卒業論文

NAS パラレルベンチマークによる PC クラスタの性能評価（Ⅱ）

氏　　名：松田 一剛
学籍番号：2210990410-8
指導教員：山崎 勝弘 教授
提 出 日：2007 年 2 月 19 日

立命館大学 理工学部 情報学科

内容梗概

並列処理は大規模な問題でも計算時間が大幅に短縮できるというメリットがあり、欠かせない技術の 1 つといえる。また、本研究は PC クラスタで行うが、近年高性能な PC が安価で手に入るようになり、Myrinet などの高速なネットワーク環境が普及してきたことから高性能な PC クラスタの構築が可能になった。
また、ベンチマークの目的とは計算機システムの性能を計測することである。近年誰もが構築可能となった高性能な PC クラスタであるが、独自で構築した PC クラスタの性能は自分で測定しなければならない。ベンチマークを行うことで、独自で構築した PC クラスタの性能を計測できる上に、スーパーコンピュータとの性能比較も可能となる。

本論文では、NAS パラレルベンチマークにおける、本研究室で構築した SMP クラスタの性能評価について述べる。NAS パラレルベンチマークのカーネル問題における、EP、MG、CG、IS の 4 つの問題で性能評価を行った。EP ベンチマークに関しては 16 台実行で約 16 倍と理想的な並列効果を得られることができた。MG、CG ベンチマークは 16 台実行で約 5 倍とある程度の並列効果は得られた。IS ベンチマークはほぼ並列効果が得られず、並列計算に不向きであることがわかった。

目次

図目次

表目次

1. はじめに

並列処理は大量のデータを用いてより高速な計算を行う手法の 1 つである。処理時間を短縮するために複数のコンピュータで同時に処理を行うのが並列コンピューティングである。

現在、コンピュータの性能向上はすさまじい勢いで進んでいるが、それでも並列処理の応用分野である、気象予測、流体計算、遺伝子の解明、環境問題、データベース処理、ディジタル画像処理などの大規模な問題では、並列コンピュータが用いられている。

並列処理は、いまや欠かせない技術の一つといえる。プロセッサと主記憶からなるシステムが複数個互いに接続された形態で、プロセッサは他のプロセッサの主記憶の読み書きが出来ない分散メモリ環境の並列プログラミングライブラリとして、PVM（Parallel Virtual Machine）や MPI（Message Passing Interface）が有名である。また、複数のプロセッサがメモリバス/スイッチ経由で、主記憶に接続された共有メモリ環境の並列プログラミングモデルとして OpenMP が主流となってきている。

また近年では、汎用マイクロプロセッサの高性能化もすさまじく、一昔前までは手の出なかった高性能スーパーコンピュータに匹敵するマイクロプロセッサが、ワークステーションやパソコンに使用され、さらにネットワークにおいても著しい発展がある。その結果、汎用計算機と汎用ネットワークを用いた計算機クラスタがコストだけでなく、性能の面でも大きな意味を持つようになってきた。[3]

その中で、PC クラスタは誰もが比較的簡単に構築可能なコストパフォーマンスの高い並列計算環境といえることが出来る。しかし、その性能は実際に測定しなければ、性能比較、コストパフォーマンスの良し悪しは判断することが出来ない。独自で構築した PC クラスタの性能を測定するために用いられるのがベンチマークである。このベンチマークを行うことによって、スーパーコンピュータとの性能比較やコストパフォーマンスの比較も可能になるのである。

コンピュータの性能を計測するベンチマークであるが、コンピュータはプログラムによってさまざまな挙動を示すうえ、その用途もさまざまであるため、比較の指標を一意に決めることは困難である。そこで、各用途に応じた性能評価の指標を決定し、その尺度に基づいて性能を評価するということが重要になってくる。[1]

本研究では、ベンチマークの中でも特に PC クラスタに関係が深いと考えられる並列処理ベンチマークである NAS パラレルベンチマークを用いることによって、本研究室の PC クラスタの性能評価を行っている。今回は NAS Parallel Benchmarks の 5 つの Parallel Kernel Benchmarks のうち、EP、MG、CG、IS の 4 つのベンチマークを用いて処理速度の計測を行った。また、本研究室の PC クラスタは SMP クラスタであるので、ノードとプロセッサの割り当てを変化させてみての実験も行い、並列効果について比較を行った。2 章では PC クラスタと並列プログラミングについて、3 章、4 章では並列処理ベンチマーク、

NAS パラレルベンチマークについて、5 章で実験環境、実験結果、考察を記述した。

2. PC クラスタと並列プログラミング

2.1 PCクラスタ

PC クラスタとは、安価な PC にフリーの OS を載せ、それを高速のネットワークで複数接続し、分散して処理を行うシステムである。以前の並列計算機には、超高性能な処理を高速に実行するための高価で大規模なものしかなかった。1990 年前後に、数千から数万台の CPU を搭載する超並列計算機の開発が進む一方で、TCP/IP ベースのネットワークで接続された計算機群を仮想的に 1 台の並列計算機として利用する PVM（Parallel Virtual Machine）が開発された。これが PC クラスタの幕開けといえる。PC クラスタは、2～8 台の小規模なものから数千台の規模のものまで、予算や要求のよって自由に構成することができるうえに、PC 向けのプロセッサの低価格化と急速な性能向上で、非常にコストパフォーマンスに優れているといえる。また、PC の高性能化とともに、接続するネットワークの高速化が進み、専用の並列計算機並みのスループットを持つネットワークの構築が可能となり、性能面で比較してもスーパーコンピュータに引けをとらない高速計算の技術の 1 つであるといえる。また、複数のプロセス間でのデータをやりとりするために用いられるメッセージ通信操作の使用標準である MPI（Message Passing Interface）が広く普及し、現在では複数のプロセッサを搭載した SMP 型の WS や PC が比較的容易に入手可能となっており、これらをベースにした SMP クラスタが登場している。

本研究室の PC クラスタは計算ノードが 4 台で、それぞれのノードに 2 台のプロセッサが搭載されている。つまり 1 台の計算ノードに 4 台のプロセッサが搭載されている。これらのノードは Gigabit Ethernet Switch を介して、Server Host とつながっている。クラスタへの操作は Server Host を介して行う。

Server Host は Dual Intel Xeon プロセッサの 3GHz、メモリは 4GB の RAM であり、計算ノードは Dual Intel Xeon プロセッサの 3GHz、メモリは 4 GB の RAM である。

クラスタ上には,SCore は動作していない。その代わりに、MPICH や Intel compiler などの並列プログラミング環境がインストールされている。そのため SCore の起動などは行う必要がない。本研究室の PC クラスタの構成図を図 1 に示す。

