Vine Linux による PC Cluster の構築
Version 2

幸谷智紀
tkouya@na-net.ornl.gov

2004 年 2 月 28 日 (土)

1 初めに

本文書は，Intel Pentium IV(IA-32) を搭載した PC11 台に Vine 2.6r3[13](追記参照) を導入し，PC Cluster(cs-pccluster2) として使用するためのネットワーク及びソフトウェア環境整備の手順を述べたものである。

著者は昨年 (2003 年) にも，Pentium III/Celeron ベースの PC Cluster(cs-pccluster) を構築し，その手順を文書 [1] にまとめ，公開した。どういう訳か，毎月のべ 500 アクセス程度，参照されているようである[1]。著者は根が単純なので嬉しくなり，今回の cs-pccluster2 構築に際してその文書のアップデートを行うことにした，というのが本文書成立の一番の理由である。但し，歴史的な資料ぐらいにはなると思われるので，前の文書も引き続き公開する。重複する記述は多いが，これはこれで閉じた資料にするためである。その点はお許し頂きたい。

前の文書と同様，以降の記述に基づいて読者が PC Cluster を構築し，何らかの不利益を被ったとしても[2]，著者は一切関知しない。全ては自身の責任で行って頂きたい。

You must do at your own risk!

本文書に関して，bug 報告やご意見などあれば，タイトル部にあるメールアドレスまでご一報頂ければ幸いである。

追記

Vine 2.6r3 にはファイルシステムの不具合 [14] が見つかり，現在 (2004 年 2 月 28 日現在) は 2.6r4 を使用することが推奨されている。以降で述べるインストール作業は 2.6r3 ベースで実行したものであるので，2.6r4 では少し異なる点があるかもしれない。その点，了承されたい[3]。

[1] ちなみに，2004 年 1 月の著者のサイト (http://na-inet.jp/) における全ヒット数は約 37000。一日単位では 1300 ヒット程度である。
[2] 大層なことは起こらないとは思うが，念のため。
[3] ちなみに著者の PC Cluster は手作業で修正作業を行い，今も 2.6r3 ベースのシステムとして動作中である。

表 1: cs-pccluster のハードウェア

	親: cs-southpole 子 1: cs-room443-b01〜b04	子 2: cs-room443-s01〜s04
CPU	Pentium III 1GHz Coppermine	Celeron 1GHz
L1 Cache (KB)	16	16/16(L1 D)
L2 Cache (KB)	256	256
RAM(MB)	512	128
IDE HDD(GB)	40	40
100BASE-TX NIC	3COM 3c905	RealTek RTL-8139B
100BASE L2 SW	ACCTON ES3016A	

2 cs-pccluster2 に至る経緯と雑感

昨年 cs-pccluster を作ったのに，何故今年も作るのか？ 一言で言えば，お金がついたからである。職場 (静岡理工科大学) に,「あれもこれもこんなことも出来る上に経費削減につながる画期的な研究をしたいからお金を下さい」と言ったら学内研究費を支給して頂けたので作ったのである。ここで関係者に深く感謝いたします。

作るに際しては，予算で購入できる範囲のハードウェアを購入し，ソフトウェアはタダで済ませるという方針を立て，その通り実行した。パーツごとに購入した PC の組み立てに際して動員したウチの学生さんの労賃と，ソフトウェアやネットワークの設定に費やされた私自身の賃金は考えないことにした。

… と，これではあらぬ誤解を招きそうなので，もう少し詳しくこの経緯を述べる。以降の節と異なり，この部分はかなり主観と感情と諦観と偏見が混入しているので，眉を唾液でベトベトとにしながら読み進められたい。

2.1 何故，cs-pccluster2 を作ろうと考えたのか？

2.1.1 ネットワーク構成についての問題

まず，昨年に構築した cs-pccluster のスペック (表 1) と，ネットワーク構成 (図 1) をご覧頂きたい。

このような構成の PC Cluster の上で，MPI[9] と BNCpack[4] を組み合わせて並列分散多倍長数値計算ライブラリ MPIBNCpack[5] を開発し，Linux Conferernce 2003 や FIT2003 でお披露目した。どちらも査読付で,「オリジナリティに欠ける研究である」という，至極真っ当な評価を頂いた。

ことに FIT2003 では座長からは，Pentium III と Celeron とを混在させている環境について,「同じハードウェアというのは無理があるのでは？」というコメントが寄せられた。数値計算によるベンチマークテストを行った限りでは有意な差を見つけられず，無視してもよいコメントではあるが，気にはなっていた。

それよりも，cs-pccluster は一つの Ethernet を MPI と NIS/NFS で共有する構成になっており，通信に負荷のかかるプログラムを実行させた時に悪影響が出るのではないかということが，構築当時から引っかかっていた。2 重化する試みはあったのだが，ある事情[4]により頓挫したままになってい

[4] 自分が使いたい環境は自分で作る以外にない，という教訓を得た。

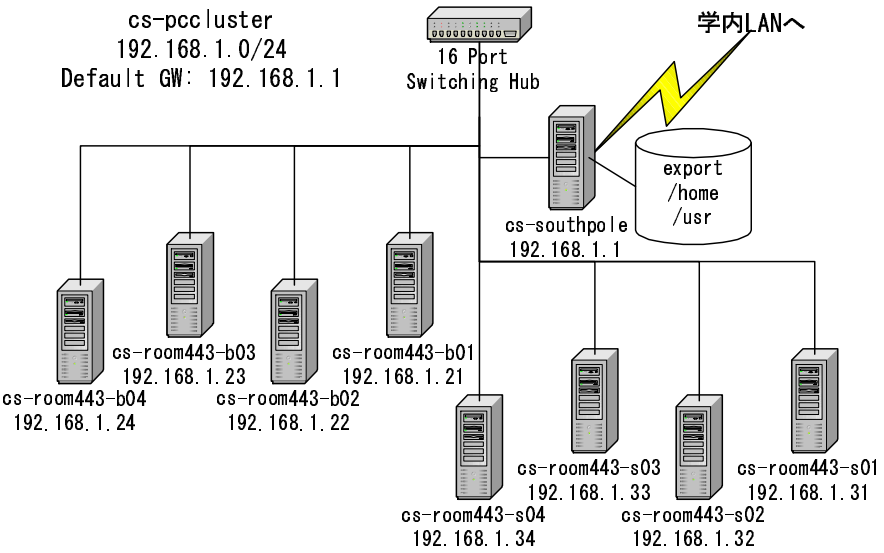

図 1: cs-pccluster

たのである。MPI が使用するネットワークは，他のサービスが使用するネットワークとは分離すべきである，という指摘は以前からなされている [7]。とはいえ，実際にやるとなると難しくはないが面倒である。SCore を紹介した書籍 [8] でも，例として提示されているのはネットワークが一つしかない構成である。

しかし，2003 年度の卒業研究で，ある学生さんに Romberg 積分を実行させたところ，2CPU 以上を使用した場合，0.1 秒程度の実行時間のずれが発生することが判明した。これがネットワークを共有しているために起こったのか，単なる誤差なのかは，まだ解析していないので判然としない。が，実行時間が遅延する事由は極力減らすべきである。

2.1.2 教育に関する問題

それと平行して，cs-pccluster のハードウェアを設置している実験室の環境を更新する必要も出てきた。最近はどこの大学でも，教育に力を注ぐべきであるという雰囲気になっており，よほど場所と予算に余裕のある研究室でない限り，研究のためだけに設備を維持することは許されない。cs-pccluster も，並列計算とは関係のない学部生の実験にも兼用してきたハードウェアを使っており，Ethernet と TCP/IP で LAN を構築してネットワーク性能を計測する実習を行ってきた。学生募集活動においては，高校生向けの実験講座を受け持ち，これも cs-pccluster を使わざるを得ない上，自分の研究紹介をデモする機会も多く，否が応でも使用頻度が増えていく。更に，大学院生に対しては MPI プログラミングを教えることになり，自分の研究以外でも常に PC Cluster として活用する必要が出てきたのである。

こうなると，現状の cs-pccluster ではいろいろと不満が出てくる。まず，ネットワークが遅すぎる。1000BASE-T の PCI カードが数千円で購入できるご時勢に，今更 100BASE でもあるまい。必ずしも “The newest is the best”ではないだろうが，IT は新しいものに触れることが常に必要な分野である。購入可能な範囲で実験環境を更新し続けていくことが，それを使う学生さんにとって有用である。

1000BASE を導入するとなると，32bit PCI ではちと心もとないので，PC 本体も更新する必要がある。とはいえ，IA-64 環境はまだ高価すぎる上，それに対応した OS も限られる。安価で広く使用されている IA-32 で，1000BASE にふさわしい CSA 接続をサポートした Chipset を備えた PC に更新すべきである。

2.1.3 学内 Web サーバの問題

広いネットワーク帯域が望まれるのは，主として動画を流す必要があるからである。リアルタイムに動画を再生するだけなら，現状の PC でもなんとかなるが，それを自在に編集してマルチメディアコンテンツを作成するとなると，最低でも 1G byte の RAM と，より高速な Pentium IV 以上の CPU パワーが必要である。2003 年からは e-Learning にも手を染めるようになっており，卒業研究と絡めてより強力な PC が必要な状況が生まれつつあった。

