ハイパフォーマンス
コンピューティング 62－7
（1996. 8. 28）

メモリ参照の局所性に関する定量的な評価

田中 淳裕
NEC C&C 研究所
E-mail omochi@sbl.cl.nec.co.jp

キャッシュや仮想記憶が有効に働くための原理として『メモリ参照の局所性』が一般に信じられている。しかしながらこれらの性質は依然として定式化されていない。そのため、定量的な議論が困難であった。本稿では、時間的・空間的なパラメータを含んだ形でヒット率を定義し、これを元に、局所性の指標を定義する。このような指標を定義することにより、同様の計算を行なう異なるアルゴリズム間の優劣・効率を考慮することが可能となる。具体例としてウエーブフロントの計算を行なう複数のアルゴリズムを考え、それらの局所性の評価を行なった。

Quantitative Evaluation of the Locality of Memory References

Atsuhiro TANAKA
C&C Research Laboratories, NEC Corp.
E-mail omochi@sbl.cl.nec.co.jp

"The locality of memory references" have been believed as the reason that cache and virtual memory system work effectively. This property have not been previously formalized, therefore, quantiative evaluation have been difficult to carry out. This article first defines hit ratio including spatial and temporal parameters, and then defines locality indexes. Comparisons among different algorithms based on locality become possible with these indexes. Localities for some programs of wavefront algorithm, and optimization method of blocked algorithm are quantitatively evaluated.

1 はじめに

ノイマン型計算機におけるプログラムの仕事とは，何らかの方法で記憶装置の内容を変化させることである．プログラマはノイマン的プログラミング言語を用いてメモリと CPU との間のデータの流れ逐一制御する必要がある．記憶装置が階層化された計算機では，記憶領域へのアクセスパターンを意識したプログラムでなければ，その能力 (特に CPU) を十分に発揮することはできない．

キャッシュや仮想記憶が有効に働くための原理として『参照の局所性』の原理が一般に信じられている．つまり，プログラムは一般に次のような特徴を有していると言われている．

- 時間的局所性： あるデータをアクセスした時には，近い将来にそのデータをもう一度参照することが多い．
- 空間的局所性： あるデータをアクセスした時には，その近くのデータを参照しやすい．

この原理は古くから指摘されているのにもかかわらず，一向に定式化されていない．本稿の目的の一つは，この『参照の局所性』を定量的に測る指標を提案することである．

局所性を定義することにより、プログラムの優劣・効率を考慮することが可能になる。例えば、アルゴリズムの評価指標としては、計算量のオーダに基づくものが一般的であるが、実際に計算を行なうとオーダが同じ場合でも計算時間にかなりの差が出る場合が多い。今回提案する指標を用いることにより、オーダが同じプログラム同士に、局所性という物差しで測った優劣をつけることができる。

本稿では，最初に，時間的・空間的パラメータを含んだ形でヒット率を定義し，この定義に基づく局所性の指標を示す．次にウェーブフロントの計算を具体例として用い，指標の妥当性を示す．

2 局所性

この節では局所性を示す指標の定義を行なうために必要な幾つかの概念を最初に定義する。その後、局所性を表す指標を述べる。

2.1 諸定義

参照列 ：プログラムが時刻 t に参照するデータのアドレスを r_t で示す．ここで，時刻 t は離散時間とし，1 回のメモリ参照を単位として増加するものとする (つまり，実時間とは物差しが異なる)．またプログラムが参照するアドレス空間は 0 以上の整数と仮定する。

長さ k の参照系列 $r_1\ r_2\ \cdots\ r_k$ を ρ_k と書く．

以下では数学的議論を簡単にするために，参照列にエルゴード性の成り立つ弱定常過程を仮定する．

時間近傍 ：時刻 t での参照アドレス r_t に対して，ウィンドウサイズと呼ばれるパラメータ $T(>0)$ を用いて，以下のような量 $W_T(r_t)$ を定義し，これを r_t の T- 時間近傍と呼ぶ．

$$W_T(r_t) := \{r_k \mid k \in [\max(0, t-T+1), t]\} \quad (1)$$

これは，ワーキングセットの定義 [2] に他ならない．

空間近傍 ：時刻 t での参照アドレス r_t に対して，スケールサイズというパラメータ $\delta(>0)$ を用いて，以下のような量 $U_\delta(r_t)$ を定義し，これを r_t の δ- 空間近傍と呼ぶ．

$$U_\delta(r_t) := \{\, r \mid r \ \mathrm{div}\ \delta = r_t \ \mathrm{div}\ \delta\} \quad (2)$$

