COMPaS：Pentium Proを用いたSMPクラスタとその評価

田中良夫† 松田元彦†
久保田和人† 佐藤三久†

我々はPentium Proを4台搭載した対称型マルチプロセッサであるサーバ型PC 8台をMyrinetと100Base-T Ethernetで結合したSMPクラスタCOMPaSを構築した．階層構造を持つクラスタシステムにおいて高い性能を獲得するため，通常の分散プログラミングとマルチスレッドプログラミングを組み合わせたプログラミング（共有/分散融合プログラミング）を行ってCOMPaSの性能を測定した．COMPaSの性能は，データの局所性（アクセスパターン）に依存する．主記憶に対して高いバンド幅を必要とするプログラムにおいてはメモリバスの性能がネックとなるが，データの局所性が高く，キャッシュを有効に利用することができる場合には高い性能を得ることができる．

COMPaS: A Pentium Pro PC-based SMP Cluster and Its Experience

YOSHIO TANAKA,† MOTOHIKO MATSUDA,† KAZUTO KUBOTA† and MITSUHISA SATO†

We have built an eight node SMP cluster called COMPaS (Cluster Of Multi-Processor System), each node of which is a quad-processor Pentium Pro PC. In this paper we report on this hybrid shared memory/distributed memory programming on COMPaS and its preliminary evaluation. The performance of COMPaS is affected by data size and access patterns. COMPaS can provide high performance if we can get the high data locality and take the advantage of cache.

1. はじめに

近年，価格・性能比が優れているという利点により，Pentium Proプロセッサのシステムが広く普及しており，Pentium Proを用いたクラスタシステムに関する研究が幅広く行われている．また，複数のプロセッサを搭載した**対称型マルチプロセッサ（Symmetric Multi-Processor, SMP）**システムも急速に普及し，サーバシステムや高性能計算のプラットフォームとして用いられるようになった．我々は4プロセッサ（Pentium Pro, 200 MHz）のPCサーバ8台をMyrinet[1]および100Base-T Ethernetで結合したSMPクラスタ**COMPaS（Cluster Of Multi-Processor System）**を構築した[2)~5)]．また，Myrinet上でのユーザレベル通信レイヤとして**NICAM（Network Interface Communication layer using Active Message）**[6)]を設計・開発した．NICAMはリモートメモリ転送を基本とする低オーバヘッドかつ高いバンド幅の転送プリミティブおよび高速な同期プリミティブを提供する．

SMPクラスタにはコンパクトな大きさで多くのプロセッサを搭載でき，保守，管理も容易になるという利点がある．今後，より安価で高性能なSMPシステムの登場にともない，SMPクラスタはクラスタコンピューティングの重要なプラットフォームとなると思われる．SMPクラスタのアーキテクチャは分散メモリアーキテクチャと共有メモリアーキテクチャの中間的な形態であり，SMPクラスタで高い性能を得るためには両方のアーキテクチャの利点を引き出す必要がある．現在SMPクラスタはクラスタシステムの1つの有効な選択肢となっているが，プログラミングおよび性能特性に関して未知の部分が多い．これらを明らかにすることは，今後SMPクラスタを構築する際のガイドラインとして重要な指標を与えることになる．我々は各ノード間ではメッセージパッシングやリモートメモリ転送により通信し，各ノード内のプロセッサ間ではマルチスレッドを用いて共有メモリを介して通信する**共有/分散融合プログラミング**を行ってCOMPaSの性能を測定し，その性能特性を調査した．ノード内の

† 新情報処理開発機構
Real World Computing Partnership

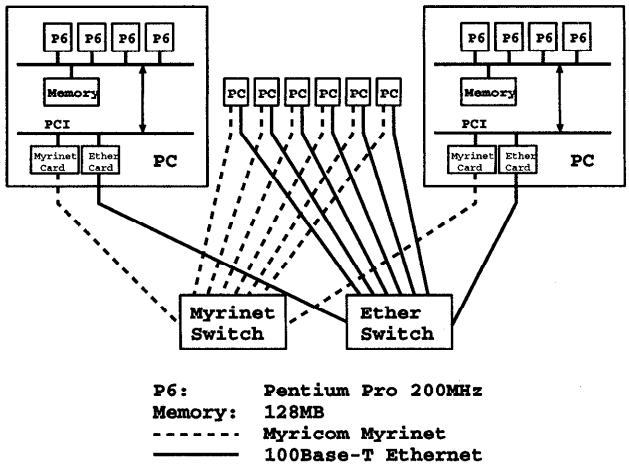


図1 COMPaSの仕様
Fig. 1 Configuration of COMPaS.

計算を複数のスレッドを用いて処理することにより，SMPの利点を引き出すことが期待される．

PCを用いたクラスタシステムに関してはいくつかの報告がなされているが，COMPaSのような均質なSMPクラスタ上でのプログラミングおよびその性能に関するものはない．バークレーのNOWプロジェクト[7]はワークステーションクラスタのシステムを構築している．NOWプロジェクトのクラスタはSunのUltraSPARCとSPARCStation，SMPを含めたIntelのPCなどをノードプロセッサとしている．しかし，COMPaSのような単一機種のクラスタではなく，共有/分散融合プログラミングのようなSMPクラスタ上でのプログラミングに関する報告もない．Jazznet[8]は数学あるいは科学計算のためのPCクラスタである．初期の仕様はシングルプロセッサPCとマルチプロセッサPCが混在したものであり，3台のシングルプロセッサPC，1台の2プロセッサPCおよび1台の4プロセッサPCがネットワークで結合されたものである．Jazznetは分散環境と共有環境の両方を提供しており，基本的な性能が報告されている．しかし，共有と分散を融合させたプログラミング環境および性能に関しては言及していない．

