ISSN 0915-6356

NATIONAL INSTITUTE FOR FUSION SCIENCE

高速通信ソフトウェアを利用したPCクラスター計算機
Beowulf Cluster Equipped with High-Speed Communication Software

田中基彦
M. Tanaka

(Received - May 10, 2004)

NIFS-TECH-12 May 2004

RESEARCH REPORT
NIFS-TECH Series

TOKI, JAPAN

高速通信ソフトウェアを利用したPCクラスター計算機
Beowulf Cluster Equipped with High-Speed Communication Software

田中基彦
核融合科学研究所、連携研究推進センター

Motohiko Tanaka
National Institute for Fusion Science, Toki 509-5292, Japan
Email: mtanaka@nifs.ac.jp URL: http://dphysique.nifs.ac.jp/

Abstract

A high performance PC cluster computer installed with Linux operating system and MPI (Message Passing Interface) for interprocessor communication has been constructed using a communication software GAMMA (Genoa Active Message Machine) instead of the standard TCP/IP protocol. Fast C/Fortran compilers have been exploited with the GAMMA communication libraries. This method has resulted in drastic reduction of the communication overhead and significant increase in the computation performance of real application programs including the first-principle molecular dynamics simulation code.

Keywords: PC cluster machine, non-TCP/IP communication, small latency and high throughput, fast C/Fortran compilers

1. はじめに

多くのパーソナルコンピュータ(PC)やワークステーションをネットワークで結び長時間、大容量の計算を行うグリッド計算(Grid Computing)の試みが世界各国の研究所や大学で進められている。これは、Purdue 大学や NASA で 1990 年代に開始された、分散メモリをもつ多数個のプロセッサで構成されるクラスター(並列)計算機による高速計算の研究[1,2]がベースになっている。これまで日本でも、大学の研究室を中心に PC クラスター計算機の導入が行われてきた[3,4]。PC クラスター計算機は Beowulf クラスターマシンとも呼ばれる。その特徴は、スカラー型(逐次)演算器と分散メモリをもつ多数個の PC に協調して作業を行わせ、全体として高い演算能力を実現することにあり、安価にスーパーコンピュータ並みの演算性能を持つ計算機を構築できる。私のグループでも Linux を処理系とする Pentium III (1GHz)を 16 台並列にして、MPI (Message Passing Interface)[5] により PC 間の通信を行うクラスター計算機を製作し、それ以後 CPU を更新して高速 CPU 下での並列プログラムの開発と研究に利用している [6]。

ところで、日常の計算機シミュレーション研究において重要なことは、標準的ベンチマークプログラムが高速に動くことではなく、各自がもつ応用プログラムがどれほど速く実行されるかである[7]。高速演算を行う並列計算機のハードウェアには、(1) 高速の CPU、(2) CPU 周囲での高速なメモリフロー、(3) プロセッサ(分散メモリ)間の高速な通信、のすべてが必要である。一般ユーザーが単体で利用することが主流のPCは、市場に直結して驚くほど急速にCPUの高速化が成し遂げられ、われわれ研究者もその恩恵にあずかっている。しかし、複数の PC をネットワークでつないで実現する高速化は、計算サーバなど用途が科学や工学の研究分野に限られるため、PC においては(1)(2)に比べて発展が遅かった。もちろん、開発目的が明確で予算が豊富なスーパーコンピュータではこれらの技術は早く確立され、ワークステーションでも一部の高速計算は専用ハードウェアで実現されている。その例として CPU では、重力場やクーロン場などの力計算に特化した超高速な Grape チップ[8]、分散メモリ間の高速通信では Myrinet [9]が有名である。ただ、これらのハードウェアはPCクラスター本体よりも高価であり、そのPCクラスターへの導入は経済性の点で現実味に欠ける。

汎用部品を利用することで非常に優れたコストパフォーマンスの高速計算を実現するPCクラスター計算機であるが、その弱点は OS(処理系)である Linux に標準実装されている TCP/IP プロトコルに由来する PC 間通信の遅さにあった。これは通信の安定性を図るために挿入される動作開始までの遅延時間に原因があり、従って通信経路を Fast Ethernet (100Mbits/s)から Gigabit (1000Mbits/s)に変更しても、普通の応用プログラムでの状況はそれほど変わらない。たとえば、CPU として同じ演算性能をもつ Pentium 4 と、SGI Origin や IBM Regatta などの RISC ワークステーションについて、頻繁にプロセッサ間通信が発生

するプログラムでベンチマークテストを行うと、通信のため発生するオーバーヘッド時間の差は歴然である。前者では通信時間がプロセッサの稼動しているCPU時間の半分を占めることもあるが、後者では数パーセント程度である。例として、電子の空間相関を表すバンド行列の解法が計算時間のほとんどを占める第一原理分子動力学(First-principle molecular dynamics)コードに対して、PC クラスター計算機では Table 1 上段に示すように、実計算時間(経過時間)と CPU 時間の比が 4CPU の並列時で約 1.4 である。

