東海大学大学院平成 27 年度博士論文

並列処理時間のモデル化による性能評価方法に関する研究

折居　茂夫

目次

1章　序論

1.1　高速計算技術の進展

電子計算機による高速計算の歴史は, 真空管を使用して 1942 年に開発された ABC と 1946 年に開発された ENIAC から始まった．真空管がトランジスタに換わり始めたのは 1955 年に Manchester Mark I をトランジスタ化した改造版が作られてからで，世界初のプログラム内蔵式の汎用トランジスタ･コンピュータ ETL Mark III は 1956 年に日本で開発された．日本ではトランジスタの電子計算機の出現とほぼ並行した 1957 年にパラメトロンコンピュータ MUSASINO-1 が開発された．トランジスタとパラメトロンはしばらく競合状態にあったが, 使用電力と発熱に起因する速度向上の差でトランジスタが生き残った．1964 年シリコントランジスタを使用したスーパーコンピュータ CDC6600（クロック周波数 10MHz）がリリースされた．高速化技術として命令パイプラインを採用したその後継機の CDC7600（クロック周波数 36.4MHz）が 1969 年にリリースされた． 1969 年にリリースされた System/360 モデル 85 ではキャッシュメモリが導入された．これら 2 つの計算機は同じ時代に存在し性能を競い合う関係にあった．1974 年世界で初の演算パイプラインのベクトル計算機 STAR-100 が開発されたが性能が出なかった. そのボトルネックの一つデータアクセス時間をベクトルレジスタを搭載してクリアし，IC を用いたピーク性能 160Mflops のパイプラインベクトル計算機 Cray-1（クロック 80MHz）が 1975 年にリリースされた．ベクトルレジスタを搭載したパイプラインベクトル計算機は，1980 年代のスーパーコンピュータの主流となった.

高速計算に並列処理を使う並列計算機は，1970 年代に 64 台の PE(Processing Element) を用い Megaflops の性能を達成した ILLIAC IV から始まるとするのが一般的である．一方並列処理が現在のように普及するにはしばらく時間を必要とした. 当時は半導体技術が急速に進歩し, 並列処理で達成された性能が翌年 1 台の計算機でクリアできたことが大きく影響した.

並列処理を採用したスーパーコンピュータは 1984 年，4 つのパイプラインベクトルプロセッサの結合により最大 800Mflops の性能が出た Cray X-MP/48 が始まりである．1990 年代になるとスーパーコンピュータに限らず科学技術計算を並列処理で行うことが普及した．並列処理で使用されるプロセッサ数が増えるに伴い，2000 年台にはパイプラインベクトル方式ではなく高速なスカラ方式のプロセッサが使われるようになって今日に至る．

このような歴史において計算機が半導体を採用して以降，1965 年に Moore が示した経験則「半導体の集積密度は 18～24 ヶ月で倍増する」[1-1)]が最近まで成り立ち，半導体テクノロジの進歩により計算機の性能が向上する状態が続いていた．昨今はその限界が真剣に議論され，2020 年以降にリリースが想定されているエクサスケールコンピュータまでは現在のテクノロジの延長で開発できるだろうという見解[1-2)]もある．

図 1-1 は計算機の性能を決定する 4 つの技術の年度毎の年間成長率(CAGR)を 1996 年から 2012 年まで示したもので，Linpack ベンチマークテストを用いて最高性能を競う TOP500[1-3)]の 10 位までのデータから作成したものである[1-4)]．図中，青で示された R_{max} は flops で現わした Linpack ベンチマークテストの最高性能，緑はコア当たりの平均クロック周波数，紫はコア当たりのメモリ量，赤は全コア数である．

この図は上位 10 スーパーコンピュータでは 2004 年以降 Moore の経験則が適用できなくなり，コア数を増やすことにより性能向上が図られていることを示す．即ち，2004 年から緑と紫の年間成長率がほぼ 1 になって現在まで継続していることである．一方全コア数を示す赤は急激に増加し近年また増加傾向にある．これは 2004 年以前の計算機の性能 R_{max} の成長がクロック周波数の増加と集積度の増加で支えられていたのに対し，2004 年以降は R_{max} の成長を保つためにコア数増加のみに頼るようになったことを意味する．故に今後の高速化技術はより高並列に向かっていくことが予想される．

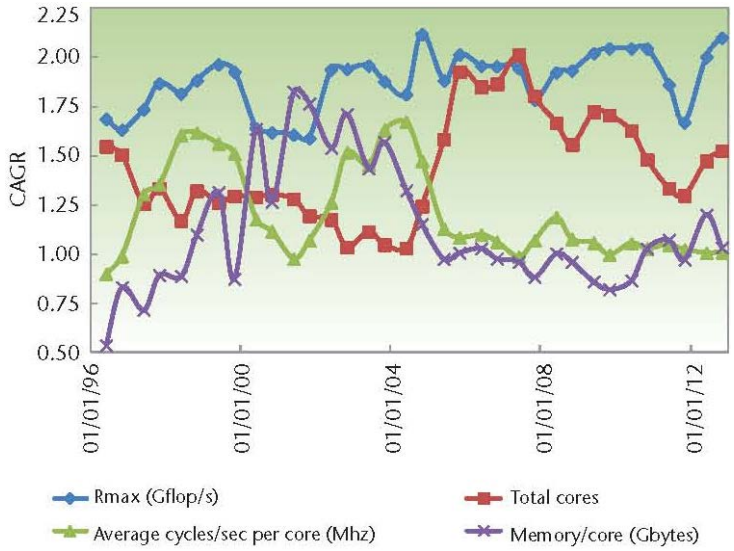


図 1-1 Summary of technology trends. The trends show the rate of improvement[1-4] (compounded annual growth rate, or CAGR) for R_{max} (peak floating-point operation rate) core count, clock frequency, and memory per core. After 2004, we can see a clear separation of trends where core counts and Rmax rates increase at 1.5× to 2× per year, but clock frequencies and memory growth remain essentially flat to negative.

1.2 並列処理の性能評価方法の進展と課題

並列処理の性能に関する重要な指摘が行われたのは 1967 年で，1.1 節で紹介した System/360 の開発者 Amdahl によるものである．彼の指摘は，並列処理の性能は処理全体の時間に占める並列処理できない部分（逐次処理部）の割合で決まる [1-5]というもので，例えば逐次処理部が 10%あるとプロセッサ数を無限に増やしても高々10 倍の性能しか得られないというものである．

この問題がプロセッサ数の増加と共に問題の大きさを大きくすればある意味解決できることを Gustafson が示したのは 1988 年であった [1-6]. 図 1-2(a)は Amdahl の法則で s：逐次処理時間，p：並列処理時間，N：プロセッサ数，$s+p=1$ である．問題の大きさが 1 に固定されているため，$N\to\infty$でもスピードアップは $1/s$ が並列処理の限界となる．

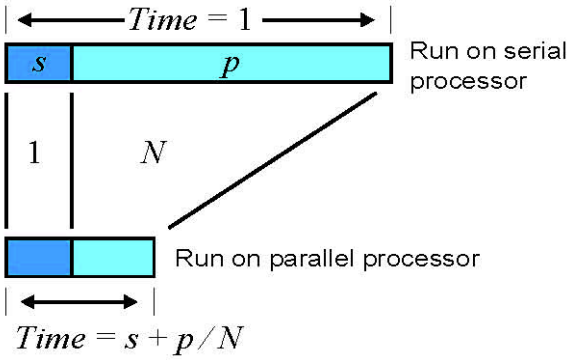


図 1-2(a) Fixed size model: Speedup=1/(*s*+*p*/*N*) [1-5)]

これに対する Gustafson の Scaled-size モデルを図 1-2(b)に示す．このモデルでは逐次処理部の時間はそのままで，並列処理部を $N \cdot p'$倍した問題を用意し，並列処理の時間が $s'+p'=1$ になるようにすれば Speedup に $s'+ N \cdot p'$を得る．従って問題の大きさを大きくすることにより s が変わらないで p が大きくなれる場合は，並列処理により大きな性能向上を期待できる．

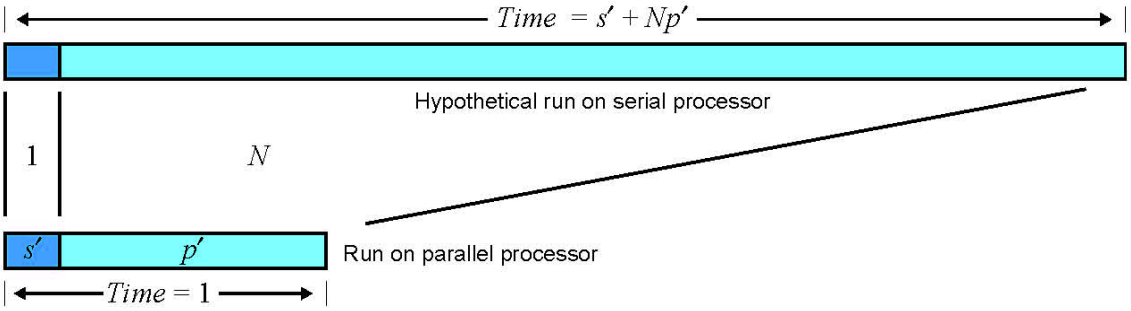


図 1-2(b) Scaled-size model: Speedup=$s'+N \cdot p'$ [1-6)]

Gustafson の指摘以前まではプロセッサ数の増加に対する処理時間，スピードアップによる性能評価が行われてきたが，彼の指摘の後はプロセッサ数と問題の大きさを変数とした性能評価が行われるようになった．

またこれら 2 つの変数を性能評価に用いることにより, プロセッサ数に比例して性能向上するかどうかという, スケーラビリティの評価を定量的に議論できるようになった. プロセッサ数のみでスケーラビリティを議論しようとすると，図 1-2(a)の逐次処理 *s* を含む並列オーバーヘッドのため, 実測においてプロセッサ数に比例して性能向上するように見える場合でも，プロセッサ数を増やしていくといずれ比例しなくなる．従って並列処理でスケーラビリティを達成することはできないと帰結される不具合がある. これに対し問題の大きさを可変にするとスケーラビリティの存在が定量的に定義できるようになる．並列効率一定(Isoefficiency)でスケーラビリティを評価する性能評価方法が 1993 年に提案された [1-7]. 速度一定(IsoSpeed)でスケーラビリティを評価する性能評価方法が 1994 年に提案された [1-8].

上記のように幾つかのスケーリング方法が考案されたが, 現在は問題の大きさが増すとき 1 プロセッサの計算量が一定になるようにプロセッサ数を増す weak scaling による評価方法がよく用いられる. また問題の大きさを固定しプロセッサ数のみで評価を行う旧来の Amdahl の方法を strong scaling と呼んで，これも合わせて用いられる.

これらのスケーリングは実測から行うこともできるが, 計算機資源をセーブするためには予測することが重要である. 即ち小規模な問題と少ないプロセッサ数で測定した処理時間からプロセッサ数を増やした大規模な問題の処理時間を予測することが, 並列性能評価として重要となる. 主な予測用法の一つとして処理時間のモデルを作って外挿する方法があるが, 実際に並列処理されるプログラムは大規模かつ複雑でモデルパラメータが増えることが課題となる．またモデルパラメータ値の推定には時間測定が必要で, モデルパラメータが多いほど測定オーバーヘッドが大きくなる課題がある. さらに複雑なモデルでは，モデル式の各項が観測データのノイズをモデル化する「過剰適合」が生じて予測精度が著しく落ちるという課題がある. 処理時間をモデル化して性能予測を行う際に生じるこれらの課題は，まだ研究段階にあると考えられる.

1.3　近年の並列性能評価の課題

並列処理の性能評価を行うときは，前節で述べた性能予測の課題に加え，評価の基準となる量を何にするかが課題となる．並列処理では，その基準として 1 プロセッサの処理時間が用いられてきた. そしてこの時間と並列処理の時間を比較したスピードアップが性能評価指標として使われてきた. よく使われてきたこの指標には幾つかの問題がある. 第 1 に 1 プロセッサの処理時間に何を選ぶかという問題がある. 1.1 節で述べた「並列処理で達成された性能が翌年 1 台の計算機でクリアできた時代」には，基準値としてその時の最速の計算機の処理時間を用いる，並列化アルゴリズムを使って作られたプログラムを 1 プロセッサで実行するのではなく逐次処理向けの最適アルゴリズムを用いたプログラムを実行してそれを基準にする場合もあった．理由は，この 1 プロセッサの処理時間に遅いものを選ぶと並列処理の性能が良く見えることにある. スピードアップを性能評価指標として不具合が生じた例に，原子力コード KENO Ⅳをベクトル計算機で高速化した例がある. ベクトル計算機ではスカラ処理時間を基準にしベクトル処理時間と比較してスピードアップを計算するが，CYBER205 では 23 倍，VP-100 では 3.8 倍という結果がでた．数年にわたりこの差の議論が行われたが，この差は CYBER205 のベクトル処理とスカラ処理の論理性能差が 10 倍以上だったことに対し，VP-100 の評価基準にスカラ計算機に最適化された KENO Ⅳの処理時間を使ったため，ベクトル処理とスカラ処理の性能差が最大 4 倍になったためであった [1-9]．このように基準が違う倍率を比較して性能評価することは大変危険である.

第 2 の問題は高並列処理の処理を 1 プロセッサで実行することができないことである. 即ち今まで使ってきた 1 プロセッサの処理時間という基準が高並列処理では使えない．1.1 節で述べた Linpack ベンチマークテストでは，浮動小数点の実行回数を式化して flops 値を算出し，基準に計算機の論理 flops 値を用いて「効率」を評価している．Linpack では計算機間の性能比較の公平を期すため四則演算の総数を式で与えているが，

計算機が四則演算数を動的に計測する機能があれば, 大規模なプログラムに対してもこの「効率」を並列処理の性能評価指標として用いることができる．一方この「効率」は浮動小数点演算が少ない処理の並列性能の評価には向かない.

第 3 の問題は基準値の測定が必要なことである. たとえ 1 プロセッサで処理時間が測定できたとしても，通常の計算機利用において性能評価のために同一入力値の処理を 2 度行うことはない. つまりスピードアップと並列効率というこれまで用いられてきた性能評価指標は，通常の計算機利用では使えない性能評価指標なのである.

現在, PC において数コアで行われる並列処理から, 数万プロセッサを用いて Petaflops の性能を達成したスーパーコンピュータで並列処理が使われる時代である． 1.1 節で述べたように Moore の法則の限界が見えた現在，使用されるプロセッサ数は今後も増加することが予想される. また並列処理の性能はプログラミングや入力値に対して敏感であることを考えると, 通常の計算機利用における全ての並列処理の性能評価を可能にする性能評価指標が必要と考えられる.

1.4 本研究の目的

本論文では第 1 に高並列処理を含む全ての並列処理の並列効率の評価が可能な, 新しい性能評価指標を提案したことについて述べる．1.3 節で述べたように従来の性能評価指標「並列効率」は，1 プロセッサで実行した処理時間と並列処理時間を比較して定義するが, この 1 プロセッサで実行した処理時間は通常の計算機利用では得られない値である．更に数万台規模のプロセッサが用いられる現在の高並列処理では，同じ処理を 1 プロセッサで実行すること自体不可能である．一方効率の評価は，数万台の高並列処理を行う昨今，使用するプロセッサ数，電力をセーブするための重要である．そこで今まで使われてきた性能評価指標の処理時間モデルを再考し, 実務の並列処理で得られる測定値のみで決定できる新しい効率評価指標とその補助評価指標を提案した [1-10].

第 2 に 2 つの性能予測方法を提案したことについて述べる. 性能予測のためにはアプリケーションプログラムと並列計算機の構造を考慮して処理時間をモデル化するホワイトボックスアプローチと，構造を考慮しないブラックボックスアプローチがある．そこでホワイトボックスアプローチにより数値シミュレーションプログラムで時間を費やしている箇所（繰り返し処理）をモデル化して性能予測をする方法を研究した．提案したモデル化方法により, プログラム中のループの四則演算と通信の性能を評価することが可能になった. そこでそれらを計算機の理論値と比較した効率で評価する方法を提案し [1-11)]，プラズマ粒子コードの並列処理の評価に応用した [1-12)].

ホワイトボックスアプローチではプログラムや計算機の構造に基づいた予測ができるが, プログラムを解析してモデルを構築するのにけっこうな時間を必要とする場合や, モデルパラメータを決定のための時間測定により処理時間が増加するという問題が生じる場合がある. そこで処理全体の時間を多項式でモデル化したブラックボックスアプローチによるモデル化方法を提案した [1-13), 1-14)].

これら２つのアプローチでは, モデルパラメータを推定する際に入力データの揺らぎ等がモデル化される過剰適合が生じ, 入力データの内挿はできるが外挿即ち予測ができないモデルとなる場合がある．そこで第 3 に，モデルパラメータに非負条件を課した連立残差不等式を有理数計算のシンプレックス法を用いて解く, 過剰適合を抑制して予測精度を向上するモデルパラメータ推定方法を提案し, ホワイトボックスアプローチで作成したモデルを基にして作成した基底関数モデルを用いて, 予測力が向上することを確認した [1-15), 1-16)].

これら並列性能評価指標の提案，モデル化方法の提案，モデルパラメータ推定方法の提案を通常の計算機利用に適用することにより, 多くの並列計算機のプロセッサ数や電力資源のセーブに貢献すると考え，ここにその内容を公表させていただくことにした.

参考文献

1-1) G. E. Moore, Cramming more components onto integrated circuits, Electronics, 38 (8) (April 19, 1965).

1-2) 追永勇次，私信，富士通 SS 研 HPC フォーラム「エクサスケール時代のコンピュータ技術」(2013).

1-3) http://www.top500.org/

1-4) P. Kogge, J. Shalf, Exascale computing trends, Computing in Science & Engineering Vol. 15 (6) (2013) 16–26.

1-5) G. Amdahl, Validity of the single-processor approach to achieving large scale computing capabilities, in: AFIPS Conference Proceedings, vol. 30 (1967) 483–485.

1-6) J.L. Gustafson, Re-evaluating Amdahl's Law, CACM 31 (5) (1988) 532–533.

1-7) V. Kumar, V.N. Rao, Parallel depth first search, Part II: Analysis, J. Parallel Program. 16 (6) (1987) 501–519.

1-8) X. H. Sun, D. T. Rover, Scalability of parallel algorithm-machine combinations, IEEE Trans. Parallel Distrib. Systems, 5 (6) (1994) 599–613.

1-9) 折居茂夫，樋口健二，浅井清，ベクトル化 KENO Ⅳコードの性能評価，JAERI-M 90-135 (1990).

1-10) S. Orii, Metrics for evaluation of parallel efficiency toward highly parallel processing, Parallel Comput. 36 (2010) 16–25.

1-11) 折居茂夫, 数値計算のための並列計算機性能評価方法, 情報処理学会論文誌, 39, (3) (1998) 529–541.

1-12) 折居茂夫，プラズマ粒子コードのためのベクトル並列計算法，プラズマ・核融合学会誌，75 (6) (1999) 704–716.

1-13) 折居茂夫，高並列処理におけるスケーラビリティ評価方法（Ⅱ），情報処理学会研究報告, HPC-126 (48) (2010).

1-14) 折居茂夫，時間モデルを用いた並列処理の性能評価 － 並列化部に隠れた並列オーバーヘッド －, HPC-130 (1) (2011).

1-15) 折居茂夫，山本義郎，限量記号消去法を用いた回帰モデルの予測力向上，情報処理学会研究報告, HPC-139 (1) (2013).

1-16) 折居茂夫，山本義郎，シンプレックス法を用いた非負のモデルパラメータを持つ並列処理時間モデルの予測力向上，情報処理学会論文誌 56, (6) (2015) 1481–1495.

2 章　並列処理時間のモデル化による性能評価（理論）

並列処理の処理時間は，プロセッサへの負荷分散のバランス，並列処理できない処理時間，プロセッサ間のデータアクセス等の並列処理により余分に発生する処理時間，という３種類の並列オーバーヘッドで決まる．従って理想の並列処理は，処理が各プロセッサに均等に分割され，並列処理できない（逐次）処理時間が零，並列処理により余分に発生する処理時間が零のときである．実際に並列オーバーヘッドを最小にするためには，ハードウェアの特性，そのハードを制御するソフトウェアの特性及び，プログラムの特性がマッチすることが必要となる．並列処理の性能評価とはこのようなマッチングを定量的に把握し，計算機のスペックに見合った性能が出ているか，出ていないとすれば何が原因かを分析することと，定量的に把握した情報から性能を予測することである．

図 2-1 は実際の並列処理のイベントトレースである．図はプロセッサ毎のイベントを横軸の時系列に沿って表示したものである．茶，青，赤はイベントを種類別に色分けしたもので，茶と青は並列処理，赤は MPI(Message Passing Library)通信に起因するイベントである．黒の直線はプロセッサ間通信が行われたことを示す．これに続く赤で表示された MPI_Wait はこの通信によって待ちが生じたことを示す．

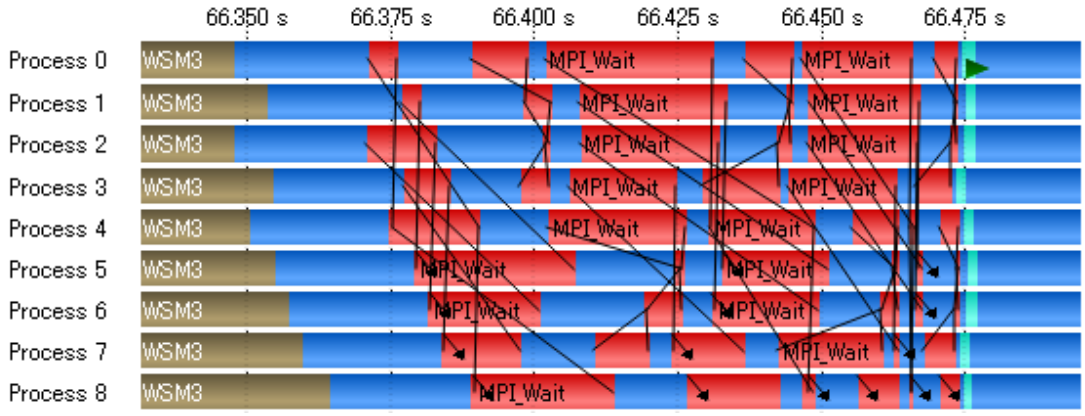


図 2-1　並列処理のイベントのトレース [2-1)]

図は茶や青で表示される並列処理の時間が各プロセッサで異なることを示す．プロセッサ間の通信は同時刻に始まらず，赤で示される待ち時間もプロセッサにより異なる．

このように図は，この並列処理の負荷分散が均等でない現象（ロードインバランス）が存在することを示す．このような並列処理を用いて，ジョブを投入してから終了するまでの時間即ち「ウォールクロックタイム」の短縮が図られる．多くの場合，最も長くジョブに関わったプロセッサの経過時間がこれに当たる．ロードインバランスが生じると，特定のプロセッサが稼働し他のプロセッサが遊んでいる状態が生じる．このような複雑な並列処理の性能評価を定量的に行う主な方法の一つに，処理時間のモデルを構築してそれを評価する方法がある．

本論文では処理時間のモデルを構築して性能評価指標と性能予測する方法の研究を報告するが，それはつまり図 2-1 のような複雑な並列処理の振る舞いをどのようにモデル化するかということである．そこで本論文の「性能評価指標の提案」では，従来の性能評価指標が 4 つの時間モデルに分類できることを示し，その中から新しい性能評価指標を提案する．性能予測の研究では，プログラムと計算機の構造の情報を使うホワイトボックスアプローチとは対照的な，プログラムと計算機の構造の情報を使わない手法であるブラックボックスアプローチの視点から並列処理の時間モデルを構築する方法を提案する．また予測精度を向上するモデルパラメータ決定方法を提案する．これまでのモデルの精度向上は，プログラムや計算機の構造をより詳細にモデルに反映することにより行われてきた．ここで提案する方法は，モデル式が与えられたとき，モデルパラメータ推定で生じる過剰適合（ノイズをモデル化する現象）を抑制して予測精度を向上する，従来のモデル化にはない新しい方法である．

2.1　並列性能評価指標の提案

並列処理では並列オーバーヘッドの影響を定量化できる，「並列効率」が重要な性能指標で，古くから用いられてきた．ところがこの指標の決定には，並列処理で用いたプログラムと入力データを 1 プロセッサで実行した処理時間（逐次処理時間）が必要とな

る．故に，逐次処理時間を測定できない日常の計算機利用における並列処理をこの指標で評価することは困難であった．そこで本研究では 1 プロセッサの実行時間を必要としない新たな並列効率指標を創出することを試みた．性能指標の導出は図 2-1 で示した並列処理時間をどのようにモデル化するかに密接に関係する．そこで今まで並列処理の性能評価に用いられた時間モデルを調査したところ，図 2-2 のように 4 つの型のモデルに分類できた [2-20]．

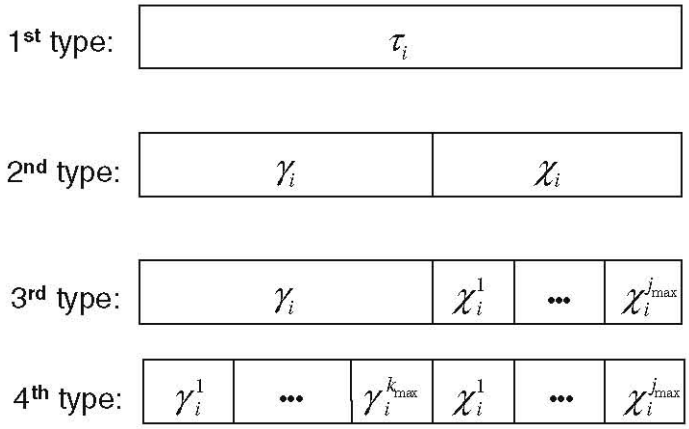


図 2-2 並列性能を特定する 4 つの時間モデル．

（k_{max} は並列プロセス数．j_{max} は並列オーバーヘッドの要因数）

第 1 型モデルは並列処理をしている各プロセッサの処理時間のみを考慮する．第 1 型は最も簡単であるが，例えば式(2-1)のようなスピードアップ $S(p,n)$ をプロセッサ数 p で除した並列効率 E_p で並列性能を評価することができる [2-2]．ここに n は問題の大きさである．

$$E_p(p,n) \equiv \frac{S(p,n)}{p} \tag{2-1}$$

$S(p, n)$は通常式(2-2)が用いられる．

$$S(p,n) = \frac{\tau(1,n)}{\tau(p,n)} \tag{2-2}$$

ここに $\tau(1,n)$，$\tau(p,n)$は処理を計算機に投入してから結果が出るまでの時間即ち，ウォールクロックタイムである．

この並列効率を用いて isoefficency と呼ばれる並列処理のスケーラビリティに対する判定条件が提案された [2-3]．

プログラムが実行した全四則演算数 $F(n)$が求まると，演算器の論理性能 r_a (flops)を用いて理想的な単一プロセッサの処理時間 $F(n)/r_a$ を計算することができる．これを基準として次のように総合的なスピードアップ S_t を定義し，これに基づく効率 E_t を得る [2-4]．これらを式(2-3)，式(2-4)に示す．

$$S_t(p,n) \equiv \frac{F(n)/r_a}{\tau(p,n)} \tag{2-3}$$

$$E_t(p,n) \equiv \frac{S_t(p,n)}{p} \tag{2-4}$$

並列処理では，プロセッサ数を増やしていくとデータが全てキャッシュメモリに乗り急激に S や E_p の値が大きくなり，E_p が 1 を越える $\tau(1,n)>\tau(p,n)\cdot p$ 現象が生じる．このような，何らかの理由で計算機の性能が低下したときに測定した基準値 $\tau(1,n)$を用いた S や E_p は，劇的な性能改善値を示す場合があるため，良い性能評価指標とは言えない．その点 E_t はメモリアクセス等で計算機の性能が低下する影響が含まれない基準 $F(n)/r_a$ により最大値が 1 を超える可能性は少なく，浮動用数点演算主体の計算では適切な性能評価指標と言える．

一方式(2-3)の代わりに r_a を除いた式(2-5)の性能評価指標がしばしば使われる．$F(n)$を計測できる計算機では，浮動用数点演算主体の計算では良い性能評価指標となる．式(2-5)のスピードを用いた Isospeed と呼ばれるスケーラビリティに対する判定条件が提案された [2-5]．有名な Linpack ベンチマークテスト [2-6]も性能評価指標としてこの指標を用いる．この論理性能 r_a で除した効率は並列計算機を評価する指標としてよく用いられる．

$$S_{flops}(p,n) \equiv \frac{F(n)}{\tau(p,n)} \tag{2-5}$$

第 1 型モデルを用いた新しい指標としては，heterogeneous な並列システム（性能が異なるプロセッサを用いた並列システム）に対して提案された，n_{eff}（実効プロセッサ数）と呼ばれるものがある [2-7]．これを式(2-6)に示す．

$$n_{eff} = \frac{\sum_{i=1}^{p} \tau_i}{\tau} \tag{2-6}$$

第 1 型のモデルは，ハードウェア，ソフトウェア，プログラムの構造をモデル化しないので，全ての並列処理に適用できる．従って第 1 型のモデルを基にした性能評価指標は任意の並列計算機とプログラムから成る並列処理の性能を評価することができる．

第 2 型のモデルは並列処理部と並列オーバーヘッドの 2 つの部分に分けたモデルである．スピードアップ値とプロセッサ数から得られる逐次実行部の時間を算出し，これを性能指標とした性能評価方法が示された [2-8]．待ち時間の性能指標が並列処理のスケーラビリティ評価のために提案された [2-9]．

式(2-7)はスピードアップとプロセッサ数の関係を表す Amdahl の法則 [2-10]である．

$$S_A(p,n) = \frac{1}{1 - R_p + R_p / p} \tag{2-7}$$

ここで R_p は並列化率で，1 プロセッサで処理した時の並列処理部の処理時間とジョブ全体の処理時間の比である．プロセッサ数の増加と共に問題の大きさを大きくしたときのスピードアップを表す Gustafson の法則 [2-11]も第 2 型モデルに分類される．第 2 型モデルで構築された性能評価指標と法則は，ロードインバランスが無い全ての種類の並列処理に適用できる．

第 3 型モデルでは，第 2 型モデルで導入した並列オーバーヘッドの要因が考慮される．要因としては逐次実行部，通信，同期等々がある．並列オーバーヘッドにより生じる並列性能の上限が，問題の大きさを固定して解析的に研究された [2-12]．5 つの性能要因，ロードインバランス，並列性，同期，通信，計算機資源の競合，を用いた性能予測

が行われた[2-13]．一般にこれらの要因を用いた時間モデルは多くのモデルパラメータを持つ．ネットワークのレイテンシとバンド幅は，ネットワーク性能の性能評価指標であり，通信時間モデルのパラメータでもある．例えば，並列化した数値シミュレーションプログラムを用いると異なった種類の通信が複数発生する．従って並列処理の性能を評価しようとすると，複数の通信時間モデルを構築してアッセンブルする必要が生じる．例えば MPI-SIM と呼ばれるプログラムでは，プログラムを実行しながら通信時間モデルを用いて MPI ライブラリの処理時間を見積もり，NAS パラレルベンチマークの処理時間を予測した[2-14]．

第 4 型モデルでは，並列処理部がプロセッサ数と問題の大きさ等の関数としてモデル化される．最大性能の半分を達成するデータ数が，パイプラインベクトル計算機の四則演算を特徴付ける指標として導入された[2-15]．最大性能の半分を達成するプロセッサ数が，性能指標として導入された[2-16]．アルゴリズムのスケーラビリティを特徴付けるため，問題の大きさが変化する場合の並列性能が解析的に研究された[2-17]．第 4 型モデルを用いると，第 1 型モデルで述べた isoefficiency はプロセッサ数と問題の大きさの関数で現わすことができるようになる．これにより，プロセッサ数と問題の大きさが変化する並列処理の性能を評価できる．第 4 型モデルを基にアルゴリズムに対して isoefficiency 関数を決定してスケーラビリティ解析が行われた[2-18]．第 4 型モデルで描けるようになったプロセッサ数と問題の大きさに対する isoefficiency 曲線を用い，複数のアルゴリズムから成る数値シミュレーションの性能評価ができることが示された[2-19]．

2.1.1　並列効率指標の提案

第 1 型モデルで定義された「並列効率」は，1 型から 4 型の性能評価指標と共に用いることができる共通の性能評価指標である．また殆ど全ての並列処理で利用できる指標でもある．しかし同じ計算結果となる処理を逐次処理と並列処理に対して行うことが必

要となり，常時の並列処理（プロダクションラン）の評価には向かない．さらに高並列処理の逐次処理時間を測定することは不可能である．この問題を解決するため逐次処理時間を用いずに第 2 型モデルて定義できる新しい並列効率（以後，新並列効率）を提案した[2-20]．同時にこの新並列効率とロードインバランスと他の並列オーバーヘッドを表す指標を第 1，2，3 型モデルで定義して導入し，新並列効率と並列オーバーヘッドを 1 つの関係式に統合し，並列効率への並列オーバーヘッドの寄与を定量的に把握することを可能にした．さらに新並列効率から限界加速率を導出して並列処理で達成できる性能のポテンシャルを評価することを可能にした．

式(2-8)は式(2-1)と(2-2)を 1 つにした並列効率 E_p の定義である．p はプロセッサ数，$\tau(1)$ は逐次処理時間，τ_i は並列処理をしている各プロセッサの処理時間であり，$\tau \equiv \underset{i=1}{\overset{p}{\mathrm{Max}}}(\tau_i)$ である．

$$E_p \equiv \frac{\tau(1)}{p \cdot \tau} \tag{2-8}$$

$\tau(1)$ に依存しない効率指標を抽出するために，既存の効率を式(2-9)のように 2 つの指標に分解する．

$$E_p \equiv \frac{\varepsilon_p}{R_{CPU}} \tag{2-9}$$

ここに $\varepsilon_p = Y/p \cdot \tau$ で $R_{CPU} = Y/\tau(1)$ である．Y は任意の関数で，図 2-2 の第 1 型のモデルは $Y = 1$ である．

ここで γ_i を図 2-2 の第 2 型モデルの並列部の処理時間として $Y \equiv \sum_{i=1}^{p} \gamma_i$ とすると，式(2-10)で示すような τ で規格化された効率 ε_p となる．これを新しい並列効率の指標（新並列効率）とすることを提案する．

$$\varepsilon_p = \frac{\sum_{i=1}^{p} \gamma_i}{p \cdot \tau} \tag{2-10}$$

ウォールクロックタイムτはほぼ全ての並列処理で出力される．γ_iは日常の並列処理でも観測できる．従ってε_pはほぼ全ての並列処理の性能を評価できる性能評価指標である．本論文では以後ε_pを新並列効率と呼ぶことにする．

一方R_{CPU}は式(2-11)に示すように p 個のプロセッサに分割して並列処理したときの時間の総和を$\tau(1)$で規格化したもので，逐次処理と並列処理が同じであればR_{CPU}=1である．もし並列処理によりキャッシュメモリ上のデータを使った計算が増え，データアクセススピードが上がった場合にR_{CPU}<1となる，スーパースカラ状態になることもある．逆にもし並列処理によりデータアクセススピードが遅くなればR_{CPU}>1となる．従ってR_{CPU}はCPUの効率指標となる．

$$R_{CPU} = \frac{\sum_{i=1}^{p} \gamma_i}{\tau(1)} \qquad (2\text{-}11)$$

ε_pのレンジは$0 \le \varepsilon_p \le 1$で，$E_p > 1$となるスーパースカラの影響即ち個々のプロセッサの性能効率はR_{CPU}に分離され，ε_pは処理を各プロセッサに分割する効率を表す指標となる．

2.1.2 並列オーバーヘッド指標の提案

新並列効率ε_pは 2 つのタイプの並列オーバーヘッド指標で定義することができる．1つ目は式(2-12)で定義されるR_bで，図 2-2 の第 1 型モデルで定義できるロードバランスを記述する．もう一つの指標，式(2-13)で定義されるR_{imp}は，図 2-2 の第 2 型モデルで示した並列オーバーヘッドχ_iの総和である．以後本論文ではR_bをロードバランス指標，ロードバランスを除いた並列オーバーヘッドとしてR_{imp}を並列阻害指標と呼ぶこと

にする．これら 2 つの指標と $\tau_i = \gamma_i + \chi_i$ 及び $\varepsilon_p = \sum_{i=1}^{p}(\tau_i - \chi_i)/(p \cdot \tau)$ の関係を用いると，新並列効率とロードバランス指標，並列阻害指標の関係は式(2-14)として簡単な関係で結び付けることができる．

$$R_b = \frac{\sum_{i=1}^{p} \tau_i}{p \cdot \tau} \tag{2-12}$$

$$R_{imp} = \frac{\sum_{i=1}^{p} \chi_i}{\sum_{i=1}^{p} \tau_i} \tag{2-13}$$

$$\varepsilon_p = R_b \cdot (1 - R_{imp}) \tag{2-14}$$

式(2-14)を用いることにより，並列効率はロードバランス指標と並列阻害指標を用いて定量的に説明できるようになり，これら 3 つの性能評価指標を用いた並列性能評価ができるようになる．これらの指標のレンジは $1/p \le R_b \le 1$，$0 \le R_{imp} \le 1$ である．これにより並列オーバーヘッドを並列効率と定量的に関係付けることが可能となる．

