多倍長精度演算の性能評価

日時　2011年12月10日　11:30-12:00
場所　工学院大学 新宿校舎 28階第4会議室

高エネルギー加速器研究機構

濱口信行

hamagu@post.kek.jp

1. はじめに

計算センター　＝＞ユーザ

プログラムの実行効率は何%です。（よく出ていますor改善してください)

実行性能=演算量/実行時間

実行効率=実行性能/理論性能

ユーザ

実行時間=演算量/実行性能　に関心

例えば,フーリエ変換とFFT：フーリエ変換のほうがFFTより実行時間がかかるのに,実行性能がよい.

両者の実感が一致する⇔演算量が確定,理論性能の定義が明確

最近では実数の倍精度演算の場合のみ明確化されている.

今回の発表の動機

4倍精度の性能がどうこうと言う話を聞くが、そもそも
4倍精度以上の多倍長精度演算の性能とは何？

2. 多倍長精度演算

演算方式には,浮動小数点演算方式と整数演算方式がある.

2.1 浮動小数点演算方式

多倍長精度変数を複数個の倍精度変数の和で表す。
表現できる数値範囲は,ieee754の倍精度と同じ.

a : $2n$倍精度変数

$a_1, a_2, ..., a_n$: 倍精度変数

$a = a_1 + a_2 + ... + a_n$

有効ビット数＝仮数部のビット数＋1

$2n$倍精度$(52+1)\times n$　　ビット

$n=2$　　106ビット

$n=3$　　159ビット

$n=4$　　212ビット

乗加算命令がある場合とない場合の処理に分かれる。
コンパイラの最適化方式に性能が大きく依存する。

基本の演算

倍精度変数の加減算,乗除算の結果を２つの倍精度変数の和で表す。

a,b,c,d倍精度変数;　$a \pm b, a \times b, a \div b = c + d$

加算 : $c = a + b$

$t = c - a$

$d = (a - (c - t)) + (b - t)$

減算 : $c = a - b$

$t = c - a$

$d = (a - (c - t)) - (b - t)$

加算,減算ともに最適化オプションで括弧をはずした演算順序に
ならない様に注意が必要

乗算(乗加算命令あり.)

$c = a * b$

$d = a * b - c$　　（最適化オプションにより$d = 0$とならないように
注意が必要)

乗算(乗加算命令なし)

$r = 134217729.0d0$　$(= 2^{27} + 1)$

$c = a * b$

$t1 = a * r$

$a1 = t1 - (t1 - a)$

$a2 = a - a1$

$t2 = b * r$

$b1 = t2 - (t2 - b)$

$b2 = b - b1$

$d = ((a1 * b1 - c) + a1 * b2 + a2 * b1) + a2 * b2$

演算量　　加減算　　n^2に比例
　　乗算　　n^3に比例

$n \geq 3$では

n個の変数の絶対値が大きい順にならべるソート処理が必要.

2.2 整数演算方式

多倍長精度変数はieee754-2008の4倍精度の仮数部を変えたものであらわす。
表現できる数値範囲はieee754-2008の4倍精度と同じ.

a：p倍精度($p \geq 3$)変数
　符号部　1ビット
　指数部　15ビット
　仮数部　$32 \times p - 16$ビット
有効ビット数
　$p = 4$　113ビット
　$p = 6$　177ビット
　$p = 8$　241ビット

p 倍精度

演算量　　加減算　　pに比例
　　乗算　　p^2に比例

符号部 1ビット	指数部 15ビット	仮数部 32*p-16ビット(p=4: 112,p=6: 176,p=8: 240)

これらから多倍長精度演算の性能に関しては,以下の2つの影響が大きい.

(1)浮動小数点演算方式(dd形式)か整数演算方式(ieee形式)か.
(2)乗加算命令(fma)を使用するかしないか.

またこれらの性能には,計算機アーキテクチャーとコンパイラの最適化手法が大きく左右する.

これらを,具体的な例題を用いて計算方法と精度の解説を加え,

精度上問題のないものに対して性能を評価した.