本稿では，COMPaS上での共有/分散融合プログラミングの方法とその評価，およびCOMPaSの性能特性に関して報告する．SMPクラスタシステムのケーススタディとして，SMPクラスタ構築のための有用なガイドラインを示す．次章では我々が作成したCOMPaSの仕様およびノードプロセッサの基本性能を示す．3章ではNICAMの仕組みおよびノード間通信の基本性能に関して説明する．4章では共有プログラミングと分散プログラミングを融合した共有/分散融合プログラミングの方法に関して説明する．5章ではCOMPaSの性能測定に関する実験結果を示し，6章ではCOMPaSの性能特性に関して考察し，最後にまとめを行う．

2. COMPaSの仕様とノードプロセッサの基本性能

本章ではCOMPaSの仕様およびメモリバスのバンド幅やスレッド間のバリア同期などのノードプロセッサの基本性能に関して述べる．**図1**にCOMPaSの仕様を示す．COMPaSは4台のPentium Proを搭載したPCサーバ東芝GS700を8台100Base-T EthernetとMyrinetにより接続したシステムである．ネットワークは2系統用意されているが，計算を行う際にそれらを混在させて用いることはない．Myrinetを介して計算を行う場合にはNICAMを用いてプログラミングを行う．Ethernetは主にファイルシステムの共有のために用いているが，Message Passing Interface（mpich-1.1.1 on TCP/IP）[9]も実装されており，Ethernetを介して計算を行うこともできる．Ethernet上で測定した性能はPentium Pro PCを用いた標準的な構成のSMPクラスタの性能と考えることができる．GS700は4-Way SMPのPentium Pro PC（200 MHz，450 GX chip-set，16 KB L1キャッシュ，512 KB L2キャッシュ，384 MB主記憶）である．各ノードのオペレーティングシステムはSolaris2.5.1である．

[メモリバスの性能]

表1に複数のスレッドを起動した際のメモリリード，

表1 メモリバスのバンド幅（MB/s）
Table 1 Memory bus bandwidth (MB/s).

スレッド数	read	write	copy	*memcpy*
1	360.0	91.1	66.5	85.6
2	258.2	106.6	70.6	88.9
3	251.7	106.6	70.1	87.7
4	250.3	99.7	70.7	88.3

表2 スレッド間のバリア同期
Table 2 Barrier synchronization time between threads.

スレッド数	2	3	4
バリア同期の時間（μsec）	1.222	1.761	1.960

```
procedure barrier(my_id, num_threads)
  begin
    if my_id = 0 then
      for i := 1 to num_threads do
        while barrier_slot[i] ≠ 1
        end
      end
      wakeup_flag := 1
    else
      barrier_slot[i] := 1;
      while wakeup_flag ≠ 1
      end
    end
  end;
```

図2 バリア同期のアルゴリズム
Fig. 2 Algorithm of barrier synchronization.

メモリライト，メモリコピー（copy），*memcpy*のバンド幅を示す．示されているバンド幅は全スレッドのバンド幅の合計である．メモリコピーはdoubleデータのコピーを繰り返し行うことにより，メモリのコピーを行う．*memcpy*はC言語のライブラリ関数であり，Pentiumが提供しているメモリコピーを行うアセンブリ言語の命令を用いて実装されている．バンド幅の合計値はスレッド数には関係なく一定であり，複数のスレッドを起動した場合にはスレッドあたりのバンド幅はスレッド数に反比例して低くなることが分かる．

[スレッド間のバリア同期]

表2にスレッド間でのバリア同期にかかる時間を示す．バリア同期のアルゴリズムを**図2**に示す．本バリアはスピンロックを用いており，OSによって提供されるmutex変数やcondition変数は用いていない．本バリアは4スレッドで2マイクロ秒以下と，高速なバリア同期を提供することができる．ちなみに，Solarisのmutex変数を用いたバリア同期の場合は4スレッドで約180マイクロ秒かかってしまう．

3. ノード間通信の基本性能

COMPaSはMyrinetと100Base-T Ethernetの2系統のネットワークを提供している．Myrinetを介した通信はNICAMを用いて行い，Ethernetを介した通信はmpichを用いて行う．本章ではNICAMの仕組みと性能，およびmpichを用いてEthernetを介した場合のノード間通信の性能に関して述べる．

3.1 NICAM: SMPクラスタ向けMyrinet上のユーザレベル通信レイヤ

SMPクラスタにおいては，各ノード内で複数のスレッドが動作する．ノード間通信をメッセージパッシングにより行うと，メッセージバッファに対する排他処理が必要となる．また，メッセージのコピーによりメモリバスに対する負荷が増加してしまう．リモートメモリ転送はこれらの問題を回避することができる．我々はリモートメモリ転送に基づくMyrinet上のユーザレベル通信レイヤであるNICAMを設計，実装した．NICAMは低オーバヘッドで高いバンド幅のデータ転送および同期機能を提供する．

3.1.1 Active Messageを用いた通信プリミティブ

ノード間通信においてはレイテンシが通信性能に大きな影響を与える．しかし，SMPクラスタにおいてはCPUのオーバヘッドはメモリバスを占有し，他のCPUに影響を与えてしまうため，より重要な要素であると考えられる．NICAMはNI上のActive Message[10]を用いたリモートメモリデータ転送および同期命令を提供し，以下のような特徴を持つ．

- リモートメモリ転送により，通信にかかわる排他処理を最小限にする．
- メモリバスへの負荷を抑えることができる．
- NI上のDMAエンジンを用いて転送することにより，データ転送にCPUが関与しない．

