計算機アーキテクチャ 123－11
（1997. 3. 7）

WSの大規模数値演算におけるメモリ参照能力

上原 哲太郎† 國枝 義敏‡

† 和歌山大学 情報処理センター ‡ 和歌山大学 システム工学部

〒640 和歌山市栄谷930

TEL 0734-54-0361 FAX 0734-54-0402(Center), 0734-54-0134(Fac. of Sys. Eng.)

E-Mail: tetsu@center.wayakama-u.ac.jp, kunieda@sys.wakayama-u.ac.jp

コンパイラでのメモリ参照の最適化の実現に向けて予備データを得るため、パソコンやワークステーションのメモリ参照性能を行列積の実行時間によって測定した。多重ループのブロック化によって簡単に性能の改善が実現できた。そのブロックサイズの決定は2次キャッシュ容量だけで決定すればよく、他の要因はほとんど無視できるとの結論を得た。

Memory-access performance of the large-scaled numerical computation on current workstations

Tetsutaro UEHARA† Yoshitoshi KUNIEDA‡

‡ Information Processing Center, Wakayama University.

‡ Faculty of Systems Engineering, Wakayama University.

930 Sakaedani, Wakayama-city, Wakayama 640, Japan.

Toward the realization of automatic localization of memory access by optimizing compilers, we have evaluated the memory performance of current workstations and PCs by measuring the time of matrix multiply. According to the results, it is prospective that we can easily improve the performance of large-scaled numerical computations on WSs and PCs by tiling the nested loop. The size would be easily determined by the size of the secondary cache and the other influences are almost ignorable.

1 研究の背景

マイクロプロセサの急激な発達により、ワークステーション (以下 WS) やパーソナルコンピュータ (以下 PC) の演算速度は近年飛躍的な伸びを示している。これに伴い、従来スーパコンピュータで解かれてきた大規模数値演算の一部はすでに WS や PC でも処理可能となってきた。計算に多くの資金を投じることの出来ない研究者にとって、WS や PC を大規模数値演算のために利用する機会はますます増えてきていると考えられる。

WS や PC で巨大配列上での演算を行う上で問題となるのは、主記憶-CPU 間のデータ転送速度である。現在の WS,PC の主記憶は CPU の演算速度に対して十分なデータ供給能力を持っているとはいいがたく、データがキャッシュに収まり切らないような演算では性能は著しく低下する。そこで通常はユーザはブロック化 [1] と呼ばれる手法の採用などにより、できるだけキャッシュ上で演算が行われるよう工夫を加え、速度の向上をはかるべきである。

しかし、実際にはこの作業は以下のような理由でユーザに大きな負担を強いる。

- ターゲットアーキテクチャ、特にキャッシュ容量、連想方式やキャッシュラインの長さなどを考慮してプログラムをチューニングする必要がある。キャッシュラインの重複の回避などでは、キャッシュの挙動に関しての知識も必要である。

- 一種のアルゴリズム変換になるため、元の演算アルゴリズムを熟知していないと作業自体が困難である。

- ブロック化などの作業後のプログラムは複雑になり、デバッグ、改良などが困難になる。また、アーキテクチャへの依存度が高まるため、問題が大きくなった場合にベクトルスーパコンピュータなどへプログラムを移植する作業が困難になる。

特に最後の問題は、WS や PC とスーパコンピュータとの間でプログラムを共有し、計算規模によって使い分けようとする上で大きな障害となる。我々は、PC,WS を「ミニスーパコンピュータ」として利用してゆくために、ユーザをこのメモリ問題のチューニング作業から解放することが必要である。

ユーザをこの作業から解放するためには、以下の 3 つの解決策が考えられる。

1. 行列の基本的演算や主要なアプリケーションについてキャッシュなどに対し最適化されたライブラリを提供する。例えば LAPACK の blas[3] に相当するライブラリを提供する。

2. コンパイラにより自動的にブロック化などの処理を行わせ、キャッシュの有効利用を図る。商用コンパイラでは一般的ではないが、研究段階では SUIF などで実装例がある [5]。

3. ハードウェアの支援により、キャッシュに収まらない場合もある程度の性能が出るよう工夫する。例えば Alpha MicroProcessor などに見られる Cache のプリフェッチ機構 [6] や、SR2201 の要素プロセサ HARP-1 の疑似ベクトル機構 [4] などが挙げられる。

この 3 つの解決策は、番号が若いものほど実現が容易だが、ユーザにとっては番号の大きいものほど柔軟な解決で好ましいと考えられる。

このいずれの解決策をとるにせよ、PC や WS において、いわゆる「メモリウォール問題」がどれほど深刻であるかその度合いを示し、どのような対策を行えばどれほど性能が改善されるかの予測に役立つような何らかの指標が必要なのではないかと、我々は考えている。則ち、