ただし，空間近傍の定義は上記の一種類だけとは限らない．例えば，

$$U_\delta(r_t) := \{r \mid r_t - \delta/2 \le r < r_t + \delta/2\} \quad (3)$$

のように定義してもかまわない．要は，何らかの意味で，参照したデータの隣近所のデータからなる集合を構成すれば良いのである．

ヒット率 ：時刻 t でのヒット率を，ウインドウサイズ T とスケールパラメータ δ を用いて以下のような確率として定義する．

$$H_t(T,\delta) := Pr\{r_t \in \bigcup_{r \in W_T(r_{t-1})} U_\delta(r)\} \quad (4)$$

この定義の心は，現在参照したアドレス r_t が，過去 $t-T, \cdots, t-1$ に参照したアドレス，及びその周辺のアドレスに含まれるかどうかを確率としてとらえているということである．

このように定義することにより，時間的・空間的な要因がヒット率へ与える影響を統一的に考えることが可能になる．例えば，$\delta = 1$ として考えると，$H_t(T,1)$ は，同一アドレスへの参照が起こる頻度 (確率) と考えられる．すなわち時間的局所性の形式的な表現である．同様に，$T = 1$ とした時には，連続したアクセスが空間的にどのような広がりを持っているかを表している．すなわち空間的局所性の表現である。

上記の定義は全て、ある時刻 t での量に関する定義である。以下での議論は、これらの量の時間平均の極限値 $\lim_{t\to\infty}\sum_t F_t(\cdot)/t$ を考えて、それらを集合平均とみなすことにする[1]。

ヒット率 / ミス率の計算方法　異なるウインドウサイズに対して、平均ワーキングセットの大きさをワンパスで求めるアルゴリズムは、1974年に既に提案されている。今回は、このアルゴリズムを援用して、(4) 式で定義した $H(T,\delta)$ のテーブルの値を計算した。スペースの関係で詳しくは述べないが、事前に参照アドレスの最大値と最小値がわかっていれば、ワンパスで全ての値を計算することができる。

2.2　局所性の評価指標

全体的な局所性　ミス率 α が与えられた時に、$\{(T,\delta)\mid 1-H(T,\delta)\geq\alpha\}$ という集合を考え、その面積 $S(\alpha)$ で全体的な局所性を定義する。これは、あるミス率 (α) を達成するために必要なウインドウサイズ・スケールパラメータの領域の大小で、プログラムの局所性を評価しようということである。従って、$S(\alpha)$ の値が小さいほど局所性は大きいことになる。

例えば，同様の計算をするが具体的な計算方法の異なる 2 つのプログラム A, B があるときに，$S_A(\alpha)$ と $S_B(\alpha)$ の大きさを比べることにより、プログラム A と B との局所性を比較することができる。

ここで注意すべきことは，$S_A(\alpha)\leq S_B(\alpha)$ が成り立っても，$S_A(\beta)\leq S_B(\beta)$ が必ずしも成り立つとは限らないということである (具体例は後述する).

時間局所性 / 空間局所性　時間方向の局所性を表す指標を次のように考える。まず δ を固定し、ミス率 $M(T)=1-H(T,\delta)$ を T の関数と考える。次にこの関数の n 次モーメント $\int T^n M(T)dT$ の n 乗根が時間方向全体の局所性と考える。ミス率が早く 0 に近づくほど局所性が高いはずなので、n 乗根が小さいほど局所性が高いことになる。空間方向の局所性も同様である。

