解説

# 円周率世界記録更新
## 2兆5769億8037万桁への道

高橋 大介
筑波大学大学院システム情報工学研究科

2009年4月に筑波大学計算科学研究センターのスーパーコンピュータ「T2K筑波システム」を用いて，円周率2兆5769億8037万桁が計算された．本稿では，円周率2兆5769億8037万桁計算の概要，および記録更新に至るまでの道程について述べる．

## 円周率の計算桁数で世界記録を更新

2009年4月29日午後6時6分に，円周率2兆5769億8037万7600桁の検証計算が完了した．2009年4月10日に完了していた主計算の結果と比較したところ，最後の76桁を除きすべて一致していることが確認された．

計算に使用したのは2008年6月から筑波大学計算科学研究センターで運用が開始されているスーパーコンピュータ「T2K筑波システム」（米国Appro International社製の高性能PCサーバを基本とする，総演算性能95TFlopsの超並列クラスタ型スーパーコンピュータ）で，合計で73時間36分（主計算：29時間5分，検証計算：44時間30分．主計算結果と検証計算結果の比較時間は含まない）を要した．

これまでの世界記録は東京大学情報基盤センターの金田康正教授らと日立製作所のグループが2002年に達成した1兆2411億桁であった．

$\pi$の小数点以下の数列について統計性の調査等が完了したことから，2009年8月17日に円周率2兆5769億8037万桁計算について，大学の広報室を通じてプレスリリースを行ったところ，新聞やテレビなどから取材を受けた．さらに，海外を含むいくつかの新聞やニュースサイトにも記事が掲載された．ニュースサイトに寄せられているコメントには称賛するものもあったが批判的なものもあった．

また，さまざまな国の人からメールを受け取った．メールの中身は，円周率の数列のパターンについて教えてもらいたいというものや，円周率の計算結果のデータをいただけないかといったものなどである．これらの反応から，円周率について興味を持っている人が少なからずいることを感じている．

## 計算の概要

### ■計算機による円周率の計算

円周率は無理数として知られており，これまでに円周率の値をより正確に求める試みが続けられてきた．計算機による円周率の計算は，1949年にENIACによって2037桁まで求められたのが最初で，1973年には100万桁，1989年には10億桁まで求められている．1990年代からは並列処理により計算時間を短縮する手法が用いられるようになり，2002年には並列型スーパーコンピュータを用いて1兆2411億桁まで求められている．計算機による円周率計算の桁数の伸びをグラフにしたものを**図-1**に，計算機による円周率計算の記録を**表-1**に示す．

1981年までの計算機による計算のすべてと1983年の1000万桁計算（表-1の記録17）では，次のMachinの公式（あるいは類似の公式）とarctanのTaylor展開を組み合わせた以下の方法により，桁数を$n$とした場合に計算量が$O(n^2)$となる方法が用いられていた．しかし，この方法では計算時間は桁数の二乗に比例して増加するという問題があった．