1. プログラムをチューンするユーザやライブラリの作成者にとっては、そのチューニングによってどの程度の性能向上が得られるのか。

2. コンパイラ作成者にとっては、どういうプログラムパターンを検出し、どのような変換を行えばどの程度の性能向上が得られるのか。

3. アーキテクチャ設計者にとっては、現状で主記憶の性能がアプリケーションにどの程度の性能低下をもたらしており、どのような解決を取りうるのか。

このような考察の手がかりが必要であると感じている。とりわけ、我々の研究的興味は、最初の1,2の項目にある。

そこで本稿では、現状のPC,WSにおいてメモリ性能がどの程度の状況にあるかまとめ、ソフトウェアのチューニングやコンパイラでの利用のためにその性能を表す指標について考察する。

2 現状のPC,WSの主記憶構成

現在出荷されている主な高性能マイクロプロセサは、200MHzから500MHz程度のクロックで動作し、1つないし2つ程度の浮動小数点ユニットを備える。浮動小数点ユニットはパイプライン化されているので1サイクル毎に結果を生成することができる。例えば現在最高速のマイクロプロセサとされるAlpha21164-500は、500MHzで動作し、浮動小数点乗算と加減算の2つのパイプラインを持つ[6]。これらは並列動作可能なので、ピーク時には1GFlopsの性能がある。PentiumPro-200なら、200MHz動作でピークは200MFlopsとなる。

一方データ供給能力は、レジスタ-キャッシュに比べ、主記憶への直接能力は極端に低下する。1GFlopsを達成するためのデータ供給能力は最大の場合で、読み込みに16GB/s、書き出しに8GB/sとなるが、Alpha21164-500の場合、チップ上の8kBの1次データキャッシュの性能がこの半分程度あるのに対し、主記憶では通常のDRAMを利用している場合で最大でも約1GB/s程度の能力である[1][9]。PentiumPro-200でも、必要なデータ転送量(読みだし3.2GB/s書き込み1.6GB/s)に対し、1次キャッシュでデータ供給能力のみ半分、主記憶は266MB/s程度[2]である。各キャッシュ間のデータ転送速度も含めて表にしたものを表1に示す。

このいずれの例をとってもわかる通り、データの供給能力(すなわち読み取り性能)で比べると、1次/2次キャッシュと主記憶にはAlphaで8倍、PentiumProで6倍の能力差がある。すなわち、ある計算についてデータがキャッシュにできるだけ載るように変換するだけで、数倍は性能が向上することが想像できる。これは他の各種最適化技法の効果と比べても極めて大きな効果であるといえる。

3 主記憶参照の最適化技法

一般に、ワークステーションで行列演算などを行う際には次のような手法を用いてチューニングを行うべきである。

1. 最内ループでの主記憶参照ができるだけ連続アドレスになるように、かつ最内ループで可能な限りデータの再利用が行われるようにループ交換や添字変換、展開[3]する。また、例えば2次元行列で横方向の要素数が2の巾乗にあるような場合は、それを2の巾でないように変換する。後者は、キャッシュラインの重複を防ぐ効果が大きい。

2. 最内ループで可能な限りデータがキャッシュに載るように計算のブロック化を行う。

3. 必要なら、最内ループでキャッシュ利用が疎にならないように頻繁に引用されるブロックを連続アドレス領域にコピーする。これによりロードストアコストが増大するが、キャッシュ利用率が上がる。

[1] 33MHz,256bit幅構成の場合

[2] 外部クロックを66MHzで使用し、SDRAMを利用した場合。1回のバースト転送(32byte)に8サイクルかかると仮定。

[3] Wavefrontのようなデータフローの演算では添字変換とSkewingが効果を発揮する。例は文献[5]などに詳しい。

CPU	Alpha21164	PentiumPro	R10000
Clock(MHz)	500	200	195
1st Cache size	Inst. 8kB Data8kB	Inst.8kB Data 8kB	Inst.32kB Data.32kB
2nd Cache size	Unified 96kB	Unified 256 or 512kB	(Off-chip)
Peak Perf.(GFlops)	1	0.2	0.39
Reg. ⇐ 1st Cache (GB/s)	8	1.6	1.56?(未確認)
Reg. ⇒ 1st Cache (GB/s)	4	1.6	1.56?(未確認)
1st ⇔ 2nd Cache (GB/s)	8	1.6	3.12
2nd ⇔ 3rd Cache (GB/s)	2	–	–
Main Memory (GB/s)	1	0.266	?(未確認)

