Table Ⅱのような自作機を用いたがメーカー製マシンでもかまわない．但し最新のマザーボードや CPU を用いているマシンの中には Linux のカーネルによってサポートされていないものもあるので注意が必要となる．メーカー製品の中で Linux 対応が明記してあるものを購入するか，対応するハードウェアを調べた上で購入し，

Compute node	
CPU	Intel Pentium4 processor 2.40GHz
FSB	533MHz
2nd cash	512KB L2 cash
M/B	ASUS P4GE-VM
Chipset	Intel845GE
Memory	Transcend DDR266 512MB×2
HDD	Seagate 7200rpm 40GB
Drive	MITSUMI 2mode Floppy
Video	Onboard Intel 82845GE GMCH
Case	OWLTECH OWL-103-Silent
Network	Onboard Intel 10/100BASE-TX
	Intel PRO/1000 MT Desktop

Table Ⅱ Machine spec for Compute node.

自作することを薦める.

2.2.1 CPU

CPU は Intel や AMD の x86 系のものであればおおむね動作する．また Dual CPU などの SMP システムも考えられるが, オーバーヘッドにより CPU１個あたりの性能はシングル CPU システムより低下してしまう.

2.2.2 マザーボード

マザーボードは Linux がデバイスを認識できるように長く使用されているチップセットを搭載した物を用いる.

2.2.3 メモリ

Linux で 32bitCPU を扱う場合，1 プロセスでは最大 3GB までのメモリまでしか扱えない．したがって 1 ノードあたり最大 3GB までの範囲内でできるだけ大容量で高速なものを搭載することが望ましい．特に負荷の高い計算を連続して行う場合などは，信頼性の高いメモリを導入するべきである.

2.2.4 ハードディスク

一部のクラスタではディスクレスシステムを導入しているが，ネットワーク負荷が高くなってしまうのでディスクフルシステムを構築する.

2.2.5 グラフィックカード

X window system はメモリを浪費し，セキュリティ的にも問題があるのでインストールしない．したがってグラフィックカードは特に何を選んでも問題にならない.

2.2.6 キーボード・マウス

高速で安定した環境を構築するために日本語環境の設定は行わないため，英語キーボードを用いる.
また X Window system もセキュリティやリソースの関係からインストールしないのでマウスは用いない.

2.3 KVM スイッチ

キーボード, モニタはサーバに１セット, 演算ノードでは８台兼用で１セット用いる. そのために corega 製 8 分岐 KVM（Keyboard, Video, Mouse）スイッチ CG-CKVM8P を使用する.

2.4 ネットワークハブ

ネットワークハブは MPI 通信時の通信性能に大きく影響し，演算性能に大きな影響を与えるので，スイッチングハブを用いる.
一般に使用されているスイッチングハブは全てのポートが同時にフルバンドでアクセスすることは想定していない.

そこで最も価格性能比の良いと思われるプラネックスコミュニケーションズ製の FHSW-1616NR と FXG-08TL をそれぞれ 16port Fast Ethernet スイッチングハブと 8port Gigabit Ethernet スイッチングハブとして用意する.

2.5 価格

最終的に Beowulf 型 PC クラスタを構築するために必要となったハードウェアの価格は Table Ⅲの通りである.

ファイルサーバ 1 台と演算ノード 8 台の構成で 100 万円を割る価格での構築が可能となる. 但し, 価格は本稿の執筆時点（2003 年 12 月）での購入価格（税込み）となっている.

Description	Quantity	Subtotal
File server (Dell)	1	90,090
Compute node (self-made)	8	90,177*8
16port Fast Ethernet SW	1	21,546
8port Gigabit Ethernet SW	1	56,511
KVM Switch	1	45,171
LAN cable	18	336*18
Total (¥)		941,038

Table Ⅲ Price List.

3.ソフトウェア

Beowulf 型 PC クラスタを構築するためには様々なソフトウェアが必要になる. そのいずれもオープンソースソフトウェアとして提供され, Beowulf 型 PC クラスタシステムの存在そのものを支えている.

3.1 オペレーティングシステム

各ノードには OS としてフリーな UNIX ライクな OS である Linux をインストールする. Linux には様々なディストリビューションがあるが, クラスタリングの実績の多い Red Hat Linux[8] を採用する.

