ハイパフォーマンス
コンピューティング 52−22
（1994．7．23）

スケーラビリティによる並列計算機システムの比較

佐藤三久 関口智嗣 長嶋雲兵†
電子技術総合研究所
お茶の水女子大学†
E-mail:{msato, sekiguti}@etl.go.jp

並列計算機システムのスケーラビリティとはプロセッサを増やした時の性能の振舞いであるが、そのスケーラビリティ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。我々は、その指標として、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や現実的な計算時間を基準とする時間固定効率などの指標を提案している。本稿では、複数の並列計算機システムをとり上げ、簡単な行列乗算を例にそれらの指標により、スケーラビリティの比較を試みる。

Scalability Analysis on Parallel Computing Systems: A Case Study

SATO Mitsuhisa, SEKIGUCHI Satoshi, NAGASIMA Umpei†
Electrotechnical Laboratory
Ochanomizu University†

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. We proposed a new metric, *size-scaled efficiency,* based on the scaling rule of the parallel program, which allows us to understand parallel behavior without the sequential execution time, and fixed-time efficiency. In this paper, we compare a few parallel systems on parallel matrix multiply program with these proposed scalability metrics.

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開発されている。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。また、イーサネット等で結合されたワークステーションで並列プログラムの実行を行うワークステーションクラスタもコストパフォーマンスのよい並列システムとして期待されている。

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。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。

広い意味での並列システムのスケーラビリティ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。

我々は、前稿 [4] において、並列プログラムのスケーリング則に基づく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、この指標によって、逐次実行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。さらに、現実的な計算時間を基準とする固定時間効率を提案した。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。

本稿では、超並列マシンである CM-5 と Paragon、SUN4 のワークステーションクラスタについて、これら並列計算機システムを、並列行列乗算プログラムを例に、スケーラビリティという観点から、比較を試みる。2 章において、スケーラビリティ評価の問題点について議論し、3 章において、我々が用いるスケーラビリティの評価指標について述べる。4 章では、実際のデータからのスケーラビリティの評価について示す。

2 スケーラビリティ評価の問題点

これまで、並列システムの、とくにスケーラビリティに関する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かった。すなわち、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）を用いて評価を行う。この評価は、「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づくものである。逐次実行時間からの性能向上率から、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線形性能からの低下分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、逐次実行時間が計測できなければ、求めることができない。実際上、測定不能であることがある。
- 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 (ことが多い)。

また、Gustafson[1] が指摘しているように、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。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、スケーラビリティを論ずる場合には、サイズによる評価は不可欠である。

3 スケーラビリティ評価指標

サイズ n の問題を計算するプログラムを p プロセッサで実行した時の実行時間を、$T(p,n)$ とする。理想的なモデルでの実行時間 $T^*(p,n)$ をもとに、実際の実行時間との比として真の効率 $E(p,n)$ が求まる。

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

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** といえる。実際、対逐次性能向上率は並列実行によりプロセッサ分だけ、向上するという線形モデルを理想のモデルにしたものである。

スケーリング則によるモデル化 — 真の効率を求めるためには、「理想的な実行時間」T^* を与える必要がある。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見積りを与える式である。しかし、スケーリング則では基本的な演算の実行時間やデータの通信速度などをパラメータとしているので、その値を求めるためには実行するシステムごとにそれらのパラメータを決定する必要がある。しかしながら、現実のシステムではこれらのパラメータを決定するのは非常に難しい。実際には、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。

サイズ相対効率 — スケーリング則から、T^* を n と p の初等関数 S で以下のように近似できる時、

$$T^*(p,n) \simeq \alpha S(n,p)$$

これを使って、効率を求めることにする。具体的には、関数 S はプロセッサ数でスケーリングした問題サイズを与える関数である。また、α は問題サイズあたりの理想的な計算量を与える未知のパラメータである。効率を S を用いて計算し、

$$e(p,n) = E(p,n)/\alpha = S(p,n)/T(p,n)$$

とすると、e は、これはサイズ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。これは、実際には各プロセッサにおいて単位時間で計算できる問題サイズを与えることになる。サイズ相対効率の大きな利点は逐次実行時間を必要としないことである。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。

