数値計算のための並列計算機性能評価方法

折 居 茂 夫†

プログラムレベルからループレベルまでの性能を一貫して評価する，数値計算のための並列計算機性能評価方法を提案する．この方法に使用するループの処理時間モデルの特徴は，理論最大性能flop/s等で規格化された効率を表す係数を導入し，ループレベルの性能評価を可能にした点にある．このループの処理時間モデルからプログラムの処理時間モデルを作成し，時間の測定によりモデル係数を決定し，プログラムレベルの性能評価を行うことができる．例として，この性能評価方法を粒子分割方法で並列化した分子動力学プログラムに対して適用し，ベクトル並列計算機VPP300の性能を評価した．その結果，分子動力学プログラムの性能は，粒子数，プロセッサ数に加えて遮蔽距離と物理量の観測回数に依存していることが分かった．またプログラムレベルの性能は，粒子数の増加とともに大きく向上する．理由は，粒子が少ないときに最も計算時間を費やすループが，粒子の増加とともに演算器を有効に使う他のループに入れ代わるためである．この性能評価方法は，計算機利用，プログラム開発，計算機設計という3つの見地から成る性能評価を可能にする．

A Performance Evaluation Method of Parallel Computers for Numerical Analysis

SHIGEO ORII†

A performance evaluation method of parallel computers for numerical analysis which enables to predict the performance consistently from a program level to a loop level is offered. The character of a processing time model of loops used this method is that the loop level performance evaluation is executable by introducing model parameters including normalized efficiency by theoretical peak performance flop/s etc. In case of a program level the evaluation is carried out by using both the model and the model parameters decided by actual measurement of time. As an example, this performance evaluation method is applied for molecular dynamics program parallelized with the particle decomposition approach, and the performance of vector parallel computer VPP300 is evaluated. As a result, it is shown that the performance of a program level depends on shielding distance and observation times of physical variables in addition to the number of particles and the number of processors. The performance of a program level is greatly improved when the number of particles increases because the most time-consuming loop in case of a small number of particles is exchanged for another loop which uses processing units efficiently when the particles increase. This performance evaluation method allow us performance evaluation consisting of the three view points which are use of computer, development of program and design of computer.

1. はじめに

並列プログラムの並列計算機上での性能は，並列化できる部分とできない部分の時間比率と通信等の並列オーバヘッドによって決まる．プロセッサ数を増したとき，並列計算できる部分の時間は減少し，できない部分の時間の割合が増加する．これに対する性能評価は，Amdahlの法則[1]によって行うことができる．この法則によれば，性能はプロセッサ数に対して単調増加するため，逐次プログラム計算時間とその中で並列計算できる部分の時間が分かれば，簡単に性能評価を行うことができる．Amdahlの法則は，ベクトル計算機に対しては大変有効な方法であった．一方，並列計算機では，プロセッサの増加にともない通信量等並列オーバヘッドが増加する[2]．したがって，プログラムの並列性能は，プロセッサ数の増加にともなう計算時間減少と並列オーバヘッドの増加のバランスによって決まる．計算時間に比べて通信時間が十分に小さければ，並列計算機の性能評価はAmdahlの法則で簡単に行うことができる．しかし実際の並列計算機は，1要素を通信する時間が1要素を四則演算する時間の十倍

† 日本原子力研究所計算科学技術推進センター
Center for Promotion of Computational Sci. & Eng., Japan Atomic Energy Research Institute

以上の場合が多く，その性能評価には通信時間の考慮が必要となる．このように相反するもののバランスにより性能が決定する場合，計算時間と並列オーバヘッド時間を定量的にとらえて性能評価することが不可欠となる．またプロセッサ数，問題の規模，計算条件により性能が異なることが予想され，パラメータサーベイを行うことが不可欠となる．このような状況下では，これまでの多くのベンチマークテストに見られるような定点観測による性能評価では不十分である．しかし実際に広い領域を実測によりパラメータサーベイすることは，多大の時間を必要とし非現実的である．

この問題は，処理時間をモデル化すれば解決できる．しかし，ただ定式化したのでは，そのプログラムと計算機システムが偶然に実現した処理時間を記述したにすぎず，性能評価には十分でない．たとえば，四則演算が最大理論性能に比べ著しく低い場合，通信時間の占める割合が減少して並列効率が向上する．並列処理の性能評価では，このような状態をも正確に評価する必要がある．Hockney[3]は簡単な処理時間をモデル化し，そのモデル係数により並列計算機の性能を評価できることを示したが，そのモデル係数の性質上，実際のプログラムへの適用は困難であった．

