

記号処理ベクトル化のための FOL 法の効率評価について

4 H-9

目木 信太郎

岡山県立大学情報工学部

1 まえがき

大規模数値計算に対する要求から、強力なパイプライン演算器を備えたベクトル計算機が開発され、実用に供されている。ベクトル計算機はもともと数値計算用として開発されたが、浮動小数点数だけでなく整数もベクトル処理可能であるために、ソーティングなどの非数値計算に応用する研究もなされている [2]。

ベクトル計算機では複数の演算が並列に実行されるので、一度にベクトル処理する演算は並列に実行可能なものでなければならない。したがって、記号処理のような並列性を持たない処理はそのままではベクトル処理することはできない。

金田はこの点に注目し、記号処理のような処理もベクトル化する手法として FOL(Filtering-Overwritten-Label) 法を開発した [1]。FOL 法を用いると記号処理などもベクトル化できるが、そのために若干のオーバーヘッドを伴う。[1] ではうまくいく場合の計算量が $O(N)$ で、最悪の場合の計算量が $O(N^2)$ であることが示されている。しかし、ベクトル化したからといって時間計算量のオーダーが改善されるというものではないから、やはり定量的な効率の評価は必要である。

本稿ではこの観点から FOL 法の一つの評価式を与え、それが実験結果と符合することを示す。

2 FOL 法

前章で述べたように一度にベクトル処理する処理は並列に実行可能なものでなければならない。並列に実行が可能かどうかはデータを指し示しているポインタ（またはインデックス）を一つずつ比較をしていけばわかるけれども、この方法ではその比較のために $O(N^2)$ の計算量がかかるが、これは効率的とはいえない。

そこで、FOL 法では表に各データを書き込んでいき、衝突しているかどうかで並列に実行が可能かどうかを判定する。衝突が起きると上書きされてしまうものが出るが、上書きされずに残っているものどうしは並列に実行可能であるからである。

FOL 法の形式的な記述は次のようになる。

手続き 1 FOL 法

Efficiency Evaluation of FOL Method for Vector Processing of Symbolic Data

Shintaro MEKI

Faculty of Computer Science & System Engineering,
Okayama Prefectural University

1. すべてのデータに識別のために一意なラベルをつける
2. ハッシュを用いてすべてのデータのラベルをハッシュ表に書き込む
3. ハッシュ表を引いて上書きされていないかどうかを調べ、上書きされていないものを一つの集合とする
4. 上書きされているものに対して 2 と 3 を繰り返し、上書きされているものがなくなった時点で終了する

3 で得られたそれぞれの集合が、並列に処理可能なデータの集合を形成する。また、FOL 法自身もベクトル処理可能である。

3 FOL 法の効率解析

FOL 法ではハッシュの衝突を無視してハッシュ表に書き込んでいき、上書きされたものに対して繰り返しこれを行なう。そこで、ハッシュ表に書き込んでいった時に、どのくらいの数のデータが上書きされるかを見積もることにする。

次の記号を使うことにする。

m : ハッシュ表のエントリの数

n : ハッシュ表に書き込むデータの数

r : データの数のハッシュ表のエントリの数に対する割合
($= n/m$)

データの書き込みにおいて、ある一つのデータは確率 $1-1/m$ で、ハッシュ表のあるエントリには書き込まれない。そのため、すべてのデータの書き込みの間に、次に示す確率でハッシュ表のあるエントリにはデータが一つも書き込まれない。

$$\begin{aligned}(1-1/m)^n &= (1-1/m)^{rm} \\ &= e^{-r}(m \rightarrow \infty)\end{aligned}$$

したがって、すべてのデータを書き込んだ後の空のエントリ数は me^{-r} であり、データが書き込まれたエントリ数は次のようになる。

$$m - me^{-r} = m(1 - e^{-r})$$

n 個のデータを挿入した後で、 $m(1 - e^{-r})$ 個のエントリにデータが書き込まれているので、上書きされたデータの数は次のようになる。

$$\begin{aligned}n - m(1 - e^{-r}) &= rm - m(1 - e^{-r}) \\ &= m(r - 1 + e^{-r})\end{aligned}$$

