ハイパフォーマンス
コンピューティング 49－2
（1993．10．14）

並列計算機システムのスケーラビリティについて

佐藤三久　関口智嗣
（電子技術総合研究所）
E-mail:{msato, sekiguti}@etl.go.jp

スケーラビリティは並列計算機システムを特徴ずける用語として頻繁に使われているが、直観的な定義はあるものの、実際には、スケーラビリティとはプロセッサ数に関係するいくつかの性能評価指標で表現される。本稿では、その指標の一つとして、並列プログラムのスケーリング則を基にして、プログラムの問題サイズで正規化するサイズ相対効率を提案する。これによって、逐次時間を用いなくとも、効率についての振舞いを表現することができる。また、逐次部分比、現実的な計算時間を基準とする時間固定効率などの理解し易い性能指標についても述べる。これらを用いて、CM-5 上での並列プログラムの性能を解析を試みる。

Scalability Analysis on Parallel Computing Systems

SATO Mitsuhisa, SEKIGUCHI Satoshi
Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba, Ibalaki 305, Japan

Scalability is a frequently-claimed attribute of parallel systems. While the basic notion is intuitive, the scalability is described as a set of performance metrics in parallel computing systems. In this paper, we propose a new metric based on the scaling rule of the parallel program, which allows us to understand parallel behavior without the sequential execution time. Other useful metrics such as serial fraction and fixed-time efficiency are also discussed. We show these analysises of a parallel program on CM-5 as an example.

1 はじめに

ハイパフォーマンスコンピューテングを目指して、様々な並列アーキテクチャが提案されて、実用化されてきている。並列アーキテクチャでは、プロセッサ数を増やすことによりプロセッサ数に見合う性能向上が期待され、数百から数千プロセッサ規模の超並列コンピュータが登場するにいたっている。並列計算機システムでは、プロセッサ自体の性能に加えて、プロセッサ数とそれを結合する機構の性能、あるいは形態が大きく性能を左右する。逐次システムと比べて、アルゴリズムあるいはプログラムによっても、得られる性能が大きく異なることが多くなる。

本稿では、並列計算機システムのスケーラビリティに関する性能評価指標について議論する。ここでいう並列計算機システム（以降、並列システム）とは、並列計算機自体だけでなく、並列アルゴリズム、並列プログラミング、実行時のオペレーテングシステムなどを含むものである。「スケーラビリティ」なる用語は、「スケーラブルな並列コンピュータ」や「... の並列マシンはスケーラブルである」というように、マルチプロセッサシステムを特徴づける用語として頻繁に使われている。並列システムのスケーラビリティとは、直観的にいえば、

> 並列システムでの重要なパラメータであるプロセッサ数を増やすことにより、（プロセッサ数に見合う）多くの性能を得られる（あるいは、効率が低下しない）。

ことを意味する言葉である。しかし、「スケーラブルな」とか「スケーラビリティの良い」というような定性的で、かつ直観的な表現として使われてはいるものの、定量的な定義として広く受け入れられているものはない。

広い意味での並列計算機システムのスケーラビリティ（の意味）とはプロセッサ数を変化させた時のシステム性能の振舞いであるといえる。したがって、スケーラビリティはいくつかのプロセッサ数に関連するいくつかの性能評価指標によって表現されると考えられる。そこでは、何がどの様に良ければ、スケーラブルなのかが明確になっており、それによって適切な示唆が得られるものであることが望ましい。これまでの評価において、対逐次性能向上率がよく用いられているが、どの様な要因により変化しているのかを読みとるのが難しい場合が多く、実際上でも逐次性能が計測不能なことがあるなど、問題点が多い。

本稿では、2章において、スケーラビリティを考えるためのモデルについて述べ、3章において、並列システムの効率を中心としたいくつかの指標について提案し、考察を行なう。