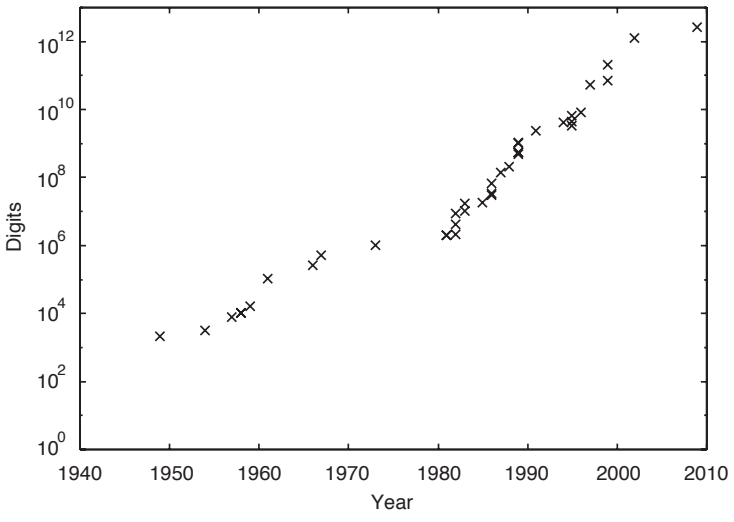


**図-1　計算機による円周率計算の桁数の伸び**

| 記録 | 記録保持者 | 使用計算機 | 実行年 | （計算桁数）<br>宣言した計算桁数 | 正確な計算桁数 | 計算時間<br>（検証時間） | 公式<br>（検証） |
|---|---|---|---|---|---|---|---|
| 1 | Reitwiesner ほか | ENIAC | 1949 | (2040) | 2037 | ～ 70 時間<br>(～ 70 時間) | M（M） |
| 2 | Nicholson, Jeenel | NORC | 1954 | (3093) | 3092 | 13 分<br>(13 分) | M（M） |
| 3 | Felton | Pegasus | 1957 | (10021) | 7480 | 33 時間<br>(33 時間) | K（G） |
| 4 | Genuys | IBM 704 | 1958 | (10000) | 10000 | 1 時間 40 分<br>(1 時間 40 分) | M（M） |
| 5 | Felton | Pegasus | 1958 | (10021) | 10020 | 33 時間<br>(33 時間) | K（G） |
| 6 | Guilloud | IBM 704 | 1959 | (16167) | 16167 | 4 時間 18 分<br>(4 時間 18 分) | M（M） |
| 7 | Shanks, Wrench | IBM 7090 | 1961 | (100265) | 100265 | 8 時間 43 分<br>(4 時間 22 分) | S3（G） |
| 8 | Guilloud, Filliatre | IBM 7030 | 1966 | (250000) | 250000 | 41 時間 55 分<br>(24 時間 35 分) | G（S3） |
| 9 | Guilloud, Dichampt | CDC 6600 | 1967 | (500000) | 500000 | 28 時間 10 分<br>(16 時間 35 分) | G（S3） |
| 10 | Guilloud, Bouyer | CDC 7600 | 1973 | (1001250) | 1001250 | 23 時間 18 分<br>(13 時間 40 分) | G（S3） |
| 11 | 三好，金田 | FACOM M-200 | 1981 | (2000040)<br>2000000 | 2000036 | 137 時間 18 分<br>(143 時間 18 分) | K（M） |
| 12 | Guilloud | 不明 | 1981-82 | (不明)<br>2000050 | 2000050 | 不明<br>(不明) | 不明<br>(不明) |
| 13 | 田村 | MELCOM 900II | 1982 | (2097152) | 2097144 | 7 時間 14 分<br>(2 時間 21 分) | L（L） |
| 14 | 田村，金田 | HITAC M-280H | 1982 | (4194304) | 4194288 | 2 時間 21 分<br>(6 時間 52 分) | L（L） |
| 15 | 田村，金田 | HITAC M-280H | 1982 | (8388608) | 8388576 | 6 時間 52 分<br>(30 時間以上) | L（L） |
| 16 | 金田，吉野，田村 | HITAC M-280H | 1983 | (16777216) | 16777206 | 30 時間以上<br>(6 時間 36 分) | L（L） |
| 17 | 後，金田 | HITAC S-810/20 | 1983.10 | (10013400)<br>10000000 | 10013395 | 24 時間以下<br>(30 時間以上) | G（L） |
| 18 | Gosper | Symbolics 3670 | 1985.10 | (17526200 以上) | 17526200 | 不明<br>(28 時間) | R（B4） |
| 19 | Bailey | CRAY-2 | 1986.1 | (29360128)<br>29360000 | 29360111 | 28 時間<br>(40 時間) | B4（B2） |
| 20 | 金田，田村 | HITAC S-810/20 | 1986.9 | (33554432)<br>33554400 | 33554414 | 6 時間 36 分<br>(23 時間) | L（L） |
| 21 | 金田，田村 | HITAC S-810/20 | 1986.10 | (67108864) | 67108839 | 23 時間<br>(35 時間 15 分) | L（L） |
| 22 | 金田，田村，久保，<br>小林および花村 | NEC SX-2 | 1987.1 | (134217728)<br>133554400 | 134217700 | 35 時間 15 分<br>(48 時間 2 分) | L（B4） |
