分散メモリ型並列計算機による円周率の515億桁計算

高橋大介† 金田康正†

本論文では，分散メモリ型並列計算機により高精度の円周率を高速に計算する方法について述べる．高精度の円周率は，Gauss-Legendreの公式およびBorweinの4次の収束の公式を用いると効率良く計算できることが知られている．これら2つの公式には平方根や4乗根，そして逆数計算が含まれているが，それらの計算はNewton法を適用することで，多倍長数の加減乗算に帰着させることができる．n 桁どうしの多倍長乗算は高速Fourier変換（FFT）を用いれば $O(n \log n \log\log n)$ で行えるが，多倍長乗算の主要部分であるFFTの計算および多倍長数の加減乗算における正規化の部分を並列化した．その結果，1024プロセッサから成る分散メモリ型並列計算機HITACHI SR2201で515億桁余りの円周率の計算が検証時間を含めて66時間11分で終了した．

Calculation of π to 51.5 Billion Decimal Digits on Distributed Memory Parallel Processors

DAISUKE TAKAHASHI† and YASUMASA KANADA†

This paper discusses the fast multiple-precision calculation of π on distributed memory parallel processors. It is well known that the multiple-precision π can be efficiently computed by the Gauss-Legendre algorithm and the Borweins' quartically convergent algorithm. Although two algorithms include the square root, 4th root and reciprocal calculation, these calculations can be reduced to the multiple-precision addition, subtraction and multiplication by using Newton method. Multiple-precision multiplication of n digits numbers can be realized with the computational complexity of $O(n \log n \log\log n)$ by using fast Fourier transform (FFT). Calculation of FFT which is crucial to the multiple-precision multiplication and normalization of the multiple-precision addition, subtraction and multiplication can be parallelized. More than 51.5 billion decimal digits of π were calculated on the distributed memory parallel processor HITACHI SR2201 (1024 PEs) within computing elapsed time of 66 hours 11 minutes which includes the time for the verification.

1. はじめに

円周率の高精度計算は昔から多くの人々によって行われており，計算機が出現してから，**表1** に示すように円周率の計算桁数は飛躍的に伸びている．

1970年代までの計算機による計算すべてとそれ以降の一部の計算では，次の公式（あるいは類似の公式）

$$\frac{\pi}{4} = 4\arctan\frac{1}{5} - \arctan\frac{1}{239} \qquad (1)$$

とarctanのTaylor展開を組み合わせた方法が使われている．しかし，この方法は桁数を n とすると計算量は $O(n^2)$ となり，計算時間は桁数の自乗に比例して増加するという問題がある．

1976年にBrent[1]とSalamin[2]の2人によってまったく独立に再発見された，楕円積分に関するGaussの計算法と，Legendreの関係式を用いた円周率を求める公式（以後，Gauss-Legendreの公式と呼ぶ）を用いることで，n 桁どうしの多倍長乗算の計算量を $M(n)$ とした場合，$O(M(n)\log n)$ で円周率が計算できることが示された．n 桁どうしの多倍長乗算は高速Fourier変換（FFT）[3]を用いることで $M(n) = n\log n\log\log n$ にできることが知られており[4]，この方法は桁数 n が大きな領域でarctanのTaylor展開を用いる方法よりも有利である．

この公式に基づく円周率の高精度計算として，1976～1979年頃にStanford大学のD.E. Knuthの学生らにより，SIMD型並列計算機ILLIAC IVを用いて概算4時間で3300万ビット（10進で約1000万桁）の円周率を計算する計画があった[2]．この計算では，Gauss-Legendreの公式とSchönhage-Strassenの乗算アルゴリズム[4),5]を用いることになっていたが，結局計算は成功していない．D.V. ChudnovskyとG.V. Chudnovskyは彼らが発見した

† 東京大学大型計算機センター
Computer Centre, University of Tokyo

表1 計算機による円周率計算の記録
Table 1 Historical records of the π calculation by computers.