表 1: マイクロプロセサにおけるキャッシュと主記憶の参照性能

4. 最内周の計算が演算器の並列性を生かせるように展開する。これにはアンロールなどの手法が簡単に利用できる。

文献 [10] によると、PentiumPro-200 を搭載した PC で 500x500 の 2 次元行列の積の計算に内積型ループを用いた場合、ブロック化前は 20MFlops 程度だったのがブロック化後 90MFlops となっている。それに対し他のチューニング技法、上記 3,4 を適用すると 100MFlops 以上にはなるが、ブロック化ほど劇的改善ではない。、他の操作を行う余裕がない場合はまずブロック化だけは行っておくと、比較的少ない作業量で大きな効果を得ることができると予想できる。

ここで、どの程度の効果が得られるか事前に予想できないかという疑問が生じる。図 1を見る限り、PentiumPro の場合 2 次キャッシュにデータが載ればデータ参照は主記憶より 6 倍程度速い。逆にこの数値を越えることは絶対にないので、ある種の目安となる。文献 [10] では 4.5 ~5 倍強の性能が出ていることになるので、比較的良い目安といえる。

4 WS,PC における行列積の計算とブロック化

我々自身も文献 [10] と同様のテストを行った。利用した WS, PC は以下のようなものである。

Pentium Pro • CPU : Pentium Pro(200MHz)
- Cache : 1 次:命令 8kB、データ 8kB、2 次:統合 256kB
- 主記憶: 64MB(ファストページモード DRAM)
- OS : BSD/OS 2.1
- コンパイラ : GCC 2.7.2 (-O3 -m486 -funroll-loops を指定)

Alpha VT433 • CPU : Alpha21164(433MHz)
- Cache : 1 次:命令 8kB、データ 8kB、2 次:統合 96kB、3 次:統合 1MB
- 主記憶: 64MB(ファストページモード DRAM)
- OS : WindowsNT 4.0
- コンパイラ : Visual C++ 4.0J (「最大限に最適化する」を指定)

PowerChallenge10000 • CPU : R10000(195MHz)
- Cache : 1 次:命令 32kB、データ 32kB、2 次:統合 1MB
- 主記憶: 2GB
- OS : IRIX 6.2
- コンパイラ : MIPSpro C 7.0 (-mips4 -O3 -LNO:opt=1)

プログラムは単純な内積型行列積と、それをブロック化したもののみ用意した。それぞれについて行列の大きさは 100x100 から 100 刻みで 500x500 まで、またブロックの大きさはそれ

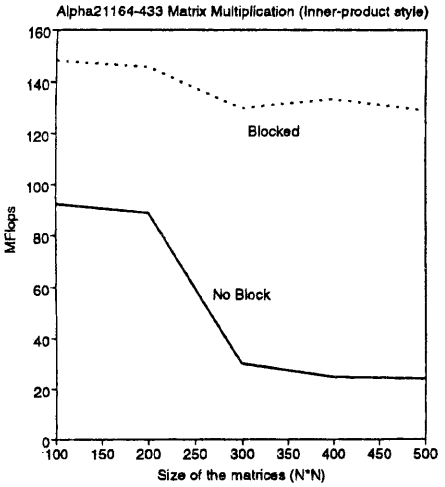

図 1: Alpha21164-433 の行列積の性能 (ブロック化前/後)

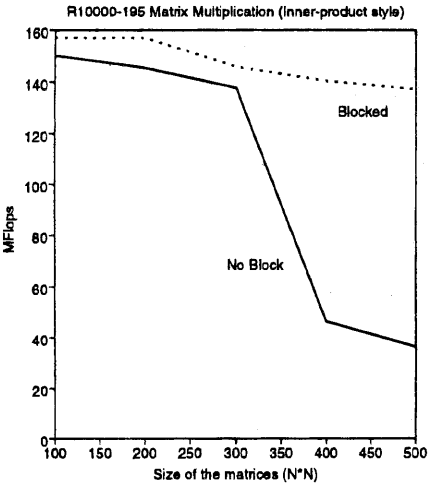

図 2: R10000-195 の行列積の性能 (ブロック化前/後)

それについて 10x10 から 5 刻みで行列全体の 1/2 (つまり 100x100 の行列の積を求めるときは 50x50 まで) を計算した。結果を図 1234に示す。なお図 123にあるブロック化時の計算速度は、大きさ毎にもっとも速かったブロック長によるものから抜粋されている。

この実験結果から以下のようなことが読み取れる。

- どの CPU においても、500x500 程度に大きな行列に対する計算では、単純なブロック化によって 3〜6 倍程度の性能向上が望める。これは主記憶のメモリ参照能力と 1 次キャッシュでのデータ供給能力の比と大きくは違わない。

