博 士 論 文

有限差分法による非圧縮粘性流体シミュレーションの高性能並列計算の研究

指導教官　松澤 照男 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

黒川 原佳

2002 年 1 月 10 日

要　旨

本研究では，高効率な CFD (Computational Fluid Dynamics) 計算を行なう 3 つの手法について検討し，その手法を適用した CFD 計算における並列性能を明らかにした．CFD 並列計算の効率を効率よく向上させるためには，3 つの方法を個々に検討することでは不十分である．これらの方法は，個々に検討が行なわれていたが，3 つの手法によって総合的に性能を向上させる試みは行なわれていない．これら 3 つの方法を MAC (Maker And Cell) 法を用いた非圧縮粘性流体のシミュレーションプログラムに対して適用した．本研究で用いる各方法は，既存シミュレーションコードや方程式解法アルゴリズムから大きな変更なしに実現できる．そのため，CFD 分野の研究者が容易に実現可能な方法であるため，並列 CFD 計算の効率向上に非常に役立つと考えられる．

MAC 法を用いる非圧縮粘性流体シミュレーションを行なった．計算対象は，三次元計算モデルとして，管内流れ問題，キャビティー問題と円柱周りの流れを対象とした．キャビティー問題では，単純形状であり計算精度の検証が行ないやすい．また円柱周りの流れ計算では，非定常非圧縮粘性流体の並列シミュレーションの総合的な性能 (計算性能，収束性能等) と有効性を議論した．

本研究で検討した 3 つの並列 CFD 計算の性能を向上させる手法は次の通りである．

1. 領域分割パターンの適値の選択方法
2. 通信時間の削除方法
3. スカラー型並列計算機に適した Multi-Colored Line SOR (Successive Over Relaxation) (MCLSOR) 法

領域分割パターンの適値の選択方法は，領域分割法における分割パターンによる性能変化を予測し，最も性能が高くなる分割パターンを選択する方法を示した．通信時間の削減方法は，1 の手法を踏まえて CFD 計算における通信時間を通信隠蔽法 (非同期通信-計算重複法) を用いて削減し，並列計算性能を向上させる方法を示した．MCLSOR 法は，スカラー型並列計算機において計算処理性能を向上させる方法であり，上記の 2 方法と組み合わせれば，並列 CFD 計算をより効果的に実行できる．

これらを組み合わせることで得られる CFD シミュレーションの並列化性能の向上は非常に高いものになった．最適な領域分割パターンの選択方法において，ほぼ

100 % の確率で最適な分割方法が得られた．最適な分割パターンの選択を行なった，通信隠蔽処理において 15 ∼ 30 % 程度の並列化性能の向上が得られた．MCLSOR 法においては，逐次処理の性能で Multi-color SOR 法に対して 40 % 程度の性能向上が得られ，並列化においても性能低下は大きく見られず．通信隠蔽処理を適用した場合はさらに効率が向上することが分かった．これら全ての方法を適用した場合，経験則と従来法による方法に対して，約 2 ∼ 3 倍の性能向上が可能である．

Abstract

In this research, I examined three effective techniques for highly effective, parallel Computational Fluid Dynamics (CFD) simulation. I clarified the parallel performance in parallel CFD simulations to which applied those techniques. It is insufficient to examine three methods individually to improve the efficiency of a parallel CFD simulation. These methods have been individually examined already. However, the performance improvement using all techniques examined. To show the effectiveness of the three techniques, I discussed the effectiveness of the individual technique. Then, at first I applied the methods at the same time to the incompressible viscous flow simulation program, which used the MAC (Maker And Cell) method. Each method used in this research can be achieved without a big change from the existing simulation program code. This is very useful for the efficiency improvement of a parallel CFD simulation. I measured the improvement of the parallel computation performance of the CFD simulation and found it very high. In the CFD simulations, I measured incompressible viscous flow problems using the MAC method. These were the pipe flow problem, the cavity flow problem and the flow around circular cylinder problem as a three-dimensional model.

The methods of the performance improvement of the parallel CFD computation examined by this research are as follows.

1. a selection method for better domain partitioning pattern
2. a method of cutting down the communication time
3. a Multi Colored Line SOR (MCLSOR) method for a scalar type parallel machine

I propose the selection method for better domain partitioning pattern, which can predict the performance change in the domain decomposition method by the partitioning pattern. In addition, the selection method of the partitioning pattern prevents the performance deterioration for the domain decomposition method. The method of cutting down the communication time is an overlapping communication with computation method (the systolic communication-computation overlap method). The systolic communication-calculation overlap method cut down

the communication time in the parallel CFD simulation. I showed the actual improvement of the parallel performance. The MCLSOR method is an improvement method of the computation processing performance on a scalar parallel computer. The program code of the CFD simulation will execute a parallel computation more effectively, when the selection method and the systolic communication-computation overlap method, the MCLSOR method combine with the parallel computation.

The CFD simulation that used these methods, obtained a high parallel performance. When the selection method of the best partitioning pattern for domain decomposition was used, the best partitioning pattern was obtained by almost 100 %. The systolic communication-computation overlap method by which the best partitioning pattern for domain decomposition was obtained the improvement of the parallel performance about 15$\sim$30%. In the MCLSOR method, the serial version of the MCLSOR method obtained about 40 % higher performance against the Multi-color SOR method, in addition, the performance did not decrease seen greatly in the parallel processing. Parallel efficiency improved when the systolic communication-computation overlap method was applied to the MCLSOR method. The parallel performance improves about 2 $\sim$3 times of the conventional system, when the methods are applied to CFD simulation at the same time.

目 次

第 1章

緒　　言

1.1　研究の背景

近年の計算機システム全般における性能向上は，計算科学分野に対して大きな発展をもたらせた．特に数値流体力学 (CFD:Computational Fluid Dynamics) 等のシミュレーション系分野における研究の進展は著しい．自然現象をより正確にシミュレートするために必要な計算性能と容量は並列計算機システムを用いることで容易に得ることが可能となり，自然現象シミュレーションは高精度でかつ実用時間内に実行可能となった．その結果，シミュレーションの技術や成果は広く用いられ，科学・工学等における現象解明の応用技術として確立し，一般社会に広く貢献している．

1.1.1　並列計算

並列計算は，並列計算機システムを構成するハードウェアと並列プログラミングのためのソフトウェア技術から成り立つ．次のような理由から高性能な並列計算機が多く存在し，プログラミングスタイルの標準化が進んだことから CFD シミュレーション等の科学技術計算において並列計算の重要性が一層増した．

1. 高性能並列計算機システム
   (a) 高性能な計算処理要素を多数用いることが可能
   (b) メモリシステムの大容量化が容易に可能

(c) 通信処理要素の高速化あるいは低価格化

2. プログラムコードの汎用化

(a) 並列ソフトウェア記述方法の標準化

(b) 標準化に伴うプログラム実行環境の非依存性

近年の高性能計算機システムは大きく分けて，計算と通信の 2 つの要素から成る．計算処理要素は，計算処理とメモリから成り，通信処理要素は，I/O 処理やネットワーク技術から成る．また，計算機システムを実際に用いるソフトウェア技術が必要である．

計算処理要素 (PE:Processing Element) には，スカラ型とベクトル型のシステムがある．CFD 分野では，ベクトル型計算機が良く用いられている．ベクトル計算機は，パイプライン処理 (ベクトル処理) と広いメモリ帯域技術を用い，配列データ (ベクトルデータ) の処理を非常に高速な処理が可能である．CFD のシミュレーションではベクトル計算処理に適したアルゴリズムを用いることにより，ベクトル型計算機が非常に有効に用いられている．また，近年スカラ型計算機も，ベクトル型計算機の計算システムの特性を取り込み，動作周波数の高速化というアプローチによって計算性能が著しく向上し，メモリアクセスの高速化も進んでいる．ベクトル型計算機もスカラ型計算機の動作周波数の高速化というアプローチを取り込み，一般に高価で巨大なシステムであるベクトル型計算機システムが小型で比較的安価な計算機システムに成りつつある．また，1 つのメモリ空間を多数の PE で共有して，1 つの計算処理要素として扱える，共有メモリ型並列計算機システムがある．後述するが，自動並列コンパイラや指示行型並列言語などによって比較的容易にプログラミングが可能であり，内部 PE が比較的少数の場合，性能が得やすい計算機システムである．しかし，内部 PE が多くなると計算処理性能が得難くなるため，現状では内部 PE は，4 $\sim$ 16 程度の共有メモリ並列システムが主流である．

通信処理要素は，分散メモリ型並列計算機システムの根幹である．近年の高性能計算機システムは並列技術 (ハードウェア，ソフトウェア等) 無しにはあり得ない．I/O やネットワーク技術は，計算要素の高動作周波数化や広帯域化に伴い高速処理が進み，専用並列計算機技術だけではなく，コモディティな製品も高性能化

が進んでいる．そのため，PC/WS クラスタといったコモディティな計算要素をコモディティな通信要素を用いた接続した並列計算機システムが広く普及している．そして，市販 PC やスイッチハブを用いた並列計算システムを個々のユーザーが容易に作成でき，並列計算・並列プログラムのユーザー底辺を引き上げている．そして，それら小規模な並列システムは，対費用効果の高いシミュレーションが可能である．ベクトル型計算機を並列接続した並列・ベクトル型計算機システムでは，ベクトル型計算機の高速演算性能と高価で超高速なクロスバネットワークを有したシステムが多く，スケーラビリティの高い並列計算性能を有し，超大規模問題シミュレーション (全地球環境シミュレーション等) に威力を発揮する．また，前述の共有メモリ並列計算機システムをネットワークで接続した共有・分散システムも数多く存在する．

並列計算機システムのプログラミングは，並列化コンパイラ等によるものと既存プログラム言語と外部処理ライブラリを用いたものが存在する．並列化コンパイラを用いるものは，自動並列コンパイラと指示行型並列言語に分けることが出来る．また，外部処理ライブラリには，データ通信を行なうものとデータの依存関係処理を行なうものがある．

自動並列化コンパイラは，コンパイラ (あるいはプリプロセッサ) が従来の逐次プログラムコードを解釈して，自動的に並列化された実行プログラムを生成するものである．ただし，並列処理部分はプログラマに対して陰に隠れて明確に見えない．また，分散メモリ型並列システムにはほとんど実装されていない．また，自動並列化コンパイラは基本的に単純に並列化可能なアルゴリズムしか並列化できないため，並列化に対する自由度は小さい．指示行型並列言語は，本来のプログラミング言語には解釈されない指示行をプログラムコード内に付加することで，プログラマが並列化を明示的する言語 (HPF:High Performance Fortran，OpenMP 等) である．指示行型並列言語は，プログラムの並列化自体はプログラマが行なうため多少複雑な並列化が可能である．また，プログラミングも比較的容易であり，並列化による計算性能も比較的良い傾向を示す．ただし，両方のプログラミングにおいて，並列処理部分が陽に見えないため，プログラムコードのデバックが困難である．

外部処理ライブラリを用いるものは，分散メモリ型並列計算機システムで用いられるデータ通信ライブラリ (MPI:Message Passing Interface，PVM:Parallel Vertual Machine 等) と共有メモリ型並列計算機システムで用いられるデータ依存関係処理ライブラリ (POSIX Thread，ISO Thread 等) である．外部処理ライブラリを用いるものは，既存プログラム言語から機能を呼び出す方法である．既存プログラムの文法と文法や使用法に変化がないため，あるプログラミング言語を習熟している場合，容易に用いることが可能である．どちらの場合も並列処理に必要な処理を詳細に記述できるため，より細やかな並列処理の記述が可能であり，プログラミングの自由度は非常に高い．また，プログラマが通信を陽に記述するため，プログラムコードのデバックは比較的容易である．しかし，習熟にはある程度の時間を要し，自動並列コンパイラや指示行型並列言語ほど簡単ではない．

近年の並列プログラミングは，並列言語仕様の標準化が進んでいるため，用いるハードウェアにほとんど依存せず，実行環境を意識する必要がない．データ通信ライブラリ (MPI や PVM) を用いたプログラムコードは，ほとんどの分散メモリ型並列計算機で実行可能であり，データ依存関係処理ライブラリ (Thread) を用いたプログラムコードは，ほとんどの共有メモリ型並列計算機で実行可能である．指示行型並列言語も同様であり，HPF も OpenMP もほとんどの場合実行可能であろう．ただし，自動並列コンパイラはコンパイラ次第であり，必ず逐次プログラムコードが並列で実行されるとは限らない．また，分散・共有並列型並列計算機システムでは，共有メモリ部分を分散メモリ的に扱って，全てを MPI 等の分散メモリプログラミングを行なう場合が多いが，共有メモリ部分を OpenMP や Thread で記述し，分散メモリ部分を MPI や PVM を用いて記述する分散・共有型のプログラミングを行なう場合があり，共有メモリシステムを 1 つの計算要素として捉えることで，より高性能な並列プログラムが可能である．

1.1.2 並列 CFD シミュレーション

計算機を用いた CFD シミュレーションは，1950 年代前半の手回し計算機による 2 次元円柱周りの流れ計算から始まり，シミュレーション技術と計算機技術は相互関係的に進歩してきた．特にベクトル型計算機の出現が CFD シミュレーションの発展に大きな影響を及ぼした．ベクトル型計算機の高速化・大容量化に伴い，

CFD シミュレーションの高速化・高精度化が可能となった．また，近年では上述のように高性能計算機システムは並列計算機システムが主流であり，PC/WS クラスタシステムのようにユーザー主導の並列計算機システムや並列プログラミングの標準化などによって並列 CFD シミュレーションの一般化が進んでいる．

CFD の数値解法には，大きく分けて有限差分法，有限体積法そして有限要素法がある．有限差分法の歴史は古く，微分方程式を差分方程式に置き換えることで容易に計算可能であり，計算に用いる格子点数を増やすことで高次精度化も容易である．一方，有限要素法の歴史は新しく，固体力学の数値計算法である変分法や Galerkin 法が元である．流れへの応用研究は 1970 年代前半から行なわれている．有限要素法は，要素ごとの近似計算で全体の計算を進める．有限差分法は有限要素法に比べて計算量が少なく済むため，総計算時間が短く，メモリ使用量も少なく済む．また，有限差分法は差分方程式をそのままプログラミングするだけであるため，プログラムの実装が容易である．

差分法は構造格子 (structured grid) を用い，有限要素法は任意分割のメッシュ (mesh,element) を用いる．有限体積法は構造格子でもメッシュどちらでも良い．境界適合格子やメッシュの生成に必要な時間は非常に長い．また，境界適合格子やメッシュの良し悪しは計算精度に大きく影響するため注意深く生成する必要がある．一般に境界適合格子は滑らかに変化して直交性がよく必要な部分で細かいものが良い．メッシュ生成でもメッシュの大きさが徐々に変化し，形状は正立法形や正 4 面体等の各ノードの長さが同じであるほど良い．複雑形状での境界適合格子の生成は膨大な時間を要するか生成不可能な場合も多い．この場合，細かな直交等間隔格子を用いることで，複雑形状も容易に計算可能であり，実用上問題の無い精度が得られている．

科学技術シミュレーションに陰的解法を用いる場合，シミュレーションを解くことは，大規模な連立 1 次方程式を解くことに帰着する．非定常流を対象とした CFD シミュレーションは， Poisson 方程式 (楕円型方程式) や移流拡散方程式 (放物/双曲型方程式) から得られる巨大な連立方程式を時間ステップ毎に解く必要がある．楕円型方程式の多くは，並列化はもとよりベクトル化さえ出来ないものがほとんどであった．しかし，ベクトル計算機の発達により方程式解法アルゴリズムのベクトル化に関する研究が進み，CFD シミュレーションで良く用いる方程式

解法の多くのアルゴリズムは，ベクトル化可能となった．アルゴリズムのベクトル化は，データ依存性の解決である．逐次 CFD シミュレーションプログラムコードの並列化にもこの考え方が有用である．

CFD シミュレーションの並列計算は，領域分割法を用いたアプローチが一般的である．領域分割法は全体領域を複数の小領域に分割して全体領域を計算する方法である．CFD の分野では，並列計算のみならず逐次処理におけるマルチブロック格子やオーバーセット格子を用いる場合にも用いられる．また，領域分割法の特性を利用した方法の 1 つに解を得るために必要な全計算量を抑える技術もある．並列計算に領域分割法を用いる場合にも様々な領域分割手法があり，アルゴリズム毎に長所・短所がある．各領域分割手法の性能は，CFD ソルバー (あるいは連立方程式解法) の特性とも複雑に関係するため，CFD ソルバー毎に検討が必要である．実際の有限差分法での領域分割の分割パターンを考慮した場合にも 用いる PE 数が増加するほど，分割パターンは増える．一般には分割パターンの決定はデータ通信量を基準に行なうことが多い．しかし，通信要素性能が十分高性能である場合，その方針は間違いである．また，これらの問題には方程式解法に反復法を用いた場合，収束性能の問題も関わる非常に複雑な選択となる．

有限差分法では，連立方程式解法に緩和法 (反復法) が良く用いられるため，領域分割法には重複領域を設定するものが多い．この方法は本質的に逐次処理の計算手順を踏まえた方法である．反復法はデータ通信量が比較的多く通信要素性能が高い場合，大きな性能低下の原因とはならないが，PC/WS クラスタ等の通信性能が低調な並列計算機では，台数効果が出にくい場合が多い．また，有限要素法では，共役勾配法系の連立方程式解法がよく用いられており，領域分割法も領域分割を強く意識した領域分割型有限要素法が存在する．また，収束性の影響や計算性能を考慮した場合，どの方法が良い方法とはいえない．CFD シミュレーションに用いる連立方程式解法は，高い収束性能が求められる．しかし，一般に収束性能が高い解法は，アルゴリズムが複雑で計算量が非常に多い．そのため，計算処理時間で考えた場合，多少の収束性能を犠牲にしても，アルゴリズムが単純で計算量が少ない解法を用いた方が，短い計算処理時間になることが少なくない．

以上のように CFD シミュレーションは，並列計算機システムの成熟や使用できる環境の広がり，シミュレーションプログラムの並列化に対する基礎的研究が進め

られ，多くの成果が得られている．しかし，並列シミュレーションを高性能に行なう場合，ユーザーは並列計算機システムの特性や並列プログラミングをかなり熟知している必要があるが，全てを知ることは難しい．このため，CFD シミュレーションにおける高性能化の方法論を示す必要がある．

1.2 研究の目的

以上のように CFD において，高速計算，大規模計算のために並列計算の必要性は極めて高い．特に，航空・宇宙分野での CFD シミュレーションでは，多数の並列化に関する研究事例が存在し，実際の解析に大きな効果を上げている．また，近年ではユーザーフレンドリな並列計算機システムとして PC/WS クラスタ等が普及し，大規模，高速計算の要求は，さらに高くなるものと考えられる．しかし，CFD の並列計算は研究途上であり，効率的な並列化手法に対する十分な検討は行なわれていない．CFD シミュレーションの並列化は，計算時間の高速化の比重が高いが，高効率と拙速性を含めた効果の検 討は行なわれていない．本研究では，有限差分法による非圧縮粘性流体計算の並列性能の向上における様々な問題点について議論し，拙速性を失わず，高効率な並列化が行なえる方法を提案し，実際に様々な並列計算機システムに実装して評価を行なう．

高効率な並列計算を行なう幾つかの手法について検討する．第 1 に，CFD 計算を並列化する場合に用いられる領域分割法による分割パターンの影響による並列計算性能の変化について検討し，最も並列計算性能が高くなる分割パターンを実際に流体計算を行なわず，選択する方法を提案する．その結果，分割パターンを最適に保つことで，並列化による付加効果で性能が損なわれる部分の性能低下を最低限に留めることが可能となる．第 2 に，並列 CFD 計算では，必ず隣接 PE との通信処理が発生する．しかし，CFD 計算における通信処理は，容易に通信隠蔽処理が可能である．通信隠蔽処理の方法として，非同期通信-計算重複法を用いた並列 CFD 計算を提案する．この手法を用いることで，並列計算における最も大きな性能欠損部分である通信処理での性能低下を効果的に防ぐことが可能である．第 3 に，非圧縮粘性流体計算は，大規模な連立方程式を解くことに帰着する．CFD 計算では，SOR (Successive Over Relaxation) 法が用いられることが多い．本研究

では，スカラー型計算機向けで通信隠蔽処理にも適した Multi Colored Line SOR (MCLSOR) 法を提案する．この方法を用いることで通信隠蔽処理において，高い収束性を持ち，並列計算による収束性の低下を防ぎ，プログラムの拙速性を維持することが可能となる．そのため，既存プログラムコードの並列化や新規プログラムを作成する際にも効率の良く通信隠蔽処理を用いることが可能である．

1.3 研究の概要

本研究では，高効率な CFD 計算を行なう 3 つの手法について検討し，その手法を適用した CFD 計算における並列性能を明らかにする．CFD 計算において効率よく適用するためには，個々の検討では不十分である．これらの方法は，個々に検討が行なわれていたが，3 つの手法によって総合的に性能を向上させる試みは行なわれていない．本研究では，個々の手法の有効性を示すため，ベンチマーク問題として Poisson 方程式ソルバーに対して適用し，個々の手法自身の有効性を議論する．そして，3 つの手法を全て用いた場合の総合的な性能向上を示し，有効性を確認する．そして，MAC 法を用いた非圧縮粘性流体のシミュレーションプログラムに対して上述の 3 つの方法を適用する．本研究で用いる各方法は，既存シミュレーションコードやアルゴリズムから大きな変更なしに実現できる．そのため，CFD 分野の研究者が容易に実現可能な方法であるため，並列計算の効率向上に非常に役立つと考えられる．

検討する 3 つの並列計算性能を向上させる手法は，それぞれ個々を詳細に検討することが必要となる．領域分割法において最適な分割パターンの選択は，経験則での分割パターン選択より多くの場合並列化性能を高めることが可能である．分割パターンによる性能への影響は，ベクトル長の変化からベクトル型計算機を用いる場合に大きく影響すると思われるが，近年のスカラー型計算機は，ベクトル型計算機の特性を踏まえている場合が多いため，分割パターンは計算性能に直結する．また，分割パターンにおける分割面の格子点数による通信量が変化するため，通信性能にも大きな影響がある．分割パターンの選択は，計算性能と通信性能の両方の検討が必要である．しかし，両者を踏まえた検討は行なわれていない．並列計算における分割パターンによる性能変化への影響は，ネットワーク性能が

低い場合を仮定して通信量が最小になる場合の直交格子型の Poisson 問題に対して Crandall ら [1] によって検討された．しかし，近年の商用並列計算機においてこの仮定は，適切ではなくなっている．最大性能は，通信量最小である必要はない．しかし，PC Cluster のようなコモディティなネットワーク機器を用いる場合に有効な結果が得られている．また，スカラー型計算機における計算性能に対する検討は，Wijngaart ら [2] によって理論性能と簡易型キャッシュモデルを用いる方法が提案されたが，性能予測を正確に行なうことは非常に困難であるとされた．本研究では，理論性能だけから性能モデルを作成するのではなく，計算性能の実測値を用いて，計算と通信の両方性能を評価するモデルを提案する．そして，最も計算性能が高くなる分割パターンの検討を行ない，最も並列計算性能が高くなる分割パターンを選択する方法を提案する．

並列 CFD 計算では，必ず隣接 Processing Element (PE) との通信処理が発生する．しかし，CFD 計算における通信処理は，容易に通信隠蔽処理が可能である．通信隠蔽処理は，通信処理 (付加的な処理) を計算処理 (主要な処理) を同時に行なうことで通信処理を覆い隠すことである．通信隠蔽処理は大きく分けて，パイプライン処理と非同期通信計算重複処理の 2 つがある．パイプライン処理は，CFD 計算における通信隠蔽の研究によく用いられる．非同期通信計算重複処理の基礎研究として Quinn ら [3] は，2 次元モデルについて自動並列コンパイラ技術を用いた有効性や性能限界を理論的に示した．さらに，2 次元モデルについて Fink ら [4] や 田中ら [5] は，SMP クラスタを用いた研究を行なっている．しかし，3 次元モデルにおける有効性や問題点は，明らかにされていない．また，非同期通信計算重複処理は，CFD 計算にはほとんど用いられていない．本研究では，並列 CFD 計算における非同期通信-計算重複法を用いた実装方法を提案し，その並列計算での有効性を確認する．

また，非圧縮粘性流体計算は，大規模な連立方程式を解くために CFD 計算では，SOR 法等の反復解法が用いられる．SOR 法は，プログラミングが容易であり，使用するメモリ量も少ない．そして，非定常の CFD のように直前の解を初期値として，解ベクトルや右辺ベクトルを僅かに修正しながら何回も解く場合に有利である．SOR 法の並列化の研究は多く存在し，Xie ら [6] は，通信隠蔽をしない場合の改良 PSOR 法や Multi-color SOR 法の並列性能について議論している．方

程式解法に SOR 法を用い，通信隠蔽処理を行なう場合，SSOR 法を用いたパイプライン処理が用いられる場合が多い．NAS Parallel Benchmark [7] では，圧縮性流れを擬似的に解くベンチマークプログラムに実装されている．しかし，パイプラン処理は，通信処理が煩雑であり，通信性能が低い並列計算機の場合や Poisson 方程式ソルバーのような計算粒度が小さな計算では，十分な並列計算性能が得られない．そのため，通信隠蔽処理向けの MCLSOR 法を提案する．SOR 法の改良研究は，比較的古くから存在する．しかし，多くはベクトル型計算機向けの研究であり，通信隠蔽を踏まえた並列化効率や計算効率向上の研究はほとんどなされていない．本研究では，スカラー型計算機における MCLSOR 法の並列化において，計算順序の入れ替え (色付け) による通信隠蔽を踏まえた MCLSOR 法の研究を行い，既存の Multi-color SOR 法やパイプライン処理による SSOR 法と MCLSOR 法の収束性や計算性能について議論し，MCLSOR 法の有効性を明らかにする．

上記の手法を用いて，MAC 法を用いる非圧縮粘性流体シミュレーションを行なった．本研究では，3 次元計算モデル (管内流れ問題，キャビティー問題と円柱周り周りの流れ) を対象とした．全ての CFD 計算において一般座標変換を行い，任意形状の流れ場の計算が取り扱える．また，円柱周りの流れでは，レイノルズ数の高い流れを対象とするため，Kawamura-Kuwahara スキーム [8] を用いた．非圧縮粘性流体の並列シミュレーションの総合的な性能を検討し，個々の方法の有効性を含めて，総合的な有効性を議論した．

第 2章

CFDの基礎方程式

2.1 MAC 法

本研究では，有限差分法による代表的な CFD の計算方法の 1 つである Maker And Cell (MAC) 法を用いた．元来の MAC 法は，1965 年に Los Alamos 研究所の Harlow と Welch によって開発された自由表面を含んだ流れを解析する方法である [9]．仮想的なマーカー粒子の動きを計算することでセル内のマーカー粒子の存在する部分を流体要素として扱いうことで自由表面を表現した．また，連続の式に圧力を上手く取り入れ，安定した計算が進められる．この特徴は非常に有効であり，自由境界を扱わない場合マーカー粒子を用いず，この特徴を取り入れた計算例が多数存在する．このため，MAC 法という名称は，必ずしもマーカー粒子を用いる方法を示さず，上述の特徴を取り入れた CFD 計算法の総称として用いられている．

一般に MAC 法は，非定常流体計算を行なうための圧力と速度の分離解法である．次時間ステップの圧力と速度は，以下のように導かれる．

1. 連続の式と Navier-Stokes (NS) 方程式から導かれた Poisson 方程式を解くことで圧力を求める．

2. 既値の速度と方程式から得られた圧力を NS 方程式に代入して，速度を求める．

このステップを繰り返すことで，時間ステップ毎の流体計算を行なう．

また，MAC 法の原論文では，計算格子に Staggered Grid を用いて計算を行なった．Staggered Grid とは，速度と圧力の評価点を同一の格子点上で評価せず，それぞれずれた点で評価する格子系である．Staggered Grid は，計算を安定的に進めることが可能であり[1]，1 つの単位体積毎に連続の式が評価できる．また，圧力差が速度を決めるという NS 方程式を自然に表現できる．しかし，Staggered Grid は，速度や圧力を格子点上で評価しないため，境界条件等の条件設定は非常に繁雑であり，一般座標を用いる場合にも不都合な点が多い．そのため，本研究では Regular Grid を用いた．Regular Grid は，Staggered Grid と異なり 1 つの格子点で全ての物理量を定義する格子系である．全ての物理用を格子点表で定義するため，境界条件等の初期条件を容易に与えることが出来る有効な方法である．

2.2 MAC 法の基礎方程式

MAC 法による CFD 計算に用いる基礎方程式を導出する．

非圧縮粘性流体解析の支配方程式は，速度ベクトルを $\mathbf{V}$，圧力を p として，ベクトル形式で表記すると質量保存をあらわす連続の式

$$\nabla \cdot \mathbf{V} = 0 \tag{2.1}$$

と運動量保存をあらわす NS 方程式

$$\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \nabla)\mathbf{V} = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{V} \tag{2.2}$$

$$\nabla = \mathbf{i}\frac{\partial}{\partial \mathbf{x}} + \mathbf{j}\frac{\partial}{\partial \mathbf{y}} + \mathbf{k}\frac{\partial}{\partial \mathbf{z}}$$

ただし，i, j, k は各軸方向の単位ベクトル

(Re は，$(LU)/\nu$ で定義する無次元量のレイノルズ数である．ただし，L は代表長さ，U は代表速度，ν は動粘性係数) となる．式 (2.2) の左辺第 2 項が非線形であること，速度 $\mathbf{V}$ が時間発展であるのに対し圧力 p は時間発展ではないなどによって，安定な数値解を得るために様々なスキームが開発されている．

表記を簡略化するために式 (2.2) を

[1]圧力解の振動を防ぐ効果が大きい

$$\frac{\partial \mathbf{V}}{\partial t} = f(\mathbf{V}) - \nabla p \tag{2.3}$$

$$f(\mathbf{V}) = \frac{1}{Re}\nabla^2 \mathbf{V} - (\mathbf{V}\cdot\nabla)\mathbf{V}$$

とする．式 (2.3) を時間に関して前進差分し，さらに両辺の発散をとると，

$$\frac{\nabla\cdot\mathbf{V}^{n+1} - \nabla\cdot\mathbf{V}^{n}}{\Delta t} = \nabla\cdot f(\mathbf{V}^n) - \nabla^2 p^{n+1} \tag{2.4}$$

が得られる．さらに時刻 $n+1$ では連続の式を満たす必要があるため，$\nabla\cdot\mathbf{V}^{n+1} = 0$ である必要がある．よって，

$$\nabla^2 p^{n+1} = \frac{\nabla\cdot\mathbf{V}^{n}}{\Delta t} + \nabla\cdot f(\mathbf{V}^n) \tag{2.5}$$

のような圧力に関する Poisson 方程式が得られる．

次に，式 (2.3) にオイラー陽解法を適用し，式 (2.5) から得た p^{n+1} を代入すると速度の時間発展をあらわす次式が得られる．

$$\mathbf{V}^{n+1} = \mathbf{V}^n + (f(\mathbf{V}^n) - \nabla p^{n+1})\cdot\Delta t \tag{2.6}$$

また，時間進行の計算精度の向上に対応するために，式 (2.3) の移流項に半陰解法を用いて線形化して，粘性項に陰解法を適用し，p^{n+1} を代入すると

$$\mathbf{V}^{n+1} = \mathbf{V}^n + (\frac{1}{Re}\nabla^2\mathbf{V}^{n+1} - (\mathbf{V}^n\cdot\nabla)\mathbf{V}^{n+1} - \nabla p^{n+1})\cdot\Delta t$$

が得られる．ただし，上の式 (2.7) は，連立方程式を解く必要がある．

式 (2.5)，(2.6) あるいは (2.7) が MAC 法の基礎方程式である [10, 11, 12].

2.3 一般座標変換

流体計算を行なう場合，計算領域が必ず矩形であるとは限らないため，直方体格子では物体を正確に表現できない場合が多い．物体周りの計算を正確に行なう

ため，物体を取り巻くように計算格子を生成する必要がある (図 2.1)．また，ある流体現象が領域内部の特定部分で急激に変化する場合，解の精度を高く保つために，その特定部分付近の格子を細かくとる必要がある．例えば，領域境界付近での微分方程式の解が急激に変化する場合，数値解の精度を高く保つために必要である．一方で，流体現象が大きく変化しない広い領域も存在する．これらの領域を同一密度での格子分割を行なうことは効率的ではない．また，一般座標系では物体表面等の境界を忠実に再現可能であるため，物体形状を忠実に計算に反映でき，数値計算における境界条件の取り扱いも容易になる [13](図 2.2).

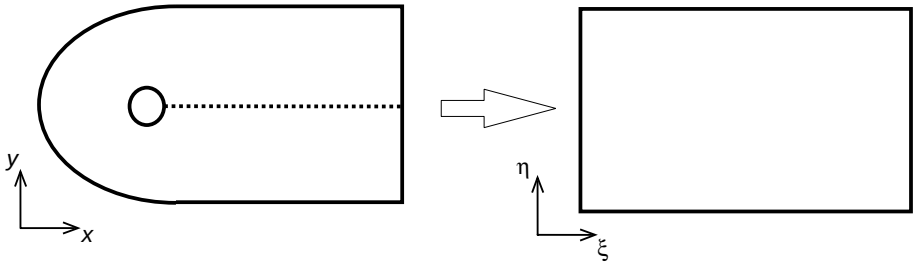

図 2.1: The general coordinated grid system

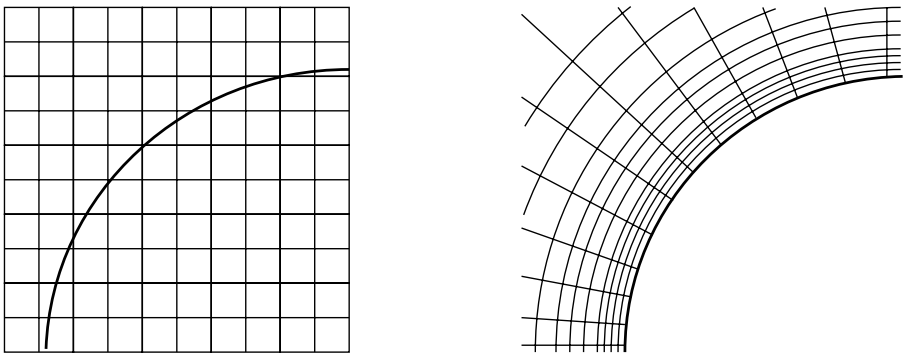

図 2.2: The boundary fitted coordinated grid system

3 次元座標変換は一般に

$$\begin{cases} \xi = \xi(x, y, z) \\ \eta = \eta(x, y, z) \\ \zeta = \zeta(x, y, z) \end{cases} \tag{2.7}$$

あるいは，

$$\begin{cases} x = x(\xi, \eta, \zeta) \\ y = y(\xi, \eta, \zeta) \\ z = z(\xi, \eta, \zeta) \end{cases} \tag{2.8}$$

で与えられる．ただし，(x,y,z) は，デカルト座標系であり，(ξ, η, ζ) は，一般座標系である．

式 (2.8) における物理領域の微係数を変換領域での微係数で表すと

$$\begin{cases} f_x = \xi_x f_\xi + \eta_x f_\eta + \zeta_x f_\zeta \\ f_y = \xi_y f_\xi + \eta_y f_\eta + \zeta_y f_\zeta \\ f_z = \xi_z f_\xi + \eta_z f_\eta + \zeta_z f_\zeta \end{cases} \tag{2.9}$$

が成り立つ．

また，式 (2.7)，(2.8) から

$$\begin{bmatrix} d\xi \\ d\eta \\ d\zeta \end{bmatrix} = \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \tag{2.10}$$

$$\begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \\ d\zeta \end{bmatrix} \tag{2.11}$$

が得られる．よって，

$$\begin{aligned} \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} &= \begin{bmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{bmatrix}^{-1} \\ &= \frac{1}{J} \begin{bmatrix} y_\eta z_\zeta - y_\zeta z_\eta & -(x_\eta z_\zeta - x_\zeta z_\eta) & x_\eta y_\zeta - x_\zeta y_\eta \\ -(y_\xi z_\zeta - y_\zeta z_\xi) & x_\xi z_\zeta - x_\zeta z_\xi & -(x_\xi y_\zeta - x_\zeta y_\xi) \\ y_\xi z_\eta - y_\eta z_\xi & -(x_\xi y_\eta - x_\eta z_\xi) & x_\xi y_\eta - x_\eta y_\xi \end{bmatrix} \end{aligned} \tag{2.12}$$

ただし，

$$J = \begin{vmatrix} x_\xi & x_\eta & z_\zeta \\ y_\xi & y_\eta & z_\zeta \\ z_\xi & z_\eta & z_\zeta \end{vmatrix}$$
$$= x_\xi y_\eta z_\zeta + x_\eta y_\zeta z_\xi + x_\zeta y_\xi z_\eta - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta - x_\zeta y_\eta z_\xi \tag{2.13}$$

が成り立つため，

$$\begin{cases} \xi_x = (y_\eta z_\zeta - y_\zeta z_\eta)/J \\ \eta_x = (y_\zeta z_\xi - y_\xi z_\zeta)/J \\ \zeta_x = (y_\xi z_\eta - y_\eta z_\xi)/J \end{cases} \tag{2.14}$$

$$\begin{cases} \xi_y = (x_\zeta z_\eta - x_\eta z_\zeta)/J \\ \eta_y = (x_\xi z_\zeta - x_\zeta z_\xi)/J \\ \zeta_y = (x_\eta z_\xi - x_\xi z_\eta)/J \end{cases} \tag{2.15}$$

$$\begin{cases} \xi_z = (x_\eta y_\zeta - x_\zeta y_\eta)/J \\ \eta_z = (x_\zeta y_\xi - x_\xi y_\zeta)/J \\ \zeta_z = (x_\xi y_\eta - x_\eta y_\xi)/J \end{cases} \tag{2.16}$$

が得られる．

よって，1 階微分は，

$$\begin{aligned} f_x &= (y_\eta z_\xi - y_\zeta z_\eta) f_\xi/J + (y_\zeta z_\xi - x_\xi z_\zeta) f_\eta/J + (y_\xi y_\eta - y_\eta z_\xi) f_\zeta/J \\ f_y &= (x_\zeta z_\eta - x_\eta z_\zeta) f_\xi/J + (x_\xi z_\zeta - x_\zeta z_\xi) f_\eta/J + (x_\eta z_\xi - x_\xi z_\eta) f_\zeta/J \\ f_z &= (x_\eta y_\zeta - x_\zeta y_\eta) f_\xi/J + (x_\zeta y_\xi - x_\xi y_\zeta) f_\eta/J + (x_\xi y_\eta - x_\eta y_\xi) f_\zeta/J \end{aligned} \tag{2.17}$$

以上の値を用いて，MAC 法の基礎方程式は，次のように変換される．NS 方程式は，

$$u_t + Uu_\xi + Vu_\eta + Wu_\zeta = -(\xi_x p_\xi + \eta_x p_\eta + \zeta_x p_\zeta) + \frac{1}{Re}\overline{\Delta} u \tag{2.18}$$

$$v_t + Uv_\xi + Vv_\eta + Wv_\zeta = -(\xi_y p_\xi + \eta_y p_\eta + \zeta_y p_\zeta) + \frac{1}{Re}\overline{\Delta} v \tag{2.19}$$

$$w_t + Uw_\xi + Vw_\eta + Ww_\zeta = -(\xi_w p_\xi + \eta_w p_\eta + \zeta_w p_\zeta) + \frac{1}{Re}\overline{\Delta} w \tag{2.20}$$

となり，Poisson 方程式は

$$\begin{aligned}\overline{\Delta}p = &-(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta)^2 - (\xi_y v_\xi + \eta_y v_\eta + \zeta_y v_\zeta)^2 - (\xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta)^2 \\ &- 2\Big((\xi_y u_\xi + \eta_y u_\eta + \zeta_y u_\zeta)(\xi_x v_\xi + \eta_x v_\eta + \zeta_x v_\zeta) \\ &+ (\xi_z v_\xi + \eta_z v_\eta + \zeta_z v_\zeta)(\xi_y w_\xi + \eta_y w_\eta + \zeta_y w_\zeta) \\ &+ (\xi_x w_\xi + \eta_x w_\eta + \zeta_x w_\zeta)(\xi_z u_\xi + \eta_z u_\eta + \zeta_z u_\zeta)\Big) \\ &+ \frac{1}{\Delta t}(\xi_x u_\xi + \eta_x u_\eta + \zeta_x u_\zeta + \xi_y v_\xi + \eta_y v_\eta \\ &\qquad + \zeta_y v_\zeta + \xi_z w_\xi + \eta_z w_\eta + \zeta_z w_\zeta) \end{aligned} \tag{2.21}$$

となる．ただし，

$$U = u\xi_x + v\xi_y + w\xi_z \tag{2.22}$$

$$V = u\eta_x + v\eta_y + w\eta_z \tag{2.23}$$

$$W = u\zeta_x + v\zeta_y + w\zeta_z \tag{2.24}$$

$$\begin{aligned}\overline{\Delta}f = &C_1 f_{\xi\xi} + C_2 f_{\eta\eta} + C_3 f_{\zeta\zeta} \\ &+ 2(C_4 f_{\xi\eta} + C_5 f_{\eta\zeta} + C_6 f_{\zeta\xi}) \\ &+ C_7 f_\xi + C_8 f_\eta + C_9 f_\zeta \end{aligned} \tag{2.25}$$

$$C_1 = \xi_x^2 + \xi_y^2 + \xi_z^2 \tag{2.26}$$

$$C_2 = \eta_x^2 + \eta_y^2 + \eta_z^2 \tag{2.27}$$

$$C_3 = \zeta_x^2 + \zeta_y^2 + \zeta_z^2 \tag{2.28}$$

$$C_4 = \xi_x\eta_x + \xi_y\eta_y + \xi_z\eta_z \tag{2.29}$$

$$C_5 = \eta_x\zeta_x + \eta_y\zeta_y + \eta_z\zeta_z \tag{2.30}$$

$$C_6 = \zeta_x\xi_x + \zeta_y\xi_y + \zeta_z\xi_z \tag{2.31}$$

$$C_7 = \xi_{xx} + \xi_{yy} + \xi_{zz} \tag{2.32}$$

$$C_8 = \eta_{xx} + \eta_{yy} + \eta_{zz} \tag{2.33}$$

$$C_9 = \zeta_{xx} + \zeta_{yy} + \zeta_{zz} \tag{2.34}$$

また，$\xi_{xx}, \xi_{yy}, \xi_{zz}, \eta_{xx}, \eta_{yy}, \eta_{zz}, \zeta_{xx}, \zeta_{yy}, \zeta_{zz}$ の 2 階微分項は

$$\begin{aligned}\xi_{xx}+\xi_{yy}+\xi_{zz} &= \xi_x(\xi_x)_\xi+\eta_x(\xi_x)_\eta+\zeta_x(\xi_x)_\zeta+\xi_y(\xi_y)_\xi+\eta_y(\xi_y)_\eta \\ &\quad +\zeta_y(\xi_y)_\zeta+\xi_z(\xi_z)_\xi+\eta_z(\xi_z)_\eta+\zeta_z(\xi_z)_\zeta \qquad (2.35)\\ \eta_{xx}+\eta_{yy}+\eta_{zz} &= \xi_x(\eta_x)_\xi+\eta_x(\eta_x)_\eta+\zeta_x(\eta_x)_\zeta+\xi_y(\eta_y)_\xi+\eta_y(\eta_y)_\eta \\ &\quad +\zeta_y(\eta_y)_\zeta+\xi_z(\eta_z)_\xi+\eta_z(\eta_z)_\eta+\zeta_z(\eta_z)_\zeta \qquad (2.36)\\ \zeta_{xx}+\zeta_{yy}+\zeta_{zz} &= \xi_x(\zeta_x)_\xi+\eta_x(\zeta_x)_\eta+\zeta_x(\zeta_x)_\zeta+\xi_y(\zeta_y)_\xi+\eta_y(\zeta_y)_\eta \\ &\quad +\zeta_y(\zeta_y)_\zeta+\xi_z(\zeta_z)_\xi+\eta_z(\zeta_z)_\eta+\zeta_z(\zeta_z)_\zeta \qquad (2.37)\end{aligned}$$

となる．

$$(\xi_x)_\xi=\Big[(y_{\xi\eta}z_\zeta+y_\eta z_{\xi\zeta}-y_{\xi\zeta}z_\eta-y_\zeta z_{\xi\eta})J-(y_\eta z_\zeta-y_\zeta z_\eta)J_\xi\Big]/J^2 \qquad (2.38)$$

そして，$x_\xi, x_\eta, x_\zeta, x_{\xi\xi}, x_{\eta\eta}, x_{\zeta\zeta} x_{\xi\eta}, x_{\eta\zeta}, x_{\xi\zeta}$ は,

$$(x_\xi)_{i,j,k}=\frac{x_{i+1,j,k}-x_{i-1,j,k}}{2\Delta\xi} \qquad (2.39)$$

$$(x_\eta)_{i,j,k}=\frac{x_{i,j+1,k}-x_{i,j-1,k}}{2\Delta\eta} \qquad (2.40)$$

$$(x_\zeta)_{i,j,k}=\frac{x_{i,j,k+1}-x_{i,j,k-1}}{2\Delta\zeta} \qquad (2.41)$$

$$(x_{\xi\xi})_{i,j,k}=\frac{x_{i+1,j,k}-2x_{i,j,k}+x_{i-1,j,k}}{\Delta\xi^2} \qquad (2.42)$$

$$(x_{\eta\eta})_{i,j,k}=\frac{x_{i,j+1,k}-2x_{i,j,k}+x_{i,j-1,k}}{\Delta\eta^2} \qquad (2.43)$$

$$(x_{\zeta\zeta})_{i,j,k}=\frac{x_{i,j,k+1}-2x_{i,j,k}+x_{i,j,k-1}}{\Delta\zeta^2} \qquad (2.44)$$

$$(x_{\xi\eta})_{i,j,k}=\frac{x_{i+1,j+1,k}-x_{i-1,j+1,k}-x_{i+1,j-1,k}+x_{i-1,j-1,k}}{4\Delta\xi\Delta\eta} \qquad (2.45)$$

$$(x_{\eta\zeta})_{i,j,k}=\frac{x_{i,j+1,k+1}-x_{i,j-1,k+1}-x_{i,j+1,k-1}+x_{i,j-1,k-1}}{4\Delta\eta\Delta\zeta} \qquad (2.46)$$

$$(x_{\xi\zeta})_{i,j,k}=\frac{x_{i+1,j,k+1}-x_{i-1,j,k+1}-x_{i+1,j,k-1}+x_{i-1,j,k-1}}{4\Delta\xi\Delta\zeta} \qquad (2.47)$$

の関係を用いることで得ることが出来る．これらの計算を行なうことで，物理座標から一般座標への変換が行なえる．

2.4 離散化

式 (2.5)，(2.3) を一般座標変換した式 (2.18)，(2.19)，(2.20) と (2.21) を空間項で離散化する．式 (2.3) の移流項 (非線形項) を除く全ての空間の離散化は，2 次精度中心差分を用いた．例えば，一般座標系での関数 f の ξ 方向の離散化は，

$$(f_\xi)_{i,j,k} = \frac{f_{i+1,j,k} - f_{i-1,j,k}}{2\Delta\xi} \tag{2.48}$$

$$(f_{\xi\xi})_{i,j,k} = \frac{f_{i+1,j,k} - 2f_{i,j,k} + f_{i-1,j,k}}{\Delta\xi^2} \tag{2.49}$$

ただし，関数 f は，u, v, w あるいは p に対応する．

移流項 (非線形項) の離散化は 3 次精度風上差分を用いた．式 (2.2) の NS 方程式の右辺第 2 項の $1/Re$ が小さくなる (即ち，Re が大きくなる) と拡散効果が衰え，NS 方程式は局所的に急激に変化するようになる．この時，Δx を十二分に小さく取らなければ，解は移流項成分の影響で数値振動することになる．風上差分の適用は，高レイノルズ数流れを安定して計算するために必要である．

例えば，一般座標系での関数 f の ξ 方向に対する 3 次精度風上差分は，

$$\begin{aligned}
&(Ff_\xi)_{i,j,k} \\
&= \begin{cases}
F_{i,j,k}\dfrac{2f_{i+1,j,k} + 3f_{i,j,k} - 6f_{i-1,j,k} + f_{i-2,j,k}}{6\Delta\xi} & \text{for} \quad F_{i,j,k} \le 0 \\
F_{i,j,k}\dfrac{-f_{i+2,j,k} + 6f_{i+1,j,k} - 3f_{i,j,k} + 2f_{i-1,j,k}}{6\Delta\xi} & \text{for} \quad F_{i,j,k} > 0
\end{cases} \qquad (2.50) \\
&= F_{i,j,k}\frac{-f_{i+2,j,k} + 8(f_{i+1,j,k} - f_{i-1,j,k}) + f_{i-2,j,k}}{12\Delta\xi} \\
&\qquad + \frac{|F_{i,j,k}|}{12}\Delta\xi^3\frac{f_{i+2,j,k} - 4f_{i+1,j,k} + 6f_{i,j,k} - 4f_{i-1,j,k} + f_{i-1,j,k}}{\Delta\xi^4} \qquad (2.51)
\end{aligned}$$

ただし，関数 f は，u, v, w に対応し，F は，式 (2.22)，(2.23)，(2.24) に対応する．

式 (2.51) は，4 次精度中心差分と数値拡散項で構成されている．そのため，式

(2.51) を

$$(Ff_\xi)_{i,j,k} = F_{i,j,k}\frac{-f_{i+2,j,k}+8(f_{i+1,j,k}-f_{i-1,j,k})+f_{i-2,j,k}}{12\Delta\xi} + \alpha\frac{|F_{i,j,k}|}{12}\frac{f_{i+2,j,k}-4f_{i+1,j,k}+6f_{i,j,k}-4f_{i-1,j,k}+f_{i-1,j,k}}{\Delta\xi} \quad (2.52)$$

として，数値拡散項に重み α を加えることで，3 次精度風上差分法に変形を加え，α を定数または場所の関数として用いることで数値拡散を制御し，高レイノルズ数流れを安定して解く方法がある．$\alpha = 1$ の場合が通常の 3 次精度風上差分スキーム，$\alpha = 3$ の場合が Kawamura-Kuwahara(K-K) スキームとして知られている．

また，時間のステップ幅は，用いる解法に大きく依存する．陽解法は，解を得ることが容易であるが，数値的な安定性を得るためにあまり大きな時間ステップを取ることが出来ない．陰解法を用いる場合，時間ステップは，陽解法よりも大きく取ることが可能であるが，解を得るために連立方程式を解く必要があり，計算量が多くなる．CFD 計算を安定に解くにはクーラン数 $(= (u\Delta t)/(\Delta x_{min}))$ を考慮する必要がある．速度計算に陽解法を用いた場合，クーラン数は 0.2 とし，陰解法を用いた場合，クーラン数は 0.5 とした．

第 3章

CFD 計算の並列化

本章では，CFD 計算に必要な並列化手法とそれに伴って発生する通信処理について述べる．

3.1 領域分割法

本研究における MAC 法による並列 CFD 計算には，領域分割法を用いた．本章では，領域分割法の概要について記述し，並列 CFD 計算の現状について述べる．

3.1.1 領域分割法の概要

領域分割法は全計算領域を小領域に分割し，各小領域を独自に計算することで，全体領域の計算を進める方法が領域分割法である．領域分割法は様々な用途や方法が考案され，古くは Schwarz [14] によって考案された大規模計算を小さな計算量に分割して計算を進める方法があり，近年では領域分割法の並列計算への適用に関する Keys らの研究 [15] がある．