この大きな通信オーバーヘッドを解消することが、PCクラスター計算機の能力を十分に発揮させる上で大きな課題だった。日本では、異種のPCや通信方法を統合する高速クラスターの研究として SCore プロジェクトが行われ、Myrinet 通信を用いたベンチマークで高いスコアを記録している[10]。ここでは多くの研究室で手持ちである汎用の Ethernet を活用し、非 TCP/IP 通信で動作する GAMMA (Genoa Active Message Machine) [11] ソフトウェアを導入し、かつ高速 C/Fortran コンパイラを応用プログラムに利用する方法を紹介する。この方法により、通信の遅延時間を除き応用プログラムのレベルで高速演算を行うことに成功したが、その性能は SCore とほぼ同等(以上)である。

2. 高速通信システムのインストールと設定

ここではジェノバ大学(イタリア)で開発された非 TCP/IP の通信ソフトウェアで、フリーウェアの GAMMA をダウンロードしてインストールする手順と設定法を述べる。PC の OS は Linux であることが必要だが、市販の NIC(ネットワークインターフェース・カード)が利用できる。

はじめに、GAMMA プログラム本体と MPI、このプラットフォーム上で MPI を走らせるためのインターフェースプログラム、の3種をダウンロードする。注意すべき点は、サポートされている NIC が Fast Ethernet (100Mbps) /Gigabit Ethernet (1000Mbps)のいずれも限定され、また Linux のカーネル(OS の基本部分)は 2.4.21 版である。前者の制約はこの高速通信がハードウェアの基礎部分を利用するためであり、将来的にサポートされる NIC 種類の拡大が望まれる。後者はアセンブラ言語がソースコードに含まれるためで、必要であればカーネルのソースコードをダウンロードして各自の PC 環境でコンパイルする。

(a) カーネルのアップグレード

特定のカーネルが必要となるのは、GAMMA でデータ移動をつかさどる部分がアセンブラ言語で書かれているが、これをコンパイルする Linux 標準の GNU C コンパイラ gcc の表記法が版ごとに少し違うためである。カーネルのインストールはかなり微妙な作業で、失敗すると PC が再起動できなくなるが、これまでに確実に動作している現在のカーネルをデュアルブート環境として残すことで最悪の事態が避けられる。(つまり作成に失敗しても、以前の Linux カーネ

ルを選択することでリブートできる。ただし、成功後のカーネルの混用は動作一般が不安定となるため避けたほうがよい)。

作業は OS の一部を書き変えるため root 権限で行い、インストールする領域は以前とは別ディレクトリとする。ダウンロードしたソースコードをディレクトリ /usr/src にコピーして解凍すると、/usr/src/linux-2.4.21(または /usr/src/linux)が生成され、それ以下にプログラムが展開される。作業は、ソースコードの解凍、各種設定、コンパイル、書き込み、モジュールの作成から成り、やや難しいのは設定部分だけである(Appendix A を参照)。リブート後に、ビデオやネットワークが動作しなくなることがあるが、それは多くの場合、新しいカーネル用のビデオやネットワークドライバーがインストールされなかったためであり、後で追加すればよい。いずれにしても、ネット検索などで得られる作業マニュアルをよく読み作業を進めることが大切である。

(b) GAMMA のインストール

複数の PC がネットワークで接続されていること、NIC が指定された機種であることを確認する。この作業はカーネルのインストールに比べると易しく、設定、コンパイル、Linux カーネルへの組み込みから成る(Appendix B を参照)。注意する点は、設定で、ネットワークの1つを GAMMA 専用にすること、フロー制御をオンにすること、通信には rsh (パスワードなしで通信を許可する)を利用すること、そのため PC クラスター内ではファイアウォールをオフにする、などである。

ところで、ネットワークは、GAMMA 通信専用のものと、並列実行中のプログラムが他プロセッサ上のファイルを NFS(ネットワークファイルシステム)経由で参照するための、2系統を設定することが推奨される (Fig.1)。つまり、PC1台あたり2枚の NIC、ネットワーク系統ごとに1台のスイッチングハブ(データのあて先別の仕分け機能がある)が必要である。2枚の NIC はインストール設定での混乱を避けるために、別ブランドを選ぶほうが無難である(NIC の IP アドレスは、192.168.1.100、192.168.2.100 のように系統別に命名する)。なお、規模の小さな PC クラスターでは、通信効率の低下と引き換えに、1系統のネットワークで済ますこともできる。この場合、TCP/IP と GAMMA の通信が混在する設定を選択する。

(c) MPI/GAMMA のインストール

これは C/Fortran で書かれた、ユーザーの応用プログラム中から、PC 間の通信をつかさどる MPI を GAMMA 通信システムで利用できるようにするインターフェイスを含む。MPI パッケージは、Point-to-Point(2点間)通信と Collective(グループ内)通信の2種類のルーチンに大別され、このほかにランのはじめと終わりに初期化と終了処理をするもの、そして PC グループ管理などの付随ルーチン群がある。