著者は学科内のメールサーバと Web サーバ (cs-www) の管理も引き受けている。前者はさほどではないが，後者は Server side で実行される CGI を多用したコンテンツを大量に保持しており，負荷はかなり高い。現状では Pentium IV マシンを使っているが，本格的に e-Learning 用として使用するには，fail over 機能を持ち，少なくとも 1000 人程度の同時接続にも耐えられるような load balancing 機能を持った，Web cluster へアップグレードする必要がある。今すぐに実行する必要はないが，そのような環境を実現するための試験は行っていかねばならない。

2.1.4 cs-pccluster2 の構成

このような研究と教育の両面から更新の必要に迫られ，やむを得ず[5]新たな PC Cluster である，cs-pccluster2 を構築しようと計画した。必要な要件は

1. PC Cluster においてはネットワークを 2 重化
2. ネットワークは 1000BASE を使用
3. PC は Pentium IV 以上の CPU を使用
4. 既存の Web サーバを生かした Web cluster の実験環境

である。その結果，図 2 に示すような構成になったのである。

cs-minerva, cs-hera, cs-www の 3 台は既存のマシンである。これらは，新たに加えられる cs-northpole と併せて，Web cluster として動作する予定である。ハードウェアの Load Balancer は高価過ぎて手が出ないので，free なソフトウェア実装である PLB(Pure Load Balancer)[15] を用いて実験を行っている。作りはまだ未成熟な点が見受けられるので，実用に供するには堅実な実験を積み重ねる必要があると思われる。

cs-pccluster2 として動作するのは cs-northpole と，cs-room443-01〜10 までの 11 台である。これらは 1000BASE(太線, 192.168.2.0/24) と 100BASE(細線, 192.168.1.0/24) の 2 重ネットワーク構成になっており，前者は MPI で利用し，後者は既存の 3 サーバと同じネットワークに属し，MPI 以外のネットワークサービスで利用する。もちろん，ちょっとした設定変更で，Web cluster 群も巻き込んだ 1000BASE, 100BASE 混在の hetero 環境にすることもできる。従って，cs-pccluster2 は homo cluster と hetero cluster の両方の実験が可能である。

[5] 自分の楽しみがかなりの割合で混入していることは否定しない。

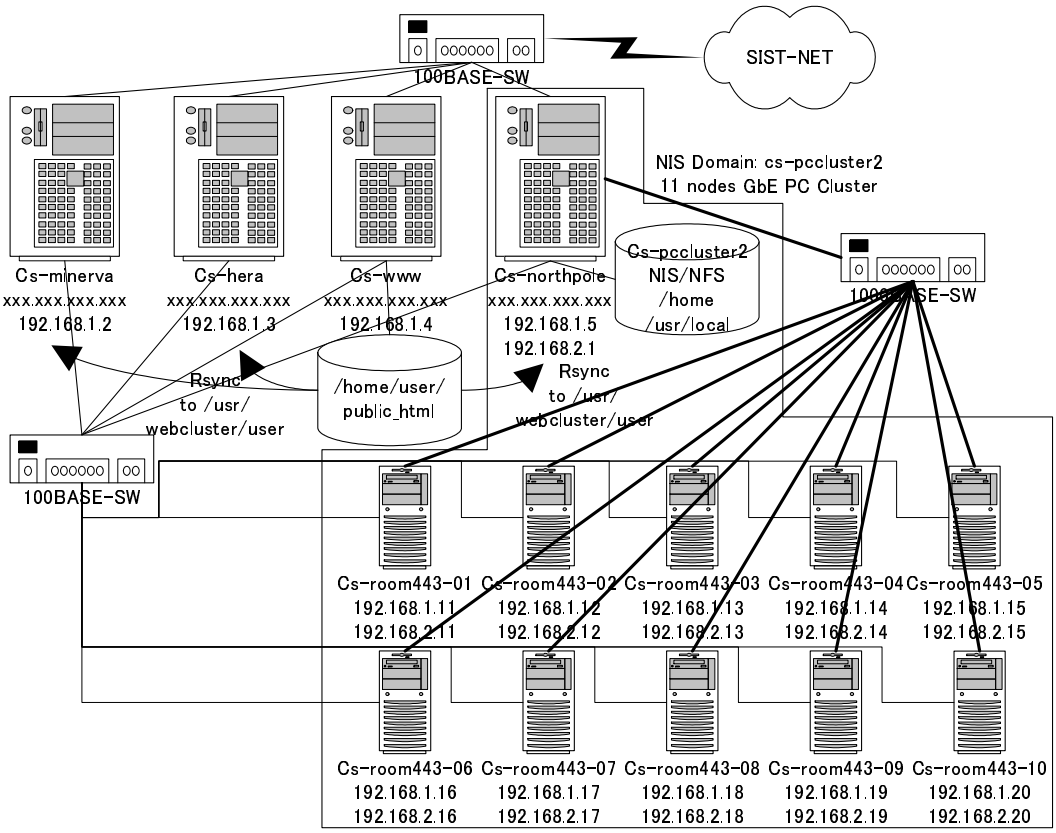

図 2: cs-pccluster2

表 2: cs-pccluster2 のハードウェア

	親: cs-northpole	子: cs-room443-01〜10
CPU	Pentium IV 2.8C GHz	
Mother Boad	Gigabyte GA-8IPE1000 Pro2	
L2 Cache (KB)	512	
RAM(MB)	1024	512
SATA IDE HDD(GB)	80(ST380013AS)	
100BASE-TX NIC	Intel Pro/100, 3COM 3c905	3COM 3c905
100BASE SW	Allied-Telesis CenterCom 8124XL	
1000BASE-T	Intel Kenai II(CSA)	
1000BASE L2 SW	Planex FXG-16TX	

当初は，cs-room443-xx マシンも学内ネットに直接接続し，全て Web cluster 兼用にしようと考えた。しかし，既存マシンが Web cluster として十分活用できる上，実験用マシンを外に出すと，実習中に Web 閲覧にうつつを抜かす輩が出ないとも限らない[6]。外部への接続を教師側が容易に切断できるのが望ましい。また，シビアなベンチマークテストの際には，外部のトラフィックの影響を極力避けたいという事情もある。よって，cs-pccluster 同様，cs-northpole 以外は内部 LAN にのみ接続する形態を取ることにした。

最後に，今回追加した PC のスペックを表 1 に示す。CSA 接続の 1000BASE-T をサポートする安価な Mother Board の登場を待ち，更に CPU の値下げのタイミングを図って，予算ギリギリで購入可能な内容となった。なお，Graphics カード，100BASE-TX カードは既存のものを使用した。最大 10 人で同時に実習が出来るよう，CPU スイッチは使用せず，液晶ディスプレイ，キーボード，マウスは各 PC 毎に取り付けてある。cs-pccluster はこれを怠ったため，実習ごとに取り外したり取り付けたりして面倒なことこの上なかったからである。

これによって，当初目指した「あれ (Web cluster) もこれ (PC Cluster) もこんなこと (多様な学生向け実験や hetero/home cluster を使った研究) も出来る上に経費削減 (一つのハードウェア群が複数の用途で利用可能) につながる画期的 (?[7]) な研究」が実現可能になったのである。

3 セットアップ手順

図 2 において，新たに導入した PC のセットアップ手順は以下の通りである。大筋は cs-pccluster[1] の手順と変わりないが，実施した日時を最後に付してある。

1. [4 節] PC の組み立てと動作チェック…2003 年 12 月 17 日 (水)
2. [5 節] Vine Linux のインストールと TCP/IP の設定…2004 年 1 月下旬
3. [6 節] NFS/NIS Server におけるソフトウェアのインストール…1 月 27 日 (火)
   (a) 最新の gcc/g++/g77

[6] というより，今までの経験上，確実に出現する。

[7] ネットワーク管理の経験があり，そこそこプログラミングが可能なスキルがあれば，この程度の仕様を実現するのは難しくはない。「画期的」という形容詞が誇大広告的なのは，著者も承知している。すいません。

(b) BLAS(+ATLAS)

(c) LAPACK

(d) GMP(+MPFR), BNCpack

(e) JDK, Omni

4. [7 節] NFS/NIS Server/Clients の設定
5. [8 節] mpich の設定
6. [9 節] 並列分散数値計算ソフトウェアのインストール

(a) MPIBNCpack

以降の節では各手順を解説する。

4 PC の組み立てと動作チェック

PC は最初からパーツ単位で購入することにしていた。デスクトップ PC が低価格化している現在，自分で組むよりも，大量に供給されているショップブランドやメーカー製の完成品の方がコストパフォーマンスが良いことも多い。しかし今回は，グラフィックスカードやキーボード・マウスは既存のものを使い，なおかつ CPU や RAM，Ethernet はなるべく高性能のものが必要だったため，必要なパーツのみ購入することとした。逆に，PC ケースや HDD は極力安いものを選択した。

部材の発注は 12 月に入ってから行った。CSA 接続の 1000BASE-T を搭載している，それでいて安価な Mother board の登場を待ち，なおかつ恒例行事となっている Intel CPU の値下げ (これは PC Watch[6] の記事を定期的にチェックしていれば分かる) のタイミングを見計らっていたからである。

全てのパーツが揃ったのは 12 月の半ばに入ってからである。手本となる 1 台と，cs-northpole の 2 台分のみ著者が組み立て，それ以外のマシンは，2003 年最後のゼミがあった 12 月 17 日 (水) に当ゼミ生 (3 年生) に組み立ててもらった (図 3)。