実装に際しては，CPU-NI間の呼び出しもAMを用いている．**図3**にNICAMの通信メカニズムを示す．

DMA転送の対象領域が物理メモリからページアウトされるのを防ぐために，ユーザはあらかじめその領域をピンダウンしておく必要がある．リモートメモリ転送においては，ユーザはローカルアドレスと同じ方法でリモートアドレスを指定する．NICAMは指定されたアドレスを自動的に物理アドレスに変換する．

3.2 NICAMの性能

図4にNICAMを用いた場合のpoint-to-pointのバンド幅および通信時間を示す．

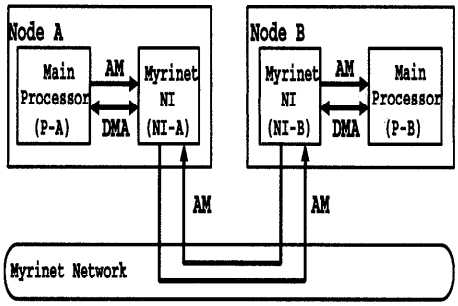


図3 NICAMの通信メカニズム
Fig. 3 Communication mechanism of NICAM.

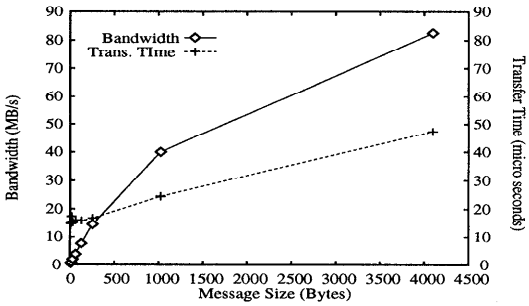


図4 NICAMによるリモートメモリ転送の性能
Fig. 4 Performance of NICAM.

表3 NICAMおよびPMのノード間の同期時間
Table 3 Synchronization time between nodes (NICAM vs. PM).

Num. of Nodes	2	4	8
NICAM (μsec)	23.79	31.45	38.67
PM (μsec)	13.47	30.37	47.42

短いメッセージを転送する際の最小レイテンシは約16マイクロ秒であり，最大のバンド幅は約82.5 MB/sである．**表3**にCOMPaS上でNICAMおよびPM[11]を用いた場合のバリア同期にかかる時間を示す．

PMは手塚らによって開発されたMyrinet上のユーザレベルの高速メッセージパッシングライブラリであり，クラスタシステムにおいて高い性能を示している．NICAMの結果は *nicam_barrier()* を用いたものである．PMはバリア同期のためのプリミティブを提供していないため，PMの結果はpoint-to-point通信をshuffle exchangeアルゴリズムを用いて組み合わせてバリア同期をとったものである．CPU-NI間の通信コストのため2ノードの場合の同期時間はNICAMの方が長くなっているが，NI間で処理が行われる各ステップに必要なコストは約7.5マイクロ秒とPMの約17マイクロ秒に比べて半分以下となり，8ノードの場合にはNICAMの方が高速に同期を行うことができる．

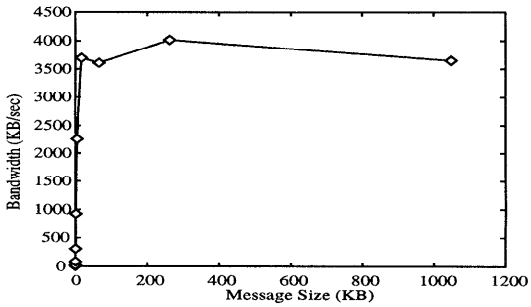


図5 point-to-pointの通信性能（mpich, 100Base-T Ethernet）
Fig. 5 Point-to-point bandwidth (mpich, 100Base-T Ethernet).

表4 ノード間のバリア同期
Table 4 Barrier synchronization time between nodes (mpich, 100Base-T Ethernet).

ノード数	2	4	8
バリア同期の時間（msec）	0.493	1.066	1.645

3.3 Ethernet上の通信性能

COMPaSはEthernet上の標準的な通信レイヤとしてmpichを使うことができる．**図5**に *MPI_Send()* および *MPI_Recv()* を用いた場合の *point-to-point* の通信性能を，**表4**に *MPI_Barrier()* を用いた場合のノード間のバリア同期にかかる時間を示す．バンド幅は最大で約4 MB/sである．

4. 共有/分散融合プログラミング

SMPクラスタ上でのプログラミング方法としては，以下の3種類の方法が考えられる．

[分散プログラミング]
すべてのプロセスはノード間/ノード内を問わず，すべてメッセージパッシングにより通信を行う．通常の分散メモリシステム上でのプログラミングと同じ方法であり，プログラミングが容易であり，様々な分散メモリシステムとの互換性もある．しかし，ノード内の通信もメッセージパッシングにより行われるため，SMPシステムの利点を活かすことができない．

[共有プログラミング]
DSMシステムが提供するグローバルアドレスを用いて，通常の共有メモリシステム上でのプログラミングと同じ方法でプログラミングを行う．プログラミングが容易であるという利点はあるが，DSMシステムによって陰に通信が発生し，高い性能を獲得するためには注意深いプログラミングが必要となってしまう．また，DSMシステムのオーバーヘッドによって性能に大きな影響を受けてしまう可能性が高い．

[共有/分散融合プログラミング]

ノード内の計算はマルチスレッドを用いたプログラミングにより行い，ノード間の通信はメッセージパッシングやリモートメモリ転送に基づいて行う．プログラミングは若干複雑になるが，SMPノードの共有メモリの利点を活かすことができれば高い性能を得ることができる．