| 23 | 金田，田村 | HITAC S-820/80 | 1988.1 | (201326572)<br>201326000 | 201326551 | 5 時間 57 分<br>(7 時間 30 分) | L（B4） |
| 24 | G. V. Chudnovsky,<br>D. V. Chudnovsky | CRAY-2，IBM 3090/VF | 1989.5 | (480000000 以上)<br>480000000 | 480000000 ？ | ～ 6 カ月？<br>(不明) | C（C） |
| 25 | G. V. Chudnovsky,<br>D. V. Chudnovsky | IBM 3090 | 1989.6 | (535339270 以上)<br>535339270 | 535339270 | 1 カ月以上？<br>(不明) | C（不明） |
| 26 | 金田，田村 | HITAC S-820/80 | 1989.7 | (536870912)<br>536870000 | 536870898 | 67 時間 13 分<br>(80 時間 39 分) | L（B4） |
| 27 | G. V. Chudnovsky,<br>D. V. Chudnovsky | IBM 3090 | 1989.8 | (1011196691 以上)<br>1011196691 | 1011196691 ？ | 2 カ月以上？<br>(不明) | C（C） |
| 28 | 金田，田村 | HITAC S-820/80 | 1989.11 | (1073741824)<br>1073740000 | 1073741799 | 74 時間 30 分<br>(85 時間 57 分) | L（B4） |
| 29 | G. V. Chudnovsky,<br>D. V. Chudnovsky | m zero<br>（自作の計算機） | 1991.8 | (2260000000 以上)<br>2260000000 ？ | 2260000000 ？ | 250 時間？<br>(不明) | C（C） |
| 30 | G. V. Chudnovsky,<br>D. V. Chudnovsky | 不明 | 1994.5 | (4044000000 以上)<br>4044000000 ？ | 4044000000 ？ | 不明<br>(不明) | C（C） |
| 31 | 高橋，金田 | HITAC S-3800/480 | 1995.6 | (3221225472)<br>3221220000 | 3221225466 | 36 時間 52 分<br>(53 時間 43 分) | B4（L） |
| 32 | 高橋，金田 | HITAC S-3800/480 | 1995.8 | (4294967296)<br>4294960000 | 4294967286 | 113 時間 41 分<br>(130 時間 20 分) | B4（L） |
| 33 | 高橋，金田 | HITAC S-3800/480 | 1995.10 | (6442450944)<br>6442450000 | 6442450938 | 116 時間 38 分<br>(131 時間 40 分) | B4（L） |
| 34 | G. V. Chudnovsky,<br>D. V. Chudnovsky | 不明 | 1996.3 | (8000000000 以上)<br>8000000000 ？ | 8000000000 ？ | 1 週間？<br>(不明) | C（C） |
| 35 | 高橋，金田 | HITACHI SR2201 | 1997.7 | (51539607552)<br>51539600000 | 51539607510 | 29 時間 3 分<br>(37 時間 8 分) | B4（L） |
| 36 | 高橋，金田 | HITACHI SR8000 | 1999.4 | (68719476736)<br>68719470000 | 68719476693 | 32 時間 54 分<br>(39 時間 20 分) | L（B4） |
| 37 | 高橋，金田 | HITACHI SR8000 | 1999.9 | (206158430208)<br>206158430000 | 206158430163 | 37 時間 21 分<br>(46 時間 7 分) | L（B4） |
| 38 | 金田，後，黒田ほか | HITACHI SR8000/MPP | 2002.11 | (1241177304180)<br>1241100000000 | 1241177304180 | 423 時間 20 分<br>(178 時間 36 分) | T（S4） |
| 39 | 高橋 | T2K 筑波システム<br>(Appro Xtreme-X3 Server) | 2009.4 | (2576980377600)<br>2576980370000 | 2576980377524 | 29 時間 5 分<br>(44 時間 30 分) | L（B4） |