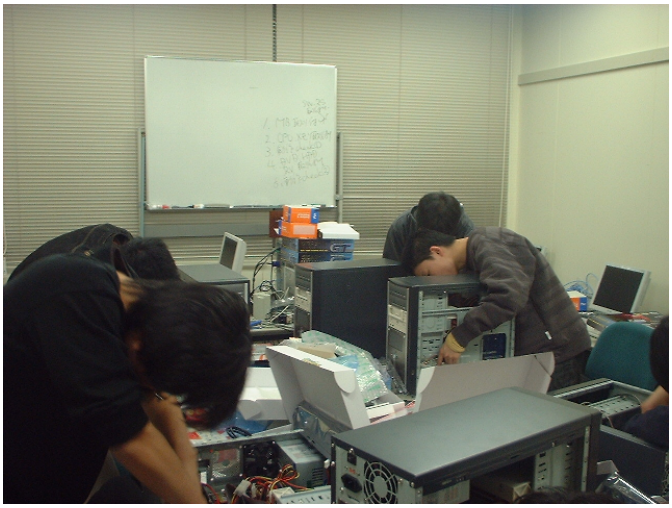

図 3: PC 組み立て作業の様子

当学科では PC の組み立てを実験実習で行っており，全くの初心者はおらず，割とスムーズに組立作業が完了した。殆どの学生さんが 90 分程度の時間で，電源チェックまでたどり着いた。

が，後に OS のインストール作業をする段になって，幾つかのマシンの組み立てに問題があることが分かった。具体的には

- SATA IDE HDD, IDE DVD-ROM の Master/Slave 位置がバラバラ
- Mother Board と PC ケースがきちんと固定されていない (ネジ穴が余っている，ミリネジ穴に無理やりインチネジをねじ込んでいる等)
- 電源スイッチや HDD アクセスランプの+/-が逆

という不具合があった。もっともこれらを修正するのは，一から組み立てるよりはずっと容易な作業で，1 時間もかからずに，OS をインストールしながら修正できた。

改めて，お手伝い頂いた学生さんたちに感謝申し上げたい。

5 Vine Linux のインストールと TCP/IP 設定

5.1 インストール作業の手順

Vine 2.6r3 のインストールについては手順のみを簡単に記す。

1. Ring サーバ [12] から適切な Mirror を探し，Vine 2.6r3 の CD イメージをダウンロードして Install CD を作成する。CD イメージの MD5 チェックは

   ```
   $ md5sum -c MD5SUM.Vine26r3
   Vine26r3-SRPMS.iso: OK
   Vine26r3-i386.iso: OK
   ```

   のように行う。OK の表示が出ることを確認する。

2. 各マシンにインストールする。表 2 構成では問題なくインストールでき，SMP kernel が設定され，On board の 1000BASE-T(eth0) も認識した。

3. boot して動作することを確認したら電源を切り，100BASE-TX の PCI カードを増設する。

4. 再び boot して，自動的に増設したカードを認識することを確認し，eth1 に TCP/IP 設定を行う。

5. 起動後，eth0, eth1 共に通信できることを確認する。

6. kernel を 2.4.24 にする。

7. /etc/hosts に自分の NIC のエントリと，NIS/NFS サーバのエントリ

   ```
   192.168.1.5             cs-northpole-nis
   ```

   を追加しておく。

8. /etc/inetd.conf を編集し，telnet, ftp を有効にして inetd を再起動する。

まず，NIS/NFS サーバ (以下，親マシンもしくは親と略記する) に上記手順にて Vine をインストールし，更にもう一枚 Intel Pro/100 の PCI カード (eth2) を追加して，学内 LAN に接続する。その後，NIS/NFS Client になるマシン (子マシンもしくは子と略記) マシンを 1 台，上記手順に従ってインストール。親との通信を確認し，9 節までの作業とテストが完了してから，他の子マシンを設定する。もしくは，全ての子マシンに Vine のみインストールして eth0 のみ使用可能にしておき，それから eth1 を追加した子マシンを構築して一台ずつ加えていくのもよい。

最初から全ての子マシンを巻き込んで作業すると，トラブルの原因が掴みづらくなる。小さい PC Cluster から始めて，徐々に大きくして要領で作っていくこと。

5.2 Vine 2.6r3 にまつわる問題

5.2.1 SATA IDE HDD のシーク速度が遅い

Kernel を 2.4.24 へ update したのは，SATA HDD のシークスピードがやたらに遅かったからである。きっちり 10^9bytes のファイルをローカル HDD(SATA 80GB) 内で cp すると，最初 6 分以上もかかっていた。試し hdparm コマンドを動かしてみると

```
[root@cs-room443-02 root]# /sbin/hdparm -t /dev/hdc

/dev/hdc:
 Timing buffered disk reads:  64 MB in 19.04 seconds =  3.36 MB/sec
```

となって，3MB/sec しか出ていないことが分かる。Fedora Core 1 を突っ込んだ別のマシンで試すと，数年前の 20GB ATA にもかかわらず，1 分程度で終了する。

Vine をインストールした直後の状態は

```
[root@cs-room443-02 root]# /sbin/hdparm /dev/hdc

/dev/hdc:
 multcount    = 16 (on)
 I/O support  =  0 (default 16-bit)
 unmaskirq    =  0 (off)
 using_dma    =  0 (off)
 keepsettings =  0 (off)
 nowerr       =  0 (off)
 readonly     =  0 (off)
 readahead    =  8 (on)
 geometry     = 9729/255/63, sectors = 156301488, start = 0
```

である。

結局，kernel を 2.4.24 にしてみたらあっさり解決し，

```
[root@cs-northpole root]# /sbin/hdparm -t /dev/hdc

/dev/hdc:
 Timing buffered disk reads:  64 MB in  1.16 seconds = 55.17 MB/sec
```

というレベルまで改善し，同様に cp コマンドを試すと，40 秒程で終了するようになった。これだと/sbin/hdparm -t の結果，55MB/sec とほぼ一致する。

Kernel を update した後の状態は

```
[root@cs-northpole ATLAS]# /sbin/hdparm /dev/hdc

/dev/hdc:
 multcount    = 16 (on)
 I/O support  =  0 (default 16-bit)
 unmaskirq    =  0 (off)
 using_dma    =  1 (on)
 keepsettings =  0 (off)
 nowerr       =  0 (off)
 readonly     =  0 (off)
 readahead    =  8 (on)
 geometry     = 9729/255/63, sectors = 156301488, start = 0
```

となっていた。つまり DMA モードが on になっていないのが原因であるらしい。しからばと，DMA モードを変更しようにも

```
[root@cs-room443-02 tkouya]# /sbin/hdparm -d1 /dev/hdc

/dev/hdc:
 setting using_dma to 1 (on)
 HDIO_SET_DMA failed: Operation not permitted
 using_dma    =  0 (off)
```

となって，弾かれてしまう。Fedora も Vine も kernel 2.4.22 系列なのだが，いったいどこが悪いのか，不明なままである。

5.2.2 GCC が古い

gcc が 2.95 系列だったのと，g77 が入っていなかったので gcc-3.3.2 を tar.gz からビルドしてインストールした。

RPM 取ってくるとか，自分でつくるとか，apt-get するとか，もうちっとスマートな方法も考えたが，面倒なので思考停止することにした。g77 が入っていないと，LAPACK/ScaLAPACK のインストールが出来ないのである。

なお，kernel コンパイルは，gcc-3.3.2 でも問題なく実行できた。

6 NFS/NIS Server におけるソフトウェアのインストール

各子マシンで共通に必要となるソフトウェアは，親マシンの/usr/local 以下に配置し，これを NFS で共有して管理の手間を省くことにする。但し，これを実行するには親と子が同じ CPU 及び OS でなければならない。異機種環境でアプリケーションソフトウェアを NFS 共有する際には，機種ごとにディレクトリ配置を検討する必要がある。

6.1 最新の gcc/g++/g77

gcc-3.3.2.tar.gz を Ring サーバからダウンロードした後の実行手順は以下の通りである。出力結果は冗長になるので略してある。

```
[tkouya@cs-northpole cluster]$ tar zxvf gcc-3.3.2.tar.gz
(略)
[tkouya@cs-northpole gcc-3.3.2]$ ./configure --prefix=/usr/local
(略)
[tkouya@cs-northpole gcc-3.3.2]$ make
(略)
[tkouya@cs-northpole gcc-3.3.2]$ su
[tkouya@cs-northpole gcc-3.3.2]# make install
(略)
[tkouya@cs-northpole gcc-3.3.2]# exit
```

g77 コマンド (Fortran77 コンパイラ) が使用できることを確認する。

```
[tkouya@cs-northpole gcc-3.3.2]$ which g77
/usr/local/bin/g77
[tkouya@cs-northpole gcc-3.3.2]$ cat > test.f
        write(6, *) sqrt(2.0)
        stop
        end

[1]+  Stopped                 cat >test.f
[tkouya@cs-northpole gcc-3.3.2]$ g77 test.f
[tkouya@cs-northpole gcc-3.3.2]$ ./a.out
  1.41421354
[tkouya@cs-northpole gcc-3.3.2]
$
```