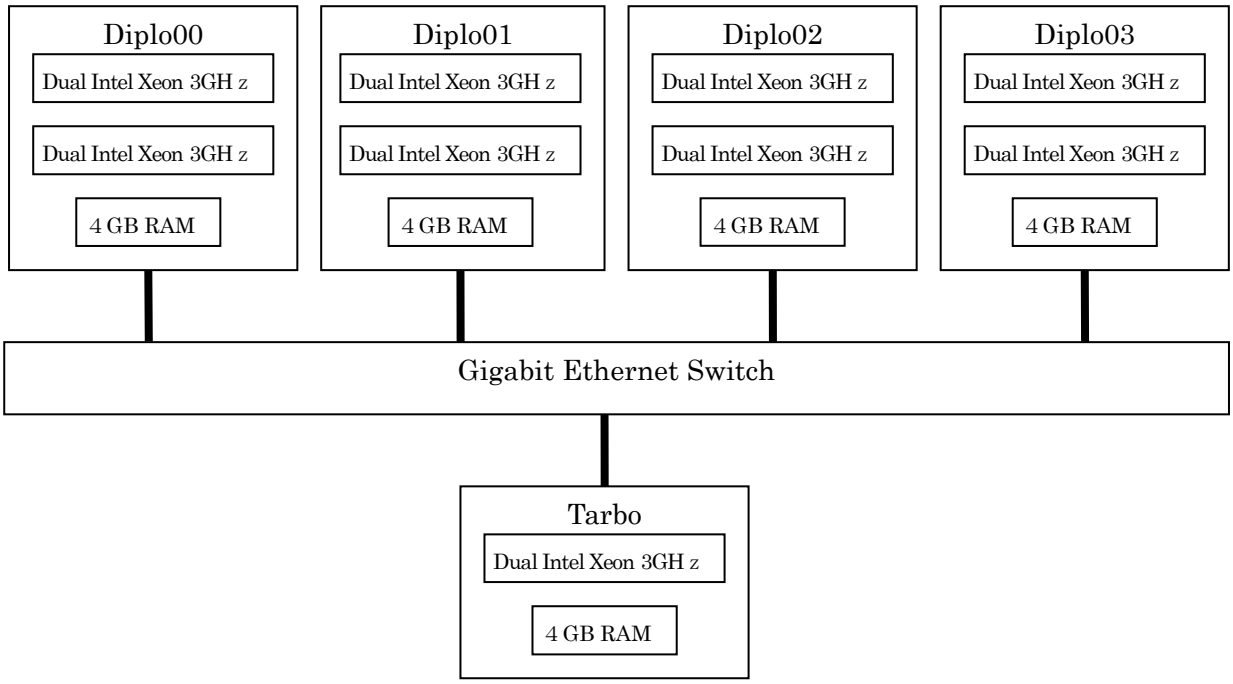

図 1：本研究室のＰＣクラスタの構成

2.1.1 SCore型クラスタ

SCore 型クラスタは RWC が開発したクラスタ上で必要なシステムソフトウェア（SCore クラスタシステムソフトウェア）を搭載しているクラスタである。既存のフリーソフトウェアの組み合わせで製作された Beowulf 型クラスタではインターネットで使用されている TCP/IP ネットワークプロトコルを使用しているために専用並列計算機に比べて通信性能の面で劣っている。また PC 上で独立したオペレーティングシステムが動いているため、すべての PC を取りまとめる機能がないという欠点がある。そこで SCore では、PM2 という通信ライブラリを用意して、ユーザが意識せずに通常よりも高速なネットワーク環境を利用することができるようにし、またクラスタ全体を管理するためのデータベースを作り、このデータを修正するだけでクラスタの設定の変更を可能にすることで Beowulf 型クラスタの欠点を取り除き、専用並列計算機と同等の性能をもつクラスタが実現できた。

SCore は WS や PC 等で稼働しているオペレーティングシステムである Linux 上に構築したトータルシステムソフトウェアである。SCore クラスタシステムソフトウェアを図 2 に示す。

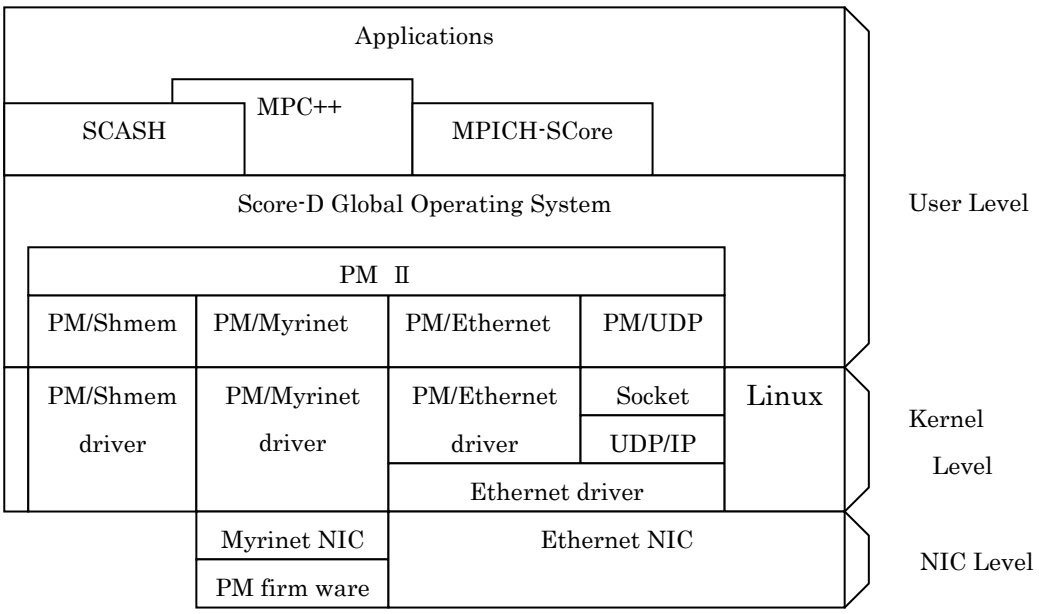

図 2：SCore クラスタシステムソフトウェア

SCore の特徴として、以下の点が挙げられる。

（1） 高性能通信

通信ハードウェアの性能を引き出すために PM2 通信ライブラリを開発した。PM2 は複数のネットワークハードウェアを支援できるよう,PM2API レイヤ、PM2 デバイスレイヤの 2 つのレイヤから構成されている。PV2 を使用することでクラスタにおける複数種類のネットワークに同一のインタフェースでアクセスすることができる。

（2） シングルシステムイメージによる効率的なシステム管理

Linux カーネル上に構築した Score-D グローバルオペレーティングシステムを用いて、クラスタの構成要素であるコンピュータを全て制御し、ギャングスケジューリングという手法を用いて複数の並列アプリケーションを効果的にスケジューリングすることができる。

（3）高利用性および高可用性

SCore-D グローバルオペレーティングシステムは、シングルユーザ、マルチユーザ、バッチジョブスケジューリングを提供している。また、チェックポイント･リスタート機能も提供しており、プログラムが途中で中断してもチェックポイントから再開することができる。