2 スケーラビリティの評価モデル

2.1 線形性能モデル

これまで、並列システムの評価は逐次的なシステムを基準として行なわれていることが多い。性能（速度）向上率は逐次のプログラムの実行時間（性能）からのプロセッサ数を増やした時の実行時間の比（速度向上率）である。単一プロセッサでの逐次実行の時間を $T(1)$、p プロセッサでの実行時間を $T(p)$ とすると、性能向上率 $Speedup(p)$ は、以下の式で与えられる。

$$Speedup(p) = \frac{T(1)}{T(p)}$$

ここで、理想的な並列システムでは、

$$Speedup(p) = p$$

とするモデルが考えられる。このモデルは、「プロセッサ数を増やした分だけ、（線形に）性能が向上する」という素朴な期待に基ずくものである。この性能向上率はこのモデルに基ずく性能指標である。

2.2 Amdahl のモデル

並列システムを論ずる際の限界について、頻繁に言及されてきたのが有名な Amdahl の法則 [1] である。並列化されて実行されるプログラムは、並列に実行できる部分が限られている。あるいは、必ずなにがしかの逐次実行を伴うであろう。逐次実行という観点から、並列に実行できる部分の逐次での実行時間を T_p、逐次に実行しなくてはならない部分の実行時間を T_s とすると、理想的な並列システムではプロセッサ数 p での実行時間は、$T_p/p+T_s$ となる。そこでの性能向上率 $Speedup(p)$ は、以下の式で計算される。

$$Speedup(p) = \frac{T_p + T_s}{T_p/p + T_s}$$

Amdahl は、無限大のプロセッサを用いても、逐次実行の部分がなくならないため、T_p/T_s+1 を越えられないことを指摘した。例えば、並列プログラムのわずか 1% が逐次時間だとしても、1000 のプロセッサを用いても、実行時間を 1/100 にするこ

とはできないということである。線形なモデルに基ずく性能向上率は、実際はこの限界に抑えられていることになる。

2.3 並列プログラムのスケーリング則

さて、並列システムでのスケーラビリティは、どのようなプログラムを実行させるかによって大きく異なる。例えば、ほとんど同期・通信を行なわなくてもよいアルゴリズムのプログラムではほとんどの並列システムにおいて、良いスケーラビリティを達成できるであろう。また、逐次実行の部分はプログラムによって異なるため、達成できる性能も異なる。

並列プログラムのスケーリング則は、その並列プログラムの実行時間の理論的な式を与えるものである。ここで、プログラムが問題のサイズを与えるパラメータを持つ時にスケーラブルな並列プログラムということにする。この時、仮定している並列アーキテクチャモデルにおいて、プロセッサ数pでのサイズnの問題の実行時間を$T^*(p,n)$とする。スケーリング則は、この実行時間の理論的な見積もりを与える。

例として、行列の乗算を行なう並列プログラムを考えて見る。ここでは、配列は行ごとに分散配置するプログラムを考えると、それぞれのプロセッサ内にある部分配列を用いた計算と配列要素の転送の繰り返しになる。それぞれの繰り返しでの計算回数はn^3/p^2、転送するデータのサイズは、n^2/pになる。したがって、スケーリング則は、以下の様になる。

$$T^*(p,n) = \alpha\frac{n^3}{p} + \beta n^2$$

ここではアーキテクチャとして、メッセージ通信を用い、演算と通信がオーバーラップしないモデルを仮定しているが、スケーリング則は、仮定するアーキテクチャモデルによって異なる。αとβが決まれば、仮定するアーキテクチャモデルでの理想の実行時間を求めることができるが、実際問題として、非常に難しい。

スケーリング則の有用な点は、実際の理想的な性能を与えるというよりはむしろ、アルゴリズムの設計に際して、並列アルゴリズム自身の持つ漸近的なスケーラビリティを与えることにある。例えば、上の例では、計算部分がn^3であり、並列プログラムのオーバーヘッドであるn^2は、nが十分大きくなれば、十分な性能を得ることができることがわかる。