以下では、時間方向の n 次モーメントの n 乗根を μ_n、空間方向の n 次モーメントの n 乗根を ν_n と記述する。

[1] 参照列にエルゴード性と弱定常性を仮定したのはこのためである。

3　具体例：ウェーブフロント

例としてウェーブフロントの計算を行なう複数のプログラムを取り上げる。ウェーブフロントの計算とは、n 次元配列 $g[]$ の要素間に以下のような関係が成り立つ場合に、計算の進み具合がちょうど波の先頭に似ていることから名付けられた。

$$g[\boldsymbol{x}]=\sum_{i=1}^{n} g[\boldsymbol{x}-\boldsymbol{e}_i]$$

ここで、$\boldsymbol{x}$ は n 次元ベクトル、e_i は n 次元単位ベクトルである。

本稿では、特に 2 次元の場合に関して、上式に基づき計算を行う異なる 2 つのアルゴリズムを取り上げ，それらの局所性に関して論じる．また，局所性を増加させる手法と言われているブロック化の効果も調べる．

超平面法　$k=x+y$ を満たす直線上の配列の値を更新していく方法。k の値が小さい方から大きい方へ向かって計算すれば、配列を上書きしながら計算が行なえる。

```
for ( k = 1; k ≤ 2n; k++ )
    while ( x + y = k )
        g[x][y] = g[x − 1][y] + g[x][y − 1]
```

単純ループ　次のような単純なループのネスト構造で配列の値を計算する方法。

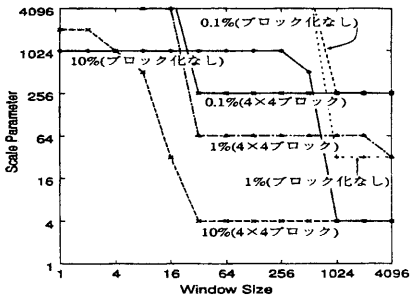
```
for ( x = 1; x ≤ n; x++ )
    for ( y = 1; y ≤ n; y++ )
        g[x][y] = g[x − 1][y] + g[x][y − 1]
```

ブロック化　配列のサイズが大きい場合に、$n\times n$ の領域を、$m\times m$ の小領域 (もちろん $m<n$ である) に分割し、それぞれの領域に前述の超平面法や単純ループ法を適用する技術である。局所性を増加させる技法と言われている [1, 6]。

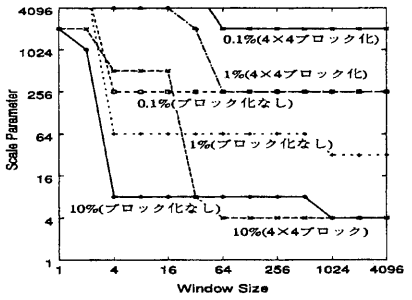
4　結果と評価

表 1 に 256×256 の配列に対して演算を行なった時のミス率を示す。ウインドウサイズ、スケールパラメータをそれぞれ 2^0 から 2^{12} まで変化させ、2 の冪に対して測定を行なった。表には 4 の冪のミス率が示されている。

この表の結果をグラフ表示したものが、**図 2, 3** である。



(A) 超平面法



(B) 単純ループ

図 1: 全体的な局所性を示すグラフ

表 1: 256 × 256 の配列演算に対するミス率

(A) 超平面法: ブロック化なし

	T =4	16	64	256	1024	4096
δ =4	0.407	0.407	0.404	0.367	0.0653	0.0653
16	0.353	0.353	0.351	0.314	0.0171	0.0171
64	0.340	0.340	0.337	0.301	5.2e-3	5.2e-3
256	0.336	0.335	0.333	0.297	1.0e-3	1.0e-3
1024	0.0855	0.0854	0.0845	0.0746	2.5e-4	2.5e-4
4096	0.0232	0.0231	0.0226	0.0192	7.0e-4	7.0e-4

(B) 超平面法: 4×4 ブロック

	T =4	16	64	256	1024	4096
δ =4	0.425	0.170	0.0863	0.0863	0.0863	0.0643
16	0.371	0.104	0.0213	0.0213	0.0213	0.0158
64	0.358	0.0884	6.2e-3	6.2e-3	6.2e-3	4.0e-3
256	0.353	0.0834	1.0e-3	1.0e-3	1.0e-3	1.0e-3
1024	0.0938	0.0208	2.5e-3	2.5e-3	2.5e-3	2.5e-3
4096	0.0238	5.2e-3	7.0e-4	7.0e-4	7.0e-4	7.0e-4