通常，並列阻害指標の要因は逐次処理，通信，その他となるので，これらの処理時間を χ_i^s，χ_i^{com}，χ_i^{others} とすると，$\chi_i(p) = \chi_i^s + \chi_i^{com} + \chi_i^{others}$ となる．式(2-13)にこの関係を代入して式(2-15)を得る．

$$\begin{aligned} \varepsilon_p &= R_b \cdot \left(1 - \frac{\sum_{i=1}^{p} (\chi_i^s + \chi_i^{com} + \chi_i^{others})}{\sum_{i=1}^{p} \tau_i} \right) \\ &= R_b \cdot (1 - R_s - R_{com} - R_{otheers}) \end{aligned} \tag{2-15}$$

ここに $R_s = \sum_{i=1}^{p} \chi_i^s / \sum_{i=1}^{p} \tau_i$，$R_{com} = \sum_{i=1}^{p} \chi_i^{com} / \sum_{i=1}^{p} \tau_i$，$R_{others} = \sum_{i=1}^{p} \chi_i^{others} / \sum_{i=1}^{p} \tau_i$ は R_{imp} の要素を説明するサブ指標である．

一般に，図 2-2 の第 3 型モデルで示す $j_{\max}$ の並列阻害要因は $R_j \equiv \sum_{i=1}^{p} \chi_i^j / \sum_{i=1}^{p} \tau_i$ により一律にその並列効率に対する影響を評価することができる．一般化した式(2-15)を式(2-16)に示す．

$$\varepsilon_p = R_b \cdot \left(1 - \frac{\sum_{j=1}^{j_{\max}} \sum_{i=1}^{p} \chi_i^j}{\sum_{i=1}^{p} \tau_i} \right) = R_b \cdot (1 - R_1 - R_2 - \cdots - R_{j_{\max}}) . \qquad (2\text{-}16)$$

上記に示すように，並列阻害指標はサブ指標の総和 $R_{imp} = \sum_{j=1}^{j_{\max}} R_j$ として記述できる．

式(2-15)と式(2-16)を見比べ， $R_s \to R_1$ ， $R_{com} \to R_2$ ， $R_{others} \to R_3 + \square + R_{j\max}$ が確認できる．

ここで再び時間モデルの分類をした図 2-2 を見ると， ε_p と R_{imp} は第 2 型モデルで定義されたことが確認できる．サブ指標 R_j は第 3 型モデルで定義できるようになる．また E_p と R_b は第 1 型モデルである．このようにモデルを第 1 型から第 2 型，3 型と複雑化することにより，既存の並列効率から $\tau(1)$ を分離することが可能になった．

2.1.3 加速限界指標の提案

並列効率指標から，並列処理の加速限界を導くことができる．図 2-3 (a)に p=4 の場合の典型的な並列処理時間を示す．図 2-3 (b)は限界加速率を見積もるための時間モデルである．このモデルは図 2-3 (a)と同じ ε_p と τ であるとする．また全てのプロセッサの処理時間が等しい $R_b = 1$ のロードバランスであると仮定し，その並列処理部の処理時間は $\sum_{i=1}^{p} \gamma_i / p$ とする．このとき加速限界指標 A_{LT} は $\sum_{i=1}^{p} \gamma_i / p \to 0$ とした式(2-17)で現わすことができる．

$$A_{LT} \equiv \frac{1}{1-\varepsilon_p} \tag{2-17}$$

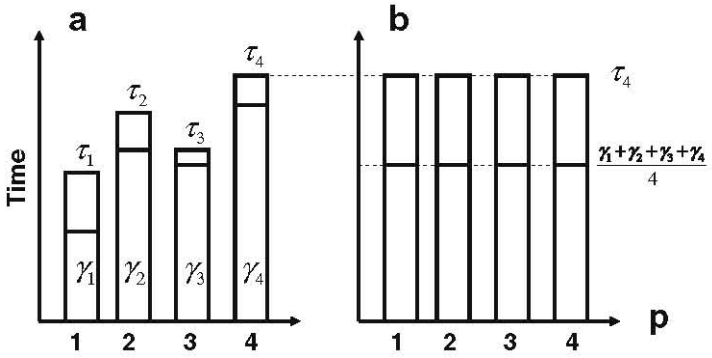


図 2-3　ロードインバランスがある場合の加速限界

(a)　測定データ．(b) $A_{LT}(4)$は γ_i の平均値から決定．

2.1.4 ループレベルの性能評価指標の提案

あるプログラムに適した計算機システムを選ぶ場合はプログラム毎に性能評価が必要となる．一方並列計算機に適したプログラムやアルゴリズムを構築する場合や，プログラムに適した計算機の設計を行う場合は，典型的な処理（大抵の場合繰り返し処理を記述したループ）単位の性能評価が必要となる．そこでループ単位の性能評価指標を提案した [2-19]．

ループ単位の逐次実行処理時間 $t_u(n)$は，この時間が四則演算実行回数 $f(n)$に比例すると仮定して，式(2-18)のようにモデル化する．ここに n はループの回転数である．するとループ単位の性能評価は，理論最大性能 r_a(Mflops)を基準とした効率 a_u で可能となる．ここに，t_{0u} は種々の要因から成る前処理，後処理時間である．

$$t_u(n) \equiv t_{0u} + \frac{f(n)}{r_a \cdot a_u} \tag{2-18}$$

$t_u(n)$が並列化可能なループの逐次実行処理時間の場合，ループ単位の並列処理時間 $t_p(p, n)$は，プロセッサ数 p を基準とした効率を e_p として，式(2-19)のようにモデル化すると，理論最大性能 p 倍を基準とした性能評価が可能となる．

$$t_p(p,n) \equiv t_{0p} + \frac{t_u(n)}{p \cdot e_p} \tag{2-19}$$

t_{0p} は並列処理のため生じる種々の要因から成る前処理，後処理時間と捉える事ができる．ロードインバランスによる並列オーバーヘッドは e_p で捉えることができる．

通信時間 $t_c(p, n)$は，通信量 $g(p, n)$に比例していると仮定して，次式の様にモデル化する．

$$t_c(p,n) = t_{0c} + g(p,n) \cdot \frac{X_B}{r_c \cdot e_c}$$

ここに，通信の理論最大性能 r_c(MB/s)を基準とした比例係数 e_c は通信の効率を表わす．t_{0c} は立ち上がり時間である．$g(p, n)$は，放送，転置転送等，通信を実現する処理内容により異なり，ネットワークの形状を考慮したモデルとなる．ここで，X_B は 1 データのバイト数で，Fortran の単精度で 4 バイト，倍精度で 8 バイト等である．

ループのモデル係数 a_u，e_p，e_c は理論最大性能という基準値に対する効率と考えることができ，ループの性能評価のための性能評価指標となる．また t_{0u}，t_{0p}，t_{0c} はループのオーバーヘッドの性能評価指標である．

これらの性能評価指標で以下のような複数の要因が評価できると考えられる．

e_p：ロードバランス

t_{0p}：並列処理の前処理・後処理オーバーヘッド

e_c：通信の実効バンド巾

t_{0c}：通信の立ち上がりオーバーヘッド

a_u：複数演算機による四則演算の同時実行状況を含んだ演算器の効率

t_{0u}：四則演算のパイプライン処理

a_u，t_{0u}：メモリアクセス処理

ループ毎のロードバランスが e_p で評価できる．並列ライブラリ等の前処理・後処理計算のため新たに処理が生じる場合は t_{0p} で検知できる．

パケットの作成等, 実際にデータ転送が始まるまでの種々の処理が t_{0c} に反映される．通信網の途中に介在するスイッチ等の中継効率を e_c で評価できる．

論理最大 flops 値は，複数の演算器が同時に実行して達成されるように設計されている場合が多い．a_u 値から，複数の演算器やパイプライン処理による四則演算の同時実行状況を含んだ演算器の効率を推測することができる．例えば a_u が 0.6 であれば，2 つの四則演算器が同時実行していることがわかる．a_u が 0.5 以下の場合，演算器が 1 つしか動かなかった場合或は，2 つ動いているが演算効率が悪い場合が考えられる．またメモリアクセス性能が悪い場合や, 段数が多いパイプライン四則演算器のでパイプライン処理ができないと a_u は 0.1 以下になる可能性もある．スカラ計算機ではキャッシュミスヒット，ベクトル計算機ではバンク競合がこれに当たる．分岐処理は，スカラ計算機ではパイプライン処理を乱す原因となり，a_u が低くなる原因となる．演算の立ち上がりが問題になる場合は，t_{0u} に値が反映される．例えば，パイプライン処理の立ち上がりがこれに当たる．

2.2　性能予測のための処理時間のモデル化

2.1 節ではあるプロセッサ数，ある問題の大きさのときの新並列効率とループレベルの性能評価指標を提案した．並列処理の性能評価では，小規模な問題で測定した処理時間データに基づいて大規模な問題の処理時間と必要なプロセッサ数を予測することが重要となる．そこで図 2-2 の第 4 型モデルで並列処理の性能予測に関する研究が行われてきた．本節の提案の一つは第 4 型モデルを用いたものである．

始めに処理時間のモデル化の課題を整理し本研究が解決した課題について述べる．並列処理時間は種々の要因の影響を受けて決まる．主に解く問題の性質，問題を解くために用いたプログラム，そのプログラムの入力，プログラムを実行する計算機がそれに当たる．これらの要因に取り込んだモデルを構築しようとすると，プロセッサ数と問題の

大きさに影響されるこれらの要因を含んだ処理時間モデルを構築することになる．このモデル構築には大別して２つのアプローチがある．一つはホワイトボックスアプローチで，プログラムと計算機に関する情報を用いてモデルを構築する方法である．従ってこのモデルによる予測は使用した情報を根拠として行われ，情報量を増やすことにより予測精度の向上が期待できる．

ここ 30 年間に提案された多くのモデル化方法がこのホワイトボックスアプローチに分類でき，多くのモデルはプログラムで用いたアルゴリズムに基づいて演算回数を式化している．文献 2-21), 2-22), 2-23)の方法はこのタイプのモデル化方法である．

コンパイル時にプログラム中の逐次処理，並列処理，通信の部分を識別し，ループの繰り返し回数を決定する静的モデル化方法が提案された [2-24]．この方法で，幾つかの数値シミュレーションプログラムの処理時間を正確に予測することができた．しかしループの繰り返し回数は普通入力データによって決まるため，適用できるケースが限られる．

処理時間モデルを 2 つのレベルに階層分けして構築することが提案されてきた．上位のレベルを無閉路有向グラフで並列タスクを記述し，下位レベルの待ち行列ネットワークソルバを用いてタスクの状態を決定するモデル化が行われた [2-25]．タスクの状態数の軽減が，逐次と並列処理を無閉路化された有向グラフ (Series-parallel directed acyclic graph) を用いて表すことにより図られた [2-26]．累計されたタスクグラフ(Augmented task graph)を用いたプログラムの動きの記述の簡略化が提案され，プロセッサ数と問題の大きさを関数とするモデルが LU 分解法に対して示された [2-27]．通信と同期を記述するスレッドグラフによる方法が提案された [2-28]．グラフを用いると他の方法では難しいロードインバランスのモデル化が可能となる場合がある．2 レベルの階層でモデルを構築する意図は，上位のレベルでプログラム依存の部分をグラフで記述し，計算機依存の部分を下位レベルで記述することにある．このように分離すると原理的には異なる計算機に対して低レベルの変更だけで済むため，構築したモデルのポータビリティが得られると

考えられる．一方 2 レベルの階層でのモデル構築には 2 つの問題が伴う．1 つはデータアクセスである. データアクセスパターンはプログラムを実行する時のデータの入力の仕方により異なる場合がある．2 つ目の問題は昨今の計算機では複数の四則演算器が同時に稼働するように設計されているが，同時に稼働するかどうかは計算の種類，プログラミングスタイル，使用したコンパイラの最適化の度合い，演算器の状態に依存することである. 例えば入力データが飛び飛びでキャッシュミスヒットが頻発するような計算は論理性能の 10 分の 1 の性能も出ないことがしばしばあるが，これを予め下位レベルのモデルとして構築することは難しい．従ってこの方法を現在の計算機に適用すると，予測モデルとしての精度が得られない場合があると考えられる.

これら 2 つの問題は，従来から行われてきた測定した時間にアルゴリズムの演算数を基にしたモデル式をフィッティングすることにより解決できる [2-29), 2-22), 2-23)]．この方法がプログラム規模の処理時間のモデル化に対しても有効であることが，ベクトル並列計算機とスカラ並列計算機で数値シミュレーションコードを実行した並列処理のモデル化により確認された [2-19), 2-30)]．この方法は性能が異なる計算機をつなげた並列処理にも適用された [2-31)]．一方この方法の問題点は，プログラムが大きくなるにつれモデルパラメータが増加すること，それに伴いモデルパラメータを決定する測定箇所が増えることである．また測定により処理時間の増加，時間測定用の関数がコンパイラの最適化を乱す可能性が生じる. さらにモデルパラメータの推定に膨大な時間を必要とする場合がある.

このモデルパラメータ増加の問題は，プログラムを各フェーズに分割してモデルを作り通信モデルを重ね合わせることによりランタイムをモデル化する，フェーズ並列モデル化方法(phase-parallel modeling method)が適用できれば解決することができる [2-32)]．巨大で複雑なモンテカルロコード MCNP のモデル化 [2-34))]はこの手法が適用されていると捉えることができる. このようにこの方法では巨大で複雑なモデル構築が可能と考えら

れるが, プログラムをフェーズに分割するためにプログラムの構造情報とその内容の理解が必要となる.

ここでプログラムの構造情報を用いないブラックボックスアプローチを考える. このアプローチの処理時間のモデルは, 並列処理の入力と出力を基に, ブラックボックスを記述する何らかの関数のパラメータを決定してして構築される. このためホワイトボックスアプローチのフェーズ並列モデル化方法で生じた問題「プログラムの構造情報とその内容の理解」は不要となる. またモデルパラメータ数もホワイトボックスアプローチと比較して少なく抑えることができる.

ブラックボックスのアウトプットは, ホワイトボックスアプローチでは捉える事が難しかったコーディングには書かれていない処理の費やす時間をモデル化できる場合がある. これの例については 4.2.2 節の(3)で述べる.

幾つかの研究がこのアプローチに分類できると考える. 同期をモデル化するためにプロセッサ数を変数としたパイプライン関数が基底関数として用いられた [2-16]. 並列オーバーヘッドのモデルとしてプロセッサ数に比例する基底関数が用いられた [2-35]. 一方これらの基底関数だけでは不十分で, 例えば基底関数がプロセッサ数の対数の場合もある. 従ってブラックボックスアプローチでは, 並列オーバーヘッドを表す普遍的な基底関数が必要となる. 全ての並列オーバーヘッドの情報を, ロードバランス, 並列度, 同期, 通信, 資源の競合によるロスに分類して出力する並列計算機 Kendall Square Research KSR1 が製造され, その出力に基づくプロセッサ数を変数としたモデル化方法が提案された [2-13]. この全並列オーバーヘッドの情報は KSR1 のハードウェアとコンパイラによって得られたが, 今日の計算機でそのような全並列オーバーヘッドの出力は一般的で無い.

上記 3 つのブラックボックスアプローチはプロセッサ数のみを変数とするモデル化であった. これにより従来から行われてきた問題の大きさを一定にしてプロセッサ数の

増加に伴う処理時間の増減を評価するストロングスケーリングを行うことができる. 一方現在は 1 プロセッサの計算量を固定し, 問題の大きさが大きくなったらプロセッサ数を増やすウィークスケーリングが良く行われる. このためにはプロセッサ数と問題の大きさを変数としたモデル化が必要である. 最近の研究では, 特定の入力と処理時間を回帰モデルにより関係づけ, 問題の規模とプロセッサ数を大きくした時もプロセッサ毎の処理時間を一定に保つ入力値を予測する方法が提案されている [2-36].

順序統計を用いてロードインバランスを伴う並列処理の処理時間を予測する方法がある. 非同期で行われる並列処理に順序統計が適用され, 平均実行時間の幅がモデル化された [2-37]. 逐次処理と並列処理が交互に現れる処理の同期に起因したスピードアップの上限の幅が, スレーブタスクの処理時間が指数分布や正規分布に従う場合に対して導かれた [2-38]. 二項分布をする同期間の時間から, 繰り返し法を用いたアルゴリズムのスピードアップの下限の幅が求められた [2-39]. ランタイムに依存した要因とシステムに起因した要因, 例えばキャッシュのミスヒット, を含んだランタイムが, 「分割して統治する並列化方法」と SPMD の共分散効果を考慮することにより, 並列処理をシュミレーションした結果を精度よく予測できた [2-40]. これら 3 つの結果は実際の並列処理による確認が待たれるところである. 特に高並列で上下限が生じると予想されている文献 [2-38), 2-39)]の予測の確認は重要であると考える.

これらの背景において, 本論文では実務で行われる並列処理のモデル化の研究について述べる. 日常行われている並列処理は, 複数の並列処理や逐次処理の集合体であり, モデルも複雑となる. そこで「数値シミュレーションプログラム」を実行したときの並列処理をモデル化の対象とし, まずホワイトボックスアプローチでモデル化する方法を提案する [2-19]. 次に数値シミュレーションプログラムを用いてプロセッサ数と問題の規模を変数としたモデル化をブラックボックスアプローチで行う方法を提案する [2-41), 2-42)]. これらのアプローチでモデルパラメータを一度に決定しようとすると過剰

適合という問題が生じる．また上記のホワイトボックスアプローチで述べたフェーズ並列モデル化方法でも，フェーズ中の処理をモデル化しようとすると過剰適合問題が生じことが予想される．そこでこの過剰適合をモデルパラメータに非負条件を課し，数式処理のアルゴリズムの一つ限量記号消去法(QE)を用いて決定する方法を提案した[2-45), 2-46)]．本論文では，扱えるモデルパラメータとデータ数を QE に比べて多くすることができる線形計画法のアルゴリズムの一つシンプレックス法を用いる方法[2-47)]を提案する．

2.2.1　ホワイトボックスアプローチ

数値計算プログラムの主要処理処理をループ処理単位にモデル化し，それらを合成してプログラム全体の処理時間をモデル化する方法を提案する[2-19)]．

並列処理の処理時間 τ は，プロセッサ数を p，問題の大きさを n とした時に式(2-20)のようにモデル化することができる．

$$\tau(p,n) \equiv \alpha(n) + \frac{\beta(n)}{p} + \sigma(p,n) \qquad (2\text{-}20)$$

ここに，α：逐次実行部の処理時間，β：並列化可能部分の逐次実行時の処理時間，σ：通信，同期等々から生じる並列オーバーヘッドである．

(1)　α，β のモデル化方法

ループ単位の逐次実行処理時間 $t_u(n)$は，原則的に $t_u(n)$が四則演算実行回数 $f(n)$に比例すると仮定して，式(2-21)のようにモデル化する．ここに $t_{0u}(n)$は種々の要因から成る前処理，後処理時間と捉える事ができ，$c_1 = 1/(r_a \cdot a_u)$ とすると(2-18)に一致する．

$$t_u(n) \equiv t_{0u} + c_1 \cdot f(n) \qquad (2\text{-}21)$$

$t_u(n)$が並列化可能なループの逐次実行処理時間の場合，ループ単位の並列処理時間$t_p(p, n)$は式(2-22)のように記述できる．t_{0p} は並列処理のため生じる種々の要因から成る前処理，後処理時間と捉える事ができ，$c_2 = 1/e_p$ とすると(2-19)に一致する．

$$t_p(p,n) \equiv t_{0p} + c_2 \cdot \frac{t_u(n)}{p} \tag{2-22}$$

ここで，プログラム全体の並列化可能なループ数を L_p，非並列化ループ数を L_{nonp} とすると，式(2-20)の $\alpha(n)$，$\beta(n)$は次のように表わす事ができる．

$$\alpha(n) = \sum_{i=1}^{L_{nonp}} \{t_u(n)\}_i + \sum_{j=1}^{L_p} \{t_{0p}\}_j \tag{2-23}$$

$$\beta(n) = \sum_{j=1}^{L_p} \{\frac{t_u(n)}{e_p}\}_j \tag{2-24}$$

(2)　σ のモデル化方法

σ には，通信を含んだ処理，同期処理，タスクの生成消滅等の並列オーバーヘッドが含まれるが，ここでは，例えばプロセッサ間の総和計算のような，通信を含んだ処理のモデル化方法を示す．

通信時間 $t_c(p, n)$は，通信量 $g(p, n)$に比例していると仮定して，式(2-25)の様にモデル化する．

$$t_c(p,n) = t_{0c} + g(p,n) \cdot \frac{X_B}{r_c \cdot e_c} \tag{2-25}$$

ここに通信の理論最大性能 r_c(MB/s)を基準とした比例係数 e_c は，通信の効率を表わす．t_{0c} は立ち上がり時間である．$g(p, n)$は，放送，転置転送等，通信を実現する処理内容により異なり，ネットワークの形状を考慮したモデルとなる．ここで，X_B は 1 データのバイト数で，Fortran の単精度で 4 バイト，倍精度で 8 バイト等である．

プログラム全体の通信を含んだ処理の並列オーバーヘッド σ は，その総数を m_{com} とすると，式(2-26)の様に表わす事ができる．

$$\sigma(p,n)=\sum_{m=1}^{m_{com}}\{t_c(p,n)+t_{others}(p,n)\}_m \qquad (2\text{-}26)$$

ここに $t_{others}(p, n)$は通信処理等の並列化のため新たに生じる演算で，式(2-21)，(2-22)を用いてモデル化する．例えばプロセッサ間の総和を取る場合，逐次計算には無い新たな四則演算が発生するため，この項でそれを考慮する．

2.2.2 ブラックボックスアプローチ

数値シミュレーションプログラムの並列処理を図 2-4 のようにブラックボックスの並列処理システムと考え，その出力から得られた情報で並列処理時間をモデル化する方法[2-41), 2-42)]を提案する．

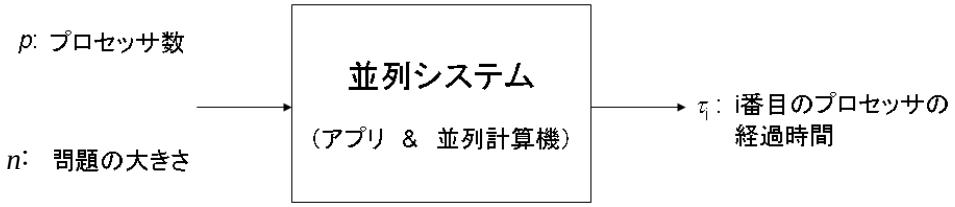


図 2-4　1 出力並列処理システム

図 2-4 はある並列システムに対し，プロセッサ数 p と問題の大きさ n を入力した場合，出力として各プロセッサ i の経過時間 τ_i が得られることを想定している．提案するモデル化方法は，これらの情報のみから並列処理システムの時間モデルを構築する．モデル化する時間は $\tau=Max_{i=1}^{p}(\tau_i)$ で，ロードインバランスでばらつく τ_i の中で一番長い経過時間とすることし，プロセッサ数と問題の大きさの関数としてモデル化し，式(2-27)でモデル化できると仮定した．ここに a は並列化部の時間中 p に対して反比例して減少

する時間で，X は並列オーバーヘッドである．また τ と X は問題の大きさと p の関数，a は問題の大きさの関数である．

$$p \cdot \tau \equiv a + p \cdot \mathrm{X} \tag{2-27}$$

ここで a の値に近い定数 Γ を導入し，式(2-27)の両辺から Γ を引いて変形すると式(2-28)を得る．

$$\frac{1-\varepsilon_p}{\varepsilon_p} \equiv \frac{1}{\Gamma}\left(a - \Gamma + p \cdot \mathrm{X}\right) \tag{2-28}$$

ここに $\varepsilon_p = \dfrac{\Gamma}{p \cdot \tau}$，$\tau$ は測定値であるので式(2-28)の左辺は測定値より求まる．

ここで式(2-28)の左辺が多項式で近似できると仮定して式(2-29)を得る．

$$\frac{1-\varepsilon_p}{\varepsilon_p} \equiv c_0 + p \cdot \left|c_1\right| + p \cdot \sum_{j=2} c_j (p-1)^{j-1} \tag{2-29}$$

式(2-28)と(2-29)を比較すると a と X について式(2-30)，(2-31)の関係を得る．ここで c_1 は逐次処理時間に対応する.しかし他項に比べて小さくなる場合やフィッティングにおいて負の値になる場合があるため絶対値を取る．

$$a = \Gamma \cdot (1 + c_0) \tag{2-30}$$

$$X = \Gamma \cdot \left(\left|c_1\right| + \sum_{j=2} c_j (p-1)^{j-1} \right) \tag{2-31}$$

問題の大きさ x を固定し，プロセッサ数 p を変えて τ を測定し，その値を式(2-29)でフィッティングして $c_0, c_1, c_2 \ldots$を決定すると，式(2-30)と(2-31)，式(2-27)より p を変数とした処理時間 τ のモデルを構築することができる．

フィッティングに式(2-29)を用いる利点は 3 つある．第 1 に測定値 τ のみからモデルが構築できること．第 2 に処理中の全ての並列オーバーヘッドが X としてモデル化さ

れること．第 3 にフィッティングする時間に $p\cdot\tau$ を用いており，τ 以外の τ_i の変動を考慮しなくてすむことにある．一方欠点としては，多項式でフィッティングできる並列オーバーヘッドを持つ並列システムに，適用が限定されてしまうことである．また Γ 値を試行錯誤で探さなければならないことも問題となる．

これらを改善するため図 2-5 のように新たに並列化部の経過時間 γ_i を測定し，a を式(2-30)からではなく τ_i とは独立に決定することを提案する．

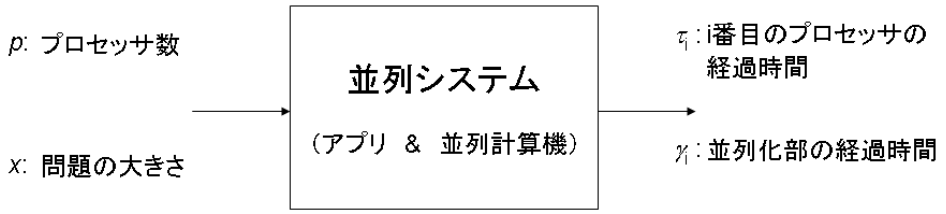


図 2-5　2 出力並列処理システム

測定した γ_i は式(2-32)のようにモデル化し，フィッティングにより a を決定する．ここに χ_{inv} は並列化部に隠れている並列オーバーヘッドである．

$$\sum_{i=1}^{p}\gamma_i / p \equiv a / p + \chi_{inv} \tag{2-32}$$

求めた a を $\Gamma = a$，$\varepsilon'_p = a/(p\cdot\tau)$ として式(2-28)に代入すると式(2-33)を得る． このように γ_i を用いることにより Γ を a として決定することができるようになる．

$$\frac{1-\varepsilon'_p}{\varepsilon'_p} \equiv \frac{1}{a}(p\cdot \mathrm{X}) \tag{2-33}$$

式(2-29)と同様，式(2-33)を式(2-34)のように多項式展開する．

$$\frac{1-\varepsilon'_p}{\varepsilon'_p} \equiv c_0 + p\cdot|c_1| + p\cdot\sum_{j=2} c_j (p-1)^{j-1} \tag{2-34}$$

式(2-33)と比較すると並列オーバーヘッドとして式(2-35)を得る．

$$X = a \cdot \left(\frac{c_0}{p} + |c_1| + \sum_{j=2} c_j (p-1)^{j-1} \right) \tag{2-35}$$

負の冪の項を持った式(2-35)により，図 2-4 で示した 1 出力並列処理システム即ち，式(2-31)でモデル化できなかった並列オーバーヘッドをモデル化できるようになる．図 2-6 に X=Log(p)の例を示す．図 2-6 (a)は Log(p)の数値を生成して左辺とし式(2-33)でフィッティングした結果である．Fitting 1 は j=0, 1, 2 項を用いたフィッティング，Fitting 2 は j=0, 1, 2, 3 項を用いた．図 2-6 (b)は Log(p)値を点で，p･Log(p)を式(2-34)でフィッティングして得た X のモデルを線で示す．図中 Model 1 は Fitting 1 に，Model 2 は Fitting 2 に対応する時間モデルの線である．図は Model 2 が Log(p)点ほぼ結べることを示す．

X が式(2-29)のように通常の多項式で近似できる場合は，図 2-4 のように τ_i のみから時間のモデル化が可能である．近似できない場合は，図 2-5 の τ_i と γ_i を用いた式(2-34), (2-35)によりモデル化できる場合がある．式(2-34)における c_0 が他項と比べて無視できる場合は X のモデルは通常の式(2-29)の多項式となる．

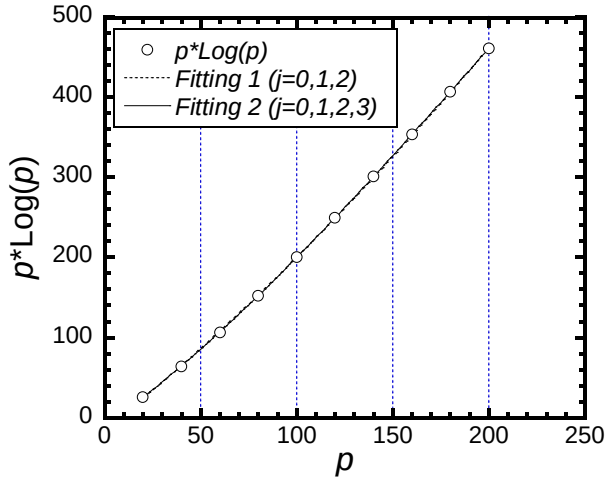


図 2-6 (a)　X=Log(p)に対する式(2-34)を用いたフィッティング結果

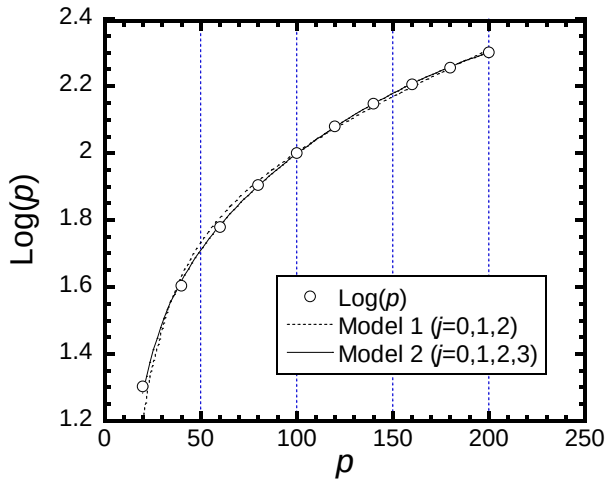


図 2-6 (b)　X=Log(p)に対する式(2-34)を用いた時間モデル

2.3 シンプレックス法による処理時間モデルの予測力の向上の提案

2.2 節で提案した処理時間のモデル化方法は，主に並列処理の手続きとその処理時間の関係をより詳細に記述することを通してモデルの精度を向上するという観点で行った提案とも言える．従来からの研究もこの観点から行われてきた．プロセッサ数の増加により処理時間が短縮される並列化部と短縮されないか増加する並列オーバーヘッドをモデル化して並列処理時間モデルを構築する方法が提案され [2-21), 2-22), 2-23), 2-16), 2-13), 2-4)]，プログラムと計算機を別々にモデル化しそれらを統合することによりモデル化する方法が提案された [2-25), 2-27), 2-28)]．性能が異なるプロセッサによる並列処理のモデル化方法も提案された [2-31)]．実際の数値シミュレーションという複雑な並列処理の時間をモデル化する方法 [2-32), 2-19), 2-34)]，複数の変数を扱うモデル化方法 [2-33), 2-36)]，モデル式を動的に補完する方法 [2-43)]などが提案されている．

一方本節では，基底関数の候補を想定したモデル式の，モデルパラメータの決定方法を工夫してモデルの予測力を向上する方法について提案する．

並列処理時間モデルは当初プロセッサ数を変数とした 1 変数モデルであったが，Gustafson[2-11)]が問題のプロセッサ数の増加と共に問題を大きくすると並列処理の性能が

維持できることを指摘して以来プロセッサ数と問題の大きさを変数とするモデルが重要となり，現在ではこの 2 変数モデルをベースにしたモデル化が一般的である．さらに実際の並列処理のモデル化では使用するモデルパラメータ数が増える．これを最小二乗法で決定すると，外挿領域の予測が大きく外れることをしばしば経験する．この原因としては，用いた基底関数がモデル化に使った観測データの振舞いをうまく記述できない，使用した観測データが外挿領域で優勢となる振舞いの情報をあまり含んでいない，観測データ中のノイズがモデル化に不要な基底関数によりモデル化される等がある．特に予測に不要な基底関数の中に冪乗等の急激に増大するものがあると，データを外挿した領域での予測値は実際とは大きく異なることになる．故に，モデルによる予測ではこのような事態の回避が重要な課題であると考えられる．

そこで，非負制約を課したモデルパラメータを持つ残差不等式を有理数演算のシンプレックス法で解いて，ノイズの主な要因の一つである観測の揺らぎのモデル化を抑制してモデルパラメータを推定する計算法を提案した [2-47]．解法にシンプレックス法を用いると，モデル化に必要ない基底関数のモデルパラメータは零となりその基底変数が用いられないため，予測を阻害する過剰適合を自動的に抑制しながら回帰モデルの予測精度の向上を図ることができる．

2.3.1 予測モデル構築における課題

最小二乗法(LSM)で予測モデルを作る際の課題を，正規方程式を使って説明する．目的変数 y を説明変数 x の M 個の基底関数 $f_k(x)$で説明する回帰モデル $\hat{y}(x)=a_0+=a_k f_k(x)$ $(k=1,\ldots,M)$を考える．係数 a_k はモデルパラメータである．このパラメータを(x_i, y_i) $(i=1,\ldots,N)$の観測データで推定する．LSM では残差平方和 E (=残$\{y_i-\hat{y}(x_i)\}^2$)を最小にする a_k を推定する．式を簡単にするため $f_0(x_i)=1$ とすると E は式(2-36)となる．

$$E = \sum_{i=1}^{N} \left\{ y_i - \sum_{k=0}^{M} a_k f_k(x_i) \right\}^2 \tag{2-36}$$

E を最小にするため，両辺をモデルパラメータ a_k で偏微分して $\partial E/\partial a_k = 0$ (k=0,…,M)とすると式(2-37)を得る．

$$\frac{\partial E}{\partial a_k} = -2\sum_{i=1}^{N} \left[f_k(x_i) \left\{ y_i - \sum_{k=0}^{M} a_k f_k(x_i) \right\} \right]$$

$$= \sum_{i=1}^{N} \left[f_k(x_i) y_i - f_k(x_i) \left(a_0 f_0(x_i) + a_1 f_1(x_i) + \cdots + a_M f_M(x_i) \right) \right] = 0$$

$$a_0 \sum_{i=1}^{N} f_k(x_i) + a_1 \sum_{i=1}^{N} f_k(x_i) f_1(x_i) + \cdots + a_M \sum_{i=1}^{N} f_k(x_i) f_M(x_i) = \sum_{i=1}^{N} f_k(x_i) y_i \tag{2-37}$$

ここで式を簡潔にするため，観測データ y_i を $\mathbf{y}= [y_1, y_2, \ \ldots, y_N]^{\mathrm{T}}$，モデルパラメータ $\mathbf{a}= [a_0, a_1, \ldots, a_M]^{\mathrm{T}}$，$\mathbf{f} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ f_1(x_1) & f_1(x_2) & \cdots & f_1(x_N) \\ \vdots & \vdots & & \vdots \\ f_M(x_1) & f_M(x_2) & \cdots & f_M(x_N) \end{bmatrix}$ とベクトルと行列で表記すると式(2-37)は式(2-38)と表現でき，モデルパラメータ $\mathbf{a}$ はこの正規方程式の解として決定することができる．

$$\mathbf{f}\,\mathbf{f}^{\mathrm{T}}\mathbf{a} = \mathbf{f}\mathbf{y} \tag{2-38}$$