我々はSMPクラスタの特徴を活かすという点と，SMPクラスタの性能特性を調査するという目的のため，共有/分散融合プログラミングを行った．本稿では対象とするプログラムをデータ並列プログラムとした．その理由は，データ並列プログラムは行列やベクトルなどのデータの段階的な分割が容易であるからである．まず始めにデータを各ノードに分割し，続いて各ノード内で与えられたデータをさらに分割して起動したスレッドに割り当てればよい．このようにすれば，COMPaS上で共有/分散融合プログラミングによりデータ並列プログラムを容易に実装することができる．共有/分散融合プログラミングはSPMDプログラミングスタイルに基づいており，複数のノードで実行されるプログラムがマルチスレッドプログラムであると考えればよい．以下に共有/分散融合プログラミングのおおまかな手順を示す．

step 1 データをノード数に応じて分割，分散する．

step 2 各ノードでは複数のスレッドを起動し，それらにデータを割り当てる．

step 3 各スレッドは割り当てられた領域を参照し，局所計算を行う．

step 4 全ノードでstep 3が終了したら，ノード間でデータの交換を行う．

step 5 必要な回数だけstep 3とstep 4を繰り返し行う．

step 1におけるデータの分散方法と，step 4におけるノード間のデータ交換方法は分散プログラミングのアルゴリズムに従う．step 2におけるスレッドに対するデータの割当てとstep 3における局所計算は共有プログラミングのアルゴリズムにより実行される．つまり，step 3においてノード内の計算が複数のスレッドによって処理されていることを意識しなければ単純な分散プログラミングとなり，逆にstep 3においては与えられたデータに対する計算をマルチスレッドによって行うという共有プログラミングとなっている．全スレッドで同期をとる場合はスレッド間の同期とノード間の同期の両方をとる必要があることや，*reduction*などのいくつかの命令に関しては特殊な処理が必要となることなど，いくつかの点においては共有/分散融合プログラミング特有の処理が必要となるが，基本的には分散プログラミングの局所計算部分を共有プログラミングによって処理するという形で容易に実装することができる．

5. 実験結果

我々は以下の4種類のベンチマークを共有/分散プログラミングにより実装し，COMPaSの性能を測定した．

ラプラス方程式の陽解法（2次元）

反復法に基づく方程式の解法であり，我々の実装はヤコビ法を用いている．分散プログラミングの場合，ノード間通信は隣接間転送のみとなる．実験では行列のサイズは640×640とした．

行列の乗算

並列計算機における行列の乗算には行分割，列分割およびブロック分割などのデータの分割方法と，ijk，jkiなどの演算順序の組合せにより様々なアルゴリズムが存在する．今回の実装ではデータの局所性を最大限に引き出すようにブロッキングとタイリングを組み合わせた方法を採用した．実験では行列のサイズは1800×1800とした．

radixソート

radixソートは各データをn進数で表し，各桁ごとのソートを繰り返すことにより全体のソートを行う方法である．実験では4M個の整数のソーティングを行った．

疎行列CGカーネル

NAS Parallel Benchmarksに基づいて実装を行った．実験でのサイズはClass Aである．

実験に際しては1，2，4，8の4通りのノード数に関し，各ノード数ごとにスレッド数を1，2，4の3通りに変化させて実行時間を計測した．本実装では起動したスレッドはSolarisが提供している*processor_bind()*関数を用いて別々のプロセッサに割り当てられるようになっている．ノード数とスレッド数の積が動作するプロセッサ数（スレッド数）となる．1ノードの場合の実験結果は共有メモリシステム上でのマルチスレッドプログラミングの結果と同じである．1スレッドの場合の実験結果は分散プログラミングの結果を表している．

図6と**図7**にラプラス方程式の陽解法の実行時間を示す．図6はMyrinet上でNICAMを用いた場合の実行時間を，図7はEthernet上でmpichを用いた場合の実行時間を示す．

図6および図7はそれぞれ1ノード，2ノード，4

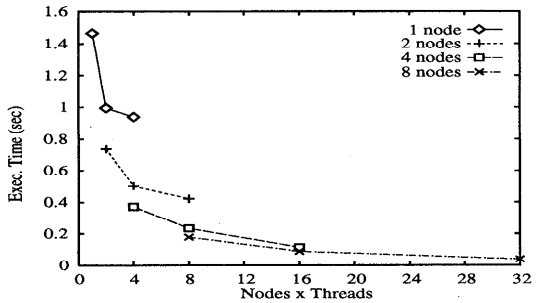


図6 ラプラス方程式の陽解法の実行時間（$N = 640$, NICAM/Myrinet）

Fig. 6 Execution time of Laplace solver ($N = 640$, NICAM/Myrinet).

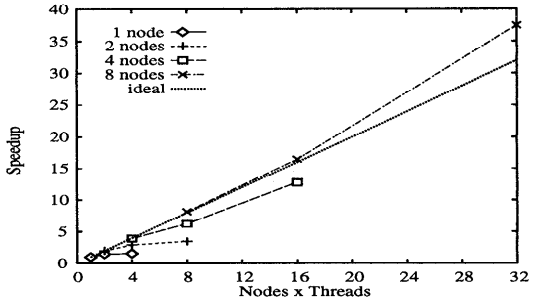


図8 ラプラス方程式の陽解法の速度向上率（$N = 640$, NICAM/Myrinet）

Fig. 8 Speedup of Laplace solver ($N = 640$, NICAM/Myrinet).