並列計算において良く用いられる領域分割法は，全体領域を Processing Element (PE) 数分と同等かそれ以上の小領域に分割し，各小領域を PE 毎に割り当て，必要な情報をデータ通信して全体の計算を進める方法である．全体領域を PE 数分に分割する方法 (図 3.1) は，有限差分法でよく用いられ，PE 数分以上に分割する方法は，領域分割型有限要素法においてよく用いられる．有限要素法では，PE 数

分以上に小領域を分割し，親 PE が子 PE に計算領域を分配する方法を取る場合が多い．その場合，境界領域の計算を親 PE が行なう．

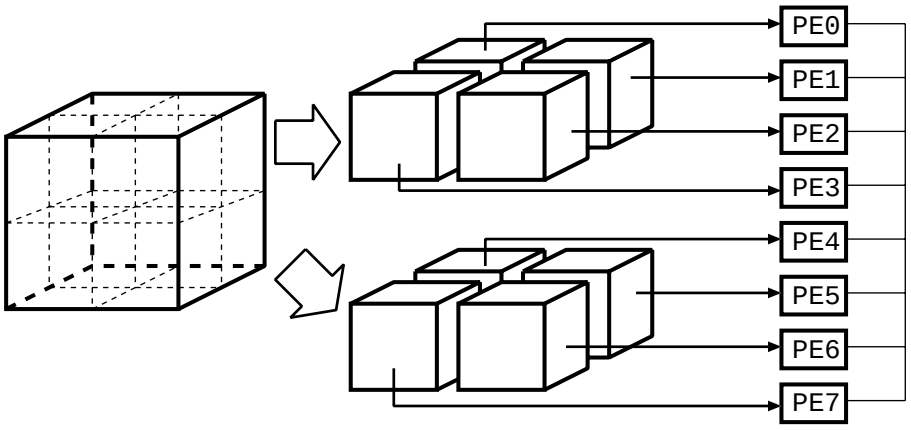

図 3.1: For example domain decomposition

領域の分割方法についても，格子点が分割境界上にある方法と分割境界間にある方法がある (図 3.2)．一般に有限要素法や有限体積法では，前者が用いられ，有限差分法では，後者が用いられることが多い．また，適応格子法のように格子点数が計算を進めるにつれて，変化するような場合には，動的に分割方法が変化する方法を用いる場合があるが，格子点数に変化がない場合には，静的な領域分割を用いることが一般的である．

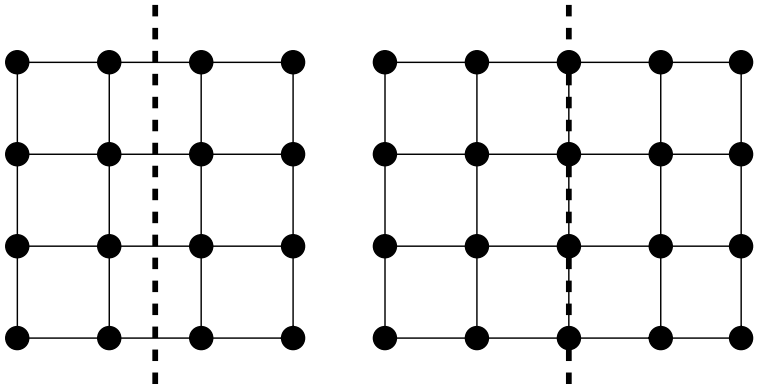

図 3.2: The partitioning location on grid system

3.1.2 有限差分法における領域分割法

本研究では，有限差分法での領域分割法として最も良く用いられている方法として，PE 数分に小領域を分割し，分割境界上に格子点を持たない，重複領域を持つ静的領域分割法を用いた．重複領域を持つ領域分割法は，図 3.1 のような小領域に分割を行なう．そして，図 3.3 のように小領域の内部をデータ通信との並列性を意識した領域に分けて考える．図中の太線内部が実際の計算領域である．この方法は，他領域の境界領域での計算結果の一部を自領域の重複領域のデータとして確保する．重複領域の値が決定されれば，自領域内部は他領域のデータに依存していないので独立に解くことが出来る．重複領域と境界領域の幅は，以下に述べるように有限差分法の差分近似精度に依存する．

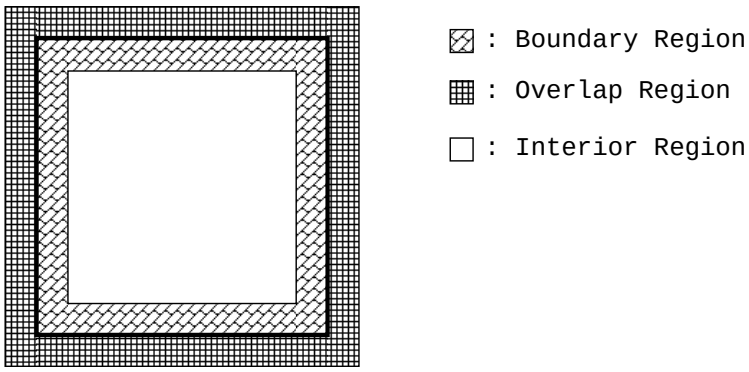

図 3.3: The partitioning location on grid system

そのため，重複領域幅の設定は差分精度によって異なる．本研究で用いた通信処理に関わる差分近似は，2 次精度中心差分と 3 次精度風上差分である．図 3.4 に差分精度による重複領域の状態を示した．Poisson 方程式では 2 次精度中心差分を用いたため，圧力値の計算に必要な重複領域は 1 格子点分必要となる．そのため，圧力データは，1 格子点分のメモリ領域が必要である．そして，NS 方程式では，移流項成分の計算に 3 次精度風上差分を用い，その他の項は 2 次精度中心差分であるため，速度の計算に必要な重複領域は 2 格子点分必要となる．そのため，速度データは，2 格子点分余分なメモリ領域が必要である．もし，全ての PE において，全計算領域を保持する仮想的な領域分割を行なうならば，重複領域を注意する必要はない．しかし，各 PE が自らの計算する領域のみを保持する場合，重複領域分はメモリを余分に確保する必要がある．本研究では，PE は計算する領域の

みの領域データを保持する．そして，メモリ配列は，各方向の格子点数分確保する．一般座標変換後，物理領域は矩形領域に変換されるため，メモリに格納されるデータは，メモリ上の配列位置と格子点上の位置は一致する．

有限差分法において，静的領域分割法を用いて並列計算を行なう場合，格子サイズや使用する PE 数そして用いるハードウェアによってどのように領域分割を行なうかが問題となる．これらの問題は，従来プログラマの経験によって行ってきたのが現状である．そして，多くの場合において通信量を減少させるため，多次元の分割を行なうこと多い．しかし，分割次元が一次元以外では，分割過程が煩雑になる問題がある．分割が一次元でも性能が得られるような場合には，分割次元を落とすことが可能となる．そして，高速な通信機構を持つ並列計算機では，通信量の削減が計算性能の向上に繋がらない場合がほとんどである．

3.1.3 分割設定

重複領域を持つ静的領域分割法は，並列化に伴う分割が非常に容易であり，注意すべき点は重複領域分を考えるだけである．しかし，計算パフォーマンスを考慮した場合，領域確保の単純な問題ではなくなる．どのように領域を分割するかを決定する方法は，一意に決まっていない．一般に領域分割の分割パターンは，データ通信量を少なくするようなブロック形状の分割パターン (3 次元分割) を用いる場合や分割を容易に行なうためやベクトル計算機向きなスライス形状の分割パターン (1 次元分割) を用いる場合が良く用いられる．

本研究では，領域分割パターンを容易にかつ機械的に予測する方法を提案する．そして，その方法によって得られた分割パターン毎の性能予測値と実際に流体計算を行って得られたパフォーマンスを示し，予測方法が有効であることを示す．この結果，様々な並列 CFD の計算を様々な並列計算機上で領域分割法を用いて並列計算する場合に最も効率的な分割パターンを用いることを容易にする．

プログラミングでは，全ての場合において Fortran 77 と Message Passing Interface (MPI) ライブラリを用い，計算精度は倍精度とした．3 次元計算空間の格子点における物理量を直接 3 次元の配列に割り当てた．また並列化に際して，PE 内で計算に必要なデータのみを確保し，計算に用いない部分のデータは確保しない．

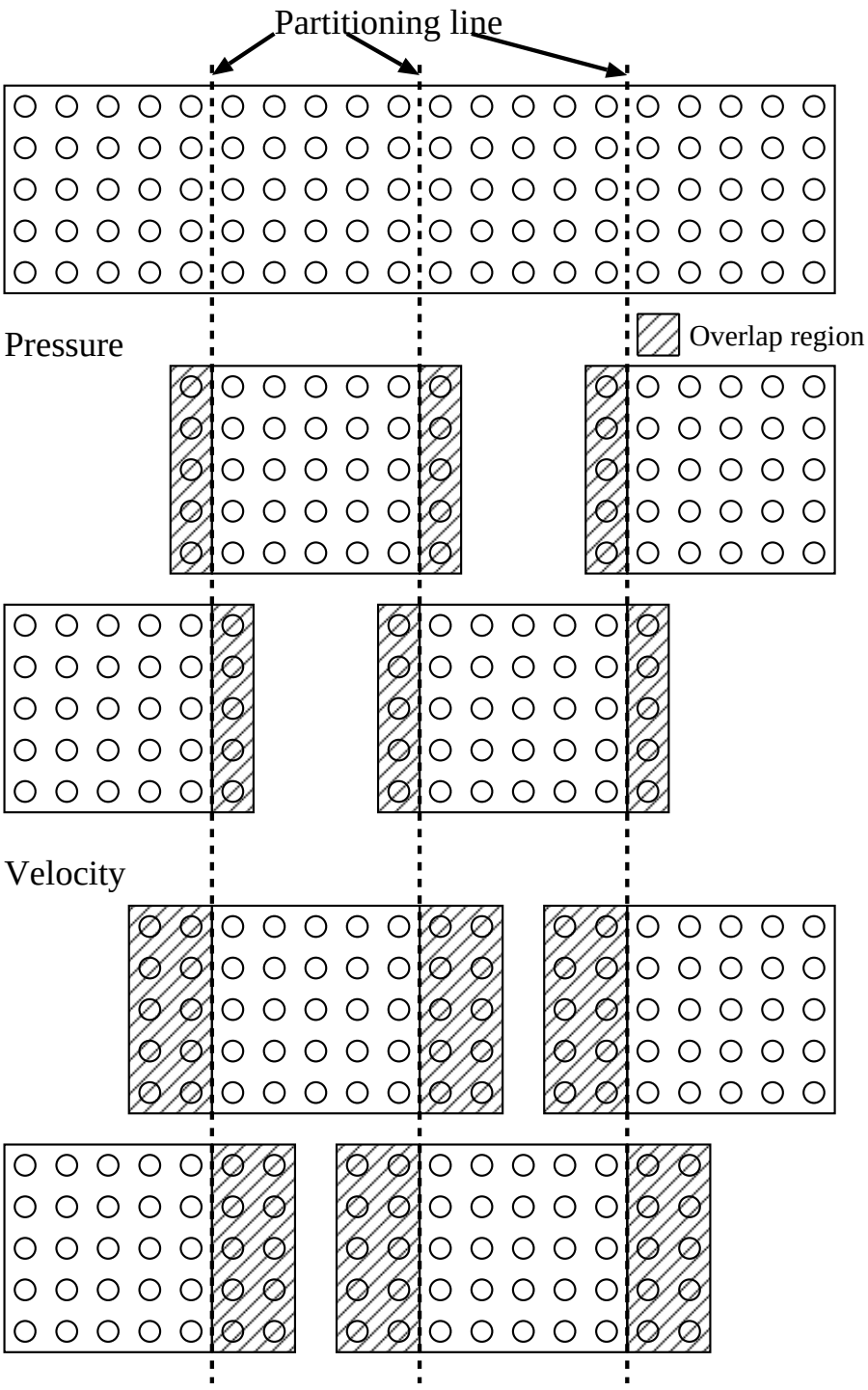

図 3.4: The number of overlap grid plane flagment

3.2 データ通信

本節では，領域分割法を用いた並列化に伴う，通信処理全般について述べる．

3.2.1 データ交換通信

領域分割法を用いて並列化を施した MAC 法による CFD プログラムでは，自領域での境界領域 (Boundary region) の結果を隣接他領域の重複領域 (Overlap region) にデータを通信する

必要がある．すなわち，

1. 自領域の境界領域を隣接領域の重複領域に送信
2. 隣接領域の境界領域を自領域の重複領域に受信

となり，通信の手順は図 3.5 のようになる．

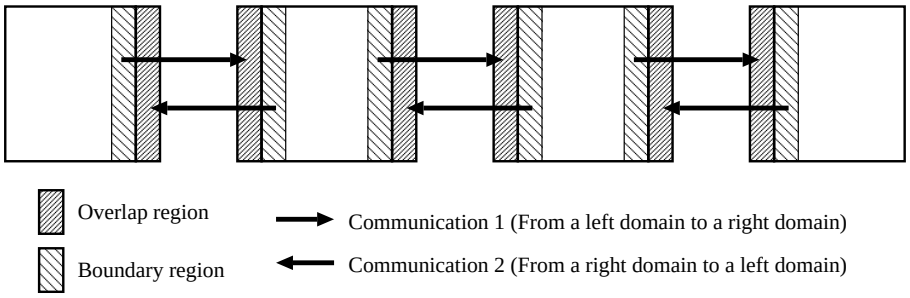

図 3.5: The overview of the shift communication

上述の通信手順は，3 次元領域での 1 次元分割 (スライス分割) の場合における概念図となるが，一般座標系における多次元分割を用いた場合には，通信手順は多少複雑になる．仮に 2 次元分割を考えると，通信パターンは図 3.6 のように 2 次元表示できる．2 次元分割における通信手順の複雑さは，境界領域中の黒点部分の通信にある．本来直交等間隔格子を用いた場合，この部分は両向いの隣接領域に通信するだけで良い．しかし，一般座標変換を行なった場合，Poisson 方程式や NS 方程式に交差微分項があらわれる．そのため，この斜め方向に隣接領域を有する部分の計算には，斜め向いの隣接領域での角部分のデータが必要となる．そのため，斜め方向のデータ通信が必要となる．しかし，上下左右のデータ交換通信

と斜め 4 方向のデータ交換通信を行なうことは，通信回数を増加させるため，通信の立ち上がり時間の影響が大きくなる．特に 3 次元分割を用いた場合さらに長い時間となるため，通信時間を考慮した場合良い方法ではない．

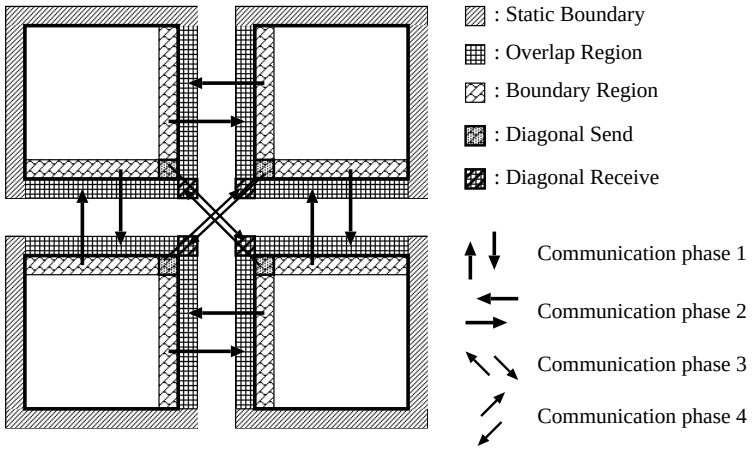

図 3.6: The overview of the communication pattern for the 2-dimensional partitioning

本研究では，通信順序の考慮と通信量の増長を用いて，斜め方向通信をしない方法を用いた．通信順序は，1 つの小領域を考えた場合，実行すべきデータ通信処理を各方向に行なう．例えば，2 次元分割した場合，まず，ある 1 方向のデータ通信処理を実行し終了後，他の 1 方向のデータ通信処理を実行する．また，通信量の増長は，本来通信する必要がない領域 (すなわち，境界領域に隣接する重複領域) を隣接領域に通信することである．これら 2 つの概念を用いることで，図 3.7 のように斜め方向の通信を行なわず，斜め方向の通信と同様の結果が得られる．図中の黒色部分のデータは，1 ステップ目でどちらかの向いの隣接領域に通信され，次のステップにおいて，通信量を増長させることで，斜め向いの隣接領域の重複領域に到達する．この方法は 3 次元分割を用いた場合にも有効である．本研究では，全ての場合において斜め方向の通信は行なわず，全て上述の方法を用いた．

3.2.2 縮約通信

Poisson 方程式や NS 方程式を陰的に解く場合，連立一次方程式を計算する必要がある．本研究では，方程式計算に反復法を用いる．反復法では計算の終了を知

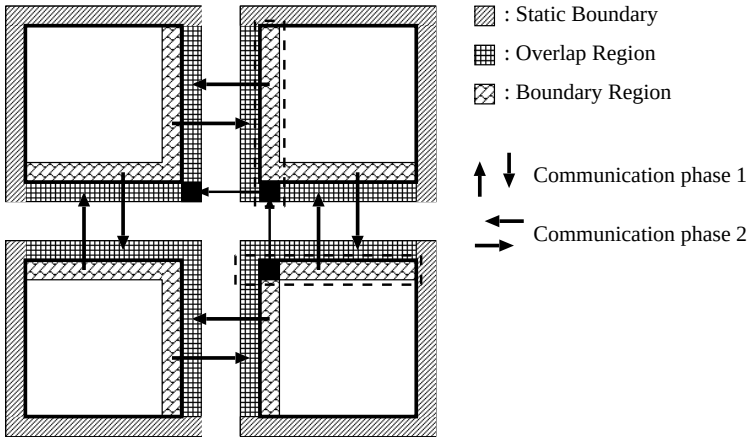

図 3.7: The overview of the new communication pattern for the 2-dimensional partitioning

るため，残差値 (誤差値) を確認する必要がある．しかし，領域分割を行なっているため，各小領域ごとの残差値は明らかであるが，全体領域での残差値は明らかではない．そのため，残差値を全領域に対する値である必要があるため，PE 全体 (全小領域で同じ値) に知らせる必要がある．

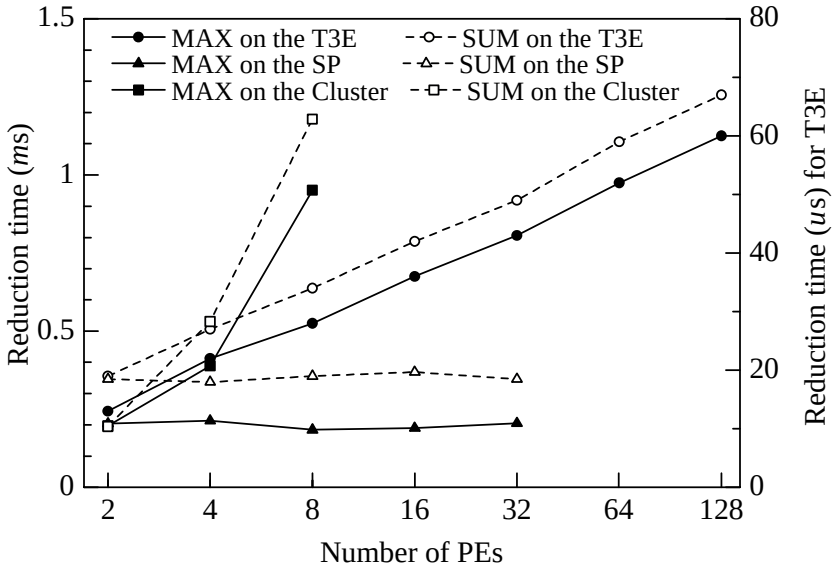

図 3.8: reduction communication time

通信量の削減と処理速度を考慮した場合，残差値自体の縮約通信を行なうより，残差値の処理を各 PE が行い，その結果によってフラグを立てることで，残差値自身のデータ通信をせず，データ量を減少させることが出来る．しかし，残差値

は収束判定のみならず，流れの状態を示す様々な情報を含んでいるため，フラグのやり取りだけでなく，残差値自身を用いた方が良い．通常，残差値は各格子点における誤差値の総和や最大値が用いられることが多い．計算処理性能において，この 2 つ処理の違いは大きな差はない．しかし，通信処理性能において大きな影響が現れる．多くの場合，総和による縮約通信の時間より最大値による縮約通信の時間の方が短い (図 3.8)．本研究では，反復法の残差評価を全て最大値によって行なった．

3.2.3 並列計算機システム

本研究で用いた並列計算機システムは，CRAY Inc. CRAY-T3E/1200E(以後 T3E)，IBM RS/6000 SP(以後 SP)，SGI Onyx2 Infinite reality(以後 Onyx)，PC Cluster Type1(以後 Cluster1)，PC Cluster Type2(以後 Cluster2) である．それぞれの並列計算機システムの概要を以下の表に示した．

表 3.1: Specification CRAY-T3E/1200E

CPU	Alpha 21164 600MHz
Total PE	128
Memory	512MB/PE
Secondary cache	96KB
Interconnect	3D torus
Network bandwidth	650MB/s
OS	Unicosmk 2.0.5.40
Compiler	Cray Fortran Verion 3.5.0.0
Compile Option	-O3 -M1110

表 3.2: Specification RS/6000 SP

CPU	PowerPC604e 334MHz
Total	256CPU(64Node)
Memory	256MB/Node
Secondary cache	256KB
Interconnect	Multistage-Switch
Network bandwidth	150MB/s
OS	AIX 4.3
Compier	XL Fortran 5.1[16]
Compile option	-O3 -qstrict -qarch=604

表 3.3: Specification Onyx2 reality monster

CPU	R10000 250MHz
Total PE	16
Memory	16GB(shared)
Secondary cache	2MB
Interconnect	HyperCube
Network bandwidth	800MB/s
OS	IRIX6.5
Compier	MIPSPro Fortran 7.0
Compile option	-fast -apo

表 3.4: Specification PC Cluster Type1

CPU	Pentium-III 450MHz
Total PE	8
Memory	256MB/PE
Secondary cache	512KB
Interconnect	Switch-Hub
Network bandwidth	12.8MB/s
OS	Linux 2.2.14
Compiler	Fujitsu Linux Fortran V2
Compile option	-O3

表 3.5: Specification PC Cluster Type2

CPU	Pentium-4 1.5GHz
Total PE	8
Memory	512MB/PE
Secondary cache	256KB
Interconnect	Switch-Hub
Network bandwidth	12.8MB/s
OS	Linux 2.4.1
Compiler	PGI Compiler 3.2-4
Compile option	-fast -Mvect

第 4章

領域分割パターンの決定

本章では，領域分割法を用いた MAC 法による並列 CFD 計算での領域分割パターンによる性能変化と最適な分割パターンの選択方法について述べる．

4.1 分割パターンの概要

静的領域分割法を用いて並列計算を行なう場合，格子形状や使用するハードウェアの特性によって計算性能が大きく変化する．そのため，どのように領域分割を行なうかが高性能を維持するために問題となる．これらの問題は，プログラマの経験によって行ってきたのが現状である．一般に MAC 法での領域分割時の分割パターンは，従来の有限差分法の並列計算全般において，使用する PE 数が増え小領域数が増える場合，通信量が少なくなるような分割を用い，通信処理の影響を最小にするという戦略で領域分割を実行することが多く行われてきた [1]．また，逆に領域分割のプログラミングを容易にするため，1 次元スライス分割を採用する．特に 1 次元分割によるネットワーク性能の低下が起こらないクロスバネットワーク接続の並列ベクトル型計算機の場合には用いる場合も多い [17].

分割パターンはデータ通信量やループ長の違いによる計算性能の両方から並列計算性能全体に影響を及ぼす [18]．従来の分割パターンの決定は，領域分割の必要性と既成概念から得られた方法であり，高性能を維持できる分割パターンではない．しかし，PE 数が多くなった場合，膨大な分割パターンの選択肢がある．例えば 8 PE 用いるために 8 領域に分割する場合を図 4.1 に示した．また，格子形状

は必ず正立方体であるとは限らないため，格子分割が各方向で異なる場合の格子形状の設定方向も 6 つの設定が可能であり，2 方向が同一分割数の場合 3 つの設定が可能である (図 4.2)．また，反復法の種類によっては，格子形状の取り方 (すなわち境界条件の設定位置) によって収束性も変化する．

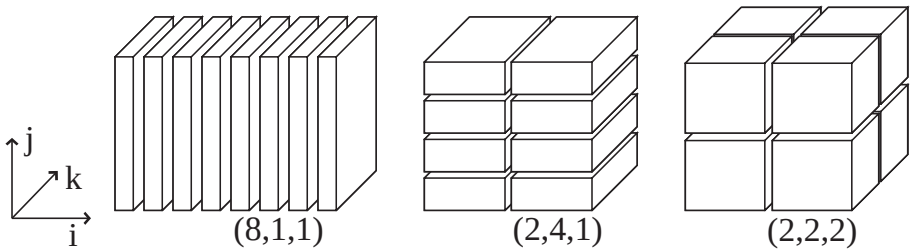

図 4.1: For example, Partitioning patterns for domain decomposition method

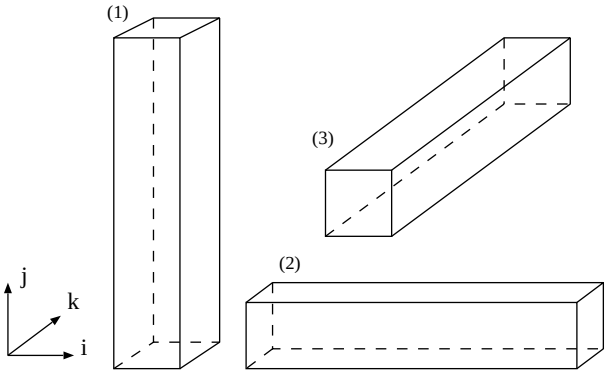

図 4.2: For example, grid style

分割パターンの決定の考慮点は，用いる並列計算機の実効性能を落とさないことであると最適な (付近の) 分割パターンを事前に知る方法である．性能評価の研究事例として，接合格子による偏微分方程式の並列計算性能を通信量と計算量組み合わせ最適化問題として捉え，性能を予測する試みがある．しかし，最適化問題を解く必要があるため最適解を得るために非常に時間を要し，比較的誤差が大きい [19]．また，スカラー計算機における逐次計算性能の評価としては，Wijngaart ら [2] による簡易キャッシュモデルを構築し，計算処理のキャッシュ動作をシミュレートして性能評価をする方法がある．しかし，この手法もキャッシュアーキテクチャに精通している必要があり，性能予測もあまり正確ではない．従来，有限差分法による領域分割法による分割パターンの研究は，ほとんど行なわれていない．多く

の並列シミュレーションを用いた研究においてデータ通信量を少なくする 3 次元分割や単純な 1 次元分割を用いる場合が多い．しかし，通信性能の向上やスカラー計算機の高性能化によって，領域分割パターンによる性能への影響は非常に大きい．

そのため，並列シミュレーションを実行する前に，機械的に並列計算効率の高いより良い領域分割パターンを決定する指標が必要である．また，分割次元が 1 次元以外では，分割過程が煩雑になる問題がある．分割が 1 次元でも性能が得られるような場合には，分割次元を落とすことでプログラムの生産性が向上する．もし，どのような分割パターンを用いても性能に大きな変化が見られないのであれば，分割が最も単純なスライス分割を用いればよい．また，ハードウェアの特性の違いをプログラマが熟知する必要が無くなる．しかし，収束性の変化の考慮は現状では不可能に近いため考慮せず，計算性能のみが最も良い値を示す方法を考慮する．

4.2 分割パターンによる並列 CFD 計算性能の予測

逐次処理時における CFD 計算性能，特に MAC 法ベースの解法を用いた性能予測には，Poisson 方程式ソルバーの性能を評価する場合が多い．MAC 法ベースの計算は，物理量の計算部分によってかなりの計算負荷の片寄りが見られる．通常，圧力解法部分の計算負荷が非常に高くなる傾向がある．そのため，計算負荷の見積もりを行なう場合，計算のカーネル部分である Poisson 方程式を解くことで CFD シミュレーション時間全体の計算負荷をある程度見積もる事が可能であり，ベンチマークプログラムとして良く用いられる [20][7]．この考え方は，並列 CFD シミュレーション時間の評価においても十分に通用するものと考えられる．

もし，この考え方を並列計算時に適用した場合，Poisson 方程式ソルバーを並列計算することで，予測可能になると思われる．実際にシミュレーション条件を元に並列計算を行なうには，同じ PE 数を用いる必要がある．しかし，PE 数の数を確保することは，使用する PE 数が増加するにつれて難しくなる．そのため，予測において用いる PE 数は少なければ少ないほど良い．そのため通常の並列 CFD シミュレーションの性能予測では，1 PE での計算性能評価とその結果を元に通信性能評価を行なう 2 段階の評価を行なうことが良いと考えられる．

1 PE 時の計算性能評価は，理論性能値を用いて全てを計算で算出する方法と実

際の計算を行なう方法がある．通常，理論性能を元に理論的に計算性能を評価する場合，計算量を元にした計算性能モデルの場合，分割数が固定で，分割パターンが違う小領域内の計算時間は，計算量が変化しないため，計算性能はどの分割数でも同じとなるが，スカラー計算機の場合，ループ長によるキャッシュや TLB 等の要因による挙動変化が大きく，現状では全てを理論で行なうことは非常に難しい．また，ベクトル計算機でも同じような状態である．実際に計算してみる場合，単一 PE で大規模問題を解くような場合，評価計算のために大きな時間が掛かる．しかし，領域分割法を用いた並列計算における 1 PE 分の計算量は，PE 数が増えるほど小さくなる．そのため，PE 数が増加して分割パターンが多くなった場合でも，評価に要する計算時間は非常に小さいものになる．

4.3 評価手法

4.3.1 計算性能

並列計算処理性能は，各分割パターンでの 1 PE 分の計算領域の計算性能で見積もる．ただし，上述のように流体コード全体の計算性能ではなく，流体コードの計算負荷の高い部分だけを領域分割から得られる各分割パターンによる分割形状を総当りで計測する．計算負荷が高い部分のみに限定することで，計測時間はすべての分割パターンを総当たりで計測しても大きくない．1 PE のみを利用するため，評価に必要な計算資源も少なくすむ．そして，かなり高精度に並列計算処理性能を評価することが可能である．

通常，スカラー型計算機やベクトル型計算機でも実効性能と理論性能には，かなりの開きがある．この原因は，様々な問題が複雑に絡む問題であるため，一意に決めることは出来ない．しかし，最も大きな原因は，主メモリからのデータ処理等のデータ移動動作遅延時間であると考えられる．上記の方法で得られる 1 PE の計算性能は，並列計算処理性能を見積もるではなく，データ移動動作の遅延，即ち計算機システムの理論性能と実効性能の差を予測することも可能であると考えられる．

4.3.2 通信性能

並列計算機の通信性能の予測モデルは，データ通信量，通信帯域，レイテンシそしてデータ通信モデルによって決定される．通信帯域とレイテンシは，ハードウェア理論性能を用いることで簡単に決定できる．そして，理論値と実測値のギャップは，メモリ移動時間も上記の計算性能を用いることで予測が可能であると考える．そして，データ通信モデルは，用いる並列計算機のネットワークのトポロジーや用いるデータ通信アルゴリズムから割り出すことが可能である．構造格子の静的領域分割の差分計算では，基本的にデータを隣接 PE に対して通信するデータ通信パターンが採用されている．

T3E は，3D トーラスネットワークである．そのためデータ通信は $O(1)$ で終了すると考えられる．SP の場合，多段スイッチネットワークを用いており，クロスバールーティング [21] によって，どの PE に対する通信も $O(1)$ での通信となる．PC/WS クラスタの場合，一般にスイッチ Hub (インテリジェンス or ブリッジ) を用いて，全ての PC を接続するか，スイッチ Hub を数台用いてカスケード接続するかである．本研究で用いた通信処理を単一スイッチ Hub 接続の PC Cluster で実行することを考えると，2 分割の場合 $O(1)$ となるが，4 分割以上では $O(2)$ となる．そのため，各方向の分割数を 2 分割にする意味は大きい．しかし，最近のブリッジ Hub の性能は，用いるスイッチ本体の性能によってかなり個体差があり，かならずしも一意には得られない場合が多い．

また，アルゴリズムによるデータ通信コストは，同期通信モデとル非同期通信モデルによってデータ通信コストは異なる．同期通信モデルの場合，データ通信において通信は階段状に通信が進むため，分割数を N とした場合の通信コストのオーダーは $O(N-1)$ となる．この場合，各次元の分割数が少なければ良い．しかし，これでは PE 数が増加した際のデータ通信効率が非常に低くなる．そのため，通信パターンに Tree モデルを用いて，同期通信時に通信コストのオーダーを $O(log_2(N))$ とすることが出来る．分割数が 2 以上の場合，効率の良い同期データ通信が可能となる．非同期通信モデルでは，通信パターンを考慮する必要は無く，ハードウェアの通信コストに依存する．そのため，比較的通信性能の一貫性が得られない場合が多い．しかし，分割数が多くなった場合非常に有効である．

通信処理においても理論性能値から正確な通信処理性能は得られない．この原

因も計算処理の評価で述べたように，データ移動の遅延に起因するものと仮定した．通信処理に対するデータ移動の遅延をモデル化するために，計算性能で得られた実測値を用いた．通信性能の予測にも計算性能の実測値を用いることで有効な通信性能予測を行えるものと考えられる．以上のことから，分割パターンによる通信時間を算出することが可能となる．この通信時間と 1 PE が計算する小領域の計算量から 1 秒当りの浮動小数点演算量 (Mega Floating point operation per second : MFlop/s) 値とする．しかし，MFlop/s は性能の評価値を示す無次元量として取り扱う．

また，並列 CFD 計算時間に占める通信時間として大きな影響がある反復制御のリダクション通信 (総和や最大値を求める通信，例 : MPI_Reduce 等) は，用いる PE 数に依存する。そのため，PE 数固定の静的領域分割の性能評価に PE 数に依存するリダクション通信は考慮する必要はない．

4.4 並列処理コストモデル

上述の点を踏まえて，並列 CFD の計算性能予測モデルを計算カーネル部 (Poisson 方程式解法部) の性能モデルとして作成する．モデルから得られる値は，全て MFlop/s に換算して評価する．

4.4.1 並列計算コスト

並列計算コストは，1 PE の処理時間を並列処理の計算コストと考える．そして，1 PE の計算時間を MFlop/s として評価する．1 反復中の総浮動小数点演算量を $FLOPs$ と計算時間 $Time$ すると，計算コスト $Cost_{comp}$ は，

$$Cost_{comp} = \frac{FLOPs}{Time(\mu s)} \tag{4.1}$$

となる．実際に測定した $Cost_{comp}$ は，

$$Cost_{comp} = \frac{FLOPs \times ITR}{Time(\mu s)} \tag{4.2}$$

$ITR : Number\ of\ Iterations$

となるが，式 (4.1) の $Cost_{comp}$ は 1 iteration での $Cost_{comp}$ となるため，式 (4.2) と違いはない．$ITR = 200$ として計測した．

4.4.2 通信処理コスト

カーネル計算部分の 1 反復に必要な通信処理時間 (s) を理論性能からモデル化する．同期通信モデルでは，

$$\text{Comm Time} = BW \times \sum_{n=1}^{N} (DS_n \times log_2 DD_n) + L \times N \tag{4.3}$$

非同期通信モデルでは，

$$\text{Comm Time} = BW \times \sum_{n=1}^{N} (DS_n) + L \times N \tag{4.4}$$

となり，N は分割次元数，DS は各次元での通信データ量 (Byte)，BW は理論通信帯域幅 (Byte/s)，DD は各分割次元での分割数，L は通信立ち上がり時間 (レイテンシ) である．このままの状態でもある程度の通信処理コストの検討は可能であるが，本研究では，並列計算コストの値を用いて，対理論性能差 (データ移動の遅延) DT を

$$DT = \frac{Cost_{comp}}{CF} \tag{4.5}$$

として評価する．CF にどの値を用いるかはハードウェアにある程度依存するが，本研究では，CPU の駆動周波数 (MHz) とした．式 (5.6)，(4.5) を考慮すると通信処理時間は，

$$\text{Comm Time} = BW \times DT \times \sum_{n=1}^{N} \left(DS_n \times log_2 DD_n \right) + L \times N \tag{4.6}$$

となる．式 (4.6) と $FLOPs$ から通信コスト $Cost_{comm}$ は，

$$Cost_{comm} = \frac{FLOPs}{\text{Comm Tine} \times 1.0 \times 10^{-6}} \tag{4.7}$$

となる．

4.4.3 モデル

以上のことから，並列処理コスト $Cost_{Total}$ は，

$$Cost_{Total} = Cost_{comp} + Cost_{comm} \tag{4.8}$$

となり，並列処理の評価値は，$Cost_{Total}$ で表す．通信機構の理論性能は表 4.1 に示した．

表 4.1: The data communication parameter

Machine	CRAY/T3E-1200E	RS/6000 SP	PC Cluster
通信帯域	650MB/s	150MB/s	12.8MB/s
立上り時間	1.0×10^{-8}	1.2×10^{-6}	1.0×10^{-5}

また，格子サイズは，$34 \times 34 \times 130$ であるが，3 種類の設定方法があるため，表 4.2 のように示す．

表 4.2: Grid type

Type-A	$130 \times 34 \times 34$
Type-B	$34 \times 130 \times 34$
Type-C	$34 \times 34 \times 130$

4.5 評価および結果

モデル式の評価のために，格子形状が偏っている 3 次元 180 度曲がり管内流の CFD シミュレーションを行なった (図 4.3)．格子サイズは，$130 \times 34 \times 34$ とした．格子設定が 3 方向存在するため，PE 数分の分割パターン $\times$ 3 を全て計算した．管断面は正方形とし，管の曲率半径比 R は 3 である．管 1 辺の長さと平均流入流速からレイノルズ数は 100 とした．このシミュレーションの無次元時間 1 までの経過時間を，T3E は 128 PE，SP と Cluster2 は 8 PE 用いて計測した．

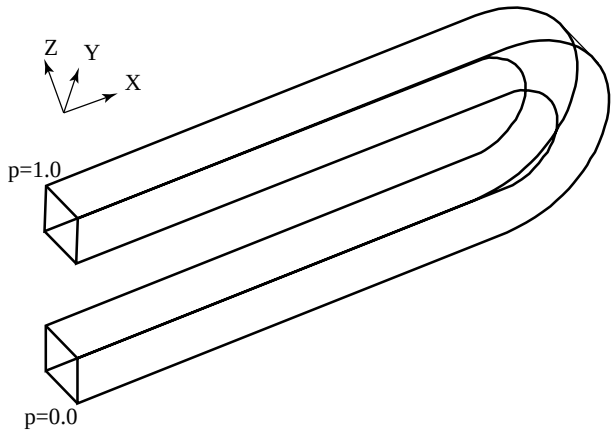

図 4.3: Overview of the bend pipe

4.5.1 計算条件

式 (2.5) の Poisson 方程式の離散化は，2 次精度中心差分を用いた．式 (2.6) の NS 方程式の離散化は，左辺第 2 項の移流項に 3 次精度風上差分を用い，その他の空間微分項は全て 2 次精度中心差分を用いた．また，速度の時間微分項は，1 次精度前進差分を用いた．

x 方向の流速を u，y 方向の流速を v，z 方向の流速を w とした．速度の境界条件は，流入，流出境界が自由境界とし，その他の境界は $u, v, w = 0.0$ とした．圧力の境界条件は，流入境界で $p = 1.0$，流出境界で $p = 0.0$ とし，その他の境界で圧力勾配を 0.0 とした．

Jacobi 法の収束判定は，各格子点での残差の最大値が 1.0×10^{-5} になるまでとした．

4.5.2 流れ場

時間ステップを進めた場合の結果としてベクトル図と 2 次流れコンタ図と流線図を図 4.4，4.5 と 4.6 に示す．

図 4.5 に曲がり管全般に観察することが出来る 2 次流れの様子が観察できた．また，2 次流れの影響から図 4.6 の流跡線が旋回する様子も見られた．また，図 4.4 から圧力値の初期条件から流れが角管における Poiseuille 流れに発達した．曲がり管における流れの特徴が見受けられ，計算結果は妥当であると考えられる．

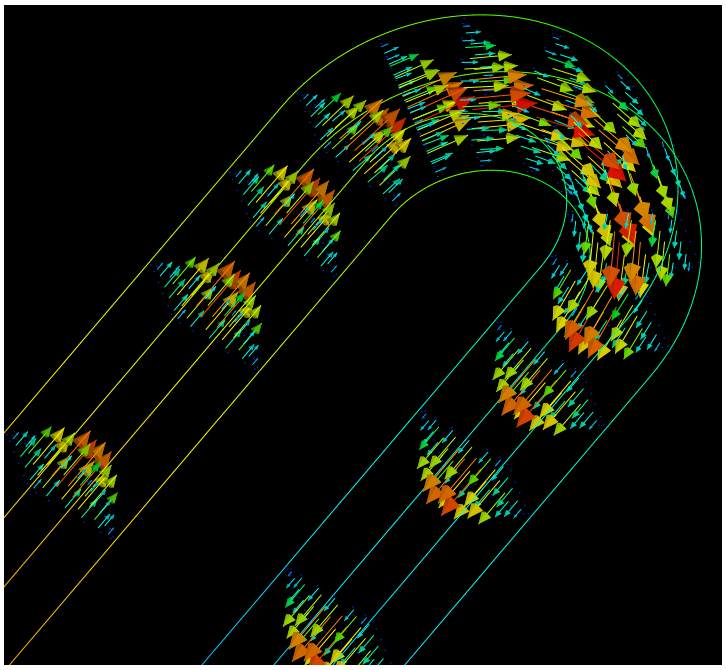

図 4.4: Vector fields

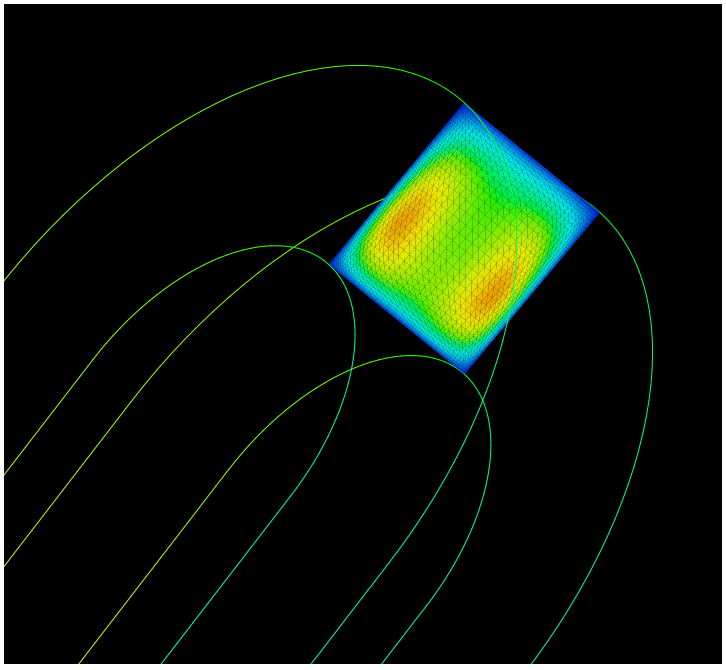

図 4.5: Vorticity

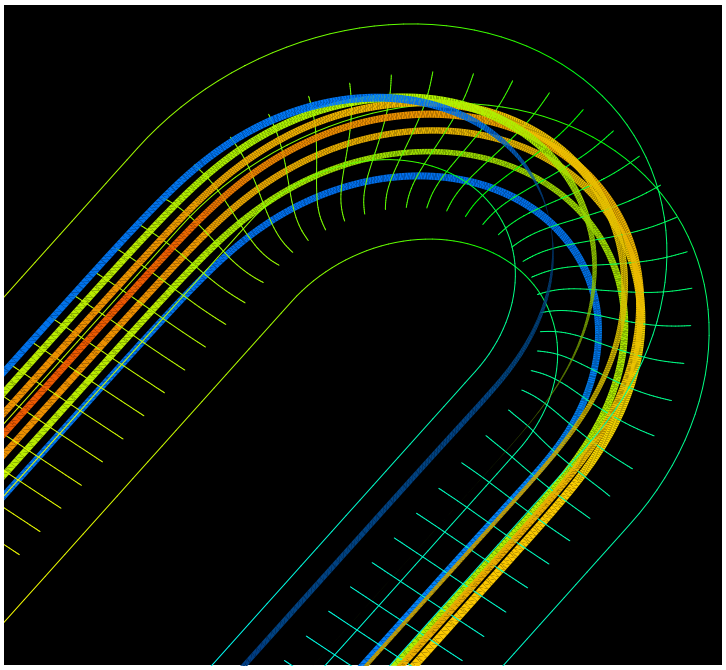

図 4.6: Streamlines and pressure contour

4.5.3 並列計算性能予測

並列 CFD 計算では，以下の各時間を計測した．並列処理の総経過時間 $Etime$ ，並列処理時の総計算処理時間

$$Ctime = \left(\sum_{n=1}^{NPE} Ctime_n \right) /NPE$$

，並列処理時の通信時間 $Ptime = Etime - Ctime$ である．NPE は使用 PE 数とする．

各結果を以下に示した．

図 4.7，4.8，4.9，4.16，4.17，4.18，は，縦軸に並列 CFD の総経過時間 $Etime$，横軸に評価値をとり，図 4.10，4.11，4.12 は縦軸に総経過時間 $Etime$，横軸に計算性能パラメータ $Cost_{comp}$ をとり，図 4.13，4.14，4.15 は縦軸にデータ通信時間 $Ptime$，横軸に通信性能パラメータ $Cost_{comm}$ を取った．

同期通信を用いた場合の全体性能予測

同期通信を用いた場合において，図 4.7，4.8，4.9 に $Etime$ と性能予測値の相関を示した．

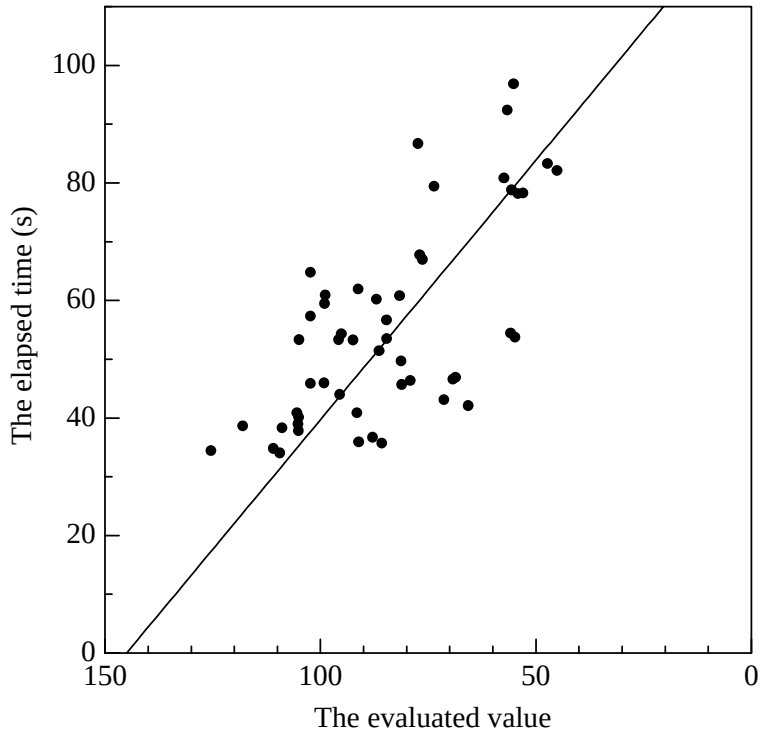

図 4.7: Correlation between experiment and evaluation on the T3E

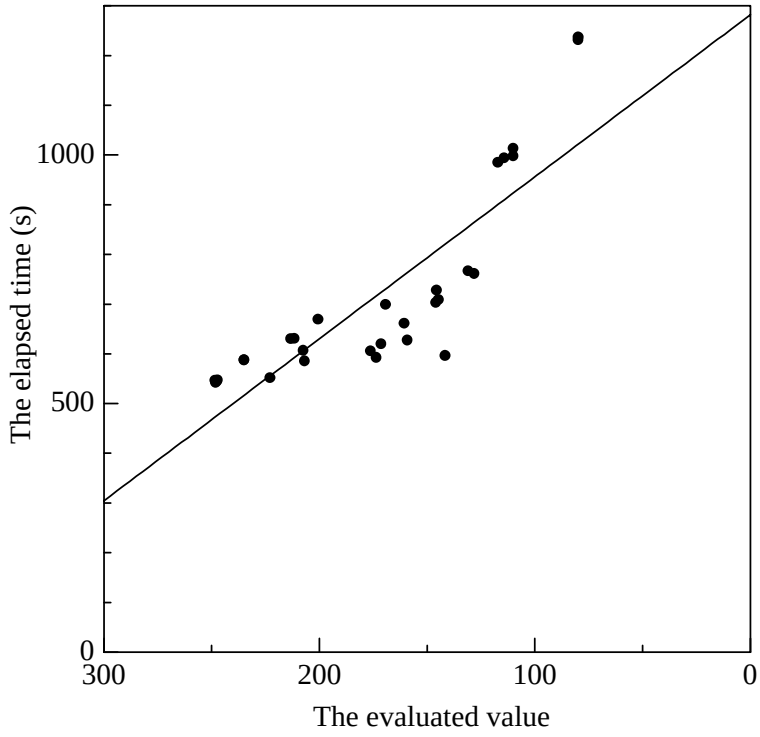

図 4.8: Correlation between experiment and evaluation on the SP

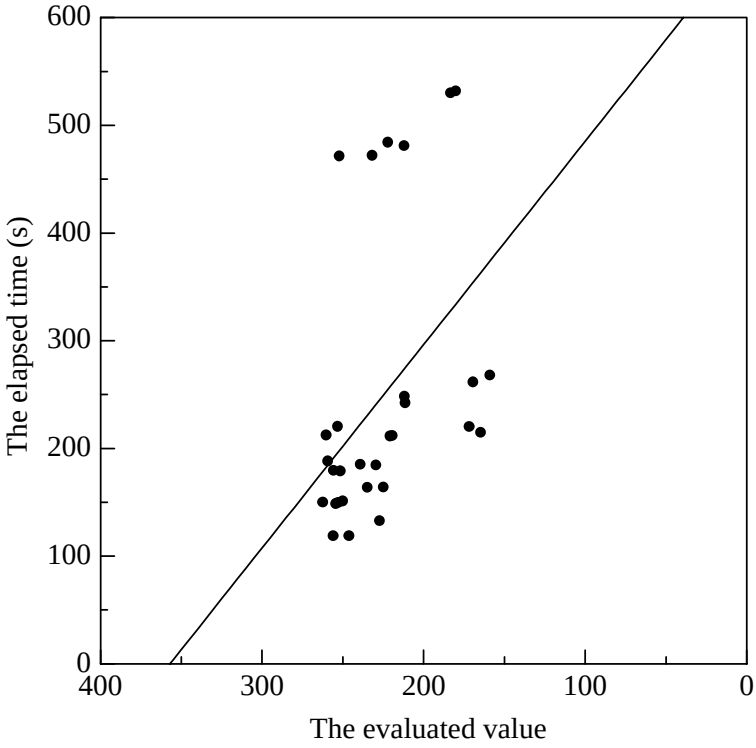

図 4.9: Correlation between experiment and evaluation on the Cluster2

T3E において，図 4.7 から全性能予測値と実際の流体計算との総経過時間の相関係数 -0.70 が得られた．同様に SP において，図 4.8 から相関係数 -0.87 が得られ，Cluster2 において，図 4.9 から相関係数 -0.61 が得られた．SP では，比較的良好な結果が得られた．T3E では，予測値に多少のばらつきが見られた．Cluster2 では，大きく異なる予測値が得られ，その結果として相関値が大きく低下した．

