Vol.2010-HPC-126 No.44
2010/8/5

高並列処理における並列性能評価方法（Ⅱ）

折居茂夫†

測定値から並列処理時間モデルを構築する方法を提案する．この方法により構築した処理時間モデルにより，Strong scaling と Weak scaling に対する並列性能評価を行うことができる．分子動力学コードと IBM SP2 から成るシステムに対して処理時間モデルを構築し，そのモデルを用いて並列性能評価を行った結果を報告する．

Method of Parallel Performance Evaluation for Highly Parallel Processing (Ⅱ)

Shigeo Orii†

The author has proposed a modeling method of parallel processing time based on measured data. A derived processing time model with the modeling method has made possible parallel performance evaluation for both of Weak and Strong scaling. This paper also reports a study on parallel performance evaluation for a system consisting of a molecular dynamics code and IBM SP2 using a processing time model based on the modeling method.

1. はじめに

ペタフロプスの計算能力を持つ数万台規模のプロセッサを持つ計算機が稼動する現在，高並列処理の並列性能評価方法の研究は，巨大な資源の有効利用という観点から益々重要になる．

その一方法として処理時間をプロセッサ数と問題の大きさを変数としてモデル化することが古くから行われてきており，高並列処理の研究においても適用されている[1]．一方このモデル化方法でプログラムの処理を一字一句モデル化すると，小規模なプログラムの処理時間を再現するモデルを作るだけでも，モデルが複雑になりモデルパラメータ数も増える[2]．そこで一字一句ではなく，文献[3]のようにアルゴリズムをある塊にしてモデル化することが必要になる．しかしこれらの方法を任意のプログラムと並列計算機から成るシステムに適用することは容易ではない．またモデルパラメータ決定において並列オーバーヘッドの時間測定を必要とする場合，この測定がプログラムの最適化に擾乱を与える場合がある．

前回の発表では，問題の大きさを固定した場合の処理時間のモデル化方法を提案した[4]．この方法はプログラム全体の処理時間と並列化部の処理時間の測定値から処理時間モデルを構築でき，測定による最適化等への擾乱を軽減できる．さらにこのモデル化方法は，並列化部に隠れた並列オーバーヘッドを分離可能にした．

本論文では，この処理時間のモデル化方法を Weak scaling できるように拡張する方法を提案する．Weak scaling をプロセッサ当りの問題の大きさを固定してプロセッサ数を増やすこととすると，計算量が問題の大きさに比例しかつ並列オーバーヘッドが非常に小さい場合，システムの処理時間はプロセッサ数の増加に対してほぼ一定となる．このようなシステムは，プロセッサ数を増すことにより大きな問題を扱うことができると考えられ，Weak scaling が行えればこのようなシステムであるかどうかを評価することができる．しかしながら実際のシステムにおいて，プロセッサ当りの問題の大きさを固定してプロセッサ数を増やし，処理時間を測定することはそれほど容易でない．そこで Strong scaling のために問題の大きさを固定してプロセッサ数を増やす測定で得られたデータから，Weak scaling 可能なモデルを構築する方法を考案した．このモデル化方法はプログラムの内部構造に踏み込まず，プログラムを一つの塊として問題の大きさに対してモデル化するため，任意のシステムに容易に適用できる可能性がある．そこでまず分子動力学コードと IBM SP2 から成るシステムにこのモデル化方法を適用し，モデル化できることを確認した．

2 章では測定した処理時間からその処理時間モデルを構築する方法を提案する．3 章では 2 章の方法を用い，分子動力学コードと IBM SP2 から成る並列処理システム[5]

†富士通株式会社
Fujitsu Limited

の処理時間をモデル化する．4 章では得られた処理時間モデルを用い Strong scaling と Weak scaling についてシステムの性能評価を示す．

2. 処理時間モデルの構築

Weak scaling できる処理時間モデルを構築する方法を提案する．まず並列効率メトリックを介して測定時間とプロセッサ数を変数とする処理時間モデルをフィッティングし，モデルパラメータを求める前回の方法[4]を示し，この方法で得られたモデルパラメータと問題の大きさを関係付ける．