6.2 ATLAS

本家 [18] から atlas3.6.0.tar.gz[8]をダウンロードし，/usr/local ディレクトリにインストールする。

```
[tkouya@cs-northpole cluster]$ tar zxvf atlas3.6.0.tar.gz
```

[8]Pentium IV 用のバイナリも存在するが，今回は Source からビルドした。

```
[tkouya@cs-northpole cluster]$ cd ATLAS
[tkouya@cs-northpole ATLAS]$ ls
CONFIG/      Make.top  README  config.c  include/     lib/    src/
INSTALL.txt  Makefile  bin/    doc/      interfaces/  makes/  tune/
[tkouya@cs-northpole ATLAS]$ make
(幾つかの質問に答える必要あり)
Probing for supported ISA extensions:
   AltiVec: NO.
   AltiVec: NO.
   SSE2: DETECTED!
(略)

ATLAS can provide SMP support for the Level 3 BLAS via Posix threads.
If you choose to build a threaded library, ATLAS will compile all
aspects of the library (including the serial components) with the
threaded compiler/link flags.  Most machines can use the serial
library even when it is compiled with threaded options, but this
is not guaranteed to work, so if you want a true serial library,
answer no to threading below.
   enable Posix threads support? [y]:
Number of CPUs: 2

Looking for compilers (this may take a while):
   /usr/local/bin/gcc  : v3.3.2
F77 = /usr/local/bin/g77 -fomit-frame-pointer -O
CC = /usr/local/bin/gcc -fomit-frame-pointer -O3 -funroll-all-loops
MCC = /usr/local/bin/gcc -fomit-frame-pointer -O

FINDING tar, gzip, AND gunzip
   tar    : /bin/tar
   gzip   : /bin/gzip
   gunzip : /bin/gunzip

(略)

   Enter Architecture name (ARCH) [Linux_P4SSE2_2]:
<arch> set to 'Linux_P4SSE2_2'

(略)

   Enter Maximum cache size (KB) [512]:

(略)
```

```
F77 & FLAGS: /usr/local/bin/g77 -fomit-frame-pointer -O
FLINKER & FLAGS: $(F77) $(F77FLAGS)

CC & FLAGS: /usr/local/bin/gcc -fomit-frame-pointer -O3 -funroll-all-loops
MCC & FLAGS: /usr/local/bin/gcc -fomit-frame-pointer -O
CLINKER & FLAGS: $(CC) $(CCFLAGS)


Finding F77 to C calling conventions (this may take a while):

Calculated F77/C interoperation conventions:
   Suffix F77 names with underscores with __
   F77 INTEGER -> C int
   F77 strings handled via standard sun style

(略)

Use supplied default values for install? [y]:


Unpacking Architectural defaults . . . done.

Creating make include file Make.Linux_P4SSE2_2
Make.Linux_P4SSE2_2 successfully created.



Creating ATLrun.sh

Creating subdirectories:
   Checking for already existing subdirectories  ........ no
Subdirectories successfully created.


Moving config logfiles ConfSummary.log and ConfDump.log to bin/Linux_P4SSE2_2/IN
STALL_LOG/


Configuration completed successfully.  You may want to examine the make include
file (Make.Linux_P4SSE2_2) for accuracy before starting the install with the com
mand:
   make install arch=Linux_P4SSE2_2
```

```
rm -f ./xconfig
[tkouya@cs-northpole ATLAS]$ make install arch=Linux_P4SSE2_2
(略)
ATLAS install complete.  Examine
ATLAS/bin/<arch>/INSTALL_LOG/SUMMARY.LOG for details.
make[1]: 出ます ディレクトリ '/home/tkouya/cluster/ATLAS'
[tkouya@cs-northpole ATLAS]$ ls lib
Linux_P4SSE2_2/  Make.ext  test_dynlink.c
[tkouya@cs-northpole ATLAS]$ ls lib/Linux_P4SSE2_2
Make.inc@  libatlas.a  libf77blas.a  libptcblas.a    libtstatlas.a
Makefile   libcblas.a  liblapack.a   libptf77blas.a
[tkouya@cs-northpole ATLAS]$ su
[root@cs-northpole ATLAS]# cp lib/Linux_P4SSE2_2/*.a /usr/local/lib
[root@cs-northpole ATLAS]# exit
```

今回は，Pthread を使って 2CPU 用のチューニングを施したものをインストールしたが，果たしてこれが最善かどうかはベンチマークテストを行ってみないとなんともいえない。

6.3 BLAS, LAPACK

本家 [19] もしくは Phase サーバから lapack.tgz をダウンロードして，BLAS と共にインストールする。

まず，同封されている BLAS を make する。

```
[tkouya@cs-northpole cluster]$ tar zxvf lapack.tgz
[tkouya@cs-northpole LAPACK]$ cp INSTALL/make.inc.LINUX ./make.inc
[tkouya@cs-northpole LAPACK]$ cd BLAS
[tkouya@cs-northpole BLAS]$ ls
SRC/      cblat2.in  dblat2.in  sblat2.in  zblat2.in
TESTING/  cblat3.in  dblat3.in  sblat3.in  zblat3.in
[tkouya@cs-northpole BLAS]$ cd SRC
[tkouya@cs-northpole SRC]$ ls
Makefile  chpr.f    daxpy.f   dsymv.f   izamax.f  sspr.f    zaxpy.f   zhpr.f
caxpy.f   chpr2.f   dcabs1.f  dsyr.f    lsame.f   sspr2.f   zcopy.f   zhpr2.f
ccopy.f   crotg.f   dcopy.f   dsyr2.f   sasum.f   sswap.f   zdotc.f   zrotg.f
cdotc.f   cscal.f   ddot.f    dsyr2k.f  saxpy.f   ssymm.f   zdotu.f   zscal.f
cdotu.f   csscal.f  dgbmv.f   dsyrk.f   scasum.f  ssymv.f   zdscal.f  zswap.f
cgbmv.f   cswap.f   dgemm.f   dtbmv.f   scnrm2.f  ssyr.f    zgbmv.f   zsymm.f
cgemm.f   csymm.f   dgemv.f   dtbsv.f   scopy.f   ssyr2.f   zgemm.f   zsyr2k.f
cgemv.f   csyr2k.f  dger.f    dtpmv.f   sdot.f    ssyr2k.f  zgemv.f   zsyrk.f
cgerc.f   csyrk.f   dnrm2.f   dtpsv.f   sgbmv.f   ssyrk.f   zgerc.f   ztbmv.f
cgeru.f   ctbmv.f   drot.f    dtrmm.f   sgemm.f   stbmv.f   zgeru.f   ztbsv.f
chbmv.f   ctbsv.f   drotg.f   dtrmv.f   sgemv.f   stbsv.f   zhbmv.f   ztpmv.f
```

```
chemm.f   ctpmv.f   dsbmv.f   dtrsm.f   sger.f    stpmv.f   zhemm.f   ztpsv.f
chemv.f   ctpsv.f   dscal.f   dtrsv.f   snrm2.f   stpsv.f   zhemv.f   ztrmm.f
cher.f    ctrmm.f   dspmv.f   dzasum.f  srot.f    strmm.f   zher.f    ztrmv.f
cher2.f   ctrmv.f   dspr.f    dznrm2.f  srotg.f   strmv.f   zher2.f   ztrsm.f
cher2k.f  ctrsm.f   dspr2.f   icamax.f  ssbmv.f   strsm.f   zher2k.f  ztrsv.f
cherk.f   ctrsv.f   dswap.f   idamax.f  sscal.f   strsv.f   zherk.f
chpmv.f   dasum.f   dsymm.f   isamax.f  sspmv.f   xerbla.f  zhpmv.f
[tkouya@cs-northpole SRC]$ make
(略)
g77  -funroll-all-loops -fno-f2c -O3 -c zher2k.f
ar cr ../../blas_LINUX.a dcabs1.o dzasum.o dznrm2.o izamax.o zaxpy.o zcopy.o zdo
tc.o zdotu.o zdscal.o zrotg.o zscal.o zswap.o idamax.o dasum.o daxpy.o dcopy.o d
nrm2.o dscal.o \
lsame.o xerbla.o zgemv.o zgbmv.o zhemv.o zhbmv.o zhpmv.o ztrmv.o ztbmv.o ztpmv.o
 ztrsv.o ztbsv.o ztpsv.o zgerc.o zgeru.o zher.o zhpr.o zher2.o zhpr2.o zgemm.o z
symm.o zsyrk.o zsyr2k.o ztrmm.o ztrsm.o zhemm.o zherk.o zher2k.o
ranlib ../../blas_LINUX.a
[tkouya@cs-northpole SRC]$
```