公式欄の M，K，G，S3，L，R，B4，B2，C，T，S4 はそれぞれ Machin，Klingenstierna，Gauss，Störmer の 3 項，Gauss-Legendre，Ramanujan，Borwein の 4 次の収束，Borwein の 2 次の収束，Chudnovsky，高野，Störmer の 4 項の各公式である．

表 -1　計算機による円周率計算の記録

$$\frac{\pi}{4} = 4\arctan\frac{1}{5} - \arctan\frac{1}{239}$$
$$= 4\left(\frac{1}{5} - \frac{1}{3\cdot 5^5} + \frac{1}{5\cdot 5^5} - \cdots\right)$$
$$-\left(\frac{1}{239} - \frac{1}{3\cdot 239^5} + \frac{1}{5\cdot 239^5} - \cdots\right) \quad (1)$$

ところが，Gauss の算術幾何平均による完全楕円積分の計算アルゴリズムと，楕円積分に関する Legendre の関係式を組み合わせた公式が，1976 年に Brent[1] と Salamin[2] の 2 人によって独立に発見された（以後，Gauss-Legendre の公式と呼ぶ）．

この公式を用いることで，$n$ 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）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いる[3]．したがって，この方法は桁数 $n$ が大きな領域で，arctan の Taylor 展開を用いた計算量 $O(n^2)$ の方法よりも有利であり，1982 年から円周率の計算に用いられるようになった．

一方，2002 年の円周率 1 兆 2411 億桁計算（表 -1 の記録 38）では，arctan の Taylor 展開を分割有理数化法[4] という方法で計算することにより，$O(M(n)(\log n)^2)$ の計算量で $n$ 桁の円周率の値を求めることを可能にしている．

また，D. V. Chudnovsky と G. V. Chudnovsky は彼らが発見した

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

の公式を用いて，1989 年から 1996 年まで記録を更新している．Chudnovsky の公式は，arctan の Taylor 展開による公式よりも収束が速く，計算量も少ないことが知られている．その反面，平方根や逆数計算が必要になることから，必要とするメモリ容量は arctan の Taylor 展開による公式よりも多くなる．彼らの計算の詳細は明らかになっていないが，前述の分割有理数化法と類似の方法を用いていると推測される．

## ■ 今回の計算に使ったプログラムについて

Gauss-Legendre の公式を用いて円周率を $n$ 桁計算すると，$n$ 桁どうしの多倍長乗算の計算量を $M(n) = n\log n\log\log n$ とした場合，$O(n(\log n)^2\log\log n)$ の計算量が必要となる．一方，arctan の Taylor 展開や Chudnocsky の公式を分割有理数化法（または類似の方法）により計算すると，$O(n(\log n)^3\log\log n)$ の計算量となるため，漸近的な計算量としては Gauss-Legendre の公式が有利である．しかし，実際には定数係数の関係で，2 兆桁程度の桁数では Gauss-Legendre の公式と arctan の Taylor 展開ではほぼ同等の計算量となり，Chudnovsky の公式の計算量が最も少なくなる．

したがって，計算時間の観点からは計算量が最も少ない Chudnovsky の公式が有利であるが，今回は 1997 年の円周率 515 億桁計算[5]，1999 年の 687 億桁計算および 2061 億計算で用いた実績がある Gauss-Legendre の公式を用いた．

プログラムはすべて Fortran，OpenMP および MPI（Message Passing Interface）により記述されており，主計算用が 3124 行，検証計算用が 3248 行（共にコメントを含む）からなっている．それぞれのプログラムのうち 1179 行(コメントを含む)が FFT に関するものである．

## ■ 使用した計算式と特徴

### ❍主計算に用いられた方法

主計算に用いられた Gauss-Legendre の公式を以下に示す．

$a_0 = 1$, $b_0 = 1/\sqrt{2}$, $t_0 = 1/4$, $p_0 = 1$ として，

$$a_{k+1} = \frac{a_k + b_k}{2} \quad (3)$$
$$b_{k+1} = \sqrt{a_k b_k} \quad (4)$$
$$t_{k+1} = t_k - p_k(a_k - a_{k+1})^2 \quad (5)$$
$$p_{k+1} = 2p_k \quad (6)$$

とすると，

$$\pi = \lim_{k\to\infty}\frac{(a_k+b_k)^2}{4t_k} \quad (7)$$

である．この公式は 2 次の収束を示すので，$n$ 桁の $\pi$ は $\log_2 n$ 回程度の反復で求まる．

### ❍検証計算に用いられた方法

1980 年代以降，J. M. Borwein と P. B. Borwein が発見した円周率を計算するいくつかの公式[6]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 4 次の収束の公式を以下に示す．

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

$$y_{k+1} = \frac{1-(1-y_k^4)^{1/4}}{1+(1-y_k^4)^{1/4}} \quad (8)$$
$$a_{k+1} = a_k(1+y_{k+1})^4 - 2^{2k+3}y_{k+1}(1+y_{k+1}+y_{k+1}^2) \quad (9)$$

とすると，

$$\pi = \lim_{k\to\infty}\frac{1}{a_k} \quad (10)$$

である．Borwein の 4 次の収束の公式では，必要となる反復回数は 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 2 倍以上になっていることから，全体の計算量としては Gauss-Legendre の公式の方が少なくなる．どちらの公式でも，$O(M(n)\log n)$ の計算量で $n$ 桁の円周率が計算できることになる．

## ■ 多倍長数の平方根，4 乗根および逆数計算アルゴリズム

Gauss-Legendre の公式や Borwein の 4 次の収束の公式には，加減乗算のほかに平方根，4 乗根および逆数計算が含まれている．多倍長数の平方根，4 乗根および逆数計算は Newton 法を用いると効率よく計算できることが知られている．

