Vol.2014-ARC-213 No.8
Vol.2014-HPC-147 No.8
2014/12/9

ファインマンループ積分によるアクセラレータの精度,性能評価

濱口信行[†1]　　石川正[†1]

ファインマンループ積分に於いては,扱う問題により必要となる演算精度と,変数の数値範囲が多岐に渡ります。本報告では,この数値積分と複数のアクセラレータでの精度及び性能の関係とその高速化に関して紹介します。

Accuracy and Performance evaluation of accelerators by Multi-loop Feynman integrals

1. はじめに

素粒子反応の計算で,多数の問題を扱う場合,特定のアーキテクチャーの計算機で実行するよりも,異なるアーキテクチャーの複数の計算機で実行する方が作業が効率的に進みます。素粒子反応の計算の特徴とそれに適したアーキテクチャーを考えるために使用した計算機とそのカタログ性能は以下のものです。

HD5870 :1088GFLOPs,HD6970:2703GFLOPs,
HD7970:947GFLOPs,Phi5110P:1011GFLOPs,
E5-2670:333GFLOPs,SR16000(1 ノード): 980GFLOPs。

性能測定は HD5870 3200 演算器,HD6970 6144 演算器,HD7970 2048 演算器,Phi5110P 240 演算器を使用して実行し,E5-2670 では 16SMP,SR16000 では 64SMP で実行し,E5-2670,Phi5110P は別々にスレッド並列で実行しています。

SR16000 は演算量測定や最適化効果などのソフトウエア面の比較に用い,他にサーバーX5570(周波数 2.93GHz)1 CPU 実行と比較してアクセラレータの効果などを見ています。

本報告の記述では浮動小数点数の精度,演算に関して,指数部 11 ビットの変数,演算に関しては”倍精度”,”4 倍精度”,指数部 15 ビットの変数,演算に関しては拡張と言う言葉を付加し,”拡張倍精度”,”拡張 4 倍精度”,と称することにしています。また浮動小数点数を一つの変数で表わす場合は単一,複数の変数の和で表わす場合は合成という言葉を付加して区別しています。

2. ループ積分と演算精度,変数の数値範囲

2.1 ループ積分概要

素粒子反応の計算では,Feynman 図を作成し,この図に対する散乱振幅を導出するループ積分が現れ,これを数値計算で求める事により,通常の数値計算の問題に帰着されます。一般的な表式は以下の様になります。[1]

$$I(\varepsilon) = (-1)^N \left(\frac{1}{4\pi}\right)^{nL/2} \Gamma(N - nL/2) \times$$
$$\int_0^1 \prod_{i=1}^{N} dx_i \delta(1 - x_1 - x_2 - ..... - x_N) \frac{C^{N-n(L+1)/2}}{(D - i\varepsilon C)^{N-nL/2}}$$

n : 時空次元($n = 4$),
N : ループ内の素粒子の数(内線の数),
L : ループの多重度,
ε : 実定数, C, D : $x_1, x_2, ..., x_N$ の多項式

この積分では,一般的に解析解または解析近似解はないので数値積分を使用する事になります。ここで演算精度及び変数の数値範囲を決めるのは,式の中に仮想光子(λ : 質量0) を含むか含まないかです。

これ以降,本報告では式の中に仮想光子を含む場合を,インフラ問題と記述しています。インフラ以外の問題では,単位が GeV からプログラム上に現れる変数の値は $10^{-7} \sim 10^{6}$ に限られます。積分領域で分母が 0(特異点)にならなければ,$\varepsilon = 0$ として演算精度は,倍精度で事足ります。また積分域内に特異点が存在する場合,微小量 ε を用いて,有理化して,ε ->0 で得た値を求める積分値としますが,変数の数値範囲は倍精度で事足り,演算中に桁落ちが発生しても 4 倍精度で事足ります。