次に，LAPACK をコンパイルする。

```
[tkouya@cs-northpole SRC]$ cd ..
[tkouya@cs-northpole BLAS]$ cd ..
[tkouya@cs-northpole LAPACK]$ make
( cd INSTALL; make; ./testlsame; ./testslamch; \
  ./testdlamch; ./testsecond; ./testdsecnd; \
  cp lsame.f ../BLAS/SRC/; cp lsame.f ../SRC; \
  cp slamch.f ../SRC/; cp dlamch.f ../SRC/; \
  cp second.f ../SRC/; cp dsecnd.f ../SRC/ )
make[1]: 入ります ディレクトリ ‘/home/tkouya/cluster/LAPACK/INSTALL’
make[1]: ‘all’ に対して行うべき事はありません。
make[1]: 出ます ディレクトリ ‘/home/tkouya/cluster/LAPACK/INSTALL’
  ASCII character set
  Tests completed
  Epsilon                      =   5.96046448E-08
  Safe minimum                 =   1.17549435E-38
  Base                         =   2.
  Precision                    =   1.1920929E-07
  Number of digits in mantissa =   24.
  Rounding mode                =   1.
  Minimum exponent             =  -125.
  Underflow threshold          =   1.17549435E-38
  Largest exponent             =   128.
  Overflow threshold           =   3.40282347E+38
```

```
  Reciprocal of safe minimum   =   8.50705917E+37
  Epsilon                      =   1.11022302E-16
  Safe minimum                 =   2.22507386E-308
  Base                         =   2.
  Precision                    =   2.22044605E-16
  Number of digits in mantissa =   53.
  Rounding mode                =   1.
  Minimum exponent             =  -1021.
  Underflow threshold          =   2.22507386E-308
  Largest exponent             =   1024.
  Overflow threshold           =   1.79769313E+308
  Reciprocal of safe minimum   =   4.49423284E+307
 Time for 1,000,000 SAXPY ops  =   0.00     seconds
 *** Error:  Time for operations was zero
 Including SECOND, time        =   0.00     seconds
 Average time for SECOND       =   0.00     milliseconds
 Time for 1,000,000 DAXPY ops  =   0.00     seconds
 *** Error:  Time for operations was zero
 Including DSECND, time        =   0.00     seconds
 Average time for DSECND       =   0.00     milliseconds
( cd SRC; make )
make[1]: 入ります ディレクトリ `/home/tkouya/cluster/LAPACK/SRC'
(略)
make[2]: 出ます ディレクトリ `/home/tkouya/cluster/LAPACK/TIMING/LIN'
Timing square REAL LAPACK linear equation routines
xlintims < stime.in > stime.out 2>&1
make[1]: *** [stime.out] エラー 127
make[1]: 出ます ディレクトリ `/home/tkouya/cluster/LAPACK/TIMING'
make: *** [timing] エラー 2
[tkouya@cs-northpole LAPACK]$
```

最後にエラーが出るのが気にかかるが,

```
[tkouya@cs-northpole LAPACK]$ ls
BLAS/     Makefile  SRC/      TIMING/       lapack_LINUX.a  make.inc
INSTALL/  README    TESTING/  blas_LINUX.a  latape          tmglib_LINUX.a
```

というように，この時点ではちゃんとライブラリが出来上がっているので，あまり気にせずインストールする。

```
[tkouya@cs-northpole LAPACK]$ su
[root@cs-northpole LAPACK]# cp *.a /usr/local/lib
[root@cs-northpole LAPACK]# exit
[tkouya@cs-northpole LAPACK]$
```

ついでに，C 用の include ファイル (clapack.h) も/usr/local/include にコピーしておく。

6.4 GMP(+MPFR), BNCpack

本家，もしくは Ring サーバから gmp-4.1.2.tar.gz をダウンロードし，/usr/local ディレクトリにインストールし，併せて MPFR, C++クラスライブラリを使用可能にする。

```
[tkouya@cs-northpole gmp-4.1.2]$ ./configure --prefix=/usr/local --enable-mpfr --enable-cxx
[tkouya@cs-northpole gmp-4.1.2]$ make
[tkouya@cs-northpole gmp-4.1.2]$ su
[tkouya@cs-northpole gmp-4.1.2]# make install
```

次に，bnc ディレクトリを作ってそこへ移動し，BNCpack をダウンロードしてコンパイルする。

```
[tkouya@cs-northpole cluster]$ mkdir bnc
[tkouya@cs-northpole cluster]$ cd bnc
[tkouya@cs-northpole bnc]$ wget http://na-inet.jp/na/bnc/bnc-0.3e.tar.gz
[tkouya@cs-northpole bnc]$ make
cc -c -DUSE_GMP -DUSE_MPFR -I/usr/local/include:/usr/include:./ get_sec.c
(略)
tkouya@cs-northpole bnc]$ ./test_dka
    0  -1.2000000e+02
    1   2.7400000e+02
    2  -2.2500000e+02
    3   8.5000000e+01
    4  -1.5000000e+01
    5   1.0000000e+00
    0 (  8.8980286e+01,   2.6596819e+01)
    1 (  4.2742925e+00,   8.9990974e+01)
    2 ( -8.2192734e+01,   2.9020662e+01)
    3 ( -5.0926308e+01,  -7.2055214e+01)
    4 (  5.4864437e+01,  -7.3553246e+01)
Iterative times: 101
    0 (  5.0000033e+00,   0.0000000e+00)
    1 (  2.9999955e+00,   0.0000000e+00)
    2 (  9.9999988e-01,   0.0000000e+00)
    3 (  1.9999964e+00,   0.0000000e+00)
    4 (  4.0000000e+00,   0.0000000e+00)
(略)
--------------------------------------------------------------------------------
BNC Default Precision    : 512 bits(154.1 decimal digits)
BNC Default Rounding Mode: Round to Nearest
--------------------------------------------------------------------------------
(略)
ans->prec: 512
Iterative times: 143
    0 (1.99999999999999999999999999999999999999999999999999999999999999999999999999999
```

```
99999999999999999999999999999986577097200753607832672967402832634719761119809447
28e1, -6.109145669860526688037151491112995410796635172013236468868597779187642255
57990704046472160333122626597693801777729478021838075061291874095362244031057503
61e-101)
(略)
   19 (1.89999999999999999999999999999999999999999999999999999999999999999999999999
99999999999999999999999998662775491321320243988322312300759980428643477774413872586e1, 9.66865623864388195004215885190460443184236023553008766540883229208423385550754670009157610527390900891535271036194208075292346181979483109957595053481747
3e-96)
```

上記のような出力が得られれば良い。

最後に，ライブラリファイル (libbnc.a) とヘッダファイル (_bnc_mpf.h, bnc.h) を/usr/local 以下にコピーする。

```
[tkouya@cs-northpole bnc]$ su
[root@cs-northpole bnc]# cp libbnc.a /usr/local/lib
[root@cs-northpole bnc]# cp *.h /usr/local/include
```

6.5 JDK, Omni

OpenMP コンパイラである Omni をインストールするには，Java 環境が必須である。そのため，まず，Java 2 SDK[30] をダウンロードし，解凍して，/usr/local 以下にインストールする。途中，ライセンス条項の確認が行われるので，yes を選択すること。

```
[tkouya@cs-northpole cluster]$ chmod +x j2sdk-1_4_2_02-linux-i586.bin
[tkouya@cs-northpole cluster]$ ./j2sdk-1_4_2_02-linux-i586.bin
(略)
Done.
[tkouya@cs-northpole cluster]$ su
[root@cs-northpole cluster]# cp -R j2sdk1.4.2_02/* /usr/local
cp: ‘‘/usr/local/bin/jar’’ を上書きしてもよろしいですか (yes/no)? yes
cp: ‘‘/usr/local/bin/rmic’’ を上書きしてもよろしいですか (yes/no)? y
cp: ‘‘/usr/local/bin/rmiregistry’’ を上書きしてもよろしいですか (yes/no)? y
cp: ‘‘/usr/local/include/jni.h’’ を上書きしてもよろしいですか (yes/no)? y
cp: ‘‘/usr/local/include/jvmpi.h’’ を上書きしてもよろしいですか (yes/no)? y
cp: ‘‘/usr/local/man/man1/jar.1’’ を上書きしてもよろしいですか (yes/no)? y
cp: ‘‘/usr/local/man/man1/rmic.1’’ を上書きしてもよろしいですか (yes/no)? y
cp: ‘‘/usr/local/man/man1/rmiregistry.1’’ を上書きしてもよろしいですか (yes/no)?
 y
```

次に，Omni-1.6 をダウンロードし，インストールする。

```
[tkouya@cs-northpole cluster]$ cd Omuni-1.6
bash: cd: Omuni-1.6: No such file or directory
```

```
[tkouya@cs-northpole cluster]$ cd Omni-1.6
[tkouya@cs-northpole Omni-1.6]$ ./configure --prefix=/usr/local
[tkouya@cs-northpole Omni-1.6]$ make
[tkouya@cs-northpole Omni-1.6]$ su
[root@cs-northpole Omni-1.6]# make install
[root@cs-northpole Omni-1.6]# exit
```