並列効率メトリックを式(1)に示す[6]．ここに$\tau(p,n)$：処理の経過時間，$\gamma_i(p,n)$：プロセッサ i の並列化部分の処理時間，p：プロセッサ数，n：問題の大きさである．

$$\varepsilon_p(p,n) \equiv \frac{\sum_{i=1}^{p}\gamma_i(p,n)}{p\cdot\tau(p,n)} \tag{1}$$

2.1 フィッティング式

フィッティングは，プロセッサ数に対する時間の変化を並列オーバーヘッド部分と並列化部分の時間比$\Sigma\chi_i(p,n)/\Sigma\gamma_i(p,n)$で捕えることにより行う．ここで"Σ"は全プロセッサの総和を表す．また$\chi_i(p,n)$は$\tau_i(p,n)$-$\gamma_i(p,n)$から算出し直接測定しないことにする．この比は$\varepsilon_p(p,n)$で表すことができ，式(1)を用いると式(2)となる．左辺は測定時間値により決定される項，右辺の$\tau(p,n)$は処理時間モデル，$\Sigma\gamma_i(p,n)$は測定値である．

フィッティングに際してはデータを$\varepsilon_{pc}<\varepsilon_p<1$ のように選ぶ．ここにε_{pc}はε_pの下限値である．ε_{pc}を設定することにより，処理時間が最小になって再び増加に転じる領域のデータがフィッティングで大量に使われることを効率という観点から制限することができる．例えばε_{pc}=0.1 のとき，式(2)の左辺は 0 ～9 で，この範囲は問題の規模や時間の大きさに関係なく設定できるメリットがある．

$$\frac{1-\varepsilon_p(p,n)}{\varepsilon_p(p,n)} = \frac{p\cdot\tau(p,n)}{\sum_{i=1}^{p}\gamma_i(p,n)} - 1 \tag{2}$$

2.2 並列処理部に含まれる並列オーバーヘッドの考慮

理想的な並列処理では，全プロセッサの並列処理時間$\Sigma\gamma_i(p,n)$ はプロセッサ数が変化しても変化しないはずである．一方測定値の$\Sigma\gamma^M_i(p,n)$はしばしばプロセッサ数の増加とともに増加する．この増加を並列化部分に隠れている並列オーバーヘッドと考え，本論文では「目に見えない並列オーバーヘッド」と呼ぶことにする．これを分離するため$\gamma^M_i(p,n)$をあるプロセッサ数 p_1 で固定し，式(1)を書き直すと次式を得る．ここに測定値とモデル変数を区別するため，測定値は$\tau^M_i(p,n)$と$\gamma^M_i(p,n)$を用いて表す．

$$\varepsilon'_p(p,n) \equiv \frac{\sum_{i=1}^{p}\gamma^M_i(p_1,n)}{p\cdot\tau^M(p,n)}$$

この$\varepsilon'_p(p,n)$を用いると式(2)は式(3)となる．フィッティングにおいて$\Sigma\gamma^M_i(p_1,n)$が定数になるため，フィッティングにおいて並列処理部として測定した時間に含まれる「目に見えない並列オーバーヘッド」の増加分を分離することが可能となる．

$$\frac{1-\varepsilon'_p(p,n)}{\varepsilon'_p(p,n)} = \frac{p\cdot\tau(p,n)}{\sum_{i=1}^{p}\gamma^M_i(p_1,n)} - 1 \tag{3}$$

2.3 処理時間モデル

処理時間モデルを式(4)で表す．ここに $a(n)$は全プロセッサの並列処理部の処理時間の総和，$\chi(p,n)$は並列オーバーヘッドである．

$$p\cdot\tau(p,n) = a(n) + p\cdot\chi(p,n) \tag{4}$$

式(4)を式(3)に代入すると次の式を得る．

$$\frac{1-\varepsilon'_p(p,n)}{\varepsilon'_p(p,n)} = \frac{a(n)+p\cdot\chi(p,n)}{\sum_{i=1}^{p}\gamma^M_i(p_1,n)} - 1 \tag{5}$$