- ブロックの大きさは、2 次キャッシュにおさまる程度がもっとも性能が良い。つまり、行列積の場合、ブロック長を B とすると B 行 B 列の大きさの小行列が繰り返し参照されるからである。PowerChallenge10000 のように大きな 2 次キャッシュを備えた計算機では、125/250 といった大きなブロックを切っても性能が低下しない。

- 図中にはないが、R10000-195 で行列の大きさを 512x512 としたところ、単純な行列積の性能は 3.47MFLops にまで低下した。これは 500x500 の時の 1/10 以下である。これは明らかにキャッシュラインの重複による現象である。これを防ぐため、行列そのものの大きさが 2 の巾乗 (大きな行列ではその整数倍も) にならないように注意する必要がある。

- 逆に、2 の巾乗の行列など極端な例を除いては、キャッシュの細かい挙動は性能に大きくは影響していないようである。

- キャッシュの利用効率はそれほど限界まで高めなくともかなりの性能が出ることが図 4からわかる。例えば Alpha ではブロック長が 20 でも 25 でも 50 でもそれほど性能はかわらない。

5 おわりに

行列積による実験を通して、WS,PC での大規模行列演算におけるメモリ性能の低さは、極端なキャッシュラインの重複だけは回避しつつ、2 次キャッシュ容量だけに着目したブロック化を

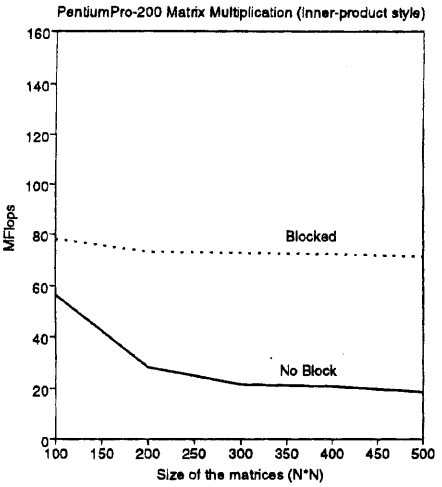

図 3: PentiumPro-200 の行列積の性能 (ブロック化前/後)

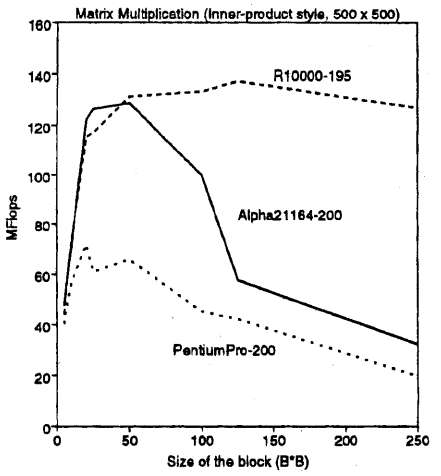

図 4: ブロック長による性能の変化 (500x500 行列の積の場合)

行うだけでかなり改善できることがわかった。よってチューニングなどの性能改善の手がかりとしては、キャッシュ容量だけを指標としてこのような方針をとっても十分実用的であろうと考えられる。

今後の研究としては、この結果に基づいて自動的にメモリの参照局所性を改善するコンパイラの開発などの研究につなげてゆきたい。

参考文献

[1] 寒川光, "数値計算プログラミングにおけるデータ移動制御のためのブロック化アルゴリズム". 情報処理学会論文誌 Vol.33, No.10, pp.1183–1192, 1992.

[2] M.J.Wolfe, "More iteration space tiling". *Supercomputing '89*, Nov. 1989.

[3] J.Dongarra, J.Du Croz, I.Duff, and S.Hammarling. "A Set of Level 3 Basic Linear Algebra Subprograms". *ACM Transactions on Mathematical Software*, 16(1):1–17, 1990.

[4] 斎藤 拡二 他, "疑似ベクトル機構を有する並列コンピュータ向け RISC プロセッサ". 信学技報 DSP95-104/ICD95-153, pp.1–6, 1995.

[5] M.E.Wolf and M.S.Lam, "A Data Locality Optimizing Algorithm", *Proceedings of the ACM SIGPLAN'91 Conference on Programming Language Design and Implementation*, 1991.

[6] Digital Equipment Corporation, "Alpha21164 Microprocessor DataSheet", July 1996.

[7] MIPS Technologies, Inc. "R10000 Microprocessor User's Manual Version 2.0", Dec. 1996.

[8] R.P.Colwell, R.L.Steck, "A 0.6um BiCMOS Processor with Dynamic Execution", *Proceedings of ISSCC95*, Feb. 1995.

[9] Digital Equipment Corporation, "AlphaPC164 Motherboard Technical Reference Manual", Aug. 1996.

[10] 前野年紀, 村上弘, "パソコンの行列演算性能の評価", 情報処理学会研究報告 06-HPC-64, pp.7–12, 1996.