最後に，テストプログラムをコンパイルし実行してみる。

```
[tkouya@cs-northpole Omni-1.6]$ cd tests
[tkouya@cs-northpole tests]$ ls
C-test/  Makefile     cg/        laplace/      scash-test/
F-test/  Makefile.in  clinpack/  laplace-f77/  tiny/
[tkouya@cs-northpole tests]$ cd cg
[tkouya@cs-northpole cg]$ make
cc    -c -o second.o second.c
cc  -o cg cg.c second.o -lm
${OMNI_HOME:=/usr/local}/bin/omcc  -o cg-omp cg.c second.o -lm
Compiling 'cg.c'...
${OMNI_HOME:=/usr/local}/bin/omcc  -o cg-orphan cg-orphan.c second.o -lm
Compiling 'cg-orphan.c'...
cc  -o cg-makedata cg-makedata.c -lm
[tkouya@cs-northpole cg]$ ./cg-makedata data
final nonzero count in sparse number of nonzeros  =            78148
[tkouya@cs-northpole cg]$ ./cg data
    0    1.3769e-13      9.99864415791401
    1    2.1517e-15      8.57332792032217
    2    2.1316e-15      8.59545103740579
    3    1.9500e-15      8.59699723407375
    4    1.8489e-15      8.59715491517665
    5    1.9902e-15      8.59717443116078
    6    1.8073e-15      8.59717707049128
    7    1.8800e-15      8.59717744406296
    8    1.7952e-15      8.59717749839416
    9    1.7612e-15      8.59717750644093
   10    1.9035e-15      8.59717750764864
   11    1.8355e-15      8.59717750783180
   12    1.8144e-15      8.59717750785982
   13    1.7342e-15      8.59717750786414
   14    1.8540e-15      8.59717750786481
time = 0.330621, 0.022041 (0.000000e+00 - 3.306210e-01)/15, NITCG=25
[tkouya@cs-northpole cg]$ ./cg-omp data
omp_num_thread=1
omp_max_thread=2
```

```
    0    1.3755e-13      9.99864415791401
    1    2.1815e-15      8.57332792032217
    2    2.1749e-15      8.59545103740579
    3    2.0147e-15      8.59699723407375
    4    1.8920e-15      8.59715491517666
    5    2.0422e-15      8.59717443116078
    6    1.9023e-15      8.59717707049128
    7    1.9085e-15      8.59717744406296
    8    1.8410e-15      8.59717749839416
    9    1.8659e-15      8.59717750644093
   10    1.9250e-15      8.59717750764864
   11    1.7752e-15      8.59717750783180
   12    1.6886e-15      8.59717750785982
   13    1.7721e-15      8.59717750786414
   14    1.8439e-15      8.59717750786481
time = 0.252650, 0.016843 (0.000000e+00 - 2.526500e-01)/15, NITCG=25
[tkouya@cs-northpole cg]$
```

HT を用いて擬似的に OpenMP を体験できるものの，PC Cluster のような分散メモリ環境で本格的にこのコンパイラを使うためには，SCASH 環境が不可欠である。その場合は，SCore[8] の導入を検討するべきだろう[9]。

7 NFS/NIS Server/Clients の設定

7.1 親マシンの NFS 設定

親マシンは，前述の通り，共通アプリケーション用に/usr/local 以下を，NIS のために/home 以下を export する。

まず，/etc/exports を

```
[root@cs-northpole gcc-3.3.2]# cat /etc/exports
/home           192.168.1.*(rw)
/usr/local      192.168.1.*(rw)
```

とし，/usr/sbin/exportfs コマンドを実行する。

```
[root@cs-northpole tkouya]# /usr/sbin/exportfs -v -a
exporting 192.168.1.*:/usr/local
exporting 192.168.1.*:/home
[root@cs-northpole tkouya]# /usr/sbin/exportfs -v
/usr/local      192.168.1.*(rw,async,wdelay,root_squash)
/home           192.168.1.*(rw,async,wdelay,root_squash)
```

親マシンの NFS サービスを on にする。

[9]著者はまだ未経験 (2004 年 2 月現在)。

```
[root@cs-northpole tkouya]# /sbin/chkconfig --list
(略)
nfs             0:off   1:off   2:off   3:off   4:off   5:off   6:off
nfslock         0:off   1:off   2:off   3:on    4:on    5:on    6:off
(略)
[root@cs-northpole tkouya]# /sbin/chkconfig nfs on
[root@cs-northpole tkouya]# /sbin/chkconfig --list
(略)
nfs             0:off   1:off   2:off   3:on    4:on    5:on    6:off
nfslock         0:off   1:off   2:off   3:on    4:on    5:on    6:off
(略)
[root@cs-northpole tkouya]#
```

reboot して NFS が有効になっていることを確認する。

7.2 子マシンの NFS 設定 (1)

NIS が有効になっていないと/home へのアクセスが出来なくなるので，まずは/usr/local の共有を実行してチェックする。

NFS サービスを on にする。

```
[root@cs-room443-02 tkouya]# /sbin/chkconfig --list
(略)
nfs             0:off   1:off   2:off   3:off   4:off   5:off   6:off
nfslock         0:off   1:off   2:off   3:on    4:on    5:on    6:off
(略)
[root@cs-room443-02 tkouya]# /sbin/chkconfig nfs on
[root@cs-room443-02 tkouya]# /sbin/chkconfig --list
(略)
nfs             0:off   1:off   2:off   3:on    4:on    5:on    6:off
nfslock         0:off   1:off   2:off   3:on    4:on    5:on    6:off
(略)
[root@cs-room443-02 tkouya]#
```

/etc/fstab に次の行を追加する[10]。

```
cs-northpole-nis:/usr/local     /usr/local      nfs     rw,hard,intr    0 0
```

reboot して，/usr/local が NFS mount されていることを確認する。

```
[tkouya@cs-room443-02 tkouya]$ mount
(略)
cs-northpole-nis:/usr/local on /usr/local type nfs (rw,hard,intr,addr=192.168.1.5)
```

[10] この設定では，NFS mount 出来るまで retry を繰り返す。改善の余地あり

7.3 親マシンの NIS サーバ設定

portmap, ypserv, ypbind, yppasswdd を有効にする。

```
[root@cs-northpole tkouya]# /sbin/chkconfig --list
portmap         0:off   1:off   2:off   3:on    4:on    5:on    6:off
ypbind          0:off   1:off   2:off   3:off   4:off   5:off   6:off
yppasswdd       0:off   1:off   2:off   3:off   4:off   5:off   6:off
ypserv          0:off   1:off   2:off   3:off   4:off   5:off   6:off
[root@cs-northpole tkouya]# /sbin/chkconfig ypbind on
[root@cs-northpole tkouya]# /sbin/chkconfig yppasswdd on
[root@cs-northpole tkouya]# /sbin/chkconfig ypserv on
[root@cs-northpole tkouya]# /sbin/chkconfig --list
portmap         0:off   1:off   2:off   3:on    4:on    5:on    6:off
postfix         0:off   1:off   2:on    3:on    4:on    5:on    6:off
ypbind          0:off   1:off   2:off   3:on    4:on    5:on    6:off
yppasswdd       0:off   1:off   2:off   3:on    4:on    5:on    6:off
ypserv          0:off   1:off   2:off   3:on    4:on    5:on    6:off
```

/varyp/securenets[11]の設定を行い，MPI以外のネットワークサービス用 Private Address(192.168.1.0/24) にのみ，NIS Client 権限を持たせる。

```
# Always allow access for localhost
255.0.0.0       127.0.0.0

# This line gives access to everybody. PLEASE ADJUST!
255.255.255.0   192.168.1.0
```

NIS ドメインは”cs-pccluster2”なので，/etc/yp.conf に以下の行を追加する。

```
domain  cs-pccluster2   server  cs-northpole-nis
```

/etc/sysconfig/network に以下の行を追加。

```
NISDOMAIN="cs-pccluster2"
```

reboot するとこの NIS ドメイン名が有効になるが，NIS の設定が完了する前に再起動してしまうと，”YPBINDPROC_DOMAIN: Domain not bound”というメッセージが出てうるさい。とりあえず，一時的に

```
[root@cs-northpole tkouya]# domainname cs-pccluster2
[root@cs-northpole tkouya]# domainname
cs-pccluster2
[root@cs-northpole tkouya]#
```

として NIS ドメイン名を設定しておく。

以上の手続き後，/var/yp ディレクトリに移動し，make コマンドを使って NIS サーバとして必要なファイルを用意する。

[11] Vine 2.6r3 にこのファイルは存在しなかった。不要になった？

```
[root@cs-northpole tkouya]# cd /var/yp
[root@cs-northpole yp]# make
gmake[1]: 入ります ディレクトリ ‘/var/yp/cs-pccluster2’
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating hosts.byname...
Updating hosts.byaddr...
Updating rpc.byname...
Updating rpc.bynumber...
Updating services.byname...
Updating services.byservicename...
Updating netid.byname...
Updating protocols.bynumber...
Updating protocols.byname...
Updating mail.aliases...
gmake[1]: 出ます ディレクトリ ‘/var/yp/cs-pccluster2’
[root@cs-northpole yp]#
```

reboot しても良いが，面倒なので ypserv(NIS server daemon) と ypbind(NIS Client daemon) を再起動し，ypcat コマンドで NIS の動作状況をチェックする。

```
[root@cs-northpole yp]# /sbin/service ypserv restart
YP サーバサービスを停止中:                                 [  OK  ]
YP サーバサービスを起動中:                                 [  OK  ]
[root@cs-northpole yp]# /sbin/service ypbind restart
Shutting down NIS services:                                [  OK  ]
Binding to the NIS domain...                               [  OK  ]
Listening for an NIS domain server: cs-northpole-nis
[root@cs-northpole yp]# ypcat passwd
tkouya:$1$gCNfIloO$N5i/QaHQPQ0rnjAECDlNZ0:500:500:Tomonori Kouya:/home/tkouya:/b
in/bash
[root@cs-northpole yp]# ypcat hosts
192.168.2.13            cs-room443-03-g
192.168.2.12            cs-room443-02-g
127.0.0.1               localhost localhost.localdomain
127.0.0.1               localhost localhost.localdomain
192.168.1.5             cs-northpole-nis
133.88.121.78           cs-southpole
192.168.2.1             cs-northpole-g
192.168.1.14            cs-room443-04
192.168.1.13            cs-room443-03
192.168.1.12            cs-room443-02
```

```
192.168.2.14            cs-room443-04-g
[root@cs-northpole yp]#
```

reboot して，もう一度 ypcat コマンドで確認する。

```
[root@cs-northpole tkouya]# domainname
cs-pccluster2
[root@cs-northpole tkouya]# ypcat passwd
tkouya:$1$gCNfIloO$N5i/QaHQPQ0rnjAECDlNZ0:500:500:Tomonori Kouya:/home/tkouya:/bin/bash
[root@cs-northpole tkouya]# ypcat hosts
192.168.2.13            cs-room443-03-g
192.168.2.12            cs-room443-02-g
127.0.0.1               localhost localhost.localdomain
127.0.0.1               localhost localhost.localdomain
192.168.1.5             cs-northpole-nis
133.88.121.78           cs-southpole
192.168.2.1             cs-northpole-g
192.168.1.14            cs-room443-04
192.168.1.13            cs-room443-03
192.168.1.12            cs-room443-02
192.168.2.14            cs-room443-04-g
[root@cs-northpole tkouya]#
```