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，

$$x_{k+1} = x_k + \frac{x_k}{2}(1 - ax_k^2) \tag{11}$$

となるが，式 (11) 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．

また，最後の反復においては $\sqrt{a}$ を

$$\sqrt{a} \approx (ax_k) + \frac{x_k}{2}(a - (ax_k)^2) \tag{12}$$

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．

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．

$a$ 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，

$$x_{k+1} = x_k + \frac{x_k}{4}(1 - ax_k^4) \tag{13}$$

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 (13) においても式 (11) 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．

$a$ の逆数 $1/a$ 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，

$$x_{k+1} = x_k + x_k(1 - ax_k) \tag{14}$$

で求める．式 (14) においても式 (11)，(13) 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．

## ■ 並列計算機上での多倍長計算の実現方法

並列計算機上で多倍長計算を行う際には，各プロセッサで桁数を分割して格納し，計算を行うことになる．

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，サイクリック分割で 4 バイト整数の配列に格納している．4 バイト整数には，10 進 9 桁まで格納可能であるが，多倍長乗算において FFT を計算する際には，FFT の精度の関係で 10 進 4 桁で計算を行う必要がある．したがって，10 進 4 桁との変換が容易である 10 進 8 桁を採用している．

なお，2002 年の 1 兆 2411 億桁計算では，まず 16 進数で計算した後に 10 進数に変換している．この方法では，途中に出てくるキャリーやボローの処理において，2 のべき乗による除算をシフト演算で行うことができ，さらに 4 バイト（または 8 バイト）整数の配列の各要素にデータをぎりぎりまで詰めることができるというメリットがある．

しかし，この場合基数変換ルーチンが別途必要になり，バグが入り込む余地が増えることから，今回は計算時間の短縮よりも確実性を重視して 10 進数ベースで計算を行った．

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．これらの処理を並列化するには，桁上げ先見 (carry look-ahead) や桁上げ飛び越し (carry skip) 方式などを用いることが考えられるが，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．また多倍長数どうしの減算や，多倍長数どうしの乗算においても，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．

多倍長数どうしの乗算の並列化においては，FFT も並列化する必要があるが，MPI を用いた並列数値計算ライブラリに含まれている並列 FFT ルーチンを用いることも可能である．

今回用いたプログラムでは，FFT を計算する際にデータがサイクリック分割になっていることが通信量の観点からは望ましいが，商用の並列数値計算ライブラリではデータがブロック分割になっていることが多い．

そこで，筆者が作成した並列 FFT ライブラリである FFTE[☆1] をデータがサイクリック分割になるように修正したものを用いた．

## ■ 計算機環境

T2K 筑波システムは，「T2K オープンスパコン仕様」に基づいた，Appro Xtreme-X3 Server が 648 ノード，

☆1 http://www.ffte.jp/

| | |
|---|---|
| ノード台数 | 648 |
| 理論ピーク性能 | 95.39TFlops |
| ノード構成 | 4 ソケット／ノード |
| CPU | Quad-Core AMD Opteron 8356（Barcelona，2.3GHz） |
| L1 キャッシュ | 64KB × 4（命令）＋ 64KB × 4（データ）（各コア独立） |
| L2 キャッシュ | 512KB × 4（各コア独立） |
| L3 キャッシュ | 2MB（コア間共有） |
| ノード当たりのメモリ容量 | 32GB（DDR2 667MHz） |
| 総メモリ容量 | 20TB |
| ノード当たりの最大メモリバンド幅 | 42.7GB/s |
| ローカルディスク容量 | 250GB × 4（SATA-II，RAID-1） |
| ファイルサーバディスク容量 | 800TB（RAID-6） |
| ネットワークインタフェース | DDR InfiniBand Mellanox ConnectX HCA × 4 |
| ネットワークトポロジ | Fat Tree（full-bisection bandwidth） |
| 最大リンクバンド幅 | 8GB/s |
| OS | Red Hat Enterprise Linux version 4 WS (Linux kernel 2.6) |
| メッセージ通信ライブラリ | MVAPICH（Appro による修正版） |
| システム規模 | 総ラック数：74（ノード，スイッチ他：69 ラック，ファイルサーバ：5 ラック）<br>総電源容量：745kVA |