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。**Marginal 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。プロセッサ数 p_1 からプロセッサ数 p_2 への Marginal 効率は $(T(p_1,n)*p_1)/(T(p_2,n)*p_2)$ で計算されるため、サイズ相対効率が、$S(p,n) = S(1,n)/p$ で与えられている場合には、$e(p,n) = S(1,n)/(p * T(p,n))$ となるため、サイズ相対効率の比になり、サイズ相対効率の傾向を示すものになる。

固定時間スケーラビリティ — 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。また、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。

Gustafson は、ベンチマーク slalom[2] 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。この指標を**固定時間サイズ**と呼ぶことにする。さらに、このサイズでのサイズ相対効率を**固定時間（サイズ相対）効率**と呼ぶことにする。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。固定時間効率によって、「**与えられた問題に関して、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を与えることができる。

4 並列行列乗算でのスケーラビリティ

ここでは、実際の例として、2次元ブロックに分割配置した行列乗算の並列プログラムについて、以下のシステムのスケーラビリティを考えてみることにする。

Intel Paragon/XP — 32 プロセッサシステム (プロセッサは i860 50MHz, ネットワークは 2 次元メッシュ、200 MB/sec(カタログ値、実行速度は 30MB/sec 程度))

Thinking Machine Corp. CM-5 — 32 プロセッサ (プロセッサは、SPARC 32 MHz、キャッシュ 64KB。ネットワークは、fat tree。速度は 5MB から 20MB。)

ワークステーションクラスタ — SUN4 IPC(SPARC 25 MHz) をイーサネットで結合したワークステーションクラスタ。25 台まで用いた。

いずれも、メッセージ通信のライブラリ (CM5 では、CMMD、Paragon では、NX ライブラリ、

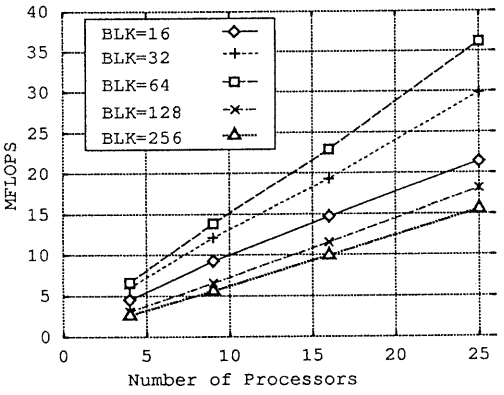

図 1: CM-5 の実行結果

ワークステーションクラスタでは、PVM3.2) を用いた。各システムのプログラムは、ライブラリの違いを除けば、ほぼ同じものである。なお、プログラムはシステムのデフォールトの最適化オプションでコンパイルした。

プログラムとスケーリング則 — 行列乗算プログラムは、各プロセッサに対して 2 次元にブロックに分割された行列 A と B を乗算し、結果の行列 C も同じく 2 次元ブロック配置で求めるものである。計算を行う行列全体のサイズを $N \times N$、プロセッサ数を $P = p^2$ とすると各プロセッサごとに配置されているブロックのサイズは $n = N/p$ となる。

アルゴリズムは、まず、列のうちの 1 つのプロセッサが B のブロック行列を同じ列の他のプロセッサに対してブロードキャストし、その行列と各プロセッサの A の行列との乗算を行う。次に、行列 A を行方向のプロセッサにシフトする。これを、p ステップだけ繰り返す。1 要素あたりの計算時間を α, 1 データあたりの転送時間を β とすると、1 ステップで必要な演算時間は、$\alpha n^3 + \beta p n^2$ になる。したがって、スケーリング則は、$T^*(P,N) = \alpha N^3/P + \beta N^2$ となる。

測定データとして、各プロセッサごとのブロックのサイズを、16,32,64,128,256 として、プロセッサ数を 4,9,16,25 とし、25 プロセッサまでの実行時間を求めた。最大で、5*256 = 1280 までの図 1、2、3に実行結果を示す。各ブロックサイズでは、ほぼ、linear に性能が向上していることがわかる。