このスケーリング則は、問題サイズとプロセッサ数が同一ないわゆる並列アルゴリズムのオーダ式と異なるものである。

2.4 Gustafson のモデル

Gustafson[2] は、Amdahl の法則の重大な欠点として、問題のサイズによって、逐次部分と並列部分の実行時間比が変化することを考慮していないことを指摘した。現実的な場面において、プロセッサ数を多くするにしたがって、大きなサイズの問題のプログラムを実行させることが期待される。そこでは、前の節で取り上げた行列の乗算をはじめ、多くの問題において、問題のサイズを大きくすることによって、十分な性能を得ることができる。

Amdahl の法則は逐次実行からの性能向上率の限界を指摘するものであったが、実際の並列システムでは並列実行に対応する逐次時間を求めるのが非現実的な場合が多い。したがって、この限界はすでに意味を持たないことになる。そこで、並列実行を基準するとして、並列実行での並列部分の実行時間をT'_pとすると、この部分は逐次実行では、pT'_pだけの時間がかかるはずである。したがって、性能向上率は、

$$Speedup'(p) = \frac{T_s + pT'_p}{T_s + T'_p} = p + \frac{(1-p)T_s}{T_s + T'_p}$$

となり、並列実行での逐次部分での比でしか性能が低下しないことになる。もちろん、これは単に言い替えでしかない。この比は当然、プロセッサ数によって変化するが、プロセッサ数によって十分小さい値になっていれば、十分な性能向上を得ることができることを示している。

3 スケーラビリティに関する性能評価指標

本章では、いくつかのスケーラビリティに関する性能評価指標とその意味について議論する。その指標に関して、CM-5 上で実行した行列の乗算プログラムの結果の解析例を示す。