3.2 並列演算ライブラリ

並列演算を行うためにはノード間で通信が行えなくてはならない. そのために並列演算ライブラリを用いる.

主な並列演算ライブラリには MPI と PVM がある.

MPI は 1994 年に当時まちまちであったプロセッサ間の通信関数を統一するために MPI Forum[9] によってリリースされた, C または Fortran から呼び出せるメッセージパッシングの関数群の仕様である. 多様な通信方式に対応し, 多くの計算システムに実装されている. また MPI の方が PVM よりも自由度が大きく様々なタイプの並列化のアルゴリズムを実装することができる[10].

そこで本稿では米国 Argonne National Laboratory (ANL)と Mississippi 州立大学との共同で開発された MPI のフリーの実装である MPICH[11] をインストールする.

4.構築手順

4.1 ハードウェアのセッティング

各演算ノードと 16port Fast Ethernet スイッチングハブ,および 8portGigabit Ethernet スイッチングハブを LAN ケーブルで接続する.

また演算ノードは 8 分岐 KVM スイッチを用いて 1 台のモニタとキーボードを分岐させてそれぞれ接続する.

ハードウェアのセッティングは Fig.2 のようになる.

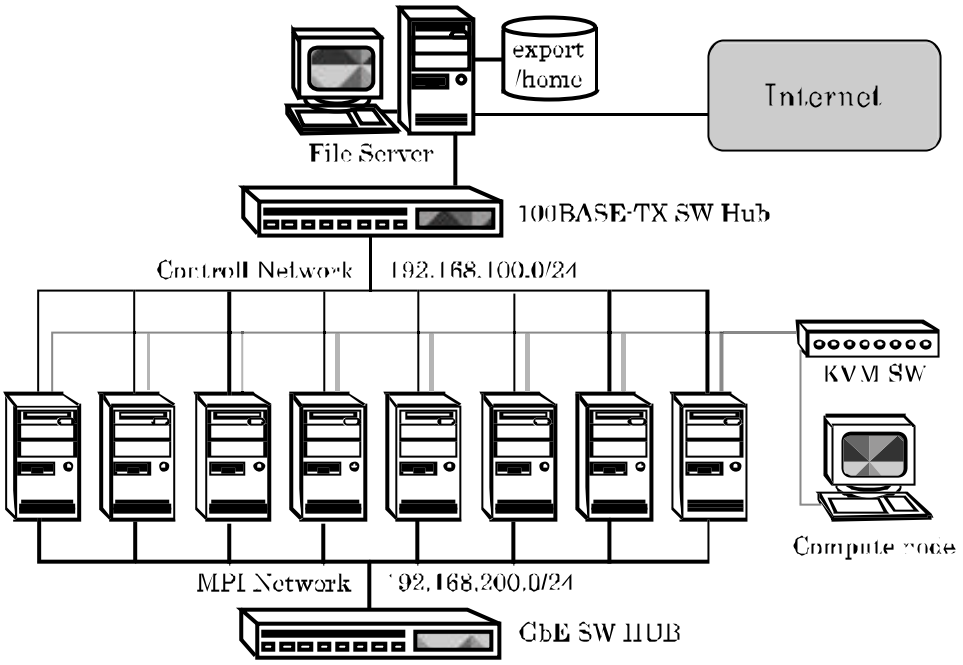

Fig. 2. Hardware Setting.

4.2 ネットワーク構成

クラスタ内にはコマンド/NFS 用に Fast Ethernet による fastlocal と MPI 用に Gigabit Ethernet による gigalocal という 2 つのローカルネットワークを用意する.

それぞれ Table Ⅳ, Table Ⅴのようになっている.

IP	FQDN
192.168.100.100	node0.fastlocal
192.168.100.101	node1.fastlocal
192.168.100.102	node2.fastlocal
192.168.100.103	node3.fastlocal
192.168.100.104	node4.fastlocal
192.168.100.105	node5.fastlocal
192.168.100.106	node6.fastlocal
192.168.100.107	node7.fastlocal
192.168.100.254	server.fastlocal

Table Ⅳ Network table for Fast Ethernet.