（4）シームレスなクラスタ環境

Ethernet でつながれた小規模クラスタから Gigabit Ethernet や Myrinet のような大規模クラスタまで連続的に利用することができ、ユーザはネットワークの種類を気にすることなくアプリケーションを実行することができる。

2.1.2 SMPクラスタ

SMP クラスタとは、共有メモリ型並列コンピュータのことであり、複数のプロセッサがメモリバス/スイッチ経由で、主記憶に接続される形態である。このアーキテクチャを有するシステムのことを SMP（対称型マルチプロセッサ）と呼ぶ。このシステムを有効に使用するためには、OS でのプロセッサ間の通信制御、プロセッサの割り当てが必要となる。この形態の特徴としては、複数のプロセスによる並列化のみではなく、スレッド単位の並列化が行われることである。これを用いる理由としては、同じプロセス中のスレッド間では同じメモリ領域にアクセスすることができるからである。つまり、マルチプロセスモデルでは、データの受け渡しは通信を介して行うが、マルチスレッドでは、得たいデータ領域に対して直接そのアドレスを参照することが出来るのである。共有メモリ型並列コンピュータは、プロセッサと主記憶から構成されるシステムが複数個互いに接続された形態である分散メモリ型並列コンピュータと比べて以下の２点で優れている。まず、プログラミング時にデータ分割を考慮する必要がなく、容易に自動並列化が可能である点、そしてメモリ間通信がないため、プロセッサ数が一桁程度の範囲では実効性能の理論最大性能に対する落ち込みが少ない点である。

2.2 並列プログラミング

2.2.1 分散メモリ並列プログラミング

（1）PVM(Parallel Virtual Machine)

PVM は、米国のオークリッジ国立研究所を中心に開発された、ネットワークに接続された異機種 UNIX コンピュータ群を、単一の並列コンピュータとして利用することを可能にするソフトウェアシステムである。これによって、多数のコンピュータの持つ計算パワーを、一つの大規模計算問題に結集して処理を行うことが出来る。対応する計算機には、パーソナルコンピュータから並列計算機／スーパーコンピュータまだ、多くの種類のものがある。

PVM では、異なったアーキテクチャのホスト間で整数や浮動小数のデータを交換するため、送信されるデータは XDR（eXternal Data Representation）Standard によりエンコードされる。送信側はデータをバッファにパックした後、送信を行い、受信側はバッファにデータを受信した後、アンパックを行いデータを取り出す。基本的には送受信は非同期で、送信と受信が一致しない場合でも送信がブロックされることはない。なお、このデータのパック／アンパックには、データをまとめて送受信することによりオーバーヘッドを減少

させる目的もある。

（2）MPI（Message Passing Interface）

MPI とは、分散メモリ型の並列計算機で複数のプロセス間でのデータをやり取りするために用いるメッセージ通信操作の仕様標準である。その名の通り、メッセージパッシング方式に基づいた仕様であり、近年では分散メモリ型のプログラミングインタフェースとして、最も標準的に用いられている。

MPI は、並列処理用のメッセージパッシングの規格である。メッセージパッシングとは、メッセージと呼ばれる特定のデータ形式の受け渡しと、これらの一元的な管理に基づく通信手段の一つである。並列計算機の各 CPU 間では数値計算の過程で多くのデータの交換が行われているが、その交換手順を定めたものが MPI である。多くの並列計算機に標準実装され、またフリーウェアとして MPICH、LAM 等のパッケージが入手可能である。

2.2.2 共有メモリ並列プログラミング

共有メモリ並列プログラミングのための言語として OpenMP が挙げられる。OpenMP は、1997 年に OpenMP Architecture Review Board が Fortran をベースとして API 仕様で開発したものである。その後 C/C++などにおいても API 仕様で開発された。

OpenMP は Fortran や C/C++をベースにコンパイラ指示文、ライブラリ、環境変数によって並列処理を行えるように拡張したものである。したがって、プログラムのソースコードの中に逐次部分と並列部分が存在する。

OpenMP の実行モデルには fork-join 並列実行モデルを用いている。プログラムはまず、マスタースレッドという単一のスレッドのみで実行が開始される。プログラムの実行が並列指示文に達するとスレーブスレッドと呼ばれる複数のスレッドを生成し、並列指示文で指定されている範囲の処理が全スレッドにおいて実行される。全てのスレッドにより並列実行される部分を並列リージョン、マスタースレッドのみで実行される部分を逐次リージョンと呼ぶ。図 3 に fork-join モデルを示す。

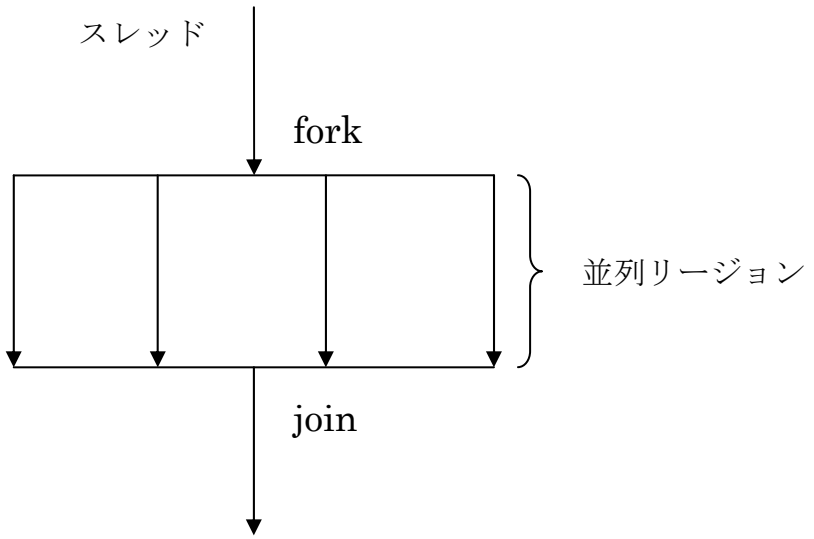

図 3 : fork-join モデル

2.3 ハイブリッド並列化

ハイブリッド並列化とは、OpenMP などによるスレッド並列と MPI などによるプロセス並列を組み合わせて並列化を行う手法である。これは粗粒戸の並列化をプロセス並列で行い、細粒度の並列化をスレッド並列で行うケースが多く、コーディングから見た場合で最も多い適用例としては、多重構造のループにおいて外側のループ処理をプロセス並列で行い、内側のループ処理をスレッド並列にする場合である。実行時の形態としては、ノード間をプロセス並列、ノード内をスレッド並列にするケースが一般的である。このとき、OpenMPのようなスレッド形の並列処理とMPIのような通信を伴う並列処理を混在するときには、ノード内で並列処理を行う適当なブロックを並列実行した後、マスタースレッドのみがノードを代表して通信を行うことにより、安全な通信が可能になるということに特に注意が必要である。