上の(b)と同じホームページから、修正された MPI ソースコード mpich-1.1.2.tar.gz および mpigamma-(version).tar.gz をダウンロードして、この順序で解凍、一緒にコンパイルする (Appendix C)。ここでも、GAMMA との整合性をとるため、GNU C/Fortran コンパイラを使うべきである。

(d) GAMMA の設定

GAMMA のインストール後、添付のテストプログラム(サブディレクトリ apps/pingpong/内にある)をコンパイルして走らせ、正しく通信できることを確認する。また、MPI のインストール後も、簡単なテストプログラム(たとえば通信に参加する PC の数とランクを尋ねる MPI ルーチンだけを引用)を作成、走らせて検査を行う。そのためには初期設定が必要である(Appendix E)。

実際に GAMMA が正常に動作するまでには、幾つかのハードルを越えることになる。PC クラスターでネットワークが正常に起動して、NFS(ネットワークファイルシステム)が働き、マスターノードのホームディレクトリがすべての PC ノードから参照可能状態にあることが必要条件である。

ところで、GAMMA 運用中にタスクが原因を問わず異常終了したときは、ノード間で状態の同期が失われ通信が混乱している可能性がある。従って、マスターノードでリセットコマンド(gammaresetall)を投入し、状態を初期化することが推奨される。インストールマニュアルの最後に Trouble shooting 用の簡単なエラー対処法が書かれおり、トラブル時の参考になる。

3. 高速 C/Fortran コンパイラの利用

前節の Linux カーネル、GAMMA、MPI プログラムは GNU C/Fortran でコンパイルするように設計されている。しかし、一般的には高速性の追求やソースコードの書式(たとえば Fortran90 仕様)のため、市販のコンパイラを使う場合が多い。その際、ユーザープログラムをコンパイルして生成されるオブジェクトと MPI/GAMMA ライブラリの整合性を図る必要がある。

その整合性をとる要点は、(i) GNU gcc/g77 がコンパイル後のオブジェクトモジュールに 2 個のアンダースコア(__) を添付するので、市販コンパイラでもこの条件にあわせてコンパイルする、(ii) 入出力の定数などを定義するインクルードファイルは、ユーザ指定のコンパイラと GAMMA が生成する2系統あり、それぞれをこの順序で引用する、(iii) 線形計算ライブラリ BLAS、LAPACK、さらにその並列計算への拡張版 BLACS、SCALAPACK [12]を、2つのアンダースコアが添付されるようにユーザー自身でコンパイルする(つまり、コンパイラに標準添付のライブラリは使えない)、(iv) 系統の異なる C と Fortran コンパイラのあいだで、引数の相違を仲介するプログラムをリンクする [13]、である (Appendix D)。

ところで、PC 用の高速コンパイラとして広く利用されている pgf90 を応用プログラムのコン

パイルに用いた場合、PC グループ内での総和演算 MPI_Allreduce が、すべての論理演算(論理 and や論理 or)で常に.false.となる。これは、Fortran 処理系が論理定数の真(.true.)にどういう整数値を対応させるかの違いによる。この問題は、論理型の場合に論理変数を数値に置き換え、MPI_Allreduce に値を渡して処理を行わせる「ラッパープログラム」で回避できる(オプション-Munixlogical の指定では対応できない)。

4. 高速通信利用による演算性能の向上

それでは、実際の応用プログラムの計算はどれくらい高速化されるだろうか。はじめに基礎データとして、通信速度が送信データ量とともに向上する様子を Fig.2に示す。これは添付の pingpong プログラムを用いた2つのプロセッサ間通信の測定であり、[転送処理能力]=[データ量]/[所要時間]をデータ量に対して描いてある。使用した環境は Pentium 4 (3GHz)と 3Com996 NIC を装備したPCである。送信データ量が小さいときは、通信開始の遅延時間(Latency)があるために処理能力は小さく、データ量1バイトでの 0.6Mbits/s は遅延時間の 15μs に対応している。しかし、TCP/IP プロトコルでの大きな遅延時間 130～150μs がここで 15μs に減少するため、小さなデータの転送が頻繁に発生するプログラムでは、GAMMA の通信性能が顕著に現れる(以下 Table 1 で言及)。送信データ量が大きくなるにつれて処理能力は向上し、10^5 バイト付近で飽和する。非常に大きいデータ量での極限帯域幅は 706Mbits/s であり、Gigabit NIC の最大処理能力の 70%に達し、ネットワークの能力を非常に良く利用している [14]。