IP	FQDN
192.168.200.100	master.gigalocal
192.168.200.101	slave1.gigalocal
192.168.200.102	slave2.gigalocal
192.168.200.103	slave3.gigalocal
192.168.200.104	slave4.gigalocal
192.168.200.105	slave5.gigalocal
192.168.200.106	slave6.gigalocal
192.168.200.107	slave7.gigalocal

Table.Ⅴ Network table for Gigabit Ethernet.

このようにネットワークを 2 重化することによってネットワーク上のトラフィックの効率化が行え，最小のコストで最大限の演算効果が狙えるようになる.

4.3 ファイルサーバの構築

4.3.1 Red Hat Linux9 のインストール

まずは Red Hat Linux9 のインストールを行う.
インストール時に用いる言語及びシステムインストール言語は英語のみとし，マウスは no-mouse を選択する.
ネットワークデバイスの設定ではインターネット側のグローバル IP アドレスと fastlocal の 192.168.100.254/24 をそれぞれ設定する.
またセキュリティの設定では NFS を用いるために「no firewall」を選択する.
インストールタイプは Custom を選択し, Table VI のパッケージのみインストールする. 丸括弧で囲んだパッケージは依存関係により必要となるパッケージである.

Applications	Internet	(openssh) rsh
	Publishing	(groff)
System	Base	logrotate man
	Daemons	nfs-utils ntp openssh-server portmap tcp_wrappers
	Libraries	(libcap) (libstdc++)
Development	Language	(perl)
	Libraries	(perl-Filter)

Table VI Packages for File server

4.3.2 サービスの制御

/sbin/chkconfig を用いて不要なデーモン（kudzu, netfs, rawdevices, saslauthd, keytable, syslogd）の自動起動を停止し，reboot する.

4.3.3 TCP Wrapper によるアクセス制御

外部からアクセスがあった場合，TCP Wrapper はまず/etc/hosts.allow を読み込み，サービスごとに許可されたホストからの接続であった場合は接続が許可される.
そこで/etc/hosts.allow に ALL:127.0.0.1 及び sshd:ALL と portmap:192.168.100.を設定する.
/etc/hosts.allow で認証されなかった場合は/etc/hosts.deny を参照する．そこで/etc/hosts.deny に ALL:ALL を設定する.

4.3.4 ホストファイルの設定

マシン名による通信を行うため，Table IVを参考に/etc/hosts に演算ノードの fastlocal 側の IP，Fully Qualified Domain Name(FQDN)，ホスト名を記入する．

4.3.5 SSH サーバの設定

Telnet などの通常のリモートログインサービスでは，パスワードが平文のままネットワーク上を流れてしまい，セキュリティ上非常に危険な状態となってしまう．そこで SSH を用いることで全ての通信を暗号化し，パスワード等の情報の漏洩を防ぐことが可能になる．

まずは sshd の自動起動の設定を行い，続いてサービス制御スクリプトによってサービスを手動で起動する．

続いて/etc/ssh/sshd_config の設定を行う．root での SSH への接続を拒否するために PermitRootLogin で no を設定する．また telnet のように ID とパスワードによる認証を許可するために PasswordAuthentication で yes を設定する．

以上の設定が終わったら sshd を再起動する．

4.3.6 NFS サーバの設定

MPICH では各ノードの同じ階層に実行するプログラムが存在する必要がある．そこで NFS を用いてファイルサーバの/home を演算ノードに NFS マウントする．

NFS などの RPC サービスを提供する際は portmapper が起動していなくてはいけない．そこで portmap デーモンの自動起動の設定後，手動起動を行う．

続いて nfs, nfslock についても同様に自動起動の設定と手動起動を行う．

次に/etc/exports に fastlocal 内のマシンに/home を rw, sync, no_root_squash オプションでエクスポートするように設定する。no root squash オプションによって root ユーザーが NFS の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ようになる．設定後 # /usr/sbin/exportfs -a で/etc/exports にかかれた全ての内容をカーネルが保管するエクスポートテーブルに反映させ、最後に nfs を再起動する．

4.3.7 NTP サーバの設定

MPICH において make を使用する時，各ノード間でシステムクロックが同期されている必要がある．そこでファイルサーバを NTP サーバとして計算ノードを同期させる．