ここでノイズのモデル化を具体的に考察する．回帰モデル $f_k(x)$において k=1 の基底関数 f_1 が過剰適合項の場合を考え，$\mathbf{a}' = [a_0', 0, a_2', \cdots, a_M']^T$，観測データが持つ測定誤差等のノイズを表わす定数を $\mathbf{d}=[d_1, d_2, ..., d_N]^{\mathrm{T}}$ とし，このような観測データを式(2-39)で生成する．

$$\mathbf{y} = \mathbf{f}^{\mathrm{T}}\mathbf{a}' + \mathbf{d} \tag{2-39}$$

式(2-39)を式(2-38)に代入すると，$\mathbf{f}\,\mathbf{f}^{\mathrm{T}}\mathbf{a} = \mathbf{f}\,\mathbf{f}^{\mathrm{T}}\mathbf{a}' + \mathbf{f}\mathbf{d}$ となり式(2-40)を得る．

$$\mathbf{f}\,\mathbf{f}^{\mathrm{T}}(\mathbf{a} - \mathbf{a}') = \mathbf{f}\mathbf{d} \tag{2-40}$$

式(2-40)は，観測データに $d_i \neq 0$，が 1 点でもあると右辺が非零となり $\mathbf{a}-\mathbf{a}' \neq \mathbf{0}$ 即ち $a_1 \neq 0$ となることを示す．このようにモデル化に必要のない基底関数があると，観測データのノイズによりそのモデルパラメータが値を持つようになる．これが過剰適合である．観測データには常にノイズが存在するため， f_1 のような必要ない基底関数がモデル中に含まれていれば，過剰適合は常に発生していると考えられる．もしこの f_1 が x の冪乗等の急激に増加する関数であれば，観測データ $\mathbf{y}$ の外挿領域においてその項の影響が大きくなり，結果モデルの予測精度が低下する．従ってモデルパラメータの決定時に過剰適合となっている基底関数を見つけて削除することが，予測モデル構築における重要な課題の一つとなる．

2.3.2 基本的アイディア

2.3.1 節で述べたノイズ $\mathbf{d}$ を観測における「揺らぎ」に限定し，揺らぎがモデル化されて生じる過剰適合を抑制する方法の基本的アイディアを述べる．

図 2-7 の×印は $\mathbf{d}$ を「揺らぎ」と想定したシミュレーションデータ $\mathbf{y}$ を観測データとしたときの一例で，真の値 $y^0 = 0.1$, d_i $(i=1,\ldots,N)$を平均 0，標準偏差 σ の正規分布に従う乱数で生成した， $\mathbf{y} = [y^0, y^0, \ldots, y^0]^{\mathrm{T}} + [d_1, d_2, \ldots, d_N]^{\mathrm{T}}$ で， ここに $N=8$ である．

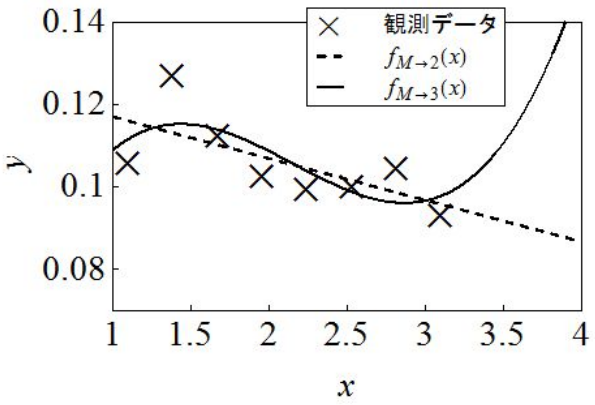


図 2-7　シミュレーションデータの多項式によるモデル化

このシミュレーションデータを多項式 $f(x)=a_0+a_1x$ (M=1) を用い LSM でモデル化すると，図 2-7 の破線で示すモデルとなり，x>3.1 での外挿は負の方向に真の値 y^0=0.1 から外れていく．モデルパラメータは $\mathbf{a}=[0.127, -0.0101]^T$ で，この過剰適合は a_1 項により生じる．次に $f(x)=a_0+a_1x+a_2x^2+a_3x^3$ (M=3)を用い LSM でモデル化すると，図の実線で示すようにデータの内挿区間で波打ったように見える曲線となる．モデルパラメータ値は $\mathbf{a}=[0.0194, 0.161, -0.0840, 0.0130]^T$ である．この曲線を観測データの範囲外 x>3.1 に外挿すると今度は正の方向に真値 y^0=0.1 から外れていく．このように揺らぎをモデル化して生ずる過剰適合により，これらのモデルでは外挿時の当てはまりが悪い．シミュレーションデータを複数回作成し同様の M=1 と M=3 のモデル化を行うと，その全てにおいて外挿時の当てはまりが悪いモデル化となる．

このような揺らぎのモデル化を，モデルパラメータに非負制約を課して推定することを考える．まず M=1 では図 2-7 のような負の傾きになるモデル化が抑制できる．次に M=3 では，上述の **a** の要素のように正負のモデルパラメータを持った基底関数項の重ね合わせで波打ったように見える揺らぎがモデル化されることが抑制できる．

ここで本研究のモデル化対象である並列計算機の処理時間のモデル化を考えると，上記で用いたような多項式を用いてモデル化する場合も多く，LSM を用いれば上記のように揺らぎがモデル化される可能性がある．また並列処理時間のモデル化でよく使われる $\log_2(x)$のようなテーラー展開で正負のモデルパラメータを持つ関数が基底関数で使われると，多項式と $\log_2(x)$の重ね合わせで $\log_2(x)$由来の観測データがモデル化される可能性が生じる．

幸いなことに並列計算機の処理時間のモデル化において観測データは時間であり非負である．そこで基底関数を演算時間，通信時間等の実際の処理に対応するようにモデル化すると，基底関数を非負に設定できる．非負制約を課すことができれば，図 2-7 で

示した例のように，揺らぎがモデル化される過剰適合を抑制し外挿時に予測を大きく外さない「予測モデル」の構築が可能と考えた．

上述の $f(x)$のモデルパラメータに非負制約を課すと各基底関数は単調増加関数となる．従って基底関数の中には残差を小さくする働きができなくなるものが出ると考えられる．その場合，残差を小さくしてモデルパラメータを決定する計算法では，その基底関数のモデルパラメータは零に漸近すると考えられる．このモデルパラメータの値を次の 2.3.4 節で述べるシンプレックス法で零値にすることにより，過剰適合を引き起こす基底関数が取り除かれたモデルを構築する．これにより並列処理の時間モデルの予測力向上を図ることが本提案の基本的アイディアである．

2.3.3 シンプレックス法によるモデルパラメータの決定

2.3.2 節で述べた非負制約を課したモデルパラメータの推定には，ラグランジュ乗数が最大になる基底ベクトルを探す非負制約最小二乗法[2-48]を適用することができる．この計算法を用いると，4.3.1 節の説明変数が 1 変数の例示のように外挿ができるモデル「予測モデル」が得られる場合がある．一方 2 変数モデル，3 変数モデルで例示するようにこの計算法ではモデル化できない場合もある．並列処理の処理時間のモデルは，2.3 節の冒頭で述べたようにウィークスケーリング等ができることが求められるので，説明変数がプロセッサ数と問題の大きさの，2 変数以上で構成されるモデルのモデルパラメータを決定できる計算法が必要となる．

そこで我々は，2 変数 3 変数のモデルパラメータが決定できる計算法として，解の候補を全て列挙して選択する，数式処理のアルゴリズムの 1 つ限量記号消去法(QE)を用いる方法[2-45]を提案してきたが，本論文では QE より多くの変数や入力データが扱え，実行可能解から解を選択する線形計画法の解法の一つシンプレックス法を用いた計算法を提案する．

シンプレックス法を適用するためには，LSM のように残差平方和を最小化するのではなく，残差 e_i ($=y_i-f(x_i)$) (i=1,…,N)の絶対値の最大値を最小化する．具体的に話しを進めるためにまず式(2-41)の連立残差方程式を考える．ここに $\mathbf{e}=[e_1, e_2, \ldots, e_N]^T$ である．

$$\mathbf{e}= \mathbf{y}-\mathbf{f}^T\mathbf{a} \quad (\mathbf{a}\geq 0) \tag{2-41}$$

残差の最大値をε(=Max $|e_i|$)とし，要素数 N のベクトルを$\boldsymbol{\varepsilon}=[\varepsilon,\ldots,\varepsilon]^T$ として，式(2-41)から式(2-42)の連立残差不等式を得る．

$$\boldsymbol{\varepsilon} \geq |\mathbf{y}-\mathbf{f}^T\mathbf{a}| \quad (\mathbf{a}\geq 0) \tag{2-42}$$

式(2-42)を解いて得られたεの最小値（e_i の最大値でもある）を$\varepsilon_{\min}$，要素数 N のベクトルを$\boldsymbol{\varepsilon}_{\min}=[\varepsilon_{\min},\ldots,\varepsilon_{\min}]^T$ と置くと，モデルパラメータは式(2-43)の連立残差不等式を解くことで決定できる．

$$\boldsymbol{\varepsilon}_{\min} \geq |\mathbf{y}-\mathbf{f}^T\mathbf{a}| \quad (\mathbf{a}\geq 0) \tag{2-43}$$

次に式(2-42)と(2-43)を線形計画法の問題として記述する．まず式(2-42)の絶対値を外し，式(2-44)の連立残差不等式とする．

$$-\boldsymbol{\varepsilon} \leq \mathbf{y}-\mathbf{f}^T\mathbf{a} \leq \boldsymbol{\varepsilon} \quad (\mathbf{a}\geq 0) \tag{2-44}$$

さらに $\mathbf{f}$ に x_i の値を代入した $\mathbf{F}$ を用いて式(2-44)を線形化し式(2-45)とする．

$$-\boldsymbol{\varepsilon} \leq \mathbf{y}-\mathbf{F}^T\mathbf{a} \leq \boldsymbol{\varepsilon} \quad (\mathbf{a}\geq 0) \tag{2-45}$$

式(2-45)を制約条件としεを最小化する線形計画法の問題として記述すると式(2-46)となり，式(2-42)の解$\varepsilon_{\min}$= Min(ε)とそれを用いた式(2-43)の解 $\mathbf{a}$ を同時に得ることができる．なお表示の簡略化のため説明変数が 1 変数の基底関数の場合で話しを進めてきた

が，x_i を代入したことにより $\mathbf{F}$ はモデルパラメータ a_k を変数とする線形基底関数の要素となるので，説明変数 x_i が $x_{1i}, x_{2i}, \cdots$ と複数となっても式(2-46)を適用できる．

$$\begin{aligned} &\text{Min} \quad \varepsilon \\ &\text{s.t.} \quad \boldsymbol{\varepsilon} + \mathbf{F}^{\mathrm{T}}\mathbf{a} \geq \mathbf{y} \\ &\qquad \boldsymbol{\varepsilon} - \mathbf{F}^{\mathrm{T}}\mathbf{a} \geq -\mathbf{y} \\ &\qquad \mathbf{a} \geq 0 \end{aligned} \tag{2-46}$$

本論文では式(2-46)をシンプレックス法で解くことを提案する．シンプレックス法を用いることにより，εを減らすモデル化に必要なモデルパラメータは基底変数として選択され，εを増やす不要なものは非基底変数となり零値を得る．この零値により，小さな値の項がモデル化に必要か否かを検討する手間が省略できる．またこの零値により，予測に不要な基底関数の一部が削除されたモデルが自動的に構築できる．更に，シンプレックス法を有理数演算することを提案する．有理数演算を用いると厳密な計算結果が得られ，計算誤差による誤ったモデルパラメータ選択が回避できる．

図 2-8 に線形計画法の関数を適用して式(2-46)を解きモデルパラメータを推定する手順を示す．始めにモデル化に用いる観測データとモデル式を入力する．2）番目に入力した観測データを有理数に変換する．3)番目に観測データをモデル式に代入して，$\mathbf{F}^{\mathrm{T}}$ の要素 m_{ik} を求める．4)番目に目的関数を設定する．最小化するのはεなので1，その他のモデルパラメータは最小化対象外なので M+1 個の 0 をセットする．5)番目に LP を適用するための制約条件をセットする．m_{ik}', m_{ik}''の要素の先頭に加えた 1 は式(2-46)の制約式の$\boldsymbol{\varepsilon}$に対応する．6)番目にシンプレックス法を計算法とする線形計画法の関数を実行する．この関数にはシンプレックス法を有理数演算で実行できるものが必要となる．最後に，結果は有理数で得られるので，他の結果と比較するため浮動小数点に変換して出

力する．なお 4), 5), 6)のインプットフォームはツール依存なので，4.3 節で用いる Mathematica の関数 LinearProgramming のインプットフォームで例示した．

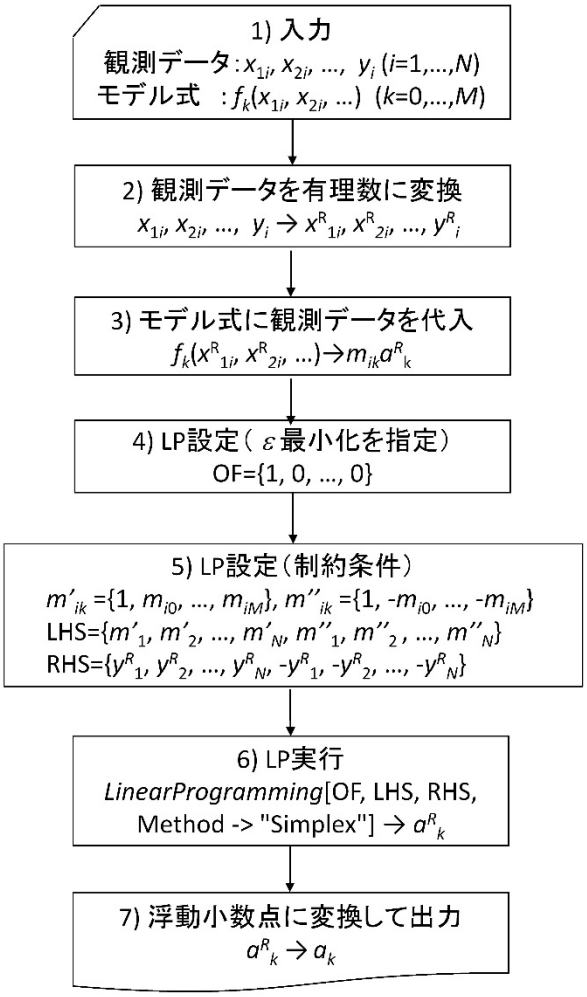


図 2-8　モデルパラメータ推定手順

参考文献

2-1) http://www.vampir.eu/

2-2) D.J. Kuck, The Structure of Computers and Computations, vol. 1, John Wiley & Sons, (1978) 33.

2-3) A.Y. Grama, A. Gupta, V. Kumar, Isoefficiency: measuring the scalability of parallel algorithms and architectures, IEEE Parallel Distrib. Technol. 1 (3) (1993) 12–21.

2-4) 古市実裕, 永松礼夫, 出口光一郎, スケーラビリティに基づく高並列計算機のパフォーマンス予測, 情報処理学会研究報告 HPC057-11, (1995) 61–66.

2-5) X.H. Sun, D.T. Rover, Scalability of parallel algorithm-machine combinations, IEEE Trans. Parallel Distrib. Syst. 5 (6) (1994) 599–613.

2-6) A. Petitet, R. C. Whaley, J. Dongarra and A. Cleary: HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers.
http://www.netlib.org/benchmark/hpl/

2-7) V.E. Bazterra, M. Cuma, M.B. Ferraro, J.C. Facelli, A general framework to understand parallel performance in heterogeneous clusters: analysis of a new adaptive parallel genetic algorithm, J. Parallel Distrib. Comput. 65 (2005) 48–57.

2-8) A.H. Karp, H.P. Flatt, Measuring parallel processor performance, Commun. ACM 33 (5) (1990) 539–543.

2-9) X. Zhang, Y. Yan, K. He, Latency metric: an experimental method for measuring and evaluating parallel program and architecture scalability, J. Parallel
Distrib. Comput. 22 (3) (1994) 392–410.

2-10) G. Amdahl, Validity of the single-processor approach to achieving large scale computing capabilities, in: AFIPS Conference Proceedings, vol. 30 (1967) 483–485.

2-11) J.L. Gustafson, Re-evaluating Amdahl’s Law, CACM 31 (5) (1988) 532–533.

2-12) H.P. Flatt, K. Kennedy, Performance of parallel processors, Parallel Comput. 12 (1989) 1–20.

2-13) M.E. Crovella, T.J. LeBlanc, Parallel performance prediction using lost cycle analysis, in: Proceedings of Supercomputing ’94 (1994) 600–610.

2-14) S. Prakash, R.L. Bagrodia, MPI-SIM: using parallel simulation to evaluate MPI programs, in: Proceedings of the 30th Conference on Winter Simulation, (1998) 467–474.

2-15) R.W. Hockney, C.R. Jesshope, Parallel Computers, Adam Hilger, Bristol (1981).

2-16) R.W. Hockney, Parameterization of computer performance, Parallel Comput. 5 (1987) 97–103.

2-17) D. Müller-Wichards, W. Rönsch, Scalability of algorithms: an analytic approach, Parallel Comput. 21 (1995) 937–952.

2-18) V. Kumar, V.N. Rao, Parallel depth first search, Part II: Analysis, J. Parallel Program. 16 (6) (1987) 501–519.

2-19) 折居茂夫, 数値計算のための並列計算機性能評価方法, 情報処理学会論文誌, 39, (3) (1998) 529–541.

2-20) S. Orii, Metrics for evaluation of parallel efficiency toward highly parallel processing, Parallel Comput. 36 (2010) 16–25.

2-21) B. L. Buzbee, Plasma simulation and fusion calculation, LA–UR–83–1633, Los Alamos National Laboratry (1983).

2-22) L. M. Adams, and T. W. Crockett, Modeling algorithm execution time on processor arrays, Computer, IEEE, 17 (1984) 38–43.

2-23) C. Moler, Matrix computation on distributed memory multiprocessors, in M. Heath (ed.) Hypercube Multiprocessors, SIAM (1986) 181–195.

2-24) M. J. Clement, M.J. Quinn, Analytical performance prediction on multicomputers, in: Proceedings of Supercomputing ’93 (1993) 886–894.

2-25) A. Thomasian, P. F. Bay, Analytic queueing network models for parallel processing of task systems, IEEE Tras. on Computers, C-35 (12) (1986) 1045–1054.

2-26) V. W. Mak, S. F. Lundstrom, Predicting performance of parallel computations, IEEE Tras. Parallel and Distrib. Systems, 1 (3), (1990) 257–270.

2-27) D. Menasce, S. H. Hoh, S. K. Tripathi, A methodology for performance prediction of massively parallel applications, in: Proceedings of IEEE Symposium on Parallel and Distributed Processing, (1993) 250–257.

2-28) Z. Xu, X. Zhang, L. Sun, Semi-empirical multiprocessor performance predictions, J. Parallel Distrib. Comput. 39 (1996) 14–28.

2-29) X. -H. Sun, J. Zhu, Performance prediction: a case study using a scalable shared-virtual-memory machine, IEEE Paralel & Distributed Technology, Winter (1996) 36–49.

2-30) 折居茂夫, レベル 1･2 並列ベンチマーク仕様及びそれに基づくスカラ並列計算機 SP2 のベンチマークテスト，JAERI-Data/Code 98–020 (1998).

2-31) Y. Kishimoto, S. Ichikawa, Optimizing the configuration of a heterogeneous cluster with multiprocessing and execution-time estimation, Parallel Comput. 31 (2005) 691–710.

2-32) Z. Xu, K. Hwang, Early prediction of MPP performance: The SP2, T3D, and Paragon experiences, Parallel Comput. 22 (1996) 917–942.

2-33) M. J. Clement, M. J. Quinn, Automated performance prediction for scalable parallel computing, Parallel Comput. 23 (1997) 1405–1420.

2-34) M. Mathis, D. J. Kerbyson, A. Hoisie, A performance model of non-deterministic particle transport on large-scale systems, Future Generation Comput. Systems 22 (2006), 324–335.

2-35) M. A. Driscoll, W. R. Daasch, Accurate prodictions of parallel program execution time, J.

Parallel Distrib. Comput. 25 (1995) 16–30.

2-36) B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. de Supinski, M. Schulz, A regression-based approach to scalability prediction, Twenty Second International Conference on Supercomputing (ICS 2008), pp. 368–377 (2008).

2-37) J. T. Robinson, Some analysis techniques for asynchronous multiprocessor algorithms, IEEE Trans. Soft. Eng., SE-5 (1979) 24–31.

2-38) S. Madala, J. B. Sinclair, Performance of synchronous parallel algorithms with regular structures, IEEE Trans. Paralel & Distrib. Systems 2 (1991), 105–116.

2-39) V. D. Agrawal, S. T. Chakradhar, Performance analysis of synchronized iterative algorithms on multiprocessor systems, IEEE Trans. Paralel & Distrib. Systems, 3 (1992), 739–746.

2-40) W. M. Lin, Performance modeling and analysis of correlated parallel computations, Parallel Comput, 34 (2008) 521–538.

2-41) 折居茂夫, 高並列処理におけるスケーラビリティ評価方法（Ⅱ）, 情報処理学会研究報告, HPC-126 (48) (2010).

2-42) 折居茂夫, 時間モデルを用いた並列処理の性能評価 － 並列化部に隠れた並列オーバーヘッド －, HPC-130 (1) (2011).

2-43) K. Raghavachar, K. Mahinthakumar, P. Worley, E. Zechman, R. Ranjithan, Parallel performance modeling using a genetic programming-based error correction procedure, SIMULATION, 83 (7) (2007) 515–527.

2-44) 折居茂夫, 穴井宏和, 時間モデルを用いた並列性能予測の誤差を検討する一方法, 情報処理学会研究報告, HPC-133 (4) (2012).

2-45) 折居茂夫, 限量記号消去法による時間モデルパラメータの決定, 情報処理学会研究報告, HPC-134 (11) (2012).

2-46) 折居茂夫, 山本義郎, 限量記号消去法を用いた回帰モデルの予測力向上, 情報処

理学会研究報告，情報処理学会研究報告, HPC-139 (1) (2013).

2-47）折居茂夫，山本義郎，シンプレックス法を用いた非負のモデルパラメータを持つ並列処理時間モデルの予測力向上，情報処理学会論文誌 56, (6) (2015) 1481–1495.

2-48) C.L. Lawson and R.J. Hanson, Solving Least Squares Problems, Prentice-Hall, Chapter 23 (1974).

3章　並列処理の性能評価の研究（実験１）

2 章で提案した並列性能評価指標の有効性を確認するため，数値シミュレーションプログラムを並列計算機で実行した処理時間を用いて並列性能評価指標を決定し，性能を評価した．3.1 節では 2.1.1，2.1.2，2.1.3 節で提案した性能評価指標を用いた並列処理の性能評価について確認する [3-1)]．3.2 節では 2.1.4 節で提案したループレベルの性能評価指標を用いた並列計算機の性能評価について確認する [3-2)]．

3.1　提案した性能評価指標による並列処理の性能評価

結晶構造解析プログラムを並列計算機 NEC SX-4 で実行した処理時間と，分子動力学コードを並列計算機 IBM SP2 で実行した処理時間を用いて決定し，これらの並列処理の性能を評価した [3-1)]．

3.1.1　ロードバランスと並列阻害指標を用いた性能評価

(1)　共有メモリ計算機上でマイクロタスクプログラムを実行した場合

マイクロタスクプログラムと SX-4 の共有メモリモードの組み合わせによる並列処理の性能評価について述べる．SX-4 は共有メモリと分散メモリを持つベクトル並列計算機である．CPU の論理性能は 2Gflops で，使用した計算機の各ノードは 16GB の共有メモリを持ち，X 線結晶構造解析プログラムを実行した．プログラムの主要ホットスポットであるフーリエ変換はマイクロタスクにより並列化されている．一方逐次処理部はプライマリプロセッサで実行される [3-3)]．

並列効率算出に使うため測定した各プロセッサの経過時間 τ_i と並列処理部の時間 γ_i の処理時間を
表 3-1 に示す．全ての γ_i がプロセッサ番号 1 の並列処理部の時間 γ_1 と同じと仮定して，式(2-10), (2-12), (2-13)と $\tau_i = \gamma_i + \chi_i$ の関係から，2 章で提案した性能評価指標 ε_p，R_b，R_{imp} を計算することができる．

表 3-1　SX-2 における X 線結晶構造解析プログラムの処理時間 [3-3)]

$p = 1$														
i	1													
τ_i	834.8													
γ_i	605.6													
$p = 14$														
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14
τ_i	273.5	57.4	57.4	57.3	57.3	57.2	57.2	57.1	57.1	57.0	56.9	56.9	56.8	56.6
γ_i	44.3	–	–	–	–	–	–	–	–	–	–	–	–	–

図 3-1 (a)はこれらの指標の値で，p=1 では ε_p =0.725,, R_b =1, R_{imp} =0.275 となる．この場合 R_b =1 であるので ε_p は R_{imp} の余数となる．これは並列効率が並列阻害要因によって 0.275 抑制されることを意味する．式(2-17)を用いて ε_p から加速限界 A_{LT} を求めると，その値は 3.34 と非常に低い．図 3-1 (b)に A_{LT} を示す．性能評価指標 R_b により p=14 においてロードインバランスがあることが検知できる．その値は 0.265 であるので，もし R_{imp} =0 と並列阻害要因が無くても，新並列効率指標は 0.265 という低い値となり結果 A_{LT} =1.36 となってしまうことがわかる．そこで p=14 において R_b を 0.265 から 1 に改善できるとすると， R_{imp} =0.389 から ε_p =0.611 となり加速限界は A_{LT} =2.57 に改善する．

結果, ロードインバランスと並列阻害要因のどちらも並列効率に大きな影響を及ぼすことがわかる．このように提案した新性能評価指標により，並列オーバーヘッドの並列効率に及ぼす影響を定量的に捉えることができることが確認できた.

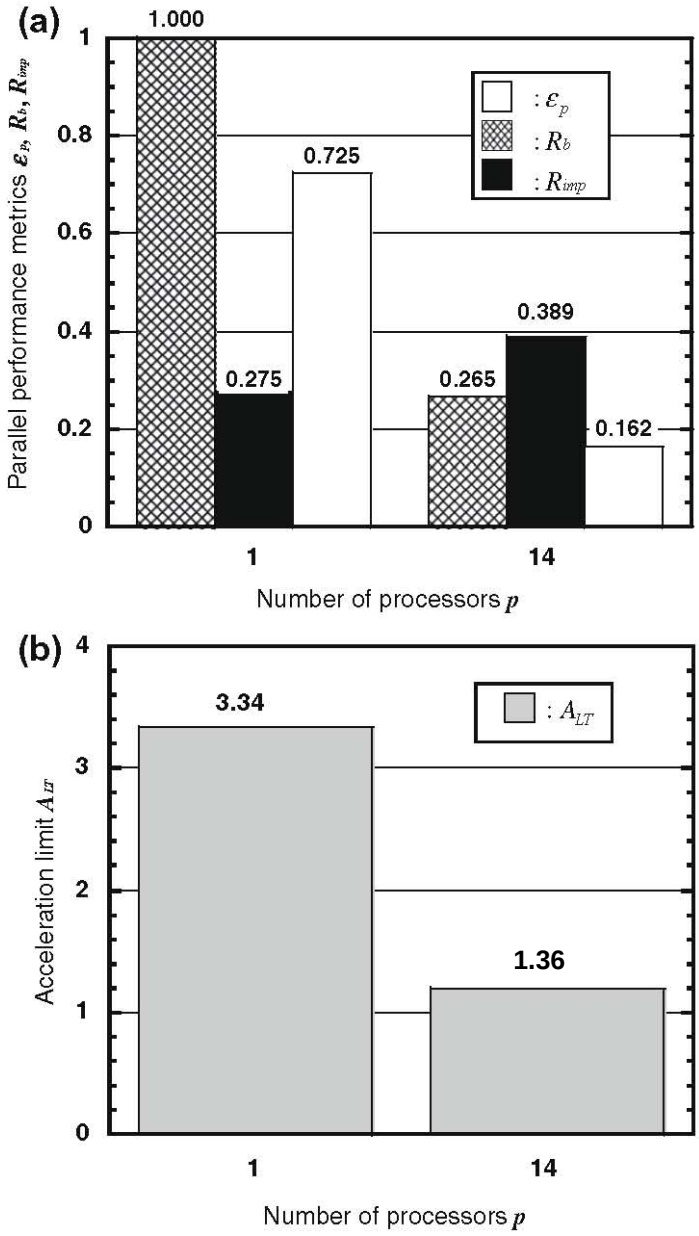


図 3-1　SX-2 での X 線結晶構造解析プログラムの新性能評価指標 ε_p, R_b, R_{imp} と加速率

(2)　分散メモリ計算機上でメッセージパッシングプログラムを実行した場合

新並列性能評価指標が分散メモリ計算機上でメッセージパッシングプログラムを実行した場合の並列性能評価においても有効であることを，2 次元分子動力学コード（以後，MD コード）[3-4]を SP2 で実行して確認した [3-5]．SP2 は分散メモリ方式を採用したスカラ並列計算機である．1 プロセッサの最大論理性能は 266 Mflops，各ノードは 128 MB

のメモリである．ノードは 4 段の多重結合スイッチで結合され，双方向 40 MB/sec のデータ転送能力を持つ.「粒子ーメッシュアルゴリズム」で構築された MD コードは, SPMD プログラミングと MPI を用いて粒子分割法で並列化されている．また MD コードに含まれる逐次実行部の計算は，(1)の事例のように特定な 1 プロセッサで実行するのではなく，全てのプロセッサが同じ計算を行う「冗長実行」で行った.

表 3-2 と図 3-2 (a)は各プロセッサで測定した処理時間を基に計算した新性能評価指標で，p=1 の場合は ε_p =0.952，R_b =1 ，R_{imp} =0.048 である．この ε_p から加速限界 A_{LT} =20.6 が得らる．これを図 3-2 (b)に示す．一方 p=10 の場合， ε_p =0.47，従って A_{LT} =1.89 となる．この原因は，性能評価指標が R_b =0.999， R_{imp} =0.528 なので，並列阻害要因にあることが確認できる．これらのことから，並列効率を考えるなら，この MD コードと SP2 の組み合わせでは数十プロセッサ以下の低並列処理が適当であることがわかる.

表 3-2　SP2 における分子動力学コードの処理時間

$p = 1$										
i	1									
τ_i	2591.4									
γ_i	2466.0									
$p = 10$										
i	1	2	3	4	5	6	7	8	9	10
τ_i	565.7	565.0	565.0	565.0	565.0	565.0	565.0	565.0	565.0	565.0
γ_i	265.8	266.6	267.2	266.6	266.9	267.2	266.8	266.9	267.4	266.8

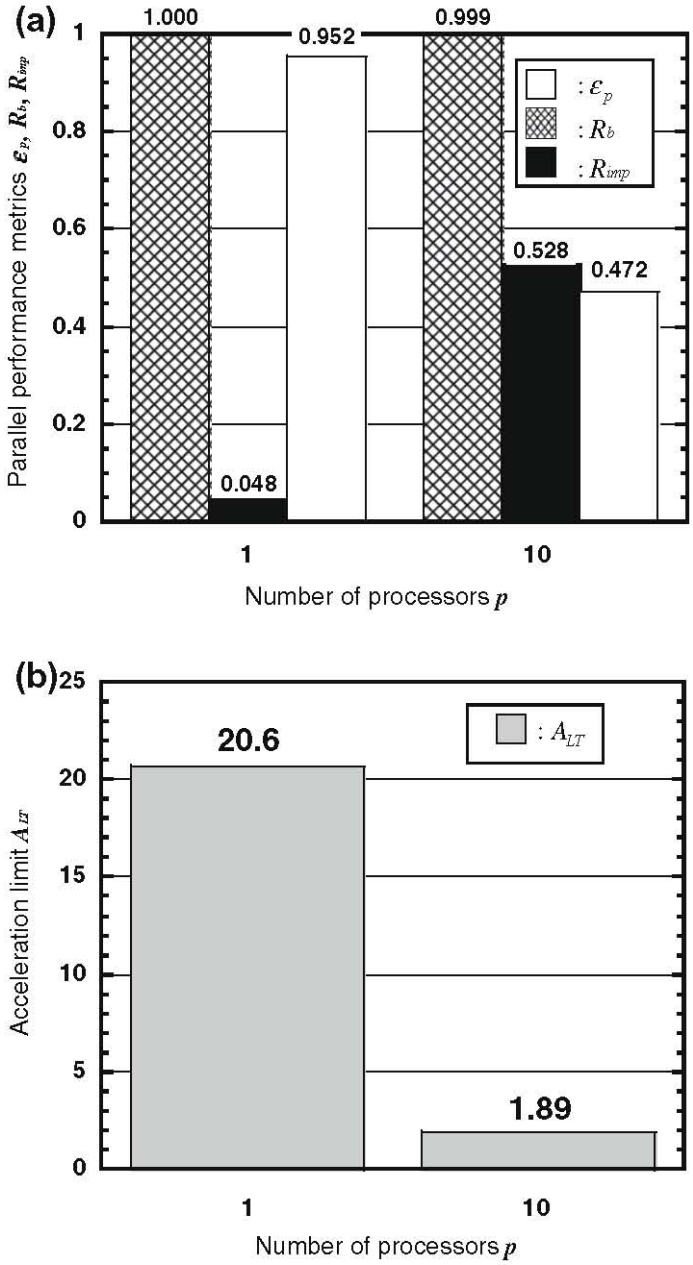
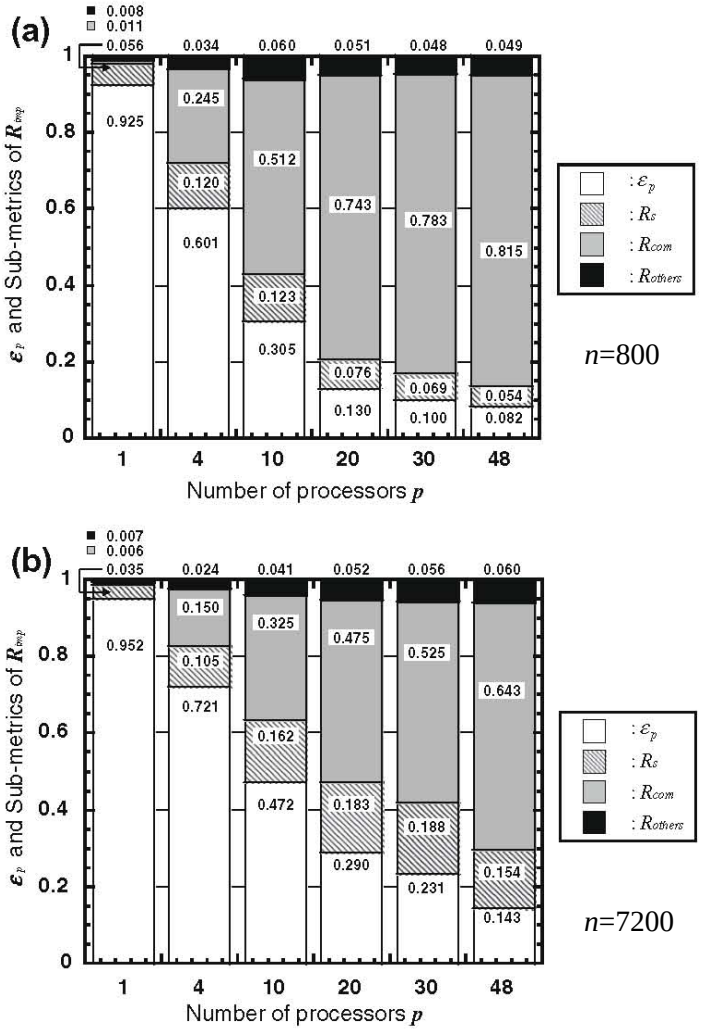


図 3-2　　SP2 での分子動力学コードの新性能評価指標 ε_p, R_b, R_{imp} と加速率

3.1.2　サブ性能評価指標による詳細な性能評価

提案した式(2-15)で示したサブ性能評価指標を用いると，より詳細な性能評価ができることを確認する． 3.1.1 節の(2)のケースでは，R_b ~1 なのでサブ性能評価指標の和の余数は並列効率指標となる. そこでプロセッサ数に対するサブ性能評価指標の割合を図

3-3 の(a)，(b)，(c)に各々問題の大きさ n=800，7200，98600 に対して表示した．ここに縦軸の最大値は 1 である．プロセッサ数や問題の大きさが変化すると処理時間は変化するが，並列効率指標の最大値は常に 1 であるので，このような図を描いて効率を評価することが可能となる．