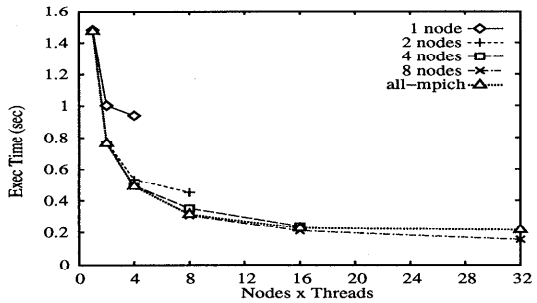


図7 ラプラス方程式の陽解法の実行時間（$N = 640$, mpich/Ethernet）

Fig. 7 Execution time of Laplace solver ($N = 640$, mpich/Ethernet).

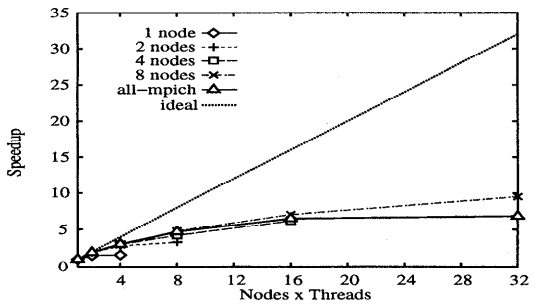


図9 ラプラス方程式の陽解法の速度向上率（$N = 640$, mpich/Ethernet）

Fig. 9 Speedup of Laplace solver ($N = 640$, mpich/Ethernet).

ノード，8ノードの場合の結果を示し，X座標はノード数とスレッド数の積，つまり全プロセッサ数を表す．1ノードの場合にスレッド数を増やしても効率は上がらない．プロセッサ間での排他制御は必要なく，同期処理も無視できる程度のものであり，これはデータサイズが大きく主記憶に対するアクセスが多数発生し，メモリバスの性能がボトルネックになっているためと考えられる．ラプラス方程式の陽解法ではノード間通信は隣接間転送のみであるが，ノード数が増えると計算量が減るのに対してノード間通信は変わらないため，ノード間通信の性能差が顕著に表れる．たとえば，8ノードの場合，Myrinet上のNICAMを用いた性能はEthernet上でmpichを用いた場合に比べて約2〜3倍程度になっていることが分かる．図7中の`all-mpich`で示されている値は，共有/分散融合プログラミングではなく，通常の分散プログラミングを行った場合の結果である．MPIプロセスが8以下の場合はMPIプロセスの数だけノードが使用され，各ノードあたり1つのMPIプロセスが起動される．MPIプロセスが16および32の場合は，8ノードが使用され，各ノードあたり2または4のMPIプロセスが起動される．したがって，16プロセス以下の場合は共有/分散融合プログラミングとの差はあまりない．しかし，32プロセッサになるとノード内のプロセスもメッセージパッシングにより通信を行うオーバヘッドにより共有/分散融合プログラミングに比べて性能が劣ってくる．実際，CGカーネルなどの全対全通信が起こるようなベンチマークを通常の分散プログラミングにより実装すると，32プロセッサで1プロセッサよりも実行時間がかかってしまうことが確認された．

図8および**図9**にMyrinet上でNICAMを用いた場合およびEthernet上でmpichを用いた場合の速度向上率を示す．idealは理想的な速度向上率を示したものである．

Myrinet上でNICAMを用いると，8ノードの場合に理想的な速度向上率以上の結果が得られる．これは，ノード間通信の高速化に加えデータサイズが小さくなり，データがキャッシュに収まって主記憶に対するアクセスが減ったためである．Ethernet上でmpichを用いた場合，ノード内の処理は同様に高速に行われる

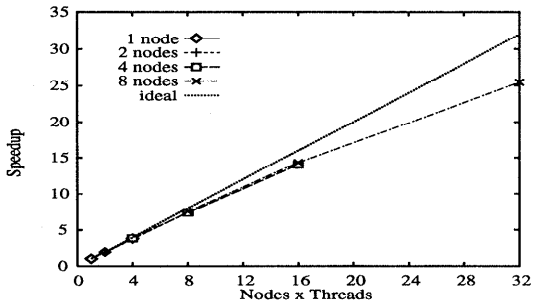


図10 行列の乗算の速度向上率（$N = 1800$, NICAM/Myrinet）
Fig. 10 Speedup of matrix multiplication ($N = 1800$, NICAM/Myrinet).

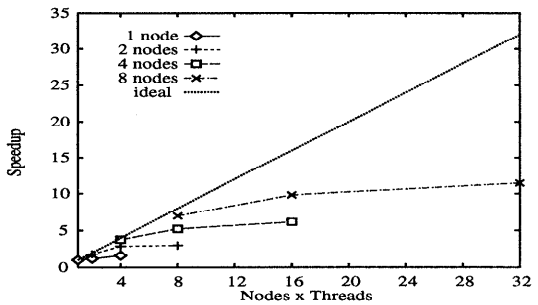


図11 radixソートの速度向上率（4M int, NICAM/Myrinet）
Fig. 11 Speedup of radix sort (4M int, NICAM/Myrinet).

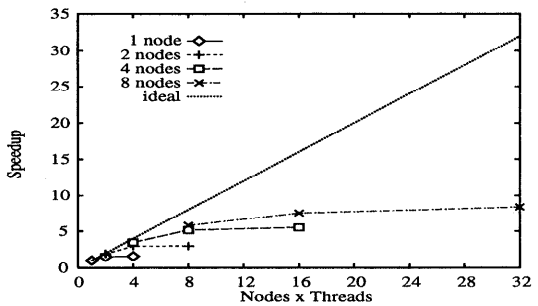


図12 CGカーネルの速度向上率（Class A, NICAM/Myrinet）
Fig. 12 Speedup of CG Kernel (Class A, NICAM/Myrinet).