一方，モデル係数の意味を厳密に定義する代わりに，最大理論 flop/s 値等の性能に関する基準を導入し，その値で規格化したモデル係数を導入してプログラムの処理時間をモデル化すると，プログラムレベルの性能とそれを実現する個々のループレベルの性能評価を同時に行うことが可能となる[4]．測定を基に決定されたこのモデルの係数は，アルゴリズム，コーディング，ハードウェア，コンパイラ等のプログラムと計算機システムすべての要因が内包されているが，数値計算を行うため使用されたハードウェアやコンパイラの最適化がそのプログラムのループをいかにうまく処理しているかを示す指標ととらえることができる．

プログラムレベルの性能評価は，一般的にスケーラビリティという概念を導入して行う．しかしその定義は直感的であり，定量的な性能評価のためには新たに性能評価指標を導入する必要がある[5]．そこで本論文では，よく知られた並列効率と，その考え方に基づいて導出した指標を性能評価指標として用いて，プログラムレベルの性能評価を行うことにした．しかる後，これらの指標を，プロセッサ数と問題の規模を軸とした2次元平面の等高線図で表すことにより，計算機利用，プログラム開発，計算機設計という3つの見地からの性能評価を行うことができることを示す．

2章では数値計算に限定した処理時間のモデル化方法について述べる．3章では性能評価指標について論じる．4章では粒子分割法で並列化された分子動力学プログラムに対する処理時間モデルを示す．5章ではこのモデルを用いてVPP300の性能評価を，計算機利用，プログラム開発，計算機設計という3つの見地から示す．6章ではまとめと議論を行う．

2. プログラムの処理時間のモデル化

数値計算プログラムの処理時間をプログラム中のループ処理単位にモデル化する方法を示す．モデル化したループの処理時間を合成してプログラム全体の処理時間をモデル化する．

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 としたときに次のようにモデル化することができる[6]．

$$\tau(p,n) := \alpha(n) + \beta(n)/p + \sigma(p,n) \qquad (1)$$

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ヘッドである．

2.1 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) のようにモデル化する．ループ単位の性能評価を可能とするため，理論最大性能 r_a（Mflop/s）を基準とした比例係数 a_u を導入する．ここに，t_{0u} は種々の要因から成る前処理，後処理時間ととらえることができる．

$$t_u(n) := t_{0u} + f(n)/(r_a \cdot a_u) \qquad (2)$$

$t_u(n)$ 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，理論最大性能 p（倍）を基準とした比例係数を e_p とすると，$t_p(p,n)$ は式 (3) のように記述できる．

$$t_p(p,n) := t_{0p} + t_u(n)/(p \cdot e_p) \qquad (3)$$

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，後処理時間ととらえることができる．ロードバランス等の並列オーバヘッドは e_p でとらえることができる．

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 (1) の $\alpha(n)$，$\beta(n)$ は次のように表すことができる．

$$\alpha(n) = \sum_{i=1}^{L_{nonp}} \{t_u(n)\}_i + \sum_{j=1}^{L_p} \{t_{0p}\}_j \qquad (4)$$

$$\beta(n) = \sum_{j=1}^{L_p} \{t_u(n)/e_p\}_j \qquad (5)$$

2.2 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ヘッドが含まれるが，ここでは，たと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，通信するデータの個数 $g(p,n)$ に比例していると仮定して，式 (6) のようにモデル化する．

$$t_c(p,n) = t_{0c} + g(p,n) \cdot X_B/(r_c \cdot e_c) \qquad (6)$$

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，通信の効率を表す．t_{0c} は立ち上がり時間である．g 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ヘッド σ は，その総数を m_{com} とすると，式 (7) のように表すことができる．

$$\sigma(p,n) = \sum_{m=1}^{m_{com}} [t_c(p,n) + t_{others}(p,n)]_m \qquad (7)$$

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)，(3) を用いてモデル化する．たとえばプロセッサ間の総和をとる場合，逐次計算にはな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．

3. 性能評価指標

並列処理の性能評価で使用する指標について述べる．プログラム単位の並列処理の性能評価指標は，並列効率，および本論文で総合効率と呼ぶ2つの指標から成る．これらの指標は，3.2 節で述べるループレベルの性能評価指標を用いて評価できる．

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単位の性能評価が必要となる．計算機システムに則したプログラムやアルゴリズムを構築する場合，あるプログラムに適した計算機の設計を行う場合，ループ単位の性能評価を必要とする．

3.1 プログラムレベルの性能評価指標

プログラム全体に対する性能評価の指標として，並列効率 E_p を用いる．これを式 (8) に示す．E_p は並列計算機の台数効果を効率の形で表現したもので，計算機システム間での比較が可能な量である．

$$E_p(p,n) := S(p,n)/p \qquad (8)$$

$S(p,n)$ は並列化によるスピードアップで，Amdahl の法則では式 (9) のように表せる．

$$S_A(p,n) = 1/(1 - R_p + R_p/p) \qquad (9)$$

ここで R_p は並列化率である．この式から E_p を計算した場合，並列オーバヘッドを含まない並列効率が計算できる．並列オーバヘッドを議論する前に，まずこの値を議論する必要がある．なぜなら，この値が並列処理による性能向上の上限値だからである．並列化率は次のように定義できる．

$$R_p(n) := 1 \Bigg/ \left[1 + \sum_{i=1}^{L_{nonp}} \{t_u(n)\}_i \Bigg/ \sum_{j=1}^{L_p} \{t_u(n)\}_j \right]$$