3.Rump's 例題

$$a = 77617.0, b = 33096.0$$

$$f = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + \frac{a}{2b}$$

$$a^2 = 5.5b^2 + 1 \qquad \text{より}$$

$$f = -2 + \frac{a}{2b} = -\frac{54767}{66192} = -0.82736059946.....$$

倍精度演算　$f = -0.11805916207174113 \times 10^{22}$

4倍精度演算　$f = 1.17260394053.....$

全く異なった結果となる.!

原因

$$A = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2)$$
$$= -79171113406689613611011347015429428 50.0$$
$$B = 5.5b^8 = \quad 79171113406689613611011347015429428 48.0$$

A＋Bの演算で121ビットの桁落ちが発生する事

有効桁数10進37桁以上の多倍長精度演算が必要となる.

幾つかの計算機でこの計算を1,000,000回実行したときの実行時間（秒）
コンパイラはIntel fortran,ieee形式は整数型32ビット演算.

計算機アーキテクチャーとコンパイラの最適化手法が大きく左右する事を示している.

xeon(3.06GHz)
V8.1

精度	ieee	dd
5倍精度	6	
6倍精度	7.63	3.44
7倍精度	8.69	
8倍精度	11.91	7.8

amd275(2.2GHz)
V9.1

精度	ieee	dd
5倍精度	2.53	
6倍精度	2.79	3.42
7倍精度	3.28	
8倍精度	3.97	7.71

amd285(2.6GHz)
V9.1

精度	ieee	dd
5倍精度	2.04	
6倍精度	2.36	2.63
7倍精度	2.52	
8倍精度	3.34	6.08

x5355(2.66GHz)
V9.1

精度	ieee	dd
5倍精度	1.75	
6倍精度	1.99	1.95
7倍精度	2.21	
8倍精度	2.83	4.52

xeon とx5355　同じインテル系でもコンパイラV8.1とV9.1 で様子が異なる.
同じコンパイラV9.1でもインテル系とオプテロン系て様子が異なる.

Cプログラム:拡張倍精度変数を2つつなげると,有効ビット数130,有効桁数10進39桁.

<=正しく計算できる条件を満たす.

gcc標準でコンパイル.1,000,000回実行したときの実行時間(秒)は以下の表。

	dd形式のCプログラム		
精度		e5430	x5570
周波数		2.66GHz	2.93GHz
long double*2		0.3509	0.284462
double*3		0.538608	0.412117
double*4		1.200533	0.867511

4.数学関数

評価関数:　arctan(x)を1,000,000回実行したときの実行時間（秒）の比較
dd形式:各精度に特化したものと一般化したもの
Ieee形式:整数型32ビット演算,整数型64ビット演算

4.1 arctan(x)の計算概略

$$\arctan(x)=\sum_{n=1}^{\infty}(-1)^{n-1}\frac{x^{2n-1}}{2n-1} \qquad (|x|\le 1)$$

$$\arctan(-x)=-\arctan(x) \qquad (x>0)$$

$$\arctan(x)=(sign\quad x)[\frac{\pi}{4}+\arctan(\frac{|x|-1}{|x|+1})]$$

これにより, $\arctan(x) \quad (0\le x<1)$ の計算ができれば良い.

$\frac{1}{16}\le x<1$に対して, $w=\frac{j}{16}$ $(j=1,2,...,15)$として, $v=\frac{x-w}{1+xw}$とすると,

$\arctan(x)=\arctan(w)+\arctan(v)$から求める.$(0\le v<\frac{1}{16})$

$0\le x<\frac{1}{16}$での$\arctan(x)$はテーラー展開で求める.

（注）$\frac{j}{16}$付近の値でのxに対し, vを求める際の桁落ち防止の処理が必要

4.2 性能評価に関して

テーラー展開での項数は 有効ビット数mに対して,$\frac{(2^{-4})^{2n+1}}{2n+1} < 2^{-m}$となる$n$の最小値を取っている

8倍精度までは、有効ビット数mはieee形式の値をとり,10倍精度以上では,dd形式とieee形式で差が出るため,性能測定では,dd形式とieee形式の平均に設定した.