3.1 性能向上率の落し穴

$T(p,n)$をサイズnの問題をpプロセッサのシステムで実行した時の実行時間とする。ここで性能向上率$Speedup(p,n)$は以下の式で与えられる。

$$Speedup(p,n) = \frac{T(1,n)}{T(p,n)}$$

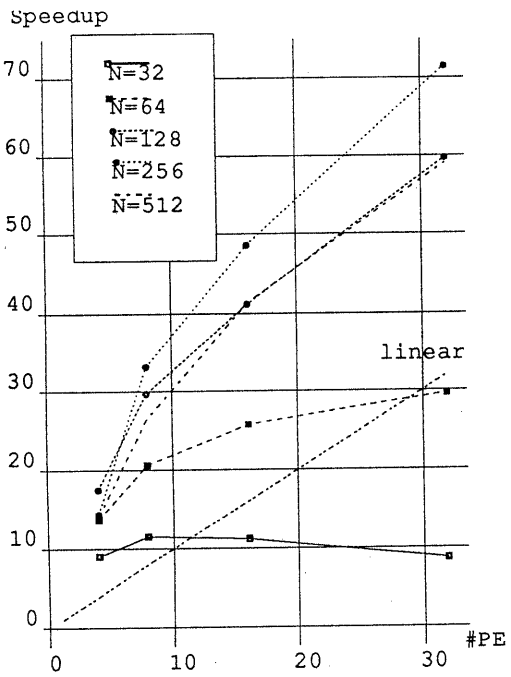

図 1: 性能向上率

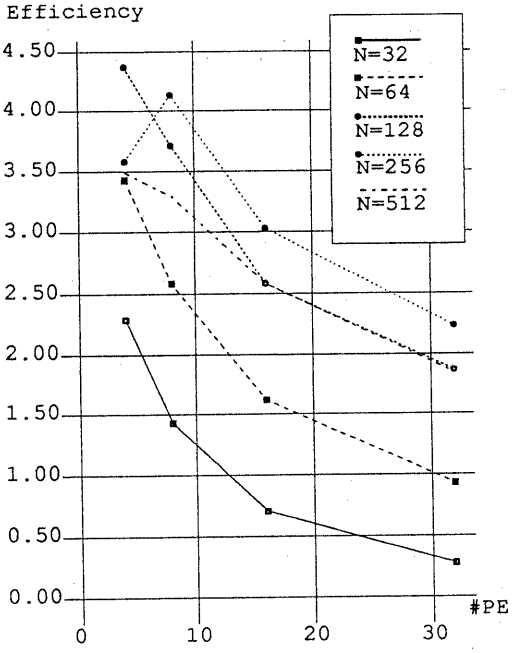

図 2: 線形性能モデルでの効率

この指標は基本的に線形な性能モデルに基づいたものであり、線形な性能からの低下はオーバーヘッドとして表される。図 1に性能向上率を示す。ちなみに、CM-5 では並列化することによって、キャッシュの効果が現れ、super-linear な性能向上が得られていることがわかる。対逐次性能向上率から、スケーラビリティを読みとるには、以下の問題点がある。

- 本質的に必要な逐次部分があったとしてもこのオーバーヘッドに含まれることになる。
- 逐次実行を基準するため、この実行時間が計測できなければ、求めることができない。
- 逐次時間を基準にしなくてはならない根拠はない。逐次プログラムは同じ計算をしているものの、並列プログラムとは異なるプログラムである。

3.2 並列システムの効率

モデルでの理想的な実行時間をもとに、実際の実行時間との比として効率 $E(p,n)$ を計算することができる。

$$E(p,n) = \frac{T^*(p,n)}{T(p,n)}$$

非常に制限された理想的な意味でのスケーラブルなシステムとは、

> すべてのプロセッサ数 p、問題サイズ n、
> すべてのアルゴリズム（プログラム）
> に対して、$E(p,n) = 1$

となるが、実際はこの効率がスケーラビリティの指標を与えることになる。すなわち、

> プロセッサ数が増加しても、効率が低
> 下しない（かつ 1 に近い）並列システ
> ムはスケーラブルである。

線形モデルに基ずく場合は、効率 $E(p,n)$ 以下の式で与えられる。

$$E(p,n) = \frac{T(1,n)/p}{T(p,n)}$$

これは、$Speedup(p,n)/p$ であり、速度向上率をプロセッサ数 p で正規化したものである。図 2に線形モデルでの効率を示す。super-linear な性能向上している場合は、効率は 1 以上になる。対逐次性