この定義は，$\sigma = 0$，$t_{0p} = 0$ および $e_p = 1$ としたときの，$\beta(n)/(\alpha(n) + \beta(n))$ に等しく，並列化されていないプログラムの並列効率の予測ができ，便利である．

一方，並列オーバヘッドを考慮する場合，E_p の計算に式 (10) のスピードアップ S_M を用いる．

$$S_M(p,n) = \tau(1,n)/\tau(p,n) \qquad (10)$$

これまで示した性能評価指標 E_p は相対値であるため，個々のプロセッサの演算器の使用効率が考慮されていない．したがって，スカラプロセッサのスーパリニア現象では，E_p が1を超える場合がある．この問題は，演算器の使用効率を考慮した基準を設けて解決することができる[7]．まず，プログラム中で実行された全四則演算数 $F(n) := \Sigma_{ij=1}^{L_{nonp}+L_p} \{f(n)\}_{ij}$ を用い，単一プロセッサの理想的な処理時間 $F(n)/r_a$ を計算する．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，式 (11) を得る．演算器の使用効率を考慮したこの E_t を総合効率と呼ぶことにする．

$$S_t(p,n) := F(n)/r_a/\tau(p,n)$$
$$E_t(p,n) := S_t(p,n)/p \qquad (11)$$

プログラムの処理時間で四則演算が占める割合が支配的で，並列計算機システムが四則演算と同時に他の演算を実行できる条件が満たされたとき，E_t は1に近づく．したがって，$E_t(1,n)$ は，そのプログラムに対し四則演算がいかに速く動くように作られているかを示す指標である．また，四則演算の占める割合が少ないプログラムを見分ける指標ともなる．この指標 E_t を基にして，そのプログラムを効率良く実行する並列計算機システムを探すことが可能となる．また，必ず $E_t(p,n) \leq 1$ となり，スーパーリニア現象も包含することができる．

S，R_p，E_p は，時間測定から容易に計算することができる．たとえ $\tau(1,n)$ が直接実測できない場合でも，$\tau(p,n)$ と $\sigma(p,n)$ を測定し，式 (1) より計算することができる．S_t，E_t は，さらに $F(n)$ を求めて計

算する．多くの計算機で，$F(n)$ を実測するツールがある．

式 (2)，(3) より処理時間のモデルを構築し，時間測定を基にモデル係数を決定すれば，時間測定が容易でない問題の規模，プロセッサ数等での性能評価を行うことができる．

3.2 ループレベルの性能評価指標

$E_t(p,n)$ に寄与しているループのモデル係数 a_u，e_p，e_c を性能評価指標として，ループの性能評価ができる．これらのモデル係数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る．また t_{0u}，t_{0p}，t_{0c} はその処理のオーバヘッドであり，そのループの処理時間に占める割合が重要となる．これらの指標値は複数の要因で決まる．そこでその要因を考察する．

(1) e_p と t_{0p} に関する要因
- ロードバランス
- 並列処理の前処理・後処理

ロードバランスは e_p に反映される．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，t_{0p} に反映される．

(2) e_c と t_{0c} に関する要因
- 通信の立ち上がり
- 通信の実効バンド巾

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が e_c に反映される．

(3) a_u と t_{0u} に関する要因
- 複数演算機による四則演算の同時処理
- 四則演算のパイプライン処理
- メモリアクセス処理
- 分岐処理

論理最大 flop/s 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，したがって，a_u は，複数の演算器とパイプライン処理による四則演算の同時実行の影響を受ける．同時実行される四則演算が 1 個の場合でも a_u に与える影響は数十%である．一方，パイプライン処理ができず，パイプライン段数が多い場合は a_u にオーダーの影響を与える．ベクトル計算機でベクトル長が極端に短い場合がこれに当たる．またメモリアクセス時間が長くなるときも，a_u にオーダーの影響を与える場合が多い．ベクトル計算機ではバンク競合，スカラ計算機ではキャッシュミスヒットがこれに当たる．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 はファクターからオーダーまで様々な値となる．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．たとえば，パイプライン処理の立ち上がりがこれに当たる．これらの要因は，プログラムのコーディングから知ることができる場合が多く，その場合は個々の要因を取り除いて時間測定することにより，性能を決める要因を特定できる．

4. 並列化分子動力学プログラムの性能評価

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[8]の並列化版[9]に対して，2 章で示した処理時間のモデル化を行い，VPP300 のモデル係数を求める．