記録	記録保持者	使用計算機	実行年	(計算桁数) 宣言した計算桁数	正確な計算桁数	計算時間 (検証時間)	公式 (検証)
1	Reitwiesner ほか	ENIAC	1949	(2040)	2037	~70時間 (~70時間)	M (M)
2	Nicholson, Jeenel	NORC	1954	(3093)	3092	13分 (13分)	M (M)
3	Felton	Pegasus	1957	(10021)	7480	33時間 (33時間)	K (G)
4	Genuys	IBM 704	1958	(10000)	10000	1時間40分 (1時間40分)	M (M)
5	Felton	Pegasus	1958	(10021)	10020	33時間 (33時間)	K (G)
6	Guilloud	IBM 704	1959	(16167)	16167	4時間18分 (4時間18分)	M (M)
7	Shanks, Wrench	IBM 7090	1961	(100265)	100265	8時間43分 (4時間22分)	S (G)
8	Guilloud, Filliatre	IBM 7030	1966	(250000)	250000	41時間55分 (24時間35分)	G (S)
9	Guilloud, Dichampt	CDC 6600	1967	(500000)	500000	28時間10分 (16時間35分)	G (S)
10	Guilloud, Bouyer	CDC 7600	1973	(1001250)	1001250	23時間18分 (13時間40分)	G (S)
11	三好, 金田	FACOM M-200	1981	(2000040) 2000000	2000036	137時間18分 (143時間18分)	K (M)
12	Guilloud	不明	1981-82	(不明) 2000050	2000050	不明 (不明)	不明 (不明)
13	田村	MELCOM 900II	1982	(2097152)	2097144	7時間14分 (2時間21分)	L (L)
14	田村, 金田	HITAC M-280H	1982	(4194304)	4194288	2時間21分 (6時間52分)	L (L)
15	田村, 金田	HITAC M-280H	1982	(8388608)	8388576	6時間52分 (30時間以上)	L (L)
16	金田, 吉野, 田村	HITAC M-280H	1982	(16777216)	16777206	30時間以上 (6時間36分)	L (L)
17	後, 金田	HITAC S-810/20	1983.10	(10013400) 10013395	10000000	24時間以下 (30時間以上)	G (L)
18	Gosper	Symbolics 3670	1985.10	(17526200以上)	17526200	不明 (28時間)	R (B4)
19	Bailey	CRAY-2	1986.1	(29360128) 29360000	29360111	28時間 (40時間)	B4 (B2)
20	金田, 田村	HITAC S-810/20	1986.9	(33554432) 33554400	33554414	6時間36分 (23時間)	L (L)
21	金田, 田村	HITAC S-810/20	1986.10	(67108864)	67108839	23時間 (35時間15分)	L (L)
22	金田, 田村, 久保, 小林および花村	NEC SX-2	1987.1	(134217728) 133554400	134217700	35時間15分 (48時間2分)	L (B4)
23	金田, 田村	HITAC S-820/80	1988.1	(201326572) 201326000	201326551	5時間57分 (7時間30分)	L (B4)
24	Chudnovskys	CRAY-2 IBM-3090/VF	1989.5	(480000000以上) 480000000	480000000?	~6ヶ月? (不明)	C (C)
25	Chudnovskys	IBM-3090	1989.6	(525229270以上) 525229270	525229270	1ヶ月以上? (不明)	C (不明)
26	金田, 田村	HITAC S-820/80	1989.7	(536870912) 536870000	536870898	67時間13分 (80時間39分)	L (B4)
27	Chudnovskys	IBM-3090	1989.8	(1011196691以上) 1011196691	1011196691?	2ヶ月以上? (不明)	C (C)
28	金田, 田村	HITAC S-820/80	1989.11	(1073741824) 1073740000	1073741799	74時間30分 (85時間57分)	L (B4)
29	Chudnovskys	m zero (手作り計算機)	1991.8	(2260000000以上) 2260000000?	2260000000?	250時間? (不明)	C (C)
30	Chudnovskys	不明	1994.5	(4044000000以上) 4044000000?	4044000000?	不明 (不明)	C (C)
31	高橋, 金田	HITAC S-3800/480	1995.6	(3221225472) 3221220000	3221225466	36時間52分 (53時間43分)	B4 (L)
32	高橋, 金田	HITAC S-3800/480	1995.8	(4294967296) 4294960000	4294967286	113時間41分 (130時間20分)	B4 (L)
33	高橋, 金田	HITAC S-3800/480	1995.10	(6442450944) 6442450000	6442450938	116時間38分 (131時間40分)	B4 (L)
34	Chudnovskys	不明	1996.3	(8000000000以上) 8000000000?	8000000000?	1週間? (不明)	C (C)
35	高橋, 金田	HITACHI SR2201	1997.4	(17179869184)	17179869142	5時間11分 (5時間26分)	L (B4)
36	高橋, 金田	HITACHI SR2201	1997.5	(34359738368)	34359738327	15時間19分 (20時間34分)	B4 (L)
37	高橋, 金田	HITACHI SR2201	1997.7	(51539607552) 51539600000	51539607510	29時間3分 (37時間8分)	B4 (L)

公式欄のM, K, G, S, L, R, B4, B2, Cはそれぞれ Machin, Klingenstierna, Gauss, Störmer, Gauss-Legendre, Ramanujan, Borwein の4次の収束, Borwein の2次の収束, Chudnovsky の各公式である.

$$\frac{1}{\pi} = \frac{6541681608}{640320^{3/2}} \sum_{k=0}^{\infty} \left(\frac{13591409}{545140134} + k \right) \cdot \frac{(6k)!}{(3k)!(k!)^3} \cdot \frac{(-1)^k}{(640320)^{3k}} \qquad (2)$$

の公式を使用し，自作の計算機を並列に用いて，約1週間程度の計算時間で円周率を80億桁以上計算した[6]．この方法では，式(2)において級数の各項を複数のプロセッサに割り当てて独立に計算ができるので，並列実行は比較的容易である．

この公式は，arctan の Taylor 展開に基づく計算方法に比べると収束が速いという利点がある．しかし，桁数を n とすると計算量は $O(n^2)$ であるので，arctan の Taylor 展開に基づく計算方法と同様に，計算時間は桁数の自乗に比例して増加するという問題がある．

今回分散メモリ型並列計算機を用いて 515 億桁の円周率を計算するにあたって，主計算に Borwein の 4 次の収束の公式[7]，検証計算に Gauss-Legendre の公式を用いた．これら 2 つの公式には多倍長数の平方根や 4 乗根，そして逆数計算が含まれているが，それらの計算は Newton 法を適用することで，多倍長数の加減乗算に帰着させることができる．FFT を用いた多倍長乗算の主要部分である FFT 計算および多倍長数の加減乗算における正規化の部分を並列化した．

以下，2 章で Gauss-Legendre の公式を，3 章で Borwein の 4 次の収束の公式を示す．4 章で多倍長数の加減乗算アルゴリズムについて，5 章で多倍長数の平方根および 4 乗根，逆数計算アルゴリズムについて，6 章で多倍長数の加減乗算の並列アルゴリズムについて述べる．7 章で並列化の評価について，8 章で円周率 515 億 3960 万桁計算の結果について述べる．

2. Gauss-Legendre の公式

Gauss-Legendre の公式[1),2)]による円周率の計算方法を以下に示す．

$A=1$，$B=1/\sqrt{2}$，$T=1/4$，$X=1$ として，A と B の差が必要とする精度より大きな間，次の { と } で囲まれる部分を繰り返し計算する．

$$\{Y := A;\ A := \frac{A+B}{2};\ B := \sqrt{B \times Y};$$
$$T := T - X \times (Y-A)^2;\ X := 2 \times X\}$$

すると，π の値は $(A+B)^2/(4T)$ となる．ただし，A，B，T，Y の値は，求める精度以上の精度で計算する必要がある．この公式は 2 次の収束を示すので，n 桁の π は $\log_2 n$ 回程度の反復で求まる．

3. Borwein の 4 次の収束の公式

1980 年代以降，Borwein と Borwein[7)]が発見した円周率を計算するいくつかの公式の中で，これまでに円周率の高精度計算で使用されている[8)~10)]4 次の収束の公式を以下に示す．