ちなみに、測定に用いたプログラムでは、各プロセッサで各ステップで行っている行列乗算はブロック化による最適化はおこなっていない。充分に大きいサイズでの、逐次実行による行列演算の性能はそれぞれ、CM5 では 0.5MFLOPS, Paragon では 1 ～ 2 MFLOPS, SUN4 IPC では、0.6 MFLOPS

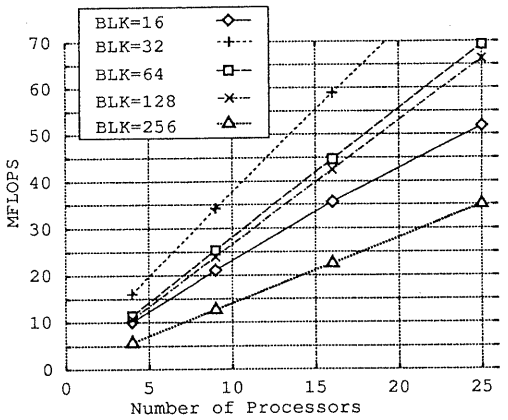

図 2: Paragon の実行結果

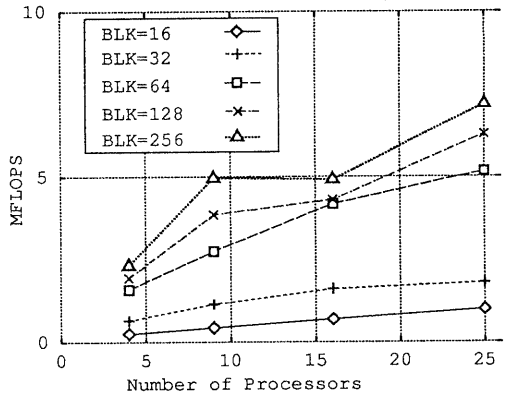

図 3: PVM/SUN4 の実行結果

である (FLOPS は行列乗算に必要な要素の演算を 1FLOPS とする)。

サイズ相対効率 — 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、$S(p,n) = n^3/p$ とすることができ、$E(p,n) = (\alpha n^3/p)/T(p,n)$ となる。したがって、サイズ相対効率は、$e(p,n) = n^3/pT(p,n)$ で与えられる。図 4、5、6に問題サイズごとのサイズ相対効率を示す。問題サイズは離散的であるが、スケーリング則から、N^2 で正規化することによって、N の一次式になることから、これで補間して求めた。

問題サイズが小さい場合は、プロセッサ数が増えることによって、各プロセッサごとのサイズが極端に小さくなるため、効率の低下が著しい。逆に、問題サイズが大きい場合には、プロセッサ数が増えることにより、ブロックのサイズが小さくなっていくため、キャッシュの効果により、効率が向上する傾向が見られる。しかし、問題のサイズがさらに大