OpenMP は、プログラム中で並列実行の指示があった時点で、複数のスレッドを生成し、並列実行部分が終了した時点でマスタースレッド以外のスレッドは消滅する。したがって、OpenMPディレクティブで指定される並列に実行される部分領域内にMPI関数がなければ、MPI 通信は自然にマスタースレッドが行うことになるため、安全に通信が行える。

MPI と OpenMP によるハイブリッド並列化を行う際には、MPI で並列化されているプログラムに OpenMP の並列化ディレクティブを追加することによって並列化を行う。このとき、プログラム中で対象となるループの並列化可能性やプログラムの整合性はユーザが判断する。また、SMP クラスタがベクトル樹の場合には、並列ループ長がパフォーマンス

に大きく影響するので、最適なループ長の確保なども考慮に入れる必要がある。

また、通常はパフォーマンスの観点からハイブリッド並列化をする場合が多いが、メモリ使用の観点からも、1 ノード内で複数のプロセスを起動させると、ノードあたりのメモリ使用量が増えてしまう場合、ノード内では 1 プロセス/複数スレッドにして、メモリ量を抑制するためにハイブリッド並列化をする場合もある。ハイブリッド並列化モデルを図 4 に示す。

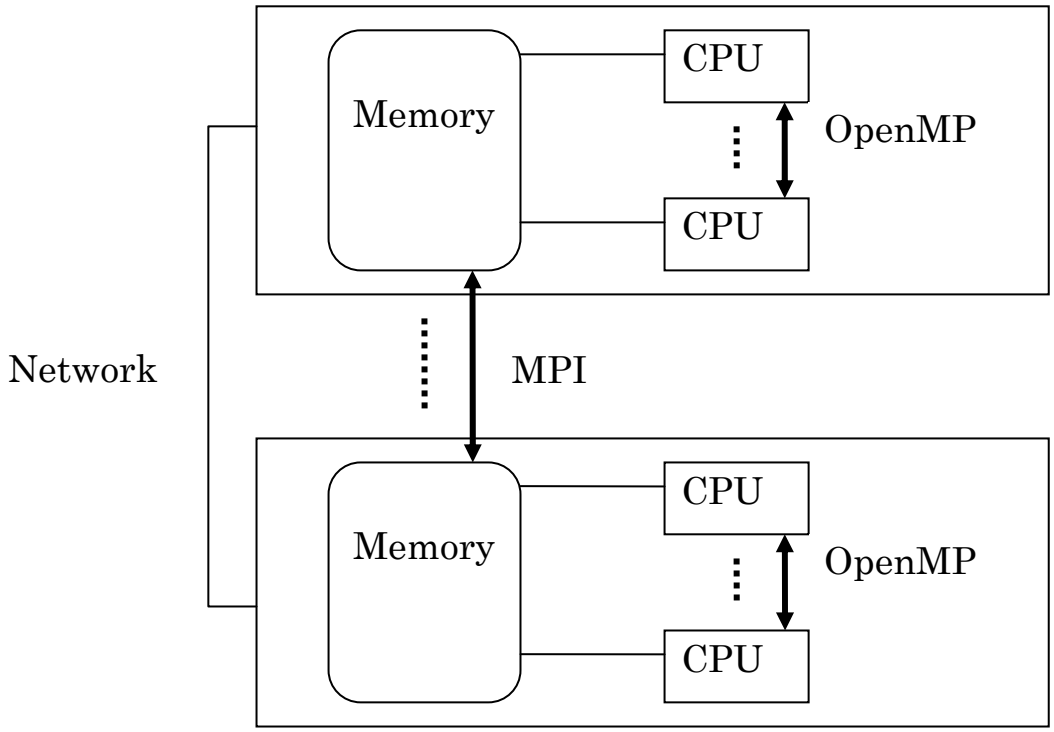

図 4： ハイブリッド並列化モデル

3. 並列処理ベンチマーク

3.1 ベンチマーク

ベンチマークとは、コンピュータのハードウェアやソフトウェアの処理速度を計測する試験、およびそれによる性能評価のことである。試験用に作成されたソフトウェアを実行し、処理の完了までにかかる時間を計測することで、製品間の比較を行う。また、ある特定の装置やソフトウェアの性能を計測するものと、コンピュータシステム全体の処理性能を評価するものもある。ただ、コンピュータはプログラムによってさまざまな挙動を示すうえ、その用途もさまざまであるため、比較の指標を一意に決めることは困難である。そこで、各用途に応じた性能評価の指標を決定し、その尺度に基づいて性能を評価するということが重要になってくる。その結果として、さまざまな用途に応じた性能を評価するためのベンチマークが必要となるのである。ベンチマークの対象となる機能を分類すると以下のようになる。

- 演算性能ベンチマーク
- I/O 性能ベンチマーク
- グラフィックス関連性能ベンチマーク
- ネットワーク関連性能ベンチマーク
- データベース性能ベンチマーク
- アプリケーション指向ベンチマーク
- 並列/大規模計算ベンチマーク

この中でも、特に PC クラスタの性能評価において重要なものに、演算性能ベンチマーク、並列計算ベンチマーク、アプリケーション指向ベンチマーク、ネットワーク性能ベンチマークなどが考えられる。その中で並列計算ベンチマークとして、NAS Parallel Benchmark、ScaLAPACK、姫野ベンチが挙げられる。また、ネットワーク性能ベンチマークとしては netperf が挙げられる。

3.2 NASパラレルベンチマーク

NAS Parallel Benchmarks（NPB）は並列コンピュータのためのベンチマークであり、NASA Ames Research Center で開発された。NPB は、5 つの Parallel Kernel Benchmarks と 3 つの Parallel CFD(Computational Fluid Dynamics)Application Benchmarks から構成されている。それぞれの問題に対し、問題サイズや反復回数が異なる 5 つのクラス（A、B、C、W（Workstation）、S（Sample））が定義されており、それぞれの環境に応じたベンチマークを行うことが出来る。NPB は、並列コンピュータの実効性能を知るうえで、権威あるベンチマークの 1 つであるといえる。

また、NASA Ames Research Center より定期的に WWW などで発行されているレポート（NAS Parallel Benchmark Results）では、プログラム実行に要した経過時間（elapsed time）と、Y-MP/1 または C90/1 との性能比などについて報告がなされており、大規模計算機と PC クラスタの性能比較も可能である。NPB の対象問題を表 1 に示す。

表 1：NAS Parallel Benchmarks の対象問題

Kernel	
EP	乗算合同法による一様乱数、正規乱数の生成
MG	簡略化されたマルチグリッド法のカーネル
CG	正値対称な大規模疎行列の最小固有値を求めるための共役勾配法
FT	FFTを用いた 3 次元偏微分方程式の解法
IS	大規模整数ソート
Simulated CFD Application Benchmarks	
LU	Synmetric SOR iteration による CFD アプリケーション
SP	Scalar ADI iteration による CFD アプリケーション
BT	5x5 block size ADI iteration による CFD アプリケーション