インフラ問題では,積分値は $\log(\frac{1}{\lambda^2})$ の多項式の多項式(主に 1 次式,2 次式)を含みますが,数値積分の制限により,λ の値の取り方で,演算精度及び変数の数値範囲が大きく異なってきます。

2.2 数値積分による制限事項

被積分関数 f(x) が積分区間[0, 1]の x=0 で端点特異点を持つとします。数値積分に於いては ε に下限値が存在します

†1 高エネルギー加速器研究機構

ので, その下限値をε_0とします。

f(x)の形により以下の例の様に所要の誤差の範囲内で正しく計算出来る場合と出来ない場合があります。

$$I=\int_0^1 \frac{1}{\sqrt{x}}dx = 2(1-\sqrt{\varepsilon_0}) \fallingdotseq 2 \quad (I\text{が有限値})$$

$$I=\int_0^1 \frac{1}{x}dx = \log(\frac{1}{\varepsilon_0}) \qquad (I\text{が発散})$$

この後者の例がインフラ問題に大きく影響します。

簡単な例として以下のもので検証する場合を考えます。

$$I_n=\int_0^1 \frac{1}{xM^2+(1-x)\lambda_n^2}dx = \frac{1}{M^2-\lambda_n^2}\log(\frac{M^2}{\lambda_n^2})$$
$$(\lambda_n = 10^{-n}, M=100)$$

を考え, $I_{n+1}-I_n$ よりこの積分値が $\log(\frac{1}{\lambda^2})$ の一次式となる事を検証します。

数学的には$\mathrm{I_n}$は単調増加数列で発散しますが, 数値積分では

$$I_n \le \frac{1}{M^2}\log(\frac{M^2}{\varepsilon_0})$$
$$(\varepsilon_0 = 10^{-300}\text{とすると}, I_n \le 0.07)$$

と有界なため収束します。このため検証には n の値に対して

$\varepsilon_0 << 10^{-2n}$ という必要条件が加わります。変数の数が多くなると, 式中に変数の 5 乗や 6 乗(例えば, 4 次元積分の変数変換の Jacobian は変数の 6 乗)が現れ, 指数部 11 ビットではεの下限値として $\varepsilon_0 = 10^{-60}, 10^{-50}$ しか取れないため n=30 の場合には”倍精度“の検証が出来なくなり, 拡張精度が必要となります。この場合, 現状の GPGPU では高速化が図れないという問題が発生します。

2.3 インフラ問題と計算可能性

使用している数値積分法は二重指数関数型積分法[2]で, 今以下の三次元問題を考えます。

$$I=\int_0^1\int_0^{1-x}\int_0^{1-x-y}\frac{1}{D^2}\quad dzdydx$$
$$(D=-xys-tz(1-x-y-z)+(x+y)\lambda^2$$
$$+(1-x-y-z)(1-x-y)m_e^2$$
$$+z(1-x-y)m_f^2)$$

パラメータの物理的意味

m_e : 電子の質量 $0.511\times10^{-3}\,GeV$,
m_f :フェルミ粒子の質量 GeV ,
s,t : 衝突エネルギー $(GeV^2), t<0$
λ : 仮想光子質量 (GeV)。

s<0 の場合の解析近似解は

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

となります。[3]

インフラ問題では, λの値に関して以下の 2 つの条件を満たす必要があります。

(1) 物理の問題では 10 進 10 桁の精度が必要なときがあるため,

モデル化の相対誤差$\frac{\lambda}{m_e}$から $\lambda \le 5.11\times10^{-14}$。

(2) 数値積分法で精度の良い結果を得るための条件から, 1に最も近い数$1-\varepsilon(\varepsilon>0)$に対し

$0<\varepsilon<\frac{\lambda}{\sqrt{-s}}$。$s=-500^2$ の場合は,

$\lambda=5.11\times10^{-14}$で$\frac{\lambda}{\sqrt{-s}}=1.022\times10^{-16}$

倍精度演算での $\varepsilon=2^{-53}>1.022\times10^{-16}$ から, インフラ問題では倍精度演算では物理的に意味のある計算が出来ないと言う事になります。