4.1 計算内容

m をアルゴン原子の質量，$\boldsymbol{r}_i$ を i 番目のアルゴン原子の位置座標，$\boldsymbol{F}_i$ 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次のようになる．

$$\frac{d^2\boldsymbol{r}_i(t)}{dt^2} = \boldsymbol{F}_i/m + \boldsymbol{g} \tag{12}$$

ここに，g は重力加速度である．$\boldsymbol{F}_i/m$ は以下のように書ける．

$$\boldsymbol{F}_i/m = -\sum_{j\neq i}^{n} \frac{\partial\phi(r_{ij})}{\partial\boldsymbol{r}_i} \tag{13}$$

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．$\phi(r)$ は Lennard-Jones ポテンシャルで式 (14) となる．

$$\phi(r) = 4\epsilon\left\{\left(\frac{\sigma_c}{r}\right)^{12} - \left(\frac{\sigma_c}{r}\right)^{6}\right\} \tag{14}$$

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うえで時間平均を行う．たとえば，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 $t \sim \Delta t \cdot I_s$ 内の時間平均物理量 $\langle a_k\rangle_t$ を式 (15) として表すことができる．

$$\langle a_k\rangle_t = \frac{1}{\Delta t \cdot I_s}\sum_{s=1}^{I_s}\left(\frac{A_k(s)}{N_k(s)}\right) \tag{15}$$

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で底と上を，鏡面反射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．初期状態では，空間的に等間隔配置で，初期速度は Maxwell 分布である．

4.2 並列化分子動力学プログラム

分子動力学では，各々の粒子は自分以外のすべ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，さらに遮蔽距離 r_{cut} 内の粒子か否かを判定し，r_{cut} 内の粒子のみ式 (13) 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 $3\sigma_c$ 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 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，物理量の計算をする式 (15) の propcel が並列化されている．並列化にともなう主な通信は force における力の総和計算と，propcel における式 (15) の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．

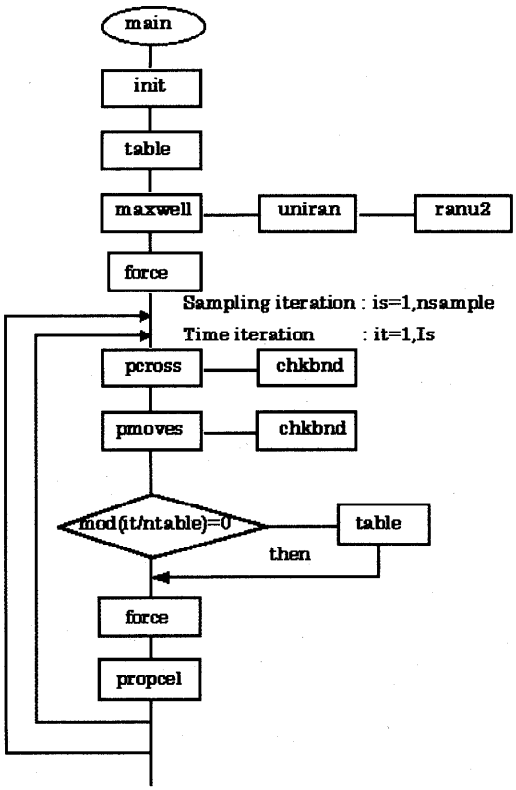

図 1 分子動力学プログラムのフローチャート
Fig. 1 Flowchart of molecular dynamics (MD) program.

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} ごとに呼ばれる．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 で境界を判定し，pmoves で粒子の移動と境界の反射を計算する．式 (12) 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，時間積分の繰返し回数 I_s とサンプリング回数 n_{sample} の制御を行う．

4.3 処理時間のモデル化

2 章で述べたモデル化方法で，プログラム中の処理時間を多く費やす部分をモデル化した．これを**表 1** に示す．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 $n_x \cdot n_y$）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．付録 **A.1** に force の並列化プログラムを示す．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 $C_1 \cdot n/2$ 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 (13) 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 $I_s = 6 \times 2000$) プラス初期設定の 1 回で，計 12001 回である．また通信をともなう fx と fy 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]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

表1 並列化分子動力学プログラムの処理時間モデル
Table 1 Processing time models of a parallelized molecular dynamics program.

(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）

判定で調べ，テーブルに書き出す．このとき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．

propcel では粒子の物理量を式 (15) 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 および 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1個の足し算および，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 $n_x\cdot n_y$ の各セル k ごとに積算する．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，各セルごとに物理量を粒子数で除して密度を計算し，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ごとに do305 で計算した各セルごとの物理量を I_s で除して時間平均を計算する．このとき物理量すなわち運動量の2成分およ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．

表2 VPP300のモデル係数値
Table 2 Value of model parameters of VPP300.

