JAMSTECが保有する並列計算機におけるMPI実行性能

齋藤　秀亮*1　古田　和史*1　直井　　純*1

MPI（Message Passing Interface）は並列処理を目的とした通信ライブラリ群のインターフェース規約であるが，ベンダーはそのプラットフォームに即した実装を行っている場合が多い。

本研究では，JAMSTECに導入されているNEC SX-4/20とIBM RS6000/SP, 自作のPCクラスタ並列計算機を用いて，各プラットフォームでのMPI通信ライブラリ特性，姫野ベンチマークによる計算性能，各種テストプログラムによる並列拡張性，並列化効率を求めた。この知見を基にプログラムの規模や構造を見極め，計算量とデータ通信量を評価することで，十分効果的な並列化を行うための有益な指針が得られた。その内容は（1）どのプラットフォームでも1回のコールでデータ転送を行うMPI_SENDRECVが他の1対1転送ルーチンを上回る転送速度を得た（2）SX-4はベクトル化可能で大容量のデータ転送があるプログラムに対して極めて有効である（3）SPは小規模から中規模のプログラムの実行で安定した性能を発揮する（4）PCCは現構成では転送データが相対的に少ない粗粒度の処理に向いている，などである。

上記の指針を基に海洋大循環モデルプログラムにMPIで並列化を施し，実際的な計算を行うことを予定している。

キーワード：MPI, 並列化，並列拡張性，並列化効率

Performance of MPI on parallel computers in JAMSTEC

Hideaki SAITO*2　Kazushi FURUTA*2　Jun NAOI*2

MPI（Message Passing Interface）is the interface agreement of communication library which aimed at carrying out parallel processing. Vendors implement MPI adapted to the parallel computers with which they deal.

This research shows MPI library's characteristics, performance for Himeno Benchmark Test, scalability and efficiency in parallel computing for test programs with NEC SX-4/20 , IBM RS6000/SP in JAMSTEC and PC cluster parallel computer of our own assembling. Some instructive information for efficient parallelization is obtained by evaluating scale and structure of a serial processing program and its estimating amount of computation and communication based on this store of knowledge.(1) On those platforms, MPI_SENDRECV that can transfer data at one call indicates the highest bandwidth than any other one-to-one routines of MPI. (2) SX-4 is especially effective for executing the programs that need to transfer big data and can be vectorized. (3) SP shows a good performance in executing the programs that range from small-scale to middle-scale. (4) PCC at the present is suitable for processes that need relatively small data-transferring. (i.e. the coarse-grained process)

Along the above information, we are planning to parallelize an Ocean Global Circulation Model program with MPI and to carry out the realistic computation.

Key Words : MPI, Parallelization, Scalability, Parallel efficiency

＊1　海洋科学技術センター情報管理室

＊2　Computer and Information Department, Japan Marine Science and Technology Center

1. はじめに

1.1. 概　要

MPI（Message Passing Interface）は標準化したメッセージ交換用ライブラリのインターフェース規約であるが，良好な性能を引き出すため，各ベンダーはそのプラットフォームに即した実装を行っている場合が多い。従ってMPIを用いたプログラムの実行時の総合的な性能はアルゴリズムなどのソフトウェア的側面のみならずプラットフォーム特性にも依存し，それに伴った特徴が現れると言われている。

MPI以前にはメーカーが独自にPVMなどを改良したメッセージ交換用ライブラリを提供していたが，互換性がなく，異機種間でプログラムの移植を妨げる要因の一つともなっていた。その弊害を考慮して計算機メーカー，研究者などで構成されたMPI Forumが1993年2月にメッセージ通信の仕様標準を定めた。MPIは本来非共有メモリ型計算機（MPP, Workstation Cluster etc）で並列処理を実行することを主眼とし実用性，移植性，効率性，柔軟性を意図されていたが，現在では非共有メモリ型計算機のみならず，共有メモリ型計算機への実装も進んできている。また，並列プログラム開発にMPIが広く使用されてきていることと併せて，様々なプラットフォームへのプログラム移植が容易となっている。

現在，海洋科学技術センター（以下センター）が保有する共用計算機のうちNEC SX-4/20（以下SX-4），IBM RS/6000 SP（以下SP）には，それぞれの製造会社からMPIプロダクトが導入されている。また，MPIは標準化を目指したライブラリであるため，インターネット上などでフリーに公開・配布されているものもある。市販パーソナルコンピュータ（PC）をスイッチングハブで並列につないだPCクラスタ並列計算機（以下PCC）にはそのフリーに提供されているMPICHを導入している。

本調査では，数種のMPIプログラムをSX-4，SP, PCCの異なるタイプの計算機で実行し，その結果を比較，検討して各プラットホームにおけるMPIの特性を把握する。それに基づきMPIを用いた並列化手法の指針，適切なプラットフォームの選択基準を明らかにすることを目的とする。

2. では各プラットフォームのMPI通信特性を検証する。3. では姫野ベンチマークテストにより，プロセッサ単体，及び各プラットフォームにおける処理能力の特性を検証する。4. では各種シミュレーションコードでしばしば用いられるアルゴリズムを使用してのMPI実行処理性能を測定し，結果を考察する。5. においてこれらの結果を総括し，MPI特性と並列化の指針を示す。

1.2. プラットフォーム

並列処理を実行する計算機はメモリアーキテクチャの観点で共有メモリ型と非共有メモリ型に分けられる。プログラムを実行する際，前者では全てのプロセッサエレメント（以下PE）が全メモリ空間にアクセスできるが，後者ではPEが直接アクセスできるのは通常，それ自身のローカルメモリに限定される。従ってほかのPEに属するローカルメモリを参照するには何らかの仕組みを必要とし，その方法の一つとしてPE間ネットワークを経由して通信を行い，メモリ内容を受け渡すメッセージ交換方式がある。この機能を具体化したルーチン群がライブラリとして提供されている。

1.2.1. SX-4/20

センターに導入されているSX-4はシングルノードタイプのいわゆる共有メモリ型ベクトル・パラレル計算機に分類される。ベクトル処理機構を備える全てのCPUがメインメモリユニット（以下MMU）を共有するシステムであり，さらにこれに対応したFortran77/SXやFortran90/SXなどのコンパイラを用いることで，並列タスク間でのメッセージ交換を記述しない並列化プログラミングが可能になっていて並列化作業への障壁が比較的低いという利点がある。MPIに関してはそのメモリアーキテクチャをいかした実装が行われている。

センターのSX-4の構成では，CPUはMMUに対し，32ウェイ・インターリーブ（各0.5GB/sec）でアクセスを行う。すなわちCPU-MMU間は1CPUあたり16GB/secのデータ転送幅を持つことになる。これは共有メモリ型計算機の一般的なデメリットであるCPU（同一，複数問わず）からメモリアクセスが頻繁に生じるとシステムバスを急激に消費し全体のパフォーマンスのボトルネックとなることを回避するための処置である。

1.2.2. RS/6000 SP

ノードはCPU, メモリ，ハードディスク，ネットワークインターフェースを備えたコンピュータ単体として動作し，ノード間ネットワーク接続（粗結合）によりいわゆるマルチコンピュータを構成している。従ってメモリアーキテクチャの観点から言えば，後述のPCクラスタ並列計算機と共に非共有メモリ型並列計算機に分類される。ノード間接続ネットワークとしては外部接続にも利用されるEthernetとそのプロトコルipに加え，ノード間通信専用の高速な媒体SPスイッチを備え，対応するUS通信サブシステムが用意されている。この理論性能は150MB/sec（ノード間）である。

ソフトウェア面では，マルチユーザの並列処理環境を運用するためのリソースは資源管理プログラムで取り扱われ，ジョブスケジューリングや負荷分散を行うロードレベラーや各種ツール類も充実している。