具体的項数

4倍精度 15, 5倍精度 19, 6倍精度 22, 7倍精度 26, 8倍精度 29
10倍精度 35, 12倍精度 42, 14倍精度 49, 16倍精度 56

次ページ以降の結果では,10倍精度以上の演算精度が必要な場合は性能面より,ieee形式を使用するのが良い事を示している.

4.3 性能測定結果

arctan(x)の1,000,000回実行時間(秒)

amd275(2.2GHz)+V9.1

演算精度	dd(一般化)	ieee(32)	dd(特化)	ieee(64)
4倍精度		2.924555	2.121677	
5倍精度		6.262048		5.705133
6倍精度	12.21914	7.783816	6.574001	7.360882
7倍精度		11.36327		9.455562
8倍精度	29.48752	15.54964	20.93982	13.30498
10倍精度	61.25069	27.91676		
12倍精度	113.8389	48.94156		
14倍精度	196.2212	67.93167		
16倍精度	324.4177	92.71891		

arctan(x)の1,000,000回実行時間(秒)

e5430(2.666GHz)+V9.1

演算精度	dd(一般化)	ieee(32)	dd(特化)	ieee(64)
4倍精度		1.555764	1.12483	
5倍精度		4.597301		3.774426
6倍精度	6.18406	5.826114	3.225509	4.729281
7倍精度		8.623688		6.428023
8倍精度	15.45765	11.60424	9.941489	8.112767
10倍精度	34.8487	19.67401		
12倍精度	63.7993	32.04013		
14倍精度	111.902	45.46109		
16倍精度	178.8458	65.56403		

arctan(x)の1,000,000回実行時間(秒)

corei7(3.2GHz)+V11.1

演算精度	dd(一般化)	ieee(32)	dd(特化)	ieee(64)
4倍精度		1.141826	1.009846	
5倍精度		3.580456		3.009543
6倍精度	4.835265	4.662291	2.324648	3.835417
7倍精度		6.78197		5.388181
8倍精度	11.8282	9.373575	6.942944	6.727977
10倍精度	24.75824	15.96657		
12倍精度	48.73459	24.53227		
14倍精度	86.98378	34.2338		
16倍精度	135.8693	46.25797		

arctan(x)の1,000,000回実行時間(秒)

x5570(2.93GHz)+V11.1

演算精度	dd(一般化)	ieee(32)	dd(特化)	ieee(64)
4倍精度		1.004848	0.963853	
5倍精度		3.319495		2.877563
6倍精度	4.618297	4.197361	2.214663	3.597453
7倍精度		6.025085		5.079228
8倍精度	11.32728	8.35073	6.650989	6.351034
10倍精度	26.05304	14.54879		
12倍精度	51.89811	22.76154		
14倍精度	89.85934	31.17426		
16倍精度	145.4339	41.33872		

5 行列積の計算

dd形式での演算量（計算式:倍精度浮動小数点演算命令を単位）

精度	加減算	乗算（fmaなし）	乗算（fmaあり）
4倍精度	11	24	10（12）
6倍精度	27	78	36（40）
8倍精度	49	175	91（98）

（注） fma命令ありの場合,fma命令は2flopとなるので,flopを単位としたときの演算量は（ ）内の値となる.

SR16000/M1 での性能モニターで測定した値(MFLOP)
(n=1024)

精度	fma なし	対倍精度	計算	fma あり	対倍精度	計算
4倍精度	39788	18.5	17.5	25791	12	11.5
6倍精度	117353	54.7	52.5	80572	37.5	33.5
8倍精度	235232	109.6	112	169719	79	73.5

（注）倍精度演算は2147

4倍精度以上の多倍長精度演算の演算量はモニター値などを使用するのが良い.