(1) プロセッサ数に対する処理時間のモデル化

測定値から決まる式(5)左辺に右辺の処理時間モデルをフィッティングするため，右辺を式(6)の多項式で表す．

$$\frac{1-\varepsilon'_p(p,n)}{\varepsilon'_p(p,n)} = c_0(n) + |c_1(n)|\cdot p + p\cdot\sum_{j=2}^{j_{\max}} c_j(n)\cdot(p-1)^{j-1} \tag{6}$$

ここに c_i は多項式の係数で j_{max} は項数の上限値である．$p\cdot\tau(p,n)$でモデル化したメリットは，逐次処理時間が右辺第 2 項のように直線の傾きとして得られることである．ここで c_1 は逐次処理時間を表すので負にならないように$|c_1|$とした．一方ネットワークのトポロジによっては負になる可能性がある j>1 の係数は，絶対値をとらないようにした．また c_0 は 1+c_0 >0 を満たす範囲で正負の値となる．

多項式の係数と処理時間の間には次の関係がある．

$$c_0(n) = a(n) / \sum_{i=1}^{p} \gamma_i^M(p_1,n) - 1$$

$$|c_1| + \sum_{j=2}^{j_{max}} c_i(n)\cdot(p-1)^{j-1} = \chi(p,n) / \sum_{i=1}^{p} \gamma_i^M(p_1,n)$$

本論文では j_{max}=2 を採用したので式(6)は式(7)となる．

$$\frac{1-\varepsilon'_p(p,n)}{\varepsilon'_p(p,n)} \approx c_0(n) + |c_1(n)|\cdot p + c_2(n)\cdot p\cdot(p-1) \qquad (7)$$

従って式(4)の処理時間モデルのパラメータ $a(n)$と並列オーバーヘッド$\chi(p,n)$は下記の式(8)と(9)のように記述できる．

$$a(n) = \sum_{i=1}^{p} \gamma_i^M(p_1,n)\cdot\left(1+c_0(n)\right) \qquad (8)$$

$$\chi(p,n) = \sum_{i=1}^{p} \gamma_i^M(p_1,n)\cdot\left(|c_1(n)| + c_2(n)\cdot(p-1)\right). \qquad (9)$$

(2) **問題の大きさに対するモデル化**

問題の大きさ n に関係しているモデルパラメータは，式(7)，(8)より，$a(n)$と多項式の係数$|c_1(n)|$，$c_2(n)$である．そこで n に対する各パラメータ値を多項式でフィッティングして問題の大きさに対するモデルを構築する．多項式の次数の上限について，$a(n)$はプログラムの処理時間を支配するアルゴリズムに依存する．本論文の例では $O(n^2)$が上限値である．$|c_1(n)|$は逐次化処理を構成するアルゴリズムに依存する．これもアルゴリズムに依存すると考えられるが，n とは異なる問題の大きさの変数で支配されている可能性もある．$c_2(n)$は主に通信の処理時間がモデル化されている項である．一般にこの項は O(1)~O(n)になるように設計されていると考えられる．従って問題の大きさに対するモデル化が可能な場合，各モデルパラメータは簡単な多項式で記述できると考えられる．

3. 並列処理システムのモデル化

測定した並列処理の時間$\tau^M_i(p,n)$，$\gamma^M_i(p,n)$に 2 章のモデル化方法を適用し，モデルパラメータを決定し，並列処理システムのモデル化を行う．

3.1 対象システム

評価したアプリケーションプログラムと計算機からなる並列処理システムは，分子動力学コード（以下，MDコード）とIBM SP2 である[5]．このMDコードは，2 次元の矩形領域内での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コードで, Lennard-Jonesポテンシャルと重力を力の源とした運動方程式を離散化して解く．