$a_0 = 6-4\sqrt{2}$ および $y_0 = \sqrt{2}-1$ として

$$y_{k+1} = \frac{1-(1-y_k^4)^{1/4}}{1+(1-y_k^4)^{1/4}} \quad (3)$$

$$a_{k+1} = a_k(1+y_{k+1})^4 - 2^{2k+3}y_{k+1}(1+y_{k+1}+y_{k+1}^2) \quad (4)$$

の級数を求める精度まで繰り返し計算すると，π の値は $1/a_k$．ただし，a_k，y_k の値は求める精度以上の精度で計算する必要がある．

Borwein の 4 次の収束の公式では，反復回数は 2 次の収束の Gauss-Legendre の公式の約半分であるが，反復 1 回あたりの計算量が Gauss-Legendre の公式の約 2 倍になっていることから，結局全体の計算量としてはどちらもほぼ同じとなる．また，Gauss-Legendre の公式および Borwein の 4 次の収束の公式においては，式を変形することにより，計算量がさらに減ることが知られている[11)]．

4. 多倍長数の加減乗算アルゴリズム

Gauss-Legendre の公式，Borwein の 4 次の収束の公式では，前述の A，B，T，Y および a_k，y_k は，正確に計算しようとする桁数 $+\alpha$ の精度で繰り返し計算を行わなくてはならない．ただし，α は計算桁数によるが，40~50 桁のオーダーである．したがって，多倍長計算が計算時間の大半を占めることになる．

4.1 多倍長数の加減算および多倍長数と単精度数との乗算アルゴリズム

多倍長数どうしの加減算や，多倍長数と単精度数との乗算は，桁数を n とした場合明らかに $O(n)$ の計算量で行えることが分かる．

4.2 多倍長数の乗算アルゴリズム

種々の多倍長乗算のアルゴリズムの中で[4)]，数千桁以上の乗算で高速性が発揮できる FFT を用いた多倍長乗算アルゴリズム[7),9),12)]を採用した．

以下，FFT を用いた多倍長乗算アルゴリズムを説明する．まず，基数 b の n word の多倍長数 $X \equiv (x_0, x_1, x_2, \cdots, x_{n-1})$ と $Y \equiv (y_0, y_1, y_2, \cdots, y_{n-1})$ の乗算を行うことを考える．

X と Y の積 $Z \equiv (z_0, z_1, z_2, \cdots, z_{2n-1})$ は，以下のように定義される．

$$\begin{aligned}
z_0 &= x_0y_0\\
z_1 &= x_0y_1 + x_1y_0\\
z_2 &= x_0y_2 + x_1y_1 + x_2y_0\\
&\vdots\\
z_{n-1} &= x_0y_{n-1} + x_1y_{n-2} + \cdots + x_{n-1}y_0\\
&\vdots\\
z_{2n-3} &= x_{n-1}y_{n-2} + x_{n-2}y_{n-1}\\
z_{2n-2} &= x_{n-1}y_{n-1}\\
z_{2n-1} &= 0
\end{aligned}$$

ここで，X と Y の後に n 個の 0 を付け加えると，X，Y そして Z の長さ N は $N=2n$ となる．すると，

z_k は次のように表すことができる．

$$z_k = C_k(x, y) = \sum_{j=0}^{N-1} x_j y_{k-j} \tag{5}$$

式(5)において，y の添字 $k-j$ が負のときは $k-j+N$ に置き換えるものとする．

ここで，離散Fourier変換（DFT）$F(x)$ および離散逆Fourier変換（逆DFT）$F^{-1}(x)$ は

$$F_k(x) = \sum_{j=0}^{N-1} x_j e^{-2\pi ijk/N} \tag{6}$$

$$F_k^{-1}(x) = \frac{1}{N} \sum_{j=0}^{N-1} x_j e^{2\pi ijk/N} \tag{7}$$

で表されるが，ここで畳み込み定理を用いれば，

$$F(C(x, y)) = F(x)F(y) \tag{8}$$

と表すことができ，これより，

$$C(x, y) = F^{-1}(F(x)F(y)) \tag{9}$$

となる．

すなわち，多倍長数 X と Y の積 Z を求めるには，数列 $(x_0, x_1, x_2, \cdots, x_{n-1})$ と $(y_0, y_1, y_2, \cdots, y_{n-1})$ の後に n 個の0を付け加えた数列のDFTを求め，得られた数列の要素ごとの積を計算し，その逆DFTを行い，得られた結果の実数部を整数値に丸めて，基数 b で正規化すればよい．ここで，DFTを求める際にFFTを用いれば，N 点DFTは $O(N \log N)$ の計算量で求めることができる．

FFTの実現方法にもよるが，有限の精度の浮動小数点演算によるFFTで多倍長乗算を行う際，IEEE表現の倍精度浮動小数点計算においては，1万進数では最悪の場合10進換算で $2^{25} \approx 3355$ 万桁どうしの乗算でも情報落ちの影響が出て，正確に多倍長乗算が行えない場合がある．

これは，1万進数どうしの乗算を求める際のFFTの計算で，4桁×4桁＝8桁の数値の数千万項の和を求めるために，総情報量としては $10^8 \times 10^7 = 10^{15}$ 以上となり，そこで仮数部がoverflowしてしまい，結果として情報落ちが生じるのが主な原因である．

しかし，Mersenne数の素数判定でも使用されている[13]次のような工夫を行うことにより，FFTにおける情報落ちの影響を防ぐことが可能である．

たとえば，

```
|   3.|1415|9265|3589|7932|
```

のように格納されているデータを

```
|   3.|1416|-735|3590|-2068|
```

のように前処理を行ってからFFTを計算する．つまり，配列IA(1:N)に基数BASEでN wordの多倍長数が格納されているものとすると，次のようなFORTRANプログラムで書かれているような前処理を行う．

```
1  DO I=N,2,-1
2    IF (IA(I) .GE. BASE/2) THEN
3      IA(I)=IA(I)-BASE
4      IA(I-1)=IA(I-1)+1
5    END IF
6  END DO
```