6.Two loop s221

S221の一般式

$$S^{221}(s;m_1^2,m_2^2,m_3^2,m_4^2,m_5^2)=\int_0^1\int_0^{1-x}\int_0^{1-x-y}\int_0^{1-x-y-z}\frac{1}{CD}dudzdydx$$

$$C=(x+y+z+u)(1-x-y-z-u)+(x+y)(z+u)$$

$$E=(1-x-y-z-u)(x+z)(y+u)+(x+y)zu+(z+u)xy$$

$$M^2=xm_1^2+ym_2^2+zm_3^2+um_4^2+(1-x-y-z-u)m_5^2$$

$$D=-sE+M^2C$$

今回の扱ったケースと解析近似解

$$S^{221}(s;m^2,m^2,0,0,m^2)=\frac{1}{2m^2}\sum_{n=0}^{\infty}\frac{n!\Gamma(\frac{3}{2})}{\Gamma(n+\frac{3}{2})}(\frac{s}{4m^2})^n[\frac{3}{(n+1)^2}-\frac{\ln(y)}{n+1}]$$

$$y=-\frac{s}{m^2}-i\varepsilon$$

精度検証結果

$$s=-1, m_1^{\,2}=m_2^{\,2}=m_5^{\,2}=100, m_3^{\,2}=m_4^{\,2}=0$$

解析近似解＝3.8000443813e-02
倍精度演算結果＝3.8000443813e-02
4倍精度演算結果＝3.8000443813e-02

$$s=10, m_1^{\,2}=m_2^{\,2}=m_5^{\,2}=100, m_3^{\,2}=m_4^{\,2}=0$$

解析近似解　　=0.026673229004550 5973
倍精度演算結果　=0.026673229277280 3997　積分誤差 0.191D-08

積分法 ： 二重指数関数型積分法
加速法 ： ε一算法 s=10の場合１５回反復

性能評価結果

$$s=-1, m_1{}^2=m_2{}^2=m_5{}^2=100, m_3{}^2=m_4{}^2=0$$

二重指数関数型積分 (n=576)

GPGPU(理論性能1088GFLOPs)

倍精度　　13.813 秒　　582GFLOPs （ソースから倍精度演算が単位）
4倍精度　225.517 秒　　35.6GFLOPs （ソースから4倍精度演算が単位）

性能はプログラムでカウントしているので,最適化による演算削減分は含まれていない.
倍精度/4倍精度=16.3 倍　(実行時間比)

SR16000/M1(理論性能980GFLOPs)

倍精度　　11.777 秒　　性能モニター　346GFLOPs （倍精度演算が単位)
4倍精度　　332.393秒　　性能モニター　127GFLOPs (倍精度演算が単位）

倍精度/4倍精度=28.2 倍　（実行時間比)
倍精度演算数=346*11.777=4075GFLOP
4倍精度演算数=127*332.393=42214GFLOP
演算数比　4倍精度演算数/倍精度演算数=10.4
最適化による演算削減より,演算量はプログラムでカウント　した場合より,性能モニターでの値は少なくなっている.

4倍精度以上の多倍長精度演算では,プログラムから演算量を算出して,例えば4倍精度演算性能をQGFLOPs等と表現する方法がある.

特に実行効率を重視するサイトで4倍精度演算、および多倍長精度演算を使用するユーザには.

モニター等で倍精度演算量を採取し,4倍精度演算および多倍長精度演算を同じベースで評価する方法が有効.

7.特異点のないinfra box

積分式

$$I=\int_0^1\int_0^{1-x}\int_0^{1-x-y}\frac{1}{D^2}dzdydx$$

$$D=-sxy-tz(1-x-y-z)+(x+y)\lambda^2+(1-x-y-z)(1-x-y)m_e{}^2$$
$$+z(1-x-y)m_f{}^2$$

$$s=-500^2, t=-150^2, m_f=150, m_e=0.0005, \lambda=10^{-30}$$

解析近似解

$$I=\frac{1}{-s(-t+m_f{}^2)}\ln(\frac{-s}{\lambda^2})\ln\frac{(-t+m_f{}^2)^2}{m_e{}^2m_f{}^2}$$