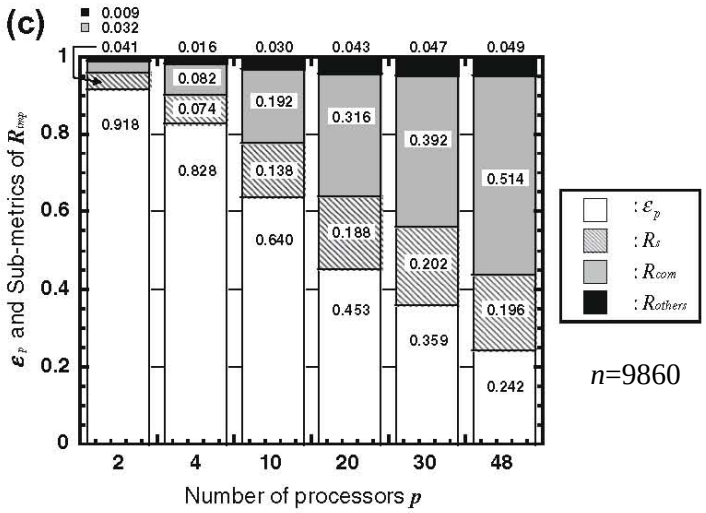


図 3-3　新並列効率指標 ε_{p} とその阻害要因 R_{s}, R_{com}, R_{other} の定量的関係

今 R_b ~1 なので，式(2-15)から $R_s = 1 - \gamma_i / \tau_i - \chi_i^{com} / \tau_i$ となる．従ってもし χ_i^s が定数で χ_i^{com} =0 なら，γ_i が p の増加と共に減少するので $R_s(p \to \infty)$ は 1 となるはずである．しかし図 3-3 の(a)，(b)，(c)は R_s が 1 にならず，p=4 以上で飽和傾向に，p =48 では減少傾向にあることを示す．この現象は，図で p の増加と共に χ_i^{com} が増加しているので，理解できる．

サブ性能評価指標 R_{com} はプロセッサ数の増加と共に χ_i^{com} の割合が増加することを検知する．図 3-3 の(a)，(b)，(c)では p=10 の R_{com} は p=4 と比べると約 2 倍大きくなる．図 3-3 の(a)と(b)は，プロセッサ数を増加しても R_s が小さな値であるにもかかわらず ε_p が減少する原因が χ_i^{com} の増加にあることを示す．図 3-3 (c)においても，p=4 と 10 の間で R_{com} は 1.65 倍になり，同じような現象が生じていることがわかる．このように，サブ並列性能指標を解析すると，この MD コードと SP2 の組み合わせでは数十プロセッサという低並列処理で行う必要がある原因が，問題の大きさを大きくしても χ_i^{com} が大きくなり ε_p が小さくなってしまうことにあることを突き止めることができる．

図 3-4 は問題の大きさを変えたときの処理時間を示す．この図のように処理時間からは異なった問題の大きさの定量的な性能比較はできない．これに対し，提案した並列効率指標とサブ指標は，異なった問題の大きさの性能の比較を可能し，異なった問題の大きさ間の性能比較を可能にする．

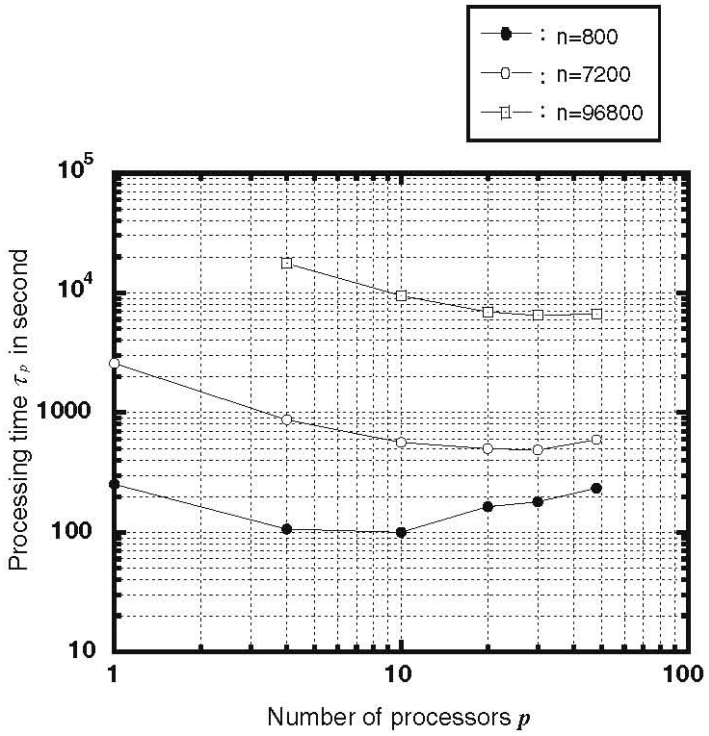


図 3-4　問題の大きさを変えたときの処理時間

3.1.3　議論

ロードバランス指標 R_b は文献 3-6)の実行プロセッサ数をプロセッサ数で除したものと同じである．従って指標 R_b，R_{imp}, and ε_p を関係付ける式(2-14)，(2-15)，(2-16)は，ヘテロジニアスな並列システムの処理にも適用できると考える．ヘテロジニアスな並列システムにおいてこれらの指標とサブ指標の有効性を確認することは今後の研究課題である．

図 3-2 (a)の R_b はロードインバランスが無い事を示すが，分子動力学コードの総和計

算にはロードインバランスの可能性が存在する．これは総和計算が同期をとるため R_b によって検知できないためである．もし総和計算中に生じる待ち時間を計測できれば，サブ指標 R_j の要因の一つにこの待ち時間を加えることによりその性能に及ぼす影響を評価することができる．

逐次実行時間を測定することが可能ならば，式(2-11)で定義した R_{CPU} を使って MD コードの並列処理部の CPU 性能の変化を検出できる．図 3-5 は n=7200 に対するプロセッサ数に対する R_{CPU} 値である．プロセッサ数を増したとき R_{CPU} は p=1 で 0.95，p=10 で 1.03，p=48 で 1.6 とプロセッサ数と共に増加する．これは並列化部の時間が p=1 で逐次処理時間の 95%であるのに対し，p=48 では逐次処理時間の 160%に増加することを意味する．図 3-5 に従来の並列効率 E_p と新並列効率指標 ε_p の違いを示す．この差は R_{CPU} により検知された CPU 性能の変化により生じる．この差が小さいのはプロセッサ数と共に並列効率が減少し p=48 では両者が約 0.1 と値が小さいためであるが，もし ε_p が高い状態で CPU 性能即ち R_{CPU} が大きくなると，ε_p は E_p より大きな値になりその差も大きくなる．一方高並列処理では R_{CPU} を決定できないので，並列処理の性能評価にはなんらかの手段で CPU の性能の変動の考慮が必要と考える．

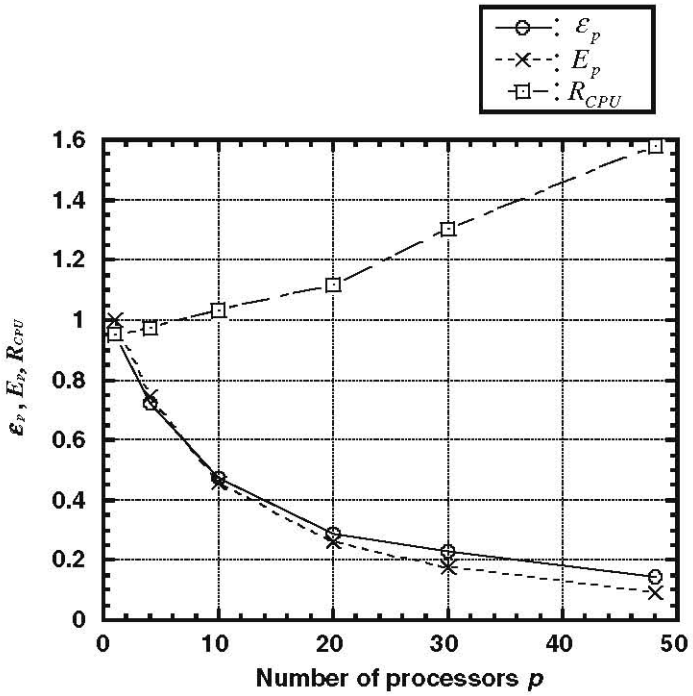


図 3-5　プロセッサ数により変化する ε_p と E_p の差とそれを説明する R_{CPU} 値

3.2　提案した性能評価指標による並列計算機の性能評価

3.1 節では並列処理の性能評価を行ったが，計算機が設計通りの性能で稼働しているかを評価する，並列計算機の性能評価もまた重要である．計算機はアーキテクチャの違い等により得手不得手の処理の種類が異なるためである．そこで 2.1.4 節で述べたループレベルの性能を評価する指標 a_u と e_c を用い，プラズマ粒子コードを用いて並列計算機の演算と通信を評価した事例を示す [3-2)].

3.2.1　プラズマ粒子コード

上記で記したように計算機の違いにより得手不得手の処理の種類が異なるため，どのような処理に対して並列計算機の性能評価を行うかを特定しておくことが必要となる．そこでまず，性能評価に用いたプラズマ粒子コードの処理について述べる．図 3-6 はこのコードのフローチャートで，計算内容は，粒子電荷のメッシュへの分配，場の計算，粒子加速，及び t_{obs} 毎の観測から成る．

これらの処理はデータマッピングの観点から，次の[Ⅰ]から[Ⅳ]の 4 つの計算に分類することができる．[Ⅰ]は電荷等の粒子データ B を粒子の位置 IX を基にしてメッシュ上 F に集める粒子分配と観測，[Ⅱ]は場 F の計算，[Ⅲ]は場の量 F を粒子の位置 IX に内挿する粒子加速，[Ⅳ]は粒子加速及び観測である．

[Ⅰ]粒子からメッシュへのデータマッピングを含む計算

$$F(IX(N))=F(IX(N))+B(N) \qquad (N=1, N_{max})$$

[Ⅱ]メッシュ上における計算

$$F(I)=FD(I)+A(I) \qquad (I=1, I_{max})$$

[Ⅲ]メッシュから粒子へのデータマッピングを含む計算

$$P(N)=F(IX(N))+B(N) \qquad (N=1, N_{max})$$

[Ⅳ]粒子上における計算

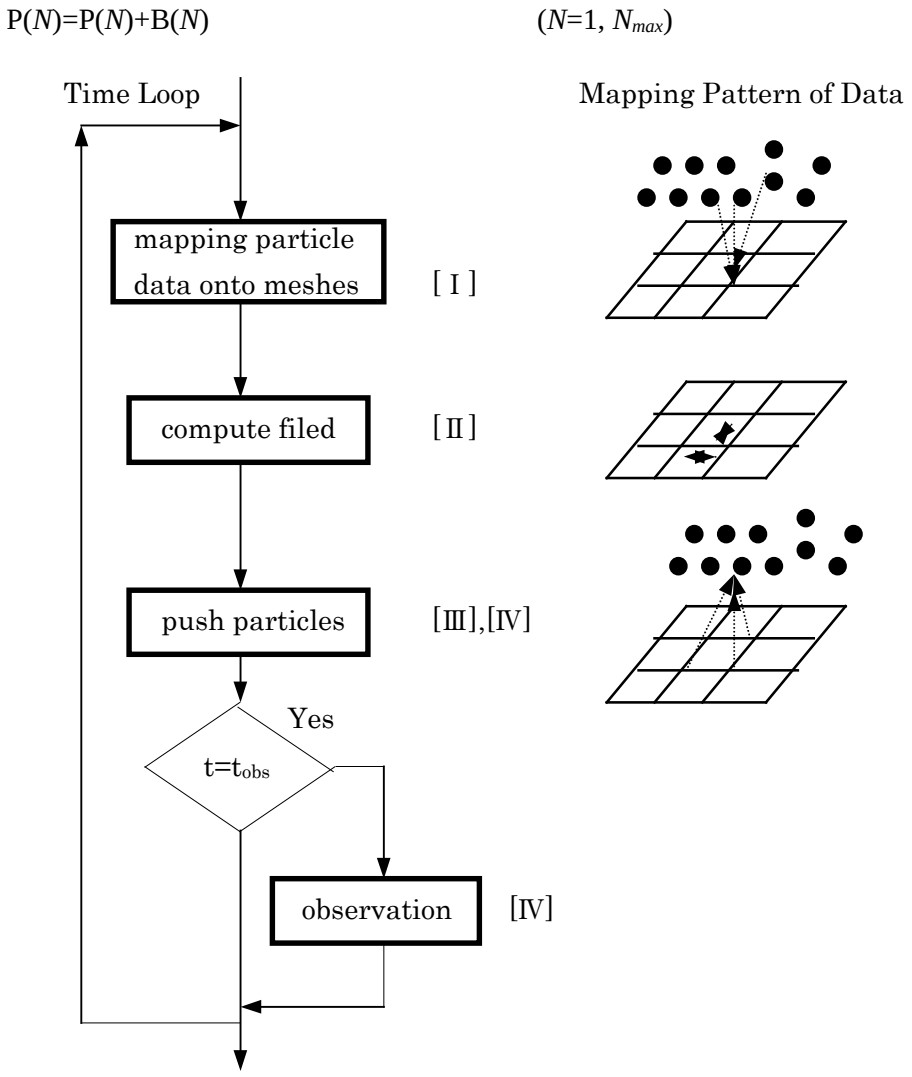


図 3-6　プラズマ粒子コードのフローチャートとデータマッピングパターン

ここに，F，FD と A はメッシュデータ，P と B は粒子データ，IX は粒子の位置，I_{max} はメッシュ数，N_{max} は粒子数を表わす．核融合プラズマ粒子シミュレーションでは，セル中の粒子数 $N_{cell}(\equiv N_{max}/I_{max})$が 10～100 を想定できる．

並列計算法に粒子分割法を用いたプラズマ粒子コードは，粒子数を各プロセッサに均等に分割するため，各プロセッサの計算時間がほぼ同じになり，優れた負荷分散を実現できる．プロセッサ数を p とし粒子数を N_{max} とすれば，[I]，[Ⅲ]，[Ⅳ]の計算を 1～

N_{max}/p，　N_{max}/p+1～ 2×N_{max}/p，・・・と各プロセッサに分割して並列計算できる．この粒子分割法は，例えば Fortran のように記述するのであれば do 1, N_{max}/p，do N_{max}/p+1, 2*N_{max}/p,…と，原則的にループの計算範囲のみを変更して各プロセッサに負荷分散すれば実現できるので，プログラミング的にも非常に容易である．

図 3-7 に本論文で述べる粒子分割法による並列計算の処理の流れを示す．並列計算は粒子計算部[Ⅰ]，[Ⅲ]，[Ⅳ]（網がけした部分）で行う．通信は，粒子分配の直後のプロセッサ間に跨がる総和計算と，観測のエネルギーの総和計算にのみに必要となる．

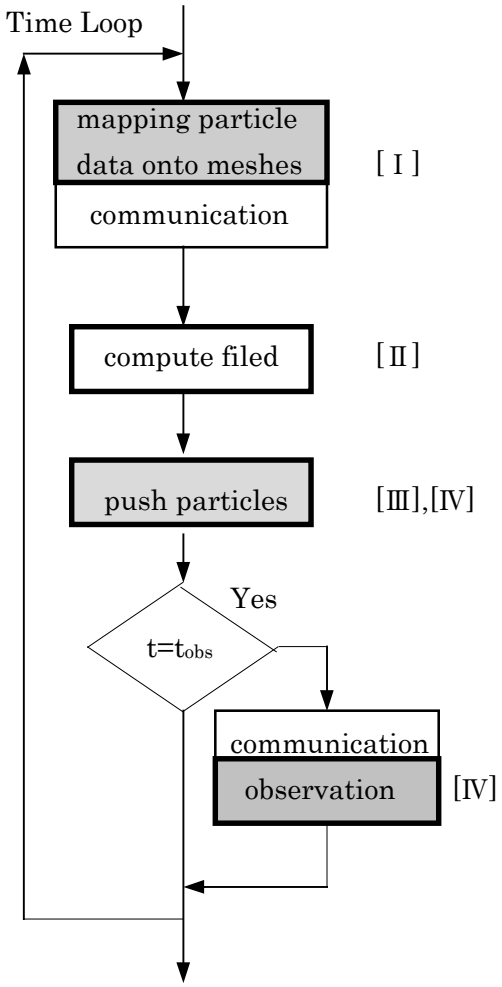


図 3-7　粒子分割法で並列化したプラズマ粒子コードのフローチャート

3.2.2　ループレベル性能評価指標 a_u，e_c の導入

一般に並列性能を阻害する主な要因は，通信時間，並列化率(=(並列計算可能な部分の実行時間／全実行時間)$_{PE=1}$)，負荷分散である．このうち負荷分散は，粒子分割法を採用したため主な要因から除外できる．粒子計算[Ⅲ]，[Ⅳ]の並列化率は 100%である．また阿部－西原の考案したベクトル計算法 [3-7), 3-8)]（以後，阿部－西原の方法と呼ぶ）を[Ⅰ]に適用した場合，粒子計算[Ⅰ]の並列化率も 100%である．従って，粒子分割法に阿部－西原のベクトル計算法を適用したプラズマ粒子コードの並列性能を決定する最大の要因は，通信となる．そこで通信の全粒子計算時間に占める影響を式(3-1)のように定義して議論する．

$$(\text{粒子計算時間})_{P=1}/((\text{粒子計算時間})_{P=1}+(\text{通信時間})_P)=1/(1+\gamma_p) \tag{3-1}$$

ここに，γ_p =(通信時間)$_P$ / (粒子計算時間)$_{P=1}$ である．粒子計算時間は，p=1 の値を基準として用いることにする．また通信時間は，プロセッサ数に依存し，通信方式や通信アルゴリズム等により変化するため，p の関数として記述した．

ここで(通信時間)$_P$ を $A(p)\cdot I_{max}/(e_c\cdot r_c/8)$，(粒子計算時間)$_{P=1}$ を $B(1)\cdot N_{max}/(a_u\cdot r_a)$とすると，$\gamma_p$ は式(3-2)のように 4 つの比の積であらわすことができる．

$$\gamma_p \equiv \frac{A(p)}{B(1)}\cdot\frac{a_u}{e_c}\cdot\frac{r_a}{r_c/8}\cdot\frac{1}{N_{cell}} \tag{3-2}$$

ここに，$A(p)$は図 3-7 の[Ⅰ]直後の通信量で，電子とイオンの電荷と電流等の場の種類と通信方式で決まる．ここに r_c(MB/s)と e_c は式(2-25)の通信の最大性能とその効率で，$r_c/8$ は 1 秒間当たりに通信できる要素数である．$B(1)$は p=1 の図 3-7 の[Ⅰ]，[Ⅲ]，[Ⅳ]のループ中の四則演算数である．r_a(Gflops)と a_u は式(2-18)の四則演算の最大性能とその効率である．上記に示したように $N_{cell}=N_{max}/I_{max}$ である．γ_p は N_{cell} に反比例することから，N_{cell} が大きい計算ほど通信時間の割合が減少することがわかる．プラズマ核融合の数値実験では通常 N_{cell}>10 である．

高い並列効果を得るためには式(3-1)の値を 1 に近づけることが必要で，そのためには γ_p の値が低くなることが要求される．しかし式(3-2)は，四則演算の効率 a_u が低い場合にも γ_p の値が低くなることを示す．この場合，計算がプロセッサの最大 flops 性能に見合わない低い性能で実行されると相対的に通信時間の割合が減少する結果，スケーラビリティを見るとリニアとなり，一見並列性能が良く見える．この現象を識別することは，ただプロセッサ数を変えて計算時間を測定しただけではできず，並列性能の原因である a_u と e_c を導入することにより初めて識別することができる．a_u と e_c は，使用した計算法がベクトル計算できない場合や通信競合を起こす場合等の本質的な問題のため低い場合もあるが，少しのコーディングの変更でコンパイラの最適化方法が変わり，大きく変わる場合もある．

ところで図 3-7 の観測[IV]では，N_{max} に比例した通信量が発生する．しかし全体の時間積分回数に対する t_{obs} 毎の観測は頻度として低く，通常無視することができる．また，場の計算[Ⅱ]は解法により並列計算法が異なる．本論文で述べる gyro3d[3-9]は FFT を用いている．FFT は並列化計算できることが知られている[3-10]．また Bi-CG 法が用いられている場合もあり[3-11]，粒子分割法とは独立に議論できるので，本論文では触れない．

3.2.3 ループレベル性能評価指標 a_u，e_c による並列計算機の性能評価

図 3-8 に拡張された阿部－西原の方法を用いた粒子分割法で gyro3d コードを並列化し，日本原子力研究所中目黒地区に設置されたベクトル並列計算機 VPP300 と SX-4 の処理時間を実測した結果を示す．VPP300 は，ピーク性能 2.2Gflops のプロセッサを，プロセッサ間通信性能 570MB/s のクロスバスイッチで結合した，15 プロセッサのシステムである．SX-4 は，ピーク性能 2Gflops のプロセッサを，ノード内を共有結合で，ノード間をクロスバスイッチで結合し，メモリ間アクセス性能 8GB/s の 6 プロセッサ（=3 ノード･2(プロセッサ/ノード)）システムである．問題の規模は，N_{max}=128×128×64，

I_{max}=32×32×16 を用いた．N_{cell}=64 である．図中 $\tau(p)$はプロセッサ数 p のときの実測値をあらわす．式(2-8)の並列効率 E_p を用いると，図はプロセッサ数が 5 台位の規模では E_p =80％以上で，10 台では E_p=約 70％の効率で並列性能が得られることを示す．測定範囲での VPP300 と SX-4 の並列効率はほぼ同じプロセッサ依存を示す．

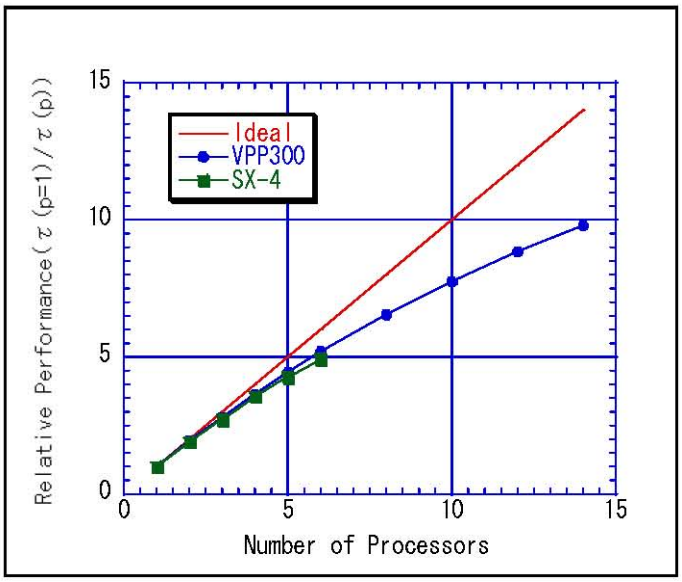


図 3-8　並列化した阿部－西口ベクトル化アルゴリズムを用いた gyro3d の並列性能

尚，VPP300 のコンパイラは Fortran90/VP (L97121)，コンパイルは frt -Sw -Ad -Ne,300 -Ei -I/usr/lang/mpi/include -Wl, -J, -P -L/usr/lang/mpi/lib -lmpi -lmp -lelf -px で行った．SX-4 のコンパイラは f90 compiler Rev.093，コンパイルは f90 -P multi -Wf”-pvctl loopcnt=1048576” -c -I/usr/include -lmpi で行った．

図 3-9 に VPP300 の粒子計算時間（図 3-7 の[Ⅰ]，[Ⅲ]，[Ⅳ]の測定値の合計）に関するスケーラビリティを示す．GSUM を含まない場合，$\tau_{particle}(p=1)/\tau_{particle}(p)$は，ほぼ理想値と同じである．GSUM を含む場合，$(\tau_{particle}(p=1) + GSUM) / (\tau_{particle}(p) + GSUM)$は，プロセッサの増加とともに幾分理想値からずれるものの，粒子分割法とベクトル並列計算機の組み合わせが有効であることが確認できる．

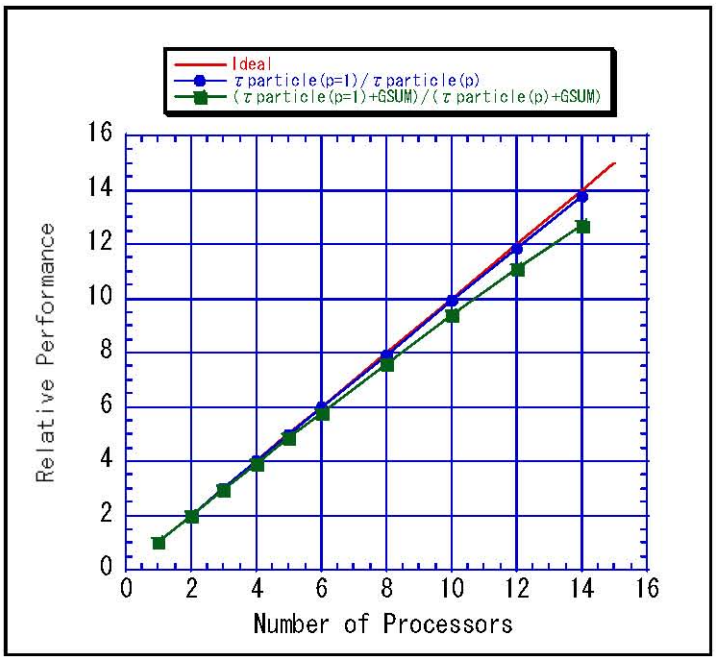


図 3-9　粒子計算全体の並列性能

図 3-9 のように粒子分割法とベクトル並列計算機の組み合わせが有効であるか否かは，式(3-2)の γ_p を評価することにより知ることができる．

まず四則演算の効率 a_u と通信の効率 e_c は，各々式(3-3)，(3-4)から計算する．

$$a_u = \frac{B(1)\cdot I_s \cdot N_{\max}}{r_a \cdot \tau_{partilce}} \qquad (p=1) \qquad (3\text{-}3)$$

$$e_c = \frac{A(p)\cdot I_{\max} \cdot I_s}{r_c \cdot \left(GSUM(p) - GSUM(1)\right)} \qquad (p>1) \qquad (3\text{-}4)$$

ここに $B(1)$=1248，I_s は時間ループの繰り返し回数，$\tau_{particle}$ (p=1)は 1PE で実行したときの実測粒子計算時間である．r_a は Mflops であらわした最大性能である．また文献 3-12)より $A(p)$=2･2･(p-1)/p･5 となる．始めの 2 は予測子・修正子の 2 回実行，次の 2 は ALLTOALL の 2 回実行，(p-1)/p は自分以外のプロセッサに送るデータの割合，5 は 5 変数の GSUM であることを意味する．GSUM(p)はプロセッサ数 p の時の communication の実測時間を，GSUM(p)-GSUM(1)は，communication 時間に含まれる総和計算時間を差し引いた p の時正味の通信時間である．r_c は B/s であらわした最大性能である．

表 3-3 に VPP300 と SX-4 に対する a_u と e_c を示す．尚 e_c は，VPP300 では p=14，SX-4 では p=6 で計算した．a_u/e_c を計算し式(3-2)に代入すると式(3-5)，(3-6)を得る．右辺の 3 つの係数は各々(A/B)，(a_u/e_c)，($r_a/r_c/8$)に対応する．

$$[\gamma_{p=14}]_{VPP300} = \frac{0.0149 \cdot 0.647 \cdot 30.9}{N_{cell}} = \frac{0.30}{N_{cell}} \tag{3-5}$$

$$[\gamma_{p=6}]_{SX-4} = \frac{0.0134 \cdot 16.9 \cdot 2.00}{N_{cell}} = \frac{0.45}{N_{cell}} \tag{3-6}$$

式(3-5)，(3-6)はプラズマ核融合炉の数値実験のように N_{cell} が大きい場合，VPP300 と SX-4 で通信の寄与を数パーセント以下に押さえることができることを示す．gyro3d では四則演算数が多いため(A/B)は 0.01 のオーダとなる．計算機の仕様の差で($r_a/r_c/8$)の値は２つの機種では大きく異なるが，効率の比(a_u/e_c)によりこれらが相殺し，係数は 0.30 と 0.45 となった．

表 3-3　VPP300 と SX-4 に対する a_u，e_c，γ_p

（I_s=1000, N_{max}=128･128･64 , I_{max}=32･32･16，N_{cell}=64）

＼	VPP300	SX-4
a_u	0.11　(r_a=2.2 Gflops)	0.27　(r_a=2 Gflops)
e_c	0.17　(r_c=570MB/s, p=14)	0.016　(r_c=8GB/s,p=6)
γ_p	0.0047　(p=14)	0.0071 (p=6)

このように並列処理の性能評価指標 γ_p が良い値であっても，計算機の論理性能を基準とした性能評価指標 a_u と e_c を用いると，個々の計算機が持つ課題が明確にできる場合がある．

参考文献

3-1) S. Orii, Metrics for evaluation of parallel efficiency toward highly parallel processing, Parallel Comput. 36 (2010) 16–25.

3-2) 折居茂夫，プラズマ粒子コードのためのベクトル並列計算法，プラズマ・核融合学会誌，75 (6) (1999) 704–716.

3-3) 渡部弘, 南正雪, 山本昭二, X 線結晶構造解析プログラムの並列化, JAERI-Data/Code 97–038 (1997).

3-4) T. Watanabe, H. Kaburaki, Increase in chaotic motions of atoms in a large-scale self-organized motion, Phys. Rev. E 54 (1996) 1504–1509.

3-5) 折居茂夫， レベル 1･2 並列ベンチマーク仕様及びそれに基づくスカラ並列計算機 SP2 のベンチマークテスト，JAERI-Data/Code 98–020 (1998).

3-6) V.E. Bazterra, M. Cuma, M.B. Ferraro, J.C. Facelli, A general framework to understand parallel performance in heterogeneous clusters: analysis of a new adaptive parallel genetic algorithm, J. Parallel Distrib. Comput. 65 (2005) 48–57.

3-7) Y. Abe, Present status of computer simulation at IPP, Supercomputing'88, (1988) 72–80.

3-8) K.Nishihara, H.Furukawa, M.Kawaguchi and Y.Abe, Japanese Supercomputing (Springer-Verlag, NY, Lecture Notes in Engineering, **36**, Eds. R. H. Mendez, S. A. Orszag), 59 (1988).

3-9) H. Naitou, T. Sonoda, S. Tokuda, V. K. Decyk, Parallelization of gyrokinetic paricle code and its application to internal kink mode simulation, Journal of Plasma and Fusion Research 72 (3) (1996) 259–269.

3-10) 例えば，福田正大，吉田正廣，中村孝，村瀬丈夫，小山隆司，並列ベクトル計算機「数値風洞」による FFT プログラムの性能評価，情報処理学会　第 47 回(平成 5 年度後期)全国大会，6H-9 (1993) 155–156.

3-11) 田中基彦，並列計算機による高速・大型粒子シミュレーション　ーその方法と問題点ー，プラズマ・核融合学会誌，72 (6) (1996) 542–548.

3-12) 折居茂夫, 数値計算のための並列計算機性能評価方法, 情報処理学会論文誌, 39, (3) (1998) 529–541.

４章　処理時間モデルを用いた性能予測の研究（実験２）

2.2 節で提案した処理時間モデル構築方法を確認する．4.1 節では，2.2.1 節で提案したホワイトボックスアプローチで式化したプログラムのホットスポットカーネルの実行回数を基に処理時間モデルが構築できることを確認する [4-1)]．4.2 節では，2.2.2 節で提案した処理時間を多項式で近似するブラックボックスアプローチを確認する [4-2)], [4-3)]．4.3 節では 2.3 節で提案したシンプレックス法を用いたモデルパラメータ決定方法による処理時間モデルの予測精度の向上を確認する [4-4)]．

ホワイトボックスアプローチではプログラムや並列計算機の情報を基にしてモデルを構築する．そこで本章の各節で用いる分子動力学プログラム（以後，MD コード）の計算内容を始めに説明する．尚使用した並列計算機は構築したモデル毎に異なるので，その都度説明する．

MD コードはアルゴン原子の熱伝導状態，対流渦の巨視的挙動を解析する分子動力学プログラム [4-5)]である．m をアルゴン原子の質量，$\mathbf{r}_i$ を i 番目のアルゴン原子の位置座標，$\mathbf{F}_i$ を i 番目の原子に働く力であるとすると，時刻 t における運動方程式は式(4-1)のようになる．

$$\frac{d^2\mathbf{r}_i(t)}{dt^2} = \frac{\mathbf{F}_i}{m} + \mathbf{g} \tag{4-1}$$

ここに，$\mathbf{g}$ は重力加速度である．$\mathbf{F}_i/m$ は式(4-2)で表す．

$$\frac{\mathbf{F}_i}{m} = -\sum_{j\neq i}^{n} \frac{\partial \phi(r_{ij})}{\mathbf{r}_i} \tag{4-2}$$

ここに，r_{ij} は i 番目の粒子と j 番目の粒子との距離である．$\varphi(r)$は Lennard-Jones ポテンシャルで式(4-3)となる．

$$\phi(r) = 4\cdot\varepsilon\cdot\left\{\left(\frac{\sigma_c}{r}\right)^{12} - \left(\frac{\sigma_c}{r}\right)^{6}\right\} \tag{4-3}$$

ここに，ε と σ_c は各々エネルギーと長さの次元を持つ量である．

系を現すマクロな物理量は，サンプリングセルに分割し，セル内の平均物理量としてまず計算した上で時間平均を行う．例えばセルの番号を k とすると，各セルの物理量の総和 A_k から時刻 t~$\Delta t \cdot I_s$ 内の時間平均物理量$\langle a_k \rangle_t$を式(4-4)として表わすことができる．

$$\left\langle a_k \right\rangle_t = \frac{1}{\Delta t \cdot I_s} \cdot \sum_{s=1}^{I_s} \frac{A_k(s)}{N_k(s)} \qquad (4\text{-}4)$$

ここに，$N_k(s)$は時刻 s におけるセル k 内の粒子数である．Δt は時間ステップ，I_s は時間積分回数である．取り扱う系は，２次元の矩形領域で，温度差がある拡散反射面で底面と上面を，鏡面反射面で左右を囲まれている．矩形領域のサンプリングセル数は 40×20 である．粒子の初期状態は，空間的に等間隔配置で，初期速度は Maxwell 分布である．

分子動力学では，各々の粒子は自分以外の全ての粒子からの力の影響を受けるため，力の計算回数は粒子数の自乗にほぼ比例する．しかしこのプログラムでは，力の及ぶ範囲を限定する遮蔽距離を導入して計算量の軽減を図るブックキーピィング法を用いるため，力の計算回数は粒子数に比例する程度に減少する．この方法は，一般に粒子 i に対し距離 $\alpha_{cut} \cdot r_{cut}$ 内の粒子番号を記述した表を作成し，表に記載されている粒子との距離を計算し，更に遮蔽距離 r_{cut} 内の粒子か否かを判定し，r_{cut} 内の粒子のみ式(4-2)の力の計算を行う．この表作成は粒子数の自乗に比例した計算量があるが，粒子の移動を考慮した一定の大きな時間間隔で再作成される．この分子動力学プログラムでは，r_{cut} は $3 \cdot \sigma_c$ をとり α_{cut} は 4 としている．

並列化プログラムのフローチャートを図 4-1 に示す． 表作成の結果，力の計算 force 以外の計算時間の割合が増す．このため force 以外に表作成の table，式(4-4)の物理量の計算をする propcel が並列化されている. 並列化に伴う主な通信は force における力の総和計算と, propcel における時間平均物理量の総和計算で生じる. プログラムは, Fortran90 の言語拡張 Fortran90/VPP を用いて並列化されている．

init では，粒子の初期位置を計算する．table は時間ループ中では間隔 n_{table} 毎に呼ばれる．maxwell では初期値として乱数から各粒子の速度を計算する．pcross と chkbnd で境界面を判定し，pmoves で粒子の移動と境界面の反射を計算する．式(4-1)の各粒子の速度の計算は main プログラムで行う． main プログラムでは，時間積分のくり返し回数 I_s の制御，サンプリング回数 n_{sample} の制御を行う．

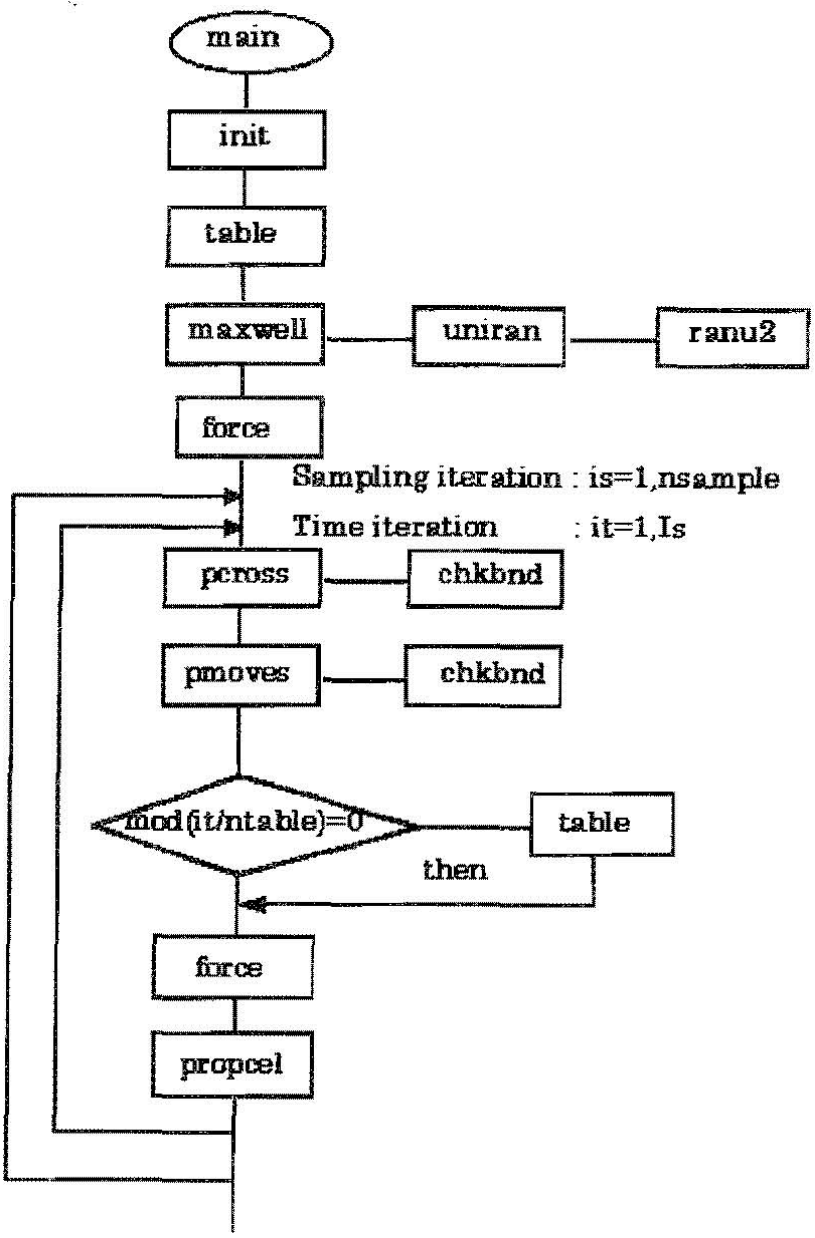


図 4-1　分子動力学プログラムのフローチャート

4.1 ホワイトボックスアプローチによる処理時間のモデル化の研究

並列計算機 VPP300 で実行される MD コードの並列処理の時間を 2.2.1 節で提案したホワイトボックスアプローチによる方法でモデル化した [4-1].