3. アクセラレータでの評価問題

アクセラレータを使用するのは, 総演算量(倍精度演算換算)1TFLOP 以上として, その演算量からインフラ以外の問題は 4 次元以上の積分, インフラ問題の場合は, 3 次元以上の積分としています。結果の検証方法としては,

(1)インフラ以外の問題では解析近似解等はなく, 別の方式で計算された結果と比較。

(2)インフラ問題では 3 次元積分は解析近似解 (相対誤差$=\lambda\log\frac{1}{\lambda^2}$)と分点数 1024 で

10 進 6 桁一致する事(通常この様な場合, 分点数を 2048 にすると 10 進 10 桁以上一致します)。

(3)4 次元以上のインフラ問題では, 比較するものがなく, 物理的に結果の主要部分が

$\log(\frac{1}{\lambda^2})$の多項式になる事を$\lambda_n = 10^{-n}$ $(n = 16,....,30)$ での積分値I_nの1次差分,2次差分が10進10桁の範囲で一定値になる事。

としています。

4. アクセラレータでの性能測定結果

4.1 立ちあがり測定

アクセラレータの立ち上がりを見るために, 2 次元および 3 次元インフラ問題を倍精度演算で実行する事により行っています。測定条件は

infra vtx

$$I = \int_0^1 \int_0^{1-x} \frac{1}{D} dydx$$

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

$s < 0$の場合の解析近似解は

$$I = \frac{1}{-s}[\log(\frac{m_e{}^2}{-s})\log(\frac{\lambda^2}{m_e{}^2}) + \frac{1}{2}\log^2(\frac{m_e{}^2}{-s}) - \frac{\pi^2}{6}]$$

です。[3]

今回は以下のパラメータで実行しています。

$s = -500^2, m_e = 0.5\times10^{-3}, \lambda = 10^{-5}$

演算量$=16N^2$(N分点数)

infra box

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

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

今回は以下のパラメータで実行しています。

$s = -500^2, t = -150^2, m_f = 150,$

$m_e = 0.5\times10^{-3}, \lambda = 10^{-5}$

演算量$=27N^3$(N分点数)

でその性能測定結果を表 1, 表 2 に示しました。

表1 infra vtx 性能測定結果　単位:GFLOPs

分点数	HD5870	HD6970	HD7970	Phi5110P
8000	147.535	100.704	161.870	152.502
16000	187.115	149.178	219.567	142.751
24000	252.808	293.420	215.177	146.744
32000	265.909	380.142	290.673	148.311
40000	304.920	411.284	254.315	150.950

表2 infra box 性能測定結果 単位:GFLOPs

分点数	HD5870	HD6970	HD7970	Phi5110P
400	124.582	153.959	161.870	318.953
800	155.242	244.556	246.333	307.431
1200	302.784	386.378	295.296	300.480
1600	342.391	479.148	316.320	306.114
2000	367.455	541.553	333.480	309.615

Phi5110P は 240smp/60core とコア数が少ないので立ち上がりが速い事がわかります。

4.2 倍精度演算計算結果

別方式で計算した問題[4],[5],[6]を実行しています。扱った問題は以下の 4 問題です。サイズを N で表しています。

(1)$S221$

$$I = \int_0^1 \int_0^{1-x} \int_0^{1-x-y} \int_0^{1-x-y-z} \frac{1}{DC} dudzdydx$$

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

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

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

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

で変数変換により,積分区間を$[0,1]^4$にして
4重DOループのものを,ループ併合して
2重DOループにしています。
$N = 576$。
$s = -1, m_1{}^2 = m_2{}^2 = 100, m_3{}^2 = m_4{}^2 = 0, m_5{}^2 = 100$