並列プログラム開発環境としてMPIに加えてIBM拡張MPE, IBM独自のメッセージ交換ライブラリMPL, 各種並列化コンパイラ，並列化した科学技術計算用ライブラリ等がプログラマに提供されている。

1.2.3. PCクラスタ並列計算機

ネットワークに複数のコンピュータが接続され（粗結合接続），それを集合体として使用する状態がブドウなどの“房”に似ていることからクラスタ（cluster）と呼ばれてい

る。SP同様マルチコンピュータであるが，SPと比較するとクラスタリングソフトが欠如していることや，専用ネットワークを備えていないことから各ノードの独立性が高い。ネットワーク接続されたノード間でデータ交換を行うことにより並列，分散処理が可能となっている。各ノードには市販されているPCを用いて構成することができることから安価で構築することができ，現在のコンピュータ技術の驚異的な発達に伴い，注目が集まっている。

本調査で使用するプラットフォームのスペック概要を表1に示す。

2. MPI通信サブルーチンの実行性能

2.1. 計算方法

メッセージ交換の基本ともいうべき一対一通信は，ブロッキング通信・ノンブロッキング通信に大別される。ブロッキング通信の場合，送信側では送信バッファからシステムバッファへメッセージが送られるまで，受信側ではシステムバッファから受信バッファにメッセージが到着するまでプロセスは継続する処理が抑止される。またノンブロッキング通信の場合，送受手続きを呼び出した後，ただちに継続する処理が可能となる。MPI_WAITを呼び出して処理を抑止する。

一般的に利用頻度が高く，その中で並列処理効率に影響を与えると考えられるメッセージ交換ライブラリのみを含んだプログラムを実行し，実行性能を測定することで各プラットフォームのプロセッサ間通信について調査する。

測定に使用したサブルーチン名は以下の通りである。

(1) ブロッキング通信

① MPI_RSEND vs MPI_RECV（レディモード：受信手続きの呼び出しが先行する）

② MPI_SEND vs MPI_RECV（標準モード）

③ MPI_SENDRECV（標準モード：一回の呼び出しで送受を行う）

④ MPI_SSEND vs MPI_RECV（同期モード：送受双方の呼び出しが揃うまで処理が抑止される）

⑤ MPI_BSEND vs MPI_RECV（バッファモード：送信側でバッファを定義，送信手続き先行可能）

(2) ノンブロッキング通信

① MPI_ISEND vs MPI_RECV

倍精度実数（8Byte）配列を一つ上のRANK番号を持つMPIプロセスに転送し，データ転送量を要した時間（計測値）で除して帯域幅を計算した。MPIプロセス数は4で固定し，転送するデータ容量を1kBから32MBまで2倍ずつ順次増加させて計測した。計算では各サブルーチンコールを計測前に10回，計測時に100回ループさせてサブルーチンの立ち上がり時のオーバーヘッドの影響を少なくしている。ただし，SX-4とSPにおいてMPI_SSENDとMPI_BSENDは1回だけのコール，PCCにおいてMPI_SSEND, MPI_BSENDはコール不可能である。

2.2. 結　果

平均バンド幅と転送データ量の関係を4プロセスの総計とデータ送受信を行う単一プロセスそれぞれについてSX-4は図1，2，SPは図3，4，PCCは図5，6に示した。なお，各プラットフォームとも現有メモリの限界までのテストである。一つ上のRANK番号を持つMPIプロセスへの転送時間の計測であるため，4プロセスでの実行時にはデータ送信，受信ともに3回ずつ行われる。

SX-4の場合，ブロッキング通信の標準モードMPI_SENDRECV, MPI_SEND vs MPI_RECVが他のルーチンと比較して安定して高速である。512kBを境にそれ以下のデータ量ではMPI_SEND vs MPI_RECV, それ以上のデータ量ではMPI_SENDRECVが最高速となっている。サブルーチンによってバンド幅にはかなり差が見受けられる。SPの場合にはブロッキング通信のMPI_SENDRECVのバンド幅が比較的高い。結果にサブルーチンのばらつきがほとんど見られず，SPのスイッチング性能が良好であること

表1　各プラットフォームのスペック概要
Table 1 Summary of each platform SPEC

	ハードウェア	ソフトウェア
SX-4	20CPU 8GBメインメモリ 理論処理性能 1CPUあたり2GFLOPS 理論データ転送幅　1CPUあたり16GB/sec	OS: SUPER-UX R7.1 コンパイラ: FORTRAN90/SX version1.0 MPI: MPI/SX version 2.0.9
SP	POWER2 SuperChip 120MHz 8ノード 1ノードあたり128MBメモリ 理論処理性能 1CPUあたり480MHz 理論ノード間転送幅　150MB/sec	OS: AIX version4.1.5 コンパイラ: HPF version 1.1.0 MPI: MPI（IBM供給）version 2.2.0
PCC	PentiumIII 700MHz 4ノード 1ノードあたり256MHzメモリ 理論ノード間転送幅　**16.67**MB/sec	OS:RedHat Linux 6.0 kernel 2.2.12.-32 コンパイラ: g77 version 0.5.24 MPI: MPICH version 1.2.0