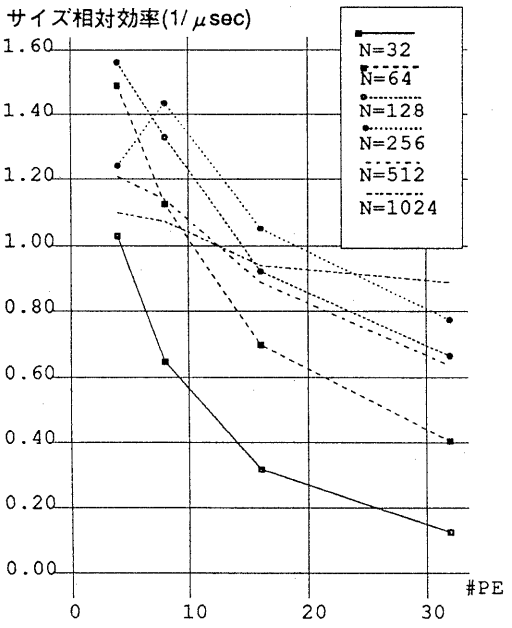

図 3: サイズ相対効率

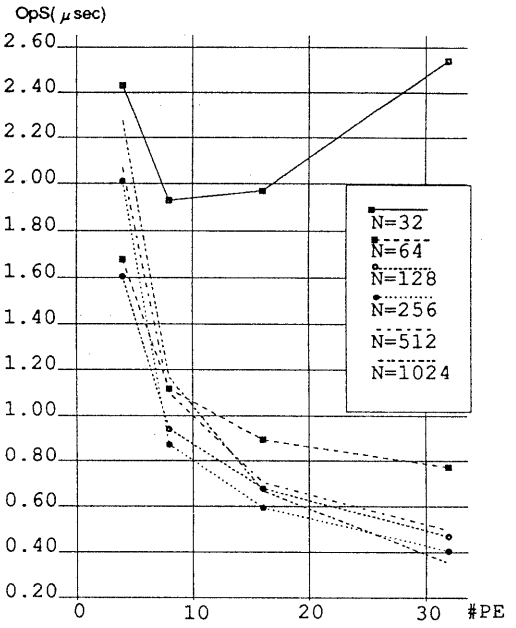

図 4: 実行 OpS

能向上率と同様な問題点があるが、正規化されているため、プロセッサを変化させた時の比較が容易になるという利点がある。

3.3 スケーリング則による正規化効率

スケーリング則は、その並列プログラムでの理論的な並列実行の時間の限界を与える。したがって、スケーリング則にしたがって、$T^*(p,n)$ を求めることができれば、真の効率を求めることができる。

ここでは、2.3 で求めた行列の乗算に関するスケーリング則から、効率を求めてみることにする。スケーリング則の式において、通信による項は実際には計算の項に比べて小さいので無視し、計算の項のみを用いれば、

$$E(p,n) = \frac{\alpha n^3/p}{T(p,n)}$$

となる。1 計算当たりの α で正規化した効率を

$$e(p,n) = \frac{n^3}{pT(p,n)}$$

で求めると、これは 1 計算あたりの効率になる。これにより、問題サイズによる影響を正規化することができるため、これをサイズ相対効率と呼ぶことにする。図 3にサイズ相対効率を示す。

この場合、スケーリング則による式は結局、逐次のものと同じであるため、サイズ相対効率は、逐次時間からの効率とほぼ同様な振舞いであることがわかる。

また、この値の逆数は 1 計算あたりの時間であり、いわゆる**実効 OpS (operations per second)** を示すものになる。(図 4)

サイズ相対効率の大きな利点は逐次時間を必要としないことである。実際、α の値を逐次実行時間から求めてみると、キャッシュが効いた場合は $0.04\mu sec$、効かない場合は $0.011\mu sec$ と大きく変動することがわかる。したがって、逐次実行時間では問題サイズを正規化することができない。サイズ相対効率では、効率が達成するべき効率にいかに近いか (すなわち、1 に近いか) は見ることはできないが、プロセッサ数の増加による効率の変化については見ることができる。

例では、スケーリング則として、通信の項は無視したが、実際、β の値は CM-5 のネットワーク性能から 5MB/s から 20MB/s の値が考えられる。5MB/s と見積もっても、通信項の値はサイズが 64 ですでに 1% 程度になる。

一般に、スケーリング則によるモデルの実行時間の計算は、想定するアーキテクチャモデルに依存する。PAX[8] では、PAX のアーキテクチャに基

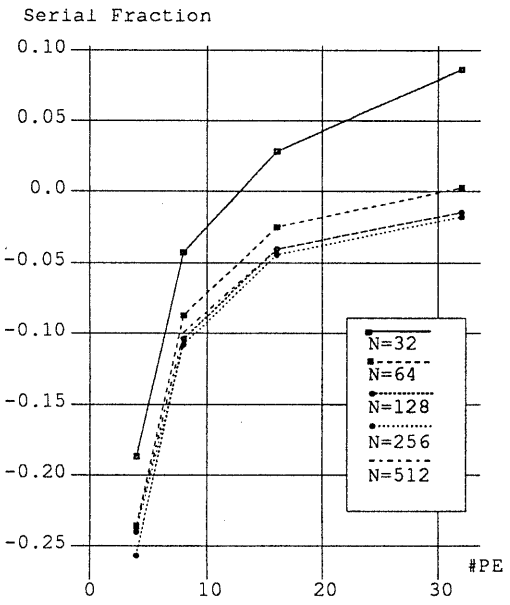

図 5: 逐次部分比 (serial fraction)