GAMMA のホームページ[11] で公表されている最高値は、ハードウェアの Myrinet (1.28Gbits/s)で遅延時間と極限帯域幅はそれぞれ 4.3μs と 1005Mbytes/s (BIP プラットフォームと Pentium Pro)であり、一方ソフトウェアの GAMMA でそれぞれ 8.5μs と 976Mbytes/s (Pentium III 1GHz と Netgear GA621 カード)である。応用プログラムで利用頻度が高い多次元行列の解法を並列プロセッサで行う場合、PC 間通信が頻繁に発生するため、遅延時間の短さが高速演算の鍵である。この点で Myrinet はやや優位に見えるが、他方、1回あたりの送信データ量が大きいプログラムでは Myrinet と GAMMA の違いは小さい。

次に、応用プログラムとして、密度汎関数法を用いた第一原理分子動力学コード Siesta [15]の実効性能を測定した結果を示そう。これは原子基底を用いた緊密結合(Tight-binding)型の量子力学分子動力学コードである。ここでは燃料電池に使われるイオン液体のイミダゾール分子 20 個(炭素、窒素、水素あわせて 180 原子)+プロトン1個の系 [16] についてのランタイムを Table 1 に示す。とくに注目したいことは、TCP/IP 通信による MPICH 利用の場合(上段)と、フロー制御をオンにした MPI /GAMMA(中段)との比較で、通信によるオーバーヘッドが 26 秒から 0.1秒に急減し、そのため、1ステップの実計算時間

(経過時間)が93秒から66秒に大きく減少する(両者のCPU時間はほぼ同じである)。また、GAMMA利用時に通信の衝突を防ぐフロー制御をオフにすると、実計算時間は115秒/stepと大きく増加し、研究で使う応用プログラムではフロー制御が必要である。

参考のため、標準的なRISCワークステーションであるIBM Power 4を先と同じ条件下で同数用いたときの測定結果を示す。これはすべてが64ビット設計のマシンである。この結果は、ワークステーションであるPower 4 (1.5GHz)とPentium 4 (3GHz)+MPI/GAMMAの実効性能がほぼ同じことを示している。オーバーヘッド時間が非常に小さい点も共通であり、RISCマシンのPower 4でも通信が非TCP/IPにより行われていることが確認される。

それでは使用するプロセッサ数に比例して、実計算速度は向上するだろうか? Figure 3に、プロセッサを複数台使用したときの演算能力(実行時間の逆数)を相対的に示す。プロセッサ数が4台までの領域では、処理能力はプロセッサ数に応じてほぼ線形に増加する。さらにプロセッサ数を増した場合も処理能力は上るが、その向上の度合いはゆるやかである。プログラム中の非並列化部分の割合をα、プロセッサ数をnとすると演算能力は$1/[\alpha+(1-\alpha)/n]$で与えられ、図から$\alpha \sim 0.1$と推定される。広い範囲にわたって演算能力がプロセッサ数にスケールするためには非並列化部分が小さいこと $\alpha \ll 1$ が必要であり、この理由からも非並列化部分である通信のオーバーヘッドが小さいことは大切である。

5. まとめ

この記事では、PCクラスター計算機の演算性能のボトルネックだった大きなオーバーヘッド時間がTCP/IPプロトコルによる通信に由来、それが通信ソフトウェアGAMMAの導入により通常のEthernetにおいて解消されることを示した。さらに、Linux標準のGNUコンパイラ以外の高速コンパイラとGAMMAシステムを一緒に利用する方法、その結果得られる応用プログラムに対する高い演算性能について記述した。GAMMA通信システムはプロセッサ間の通信で安定に稼動しており、ブロッキング通信(一連の転送が終了するまで次の動作に移らない)によるデータ転送の正しさを保障する。もし通信経路でデータが失われた場合にも、データを再送する機能を備えている。

PCクラスター計算機は、ここで述べた通信方法の非TCP/IPへの変更により、同数のRISCワークステーションに匹敵する演算性能をもつに至った。これは複雑な力場を計算する分子動力学計算に限定すれば、ベクトル型並列スーパーコンピュータに劣らない性能である。その半面で、PCクラスター計算機の運用とメインテナンスはユーザー自身が行う必要があり、これは実験装置の場合と似た状況である。結論として、PCクラスター計算機はコストパフォーマンスに優れた計算機シミュレーション研究の道具であり、豊かな科学研究の成果がこれを用いて生み出されていくことだろう。

謝辞:
MPI/GAMMA 通信システムの PC クラスター計算機へのインストールと初期運用について、開発者である Dr. Giuseppe Chiaccio の親切な助言とサポートに感謝いたします。また、第一原理分子動力学法の使用と PC クラスターの製作は、善甫康成氏との共同研究の成果であり、同氏に感謝いたします。

Appendix A: カーネルのアップグレード

ここでは手順の概略を示すが、詳細については Linux の種類、バージョンごとに多少の相違がある。詳しくはオンライン検索で入手できるマニュアル等を参照のこと。

#mv linux-2.4.21.tar.gz /usr/src

#cd /usr/src

#rm -rf linux (既存のリンクを削除。古いカーネルは保存する)

#tar xvzf linux-2.4.21.tar.gz

#ln -s linux-2.4.21 linux (新しいリンクの設定)