まずは ntpd の自動起動の設定を行い，続いてサービスを手動で起動する．

/etc/ntp.conf に参照する NTP サーバアドレスとして local clock を指す 127.127.1.0 を設定する．

このアドレスを用いることで，NTP サーバ自身の時刻のずれは補正できないが，演算ノードの時刻を同期させるための NTP サーバとして動作させることができる．

設定が終わったら ntpd を再起動し設定を反映させる．

4.4 マスター演算ノードの構築

4.4.1 ネットワークインストールの準備

演算ノードは NFS 経由でネットワークインストールを行う．そこでまずファイルサー

バ上の/images に 3 枚の Red Hat Linux 9 CD-ROM から Red Hat ディレクトリをコピーし，インストールツリーを作成する．

次に/etc/export に/images を fastlocal へ ro, sync, no_root_squash オプションで加え，/images をインストールプログラムからアクセスできるようにする．

最後に nfs を reload してそれぞれのシステムから/images を読み込み専用としてアクセスできるようにする．

次にファイルサーバに Red Hat Linux9 のインストールディスク 1 をマウントしてから /mnt/cdrom/images に移動し，フロッピードライブに新しいフロッピーディスクを挿入して # dd if=./bootdisk.img of=/dev/fd0 bs=1440k でブートディスクを作成する．作成できたら新しいフロッピーディスクと入れ替え，# dd if=./drvnet.img of=/dev/fd0 bs=1440k によってドライバディスクを作成する．

4.4.2 Red Hat Linux9 のインストール

ブートディスクをこれから構築するマスター演算ノードのフロッピードライブに挿入し，電源を入れる．言語やキーボードの設定後，Installation Method で NFS images を選択し，画面の指示に従いドライバディスクを読み込ませる．

TableⅣを参考に TCP/IP を設定し、NFS サーバ名には 192.168.100.254 を，RedHat ディレクトリには/images を設定する．

以上の操作でインストーラが立ち上がる．

Table Ⅳ，Table Ⅴの通りネットワークデバイスの設定を行う．但し MPICH では gigalocal を用いるため，ホスト名には必ず TableⅣを参考に設定する．

セキュリティの設定では rsh を用いるために「no firewall」を選択する．

インストールタイプは Custom を選択し，Table Ⅶのパッケージのみインストールする．丸括弧で囲んだパッケージは依存関係により必要となるパッケージである．

4.4.3 サービスの制御

不要なデーモン（kudzu, rawdevices, saslauthd, keytable, syslogd）の起動を停止し，reboot する．

4.4.4 TCP Wrapper によるアクセス制御

/etc/hosts.allow に ALL:127.0.0.1 ， in.rlogind:192.168.200. 192.168.100.254 ， in.rshd:192.168.200. 192.168.100.254 を設定し，/etc/hosts.deny に ALL:ALL を設定する．

4.4.5 ホストファイルの設定

/etc/hosts に各演算ノードの fastlocal の IP 及びホスト名とファイルサーバの IP 及びホスト名を設定する．

4.4.6 NFS クライアントの設定

/etc/fstab に NFS マウント情報を設定する．クラスタシステムで最良の効果を得るために読み書きバッファサイズが最大許容量(8192 バイト)になるようにマウントオプションとして rw,hard,intr,bg,rsize=8192,wsize=8192 を設定する．[12)]

netfs の自動起動の設定を行った後，サービスを手動で起動する．

reboot することによって，netfs によってファイルサーバの/home が自動的にマウントさ

れる.

4.4.7 rsh サーバの設定

MPICH を利用するにはパスワードの要求の無い rsh が必要になる．本来 rsh サーバはセキュリティ上問題があるために SSH などを用いるべきだが，SSH を用いると暗号化のオーバーヘッドが加わり rsh に比べ演算速度の低下が起こってしまう．[13)]

そこで本来はセキュリティ的に危険を伴う rsh だが，gigalocal 内とファイルサーバ演算ノード間においてのみサービスを提供する．

xinetd 経由で起動されるサービスに特に起動スクリプトは存在しないので rsh と rlogin の自動起動を設定後， xinetd を再起動する．

続いて/etc/xinetd.d/rsh 及び/etc/xinetd.d/rlogin の only_from にファイルサーバの IP アドレスと 192.168.200.0/24 を指定することで rsh，rlogin のアクセス制御を行う．

rsh サーバは TCP Wrapper による認証の後/etc/hosts.equiv と/root/.rhosts を参照する．

サーバはまず/etc/hosts.equiv を参照し，このファイルに記述されているホストの一般ユーザーに対しパスワード要求をせずに接続を許可する．したがって全ての演算ノードの/etc/hosts.equiv に全ての演算ノードの gigalocal のホスト名とファイルサーバのホスト名を記述しておく。