(2)*Laporta* *D* (5次元)

$$I=\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\int_0^{1-x_1-x_2-x_3}\int_0^{1-x_1-x_2-x_3-x_4}\frac{1}{D^2}d\Omega$$

$d\Omega=dx_5dx_4dx_3dx_2dx_1$

$x_6=1-x_1-x_2-x_3-x_4-x_5$

を変数変換で積分区間を$[0,1]^5$にして,5重*DO*ループ
を外側*DO*ループ(2つの*DO*ループを併合)×
内側*DO*ループ(3つの*DO*ループを併合)
で, $N=120$。

$$\begin{aligned}D=&-x_1^2x_2-x_1^2x_3-x_1^2x_4-x_1^2x_6-x_1x_2^2-x_1x_2x_3\\&-2x_1x_2x_4-x_1x_2x_5-x_1x_2x_6-x_1x_3^2-2x_1x_3x_4\\&-x_1x_3x_5-x_1x_3x_6-x_1x_4^2-x_1x_4x_5-2x_1x_4x_6\\&-x_1x_5x_6-x_1x_6^2-x_2^2x_4-x_2^2x_5-x_2x_3x_4\\&-x_2x_3x_5-x_2x_4^2-2x_2x_4x_5-x_2x_4x_6-x_2x_5^2\\&-x_2x_5x_6-x_3^2x_4-x_3^2x_5-x_3x_4^2-2x_3x_4x_5\\&-x_3x_4x_6-x_3x_5^2-x_3x_5x_6-x_4^2x_5-x_4^2x_6\\&-x_4x_5^2-3x_4x_5x_6-x_4x_6^2-x_5^2x_6-x_5x_6^2\end{aligned}$$

(3)*Laporta* *G* (6次元)

変数変換により積分区間を$[0,1]^6$にして3重*DO*ループ
をひとまとめにした2重*DO*ループを作成。$N=120$。

$$I=\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\int_0^{1-x_1-x_2-x_3}\int_0^{1-x_1-x_2-x_3-x_5}\int_0^{1-x_1-x_2-x_3-x_5-x_6}\frac{C}{D^3}d\Omega$$

$d\Omega=dx_7dx_6dx_5dx_3dx_2dx_1$

$x_4=1-x_1-x_2-x_3-x_5-x_6-x_7$

$$\begin{aligned}C=&x_1x_4+x_1x_5+x_1x_6+x_2x_4+x_2x_5+x_2x_6+x_3x_4\\&+x_3x_5+x_3x_6+x_4x_5+x_4x_6+x_4x_7+x_5x_7+x_6x_7\end{aligned}$$

$$\begin{aligned}D=&-(x_1^2+x_2^2+x_3^2+x_7^2+x_1x_2+x_1x_3+x_1x_7\\&+x_2x_3+x_2x_7+x_3x_7)(x_4+x_5+x_6)\\&-x_4^2(x_1+x_2+x_3+x_5+x_6+x_7)\\&-(x_5^2+x_6^2+x_5x_6)(x_1+x_2+x_3+x_4+x_7)\\&-3x_4(x_1x_5+x_6x_7)\\&-2((x_1+x_2+x_3)x_4x_6+(x_2+x_3+x_7)x_4x_5)\end{aligned}$$

(4)*Laporta* *H* (6次元)

変数変換により積分区間を$[0,1]^6$にして3重*DO*ループ
をひとまとめにした2重*DO*ループを作成。$N=120$。

$$I=\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_3}\int_0^{1-x_1-x_3-x_2}\int_0^{1-x_1-x_3-x_2-x_7}\int_0^{1-x_1-x_3-x_2-x_7-x_6}\frac{C}{D^3}d\Omega$$