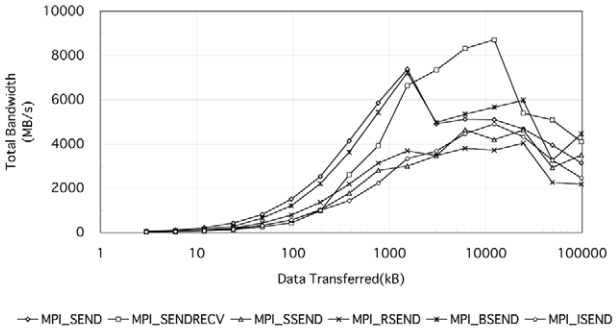

図 1　MPI使用時におけるシステムの総バンド幅（SX-4）
Fig. 1　Bandwidth with MPI :Total（SX-4）

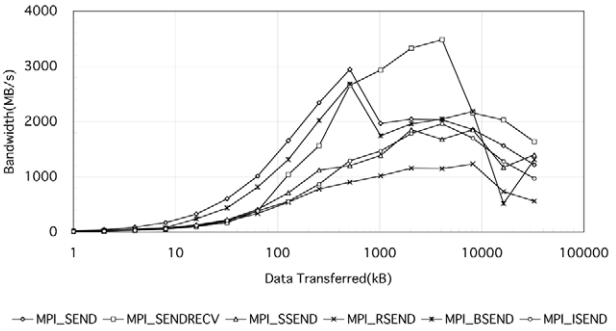

図 2　MPI使用時における1プロセスのバンド幅（SX-4）
Fig. 2　Bandwidth with MPI :Single（SX-4）

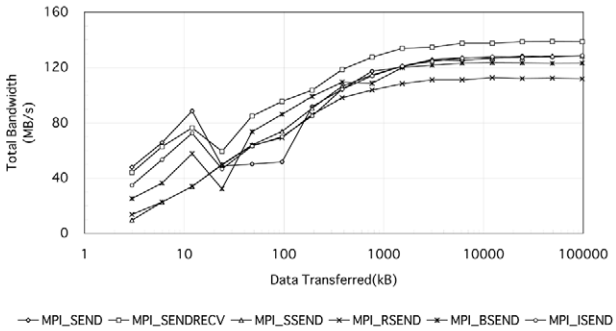

図 3　MPI使用時におけるシステムの総バンド幅（SP）
Fig. 3　Bandwidth with MPI :Total（SP）

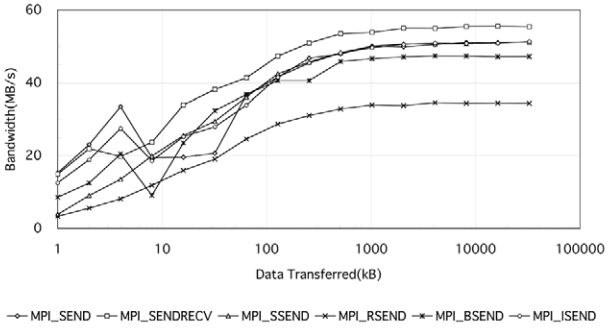

図 4　MPI使用時における1プロセスのバンド幅（SP）
Fig. 4　Bandwidth with MPI :Single（SP）

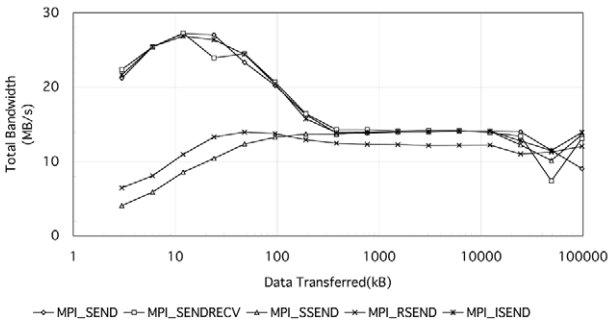

図 5　MPI使用時におけるシステムの総バンド幅（PCC）
Fig. 5　Bandwidth with MPI :Total（PCC）

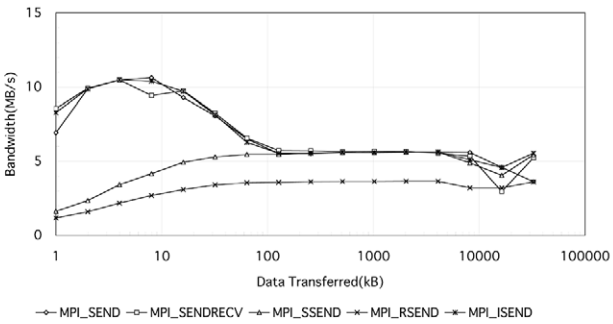

図 6　MPI使用時における1プロセスのバンド幅（PCC）
Fig. 6　Bandwidth with MPI :Single（PCC）

が分かる。データ転送量の増加に対してバンド幅は増加から飽和する傾向を示している。PCCに関しては他2つと異なり，データ転送量の増加に対しバンド幅が減少して飽和するサブルーチンが見られた。全体的にブロッキング通信のMPI_SENDRECV, MPI_SEND vs MPI_RECV, ノンブロッキング通信のMPI_ISEND vs MPI_RECVが高速であり，数値的に見てもほぼ同一である。データ転送量が少ない域にピークがあり，転送量が大きくなるに従い，1プロセスで2.5MB/secから5.5MB/secのバンド幅に収束していることが分かる。

2.3. 考　察

非共有メモリ型計算機のPE間ネットワークは安価・低速な回線が用いられているのが一般的で，上位層まで規定されたネットワークプロトコルを利用するとPE間で伝達すべきメッセージに様々なヘッダが付属し，さらにメッセージの転送を遅延化させる要因の一つとなる。それに対しSX-4ではメッセージ交換はMMU上で行われ（メッセージ転送はメモリ内でのコピーに相当），ほぼいかなるネットワーク転送幅をも上回っている。原則的に非共有メモリ型計算機と比較してメッセージのオーバーヘッドが非常に小さく済み，通信に費やされている時間は短く済むというメリットを有する。

SX-4，SPの両グラフからサブルーチンの立ち上げのオーバーヘッドがあるため，ある程度以上のメッセージ長を確保しないと十分な帯域幅が得られないことがわかる。SX-4，SPともに4MB程度の転送で最大バンド幅が得られている。計算機の平均的な負荷が少ないタイミングを見計

らって計測するようにしたが，一様な傾向を示すとは言い難いサブルーチンもある。サブルーチン呼び出しのタイミングでのシステム状況が関係すると思われる。PCCに関してはバンド幅の値が全般的に低く，転送量が大きくなるにつれ，減少する傾向さえ起きている。明らかにネットワークがボトルネックとなるような要因を見せている。

PCCは100Base-TXのEthernetスイッチングハブを使い，インターネットで使用するために標準化されたプロトコルipで通信を行う実装であるのに対し，SPはノード間専用スイッチを持ち，それに対応するUS通信サブシステムで通信を行う実装である。US（User Space）はSPスイッチへの直接アクセス用に最適化され，SP ハードウェアのパフォーマンス能力を最大にするメッセージ交換ライブラリである。ノード間のネットワークトポロジーは両者同一であるが，通信におけるプロトコルが標準仕様であるか，最適化された特別仕様であるかの違いがある。Ethernet接続であればどんな規模や形態でも実行可能であるPCCであるが，それゆえ不要な情報がデータのヘッダに含まれ，それがスイッチングにおいても影響を及ぼす。それに対しSPではスイッチを含めたノード間通信が最適化されているため実行結果に大きな差が見られたと言える。

6種類のサブルーチンで帯域幅を計測したが，これらはデータ送信前にある特別な準備を行うか否かで送信前に何も行わないサブルーチン(1)②③(2)①とデータ送信前にそれぞれで何らかの特別な準備を行うサブルーチン(1)①④⑤に大別できる。SPとPCCの結果から明らかにこの2つのグループ内のサブルーチンは類似した挙動を見せていることが分かる。しかしながら，SX-4では大容量のデータ転送においてMPI_SENDRECVが突出した結果を示している。このことからベンダーが特にMPI_SENDRECVに対してチューニングを施し，最適化していると予想される。

3. 姫野ベンチマークによるMPI並列実行性能

3.1. 計算方法

姫野ベンチマーク（以下姫野ベンチ）とは姫野龍太郎氏（理化学研究所情報環境室室長）が非圧縮流体解析コードの演算性能評価の目的で考案したベンチマークテストで，非圧縮性流体の支配方程式系から導出される圧力のPoisson方程式をJacobi法で解く場合での処理速度を測るものである。圧力のPoisson方程式の求解がホットスポットになるため，姫野ベンチでの性能が実際の流体計算においての演算性能を模擬すると言われ，しばしば流体解析システムのベンチマークテストに用いられる。これによる測定値は一般に逐次処理版であればプロセッサ単体における計算速度とメモリバンド幅の実効値の指標として用いることができる。MPI並列版においては各プロセッサで処理を分担し，MPIシステムの処理能力を測定できる。

ここではSX-4，SP, PCCのそれぞれでSサイズ（65×33×33），Mサイズ（129×65×65），Lサイズ（257×129×129）のメッシュサイズを使用して計算を行い，システムの処理速度（FLOPS）を測定する。この計算で用いているベンチマーク用プログラムはFortran77に準拠したMPI版である。プログラムではMPI_SENDRECVのみを用いている。使用したプロセッサ数はすべて4である。1次元領域分割法で並列化を行っている。単精度で計算を行っているため，1回のデータ送信における通信量はSサイズで8580Byte, Mサイズで33540Byte, Lサイズで132612Byteである。

また，参考として逐次処理版に対しても各プラットフォームで計算を行い，単体の処理能力を測定する。

3.2. 計算結果

図7～9にそれぞれSX-4，SP, PCCによるMPI版，表2に逐次処理版の姫野ベンチマークテストの結果を示す。なお，N/Aについてはプログラムサイズが実メモリ容量を超え，スワッピングが発生し，I/Oによる大きな遅延が起こるため比較対象とならない場合である。