そうすると，式(5)における畳み込み $\sum_{j=0}^{N-1} x_j y_{k-j}$ を計算する際に，基数を b としたとき $-b/2 \le (x_j, y_{k-j}) \le b/2-1$ とできる．したがって，前処理を行わない場合，つまり $0 \le (x_j, y_{k-j}) \le b-1$ の場合に比べて $x_j y_{k-j}$ の絶対値が小さくなるとともに，$\sum_{j=0}^{N-1} x_j y_{k-j}$ を求める際に各項がプラスマイナスで打ち消しあう効果が出る．このように，前処理を行うことによって仮数部のoverflowが発生する確率を低くすることができるので，今回はこの方法を採用した．

ここで，この方法を用いた際に仮数部のoverflowが発生しない十分条件を検討する．浮動小数点演算によるFFTを用いた N 点畳み込みの相対誤差は，machine epsilonを ϵ とすると，正確な三角関数値を使ってFFTを計算した場合には $\epsilon\sqrt{\log_2 N}$ となる[14]．IEEE表現の倍精度実数では $\epsilon = 2^{-52}$ であることから，N 点畳み込みにおける有効bit数は，$52 - \lceil \log_2 \sqrt{\log_2 N} \rceil = 52 - \lceil (\log_2 \log_2 N)/2 \rceil$ bitとなる．

したがって，IEEE表現の倍精度実数では，式(9)において $C(x, y)$ の各要素の絶対値が，N 点畳み込みにおける有効bit数で表すことのできる最大の数，つまり $2^{52-\lceil (\log_2 \log_2 N)/2 \rceil} - 1$ 以下であれば，仮数部のoverflowは起こらない．もしこの条件を満たさない場合には，1 wordあたりの基数 b を小さくすることで仮数部のoverflowを回避することができる．

FFTを用いて1億桁以上のオーダーの多倍長乗算を行う際は，このような工夫を行ったとしても，倍精度実数配列1個に格納できる多倍長数は10進で4桁～5桁程度となってしまうために，通常の $O(n^2)$ のアルゴリズムやKaratsubaの $O(n^{\log_2 3})$ のアルゴリズム[4),15)]に比べて，多倍長乗算には記憶領域を多く必要とする．このような場合は，二次記憶を用いたFFTを使う方法が考えられる．しかし最近のプロセッサは高速化されているにもかかわらず，二次記憶装置と主記憶装置間のデータ転送はそれほど高速化されていないため，大半の経過時間がI/Oに費やされてしまう．分散メモリ型並列計算機において，二次記憶を用いた

FFT を実現し性能を評価した結果からも分かるように[16]，最近の分散メモリ型並列計算機においては，二次記憶を用いた FFT を行うのは現実的でない．

FFT を主記憶のみで計算する場合には，二次記憶を用いて計算する場合に比べて乗算が可能な桁数が小さくなる．しかし，このような場合には Karatsuba のアルゴリズムと FFT による乗算アルゴリズムを組み合わせて計算することができる．

たとえば FFT による乗算が可能な桁数が 1 億桁であるとすれば，1 億桁までは FFT を用いて乗算を行い，それより大きな桁数の乗算は 1 億進数どうしの乗算を Karatsuba のアルゴリズムを用いて行う．今回の計算では，FFT と Karatsuba のアルゴリズムを組み合わせた方法を採用した．

次に，FFT と Karatsuba のアルゴリズムを組み合わせて乗算を行う際における，切替え条件，メモリ使用量と計算時間の間の関係を検討する．

まず多倍長数のデータ構造としては，10 進 n 桁の多倍長数を 10 進 8 桁ごとに区切り，32 bit 整数の配列に格納するものとする．また多倍長乗算において FFT を計算する際には，10 進 8 桁ごとに 32 bit 整数の配列 IA，IB に格納されている 32 bit 整数の配列のデータを，10 進 4 桁ごとに 64 bit 浮動小数点数の配列 X，Y にコピーするものとする．つまり，10 進 n 桁どうしの乗算を行う際は，$(N = n/2)$ 点 FFT を計算することになる．また，FFT のワーク配列として 64 bit 浮動小数点数の配列 Z を使用する．

すると，10 進 n 桁の多倍長数 IA と IB の積の上位の n 桁 IC を求めるとき，Karatsuba のアルゴリズムを適用しない場合のメモリ使用量は次のようになる．

- 入力 (IA, IB)：n バイト
- FFT (X, Y)：$8n$ バイト
- 出力 (IC)：$(n/2)$ バイト
- FFT ワーク (Z)：$4n$ バイト
- 合計：$(27/2)n$ バイト

ここで，Karatsuba のアルゴリズムを適用しない場合の n 桁どうしの多倍長乗算の計算時間 T_{simple} を検討する．桁数 n がある程度大きな場合には，FFT の計算時間が多倍長乗算の計算時間のほとんどを占めるので，以下では FFT の計算時間のみを検討する．

10 進 n 桁の多倍長数 IA と IB の積 IC を求めるためには，順 FFT を 2 回と逆 FFT を 1 回行う必要があるので，N 点 FFT を 1 回行うのに必要な計算時間を $c_{fft} \cdot N \log_2 N$ とすれば $N = n/2$ より，

$$\begin{aligned} T_{simple} &= 3c_{fft} \cdot N \log_2 N \\ &= 3c_{fft} \cdot \frac{n}{2}(\log_2 n - 1) \end{aligned}$$

となる．

次に，10 進 n 桁の多倍長数を $(d \geq 2)$ 分割して，Karatsuba のアルゴリズムを $\log_2 d$ 回適用した場合のメモリ使用量を以下に示す．ここで，Karatsuba のアルゴリズムにおけるワーク配列として 32 bit 整数の配列 IZ を使用する．

- 入力 (IA, IB)：n バイト
- FFT (X, Y)：$(8n/d)$ バイト
- 出力 (IC)：$(n/2)$ バイト
- FFT ワーク (Z)：$(4n/d)$ バイト
- Karatsuba ワーク (IZ)：$(3/4)n$ バイト
- 合計：$(9/4)n + (12/d)n$ バイト

ここで，d 分割して Karatsuba のアルゴリズムを適用した場合の計算時間 $T_{Karatsuba}$ を検討する．

Karatsuba のアルゴリズムを適用した場合は，$N/d = n/(2d)$ 点 FFT を用いて n/d 桁どうしの乗算を $3^{\log_2 d}$ 回行うことから，

$$\begin{aligned} &T_{Karatsuba} \\ &= 3^{\log_2 d} \times \left(3c_{fft} \cdot \frac{N}{d} \log_2 \frac{N}{d}\right) \\ &= 3^{\log_2 d} \times \left\{3c_{fft} \cdot \frac{n}{2d}(\log_2 n - \log_2 d - 1)\right\} \end{aligned}$$