$d\Omega=dx_4dx_6dx_7dx_2dx_3dx_1$

$x_5=1-x_1-x_3-x_2-x_7-x_6-x_4$

$$C=(x_1+x_2+x_3+x_4)(x_4+x_5+x_6+x_7)-x_4^2$$

$$cc=x_1m_1^2+x_2m_2^2+x_3m_3^2+x_4m_4^2+x_5m_5^2+x_6m_6^2+x_7m_7^2$$

$$\begin{aligned}D=&-C\times cc+s(x_1x_2(x_4+x_5+x_6+x_7)\\&+x_5x_6(x_1+x_2+x_3+x_4)+x_1x_4x_6+x_2x_4x_5)\\&+tx_3x_4x_7\\&+p_1^2(x_1x_3(x_4+x_5+x_6+x_7)+x_3x_4x_5)\\&+p_2^2(x_2x_3(x_4+x_5+x_6+x_7)+x_3x_4x_6)\\&+p_3^2(x_5x_7(x_1+x_2+x_3+x_4)+x_1x_4x_7)\\&+p_4^2(x_6x_7(x_1+x_2+x_3+x_4)+x_2x_4x_7)\end{aligned}$$

今回は以下のパラメータで実行しています。

$m_1^2=m_2^2=m_3^2=m_4^2=m_5^2=m_6^2=m_7^2=1$

$s=t=1$

$p_1^2=p_2^2=p_3^2=p_4^2=1$

計算結果の精度と SR16000 の性能モニターによる 2loop box planar の演算量は表 3 の様になっています。

表3 倍精度演算積分の演算量と計算結果			
プログラム名	次元数	分点数	演算量 (GFLOP)
s221	4	576	4568
Laporta D	5	120	2514
Laporta G	6	120	312064
Laporta H	6	120	243328

計算結果

S221	0.038000443813
Laporta D	0.276209225359
Laporta G	0.172336790750
Laporta H	0.103640720989

計算結果は別方式(参考文献[4],[5]を参照)と今回の結果は 10 進 12 桁一致しており精度的に充分な結果となっています。
性能測定結果は表 4 の様になっています。比較のためスーパーコンピュータ SR16000, サーバーE5-2670 の値も記載しました。

表4 倍精度演算性能測定結果

実行時間(秒)一覧表
S221とLaporta X(=D,G,H)

CPU	S221
HD5870	13.815
HD6970	3.197
HD7970	9.165
Phi5110P	17.816
SR16000	24.126
E5-2670	26.959

CPU	D	G	H
HD5870	5.868	589.471	568.483
HD6970	3.138	346.624	362.633
HD7970	7.541	718.903	684.978
Phi5110P	7.474	962.356	925.742
SR16000	5.426	1418.553	1256.823
E5-2670	27.621	2907.167	2717.488

演算量の多いケースでは,GPGPU,Phi5110P の効果,特にGPGPU の効果が出ています。

またS221で多重ＤＯループの一重化の影響を表5に示しました。

表5 多重DOループの一重化の影響
プログラムS221 分点数N=576
実行時間：単位秒
演算量:一重化なし 2976GFLOP
　　　　一重化あり 4567GFLOP

一重化	SR16000	E5-2670	Phi5110P
	実行時間	実行時間	実行時間
なし	11.6450	27.6185	17.8365
あり	24.1262	26.9587	17.8157

多重 DO ループを一重化することにより,参照する 2 つのテーブルのサイズが 4.5KB から 2.5MB になることにより
スーパーコンピュータでは性能低下が起こっています
が Phi5110P では起こらなくなっています。