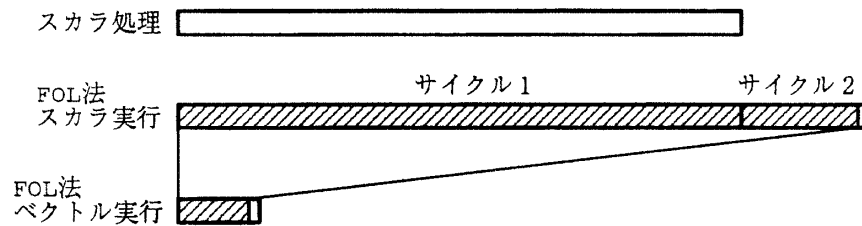


図 1: FOL 法の効率

FOL 法はこれらのデータに対して次のサイクルを実行することになる。したがって、手続きの 2 回目のサイクルを実行する時点で、データの数ハッシュ表のエントリの数に対する割合 r_1 は

$$\begin{aligned} r_1 &= m(r - 1 + e^{-r})/m \\ &= r - 1 + e^{-r} \end{aligned}$$

となる。

最初 $r = 0.5$ であったとすると、 $r_1 = 0.106531$ 、 $r_2 = 0.00547818$ となるから、手続きのサイクルを 2 回実行すれば、データのおよそ 99% が並列実行可能なデータの集合に分割されることがわかる。その後におよそ 1% のデータが残り、これに FOL 法を適用することも可能であるが、ベクトル処理するデータの数 (ベクトル長) がおよそ 1/100 になっているので、ベクトル処理しても十分な高速化を達成することはできないと考えられる。そこで、これらのデータに対してはスカラ処理を行なうことにする。この時、およそ 99% のデータがベクトル処理されることになる (ベクトル化率 99%)。

そして、2 回目のサイクルの時には最初のデータのおよそ 21% に対して処理を行うのでこれがオーバーヘッドとなるが、ベクトル計算機を用いれば 10 倍以上の高速化がはかれるので、この点は問題にはならないと考えられる。

図 1 にこのこと概念図を示す。図の斜線で示した部分がベクトル実行した時にベクトル処理される。図から多少のオーバーヘッドがあっても、全体として高速化がはかれることがわかる。

4 実験

計算機を用いて実験を行ない、前章で述べた理論が正しいのかどうかの検証を行なった。ハッシュ関数がほぼ一様分布をするであろうという仮定に基づいて、一様乱数を用いてハッシュ表にデータを書き込んでいき、上書きされてしまったものの数を調べた。ハッシュ表の大きさは 1000000 とし、10 回実験を行なって得た結果の平均を表 1 に示した。実験結果と理論から導かれた値はよく一致しており、本稿で示した評価式が正しいことを実証している。

表 1: 実験結果

r	上書きされたデータの数	
	実測値	理論値
0.1	4815	4837
0.2	18716	18730
0.3	40783	40818
0.4	70460	70320
0.5	106422	106530
0.6	148725	148811
0.7	196476	196585
0.8	249378	249328
0.9	306744	306569
1.0	367717	367879

5 むすび

本稿では記号処理をベクトル化するための FOL 法の一つの評価式を与え、実験の結果からそれが正しいことを示した。ハッシュ関数が一様分布をするという仮定は実際の応用では必ずしも成り立つとは言えないが、ハッシュ関数の計算自体は効率よくベクトル処理できるので、スカラ処理では実際的ではないような複雑なハッシュ関数を用いることもできる。したがって、一様分布とまではいなくてもそれとかなり近い分布をすると考えられるので、本稿で行なった考察は実際の応用にも適用できると考えられる。

以上のことから、FOL 法は記号処理ベクトル化のために有効であると考えられる。

謝辞

種々御議論頂いた京都大学工学研究科の上林教授に感謝致します。

参考文献

- [1] Y. Kanada, "A method of vector processing for shared symbolic data," *Parallel Computing*, Vol.19, No.10, pp.1155-1175, 1993.
- [2] 石浦, 高木, 矢島, "ベクトル計算機上でのソーティング," *情報処理学会論文誌*, Vol.29, No.4, pp.378-385, 1988.