どのグラフからもプロセッサの数とともに演算処理速

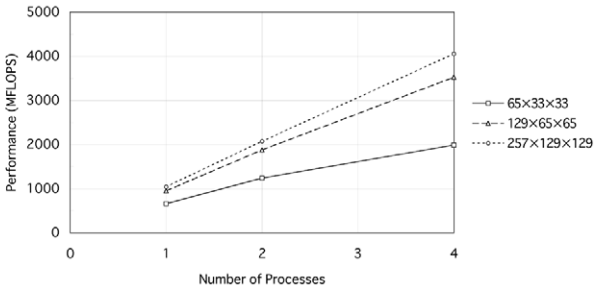

図 7　姫野ベンチマークテストMPI版（SX-4）
Fig. 7　Himeno Benchmark Test with MPI（SX-4）

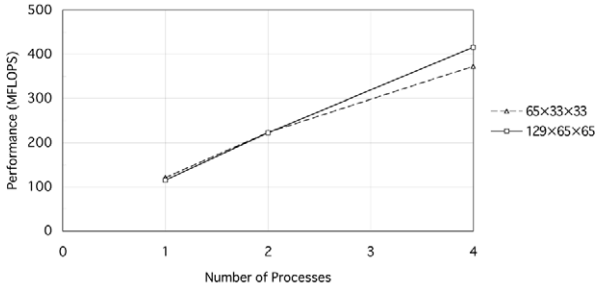

図 8　姫野ベンチマークテストMPI版（SP）
Fig. 8　Himeno Benchmark Test with MPI（SP）

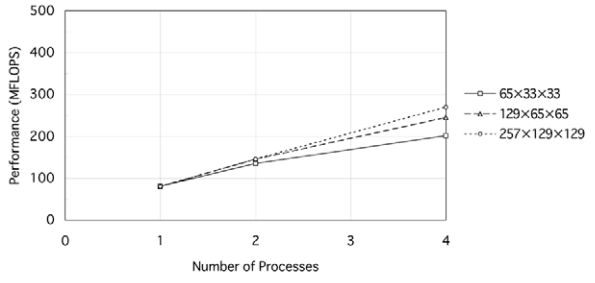

図 9　姫野ベンチマークテストMPI版（PCC）
Fig. 9　Himeno Benchmark Test with MPI（PCC）

表2　姫野ベンチマークテスト逐次処理版
Table 2 Himeno Benchmark Test with sequential processing

	SS (65×33×33)	S (129×65×65)	M (257×129×129)	L (513×257×257)
SX-4	1099.826087	1457.31823	1791.323385	1781.163
SP	117.3849487	112.7338867	N/A	N/A
PCC	80.2198792	78.529775	88.4549255	N/A

度が向上していることが分かる。しかしながら，並列化の有効性の観点から見るとSX-4，SPにおいて良好なスケーラビリティを示しているが，PCCでは並列化の効果が2者に比べ，並列化効果が発揮されていない。

逐次処理版とMPI版1プロセスの結果を比較するとSP，PCCではほぼ同一の処理速度が得られているのに対し，SX-4ではMPI版1プロセスの結果が逐次処理版の約70%程度の演算処理速度を示していることがわかる。

3.3. 考　察

一般的に流体解析プログラムはベクトル計算向きと言われ，またシステムの規模から考えて3者を比較すればSX-4のパフォーマンスが際だっていることは当然であるが，特筆すべきはSPとPCCのパフォーマンスである。SPはベンダーが最適化しているコンパイラやライブラリを提供しての性能であるが，PCCに至っては市販で購入できるPCの粗結合接続とフリーのコンパイラといったいわば寄せ集めである。しかしながら，逐次処理版の結果において性能差がそれほど見られない理由の一つはハードウェアの驚異的な発達によるものと言える。

本調査は最大4ノードまで使用した場合での計算結果であり，この段階ではどのプラットフォームにおいてもMPIプロセス数の増加に応じて処理性能が向上している。メッシュサイズを固定し，MPIプロセス数を増やした場合，各プロセスが担う計算量は減少するが，通信量は変化せず，計算量に対する通信量の割合が大きくなることから今回の結果で見られるようにプロセッサ数に比例して処理速度が同様に向上していくとは考えづらい。

MPI版ではSPと比較してPCCのスケーラビリティが低い結果であり，両者の性能差は開く傾向にある。姫野ベンチは3次元配列での計算プログラムであるため，並列化するために領域分割すると二次元領域分の通信が発生し，1回のデータ転送における通信量は大きくなる。姫野ベンチでの各メッシュサイズでのデータ転送量を2. の結果に照らし合わせるとSPではメッシュサイズが大きくなるにつれバンド幅も増加している領域にあるのに対し，PCCでは逆にバンド幅が減少する領域に相当している。すなわち，転送量が大きくなると転送効率が悪くなり，転送時間がさらに多く費やされることになる。しかも上述のようにメッシュサイズを固定してMPIプロセス数を増やした場合，1プロセスあたりの計算量は減少するが全データ転送量は増加する。従ってPCCのスケーラビリティはメッシュが増加するにつれ，さらに低くなると解釈できる。

逐次処理版とMPI版1プロセスの結果からSP, PCCではほぼ同一の処理速度が得られているのに対し，SX-4ではMPI版1プロセスの結果が逐次処理版の約70%程度の演算処理速度を示している。これはデータ転送を伴わないサブルーチンコールのみでの差を意味していることからSX-4ではサブルーチンコール時のシステム状況が敏感に処理性能に反映され，2. におけるSX-4の結果に示されているばらつきと同一の現象であると考えられる。

4. 数値解析に用いるアルゴリズムのMPI並列性能

4.1. 計算方法

並列処理において最も重要な観点は複数のプロセッサを効率よく動作させることである。MPIプロセス数をn，その場合に要した経過時間をT_nとすれば，加速率

$$S_n = T_1 / T_n \quad (1)$$

，並列化効率

$$P_n = T_n / T_1 \quad (2)$$

で並列処理による効果を表す値が定義される。加速率は並列化することで逐次処理をどれほど高速化したのか，並列化効率は逐次処理に比較して各プロセスがどれほど計算に専念したかの平均を示している。S_nがnに近いほどスケーラビリティが高く，P_nが1に近いほど通信が少ない処理と判断できる。これらの値は使用するプログラムのタイプに大きく依存し，逐次処理部分の割合が多いほど効率を下げることになる。

また効率に重大は影響を及ぼす要因としては他にプロセッサ間通信速度やロードバランスがある。プロセッサ間通信の速度が遅く，しかも通信の回数が多い場合には並列処理の効率は悪くなるし，各プロセスに与えられる計算量が不均一であると，処理に無関係な同期待ちが多くなり，やはり効率は悪くなる。

4.1.1. πの求値における性能測定

MonteCarlo法は多数のランダムな数値実験を繰り返し

行うことにより，近似解を統計的に求める方法で，確率的な挙動を示す現象に用いられる。ここでは決定的なπの値を確率論的に求めることにし，その比較として数値積分によっても求めることにする。

(1)MonteCarlo法

正方形領域(0≦x≦1，0≦y≦1)内に一様乱数で点を打つと，それが半径1の4分円に入る確率はπ/4であることを使って，πを求める。ヒットミス法で乱数を決定し，総試行回数は正の整数の取りうる最大の2,147,483,647(=2^{31}-1)回である。異なる種をMPIプロセスに割り当て各プロセスで異なる乱数系列を発生させ，発生回数と該当回数の各々の総和を取ることで求める。MPIプロセス数はSX-4で1, 2, 4, 8, 16，SPで1, 2, 4, 8，PCCで1, 2, 4である。

(2)数値積分

被積分関数$4/(1+x^2)$，積分領域$[0,1]$の分割数はMonteCarlo法の場合と同様，取りうる最大の2,147,483,647(=2^{31}-1)である。