4.3　4 倍精度演算計算結果

4.3.1 合成数演算精度

4 倍精度浮動小数点数は常に倍精度浮動小数点の合成数となります。このため,1 に近い数 $1-\varepsilon, 1-\varepsilon = a+b = 1-c$ $(a,b,c$は倍精度変数）(a>0, b>0, c>0) の様に表わす事が出来ます。$1-\varepsilon = 1-c$ の場合は ε は充分小さく取れます。3 次元インフラ問題で $\lambda = 10^{-30}$ の場合，$\frac{\lambda}{\sqrt{-s}} = 2\times10^{-33}$ のため，$1-\varepsilon = a+b$ での $\varepsilon = 2^{-106}$は$\varepsilon > \frac{\lambda}{\sqrt{-s}}$ となり,計算結果の精度が問題となります。合成数の計算アルゴリズム[7]は,幾つかありますが,分点を求める計算

x=exp(y)/(exp(y)+exp(-y))（修正なし）

x=1.0q0/(1.0q0+exp(-2.0q0*y))（修正あり）

で計算した時の結果と解析近似解は表 6 の様になっています。

表6 INFRA　BOX　計算結果

解析近似解	3.56173681291824538D-07
	HD5870 ソース修正なし
分点数	value
1024	3.56107323854343038D-07
4096	3.56152442514549826D-07
	HD5870 ソース修正あり
1024	3.56173712303736817D-07
	E5-2670 ソース修正あり
1024	3.56173681720991588D-07

3 次元インフラ問題ではソースを修正しない場合はいくら N を増やしても精度は向上しませんが,ソースの修正により精度向上が見られます。ただし,この一例の様に使用するコンパイラのアルゴリズムに依存する事や,他の問題への影響の確認などが必要となり,Feynman 積分では多くのケースを扱うため,細心の注意を払う必要があります。

4.3.2　S221 性能測定結果

4 次元積分 S221 の性能測定結果は表 7 の様になっています。SR16000 の性能モニターでの演算量は 42187GFLOP で倍精度演算での 4568GFLOP の 9.2 倍となっています。

表7　S221 4倍精度演算性能測定結果
実行時間：単位秒
時間比:対倍精度演算時間

CPU	実行時間	時間比
HD5870	232.0790	16.8
HD6970	82.0468	25.7
HD7970	267.9379	29.2
Phi5110P	156.2803	8.8
SR16000	330.4291	13.7
E5-2670	773.7886	28.7

実行時間では,GPGPU,特に Phi5110P の効果が良く出ています。

対倍精度時間比では SR16000 が GPGPU より良くなっているのは,乗加算命令の適用の最適化効果が大きい事によります。また SR16000 では対倍精度演算量比 9.2 に対し,対倍精度時間比が 13.7 となっているのは,4 倍精度演算には simd 命令が適用できないため,2smp/core で実行する事によります。Phi5110P では演算量が多くなると,演算器を効率的に使用できるため,対倍精度時間比が他に比べて小さな値となっています。

4.4 拡張 4 倍精度演算計算結果

4.4.1 拡張精度演算の利点

拡張 4 倍精度変数には単一(有効ビット数 113),合成(有効ビット数 130)の場合があります。拡張精度は 3 次元インフラ問題では,倍精度,4 倍精度に比べて非常に優位な点があります。$\lambda = 10^{-15}$ の場合拡張倍精度

$$\varepsilon = 2^{-65} = 2.71\times10^{-20} < \frac{\lambda}{\sqrt{-s}} = 2\times10^{-18}$$

でモデル化による誤差も10^{-10}以下となっています。

解析近似解　0.192786112243964116D-06 に対し,
倍精度　　　0.191153752861597116D-06
(分点数 1024)
0.191182177302367076D-06
(分点数 8192)
拡張倍精度　0.192786109420426930D-06
(分点数 1024) となります。

λ=10^{-30}でも単一拡張4倍精度

$$\varepsilon = 2^{-113} = 9.63\times10^{-35} < \frac{\lambda}{\sqrt{-s}} = 2\times10^{-33}$$

合成拡張 4 倍精度 $\varepsilon = 2^{-130} = 7.35\times10^{-40} < \dfrac{\lambda}{\sqrt{-s}} = 2\times10^{-33}$