	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	–	–	–	–	–

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．

5. VPP300の性能評価

処理時間のモデルと実測値を基に3章で示した指標を求め，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．このため，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．さらに必要に応じて，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1で示したサブルーチンとdoループに対し，VPP300の時間測定を経過時間を測るサブルーチン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：2.2 Gflop/s，r_c：570 MB/sおよび表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2**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．両者はよく一致しており，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．

5.1 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および，プロセッサ数の増加に比例して処理時間が十分減

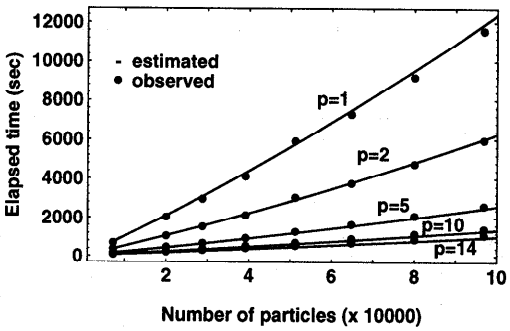

図2 VPP300における分子動力学プログラムの処理時間モデル精度
Fig. 2 Accuracy of processing time model of MD program on VPP300.

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，縦軸を問題の規模すなわ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3**に示す．

たと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

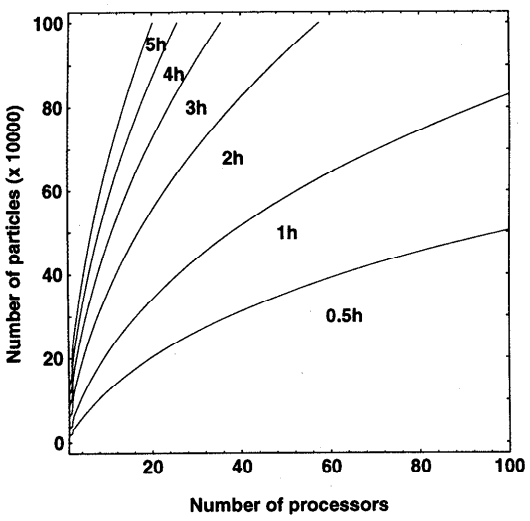

図3 VPP300における分子動力学プログラムの予測処理時間
Fig. 3 Predicted processing time of MD program on VPP300.

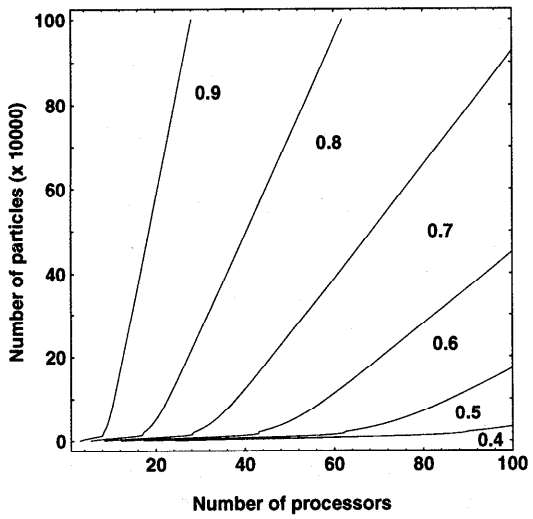

図4 MDプログラムの並列効率
Fig. 4 Parallel efficiency E_p of MD program.

とともに間隔が狭くなり，処理時間が急激に増加する．たとえば，プロセッサ数40では50万粒子を約1h，80万粒子を約2h，100万粒子を約4h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ことが分かり，問題の規模ごとの評価が必要なことが分かる．

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**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．先の50万粒子の実行での並列効率は，10プロセッサで約3時間のとき90%以上，40プロセッサで約1時間のとき約80%である．並列効率を用いると，プロセッサの利用率という観点で計算機間の比較を行うことができる．

並列効率に条件を付けると，プログラムを実行するときのプロセッサ数の決定を，容易に行うことができる．この条件は，価格，計算機の運用時間等により決まり，利用者ごとに大きく異なるのが普通である．

5.2 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見なした評価である．その中でベクトル処理されているか否か，スーパースカラであるか否かを問題としない．

一方プログラム開発の見地からの性能評価は，プログラムレベル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レベル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．

初めに並列効率を調べる．並列化率が100%で並列オーバヘッドがない場合，並列効率は100%となる．したがって，まずこの2つの要因を分離して各々の寄与を調べる．図4の並列効率は粒子数の増加とともに増し，プロセッサ数の増加とともに減少する傾向を持ち，プロセッサ数の増加とともに右方上がりの傾向を示す．この右方上がりの度合は，並列化率と並列オーバヘッドの寄与として現れ，この図では並列効率0.5以上で観測される．