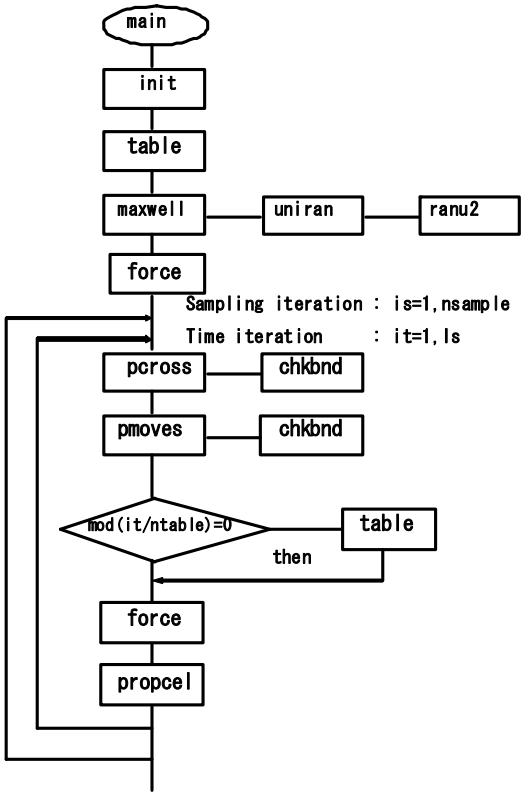

図 1　MD コードのフローチャート[5]

この MD コードの処理時間の構成要素で，処理時間を最も費やすのは粒子間力の計算である．この計算を全粒子間で行うと $O(n^2)$であるが，遮蔽距離を設定し，その 4 倍以上離れた原子間の力を計算しないことにして計算量を大幅に削減している．

並列化方法は，全粒子を各々のプロセッサに置いて，割り当てられた計算のみを各プロセッサが実行することとし，離散時間の進み毎に全対全通信を MPI_ALLTOALL で行い，各プロセッサで力の総和をとる方法が用いられている．

プログラム中の最初の実行文の前と最後の実行文の後までを処理時間を$\tau^M_i(p,n)$とする．$\gamma^M_i(p,n)$は，図 1 の force で示された上記力の計算，各粒子に対して遮蔽距離の 4 倍以内に存在する他の粒子を調べてそのリストを作る部分 table，及び粒子の情報を 2 次元座標系のメッシュ上に落とす処理の一部 propcel である．図中のそのほかのサブルーチンは逐次計算で，各プロセッサで冗長に実行される．

3.2 プロセッサ数に関するモデルパラメータの決定

測定値$\tau^M_i(p,n)$，$\gamma^M_i(p,n)$から p_1=8 として$\varepsilon'_p(p,n)$を算出し，式(7)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 1-1 に示す．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．

表 1-1　問題の大きさ毎のフィッティング結果

n	$\Sigma\gamma_i(p_1)$	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

図 2 に表 1-1 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 $0\le(1-\varepsilon'_p(p))/\varepsilon'_p(p)<9$（即ち並列効率メトリック $0.1<\varepsilon'_p(p)\le 1$）の範囲でよく一致しており，測定値から計算した式(7)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．