3.3 ScaLAPACK

ScaLAPACK は、テネシー大学の Jack Dongarra によって開発された、線形代数問題の LU 分解を行うベンチマーク LINPACK の並列版である。LINPACK は、N×N 行列について、単精度および倍精度で y(i)=y(i)+a＊x(i)を演算する。演算の特性上、ベクトル化による効果が顕著に表れるベンチマークである。また、LINPACK は世界のスーパーコンピュータランキングである「TOP500 Supercomputer」のベンチマークに採用されている。

ScaLAPACK では、行列を分割し、それぞれを 1 つにプロセスが担当する。プロセス間通信には MPI もしくは PVM を用い、これによって PC クラスタでも実行･計測が可能になっている。

3.4 姫野ベンチマーク

姫野ベンチマークは、情報基盤センター・センター長の姫野龍太郎氏が非圧縮流体解析コードの性能評価のために考えたベンチマークで、ポアッソン方程式解法をヤコビの反復法で解く場合に主要なループの処理速度を計測するものである。

特徴としては、コードは非常に短く簡単にコンパイル･実行できるので、即座に実測速度（何 MFLOPS）を求めることが可能なことである。また、Windows や Machintosh で利用可能である。

姫野ベンチマークテストは主に計算機のメモリバンド幅などの性能を測定するベンチマークテストである。C 版と Fortran 版がある。パラメータは計算サイズと分割数の種類の 2

種類だけである。計算サイズには XS、S、M、L、XL がある。計算サイズの大きさを表 2 に示す。これはプログラム中に用いられている 3 次元配列の要素数を表すものである。

表 2：姫野ベンチの計算サイズの大きさ（要素数）

記号	計算サイズ
XS	65×33×33
S	129×65×65
M	257×129×129
L	513×257×257
XL	1025×513×513

分割数の種類とは並列計算を行う場合、上の表の 3 つの次元を含む配列をどのようにそれぞれのマシンに分割するかという意味である。なお、分割は 3 つの次元にそれぞれ任意で行うことができ、その分割数の積は計算に使用する CPU 数と同じでなくてはならない。

3.5 Netperf

Netperf は、LAN を含むさまざまなタイプのネットワークの性能を計測するベンチマークソフトである。UNIX 版と Windows 版が用意されていて、相互の版の間でも利用可能である。一般にネットワーク性能ベンチマークは実ネットワークの性能評価、ネットワーク・スタックの性能評価などに利用される。また、ネットワークの性能は以下の 2 点で決定される。

・スループット：単位時間当たりの処理量

・遅延：レイテンシ（待ち時間）と RTT（Round Trip Time：往復時間）

Ethernet のようなパケット転送型ネットワークにおけるスループットは、通信ノード間の回線のバンド幅および中間ノード間のネットワークスタックの性能を示している。また、遅延には接続時のレイテンシと、ネットワークを介した場合の RRT が含まれる。

バルクデータを送受信する場合には、性能はスループットだけで決定され、遅延性能はほぼ無視することができる。これは、遅延要素が大量のデータ転送にかかる時間に隠蔽されてしまうためである。一方で、telnet などを用いたインタラクティブなジョブでは遅延性能が主要な要素となる。

Netperf はネットワークに接続された 2 ノード間における TCP および UDP を利用した通信のスループットおよび、ネットワークの遅延を計測するベンチマークである。計測時にはメモリ上のデータについて転送を実行するので、ハードディスク上のファイルアクセスは行わない。そのため、巨大なインターネットでも小規模な LAN であっても、2 ノード間におけるスループットおよびネットワーク遅延を計測できるのである。

4. NASパラレルベンチマーク

4.1 EP

EP ベンチマーク（The Embarrassingly Parallel Benchmark）は、並列計算に適しており、指定されたスキームに従い生成された、多数のガウス分布に従う擬似乱数を用いて、2 次元の統計情報を蓄積するものである。この問題は、モンテカルロ法を用いた応用プログラムによく見られる。この問題を解く際に転送をほとんど必要としないことから、ある意味において本ベンチマークは、システムが持つ浮動小数点演算の最大実効性能を示すものであるといえる。

4.2 MG

MG ベンチマーク（Multigrid Benchmark）は、3 次元ポアソン方程式を簡略化したマルチグリッド法で解くものである。B クラスでは、A クラスと同じ大きさの格子を使用しているが、内部のループの繰り返し数が A クラスより大きくなっている。

4.3 CG

CG ベンチマーク（Conjugate Gradient Benchmark）は、大規模で正値対称な疎行列の最小固有値の近似値を共役勾配法を用いて解くものである。このカーネルは非構造格子を用いたアプリケーションでよく見られる。

4.4 IS

IS ベンチマーク（Integer Sort Benchmark）は、パーティクル法を使用したアプリケーションにおいて重要なソートを評価するものである。このタイプのアプリケーションは、物理におけるパーティクルをあるセルに割り当てて、パーティクルがセルから流れ出るかどうかを見る、particle-in-cell 法のアプリケーションに似ているといえる。ソートは、パーティクルを再び適切なセルに割り当てる際に使用される。

5. NASパラレルベンチマークによる性能評価

5.1 実験環境

NAS Parallel Benchmarks（NPB）は、NASA によって開発されたメモリ分散型並列コンピュータ用のベンチマークである。今回はこれを用いて本研究室の PC クラスタ Diplo 上で実験を行い、Diplo の性能評価を行った。

また NPB は､MPI ライブラリを用いた並列計算を行ってその処理能力を計測する仕組みになっているので、このベンチマークを動かす場合には実装したパッケージがインストールされている必要がある。NPB のインストール方法を以下に示す。

まず NPB のサイトからプログラムをダウンロードし、展開する。すると、NPB3.2 というディレクトリが生成される。

（1）設定ファイルの変更

make する前に各環境に応じて設定ファイルを変更する必要がある。NPB3.2/config/に make.def.template というファイルがあるので、それをコピーして make.def として、MPICC や MPIF77、MPI のライブラリのパスなどを正しく設定する。

（2）コンパイル

make.def を正しく変更した後、NPB3.2 ディレクトリで次のように make する。NPB ではさまざまなベンチマークのそれぞれに対して、実行するプロセス数、問題のクラスを指定することができる。それらは全て別々の実行ファイルとなるので、対象問題、プロセス数、クラスを指定して make する必要がある。たとえば EP ベンチマークの A クラスを 16 プロセスで実行する場合には、次のようになる。

```
make EP NPROCS=16 CLASS=A
```

make に成功すると実行ファイルは bin 内に生成される。

（3）実行

EP ベンチマークの A クラスを 16 プロセスで実行する場合、bin 内で次のように実行する。

```
mpirun –mp16 ep.A.16
```

5.2 実験内容

今回は、NPB の 5 つの Parallel Kernel Benchmarks のうち、EP、MG 、CG、IS の 4 つのベンチマークを用いて PC クラスタの処理速度の計測を行い、並列効果について調べてみた。クラスは A、B、C を使用した。また、本研究室の PC クラスタ Diplo は SMP クラスタであるので、ノード数とプロセッサ数の割り当てを変化させることができる。よって割り当ての際、変化させることのできるものについてはそれぞれ処理速度の計測を行い、おのおのの並列効果について調べ比較を行った。ノードとプロセッサの割り当てを表 3 に示す。