分割領域をMPIプロセスに割り当て各々が部分和を計算した後，総和を求める。MPIプロセス数はSX-4で1, 2, 4, 8, 16，SPで1, 2, 4, 8，PCCで1, 2, 4である。

4.1.2. 二次元拡散方程式(放物型偏微分方程式)における性能測定

一般に物質拡散や熱伝導を支配する方程式である。

$$\frac{\partial \phi}{\partial t} = c\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right),\ \phi = \phi(x,y,t),\ c = \text{const.} \qquad (3)$$

上式を差分法を用いて離散化する。空間軸方向に交互に走査するADI法を用い，行列はJacobi法を使った反復計算により解く。領域分割法(1次元)で並列化し，各MPIプロセスに負荷分散させる。使用MPIプロセス数はSX-4で1, 2, 4, 8, 16，SPで1, 2, 4, 8，PCCで1, 2, 4である。計算したメッシュサイズはSX-4では(513×513)，(1025×1025)，(2049×2049)，SPとPCCでは(17×17)，(129×129)，(257×257)である。

4.1.3. 二次元Laplace方程式(楕円型偏微分方程式)における性能測定

静電場や重力場，定常状態の温度分布，ポテンシャル流の流れ場などの支配方程式である。

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0,\ \phi = \phi(x,y) \qquad (4)$$

差分法で離散化する。SOR法による緩和計算をするため，領域内格子をチェッカーボード状に互いに二分し(Red-Black法)反復計算を交互に行う。並列化は領域分割法(1次元)で行い，使用したMPIプロセス数はSX-4で1, 2, 4, 8，SPで1, 2, 4, 8，PCCで1, 2, 4である。計算したメッシュサイズはSX-4では(129×129)，(257×257)，(513×513)，(1025×1025)，(2049×2049)，SPとPCCでは(129×129)，(1025×1025)，(2049×2049)である。

4.1.4. 非圧縮性流体方程式系における性能測定

非圧縮性流れの支配方程式は質量保存を表す連続の式(二次元の場合(5)式)と運動量保存を表すNavier-Stokes方程式(二次元の場合(6)式，(7)式)の連立偏微分方程式で表される。x軸方向速度を$u = u(x, y, t)$，y軸方向速度を$v = v(x, y, t)$，圧力を$p = p(x, y)$，Re = const. とすると，

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \qquad (5)$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \qquad (6)$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \qquad (7)$$

上記連立偏微分方程式から圧力のPoisson方程式を導出する。差分法を用いて離散化し，MAC(Marker and Cell)法を用いて計算を行う。圧力の緩和にはRed-BlackSOR法を用いる。流れ場は二次元キャビティ流れ(Re=100)を考え，並列化には領域分割法(1次元)を用いている。MPIプロセス数はSX-4で1, 2, 4, 8, 16，SPで1, 2, 4, 8，PCCで1, 2, 4である。計算したメッシュサイズは(65×65)，(129×129)である。

なお，すべてのプログラムで領域間データ転送にはMPI_SENDRECVを用いた。

4.2. 結　果

経過時間の測定値を2つの指標(加速率Sn, 並列化効率Pn)に処理し，プロセス数に対する値をグラフ化した。

4.2.1. πの求値における性能測定

図10～12にそれぞれSX-4，SP, PCCでの並列拡張性，図13～15に並列化効率を示す。

SX-4の結果ではプロセッサ数の増加とともにほぼ線形に加速率も増加し，良好なスケーラビリティを示し，これは0.9以上の並列化効率を見ても分かる。SPでは4ノードから8ノードで変化が見られ，加速率の傾きや効率の減少があるが，これはSPは8ノード構成であるためその内の1ノードは計算以外のプロセス(OS等)で資源を消費しているためのものである。これはPCCに関しても同様である。

4.2.2. 二次元拡散方程式における性能測定

図16～18にそれぞれSX-4，SP, PCCでの並列拡張性，図19～21に並列化効率を示す。

SX-4 に関しては大きいメッシュサイズ(2049×2049)で良好なスケーラビリティ，効率が得られた。しかしそのTrade-offとして膨大なCPU時間を費やしている。SP, PCCではメッシュサイズを大きくしても効果的な性能向