がノード間通信が遅いため，高い効率を得ることができない．

図10から**図12**に各ベンチマークプログラムをMyrinet上でNICAMを用いて実行した場合の速度向上率を示す．行列の乗算ではかなり高い速度向上率が得られている．行列の乗算はラプラス方程式の陽解法に比べるとはるかに大きなデータを扱っているが，アルゴリズムを工夫してデータの局所性を高く引き出しているため，1ノードの場合に高い並列化効率を得ることができる．また，ノードを固定してスレッドを増加させる場合とスレッドは増加させずにノードを増やす場合とでほぼ同じ結果を示している．これはもともと重くないノード間通信の処理がNICAMを使うことによってほとんどオーバヘッドとして見えなくなっているためである．8ノード・4スレッドの場合でも約80%程度の並列化効率を得ることができる．これは，スレッド内でキャッシュを有効に利用する実装による高性能計算と，NICAMによる高性能ノード間通信により可能となる．radixソートでは1スレッドの場合にはノード数に対して高い並列化効率が得られている．radixソートではノード間で交換するデータ量が大きくなる（2ノードの場合は平均4MB，8ノードの場合は平均256KB）ため，ノード間通信の性能が全体の性能に大きな影響を与えるが，NICAMを用いた高速通信により高い性能が得られていると考えられる．しかし，スレッド数に対してはスケーラブルでない．これは，データサイズが大きいため，メモリバスの性能がボトルネックとなっているためと考えられる．CGカーネルでも同様にスレッド数を増やしても効率は上がらない．4スレッドを用いた場合の実行時間は2スレッドのものとほとんど変わらない．CGカーネルのデータサイズは大きく，かつデータに対するアクセスが1度走査するだけと局所性が低いため，主記憶に対する高いバンド幅が必要とされるため，メモリバスの性能がボトルネックとなってしまうためである．

6. 考　　察

6.1 データの局所性の効果・影響

ラプラス方程式の陽解法の場合，図8に示したようにノード数が少ない場合にはスレッド数を4にしても速度向上率はかなり低いが，ノード数が8になると速度向上率は理想値を超える．これは，プロセッサ数が増加した結果各プロセッサあたりのデータサイズがちょうどキャッシュにフィットする程度となり，主記憶に対するアクセスが減ってキャッシュを有効に利用することができ，SMPの利点を引き出すことができるためである．

このようにデータサイズが小さくなれば高い性能が得られるが，たとえデータサイズが大きくてもアルゴリズムを工夫することにより，データの局所性を引き出してSMPの利点を活かすことができる．キャッシュ上のデータに対してアクセスする回数を増やす（主記憶へのアクセスを減らす）ことにより，データの局所性を高めることができる．行列の乗算においては，

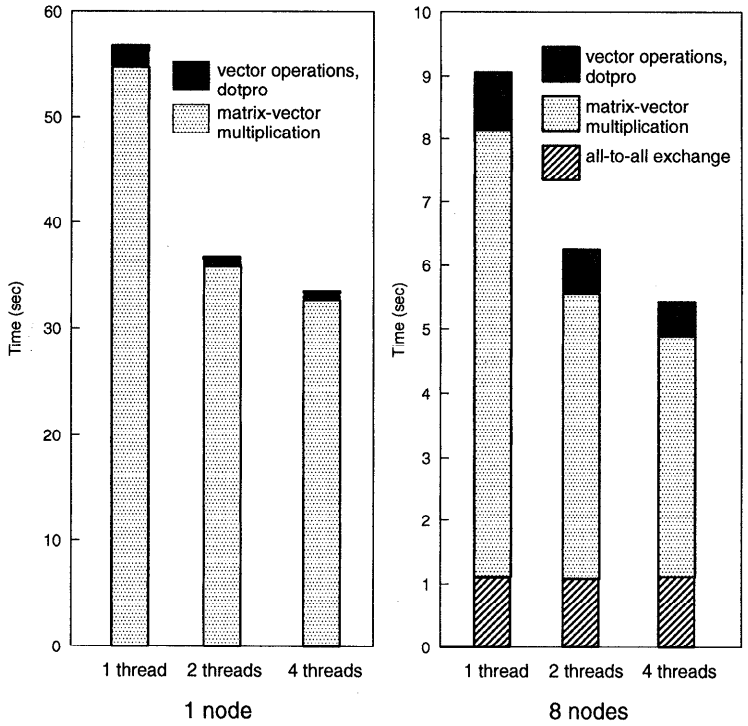


図13 CGカーネルの各処理にかかる時間
Fig. 13 Break down results of CG Kernel.

データサイズは大きい（1800×1800の場合には行列1つあたりで約25 MB）が，行列のブロッキングおよびタイリングによるレジスタブロッキングを行うことにより高いデータの局所性が得られ，結果としてスレッド数に対して高い効率を得ている．共有/分散融合プログラミングにおいて高い効率を得るためには，ノード内でのマルチスレッドプログラミングにおいて高い効率を得ることが必要である．

Pentium Proを用いたSMPシステムのメモリバスの性能はまだ十分ではなく，このようなメモリバスの性能が弱いSMPシステムを用いたクラスタ上で高い性能を得るためには，ノード内での計算アルゴリズムを工夫して高い局所性を得ることが重要である．キャッシュを有効に利用することができれば，ノード間通信の発生しないSMPクラスタが単一階層のクラスタに比べて有利になると考えられる．