となる．

したがって Karatsuba のアルゴリズムを適用する回数が大きくなると，n 桁どうしの多倍長乗算における実質的な計算量は $O(n^{\log_2 3})$ に近づいていくことになる．

これらの結果より，$3 \times 2^{34} \approx 515$ 億桁どうしの乗算における FFT と Karatsuba のアルゴリズムを組み合わせた場合のメモリ量，および Karatsuba のアルゴリズムを適用しない場合の計算時間を 1 とした場合の，$3 \times 2^{34} \approx 515$ 億桁どうしの多倍長乗算の計算時間を**表 2** に示す．

今回円周率 515 億桁計算に用いた分散メモリ型並列計算機 HITACHI SR2201 の総主記憶容量は 256 GB であるが，OS のカーネルでも I/O バッファ等でメモリを使用するので，実際の実行環境では 224 GB までしか使えない．また，Gauss-Legendre の公式や Borwein の 4 次の収束の公式を用いた円周率の計算では，多倍長乗算以外にもワーク領域を必要とするため，多倍長乗算に使えるメモリ領域は Gauss-Legendre の公式では 172 GB，Borwein の 4 次の収束の公式では 184 GB 程度となる．したがって，今回の円周率 515 億桁計算においては，Gauss-Legendre の公式では $3 \times 2^{30} \approx 32$

表2 $3\times2^{34}\approx515$ 億桁の乗算（IC＝IA×IB）における FFT と Karatsuba のアルゴリズムを組み合わせた場合のメモリ量および計算時間の比

Table 2 Memory size and calculation time ratio for $3\times2^{34}\approx51.5$ billion decimal digits multiple-precision multiplication (IC＝IA×IB) with combination of FFT and Karatsuba algorithm.

切り替える乗算桁数		3×2^{30}	3×2^{31}	3×2^{32}	3×2^{33}	3×2^{34}
メモリ量（GB）	入力 (IA, IB)	48	48	48	48	48
	FFT (X, Y)	24	48	96	192	384
	出力 (IC)	24	24	24	24	24
	FFT ワーク (Z)	12	24	48	96	192
	Karatsuba ワーク (IZ)	36	36	36	36	0
	合計	144	180	252	372	648
計算時間の比		4.477	3.082	2.120	1.457	1.000

億桁まで FFT を使った乗算，Borwein の4次の収束の公式では $3\times2^{31}\approx64$ 億桁まで FFT を使った乗算，をそれぞれ使用し，それより大きな桁数の乗算に Karatsuba のアルゴリズムを用いた．

5. 多倍長数の平方根および4乗根，逆数計算アルゴリズム

5.1 多倍長数の平方根

Newton 法により $\sqrt{a}$ を求めるには，まず $1/\sqrt{a}$ を $f(x)=1/x^2-a=0$ の解として求め，その結果に a を掛けることにより $\sqrt{a}$ を求める．具体的には，

$$x_{k+1}=x_k+\frac{x_k}{2}(1-ax_k^2) \tag{10}$$

となるが，式(10)で x_k を掛けている部分については半分の精度の乗算で済む[17]．

また，最後の反復においては $\sqrt{a}$ を

$$\sqrt{a}\approx(ax_k)+\frac{x_k}{2}(a-(ax_k)^2) \tag{11}$$

とし，x_k を掛けている部分を半分の精度の乗算で計算すればさらに計算量が減る[18]．

Newton 法による $\sqrt{a}$ の計算は2次の収束，つまり正しく求まる桁数が毎回前回の2倍になるので，始めは初期値で与えた2倍の精度の桁数で計算を始め，毎回桁数を2倍にしていけばよく，始めから全桁数の多倍長計算を行う必要はない[1),17)]．

5.2 多倍長数の4乗根

a の4乗根 $a^{1/4}$ も，平方根と同様に Newton 法を用いて計算する．まず，$a^{-1/4}$ を $f(x)=1/x^4-a=0$ の解として，

$$x_{k+1}=x_k+\frac{x_k}{4}(1-ax_k^4) \tag{12}$$

で求めた値を3乗し，それに a を掛けて $a^{1/4}$ を求める．式(12)においても式(10)と同様，x_k を掛けている部分の計算は半分の精度の乗算で済む．

5.3 多倍長数の逆数

a の逆数 a^{-1} も，平方根や4乗根と同様にして，$1/a$ を $f(x)=1/x-a=0$ の解として，

$$x_{k+1}=x_k+x_k(1-ax_k) \tag{13}$$

で求める．式(13)においても式(10)，(12)と同様，x_k を掛けている部分の計算は半分の精度の乗算で済む．

6. 多倍長数の加減乗算の並列アルゴリズム

6.1 多倍長数の加減算および多倍長数と単精度数との乗算の並列化

多倍長数どうしの加減算や，多倍長数と単精度数との乗算は，多倍長数の桁数を n とした場合明らかに $O(n)$ の計算量で行えることが分かる．

しかし，多倍長数どうしの加減算や，多倍長数と単精度数との乗算において，並列化を阻害する要因はキャリーおよびボローの処理である．

ところが次に示すようなベクトル処理にも適応可能な工夫を行うことで，キャリーおよびボローの並列化が可能になる．Fortran 90 で記述された多倍長数の並列加算プログラムの kernel 部分は次のようになる．

```
1  IA(1:N)=IA(1:N)+IB(1:N)
2  DO WHILE (ANY(IA(2:N) .GE. BASE))
3    IC(N)=0
4    IC(1:N-1)=INT(IA(2:N)/BASE)
5    IA(2:N)=IA(2:N)+IC(2:N)-IC(1:N-1)*BASE
6    IA(1)=IA(1)+IC(1)
7  END DO
```