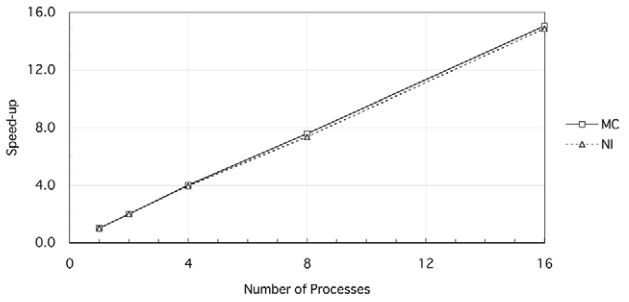

図10　πの計算における並列拡張性（SX-4）
Fig.10 Scalability for π calculation（SX-4）

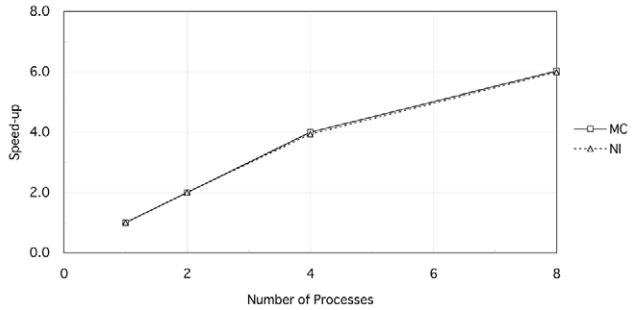

図11　πの計算における並列拡張性（SP）
Fig.11 Scalability for π calculation（SP）

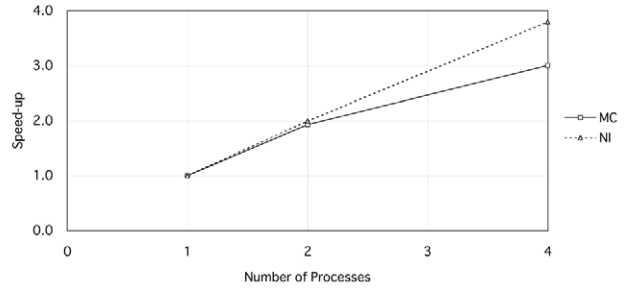

図12　πの計算における並列拡張性（PCC）
Fig.12 Scalability for π calculation（PCC）

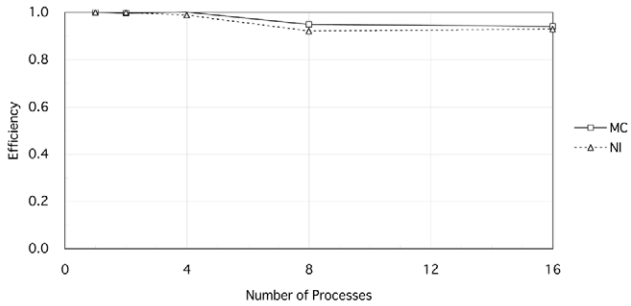

図13　πの計算における並列化効率（SX-4）
Fig.13 Parallel Efficiency for π calculation（SX-4）

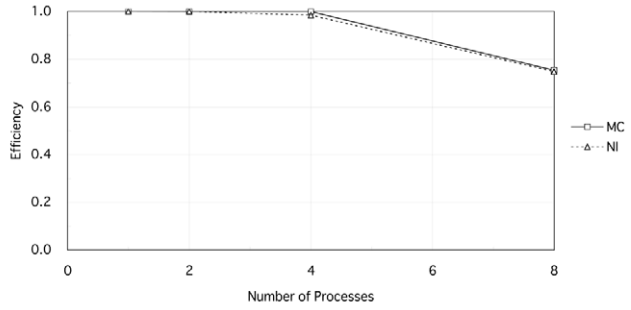

図14　πの計算における並列化効率（SP）
Fig.14 Parallel Efficiency for π calculation（SP）

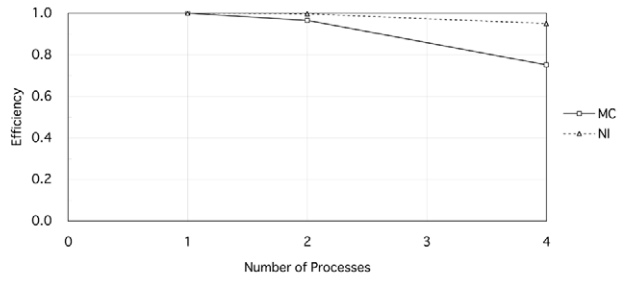

図15　πの計算における並列化効率（PCC）
Fig.15 Parallel Efficiency for π calculation（PCC）

上が得られておらず，並列化効率についても良好な結果はでていない。

4.2.3. 二次元ラプラス方程式における性能測定

図22～24にそれぞれSX-4，SP, PCCでの並列拡張性，図25～27に並列化効率を示す。

SX-4ではメッシュサイズが大きくなるにつれ良好なスケーラビリティ，効率を示している。SPでは（1025×1025）以上のメッシュサイズでほぼ線形に近い加速率が得られた。並列化効率も0.8超を示し，良好な結果が得られている。PCCに関してもSPと同様，良好な結果を示している。

4.2.4. 非圧縮性流体方程式系における性能測定

図28～30にそれぞれSX-4，SP, PCCでの並列拡張性，図31～33に並列化効率を示す。

SX-4での計算ではアルゴリズムはベクトル計算向きであるが，問題の規模が小さくベクトル長が取れないため効果的な加速率向上が得られず，加速率の飽和が見られている。SPでの計算では加速率に関してはSX-4とほぼ同様であるが，効率は比較的高くなっている。PCCに至っては逐次処理を下回る結果が得られている。

4.3. 考　察

4.1.1.での計算はMonteCarlo法・数値積分法とも各プロセスの部分和計算の終了後最後に値をまとめて加算を行うという，いわば並列処理向きの計算内容であるため，SX-4，SP, PCCどれも，ほぼ同じプロセス数に比例する性能が得られ，優れたスケーラビリティを示した。また並列化効率が0.9以上を示しているのも確認された。

行列の反復解法としてJacobi法とGauss-Seidel（SORも同

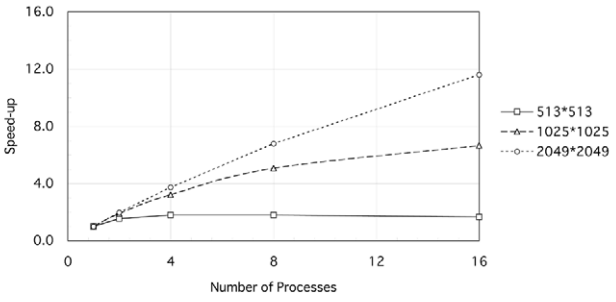

図16　拡散方程式の計算における並列拡張性（SX-4）
Fig.16 Scalability for solving Diffusion Eq（SX-4）

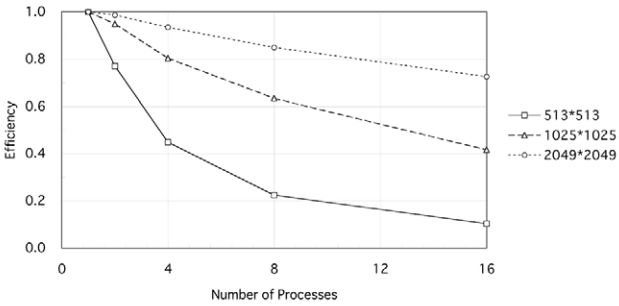

図19　拡散方程式の計算における並列化効率（SX-4）
Fig.19 Parallel Efficiency for solving Diffusion Eq（SX-4）

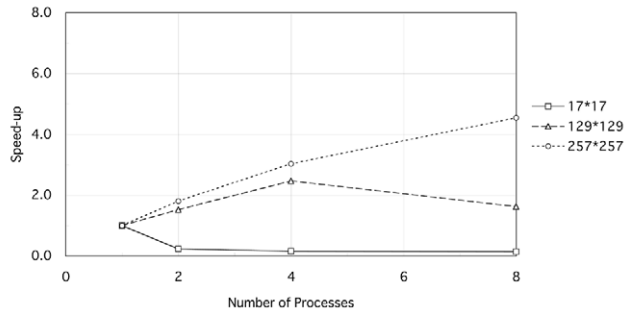

図17　拡散方程式の計算における並列拡張性（SP）
Fig.17 Scalability for solving Diffusion Eq（SP）

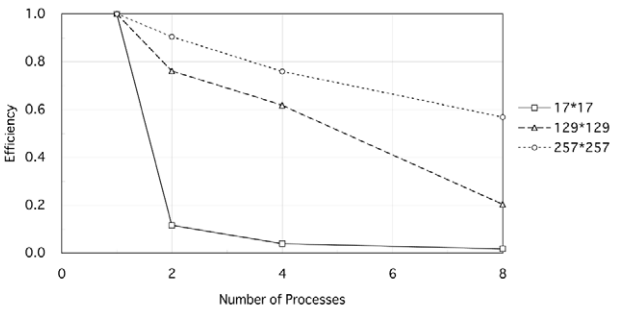

図20　拡散方程式の計算における並列化効率（SP）
Fig.20 Parallel Efficiency for solving Diffusion Eq（SP）

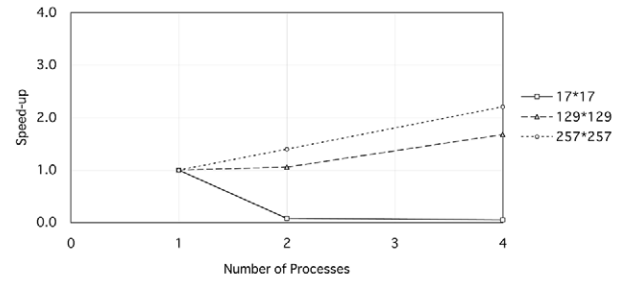

図18　拡散方程式の計算における並列拡張性（PCC）
Fig.18 Scalability for solving Diffusion Eq（PCC）

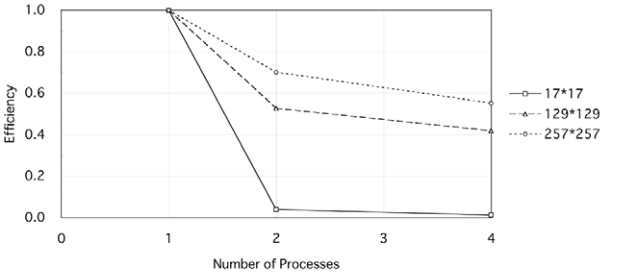

図21　拡散方程式の計算における並列化効率（PCC）
Fig.21 Parallel Efficiency for solving Diffusion Eq（PCC）

様）法が使用される場合が多い。Jacobi型は一度に配列値を更新するのに対し，Gauss-Seidel（SORも同様）型は逐次に更新する。領域分割法で並列化する場合，Jacobi法はメモリ上で連続した配列を転送すればよいが，Gauss-Seidel法はアルゴリズムから言ってメモリ上で不連続な配列を隣接したプロセスへ転送する必要がありRed-Black格子を用いると，Jacobi法と比較して1/2のデータ容量の転送となるが2倍の回数のサブルーチンコールをしなければならない。4.1.2.の計算において用いたJacobi法は逐次計算では他の反復解法に比較して収束が遅いことが知られているが，プログラムの並列化作業はプログラム構造の変更を伴わないので比較的容易に済むというメリットがある。それに対し，4.1.3.，4.1.4.ではJacobi法に較べ元来収束が速いSOR法を用いているので，4.1.2.で消費した計算時間より遙かに少ない時間で収束が見られたが，並列計算ではベクトル計算時と同様にRed-Black格子を構成するような工夫なしでは並列化できす，並列化作業は煩雑となる。