(C) 単純ループ: ブロック化なし

	T =4	16	64	256	1024	4096
δ = 4	0.149	0.149	0.149	0.149	0.0643	0.0643
16	0.0368	0.0368	0.0368	0.0368	0.0158	0.0158
64	9.16e-3	9.16e-3	9.16e-3	9.16e-3	3.95e-3	3.95e-3
256	1.0e-3	1.0e-3	1.0e-3	1.0e-3	1.0e-3	1.0e-3
1024	2.5e-4	2.5e-4	2.5e-4	2.5e-4	2.5e-4	2.5e-4
4096	7.0e-5	7.0e-5	7.0e-5	7.0e-5	7.0e-5	7.0e-5

(D) 単純ループ: 4×4 ブロック化

	T = 4	16	64	256	1024	4096
δ =4	0.255	0.191	0.0863	0.0863	0.0863	0.0643
16	0.141	0.125	0.0220	0.0220	0.0220	0.0158
64	0.113	0.109	6.21e-3	6.21e-3	6.21e-3	3.95e-3
256	0.104	0.104	1.0e-3	1.0e-3	1.0e-3	1.0e-3
1024	0.0310	0.0208	2.5e-4	2.5e-4	2.5e-4	2.5e-4
4096	7.88e-3	5.19e-3	7.0e-5	7.0e-5	7.0e-5	7.0e-5

ウインドウサイズとミス率との関係、スケールパラメータとミス率との関係がそれぞれ示されている。また、**表 2, 3** に時間方向・空間方向の局所性の指標を示す。

図 1 は全体的な局所性を示すグラフである。ミス率 $\alpha = 10\%, 1\%, 0.1\%$ に関して、測定点でのミス率が α を越えない最小の点をプロットし、それらを直線で結んでいる。囲まれたグラフの左下方向の領域が、ミス率が $\alpha\%$ を越える領域と近似的に考えられる。

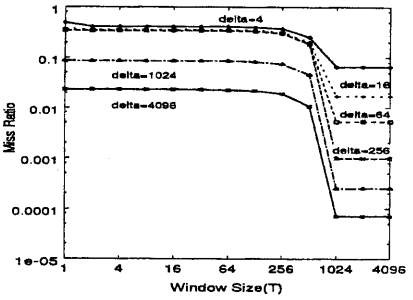
この図を用いて異なる計算方法の比較を行なうことができる。ブロック化をしない場合には、超平面法よりも単純ループ法の方が圧倒的に局所性が高いということとがわかる。また、ブロック化をした場合には、ミス率が 10% では単純ループ法の方が局所性が高くなるが、ミス率が 1%, 0.1% では超平面法の方が局所性が高くなる。

図 1 (A) を用いて、超平面法におけるブロック化の効果を確認することができる。ミス率 10%, 1%, 0.1% のいずれの場合にもブロック化をすることにより、局所性が大幅に向上している。ただし細かく見ると、ブロック化しない方がミス率が小さい領域も存在している。

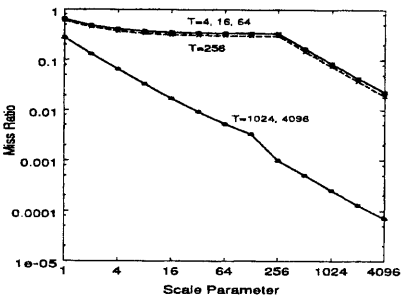
単純ループ法におけるブロック化の効果は**図 1 (B)** を見ることで確認できる。こちらの場合にはブロック化をした方が局所性は劣化している。ただし、ブロック化を行なうことによりミス率が改善される領域も存在する。

表 3, 4 に局所性の成分 (時間・空間方向) を示す。ブロック化をしない場合には、ウインドウサイズが比較的小さければ、単純ループ法の空間的局所性が、超平面法のそれに比べて圧倒的に高く (モーメントが小さく) なっている。ただし、ウインドウサイズがある程度大きくなるとほとんど変化はない。時間的局所性は、だいたいにおいて単純ループ法が高くなっている。