#cd /usr/src/linux

ここで、Makefileを編集して、#export INSTALL_PATH= /bootの#(コメント)記号を消す。

次に、すでに動作が確かな以前の設定を継続して利用するためコピーする、

#cp /usr/src/linux-(old)/configs/kernel-(old)-(arch).config /usr/src/linux/.config

ここで、(old)は以前のカーネルのバージョン番号、(arch)は i386 や i686 などである。

#make oldconfig (すべて設定値のままで)

#make xconfig (Processor Family、Symmetric multi-processing support on、内容がわからないときは、右横の HELP タブをクリックして情報を読む)

#make dep

#make clean

#make bzImage

もしここでエラーが出る場合、上で行った設定のいずれかが不適当なので、xconfig の設定を見直してから、make dep; clean; bzImage の部分を再実行する。

#mkdir /lib/modules/2.4.21

#/sbin/installkernel 2.4.21 arch/i386/boot/bzImage System.map

カーネルのインストールが済んだ後、ブートローダで LILO を使用しているときは/etc/lilo.conf を、GRUB のときは/etc/grub.conf を編集する。default= のあとの文字列で自動ブートするカーネルを指定するが、古いカーネルでのブートも残すこと。次に、モジュールを作成する。

#make modules

#make modules_install

#/sbin/depmod -a (エラーがでないことを確認する)

/sbin/mkinitrd --ifneeded /boot/initrd-2.4.21.img 2.4.21

使用するブートローダが LILO のときは #/sbin/lilo -v を、GRUB のときは

/sbin/grub-install /dev/had (IDE ハードディスクの1台目がブートディスクの場合)を実行する。新しいカーネルを利用するために、PCをリブートする。

Appendix B: GAMMA のインストール

#cp gamma-(version).tar.gz /usr/local

#tar xvzf gamma-(version).tar.gz

#cd /usr/local/gamma

#./configure

ここでは、NIC カードの選択、通信モードの設定を行う。"Flow control"は on を選択すること。そうでないと通信の衝突が多発して、現実の応用プログラムがまともに動作しない。

#cd /usr/src/linux
#make xconfig (X window 上で作業する)
上で、Linux が提供するネットワークドライバーを off に設定する。
#make dep
#cd /usr/local/gamma
#make
#make install
ここで、GAMMA が利用する PC の名称と NIC 固有番号のリストを/etc/gamma.conf に列記する。このファイルはすべての PC に必須。また、各ユーザーのホームにある.bash_profile に export PWD を追加する。最後に、ここで行った変更を Linux カーネルに反映させるため、
#cd /usr/src/linux
#make bzImage
#/sbin/installkernel 2.4.21 arch/i386/boot/bzImage System.map
を行う。この過程で/etc/grub.conf に不要な追記がされるので、どちらか一方を削除する。また、ブート時の自動初期化のため、/usr/local/bin/gammagetconfig を初期化スクリプトに追記する。そしてこれらの操作を反映させるため、最後に PC をリブートする。

Appendix C: MPI/GAMMA のインストール

#cp mpich-1.1.2.tar.gz mpigamma-(version).tar.gz /usr/local
#tar xvzf mpich-1.1.2.tar.gz (サブディレクトリ mpich が生成される)
#tar xvzf mpigamma-(version).tar.gz
#cd /usr/local/mpich
#./configure
#make (#make install は行わない)
make の途中でエラーが出たら、指示に従いインクルードファイルをコピーする。終了後、ライブラリとインクルードファイルは、それぞれ/usr/local/mpich/build/LINUX/gamma/以下の lib と include ディレクトリに収納されている。

Appendix D: ユーザー指定の C/Fortran コンパイラの利用

以下は、IA(Intel)-32 PC で定評のある Portland グループ製の pgcc /pgf90 コンパイラを MPI/GAMMA とあわせて利用する方法である。標準的なコンパイル・スクリプトは、

```
pgf90 -o ax1.out -Msecond_underscore -Mvect zzzzz.f90 ¥
-I/usr/local/pgi/linux86/.../include ¥
-I/usr/local/mpich/build/LINUX/gamma/include ¥
farg.o ¥
-L/usr/lib/libgamma.a ¥
-L/usr/ local/mpich/build/LINUX/gamma/lib -lfmpich -lmpich ¥
/usr/local/BLAS/libblas.a ¥
/usr/local/LAPACK/liblapack.a
```

最後の2つのライブラリは、Portland のコンパイラに添付のものではなく、2つのアンダースコアを添付するようにユーザー自身がコンパイルしたもの(これらは、gcc /g77 でコンパイルして

もよい)。farg.o は開発者が異なるコンパイラ間で引数の相違を橋渡しするソース farg.f のオブジェクトである [13]。

Appendix E: GAMMA 通信の設定

GAMMA 通信を開始するためには、以下の設定が必要である。

(i) 通信に参加するすべての PC のマシン名とその 0 番目 NIC の MAC アドレス(ハードウェアアドレス、/sbin/ifconfig で表示される eth0 以下の情報に含まれる 12 桁の数字)を 2 桁ずつに区切って、各 PC 上の/etc/gamma.conf に、リストアップする、