4.1.2.～4.1.4.の計算で用いた領域分割は1次元方向でのみ行っており，その境界に沿った1次元方向の配列をあるタイミングで互いに転送しあっている。メッシュサイズのみ増加させると転送すべき境界上配列は1次元的な増加にとどまる（たとえばメッシュサイズ全体が4倍になっても転送すべき配列は2倍しか増加しない）。このことは一定のMPIプロセス数の下では格子数の増加に伴う通信量の増加よりも計算量の増加が凌駕し，並列化がより効果的であることが期待できる。一方で1CPUの処理性能限界があるために加速率の向上には上限が存在することは必定である。一定の格子数の下ではMPIプロセス数の増加は分割領域数の増加であり，並列度が増すという意味では全体のスループットは上がる。しかし計算量に対し通

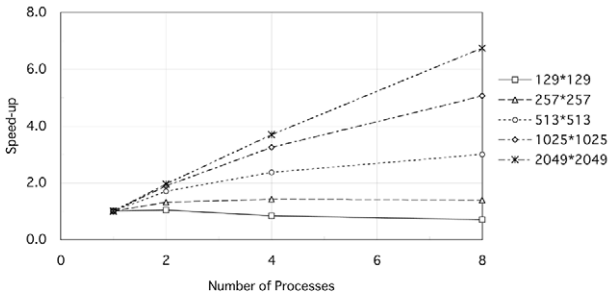

図22　Laplace方程式の計算における並列拡張性(SX-4)
Fig.22 Scalability for solving Laplace Eq(SX-4)

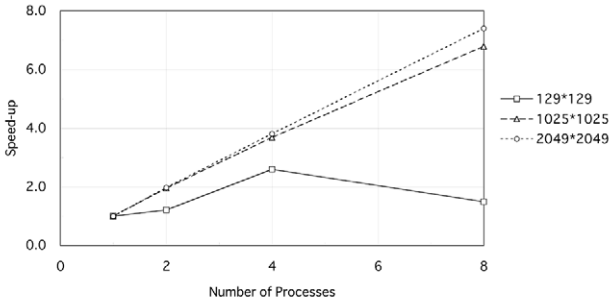

図23　Laplace方程式の計算における並列拡張性(SP)
Fig.23 Scalability for solving Laplace Eq(SP)

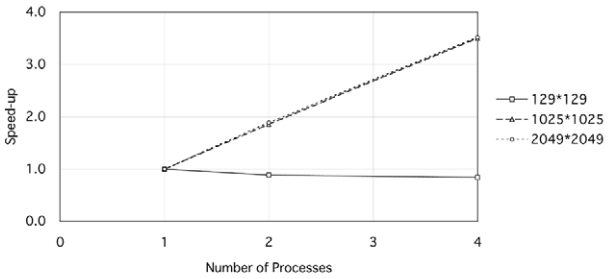

図24　Laplace方程式の計算における並列拡張性(PCC)
Fig.24 Scalability for solving Laplace Eq(PCC)

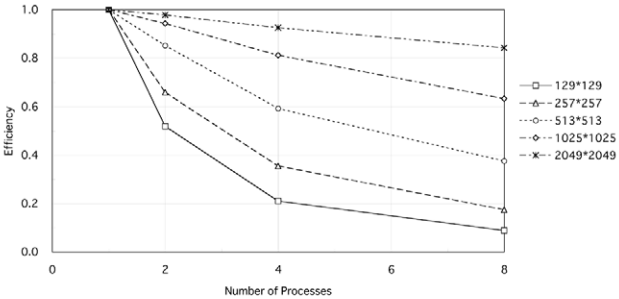

図25　Laplace方程式の計算における並列化効率(SX-4)
Fig.25 Parallel Efficiency for solving Laplace Eq(SX-4)

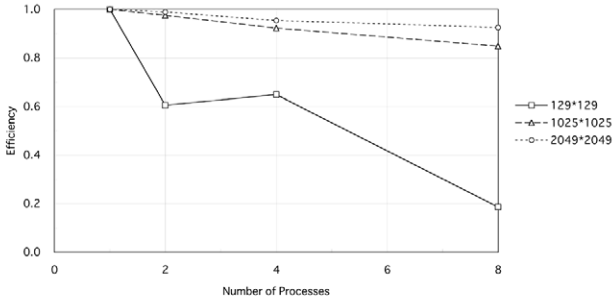

図26　Laplace方程式の計算における並列化効率(SP)
Fig.26 Parallel Efficiency for solving Laplace Eq(SP)

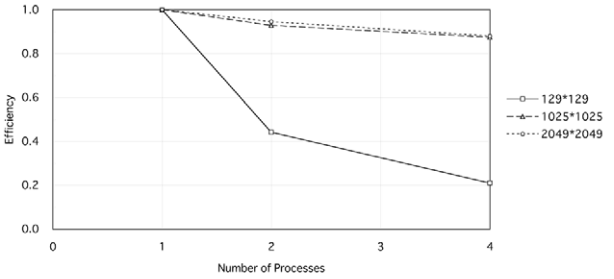

図27　Laplace方程式の計算における並列化効率(PCC)
Fig.27 Parallel Efficiency for solving Laplace Eq(PCC)

信量が増加することであるので，領域数増加が行き過ぎるとネットワークの物理的なバンド幅の不足，MPIサブルーチンの呼び出しのオーバーヘッドの増加で計算の停滞を招き，むしろ全体性能を下げることにつながる。

5. まとめ

5.1. MPI特性

2. ～4. の結果から，各プラットフォームにおけるMPI特性は以下のようにまとめられる。

SX-4：

メモリアーキテクチャの構造からして，他と比べ最良のバンド幅が得られるが，通信性能はシステム状況により大きく左右される。大容量のデータ転送時にはMPI_SENDRECVの性能が特化している。問題規模が小さい場合にはベクトル長が取れず，また，データ転送時におけるサブルーチンの立ち上げ時のオーバーヘッドによる影響も大きいため，良好なスケーラビリティは得られないが，他のいずれの場合(本調査において取り扱ったデータ容量の範囲内)でも良好な並列化効果が得られると期待できる。

SP：

スイッチング性能が良好であるゆえ，データ転送結果に大きなばらつきは発生しない。データ転送能力においてSX-4と比較すると劣るが安定している。PCCと比較すると高速かつ安定している。MPI_SENDRECVの結果が良好である。全てのノードを用いて並列化を行うと8ノードのうち1ノードはOS等計算以外のプロセスで資源を消費しているため，並列化効果が減少する。ある程度の計算量(メッシュサイズ)がないと通信量の割合が計算量に比べて大きくなり並列化効果が得られない。データ転送量