基数を BASE としたとき，入力となるデータは各要素が 0～BASE-1 に正規化されて配列 IA と IB に格納されているものとする．まず，1行目でキャリーを考慮せずに加算を行ってしまう．次に，2行目の ANY 文で配列 IA(2:N) の各要素に BASE 以上の値があるかどうかをチェックする．もしあれば，4行目でキャリーを配列 IC に求めた後に，5行目および6行目でキャ

リーの補正を行う．

このキャリーの補正においては，キャリーの伝搬を考慮していない．したがってキャリーが完全に補正されていない場合があるので，各要素がBASE未満になるまでDO WHILEループが繰り返されることになる．

しかし，多倍長数の計算において各要素の値はランダムかつ正規化されているものと仮定すれば，たとえば基数BASEを10^8とした場合にキャリーが2回続けて出現する確率は，$0.5\times(1/10^8)^2=5\times10^{-17}$となり，事実上問題はないことが分かる

$0.99999999\cdots9+0.00000000\cdots1$のようにキャリーが長い区間伝搬する場合や，キャリーの伝搬が頻繁に起こる場合は，桁上げ飛び越し（carry skip）方式[19]を用いるか，金田がベクトル計算機で行ったように[9]，桁上げ先見（carry look-ahead）方式を一次回帰演算で実現し，recursive doubling[20]によって並列化することが考えられる．今回の計算では，実現が容易である桁上げ飛び越し方式を並列化して，キャリーの伝搬を処理している．

ここでキャリーを完全に補正せず冗長表現を使用して，各要素が基数BASE未満ではなくBASEの値を使ってよいことにすれば，DO WHILEループは最大で2回しか回らない．しかし冗長表現を用いた場合，Newton法による平方根や4乗根，逆数の計算において，収束判定を行う際に前の値との比較を行うために，正規化が必要となる．したがって今回の計算においては，多少時間がかかっても毎回正規化を行って計算を行っている．

また多倍長数どうしの減算や，多倍長数と単精度数との乗算においても，加算と同様のキャリー（あるいはボロー）の処理が可能である．

6.2 多倍長数の乗算の並列化

多倍長数の乗算の並列化においては，FFTおよび正規化の部分の並列化が問題となるが，並列FFTアルゴリズムが多く提案されているので[21)~23)]，高性能な並列FFTプログラムを利用することができる．

問題は正規化の処理であるが，多倍長数の加減算および多倍長数と単精度数との乗算の並列化におけるキャリーの処理と本質的には同一であり，この部分も同様にして並列化できる．ただしキャリーの値は1とは限らないので，正規化の際にキャリーが出現する確率は加減算の場合より高くなる．

7. 並列化の評価

6章で示した各並列化の評価を行うため，多倍長数の円周率の計算をプロセッサ数Pと桁数nを変化させ，各条件下における実行時間を測定した．

計算機としては，分散メモリ型並列計算機HITACHI SR2201（1024PE，総主記憶256GB，理論ピーク性能307.2GFLOPS）のうち，4~256PEを用いた．計算経過時間の測定に際して，使用PEすべてをシングルユーザで使用した．

並列実数FFTルーチンとして，文献23)による基数2，3，5の複素FFTを，実数FFTの性質を利用し計算量および記憶領域を約半分にしたものを用いた．通信ライブラリとしてMPI[24)]を用いた．プログラムはすべてFORTRANで記述し，コンパイラは日立の最適化FORTRAN77 V02-05Bを用い，最適化オプションとして`-W0,'opt(o(ss),approx(0))'`を指定した．

多倍長数のデータ構造としては，多倍長数を10進8桁ごとに区切り，サイクリックに32bit整数の配列に格納している．また多倍長乗算においてFFTを計算する際には，10進8桁ごとに格納されている32bit整数の配列のデータを，10進4桁ごとに64bit浮動小数点数の配列にコピーしている．

利用可能な主記憶容量の関係で，計算桁数nを$n=2^{16}$~2^{30}桁まで，プロセッサ数Pを$P=4$~256と，それぞれ2のべきで変化させた．

図1，**図2**から分かるように，Gauss-Legendreの公式，Borweinの4次の収束の公式のいずれにおいても，$n=2^{20}$桁の円周率の計算においては，128台まではプロセッサ数の増加にともない実行時間が短縮されているが，256台になると逆に実行時間が増えている．

この理由としては，多倍長乗算を計算する際のFFTの計算において全対全通信を行っている[22),23)]ために，プロセッサ数が増加するに従って一度に送るデータ量が少なくなり，通信の立ち上がり時間が無視できなくなってくるためと考えられる．

しかし今回作成したプログラムでは，全対全通信においてデータが小さいときにはプロセッサ間通信のレイテンシの影響の小さいshuffle-exchangeアルゴリズムを使うようにし，データが大きいときには総データ転送量の少ないpairwiseアルゴリズムを使うように切り替えている[22)]．

なお全対全通信の方法としては，shuffle-exchangeアルゴリズムとpairwiseアルゴリズムを切り替える方法が最適というわけではなく，両者を組み合わせた方法が高速な場合もある．たとえば16台のプロセッサの場合，pairwiseアルゴリズムの通信回数と通信量をそれぞれ15回，$15N$とすると，shuffle-exchangeア

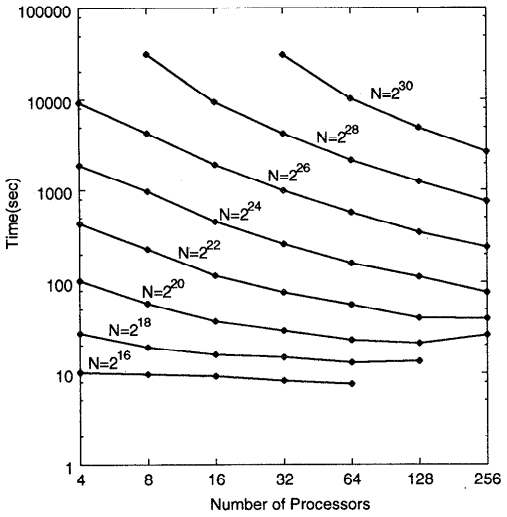

図1 HITACHI SR2201による円周率の計算時間
(Gauss-Legendreの公式, N は計算桁数, MPI版)

Fig. 1 Execution time of π calculation on HITACHI SR2201 (Gauss-Legendre algorithm, N = number of digits, MPI version).