計算性能と総経過時間

図 4.10，4.11，4.12 に 総経過時間 $Etime$ と計算性能 $Cost_{comp}$ の相関を示した．

T3E において，図 4.10 から計算性能予測値と CFD シミュレーションの総経過時間との相関係数は，-0.61 が得られた．SP において，図 4.11 から相関係数は，-0.48 が得られ，Cluster2 において，図 4.12 から相関係数は，-0.32 が得られた．計算性能予測値と CFD シミュレーションの総経過時間の相関係数は，全性能予測値よりも低い係数である．特に SP と Cluster2 では半分程度である．また，T3E では差は小さい．

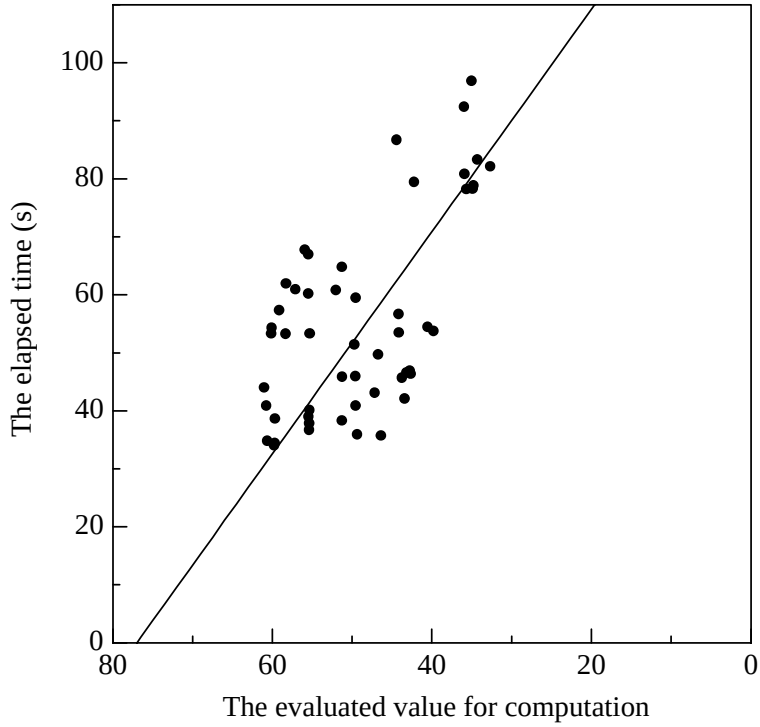

図 4.10: Correlation between $Cost_{comp}$ and $Etime$ on the T3E

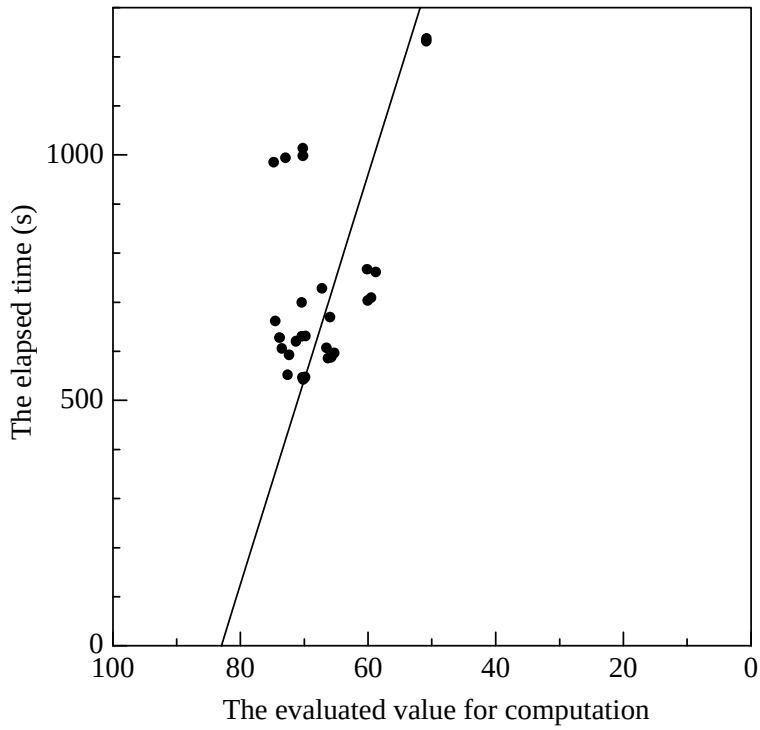

図 4.11: Correlation between $Cost_{comp}$ and $Etime$ on SP

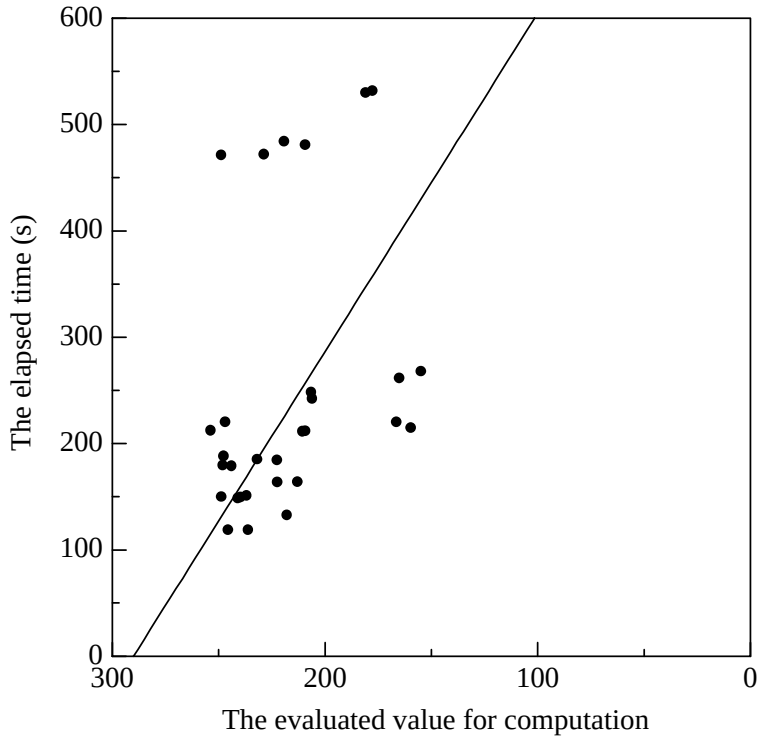

図 4.12: Correlation between $Cost_{comp}$ and $Etime$ on the Cluster2

パラメータ DT とデータ通信時間

図 4.13，4.14，4.15 にデータ通信時間 $Ptime$ と $Cost_{comm}$ の相関におけるパラメータ DT 値の効果をそれぞれ示した．

T3E において，図 4.13 から通信性能予測値と CFD シミュレーションにおける通信時間との相関係数は，DT を用いた場合 -0.53，用いない場合 -0.25 が得られた．SP において 図 4.14 から相関係数は，DT を用いた場合 -0.83，用いない場合 -0.78 が得られた．また，Cluster2 において図 4.15 から相関係数は，DT を用いた場合 -0.64，用いない場合 -0.63 が得られた．パラメータ DT を用いた場合，総じて実際の通信時間との相関係数が高くなった．しかし，Cluster2 では，大きな変化として表れなかった．

非同期通信を用いた場合の全体性能予測

非同期通信を用いた場合において， 図 4.16，4.17，4.18 に総経過時間 $Etime$ と非同期通信を用いた場合の性能予測値の相関を示した．

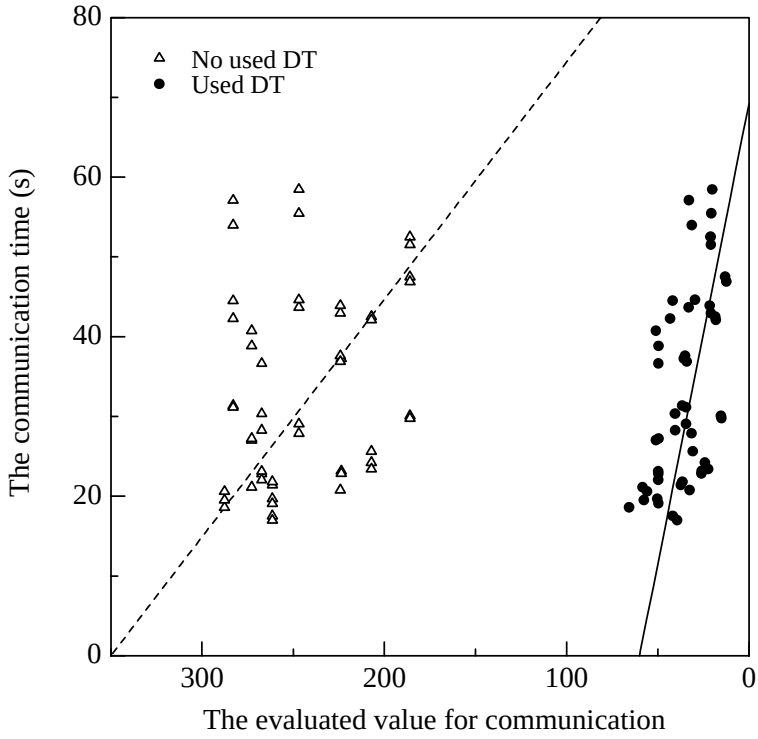

図 4.13: Correlation between $Cost_{comm}$ and $Ptime$ on the T3E

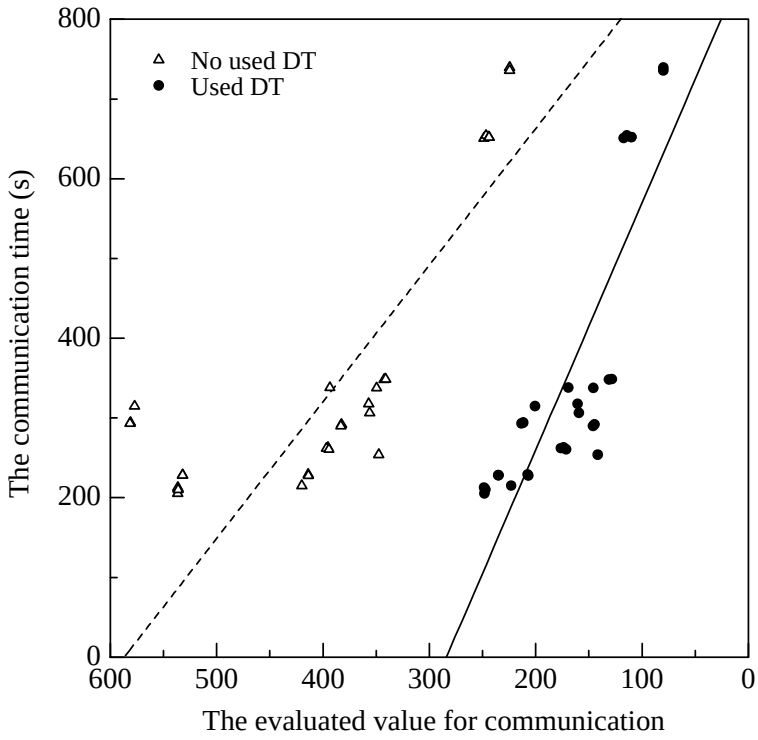

図 4.14: Correlation between $Cost_{comm}$ and $Ptime$ on the SP

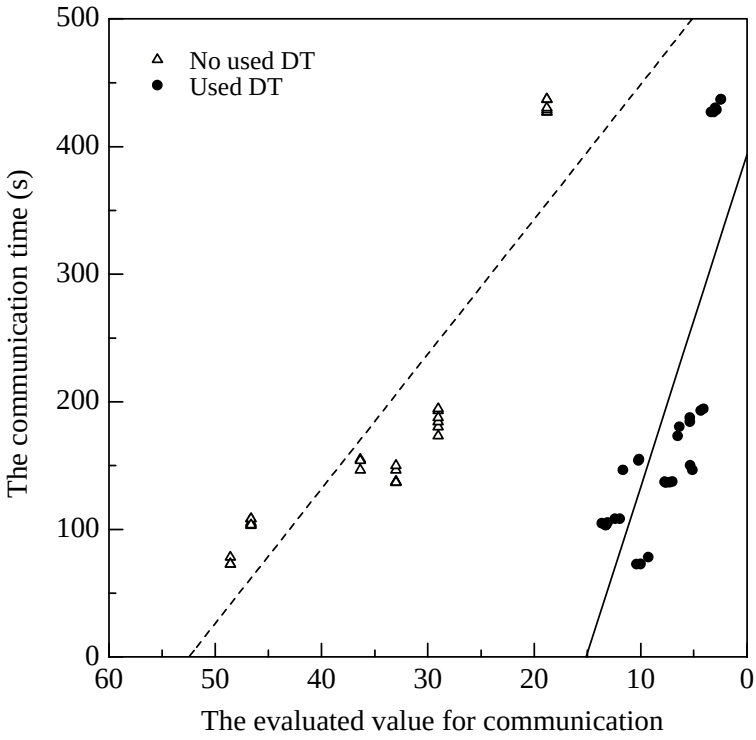

図 4.15: Correlation between $Cost_{comm}$ and $Ptime$ on the Cluster2

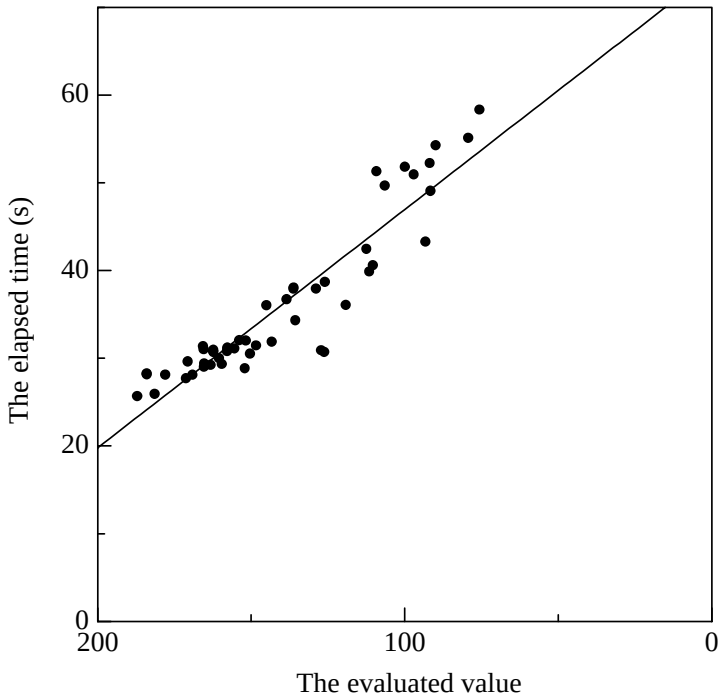

図 4.16: Correlation between experiment using asynchronous communication and evaluation on the T3E

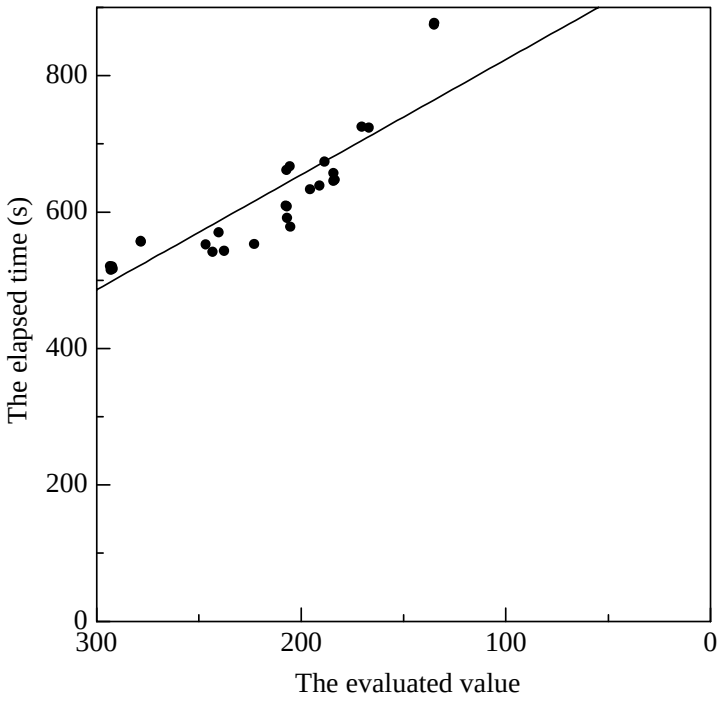

図 4.17: Correlation between experiment using asynchronous communication and evaluation on the SP

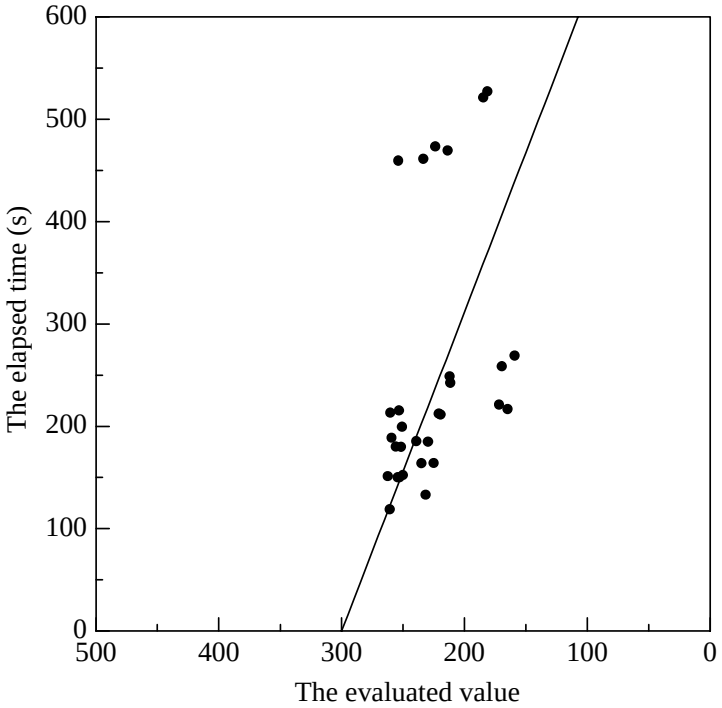

図 4.18: Correlation between experiment using asynchronous communication and evaluation on the Cluster2

CFD 計算に非同期通信を用いた場合の結果を示した．T3E において，図 4.16 から全性能予測値と実際の流体計算との総経過時間は相関係数 -0.94 が得られた．同様に SP において，図 4.17 から相関係数 -0.91 が得られ，Cluster2 において，図 4.9 から相関係数 -0.60 が得られた．T3E や SP においては，非常に良い結果が得られた．しかし，Cluster2 においては，同期通信を用いた場合と比べて，多少相関係数が低くなった．

非同期通信を用いた場合の通信性能予測

非同期通信を用いた際のデータ通信時間 $Ptime$ と通信予測値 $Cost_{comm}$ の関係を図 4.19，4.20，4.21 に示した．

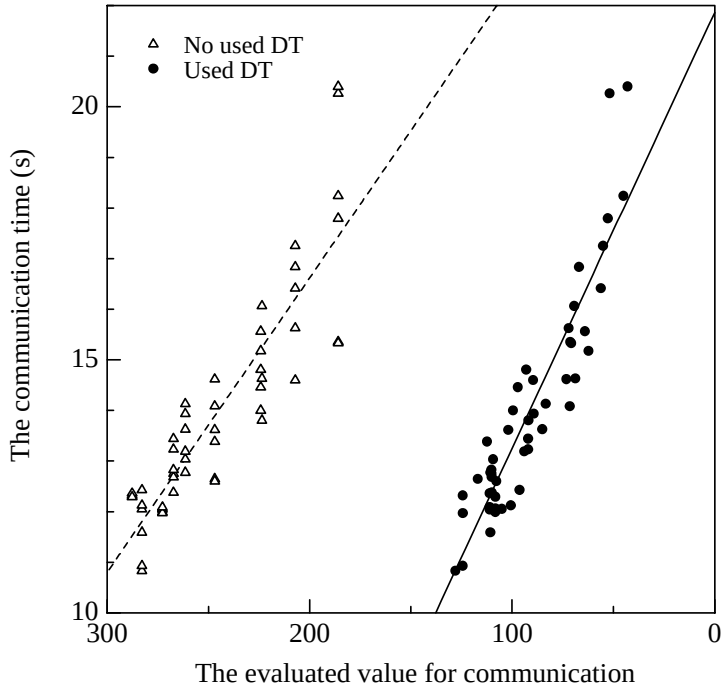

図 4.19: Correlation between $Cost_{comm}$ of asynchronous communication and $Ptime$ on the T3E

T3E において，図 4.19 から通信性能予測値と CFD シミュレーションにおける通信時間との相関係数は，DT を用いた場合 -0.89，用いない場合 -0.81 が得られた．SP において 図 4.20 から相関係数は，DT を用いた場合 -0.87，用いない場合 -0.64 が得られた．また，Cluster2 において図 4.21 から相関係数は，DT を用いた

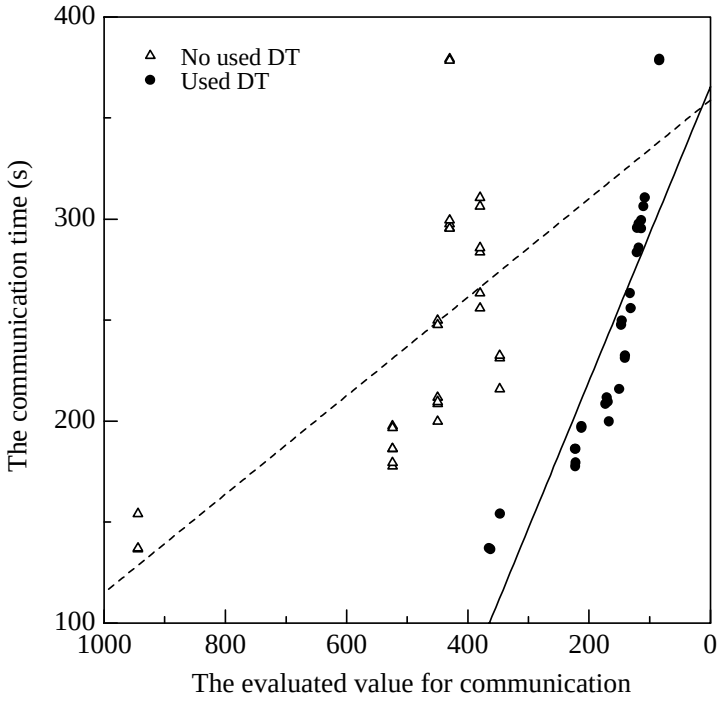

図 4.20: Correlation between $Cost_{comm}$ of asynchronous communication and $Ptime$ on the SP

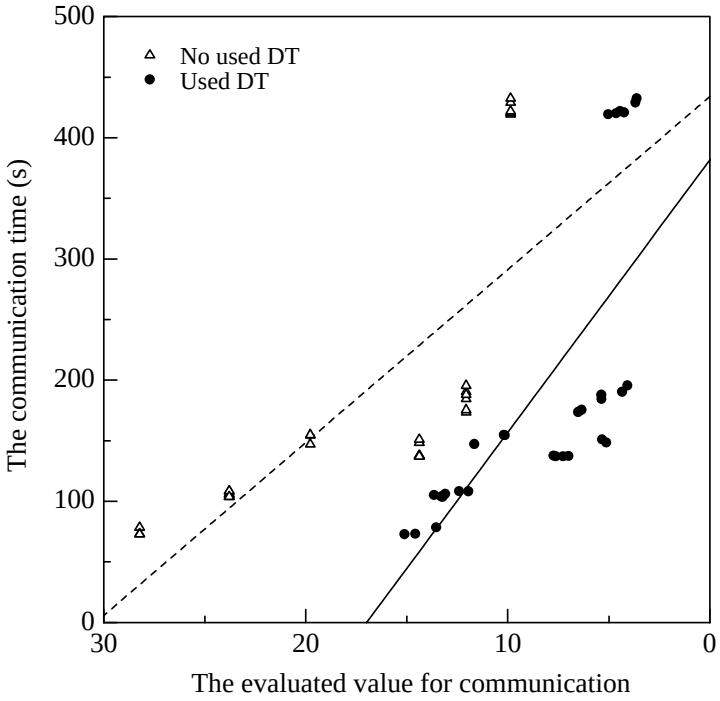

図 4.21: Correlation between $Cost_{comm}$ of asynchronous communication and $Ptime$ on the Cluster2

場合 -0.73，用いない場合 -0.77 が得られた．同期通信の時と同様に，パラメータ DT を用いた場合，総じて実際の通信時間との相関係数が高くなった．また，同期通信を用いた場合よりも，非同期通信を用いた場合の方がより高い相関関係が得られ，データ通信予測の精度が高くなった．また，Cluster2 では，DT を用いた方が多少相関係数の低下を招いた．

分割パターンによる並列性能

表 4.3，4.4，4.5 に CFD シミュレーション時間において，最も高性能な分割パターンと予測された最も良い分割パターンを示した．表中の予測値における経過時間は，予測値の分割パターンを用いた場合における，実際の総経過時間を示した．

表 4.3: Result on the T3E

For synchronous	Grid Type	Partition pattern	Elapsed Time (s)
Better pattern	Type-A	(4,4,8)	34.063
Evaluated better	Type-A	(8,4,4)	34.451
For asynchronous	Grid Type	Partition pattern	Elapsed Time (s)
Better pattern	Type-B	(1,16,8)	25.695
Evaluated better	Type-B	(1,16,8)	25.695

表 4.4: Result on the SP

For synchronous	Grid Type	Partition pattern	Elapsed Time (s)
Better pattern	Type-B	(1,4,2)	542.704
Evaluated better	Type-B	(1,4,2)	542.704
For asynchronous	Grid Type	Partition pattern	Elapsed Time (s)
Better pattern	Type-B	(1,8,1)	473.654
Evaluated better	Type-C	(1,1,8)	473.998

T3E において，表 4.3 から同期通信において最適分割パターンの総経過時間と予測値による分割パターンは約 0.4 秒の差があらわれ，最適分割パターンは予測値の第 3 位であった．非同期通信では，最適分割パターンと予測値の分割パターン

表 4.5: Result on the Cluster2

For syncronous	Grid Type	Partition pattern	Elapsed Time (s)
Better pattern	Type-B	(1,8,1)	118.956
Evaluated better	Type-C	(1,1,8)	118.955
For asynchronous	Grid Type	Partition pattern	Elapsed Time (s)
Better pattern	Type-C	(1,1,8)	118.851
Evaluated better	Type-C	(1,1,8)	118.851

は一致した．同様に SP において，表 4.4 から同期通信では最適分割パターンと予測値の分割パターンは一致した．非同期通信では，最適分割パターンと予測値の分割パターンでは，約 0.3 秒の差があらわれ，最適分割パターンは，予測値の第 2 位でだった．Cluster2 において，表 4.5 から同期通信では最適分割パターンと予測値の分割パターンでは，0.001 秒の差があらわれ，最適分割パターンは予測値の第 2 位だった．非同期通信では最適分割パターンと予測値の分割パターンは一致した．本方法の目的である最適あるいは最適に近い分割パターンを得ることであるため，本方法は十分に効果的である．

4.6 考　察

4.6.1 全体性能

図 4.7，4.8，4.9 および図 4.16，4.17，4.18 から性能予測値の精度を並列計算機ごとについて考える．

T3E の場合，同期通信で相関係数値にブレが見られ，非同期通信では非常に良い相関が得られた．同期通信を用いた場合の通信性能モデルの問題が大きいものと考えられる．通信コストの算出には，通信アルゴリズムと通信ハードウェアの値を用いているが，通信ライブラリの挙動を想定していない．そのため，通信ライブラリがプログラマの意図と異なる動作を行なう場合，予測値を大きくはずす可能性がある．同期通信を用いた T3E の場合，通信コストの算出に同期通信用の $O(log_2(N))$ を用いず，非同期通信用の $O(1)$ を用いた場合，相関係数は -0.94 まで

向上する．T3E の通信ライブラリは，同期通信を用いた場合でも非同期的な挙動を示すと考えられる．そのため性能予測の通信コストは$O(1)$ を用いておけばよい．

SP の場合，同期通信ではデータ通信量が多くなる TypeA 格子の (1,8,1) や (1,1,8) 等の 6 ヶ所が高性能になると予測された．この現象は Cluster2 での同期通信の場合と非同期通信の場合にさらに大きくあらわれた．SP の場合，どちらの通信を用いても十分な相関が得られた．さらに改良するならば，通信コストの産出にクロスバー接続を仮定した値を用いたが，本来多段結合であるため通信コストが $O(1)$ では不十分であり，通信モデルを改良する必要がある可能性が高い．また，T3E と同様に通信ライブラリの挙動の解析も必要である．しかし，SP では通信アルゴリズム通りに同期通信の動作しているように思われる．

Cluster2 の場合，SP と同様に データ通信量が多くなる分割パターンを高性能に見積もる傾向が同期，非同期通信の両方で見られた．この影響のため，Cluster2 での性能予測値と実性能に大きな開きが見られ，相関係数も十分な値が得られなかった．通信量が多い 6 ヶ所の値を省いた場合，同期通信で -0.89，非同期通信では -0.90 が得られた．IP 接続された PC Cluster システムでは，データ通信量が多い場合に必ずしもプログラマの予測した挙動を示さないことが多い．また，スイッチ Hub の性能も大きな要因である．高価なインテリジェント Hub では，4 分割を超えた場合必ず，O(2) 程度の値を示すが，安価なブリッジ Hub では，4 分割以上で O(2) 以上に成る場合がある．また，高価な Hub でもデータ量の少ないうちは，かなり大きな変動が見られることが報告されている [22]．Cluster2 においてデータ通信量が多くなった場合に，性能予測値よりも非常に低性能な結果が得られた背景には，IP 接続とスイッチ Hub の影響が大きい可能性が高い．しかし，これらを通信性能モデルとして取り込むには，非常に複雑な問題のため実用的ではない．実用的な解決方法は，100 Base 等の低速ネットワークで接続された並列計算機では，必要以上にデータ通信量が多い場合は省いて考えるべきである．

4.6.2 計算性能とパラメータ *DT*

T3E，SP，Cluster2 のどの場合も計算性能値と実性能の相関値は，並列計算性能予測値と実性能の相関値よりも低い値であった．本研究では，計算性能の予測値には，実計測値を用いているため，非常に正確であるといえる．また，全ての分割

パターンにおいて計算量は全て同量である．しかし，計算性能は，数 10 Mflop/s の差が現れた．計算性能の評価が，非常に重要であることは明らかである．理想的には，計算性能も計算要素の理論値を用いることで，数値的に計算できることが望ましい．ベクトル計算機での性能評価においては，比較的良好な性能が得られているが，非常に複雑なパラメータを多数用い，誰もが容易に計算できるものではない．また，キャッシュベースのスカラー計算機で計算性能の挙動を数値的に計算するのは非常に困難である．Wijjingert らは，簡易キャッシュモデルを用いることで，処理の簡易化を計っているが，非常に複雑な手順を踏む必要がある．並列計算処理における計算性能を正確に予測するには，本研究で用いたようにプログラムのコア部分の計算を PE 数に分割して，実際に実行することが最も容易で確実な方法である．

本研究では，通信性能評価の精度向上のため，通信のための対理論性能評価値 DT というパラメータを導入した．T3E と SP において，DT は予測どおり通信評価値と実際のデータ通信時間の相関を高める効果を発揮した．DT は，計算性能評価と通信性能評価の両方に有効に働いた．しかし，Cluster2 においては，DT の値は有効に働かず，通信性能を実際のデータ通信時間に近づける効果が少なかった．Cluster2 での DT の大きさを見た場合，T3E や SP とオーダーの違いは無い．この場合もデータ通信量が多い 6 ヶ所の性能評価の問題が大きい．そして，データ通信量が比較的少ない場合の予測精度はかなり高い．そのため，あるデータ通信量を越えた場合指数関数的にデータ通信負荷が高くなることがあることが考えられるが，実用上前述のデータ通信量が多い場合を省くことで対応は可能である．

4.6.3 領域分割パターンの決定

本手法の目的は，最も処理性能の落ちが低い分割パターンあるいはその分割パターンに近い値を機械的に容易に決定することが目的である．以上の結果からその機能は十分に果たすことは明らかである．この方法は，Jacobi 法以外のソルバーに対する効果については未知数である．しかし，CFD 計算シミュレーションの性能予測に関しては，同一の評価手法を用いることで，十分に対応可能であると考えられ，Jacobi 法を SOR 法にしても，Multi Grid(MG) 法にしても同一評価を行なうことで，最適と考えられる分割パターンを得ることは十分に可能であると思

われる.

第 5章

通信隠蔽処理

本章では，MAC 法を用いた並列 CFD 計算の通信処理を積極的に削減する方法である非同期通信-計算重複法について述べる．

5.1 通信隠蔽処理の概要

大規模な並列 CFD シミュレーションを行なうためには，効率的な並列アルゴリズムと並列プログラミングが重要な課題となっている．効率的に並列計算を行なうには，通信のオーバーヘッドをいかに少なくするかが非常に重要である．通信オーバーヘッドを少なくする手段として，計算処理時間に通信処理時間を重ね合わせて，総経過時間に対する通信処理時間を仮想的に減少させる通信隠蔽処理法がある．通信隠蔽処理法は，通信処理と計算処理が独立に行なえる場合に適用できる．

通信隠蔽処理法には，パイプライン処理法と非同期通信-計算重複法の 2 つが考えられる．パイプライン処理は NASA Ames Research Center で開発された NAS Parallel Benchmark (NPB) のアプリケーション LU で Symmetric Successive Over Relaxation (SSOR) 法に用いられている [7]．NPB LU [23] のように圧縮性 NS 方程式を解く場合には計算粒度が大きく通信機構の性能が多少低くとも有効であるが，通信機構の性能が低い場合や計算粒度が非常に小さい場合，並列計算性能が低下する [24]．特に MAC 法での最も大きな計算負荷部分である Poisson 方程式では，計算粒度が小さいため性能低下は著しい [25]．

計算粒度が非常に小さな場合や通信機構の性能が低く，パイプライン処理では

並列化効率が低下する場合でも，並列化効率の低下を防ぎ通信隠蔽を行なう方法として非同期通信-計算重複法がある．非同期通信-計算重複法は，領域分割が可能で領域間のデータ依存性が少ない場合やデータ依存が領域境界部分に集中する場合に特に有効である．Quinn ら [3] は，2 次元モデルについて自動並列コンパイラ技術を用いた非同期通信-計算重複法の有効性や性能限界を理論的に示した．さらに，2 次元モデルについて，Fink ら [4] や田中ら [5] は，SMP クラスタを用いた研究を行なっている．しかし，3 次元モデルにおける非同期通信-計算重複法の有効性や問題点は，明らかにされていない．また，通信隠蔽処理の CFD への応用研究は，以下のような研究がある．Evans ら [26] は，自動並列化ツールを用いたパイプライン処理による通信隠蔽のプログラミングを示し，NPB LU 等で良好な並列化効率を示した．しかし，非同期通信-計算重複法を実際の CFD に応用した例はほとんどない．

非同期通信-計算重複法は，Successive Over Relaxation (SOR) 法等の計算順序に依存する反復法に適用した時，計算順序を変更するため，収束に影響を与える場合がある．一方で収束性は，流体の物理状態や計算領域によっても大きく変化する．このため，まず収束性を考慮せず並列計算性能のみを議論するため Jacobi 法を用いた．Jacobi 法は，計算順序の変更や領域の分割パターンによる収束性の変化はない．また，この方法は緩和法の基本的な形態であり，収束を速めた様々な緩和法の並列性能を類推できる．

5.1.1 非同期通信-計算重複法

処理手順

並列 CFD 計算に適用した非同期通信-計算重複法は，領域の内部を図 3.2 左に示す 3 つの部分に分けて考える．小領域の境界領域 (*Boundary region*) の計算は，重複領域 (*Overlap region*) と領域内部 (*Non-boundary region*) を用いる．境界領域の値が確定した時，領域内部の計算は，小領域間の依存関係がなくなるため小領域で個別に実行できる．

非同期通信-計算重複法の手順は，最初に境界領域の一対の面 (例えば左右の面) を計算し，その結果を隣接領域へノンブロッキング通信する．この通信開始直後

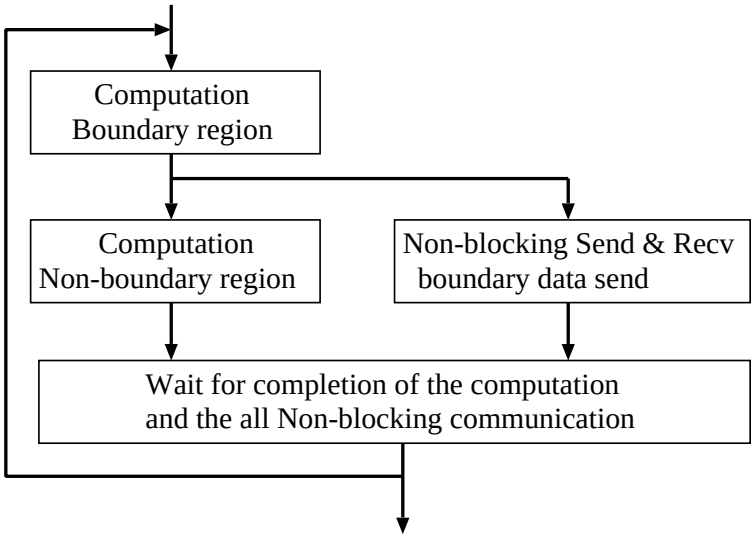

図 5.1: The systolic communication-computation overlap Method

に，2 次元以上の分割では，次の面 (例えば上下の面) を計算し，その結果を隣接領域にノンブロッキング通信する．この通信処理と平行に領域内部を計算する．領域内部の計算終了後，通信が終了していれば次反復の境界領域の計算を開始し，通信が終了していなければ待ち状態となる．Jacobi 法の場合，データの送信先 (受信バッファ) に気をつける必要は無いが，SOR 法等のようにデータ依存性が高いソルバーを用いる場合は，データ受信先の配列は，直接物理量の配列に対する実行は不都合が多い．そのため，受信バッファを別途用意し，受信終了後に物理量配列にコピーした方がよい．図 5.1 に計算の流れの概要を示した．そして，計算手順の詳細を以下に示した．

非同期通信-計算重複法の 2 つのケース

理想的な非同期通信-計算重複は，通信処理時間の全てが計算処理時間に覆い隠される．非同期通信-計算重複法の効果は，以下のモデル式が成り立つことで明らかになる．モデル式は，MPI の待ち処理 (MPI_Wait) の影響を仮定した．

非同期通信-計算重複による総経過時間の短縮には 2 つの場合がある．内部領域の計算時間が通信時間よりも長い場合 (通信圧縮，5.2 左 $overlap\ comm$) と通信時間が内部領域の計算時間よりも長い場合 (計算圧縮，5.2 中 $overlap\ comm$) である．

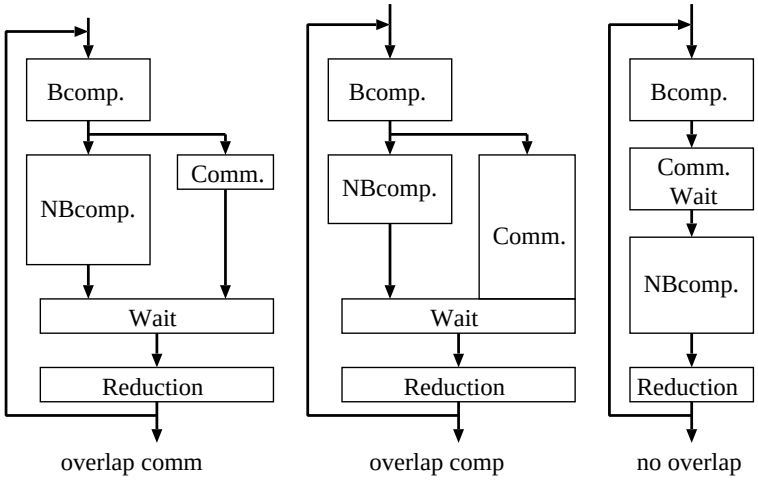

図 5.2: The overlap communication(left), the overlap computation(middle) and the no overlap(right)

通信圧縮と計算圧縮の判定は，非同期通信-計算重複法と同一の計算手順を用いるが，非同期通信-計算重複を行なわない並列計算方法 (非圧縮，5.2 右 $no\ overlap$) を用いる．非圧縮の総経過時間 (T_n) を次式に示す．

$$T_n = C + M + R \tag{5.1}$$

ただし，C は計算時間を示し，$C = C_B + C_{NB}$，C_B は境界領域の計算時間 (5.2 BComp)，C_{NB} は領域内部の計算時間 (5.2 NBcomp) である．M は通信時間 (5.2 Comm+Wait) を示し，R は，誤差評価の縮約処理時間 (5.2 Reduction) を示す．

通信圧縮の総経過時間 (T_m) は次のようになる．

$$T_m = C + W + R \tag{5.2}$$

W は非同期通信のための待ち処理 (MPI_Wait) の時間 (5.2 Wait) を示す．W は通常明確にならないが，C_{NB} が M より十分に大きい場合は待ち処理のみの時間として扱える．計算圧縮の総経過時間 (T_p) は次のようになる．

$$T_p = C_B + M + R \tag{5.3}$$

式 (5.2)，(5.3) の関係が成り立つ場合，非同期通信-計算重複法によってデータ通信時間または内部領域の計算処理時間が圧縮され，総経過時間が短縮する．

モデル式の適用は，非圧縮を計測し，通信圧縮と計算圧縮を判断する．非同期通信-計算重複法を適用した並列 CFD 計算プログラムの各成分の経過時間を計測した．

5.1.2 ベンチマーク問題での評価

非同期通信-計算重複法の効果を境界適合格子の MAC 法で用いる Poisson 方程式ソルバーのみを用いて検討した．評価において，まず並列処理性能のみを検討した．簡単化のために反復回数を 200 回に固定し，境界条件はディレクレ，残差処理の縮約通信は行なわなかった．そして，時間計測は次のように行ない，用いた格子数は表 5.1 とした．また，用いた分割パターンを表 5.2，5.3，5.4 に示した．分割パターンは，前章で示した方法を用いた．通信隠蔽を行なう場合，計算性能値の計測は通信隠蔽処理に伴う計算手順と同等に境界部分を先に計算する場合の計算手法を計測した．

$Etime_p$: 並列処理におけるサブルーチン処理の総経過時間
$Ctime_p$: 並列処理におけるサブルーチン内部の計算処理の経過時間
$Etime_s$: 逐次処理におけるサブルーチン処理の総経過時間
$(= Ctime_s)$

$Etime_s$ は，Jacobi 法サブルーチンの逐次処理時間を計測したもので，$Ctime_s$ と等しい値になる．$Etime_p$ は，各 PE で異なるが，最も大きい $Etime_p$ を代表値とした．$Ctime_p$ も各 PE で異なるが，$Etime_p$ を測定した PE の $Ctime_p$ を代表値とした．$Etime_p - Ctime_p$ は，通信処理や同期待ち時間等となる．

表 5.1: Grid size

	Grid size	Number of grid points
Type A	$130 \times 66 \times 66$	566,280
Type B	$66 \times 34 \times 34$	76,296

表 5.2: Domain partitioning patterns for T3E(left:TypeA,right:TypeB)

PE	patterns	PE	patterns
2	2,1,1	32	2,2,8
4	2,1,2	64	2,4,8
8	2,2,2	128	4,4,8
16	2,2,4		

PE	patterns	PE	patterns
2	1,1,2	32	2,4,4
4	1,2,2	64	4,4,4
8	1,2,4	128	4,4,8
16	1,4,4		

表 5.3: Domain partitioning patterns for SP(left:TypeA,right:TypeB)

PE	patterns	PE	patterns
2	1,1,2	16	1,4,4
4	1,2,2	32	1,4,8
8	1,2,4		

PE	patterns	PE	patterns
2	1,1,2	16	1,2,8
4	1,2,2	32	2,2,8
8	1,2,4		

表 5.4: Domain partitioning patterns for Cluster(left:TypeA,right:TypeB)

PE	patterns	PE	patterns
2	2,1,1	8	4,1,2
4	2,2,1		

PE	patterns	PE	patterns
2	2,1,1	8	4,1,2
4	4,1,1		

5.1.3 結　果

使用した並列計算機は，T3E，SP，Cluster1 である．また，SP では，1 プロセスに 1 ノード用いた．T3E では 128 PE，SP では 32 PE 用い，Cluster1 では 8 PE 用いた．

速度向上比

速度向上比 (Speed up ratio) を

$$Speed\ up\ ratio = \frac{Etime_s}{Etime_p} \tag{5.4}$$

とした．図 5.3 に T3E の速度向上比，図 5.4 に SP の速度向上比，図 5.5 に Cluster1 の速度向上比を示した．これ以降，通常の通信を隠蔽しないアルゴリズムを *ordinary*，非同期通信-計算重複法を *overlap* とする．

SP での最大の速度向上比は，32 PE で Type A の場合 *overlap* で 23.6 倍，*ordinary* で 23.6 倍が得られた．また，Type B の場合 *overlap* で 21.9 倍，*ordinary* で 20.5 倍の速度向上比が得られた．Type B の 32 PE の速度向上比は，*overlap* が *ordinary* に対して 7 % 上回った．しかし，Type A の 8，16 PE の速度向上比は，逆に *ordinary* が *overlap* よりも大きな値を示した．その他の場合では *overlap* と *ordinary* の差はほとんどなかった．

Cluster1 での最大の速度向上比は，8 PE で Type A の場合 *overlap* で 9.5 倍，*ordinary* で 8.1 倍が得られた．また，Type B の場合 *overlap* で 8.4 倍，*ordinary* で 5.3 倍の 速度向上比が得られた．Type A，B ともに非同期通信-計算重複法の速度向上比は大きくなり，特に計算粒度が小さい場合の *ordinary* との差は，8 PE の Type A の場合でも 17 % ，Type B の場合では 56 % の性能差が得られた．また，PE を増加させても，問題量に関わらず *overlap* は，直線的に速度向上比が増加しているが，*ordinary* では，8 PE で早くも速度向上比に飽和の兆しが現れた．

T3E での最大の速度向上比は，128 PE で Type A の場合 *overlap* で 111.2 倍，*ordinary* で 95.5 倍が得られた．また，Type B の場合 *overlap* で 87.3 倍，*ordinary* で 71.1 倍 の速度向上比が得られた．T3E においては，*overlap* は *ordinary* より低い速度向上比しか得られなかった．特に 32 PE から 64 PE の間において，*overlap* において大きな低下が見られた．このため，T3E では非同期通信-計算重複法が有

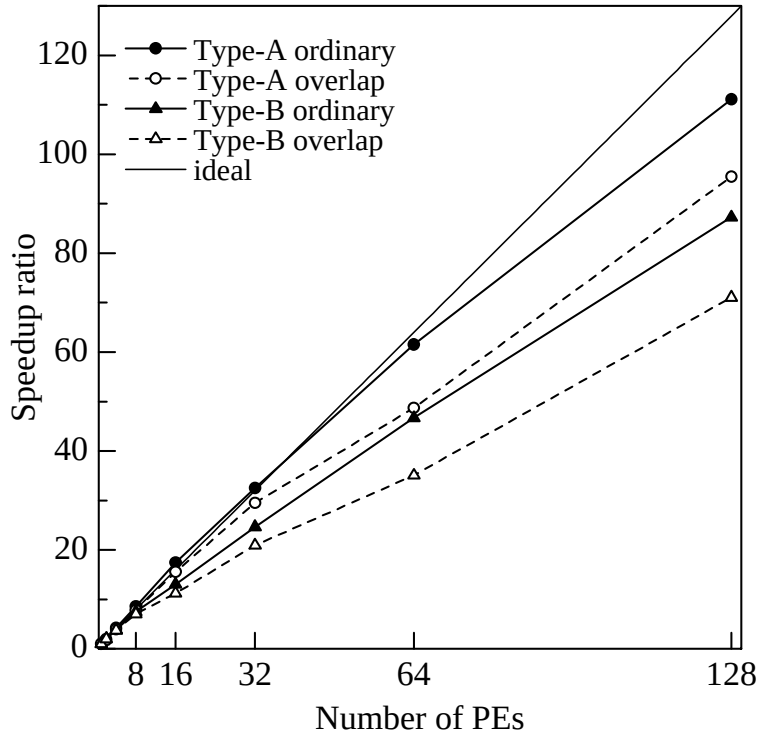

図 5.3: Speed up ratio on the T3E

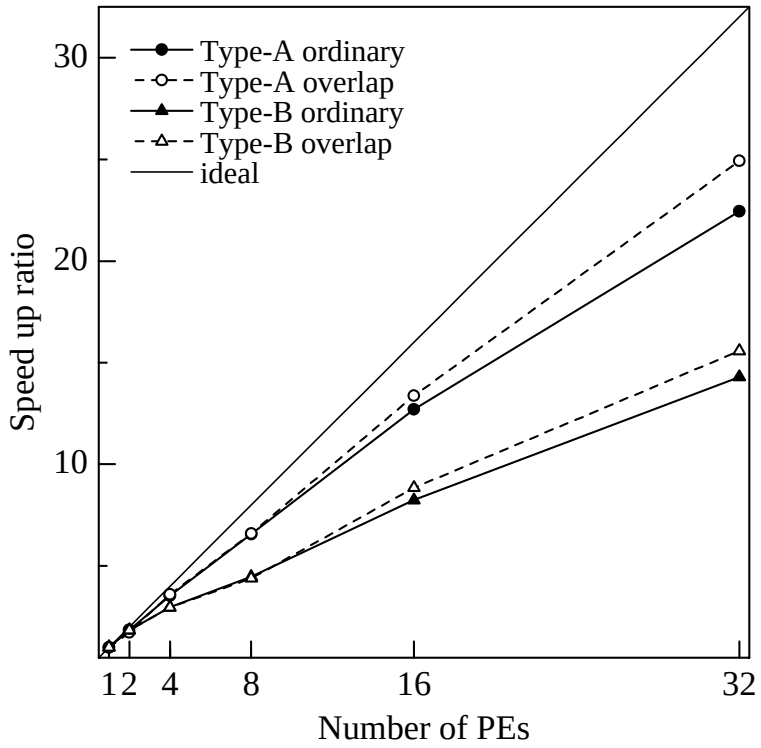

図 5.4: Speed up ratio on the SP

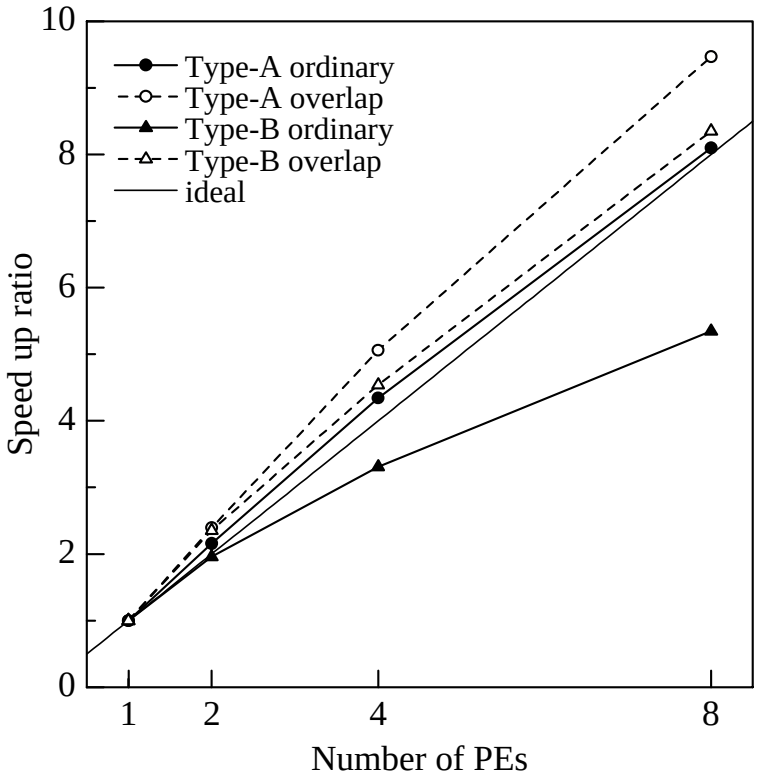

図 5.5: Speed up ratio on the Cluster1

効な方法ではないと考えられる.