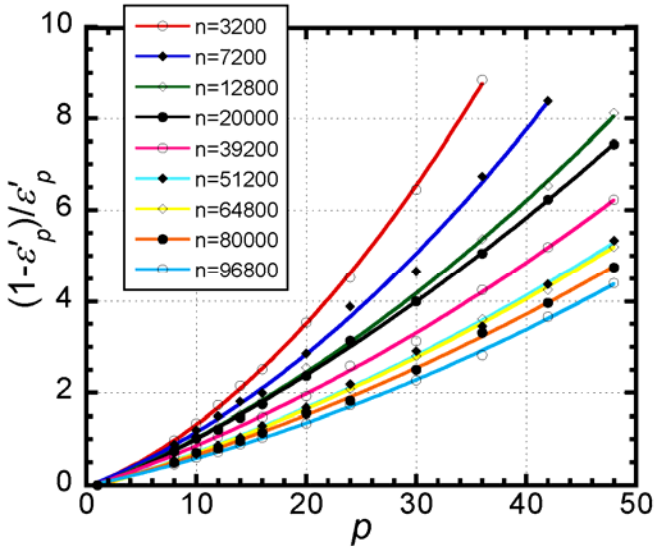

図 2　問題の大きさ毎の式(7)のフィッティング結果．
（マークは測定値，曲線はフィッティングラインを表す．）

3.3 モデルパラメータの問題の大きさに関するモデル化

表 1-1 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 1-2 に示す．表中これらのモデルのフィッティング状態を図 3 に示す．図 3 (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 3(c)は $c'_2(n)$ と n が反比例関係で近似できることを示す．図 3 (b)は n が小さいときにばらつきはあるが，$c'_1(n)$を一次式でモデル化したことを示す．

表 1-2　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

Vol.2010-HPC-126 No.44
2010/8/5

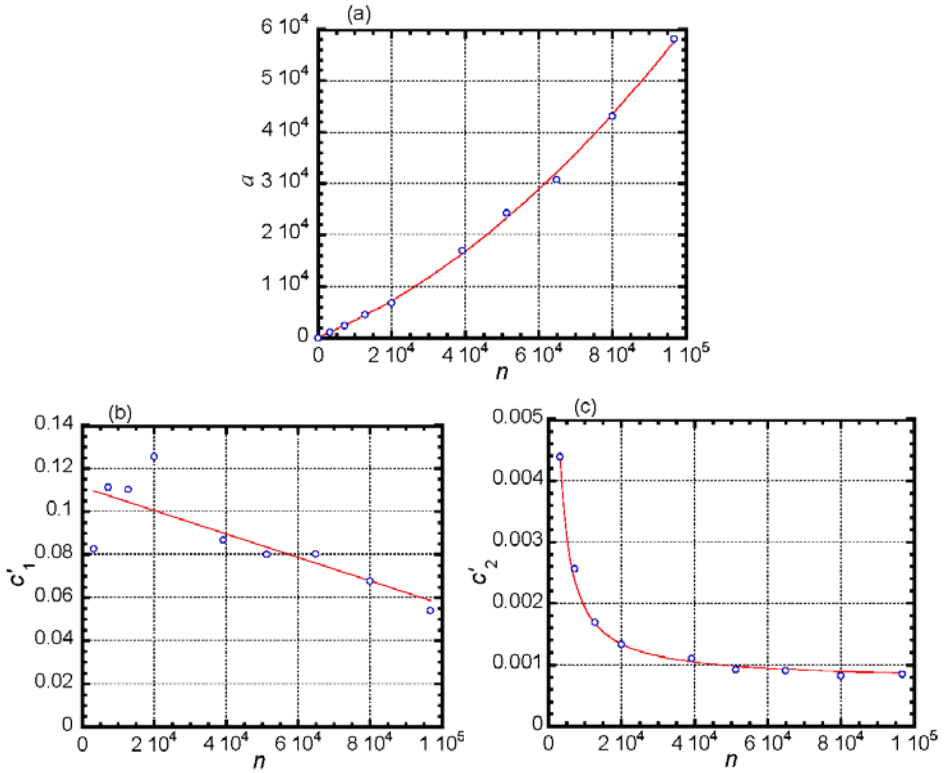

図 3　モデルパラメータの問題の大きさに関するフィッティング結果.

これら n に関するモデル式を式(8), (9)に代入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(10)を得る.

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

測定した処理時間とこの処理時間モデルの比較を図 4 に示す. シンボルは測定点, 曲線は式(10)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.