4.1.1 処理時間モデルの導出

1)モデル式の導出

MD コードを解析し，式(2-20)，(2-21)，(2-22)，(2-23)，(2-24)，(2-25)，(2-26)及び $c_1 = 1/(r_a \cdot a_u)$，$c_2 = 1/e_p$ を用いてホットスポットループの処理時間をモデル化した．これを表 4-1 に示す.

表 4-1　並列化分子動力学プログラムの処理時間モデル

(force)	
f_{force}	$= (n\cdot(n-1)/2)\cdot\{C_1\cdot[3_{\pm}+2_{*}+C_2\cdot(1_{/}+5_{\pm}+7_{*})_{if}]_{271}\}_{270}$
	$= (n-1)/2\cdot[C_1\cdot n\cdot(5+13C_2)]$
t_u	$= t_{0u270}+(n-1)\cdot\{t_{0u271}+C_1\cdot n/2\cdot[(5+13C_2)/(r_a\cdot a_{u271})]\}$
t_p	$= t_{0p270}+t_u/(p\cdot e_{p270})+\sigma_{270}$
σ_{270}	$= 2\cdot\{2\cdot[t_{0c270}\cdot p+(p-1)/p\cdot n\cdot X_8/(r_c\cdot e_{c270})]+n\cdot t_{0sum}\}$
$(\tau_p)_{force}$	$= (n_{sample}\cdot I_s\cdot+1)\cdot t_p$
(table)	
f_{table}	$= (n\cdot(n-1)/2)\cdot[(3_{\pm}+2_{*}+C_1\cdot(1_{+})_{if}]_{101}$
t_u	$= t_{0u101}+(n-1)\cdot[t_{0u100}+n/2\cdot(5+C_1)/(r_a\cdot a_{u100})]$
t_p	$= t_{0p100}+t_u/(p\cdot e_{p101})$
$(\tau_p)_{table}$	$= (n_{sample}\cdot I_s/n_{table}+1)\cdot t_p$
(propcel)	
$f_{propcel}$	$= n\cdot[(2_{max}+2_{min}+2_{int}+2_{*}+2_{+})_{302}+(5_{+}+2_{*})_{303}]$
t_u302	$= t_{0u302}+10\cdot n/(r_a\cdot a_{u302})$
t_p302	$= t_{0p302}+t_u302/(p\cdot e_{p302})$
t_u303	$= t_{0u303}+7\cdot n/(r_a\cdot a_{u303})$
t_p303	$= t_{0p303}+t_u303/(p\cdot e_{p303})$
t_u	$= t_u302+t_u303$
t_p	$= t_{p302}+t_{p303}+\sigma_{303}+C_3\cdot t_{p306}\sigma_{306}$
σ_{303}	$= 2\cdot[t_{0c303}\cdot p+(p-1)/p\cdot(n_x\cdot n_y)\cdot X_4/(r_c\cdot e_{c303})]+n_x\cdot n_y\cdot t_{0sum}$
σ_{306}	$= 3\cdot\{2\cdot[t_{0c306}\cdot p+(p-1)/p\cdot(n_x\cdot n_y)\cdot X_8/(r_c\cdot e_{c306})]+n_x\cdot n_y\cdot t_{0sum}\}$
$(\tau_p)_{propcel}$	$= n_{sample}\cdot I_s\cdot t_p$
(pcross)	
f_{pcross}	$= n\cdot[3_{+}+2_{*}+\{[1_{/}+C_4\cdot 1_{*}+(1-C_4)\cdot(1_{-}+1_{*})]_{if}+[1_{/}+C_5\cdot 1_{+}+(1-C_5)\cdot(1_{-}+1_{*})]_{if}\}_{chkbnd}]_{10}$
t_u	$= t_{0u10}+10\cdot n/(r_a\cdot a_{u10})$
$(\tau_u)_{pcross}$	$= n_{sample}\cdot I_s\cdot t_u$
(pmoves)	
f_{pmoves}	$= n\cdot(C_6\cdot(5_{+}+4_{*})_{if}+[(1-C_6)\cdot(1_{+})_{if}]_{300}$
t_u	$= t_{0u300}+9\cdot n/(r_a\cdot a_{u300})$
$(\tau_u)_{pmoves}$	$= n_{sample}\cdot I_s\cdot t_u$
[TOTAL time]	
F_{TOTAL}	$= n_{sample}\cdot I_s\cdot(f_{force}+f_{table}/n_{table}+f_{propce}+f_{pcross}+f_{pmoves})+f_{force}+f_{table}$
$(\tau)_{TOTAL}$	$= (\tau_p)_{force}+(\tau_p)_{table}+(\tau_p)_{propcel}+(\tau_u)_{pcross}+(\tau_u)_{pmoves}$

α_{cut}：4，　r_{cut}：3，　n_{sample}：6，　I_s：2000，　n_{table}：$14.16\cdot\alpha_{cut}\cdot r_{cut}$，　n_x：40，　n_y：20，
X_8：8 バイト，　X_4：4 バイト，　C_1：$\pi\cdot(\alpha_{cut}\cdot r_{cut})^2$/ 系の面積 $= \pi\cdot(\alpha_{cut}\cdot r_{cut})^2/(2.5\cdot n)$，
C_2：$1/\alpha_{cut}^2$，　C_3：$1/I_s$，　C_4：0.5（x 方向の粒子速度が正である確率），
C_5：0.5（y 方向の粒子速度が正である割合），C_6：～1（境界と接触しない粒子の割合）

表中 f の右辺の中で，下付き添字として付けた+，-，*，/は，プログラム中に記述されているその記号の四則演算をカウントしたことを，if は条件文中にある四則演算をカウントしたことを，max，min，int はそれぞれの内部関数をカウントしたことを示す．t_u，t_p はループの処理時間で，その添え字の３桁の数字は do ループ番号を表わす．添え字の付いた τ_u, τ_p は，添え字に示すサブルーチンの処理時間の総和である．処理時間は，プロセッサ数 p と問題の大きさ（この場合粒子数 n と n_x･n_y）以外に，C_1 から C_6 の 6 つのパラメータに影響を受ける．また，propcel の do303 はスカラ計算で実行されるが，a_u の値からスカラループを発見できるように，ベクトル計算と同じ r_a で評価する．force は２重ループにより構成されている．

図 4-2 に force のコンパイルリストを示す．外側の do270 は n-1 回転する．内側の do271 では table で作成された表に記された粒子のみが計算され，その平均粒子数は C_1・n/2 である．do271 では 5 個の四則演算により距離を計算し，if 判定で r_{cut} 内の粒子を探し，それら粒子に対し式(4-2)を 13 個の四則演算数で行う．このため，プログラムの処理時間は α_{cut}，r_{cut} の値に大きく左右される．また C_1 が粒子数 n に反比例するため，計算回数は n に比例する．force の呼び出し回数は，（サンプル回数 n_{sample}×時間積分回数 I_s=6×2000）プラス初期設定の 1 回で，計 12001 回である．また通信を伴う f_x と f_y の総和計算時間 σ は，各プロセッサで担当する粒子番号を決め，全対全転送で担当する粒子を集め，同粒子番号の総和を計算して再び全対全転送で各プロセッサに集めるプロセッサ間の総和計算モデル [4-6]を用いた．

table は 2 重ループで force と同じ距離計算を内側の do100 で行う．表作成のため C_1 で決まる粒子を if 判定で調べ，テーブルに書き出す．この時のカウンタのカウントアップを 1 個の足し算で行う．この処理は作用反作用を考慮した総当たり計算のため，f は n^2 に比例する．

```
      subroutine force(rcut2)
        parameter(n=7200)
        parameter(npe=10)
!xocl processor pe(npe)
!xocl subprocessor pes(npe)=pe(1:npe)
!xocl index partition ip=(pes,index=1:n,part=cyclic)
      common/winter/itab(400,n)
!xocl local itab(:,/ip)
        implicit real*8(a-h,o-z)
        include './inc2'
        rcut    = 3.0d0
c
v       do  225  i=1,n
v         fx(i) = 0.0d0
v         fy(i) = 0.0d0
v   225 continue
c
!xocl spread do/ip
s       do  270  i=1,n-1
s        xi   = x(i)
s        yi   = y(i)
v        fxi$ = 0.d0
v        fyi$ = 0.d0
*vocl loop,novrec
v       do  271  k=1,icol(i)
v         j  = itab(k,i)
v         xx = xi - x(j)
v         yy = yi - y(j)
v         rd = xx * xx + yy * yy
v         if ( rd  .gt. rcut2 ) goto 271
c
v         rdr   = 1./rd
v         rd3   = rdr**3
v         rd4   = rdr**4
v         rd    = (rd3-0.5)*rd4
v         fxx   = xx    * rd
v         fxi$  = fxi$  + fxx
v         fx(j) = fx(j) - fxx
v         fyy   = yy    * rd
v         fyi$  = fyi$  + fyy
v         fy(j) = fy(j) - fyy
v   271 continue
m        fx(i) = fx(i) + fxi$
s        fy(i) = fy(i) + fyi$
v   270 continue
!xocl end spread sum(fx),sum(fy)
c
      return
      end
```

図 4-2　VPP Fortran による force プログラムのコンパイルリスト

propcel では粒子の物理量を式(4-4)で計算する．do302 は粒子のセル上での位置の計算を行うための２個の掛け算，四捨五入のための２個の足し算，結果がセルの範囲内になるために使用する各々1 個の内部関数 max，min 及び int より成る．do303 はエネルギー

の計算のための 2 個の掛け算と 3 個の足し算及び，積算のための 4 個の足し算より成り，粒子の物理量 A_k と粒子数 N_k を $n_x \cdot n_y$ のセル k 毎に積算する．またプロセッサ間の総和を，粒子数 N_k のみに対して行う．この処理時間を σ_{303} に示す．do305 は，セル毎に物理量を粒子数で除して密度を計算して時間積分を行うが，このループの測定時間は小さく有意でないため，モデル化を省略する．do306 では，C_3 で決められた観測回数毎に do305 で計算したセル毎の物理量を I_s で除して時間平均を計算する．この時物理量即ち運動量の 2 成分及びエネルギーのプロセッサ間の総和を行う．この処理時間を σ_{306} に示す．この do ループの四則演算のモデル化は，サンプル数 n_{sample} が少なく，測定時間が有意でないため省略する．

pcross では do10 で 3 個の和と 2 個の積で速度成分を計算する．上下左右の境界との接触を調べる chkbnd が do10 にインライン展開されている．chkbnd では，すべての処理が if 文中で行われる．pmoves では do300 で境界と衝突しない粒子を if 文で抽出し，その位置を 5 個の和と 4 個の積で計算する．衝突する粒子の番号が記録され，1 個の和で数え上げられる．

2)　VPP300 に対するモデルパラメータの決定

MD コードをベクトル並列計算機富士通 VPP300 で並列処理し，処理時間のモデルと実測値を基に表 4-1 のモデルパラメータを決定する．使用した VPP300 は 16 プロセッシングエレメント(PE)で構成されたベクトル並列計算機で，各 PE の論理性能は 2.2 GMflops，搭載メモリは 512 MB である．各 PE はバンド幅 570 MB/s の双方向クロスバーネットワークで接続されている．

まずループレベルのモデル係数を求め，プログラムレベルの性能評価指標を計算する．それらの指標を用いて，計算機利用，プログラム開発，計算機設計という３つの見地か

ら性能評価を行う．このとき必要に応じて，ループレベルの性能を決める要因を特定する．

モデル係数決定には，まず表 4-1 で示したサブルーチンと do ループに対し，経過時間を測る VPP300 のサブルーチン gettod を用いて実施した．測定範囲は，粒子数 7200 から 96800 の 8 ケース，プロセッサ数 1 から 14 までの 10 ケースである．次に，この測定値と VPP300 の r_a=2.2Gflops，r_c=570MB/sec 及び表 4-1 の処理時間モデルを用いて回帰分析を行い，モデル係数を決定した．その結果を表 4-2 に示す．モデルの精度を検討するため，実測値の一部と予測処理時間を図 4-3 に示す．実測値は点，モデルとモデル係数を用いた予測処理時間は実線である．両者は良く一致しており，測定範囲内ではモデルの精度が十分であることを示す．そこでこれらを用い，測定範囲外の領域を含む広範囲の領域での処理時間のパラメータサーベイを行い，その性能評価を行った．

表 4-2　VPP300 のモデルパラメータ

	t_{0u} (μsec)	a_u	t_{0p} (μsec)	e_p	t_{0c} (μsec)	e_c	t_{0sum} ($n$sec)
(force)							
do_{270}	~0	–	$1{\cdot}10^2$	0.987	1.9	0.62	2.7
do_{271}	0.099	0.030	–	–	–	–	–
(table)							
do_{101}	$27.5{\cdot}10^2$	–	$1.6{\cdot}10^3$	0.999	–	–	–
do_{100}	~0	0.346	–	–	–	–	–
(propcel)							
do_{302}	25	0.387	60	1.00	–	–	–
do_{303}	$2.7{\cdot}10^2$	0.0107	~0	1.00	2.7	0.08	6
do_{306}	–	–	–	–	10.3	0.03	22
(pcross)							
do_{10}	1.1	0.073	–	–	–	–	–
(pmoves)							
do_{300}	38	0.343	–	–	–	–	–

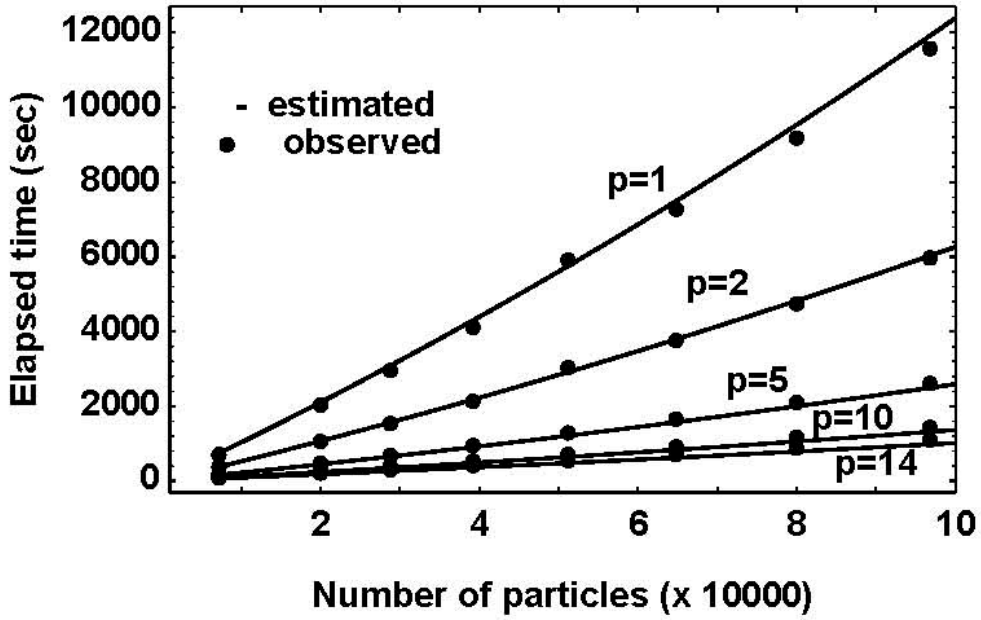


図 4-3　VPP300 における分子動力学プログラムの処理時間モデル精度

4.1.2 モデルを用いた性能予測による性能評価

MD コードを VPP300 で実行したときの処理時間モデルを用いて並列処理の性能を外挿し，予測領域の性能を評価することができる．そこで得られたモデルを用い，計算機利用，プログラム開発，計算機設計に対する 3 つの見地から性能を評価できることを示す.

(1) 計算機利用の見地からの性能評価

計算機利用の見地では, ある問題の規模の計算をある時間内に実現できるかという基準時間のクリア及び, プロセッサ数の増加に比例して処理時間が十分減少するかという, 戦略性が評価される．また，プロセッサの利用率が評価されることもある.

これらの評価は，横軸をプロセッサ数，縦軸を問題の規模即ち粒子数とする等高線図にすることにより，1 つの図で実施することができる．モデルを用いたプログラムレベルの予測処理時間を図 4-4 に示す.

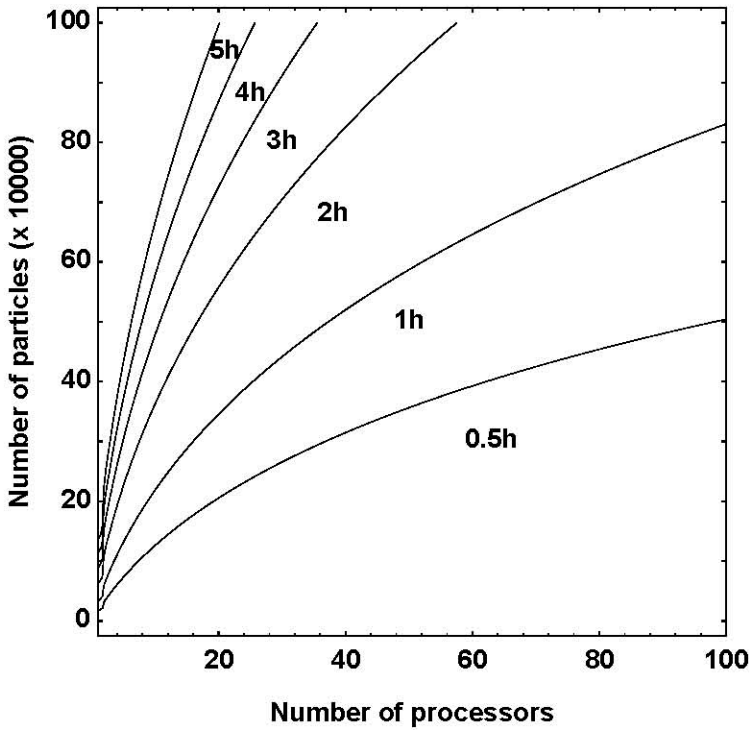


図 4-4　VPP300 における分子動力学プログラムの予測処理時間

例えば問題の規模を粒子数 50 万，基準時間を 1 時間以内と設定すると，プロセッサ数 40 以上の並列処理で実現できる．粒子数を固定すると，プロセッサ数の増加に対する処理時間の減少を評価できる．先の 50 万粒子の実行では 10 プロセッサで約 3 時間，40 プロセッサで約 1 時間であり，4 倍のプロセッサを投入して 3 倍性能が向上する．また，100 プロセッサで約 0.5 時間であり，10 倍のプロセッサを投入して 6 倍性能が向上する．プロセッサ数を固定すると，粒子数の増加に対する処理時間の増加を評価できる. 等高線は粒子数の増加と共に間隔が狭くなり, 処理時間が急激に増加する. 例えば，プロセッサ数 40 では 50 万粒子を約 1 時間，80 万粒子を約 2 時間，100 万粒子を約 4 時間で処理できる．これらの数値から，処理時間が粒子数の増加に対し線形の関係でない事がわかり，問題の規模毎の評価が必要なことがわかる．

プロセッサの利用率の評価には並列効率を用いる．これを図 4-5 に示す．粒子数の少ない領域を除いて 40%以上の並列効率であり，並列処理が有効に行われると予測できる．図 4-4 において 50 万粒子で 10 プロセッサのとき処理時間が約 3 時間の並列効率は図 4-5 で約 90%以上と確認できる．同様に 50 万粒子で 40 プロセッサのとき処理時間が約 1 時間の並列効率は約 80%である．並列効率を用いると，プロセッサの利用率の観点で計算機間の比較を行うことができる．逆に並列効率をある値に定めたときの，処理時間とプロセッサ数を知ることができる．

利用者の視点から見ると，並列効率は自分（たち）の複数のジョブに割り当てるプロセッサ数を決めるとき意識する値である．

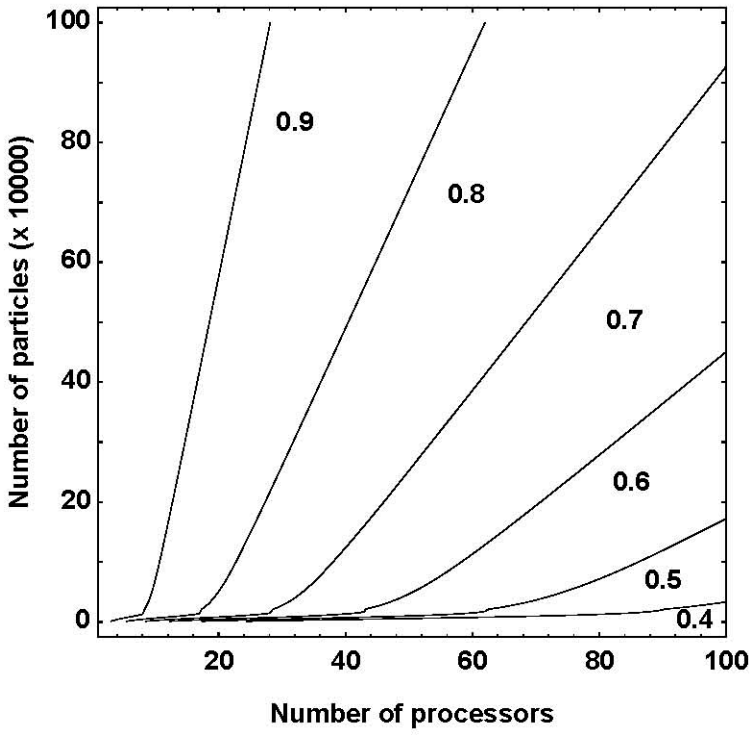


図 4-5　MD プログラムの並列効率

(2)　プログラム開発の見地からの性能評価

計算機利用の見地からの性能評価は，ハードウェア，コンパイラ，OS 等から成る計算機システムをブラックボックスとみなした評価である. その中でベクトル処理されるか否かスーパースカラであるか否か等々，個々の性能要因は問題としない.

一方プログラム開発の見地からの性能評価は, プログラム全体の処理時間を短縮するために実施する. そこで与えられた計算機で実現しているプログラム全体の処理時間の妥当性を，計算機が持つポテンシャル性能を基準として評価する．このため，並列効率に加えて，総合効率，ループのモデル係数を用いて評価する．計算時間に寄与しているルーチンの挙動，必要ならそれらの処理のループの性能要因を特定する.

初めに「並列処理できない処理（逐次処理）」だけが並列効率に及ぼす影響を調べる．これは式(2-7) の Amdahl の法則で計算することができる．これを図 4-6 に示す．この図は，図 4-5 の等高線の右方上がりの現象が逐次処理の割合，即ち R_p の違いだけで生じることを示す．さらに図 4-5 と比較すると，図 4-6 が等高線の形状はあまり変化しないまま全領域に渡って約 10%右にシフトすることがわかる．これが逐次処理以外の並列効率に対する並列オーバーヘッドの寄与である.

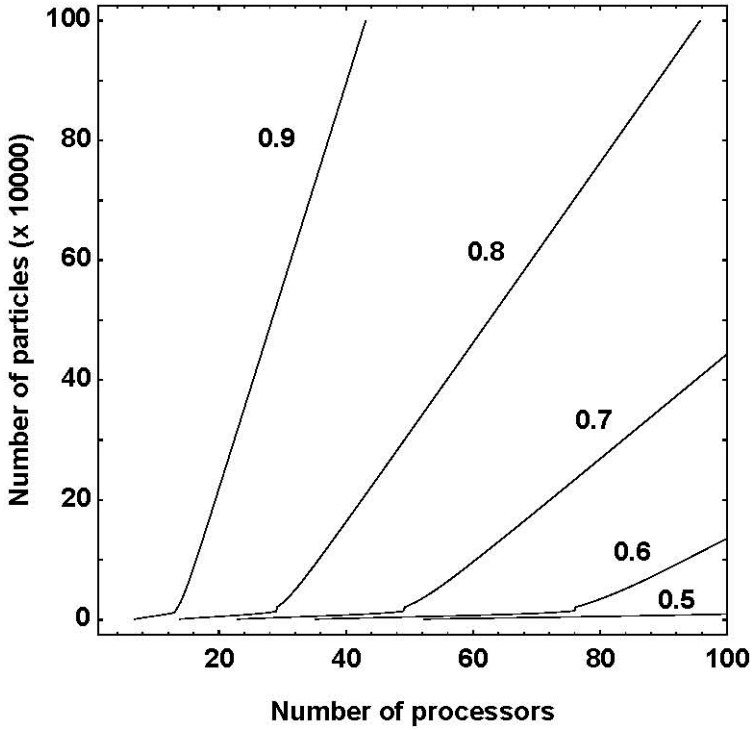


図 4-6　Amdahl の法則から計算した並列効率

並列化率と並列オーバーヘッドの寄与の度合を調べるため，プロセッサ数 100 の時の粒子数の変化に対する各処理の予測処理時間を計算し図 4-7 に示す．主要な処理時間を占めるのは，粒子数が少ない時は force，多くなると table と pcross が加わる．ここに force は，並列オーバーヘッド σ_{270} を含んだ処理時間である．そこで図 4-6 と図 4-5 の間で並列効率が異なる原因の一つが σ_{270} の並列オーバーヘッドであると特定できる．また，図 4-6 において右方上がりの傾向を作っている要因が逐次処理の pcross であることが特定できる． pcross の処理時間と並列オーバーヘッドの σ_{270} が同じオーダーであることから，プロセッサ数 100 の時の並列効率の値を決める要因には，並列化率と並列オーバーヘッドが処理時間的には同じぐらい寄与していることがわかる．

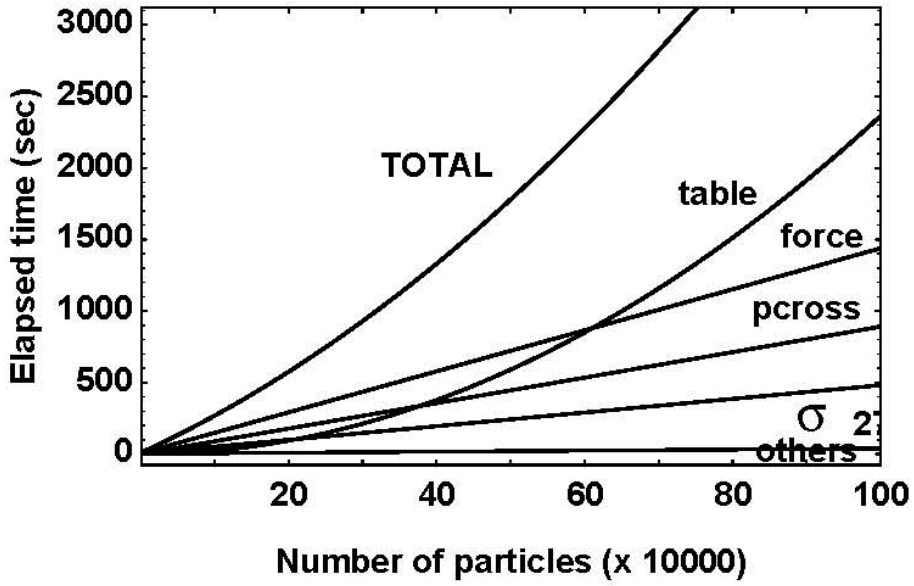


図 4-7　プロセッサ数 100 での各サブルーチンの処理時間

図 4-7 は，粒子数の増加に伴い force の並列オーバーヘッドと並列化していない pcross の占める割合が増え，特に table は force より大きな処理時間となることを示す．粒子数増加に伴い，粒子数の自乗に比例した計算量を持つ table の処理時間に占める割合が増加しプログラム全体の処理時間は増えるが，並列化率は向上する．これが粒子数の増加に伴い並列効率が上昇する原因である．また，図 4-7 において処理時間が粒子数の増加に対し線形の関係でない理由が，table の計算時間の占める割合が増え，その増え方が線形でないためであることがわかる．

次に計算機のポテンシャル性能を基準として性能評価する．これには図 4-8 の総合効率を用いる．図はこのプログラムが VPP300 の持つ全てのポテンシャル性能の何割を使用しているかを示す．図 4-5 の並列効率との差が，各プロセッサの演算器の使用効率をプログラムレベルで見たものとなる．総合効率が並列効率と大きく異なる点は，その値がプロセッサ数 1 でも 100%にならない点である．並列オーバーヘッドの寄与がほぼ零であるプロセッサ数 1 の並列効率は，プログラムレベルの演算器の使用

効率を表わす．図は，総合効率が粒子数の増加に伴い増加し，演算器の使用効率が改善されることを示す.

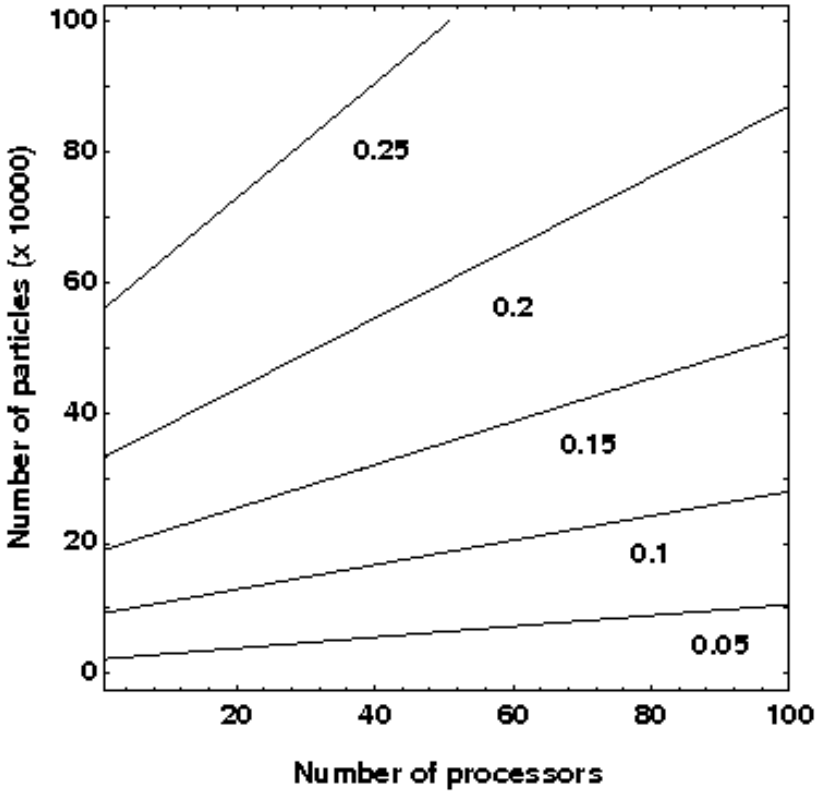


図 4-8　MD プログラムの総合効率

図 4-5 の並列効率と比較すると，粒子数 100 万個，プロセッサ数 50 から 100 の範囲では約 1/3，粒子数 10 万，プロセッサ数 10 から 100 の範囲では約 1/10 の値であることが読み取れる.

粒子数が増えると総合効率が向上するということに着目して，図 4-7 を見ると，粒子数の増加に伴い force と table の処理時間の逆転が生じていることに着目できる. 更に表 4-2 によりループレベルの性能を調べると，force の効率 a_{u271}=0.030，table の効率 a_{u100}=0.346 であることがわかる．そこでこの現象が，粒子数が多くなるに従って演算器の使用効率が高い table の計算時間の占める割合が増加するため生じると理解できる.

