



Title	スーパーコンピューターの効能 その2
Author(s)	佐藤, 透; 丹羽, 哲夫; 田村, 圭介
Citation	大阪大学大型計算機センターニュース. 1989, 75, p. 47-50
Version Type	VoR
URL	<a href="https://hdl.handle.net/11094/65856">https://hdl.handle.net/11094/65856</a>
rights	
Note	

*The University of Osaka Institutional Knowledge Archive : OUKA*

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

## スーパーコンピュータの効能 その2

大阪大学理学部物理

佐藤透、丹羽哲夫

大阪大学核物理研究センター

田村圭介

### 1. はじめに

前回単純なプログラムを用いてベクトル化によっていったいどれほど高速化がなされるのかを、いろいろな関数、ループの構造、ベクトル化技法について調べた。今回もこの計算機現象論を続け、さらにいくつかの例題について試してみた。

### 2. ベクトル化による効果の演算子、ループ長依存性。

倍精度実数について配列要素の四則演算、組み込み関数による演算についてベクトル化による効果を調べる。これは前回報告にも書いたが、サブルーチンCLOCKを呼ぶ時間が短いループでは無視できないため今回はこの時間を差し引いた結果を一応示しておく。使用したプログラムは

```
call clock(ts)
do 100 n=1,nmax
  演算
100 continue
call clock(te)
tim = te-ts
```

timからcall clockの時間を引く。図1に時間のループ長依存性を示す。縦軸はLinear Scale、単位秒。この図以上のループ長では、かかる時間はいずれの場合もほぼループ長に比例する。次の表1に各演算にかかる時間(単位ループ当り、代入も含む)のoption vector(V)、novector(NV)及びその比を示す。特に complex\*16のEXPが高速化されreal\*8の割り算は効率が悪い。

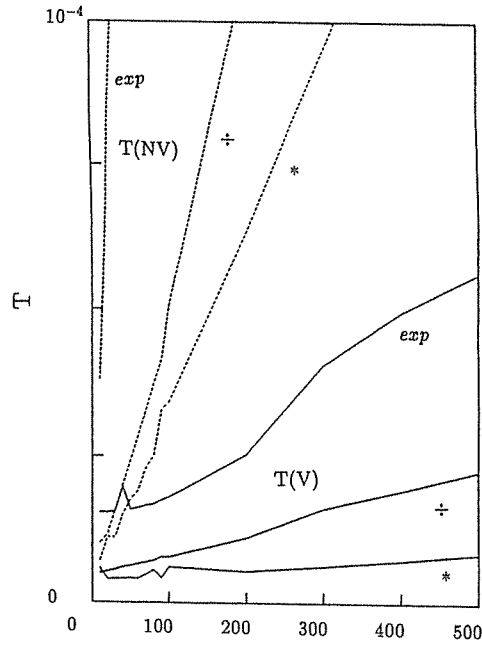


図1 N(Loop)

表 1

	T(NV)	T(V)	T(NV)/T(V)
Real#8			
=	173	6	25
*	311	9	34
+	290	9	32
/	516	39	13
SIN	3240	79	40
TAN	4740	166	28
EXP	3500	100	34
LOG	3250	124	26
Complex#16			
=	291	14	20
+	493	19	25
*	716	31	22
/	1510	66	22
EXP	16440	290	56

unit ns

### 3. 間接番地指定

間接番地指定が高速化を阻むといわれているが、ここでは間接番地指定でどれだけ遅くなるか調べてみた。(Ti(NV)/Ti(V))は各々の演算でベクトル化によりどれだけ早くなったか、また(Ti(V)/T1(V))は間接番地指定でどれだけ遅くなったかを示す。ご覧のように間接番地指定を行うと極端に遅くなる。

#### 例 1

		Ti(NV)/Ti(V)	Ti(V)/T1(V)
1	$a3(i) = a1(i) + a2(i)$	33.	1
2	$a3(i) = a1(i1(i)) + a1(i)$	9.1	4.4
3	$a3(i) = a1(i1(i)) + a2(i2(i))$	5.5	7.8

#### 例 2

1	$a3(i) = a1(i) + a3(i)$	33	1
2	$a3(i) = a1(i1(i)) + a3(i)$	9.0	4.5
3	$a3(i2(i)) = a1(i) + a3(i2(i))$	1.0	39

例 2 - 3 は基本的に 2 - 2 と同等なのでもちろん 2 - 3 の様なプログラムは避けることが出来る。

### 4. 回帰演算

D O ループにおいて前回の結果を参照する計算でしかもベクトル化により高速化されるといわれている例題を調べてみる。

	Ti(NV)/Ti(V)	Ti(V)/T1(V)
1 $s = s + a1(i)$	44	1
2 $s = s + a1(i)*a2(i)$	85	1.1
3 $xmax = \max(xmax, a1(i))$	44	2.2
4 $xmin = \min(xmin, a1(i))$	44	2.2
5 $s = s + (a1(i))$	48	0.9

5 は 1 の  $a1(i)$  によけいな括弧をつけただけである。

## 5. メモリ・アクセス

D O ループの刻みを変える事によりメモリへのアクセス順序を変えてみた。実際には

```
do 100 i=1,imax,id
として
```

### 1次元配列（次元10000）

real * 8			
	id	Ti(NV)/Ti(V)	Ti(V)/T1(V)
1	1	33	1
2	-1	30	1.1
3	2	37	1.1
4	3	40	1.2
integer			
5	1	30	1.0
6	-1	28	1.1

### 2次元配列（7, 8は201\*201 奇数次元、9, 10は200\*200 偶数次元の配列）

	id	Ti(NV)/Ti(V)	Ti(V)/T1(V)
7	1	93	1
8	-1	94	1.1
9	1	104	1.
10	-1	83	1.1

2次元の配列 A(N1,N2) の2重D O ループにおいてN1,N2どちらを中にいれてもTi(V)は変わらない。（ただしTi(NV)は50%ほど違いがある。）

## 6. おわりに

この様な単純化したプログラムでどの程度現実の状況を反映しているか疑問ではありますが、S Xを使う際の基礎データとして参考にしていただければ幸いです。

- 1) 佐藤、田村 大阪大学大型計算機センター・ニュースVol.18、45(1988)
- 2) 島崎 スーパーコンピューターとプログラミング（共立出版）