但し初期の状態では root は rsh を利用できない．そこで/etc/securetty の最後に rsh と rlogin を追加することで root のアクセスを許可する．

ところがこのままでは root で r コマンドを用いるときにはパスワードを要求されてしまうが，rcp ではパスワード要求のない接続ができることが前提となっている．

そこで/root/.rhosts にパスワード要求なしで root での接続を許可するクライアントのホスト名を書き，パーミッションを 600 に設定する．

4.4.8 NTP クライアントの設定

まずは NTP サーバと同様に ntpd の自動起動の設定を行い，サービスを手動で起動する．

続いて/etc/ntp.conf の server オプションにファイルサーバの IP アドレス 192.168.100.254 を設定する．

設定が終わったら ntpd を再起動し設定を反映させる．

4.4.9 MPICH の設定

MPICH のインストールを行う．ANL の MPI のサイト [11)] から最新版の Linux 用の MPICH (mpich.tar.gz)をダウンロードし/usr/local/src/mpich にコピーし，# tar zxvf mpich.tar.gz で解凍する．本稿の執筆時点での最新版は mpich-1.2.5.2 であるので，以後このバージョンを元に話を進める．

解凍したことで./mpich-1.2.5.2/ができるので，そこへ移動し# ./configure -prefix=/usr/local/mpich-1.2.5.2 を実行する．-prefix オプションはインストール先ディレクトリの指定である．configure によって Makefile が生成されるので make を実行し，make が完了したら# make install でインストールを行う．

次に machine ファイルに MPICH で使用するノードと，使用する順番を登録する．デフォルトでは machine ファイルとして/usr/local/mpich-1.2.5.2/share/machines.LINUX が参照されるので，このファイルに並列計算に参加するノードのホスト名を登録する．

最後に MPICH の mpicc，mpif77，mpirun などのスクリプトへパスを通すため，各

ユーザーの~/.bash_profile に PATH=$PATH:/usr/local/mpich-1.2.5.2/bin を追加する.

4.5 スレーブ演算ノードの構築

4.5.1 キックスタートインストールの準備

スレーブ演算ノードへの Red Hat Linux のインストールはキックスタートインストールによって行う．キックスタートはデフォルトで Red Hat Linux に備わっている機能で，これによって Red Hat Linux のインストールの大部分を自動化する事ができるようになる.

まずマスター演算ノードをインストールしたときに作成されたキックスタートファイル/root/anaconda-ks.cfg をファイルサーバ上の/ks/にコピーし, install の下の行に nfs --server 192.168.100.254 --dir /images を追加し，network の ip と hostname を Table Ⅳと Table Ⅴを参考に変更して ks.cfg として保存する.

次にファイルサーバに Red Hat Linux9 のインストールディスク 1 をマウントしてから /mnt/cdrom/images に移動し，フロッピードライブに新しいフロッピーディスクを挿入して # dd if=./bootdisk.img of=/dev/fd0 bs=1440k を実行してブートディスクを作成し，# /ks/ks.cfg a:によってキックスタートファイルを追記する.

4.5.2 Red Hat Linux9 のインストール

ブートディスクをこれから構築するスレーブ演算ノードのフロッピードライブに挿入し，電源を入れる．Boot プロンプトで boot: linux ks=floppy dd を指定する.

後は画面の指示に従い，ドライバディスクを読み込ませると，キックスタートインストールが始まる．パーティションの設定画面のみ表示されるのでそれのみ行うと，あとは自動的にインストールが行われる.

4.5.3 サービスの制御

不要なデーモン（kudzu, rawdevices, saslauthd, keytable, syslogd）の起動を停止し reboot する.