通信処理の影響

並列処理における通信コストは，並列処理時の経過時間 ($Etime_p$) と計算処理の経過時間 ($Ctime_p$) の差，すなわち通信処理による付加的時間 (データ通信時間及び同期待ち時間) を $Etime_p$ で割った無次元量 (式 5.5) で評価される場合が多い.

$$\frac{Etime_p - Ctime_p}{Etime_p} \tag{5.5}$$

しかし，この値は今回のように同一 PE 数で $Etime_p$ が異なる $overlap$ と $ordinary$ の比較には適切ではない．例えば，通信による付加的時間が同じでも $Etime_p$ が大きければ，通信コストが小さいと誤って評価される．そこで，分母の $Etime_p$ の代わりに，逐次処理時の経過時間 $Etime_s$ を PE 数で割った値 ($Etime_s/PE$) で置

き換えた通信負荷比 C_a を用いた (式 5.6).

$$C_a = \frac{Etime_p - Ctime_p}{\frac{Etime_s}{PE}} \tag{5.6}$$

このように定義された C_a は，$overlap$ と $ordinary$ の通信負荷を絶対的に評価することができる．C_a が大きいと通信負荷が大きくなり，C_a 値は 1 を超えることがある．

図 5.8 に Cluster1 の通信負荷比を示した．Cluster1 は，データ通信処理の負荷が高く，通信隠蔽の効果は非常に大きい．特に計算粒度が小さく，通信が大きな割合を占める場合に通信隠蔽の効果が高い．PE 4 と 8 において，Type A では，ほとんど C_a の値が変わっていない．Type B でも，僅かに上昇した程度であった．通信負荷の多くは隠蔽あるいは短縮された．

図 5.7 に SP の通信負荷比を示した．$overlap$ の C_a 値は，$ordinary$ に比べて，格子サイズと PE 数に関わらず低い値を示した．このような結果から $overlap$ の通信負荷が減少したことになる．そのため，Type B では，PE 数が少ない時でも大きな差があらわれ，Type A では，16 PE の時に最大になった．しかし，Type A の 32 PE の場合では，$overlap$ と $ordinary$ の差はほとんどなかった．Type A で PE 数が少ない場合の $overlap$ と $ordinary$ の差は小さい．

図 5.6 に T3E の通信負荷比を示した．Type A において $overlap$ の C_a 値は $ordinary$ に比べて，低い値を示した．また，Type B においては 4, 8 PE において低い値を示した．しかし，Type A では PE 数の増加とともに差が小さくなった．Type B でも多くの場合 $ordinary$ での値の方が小さくなった．通信隠蔽の効果であるデータ通信時間の短縮という意味において，T3E においてもその目的を達していることが分かった．

計算効率

通常並列計算における計算効率の評価は並列計算効率 (式 5.7) を用いることが多い．

$$Parallel\ Efficiency = \frac{Etime_s}{Etime_p \times PE} \tag{5.7}$$

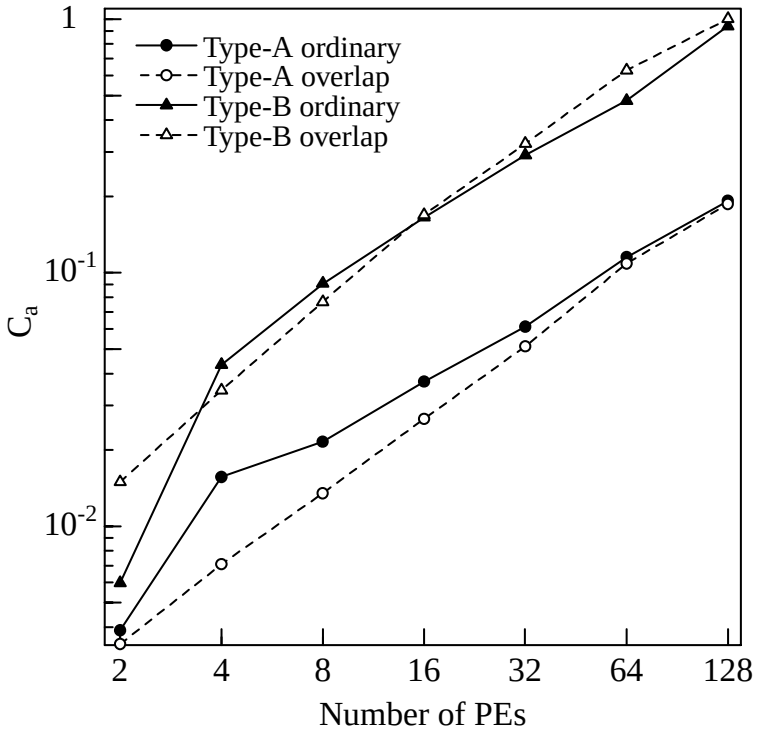

図 5.6: C_a on the T3E

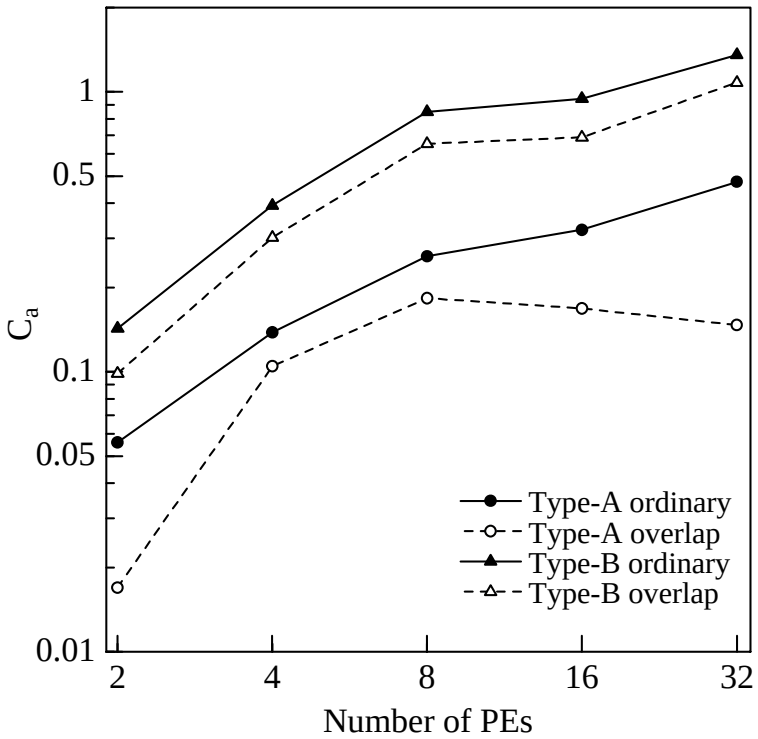

図 5.7: C_a on the SP

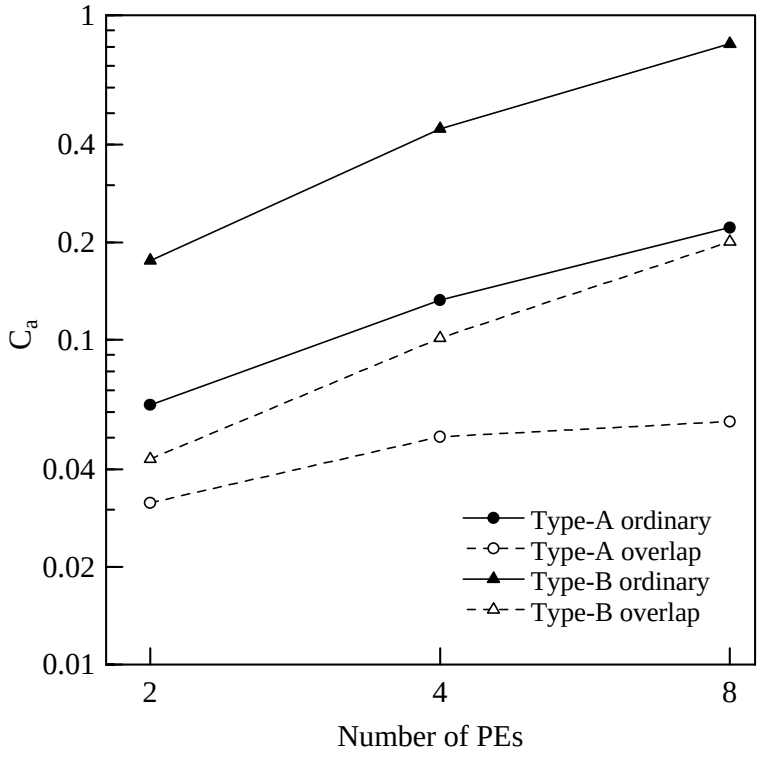

図 5.8: C_a on the Cluster1

この値は，通信負荷量を含んだものであり，通信隠蔽を行なう場合の計算効率を評価するには不適当である．なぜなら，通信隠蔽における通信処理時間と計算処理時間は重複して実行されている．そのため，それらを重ね合わせた値を評価に用いれば，通信処理時間 $>$ 計算時間であるのか，通信処理時間$<$ 計算時間であるのかが不明瞭であり，式 (5.7) で得られた値では，計算処理時間と通信処理時間のどちらの影響が大きいのか誤解を与える可能性がある．そのため，並列処理性能での計算処理時間の影響のみを評価する必要がある．1 PE あたりの計算効率の評価には，$Etime_s$ を $Ctime_p$ に PE 数を掛けたもので割った無次元量 (式 5.8) C_E (Computational Efficiency) を用いた．

$$C_E = \frac{Etime_s}{Ctime_p \times PE} \tag{5.8}$$

この値は，逐次処理時間と並列計算時の計算処理時間の比を示す．そのため，C_E が 1 ならば並列計算時の計算効率が，逐次処理時の計算効率と同等である．

図 5.9，5.10，5.11 にそれぞれ T3E，SP と Cluster1 での C_E を示した．

T3E では，格子サイズと PE 数に関わらず，$overlap$ は $ordinary$ よりも低い値

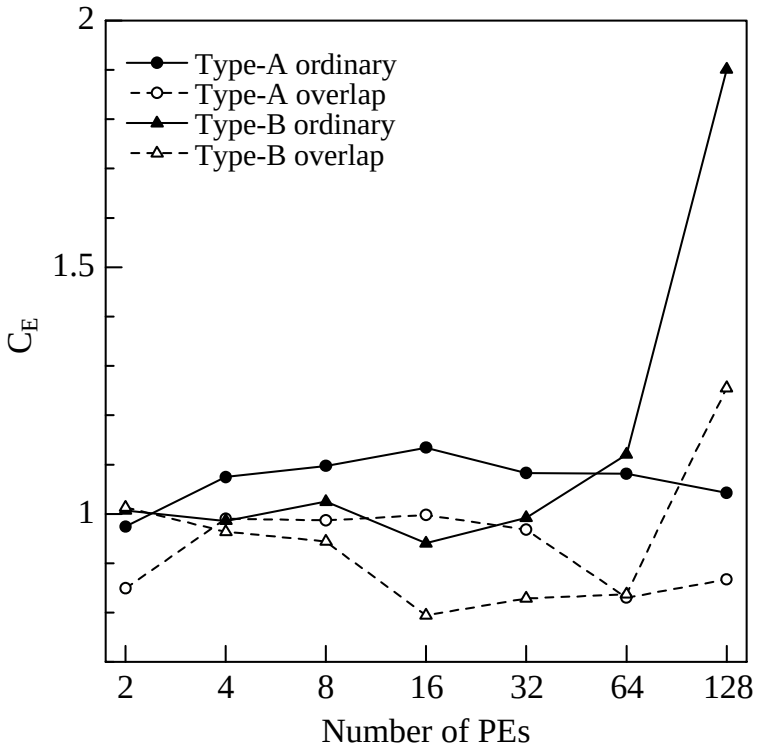

図 5.9: C_E on the CRAY-T3E

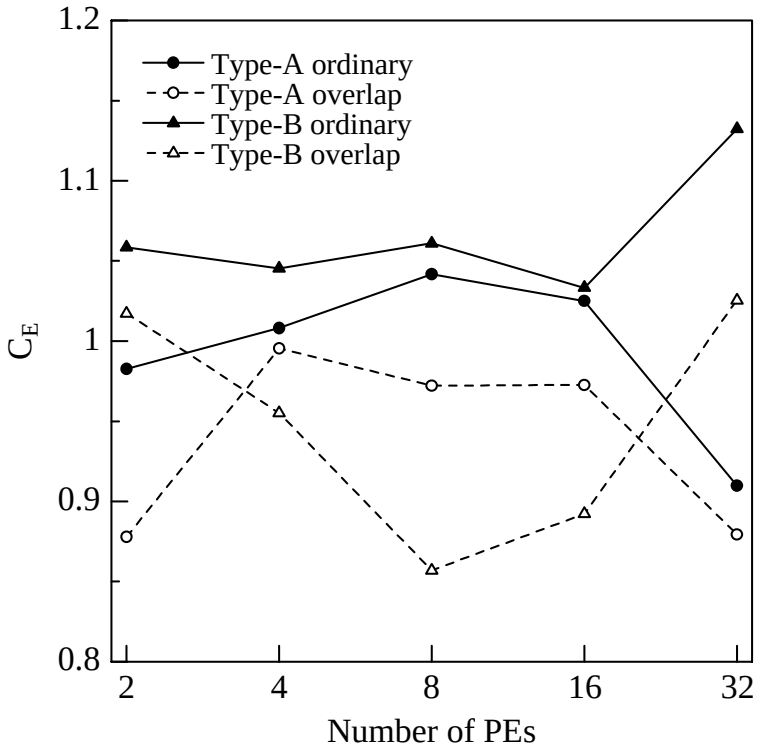

図 5.10: C_E on the RS/6000 SP

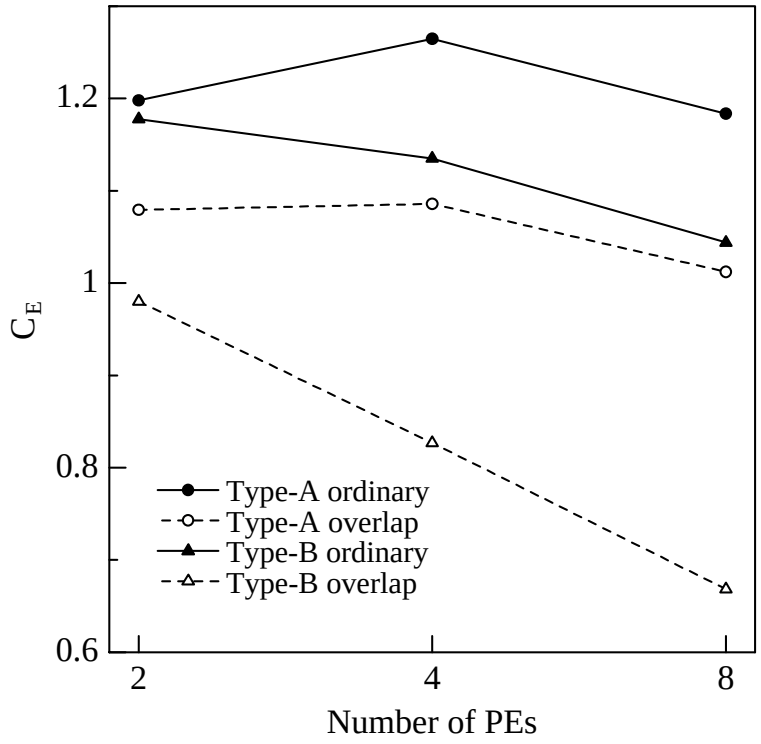

図 5.11: C_E on the PC Cluster

であった．計算効率数値変動の傾向は，*overlap* と *ordinary* は大きく変わらなかった．しかし，効率差は，PE 数が少ない場合，10 % 程度，PE 数が大きい場合 30 ~ 60 % 程度低い計算効率だった．Type B の 128 PE では，計算効率がかなり高い値を示した．

SP では，格子サイズと PE 数によって計算効率が大きく変化した．そして，*overlap* は *ordinary* よりも計算効率が低かった．計算効率の数値変動は，*overlap* において Type A で 10 % 程度，Type B で 20 % 程度の値だった．*ordinary* において Type A で 10 % 程度，Type B で 20 % 未満程度だった．傾向は T3E と同様の傾向が見られた．

Cluster1 では，格子サイズ，PE 数に関わらず，計算効率が，1.0 を越える値を示した．Type A と Type B で異なる現象を示した．Type A の場合，4 PE から 8 PE では効率が下がる傾向が見られたが，Type B では *overlap* と *ordinary* で傾きは異なるが計算効率は向上した．RS/6000 SP で見られたような *overlap* での計算効率の低下は Type A の 8 PE の場合と Type B の 4，8 PE の場合に見られた．

考　察

図 5.5 から見られるように，通信隠蔽は，計算粒度が小さくなる時に有効性が高い．特に 100 M bps Switch-HUB ネットワークで構成された Cluster1 で高い性能が得られたことは，現在コストパフォーマンスが最も高い 100 M bps のネットワーク機器を用いた PC Cluster で，多数の PE を接続したシステムでも通信コストの増加による速度向上比の低下が押さえられることを示している．また，中速ネットワークで構成された図 5.4 の SP でも PE 数が増加した場合に同様なことが言え，ギガビットネットワークを用いた PC Cluster でも同程度の通信性能を有するため同じことが言える．しかし，図 5.3 の T3E に見られるように通信性能が高いシステムにおいては，通信隠蔽処理が並列計算性能を向上させず，低下させることになった．このため，Jacobi 法を用いた通信隠蔽処理は，高速ネットワークを用いた場合にあまり効果があらわれないことが考えられる．図 5.5 の *ordinary* と *overlap* ではスーパーリニアの速度向上比が得られ，図 5.11 から PC Cluster では，格子サイズや *overlap*，*ordinary* に関わらず 1 PE が処理する小領域の計算効率は非常に高くなった．計算効率が高くなる原因は，様々考えられるが，一つにはキャッシュの影響が考えられる．領域分割によってデータのブロック化に近い影響があらわれることや，1 PE で保持する配列サイズが領域分割によって異なり，コンパイラの最適化が効率よく働いたものと考えられる．文献 [5] でも Laplace 方程式解法では，32 CPU で約 45 倍というスーパーリニアが得られている．この場合は，計算量が非常に小さくなり，キャッシュ内に全てが収まって計算性能が飛躍的に向上したと考えられる．図 5.8 から *ordinary* に対して *overlap* の C_a が小さいので通信隠蔽の効果が高い．特に Cluster1 で用いられた 100 M bps のネットワーク機器は通信性能が低いので，C_a の差が大きく通信隠蔽の効果が非常に高い．図 5.5 の PC Cluster における *overlap* と *ordinary* の速度向上比の差があらわれる最も大きな理由は通信隠蔽の主目的である通信時間の削減が有効に働いた結果であり，図 5.4 の SP での結果からも通信性能がギガビットクラスになっても十分に有効な結果が得られると考えられる．

図 5.6 から *ordinary* に対して *overlap* の C_a が小さいので通信隠蔽の効果はある．しかし，T3E における速度向上比は多くの部分で *ordinary* よりも *overlap* が低い値を示し，通信隠蔽処理の効果が実際の性能として現れない．通信隠蔽処理

による計算処理性能の低下は，SP，Cluster1 の場合にも見られ，計算順序の変更の影響があらわれたものと考えられる．全体性能として通信隠蔽の効果は見られないが，通信処理の負荷を見る限り，通信処理時間を減少させるという通信隠蔽の効果ははっきりしている現象が見られた．最も大きな原因は，図 5.9 に見られるように T3E では，計算処理性能の低下が主要因である．このため，通信隠蔽処理は通信処理性能が主要因の通信隠蔽の効果と計算処理性能の効果との対効果比が重要である．本研究の結果では，T3E において通信隠蔽の効果があらわれず，計算処理性能の低下による影響の方が大きい．この結果から CPU 処理性能が十分に高くなれば，このような結果は起こらないと考えられ，PC Cluster のように高速な CPU と比較的低性能な通信性能を組み合わせる計算機では，十分な性能が得られると考えられる．しかし，現在のベクトル型並列計算機では，計算順序の変更による計算性能の低下が考えられ，超高速ネットワークを用いているため，通信性能が非常に高いため，通信隠蔽の効果は現れないと予測される．

本研究の計算処理は，3 次元の境界適合問題を対象としているため，計算量が比較的多いため，100 Mbps のネットワークを用いた場合にも 1 PE あたり Type B の 1/8 の計算量があれば通信隠蔽の効果は十分にある．文献 [5] のような，高性能な通信機構を用いれば，2 次元の Cartesian 格子系のような非常に計算粒度が小さな問題の 32 CPU でも通信隠蔽が十分に有効であることが示されている．さらに，計算量が小さな場合，大容量キャッシュの搭載が現在のスカラ型計算機の傾向を見る限り，キャッシュの効果がさらに有効となり，スーパーリニアがあらわれる可能性が高い．また，境界部分を先に計算することによる計算性能の低下が，キャッシュが大きくなれば，この性能低下を抑えることが出来ると考えられる．また，T3E，SP，Cluster1 の全ての並列計算機において通信隠蔽処理による計算性能の低下が発生した．T3E では，計算性能低下が致命的で通信隠蔽処理の効果が相殺された．そのため，計算性能の評価が通信隠蔽処理における全体性能の評価に大きく影響すると考えられる．通信隠蔽処理での分割パターンの設定において，通信隠蔽による計算パターンを採用したことでかなり正確な予測であり，分割パターンは妥当な結果である．

5.2 CFD シミュレーションでの性能向上

Poisson 方程式解法での性能を踏まえて，非同期通信-計算重複法を実際の CFD シミュレーションコードに実装した．前述の結果の通り，T3E では，十分な結果が得られないことが予想されるため，CFD シミュレーションでの実行環境に T3E は含まなかった．

5.2.1 計算対象

非圧縮粘性流体の計算モデルは，図 5.12 に示す 3 次元正方キャビティー流れとした．Re 数は 100 である．ただし，L を正方キャビティーの 1 辺の長さ 1，U を 1，そして ν を 0.01 とした．計測対象は非定常性が強い計算初期の 100 ステップまでとした．

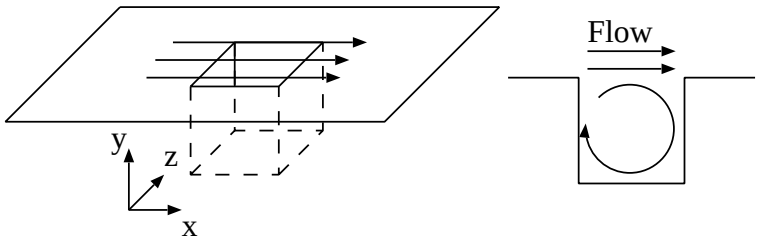

図 5.12: Overview of flow in a driven cabity

計算格子は境界付近で格子点間隔を細かくとる．計算格子の設定は直交不等間隔格子である．方程式は直交不等間隔の定式化ではなく，一般座標変換を伴った定式化を行なった．

検討に用いた計算格子サイズは，表 5.5 に示す 2 種類である．クーラン数は 0.2 とした．どちらの計算格子も最小格子点間隔 (Δx_{min}) は同じとし，時間ステップ幅 Δt も同一である．

表 5.5: Grid system size

	Grid size	Number of grid points
Type A	$26 \times 26 \times 26$	17,576
Type B	$66 \times 66 \times 66$	287,496

5.2.2 計算条件

式 (2.5) の Poisson 方程式の離散化は，2 次精度中心差分を用いた．式 (2.6) の NS 方程式の離散化は，左辺第 2 項の移流項に 3 次精度風上差分を用い，その他の空間微分項は全て 2 次精度中心差分を用いた．また，速度の時間微分項は，1 次精度前進差分を用いた．

図 5.12 において，x 方向の流速を u，y 方向の流速を v，z 方向の流速を w とした．速度の境界条件は，上の境界が $u = 1.0$，$v, w = 0.0$ とし，その他の境界は $u, v, w = 0.0$ とした．圧力の境界条件は，全ての境界で圧力勾配を 0.0 とした．

Jacobi 法の収束判定は，各格子点での残差の最大値が 1.0×10^{-5} になるまでとした．

5.2.3 結果

使用した並列計算機は，SP，Cluster1 である．SP では 32 PE，Cluster1 では 8 PE 用いた．

また，並列計算結果とした時間は，以下の方法で選択した．PE 間で総経過時間が最大である PE を選択し，その PE の総経過時間と各成分の経過時間を代表値として得る．同一プログラムを 3 回実行し，総経過時間が最小である PE の代表値を結果として採用した．

流れ場

本稿で用いたプログラムを検証するため，定常状態まで計算した結果を示した．図 5.16 に Ku らの結果 [27] と表 5.5 の Type A,B の垂直，水平の中心線上の速度プロファイルを示した．Type A では，格子点数が少ないため精度低下が見られた．また，定常状態における 3 次元キャビティー内の流跡線を図 5.13 に示した．図は流れ方向の上流側から下流側へ斜め下に見下ろしたものである．下流側上面部の中心付近の流れが，上流側の壁に近づくにしたがって，側壁方向に移動している様子が見られ，3 次元キャビティー流れの特性が良くあらわれている．また，図 5.14 に空洞中心付近の渦度分布を示し，図 5.15 に等圧力面を示した．

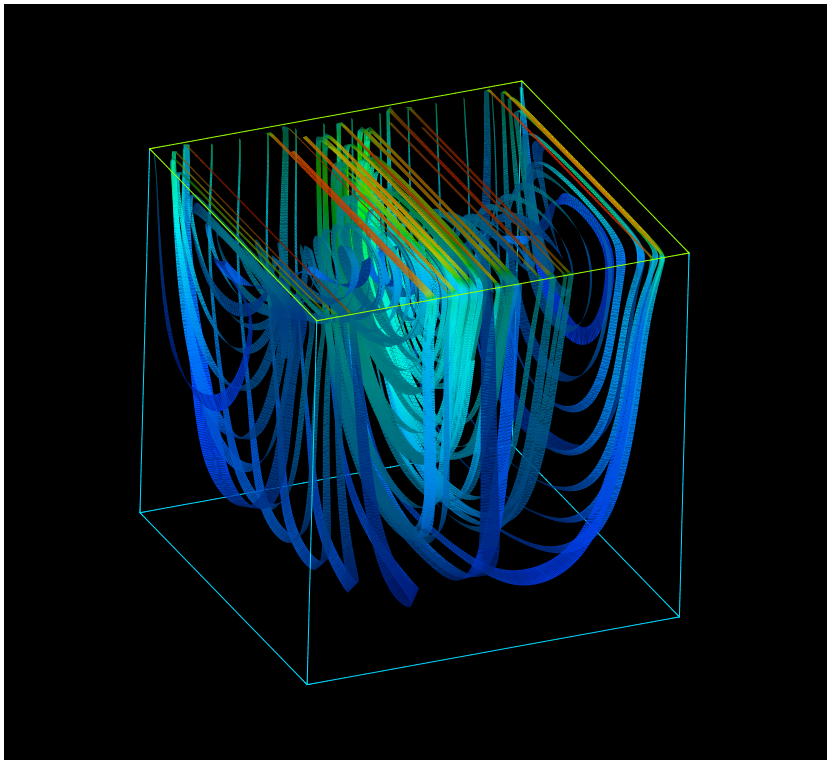

図 5.13: The streamlines for the lid-driven Cubic cavity flow

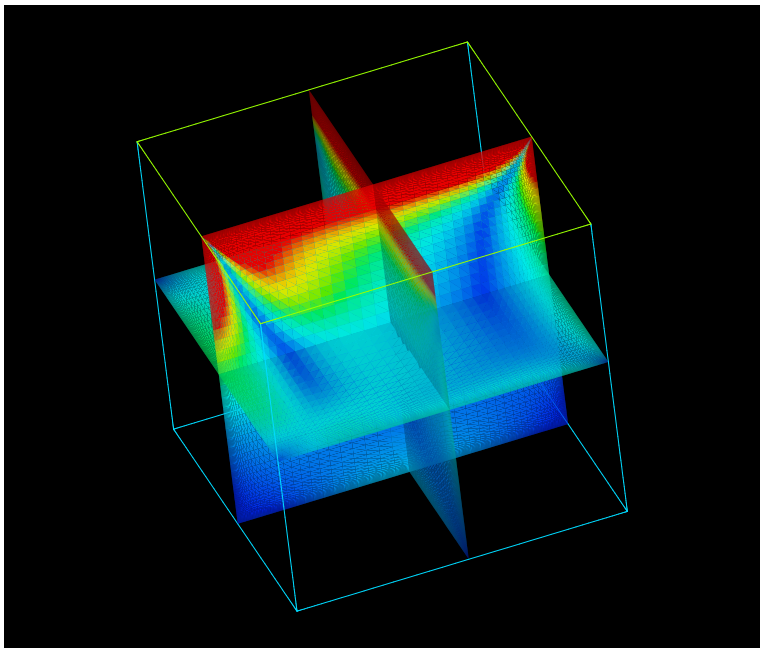

図 5.14: The vorticity contours for the lid-driven Cubic cavity flow

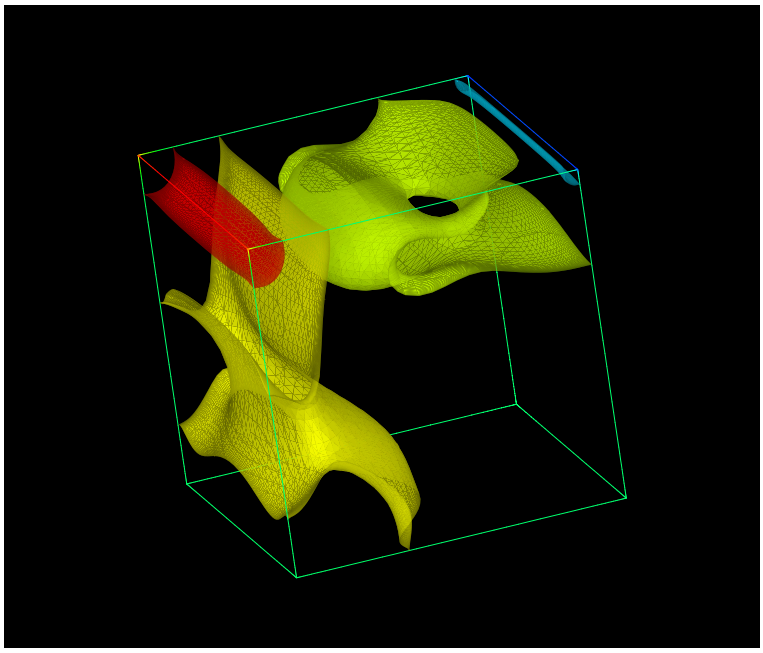

図 5.15: The pressure isosurface for the lid-driven Cubic cavity flow

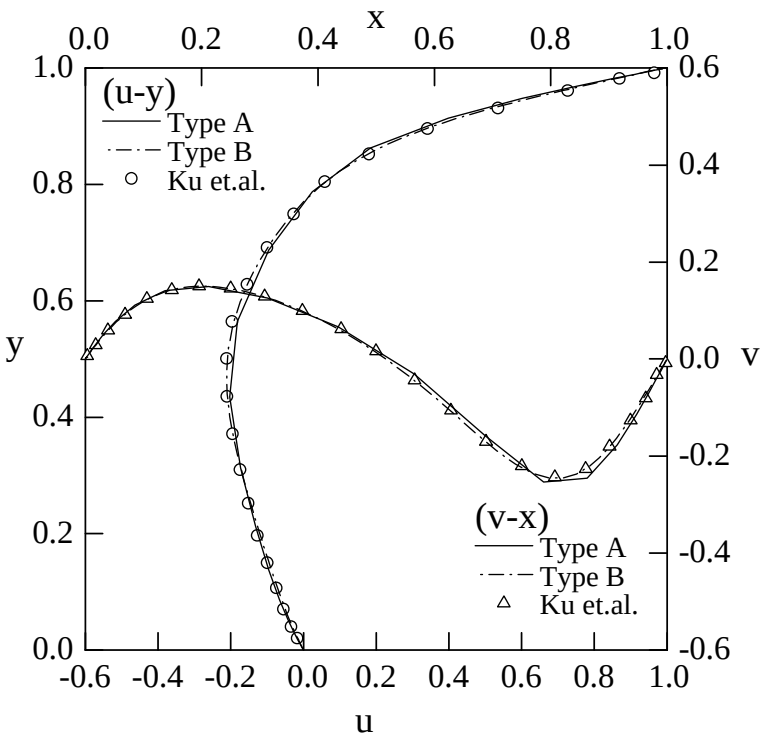

図 5.16: The flow velocity profiles of the cubic cavity on the vertical and the horizontal centerline

計算負荷率

本章の CFD 計算を 1 PE で実行した場合の圧力計算部分と速度計算部分の計算負荷比率を表 5.6 に示した.

表 5.6: Percentages of total elapsed time contributed by the major processes

Process	RS/6000 SP		PC Cluster	
	Type A	Type B	Type A	Type B
Pressure Comp.	99.0	99.2	98.9	98.8
Velocity Comp.	1.0	0.8	1.1	1.2
Total	100.0	100.0	100.0	100.0

CFD 計算の総経過時間は，圧力計算時間でほぼ 100 % を消費している．速度計算時間は 1.0 % 前後である．速度計算に非同期通信-計算重複を行なわなくても総経過時間にはほとんど影響ない.

分割パターン

用いた分割パターン表 5.7(SP) と表 5.8(Cluster1) に示した．分割パターンの決定は，前述の方法を用いた.

非同期通信-計算重複

5.2.2 節で説明した非同期通信-計算重複法の 2 つのケースを調べるため，非圧縮について，領域内部の計算処理時間と通信処理時間を調べた．非圧縮の計算には，表 5.7，5.8 に示された *overlap* の分割パターンを用いた.

図 5.17 は，計測によって得られた C_{NB} と T_m の差を示した.

問題サイズが小さい Type A で SP の 16，32 PE と Cluster1 の 4, 8 PE が計算圧縮になる．また，その他や問題サイズが大きな Type B は通信圧縮になった.

非圧縮の結果から非同期通信-計算重複法を 2 ケースに分け，非同期通信-計算重複法を適用した並列 CFD 計算の各成分の計測を行なった．図 5.18，図 5.19 は，PE 数を変化させた時の通信圧縮の場合 (式 (5.2)) と計算圧縮の場合 (式 (5.3)) の計測を行ない，各成分の経過時間を示した．図には，左側棒グラフに式 (5.2),(5.3) の

表 5.7: Domain partitioning patterns on the SP

PE	Type A		Type B	
	ordinary	*overlap*	*ordinary*	*overlap*
2	1,1,2	←	1,2,1	←
4	1,2,2	←	1,2,2	1,4,1
8	1,4,2	1,2,4	1,4,2	←
16	1,4,4	1,2,8	2,4,2	1,8,2
32	1,4,8	←	2,4,4	1,8,4

表 5.8: Domain partitioning patterns on the Cluster1

PE	Type A		Type B	
	ordinary	*overlap*	*ordinary*	*overlap*
2	1,1,2	←	1,2,1	←
4	1,1,4	1,2,2	1,2,2	←
8	1,1,8	1,2,4	2,2,2	←

左辺 T_m と T_p を，右側棒グラフに式の右辺の各成分を示した．左側棒グラフは，黒色が T_m であり，白色が T_p である．右側棒グラフは通信圧縮の場合，上から R, W, C_{NB} であり，計算圧縮の場合，上から R, C_B, M である．W は MPI_Wait, R は $MPI\_Reduction$ の各サブルーチンの処理時間であり，通信時間 M は，R を除いた総経過時間から計算時間 C を引いた値を用いた．

式 (5.2), (5.3) の非同期通信-計算重複法における仮定である $Wait$ を用いることで性能モデル式が図 5.18，5.19 の各 PE 数における左右棒グラフより成り立つことが分かった．また，通信圧縮と計算圧縮の領域内部の経過時間は，非圧縮の場合と比べ多少遅くなった．非同期通信- 計算重複法では，通信処理と計算処理を同時に行なうためと考えられえる．

SP，Cluster1 ともに待ち処理の時間 $Wait$ が大きいことが分かった．待ち処理時間は，SP と Cluster1 共に PE 数の増加に伴い減少した．一般に PE 間の同期時間は PE 数の増加に従って長くなると考えられる．しかし，本稿では問題サイズに関わらず，PE 数が増加すると，待ち処理時間が減少した．

図 5.20 に示すような，待ち処理 (MPI_Wait) のサブルーチンコールが完了す

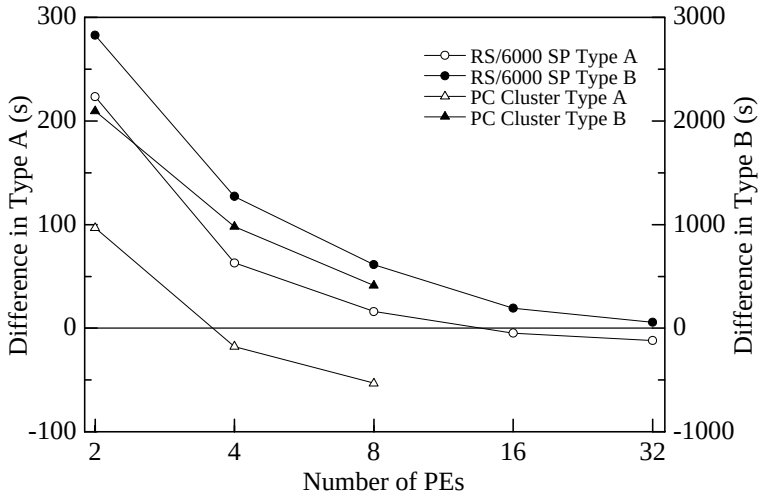

図 5.17: Difference between computation time and communication time

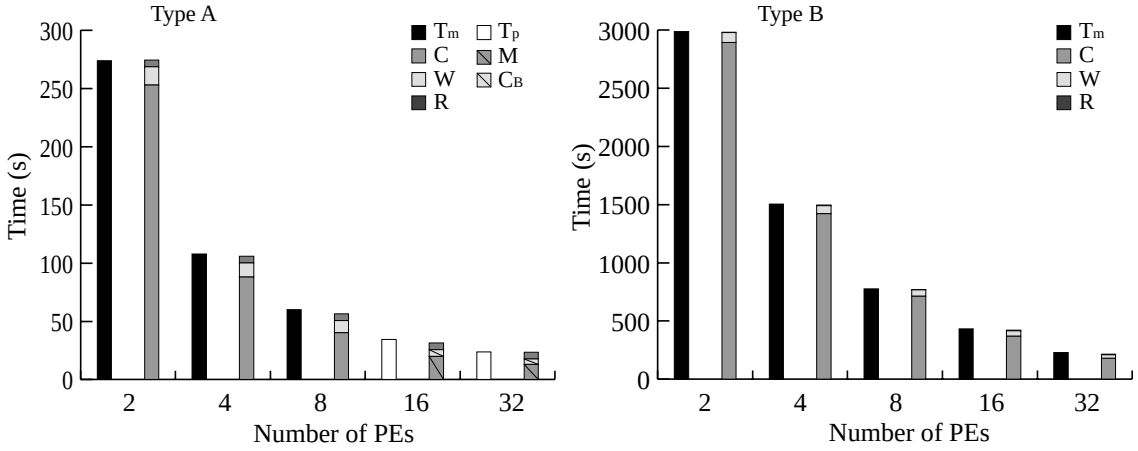

図 5.18: Effect of the systolic communication-computation overlap on the SP

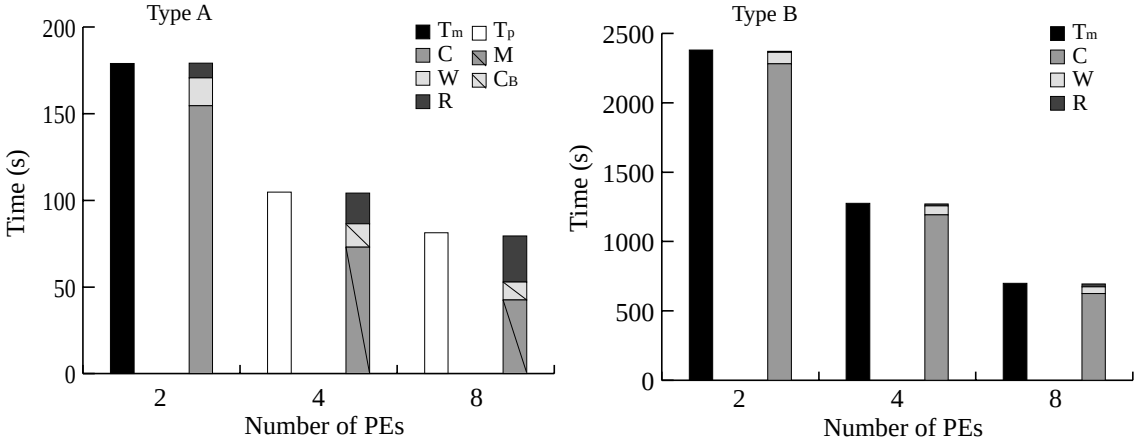

図 5.19: Effect of the systolic communication-computation overlap on the Cluster1

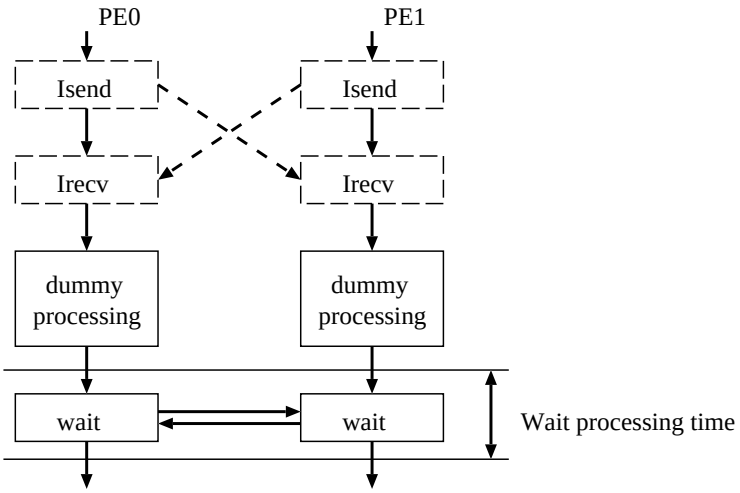

図 5.20: The overview of the asyncronous message passing and the wait processing on the SP

る時間を明確にするため追加実験を SP と Cluster1 で行なった．2 PE 間で双方向の非同期通信処理 (MPI_Isend と MPI_Irecv) 行ない，待ち処理の経過時間の計測を行なった．非同期通信のサブルーチンコールと待ち処理のサブルーチンコールの間に CPU 時間を消費する dummy ルーチン (CPU 内部で閉じた計算処理) を加え，dummy のルーチンは，通信処理が終了する程度以上である 100 反復の経過時間とした．通信処理は dummy の処理に隠蔽されているため，待ち処理は通信データ量を変化させてもほぼ一定であると考えられるが，計測すると待ち処理は通信データ量に応じて増加した．図 5.21 のように，本研究での待ち処理時間の減少の原因は，PE 数の増加とともに減少する通信データ量であると考えられる．

速度向上比

並列性能評価として速度向上比 ($Speed\ up\ ratio$= (1 PE の総経過時間)/(並列計算時の総経過時間)) を用いた．図 5.22 に SP ，図 5.23 に Cluster1 の速度向上比を示した．

図 5.22 に示した SP において，問題量の多い Type B では，$overlap$ は $ordinary$ に対して，最大約 14 % 高速であり，8 PE 以降の並列性能は直線的にのびた．また，問題量の少ない Type A では，性能の飽和が見られ，$overlap$ と $ordinary$ の差が非常に小さい．さらに PE 数が増えた場合，Type B では $ordinary$ と $overlap$

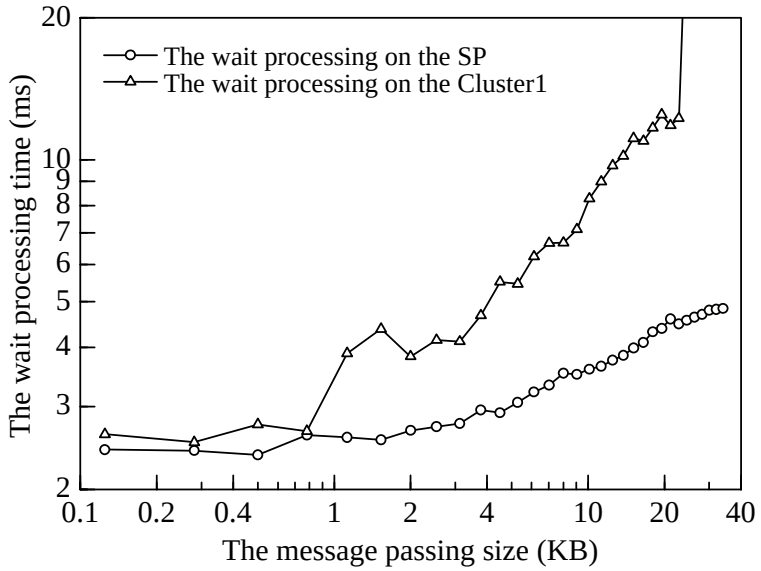

図 5.21: The Elapsed time of wait processing on the SP

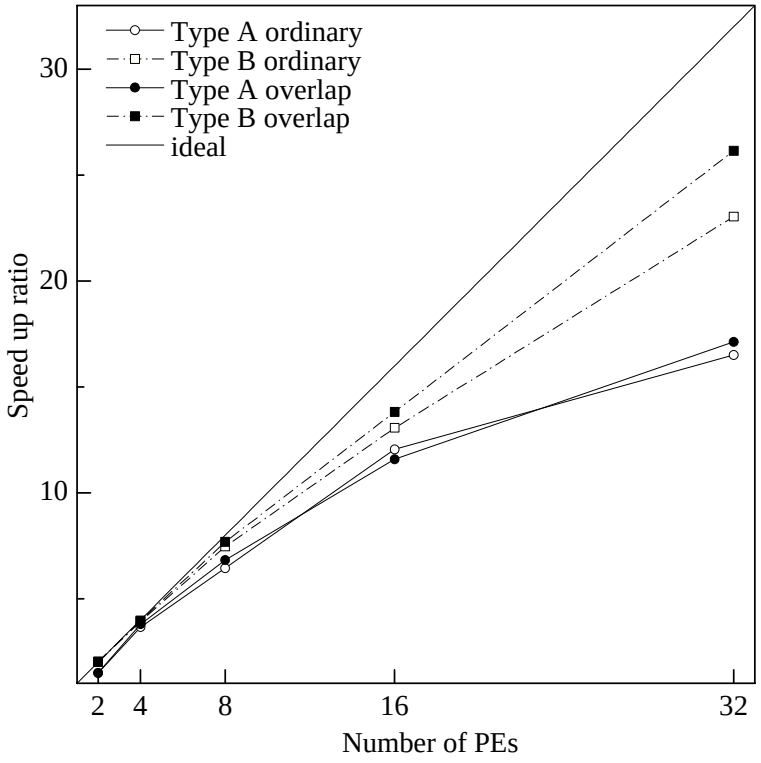

図 5.22: Speed up ratio on the SP

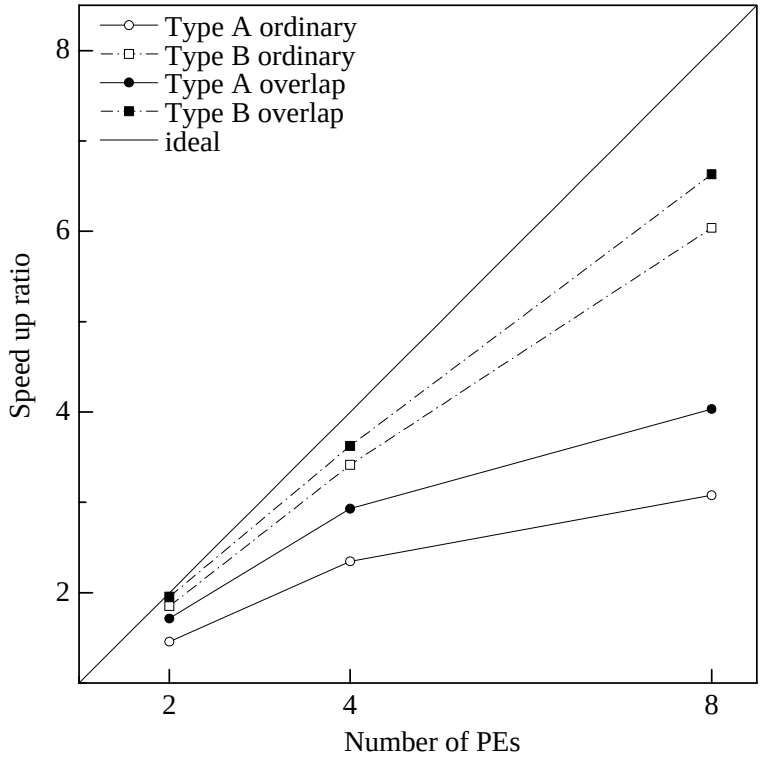

図 5.23: Speed up ratio on the Cluster1

の性能差がさらに拡大し，*overlap* の並列性能向上がさらに続くと考えられる．

図 5.23 に示した Cluster1 では，Type A で性能の飽和が見られるが，*overlap* は，8 PE においても *ordinary* ほど大きな性能低下は見られない．Type B では，*overlap* と *ordinary* ともに並列性能は直線的に推移した．Type A，B ともに *overlap* と *ordinary* の並列性能差は大きい．Type A では最大 31 %，Type B で最大 12 % 程度の差が見られた．そして，PE 数を増やした場合には，さらに並列性能差が拡大すると考えられる．

5.2.4 考　察

並列 CFD 計算シミュレーションでは，通信負荷の影響をどのように扱うかが重要な問題である．非同期通信-計算重複法はデータ依存性が少ない場合， 特に有限差分法の CFD 計算に多く発生する隣接領域のみのデータ依存性を持つ場合，データ通信処理は，計算時間で隠蔽され，通信時間の多くを擬似的に減少できるため非常に有効な手法である．また，本稿で用いた非同期通信-計算重複法は，境界領