並列化率の寄与を知るため式(9)のAmdahlの法則を用いて並列効率を計算し，並列オーバヘッドが寄与しない状況での性能を調べる．これを**図5**に示す．この図は，等高線の右方上がりの現象は並列化率の寄与のみでも現れることを示す．

また両図を比較すると，並列オーバヘッドにより等高線の形状はあまり変化せず，図5が全領域にわたって約10%右にシフトして減少することが分かる．そこで，並列オーバヘッドもこの現象に寄与していることが分かる．

並列化率と並列オーバヘッドの寄与の度合を調べるため，プロセッサ数100のとき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6**に示す．

主要な処理時間を占めるのは，粒子数が少ないときはforce，多くなるとtableとpcrossが加わる．ここ

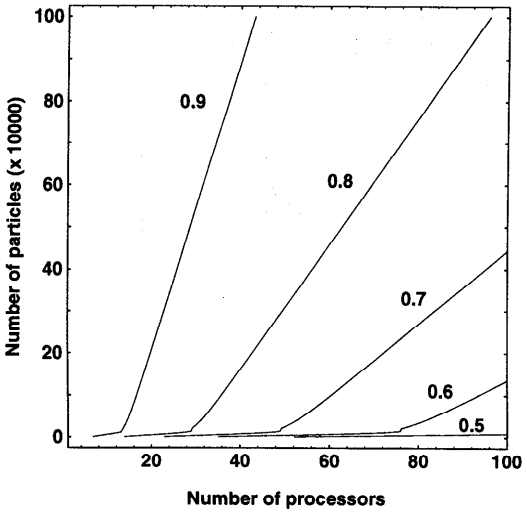

図5 Amdahlの法則から計算した並列効率
Fig. 5 Parallel efficiency based on Amdahl's law.

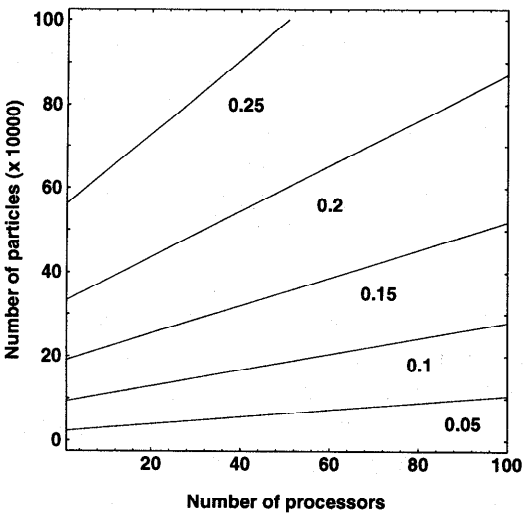

図7 MDプログラムの総合効率
Fig. 7 The total efficiency E_t of MD program.

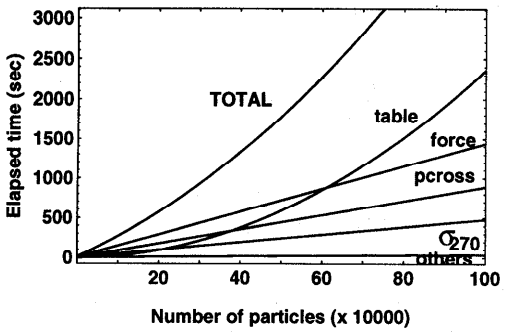

図6 プロセッサ数100での各サブルーチンの処理時間
Fig. 6 Processing time of each subroutine on 100 processors.

にforceは，並列オーバヘッド σ_{270} を含んだ処理時間である．ただちに，図5から図4の間で並列効率が約10%下がる原因は，σ_{270} の並列オーバヘッドであると特定できる．また，図5において右方上がりの傾向を作っている要因がpcrossであることが特定できる．逐次処理のpcrossの処理時間と並列オーバヘッドの σ_{270} が同じオーダーであることから，プロセッサ数100のとき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分かる．

図6は，粒子数の増加にともないforceの並列オーバ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．言い換えると，粒子数増加にともな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，並列化率が向上する．これが，粒子数の増加にともない並列効率が上昇する原因である．また，図3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るためであることが分かる．

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7の総合効率を用いる．図はこのプログラムがVPP300の持つすべ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す．図は，総合効率が粒子数の増加にともない増加し，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6を見ると，粒子数の増加にともない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．さらに表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 $a_{u100} = 0.346$ であることが分かる．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

表3 forceの性能要因
Table 3 Reason for performance of force.