これで NIS サーバとしての設定は完了した。

7.4 子マシンの NIS Client 設定

子マシンは，NIS client としてのみ動作すればよいので，portmap, ypbind, yppasswdd(不要？) を on にする。

```
[root@cs-room443-02 tkouya]# /sbin/chkconfig --list
(略)
portmap         0:off   1:off   2:off   3:on    4:on    5:on    6:off
ypbind          0:off   1:off   2:off   3:off   4:off   5:off   6:off
yppasswdd       0:off   1:off   2:off   3:off   4:off   5:off   6:off
ypserv          0:off   1:off   2:off   3:off   4:off   5:off   6:off
(略)
[root@cs-room443-02 tkouya]# /sbin/chkconfig ypbind on
[root@cs-room443-02 tkouya]# /sbin/chkconfig yppasswdd on
[root@cs-room443-02 tkouya]# /sbin/chkconfig --list
(略)
portmap         0:off   1:off   2:off   3:on    4:on    5:on    6:off
ypbind          0:off   1:off   2:off   3:on    4:on    5:on    6:off
yppasswdd       0:off   1:off   2:off   3:on    4:on    5:on    6:off
ypserv          0:off   1:off   2:off   3:off   4:off   5:off   6:off
```

(略)

親マシン同様，/etc/yp.conf に

```
domain  cs-pccluster2   server  cs-northpole-nis
```

を追加し，/etc/sysconfig/network に以下の行を追加。

```
NISDOMAIN="cs-pccluster2"
```

reboot して ypcat コマンドで NIS サービスの確認。

```
[root@cs-room443-02 tkouya]# ypcat passwd
tkouya:$1$gCNfIloO$N5i/QaHQPQ0rnjAECDlNZ0:500:500:Tomonori Kouya:/home/tkouya:/b
in/bash
[root@cs-room443-02 tkouya]# ypcat hosts
192.168.2.13            cs-room443-03-g
192.168.2.12            cs-room443-02-g
127.0.0.1               localhost localhost.localdomain
127.0.0.1               localhost localhost.localdomain
192.168.1.5             cs-northpole-nis
133.88.121.78           cs-southpole
192.168.2.1             cs-northpole-g
192.168.1.14            cs-room443-04
192.168.1.13            cs-room443-03
192.168.1.12            cs-room443-02
192.168.2.14            cs-room443-04-g
```

7.5 子マシンの NFS 設定 (2)

最後に，親サーバの/home を mount すべく，/etc/fstab に

```
cs-northpole-nis:/home          /home           nfs     rw,hard,intr    0 0
```

を追加し，reboot 後，login の確認と，mount 状況を確認して問題なければ o.k.。

```
Vine Linux 2.6r3 (La Fleur de Bouard)
Kernel 2.4.22-0vl2.8smp on a 2-processor i686
login: tkouya
Password:
Last login: Thu Jan 29 20:13:49 from cs-northpole-nis
[tkouya@cs-room443-02 tkouya]$ ls
100000.txt   10000000.txt  cluster/     mgmp/
1000000.txt  Xrootenv.0    experiment/  rpm/
[tkouya@cs-room443-02 tkouya]$ mount
(略)
cs-northpole-nis:/usr/local on /usr/local type nfs (rw,hard,intr,addr=192.168.1.5)
cs-northpole-nis:/home on /home type nfs (rw,hard,intr,addr=192.168.1.5)
[tkouya@cs-room443-02 tkouya]$
```

以上で，親マシンと子マシンの基本設定は終了した。

8 mpich の設定

基本設定は終了したので，MPI 環境を構築する。ここでは rsh を利用して，mpich[20] が動作するように設定する手順を示す。

といっても，Vine 2.6r3 のインストール CD には in.rshd が入っていない (Vine Seed の方にある)。そこでまず本家から Netkit の本家 [29] から netkit-rsh-0.17.tar.gz をダウンロードし，/usr/local 以下にインストールしておくことにする。

```
[tkouya@cs-northpole tkouya]$ wget ftp://ftp.uk.linux.org/pub/linux/Networking/net
kit/netkit-rsh-0.17.tar.gz
[tkouya@cs-northpole tkouya]$ tar zxvf netkit-rsh-0.17.tar.gz
[tkouya@cs-northpole netkit-rsh-0.17]$ ./configure --prefix=/usr/local
[tkouya@cs-northpole netkit-rsh-0.17]$ make
[tkouya@cs-northpole netkit-rsh-0.17]$ su
[root@cs-northpole netkit-rsh-0.17]# make install
[tkouya@cs-northpole netkit-rsh-0.17]$ ls /usr/local/sbin
in.rexecd*  in.rlogind*  in.rshd*
```

8.1 rsh の有効化

親，子マシンとも，inetd.conf を

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

とし，

```
[root@cs-room443-02 tkouya]# /sbin/service inet restart
Stopping INET services:                                    [  OK  ]
Starting INET services:                                    [  OK  ]
```

と inetd を再起動して rsh(Remote SHell) を有効化しておく。

次に，親マシンの home ディレクトリ直下に.rhosts(全マシンで共有される) を，もしくは/etc/hosts.equiv(この場合は全てのマシンに作成する) を作成し，rsh が使用可能なマシンとユーザ名をリストアップしておく。

```
cs-northpole-g  tkouya
cs-room443-g    tkouya
```

rsh コマンドが動作することを確認する。

```
[tkouya@cs-room443-02 tkouya]$ rsh cs-northpole-g ls
100000.txt
1000000.txt
10000000.txt
```

```
Xrootenv.0
cluster
experiment
mgmp
rpm
```

8.2 親マシンに mpich をインストール

本家 [20]，もしくは Phase サーバ [25] から mpich-1.2.5.2.tar.gz をダウンロードして，/usr/local 以下にインストールする。rsh コマンドは新たにインストールしたもの (/usr/local/bin/rsh) を使いたいので，それもオプションで指定してある。

```
[tkouya@cs-northpole mpich-1.2.5.2]$ ./configure --prefix=/usr/local -rsh=/usr/local/bin/rsh
(略)
*# --->
*# You should register your copy of MPICH with us by sending mail
*# to majordomo@mcs.anl.gov containing the message
*# subscribe mpi-users
*# This will allow us to notify you of new releases of MPICH.
*#
*# You can also check the MPICH home page at
*# http://www.mcs.anl.gov/mpi/mpich
*# ---<
Old style arguments to configure were found:
Use the environment variable RSHCOMMAND instead of configure argument -rsh=/usr/local/bin/rsh

Configuration completed.
[tkouya@cs-northpole mpich-1.2.5.2]$
```

RSHCOMMAND 環境変数を使えと警告が出ているが，無視してビルドし，インストールする。

```
[tkouya@cs-northpole mpich-1.2.5.2]$ make
[tkouya@cs-northpole mpich-1.2.5.2]$ su
[root@cs-northpole mpich-1.2.5.2]# make install
[root@cs-northpole mpich-1.2.5.2]# exit
[tkouya@cs-northpole mpich-1.2.5.2]$ ls /usr/local/bin
addr2name.awk*                iperf*              mpirun.pg*
c++*                          jar*                mpirun.rand*
clog2alog*                    jcf-dump*           mpirun_dbg.dbx*
clog2slog*                    jv-convert*         mpirun_dbg.ddd*
clog_print*                   jv-scan*            mpirun_dbg.gdb*
cpp*                          logviewer*          mpirun_dbg.ladebug*
g++*                          mpereconfig*        mpirun_dbg.totalview*
g77*                          mpereconfig.dat*    mpirun_dbg.xxgdb*
```

```
gcc*                          mpereconfig.in*     rcp*
gccbug*                       mpicc*              rexec*
gcj*                          mpif77*             rlogin*
gcjh*                         mpif90*             rmic*
gcov*                         mpiman*             rmiregistry*
gij*                          mpireconfig*        rsh*
grepjar*                      mpireconfig.dat*    serv_p4*
i686-pc-linux-gnu-c++*        mpirun*             slog_print*
i686-pc-linux-gnu-g++*        mpirun.args*        tarch*
i686-pc-linux-gnu-gcc*        mpirun.ch_p4*       tdevice*
i686-pc-linux-gnu-gcc-3.3.2*  mpirun.ch_p4.args*
i686-pc-linux-gnu-gcj*        mpirun.p4shmem*
[tkouya@cs-northpole mpich-1.2.5.2]$ ls /usr/local/share
Makefile.sample  java/    machines.LINUX   upshot/
examples/        locale/  machines.sample
[tkouya@cs-northpole mpich-1.2.5.2]$
```

mpich で使用するマシンリストを/usr/local/share/machine.LINUX に追加する。

```
# Change this file to contain the machines that you want to use
# to run MPI jobs on.  The format is one host name per line, with either
#    hostname
# or
#    hostname:n
# where n is the number of processors in an SMP.  The hostname should
# be the same as the result from the command "hostname"
cs-northpole-g
cs-room443-02-g
```