プログラムレベルの総合効率を決定している force の do271 は a_{u271}=0.030 という低い値である．プログラム開発の見地に立てば，この値を低くしている 2.1.4 節で述べた要因を取り除いたプログラム開発が可能か否かが問題となる．そこで重複している要因を取り除いたループを作成し，その性能を比較して要因の特定を行う．

図 4-2 に着目する．do271 の回転数即ちベクトル長は C_1 で与えられ，その値は約 90 である．これは粒子数が変化しても変わらない．次に，if 文中のベクトル長は 90 に C_2 をかけた値になる．その値は約 5 である．これはベクトルパイプライン処理に対しては極端に短いベクトル長である．更に距離の計算の時，ブックキーピングの表を参照するためインデックス j を itab(k,j)で間接参照する．これはメモリアクセス効率に影響を与える．これらから性能を決定する要因を特定するため，各々の要因を取り除いて do270 の時間を測定し flops 性能を比較する．j=itab(k, j)を消去し， do271 j=1,n/2 (=3600)とすれば, ブックキーピング法を用いないで長いベクトル長で連続メモリアクセスを実現できる．if 文を消去すれば短ベクトル長の回避ができる．処理((xx(k, i)=x(itab(k, i)), k=1,icol(i)), i=1,n-1)を do270 の前で行い，do270 中で x(i)の代わりに xx(k, i)を用いれば，連続メモリアクセスを実現できる．無論，y(i)も同様に置き換える．これらを行った結果を表 4-3 に示す．この表中の flop，flops，$(\tau_u)_{force}$ は VPP300 の性能測定ツール PEPA(PE Performance Analyzer)を使用して測定した値である．

表 4-3　force の性能要因

要因	Av. VL (do271)	Av. VL (if 文中)	f (7200) (Gflop)	$(\tau_u)_{force}$ (sec)	Mflop/s
連続メモリアクセス	90	5	67.9	554	123
if 文の消去	90	90	156	630	247
do271 を 3600 (=n/2) 回転	3600	5	6262	8007	782
do271 を 3600 (=n/2) 回転+if 文の消去	3600	3600	6262	7690	814
オリジナル)	90	5	67.9	650	104

この表は，連続メモリアクセスにより性能が 104Mflops から 123Mflops になり 1.2 倍向上すること，if 文の消去により演算量が 67.9Gflop から 156Gflop になったにもかかわらず$(\tau_u)_{force}$が 650 秒から 630 秒になり，性能が 104Mflops から 247Mflops と 2.5 倍向上すること，更にブックキーピング法を用いないと平均ベクトル長 Av. VL が 3600 となり，性能が 104Mflops から 782Mflops と 7.5 倍向上することから，性能を決定する主な要因は do271 の回転数が少ないために生ずる短ベクトル長で，次に if 文中の計算要素が少ないため生ずる短ベクトル長であることがわかる．

このようにプログラム開発の見地からの性能評価の結果, 粒子数の少ない領域において演算器の使用効率が低い force の性能がプログラムレベルの性能として現れ，その要因は短ベクトル長であることがわかる. そしてプログラムレベルの処理時間を短縮するためには，短ベクトル長を回避するアルゴリズムが必要となる．このベクトル長は C_1 と C_2，即ち計算の入力データ α_{cut} と r_{cut} にも依存しているため，総合性能は上記で述べた要因と，これら閾値の変化に大きく左右されることになる．

プログラム開発の見地からは, 将来生じる可能性がある条件に対してパラメータサーベイをして性能を評価する必要がある．重要なものの一つに，物理量の計算，即ち観測行為がある．観測が通信を伴うためである．図 4-9 に観測回数 n_{sample} を 600 回にした時の並列効率を示す．n_{sample}=6 の図 4-5 と比較すると，等高線は右方上がりが急になり並列オーバーヘッドの寄与が大きくなり，プロセッサ数が多い場合，高い並列効率の実現が難しくなる．これは propcel の do306 の並列オーバーヘッドの効果である．モデルによるパラメータサーベイにより，このような性能評価が可能になる．

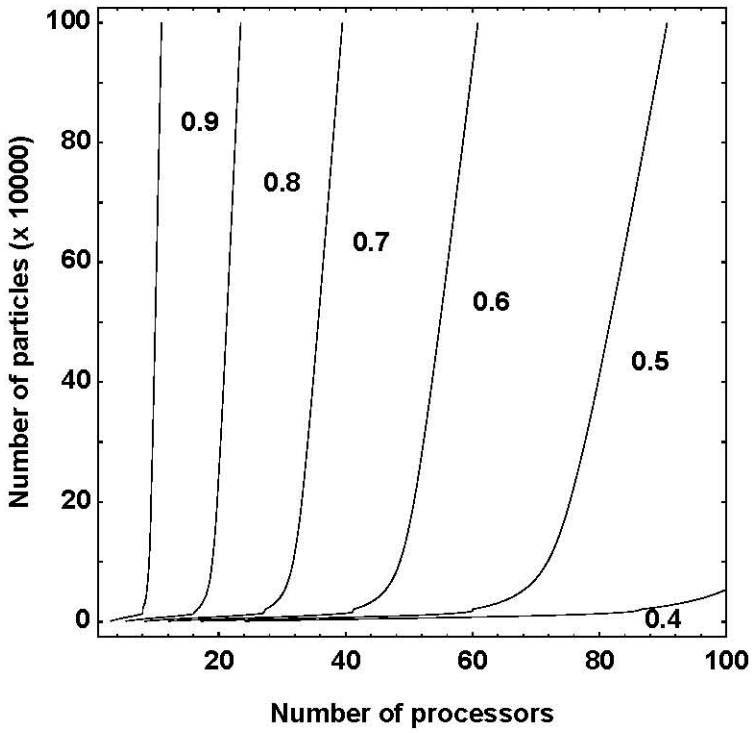


図 4-9　観測回数 600 の並列効率

(3)　計算機設計の見地からの性能評価

プログラム開発の見地からの性能評価では，プログラムレベルの性能を決定している要因を特定することができた．その要因は短ベクトル長で，要因が判明するとアルゴリズムの変更等で回避ができるかを検討できる．一方，計算機設計の見地からの性能評価では，短ベクトル長で充分な性能を発揮する演算器を設計した時に，それでどのくらいプログラムレベルの性能が向上されるかを評価することが重要となる．これはモデル係数を変えることにより可能となる．ここではモデル係数を a_{u271}=0.5 に設定した，ベクトル長で充分な性能を発揮する仮想の計算機を用いて議論する．この計算機の総合効率を図 4-10 に示す．a_{u271}=0.030 の図 4-8 と比較すると，プロセッサ数 1 の場合，粒子数 10 万の総合効率は約 0.1 から約 0.35 と 3.5 倍向上する．原点近傍の領域の粒子数で計算を行う場合，モデル係数を a_{u271}=0.5 を達成する設計は有効であることがわかる．プロセッ

サ数 20 の場合，粒子数 70 万の並列効率は約 0.25 から約 0.35 と 1.4 倍の向上である．このようにターゲットとなる問題の規模により評価結果が変わる場合があるので，設計した計算機がどのような粒子数（問題の大きさ）で実行されるかを考慮することは重要である．ところで，仮想計算機と実際の計算機の a_{u271} の比が 16.7 (=0.5/0.03)倍であることから考えると，プログラムレベルの性能向上が少ない．特に，force の処理時間の占める割合が大きい，粒子数の少ない領域では約 3.5 倍である．粒子数の多い場合は table の処理時間に占める割合が大きいことを思い出すと，a_{u100}=0.346 により総合効率が頭打ちになっていることがわかる．

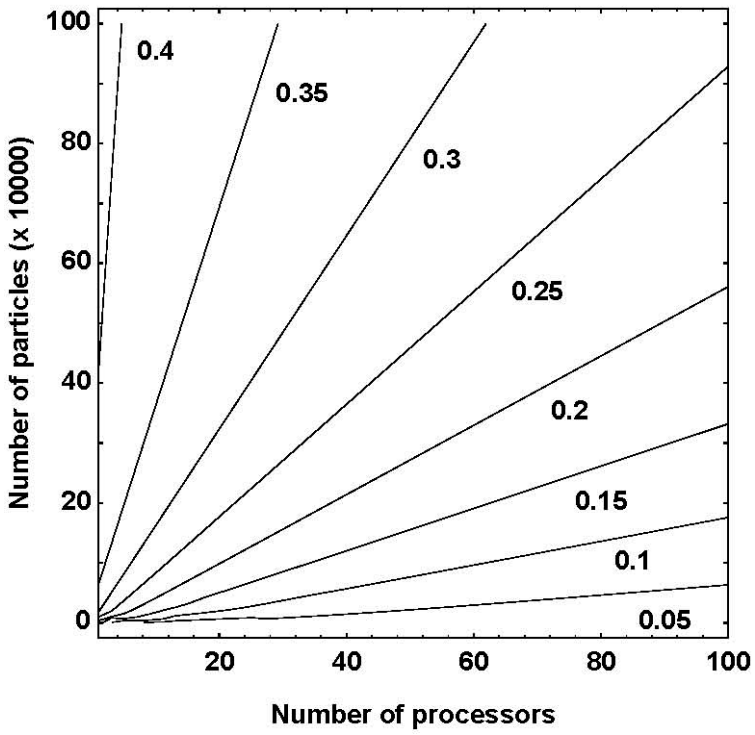


図 4-10　総合効率

原点近傍の領域についての要因は，仮想計算機の各サブルーチンの処理時間を調べることにより知ることができる．図 4-11 は force の a_{u271} の値が 0.03 から 0.5 に向上し

た結果，force の四則演算部分のみの処理時間が減少し，他の処理時間がそのままなので，table の処理時間寄与の少ない粒子数の少ない領域で σ_{270} と pcross の時間の占める割合が増加し，総合効率が余り向上しない原因になっている事がわかる．両者の処理時間はほぼ同じで，プログラムレベルの性能向上のためには，pcross の do10 の条件分岐処理の現状の効率 a_{u10}=0.073 と通信性能 r_c の改善が必要になる．

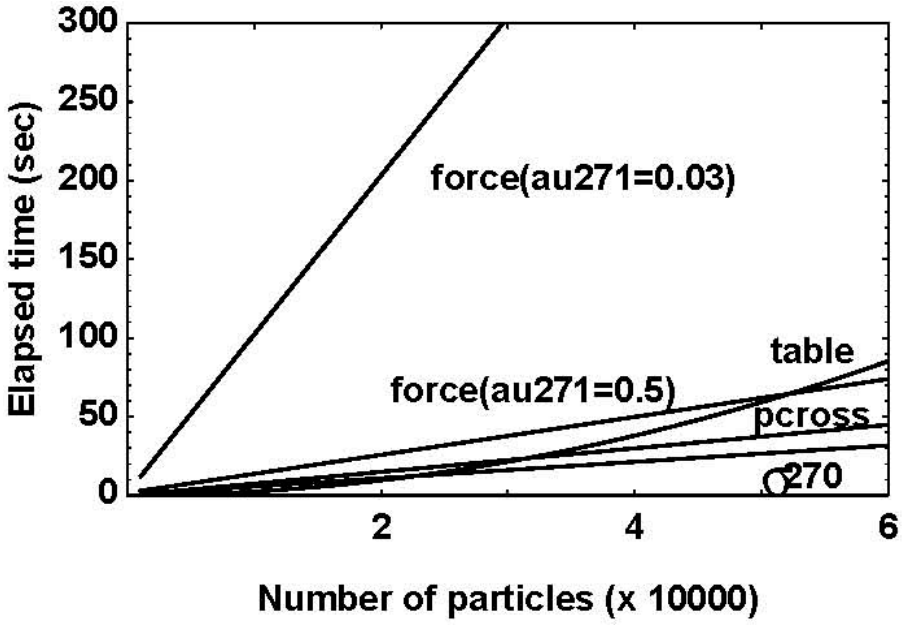


図 4-11　a_{u271}=0.03, a_{u271}=0.5, p=10 の force の処理時間

残念ながら force の効率を改善する短ベクトル長で高い効率を発揮する演算器を開発しても，a_{u10} は向上しない． pcross の do10 のベクトル長は既に十分長いためである．このことは，表 4-1 を見ると，do10 の回転数は n で if 文の真率 C_4 と C_5 が 0.5 であり，ベクトル長は約 n/2 であることからわかる．また表 4-1 からは， if 文とその中の四則演算数がほぼ同数であり，if 文の処理時間がループ効率低下の要因となっていることがわかる．従って，a_{u10} の改善には，if 文の処理時間を短縮する演算器の開発が必要となる．VPP300 は if 文の処理をベクトル処理で行っており， do10 はスカラ処理から比べると

数十倍以上高速化されている．これらのことから，a_{u271}=0.5 とすると force 以外の部分の性能設計仕様を全体的により高くすることが必要であることがわかる．

以上のように問題の大きさを変えるとプログラム中の計算時間の分布が変化する場合や, 計算時間の割合が大きい部分の性能を高くする設計をすると他の部分の性能仕様にも影響する場合があるが, ホワイトボックスアプローチでモデルを構築するとこれらの変化を考慮した計算機設計が可能となる．

4.2　ブラックボックスアプローチ

MD コードを IBM SP2 で実行したときの処理時間モデルをブラックボックスアプローチで構築できることを確認した [4-2), 4-3)]．使用した IBM SP2 は 48 ノードのスカラ並列計算機で，1 ノードの最大論理性能とメモリは各々266Mflops，512MB である．各ノードは 4 段の多段結合方式のスイッチで結合されその性能は双方向 40MB/秒である．

4.2.1 処理時間モデルの導出

1) プロセッサ数に関するモデルパラメータの決定

測定値 $\tau(p,n)$，$\gamma(p,n)$とし，$\Gamma=\gamma(8,n)$とし，式(2-34)を用いて処理時間モデルとフィッティングして多項式の係数 c_0，c_1，c_2 を決定し，これらを表 4-4 に示す．

表 4-4　問題の大きさ毎のフィッティング結果

n	Γ	a	c_0	c_1	c_2	**R**
3200	1117	1156	0.03521	0.08537	0.004541	0.9998
7200	2589	2381	-0.08016	0.10225	0.002351	0.9988
12800	5054	4523	-0.10509	0.09874	0.001512	0.9998
20000	8013	6755	-0.15695	0.10576	0.001123	0.9997
39200	17944	16953	-0.05524	0.08189	0.001039	0.9990
51200	26675	24206	-0.09256	0.07261	0.0008398	0.9994
64800	34229	30736	-0.10204	0.07196	0.0008153	0.9997
80000	45620	43063	-0.05605	0.06386	0.0007789	0.9996
96800	59413	58093	-0.02222	0.05276	0.0008270	0.9996

問題の大きさ即ち粒子数，n=3200，7200，12800，20000，39200，51200，64800，80000，96800 を用いた．ここに **R** は相関係数である．

図 4-12 に表 4-4 のモデルパラメータ決定に用いたフィッティング結果を示す．図は測定値とフッティングラインが $0\leq(1-\varepsilon'_p(p))/\varepsilon'_p(p)<9$（即ち並列効率メトリック $0.1<\varepsilon'_p(p)\leq1$）の範囲でよく一致しており，測定値から計算した式(2-34)の左辺の値が，単調に増加するモデルでフィッティングできることを示す．

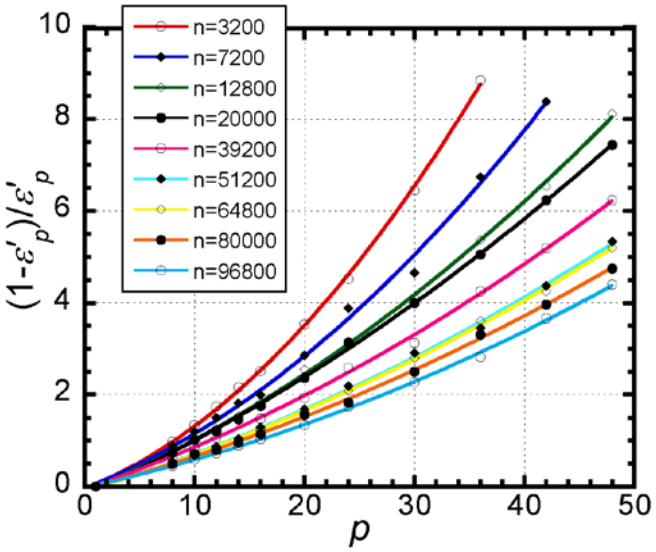


図 4-12　問題の大きさ毎の式(2-34)のフィッティング結果.

（マークは測定値，曲線はフィッティングラインを表す．）

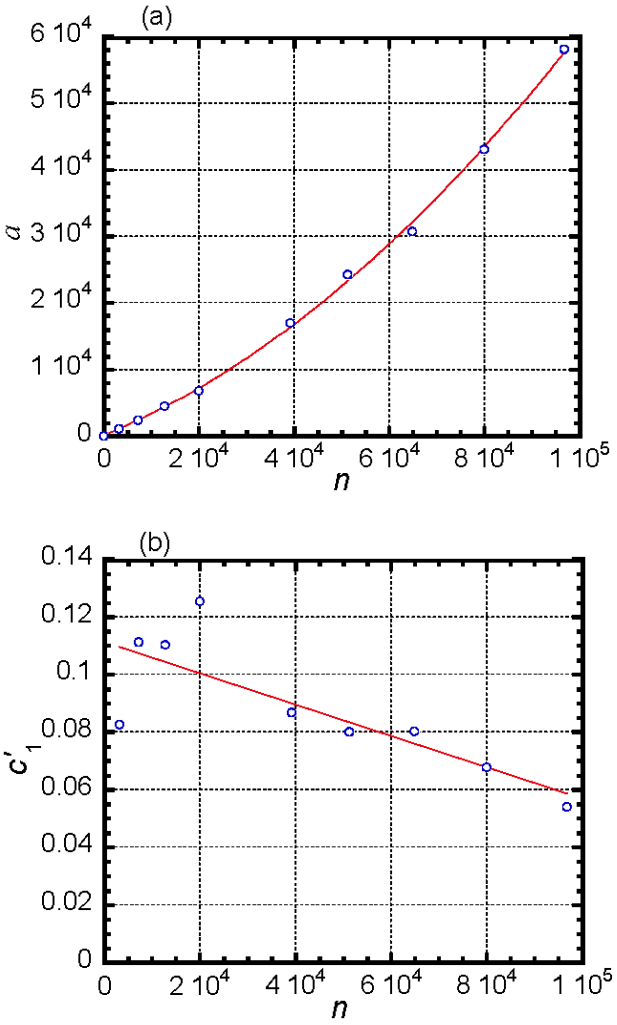
2)　モデルパラメータの問題の大きさに関するモデル化

表 4-4 のモデルパラメータ c_1，c_2 を a で規格化して c'_1，c'_2 とし，n の多項式としてモデル化した．a を多項式近似した結果と合わせてこれらを表 4-5 に示す．表中これらのモデルのフィッティング状態を図 4-13 に示す．図 4-13 (a)はモデルパラメータ $a(n)$を n の 2 次多項式で近似できることを示す．図 4-13 (c)は $c'_2(n)$ と n が反比例関係で近似で

きることを示す．$c'_1(n)$のモデルパラメータは，図 4-13 (b)に示すよう，n が小さいときにばらつきがあるデータを一次式でモデル化した．

表 4-5　モデルパラメータの問題の大きさに関するモデル化

	モデル式	**R**
$a(n)$	$155.9 + 0.2889 \cdot n + 3.148 \cdot 10^{-6} \cdot n^2$	0.9995
$c'_1(n)$	$0.1113 - 5.436 \cdot 10^{-7} \cdot n$	0.8041
$c'_2(n)$	$0.0007430 + 11.89/n$	0.9981



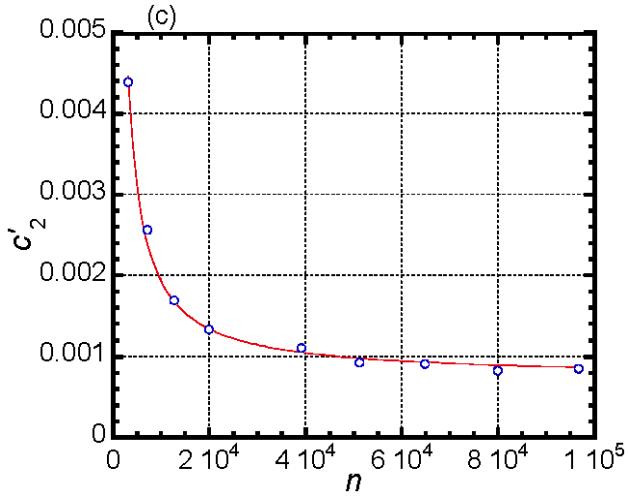


図 4-13　モデルパラメータの問題の大きさに関するフィッティング結果

表 4-4 のモデルパラメータ c_1, c_2 を a で規格化して c_1', c_2' とし，n の多項式としてモデル化することによりプロセッサ数と問題の大きさが可変の場合に対する処理時間モデルの式(4-5)を得る．

$$\tau(p,n) = (155.87 + 0.28887 \cdot n + 3.148 \cdot 10^{-6} \cdot n^2) \cdot \left(\frac{1}{p} + 0.11131 - 5.436 \cdot 10^{-7} \cdot n + (0.00074301 + \frac{11.891}{n}) \cdot p \right) \quad (4\text{-}5)$$

測定した処理時間とこの処理時間モデルの比較を図 4-14 に示す. シンボルは測定点, 曲線は式(4-5)に問題の大きさとプロセッサ数を代入して求めたものである. 赤で囲んだ部分にある測定値をフィッティングに使用した．この図より処理時間モデルは n=3200 においてややオーバーエスティメイトであるが, 他の場合測定と処理時間モデルがほぼ一致することが確認できる．

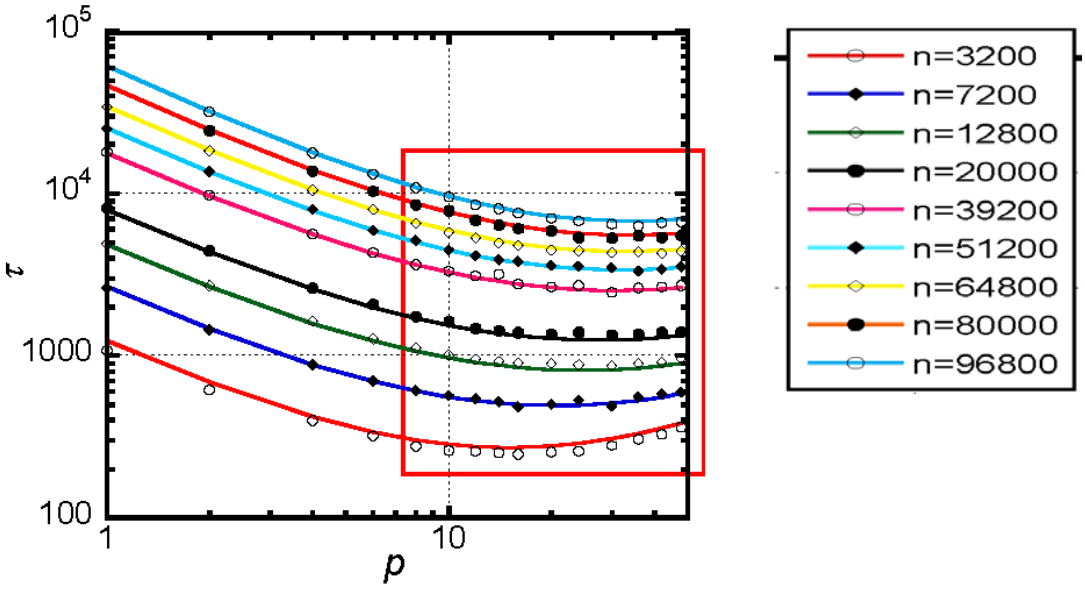


図 4-14　処理時間の測定値とモデルの比較

（赤で囲んだ測定値の内，$(1-\varepsilon'_p)/\varepsilon'_p<9$ を満たすものをフィッティングに使用）

4.2.2 モデルを用いた性能評価

処理時間モデルの式(4-5)から並列処理の性能を評価する方法を示す．フィッティングに使ったデータを包含し，評価は測定値を再現する図 4-14 の p=2~48，n=3200~96800 の範囲を対象に行った．

(1)　Strong Scaling における性能評価

処理時間モデルの式(4-5)を用い n=7200 と 80000 についての Strong scaling を図 4-15 の(a)と(b)に示す．ここに τ：MD コードの処理時間，$a(n)/p$：並列化部の処理時間，χ_0 (= $a(n)\cdot(0.1113-5.436\cdot10^{-7}\cdot n)$)：逐次処理時間等に代表されるプロセッサ数に依存しない並列オーバーヘッド，χ_1(= $a(n)\cdot(0.0007430+11.89/n)\cdot p$)：主に通信に代表されるプロセッサ数に依存する並列オーバーヘッドである．

これらを τ で除すと $\varepsilon_p = a/p/\tau$ で，χ_0/τ と χ_1/τ は ε_p を阻害する要因として説明できる．これを図 4-16 の(a)と(b)に示す．図中白抜きはε_p，赤はχ_0/τ，茶色はχ_1/τを表す．

図はプロセッサ数の増加と共に効率が下がるが，n=7200 に対し n=80000 では少し効率の改善が見られ，例えば効率 50%は，n=7200 では p=8 の付近，n=80000 では p=12 の付近で生ずることを示す.

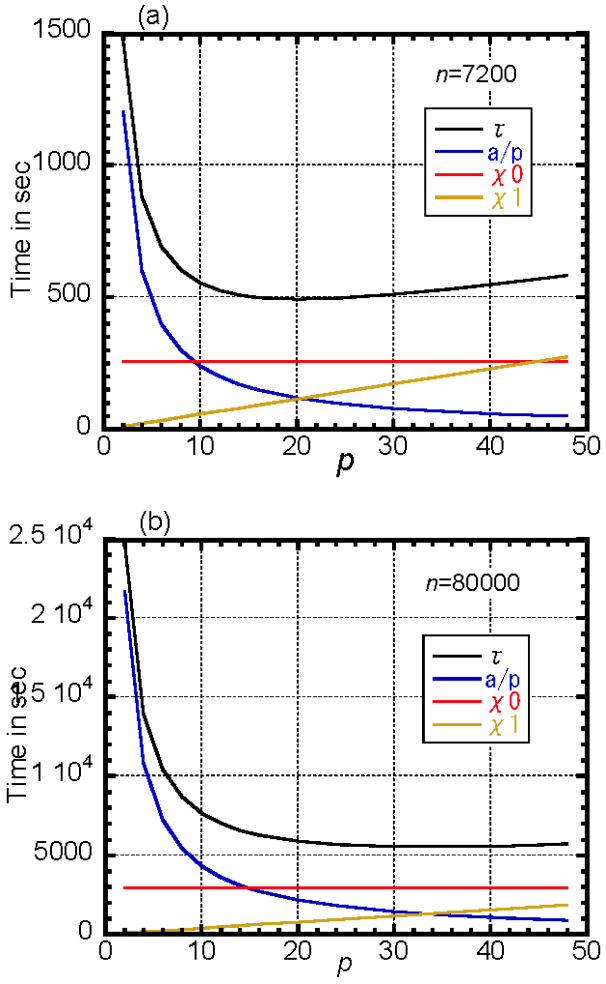


図 4-15　Strong scaling における時間変化

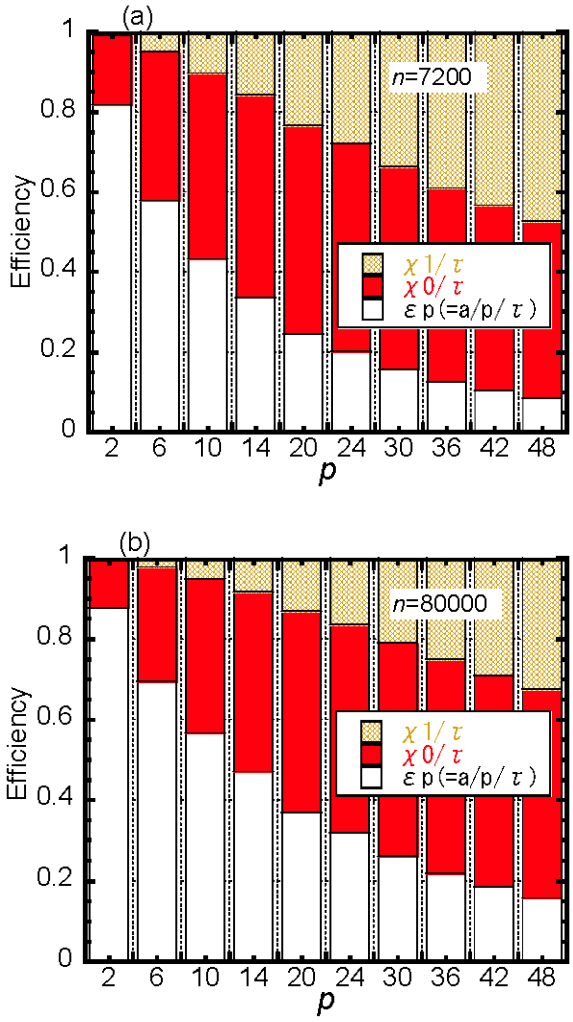


図 4-16　Strong scaling における効率の変化

ブラックボックスアプローチではプログラムの行間に隠れたホワイトボックスアプローチでは捉えられない現象を捉える事ができる．図 4-17 は並列オーバーヘッドの測定値とモデルのプロットである．図から，両者の間にはほぼ一定の差があることが見て取れる．並列オーバーヘッドとして観測した部分には含まれなかったこの差の部分は，並列化された時間として測定した中に隠れていた「目に見えない並列オーバーヘッド」と推測できる．図 4-17 (b)即ち n=80000 の場合，は明らかに「目に見えない並列オーバーヘッド」があり，並列オーバーヘッド全体の約 10%を占めており，並列性能の阻害要

因として無視できないことを示す. また測定値とモデル値の差がプロセッサ数によらずほぼ一定であることから，「目に見えない並列オーバーヘッド」が逐次処理時間に関係していることがわかる．図 4-17 (a)の n=7200 の場合も，n=80000 ほどではないが「目に見えない並列オーバーヘッド」が存在し，測定値とモデル値の差がプロセッサ数によらずほぼ一定である．従って問題の大きさ n に依存した逐次処理時間に関する「目に見えない並列オーバーヘッド」が存在することがわかる．この正体としては，並列化部として測定した処理時間の中に逐次処理が混じっている場合, データアクセスのスタートアップ時間が大きい場合等が考えられる.

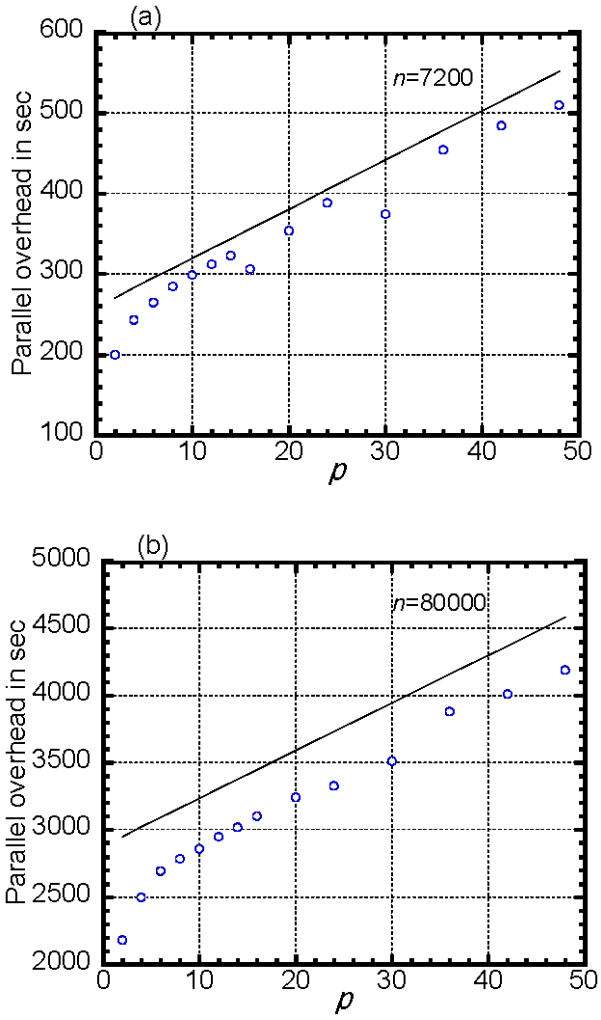


図 4-17　並列化部に隠れている並列オーバーヘッド（○：測定．線：モデル）

(2)　Weak Scaling による性能評価

Weak Scaling をプロセッサ毎の問題の大きさ *n* を一定にして，プロセッサ数を増すこととする．プロセッサ毎の問題の大きさを *k*=*n*/*p* で表し，式(4-5)から *n* を消すと式(4-6)を得る．

$$\tau(p,k) = k \cdot (\frac{155.87}{k \cdot p} + 0.28887 + 3.148 \cdot 10^{-6} \cdot k \cdot p)$$

$$\cdot \left(1 + (0.11131 - 5.436 \cdot 10^{-7} \cdot k \cdot p) \cdot p + (0.00074301 + \frac{11.891}{k \cdot p}) \cdot p^2 \right) \qquad (4\text{-}6)$$

図 4-18 は(a) k=500 と(b) k=2500 のときのプロセッサ数に対する時間変化である．Weak Scaling をするときは $\tau(p,k)$が一定であることを期待するが，これらの図は，この並列処理がプロセッサ数の増加とともに処理時間が増加することを示す．図はまた処理時間が一定にならない原因を可視化する．まず並列オーバーヘッドχ_0とχ_1がプロセッサ数と共に増加し，全処理時間に占める割合が大きくなりτが大きくなる主要な原因であることがわかる．χ_0とχ_1を比べると図の(a)と(b)共に$\chi_0 > \chi_1$であり，逐次処理部のオーバーヘッドの増加が処理時間増加の最大の原因であることがわかる．また通信を記述するχ_1は第二の原因であることがわかる．さらに *a*/*p* は *k*=500 についてはプロセッサ数に対してほぼ一定であるが，*k*=2500 になると *p*=2 から 48 の間に 2 倍以上増加する．これは *k* が大きくなったときに生じる第三の原因である．このように処理時間モデルを用いた Weak scaling により，システムの処理時間が増加することに対する原因を 3 つに分類してその大きさの違いを比べて性能評価することができる．特に Strong Scaling ではわかりづらい，問題の大きさの変化に対する逐次処理部の時間変化を可視化できる．

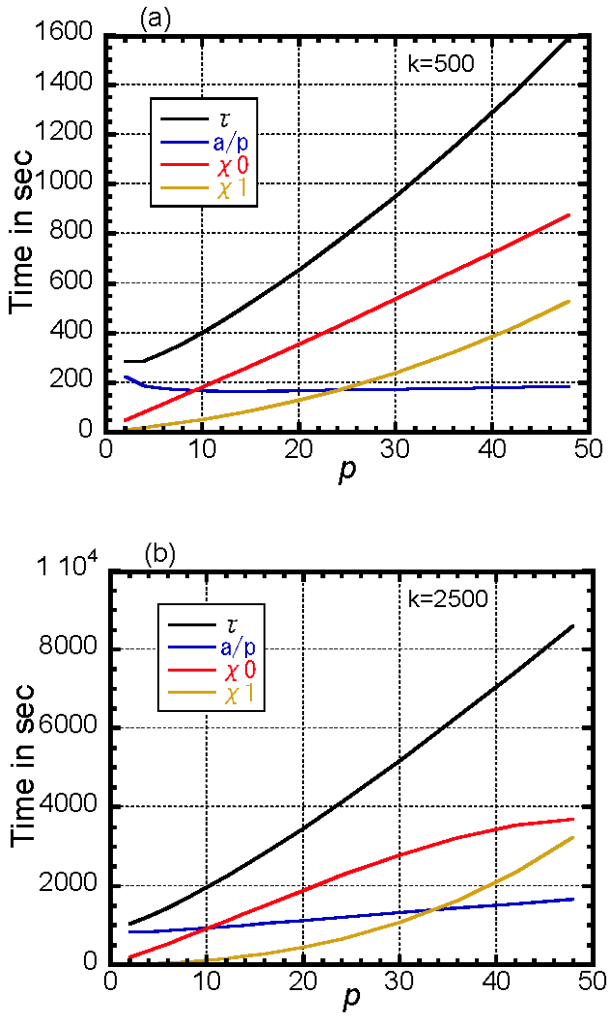


図 4-18　Weak scaling における時間変化

並列オーバーヘッドχ_0とχ_1の効率に対する寄与は，図 4-19 の並列効率メトリックから知ることができる．図はプロセッサ数の増加と共に効率が下がり，k=500 と 2500 でその区別はそれほど大きなものではなく，効率 50%は，*k*=500 では *p*=7 の付近，*k*=2500 では *p*=10 の付近で生ずることを示す．

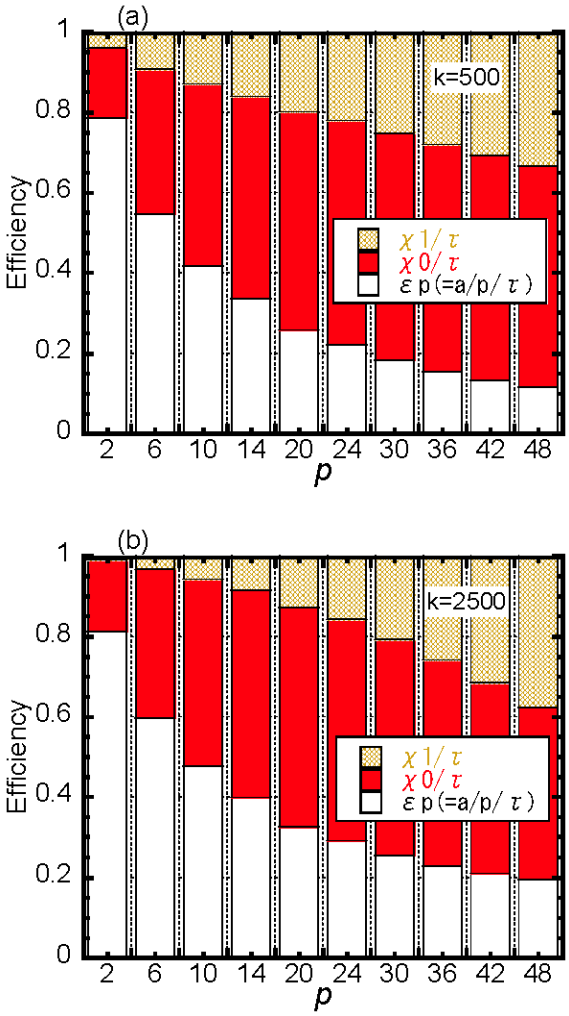


図 4-19　Weak scaling における効率の変化

(3)　並列化部に隠れた種々の並列オーバーヘッドのモデル化

(1)で示した「目に見えない並列オーバーヘッド」は種々のところで出現する．ここでは異なった 2 つの例を紹介する．

【並列化された多重ループに隠れた並列オーバーヘッド】

図 4-20 は MD コードの力の計算を Intel Paragon で並列化した MD コードの一部である [4-7]．

ここに粒子数 *n*，プロセッサ数 *p*= nnode である．遮蔽距離を設けて計算量を削減しているので，各プロセッサが計算する粒子数は予め計算して icol に，その粒子番号は itab に格納されている．計算した各粒子に働く分子間力は fx，fy に蓄えられ，tgdsum [4-8)]で全プロセッサの総和を行う．

このような 2 重ループの並列処理のモデル化はプログラムの構造を解析でモデル化するホワイトボックスアプローチによるモデル化では，do271 の回転数，データアクセスに種々のモデル化が必要であった [4-1)]が，図 2-5 のようなブラックボックスで MD コードを Paragon で実行した並列処理を簡単にモデル化することができる．

```
      subroutine force(rcut2)
      dimension tmp(n)
      ・・・
      do 270 i=mnode+1,ipmax,nnode
         iii=iii+1
         xi=x(i)
         yi=y(i)
     do 271 k=1,icol(iii)
         j=itab(k,iii)
      ・・・
271 continue
270 continue
      call tgdsum(fx,n,twp)
      call tgdsum(fy,n,twp)
      ・・・
      return
      end
```

図 4-20　分動力学プログラムの分子間力計算の並列化部 [4-7)]

ここでは図 4-20 の force の処理時間を並列処理システムの時間 τ とし, force と tgdsum の時間の測定値 [4-7]から時間モデルを構築した例を示す．図 4-21 (a) に示した a=153.0 は，Do 270 の実測値を γ として式(2-32)を用いて求めた結果である．

これ用いて ε'_p を計算して式(2-34)でフィッティングし，時間モデル X = 153.0*(1/p - 0.2043/p + 0.08023 + 0.0001941*(p-1))を得る．また tgdsum を式(2-34)でフィッティングし X_tgdsum=153.0*(-0.1962/p+(0.06613+0.0002322*(p-1)))を得る．図 4-21 (b)に測定値とモデルの比較を示す．図は force の測定時間がモデル化とよく一致することを示す．モデル化した並列オーバーヘッド X と四角形で示した tgdsum の測定値を比べると約 2 秒大きく，Do 270 中にプログラムからは判らない並列オーバーヘッドが約 2 秒の逐次処理として存在することがわかる. この tgdsum の測定値は Log(p)に比例する. そこで Log(p)のモデルを作り，X_ tgdsum，測定値と比較した．結果を図 4-21(c)に示す．丸で示したこの測定値をフィッティングした結果を実線で示す. 点線は Log(p)でフィッティングした結果である．図は Log(p)に比例する測定点を Log(p)を使わない式(2-34)で構築したモデル X_ tgdsum でフィッティングできることを示している.