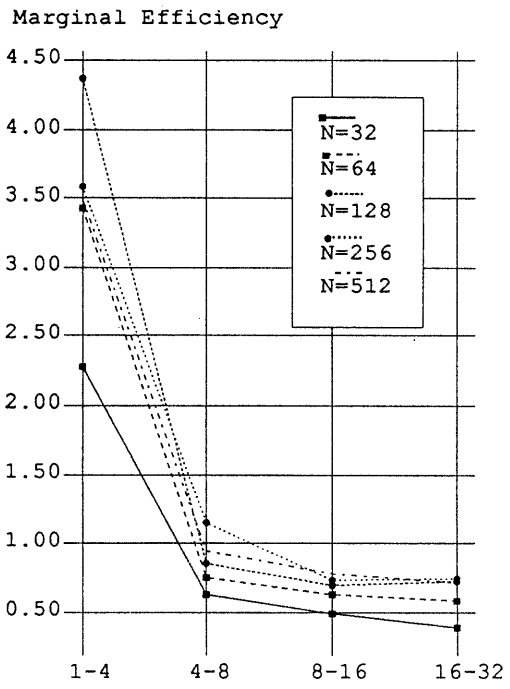

図 6: Marginal 効率

ずく正確なスケーリング則にしたがって性能評価を行なっている非常に優れた例となっている。しかし、一般的には、たとえアーキテクチャを固定したとしても、α と β などの実際の定数を決めるのは、プロセッサ間ネットワーク、同期機構などパラメータしなくてはならず、難しいことが多い。

3.4 逐次部分 (Serial fraction) の解析

Karp ら [3] は、逐次部分を解析することによって、並列システムの性能を評価する指標を提案している。Amdahl のモデルでの逐次部分比 (serial fraction) を f とすると、プロセッサ数 p での実行時間は、以下の式になる。

$$T(p,n) = T(1,n)f + \frac{T(1,n)(1-f)}{p}$$

したがって、f は、

$$f = \frac{T(1,n)/T(p,n) - 1/p}{1 - 1/p}$$

この解析では、アルゴリズム上除くことのできない逐次部分だけでなく、同期のためのオーバヘッド、ロードバランスのための影響も逐次部分の増加として観測される。図 5に逐次部分比を示す。

ここで注目したいのは、今の例では、スケーリング則での通信の時間はプロセッサ数に依存しないということである。したがって、理想の並列システムではこの値は問題サイズにしたがって、一定の値となるはずであり、サイズごとの逐次部分比の変化は並列化に伴うオーバヘッドによる。Gustafson の指摘にある問題サイズによって、逐次部分が変化する場合、その逐次部分がプロセッサ数に依存しないものであるならば、この指標ではその影響を除くことができる。そして、実際にこのような場合は多い。

また、プロセッサ数を連続値として解釈すれば、f は $1/E(p,n)$ を p で微分したものなっており、効率の低下の割合を示す値になっている。

3.5 Marginal 効率

スケーラブルな並列システムに期待されるものとして、プロセッサを増やしたときに性能が低下しないという特性がある。Merginal 効率とは、プロセッサを増やした時に、もとのプロセッサ数での効率を 100% と仮定した効率である。この性能指標は DASH[4] の性能評価に用いられている。図 6に計算した結果を示す。それぞれ、1 プロセッサから 4 プロセッサ、4 から 8、8 から 16、16 から 32 の効率を示す。

前に指摘した通り、逐次でのプログラムは並列

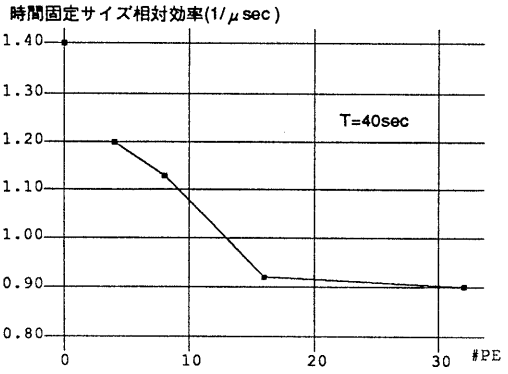

図 7: 固定時間効率

プログラムは別のプログラムである。したがって、1プロセッサから4プロセッサは他の値とは別の意味を持つ。Marginal 効率は並列システムにクラスタ構造を持つ場合などに便利である。また、超並列システムにおいて数台のプロセッサを1プロセッサとみて、数の少ない並列システムとの比較も可能である。