(ii) ユーザーのホームディレクトリの .bash_profile に、export PWD を追加する(bash シェルを使用の場合)、

(iii) ブート時に GAMMA を自動的に立ち上げるため、/etc/rc.d/rc.local (Linux ディストリビューションにより場所が異なる)に起動スクリプト gammagetconfig を追加する。

参考文献

[1] Purdue 大学、Beowulf プロジェクト, http://www.psych.purdue.edu/~beowulf/

[2] D. Becker、Beowulf プロジェクト(NASA), http://www.beowulf.org/

[3] 青山学院大学理工学部 http://www.phys.aoyama.ac.jp/~aoyama/

[4] 同志社大学工学部 http://www.is.doshisha.ac.jp/SMPP/

[5] M.Snir, S.Otto, S.Huss-Lederman, D.Walker, and J.Dongara, MPI – The Complete Reference (The MIT Press, Cambridge, 1998)

[6] 善甫康成、田中基彦、「第一原理分子動力学コードの整備と応用」、核融合科学研究所 Annual Review (2001).

[7] 田中基彦、"Plasma and Ionic Condensed Matters by Molecular Dynamics Simulations", http://dphysique.nifs.ac.jp/

[8] 国立天文台 Grape システム: http://www.astrogrape.org/

[9] Myrinet、http://www.myri.com/

[10] SCore Consortium、http://www.pccluster.org/

[11] G.Chiola and G.Ciaccio、"GAMMA Project: Genoa Active Message Machine" (Genoa 大学)、 http://www.disi.unige.it/project/gamma/

[12] 線形計算ライブラリのアーカイブ http://www.netlib.org/

[13] Portland Group http://www.pgroup.com/ FAQ の項を参照.

[14] 3Com996 NIC の最高値は処理能力が 816Mbits/s、遅延時間が 12μs。

[15] A. Garcia et al., Siesta (Spanish Initiative for Electronic Simulations with Thousands of Atoms), http://www.uam.es/departamentos/ciencias/siesta/

[16] 善甫康成、田中基彦、「第一原理分子動力学による物質科学」、核融合科学研究所ニュース 2004 年 2/3 月号.

[17] MPICH (MPI Chameleon)、Argonne National Laboratory、 http://www.mcs.anl.gov/

Table 1.

異なるPC間通信による、密度汎関数第一原理分子動力学コードSiesta [15]の実行時間。測定に用いた環境は、Pentium 4 (3GHz) とGigabit Ethernet (3Com996) のPCを4台、およびPGI Fortran pgf90であり、実計算時間は1ステップ (SCFループを1回)の計算に要する経過時間、オーバーヘッドは実計算時間とCPU時間の差、比は実計算時間とCPU時間の比である。MPICH-1.2はTCP/IPを利用するアルゴンヌ研究所のMPI [17]、MPI/GAMMAはジェノバ大学で開発された非TCP/IPの通信 [11]であり、FCは通信中でのフロー制御を意味する。比較のためRISCワークステーションIBM Power 4 (1.5GHz Regatta)の4台並列の場合を最下段に示す。

		実計算時間	CPU時間	オーバーヘッド	比
MPICH -1.2 TCP/IP		93 sec	67 sec	26 sec	1.39
MPI/ GAMMA	FC on FC off	66 sec 115 sec	66 sec 98 sec	0.1 sec 17sec	1.00 1.17
IBM Power4 / 1.5GHz		64 sec	64 sec	0.4sec	1.01

Figure Captions

Figure 1:
高速通信システムGAMMAをPCクラスターで利用するときに推奨されるネットワークの構成。1番目の系統は GAMMA によるデータ通信専用のギガビットネットワーク、2番目の系統は NFS マウントによる遠隔(異なる)PC からのファイル参照や PC クラスターの管理に使用する通常のTCP/IP ネットワークである(後者は Node 0 に置いたランの実行バイナリ参照のために必須)。

Figure 2
GAMMA 通信における、プロセッサ間で転送するデータのサイズ(単位:バイト)と転送速度(Mbits/sec)の関係。プロセッサは Pentium 4 (3GHz)で、ネットワークカードは 3Com996 (Gigabit Ethernet)。この測定では、通信開始の遅延時間は 15μs、大きなデータに対する極限転送速度は 706Mbits/sec であり、極限性能1Gbits/sec の約 70%である。

Figure 3:
第一原理分子動力学コード Siesta [15] における、GAMMA 通信下での実行速度とプロセッサ数の関係 (プロセッサ数1の能力を 1.0 として相対表示)。プロセッサは Pentium 4 (3GHz)、ネットワークカードは 3Com996 (Gigabit Ethernet)である。並列度とともに演算処理性能が上がり、とくに4プロセッサまではほぼ線形に向上する。