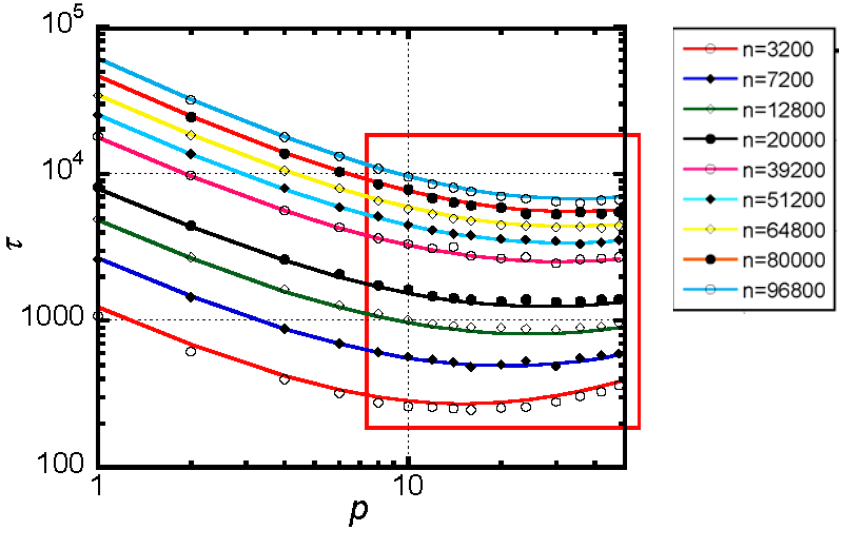

図 4　処理時間の測定値とモデルの比較
（赤で囲んだ測定値の内，$(1-\varepsilon'_p)/\varepsilon'_p$<9 を満たすものをフィッティングに使用）

4. 性能評価

処理時間モデルの式(10)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 の p=2~48，n=3200~96800 の範囲を対象に行った.

4.1 Strong Scaling における性能評価

処理時間モデルの式(10)を用い n=7200 と 80000 についての Strong scaling を図 5-1 の(a)と(b)に示す. ここに $\tau(p,n)$:MD コードの処理時間, $a(n)/p$:並列化部の処理時間, $\chi_0(p,n)$(= $a(n)\cdot(0.1113-5.436\cdot10^{-7}\cdot n)$: 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で除すと式(1)より a/τがε_pとなり，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 5-2 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．