域の処理を先行して行なうという計算順序の変更だけであるためアルゴリズムが単純であり，既存の並列 CFD 計算のプログラムへの適用や新規の並列実装においても有効であると考えられる．

非同期通信-計算重複法は，待ち処理 (MPI_Wait) のコストが予想以上に高く，非同期通信-計算重複による効果を抑えている．待ち処理に時間を要する理由は不明であるが，待ち処理の経過時間がなくなれば，非同期通信-計算重複法の効果はさらに高くなる．また，2 PE の場合と 4 PE の場合を比較した場合，データ通信量はほとんど変わらない．しかし，4 PE の待ち処理時間が短い．このため，データ通信量を少なくする努力は待ち処理時間の減少に有効ではないかと考えられる．

図 5.18，5.19 から非同期通信-計算重複法は，2 ケースの両方の場合において有効である．しかし，図 5.22 の Type A に見られるように，*ordinary* と *overlap* の並列計算性能がほとんど変化がない場合がある．原因は，計算処理性能の低下が考えられる．*overlap* は，データアクセスの順番を変更するため，順番にデータアクセスする *ordinary* に較べて計算処理効率が低下する事や通信圧縮によって計算のためのデータ移動と通信のためのデータ移動が同時に発生するため全体的なデータ移動性能が相対的に低下する事が考えられる [5]．計算処理効率の低下を抑制できれば，*overlap* の並列計算効率が *ordinary* より低くならない．また，並列化効率の向上にも役立つ．

図 5.22，5.23 より，通信圧縮が並列 CFD 計算の全体性能の向上 (あるいは性能低下の防止) に有効であることは明らかである．Type B のような問題サイズが大きな場合，さらに PE 数を増やした場合にも高い性能が得られることが予測できる．問題量が少ない Type A でさらに多数の PE を用いる場合，非同期通信- 計算重複は計算圧縮となり，通信時間のみが表面化するため通信性能に依存すると考えられる．Type A のような非常に小さな問題サイズを PC Cluster 等の通信性能が低い分散メモリ並列計算機実行しても，並列計算性能の飽和による過剰な効率低下はなかった．

PC Cluster で縮約処理に総和を用いた並列 CFD 計算の結果 [28] と比較すると，総和を用いて本研究の Type B の問題量に非同期通信-計算重複法を適用した場合，5.8 倍程度の速度向上比しか得られていない．本研究では，縮約通信に最大値を用いた．最大値を用いた本研究での結果の方が並列計算の効率が高くなった．また，

Poisson 方程式のみの場合，スーパーリニアの速度向上比が得られたが，CFD での性能ではスーパーリニアは得られなかった．この大きな原因は，縮約処理であろう．縮約処理コストの抑制を考えた場合，最も良い方法は縮約処理を行なわず，収束判定が出来ることが理想である．しかし，現実的には最大値で収束判定を行ない，判定回数を数回に一度の割合に削減することが有効であると考えられる．

第 6章

Multi Colored Line SOR 法

前章までは，分割や計算順序の変更によって収束性が変化しない Jacobi 法を用いた．しかし，Jacobi 法は収束性能が低く，さらに収束性能が高い方法を用いることを考慮する．

6.1 並列化 SOR 法の概要

有限差分法系の CFD における方程式 (Poisson 方程式や移流拡散方程式等) の解法は，Jacobi 法や SOR 法等の反復解法がよく用いられている．これらの解法は，プログラミングが容易であり，使用するメモリ量も少ない．そして，非定常の CFD のように直前の解を初期値として，係数行列や右辺ベクトルを僅かに修正しながら何回も解く場合に有利である．また，CFD や線形計算の分野では，ベクトル計算機を有効に用いるための研究が様々に行なわれ，過去にベクトル計算機上で開発されたシミュレーションプログラムの資産が非常に多く存在する．しかし，ベクトル計算機上で開発されたシミュレーションプログラムはスカラー計算機上で良い計算処理性能が得られない．

ベクトル計算で良く用いられる SOR 法は，ベクトル化するために計算順序の色付け (Multi-color SOR 法) を行なう，あるいは計算に一時的バッファを設けて (擬似 SOR 法 [29, 30])，ベクトル計算に伴う再帰計算の発生を抑止する．これらの方法は，ベクトル計算機を多用する CFD 分野において一般的な方法である．その結果，計算処理の並列性を持たせベクトル化や並列化が可能となる．しかし，メ

モリアクセスの不連続性による性能低下が，メモリアクセス性能の非常に高いベクトル計算機でも現れていた．そのため，メモリアクセス性能が低いスカラー型計算機上でのメモリアクセスの不連続性は，計算効率を非常に低下させた．既存の Multi-color SOR 法プログラムコードをスカラー型計算機に適した変更を行ない [31]，高い性能向上を得る方法が必要である．

スカラー計算機上で計算性能を向上させるには，自然順序の SOR 法を用いればよい．アルゴリズムやプログラミングは単純であり，逐次処理だけを行なう場合十分な方法である．しかし，大規模問題や高速化を考慮した場合，並列化を行なう必要がある．通常，自然順序の SOR 法を並列化した場合，収束性の悪化や収束しない状態となる場合がある．逐次処理の計算順序のまま並列化する方法には，前述の Multi-color SOR 法や Pipeline Symmetric SOR(SSOR) 法がある．Multi-color SOR 法は並列化した際に収束性に変化が現れない．また，完全な自動並列化が可能である． また，Pipeline SSOR 法も並列化による収束性の変化は無い．しかし，前章で述べたように，Poisson 方程式ソルバーでは，計算量に対する通信回数および量が多くなるため，十分な性能が得られない．スカラー型計算機での SOR 法は，Multi-color SOR 法の特性を持ち，計算性能の低下し難い方法が必要である．

SOR 法のブロック化や色付けによる計算手法の研究は古くから数多く行なわれ，様々な成果が得られている [32][33]．本研究では，既存 Multi-color SOR 法プログラムコードに対する変更点が少なく，自動並列化が可能であり，スカラー計算機での計算効率が落ちない Multi Colored Line SOR (MCLSOR) 法を提案する．MCLSOR 法は，Multi-color SOR 法の並列性を有し，スカラー型計算機でも計算性能が落ちない方法として開発した．他にも差分法ソルバーをスカラー型計算機で計算する場合にキャッシュを有効に用いる方法が提案されている [34]．この方法はキャッシュが有効に使えるようにデータ構造を考えたものである．本研究では，一定の連続データをブロック [35] として扱いキャッシュのヒット率が向上し，データのプリフェッチ機構も有効に利用できる．また，空間的な並列性を維持するため，自動並列化コンパイラ [36] や自動並列化ツール [37] を有効に用いることが可能である．

並列計算を行なう場合，Multi-color SOR 法は 3 次元の領域分割が可能であり，通信量が少なくなることである．しかし，MCLSOR 法は，ある 1 方向にデータ

依存性を持つため，領域分割は，2 次元までの制約がある．2 次元分割と 3 次元分割の通信データ量の差は，高並列になるほど大きな影響があるものと考えられるが，計算全体の処理性能を考えた場合，計算効率の向上がより重要である．また，MCLSOR 法は，領域内部で計算の並列性を持つため通信隠蔽が可能である．

本研究では，MCLSOR 法を提案し，CFD 計算で用いる Poisson 方程式ソルバーによるベンチマーク問題として，Multi-color SOR 法と MCLSOR 法を詳細に比較検討した．また，実際に並列 CFD 計算に適用して，Multi-color SOR 法に対する MCLSOR 法の並列計算での優位性と収束性を検討した．ベンチマーク問題は T3E と Onyx および PC Cluster 上に実装した．T3E では，MPI で実装したプログラムを用いて高並列時の影響を調べた．Onyx では，逐次プログラムを自動並列化コンパイラのみを用いた場合の性能を調べた．PC クラスタでは，低通信速度による全体性能への影響と非同期通信-計算重複法による通信隠蔽処理を行なった並列計算性能の検討を行なった．また SP において，MCLSOR 法の共有/分散プログラミングによる性能を調べた．

6.2 SOR 法の改良

6.2.1 Multi-color SOR 法

MCLSOR 法は，Multi-color SOR 法でのデータアクセスの不連続性による性能低下を避けつつ，Multi-color SOR 法が有する並列化に対する特性の多くを維持するための方法である．そのため，まず Multi-color SOR 法について述べる．Multi-color SOR 法は，計算順序を変更することで自然順序の SOR 法における再帰計算を解消し，ベクトル計算を行なうための一般的な方法である．Multi-color SOR 法の色付けは，経験則で付けられる事が多い．また，ベクトル計算機上では，ある差分精度における色数は，再帰計算を防げる最小数である程よい．なぜなら，色数が最小であればベクトル長が長くなる．ただし，近年では，多数の色数を用いて，収束性を向上させる試みがある．しかし，スカラー型並列計算機においてベクトル長は重大な問題ではない．有限差分法での色付けは色数ではなくプログラミングの簡便さと並列化を考慮した色数と色付けを用いた．例として次節で説明する Poisson 方程式を 2 次中心差分で離散化した場合と NS 方程式を最大 3 次風上差

分で離散化した場合を考える．

図 6.1 に Poisson 方程式のデータの依存関係 (再帰計算の発生) を示した．直交格子での計算は，上下・左右・前後の 7 点 (図 6.1 左) を必要 (7 点差分) とする．一般座標変換での計算は，座標変換によって発生する交差微分項による斜め方向のデータ依存 (19 点差分) がある (図 6.1 右)． 以上のデータ依存性を考慮して色付け

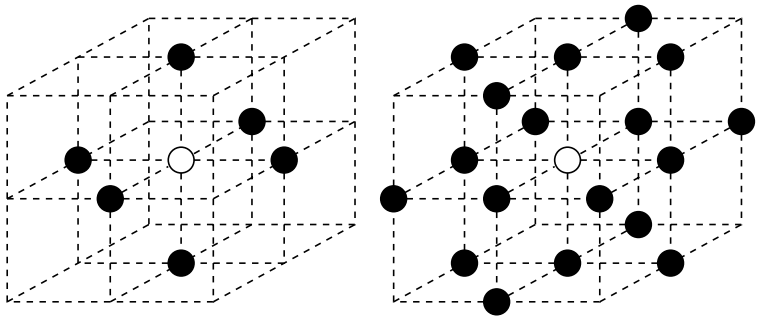

図 6.1: A 7-point stencil(left) and a 19-point stencil(right)

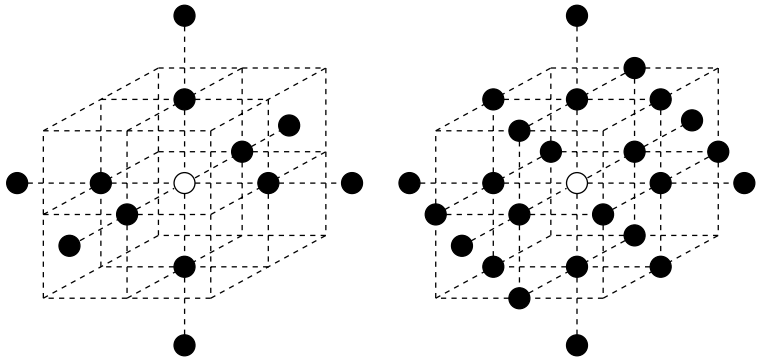

図 6.2: A 13-point stencil(left) and a 25-point stencil(right)

を行ない計算順序を変更する．色付けの方法は，求める格子点の色は，差分計算を構成する要素の格子点の色がお互いに異なるように構成する．7 点差分の場合，各隣接格子点 (上下・左右・前後) 6 点と直接接続せず，データ依存性を解消するには 2 色必要である (Red-Black)．また，19 点差分の場合，7 点差分の隣接格子点の他に斜め方向格子点とのデータ依存が発生するため，

色付けは 5 色以上必要である．色付け方法は様々な方法があるが，本研究では一般座標の 3 次元 Poisson 方程式の 3 次元でのデータ依存を考慮して 8 色用いた (図 6.3 左).

図 6.2 に NS 方程式のデータの依存関係 (再帰計算の発生) を示した．直交格子での計算は，上下・左右・前後の 13 点 (図 6.2 左) を必要 (13 点差分) とする (6 色以上)．一般座標変換での計算は，Poisson 方程式と同様に斜め方向のデータ依存 (25 点差分) がある (図 6.2 右)(7 色以上)．本研究では，一般座標の 3 次元 NS 方程式を考慮し 27-color SOR 法を用いた (図 6.3 右)．

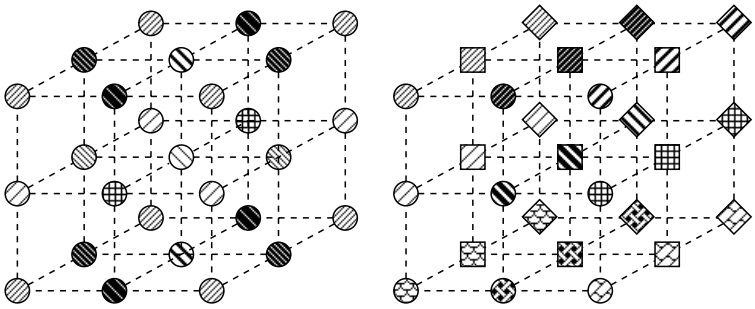

図 6.3: A 8-color coloring(left) and a 27-color coloring(right)

6.2.2 MCLSOR 法

MCLSOR 法は，Multi-color SOR 法でのデータアクセスの不連続性による性能低下を避けつつ，Multi-color SOR 法が有する並列化に対する特性の多くを維持するための方法である．Multi-color SOR 法の不連続なデータアクセスは，ベクトル計算による再帰計算を防ぐためにはやむを得ない．しかし，スカラー計算機上ではベクトル計算における再帰を考慮する必要がない．しかし，自動並列化コンパイラの利用を含め，並列化を行なう場合には，スカラー計算機上でも Multi-color SOR 法が持っている空間的なデータ並列性は必要である．スカラー並列計算機の利用における，連続データアクセスによる性能向上と並列化によるデータ依存性の解消を両立する方法を開発する必要がある．

まず，データアクセスの連続性を維持するには，Fortran におけるデータ配列の最左 index のデータアクセスを連続に行なう必要がある (図 6.4 のように 3 次元のデータ配列を取る際に，最左 index を i とし，次を j，最右 index を k とした)．i-index の処理 DO ループを最内 DO ループとし，データ処理を線 (Line) という 1 つのデータの処理単位として定義する．ただし，ベクトル計算のように最内ループだけの問題ではなく，計算処理に伴う連続データアクセスを行なうために，線は必ずメ

モリアドレスが連続である方向に取る必要がある．

```
double precision a(imax,jmax,kmax)


do k=1,kmax
  do j=1,jmax
    do i=1,imax
      ;;
```

図 6.4: kernel iterations

```
do l=1,4
  do k=ks(l),kmax,2
    do j=js(l),jmax,2
      do i=1,imax
        ;;
```

図 6.5: kernel iterations for MCLSOR method

また，線データを計算することで，Multi-color SOR 法では，前の格子点計算において用いてキャッシュ内部に存在するであろうデータが再使用できない．MCLSOR 法では，図 6.6 のように Multi-color SOR 法よりもキャッシュ内に存在するより多くのデータを再使用することができる．図中の四角は前の計算格子点，白丸は今計算する格子点である．黒丸は前格子点計算での参照データであり，斜線の丸は前格子点の四角の計算に用いて白丸の格子点の計算にも用いることが出来るデータである．

次に，データ依存関係の解消するため，線の連続データ以外の部分を解決する必要がある．空間的なデータ依存を解消するため，j,k-index の DO ループに対して，データ依存性を解消する色付けを行なう．MCLSOR 法では，j,k-index の 2 次元面上でのデータ依存のみを考えればよい．そのため，色付けは 2 次元領域での Multi-color SOR 法と同じ色数と色付けを行なう．一般座標系の Poisson 方程式に MCLSOR 法を適用した場合，j,k-index の DO ループのデータ依存を解消するために 4 本の線 (4colored line) が必要となり (図 6.7 左)，処理ループは図 6.5 のよ

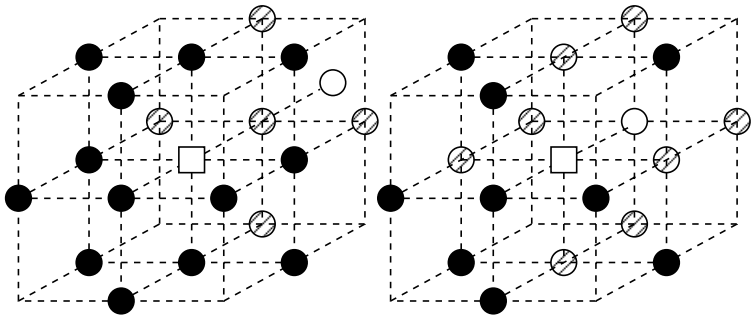

図 6.6: Cache hit (left：Multi-color SOR method，right：MCLSOR method)

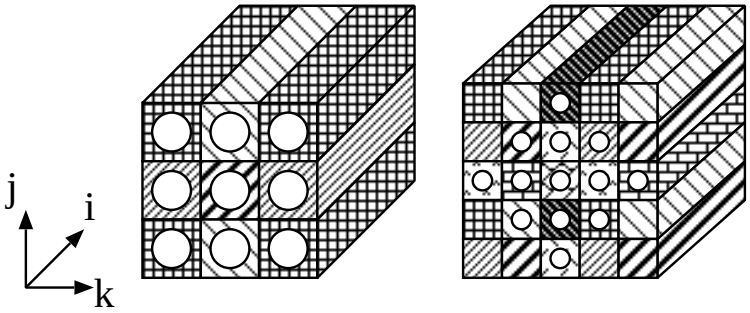

図 6.7: A 4-line coloring and a 9-line coloring

うになる．また，3 次精度風上差分で離散化した NS 方程式は，9colored line を用い (図 6.7 右)，処理ループは図 6.5 と異なり最外ループを 9 反復する処理ループとなる．

6.2.3 並列化

Multi-color SOR 法や MCLSOR 法の計算では，空間的な並列性が存在するため，単純な領域分割を行なえばよい．ただし，3 次元領域を考えた場合，Multi-color SOR 法では，全ての方向に対して空間的な並列性が存在するため，3 次元の領域分割を行なっても収束性が変化しない．そのため，並列性が高くなるほど，通信量が少なくなる．MCLSOR 法では，1 次元方向に対して並列性が存在しないため，2 次元までの領域分割で収束性の変化は現れない．Multi-color SOR 法に比べて MCLSOR 法は並列性が高くなった場合，通信量が多くなる．しかし，商用の並列計算機のように高速な通信機構を持つ場合，有限差分法系 CFD 計算の並列性能は，通信量に対して大きく影響しない場合が多い (第 3 章).

データ通信は各色や線の計算後毎に実行する．そのため，8 color SOR 法の場合，SOR 法の 1 反復毎に 8 回のデータ通信が発生する．4-line の MCLSOR 法の場合，1 反復毎に 4 回のデータ通信が発生する．図 6.8 に 4-line の MCLSOR 法でのデータ通信状態の概略を示した．図中は $j-k$ 断面を表示し，奥行き方向に i 方向を持つ．各線成分を計算後，*Boundary region* 線成分のを隣接 *Overlap region* に通信する．仮に 1 次元分割の場合，分割面と水平な格子の線を 2 本先に計算した場合，面データのデータ通信を行なうだけで良いが，2 次元分割を行なった場合には，格子の各線成分の計算後データ通信を必ず行なわなければならない．これは，8 color SOR 法でも同様で，各色成分を計算後，データ通信を行ない必要がある．

また，各方向の格子点数が偶数，奇数の場合でデータ通信の状態が大きく変化する．偶数の場合，ある PE におけるあるデータ通信処理は，1 方向に対する送信処理あるいは受信処理のどちらかである．しかし，偶数の場合，ある PE におけるあるデータ通信処理は，2 方向に送信処理あるいは受信処理を行なうことになる．データ通信効率を考慮した場合，PC クラスタ等のスイッチネットワークでは，明らかに偶数の場合での効率が高いと考えられる．そのため，本研究では，各方向の格子点数が偶数になるような格子サイズを考える．

MCLSOR 法は，3 次元の領域分割を行なえないため，Multi-color SOR 法に対してデータ通信量が多くなる．そのため PC Cluster において，通信隠蔽処理は，データ通信量の増加による並列計算性能への影響を低く抑え，さらに並列計算性能を向上させることが可能である．Multi-color SOR 法や MCLSOR 法は，空間的な計算並列性を

有するため，PC Cluster 等の通信性能が低い並列計算システムでは，通信隠蔽処理が有効であると考えられる．また，周期境界条件を用いるような場合，領域分割の状態によっては境界部分にも通信が発生するため，CFD シミュレーションにおいて，通常の実装よりも通信隠蔽を用いた方が効果的である可能性は高い．

計算手順は，Jacobi 法と同様に *Boundary region* の色あるいは線を先に計算し，非同期通信後内部の同種の色，同種の線を計算する．Jacobi 法と異なり，通信隠蔽のための計算時間は，4 line SOR 法で 1/4，8 color SOR 法で 1/8 と少なくなるが，非同期通信-計算重複法を用いた場合，通信圧縮の状態になる場合が少なくなるが，計算圧縮状態でも有効な結果が得られるものと考えられる．しかし，

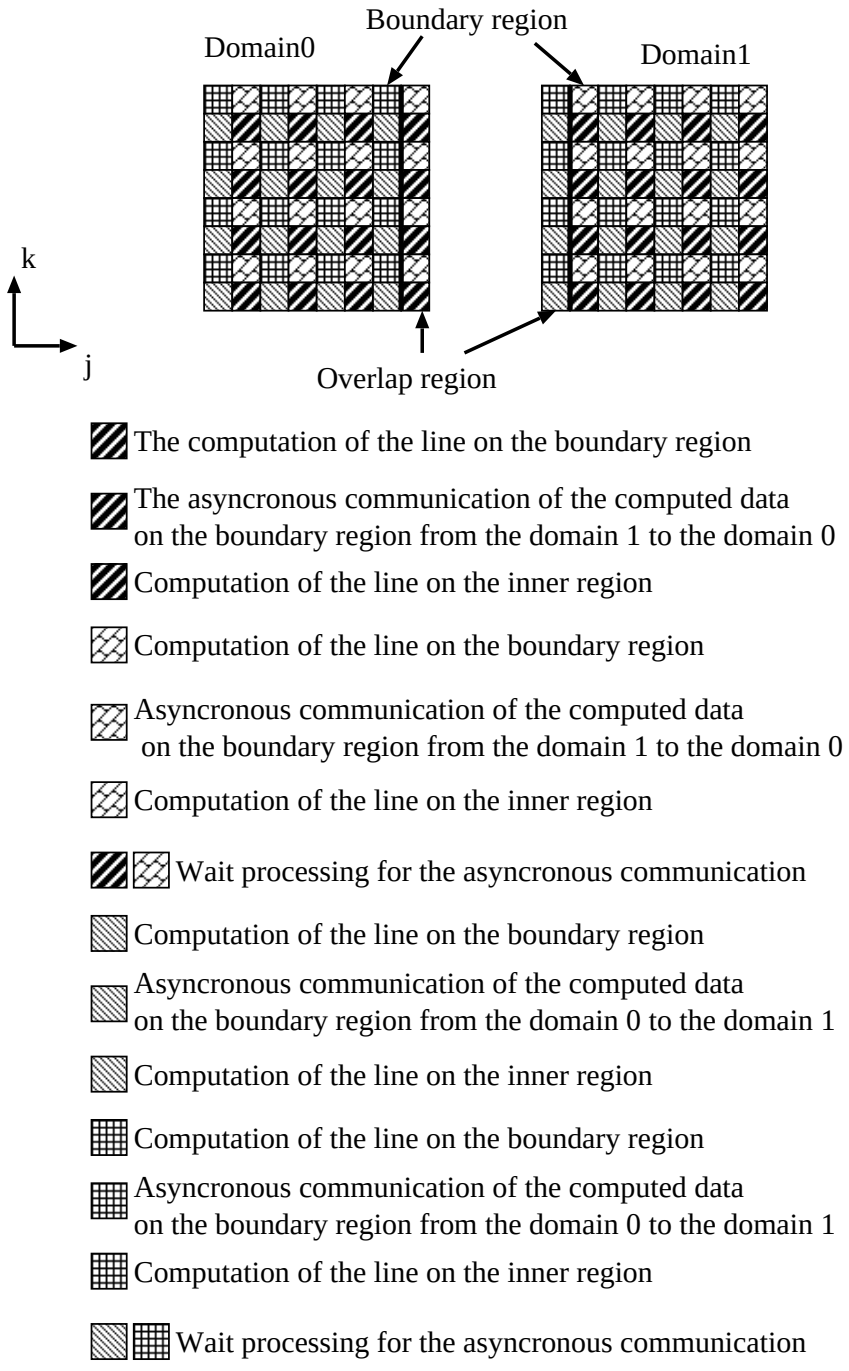

図 6.8: The communication pattern for a MCL or multi-color SOR method

Jacobi 法とは異なる傾向があらわれる可能性が高い.

6.2.4 ベンチマーク問題

並列計算性能

前述のように非圧縮粘性流体の CFD シミュレーションプログラムの性能は, Poisson 方程式ソルバーの計算処理性 能を評価することで詳細に評価できる. Poisson 方程式ソルバーを用いて, Multi-color SOR 法と MCLSOR 法の各並列計算機システムでの並列計算性能のみを評価した.

用いた Poisson 方程式ソルバーは, T3E, SP と Cluster2 では, 上述の CFD シミュレーションに用いた周期境界条件等の特殊な条件を用いないソルバーである. 反復回数は 200 回の固定である. 各 SOR 法ソルバーに Multi-color SOR 法と MCLSOR 法を適用した場合の並列計算時間を計測した. また, 自動並列コンパイラにおける評価を Onyx で行ない, Onyx では, 逐次処理プログラムコードを用い, ディレクティブ等の並列化支援のコードは含まれない. 用いた格子サイズは, 表 6.1 である.

表 6.1: Grid size for the benchmark test of the MCLSOR method

Grid	Number of grid points
Type1	26×26×26
Type2	66×66×66
Type3	34×66×130

Type3 格子は, 格子の取り方が 6 種類存在するため, 最も i ループが短くなる子形状 $34 \times 66 \times 130$ を採用した.

収束性能

Multi-color SOR 法と MCLSOR 法における収束性の違いについて, 前述の Cavity 流れの条件を用いて検討した. また, 同様に Jacobi 法と Pipeline SSOR 法 の収束性と並列計算性能についても検討した. 並列計算機は, T3E, SP と Cluster2 である. 計算サイズは, 表 6.1 の Type1, Type2 である.

Poisson 方程式の計算条件は，上述の Cavity 流れにおける CFD 計算において得られる計算格子での係数と物理量を用いた．物理量は，初期時間での圧力 p を用いた．収束判定を行なう縮約通信は，毎ステップ行なった．収束判定は CFD シミュレーションと同じ 1.0×10^{-5} である．SOR 法の加速係数は，以後の計算全て 1.5 とした．収束性能は，収束判定条件を満たすまでの反復回数を計測した．境界条件は $\frac{\partial p}{\partial X} = 0.0$ とした．ただし，定常流れの計算において，計算初期状態において収束判定条件を満たすまで計算を行なうことはほとんど無いため，実際の計算とはやや異なるが，検証条件としては有効である．

Pipeline SSOR 法の実装について説明する．前述のように Pipeline SSOR 法は，NPB LU で用いられている．NPB LU は行列を上 3 角行列と下 3 角行列に分かれる片側離散化を行ない，SSOR 法を用いて圧縮性 NS 方程式解く方法である．ここでは，計算手法はほぼ同等であり，離散化は変更せず Poisson 方程式に Pipeline SSOR 法を用いた．図 6.9 に 8 PE での領域分割と計算順序を示した．そして，図 6.10 にその時の計算/通信パターンを示した．順方向として $k = 1$ において PE 0 から PE 7 に計算を進める．その際 PE 0 はPE 1 と PE 2 にデータ通信を行ない，PE 0 は $k = 2$ の計算を進める．そして，PE 1 と PE 2 も $k = 1$ の面を計算後，$k = 2$ の計算を進める．順方向を計算後，逆方向の計算を PE 7 から同様に計算を進める．

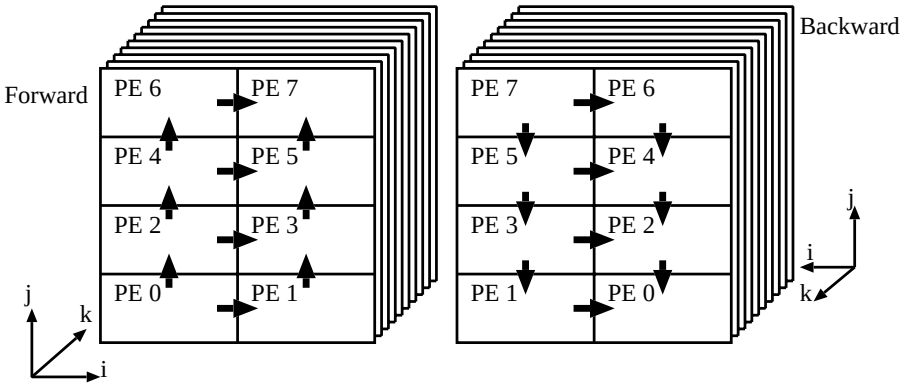

図 6.9: The domain partitioning and the computational overview on the 8 PE

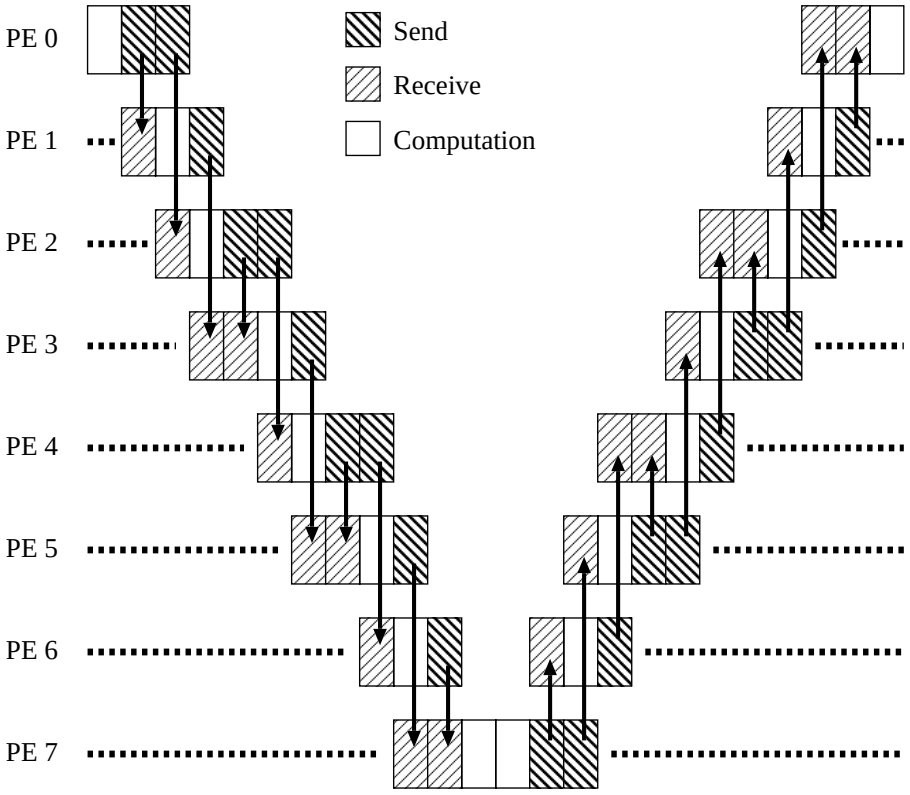

図 6.10: The computation and communication patterns on the 8 PE

6.2.5 計算対象

実際に T3E と Cluster2 を用いて非圧縮粘性流体の CFD 計算性能を検討した．非圧縮粘性流体の計算モデルは，図 6.11 に示す 3 次元円柱周り流れとした．Re_D 数は $10,000$ である．ただし，代表長さ L を円柱直径 1(1D:D は円柱直径)，代表速度 U を 1 とした．計測時間は，無次元時間 5 までとした．計算手法は，非定常問題の解法として良く用いられる MAC 法をベースとし，NS 方程式の移流項には K-K スキーム [8] を用いた．NS 方程式解法は陰解法である Crank-Nicolson 法を用いた．本研究では，0 型格子を用いたため，円柱の上下端面と円柱に垂直な断

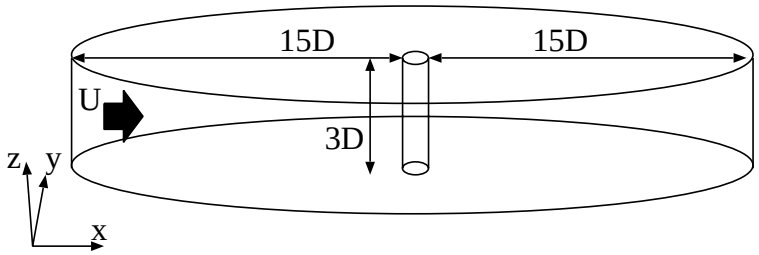

図 6.11: 円柱周りの計算領域

面方向の 2 方向の境界で周期境界条件となる．計算格子は円柱境界付近で格子点間隔を細かくとった．検討に用いた計算格子サイズは，$34 \times 66 \times 130$ (291,720 格子点) とした．図 6.11 において，x 方向の流速を u，y 方向の流速を v，z 方向の流速を w とした．速度の境界条件は，流入境界は $u = 1.0$，$v, w = 0.0$，流出境界は，自由流出とし，円柱境界はすべり無し境界とした．また，上下境界は，周期境界条件である．圧力の境界条件は，外部境界は無限遠方仮定を用い $p = 0.0$，円柱境界は，$\frac{\partial p}{\partial x} = 0.0$，上下境界は，周期境界条件である．

6.3 結　果

6.3.1 ベンチマーク問題

結果は MCLSOR 法，Multi-color SOR 法や Jacobi 法等のソルバー間の比較を行なうため，総経過時間と MCLSOR 法に対する割合 ($Ratio$)

$$Ratio = \frac{\frac{Etime_s}{Etime_p}}{PEs} \tag{6.1}$$

をグラフ化した．ただし，$Etime_s$ は，MCLSOR 法の $ordinary$ の逐次処理の総経過時間，$Etime_p$ は各場合における並列処理の経過時間，PEs は，PE 数である．MCLSOR 法においての割合は単純な並列化効率である．その他の場合は，MCLSOR 法に対する割合である．用いた並列計算機は，T3E，SP，Cluster2，Onyx である．

図 6.12，6.16，6.20 に T3E，図 6.24，6.28，6.32 に SP，図 6.36，6.40，6.44 に Cluster2，図 6.48，6.50，6.52 に Onyx における Type1，Type2，Type3 での総経過時間の結果をそれぞれ示した．また，図 6.13，6.17，6.21 に T3E，図 6.25，6.29，6.33 に SP，図 6.37，6.41，6.45 に Cluster2，図 6.49，6.51，6.53 に Onyx における Type1，Type2，Type3 での $Ratio$ の結果をそれぞれ示した．グラフ中の MCL は MCLSOR 法を示し，MC は Multi-color SOR 法を示す．

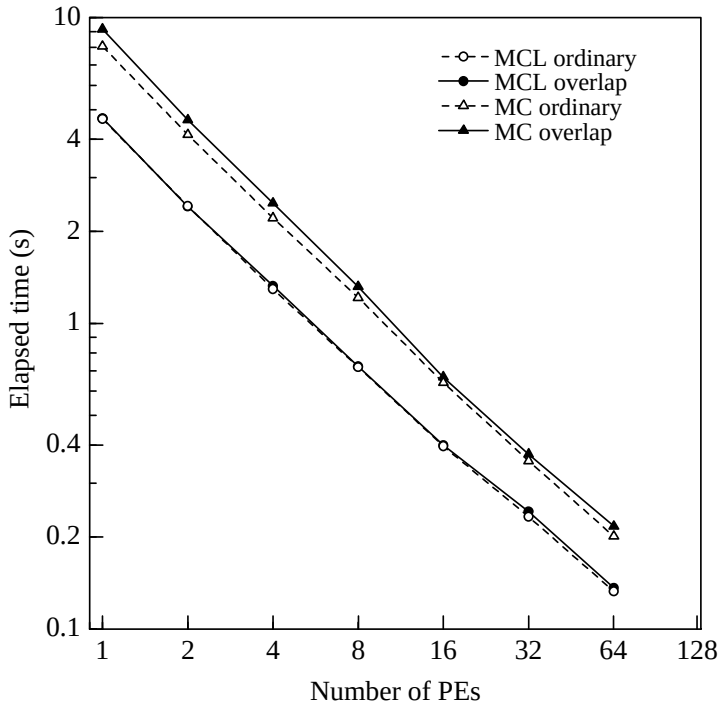

図 6.12: The elapsed time of the benchmark result about the Type1 grid on the T3E

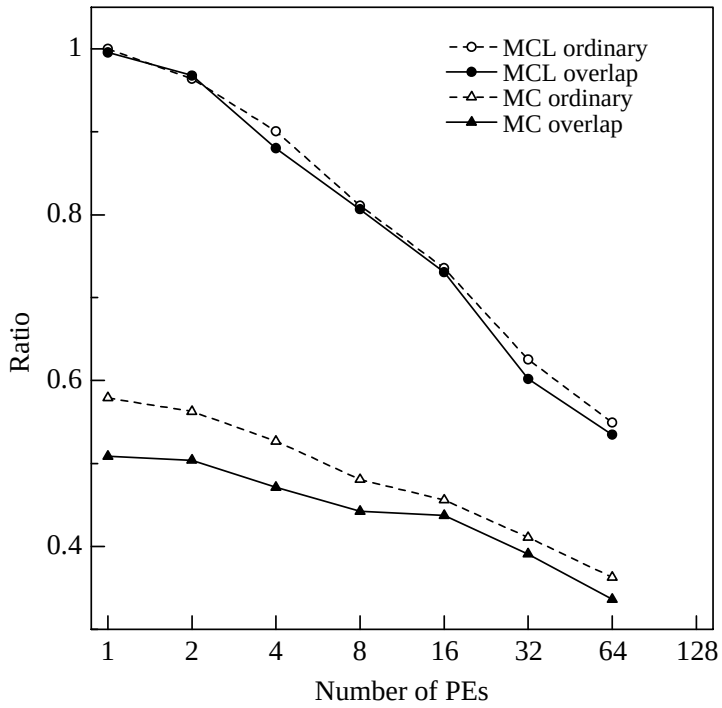

図 6.13: The efficiency of the benchmark result about the Type1 grid on the T3E

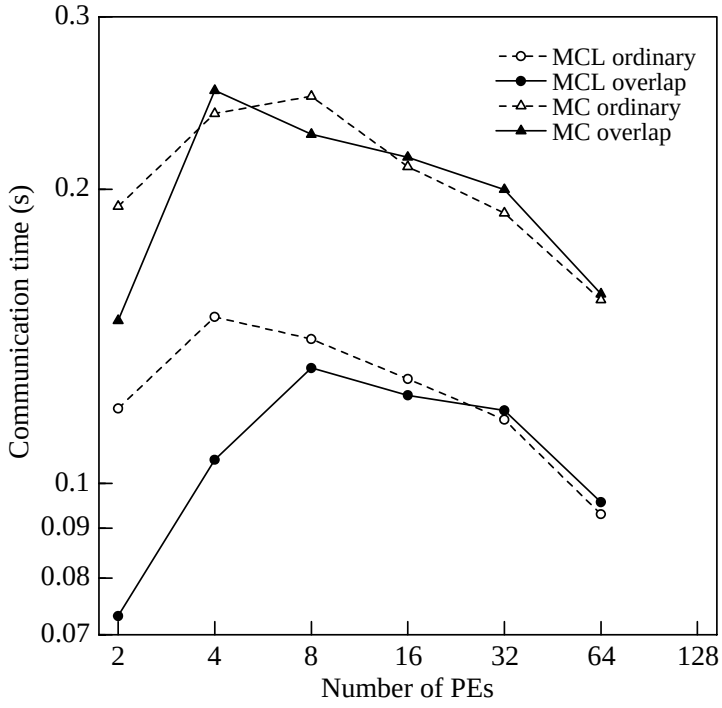

図 6.14: The communication time of the benchmark result about the Type1 grid on the T3E

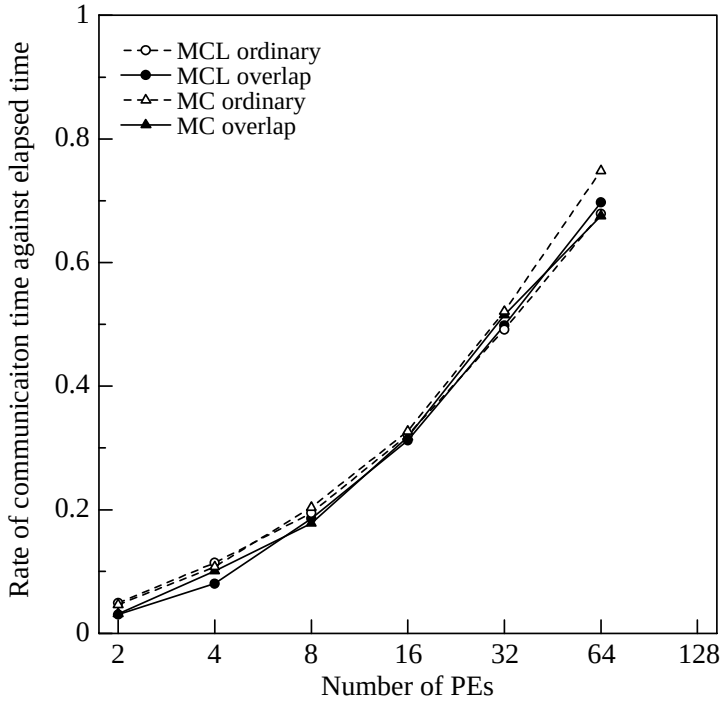

図 6.15: The rate of communication time against the elapsed time of the benchmark result about the Type1 grid on the T3E

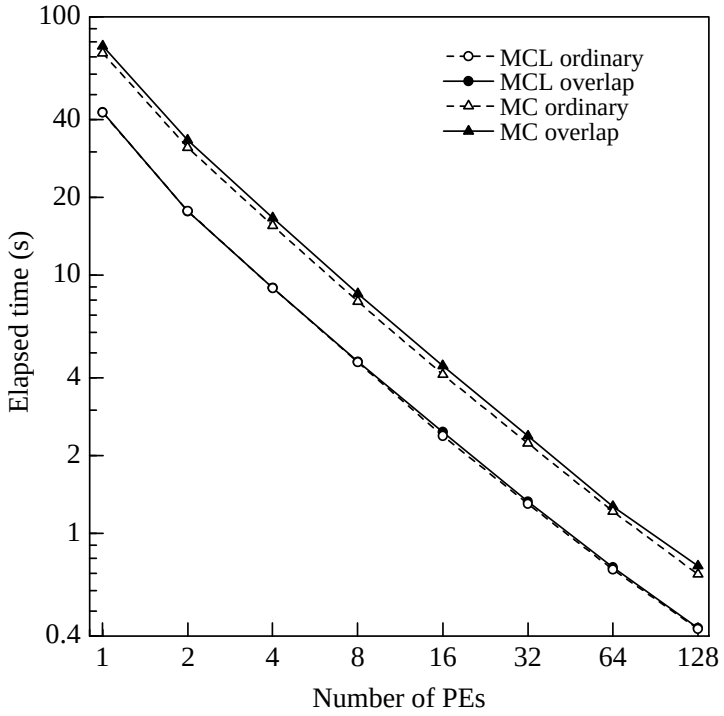

図 6.16: The elapsed time of the benchmark result about the Type2 grid on the T3E

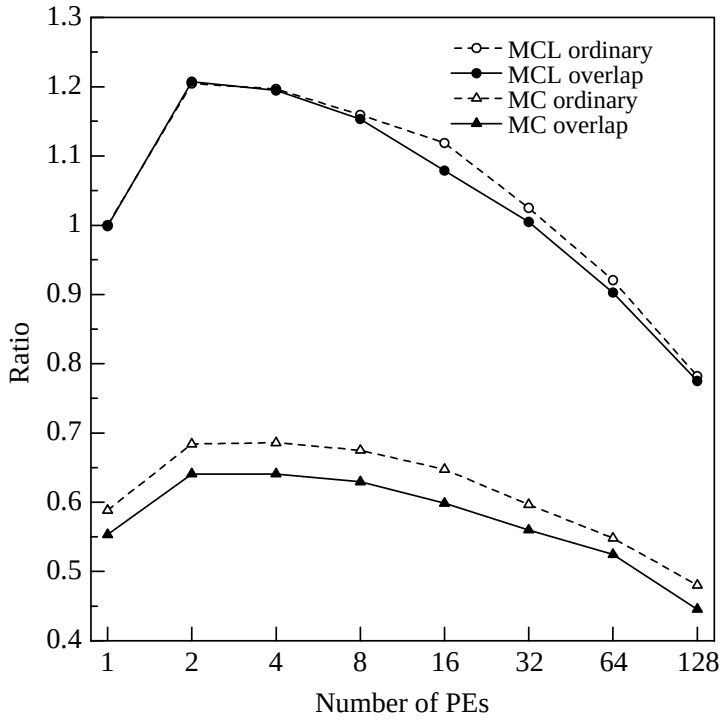

図 6.17: The efficiency of the benchmark result about the Type2 grid on the T3E

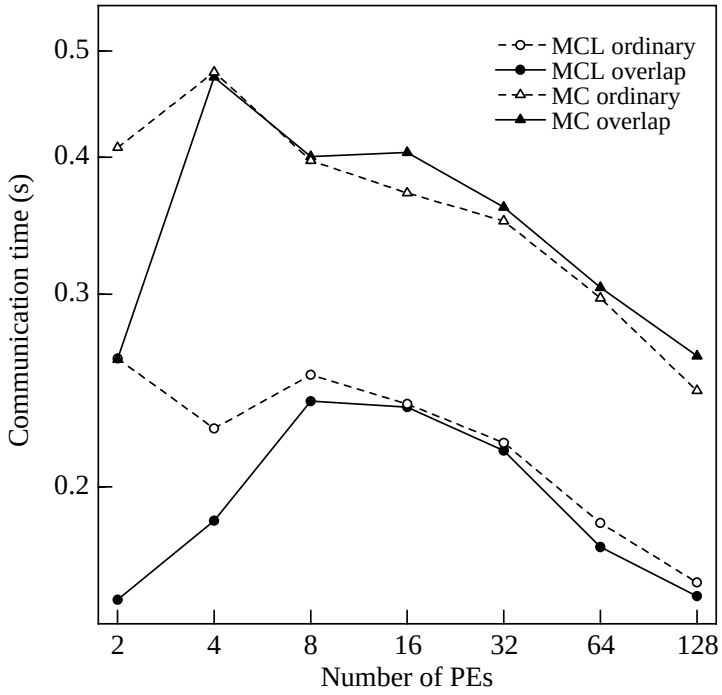

図 6.18: The communication time of the benchmark result about the Type2 grid on the T3E

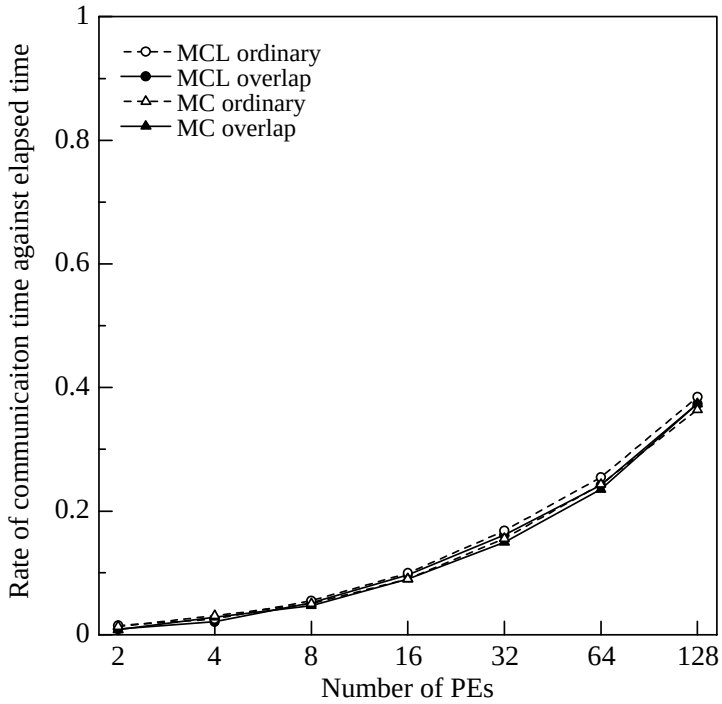

図 6.19: The rate of communication time against the elapsed time of the benchmark result about the Type2 grid on the T3E