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

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3.2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．

付録A.1に着目する．do271の回転数すなわちベクトル長は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．さらに距離の計算のとき，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flop/s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．むろん，y(i)も同様に置き換える．

これらを行った結果を**表3**に示す．この表中のflop, flop/s, $(\tau_u)_{force}$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．この表は，連続メモリアクセスにより性能が104 Mflop/sから123 Mflop/sになり1.2倍向上すること，if文の消去により演算量が67.9 Gflopから159 Gflopになったにもかかわらず$(\tau_u)_{force}$が690秒から630秒になり，性能が104 Mflop/sから247 Mflop/sと2.5倍向上すること，さら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104 Mflop/sから782 Mflop/s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分かる．

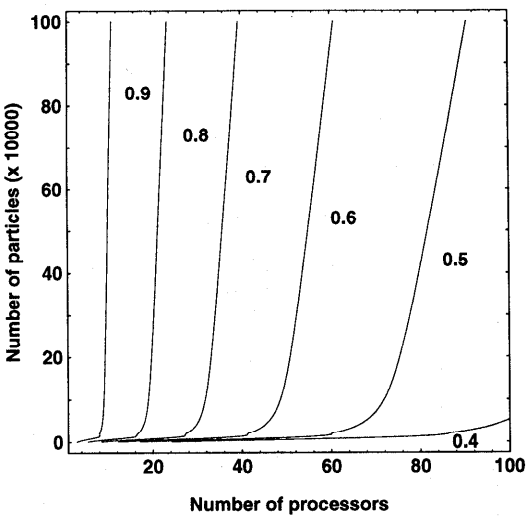

図8 観測回数600の並列効率
Fig. 8 Parallel efficiency E_p in case of 600 observations.

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分かる．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，すなわちα_{cut}とr_{cut}に依存しているため，総合性能はこれら閾値の変化に大きく左右されることも予測される．

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1つに，物理量の計算，すなわち観測行為がある．観測が通信をともなうためである．**図8**に観測回数n_{sample}を600回にしたときの並列効率を示す．図4 (n_{sample} : 6)と比較すると，等高線は右方上がりが急になって並列オーバ

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．

5.3 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，アルゴリズムの変更で回避が図られる．

一方，計算機設計の見地からの性能評価では，短ベクトル長で十分な性能を発揮する演算器を設計したときに，それが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．

そこでモデル係数を a_{u271}=0.5 に設定した，短ベクトル長で十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9**に示す．

$a_{u271}=0.030$ の図7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分かる．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．

このように計算機設計の見地からの性能評価では，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．

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分かる．

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10**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，総合効率があまり向上しない原因になっていることが分かる．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．

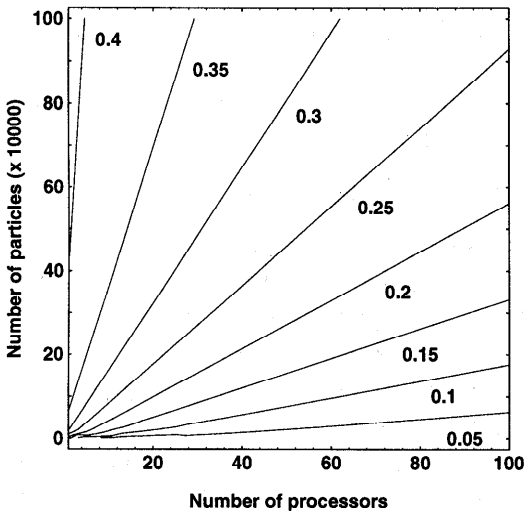

図9 $a_{u271}=0.5$ のときの総合効率
Fig. 9 The total effciency E_t in case of $a_{u271}=0.5$.

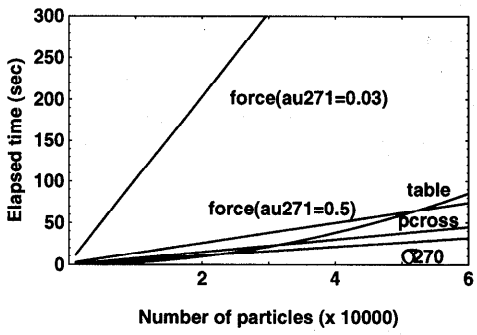

図10 $a_{u271}=0.03$, $a_{u271}=0.5$, $p=10$ のforceの処理時間
Fig. 10 Processing time of force in case of $a_{u271}=0.03$, $a_{u271}=0.5$, $p=10$.

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．すでにベクトル長は十分長いためである．このことは，表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であることから分かる．また表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分かる．したがって，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，$a_{u271}=0.5$ とすると全体的により高い性

能の設計仕様が必要であることが分かる．

このように計算機設計の見地からの性能評価は，プログラムの構造に依存する．したがって，なるべく実際のプログラムに即した並列化プログラムを評価することが不可欠である．

6. まとめと議論

本研究は，**図11** に示す並列計算機用ベンチマークテスト・システムの研究開発[10]における，並列処理性能評価方法研究に位置付けられて行われた．

並列計算機用ベンチマークテスト・システムの研究開発は，並列計算機の実用に即した性能評価と普及を目的とし，利用者コードの並列化，利用者コードを基にした並列BMTコード開発，並列BMTコードの性能評価，並列処理性能評価方法研究，およびそれらの成果公開を行う．また並列処理性能評価方法研究を通して，容易な性能評価を実現するツールを開発している[11]．