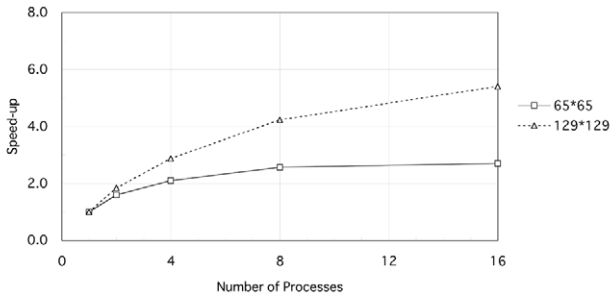

図28　流体方程式系の計算における並列拡張性（SX-4）
Fig.28 Scalability for solving Fluid Eqs（SX-4）

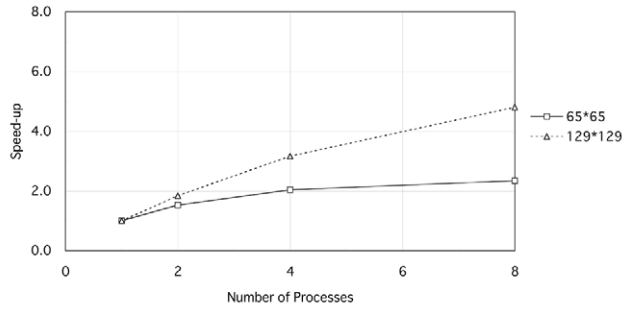

図29　流体方程式系の計算における並列拡張性（SP）
Fig.29 Scalability for solving Fluid Eqs（SP）

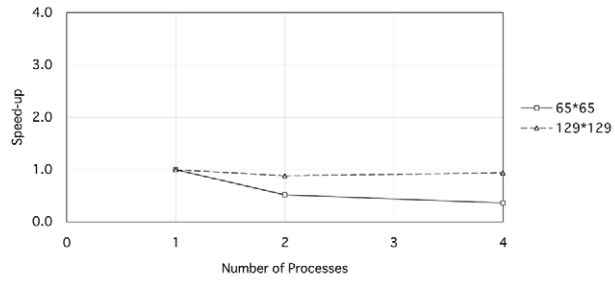

図30　流体方程式系の計算における並列拡張性（PCC）
Fig.30 Scalability for solving Fluid Eqs（PCC）

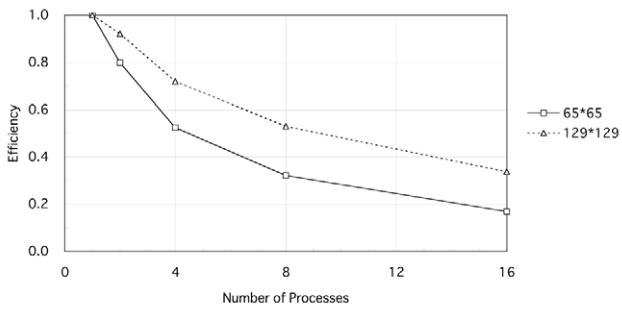

図31　流体方程式系の計算における並列化効率（SX-4）
Fig.31 Parallel Efficiency for solving Fluid Eqs（SX-4）

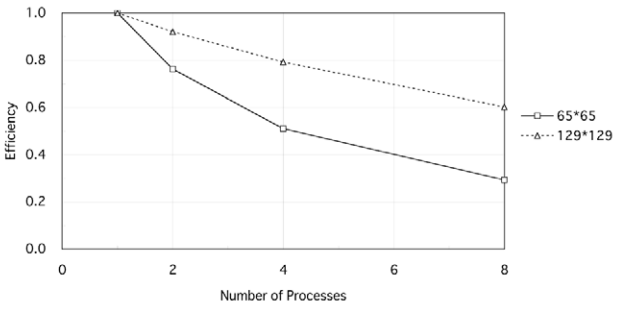

図32　流体方程式系の計算における並列化効率（SP）
Fig.32 Parallel Efficiency for solving Fluid Eqs（SP）

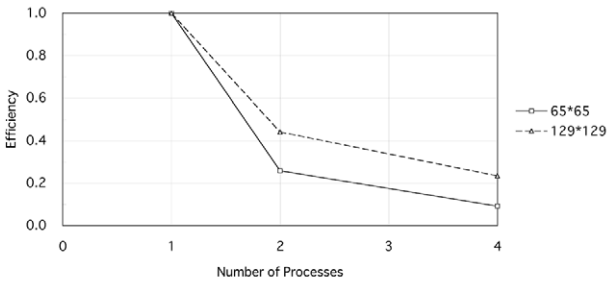

図33　流体方程式系の計算における並列化効率（PCC）
Fig.33 Parallel Efficiency for solving Fluid Eqs（PCC）

の増加に対してバンド幅は増加から飽和を示すため，効率が大きく低下するしきい値は現れない。

PCC：

ブロッキング通信の標準モードとノンブロッキング通信において他のサブルーチンと比べ，良好な結果を示すが，データ転送量の増加に伴ってバンド幅が減少し飽和する傾向がある。ゆえにノード間通信にボトルネックがあり，通信量と比べ計算量の割合が著しく大きい場合以外，並列化効果は期待できない。1ノードの演算処理能力においてはコストから見て満足できる性能がでるもののノード間通信性能が著しく悪いため，データ転送が必然である並列処理において他2機種との差は大きい。

5.2. 並列化指針

以下に5.1.の見解を基にプログラムを並列化する場合の指針を示す。

各プラットフォーム共通：

どの計算機においてもMPI_SENDRECVは並列化効果を最大に引き出すことができるサブルーチンであるゆえ，特別な処理が必要な場合以外，MPI_SENDRECVを選択することが最良である。但しデータ転送量が小さい場合にはMPI_SENDが最良となる場合もある。

プロセス間通信が多発する問題を解く場合や構成ノード数全てを使用して並列計算を実行する場合，システムの利用状況により効率にばらつきがでることを考慮しておくべきである。

通信量に対して計算量が少ない問題では並列化効果が期待できず，逐次処理で十分となる場合もある。

SX-4：

使用できるプロセッサ数の範囲内で並列化の効果が

期待できるものは，1MB以上のデータを取り扱う問題であり，なおかつ，さらに特徴的な効果を発揮するものは大通信容量で，サブルーチンコール回数が比較的少ないコード，ベクトル化可能なコードである。ただし4MB以上のデータ転送を扱う問題に関して並列化効率は飽和もしくは減少を見せる可能性がある。

安定した性能を期待する場合にはサブルーチンコールを抑えることが必要である。

SP：

演算性能と通信性能のバランスがとれたシステムであるが，（搭載メモリ容量やCPU性能などから）現システムでは小規模なプログラムでないと十分は並列化効果を期待するのが難しい。

PCC：

PCCにおいて唯一並列化効果が期待できる状況は計算量が通信量を凌駕している場合のみである。

6. おわりに

本研究において，各プラットフォームでのMPI通信ライブラリ特性，並列システムの計算性能，各種テストプログラムによる並列化拡張性，効率を調査した。テスト環境ではセンター既存のシステムを用い，各プラットフォームでの結果比較等の関係でその中でも使用ノード数など限られた条件を付加しているため，それぞれにおいて精密に性能評価をするまでには至っていない。しかしながら，各プラットフォームにおけるMPIプロダクトのおおよその特性を知見することができた。これを基にプログラムの規模や構造を見極め，計算量とデータ通信量を評価することで，十分に効果的な並列化を行う指針が得られた。

PCCについては今後，拡張できる唯一のシステムであるゆえ，本調査で性能不足が見られた点について増強可能である。ノード数の面での拡張，他スペックのPCでの構成，ノード間通信媒体について100Base-TXのEthernetより高速なネットワークへの拡張，ノード間接続のトポロジーの変更など様々な拡張要素はある。いずれかによる拡張後でのSPとの性能比較で興味深い結果が得られると期待される。

今後，海洋計算プログラムに適用できるか否かを検証するために，本調査結果を活用する。しかしながら，MPIは並列化を実行するためのライブラリであり並列化の手法ではない。検証には海洋計算プログラムを分析し，その特性を把握しなければならないのは言うまでもないが，その特性に合う並列化手法とプラットフォームにおけるMPI特性について調査し，最適な並列化条件を得ることが必要になる。それにより海洋計算プログラムに対して並列化による計算効率の向上が確実に期待できると考える。

（原稿受理：2000年8月18日）