4.5.4 設定ファイルのコピー

スレーブ演算ノードは MPICH をインストールしないことを除いてはマスター演算ノードと全く同じ構成である．したがって rcp を用いて/etc/hosts，/etc/hosts.allow, /etc/hosts.deny, /etc/fstab, /root/.rhosts, /etc/hosts.equiv, /etc/securetty, /etc/xinetd.d/rsh, /etc/xinetd.d/rlogin，/etc/ntp.conf を 192.168.200.100 からコピーする.

続いて rsh，rlogin，ntpd の自動起動を設定してから，reboot する．これによって rsh, rlogin，ntp は有効化され，/etc/fstab によって/home が NFS マウントされる.

5.運用

5.1 ユーザー管理

並列演算を行うためにはノード間でユーザーID が一致していなくてはならない．そのためには Network Information Service (NIS)を用いるという方法があり一部のクラスタシステムにおいて実際に用いられている.ところが NIS は MPI を用いた演算などにおいて，ホスト名の解決が要求されるたびに余計なネットワークのトラフィックを生み，演算性

能の低下を招いてしまう [12]．

そこでファイルサーバのユーザファイル（/etc/passwd, /etc/group, /etc/shadow）を全演算ノードへ rcp でコピーするシェルスクリプトをファイルサーバの/sbin に用意し，ファイルサーバでのユーザーの追加・削除時やパスワードの変更時に実行する方法を用いる．

5.2 ベンチマーク

構築した PC クラスタの演算性能を評価するために，ベンチマークを行う．並列演算ベンチマークとしては ScaLAPACK[14] や NAS Parallel Benchmarks[15] も有名だが，特に流体計算結果に比較的近い結果を出すと言われている国産の姫野ベンチ [16] を用いる．

5.2.1 姫野ベンチ

姫野ベンチは理化学研究所情報基盤センター室長の姫野龍太郎氏が開発したもので，非圧縮流体解析コードの性能評価のためのベンチマークソフトであり，ポアソン方程式をヤコビの反復法で解く場合の主要なループの処理速度を計る．ScaLAPACK や NAS Parallel Benchmarks では多くの計測時間が必要であるのに対し，ソースコードが非常に短いため短時間で実測速度を求めることが出来る．

5.2.2 ベンチマーク

ベンチマークは一般ユーザーで行う．まずは一般ユーザーのホームディレクトリにベンチマーク用の適当なディレクトリを作成し，そこへ姫野ベンチのサイト [16] からダウンロードした C+MPI 用の姫野ベンチマークプログラム cc_himenoBMTxp_mpi.lzh を保存する．

続いて cc_himenoBMTxp_mpi.lzh を解凍し，paramset.sh を用いて./paramset.sh S 2 1 1 のようにパラメータの初期化を行う．但し paramset.sh の第１引数は計算サイズで，XS，S，M，L の４通りから選ぶことができる．また第２から第４引数はそれぞれ分割数に対応していて，これらを全てかけた値がプロセッサ数と等しくなるように設定する．これによって param.h が更新される．

続いて Makefile のサンプルとして Makefile.sample があるのでこれを用いて Makefile を作成し make する．

最後に$ mpirun -np 2 ./bmt のようにベンチマークを実行する．mpirun の–np に続く数字は使用するプロセッサ数である．

ベンチマークを繰り返すときは./paramset.sh を実行後，$ make clean を実行して himenoBMTxps.o を削除してから make して実行する．

5.2.3 ベンチマーク結果

汎用のスーパーコンピュータとの性能比較をするために計算サイズとして S を用いる．

Fig.4 は使用したノード数と実測速度の関係を表したものである．1CPU の場合で 409.51MFLOPS，4CPU の場合で 1471.06MFLOPS，8CPU の場合で 2315.68MFLOPS と言う実測速度が出た．