並列計算機用ベンチマークテスト・システムの中での本研究の目的は，アーキテクチャとピーク性能が異なる並列計算機どうしを比較し，各計算機の長所短所を正確に把握する実用に即した性能評価方法の確立である．このためのループレベルの性能評価指標を新たに提案し，既存のプログラムレベルの性能評価指標と組み合わせた性能評価方法を，計算機利用，プログラム開発，計算機設計という3つの見地から示した．

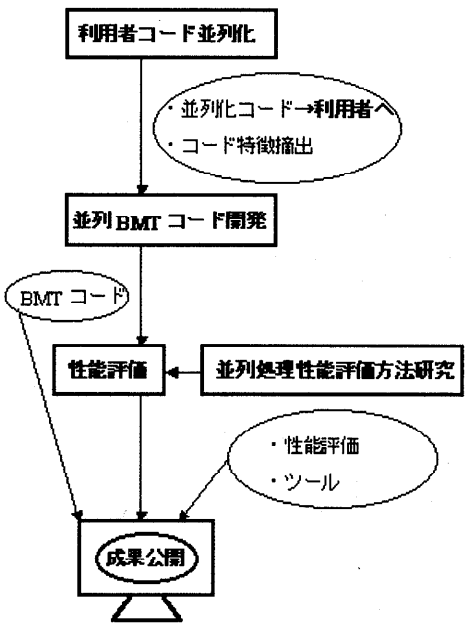

図11 並列ベンチマークシステム概要
Fig. 11 Schematic view of parallel BMT sysytem.

本論文ではベクトル並列計算機VPP300のみの性能評価を行ったが，現在スカラ並列計算機SP-2の性能評価を同様の方法で実施中である．

本研究の性能評価方法を用い，並列BMTコードの性能評価を3つの見地から種々の並列計算機で行うことにより，従来のベンチマークテストが得た結果より，さらに詳細な解析を行うことができる．これを蓄積することにより今までの性能評価のように現状を評価するだけでなく，多数のプログラムに適応した計算機システムの概念設計を行うことが可能となると考える．

謝辞 日本原子力研究所の浅井清氏，相川裕史氏には様々なご討論をいただきましたことを，また日本原子力研究所の渡辺正氏には，分子動力学プログラムを使用させていただきましたことを深く感謝いたします．

参考文献

1) Amdahl, G.: The validity of the single processor approach to achieving large scale computing capabilities, *Proc. AFIPS Conf. Spring Joint Comput. Conf.*, Vol.30, pp.483–485 (1967).
2) Flatt, H. and Kennedy, K.: Performance of parallel processors, *Parallel Comput.*, Vol.12, pp.1–20 (1989).
3) Hockney R.: Performance parameters and benchmarking of supercomputers, *Parallel Comput.*, Vol.17, pp.1111–1130 (1991).
4) 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, pp.27–32 (1995).
5) 関口智嗣，佐藤三久，井須芳実，長嶋雲兵：定量的な並列システムのスケーラビリティ評価指標，情報処理学会JSPP'96, pp.235–242 (1996).
6) Wichards, D.M.: Problem size scaling in the presence of parallel overhead, *Parallel Comput.*, Vol.17, pp.1361–1376 (1991).
7) 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，HPC57-11, pp.61–66 (1995).
8) 渡辺 正，蕪木英雄，町田昌彦：熱伝導–対流系における原子運動の乱雑さ，第9回数値流体力学シンポジウム講演論文集，pp.325–326 (1995).
9) 折居茂夫，大田敏郎：分子動力学コードの段階的並列化手法，日本原子力研究所 JAERI-Data/Code96-023 (1996).
10) 折居茂夫，松山雄次，大田敏郎，久米悦雄，相川裕史，熊倉利昌，滝川好夫：並列計算機用ベンチマークテスト・システムの研究開発，計算工学講演会論文集，Vol.2, pp.117–120 (1997).

11) 松山雄次，折居茂夫，大田敏郎，久米悦雄，相川裕史：プログラム並列化支援解析ツールkpx，日本原子力研究所 JAERI-Tech 97-017 (1997).

付　　録

A.1 forceプログラムのコンパイルリスト

```
        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
```

(平成9年 8 月6日受付)
(平成9年12月1日採録)

折居　茂夫（正会員）

昭和58年富士通（株）入社，以来スーパーコンピュータのプログラミング技術の普及に従事．昭和62年末から2年間，日本原子力研究所外来研究員として原子力コードのベクトル処理，並列処理に関する研究に従事．平成7年より日本原子力研究所に出向，計算科学技術推進センターに所属し並列処理の基礎・基盤技術研究開発に従事，現在に至る．