相対誤差7×10^{-26}以下

解析近似解の算出方法

$$I = \int_0^1 \int_0^{1-x} \int_0^{1-x-y} \frac{1}{D^2} dz dy dx$$

$$D = -sxy - tz(1-x-y-z) + (x+y)\lambda^2 + (1-x-y-z)(1-x-y)m_e^2 +$$
$$z(1-x-y)m_f^2$$
$$s, t < 0, -s, -t >> m_e^2 >> \lambda^2$$
$z = (1-x-y)p, dz = (1-x-y)dp$ で変数変換して，p を z と付け直すと，
$$I = \int_0^1 \int_0^{1-x} \int_0^1 \frac{1-x-y}{D^2} dz dy dx$$
$$D = -xys + (x+y)\lambda^2 + (1-x-y)^2 A(z)$$
$$A(z) = tz^2 + (-t + m_f^2 - m_e^2)z + m_e^2$$

$x+y=\rho$

$D=-x(\rho-x)s+\rho\lambda^2+(1-\rho)^2A(z)$

$\frac{\partial}{\partial x}(\frac{x}{D})+\frac{\partial}{\partial \rho}(\frac{\rho}{D})=\frac{\rho\lambda^2+2(1-\rho)A(z)}{D^2}$

$\frac{\rho\lambda^2}{D^2}$の項を省略すると,

$I=\frac{1}{2}\int_0^1\frac{1}{A(z)}dz\times[\int_0^1\int_x^1(\frac{\partial}{\partial x}(\frac{x}{D})+\frac{\partial}{\partial \rho}(\frac{\rho}{D}))d\rho dx]$

$\int_0^1\int_x^1(\frac{\partial}{\partial x}(\frac{x}{D})+\frac{\partial}{\partial \rho}(\frac{\rho}{D}))d\rho dx=\int_0^1\int_0^\rho\frac{\partial}{\partial x}(\frac{x}{D})dxd\rho$

$+\int_0^1\int_x^1\frac{\partial}{\partial \rho}(\frac{\rho}{D})d\rho dx$

$=\int_0^1\frac{1}{\rho\lambda^2+(1-\rho)^2A(z)}d\rho+\int_0^1\frac{1}{-x(1-x)s+\lambda^2}dx-\int_0^1\frac{1}{x\lambda^2+(1-x)^2A(z)}dx$

$=\int_0^1\frac{1}{-x(1-x)s+\lambda^2}dx$ より,

$I=\frac{1}{2}\int_0^1\frac{1}{A(z)}dz\times\int_0^1\frac{1}{-x(1-x)s+\lambda^2}dx$

$\int_0^1\frac{1}{A(z)}dz=\frac{1}{-t+m_f^2}\ln\frac{(-t+m_f^2)^2}{m_e^2m_f^2}$

$\int_0^1\frac{1}{-x(1-x)s+\lambda^2}dx=\frac{2}{-s}\ln\frac{-s}{\lambda^2}$

$I=\frac{1}{-s(-t+m_f^2)}\ln\frac{-s}{\lambda^2}\ln\frac{(-t+m_f^2)^2}{m_e^2m_f^2}$

性能測定結果

cpu :T2K筑波システム　AMD quad-core Opteron 8000 シリーズ(Barcelona)
1 node:ピーク性能　147GFLOPs,16MPI/node
積分法:二重指数関数型積分(サイズn=1024)
要求精度:解析近似解と10進6桁まで一致する事

	実行時間一覧表(秒)			
精度	64MPI	128MPI	256MPI	形式
4倍精度	8.572	4.319	2.19	dd
4倍精度	7.569	3.784	1.891	ieee
6倍精度	20.191	10.668	5.503	dd
8倍精度	88.097	46.324	23.791	dd
8倍精度	79.043	39.759	20.067	ieee
12倍精度	237.786	119.357	59.691	ieee
16倍精度	355.702	177.481	88.869	ieee
32倍精度	1307.441	651.315	326.991	ieee