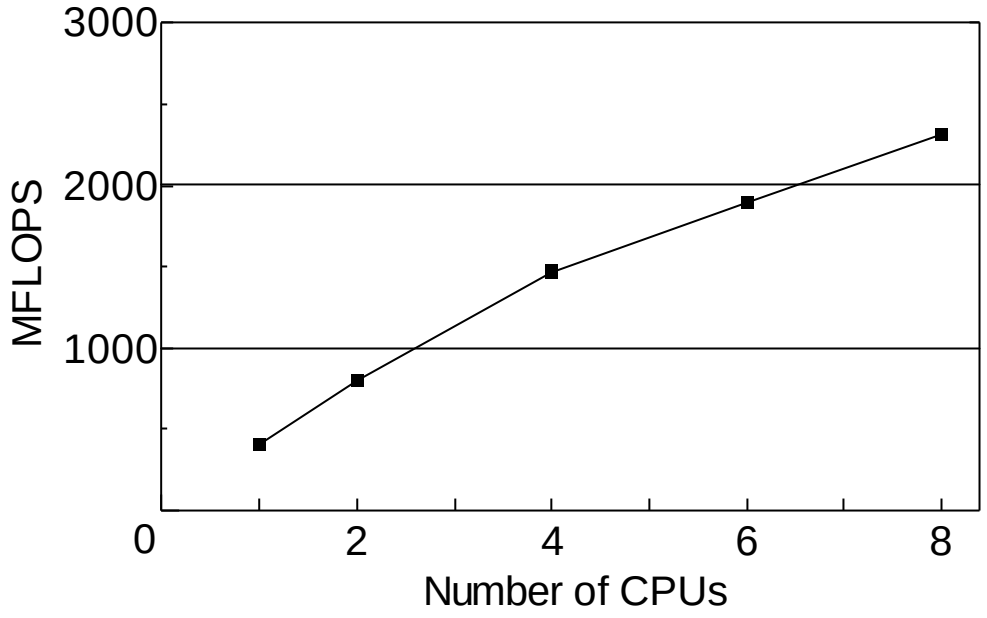

Fig. 4. HIMENO Benchmark.

並列演算においてはノード数が多くなると，ノード間通信速度がボトルネックとなり，ノード数の増加の割に実測速度が出ないが，Gigabit Ethernet を MPI 専用としたことで，ノード数の増加に伴いほぼ線形に性能が増加した．

また Table Ⅷに 2003 年 11 月 25 日に更新された姫野ベンチの実測結果[16]の一部と比較した．

本稿で構築した Beowulf 型 PC クラスタは 8 ノードで 2315.68MFLOPS を示したので，SGI の CRAY-T3E 64 ノードや Origin2000 の 8 ノードとほぼ同様の性能が出ていることがわかる．また，おそらくノード数を 16 ノードに増やせば，NEC のスーパーコンピュータ SX-5 以上の性能が出ることも予想できる．

以上のベンチマークによって Beowulf型PC クラスタの圧倒的なコストパフォーマンスが実証できた．

6. まとめ

PC をネットワークで接続し，Linux をインストールし，ほんの少しの設定を行うだけで，スーパーコンピュータ級の計算環境が手に入る．

PC パーツの性能が飛躍的に進歩した今日，そんな夢のような話が現実の物として目の前にあるのである．

そこで本稿によって一つでも多くの研究室においてBeowulf型PCクラスタが稼働し，研究室レベルでもスーパーコンピュータ級の高速な実験環境が導入可能であることを実感して頂きたいと思う．

リファレンス

1) 水谷由宏・岡村秀樹: 分光研究 **50,** 167-170 (2001).
2) 三宅川弘明・水谷由宏: 分光研究 **52,** 239-240 (2003).

3) **水谷由宏・三宅川弘明・岡村秀樹**: **分光研究 52,** 281-285 (2003).
4) http://www.beowulf.org/beowulf/history.html
5) http://www.csm.ornl.gov/pvm/pvm_home.html
6) http://www-unix.mcs.anl.gov/mpi/
7) http://www.top500.org/list/2003/11/
8) http://www.redhat.com/
9) http://www.mpi-forum.org/
10) Douglas Eadline: *Cluster Quick Start* (GNU GENERAL PUBLIC LICENSE Version 2, 1999)
11) http://www-unix.mcs.anl.gov/mpi/mpich/
12) Thomas L. Sterling, Donald J. Becker, John Salmon, Daniel F. Savarese: *How to build a Beowulf* (MIT Press, Cambridge, 1999)
13) Nathan Carstens: *BUILDING A BEOWULF CLUSTER* (http://echelon1.mit.edu/paper.pdf, 2003)
14) http://www.netlib.org/scalapack/
15) http://www.nas.nasa.gov/Software/NPB/
16) http://w3cic.riken.go.jp/HPC/HimenoBMT/