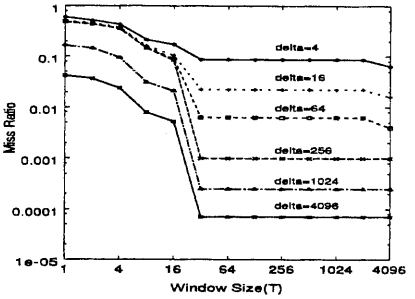
ブロック化することにより、超平面法では時間と空間方向の両方の局所性が向上している。一方で、単純ループ法では局所性の向上はほとんどほとんど認められない。興味深いことは、ブロック化をした場合の局所性が、超平面法と単純ループ法とでほとんど等しくなっていることである。



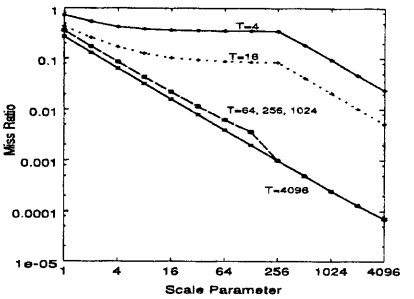
(A) ブロック化なし



(B) ブロック化なし



(C) 4 × 4 ブロック化



(D) 4 × 4 ブロック化

図 2: 超平面法のミス率

表 2: 局所性の成分 (超平面法)

(A) 時間方向の局所性 (n 次モーメント)

	ブロック化なし			4 × 4 ブロック化		
δ	μ_1	μ_2	μ_3	μ_1	μ_2	μ_3
4	596	1610	2480	467	1623	2497
16	297	878	1594	121	810	1567
64	223	569	1092	36	413	994
256	197	404	683	11	193	612
1024	50	201	431	3.5	97	386
4096	13	100	274	1.7	50	248

(B) 空間方向の局所性 (n 次モーメント)

	ブロック化なし			4 × 4 ブロック化		
T	ν_1	ν_2	ν_3	ν_1	ν_2	ν_3
4	400	1115	1836	427	1144	1859
16	400	1114	1833	98	537	1120
64	395	1104	1821	4.1	61	260
256	347	1025	1728	4.1	61	260
1024	3.7	60	260	4.1	61	260
4096	3.7	60	260	3.5	60	260

5 関連研究

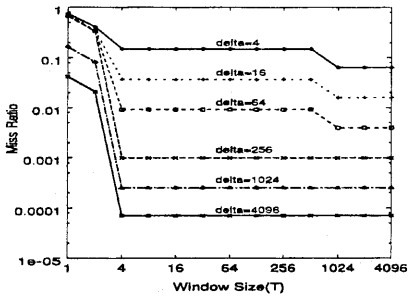
Random Walk モデル [8] や SGF(Stack Growth Function) モデル [5] では、プログラムの総メモリ参照回数 (v) と、重複を除いた参照メモリ数 (u) との関係を考察している。局所性という意味からすると u/v の大小で局所性の大小をとらえるのが直接的かもしれない。これらのモデルは、空間的・時間的局所性を明確に定義しているわけではないが、$u = f(v)$ という関数を考え、この関数の微分・差分の形でヒット率・ミス率が導き出されるのは、非常に興味深い。

文献 [4] は、形式的に局所性を定義しようとしている。ヒット率 $H(T, \delta)$ の空間 (δ) 差分を考えて、それを時間方向の確率分布として考えている。そのうえで、それが表す曲面がプログラムの動作特性 (メモリ参照特性) を示すと考えている。近傍ではなく、距離という概念で議論を進めているところが本稿の定義と異なるところである。

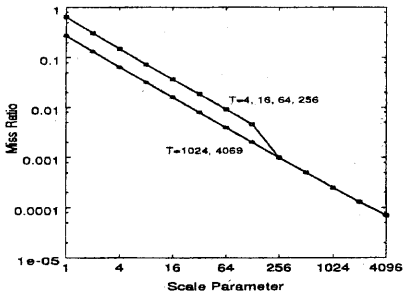
6 まとめ

プログラムのメモリ参照の局所性を表す指標を提案するために、時間的・空間的近傍を用いてヒット率というものを定義した。これにより、時間的・空間的な要因を同時に考慮することが可能となる。