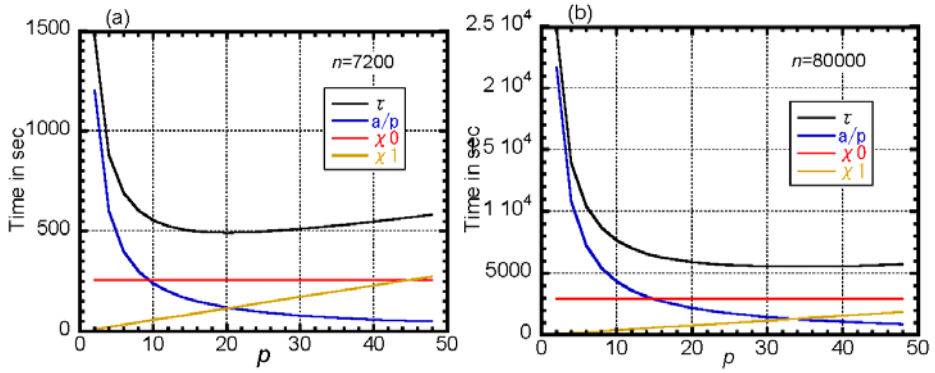

図 5-1　Strong scaling における時間変化

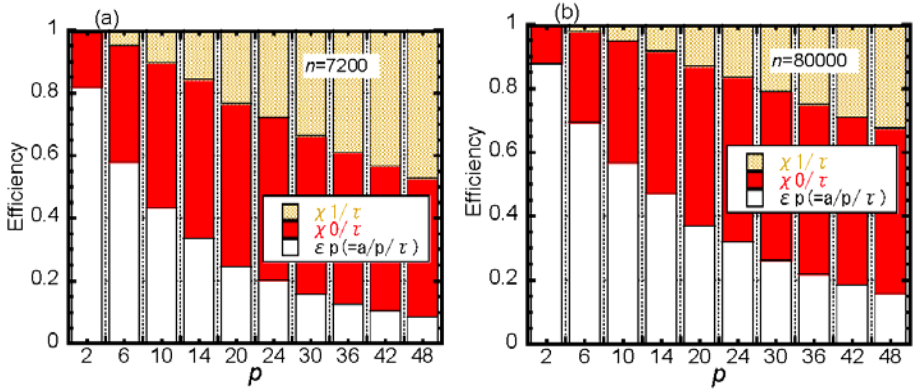

図 5-2　Strong scaling における効率の変化

図 5-3 は，並列化部に隠れていた「目に見えない並列オーバーヘッド」を，測定値とモデルの差異で示したものである．図 5-3 (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 5-3 (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．

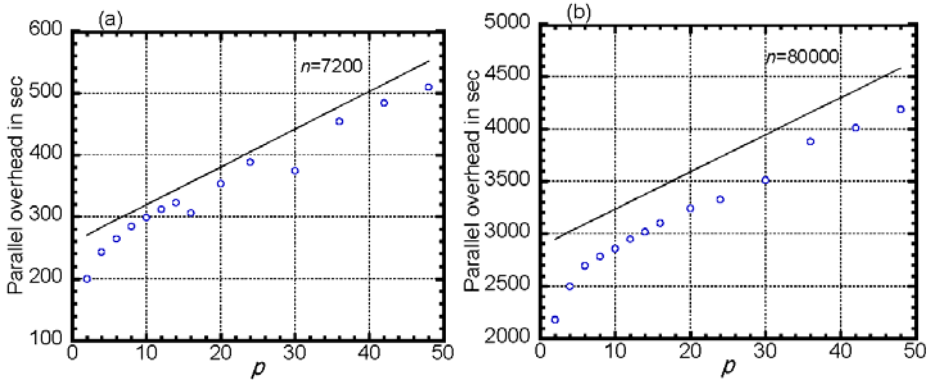

図 5-3　並列化部に隠れている並列オーバーヘッド（○：測定．線：モデル）

4.2 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 で表すと，k=一定として式(10)から n を消すと式(11)を得る．

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

$$\cdot\left(1+(0.11131-5.436\cdot 10^{-7}\cdot k\cdot p)\cdot p + (0.00074301+\frac{11.891}{k\cdot p})\cdot p^2\right) \quad (11)$$

Vol.2010-HPC-126 No.44
2010/8/5

図 6-1 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，このシステムは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．図はまたこのシステムの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$\chi_0(p,k)$と$\chi_1(p,k)$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$\tau(p,k)$が大きくなる主要な原因であることがわかる．$\chi_0(p,k)$と$\chi_1(p,k)$を比べると図の(a)と(b)共に$\chi_0(p,k)$> $\chi_1(p,k)$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$\chi_1(p,n)$は第二の原因であることがわかる．さらに $a(p,k)/p$ 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．

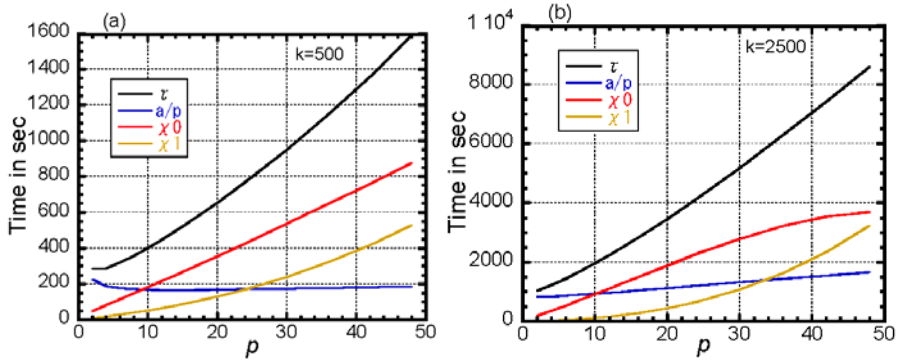

図 6-1　Weak scaling における時間変化

並列オーバーヘッド$\chi_0(p,k)$と$\chi_1(p,k)$の効率に対する寄与は，図 6-2 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．