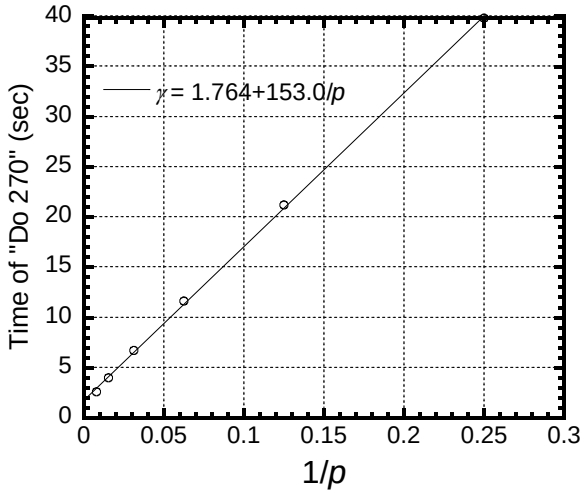


図 4-21 (a) Paragon における force 計算の a 値の決定

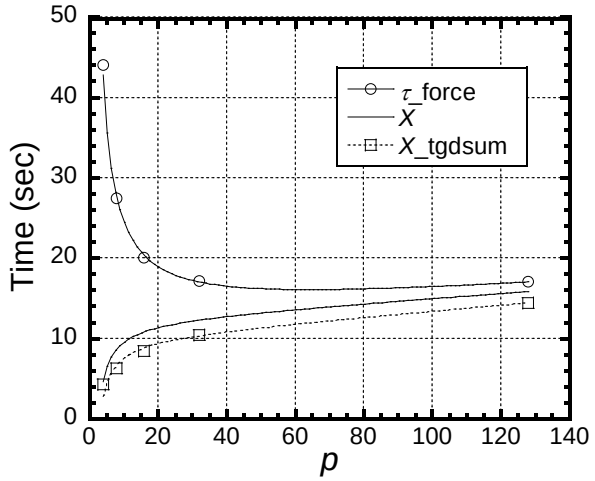


図 4-21 (b) Paragon における force 計算の測定時間とモデルの時間

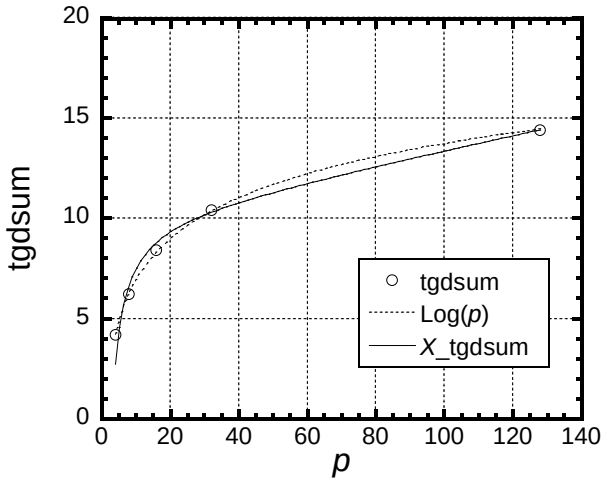


図 4-21(c) Paragon における tgdsum の測定時間とモデルの時間

【コンパイラ制御行で並列化されたループに隠れた並列オーバーヘッド】

図 4-22 は MD コードの force の部分を VPP300 の VPP Fortran のコンパイラ制御行を使って並列化したプログラムである [4-1)].

```
      subroutine force(rcut2)
      parameter(n=7200)
      parameter(npe=10)
!xocl processor pe(npe)
!xocl subprocessor pes(npe)=pe(1:npe)
! xocl index partition ip=(pes,index=1:n,part=cyclic)
      Common/winter/itab(499,n)
! xocl local itab(:/ip)
        ・・・
! xocl spread do/ip
      do 270 i=1,n-1
        ・・・
! vocl loop,novrec
      do 271 k=1,icol(i)
        ・・・
  271 continue
      fx(i)=fx(i)+fxi$
      fy(i)=fy(i)+fyi$
  270 continue
! xocl end spread sum(fx),sum(fy)
      return
      end
```

図 4-22　VPP Fortran で並列化された分動力学プログラムの分子間力計算

図中の! xocl end spread sum(fx),sum(fy)は 1 行で書かれているが，fx と fy に関するプロセッサ間の総和で，並列オーバーヘッドとなっている．この総和時間は VPP Fotran の知識がないと測定することができないが，！xocl spread do/ip と！xocl end spread sum(fx),sum(fy)を時計で挟んで測定し，2.2.2 節で提案したブラックボックスアプローチを用いると，総和時間とそれ以外の並列化された dio270 の中に隠れている並列オーバーヘッドをモデル化することができる．

図 4-23 に MD コードを VPP300 で実行した結果を示す．図中丸は force の測定値，三角は総和計算の測定値である．黒の実線は式(2-34)を用いて求めた時間モデル τ=640.9・(1/p +0.02830)で，測定値とモデルが良く一致していることを示す．尚 c_0 は小さい値なので零と置いてモデル化した．赤の実線はモデル化した並列オーバーヘッドで，X=18.1(=640.9･0.02830)とプロセッサ数に関係ない定数である．実測値は p=14 のとき X~11 で，これは並列オーバーヘッドの約 40%がプロセッサ間の総和以外のところで生じていることを意味する．モデル化により，今まで検知できなかったこの並列オーバーヘッドが並列性能に寄与しており，プロセッサ数を増やして性能を向上することに対して大きな阻害要因であると評価することができる．

図 4-23 において明るい青の四角は並列効率メトリック ε'_p の値である．それを結ぶ曲線はモデルから得た ε'_p である．また p=14 までの測定値に基づいて作成した時間モデルによる性能予測として，ε'_p=0.5 のプロセッサ数は p=35 で，そのときの τ=30 秒を得る．このように時間モデルを構築すると，処理時間，効率とそれに対するプロセッサ数を定量的に評価できるようになる．

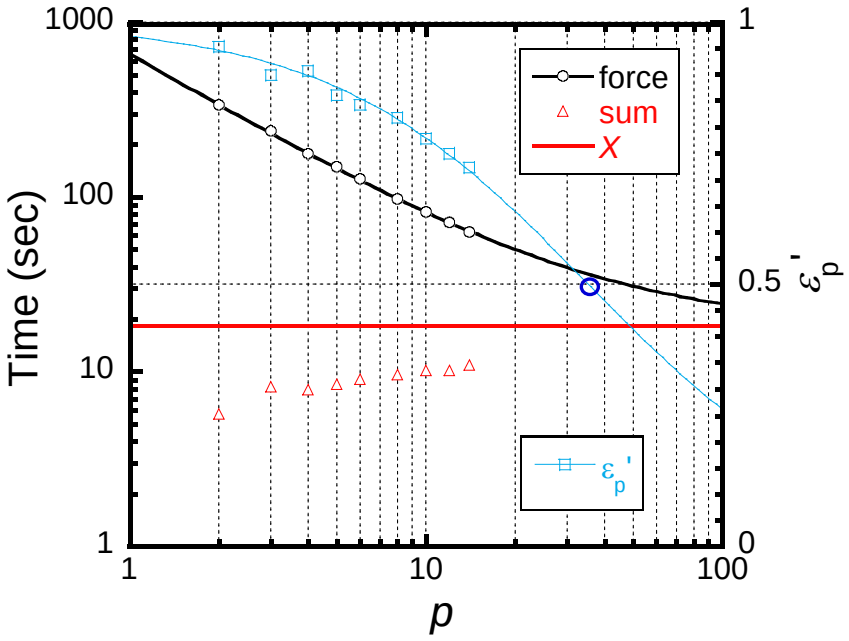


図 4-23　VPP300 における force 計算の測定時間とモデルの時間

4.3 モデルの予測力向上の研究

ホワイトボックスアプローチでは，モデル構築に時間を要するという問題がある．理由は，プログラムの構造に基づいてモデル式を構築することに加え，文献 4-13)の付録に示すように，各ループの処理時間を測定してノイズを伴うその測定値からモデルパラメータを推定するのにループ毎にケースバイケースのフィッティングを必要とするためである．この問題点は，プログラムの構造情報を用いずプログラムの処理時間のみを使ってモデル化する，ブラックボックスアプローチにより大幅に軽減できる． 一方ブラックボックスアプローチでは，モデル化のため多くの観測データを必要とするデメリットがあった（例えば図 4-14 の四角の中のテータ全てをフィッティングに使用）．予測の精度を悪くする項の影響を排除するためには多くの観測データが必要でなためである．例えば必要でないプロセッサ数の二乗の項をモデルに加えればその項のモデルパラメータにも値が入り，この項が効くモデルができる場合がある．予測においてこの項が効くと予測領域の並列処理は効率悪い並列処理であると評価することになるが，二乗項をモデルに加えなければ予測領域において効率良い並列処理である場合もある．このように必要でない項が予測領域で効くことが無いようにするためにフィッティングの観測データを増やしたところ，図 4-14 に示すような結果となった．この主な原因の一つは，モデル化に必要のない項が観測データのノイズの残差平方和を最小にするために使われる，過剰適合である．そこで 2.3 節ではモデルパラメータに非負制約を付けた残差不等式をシンプレックス法で解くことにより，過剰適合を抑制して処理時間モデルの予測精度を向上する計算法の提案を行った [4-4)]．本節ではこの計算法により過剰適合が抑制され，モデルの予測精度が向上することを確認する．まず単純な 1 変数モデルに関するシミュレーションとして 2 つの事例を用いることで手法の比較の妥当性を確認し，次に並列時間モデルについて 2 つの事例で確認する．2.3 節で提案したシンプレックス法を用いた計算法(LP)の効果を確認するため，制約なしの最小二乗法，AIC に基づく変

数選択を伴う LSM[4-11]，非負制約付最小二乗法(NNLS) [4-15]について比較する．これは，提案手法が非負制約付 LP であるため，制約付 LSM と比較するものである．また非負制約の結果 0 と推定されるパラメータに対する基底関数がモデルから除去されることは，重回帰分析で行われる変数選択と結果的に同等である．そこで多くの説明変数の候補があり予測に有用な少数の説明変数を用いた重回帰分析を行うために一般的に行われる変数選択として，説明変数の組み合わせを変え LSM でモデル化した際にモデルの AIC（赤池情報量規準）が最小の組合せを選択する方法(LSM 変数選択)との比較も行う．

LP のツールとして Mathematica [4-9]の LinearProgramming 関数を用いた．この関数は入力を有理数で与えると有理数または整数の結果を返す． NNLS のツールとしては MATLAB の lsqnonneg 関数 [4-16]を用いた．LSM には統計解析ソフトウェア R[4-10]を用い，パラメータ推定は lm 関数を，AIC に基づく変数選択は step 関数を用いた．

4.3.1 1 変数モデルによるシミュレーション

(1) $y^0=x^2$ のモデル化と過剰適合抑制

観測データとして，真のモデル $y^0=x^2$ を基に生成したシミュレーションデータ **y** について，多項式関数 $f(x)=a_0+a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5$ (M=5)でモデル化する．少ない観測データを用いたモデル化を考え，観測データ数 N が基底関数の数 M に近い観測データ数でかつ $N \geq M$ を想定し，**y** の要素は，1.1 から 3.1 を等間隔に 8 点(N=8)で分割した x_i に対して $y_i = x_i^2 + d_i$ (i=1,…,8)で，平均 0，標準偏差 σ 標準偏差 σ の正規分布に従う揺らぎ d_i を与えて生成した．

表 4-6 はこの **y** と $f(x)$を用いて LSM で得られたモデルパラメータである．このモデルは真のモデル $y^0=x^2$ とは大きく異なる．

表 4-6　LSM によるモデルパラメータ（σ=0.01）

パラメータ	推定値	標準誤差	t 値	p 値
a_0	4.30	2.00	2.16	0.164
a_1	–11.2	5.38	–2.07	0.174
a_2	12.2	5.61	2.17	0.162
a_3	–5.42	2.83	–1.92	0.195
a_4	1.29	0.691	1.86	0.204
a_5	–0.120	0.0658	–1.82	0.211

図 4-24 は表 4-6 を用いた $f(x)$の外挿の評価である．破線が $y= x^2$．実線が表 4-6 を用いたモデル．×印が入力データである．実線は×印を精度良く内挿している．一方外挿領域は全く破線と一致しない．図 4-24 は，負のモデルパラメータに由来する，典型的な過剰適合例である．

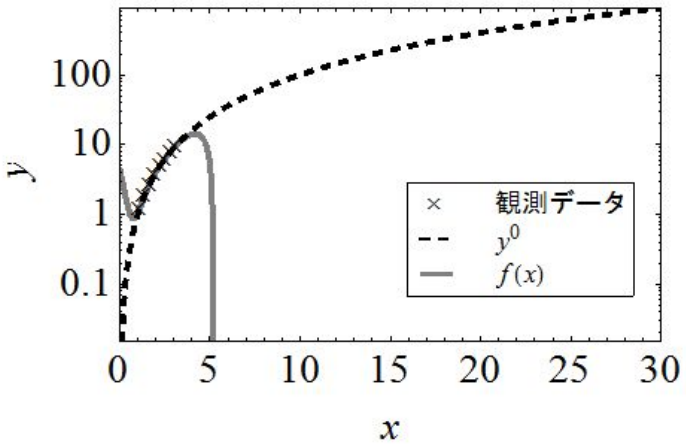


図 4-24　表 4-6 による $f(x)$の外挿

過剰適合となる基底関数を用いないモデルを構築するため，変数選択による最適なモデルの推定を試みた．LSM による回帰係数の推定についてモデルの適合度指標として，R^2 値= 0.99999，修正 R^2 値= 0.99997 でありモデルの適合は高い．回帰係数 $a_{1,\ldots,}a_5$ の p 値はいずれも 0.05 より小さな値でなく係数が 0 と判断されるものはない．またこのシミ

ュレーションデータに対して変数選択を行い AIC を比較したところ，この 5 変数を用いたモデルが AIC 最小であった．この分析結果からは，過剰適合項を含むことから x の値が大きい外挿時に予測が大きく外れることがわかる．切片項の除外も含む全ての可能な組み合わせに対する回帰モデルの適合度を比較すると，修正 R^2 最大のモデルは $a_0+a_2x^2+a_5x^5$ で AIC 最小のモデルは $a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5$ であった．このように変数選択では，必ずしも真のモデルが選択される訳ではなく，この結果のように過剰適合項が説明変数として残ることもある．この 1 回の結果から判断することはできないため，切片項の除外も含む AIC に基づく変数選択を，同様に $y_i = x_i^2 + d_i$ (i=1,…,8)を異なる乱数に対して 100 組生成したものに対して行い、各組で選択されたモデルの回数を表 4-7 に示す．

表 4-7　100 回の変数選択により選択されたモデル

モデル	回数
a_2x^2	19
$a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5$	19
$a_1x+a_3x^3+a_4x^4+a_5x^5$	12
$a_0+a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5$	9
$a_2x^2+a_3x^3+a_4x^4+a_5x^5$	7
$a_2x^2+a_4x^4+a_5x^5$	7
その他	25

真のモデル $y= a_2x^2$ が最も多く選ばれているが 2 割程度に過ぎず，逆に外挿に対して悪影響をおよぼす x^4 および x^5 の項が含まれたモデルも半分以上見られた．表 4-6 のモデルの x^4 および x^5 の項に対する回帰係数の値が大きいことから外挿として x の値が大きいときに誤差の影響は大きくなる．

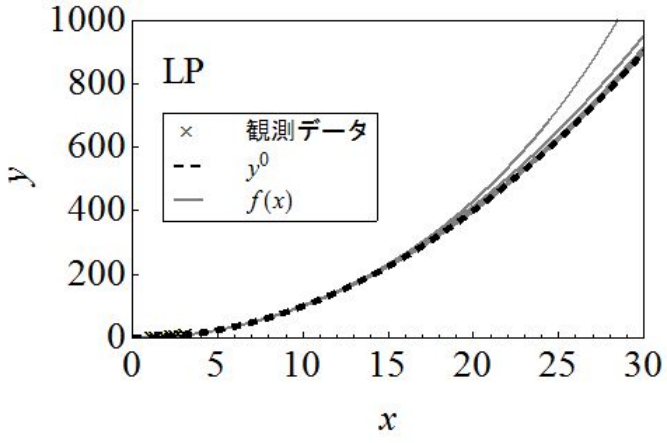
次に，表 4-6 と同一のシミュレーションデータに対し，本論文の提案である $\mathbf{a}\geq\mathbf{0}$ の制約を課したモデル化を 4.3 節の冒頭で述べた LP と NNLS を用いて行った．なお本節の

変数と LP の手順である図 2-8 の変数は $x_i \rightarrow x_{1i}$, $y_i \rightarrow y_i$, $f(x) \rightarrow f(x_{1i})$と対応する．モデルパラメータを表 4-8 に示す．両者共 a_2 と a_3 に値を持ち，a_2 値はほぼ真値 1 に近くなる．また両者とも a_0, a_1, a_4, a_5 が零となり，揺らぎをモデル化していた過剰適合が抑制されたことが確認できる．

表 4-8 非負制約付モデルパラメータ（σ=0.01）

	a_0	a_1	a_2	a_3	a_4	a_5
LP	0	0	0.994	0.00212	0	0
NNLS	0.	0.	0.999	0.000186	0.	0.

表 4-6 のシミュレーションデータと同じ方法で作った 5 組のシミュレーションデータに対する LP と NNLS の予測を図 4-25 に示す．両者による 4 回の予測は観測データ x_N の約 10 倍の x=30 においてまた y_N の約 100 倍の y=900 においてほぼ破線の $y^0 = x^2$ を予測する．このように非負制約 $\mathbf{a} \geq \mathbf{0}$ を課すことにより図 4-24 に示したような過剰適合が抑制され，モデルの予測精度向上を図れることが確認できる．



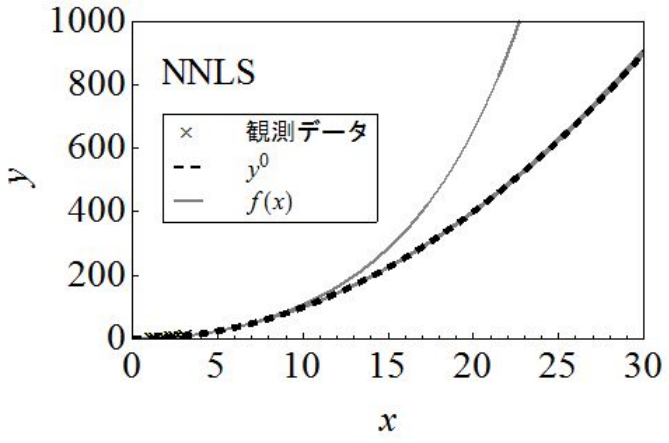


図 4-25　x^2 モデルの予測（上：LS，下：NNLS）

さらに同様の実験を 100 組のシミュレーションデータに対して行った．表 4-9 に LP と NNLS に対する全ケースの平均値を入力データ x の最大値 3.1 の約 10 倍までのレンジ x=5, 10, 20, 30 に対して示す．表より，LP と NNLS の両方で，$y^0(10)$の値 100 をほぼ予測できるモデル化ができたことが確認できる．

ところでモデルパラメータが零値ということは，自動的に過剰適合が緩和されたモデルが得られたことを意味するが，100 ケースのシミュレーションデータに対してモデルパラメータが零値であったり無かったりする基底関数は，予測モデル構築の障害となる可能性が高い．そこで 100 回の各モデルパラメータが零値になるケースをカウントした．これを表 4-10 に示す．まず真のモデル y^0 と同じ項の a_2 の回数は 0 で，モデル化において正しくこの項が選択されたことがわかる．これに対し，零回になる回数が多いのは a_0, a_4, a_5 である．このうち a_5 が表 4-9 における x=30 の分散を大きくしている原因なので，σ=0.01 の揺らぎがある観測データで 100 回モデル化すると約 70 回で真のモデル y^0 に近い予測モデルが得られたことになる．

表 4-9　100 ケースの予測値の要約（σ=0.01）

x		5	10	20	30
$y^0(x)=x^2$		25	100	400	900
LP	平均値	25.2	106.	571.	2175.
	標準偏差	(0.235)	(10.1)	(340.)	(2604.)
NNLS	平均値	25.1	105.	548.	2006.
	標準偏差	(0.204)	(8.4)	(279.)	(2130.)

表 4-10　各モデルパラメータが零になる回数

	a_0	a_1	a_2	a_3	a_4	a_5
LP	84	40	0	56	98	70
NNLS	76	41	0	69	95	68

(2) $y=x^2+\log_2(x)$のモデル化と過剰適合抑制

(1) では 1 つの関数から作ったシミュレーションデータを観測データとしたモデル化だったが，現実の並列処理時間は複数の基底関数を必要とする複雑な振舞いをする．そのような場合にも非負制約を課することにより，揺らぎのモデル化を抑制し，適切な基底関数選択ができるかを調べるため，x^2 に加えて並列処理の通信時間等のモデル化でよく使われる $\log_2(x)$を用いて真のモデル $y^0=x^2+\log_2(x)$に対してシミュレーションデータ $y_i = x_i^2+\log_2(x_i)+ d_i$ を作り，$f(x)= a_0+a_1x+a_2x^2+a_3x^3+a_4\log_2(x)+a_5\sqrt{x}$ (M=5)としてモデル化した．ここにシミュレーションデータ x_i，d_i は (1) と同様に生成した．

x_1=1.1 から x_8=3.1 という短い区間の $\log_2(x)$曲線は M=3 の多項式でもそれらしく近似できるが，外挿すると両者は全く異なる振舞いをする．HPC ではモデル化に $\log_2(x)$をしばしば使うので，$f(x)$のように多項式と $\log_2(x)$の両方をモデル化に用いた場合，観測データから両者の振舞いを分離できることが重要である．従って，予測モデルができるかということ以外に，a_4 が 1 に近い値になるということがこの実験の注目点である．

表 4-11 は LSM で得られたモデルパラメータで，真のモデルの $a_2= a_4$=1，$a_o=a_1= a_3= a_5= 0$ とは大きく異なる結果となる．また正負の推定値は揺らぎがモデル化されたことを示す．

表 4-11　LSM によるモデルパラメータ（σ=0.01）

パラメータ	推定値	標準誤差	t 値	p 値
a_o	-370.	290.	-1.28	0.330
a_1	-178.	143.	-1.24	0.34
a_2	15.8	12.2	1.29	0.327
a_3	-0.961	0.812	-1.18	0.358
a_4	-79.7	62.5	-1.28	0.330
a_5	534.	422.	1.27	0.333

LSM による回帰係数の推定についてモデルの適合度指標として，R^2 値= 0.999998，修正 R^2 値= 0.999993 であり， (1) の実験と同様に回帰係数 a_1 から a_5 の p 値がほぼ同様で

全てのパラメータは 0 とみなされない．また AIC に基づく変数選択で削除される項は無く，切片除去も含む変数選択において $f(x)=a_1x+a_2x^2+a_3x^3$ が AIC 最小であった．

同様の実験を 100 回行った際に，切片項の除外も含む変数選択で AIC 最小として選択されたモデルの個数について表 4-12 に示す．

表 4-12　100 回の変数選択により選択されたモデル

モデル	回数
$a_1x+a_4\log_2(x)+a_5\sqrt{x}$	26
$a_0+a_2x^2+a_3x^3+a_4\log_2(x)+a_5\sqrt{x}$	19
$a_1x+a_2x^2+a_3x^3+a_4\log_2(x)$	13
$a_1x+a_3x^3+a_4\log_2(x)+a_5\sqrt{x}$	13
$a_2x^2+a_3x^3+a_4\log_2(x)+a_5\sqrt{x}$	11
$a_1x+a_2x^2+a_3x^3+a_5\sqrt{x}$	8
その他	10

真のモデルは 100 回中 1 回も選択されなかった．

次に表 4-11 と同一のシミュレーションデータに対し，$\mathbf{a}\geq\mathbf{0}$ の制約を課して LP と NNLS による推定を行った結果を表 4-13 に示す．表は，これら 2 つの計算法の両方で a_2 と a_4 がほぼ 1 で選択されたことを示す． また a_0, a_1, a_5 が零となり，揺らぎをモデル化する過剰適合が抑制された予測モデルが構築できたことを示す．

表 4-13　モデルパラメータ（σ=0.01）

	a_0	a_1	a_2	a_3	a_4	a_5
LP	0	0	0.985	3.84×10^{-3}	1.02	0
NNLS	0.	0.	0.998	3.01×10^{-4}	1.00	0.

(1)と同様に，表 4-11 のシミュレーションデータと同じ方法で作った 5 組のシミュレーションデータに対する LP と NNLS の予測を図 4-26 に示す．破線は $y^0= x^2+\log_2(x)$ である．この図より，$\mathbf{a}\geq\mathbf{0}$ を用いるとモデルの予測力向上が図れることが確認できる．

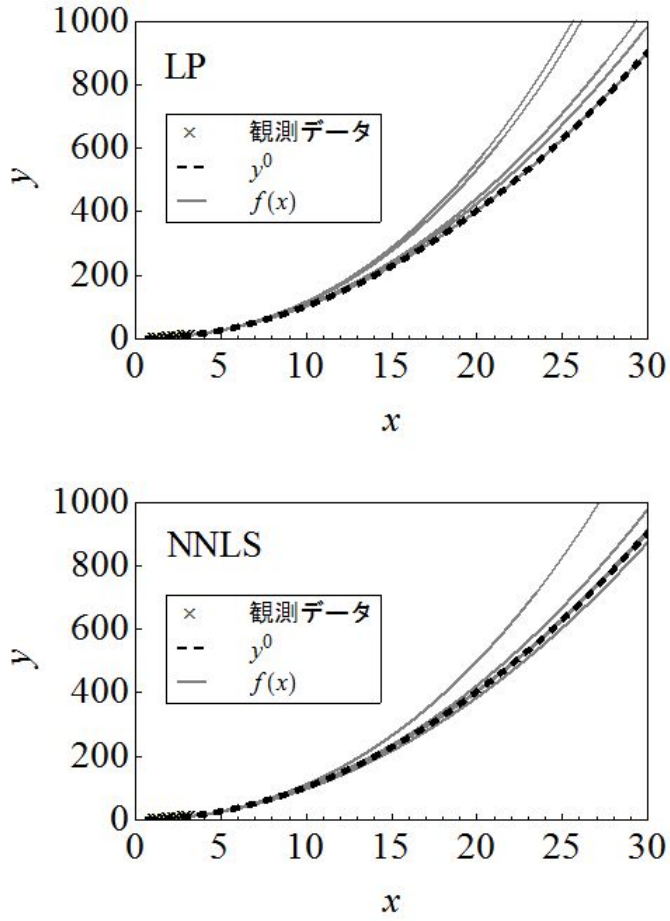


図 4-26 $x^2+\log_2(x)$ モデルの予測（上：LP，下：NNLS）

同じ y^0 と σ=0.01 に対して揺らぎが異なるシミュレーションデータを 100 ケース用意し，**a≥0** の効果を確認した．全ケースの x=5, 10, 20, 30 における平均値と標準偏差を表 4-14 においてして示す. 結果, シミュレーションデータに揺らぎがあっても LP と NNLS の両方で，シミュレーションデータ x_N=3.1 の約 10 倍の $y^0(30)$を大きく外れずに予測するモデル化ができることが確認できた.

表 4-14　100 ケースの予測値（σ=0.01）

X		5	10	20	30
$y^0(x)=x^2+\log_2(x)$		27.3	103.	404.	905.
LP	平均値	27.5	107.	451.	1076.
	(標準偏差)	(0.239)	(4.54)	(50.2)	(186.)
NNLS	平均値	27.5	107.	443.	1048.
	(標準偏差)	(0.211)	(3.95)	(43.5)	(161.)

表 4-15 に 100 ケースの各モデルパラメータが零になるケースの数を示す. x^2 と $\log_2(x)$ の a_2 と a_4 項の回数が全ケース 0 故, これらの項が σ=0.01 の揺らぎデータから確実に選択できたことが確認できる．また a_5 の回数もほぼ 100 で，$\sqrt{x}$ が残差を増加させる項であることが確認できる.

表 4-15　各モデルパラメータが零になる回数

	a_0	a_1	a_2	a_3	a_4	a_5
LP	71	65	0	30	0	99
NNLS	68	54	0	27	0	100

以上のことから，1 変数モデルのモデルパラメータに非負制約を課すことにより，揺らぎ起因の過剰適合が抑制できることがシミュレーションにより確認できた. またこの過剰適合抑制は LP と NNLS のどちらでも確認できたが，LSM の変数選択では必ずしも良いモデルは選択されなかった. このことから次節以降の実際の並列処理時間のモデル化において，非負制約付きの既存の計算法として提案手法に対して NNLS を比較対象とすることにした.

また変数選択の考え方としての比較として，真のモデルを選択する可能性もあるため，非負制約を課さない LSM による変数選択についても比較対象とすることにした.

4.3.2　2 変数モデルの場合

4.3.1 節の 2 つの例は 1 変数モデルであったが，2.3 節の冒頭で述べたように，並列処理の性能のスケーリングには少なくともプロセッサ数と問題の大きさの 2 変数から成る並列処理時間モデルが必要となる．そこで分子動力学プログラム [4-5] を並列計算機 IBM SP2 で実行した処理時間をこれら 2 変数でモデル化したモデル [4-13] を使い，LP によりモデル化した結果，過剰適合が緩和されて予測力が向上し，外挿ができる「予測モデル」が構築できることを確認した．一方比較のため行った LSM 変数選択と NNLS によるモデル化では，外挿に強い予測モデルの構築はできなかった．

並列処理時間のモデル化では，日々の並列処理の際に採取できる観測データからモデル化できることが望ましい．そこで観測データは，簡単に採取できるウォールクロックタイム（処理を投入してから終了するまでの時間）のみとした．次に日々の処理計算では同じ問題を解かないことから，問題の大きさが異なる観測データを用いるモデル化を想定した．またモデル化が必要な時は，並列処理するようになってから日が浅い場合が多いと考えられるので，観測データが少ないことを想定した．

以上の条件から観測データ $\mathbf{y}$ はプロセッサ数 p={2, 4, 6, 8, 10}と，問題の大きさ n={3200, 7200, 12800, 20000}という 2 つの変数の組み合わせで 20 通り測定したウォールクロックタイム中の 10 点を用いてモデル化した．また乱数を用いてこの 10 点の観測データの組み合わせを 5 ケース作り，任意の観測データから予測モデルが構築できるかを確認した．

(1) 処理時間モデルの基底関数の抽出

モデルの基底関数はプログラムから抽出した．モデル化した分子動力学プログラムは，ファンデルワールス力と重力の影響下で時間発展するアルゴン原子の巨視的振舞いをシミュレーションする分子動力学プログラムである [4-5]．

このような並列計算されるシミュレーションプログラムの処理時間をモデル化する場合，プログラムの全ての処理をモデル化するのではなく，処理時間の大部分を費やすカーネルを見つけてその部分の処理を式化し，それらをアッセンブルしてプログラムの処理時間モデルを構築する方法が一般的である．そこで時間全体の 99%の時間を占めるサブルーチンの処理時間を式化した[4-12]．具体的には，サブルーチン中のカーネル(do ループ，MPI 通信部)の処理時間を，問題の大きさ n の関数であるループ回転数に比例するものとし，次にその処理時間はプロセッサ数 p に反比例するとして式化した．

表 4-16 にサブルーチン毎の処理時間のモデルを示す．表で示したように，時間全体の 99%の時間を占めるサブルーチンは 5 つあり，そのうち 3 つのサブルーチンで並列処理が行われる．T_{force} は粒子間力の計算時間である，T_{tabe} は遮蔽距離内に在る粒子の表を作る，T_{propcel} は粒子の物理データをメッシュに落として物理量を計算する，T_{cross} は粒子の境界での反射を処理し，T_{moves} では粒子の移動を処理する．表中 I_{s} は時間の繰り返し数，N_{sample} は観測の回数，そして n_{table} は時間積分の繰り返しの何回目に遮蔽距離内に在る粒子の表を更新するかを決める間隔である．t_{u} はカーネルの逐次時間の計算時間，$t_{0\mathrm{u}}$ はその立ち上がり時間，a_{u} は論理性能で r_{a} により規格化した効率である．表 4-16 中のσは MPI_ALLTOALL で行った総和計算をモデル化した通信時間であり，$t_{0\mathrm{c}}$ はその立ち上がり時間，e_{c} は通信の論理性能 r_{c} で規格化した効率である．また各プロセッサの総和計算時間は $t_{0\mathrm{sum}}$ とした．t_p は t_{u} と σ の和からサブルーチンの並列処理時間で，$t_{0\mathrm{p}}$ は立ち上がり時間，e_{p} はカーネルの並列処理の効率である．

表 4-16 ホットスポットカーネルの処理時間モデル [4-12)]

$T_{force} = n_{sample} I_s t_p$ $t_u = t_{0u270} + n\left(t_{0u271} + \frac{n}{2} C_1 \frac{5+13C_2}{r_a \cdot a_{u271}}\right)$ $\sigma_{270} = 2n\left(2\left(t_{0c270} p + \frac{X_8}{r_c e_{c270}}\right) + t_{0sum270}\right)$ $t_p = t_{0p270} n + \frac{t_u}{pe_{p270}} + \sigma_{270}$	$T_{table} = n_{sample} I_s / n_{table} t_p$ $t_u = t_{0u101} + n\left(t_{0u100} + \frac{n}{2} \frac{5+C_1}{r_a a_{u100}}\right)$ $t_p = t_{0p100} n^2 + \frac{t_u}{pe_{p101}}$
$T_{prepcel} = n_{sample} I_s t_p$ $t_{u302} = t_{0u302} + \frac{10n}{r_a a_{u302}}$ $t_{u303} = t_{0u303} + \frac{7n}{r_a a_{u303}}$ $t_{p302} = t_{0p302} + t_{u302} / (pe_{p302})$ $t_{p303} = t_{0p303} + t_{u303} / (pe_{p303})$ $t_p = t_{p302} + t_{p303} + \sigma_{303}$ $\sigma_{303} = n_x n_y \left(2\left(t_{0c303} p + \frac{X_8}{r_c e_{c303}}\right) + t_{0sum303}\right)$	$T_{pcross} = n_{sample} I_s t_u$, $t_u = t_{0u10} + \frac{10n}{r_a a_{u10}}$ $T_{moves} = n_{sample} I_s t_u$, $t_u = t_{0u300} + \frac{9n}{r_a a_{u300}}$

α_{cut}=4, r_{cut}=3, n_{sample}=6, I_s=2000, n_{table}=14.16・α_{cut}・r_{cut},n_x=40, n_y=20, X_8 =8, X_4=4, C_1=π・(α_{cut}・r_{cut})^2/高さ×幅=π・(α_{cut}・r_{cut})^2/(2.5・n), C_2=1/α_{cut}, C_3=1/I_s, C_4=0.5 (x 方向で粒子速度が正の確率), C_5=0.5 (y 方向で粒子速度が正の確率), C_6=~1 (境界に接しない粒子の確率)

これらの処理時間の総和 T_{MD} (=T_{force} + T_{tabe} + $T_{propcel}$ + T_{cross} + T_{moves})に対し，変数 p と n で式をまとめると，プログラム全体の処理時間を式(4-7)のように，8 つのモデルパラメータを持つ 2 変数 p, n のモデルとして記述することができる．

モデルパラメータと式化したカーネルモデルの係数の関係を表 4-17 に示す．各パラメータは主に立ち上がり時間 t_0 に関するもの，理論性能に対する効率である r_a, r_c, e_p に関するもの及び，立ち上がり時間とこれら割合の混合から成る．これら立ち上がり時間と効率は原理的には正であると考えられるので，表 4-17 の 1 列目に示したモデルパラメータ,即ち式(4-7)のモデルパラメータは全て非負値と想定し，2.3.3 節で提案した計算

法(LP)を適用した.

$$T_{MD}(p,n) = a_0 + a_1 n^2 + a_2 \frac{n^2}{p} + a_3 n + a_4 \frac{n}{p} + a_5 np + \frac{a_6}{p} + a_7 p \qquad (4\text{-}7)$$

表 4-17 モデルパラメータとカーネル係数の関係

(× $n_{sample} I_s$　秒)

a_0	$t_{0p302} + t_{0p303} + t_{0u10} + t_{0u300} + n_x n_y t_{0sum303} + \frac{16 n_x n_y}{r_c e_{c303}}$
a_1	$\frac{1}{n_{table}}\left(t_{0p101}\right)$
a_2	$\frac{1}{n_{table}}\left(\frac{5}{2 a_{u100} r_a e_{p101}}\right)$
a_3	$t_{0p270} + 2t_{0sum270} + \frac{10}{a_{u10} r_a} + \frac{9}{a_{u300} r_a} + \frac{32}{r_c e_{c270}}$
a_4	$\frac{1}{n_{table}}\left(\frac{t_{0u100} + \frac{90.48}{a_{u100} r_a}}{e_{p101}}\right) + \frac{1}{r_a}\left(\frac{10}{a_{u302} e_{p302}} + \frac{7}{a_{u303} e_{p303}}\right) + \frac{t_{0u271} + \frac{525.9}{a_{u271} r_a}}{e_{p270}}$
a_5	$4t_{0c270}$
a_6	$\frac{1}{n_{table}}\left(\frac{t_{0u101}}{e_{p101}}\right) + \frac{t_{0u270}}{e_{p270}} + \frac{t_{0u302}}{e_{p302}} + \frac{t_{0u303}}{e_{p303}}$
a_7	$2t_{0c303}$

(2) LP によるモデル化と予測力向上

式(4-7)のモデルパラメータを $\mathbf{a} \geq \mathbf{0}$ とし，$p \rightarrow x_{1i}$，$n \rightarrow x_{2i}$，$\mathbf{y} \rightarrow y_i$，$T_{MD}(p,n), f(x_{1i}, x_{2i})$として図 2-8 を適用して，モデルパラメータを推定した．表 4-18 の No.1～5 に冒頭で述べたように乱数で観測データを 5 ケース選んだ推定値を示す．表で値が 0 の項は，自動的に削除された過剰適合項である．全ケースで零となる a_6 はどの観測データもモデル化にこの項を必要としないことを示す.