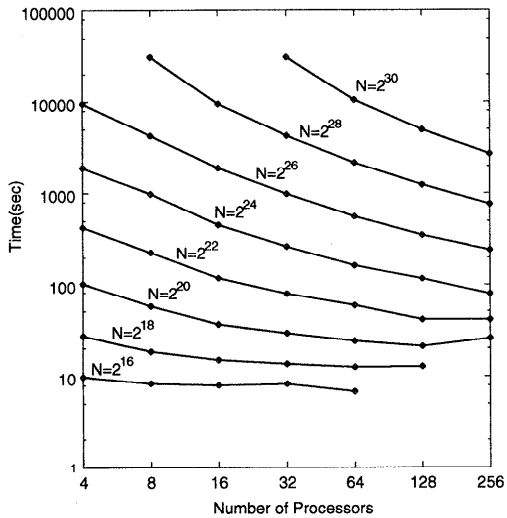

図2 HITACHI SR2201による円周率の計算時間
(Borweinの4次の収束の公式, N は計算桁数, MPI版)

Fig. 2 Execution time of π calculation on HITACHI SR2201 (Borweins' quartically convergent algorithm, N = number of digits, MPI version).

ルゴリズムではそれぞれ4回, $32N$ となるが, 4×4 の4組のpairwiseをshuffle-exchangeして2回行う場合には通信回数と通信量はそれぞれ6回, $24N$ となる.

しかし, 今回は実現の容易さから, shuffle-exchangeアルゴリズムとpairwiseアルゴリズムを切り替える方法を採用した.

また n 桁どうしの多倍長乗算にFFTを用いた場合, n が大きくなるに従って実行時間におけるFFTの比率が高くなる. なお多倍長乗算におけるFFTの時間と結果の正規化の時間の比は, $n=2^{30}$, $P=256$ において, キャリーの補正が1回だけの場合で約9:1であった. このことから, キャリーの補正に要する時間は少なくないことが分かる.

8. 円周率515億3960万桁計算の結果

7章で示した並列化の評価の結果, スケーラビリティや主記憶容量を考慮すると, 分散メモリ型並列計算機HITACHI SR2201 (1024PE, 総主記憶256GB, 理論ピーク性能307.2GFLOPS) においては, 1024PEすべてを使った場合には40時間以内で円周率500億桁余りの計算が可能であるとの見込みが立ったので, 円周率515億3960万桁計算を行った. 今回作成したプログラムのプロセッサ間通信ライブラリとして, MPIを用いた版と, MPIをリモートDMA転送[25]で高速にシミュレートする版の2種類で実行した.

今回計算に用いたHITACHI SR2201の要素プロセッサは32bitプロセッサであるため, このままではword数が32bitで表される数 (10進で約42億word) を超える多倍長数は計算できない. したがって, word数を表す変数を53bit (IEEE表現の倍精度浮動小数点の仮数部52bit+ケチ表現の1bit) としてプログラムを作成した.

今回作成したFORTRAN77によるプログラムは, 許される記憶容量に応じてFFTで使用する実メモリ量を変更することが可能になっている. したがって, FORTRAN77とMPIが動作する並列計算機ならば, MPI版のメモリパラメーターを変更するだけでこのプログラムの実行が可能である.

この円周率515億3960万桁計算を, 経過時間 (結果の出力時間を含む), 使用記憶容量の面からまとめたものを1997年8月2日から, 以下のように公開している☆.

主計算 (MPI版) について:
計算開始: 1997年6月6日22時29分
計算終了: 1997年6月8日3時32分
経過時間: 29時間3分11秒
総主記憶容量: 212GB
アルゴリズム: Borweinの4次の収束の公式

☆ www.cc.u-tokyo.ac.jpにanonymous ftpでアクセスできる.

表3 π の小数点以下500億桁までの0～9の数字の分布
Table 3 Frequency distribution for $\pi - 3$ up to 50,000,000,000 decimal digits.

数字	回数
0	5000012647
1	4999986263
2	5000020237
3	4999914405
4	5000023598
5	4999991499
6	4999928368
7	5000014860
8	5000117637
9	4999990486

表4 $1/\pi$ の小数点以下500億桁までの0～9の数字の分布
Table 4 Frequency distribution for $1/\pi$ up to 50,000,000,000 decimal digits.

数字	回数
0	4999969955
1	5000113699
2	4999987893
3	5000040906
4	4999985863
5	4999977583
6	4999990916
7	4999985552
8	4999881183
9	5000066450

主計算（リモートDMA版）について：
計算開始：1997年8月1日23時4分
計算終了：1997年8月3日0時18分
経過時間：25時間14分32秒
総主記憶容量：212 GB
アルゴリズム：Borweinの4次の収束の公式

検証計算（リモートDMA版）について：
計算開始：1997年7月4日22時11分
計算終了：1997年7月6日11時19分
経過時間：37時間8分16秒
総主記憶容量：188 GB
アルゴリズム：Gauss-Legendreの公式

なお，π および $1/\pi$ の小数点以下51,539,599,981桁から51,539,600,000桁までは，

π： 70532 46569 86142 12904
$1/\pi$： 60081 50624 62192 72973

である．また，π および $1/\pi$ の小数点500億桁までの0～9の数字の分布を**表3**と**表4**に示す．

Borweinの4次の収束の公式による18回の反復と，Gauss-Legendreの公式による35回の反復による $3 \times 2^{34} = 51,539,607,552$ 桁の円周率の計算結果は，最後の42桁（丸め誤差）を除きすべて一致していた．

9. おわりに

今回の円周率515億桁計算においては，分散メモリ型並列計算機を用いることにより，実行時間の大幅な短縮および計算桁数の増大を図ることができた．

また並列化だけでなく，

- 桁上げ飛び越し方式の採用による，キャリーの伝搬処理の高速化，
- FFTに基づく多倍長乗算における情報落ちの影響を防ぐための工夫，
- 多倍長乗算におけるFFTとKaratsubaのアルゴリズムの組合せによる，制限された記憶容量の下での計算速度の向上，
- Gauss-Legendre，Borweinの4次の収束の各公式における，式の変形による計算量の減少[11]，