3.6 問題サイズによる評価指標

プロセッサ数が多くなれば、解ける問題サイズが大きくなり、制約される逐次部分も変化する。スケーラビリティを評価する場合に、問題サイズを固定した評価は大規模のプロセッサを不当に評価していることになる。

Gustafson は、ベンチマーク slalom において、並列システムのスケーラビリティの評価を逆に問題サイズで評価することを提案している。通常ベンチマークでは、問題サイズを固定して実行時間を求め、プロセッサ数による速度向上率、あるいは効率にを評価するが、slalom では実行時間を固定し（例えば1分）、その時間内で計算できる問題サイズ（radiosity 計算のパッチ数）を性能の指標とする。計測する時間を現実的な時間に設定することによって、現実的な状況でのスケーラビリティを評価することができるという利点がある。例えば、少ないプロセッサで多くの時間を必要とする問題でも、多くのプロセッサで一瞬で終ってしまうならば、その問題での評価は現実的な状況では役に立たない結果でしかない。この指標は、すでに完成している並列システムを比較評価するのには有用である。

さて、この考え方をプロセッサ数を変化させた場合の評価に適用してみる。プロセッサ数ごとのサイズと実行時間の関係から、一定時間で計算できる問題のサイズを求める。行列の乗算の場合、サイズは離散的にしか求まらないが、便宜的にサイズを推定した。実行時間 40sec では、プロセッサが4の時には、サイズが 550、8 の時には 650、16 の時には 800、32 の時は 1050 となる。スケーリング則から、理論的には $N = (p/\alpha)^{1/3}$ の曲線となる。

これらの点でのサイズ相対効率を図7に示す。これを固定時間（サイズ相対）効率と呼ぶことにする。実際には、線形モデルでの効率を用いてもよいが、時間を固定した場合には、大きなプロセッサで求めた並列実行時間に対する逐次は求まらない場合が頻発する。ちなみに CM-5 上ではサイズ 1024 に対応する逐次時間は計測できなかった。

図3の問題サイズごとのサイズ相対効率と比較してみると、時間固定効率はいろいろなサイズでの効率をうまく summarize していることがわかる。

プロセッサと問題サイズの関係を表現する指標はスケーラビリティに関して、見透しのよい示唆を与える。例えば、固定時間効率は、「ある問題（行列の乗算）に関して、プロセッサの数を増やした時に、問題サイズを適当に大きくして、どのような効率が得られるか」の一つの現実的な解になる。

プロセッサと問題サイズの関係を与える指標としては、他に iso-efficiency[5] がある。iso-efficiency とは、プロセッサ数が与えられた時に、ある一定の効率を達成するために必要な問題サイズを与えるものである。例えば、ベクトルコンピュータの性能モデルでのパラメータである $N_{1/2}$ はこの iso-efficiency の一つであり、並列システムの場合はプロセッサ数 p の関数になる。

4 おわりに

これまでの多くの並列システムの評価では、線形性能モデルに基づいて、一つのサイズで行なわれることが多かった。その問題点とは、以下の2点にまとめることができる。

(1) 逐次時間を基準とすること。

(2) サイズによる逐次部分の変動を考慮できないこと。

(1) に関しては、実際上測定不能であることがあり、その場合は性能評価ができないということになってしまう。また、並列プログラムと逐次プログラムは別のものであり、逐次プログラムを基準にする根拠はない。(2) については、スケーラビリティ

を論ずる場合にはプロセッサ数を増やすにしたがって、問題のサイズを大きくするのは自然な要求である。問題サイズを変化させることによって、問題固有の逐次部分が変化し、並列システムで得られる性能も変化するため、サイズによる評価は不可欠である。