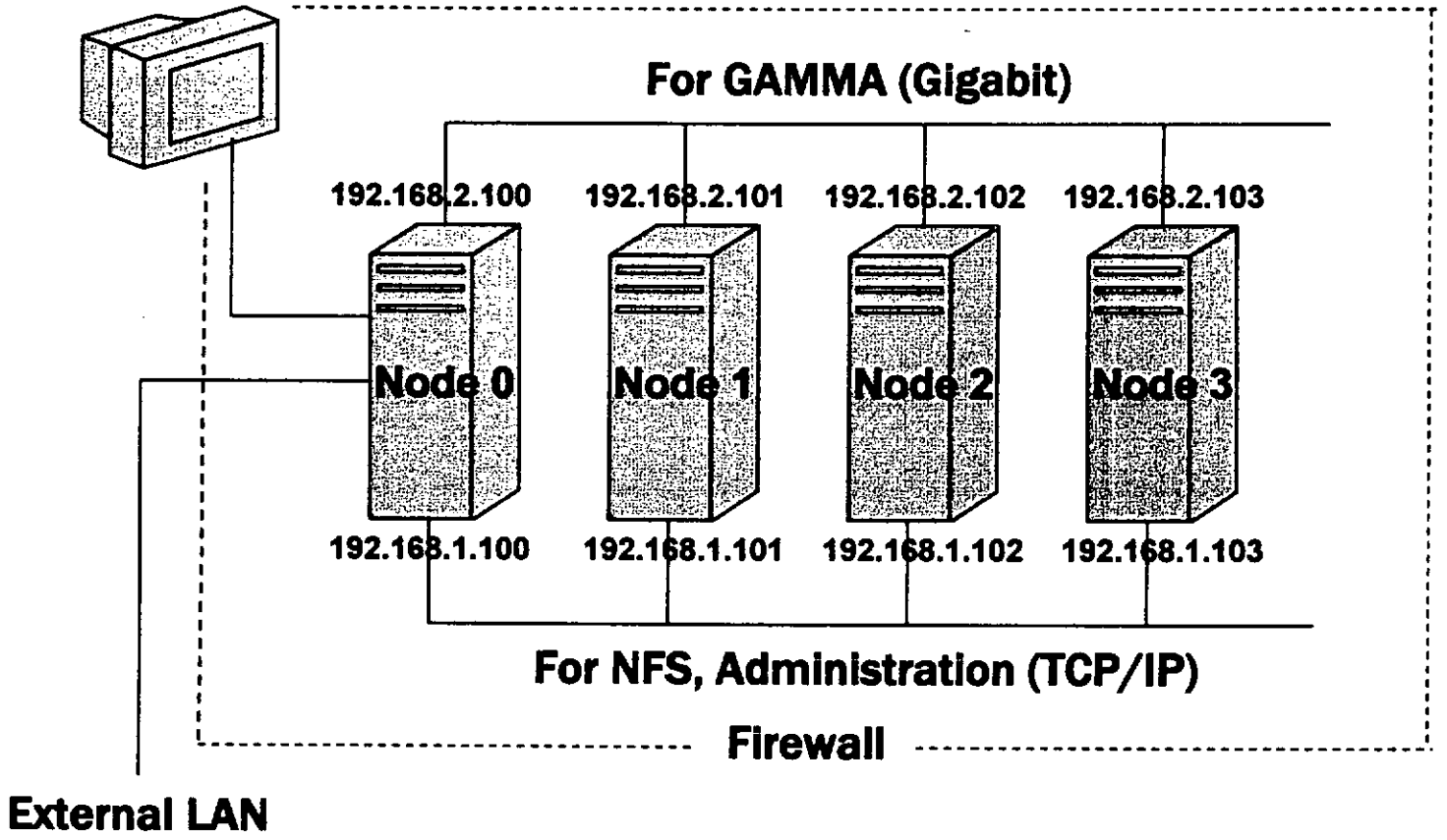

Fig.1 M.Tanaka

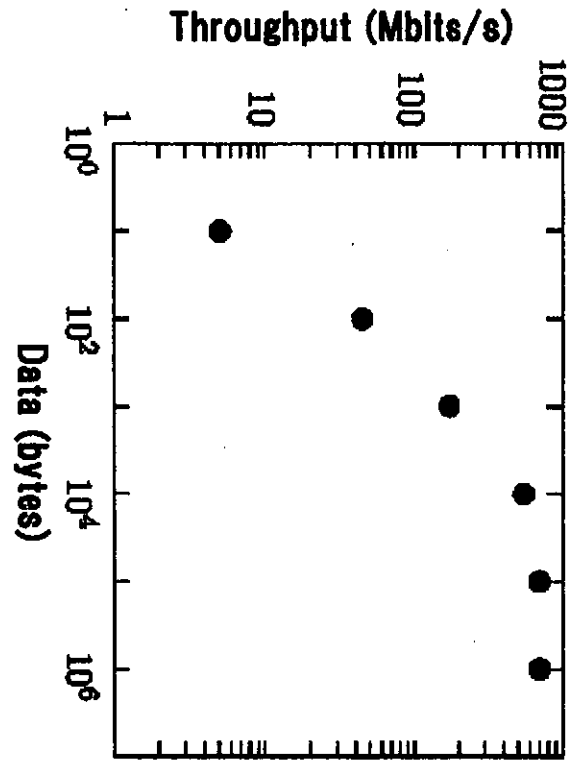

Fig.2 M.Tanaka

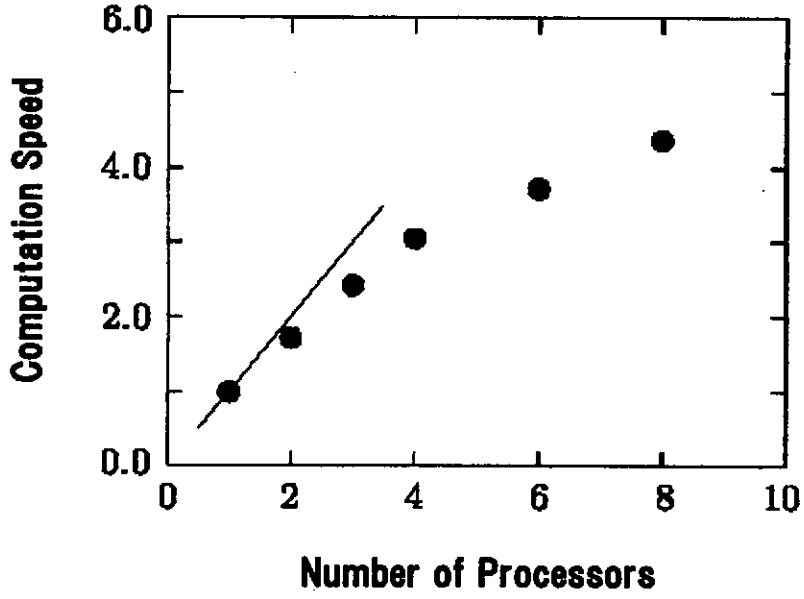

Fig.3 M.Tanaka

Publication List of NIFS-TECH Series

NIFS-TECH-1 H. Bolt and A. Miyahara,
Runaway–Electron –Materials Interaction Studies ; Mar. 1990

NIFS-TECH-2 S. Tanahashi and S. Yamada,
Dynamic Analysis of Compact Helical System Power Supply and Designs of Its Upgrade; Sep. 1991

NIFS-TECH-3 J. Fujita, K. Kawahata, S. Okajima, A. Mase, T. Suzuki, R. Kuwano, K. Mizuno, T. Nozokido, J.J.Chang and C.M.Mann,
Development of High Performance Schottky Barrier Diode and its Application to Plasma Diagnostics;
Oct. 1993 (in Japanese)

NIFS-TECH-4 K.V. Khlopenkov, S. Sudo, V.Yu. Sergeev,
Operation of the Lithium Pellet Injector; May 1996

NIFS-TECH-5 Nakanishi, H., Kojima, M. and Hidekuma, S.,
Distributed Processing and Network of Data Acquisition and Diagnostics Control for Large Helical Device (LHD); Nov. 1997

NIFS-TECH-6 Kojima, M., Nakanishi, H. and Hidekuma, S.,