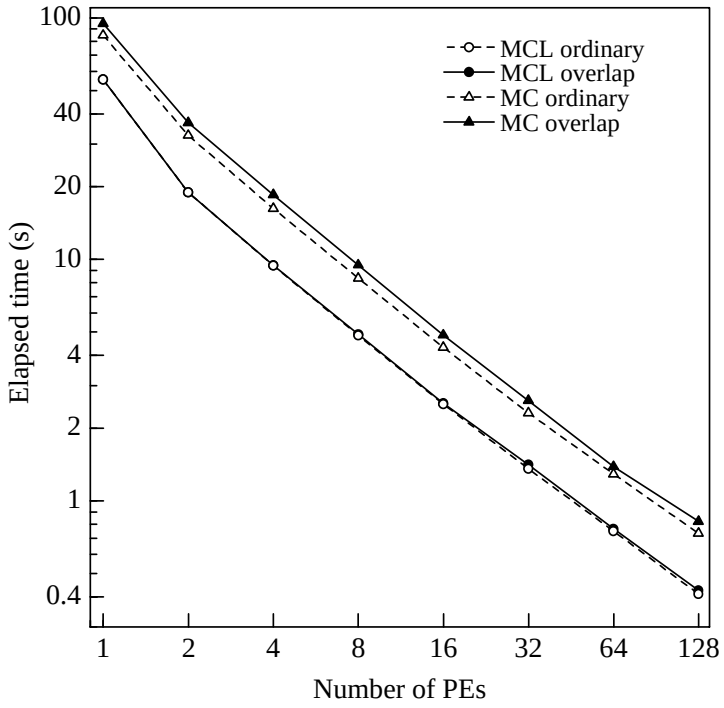

図 6.20: The elapsed time of the benchmark result about the Type3 grid on the T3E

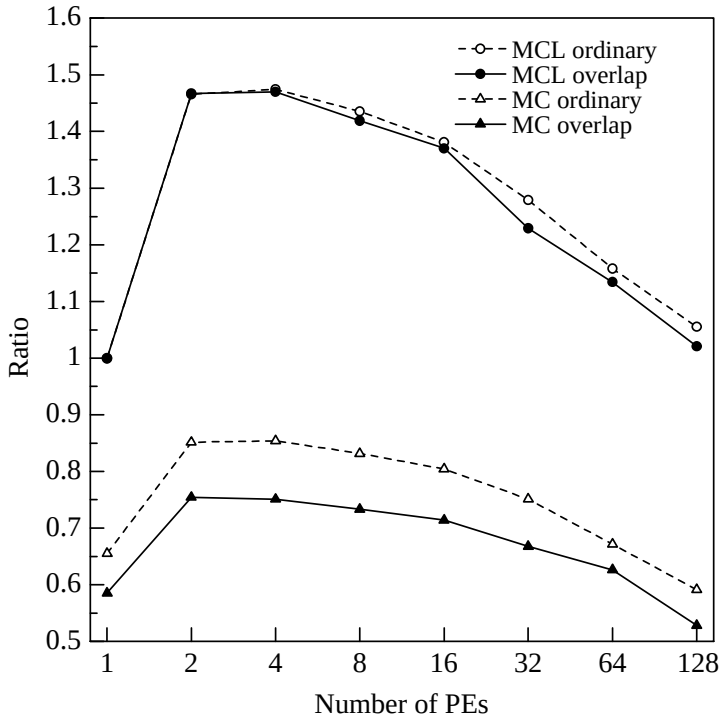

図 6.21: The efficiency of the benchmark result about the Type3 grid on the T3E

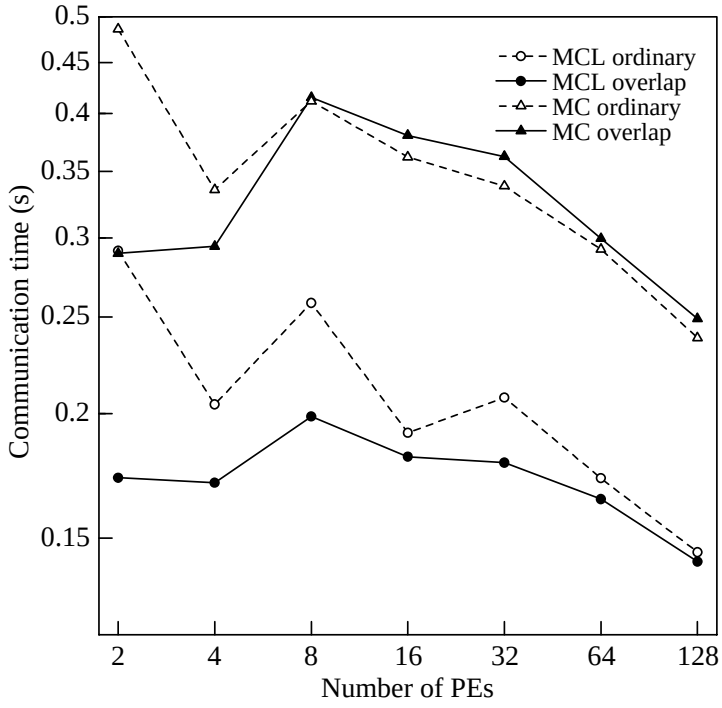

図 6.22: The communication time of the benchmark result about the Type3 grid on the T3E

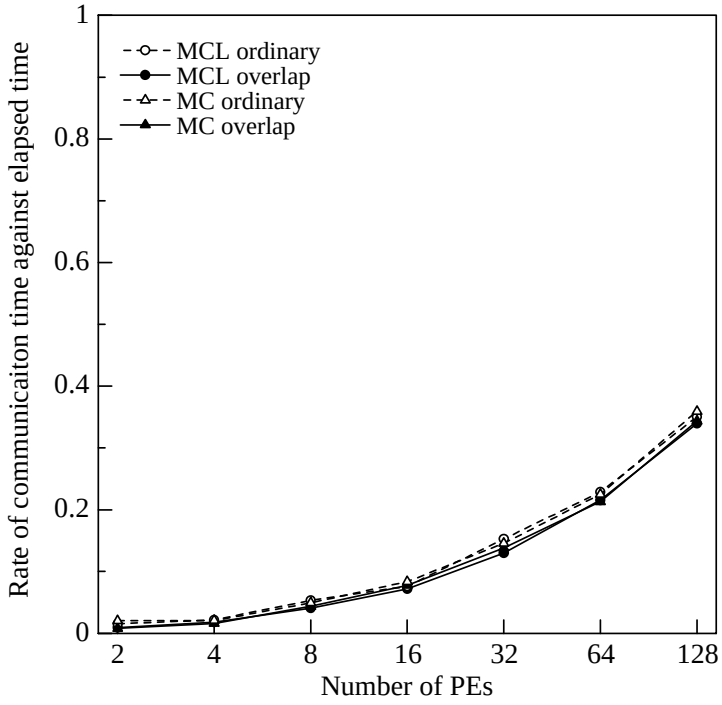

図 6.23: The rate of communication time against the elapsed time of the benchmark result about the Type3 grid on the T3E

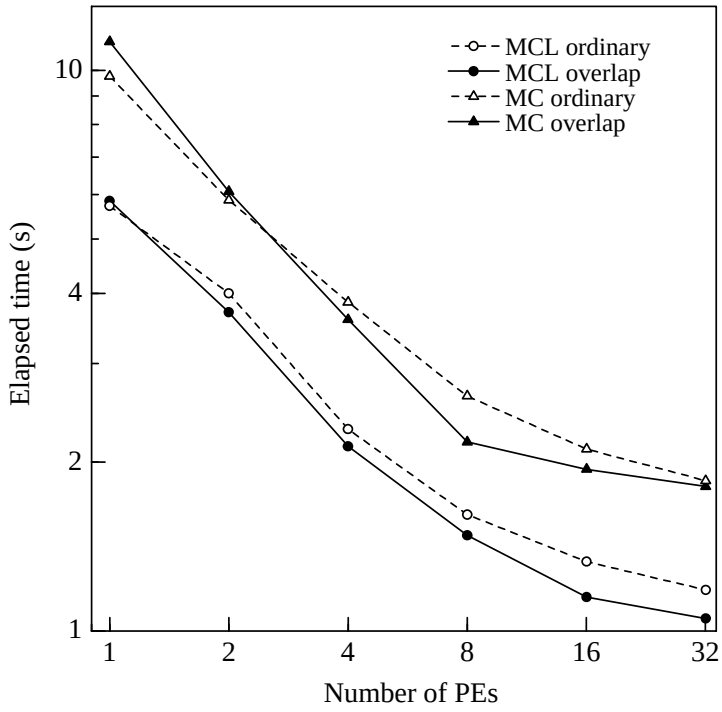

図 6.24: The elapsed time of the benchmark result about the Type1 grid on the SP

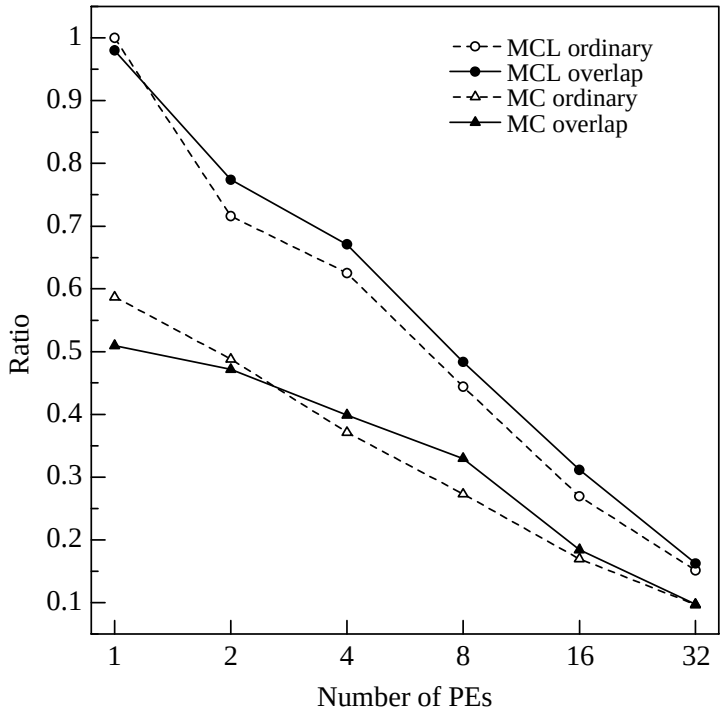

図 6.25: The efficiency of the benchmark result about the Type1 grid on the SP

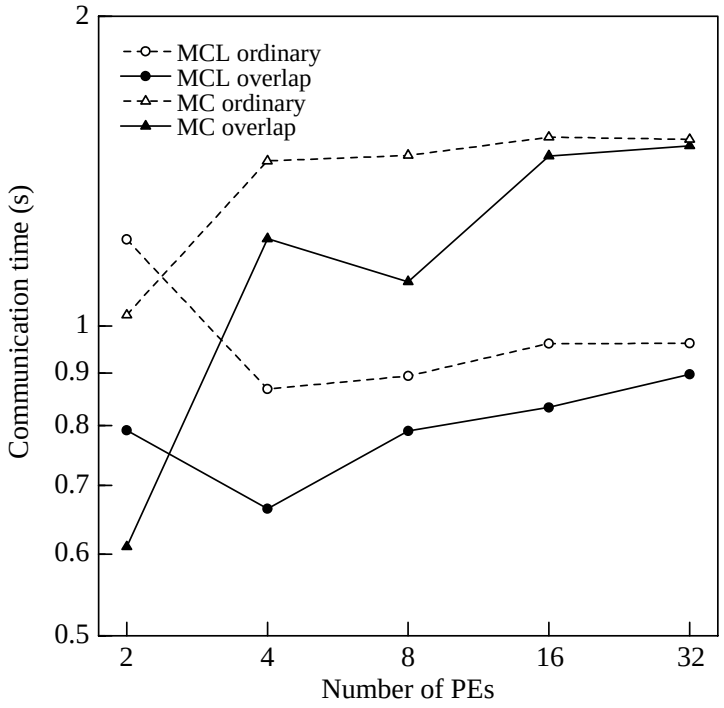

図 6.26: The communication time of the benchmark result about the Type1 grid on the SP

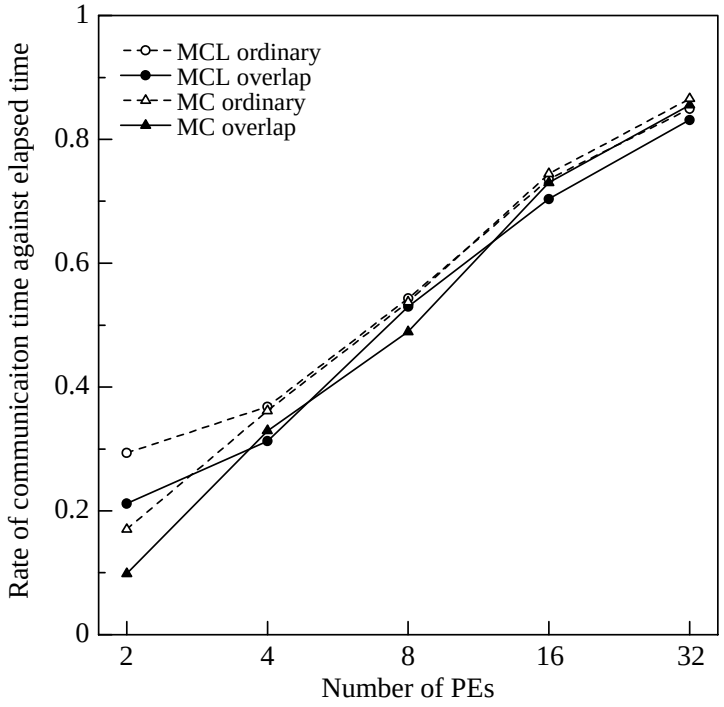

図 6.27: The rate of communication time against the elapsed time of the benchmark result about the Type1 grid on the SP

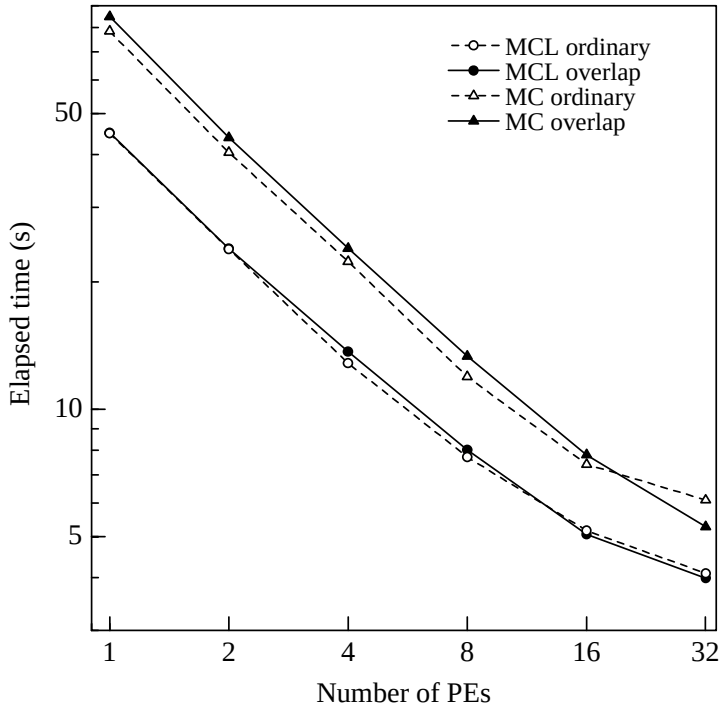

図 6.28: The elapsed time of the benchmark result about the Type2 grid on the SP

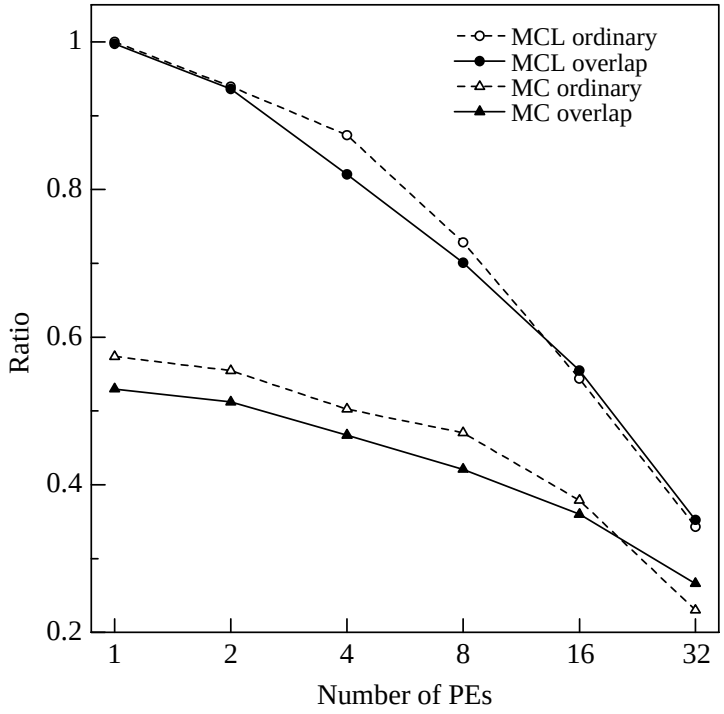

図 6.29: The efficiency of the benchmark result about the Type2 grid on the SP

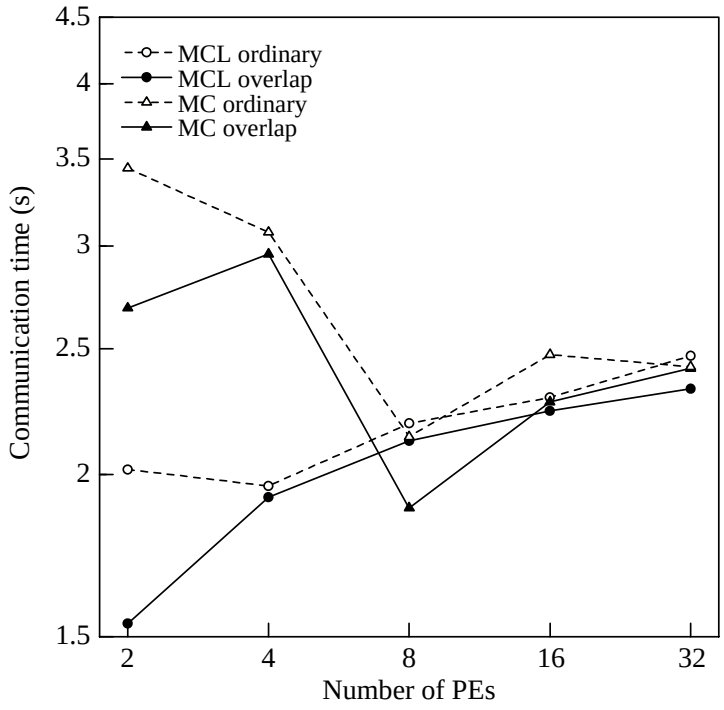

図 6.30: The communication time of the benchmark result about the Type2 grid on the SP

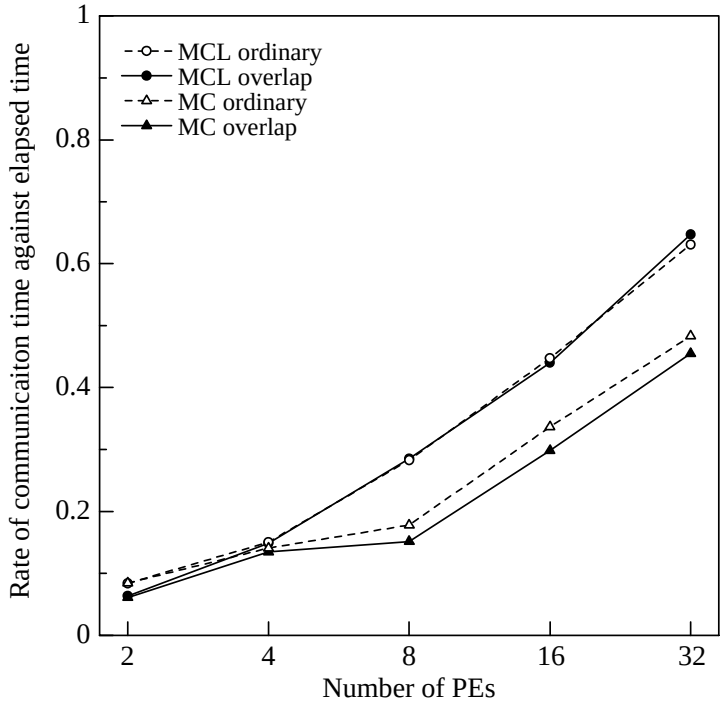

図 6.31: The rate of communication time against the elapsed time of the benchmark result about the Type2 grid on the SP

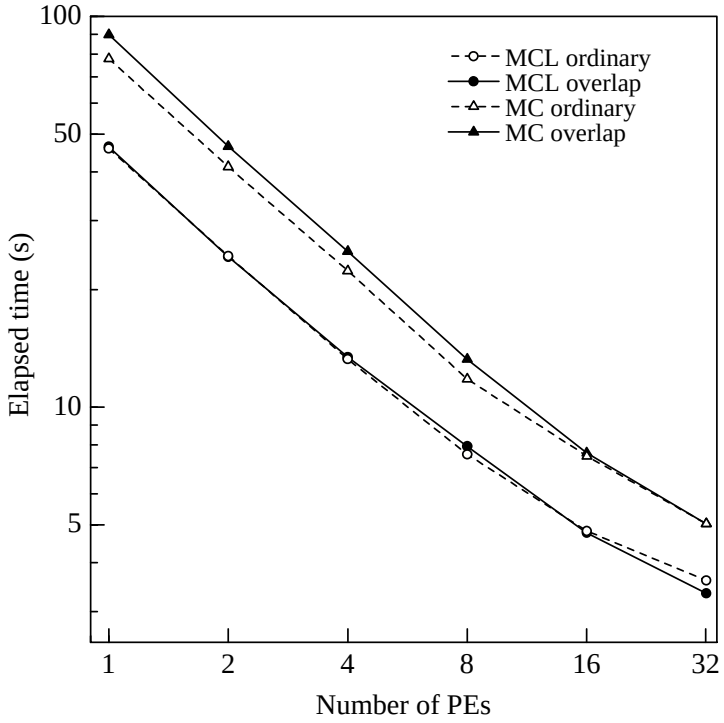

図 6.32: The elapsed time of the benchmark result about the Type3 grid on the SP

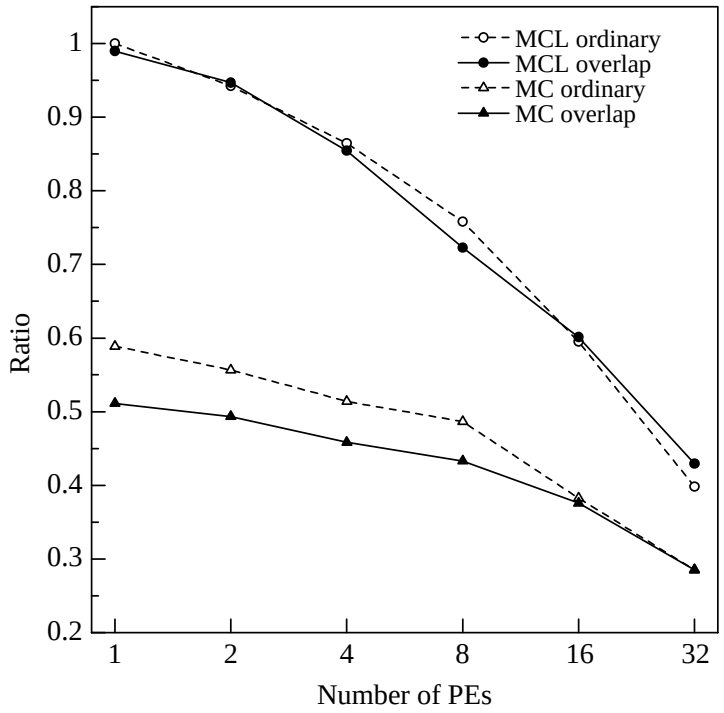

図 6.33: The efficiency of the benchmark result about the Type3 grid on the SP

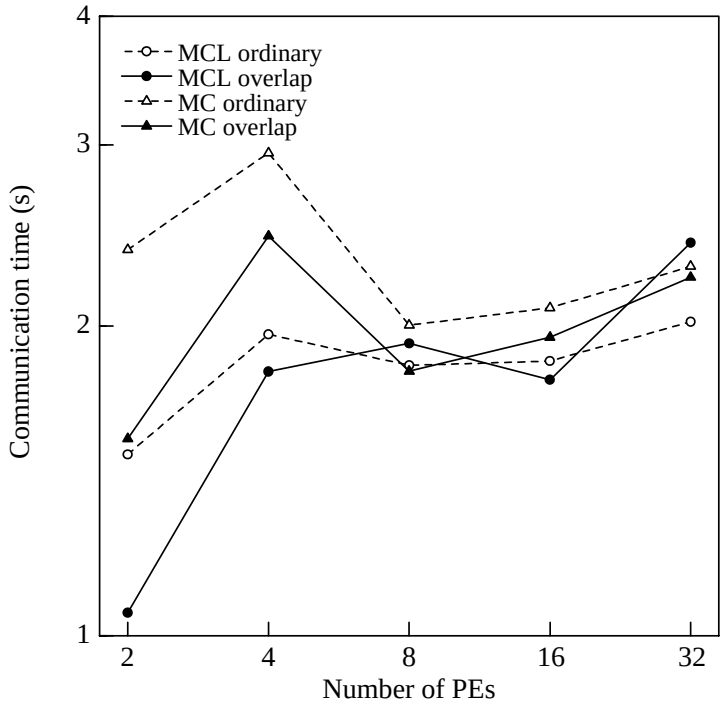

図 6.34: The communication time of the benchmark result about the Type3 grid on the SP

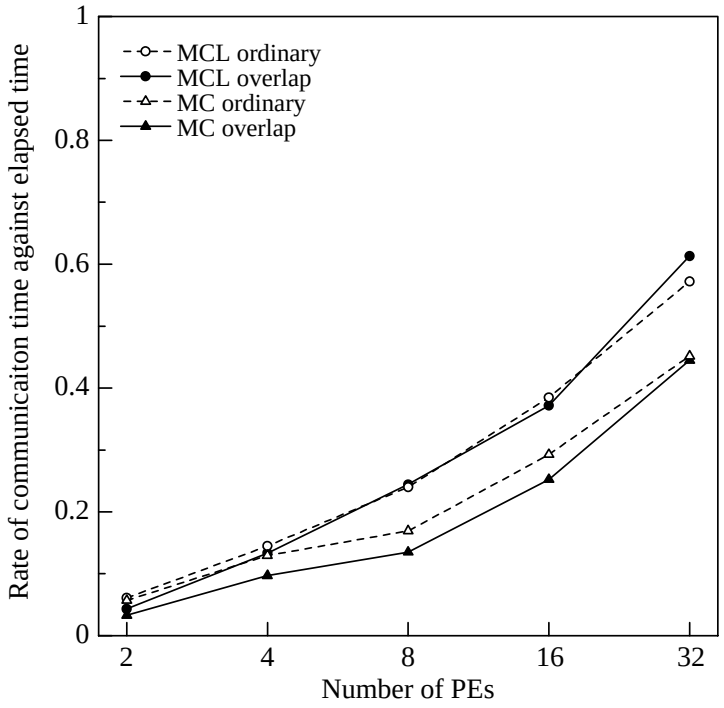

図 6.35: The rate of communication time against the elapsed time of the benchmark result about the Type3 grid on the SP

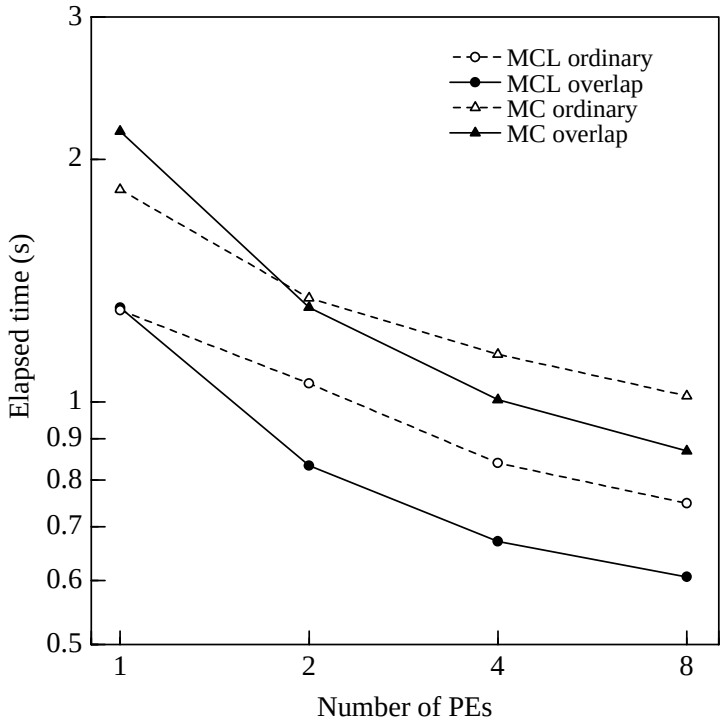

図 6.36: The elapsed time of the benchmark result about the Type1 grid on the Cluster2

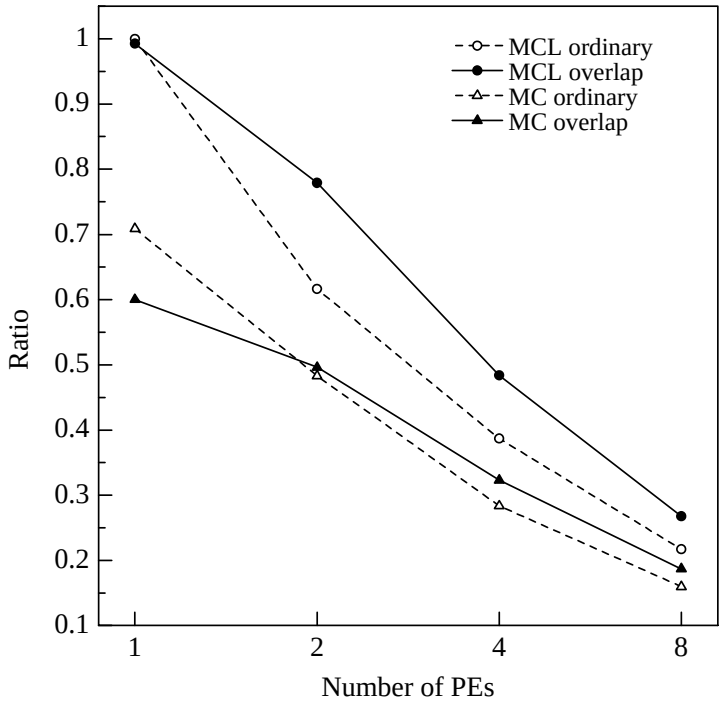

図 6.37: The efficiency of the benchmark result about the Type1 grid on the Cluster2

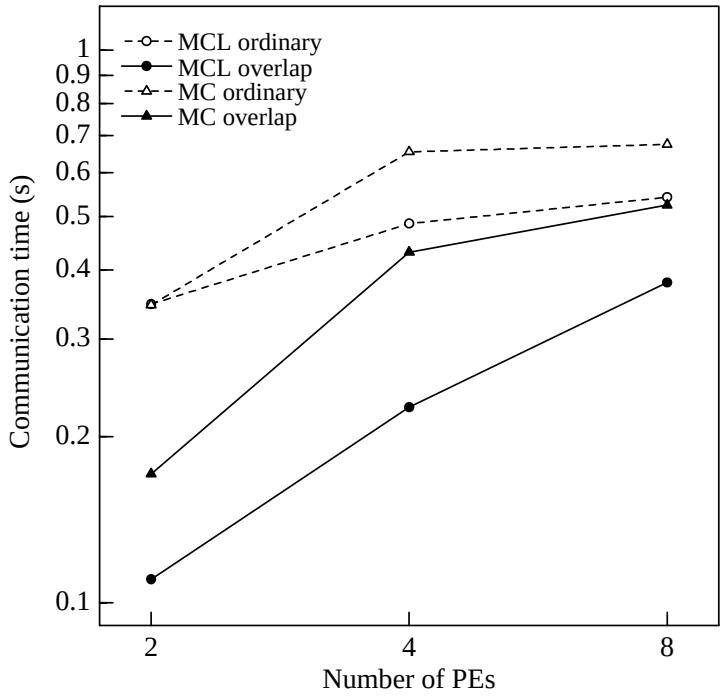

図 6.38: The communication time of the benchmark result about the Type1 grid on the Cluster2

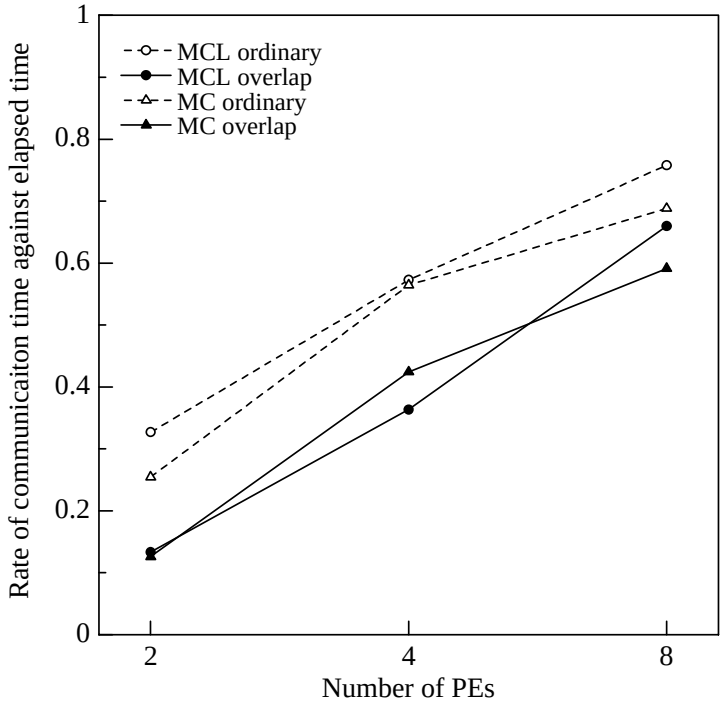

図 6.39: The rate of communication time against the elapsed time of the benchmark result about the Type1 grid on the Cluster2

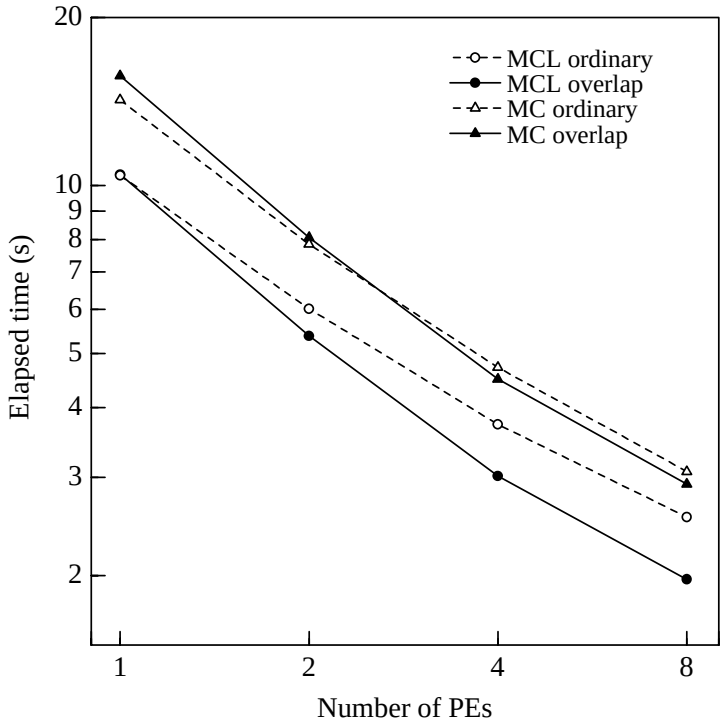

図 6.40: The elapsed time of the benchmark result about the Type2 grid on the Cluster2

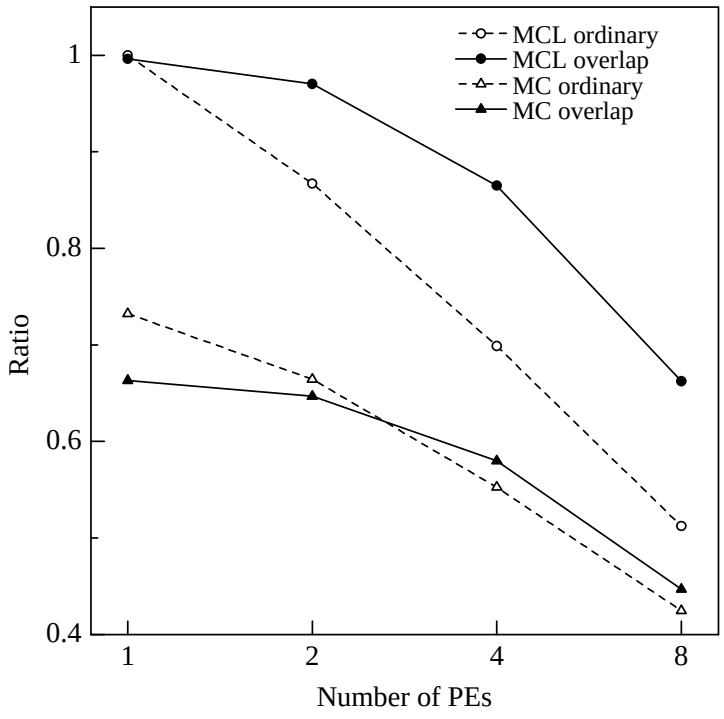

図 6.41: The efficiency of the benchmark result about the Type2 grid on the Cluster2

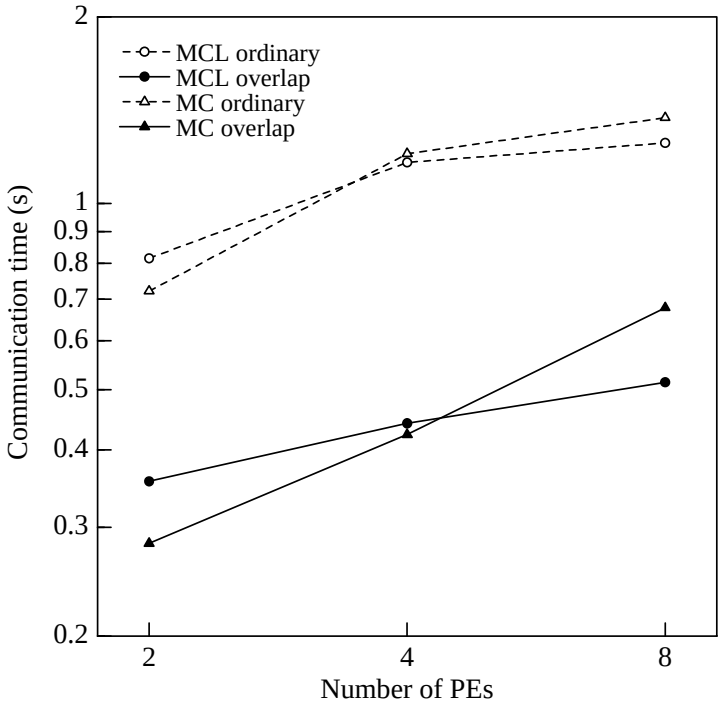

図 6.42: The communication time of the benchmark result about the Type2 grid on the Cluster2

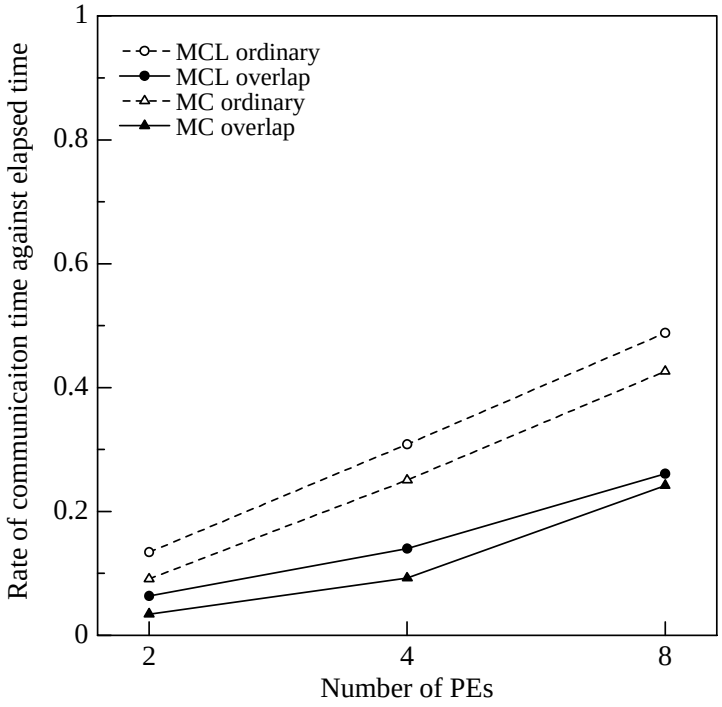

図 6.43: The rate of communication time against the elapsed time of the benchmark result about the Type2 grid on the Cluster2

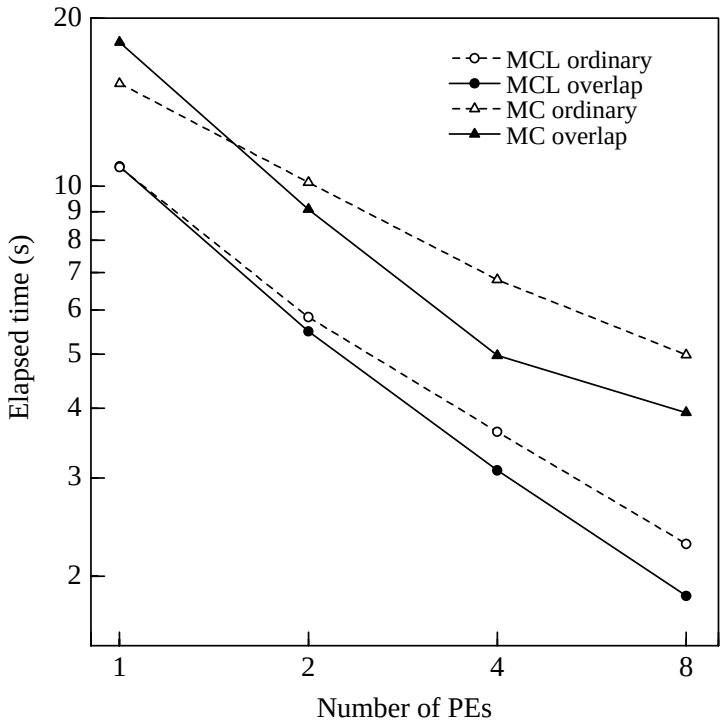

図 6.44: The elapsed time of the benchmark result about the Type3 grid on the Cluster2

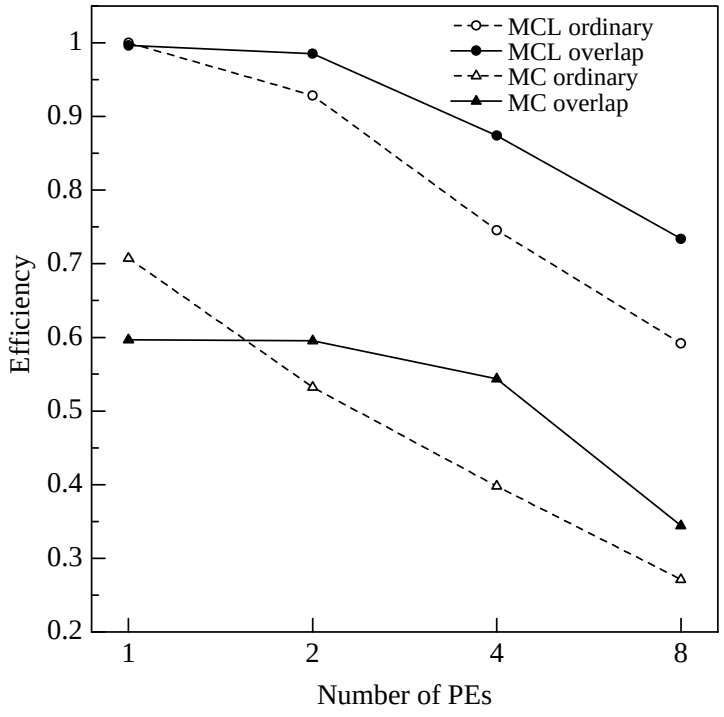

図 6.45: The efficiency of the benchmark result about the Type3 grid on the Cluster2

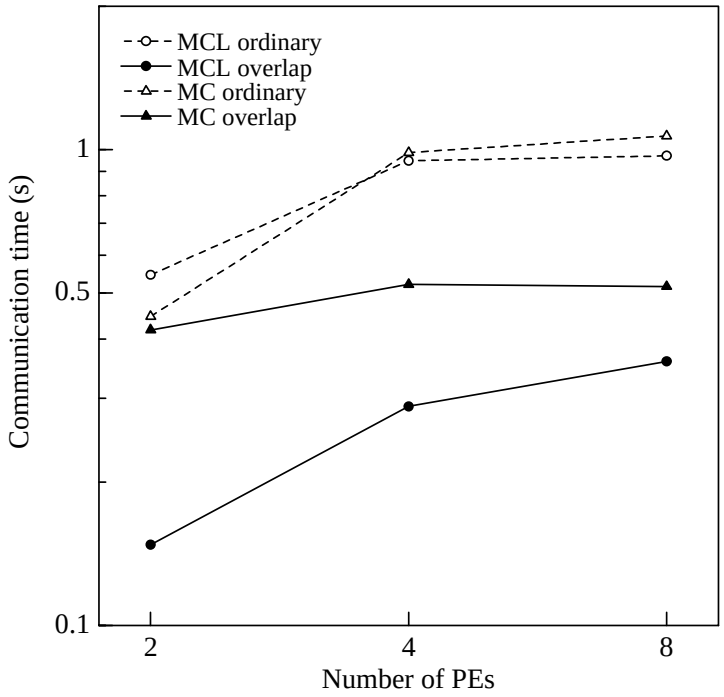

図 6.46: The communication time of the benchmark result about the Type3 grid on the Cluster2

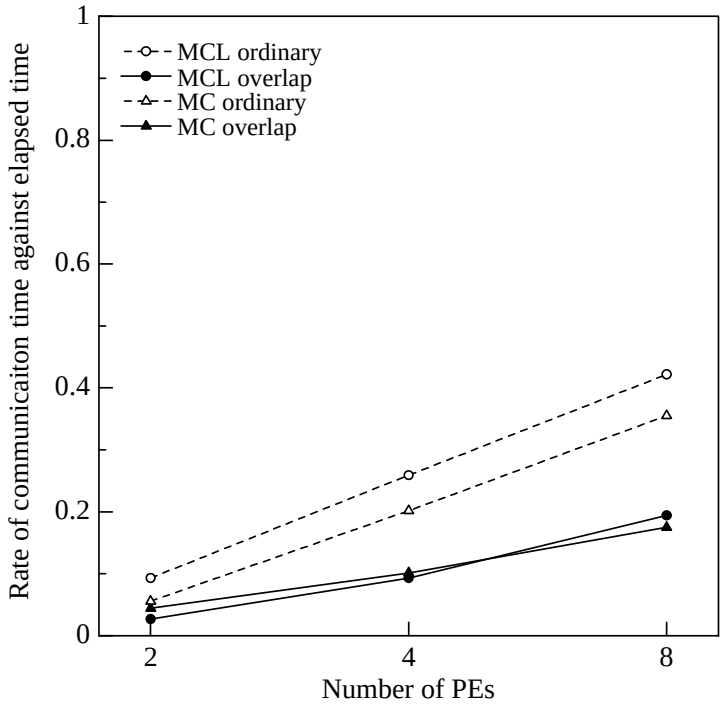

図 6.47: The rate of communication time against the elapsed time of the benchmark result about the Type3 grid on the Cluster2

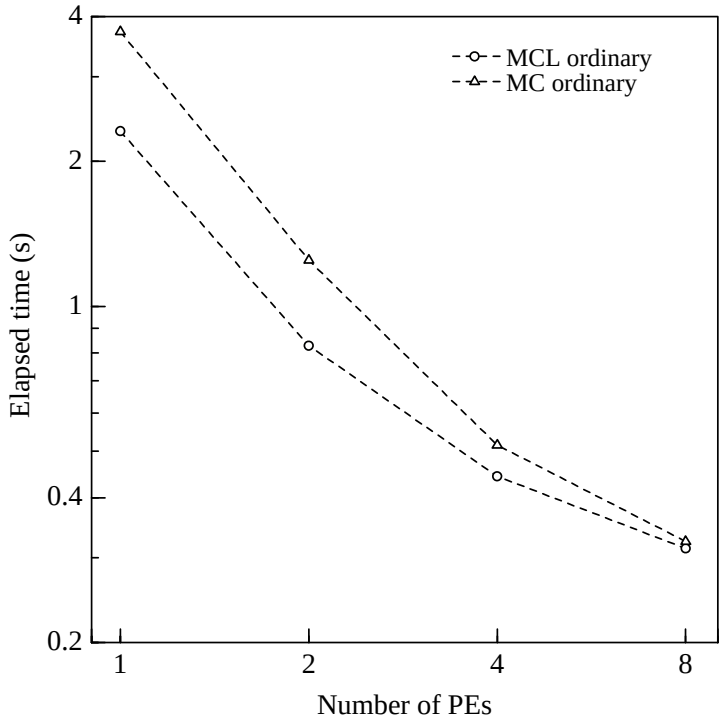

図 6.48: The elapsed time of the benchmark result about the Type1 grid on the Onyx

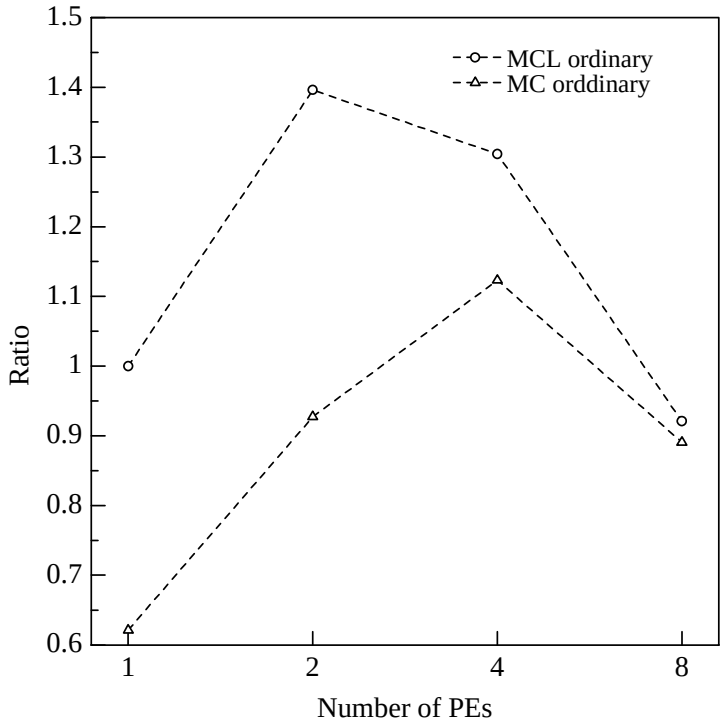

図 6.49: The efficiency of the benchmark result about the Type1 grid on the Onyx

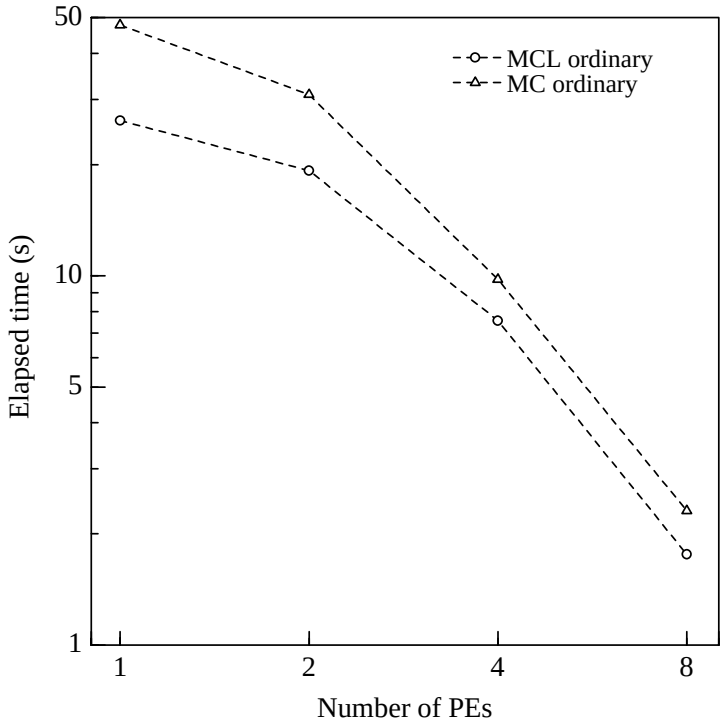

図 6.50: The elapsed time of the benchmark result about the Type2 grid on the Onyx