表 3： ノードとプロセッサの割り当て

プロセッサ数		1	2	4	8	16
ノード×プロセッサ	ノード内割り当て	1×1	1×2	1×4	2×4	4×4
	ノード間割り当て		2×1	4×1	4×2	

上の表のようにプロセッサ数 2、4、8 の場合に割り当てを変化させることができるので、それぞれ 2 通りの実験を行った。また、プロセッサ数 2、4、8 の時、ノードとプロセッサの割り当てが 1×2、1×4、2×4 の場合をノード内割り当て、2×1、4×1、4×2 の場合をノード間割り当てとして表記する。

5.3　実験結果

ノード数とプロセッサ数の割り当てで、ノード内割り当てを A、B、C、ノード間割り当てを A’、B’、C’で表した。

5.3.1　EPベンチマーク実験結果

EP ベンチマークに関して実行時間を表 4、速度向上比を図 5 に示す。プロセッサ数 16 の時、プロセッサ数 1 の時と比べて A クラスで 15.8 倍、B クラスで 15.6 倍、C クラスで 15.8 倍の速度向上が得られた。

表 4： EP ベンチマークの実行時間（秒）

プロセッサ数		1	2	4	8	16
A クラス	ノード内割り当て(A)	62.24	31.42	15.77	7.89	3.95
	ノード間割り当て(A’)		31.27	15.72	7.87	
B クラス	ノード内割り当て(B)	249.84	125.49	62.98	31.58	16.01
	ノード間割り当て(B’)		125.69	62.85	31.52	
C クラス	ノード内割り当て(C)	994.47	502.98	251.71	125.79	63.08
	ノード間割り当て(C’)		501.93	251.66	125.76	

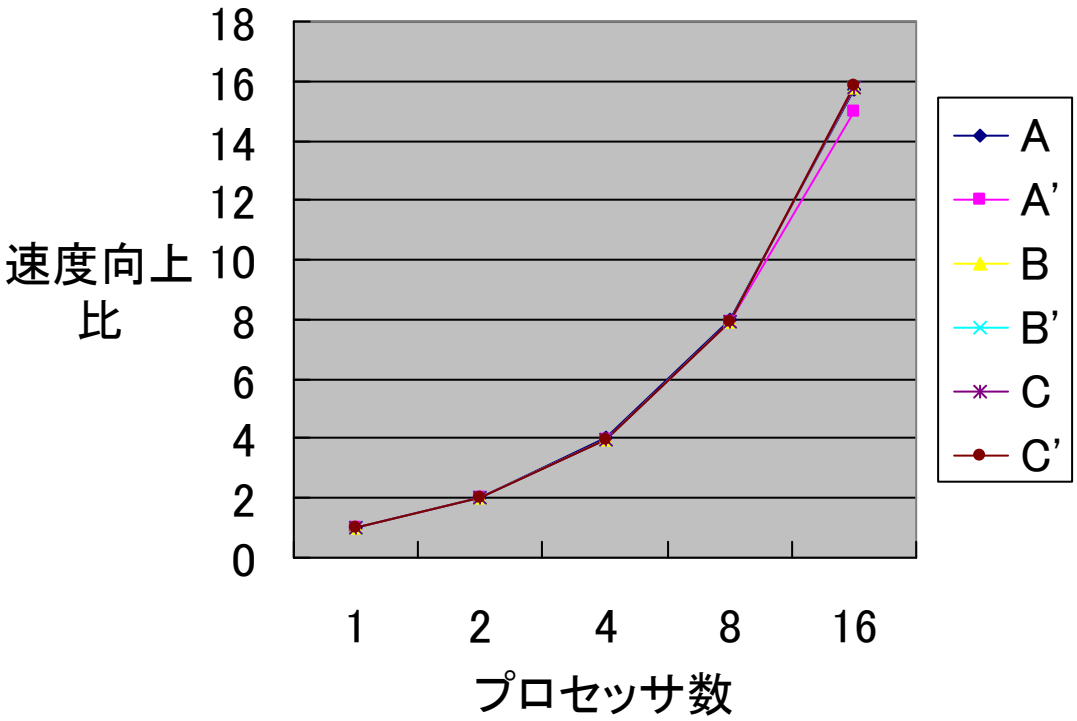

図 5： ＥＰベンチマークの速度向上比

5.3.2　MGベンチマーク実行結果

MG ベンチマークに関して実行時間を表 5、速度向上比を図 6 に示す。また、プロセッサ数 1 の C クラスでは実行できなかった。プロセッサ数 16 の時、プロセッサ数 1 の時と比べて A クラスで 5.5 倍、B クラスで 5.6 倍の速度向上が得られた。また、C クラスでは、プロセッサ数 16 の時、プロセッサ数 2 の時の 3.1 倍の速度向上が得られた。

表 5：MG ベンチマーク実行時間（秒）

プロセッサ数		1	2	4	8	16
A クラス	ノード内割り当て(A)	6.52	4.01	3.36	2.02	1.19
	ノード間割り当て(A')		4.08	2.29	1.41	
B クラス	ノード内割り当て(B)	30.43	18.82	16.19	9.55	5.41
	ノード間割り当て(B')		19.03	10.84	7.32	
C クラス	ノード内割り当て(C)		138.36	138.16	78.74	45.23
	ノード間割り当て(C')		139.71	77.89	59.93	