6.2 メモリバスボトルネックによる影響

CGカーネルのように主記憶に対する高いバンド幅が必要なプログラムでは高い効率が得られなかった．その原因を明らかにするため，CGカーネルをブレークダウンして，行列とベクタの乗算，ベクタに対する加算，内積などの演算，およびノード間での全対全通信の処理にかかる時間を求めた．**図13**に1ノードおよび8ノードにおける結果を示す．

1ノードの場合は処理のほとんどが行列とベクタの乗算に費やされており，この部分がスケーラブルでないことが分かる．8ノードの場合，今回の実装ではノード間通信は親スレッドのみが行うめに，その時間はスレッド数によらず一定となっている．また，1ノードの場合と同様に行列とベクタの乗算がスケーラブルでないことが分かる．表1に示されているように，ある時点での全スレッドのメモリバスのバンド幅の合計はメモリバスの性能（バンド幅）で抑えられるため，主記憶に対する高いバンド幅が必要な場合にはメモリバスの性能がボトルネックとなってしまう．この問題を回避するためにはキャッシュを有効に利用する必要があるが，Pentium ProのL2キャッシュは物理アドレスキャッシュであるため，実際の大きさ（COMPaSでは512 KB）の半分程度の大きさしか有効でない点に注意する必要がある．

6.3 ノード間通信の割合

表5に行列の乗算における1ノードおよび8ノードの場合の並列化効率を示す．Nはノード数，Tはスレッド数を表す．1ノードの場合にはスレッド数に対して高いスケーラビリティがあるが，8ノードの場合にはスレッド数に対するスケーラビリティが低下す

表5 行列の乗算の並列化効率（$N = 1800$, NICAM/Myrinet）
Table 5 Efficiency of matrix multiplication ($N = 1800$, NICAM/Myrinet).

	$T=1$	$T=2$	$T=4$
$N=1$	1.00	0.99	0.97
$N=8$	0.94	0.89	0.80

る．8ノードの場合の実行時間をブレークダウンした結果，1，2，4スレッドの場合の乗算にかかる時間はそれぞれ17.09秒，8.59秒，4.46秒であった．このように，8ノードの場合でも乗算演算に関してはスレッド数に対して高いスケーラビリティが得られている．一方，ノード間通信の時間はスレッド数に依存せずに一定であり，その値は0.83秒であった．しかし，全実行時間に対する通信時間の割合を見てみると，1スレッドの場合は約4.6%であるのに対し，4スレッドの場合は15.7%となる．

行列の乗算は演算量に比べると通信量が少ないベンチマークではあるが，スレッド数が増えると各スレッドが担当する領域が小さくなって演算時間が短縮され，ノード間通信にかかる時間がオーバヘッドとして顕在化するために，プロセッサ数が多くなると効率が低下する．

7. 結　論

本稿ではPentium Pro 4台を搭載したPCサーバ8台をMyrinetおよび100Base-T Ethernetで結合したSMPクラスタCOMPaSについて基本性能を示し，COMPaS上で共有/分散融合プログラミングにより実装したベンチマークの実行結果を示した．SMPクラスタ向けの通信レイヤであるNICAMはMyrinetのNI上のActive Messageを用いたリモートメモリベースの通信レイヤであり，低オーバヘッドかつ高いバンド幅を持つ通信を可能とする．

ブロック化による行列の乗算のようにキャッシュを効果的に利用することができれば，マルチスレッド処理により高い効率を得ることができる．残念ながら現在のPentium Proのメモリバスの性能ではradixソートやCGカーネルのようにデータサイズが大きく基本的にデータアクセスが必要なアプリケーションに対しては，メモリバスの性能がボトルネックとなって高い性能は得られなかった．

SMPクラスタを構築する際のガイドラインとして以下のようなことを示すことができる：

- メモリバスの性能が大きな影響を与える．Pentium Proのようにメモリバスの性能が低い場合には，主記憶に対して高いバンド幅を必要とするアプリケーションにおいては性能が劣化する可能性がある．
- 450GX chip-setのようなメモリバスの性能が低いSMPシステムにおいてSMPの利点を活かして高い性能を得るためには，データの局所性を引き出すことが重要である．データの局所性を引き出すということは，キャッシュ上のデータに対するアクセスの割合を増やすということである．データサイズが小さければデータの局所性は高くなるが，キャッシュにロードしたデータを何度もアクセスするようにアルゴリズムを工夫することにより，高いデータの局所性を引き出すことができる．
- SMPノード内で高い性能を得ることができても，100Base-T Ethernet程度のネットワーク性能ではノード間通信がネックとなって高い性能を得ることができない．Myrinetのようなプロセッサの処理能力に見合うだけの高性能なネットワークが望まれる．その場合でも，SMPクラスタにおいてはCPUオーバヘッドの少ない，マルチスレッドセーフなノード間通信の手段が必要である．また，メモリバスに対するアクセスも極力避ける必要がある．我々はMyrinetのNI上でのActive MessageおよびDMAの機能を用いたリモートメモリ転送を行うNICAMにより，この問題を解決した．

ノードプロセッサの処理能力が高くなるほど，ノード間通信のコストがオーバーヘッドとして見えてくる．リモートメモリ転送機能と複数のスレッドを用いる1つの方法として，ノード間通信と演算とをオーバラップすることによりノード間通信のコストを隠蔽することにより，さらに高い性能を得ることができる[12]．

本研究はSMPクラスタシステムのケーススタディとしてPentium ProのSMPシステムを用いてクラスタを構築し，その性能特性を調査した．現段階ではメモリバス性能の貧弱さのため，単一プロセッサのクラスタシステムに性能という点では劣ることが多いが，今後さらなる高性能SMPシステムの登場も期待され，今回得られた知見はSMPクラスタ構築の際の有効なガイドラインとなる．