**表 -2　T2K 筑波システムの諸元**

10,368 コアからなる大規模 PC クラスタである．ノード間は multi-rail の InfiniBand を用いた Fat Tree 相互結合網で接続されている．T2K 筑波システムの諸元を**表 -2** に示す．

また**図 -2** に，全 74 ラックからなる T2K 筑波システムの全景を示す．

今回の計算では，648 ノードのうち 640 ノードを用いた．プログラム上は 648 ノードすべてを用いて計算することも可能であるが，多倍長数どうしの乗算に FFT を使っていることから，ノード数に 2 のべきを多く含んでいた方が計算の効率が良くなること，またノード障害が生じた際の予備ノードを複数確保しておきたかったのが，その理由である．MPI プロセス数はノードあたり 4 プロセスとし，合計で 2560MPI プロセスとした．

## 世界記録樹立までの道

### ■円周率計算とのかかわり

筆者が円周率の計算に興味を持ったのは，中学校 1 年生の時に「π の話」（野崎昭弘著，岩波書店）という本に出会ったのがきっかけである．当時は電卓で円周率の計算を行っていたが，その後ポケットコンピュータやパソコンを購入し，触れてきたほぼすべてのコンピュータで円周率の計算を行ってきた．

その後，1995 年に東京大学大学院理学系研究科博士課程に入学してからは，金田康正教授にご指導いただき，円周率の計算に本格的に取り組む機会に恵まれたが，2001 年に筑波大学に異動してからは，円周率の計算からは遠ざかっていた．

**図 -2　T2K 筑波システムの全景（648 ノード，74 ラック）**

### ■T2K 筑波システムによる円周率計算

2008 年 6 月に T2K 筑波システムの運用が開始される前に，連立一次方程式をガウスの消去法で解く速度を測定する LINPACK ベンチマークにより T2K 筑波システムの性能を計測したところ，76.46TFlops の性能が得られた．しかし，LINPACK ベンチマークは演算量に対する通信時間の比率が小さく，またファイル I/O がほとんど生じないことから，システム全体の性能や信頼性を評価するという観点からは必ずしも十分であるとはいえない．

これに対して，円周率計算のプログラムでは，実行時間の 70% 以上を占める FFT の処理において全対全通信という非常に重い通信が頻発するとともに，メモリやファイルシステムに対しても負荷をかけることができることから，システム全体の性能や信頼性の評価に向いている．しかも，計算がうまくいけば，今まで知られていなかった円周率の値を知ることもできる．

そこで，計算機の性能・信頼性等の評価のための高性能計算の一例として円周率計算を行いたい旨を提案したところ，筑波大学計算科学研究センターに認めていただ

き，計算を行う運びとなった．

最初に円周率計算に挑戦する機会は，2008 年 10 月の T2K 筑波システムの本運用開始直前の 2008 年 9 月末に訪れた．しかし，その際は 640 ノード（2560MPI プロセス）という大規模構成において，MPI ライブラリのメモリ消費量が予想以上に大きいという問題点が見つかり，計算を断念せざるを得なかった．

その後，2009 年 4 月に入り，月 1 回行っている定期メンテナンスに合わせて T2K 筑波システムを使わせていただく機会に恵まれ，2008 年 9 月末とは別の MPI ライブラリを用いることで，円周率 2 兆 5769 億 8037 万桁の主計算と検証計算を行うことができた．

## ■ハードウェア障害に対する対策

計算機で円周率を計算する際には，プログラムにミスがないことはもちろんのことであるが，ハードウェアの信頼性も重要となる．特に，最近の並列スーパーコンピュータではマルチコアプロセッサのクラスタ構成となっており，メモリ容量も増えていることから，ハードウェア障害（たとえば，メモリのアンコレクタブルエラー）なしに長時間連続で計算することが難しくなりつつある．

今回の計算では，640 ノード（10240 コア）の構成を用いたが，そのうち 1 ノードでも障害が発生すると，計算は途中で停止してしまうことになる．そのため，障害が起こった時に備えて，計算の途中で再開に必要なデータ（主計算では約 8.2TB，検証計算では約 4.7TB）をファイルサーバに保存するチェックポイントを主計算，検証計算においてそれぞれ 4 回ずつ行っている．なお，これらのチェックポイントに要した経過時間の合計は，主計算で 1 時間 20 分，検証計算で 57 分であった．