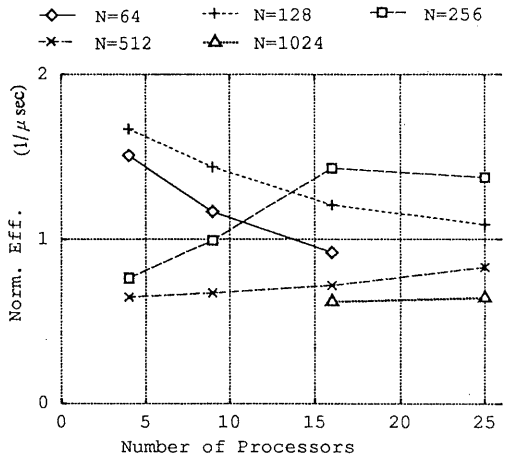

図 4: CM-5 のサイズ相対効率

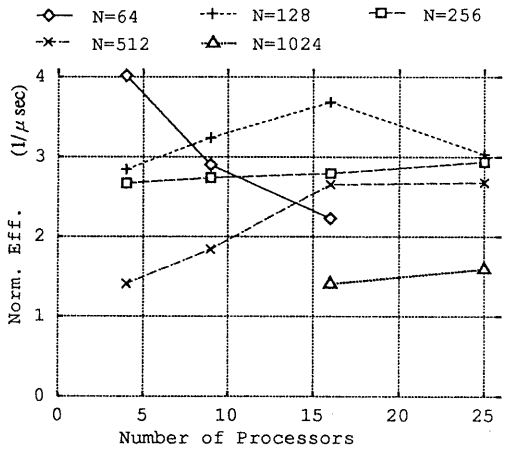

図 5: Paragon のサイズ相対効率

きい場合には、各プロセッサのブロックのサイズが依然として大きいため、あまり効率は向上しないが、計算の方が通信に比べて非常に大きいため、ほぼ一定になる。ワークステーションクラスタでは、通信が遅いため、問題サイズが小さい範囲では効率が悪いが、問題サイズが大きくなるにつれて、その影響が軽減されるようになるが、大きい問題サイズの範囲なのでキャッシュによる効果はなくなってしまう。

Marginal 効率の比較 — Marginal 効率は、サイズ相対効率の傾向を示す効率である。すなわち、サイズ相対効率が一定に近い、すなわち、効率が低下しない場合には、1 に近くなる。小さいサイズと大きいサイズでの Marginal 効率を図 7、8に示す。効率が 1 を越える場合には、逆にプロセッサ数を増やすことによって、ブロックのサイズが小さくなり、通信のオーバヘッドが多少増加しても、効率が向上することを示している。この効率が 1 以下で一定であ