謝辞 本研究にあたり，クラスタの構築に関してご指導いただいた新情報処理開発機構並列分散システムソフトウェア研究室の皆様に感謝いたします．

参考文献

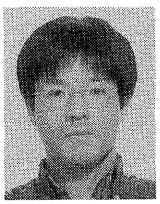
1) Boden N.J., Cohen, D., Felderman R.E., Kulawik, A.E., Seitz, C.L., Seizovic, J.N. and

Wen-King, S.: Myrinet – A Gigabit-per-Second Local-Area Network, *IEEE MICRO*, Vol.15, No.1, pp.29–36 (1996).
2) 田中良夫, 松田元彦, 安藤 誠, 久保田和人, 佐藤三久：SMP クラスタにおける共有/分散融合プログラミング, ハイパフォーマンスコンピューティング研究会資料, 97-HPC-67, No.67, pp.61–66, 情報処理学会 (1997).
3) 田中良夫, 松田元彦, 佐藤三久：SMP クラスタ COMPaS の性能評価, ハイパフォーマンスコンピューティング研究会資料, 98-HPC-70, No.70, pp.49–54, 情報処理学会 (1998).
4) Tanaka, Y., Matsuda M., Ando, M., Kubota K. and Sato, M.: COMPaS: A Pentium Pro PC-based SMP Cluster and Its Experience, *IPPS'98 Workshop on Personal Computer Based Networks of Workstations*, Chiola, G. and Conte, G. (Eds), Lecture Notes in Computer Science, Vol.1388, pp.486–497, Springer-Verlag (1998).
5) 田中良夫, 松田元彦, 安藤 誠, 久保田和人, 佐藤三久：COMPaS: Pentium Pro を用いた SMP クラスタとその評価, 並列処理シンポジウム JSPP'98, pp.343–350, 情報処理学会 (1998).
6) 松田元彦, 手塚宏史, 田中良夫, 久保田和人, 安藤 誠, 佐藤三久：SMP クラスタ向けネットワーク・インタフェース上 AM 通信, 計算機アーキテクチャ研究会資料, 97-ARC-125, No.125, pp.55–60, 情報処理学会 (1997).
7) Anderson, T., Culler, D., Patterson, D. and the NOW Team: A Case for Networks of Workstations: NOW, *IEEE MICRO*, Vol.15, No.1, pp.54–64 (1995).
8) Pozo R., et al.: The Jazznet Project. http://math.nist.gov/jazznet/index.html.
9) Gropp, W. and Lusk, E.: MPICH Working Note: Creating a new MPICH device using the Channel interface, Technical Report, ALgonne National Laboratory (1995).
10) Eicken, T., Culler, D.E., Goldstein, S.C. and Schauser, K.E.: Active Messages: A Mechanism for Integrated Communication and Computation, *Proc. 19th Int'l Symp. on Computer Architecture*, pp.256–266 (1992).
11) Tezuka, H., Hori, A., Ishikawa, Y. and Sato, M.: PM: An Operating System Coordinated High Performance Communication Library, *High-Performance Computing and Networking*, Lecture Notes in Computer Science, Vol.1225, pp.708–717, Springer-Verlag (1997).
12) Tanaka, Y., Matsuda, M., Kubota, K. and Sato, M.: Performance Improvement by Overlapping Communication and Computation on SMP Clusters, *Int'l Conf. on PDPTA'98*, Arabnia, H.R. (Ed.), Vol.1, pp.275–282 (1998)

(平成 10 年 9 月 1 日受付)
(平成 11 年 1 月 8 日採録)

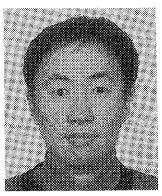
田中 良夫（正会員）

昭和 40 年生. 昭和 62 年慶応義塾大学理工学部数理科学科卒業. 平成 7 年同大学大学院理工学研究科数理科学専攻博士課程単位取得退学. 平成 8 年より技術研究組合新情報処理開発機構に勤務. 現在同組合並列分散システムパフォーマンスつくば研究室主任研究員. 工学博士. 並列システムの性能評価およびクラスタシステムでの高性能計算に関する研究に従事. ACM 会員.



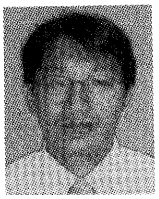
松田 元彦（正会員）

昭和 38 年生. 昭和 63 年年京都大学理学部卒業. 同年住友金属工業（株）入社. 平成 7 年より技術研究組合新情報処理開発機構に出向. データ並列計算等の研究に従事.



久保田和人（正会員）

昭和 39 年生. 昭和 63 年早稲田大学理工学部電子通信学科卒業. 平成 5 年同大学大学院理工学研究科博士課程修了. 同年（株）東芝入社. 平成 7 年 10 月より平成 10 年 9 月まで技術研究組合新情報処理開発機構に出向. 工学博士. 並列計算機のプログラミング環境, 性能評価環境, 計算機クラスタシステムに興味を持つ.



佐藤 三久（正会員）

昭和 34 年生. 昭和 57 年東京大学理学部情報科学科卒業. 昭和 61 年同大学大学院理学系研究科博士課程中退. 同年新技術事業団後藤磁束量子情報プロジェクトに参加. 平成 3 年, 通産省電子技術総合研究所入所. 平成 8 年より, 技術研究組合新情報処理開発機構つくば研究センターに出向. 現在, 同機構並列分散システムパフォーマンス研究室室長. 理学博士. 並列処理アーキテクチャ, 言語およびコンパイラ, 計算機性能評価技術等の研究に従事. 日本応用数理学会会員.