今回の計算においては，主計算（29 時間 5 分）および検証計算（44 時間 30 分）を通して，一度もマシンダウンが発生しなかったことから，チェックポイントは行わなくても計算が成功した可能性はあるが，貴重な計算機資源を無駄にしないためにも，今回のような大規模計算ではチェックポイントは必須といえる．

また，プログラムの各所で計算が正しく行われているかをチェックしている．具体的には，

- FFT を用いた多倍長乗算において，逆 FFT の結果が整数に近い値になっているかどうか．
- 多倍長数の平方根，4 乗根および逆数の計算で用いている Newton 法による反復において，正しく収束しているかどうか．
- Gauss-Legendre の公式および Borwein の 4 次の収束の公式で，正しく収束しているかどうか．
- チェックポイントにおいて，ファイルにデータを書き込む際に計算したチェックサムが，読み込み時にも一致しているかどうか．

などである．このように各所でチェックを行うことで，計算時間はその分余計にかかることになるものの，万が一プログラムにミスがあったり，ハードウェア障害が生じた際に，原因を突き止めやすくなる．

## ■円周率 2 兆 5769 億 8037 万桁計算の結果

この円周率 2 兆 5769 億 8037 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 2009 年 8 月 17 日から，以下のように公開している[☆2]．

主計算について：

計算開始：2009 年 4 月 9 日午前 07 時 37 分
計算終了：2009 年 4 月 10 日午後 12 時 43 分
経過時間：29 時間 05 分 49 秒
主記憶容量：13.5 TB
アルゴリズム：Gauss-Legendre の公式

検証計算について：

計算開始：2009 年 4 月 27 日午後 09 時 35 分
計算終了：2009 年 4 月 29 日午後 06 時 06 分
経過時間：44 時間 30 分 33 秒
主記憶容量：12.9 TB
アルゴリズム：Borwein の 4 次の収束の公式

Gauss-Legendre の公式と，Borwein の 4 次の収束の公式による $3 \times 5^2 \times 2^{35} = 2{,}576{,}980{,}377{,}600$ 桁の円周率の計算結果を比較したところ，最後の 76 桁（丸め誤差）を除きすべて一致していた．そこで，記録の安全性と記憶のしやすさを考慮し，切りの良い 2 兆 5769 億 8037 万桁を新記録と宣言した．

なお，$\pi$ の小数点以下 2,576,980,369,951 桁から 2,576,980,370,000 桁までは，

```
3616276346 5152343138 0598550567
3249553206 9855284552
```

であった．

円周率 2 兆 5769 億 8037 万桁のデータは約 1.3TB にもなり，2560 個に分割したそれぞれ約 500MB のファイルに保存している．

$\pi$ の小数点以下 2 兆 5000 億桁までの 0 ～ 9 の数字の分布を調べた結果を**表 -3** に示す．

$\pi$ の小数点以下の数列について，昇順，降順，連続する同じ数字，そして $e = 2.71828\cdots$ や $\pi = 3.14159\cdots$ の数列がどこに出現するかを調べた結果を**表 -4** に示す．

## おわりに

本稿では，円周率 2 兆 5769 億 8037 万桁計算の概要，

☆2 http://www.hpcs.cs.tsukuba.ac.jp/~daisuke/pi-j.html

| 数字 | 回数 |
|---|---|
| 0 | 249999192826 |
| 1 | 249999959334 |
| 2 | 250000751269 |
| 3 | 249999904969 |
| 4 | 250000455856 |
| 5 | 249999721513 |
| 6 | 249999564178 |
| 7 | 249999660121 |
| 8 | 250001040584 |
| 9 | 249999749350 |

**表 -3　$\pi$ の小数点以下 2 兆 5000 億桁までの 0 ～ 9 の数字の分布**

| 数列 | 出現場所 |
|---|---|
| 012345678901 | 小数点以下 1 兆 7815 億 1406 万 7534 桁目から |
| 012345678901 | 小数点以下 2 兆 3641 億 9038 万 6673 桁目から |
| 987654321098 | 小数点以下 5931 億 54 万 6152 桁目から |
| 987654321098 | 小数点以下 1 兆 1161 億 603 万 8318 桁目から |
| 8888888888888 | 小数点以下 2 兆 1641 億 6466 万 9332 桁目から |
| 271828182845 | 小数点以下 1 兆 160 億 6541 万 9627 桁目から |
| 271828182845 | 小数点以下 1 兆 5359 億 1732 万 8677 桁目から |
| 314159265358 | 小数点以下 1 兆 1429 億 531 万 8634 桁目から |