の4つの工夫も有効である．ただし，それらの工夫の効果の程度は，計算桁数，使用PE数，使用可能な主記憶容量，PE間通信速度等により変化する．

今後の円周率計算の見通しとしては，

- 計算できる桁数の上限は，使える主記憶容量でほぼ決まる，
- 計算時間は，計算機の性能と使用可能な主記憶容量の大きさで決まる，

ということから考えると，今後の技術動向からは，1999年末までには2000億桁程度の計算が可能になっているであろうが，今世紀中に1兆桁に到達するのは不可能と考えている．

今後の課題として，途中の多倍長数の計算を10進数ではなく2進数で行うことで配列の一要素あたりに入れる桁数を増やし，メモリ効率を高くすることがある．この場合，最後に2進10進変換が必要になるが，これを分散メモリ型並列計算機で行うアルゴリズムは我々の知る限りまだ知られておらず，今後の研究が必要である．また多倍長乗算を分割して行うことで[26]さらに計算量を減らし，計算時間の短縮を図ることもあげられる．

参考文献

1) Brent, R.P.: Fast Multiple-Precision Evaluation of Elementary Functions, *J. ACM*, Vol.23, pp.242–251 (1976).
2) Salamin, E.: Computation of π Using Arith-

metic-Geometric Mean, *Math. Comp.*, Vol.30, pp.565–570 (1976).
3) Cooley, J.W. and Tukey, J.W.: An Algorithm for the Machine Calculation of Complex Fourier Series, *Math. Comp.*, Vol.19, pp.297–301 (1965).
4) Knuth, D.E.: *The Art of Computer Programming, Vol.2: Seminumerical Algorithms*, Addison-Wesley, Reading, MA (1997).
5) Schönhage, A. and Strassen, V.: Schnelle Multiplikation grosser Zahlen, *Computing (Arch. Elektron. Rechnen)*, Vol.7, pp.281–292 (1971).
6) Blatner, D.: *The Joy of Pi*, Walkerbooks, New York (1997).
7) Borwein, J.M. and Borwein, P.B.: *Pi and the AGM–A Study in Analytic Number Theory and Computational Complexity*, Wiley, New York (1987).
8) Bailey, D.H.: The Computation of π to 29,360,000 Decimal Digits Using Borweins' Quartically Convergent Algorithm, *Math. Comp.*, Vol.50, pp.283–296 (1988).
9) Kanada, Y.: Vectorization of Multiple-Precision Arithmetic Program and 201,326,000 Decimal Digits of π Calculation, *Supercomputing 88: Volume II, Science and Applications*, pp.117–128, IEEE Computer Society Press (1989).
10) 高橋大介，金田康正：円周率—高速計算法と統計性（3），情報処理学会第37回プログラミングシンポジウム報告集，pp.73–84 (1996).
11) 高橋大介，金田康正：多数桁の円周率を計算するための公式の改良：ガウス-ルジャンドルの公式とボールウェインの4次の収束の公式，情報処理学会論文誌，Vol.38, pp.2406–2409 (1997).
12) 田村良明，吉野さやか，後 保範，金田康正：円周率—高速計算法と数値の統計性，情報処理学会第25回プログラミングシンポジウム報告集，pp.1–15 (1984).
13) Slowinski, D.: Personal communication (1990).
14) Schatzman, J.C.: Accuracy of the discrete Fourier transform and the fast Fourier transform, *SIAM J. Sci. Comput.*, Vol.17, pp.1150–1166 (1996).
15) Karatsuba, A. and Ofman, Y.: Multiplication of multidigit numbers on automata, *Doklady Akad. Nauk SSSR*, Vol.145, pp.293–294 (1962).
16) 高橋大介，金田康正：並列計算機における二次記憶を用いた一次元FFTの実現と評価，情報処理学会研究報告，97-ARC-123, pp.7–12 (1996).
17) Bailey, D.H.: Algorithm 719: Multiprecision Translation and Execution of FORTRAN Programs, *ACM Trans. Math. Softw.*, Vol.19, pp.288–319 (1993).
18) Karp, A. and Markstein, P.: High Precision Division and Square Root, HP Labs Report, Hewlett-Packard Laboratories, 1530 Page Mill Road, Palo Alto, CA (1993).
19) Lehman, M. and Burla, N.: Skip Techniques for High-Speed Carry Propagation in Binary Arithmetic Units, *IRE Trans. Elec. Comput.*, Vol.Ec-10, pp.691–698 (1961).
20) Stone, H.: An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations, *J. ACM*, Vol.20, pp.27–38 (1973).
21) Van Loan, C.: *Computational Frameworks for the Fast Fourier Transform*, SIAM Press, Philadelphia, PA (1992).
22) Hegland, M.: Real and Complex Fast Fourier Transforms on the Fujitsu VPP 500, *Parallel Computing*, Vol.22, pp.539–553 (1996).
23) 高橋大介，金田康正：分散メモリ型並列計算機による2，3，5基底一次元FFTの実現と評価，情報処理学会論文誌，Vol.39, No.3, pp.519–528 (1998).
24) Message Passing Interface Forum: *MPI: A Message-Passing Interface Standard*, Version 1.1 (1995).
25) 日立製作所：HI-UX/MPPリモートDMA転送使用の手引，6A20-3-021 (1996).
26) 高橋大介，金田康正：多倍長平方根の高速計算法，情報処理学会研究報告，95-HPC-58, pp.51–56 (1995).

(平成9年10月3日受付)
(平成10年5月8日採録)

高橋 大介（正会員）

1970年生．1991年呉工業高等専門学校電気工学科卒業．1993年豊橋技術科学大学工学部情報工学課程卒業．1995年同大学大学院工学研究科情報工学専攻修士課程修了．1997年東京大学大学院理学系研究科情報科学専攻博士課程中退．同年同大学大型計算機センター助手．並列数値計算アルゴリズムに関する研究に従事．日本応用数理学会会員．

金田 康正（正会員）：Vol.39, No.3, pp.519–528「分散メモリ型並列計算機による2，3，5基底一次元FFTの実現と評価」（高橋 大介，金田 康正）を参照．