Object-Oriented Design for LHD Data Acquisition Using Client-Server Model; Nov. 1997

NIFS-TECH-7 B.N. Wan, M. Goto and S. Morita,
Analysis of Visisble Spectral Lines in LHD Helium Discharge; June 1999

NIFS-TECH-8 Y. Zhao, Y. Torii, T. Mutoh, R. Kumazawa, F. Shimpo, T. Seki, K. Saito, G. Nomura and T. Watari,
ICRF Waveform Controlling; Dec. 1999

NIFS-TECH-9 H. Nakanishi, M. Kojima, M. Ohsuna, S. Komada, M. Emoto, H. Sugisaki, S. Sudo and LABCOM group,
Distributed Mass Data Acquisition System Based on PCs and Windows NT for LHD Fusion Plasma Experiment; Dec. 2000

NIFS-TECH-10 H. Nakanishi, M. Kojima and LABCOM Group,
Design for Real-Time Data Acquisition Based on Streaming Technology; Apr. 2001

NIFS-TECH-11 Y. Kondoh, M. Kondo, K. Shimoda, T. Takahashi, K. Itoh,
Innovative Direct Energy Conversion Systems Using Electronic Adiabatic Processes of Electron Fluid in Solid Conductors: New Plants of Electrical Power and Hydrogen Gas Resources without Environmental Pollutions; July 2001

NIFS-TECH-12 田中基彦
M. Tanaka
高速通信ソフトウェアを利用したPCクラスター計算機
Beowulf Cluster Equipped with High-Speed Communication Software