**表 -4　$\pi$ における興味深い数列**

および記録更新に至るまでの道程について述べた．

今回の計算においては主計算および検証計算を通して，一度もマシンダウンが発生しなかったことから，T2K 筑波システムの高い信頼性を実証することができたと考えている．

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，計算機の主記憶容量の増加と性能向上に伴い，今後さらに桁数は伸びると考えられる．

しかし，Gauss-Legendre の公式や Borwein の 4 次の収束の公式を用いて円周率を計算したとしても，桁数を $n$ とすると $O(n(\log n)^2 \log\log n)$ の計算量が必要となることから，$(\log n)^2$ の項による計算量の増加が無視できなくなってきている．したがって，今後桁数の伸びは次第に緩やかになっていく可能性がある．

今回いくつか取材を受けた際に，円周率計算の意義について聞かれることが多かったので，最後に筆者の意見をまとめておきたい．

小数点以下 37 桁までの円周率の値が分かれば，宇宙の円周を水素原子の直径ほどの精度で求めることができるといわれており，円周率の値そのものを詳しく求めても，直接何かに役立つというわけではない．

しかし，今回の円周率計算においては T2K 筑波システムの性能や信頼性を実証することができたばかりでなく，プログラムの高速化の過程において大規模システムにおける並列 FFT ライブラリの改良にもつながった．

表 -3 を見るかぎり，$\pi$ の値は小数点以下の数字が等しい確率で出現する数（正規数）であるように見えるが，$\pi$ が正規数であることは，まだ数学的に証明されていない．もしかすると $\pi$ の数列には他の数学定数（$e$，$\gamma$，$\sqrt{2}$ など）の数列と区別するような，何らかの法則が隠されている可能性があるかもしれないのである．

紀元前 3 世紀にアルキメデスが円周率の値を「3.14」と計算して以来，円周率の値をより正確に求める努力が続けられてきた．その後 2000 年以上が経過したが，これまでの円周率計算の歴史を振り返ってみると，円周率の桁数は文明の進歩の尺度の 1 つになっているといえる．

今後，円周率の桁数がどこまで伸びるのかはまったく予想がつかないが，数学やコンピュータが進歩するかぎり，人類の挑戦は続くのではないかと考えている．

**謝辞**　T2K 筑波システムにおいて円周率計算を行うにあたり，ご支援をいただいた筑波大学計算科学研究センターの佐藤三久センター長および朴泰祐副センター長，ならびにご協力をいただいたクレイ・ジャパン・インク，Appro International, Inc.，（株）HPC ソリューションズ，住商情報システム（株）の関係者諸氏に深く感謝いたします．

**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 $\pi$ Using Arithmetic-Geometric Mean, Math. Comput., Vol.30, pp.565-570 (1976).
3) Knuth, D. E. : The Art of Computer Programming, Vol.2 : Seminumerical Algorithms, Addison-Wesley, Reading, MA, 3rd ed. (1997).
4) 後　保範，金田康正，高橋大介：級数に基づく多数桁計算の演算量削減を実現する分割有理数化法，情報処理学会論文誌，Vol.41, pp.1811-1819 (2000).
5) 高橋大介，金田康正，分散メモリ型並列計算機による円周率の 515 億桁計算，情報処理学会論文誌，Vol.39, pp.2074-2083 (1998).
6) Borwein, J. M. and Borwein, P. B. : Pi and the AGM — A Study in Analytic Number Theory and Computational Complexity, Wiley, New York (1987).

（平成 21 年 10 月 2 日受付）

**高橋 大介（正会員）** daisuke@cs.tsukuba.ac.jp
昭和 45 年生．平成 3 年呉工業高等専門学校電気工学科卒業．平成 5 年豊橋技術科学大学工学部情報工学課程卒業．平成 7 年同大学院工学研究科情報工学専攻修士課程修了．平成 9 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．同年同大大型計算機センター助手．平成 11 年同大情報基盤センター助手．平成 12 年埼玉大学大学院理工学研究科助手．平成 13 年筑波大学電子・情報工学系講師．平成 16 年同大学院システム情報工学研究科講師．平成 18 年同助教授，平成 19 年同准教授．博士（理学）．並列数値計算アルゴリズムに関する研究に従事．平成 10 年度本会山下記念研究賞，平成 10 年度，平成 15 年度本会論文賞各受賞．日本応用数理学会，ACM，IEEE，SIAM 各会員．