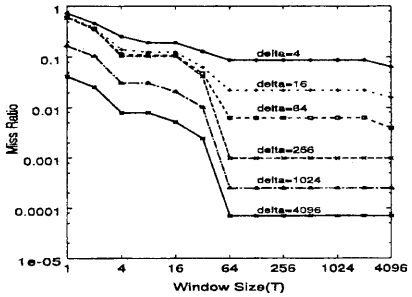
局所性の指標として、面積を用いて全体的な局所性を表すことと、空間方向・時間方向の関数の n 次モーメントで局所性を表すことを提案した。例としてウエーブフロントを取り上げ、これらの指標の効果を示すために、異なる計算方法の比較を行なった。



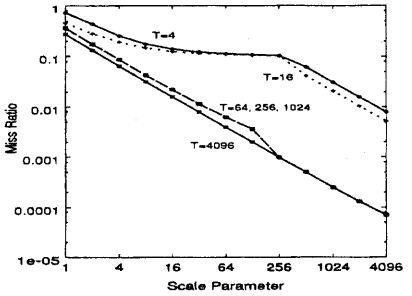
(A) ブロック化なし



(B) ブロック化なし



(C) 4 × 4 ブロック化



(D) 4 × 4 ブロック化

図 3: 単純ループ法のミス率

表 3: 局所性の成分 (単純ループ法)

(A) 時間方向の局所性 (n 次モーメント)

	ブロック化なし			4 × 4 ブロック化		
δ	μ_1	μ_2	μ_3	μ_1	μ_2	μ_3
4	462	1574	2464	468	1623	2497
16	115	781	1544	121	810	1567
64	30	390	972	36	413	994
256	7.9	193	612	11	194	612
1024	7.7	97	386	3.5	97	387
4096	1.5	50	248	1.7	50	248

(B) 空間方向の局所性 (n 次モーメント)

	ブロック化なし			4 × 4 ブロック化		
T	ν_1	ν_2	ν_3	ν_1	ν_2	ν_3
4	5.4	61	261	140	657	1285
16	5.4	61	261	107	540	120
64	5.4	61	261	4.1	61	261
256	5.4	61	261	4.1	61	261
1024	3.5	60	261	4.1	61	261
4096	3.5	60	261	3.5	60	261

参考文献

[1] D. F. Bacon, S. L. Graham, and O. J. Sharp, Compiler Transformations for High-Performance Computing, *Computing Surveys*, Vol. 26, No. 4, pp.345–420, 1994.

[2] P. J. Denning, The Working Set Model for Program Behavior, *CACM*, Vol. 11, No. 5, pp. 323–333, 1968.

[3] P. J. Denning and S. C. Schwartz, Properties of the Working-Set Model, *CACM*, Vol. 21, No. 9, pp. 750–759, 1978.

[4] K. Grimsrud, J. Archibald, R. Frost, and B. Nelson, On the Accuracy of Memory Reference Models, *LNCS 794*, pp. 369–388, 1994.

[5] M. Kobayashi and M. H. MacDougall, The Stack Growth Function: Cache Line Reference Models, *IEEE Trans. on Comput.*, Vol. 38, No. 6, pp.798–805, 1989.

[6] M. S. Lam, E. E. Rothberg, and M. E. Wolf, The Cache Performance and Optimizations of Blocked Algorithms, *ASPLOS IV*, pp.63–74, 1991.

[7] D. R. Slutz and I. L. Traiger, A Note on the Calculation of Average Working Set Size, *CACM*, Vol. 17, No. 10, pp.563–565, 1974.

[8] D. Thiébaut, On the Fractal Dimension of Computer Programs and Its Application to the Prediction of the Cache Miss Ratio, *IEEE Trans. on Comput.*, Vol. 38, No. 7, pp.1012–1026, 1989.