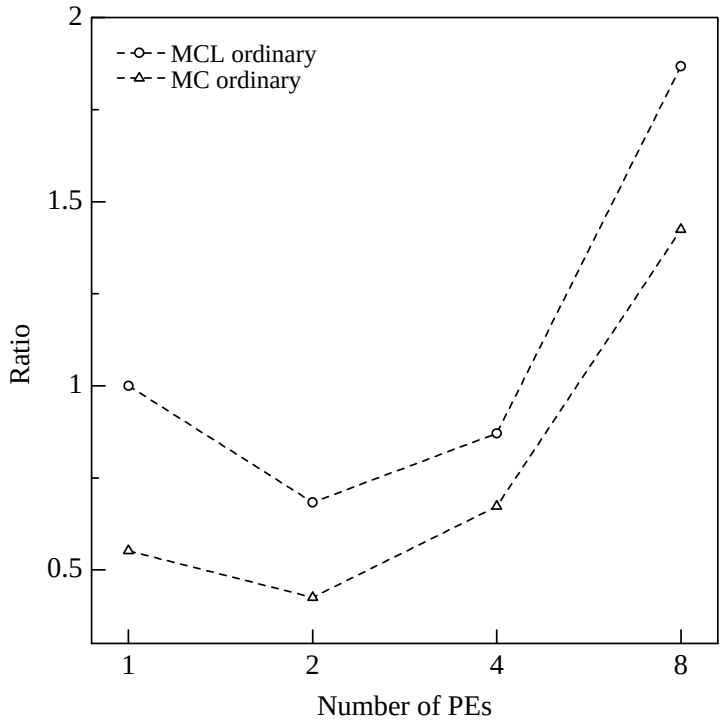

図 6.51: The efficiency of the benchmark result about the Type2 grid on the Onyx

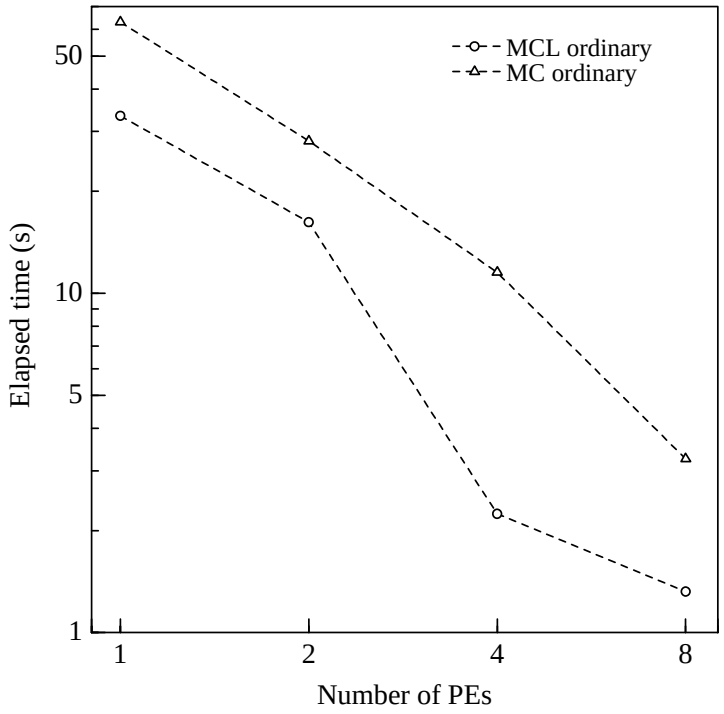

図 6.52: The elapsed time of the benchmark result about the Type3 grid on the Onyx

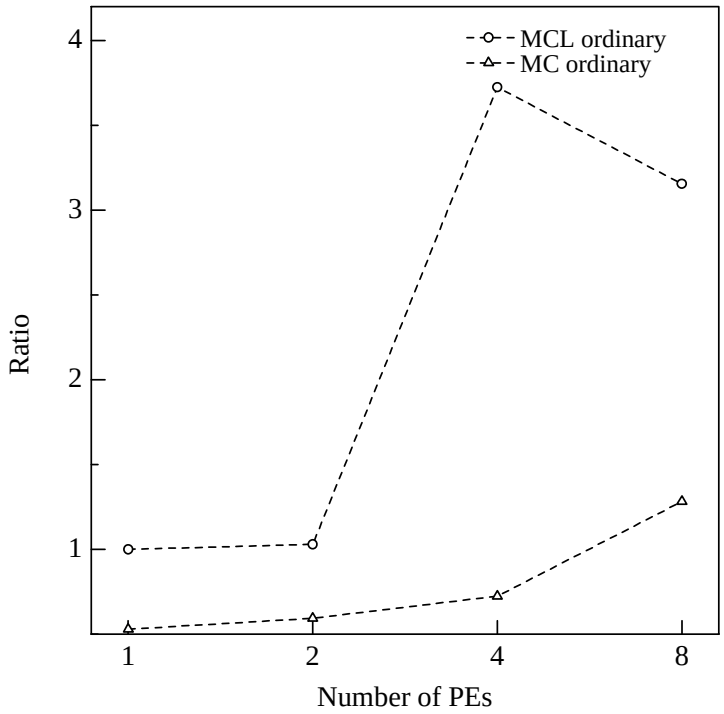

図 6.53: The efficiency of the benchmark result about the Type3 grid on the Onyx

総経過時間を示した図 6.12，6.16，6.20，6.24，6.28，6.32，6.36，6.40，6.44，6.48，6.50，6.52，から，MCLSOR 法は，Multi-color SOR 法よりも計算効率が高いことが分かった．T3E，SP においては格子サイズに関係なく 30 ～ 40 % 程度，Cluster2 では 20 ～ 50 % 程度，Onyx では最大が 80 %，最小が 3 %，全般的に 30 % 程度の性能向上が見られた．また，T3E，SP と Cluster2 での 1 PE における MCLSOR 法，Multi-color SOR 法ともに *overlap* は *ordinary* よりも処理性能が低い．実装上，計算手順の変更のため計算効率が落ちたものであり，Jacobi 法を用いた *overlap* においても見られた現象である．T3E を除いて PE 数の増加に伴って，*overlap* の処理性能が高くなった．

図 6.12, 6.16, 6.20, 6.13, 6.17, 6.21 より T3E では全ての場合において MCLSOR 法が Multi-color SOR 法に比べて並列処理性能が高い．1 を越える割合が Type2, Type3 格子において見られた．1 PE の計算処理性能が低くなったためである．*overlap* は，Jacobi 法を用いた場合と同様に *ordinary* に比べて処理性能が低くなった．得られた *Ratio* において，MCLSOR 法では *ordinary* と *overlap* の差は僅か 5 % 程度である．また，Multi-color SOR 法において *overlap* は *ordinary* に対して，Type1 格子では 5 ～ 10 % 程度，Type2 格子では 4 ～ 7 % 程度，Type3 格子では 5 ～ 15 % 程度の差が見られた．図 6.14，6.15，6.18，6.19，6.18，6.19 より Multi-color SOR 法のデータ通信時間は T3E において総じて MCLSOR 法に対して長い時間がかかった．しかし，総経過時間に対するデータ通信時間の割合に大きな差は見られなかった．また，通信隠蔽処理により MCLSOR 法の低並列の場合では *overlap* の通信時間は減少した．しかし，高並列の場合や Multi-color SOR 法では，大きな違いが見られなくなった．

図 6.24，6.28，6.32，6.25，6.29，6.33 より MCLSOR 法の場合では，Type1 格子では，全ての場合において *overlap* の処理性能が高いが，その他の格子では，計算量の低下に伴って若干高速になった．処理効率の状態を見ても PE 数が増え，計算量が低下するにつれて高速になった．また，Jacobi 法と異なり，PE 数が少ない場合，*overlap* は十分効果を示さなかった．*Ratio* を見た場合 32 PE，Type1 格子で 20 % 程度，Type2，Type3 格子で 40 % 程度となり，大きく *Ratio* が落ちることになった．また，MCLSOR 法における *overlap* は，Type1 格子では PE 数に関わらず，*ordinary* よりも *Ratio* が高い．また，Type2，Type3 格子においては，

16 PE を越えた場合 *ordinary* よりも *Ratio* が高くなった．Jacobi 法の *overlap* と *ordinary* とは異なる現象が見られた．図 6.26，6.27，6.30，6.31，6.30，6.31 より SP では，Type1 格子では高並列になるほど，MCLSOR 法と Multi-color SOR 法のデータ通信時間の差が開いた．しかし，その他では高並列になるほど 2 つの差が小さくなった．データ通信の割合は Type1 格子では全てにおいて大きな差が見られず，その他では高並列になるほど，MCLSOR 法の方が Multi-color SOR 法のデータ通信割合より小さくなった．特に 4 PE を境に大きく変化した．Type1 格子では，通信隠蔽の効果が現れデータ通信時間が減少したが，その他では大きく減少しなかった．また，通信隠蔽処理によって，総じて総経過時間中のデータ通信処理時間は短くなった．

図 6.36，6.40，6.44，6.37，6.41，6.45 より MCLSOR 法における *overlap* は *ordinary* に対して明確に有効である．Type1 格子では 20 % 程度，Type2 格子では 10 $\sim$ 20 % 程度，Type3 格子では 5 $\sim$ 20 % 程度高速に処理された．また，Multi-color SOR 法においても，1 PE の場合は *ordinary* の方が高速であるが，2 PE より多い場合 *overlap* が高速になった．また，MCLSOR 法の *Ratio* は，Type1 格子において PE 数が増加するほど *ordinary* と *overlap* の差が僅かに小さくなり，8 PE では，7 % 程度の差になった．Type2 格子では，*overlap* と *ordinary* の差は非常に大きく 18 % 程度の差があらわれた．Type3 格子でも PE の増加と伴に *Ratio* の差も増加した．8 PE 時に 15 % 程度の差があらわれた．図 6.38，6.39，6.42，6.43，6.42，6.43 より通信隠蔽処理の効果が非常に高い．特に MCLSOR 法での効果が高く，*ordinary* と比較して 2 倍程度短い時間である．そのため，データ通信処理時間の割合も 20 % 程度短くすることが出来る．Type1 格子以外での *ordinary* の Multi-color SOR 法と MCLSOR 法では，8 PE において Multi-color SOR 法は 3 次元分割を用いるため，データ通信量が小さくなる．しかし，データ通信時間に大きな差が見られなかった．

図 6.48，6.50，6.52，6.49，6.51，6.53 より MCLSOR 法と Multi-color SOR 法において，自動並列コンパイラが十分に機能し，ディレクティブの挿入もなしに十分な並列性能が得られることが分かった．Type3 格子を除いて，PE 数の増加と伴に Multi-color SOR 法と MCLSOR 法の差が小さくなった．総経過時間において，グラフの傾向が格子サイズによって異なった．Type1 格子は，PE 数が少ない

段階で 2 次キャッシュに収まるサイズのため，経過時間が飽和するような現象が見られ，Type2，Type3 では，PE 数が増えた段階で 2 次キャッシュの影響が現れるため，経過時間の減少が加速するような現象が見られた．*Ratio* にもこの影響が見られ，ほぼ全ての場合において，*Ratio* が 1 を超える現象が見られた．

6.3.2 共有/分散プログラムの効果

SP のような共有/分散型並列計算機システム (共有メモリ型並列計算機 (Symmetric Multi Processor(SMP)) システムをネットワークで疎結合した計算機システム) を並列プログラミングの観点から見た場合，多くのプログラムでは単純な分散システムと見ることが多い．すなわち，共有メモリシステムのプログラミングもプロセスを CPU 数分立ち上げることで，分散メモリシステム的に用いる方法である．この方法は，既存の MPI プログラムを修正すること無しに用いることが可能であるため，良く用いられる．しかし，性能やシステム構成を考慮した場合では，共有メモリ部分にプロセス間通信を含めたデータ通信負荷の増加やネットワーク帯域を SMP 部分の CPU 数で共有することなど性能低下を引き起こす原因となることが多い[1]．

共有/分散型並列計算機システムを有効に用いるためには，共有/分散を意識した共有/分散プログラミングが必要である．共有メモリ部分 (SMP 部分) は共有メモリプログラミングを用いて，プロセス間通信を行なう必要が無い，スレッドレベルの並列化を行い．分散メモリ部分は MPI を用いた通常の並列化を行なう．この方法を用いれば共有メモリ部分のプロセス間通信が無くなり，ネットワーク帯域の共有による通信性能の低下や並列度を下げることが可能となる．

共有メモリ部分のプログラミングは，様々な方法がある．自動並列化コンパイラを用いる方法や並列化指示行を用いる方法，Pthread 等のスレッドプログラミングを用いる方法がある．上述のように MCLSOR 法は並列化が可能である．しかし，コンパイラが自動並列できるかどうかは，コンパイラの処理性能次第であり，SP の XL Fortran は自動的に並列化出来なかった．そのため，本研究は XL Fortran の並列化指示行を用い，共有メモリ部分の並列化を行なった．挿入した指示行は 1

[1]前述の SP の使用方法は，1SMP node に 1 プロセスのみを立てているため，SP を用いる場合に通信負荷の問題を意識する必要が無く，単純な分散メモリシステムとして用いた

行だけである．

SP での通信隠蔽を用いない MCLSOR 法における，共有/分散プログラミングを用いた場合と分散プログラミングを用いた場合の処理性能を評価した．SP は 4CPU 構成の SMP システムをネットワークで疎結合した SMP クラスタであるため，共有/分散では，共有部分に 4CPU と SMP 部分 32node を用いた 128PE を用いる．分散では，1SMP 部分に 4 プロセス用いた 128PE である．図 6.54，6.56，6.58 に総経過時間と MCLSOR 法に対する割合 ($Ratio$)

$$Ratio = \frac{Etime_p}{Etime_s \times PEs} \tag{6.2}$$

を示し，図 6.55，6.57，6.59 にデータ通信時間と通信割合

$$rate\ of\ communicationtime = \frac{Mtime}{Etime_p} \tag{6.3}$$

を示した．$Mtime$ は通信時間を示し，$Mtime = Etime_p - Ctime$ から得る．$Ctime$ は計算処理時間である．

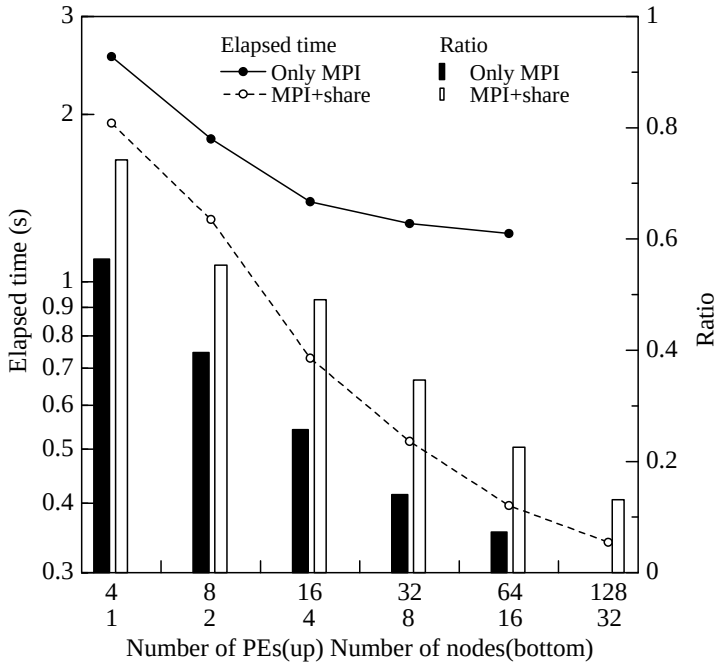

図 6.54: The elapsed time of the benchmark result about the Type1 grid on the SP using the shared-distributed programming

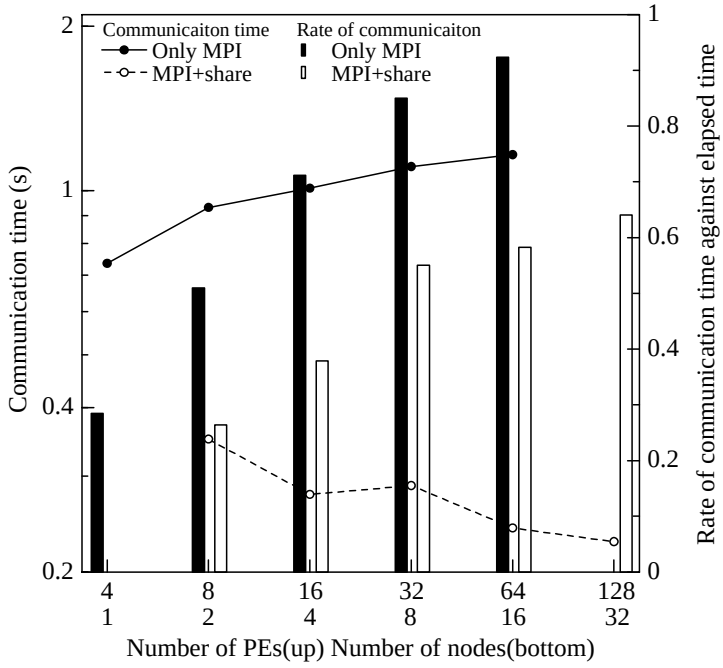

図 6.55: The communication time and the rate of the communication time of the benchmark result about the Type1 grid on the SP using the shared-distributed programming

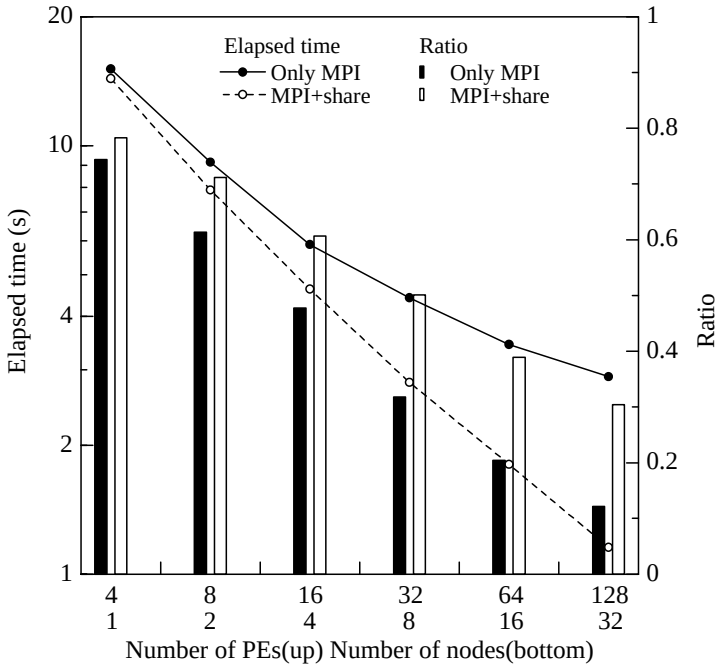

図 6.56: The elapsed time of the benchmark result about the Type2 grid on the SP using the shared-distributed programming

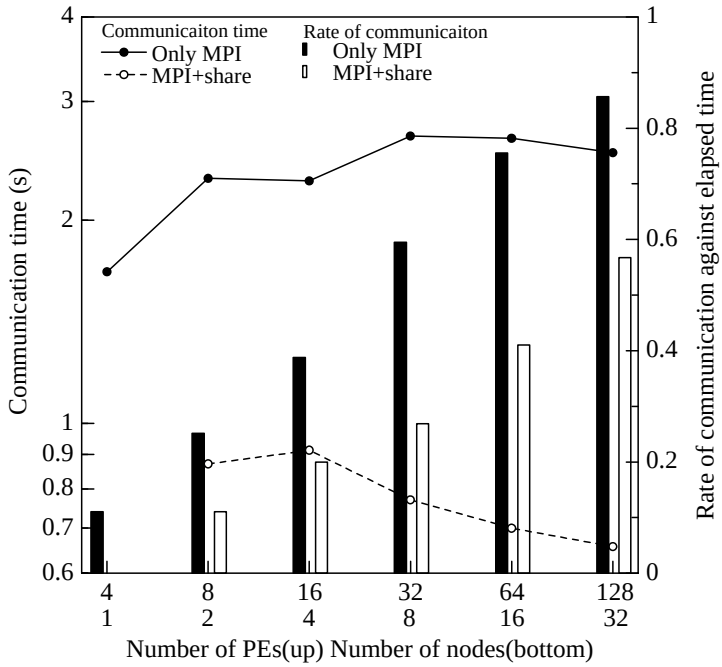

図 6.57: The communication time and the rate of the communication time of the benchmark result about the Type2 grid on the SP using the shared-distributed programming

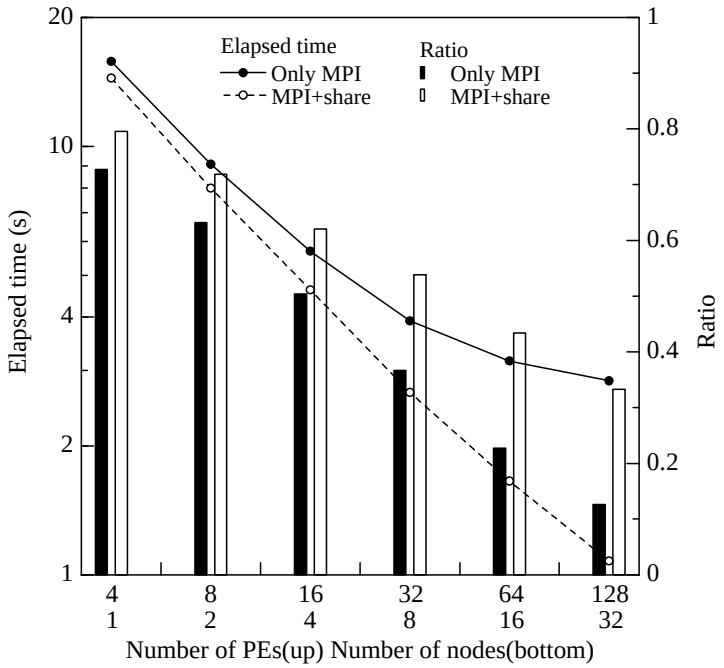

図 6.58: The elapsed time of the benchmark result about the Type3 grid on the SP using the shared-distributed programming

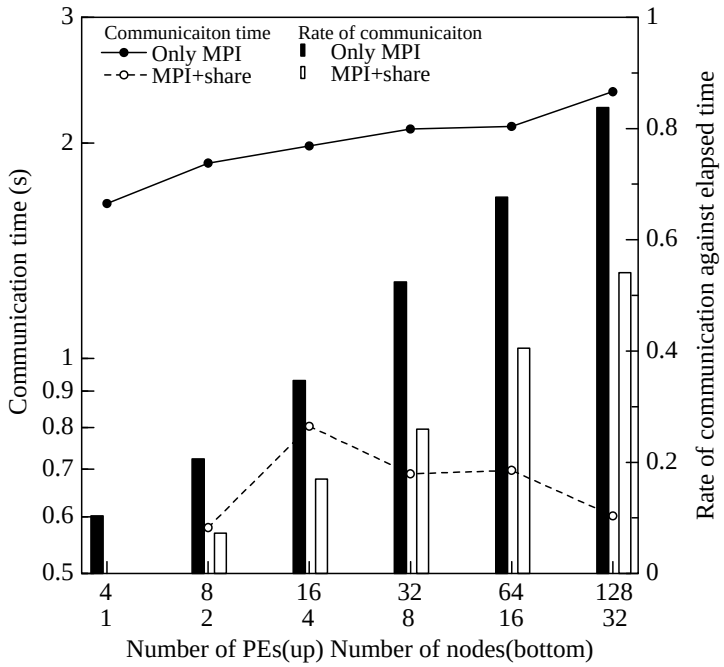

図 6.59: The communication time and the rate of the communication time of the benchmark result about the Type3 grid on the SP using the shared-distributed programming

図 6.54，6.56，6.58 より，MPI と並列化指示行を用いた共有/分散プログラムの方が MPI だけの分散プログラムより並列性能の低下が緩やかであることが分かった．特に分散プログラムで高並列になる場合に対して，共有/分散プログラムの並列性能は非常に高い．128PE(32node) において，共有/分散プログラムの *Ratio* は，分散プログラムに対して 20% 程度高い．また，単一 node の処理効率は，Type1 格子で 15%，その他では 3,4% の差がある．このことから，共有メモリシステムでのプロセス間通信は非常に重い操作であり，計算粒度が小さくなればなうほど，共有プログラムを用いる有効性が高くなった．

図 6.55，6.57，6.59 より，分散プログラムでは，共有/分散プログラムの 2 倍以上の通信時間を要していることが分かった．また，共有/分散プログラムでは，高並列につれてデータ通信時間の減少が見られた．また，総経過時間に対するデータ通信時間の割合は，128PE(32node) 時に分散プログラムでは 80 % 以上を示すのに対して，共有/分散プログラムでは 60 % 程度である．総じて 20 % 程度の差が現れた．

6.3.3 収束性能と並列性能

図 6.60，6.61 に T3E，図 6.62，6.63 に SP，図 6.64，6.65 に Cluster2 における Type1，Type2 の結果をそれぞれ示した．表 6.2 に収束までの反復回数を示した．図中には，Jacobi 法，MCLSOR 法，Multi-color SOR 法，Pipeline SSOR 法 を示した．これらの方法は，全て通信隠蔽の手法を用いたものの実装であるが，全ての場合で逐次処理と並列処理での収束性に変化は無い．収束性の高さと実際の計算時間とは別の問題であり，収束性能が高いソルバーが良いとは限らない．収束性能と MAC 法においてどの通信隠避手段を用いると効率が良いかが明らかになる．

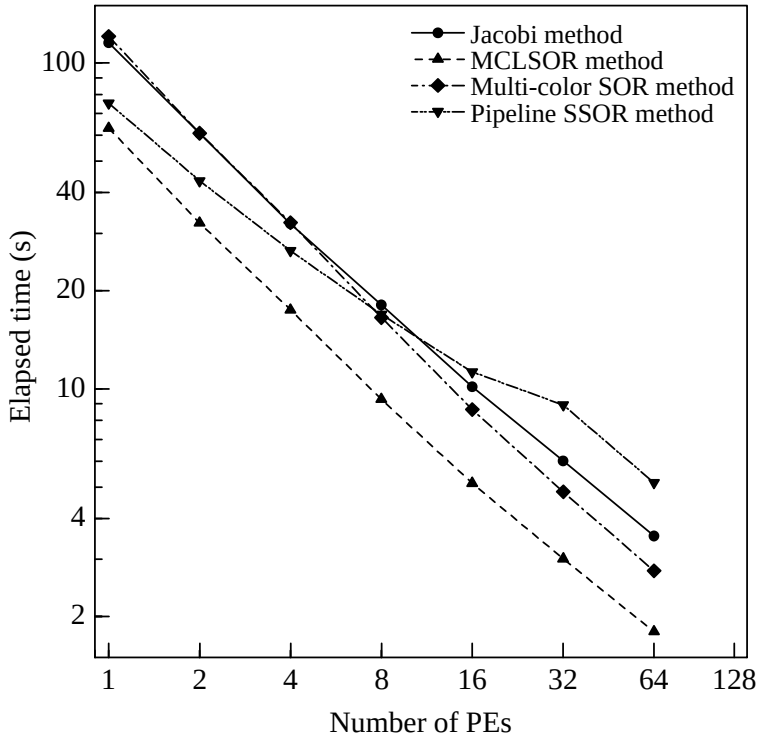

図 6.60: The benchmark result about Type1 grid on the T3E

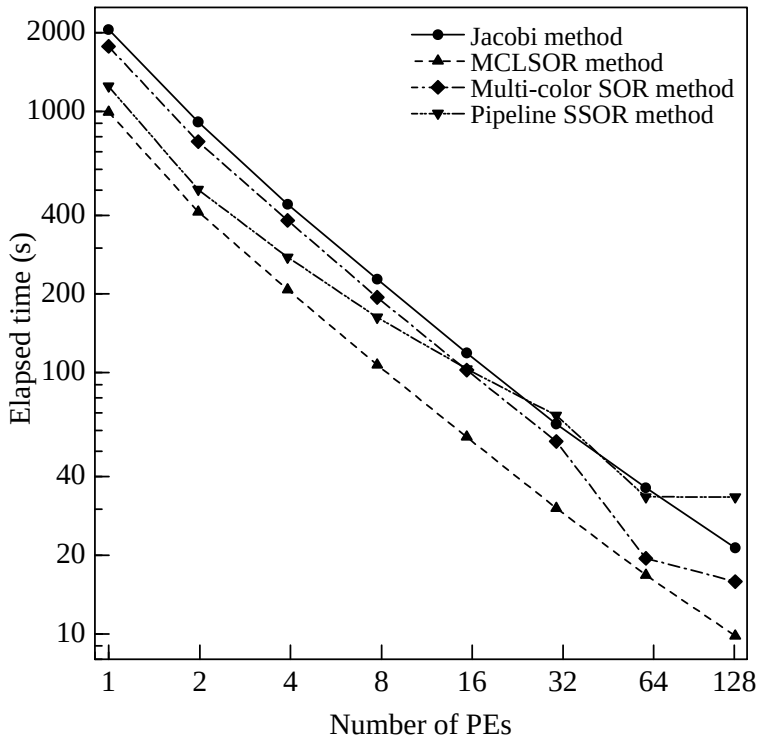

図 6.61: The benchmark result about type2 grid on the T3E

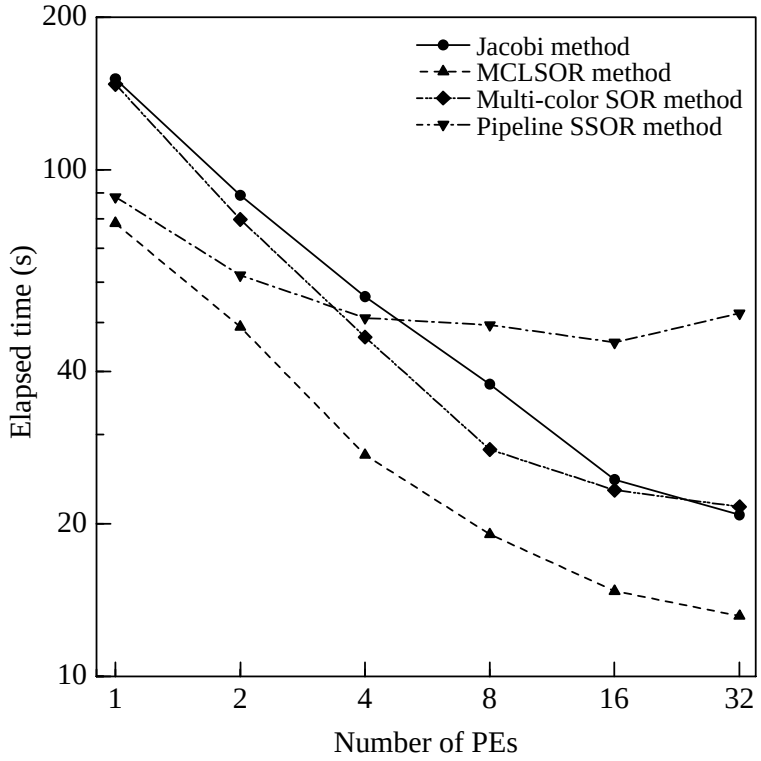

図 6.62: The benchmark result about Type1 grid on the SP

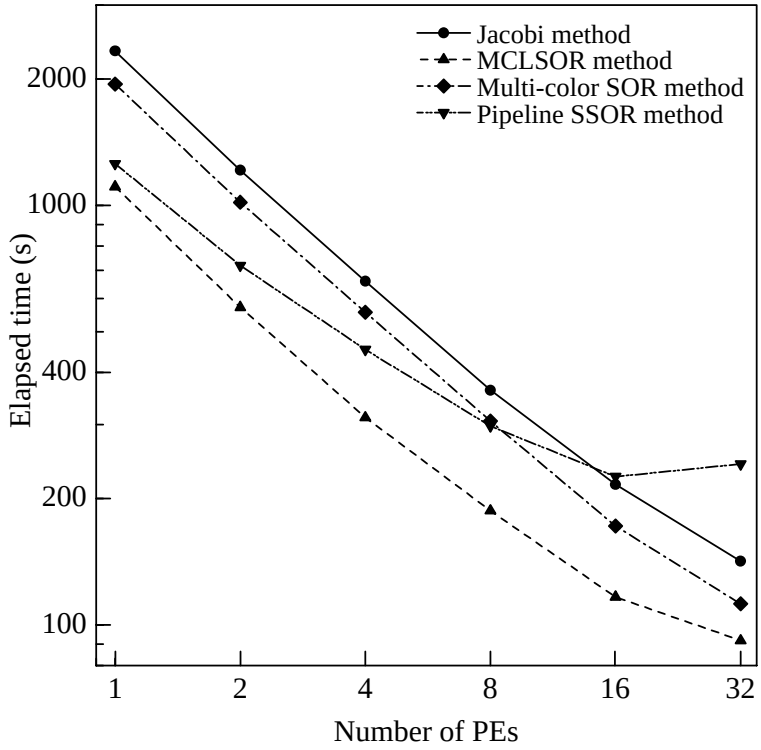

図 6.63: The benchmark result about type2 grid on the SP

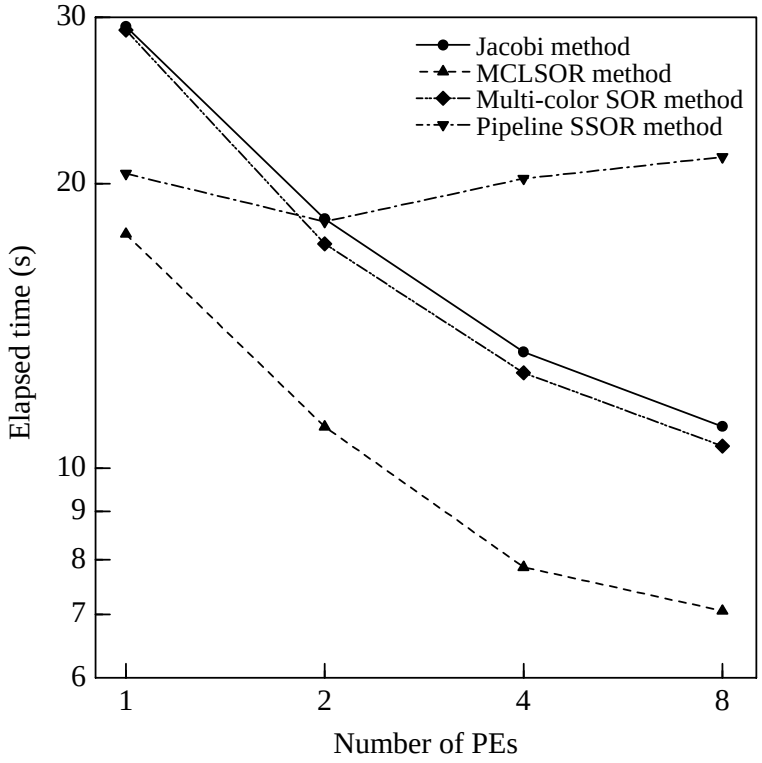

図 6.64: The benchmark result about Type1 grid on the Cluster2

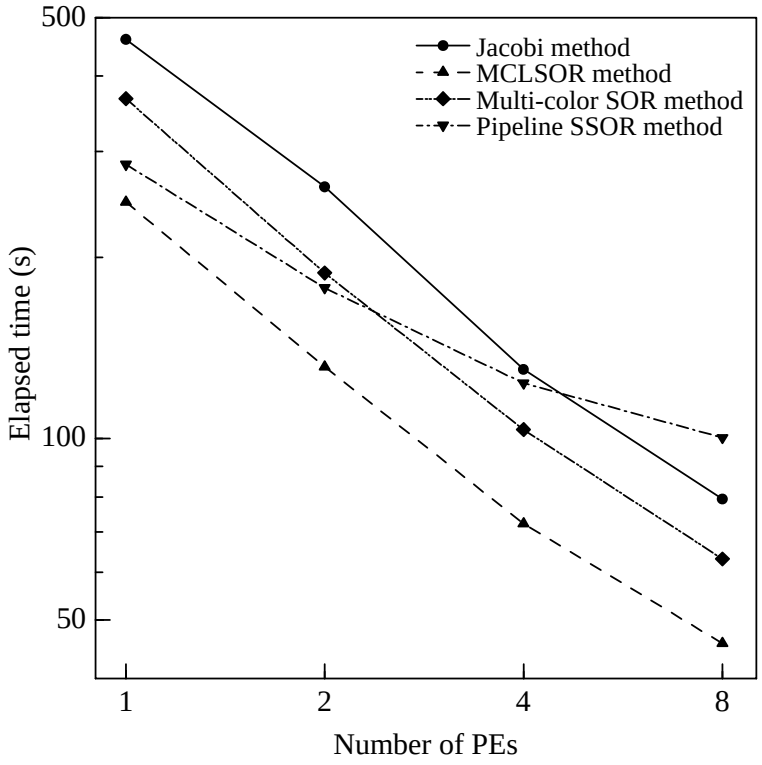

図 6.65: The benchmark result about type2 grid on the Cluster2

表 6.2: The convergence profile about the number of iterations

Method	Number of iterations	
	Type1	Type2
Jacobi	5,099	10,020
Multi-color SOR	2,560	4,538
MCLSOR	2,565	4,549
Pipeline SSOR	3,470	5,806

表 6.2 が示すように 収束性能は Multi-color SOR 法，MCLSOR 法，Pipeline SSOR 法，Jacobi 法の順に良い．Multi-color SOR 法と MCLSOR 法は，Jacobi 法の約半分の反復回数である．Multi-color SOR 法と MCLSOR 法は反復回数に数反復の違いがあった．Pipeline SSOR 法は Multi-color SOR 法や MCLSOR 法に比べて 40 % 程度反復回数が多くなった．

図 6.60，6.61，6.62，6.63，6.64，6.65 から MCLSOR 法が最も処理性能が高いことが分かった．Pipeline SSOR 法は，PE 数が少ない場合では，MCLSOR 法についで高速であるが，PE 数の増加に伴い性能が飽和した．Multi-color SOR 法は反復回数が同程度である MCLSOR 法に比べると，非常に処理性能が低いことが分かった．

T3E において，Type1 格子では MCLSOR 法は，Multi-color SOR 法に比べて 35 ～ 50 % 程度処理性能が高く，Jacobi 法に比べ 50 % 程度処理性能が高く，そして，Pipeline SSOR 法に比べて 20 ～ 70 % 程度高速であった．Multi-color SOR 法は Jacobi 法と大きな処理性能差が見られなかった．Pipeline SSOR 法は 8 PE を越えたあたりから Jacobi 法と Multi-color SOR 法より処理性能が落ちた．Type2 格子では MCLSOR 法は，Multi-color SOR 法に比べて 64 PE を除いて 40 ～ 45 % 程度処理性能が高く，Jacobi 法に比べ 50 数% 程度処理性能が高く，そして，Pipeline SSOR 法に比べて 20 ～ 70 % 程度高速であった．Multi-color SOR 法は Jacobi 法と 64，128 PE を除いて 15 % 程度の差しか見られなかった．Pipeline SSOR 法は 32 PE を越えたあたりから Jacobi 法と Multi-color SOR 法より処理性能が落ちた．

SP において，Type1 格子では MCLSOR 法は，Multi-color SOR 法に比べて

30 ～ 45 % 程度処理性能が高く，Jacobi 法に比べ 35 ～ 50 % 程度処理性能が高く，そして，Pipeline SSOR 法に比べて 10 ～ 75 % 程度高速であった．Multi-color SOR 法は PE 数が増えるにつれて Jacobi 法と大きな処理性能差が見られなくなった Pipeline SSOR 法は 8 PE を越えたあたりから Jacobi 法と Multi-color SOR 法より処理性能が落ちた．Type2 格子では MCLSOR 法は，Multi-color SOR 法に比べて 40 % 程度処理性能が高く，Jacobi 法に比べ 50 % 程度処理性能が高く，そして，Pipeline SSOR 法に比べて 10 ～ 60 % 程度高速であった．Multi-color SOR 法は Jacobi 法に対して 20 % 程度の差しか見られなかった．Pipeline SSOR 法は 16 PE を越えたあたりから Jacobi 法と Multi-color SOR 法より処理性能が落ちた．

Cluster2 において，Type1 格子では MCLSOR 法は，Multi-color SOR 法に比べて 38 % 程度処理性能が高く，Jacobi 法に比べ 40 % 程度処理性能が高く，そして，Pipeline SSOR 法に比べて 10 ～ 70 % 程度高速であった．Multi-color SOR 法は Jacobi 法と大きな処理性能差が見られなくなった Pipeline SSOR 法は 4 PE を越えたあたりから Jacobi 法と Multi-color SOR 法より処理性能が落ちた．Type2 格子では MCLSOR 法は，Multi-color SOR 法に比べて 30 % 程度処理性能が高く，Jacobi 法に比べ 45 % 程度処理性能が高く，そして，Pipeline SSOR 法に比べて 15 ～ 55 % 程度高速であった．Multi-color SOR 法は Jacobi 法に対して 25 % 程度の差しか見られなかった．Pipeline SSOR 法は 4 PE を越えたあたりから Jacobi 法と Multi-color SOR 法より処理性能が落ちた．

6.3.4 キャッシュ

図 6.66 に T3E 上での Type3 格子を並列計算する際発生するキャッシュミスの状態を模擬するため，Type3 格子を各 PE 数で分割した大きさの格子を 1 PE で計算した場合のキャッシュミスの状態を示した．縦軸は浮動小数点演算数あたりのキャッシュミス，横軸は PE 数を取った．図 6.67 は，上記の状態における Mflop/s 値の結果を示した．計測には CRAY Inc. の Performance analysis tool を用いた．図 6.66 より MCLSOR 法は Multi-color SOR 法に比べて，1 PE の場合を除いて 30 ～ 45 % 程度キャッシュミス回数が少ないことが分かった．MCLSOR 法は 1, 128 PE を除いて 10 浮動小数点演算あたり 2，3 回のキャッシュミスであるが，Multi-color SOR 法では，4 回前後のキャッシュミスが発生する．1 PE での計算の

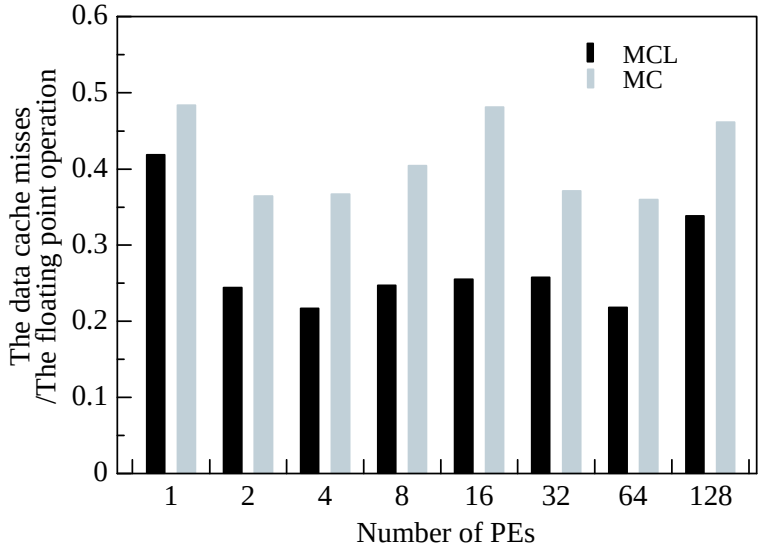

図 6.66: The cache miss ratio on the T3E

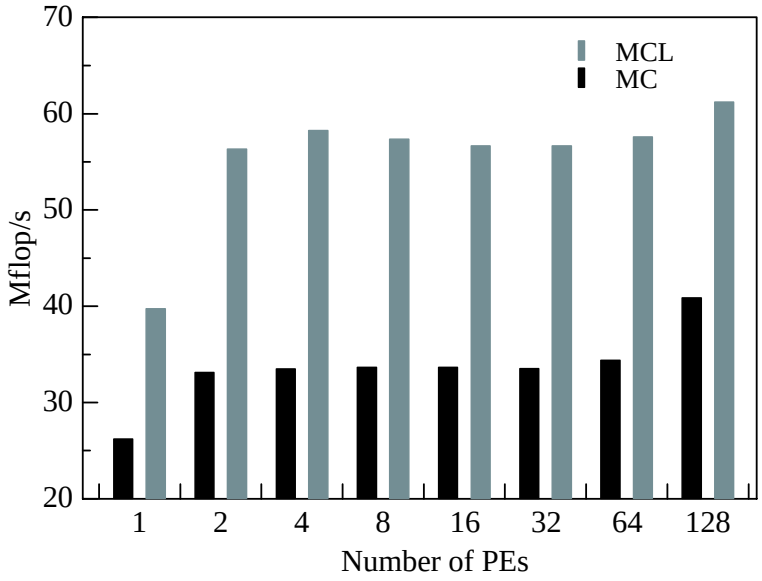

図 6.67: The floating point operations on the T3E

場合 MCLSOR 法と Multi-color SOR 法の差が小さくなった.

図 6.67 より Multi-line SOR 法は 60 Mflop/s 前後，Multi-color SOR 法は 35 Mflop/s 前後の値が得られた．純粋な計算処理性能として MCLSOR 法は 1，128 PE を除いて Multi-color SOR 法より 40 % 程度計算処理性能が高かった．1，128 PE では，33 % 程度に落ちた．また，1 PE では他の PE 数に比べて 20 % 前後処理性能が低かった.