本稿では、スケーリング則に基づいて、サイズ相対効率を提案した。スケーリング則が明らかになっている時には、逐次時間を用いなくとも、効率についての振舞いを表現することができる。また、オーバーヘッドを表現するものとして逐次部分比による評価指標を示した。逐次部分比はサイズを固定することによって、理想値が一定値になるため、理解し易い指標となる。さらに、現実的な計算時間を基準とする時間固定効率を提案し、例について示した。実際、この効率は「適当な」サイズによる効率として用いることができ、様々なサイズでの並列システムの効率のよい summary になる。

効率はその基本とするモデルにより計算されるが、アーキテクチャ側にとって、真の効率とは理想的な計算モデルからの効率である。実際にはそれを知ることは不可能である。共有メモリに対する理論的なマシンとしては、PRAM などが考えられるが、オーダはしることができても、実際の理想の実行時間を与える定数を決めることは恣意的なものになってしまわざる得ない。スケーリング則はこのモデルから計算されるものであり、(1) にある問題点はなくなったとしても、この点は本質的な問題点として残ることになる。

少なくとも、ベンチマークとしての性能評価では、サイズ固定での性能向上率を用いることはやめるべきである。ベンチマークとしては、スケーラブルなプログラムでかつスケーリング則が明確なものを用い、解析的な振舞いがわかるものについて行なうべきであろう。

性能評価（ベンチマーク）は、計算機システムの使用者に対しては対象とするプログラムの性能推定の手段として、計算機システムの設計者に対しては多くのプログラムを代表するワークロードとしての２つの面を持つ [7]。ある特定のベンチマークで得られた結果はそのプログラムのものであって、他の全く異なるプログラムに適用することは難しい。他のアプローチとしては、同期、通信などプログラムの構成要素についての性能を測定するものがある。我々は並列プロセッサの重要な操作である同期性能に関して、時間当たりの同期回数SYOPS を提案している [6]。他にも、データ並列プログラミングモデルの基本的なオペレーションについての時間を測定し、スケーラブルであるかどうかを評価する方法もある。

さて、２つの並列システムについて、どちらが「スケーラブル」であるか答えることができるであろうか？また、遅いプロセッサを多くつかった「スケーラブル」なシステムと少ない速いプロセッサの「スケーラブル」でないシステムではどちらがいいのだろうか？これからの課題として、本稿で議論した性能評価指標を異なるシステムの比較という観点から考察してみたい。

謝辞　本研究を遂行するにあたり、日頃有益な議論を頂く山口計算機方式研究室長ならびに計算機方式研究室の同僚諸氏に感謝いたします。特に、東京大学 平木助教授には貴重な助言、議論を頂きました。CM-5 の結果は、佐藤の共同研究の一環として RWC つくば研究センターにて計測させていただいたものです。

参考文献

[1] G.M. Amdahl. Validity of the single-processor approach to achieving large scale computing capabilities. *AFIPS Conference Proceedings*, 30, 1967.

[2] J.L. Gustafson. Reevaluating Amdahl's Low. *CACM*, 31(5), May 1988.

[3] A.H. Karp and H.P. Flatt. Measuring Parallel Processor Performance. *CACM*, 33(5), May 1990.

[4] D. Lenoski, J. Laudon, K. Gharachoroloo, W.D. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M.S. Lam. The Stanford Dash Multiprocessor. *IEEE Computer*, March 1992.

[5] V. Singh, V. Kumar, G. Agaha, and C. Tomlinson. Scalability of Parallel Sorting on Mesh Computer. In *Proc. of International Parallel Processing Symposium*, pages 92–101, 1991.

[6] 関口、佐藤. 細粒度並列処理における性能評価モデル - 節度ある並列性を求めて - . *1992* 並列処理シンポジウム論文集, 1992.

[7] 関口、佐藤. HPC の性能評価 — 測定から科学へ —. 情報処理学会研究報告 *HPC-46-5*, 1993.

[8] 星野、他. *PAX* コンピュータ. オーム社, 1985.