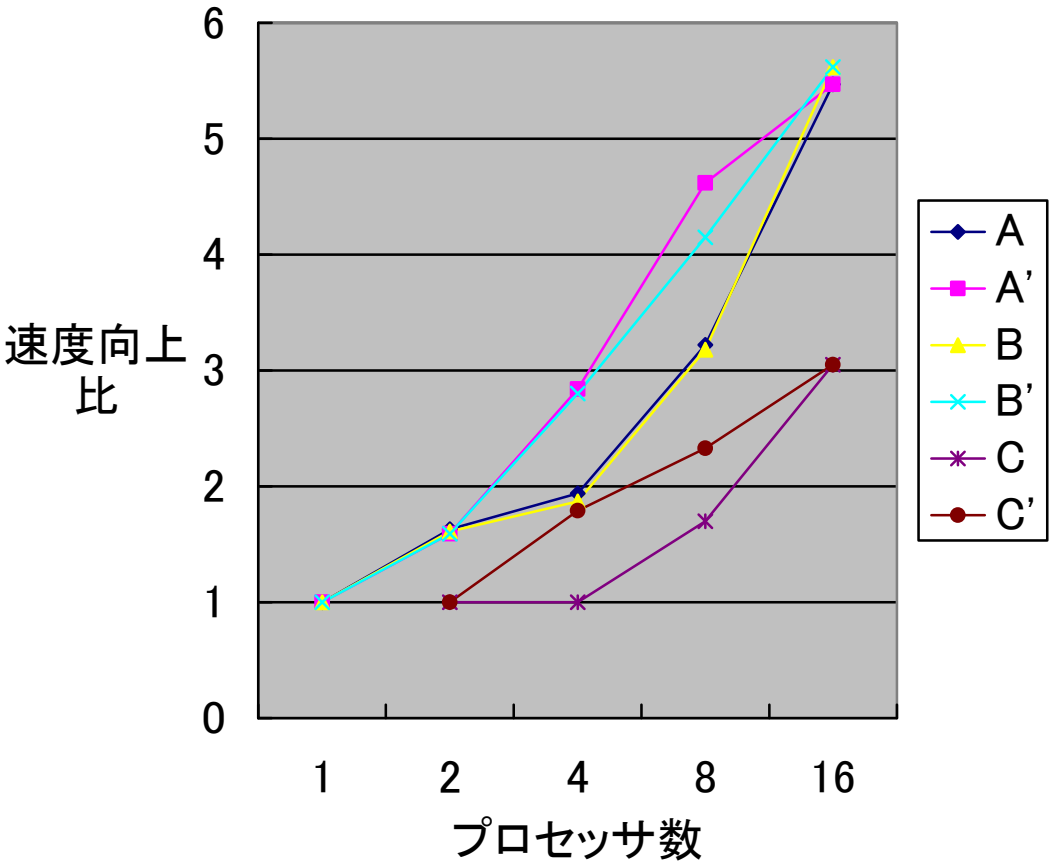

図 6：MG ベンチマークの速度向上比

5.3.3　CGベンチマーク実行結果

CG ベンチマークに関して実行時間を表 6、速度向上比を図 7 に示す。プロセッサ数 16 の時、プロセッサ数 1 の時と比べて A クラスで 3.9 倍、B クラスで 3.7 倍、C クラスで 5.5 倍の速度向上が得られた。

表 6：CG ベンチマーク実行時間（秒）

プロセッサ数		1	2	4	8	16
A クラス	ノード内割り当て(A)	5.36	3.24	2.55	1.89	1.38
	ノード間割り当て(A')		3.23	2.23	1.68	
B クラス	ノード内割り当て(B)	209.16	119.43	97.01	70.84	57.04
	ノード間割り当て(B')		119.43	76.91	58.16	
C クラス	ノード内割り当て(C)	696.66	317.67	242.40	172.69	126.09
	ノード間割り当て(C')		317.61	194.73	126.47	

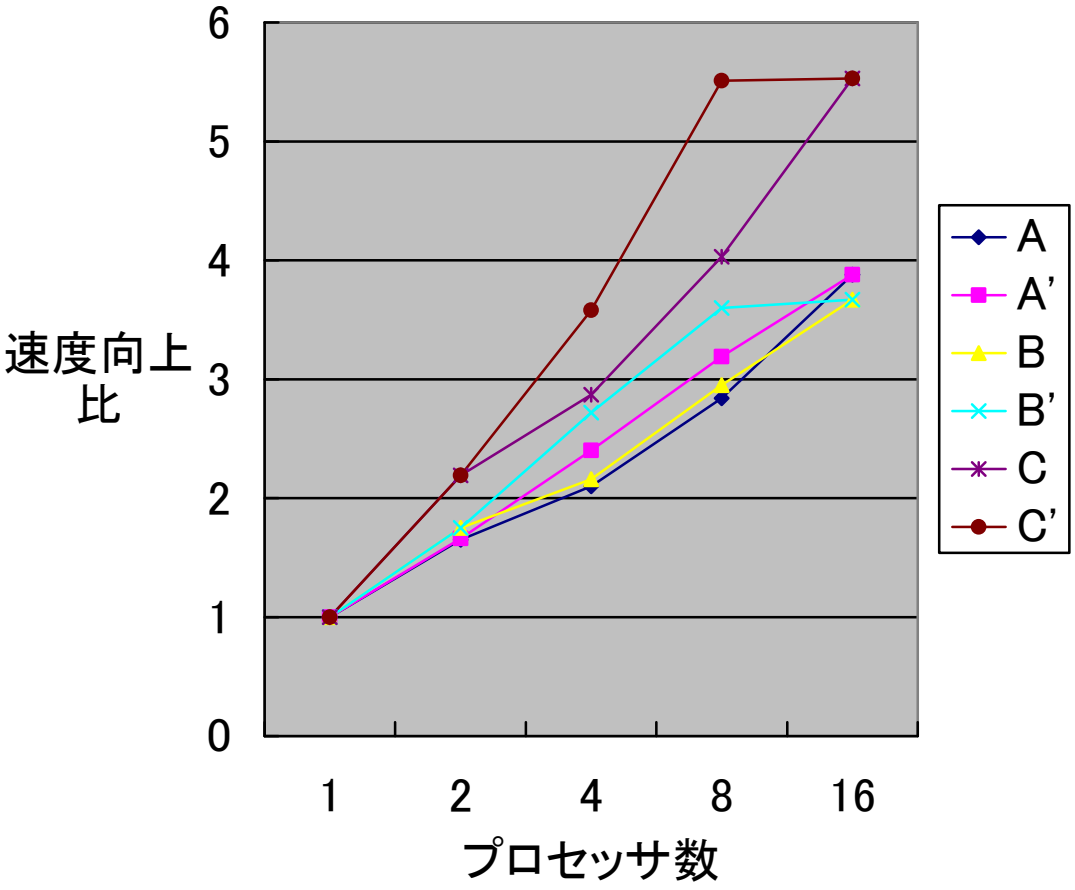

図 7：CG ベンチマークの速度向上比

5.3.4　ISベンチマークの実験結果

IS ベンチマークに関して実行時間を表 7、速度向上比を図 8 に示す。プロセッサ数 16 の時、プロセッサ数 1 の時と比べて A クラスで 1.4 倍、B クラスで 1.4 倍、C クラスで 1.5 倍の速度向上が得られた。

表 7：IS ベンチマーク実行時間（秒）

プロセッサ数		1	2	4	8	16
A クラス	ノード内割り当て(A)	2.39	2.99	2.33	2.02	1.73
	ノード間割り当て(A')		3.05	2.62	2.11	
B クラス	ノード内割り当て(B)	11.07	12.41	9.42	8.33	7.65
	ノード間割り当て(B')		12.62	10.83	9.17	
C クラス	ノード内割り当て(C)	47.04	51.06	37.80	33.73	30.90
	ノード間割り当て(C')		52.01	46.23	36.06	