最後に，テストプログラムをコンパイルして，MPI の動作チェックを行う。

```
[tkouya@cs-northpole mpich-1.2.5.2]$ cd examples
[tkouya@cs-northpole examples]$ ls
Makefile  Makefile.in  README  basic/  io@  nt/  perftest/  test/
[tkouya@cs-northpole examples]$ cd basic
[tkouya@cs-northpole basic]$ ls
Makefile     README  cpilog.c  hello++.cc*  pcp.c       prm.c     systest.c
Makefile.in  cpi.c   fpi.f     iotest.c     pi3f90.f90  srtest.c  unsafe.c
[tkouya@cs-northpole basic]$ make
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpicc   -c cpi.c
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpicc  -o cpi cpi.o -lm
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpicc   -c systest.c
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpicc  -o systest systest.o -lm
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpicc   -c srtest.c
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpicc  -o srtest srtest.o -lm
```

```
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpicc   -c iotest.c
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpicc  -o iotest iotest.o -lm
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpif77  -c fpi.f
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpif77  -o fpi fpi.o
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpicc   -I/home/tkouya/cluster/mpich-1.2.
5.2/mpe/include  -c /home/tkouya/cluster/mpich-1.2.5.2/examples/basic/cpilog.c
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpicc  -o cpilog cpilog.o -lmpe -lm
/home/tkouya/cluster/mpich-1.2.5.2/bin/mpicc  -o cpi_autolog cpi.o -mpilog -lm
[tkouya@cs-northpole basic]$ mpirun -np 1 ./cpi
Process 0 of 1 on cs-northpole
pi is approximately 3.1415926544231341, Error is 0.0000000008333410
wall clock time = 0.000235
[tkouya@cs-northpole basic]$ mpirun -np 2 ./cpi
Process 0 of 2 on cs-northpole
pi is approximately 3.1415926544231318, Error is 0.0000000008333387
wall clock time = 0.003471
Process 1 of 2 on cs-room443-02
[tkouya@cs-northpole basic]$
```

8.3 他の子マシンのセッティング

ここまで確認できれば，残りの子マシンに NIS/NFS 及び mpich の設定を行ってよい。手順をまとめると次のようになる。

1. /etc/hosts に cs-northpole-nis のエントリがあることを確認
2. NFS のサービスを on にする
3. /etc/fstab に cs-northpole-nis:/usr/local を NFS マウントする設定を記述する
4. mount /usr/local して確認
5. NIS Client サービスを on にする
6. /etc/yp.conf に NIS の設定を記述する
7. /etc/sysconfig/network に NIS domain を追加
8. reboot する
9. /usr/local マウントの確認
10. ypcat コマンドで NIS の設定を確認
11. /etc/fstab に cs-northpole-nis:/home を NFS マウントする設定を記述する
12. reboot する
13. ログインして NIS, NFS の確認

14. /etc/inetd.conf に/usr/local/sbin/rshd の設定を追加

15. /home/tkouya/.rhosts に子マシンを追加

16. inetd を再起動して rsh の動作確認

17. /usr/local/share/machines.LINUX に子マシンを追加して mpirun のテスト

18. reboot して全ての総点検

バカみたいに reboot しまくっているが，実際にはサービスの再起動だけで済む。この時は導入したてのマシンだったので，動作チェックも兼ねていたのである。

9 並列分散数値計算ソフトウェアのインストール

9.1 MPIBNCpack

BNCpack と同様，mpibnc ディレクトリを掘り，そこへカレントディレクトリを移動して，ダウンロードし，コンパイルする。

```
[tkouya@cs-northpole cluster]$ mkdir mpibnc
[tkouya@cs-northpole cluster]$ cd mpibnc
[tkouya@cs-northpole mpibnc]$ wget http://na-inet.jp/na/bnc/mpibnc.tar.gz
[tkouya@cs-northpole mpibnc]$ tar zxvf mpibnc.tar.gz
License.txt
Makefile
cpi-gmp.c
mpi_aux.c
mpi_bcastbnc.c
mpi_cg.c
mpi_complex.c
mpi_dka.c
mpi_gmp.c
mpi_integral.c
mpi_linear.c
mpi_matrix_mul.c
print_process.c
test.c
test_mpicg.c
test_mpidka.c
test_mpiefunc.c
test_mpiint.c
test_mpilinear.c
test_mpimm.c
test_mpzq.c
testc.c
```

```
mpi_bnc.h
mpi_gmp.h
test_mpidkacoef.dat
test_mpidkacoef2.dat
[tkouya@cs-northpole mpibnc]$ make
mpicc  -c -DUSE_GMP -DUSE_MPFR -I/usr/local/include:/usr/include:./ mpi_gmp.c
(略)
mpicc  -otest_mpidka -DUSE_GMP -DUSE_MPFR -I/usr/local/include:/usr/include:./ t
est_mpidka.o libmpibnc.a -L/usr/local/lib:/usr/lib:./ /usr/local/lib/libbncode.a
 /usr/local/lib/libbnc.a /usr/local/lib/libmpfr.a /usr/local/lib/libgmp.a -lm
[tkouya@cs-northpole mpibnc]$ ls *.h
mpi_bnc.h*  mpi_gmp.h*
[tkouya@cs-northpole mpibnc]$ ls *.a
libmpibnc.a
[tkouya@cs-northpole mpibnc]$
```

最後に，ライブラリファイル (libmpibnc.a) とヘッダファイル (mpi_bnc.h, mpi_gmp.h) を/usr/local 以下にコピーする。

```
[tkouya@cs-northpole mpibnc]$ su
[root@cs-northpole mpibnc]# cp libmpibnc.a /usr/local/lib
[root@cs-northpole mpibnc]# cp *.h /usr/local/include
```

参考文献

[1] 幸谷智紀,「Vine Linux による PC Cluster の構築」, http://na-inet.jp/na/mpibnc.pdf

[2] 幸谷智紀,「MPI を用いた並列多倍長数値計算の簡易な実装について」,FIT2003.

[3] 幸谷智紀,「PC Cluster 上における多倍長数値計算ライブラリ BNCpack の並列分散化」, http://lc.linux.or.jp/paper/lc2003/CP-14.pdf

[4] BNCpack, http://phase.hpcc.jp/phase/bncpack/

[5] MPIBNCpack, http://na-inet.jp/na/bnc/mpibncpack.pdf

[6] PC Watch, http://pc.watch.impress.co.jp/

[7] 超並列計算研究会, PC クラスタ超入門 2000, http://mikilab.doshisha.ac.jp/dia/smpp/cluster2000/

[8] 石川 他,「Linux で並列計算をしよう」, 共立出版, 2002.

[9] MPI Forum, http://www.mpi-forum.org/

[10] GNU MP, http://swox.com/gmp/

[11] MPFR Project, http://www.mpfr.org/

[12] Ring Servers, http://www.ring.gr.jp/

[13] Vine Linux, http://www.vinelinux.org/

[14] http://vinelinux.org/errata/25x/20040206-1.html

[15] Pure Load Balancer, http://plb.sunsite.dk/

[16] BLACS, http://www.netlib.org/blacs/

[17] 姫野ベンチマーク, http://w3cic.riken.go.jp/HPC/HimenoBMT/

[18] Automatically Tuned Linear Algebra Software, http://math-atlas.sourceforge.net/

[19] LAPACK, http://www.netlib.org/lapack/

[20] MPICH, http://www-unix.mcs.anl.gov/mpi/mpich/

[21] Netlib, http://www.netlib.org/

[22] Linux NFS-HOWTO, http://www.linux.or.jp/JF/JFdocs/NFS-HOWTO/

[23] Linux NIS-HOWTO, http://www.linux.or.jp/JF/JFdocs/NIS-HOWTO/

[24] PC Cluster Consortium, http://www.pccluster.org/

[25] PHASE Project, http://www.hpcc.jp/

[26] ScaLAPACK, http://www.netlib.org/scalapack/

[27] 超並列計算研究会, http://www.is.doshisha.ac.jp/SMPP/

[28] Iperf, http://dast.nlanr.net/Projects/Iperf/

[29] Linux Netkit, ftp://ftp.uk.linux.org/pub/linux/Networking/netkit/

[30] Java 2 SDK v.1.4.2_02, http://java.sun.com/products/archive/j2se/1.4.2_02/index.html

[31] Omni OpenMP Compiler Project, http://phase.hpcc.jp/Omni/