6.3.5 CFD での並列化性能

流れ場

時間ステップを進めた場合の結果として無次元時間 300 の 3 次元流線図を図 6.68，6.69 に，等渦度コンタ図を図 6.70，6.71 に示した．そして，無次元時間 200 ∼ 300 の物理量の平均値を円柱方向で平均化し 2 次元化したものの渦度コンタ図を図 6.72，円柱周りの圧力係数を図 6.73 に示した．圧力係数は

$$C_p = \frac{P}{(1/2)\rho U_0^2} \tag{6.4}$$

となり，P は圧力値，$(1/2)\rho U_0^2$ は一様流中の動圧である．縦軸に圧力係数，横軸に円柱上での位置を示した． 図 6.68，6.69 は，円柱周りの流れの様子を示した．円柱表面は圧力値を数値化して表示した．図 6.68 では，ある時間において，円柱上流の開始点からその時間の流れの状態をリボンであらわした．リボンの色は流速である．円柱後方で流れが剥離し，流れにうねりが生じているのが分かる．また，図 6.69 では，円柱後方に円柱に平行な渦チューブの存在が確認できた．

図 6.70，6.71 は，円柱周りでの渦度分布の等高線を示した．この図も円柱表面は圧力値である．図 6.70 では，円柱の長さ方向に平行な部分の円柱後方の渦度の等高線である．流れが複雑に 3 次元化し，円柱に平行な主渦チューブに取り巻く 2 次的な渦構造の存在が予想される．図 6.71 では，円柱の長さ方向に垂直な円柱中心付近後方の渦度の等高線を示した．渦が交互に連続的に生成されて，Karman 渦列の存在が確認できた．

図 6.72 は，時間平均，空間平均を取った渦度，図 6.73 は，円柱周りの圧力係数分布と理想気体での理論値である．図 6.72 からこの円柱周り流れは本質的に周期性を持つ規則的な流れであることが分かる．周期性を持った K' arm' an 渦列が放出されている．圧力係数は流れの剥離が発生した時点で急激に変化する．図 6.73 は，およそ 75° 付近で剥離していると見られ，一般に $10^3 < Re < 2 \times 10^6$ では層流状態で前方よどみ点から $\theta = 78°$ 付近で剥離することが知られており [38]，数値実験でも同様な結果 [39] が得られている．シミュレーション結果は妥当である．

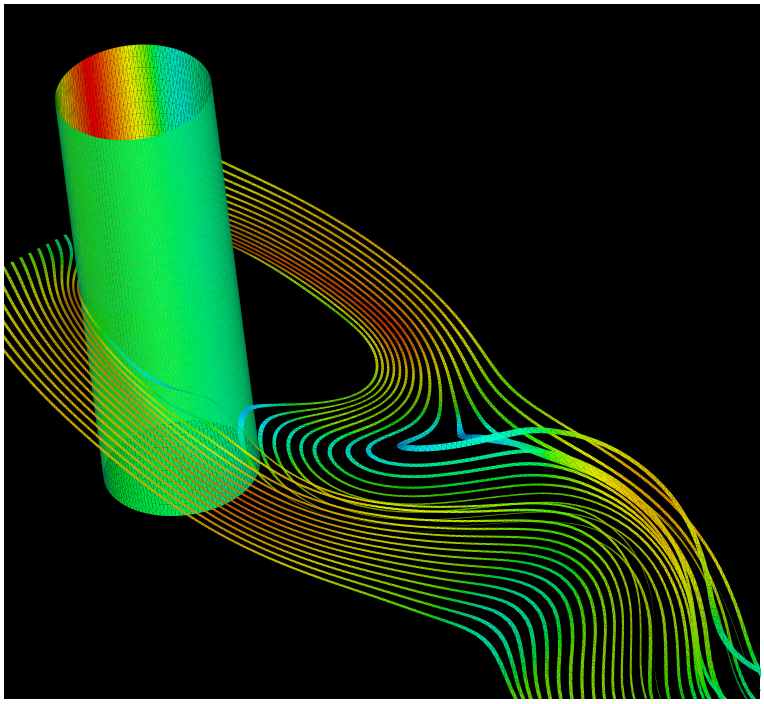

図 6.68: The streamlines

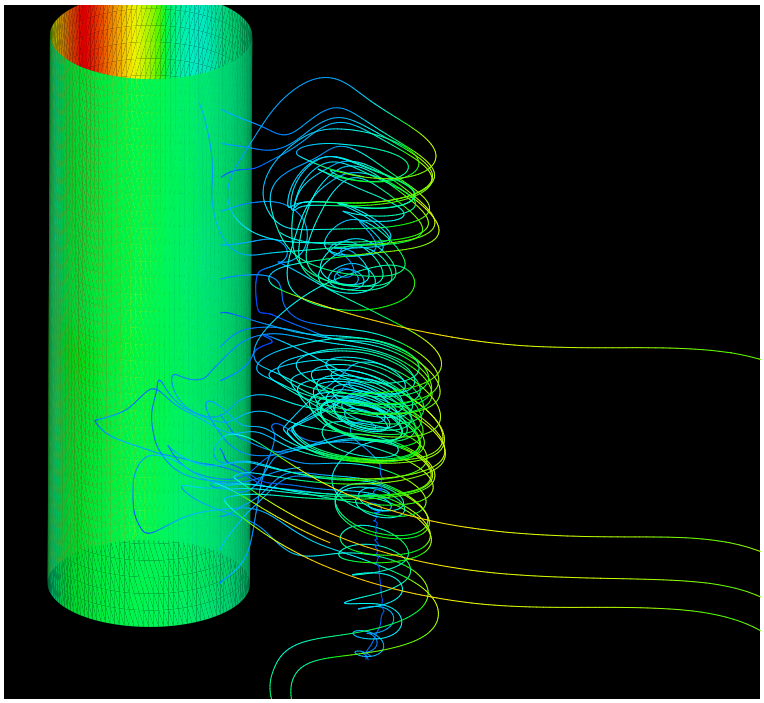

図 6.69: The streamlines

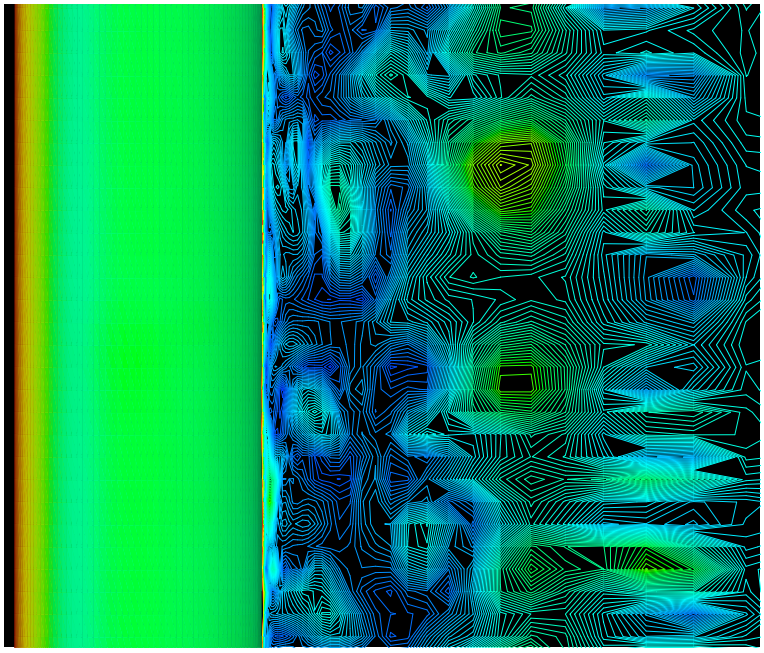

図 6.70: The contour of the vorticity magnitude at X-Y plane

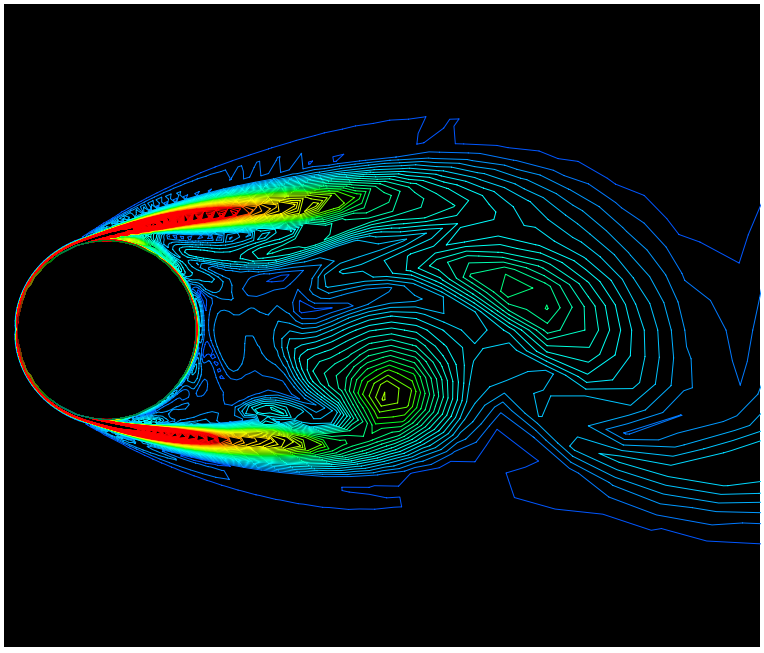

図 6.71: The contour of the vorticity magnitude at X-Z plane

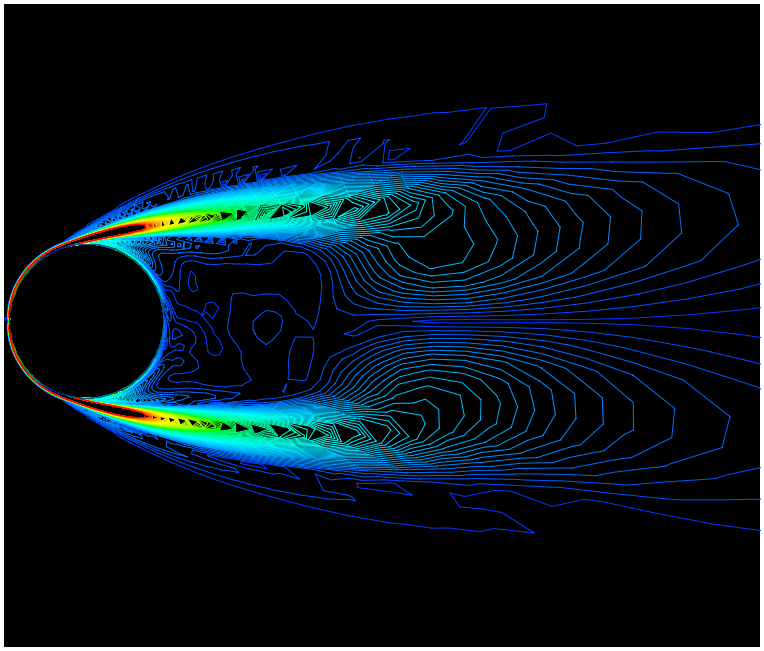

図 6.72: The contour of the averaged vorticity magnitude

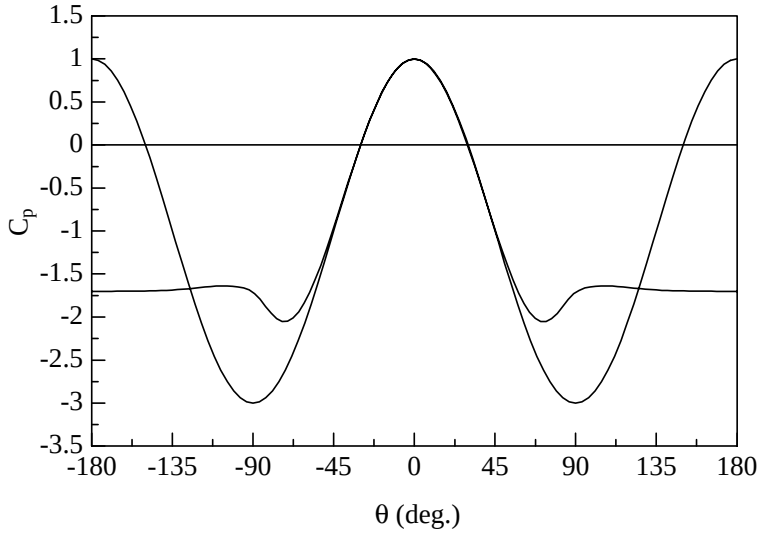

図 6.73: The averaged pressure distributions around the circular cylinder

収束性

図 6.74 に円柱周り流れシミュレーションでの無次元時間 $t = 0.2$ における収束状況を示した．縦軸は各反復ステップでの残差値，横軸は反復ステップ数である．

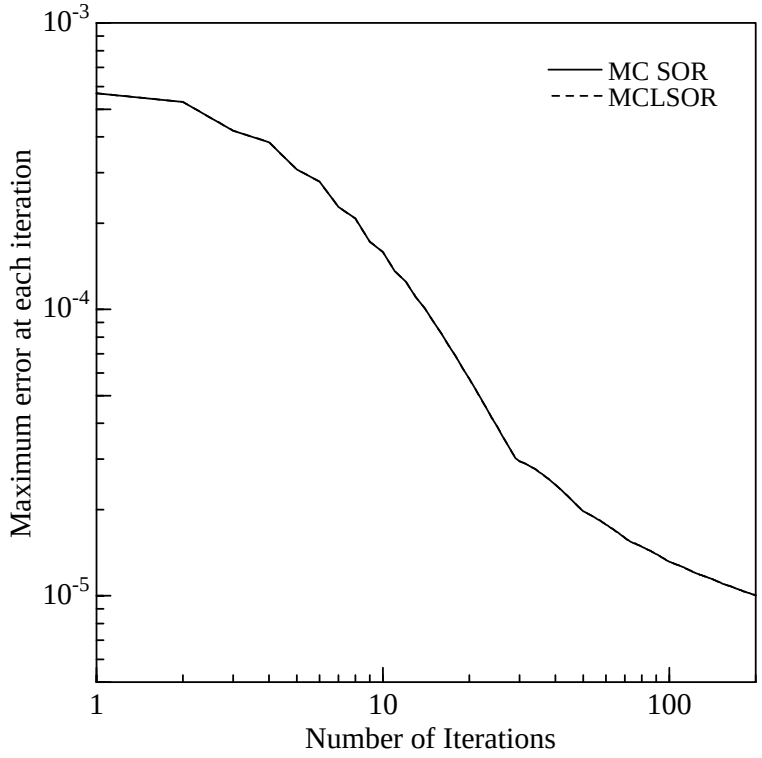

図 6.74: The convergence performance for the CFD simulation at t=0.2

図 6.74 より MCLSOR 法と Multi-color SOR 法に収束性の大きな違いは見られなかった．また，NS 方程式解法においても同様に収束性に変化が無かった．おそらく，前節のキャビティー流れの条件を用いた場合，収束反復終了までの反復回数で数ステップ程度差があったが，収束反復中の残差値の差はごく僅かであると考えられる．また，十分に時間が経過した場合，圧力に関する Poisson 方程式での反復回数は最大で数回〜数十回程度，NS 方程式での反復回数は 3，4 回だった．

並列 CFD 性能

図 6.75 に T3E，図 6.76 に Cluster2，図 6.77 に SP での円柱周り流れの CFD 計算シミュレーションの総経過時間を示した．また，図 6.78 に T3E，図 6.79 に Cluster2，図 6.80 に SP における MCLSOR 法の総経過時間 ($Line_n(Etime_s)$) を基準とした割合 ($Ratio$) を示した．T3E，Cluster2 では 2 PE，SP では共有/分散プログラムの MCLSOR 法の 1 Node(4CPU) での総経過時間を基準とした．

$$Ratio = \frac{\frac{Line_n(Etime_s)}{Etime_p}}{PEs} \tag{6.5}$$

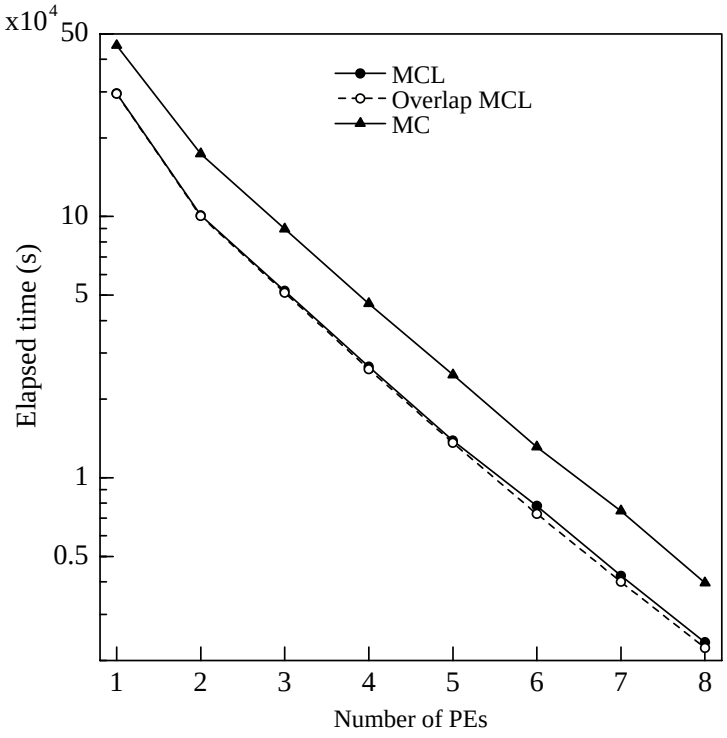

図 6.75: The elapsed time on the T3E

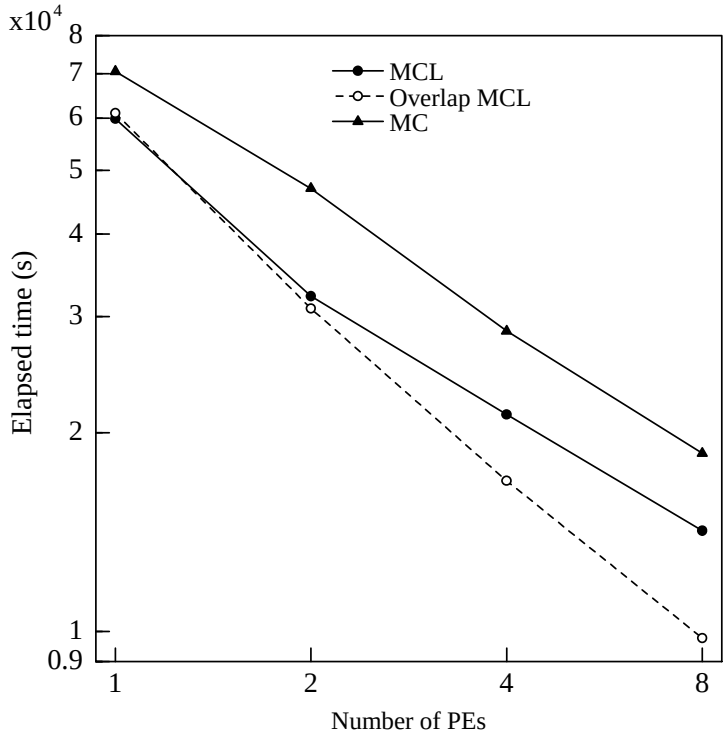

図 6.76: The elapsed time on the Cluster2

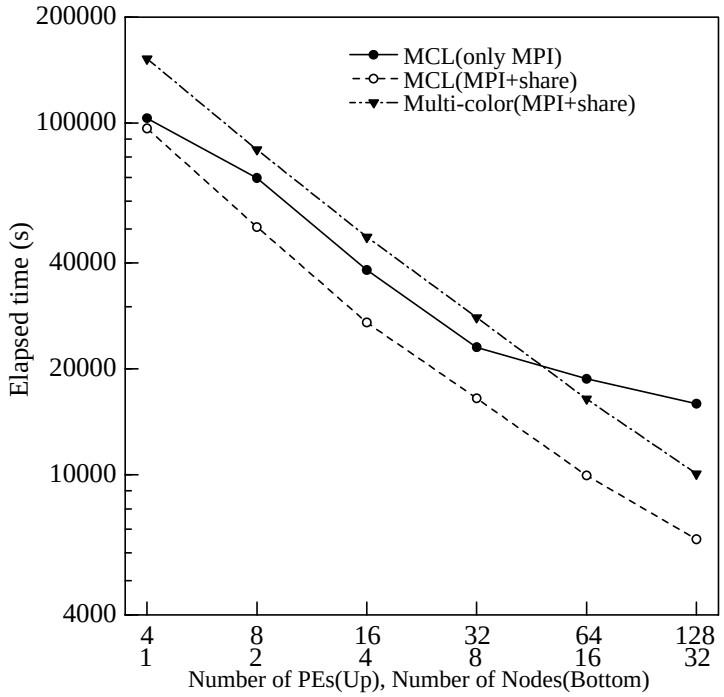

図 6.77: The elapsed time on the SP

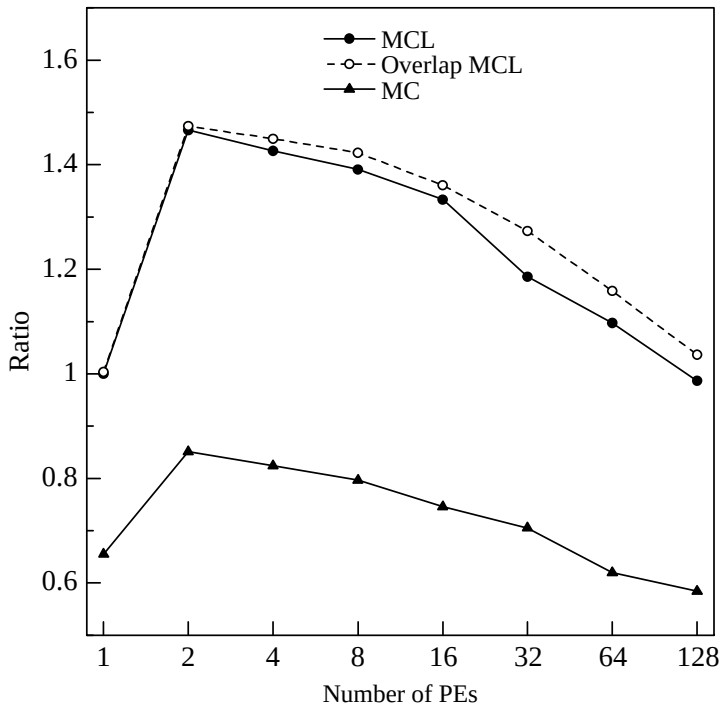

図 6.78: The efficiency on the T3E

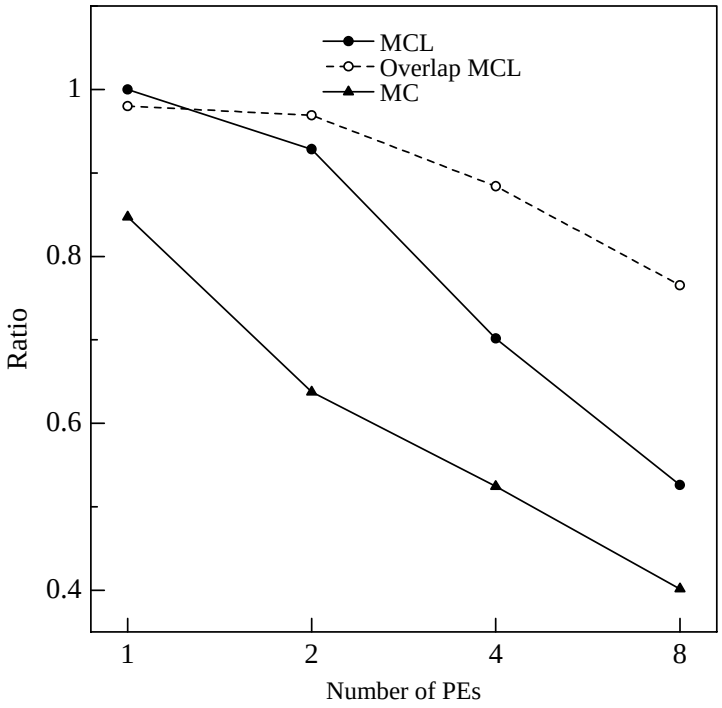

図 6.79: The efficiency on the Cluster2

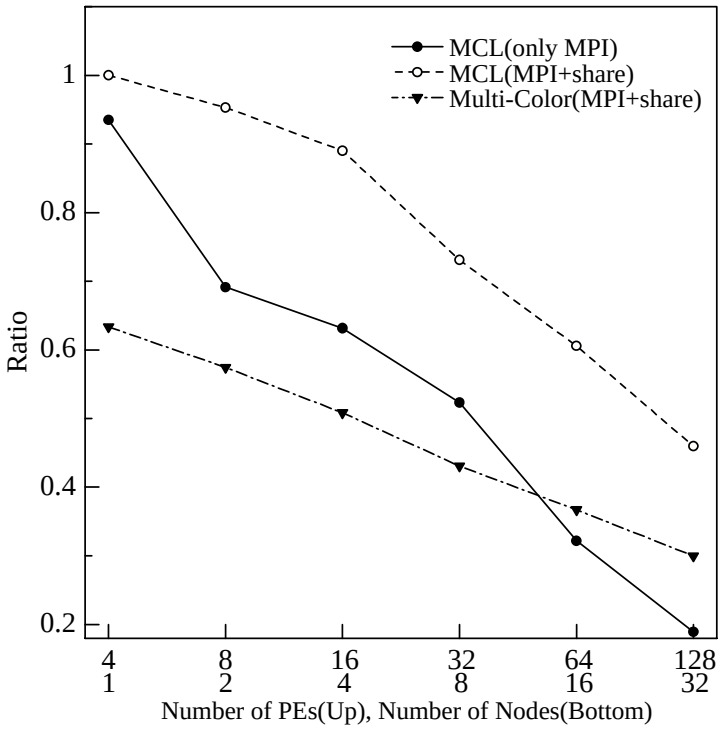

図 6.80: The efficiency on the SP

図 6.75 から T3E における CFD 計算シミュレーションにおいても MCLSOR 法が有効であることが分かった．また，非同期通信-計算重複法を用いた MCLSOR 法は，通信隠蔽を行なわないものよりも高速であった．Poisson 方程式ソルバーとは異なる結果が得られた．全ての PE 数において MCLSOR 法は Multi-color SOR 法に対して 40 % 高速だった．また，PE 数が増加するにつれて非同期通信-計算重複法の効果があらわれ，16 PE 以上の場合 5 % 以上高速に処理できた．また，図 6.76 から Cluster2 における CFD 計算シミュレーションにおいても MCLSOR 法が有効であり，さらに非同期通信-計算重複法の適用は非常に効果が高いことが分かった．MCLSOR 法は Multi-color SOR 法に対して 25 % 前後高速だった．また，非同期通信-計算重複法は，非常に効果が高く，適用していない MCLSOR 法に比べて，8 PE で 30 % 高速に処理できた．図 6.77 から SP での共有/分散プログラム (MPI+share) は分散プログラムのみ (only MPI) に比べて，ベンチマーク問題と同様に効果が非常に高かった．MPI+share の MCLSOR 法は only MPI に対して，1Node 時に 10% 程度，32Node 時に 60% 程度の性能差が見られた．また，MPI+share の Multi-color SOR 法に対して 1Node 時に 30%程度の性能差があったが，32Node 時には逆に 55%程度の差を付けられた形となった．

2 PE での割合を基準にした場合，図 6.78 から T3E において，MCLSOR 法での *Ratio* は，並列化によって 128 PE までに 32.7 % 低下した．また，Overlap MCLSOR 法では，29.7 % 低下した．Overlap MCLSOR 法は，MCLSOR 法よりも 3 % *Ratio* が高かった．Multi-color SOR 法での *Ratio* は，128 PE までに 31.3 % 低下した．計算処理効率が低いため，*Ratio* の低下は MCLSOR 法よりも抑えられた．1 PE での割合を基準にした場合，図 6.79 から Cluster2 において，MCLSOR 法での *Ratio* は 8 PE で 47.4 % 低下した．Overlap MCLSOR 法では，21.9 % 低下した．Overlap MCLSOR 法は，MCLSOR 法よりも 25.5 % *Ratio* が高かった．Multi-color SOR 法での *Ratio* は 8 PE で 52.6 % 低下した．計算処理効率の低さから処理性能の低下は抑えられるものと考えられたが，実際には MCLSOR 法よりも *Ratio* が低下した．図 6.80 から 1Node(SMP の 4CPU) を基準とした場合，MPI+share の MCLSOR 法での *Ratio* は 32 Node で 54 % 低下し，only MPI では 81 %，MPI+share の Multi-color SOR 法は 70 % 低かった．only MPI の性能低下はかなり激しく，MPI+share は並列計算における性能低下を防ぐことが出

来た.

6.4 考　察

6.4.1 ベンチマーク問題

通常の MCLSOR 法は，逐次・並列処理性能においてベンチマーク問題，収束性の検討，実際の CFD 問題シミュレーションにおいて，高い性能が得られ，用いる有用性が高いと考えられる．

通信隠蔽を用いないベンチマーク問題の経過時間において，全般的に 30 % 程度計算処理性能が高かった．これは Multi-color SOR 法が MCLSOR 法に対してスカラー型並列計算機に全く向いていない．スカラー型計算機に多く見られる，メモリアクセスの脆弱さが非常に大きな原因である．そのため，1 つ跳びにデータをアクセスする Multi-color SOR 法では，その不連続なデータアクセスのため，メモリアクセス性能の低さがさらに目立つ状態となったと考えられる．また，データ通信の状態を考えた場合，データ通信量による処理性能の若干の低下が MCLSOR 法で起こっているように見られる．しかし，T3E，SP においては大きく影響が現れていない．即ち，逐次処理時の計算効率の差が大きいため，並列計算時のデータ通信量の増加の影響が大きくあらわれなかったと考えられる．しかし，Cluster2 のような通信性能が低い並列計算機において，データ通信量が多い場合では，通信時間の削減する通信隠蔽は有効であると考えられる．

また，通信隠蔽を用いた場合において，Cluster2 において Multi-color SOR 法，MCLSOR 法ともに非常に有効な結果が得られた．また，Multi-color SOR 法と MCLSOR 法での通信隠蔽の方が高い通信隠蔽効果である．Multi-color SOR 法のデータ通信量が 1/4 で，計算量が 1/8 の場合と MCLSOR 法のデータ通信量が 1/2 で，計算量が 1/4 という状態を考慮すると Multi-color SOR 法では通信を十分に隠蔽するだけの計算時間が得られないと考えられ，Cluster2 において，Multi-color SOR 法の通信隠蔽の方が MCLSOR 法より処理性能の低下が少ない場合がある．データ通信量の量のため通信時間の多くを隠蔽できない場合があるためと考えられる．データ通信量が多く，通信時間を削減しきれない場合は，3 次元分割が必要である可能性が高い．MCLSOR 法は，データの連続性を有するため 3 次元分割に適さない．しかし，解決策として次の方法が考えられる．データ処理の連続方向に領域を分割し，連続方向に各 line をパイプライン的に処理する方法がある．例え

ば，PE 0 と PE 1 において連続方向にデータを分割した場合，PE 0 の 境界 PE 1 方向以外の境界を計算後，その部分を隣接 PE にデータ通信しつつ，内部を計算する．その後 PE 0 の 1 line が終了後，1 line の PE 1 方向の境界部分を PE 1 へデータ通信しつつ，PE 0 は 2 line を計算する．その通信後，PE 1 は 1 line を計算し，PE 0 は 3 line を計算する．この方法を用いた場合，データ通信量の削減が可能である．しかし，処理手順がパイプライン的な処理となるため，並列性が多少低下する．そのため，効果については，詳細な検討が必要である．T3E において Jacobi 法のソルバーの *overlap* 時のような並列化性能の低下が MCLSOR 法では低いレベルに抑えられた．T3E では，*overlap* での計算処理性能の低下が大きいため，Jacobi 法での並列計算性能が落ちたが，MCLSOR 法ではその落ちが低いために，大きな並列化性能の低下に結びつかなかったと考えられる．そして，SP では計算量が少ない場合，PE 数が増加した場合に効果が得られたのは，通信負荷の増加による通信隠蔽が有効に機能した結果であり，効果が得られなかったのは，T3E での Jacobi 法ソルバーの場合と同様に，計算性能の低下による通信隠蔽効果の相殺であると考えられる．

Multi-color SOR 法は，SOR 法をベクトル計算可能とするため，ベクトル型計算機における CFD シミュレーション計算，特に有限差分法系の計算ソルバーを用いているものでは非常によく利用される手法である．しかし，結果の通りスカラー型計算機においては，良い結果が得られなかった．また，後でも述べるがベクトル計算機における Multi-color SOR 法においても，Jacobi 法等の連続アクセス可能なソルバーに対する計算性能の低下が知られており，メモリシステムの弱いスカラー型計算機システムにおいて結果として極端にあらわれたものと考えられる．スカラー型計算機システムでの Multi-color SOR 法の性能低下の原因は，前述の連続なメモリアクセスではないことと後述のキャッシュミスの発生の両方が複合的に合わさったものと考えられる．

共有メモリ型並列計算機において，MCLSOR 法は自動並列化コンパイラを用いることで並列化が可能であり，処理性能も良好な結果が得られた．この結果から MCLSOR 法を用いた CFD 計算シミュレーションプログラムは，SMP システム等の共有メモリ型並列計算機を用いる場合，逐次処理プログラムをプログラムコード変更無しに，並列化できることを意味する．また，現在多く普及している共有/

分散型並列計算機においても，MCLSOR 法を並列化した場合，共有メモリ部分の並列化は自動並列化コンパイラで自動で行なうことも可能となると思われる．しかし，SUN Microsystems の自動並列コンパイラでは，完全な自動化が出来ず，並列化ディレクティブをプログラムコードに追加する必要があったが，数行のディレクティブの追加で並列化は可能だった．MCLSOR 法は，共有メモリのスカラー型並列計算機への親和性が高い計算方法であると考えられる．

6.4.2 共有/分散プログラム

SP のような共有/分散型並列計算機は，現在の並列計算機システムの主流であり，今後もその傾向は続くものと考えられる．そのため，共有/ 分散並列計算機システムを有効に用いることが出来る，共有/分散プログラムによるプログラミングは重要性が高い．本研究でも見られたように，共有/分散並列計算機システムでは，全てを分散メモリプログラムで実行するよりも，共有メモリ部分は共有メモリ型のプログラムモデルを用いるべきである．

共有/分散並列計算機システムにおける，分散メモリプログラムの実行は一般的である．そのため，分散メモリプログラムを実行させた場合でも並列計算性能が維持できるような仕組み (ゼロコピー等) が組み込まれた，メッセージ通信ライブラリが用いられる．IBM が提供している MPI ライブラリについては不明であるが，何かしらかの処理が施されているものと考えられる．しかし，結果は共有/分散プログラムを用いた方が計算処理，通信処理両方において有効だった．

CMLSOR 法の共有/分散プログラミングは容易である．ベンチマーク問題において，分散プログラムは出来ているとすると，共有プログラムのための並列化指示行を 1 行挿入するのみである．そのため，Poisson 方程式や NS 方程式に既存の分散プログラムが存在する場合，非常に容易に共有/分散化が可能である．また，Multi-color SOR 法もおそらく共有/分散化が可能である．

6.4.3 収束性能と通信隠蔽方法

通信隠蔽法を用いたソルバーの並列計算処理性能を収束性能の順に並べ替えると，Multi-color，MCL，Pipeline SSOR，Jacobi という結果になった．特に Pipeline

SSOR 法は，Multi-color SOR 法の 25 % 程度，Jacobi 法の 50 % 増加した．この収束性能は予測された結果である．ただし，MCLSOR 法の収束性能は，Multi-color SOR 法とほとんど変わらない．ベンチマーク問題ではあるが，この結果は他の場合に適用してもほぼ変わらない結果が得られる．

Multi-color SOR 法は，ベクトル型計算機においても Jacobi 法に 対して 20 % 程度の処理性能差しか得られていない．また，ハイパープレーンによる SOR 法の計算処理においてもかなりの性能低下が発生する．この傾向はスカラー型計算機においても同程度からさらに性能差が小さくなる傾向にある．この現象は，メモリアクセスの連続性が非常に重要であることを示している．メモリアクセスの連続性を保ちつつ，並列化による収束性の変化を起こさない方法がより良い方法であり，MCLSOR 法は有効な回答である．

通信隠蔽処理において，Pipeline 法は良く比較対象に用いられる方法である．しかし，本結果では，Pipeline SSOR 法は取り立てて有効であるとは考えられない．通信性能が高い T3E でも PE 数の増加による性能飽和は防げず，SP や Cluster2 においてはさらに性能飽和が PE 数の少ない段階で発生する．NPB LU は，計算に 5 つの物理量を用い，計算量 [23] は，本研究の Poisson 方程式ソルバーに対して約 8 倍多い．計算量が多く，高速な通信機構を持つ並列計算機では，パイプライン処理が有効であることは明らかである．しかし，100 M bps ネットワーク機器の通信性能で計算量が少ない場合，通信負荷の影響から全体性能は低下する．このため，通信性能が低く，計算量が少ない場合に非同期通信-計算重複法による通信隠蔽法は有効であり，特に MCLSOR 法に対して適用した場合に有効であると考えられる．

6.4.4 キャッシュ

Multi-color SOR 法のキャッシュミス回数は MCLSOR 法よりかなり多い．このミス回数はかなり大きな計算性能の差となった．T3E は，2 次キャッシュのミスをかなり抑えることが可能なストリームバッファがあり，非連続メモリアクセスや間接参照メモリアクセスにおいても性能低下が低い計算機である．しかし，他のスカラー型計算機には，そのような機構は持っておらず，非連続メモリアクセスや間接参照のメモリアクセスでの性能低下が大きい．そのため，SP や Cluster2 にお

いてキャッシュミスの回数は T3E よりも明らかに多いと考えられ，大容量キャッシュを搭載したシステムでキャッシュサイズを越える計算を行なう場合には大きな影響を与えることは確実である．

6.4.5 CFD シミュレーション

前述のように MCLSOR 法と Multi-color SOR 法では，ある時間ステップでの圧力の計算する Poisson 方程式の計算においてほぼ同じ収束プロファイルを示した．ほぼ確実に MCLSOR 法は Multi-color SOR 法と同等の収束性を示すことが分かった．

T3E では，NS 方程式解法に用いた 9 Colored Line SOR 法と 27 Color SOR 法は Poisson 方程式解法と同等の性能差がある．T3E では差分精度による線の本数や色数の問題によらず，ほとんどの場合で MCLSOR 法は Multi-Color SOR 法に対して，40% 程度の性能向上が見込まれる．

SP では，MPI+share においての性能差は同等であるが，ベンチマーク問題で見られた通信処理時間の影響が NS 方程式解法においてさらに大きくあらわれたと考えられ，MCLSOR 法の MPI only と MPI+share でさらに大きな差となった．SP 等の SMP クラスタにおいて共有/分散型のプログラムを用いる必要性は非常に高い．MPI+share における MCLSOR 法と Multi-Color SOR 法では，ベンチマーク問題と同等の性能差が見られ，Node 数の増加によってもほぼ同等の性能差が得られると考えられる．ただし，MPI only において見られるように，並列 PE 数と Node 数の増加にともなう通信時間の増加がベンチマーク問題と CFD シミュレーションの両者であらわれているため，さらに検討が必要である．そして，SP の共有/分散プログラムと通信隠蔽処理を用いた場合，T3E においても有効だったため，SP での共有/分散プログラムにおいても効果的であると考えられる．

PC Cluster では，逐次処理において MCLSOR 法と Multi-color SOR 法の性能差は，ベンチマーク問題と同程度であり，NS 方程式計算部分の 9 本線の MCLSOR 法と 27 色の Multi-color SOR 法もベンチマーク問題と同程度の性能差があるものと考えられる．しかし，並列処理においては 20 % 程度性能が縮まった．逐次処理の結果を踏まえた時，NS 方程式解法におけるデータ通信の時間が，かなりの部分を占めてしまう可能性が高く，その影響でベンチマーク問題と比べて性能差が縮

まったものと考えられる．また，通信の影響が大きくなったため，通信隠蔽処理を適用した MCLSOR 法は，ベンチマーク問題よりも CFD シミュレーションの方が性能差が広くなった．

Cluster2 では，ベンチマーク問題と同等かそれ以上の性能が得られた．T3E では，ベンチマーク問題では，通信隠蔽を用いるより用いない方が良い並列性能を得られたが，CFD シミュレーションにおいては，通信隠蔽処理を用いた方が高い処理性能が得られた．これは，O 型格子を用いる円柱周りの流れの並列計算においては，周期境界条件を 2 ヶ所用いる必要がある．MCLSOR 法は，1 ヶ所の周期境界はする必要が無い．もう 1 ヶ所の周期境界条件は，普通境界の条件設定と同時にデータ通信によって条件設定される．通信隠蔽処理においては，この過程が通信隠蔽の内部に取り込まれ，周期境界条件の設定も通信隠蔽される状態となる．そのため，ベンチマーク問題ではあらわれなかった性能が，円柱周りの流れ計算においてあらわれたと考えられる．Cluster2 で得られた通信隠蔽処理の結果がベンチマーク問題の通信隠蔽処理を用いない結果以上に性能向上したのはこのことが原因であると考えられる．

第 7章

結　言

MAC 法を用いた並列 CFD 計算における高性能並列化に関する研究を行なった．高性能な並列計算を行なう 3 つの方法を提案し，そのぞれにおいて性能維持や性能向上が得られることを示した．

最適な領域分割パターンを得る方法において，本研究における予測方法は，実際の並列 CFD 計算の総経過時間との相関係数は 0.9 が得られた．これは十分な性能予測を得られていることを示している．また，この方法の最も重要な目的である最適な分割パターンを得るという意味においては，ほぼ確実に最適な分割パターンが得られた．

通信隠蔽処理法である 1 つ非同期通信-計算重複法を並列 CFD 計算に適用し，その性能向上を示した．通信隠蔽処理を適用しない並列 CFD 計算において IBM RS/6000 SP において最大 14 %，Cluster1 において最大 31 % の性能向上が得られた．中低速の通信機構を持つ並列計算機においては，有効な方法である．高速な通信機構を持つ並列計算機では，通信隠蔽処理に伴う計算処理の低下から通信隠蔽の効果より計算性能の低下が上回り，全体性能として通信隠蔽処理の効果が上がらない場合があった．

以上の方法においては，Poisson 方程式ソルバーに Jacobi 法を用いたが，Jacobi 法では十分な収束性能が得られない場合が多いため，さらに高い収束性能が得られる SOR 法を並列計算に用いる場合について考慮した．通常の SOR 法を並列で用いた場合では，収束性能の低下や収束しない場合があるため，計算性能を維持し並列化が可能である MCLSOR 法を提案した．もう一方の方法である Multi-color

SOR 法は，ベクトル化において必然であるが，スカラー型計算機においては高性能な方法ではない．逐次の CFD 計算において MCLSOR 法は，Multi-color SOR 法に比べ，全てのスカラー型計算機において 40 % 程度高い性能が得られた．また，並列 CFD 計算においても非常に有効であり，MCLSOR 法の欠点であるデータ通信量においても，通信隠蔽処理を行なうことで，低速な通信機構を持った PC Cluster においても有効な方法であることが分かった．近年のトレンドである SMP クラスタを用いた共有分散プログラムを用いた場合にも有効性が高く，SMP クラスタの搭載 CPU 数分のプロセスで分散プログラムを実行するよりも，SMP の Node 内を共有メモリプログラムで，Node 間を分散プログラムで実行した方が高並列時においても，1Node 用いる場合においても有効性が高い．

これらの方法を全て組み合わせれば，効率の高い並列計算性能が得られ，高性能な並列計算が可能となると考えられる．通信隠蔽処理に関しては，通信機構が中低速な並列計算機システムでは，非常に効果が高い．また，これらの方法は全てを用いる必然性は無い．もし，高速な通信機構を持ったベクトル型並列計算機での並列 CFD 計算を行なう場合においては，最適な分割パターンを決定する方法を用いるだけでも非常に有効であると考えられる．また，通信隠蔽処理の考え方は，反復解法のみならず色々な計算手法への適用も可能である．そして，MCLSOR 法は，SOR 法を単純並列や Parallel Block Ordering 等を用いて並列化する場合と比べて，逐次処理のアルゴリズムとの変更点が少なく MPI 等のメッセージ通信ライブラリを用いる場合や自動並列化コンパイラを用いる場合に非常に有効な方法である．

近年において，CFD 分野における並列処理の重要性は非常に増している．今後，並列処理技術の向上とともにその重要性はさらに増すと考えられる．様々な超並列計算機を用いるシミュレーションが一般化した場合に非常に有効な方法であり，計算工学分野において重要な意味を持つことが期待できる．

謝　辞

本研究を行なうにあたり御指導賜りました 松澤 照男 教授に深く感謝致します．

また，本論文をまとめるにあたり，堀口 進 教授，阿部 亨 助教授，敷田 幹文 助教授には有益な御助言，御審査頂き，深く感謝致します．

理化学研究所情報基盤研究部情報環境室室長 姫野 龍太郎 博士には，研修生として理化学研究所に受け入れて頂きましたこと，その後も様々な御配慮，御助言等頂きましたことを深く感謝致します．また，情報環境室技師 重谷 隆之 博士には，研究に関して様々な便宜を図って頂きました．深く感謝致します．

松澤研究室の諸兄には，議論の場を設けて頂き，厚くお礼申し上げます．また，理化学研究所情報基盤研究部情報環境室の皆様には，理化学研究所滞在中，大変御世話になりました，厚くお礼申し上げます．

参考文献

[1] Crandall, P. and Quinn, M.: Three-Dimensional Grid Partitioning for Network Parallel Processing, *In Proceedings of the ACM 1994 Computer Science Conference* (1994).

[2] van der Wijngaart, R.: Efficacy of source code optimizations on cache-based processors, NAS Technical Report NAS-00-014, NASA Ames Research Center, Moffett Field, CA 94035-1000 (2000).

[3] Quinn, M. and Hatcher, P.: On the Utility of Communication-Computation Overlap in Data-Parallel Programs, *Journal of Parallel and Distributed computing*, Vol. 33, No. 2, pp. 197–204 (1996).

[4] Fink, S. and Baden, S.: Non-uniform Partitioning of Finite Difference Methods Running on SMP Clusters. http://now.cs.berkeley.edu/clumps/.

[5] 田中良夫, 松田元彦, 久保田和人, 佐藤三久: SMP クラスタ上でのリモートメモリ転送を用いた通信と計算のオーバーラップによる性能改善, 情報処理学会研究報告，98-HPC-72 (1998).

[6] Xie, D. and Adams, L.: New parallel SOR method by domain decomposition (1997). Dexuan Xie and Loyce Adams, New parallel SOR method by domain decomposition, SIAM J. Sci. Stat. Comput., submitted, 1997.

[7] Bailey, D., Harris, T., Saphir, W., van der Wijngaart, R., Woo, A. and Yarrow, M.: The NAS Parallel Benchmarks 2.0, NAS Technical Report NAS-95-020, NASA Ames Research Center, Moffett Field, CA 94035-1000 (1995).

[8] Kawamura, T. and Kuwahara, K.: Computation of high Reynolds number flow around a circular cylinder with surface roughness, *AIAA Journal*, 84-0340 (1984).

[9] Harlow, F. H. and Welch, J. E.: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, *Physics of Fluids*, Vol. 8, pp. 2182–2189 (1965).

[10] 荒川忠一: 数値流体工学, 東京大学出版会 (1994).

[11] 川村哲也: 流体解析 I, 朝倉書店 (1996).

[12] 数値流体力学編集委員会 (編): 数値流体力学シリーズ 1，非圧縮性流体解析, 東京大学出版会 (1995).

[13] Thompson, J., Wasri, A. and Mastin, W.: *Numerical Grid Generation: Foundations and Applications*, North Holland (1985).

[14] Schwarz, H. A.: *Gesammelte Mathematische Abhandlungen*, Vol. 2, Springer, Berlin (1890).

[15] Keys, D. E., Saad, Y. and Truhlar, D. G.: Domain-Based Parallelism and Problem Decomposition Methods in Computational Science and Engineering, *SIAM, Philadelphia, PA* (1995).

[16] IBM Corporation: *XL Fortran for AIX Language Reference Version5.1*. (1999).

[17] Kurokawa, M., Himeno, R., Shigetani, S. and Matsuzawa, T.: A case study of the partitioning patterns for domain decomposition method on VPP700E, RIKEN HPC Review (2000).

[18] 黒川原佳, 姫野龍太郎, 重谷隆之, 松澤照男: 三次元ポアソン方程式に対する領域分割法の 分割方法による性能への影響, 情報処理学会研究報告，2000-HPC-80 (2000).

[19] 市川周一, 川合 隆光, 島田 俊夫: 組合せ最適化による並列数値シミュレーションの静的負荷分散, 情報処理学会論文誌, Vol. 39, No. 6, pp. 1746–1756 (1998).

[20] Himeno, R.: *Implementation of an incompressible Navier-Stokes Solver on Vector/Parallel Computers and Application to the Aerodynamics Analysis of Automobilies* (Dr. Ginsberg, M.(ed.)), Society of Automotive Engineers (1992).

[21] IBM Corporation: *The RS/6000 SP High-Performance Communication Network*. http://www.rs6000.ibm.com/resource/technology/sp_sw1/spswp1.book_1.html.

[22] Pedretti, K. and Fineberg, S.: Analysis of 2D Torus and Hub Topologies of 100Mb/s Ethernet for the Whitney Commondity Computing Testbed, NAS Technical Report NAS-97-017, NASA Ames Research Center, Moffett Field, CA 94035-1000 (1997).

[23] Yarrow, M. and van der Wijngaart, R.: Communication improvement for the LU NAS Parallel Benchmark: A Model for Efficient Parallel Relaxation Schemes, NAS Technical Report NAS-97-032, NASA Ames Research Center, Moffett Field, CA 94035-1000 (1997).

[24] van der Wijngaart, R., Sarukkai, S. and Mehra, P.: Analysis and Optimization of Software Pipeline Performance on MIMD Paralle Computers, NAS Technical Report NAS-97-003, NASA Ames Research Center, Moffett Field, CA 94035-1000 (1997).

[25] Kurokawa, M., Matsuzawa, T., Himeno, R. and Shigetani, S.: Parallel CFD simulation using systolic communication-computation overlap, *5th International Conference and Exhibition on High-Performance Computing in the Asia-Pacific Region(HPC Asia 2001)* (2001).

[26] Evans, E., Johnson, S., Legget, P. and Cross, M.: Automatic code generation ov overlapped communications in a parallelisation tool, *Parallel Computing*, Vol. 23, No. 10, pp. 1493–1523 (1997).

[27] Ku, H., Hirsh, R. and Taylor, T.: A Pseudospectral Method for Solution of the Three-Dimensional Incompressible Navier-Stokes Equations, *Journal Computational Physics*, Vol. 70, pp. 739–462 (1987).

[28] 黒川原佳, 松澤照男, 姫野龍太郎, 重谷隆之: 境界領域先行計算による通信処理隠蔽を行なう並列計算アルゴリズム, 第13回計算力学講演会講演論文集 (2000).

[29] 杉原正顕, 小柳義夫, 森正武, 藤野清次: ベクトル計算機における SOR 的方法の効率について, 情報処理学会論文誌, Vol. 31, No. 6, pp. 930–938 (1990).

[30] 藤野清次, 杉原正顕, 小柳義夫, 森正武: SOR 法のベクトル計算機向き書換えによる効率の低下, 情報処理学会論文誌, Vol. 32, No. 3, pp. 373–382 (1991).

[31] Bailey, D. H.: RISC Microprocessors and Scientific Computing, RNR Technical Report RNR-93-004, NASA Ames Research Center, Moffett Field, CA 94035-1000 (1993).

[32] Fujino, S., Himeno, R., Kojima, A. and Terada, K.: Implementation of the Multicolored SOR method on a Vector Supercomputer, *IEICE TRANSACTIONS INFOMATION & SYSTEM*, Vol. E80-D, No. 4, pp. 518–522 (1997).

[33] Doi, S. and Washio, T.: Ordering strategies and related techniques to overcome the trade-off between parallelism and convergence in incomplete factorizations, *Parallel Computing*, Vol. 25, pp. 1995–2014 (1999).

[34] Frumkin, M. and van der Wijngaart, R.: Efficient cache use for stencil operations on structured discretization grids, NAS Technical Report NAS-00-015, NASA Ames Research Center, Moffett Field, CA 94035-1000 (2000).

[35] Block, U., Frommer, A. and Mayer, G.: Block colouring schemes for the SOR method on local memory parallel computers, *Parallel Computing*, Vol. 14, pp. 61–75 (1990).

[36] Kuck Association Inc.: KAP, http://www.kai.com/.

[37] Parallel Software Products Inc.: CAP Tools, http://captools.gre.ac.uk/.

[38] 日本数値流体力学会 (編): 第 2 版 流体力学ハンドブック, 丸善株式会社 (1998).

[39] Tamura, T., Ohta, I. and Kuwahara, K.: ON THE RELIABILITY OF TWO-DIMENSIONAL SIMULATION FOR UNSTEADY FLOWS AROUND A CYLINDER-TYPE STRUCTURE, *Journal of Wind Engineering and Industrial Aerodynamics*, Vol. 35, pp. 275–298 (1990).

本研究に関する発表論文

査読付き論文

1. 黒川原佳，松澤照男，姫野龍太郎，重谷隆之，“並列 CFD 計算における非同期通信-計算重複法”，情報処理学会論文誌，ハイパフォーマンスコンピューティングシステム Vol.42，No.SIG9(HPS3)，pp.54.63，2001

2. 工藤 奨，佐藤正志，町田和敏，山口隆平，黒川原佳，松澤照男，池田満里子，岡浩太郎，谷下一夫，“剥離流れ場における培養内皮細胞の高分子物質取り込みと形態変化”，日本機械学会論文集 (B 編)，65B(639)，pp.3705-3712，1999

3. Motoyoshi Kurokawa，Ryutaro Himeno，Takayuki Shigetani，Teruo Matsuzawa，“A case study of the partitioning patterns for domain decomposition method on VPP700E”，RIKEN Review，No.30，pp.30-34，2000.9.

査読付き国際会議

1. Motoyoshi Kurokawa，Teruo Matsuzawa，Ryutaro Himeno，Takayuki Shigetani，“Parallel CFD simulation using systolic communication-computation overlap”,5th International Conference and Exhibition on High-Performance Computing in the Asia-Pacific Region(HPC Asia 2001), 2001.8

査読付き国内会議

1. 黒川原佳，松澤照男，姫野龍太郎，重谷隆之，“スカラー並列計算機における Multi Colored Line SOR 法”，2002 年ハイパフォーマンスコンピューティングと計算科学シンポジウム (HPCS2002)，2002.1. (受理)

口頭発表

1. 黒川原佳，松澤照男，“自動並列コンパイラを用いた SMP 型計算機の評価”，第 9 回計算流体シンポジウム講演論文集，pp.415-416，1998.7.
2. 黒川原佳，松澤照男，“大規模 SMP システムの性能評価と効率的な並列手法の実装と実際”，第 14 回超並列計算研究会，1998.11.
3. 黒川原佳，松澤照男，“SMP 型計算機による非圧縮性粘性流れの並列計算”，第 12 回数値流体力学シンポジウム論文集，pp.533-534，1998.12.
4. 黒川原佳，松澤照男，“局所的な同期機構を用いた非圧縮粘性流体の並列計算”，第 4 回計算工学講演会論文集，Vol.1，pp.393-396，1999.5.
5. 黒川原佳，重谷隆之，“今すぐ始める 100 万円以下の PC クラスタとその性能”，第 13 回数値流体シンポジウム講演要旨集，pp.7-10，1999.12.
6. 黒川原佳，姫野龍太郎，重谷隆之，松澤照男，“三次元ポアソン方程式に対する領域分割法の分割方法による性能への影響”，2000-HPC-80，pp.137-412，2000.3.
7. 黒川原佳，姫野龍太郎，重谷隆之，松澤照男，“三次元ポアソン方程式に対する領域分割法の分割方法による性能への影響”，理研 HPC シンポジウム，2000.3.
8. 黒川原佳，重谷隆之，“PC クラスタによるハイエンドベクトル計算機の置換え可能性”，日本機械学会 2000 年年次大会，計算力学部門ワークショップ，日本機械学会 2000 年度年次大会資料集 (V)，pp.457-458，2000.8.

9. 黒川原佳，松澤照男，姫野龍太郎，重谷隆之，“境界領域先行計算による通信処理隠蔽を行う並列計算アルゴリズム”，第 13 回計算力学講演会講演論文集，pp.499-500，2000.11.

10. 黒川原佳，松澤照男，姫野龍太郎，重谷隆之，“領域分割法による MAC 法の効率的な並列計算アルゴリズム”，第 14 回数理流体力学シンポジウム講演要旨集，pp.196，2000.12. 論文掲載 (http://www.nacsis.ac.jp/jscfd/cfds14/pdf/e08-2.pdf)

11. 黒川原佳，井口寧，松澤照男，“CRAY-T3E における NAS Parallel Benchmark”，2001-HPC-142，pp.139-144，2001.3.

12. 高橋達矢，河村洋，松澤照男，黒川原佳，荒川忠一，功刀資彰，木村俊哉，“熱流体解析における並列計算のベンチマークテスト”，第 11 回計算力学講演会論文集，pp.583-584，1998.11.

その他

1. 松澤照男，黒川原佳，“並列計算機による NPB ベンチマークテスト及び並列化について”，DECUS High Performance Technical Computing 分科会，1999.3.

2. 姫野龍太郎，重谷隆之，黒川原佳，“PC クラスタは Poor Man’s Supercomputer”，サイエンティフィック・システム研究会，2000.9.

3. 井口 寧，黒川原佳，松澤照男，“Starfire の NASA Benchmark をベースにした性能評価”，サイエンティフィック・システム研究会，2000.11.

4. 井口 寧，黒川原佳，松澤照男，“超並列計算機におけるハードウェア高速化によるシステム性能の評価”，Research Report IS-RR-2001-027，2001.12.