(注)この解析近似解と10進6桁まで一致する様にするため,dd形式の4倍精度のみ,
分点のテーブル作成部分を以下の様に変更している。
解析近似解0.35617368……D-06
修正前　x30(i)=exp(y)/(exp(y)+exp(-y)),　result=0.35615337207937480 5D-06
修正後　x30(i)=1.0q0/(1.0q0+exp(-2.0q0*y)),result= 0.35617366244969523 3D-06

8.特異点のあるinfra box

$$I=\int_0^1\int_0^{1-x}\int_0^{1-x-y}\frac{1}{D^2}dzdydx$$

$$D=-sxy-tz(1-x-y-z)+(x+y)\lambda^2+(1-x-y-z)(1-x-y)m_e{}^2$$
$$+z(1-x-y)m_f{}^2$$

$$s=500^2, t=-150^2, m_f=150, m_e=0.0005, \lambda=10^{-30}$$

特異点のある場合の計算では,s<0で求めた式にs>0の値を代入した値と積分結果を比較する.

$\lambda=10^{-30}$より,この解析近似解は十分な精度をもっている.また有理化は

$s->s+i\varepsilon$で行い,$\varepsilon->0$を求める加速法はε－算法を使用している.

理由

これは$D->D-i\varepsilon$で有理化すると虚数部の計算で異なる３つの値が得られ,
プログラムは正しくても物理現象と合わない結果が出ることによる.

領域分割

実数部の値を求めるため,積分領域を4つの領域に分割し,各領域での二重指数関数型積分法で積分値を求めて最終結果を得ている.

分割の方法は積分領域を [0,1]＊[0,1]＊[0,1] にしてｚ軸を最外側にする.
また0.5に対する対称性を利用して積分領域を [0,1] から [0,0.5] にする.

積分実行前に$\alpha = \frac{4m_e^{\ 2}}{s-4m_e^{\ 2}}\left(\sqrt{1+\frac{s-4m_e^{\ 2}}{4m_e^{\ 2}}}-1\right)$を計算する.

Zの値がきまるとA(z)の値が定まる.

$\beta = \frac{4A(z)}{S-4A(z)}\left(\sqrt{1+\frac{s-4A(z)}{4A(z)}}-1\right)$を求める

この値より,$(x,y)=[0,\beta]\times[0,1]\oplus[\beta,1]\times[0,0.5]$の領域を以下の４つに分割する.

領域1$[0,\alpha]\times[0,1]$

領域2$[\alpha,\beta]\times[0,1]$

領域3$[\beta,1]\times[0,\beta]$

領域4$[\beta,1]\times[\beta,0.5]$

(注－1)領域分割では,$0<\lambda^2<<1$を使用している.

(注－2)⊕は領域の和を表している.

実数部の積分計算結果が要求される相対誤差0.1%以下をみたすのに要する時間は64スレッド（ＭＰＩ）で以下の様に変遷している.

実行時間一覧表(秒)

CPU	SR11000	XM1	T2K	SR16000
理論性能 (GFLOPs)	537.6	844.8	588	980.49
4倍精度	681	589	414	224
6倍精度	346	401	372	208
8倍精度	382	458	708	314

infra box 多倍長精度演算結果(実数部)

解析近似解　-0.3561736812D-06
4倍精度　　-0.3560594322D-06　相対誤差 0.032%
6倍精度　　-0.3561608223D-06　相対誤差　0.004%
8倍精度　　-0.3561449971D-06　相対誤差　0.008%

以上の結果をみるとこの問題にはdd形式6倍精度が最も適していると言える.

9.まとめ

(1)4倍精度,多倍長精度演算ではモニター等により,実行時の倍精度演算数を採取し、それをもとに同じ基準で性能をみるのが良い。

(2)dd形式とieee形式では4倍精度から8倍精度まで性能面では特に優劣はない.

(3)10倍精度以上の演算精度が必要な場合はieee形式が良い.

(4)4倍精度演算で精度不足の場合,8倍精度ではなく,dd形式の6倍精度を使用するのが良い.