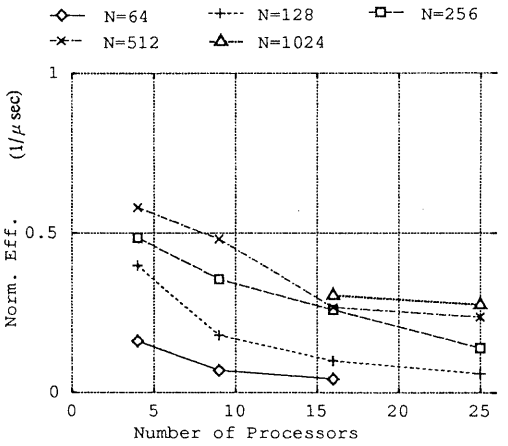

図 6: PVM/SUN4 のサイズ相対効率

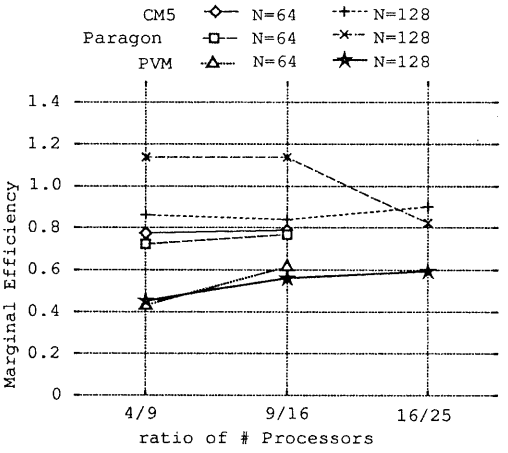

図 7: 小さいサイズでの Marginal 効率

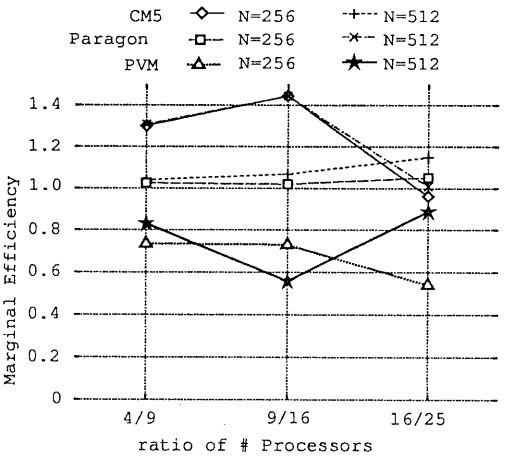

図 8: 大きいサイズでの Marginal 効率

ることは、次第に効率が低下していくことを示している。なお、この問題では計算が N^3 で増加してい

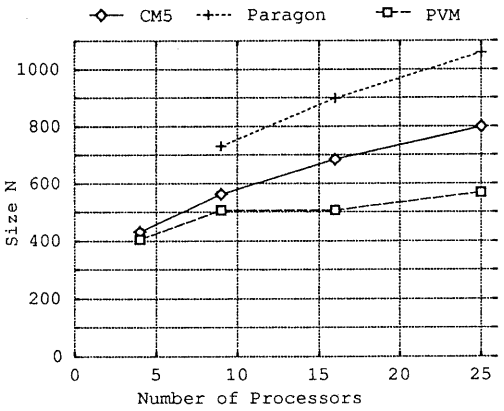

図 9: 固定時間サイズ (T=30sec)

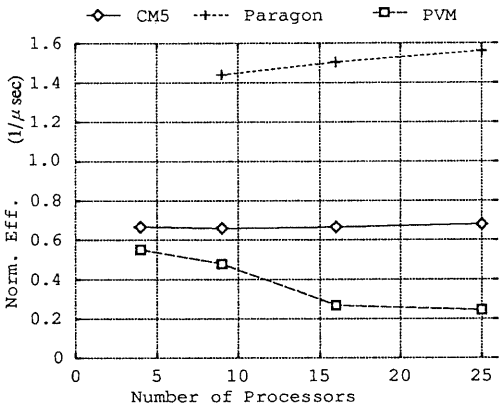

図 10: 固定時間サイズ相対効率 (T=30sec)

くため、いずれのシステムでも問題サイズが大きくなると、次第に 1.0 に近付き、性能が低下しにくくなる。

固定時間サイズ相対効率 — 最後に、全体的な傾向を比較するために、固定時間の問題サイズと効率を求めた。ここでは 30sec に時間を固定した。図 9、10に示す。問題サイズは次第に増大している。これを効率という観点からみると、CM-5 と Paragon ではほぼ一定になっている。この 30sec という時間はすでに、充分大きな問題を計算しているため、効率は変化しない。ワークステーションクラスタでは、プロセッサ数が少ないサイズでは、効率が比較的良いため、このプロセッサ数にしては大きいサイズの問題を計算することできるが、プロセッサ数を大きくしていくと効率が低下するため、あまり大きい問題を計算することができなくなってしまう。(特に、4 プロセッサの範囲では通信がそれほど障害にならないために、CM-5 に近い効率である。)

5 おわりに

本稿では、複数の並列システムについて、行列乗算を例に、スケーリング則に基づくサイズ相対効率、Marginal 効率、固定時間効率を示し、スケーラビリティについて考察した。

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。Hill[3] が指摘しているように、実際にはそれを知ることは不可能である。オーダは知ることができても、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、この点は本質的な問題点として残っている。実際のプログラムではスケーリング則は非常に複雑であり、探索問題など、スケーリング則を決めることのできないものもある。しかし、データ並列のプログラムなど、単純なスケーリング則の組合せで求まるものも多い。

これまで多くの並列システム評価に用いられている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。少なくとも、ベンチマークとしての性能評価では、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。

謝辞 本研究は科技庁原子力特別研究『複雑現象の解明における超高速計算機利用技術の研究』および，電子技術総合研究所とお茶の水女子大学の共同研究契約に基づくものである．本研究を遂行するにあたり、日頃より議論を頂く電子技術総合研究所太田公廣情報アーキテクチャ部長、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、お茶の水女子大学理学部細矢治夫教授に感謝致します。CM-5 ならびに Paragon 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。また、SUN のワークステーションクラスタの結果はお茶の水女子大学のシステムにて計測したものです。

参考文献

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

[2] John Gustafson, Stephen Elbert Diane Rover, and Michael Carter. The First Scalable Supecomputer Benchmark. *Supercomputing Review*, pages 56–61, 1990.

[3] M. D. Hill. What is Scalability? *Computer Architecture News*, 19(2), 1991.

[4] 佐藤、関口. 並列計算機システムのスケーラビリティについて. 情報処理学会研究報告 *HPC-49-2*, 1993.