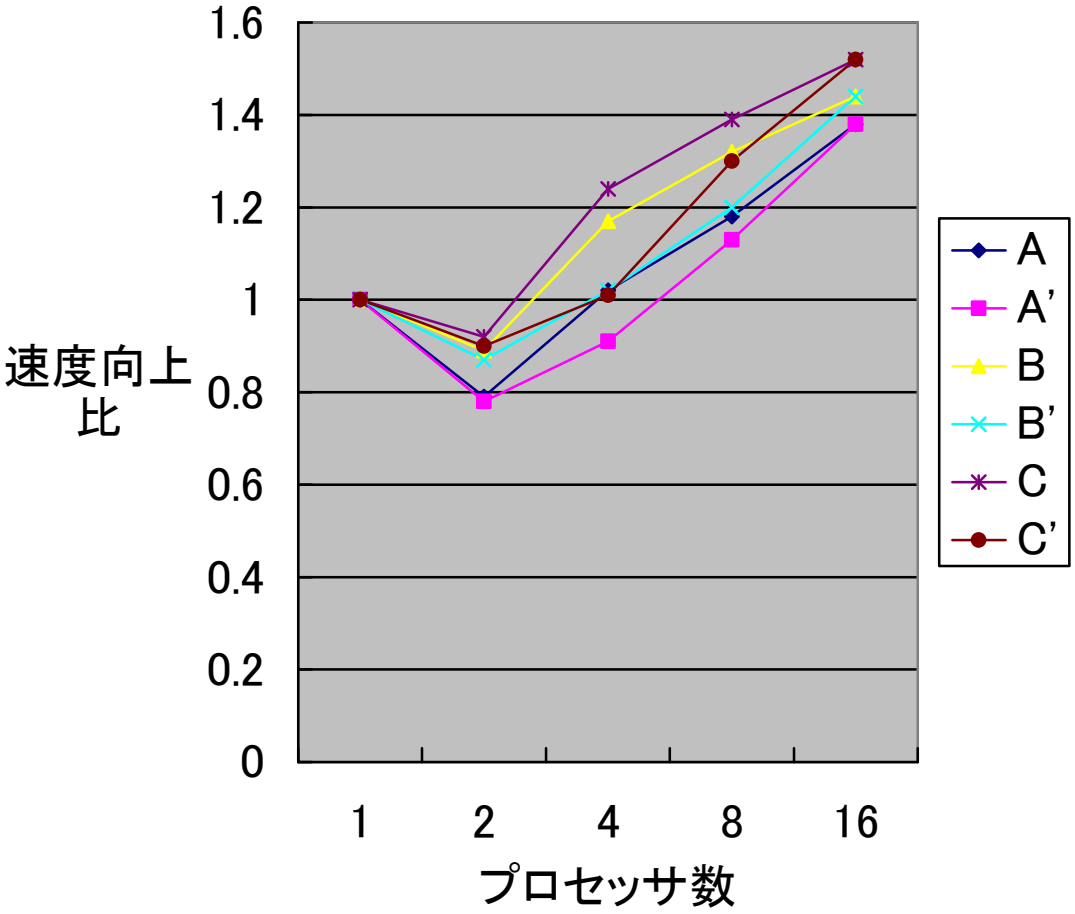

図 8：IS ベンチマーク速度向上比

5.4 考察

EP ベンチマークに関しては、ノード数とプロセッサ数の割り当てに関係なく、全ての場合において理想に近い速度向上が得られた。理由としては、EP ベンチマークが問題を解く際に転送をほとんど必要としないためだと考えられる。また、EP ベンチマークは並列計算に適した問題であるということが考えられる。

MG ベンチマークに関して、全てのクラスである程度の並列効果が得られた。さらに、ノードとプロセッサの割り当てを変化させるとさらに性能が向上した。ただし、NAS Parallel Benchmarks では本来は不要である通信が発生しているため、PC クラスタ上で実行した場合、理想どおりの並列効果は得られない。

CG ベンチマークに関して、全てのクラスにおいてある程度の並列効果が得られた。さらにノードとプロセッサの割り当てを変化させると、さらに性能が向上した。しかし、変化させた場合、プロセッサ数が 8→16 において、速度向上が得られなかった。これは、通信のオーバーヘッドの影響によるものだと考えられる。

IS ベンチマークに関して、全てのクラスにおいて、ごくわずかな並列効果しか得られなかった。また、プロセッサ数 2 に時には、性能が低下した。さらに、IS ベンチマークでは、ノードとプロセッサの割り当てを変化させると性能が低下した。理由としては、IS ベンチマークが通信を大量に行うベンチマークであるということが考えられる。

6. おわりに

本研究では、NAS Parallel Benchmarks による PC クラスタの性能評価を行ってきた。本研究で行ってきた EP、CG、MG、IS のそれぞれのベンチマークを行うことによって、PC クラスタの処理速度を計測し性能を評価することができた。16 台実行で EP ベンチマークでは、約 16 倍、MG ベンチマークでは 5 倍強、CG ベンチマークでは約 5 倍の並列効果を得た。また、IS ベンチマークは並列計算に不向きであることもわかった。

今後の課題としては、IS ベンチマークのハイブリッド並列化しての性能評価、また、Score クラスタ上での実験などが挙げられる。

謝辞

本研究の機会を与えてくださり、貴重な助言、ご指導いただきました山崎勝弘教授に深く感謝いたします。また、本研究に当たり、貴重なご意見をいただきました、中谷氏、Duy氏、およびいろいろな面で貴重なご意見や助言、励ましの言葉をいただきました本研究室の皆様には深く心から感謝いたします。

参考文献

[1] 吉田純一　他：PC クラスタ超入門、超並列計算研究会、2000、http://mikilab.doshisha.ac.jp/dia/smpp/lustecr2000/index.html
[2]湯浅太一、安村通晃、中田登志之：はじめての並列プログラミング、共立出版、1999
[3]PC Cluster Consortium：http://www.pccluster.org/
[4]Al Geist 他、PVM3 ユーザーズガイド&リファレンスマニュアル　日本語版、http://phase.hpcc.jp/phase/pvm-j/jpvmug.pdf
[5]Open MP：http://www.openmp.org/drupal/
[6]Omni OpenMP Compiler：http://phase.hpcc.jp/Omni/home.ja.html
[7]NAS Parallel Benchmarks：http://www.nas.nasa.gov/Resources/Software/npb.html
[8]加藤寛暁：SMP クラスタ上での OpenMP による MPEG2 エンコーダの並列化、立命館大学理工学部情報学科卒業論文、2006
[9]粂山直生：PC クラスタを用いた画像処理プログラムの並列化、立命館大学理工学部情報学科卒業論文、2005
[10]多田修司：PC クラスタ上での OpenMP によるマンデルブロ集合の並列化、立命館大学理工学部情報学科卒業論文、2005