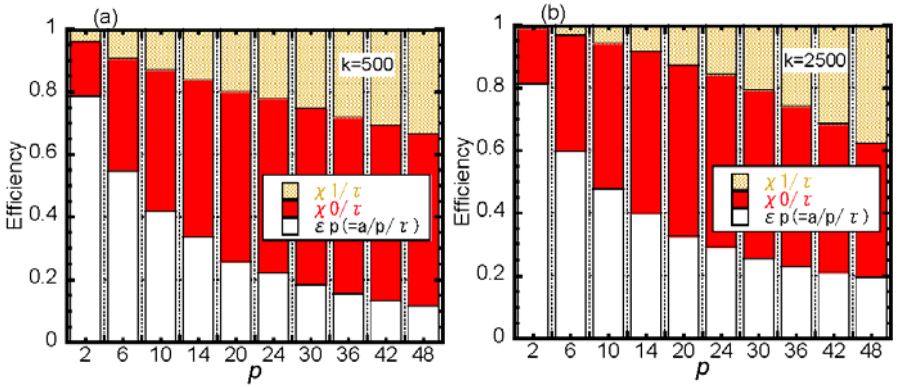

図 6-2　Weak scaling における効率の変化

5. まとめと議論

処理時間を測定データとのフィッティングによるモデル化において，各プロセッサの処理時間とその並列化部のみの測定データから処理時間のモデルを構築する方法を示した．この方法で得たモデルパラメータを問題の大きさの多項式で近似して Weak scaling 可能な処理時間モデルを構築する方法を提案した．これを物理現象を解析するプログラムである分子動力学コードと IBM SP2 から成るシステムに適用し，複数のアルゴリズムからなるプログラムに対して処理時間モデルが構築できることを示した．

今回の処理時間モデルはプロセッサ数に対するフィッティング 9 回，問題の大きさに対するフィッティング 3 回で得ることができた．実際の並列システムの処理時間のモデル化において，このように手数が少ないことは有効と考える．

この方法は，プログラムの内部構造に立ち入らず処理時間モデルを構築することができるので，あらゆる並列システムの処理時間をモデル化できる可能性がある．一方図 3(b)で示したように種々のアルゴリズムが絡む逐次処理部は，場合によってはモデル化が難しいと考えられる．これらモデル化の可能性と困難性は主にアプリケーションプログラムの性質により決まると考えられるため，種々のアプリケーションプログラムを用いてモデル構築を行うことは今後の研究課題の一つと考える．

本方法で構築したモデルは多項式のため，モデルを測定値外に外挿すると負の値になる場合がある．外挿できるモデルの構築は次の重要な研究課題と考える．

Vol.2010-HPC-126 No.44
2010/8/5

謝辞 論文作成に際し，富士通株式会社次世代テクニカルコンピューティング開発本部の安里彰氏，九州大学大学院システム情報科学研究院の井上弘士先生にコメントを頂きました事に感謝いたします．富士通株式会社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エグゼクティブアーキテクトの援助に感謝します．

参考文献

1) R. Susukita, et. al, Performance Prediction of Large-scale Parallel System and Application using Macro-level Simulation, In proceedings of SC2008.
2) 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，Vol.39, No.3, pp.529-541 (1998).
3) Y. Kishimoto and S. Ichikawa, Optimizing the configuration of a heterogeneous cluster with multiprocessing and execution-time estimation, Parallel Comput, 31, pp. 691-710 (2005).
4) 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，情報処理学会研究報告，2009-HPC-121, Vol.2009, No.28, pp.1-6 (2009).
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).
6) Shigeo Orii, Metrics for evaluation of parallel efficiency toward highly parallel processing, Parallel Comput, 36, pp. 16-25 (2010).