表 4-18　LP により決定した式(4-7)のモデルパラメータ

No.	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
1	**0**	**0**	$3.11\cdot10^{-6}$	$3.54\cdot10^{-2}$	0.307	$6.55\cdot10^{-4}$	**0**	3.53
2	**0**	**0**	$3.77\cdot10^{-6}$	$3.75\cdot10^{-2}$	0.296	$4.95\cdot10^{-4}$	**0**	3.56
3	**0**	**0**	$3.52\cdot10^{-6}$	$4.11\cdot10^{-2}$	0.293	$1.52\cdot10^{-4}$	**0**	3.21
4	**0**	**0**	$3.01\cdot10^{-6}$	$4.00\cdot10^{-2}$	0.301	**0**	**0**	3.64
5	4.34	$2.91\cdot10^{-7}$	$1.51\cdot10^{-6}$	$4.17\cdot10^{-2}$	0.302	**0**	**0**	1.21

この表のモデルパラメータを用いて問題の大きさ n=96800 の処理時間の予測を確認した結果を図 4-27 に示す．図はプロセッサ数 p に対する経過時間 T_{MD} である．大きな×印は予測確認用の測定値である．実線が LP を用いた 5 ケースのモデルの予測で，データの組み合わせが変わっても×印の値と傾向が予測できた. 図の小さい×印はモデル化に用いた観測データで，用いたプロセッサ数の最大値は p =10，問題の規模の最大値は n=20000 である．大きな×印の確認データのプロセッサ数の最大値は p =48，問題の規模の最大値は n=96800 であるので, 図は p と n で各々約 5(~48/10)倍，5(~96800/20000)倍の領域を予測できたことを示す.

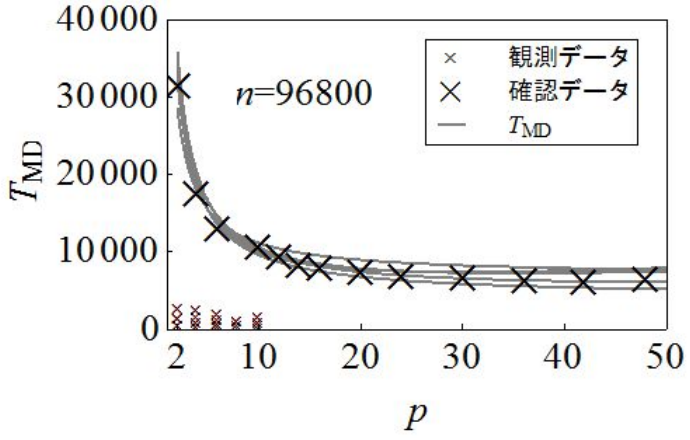


図 4-27　LP で構築した MD 処理時間モデルの予測

表 4-19 は LSM により AIC に基づいて変数選択した結果推定されたモデルパラメータである．表中「削除」と書かれているモデルパラメータは，変数選択で削除されたことを示す．表は入力データにより削除される項が異なることを示す．また負の値を持つモデルパラメータが存在していることから，基底関数の合成により揺らぎ等のノイズをモデル化する現象が生じていることがわかる．この表のモデルパラメータを用いた予測を図 4-28 に示す．入力データは表 4-18 と同じである．図中の実線が予測で，プロセッサ数の増加を全く予測できない場合が 1 ケースあることを示す．

表 4-19　LSM 変数選択により推定されたモデルパラメータ

No.	a_0 (切片)	a_1	a_2	a_3	a_4	a_5	a_6	a_7
1	**削除**	**削除**	2.13×10^{-6}	3.55×10^{-2}	0.328	9.00×10^{-4}	-93.1	**削除**
2	74.6	-13.3×10^{-7}	11.6×10^{-6}	3.18×10^{-2}	0.244	2.87×10^{-3}	**削除**	-8.56
3	72.7	-5.08×10^{-7}	6.29×10^{-6}	3.93×10^{-2}	0.264	1.46×10^{-3}	**削除**	-10.9
4	48.8	**削除**	2.83×10^{-6}	2.50×10^{-2}	0.334	1.93×10^{-3}	15.8	**削除**
5	-594.	-11.5×10^{-7}	9.02×10^{-6}	12.2×10^{-2}	**削除**	-4.06×10^{-3}	1918.	40.4

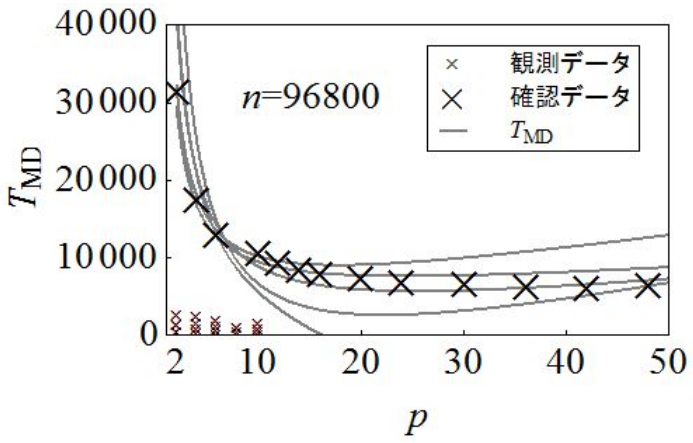


図 4-28　LSM で AIC に基づく変数選択により構築した
MD 処理時間モデルの予測

LP と同じ観測データと式(4-7)に対して $\mathbf{a}\geq 0$ の条件を附与した NNLS でモデルパラメータを決定すると，5 ケースの番号順に a_1={6.50×10^{-6}, 5.42×10^{-6}, 6.01×10^{-6}, 3.34×10^{-6}, 3.90×10^{-6}}，他のモデルパラメータは全て 0.となった．結果，得られたモデルは図 4-29 の実線となり，予測モデルを作ることができなかった．

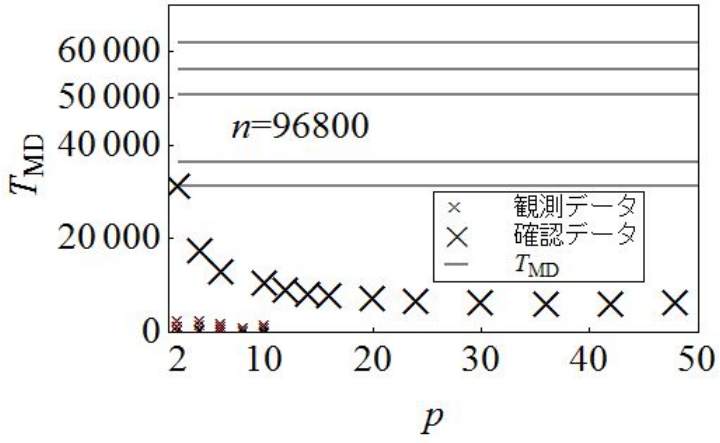


図 4-29　NNLS で構築した MD 処理時間モデルの予測

この原因を探るため，LP，NNLS の残差平方和 E を式(2-36)で決定して比較した．これを表 4-20 に示す．全ての観測データにおいて $E_{NNLS} >> E_{LP}$ であり，NNLS の残差平方和の最小化が十分でないことがわかる．

表 4-20　MD コードの RSS

No.	*1*	*2*	*3*	*4*	*5*
LP	1862.	787.	972.	2289.	784.
NNLS (×10^6)	5.66	3.09	5.44	11.3	3.26

以上のことから，2 変数モデルの予測力向上を図るのに， LP が有効である場合があることが確認できた．残差平方和の桁が 2−3 桁異なることから予測誤差について外挿時に数桁の予測誤差の違いとなるため，LP が外挿時の予測精度が高いことがわかる．

4.3.3　3 変数モデルの場合

4.3.2 節では 2 変数モデル構築において有理演算 LP を用いた提案方法が並列処理時間の予測モデル構築に有効であることを確認したが, 並列処理時間モデルではモデルが 3 変数になる場合もある．そこで提案したモデル化方法を，3 変数モデルとしてポピュラーな密行列式を解く性能ベンチマークコード High Performance Linpack[4-13] (以後 HPL) を並列計算機 FUJITSU HPC2500[4-17]で実行した処理時間をモデル化した.

(1) 処理時間モデル

HPL は次元数 N の密行列計算を $P \times Q$ の 2 次元プロセス格子に分割し，並列処理するプログラムである. その処理時間モデルは計算量が処理時間に比例すると仮定して得られる [4-13]．このとき panel factorization と trailing sub matrix 計算で生じる通信のモデルパラメータが異なることを想定して各々の通信時間を $t_{cf}= \alpha_f+ \beta_f L$，$t_{cm}= \alpha_m + \beta_m L$ とする．ここに L は通信量である．次に HPC2500 がクロスバーネットワークで通信することを考慮し，立ち上がり時間 α_f, と α_m が P に比例すると仮定して $\alpha=aP+b$ として処理時間を求めると，処理時間モデルは式(4-8)のように P, Q, N の 3 変数 7 パラメータモデルとなる.

$$
\begin{aligned}
&T_{HPL}(P,Q,N) = \\
&\gamma\frac{2N^3}{3PQ} + \beta_f\frac{N^2}{2P} + \beta_m\frac{3N^2}{2Q} \\
&+ a_f P\frac{N\left(1+NB\log_2(P)\right)}{NB} + a_m P\frac{N\left(P-1+\log_2(P)\right)}{NB} \\
&+ b_f\frac{N\left(1+NB\log_2(P)\right)}{NB} + b_m\frac{N\left(P-1+\log_2(P)\right)}{NB}
\end{aligned}
\qquad (4\text{-}8)
$$

ここに NB はプロセス格子のブロックサイズ．γ，β_f，β_m，a_f，a_m，b_f，b_m がモデルパラメータである.

観測データであるウォールクロックタイム $\mathbf{y}$ は前節と同様になるべく少なくし，NB=100 に対して問題の大きさ N={10000, 15000, 20000, 25000, 30000, 35000}とプロセッサ数 Q={10, 20, 30, 40}及び P=1 の組み合わせからランダムに選んだ 4 点，N とプロセッサ数 P={10, 20, 30, 40}及び Q=1 の組み合わせからランダムに選んだ 4 点と全組み合

わせの中から重複しない 1 点の計 9 点を 1 ケースとし, 前節と同様 5 ケースの入力を用意した．同じ問題を 2 度計算しないという観点から，ランダムに選ぶ N と Q および N と P の各組み合わせにおいて，N が重複する組み合わせを避けた．提案方法の有効性を調べるため，LSM 変数選択と NNLS で同様のモデル化を行い比較した．

⑵ LP によるモデル化と予測力向上

式(4-8)と図 2-8 のタームを $P \rightarrow x_{1i}$，$Q \rightarrow x_{2i}$，$N \rightarrow x_{3i}$，$\mathbf{y} \rightarrow y_i$，$T_{HPL3D}(P, Q, N) \rightarrow f(x_{1i}, x_{2i}, x_{3i})$ と対応付けて，推定したモデルパラメータを表 4-21 の No.1～5 に示す．なお図中のモデルパラメータとは$\gamma \rightarrow a_1$, $a_2 \rightarrow \beta_f$, $a_3 \rightarrow \beta_m$, $a_4 \rightarrow a_f$, $a_5 \rightarrow a_m$, $a_6 \rightarrow b_f$, $a_7 \rightarrow b_m$ と対応する．表で値が零の項は，過剰適合項が自動的に削除されたことを示す．5 ケースの γ，β_f，β_m の値はほぼ同じで，5 ケースの観測データに対してパラメータ決定が安定して行われていることが確認できる．a_m，b_f，b_m に零値が出現し，これらの項がその観測データのモデル化に必要ないことがわかる．

表 4-21　LP によりされた式(4-8)のモデルパラメータ

No.	γ	β_f	β_m	a_f	a_m	b_f	b_m
1	$4.10 \cdot 10^{-10}$	$2.61 \cdot 10^{-7}$	$1.23 \cdot 10^{-7}$	$2.25 \cdot 10^{-5}$	**0**	**0**	**0**
2	$4.08 \cdot 10^{-10}$	$2.73 \cdot 10^{-7}$	$1.30 \cdot 10^{-7}$	$2.26 \cdot 10^{-5}$	**0**	**0**	**0**
3	$4.17 \cdot 10^{-10}$	$2.53 \cdot 10^{-7}$	$1.10 \cdot 10^{-7}$	**0**	$2.38 \cdot 10^{-4}$	$2.18 \cdot 10^{-4}$	**0**
4	$4.22 \cdot 10^{-10}$	$2.45 \cdot 10^{-7}$	$0.866 \cdot 10^{-7}$	$1.08 \cdot 10^{-5}$	**0**	$1.36 \cdot 10^{-4}$	$7.96 \cdot 10^{-3}$
5	$4.23 \cdot 10^{-10}$	$2.35 \cdot 10^{-7}$	$1.26 \cdot 10^{-7}$	$1.97 \cdot 10^{-5}$	$0.178 \cdot 10^{-4}$	**0**	**0**

図 4-30 の実線は式(4-8)と表 4-21 のモデルパラメータを用いたモデルの確認テストである．大きな×印は確認のための測定値である．小さい×印は入力データである．P=1, N=80000 のケースでは問題の大きさにして約 12(=80000/35000)倍の領域で P=40~120 の予測ができることが確認できる．Q=1，N=45000 の場合では，問題の大きさにして約

2(=$45000^3/35000^3$)倍の領域でT_{HPL}が最小値になるまでの予測がほぼできることが確認できる.

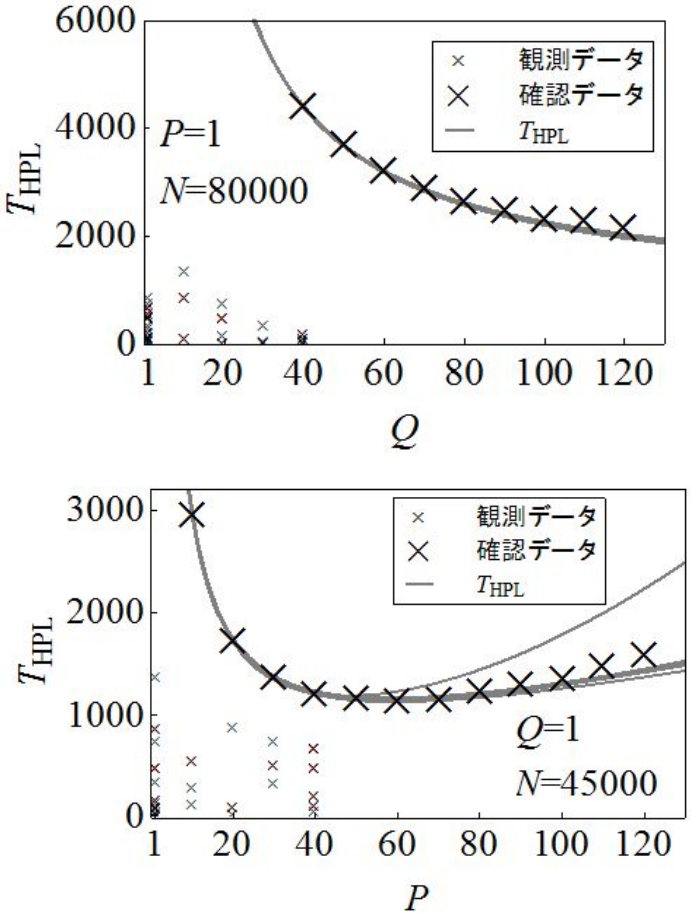


図 4-30　LP で構築した HPL 処理時間モデルの確認

結果，3 変数 7 パラメータの並列処理時間の予測モデルがウォールクロックタイムのみの入力データから構築できることが確認できた.

前節と同様に AIC による変数選択で決定したモデルパラメータを表 4-22 に，予測の確認を図 4-31 に示す．観測データは表 4-21 と同じである．図中の実線は式(4-8)の予測で，同上図 P=1，N=80000 では全入力データの予測が大きい×の確認データと一致していることを示す．一方同下図の Q=1，N=45000 に対しては，1 つの観測データで確認データと全く一致せず，残りの 4 つのデータも確認データの傾向を捉えていない

ことを示す.

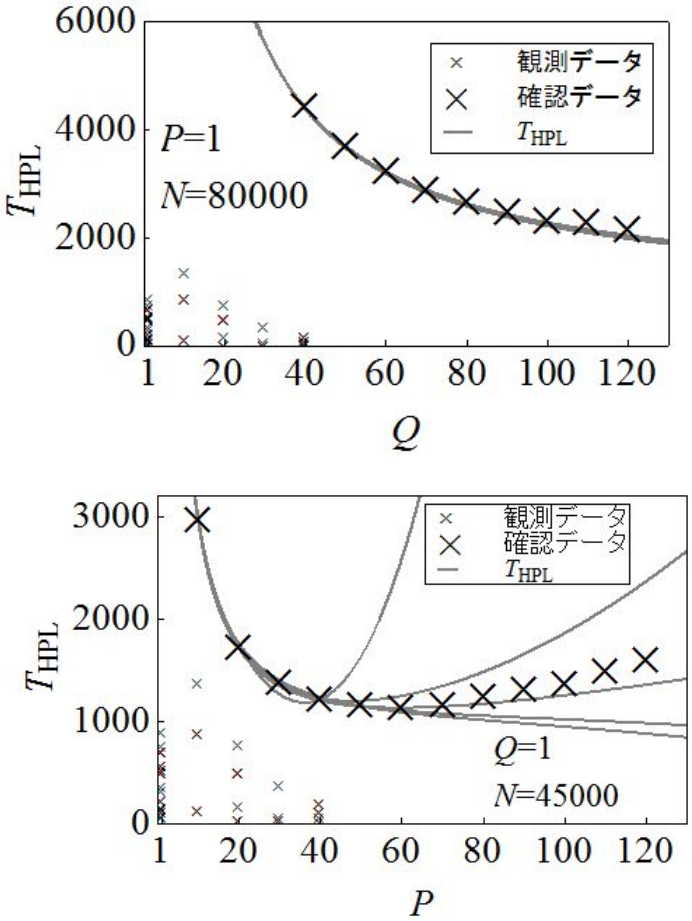


図 4-31　LSM で AIC に基づく変数選択により構築した
HPL 処理時間モデルの予

表 4-22　LSM で AIC に基づく変数選択により推定されたモデルパラメータ

No.	切片	γ	β_f	β_m	a_f	a_m	b_f	b_m
1	3.23	4.10×10^{-10}	2.83×10^{-7}	1.26×10^{-7}	-4.29×10^{-3}	1.10×10^{-2}	-3.86×10^{-2}	2.10
2	0.660	4.20×10^{-10}	2.43×10^{-7}	1.25×10^{-7}	-1.08×10^{-4}	**削除**	-2.10×10^{-3}	0.0876
3	0.121	4.14×10^{-10}	2.54×10^{-7}	1.15×10^{-7}	-2.53×10^{-5}	3.09×10^{-4}	**削除**	0.0115
4	1.03	4.23×10^{-10}	2.42×10^{-7}	0.790×10^{-7}	**削除**	**削除**	-4.66×10^{-5}	0.0162
5	0.202	4.19×10^{-10}	2.46×10^{-7}	1.22×10^{-7}	-1.01×10^{-4}	**削除**	-2.18×10^{-3}	0.0855

モデルパラメータを NNLS で計算した結果，データ番号順に γ={5.94×10^{-10}, 5.79×10^{-10}, 6.51×10^{-10}, 5.33×10^{-10}, 6.14×10^{-10}}，その他のモデルパラメータは全て零となった．図 4-32 はモデルが P=1 の場合の予測ができることを示す．一方 Q=1 で P の変化に対して凹となる現象がモデル化できなかったことを示す．

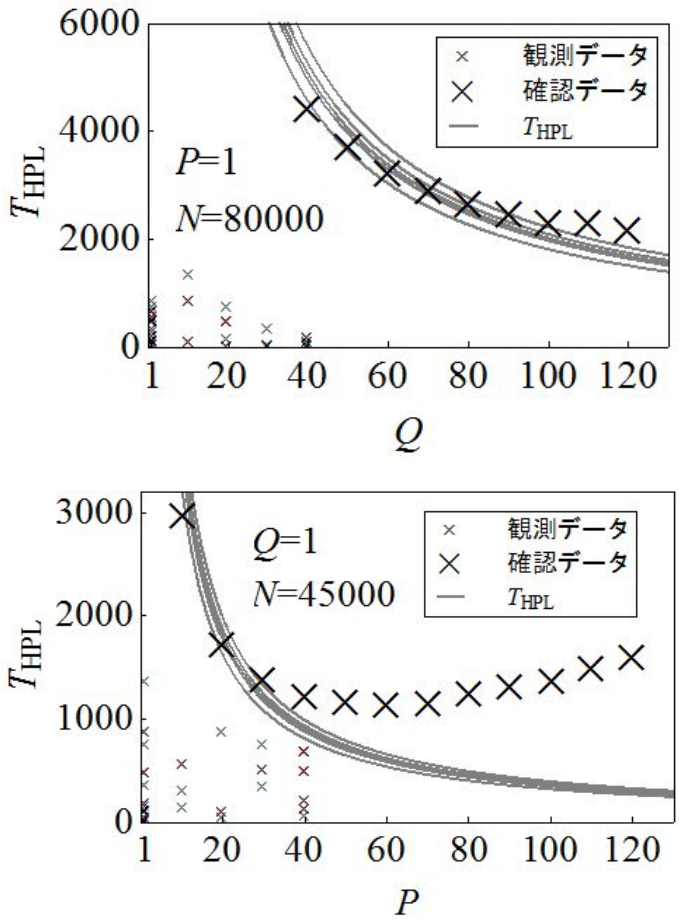


図 4-32 NNLS で構築した HPL 処理時間モデルの予測

表 4-23 に LP と NNLS の残差平方和 E を示す．LP の E は 10^0~10^2 のオーダーで，NNLS は 10^5 のオーダーである．これらのことから 4.3.2 節と同様，LP の外挿時の予測精度が高いことがわかる．

以上のことから 3 変数モデルでは，LP でのみよい予測モデル構築が可能であることが確認できた．

表 4-23　HPL モデルの残差平方和 E

No.	*1*	*2*	*3*	*4*	*5*
LP	4.53	275.	4.02	5.95	27.5
NNLS($\times 10^5$)	0.429	1.39	1.28	1.92	2.12

4.3.4　議論

シンプレックス法を有理数演算するメリットを考える．4.3.3 節の事例を有理数演算と浮動小数点演算で行い比べると，5 ケース全てで値が異なり，4 ケースで異なった変数選択となった．例えば **a** の要素を$\{\varepsilon, \gamma, \beta_f, \beta_m, a_f, a_m, b_f, b_m\}$とすると，ケース 1 は，有理数演算で$\{0.843, 4.10\times10^{-10}, 2.61\times10^{-7}, 1.23\times10^{-7}, 2.25\times10^{-5}, 0, 0, 0\}$，浮動小数点演算で$\{0., 4.16\times10^{-10}, 2.51\times10^{-7}, 1.16\times10^{-7}, 2.02\times10^{-5}, 3.38\times10^{-5}, 0, 0\}$となる．この問題は解法に改訂シンプレックス法を用いることにより解決でき，浮動小数点で表した有理数演算のシンプレックス法の結果と比較すると有効数字 6 桁で一致する．この差異は有理数演算のシンプレックス法と改訂シンプレックス法では生じなかった．また適当なピポットが見つからない等，シンプレックス法が解を決定できない場合を除くと，有理数演算の結果はモデル化に用いた観測データに対する厳密解である．従ってモデルパラメータ決定に利用した計算法の計算誤差を，他の計算法ほど気にしないで済むものと考える．

提案した計算法 LP ではモデルパラメータに非負制約が附与できることが前提である．もし非負制約が適用できないモデル化に LP を適用すると，例えば真のモデル $y^0= -0.5+x^2$ に対して揺らぎ d_i (i=1,…,N)を加え，シミュレーションデータ $y_i = -0.5 + x_i^2 + d_i$ (i=1,…,8)を生成しモデル化を行うと，　$f(x)=a_0+a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5$ に対しては　a_0=0, a_1=0, a_2=0.613, a_3=0.113, a_4=0, a_5=0 となり，モデル化に必要な a_0 項は過剰適合項として削除され，真のモデル y^0 とは異なるモデルが得られる．

並列処理時間のモデル化においても，表 4-16 のように個々のカーネルをモデル化すると，モデルパラメータの一つ立ち上がり時間 t_{0u} が負になる場合がある [4-12]．問題の大

きさを変えた時キャッシュの効果が変わることを考えると，複雑な現象を線形近似でモデル化したためにそのしわ寄せが t_{0u} に生じたと推察する．これを確かめるには非負や非正，或いはそれらの条件が混在したモデルパラメータ決定が必要と思われる．条件の混在は限量記号消去法(QE)[4-14] により可能であるが，非負で抑制した観測データの揺らぎのモデル化が復活する可能性があり，今後の研究課題である．

4.3.5　まとめと今後の課題

並列処理時間の回帰モデルの予測力向上のため，モデルパラメータに非負の条件を付与することで観測の揺らぎがモデル化される過剰適合の抑制を試みた．モデルパラメータを決定し，非負制約を課したことにより過剰適合項となった基底関数のモデルパラメータを零値にするために，シンプレックス法を用いた計算法(LP)を提案した．また LP の計算誤差を抑制するために有理数演算にすることを提案した．適用事例から次の効果を確認した．

1) 非負制約をモデルパラメータに課すことにより，複数の基底関数の重ね合わせで揺らぎがモデル化される過剰適合が抑制でき，モデルの予測力が向上する．
2) 非負制約を課すことにより残差を増やすことしかできなくなった基底関数のモデルパラメータが零になり，過剰適合を起す基底関数の幾つかを抑制したモデルが自動的に構築できる．
3) 説明変数が 1 変数のモデル化では，LP と既存の非負制約付き計算法である非負最小二乗法(NNLS)で上記 1)，2)の効果が確認できた．一方 LSM 変数選択では確認できなかった．
4) 説明変数が 2，3 変数のモデル化では，各々5 ケースの観測データを用いた LP ，LSM 変数選択，NNLS の比較では，LP でのみで 1)，2)の効果が観測できた．また異なる 5 つの観測データの組み合わせで予測確認データの値とほぼ一致する予

測モデルが構築できた.

以上のことから並列処理時間の「予測モデル」構築に LP が有効であると考える. ウォールクロックタイムを観測データとし，少ないデータで「予測モデル」を構築できることが期待できる提案した計算法 LP が，多くの並列計算機とそれに関わる資源の有効利用に役立つものと期待する.

参考文献

4-1) 折居茂夫，数値計算のための並列計算機性能評価方法，情報処理学会論文誌，39, (3) (1998) 529–541.

4-2) 折居茂夫，高並列処理におけるスケーラビリティ評価方法（Ⅱ），情報処理学会研究報告, HPC-126 (48) (2010).

4-3) 折居茂夫，時間モデルを用いた並列処理の性能評価 ― 並列化部に隠れた並列オーバーヘッド ―, HPC-130 (1) (2011).

4-4) 折居茂夫，山本義郎，シンプレックス法を用いた非負のモデルパラメータを持つ並列処理時間モデルの予測力向上，情報処理学会論文誌 56, (6) (2015) 1481–1495.

4-5) T. Watanabe and H. Kaburaki, Increase in chaotic motions of atoms in a large-scale self-organized motion, Phys, Rev. E 54 (1996) 1504–1509.

4-6) 折居茂夫，並列処理数値シミュレーションのためのスケーラビリティ検討方法，情報処理学会研究報告 HPC58-5 (1995) 27–32.

4-7) 折居茂夫，分子動力学コードの段階的並列化手法, JAERI-Data/Code 96-023 (1996).

4-8) 大田敏郎，疎結合スカラ並列計算機のグローバル総和の高速化, JAERI-Data/Code 96–009 (1996).

4-9) Mathematica, http://reference.wolfram.com/language/

4-10) R -The Comprehensive R Archive Network-, http://cran.r-project.org/

4-11) 樺島祥介，北川源四郎，甘利俊一，赤池弘次，下平英寿：赤池情報量規準 AIC―モデリング・予測・知識発見，共立出版 (2007)

4-12) 折居茂夫：レベル 1･2 並列ベンチマーク使用及びそれに基づくスカラ並列計算機 SP2 のベンチマークテスト，JAERI-Data/Code 98-020 (1998).

4-13) A. Petitet, R. C. Whaley, J. Dongarra and A. Cleary: HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers.

http://www.netlib.org/benchmark/hpl/

4-14) 折居茂夫，山本義郎：限量記号消去法を用いた回帰モデルの予測力向上，情報処理学会研究報告, 2013-HPC-139, No.1 (2013).

4-15) C.L. Lawson and R.J. Hanson, Solving Least Squares Problems, Prentice-Hall, Chapter 23 (1974).

4-16) MATLAB, http://www.mathworks.co.jp/jp/help/matlab/ref/lsqnonneg.html

4-17) Y. Matsuo and M. Tsuchiya, M. Aoki, N. Sueyasu, T. Inari, K. Yazawa, Early Experience with Aerospace CFD at JAXA on the Fujitsu PRIMEPOWER HPC2500, In proceedings of SC2004.

5章　終章

並列処理は電子計算機の歴史において早くから高速化手法として研究されたが, 1967年に, 並列処理の性能が並列処理できない部分の時間が処理全体に占める割合で決まることがAmdahlにより指摘されその実現性の問題点が指摘された．この問題がプロセッサ数の増加と共に問題の大きさを大きくすれば解決できることをGustafsonが示したのは1988年であった．この頃から半導体の性能向上の限界が叫ばれ始め，1990年台に並列処理はスーパーコンピューティングを実現する主役となり, 2015年現在ではスーパーコンピューティングを実現する唯一の手法となっている．

大きな問題を計算することでAmdahlの指摘は軽減されたが一方，並列処理に起因して出現しプロセッサ数と共に増加する通信等のその他の並列オーバーヘッドは, 現在も並列処理の性能を妨げる大きなハザードとして存在する. 並列処理を性能の低い状態で行うと処理に投入した計算機資源が無駄になる．例えば並列効率50%の処理では半分の計算機資源が無駄になる勘定となる．かつて性能世界一だった「京」スーパーコンピュータは88,128個のCPUで構成されるが，これを全て使った並列処理の並列効率50%とすると44,064個のCPUが無駄になっていることになる．1CPUの消費電力は2011年11月TOP500リストの測定時のデータでは144W/CPU(~12.66MW/705024/8)なので44,064個に換算すると6.3MWの電力が無駄に消費されることになる．

並列処理の性能評価技術は意図した性能を得るための手段である他に, このような計算機資源の浪費を避ける手段としても存在する．一方これらを実現しようとすると，並列計算機で処理されるほぼ全ての並列処理の性能評価を行う必要が生じる. ところが従来の性能評価指標で最も頻繁に使われる並列効率は逐次処理（1プロセッサ）の処理時間を基準にして決定される．従ってこの指標の決定には2回の測定が必要である．故に従来の並列効率は, 計算機システムで処理されるジョブ全てに対して決定できる指標ではない. このため全てのジョブの性能を監視し計算機資源の浪費を削減する目的には使

用できなかった．また高並列処理を逐次処理することは不可能である．これが本論文の第1の課題である．

Gustafsonが述べたようにプロセッサ数の増加と共に問題の大きさを大きくすることによりAmdahlの指摘は回避できるが，プロセッサ数の増加と共に並列オーバーヘッドは増加する．従ってプロセッサ数の増加したときの処理時間と並列効率を予測することが必要となる．これまでこれに対して全ての並列処理に適用できる予測技術はまだ確立されていない．これが本論文の第2の課題である．

2章では上記第1の課題の解決方法を提案した．また第2の課題を解決する2つの方法と，それらに付随し予測モデル構築のハザードである，モデルパラメータ推定で生じる過剰適合を抑制してモデルの予測力を向上する方法を提案した．3章では第1の課題の解決方法の事例を2つ示した．4章では第2の課題の解決方法の事例をホワイトボックスアプローチと，ブラックボックスアプローチという2つの切り口から示した．またモデルパラメータに非負条件を付けると，過剰適合を自動的に抑制してモデルの予測精度が向上することを示した．また従来の数値計算法ではうまくモデルパラメータを決定できない2変数，3変数の場合において予測モデルが構築できることを示した．

このように並列処理の性能情報を飛躍的に増大するポテンシャルを持つ上記2つの課題に対する提案が，並列処理による高速計算の進化の一助となれば幸いである．

謝辞

本論文博士活動をキックオフくださいました，元東海大学理学部長　南里　憲三氏に感謝いたします．南里先生から本件を引き継ぎ，学位取得活動をコーディネートくださいました東海大学大学院　元総合理工学研究科　研究科長，現研究推進部　部長で産官学連携センター所長の山口　滋氏に感謝いたします．本論文主査の東海大学理学部数学科教授　山本　義郎氏には，計算機科学，統計学等がクロスオーバーした「モデルの予測力向上の研究」において，論文となるまでご指導いただきました．感謝いたします．論文審査委員の山口先生，山本先生，東海大学理学部物理学科教授の新屋敷　直木氏，東海大学工学部動力機械工学科教授の高倉　葉子氏，学外論文審査委員の富士通研究所主管研究員，国立情報学研究所　客員教授，九州大学マス・フォア・インダストリ研究所　訪問教授の穴井　宏和氏に感謝いたします．穴井先生には，本論文の「モデルの予測力向上の研究」で提案した計算法「LP」の研究を始めるきっかけになった，数式処理による計算法をご指導いただきました．感謝いたします．旧日本原子力研究所　計算科学技術推進センター長の浅井　清氏，グループリーダの相川　裕史氏の下で，ホワイトボックスアプローチによるモデル化方法の研究ができました．感謝いたします．また筆者富士通在職中には，テクニカルコンピューティング・ソリューション事業本部の奥田基氏，北畠　秀俊氏，樋口　哲二氏のご配慮で研究を続けることができましたのでここに感謝いたします．最後に長期間に渡る学位論文活動を常に期待し励ましてくれた母折居　せつ　に感謝いたします．