でともにソースの修正なしで精度の良い計算が出来る事になります。

解析近似解　0.356173681291824538D-06
に対し,
単一拡張 4 倍精度　0.35617344043706959D-06
(分点数 1024)
合成拡張 4 倍精度　0.35617367523816211D-06
(分点数 1024)
分点数 2048 では合成拡張 4 倍精度は
0.356173681291800954D-06
と解析近似解と 10 進 13 桁まで一致しています。

また Feynman 積分においては,2 次元積分でも,

$$\int_0^1\int_0^{1-x}\frac{1}{(xy)^{1-\eta}}dydx = \frac{1}{\eta^2}\frac{(\Gamma(1+\eta))^2}{\Gamma(1+2\eta)}$$

$$\eta = \frac{1}{\log(\frac{1}{\lambda^2})}$$

$\eta = 0.01, \lambda = 10^{-21.714721}$で

解析解　　　　9998.37886579431687

拡張倍精度　　9998.37886579431643（$n = 738, h = 0.5^6$）

拡張 4 倍精度　9998.37886579431687（$\mathrm{n} = 772, \mathrm{h} = 0.5^6$）

4 倍精度　　　9366.55046529275386（$\mathrm{n} = 2368, \mathrm{h} = 0.5^8$）

が示されており単に仮数部の精度が長くなるだけでは問題があり[8],別分野である物性スペクトル計算で,拡張精度演算の必要性が示されています[9]。

4.4.2 拡張 4 倍精度演算性能測定結果

既存の GPGPU では拡張演算はサポートされていないので,性能測定は E5-2670,Phi5110P を使用して行いました。検証の精度の範囲は,$\lambda = 10^{-30}$ までの計算でその差分が 10 進 10 桁以上一致する事を確認するために,条件をより厳しくして,3 次元インフラ問題で $\lambda = 10^{-40}$ が分点数 1024 でソース修正なしで 10 進 6 桁一致する事を確認した結果から単一拡張 12 倍精度までとしています。

性能測定は分点数 1024,3 次元インフラ問題で行い,4 倍精度に関しては精度の関係からソースは修正しています。結果はそれぞれ最速となるオプションにより実行したものを表 8 に示しています。

表8 各種演算精度による性能測定結果

精度	E5-2670	Phi5110P
	実行時間(秒)	実行時間(秒)
4倍精度	9.6243	9.7896
単一拡張4倍精度	23.8442	44.1840
合成拡張4倍精度	14.0411	21.8673
6倍精度	39.5971	49.2443
8倍精度	95.8929	92.5810
単一拡張8倍精度	396.6969	517.6158
合成拡張8倍精度	506.8777	908.5313
10倍精度	340.5119	525.3891
単一拡張10倍精度	419.7810	1001.4391
12倍精度	722.0146	868.2430
単一拡張12倍精度	593.8716	1307.8625

E5-2670 では表の 4 倍精度演算はオプション-fp-model

precise -openmp で実行したもので,Phi5110P では 9.8170 秒,-O3 -fp-model precise -openmp で実行すると,E5-2670 では 17.4562 秒と合成拡張 4 倍精度演算より遅くなり,Phi5110P では,表 8 の値で差が見られないと言う最適化の影響がでています。

単一拡張 8 倍精度,単一拡張 10 倍精度,単一拡張 12 倍精度演算は,整数演算で行っているため,8 倍精度,10 倍精度,12 倍精度演算と比較すると,E5-2670 では単一拡張 12 倍精度演算が 12 倍精度演算より速く,Phi5110P では 1.5 倍遅いという結果となっています。これは E5-2670 と Phi5110P の整数演算器構成の差による影響と言えます。

また単一拡張 4 倍精度演算は現在ソフトウエアサポートで,単一拡張 4 倍精度演算器が出来れば,合成拡張 8 倍精度演算の性能が向上すると考えられます。

4.4.3 5 次元インフラ問題性能測定結果

５次元インフラ問題

$$I=\int_0^1\int_0^{1-x_1}\int_0^{1-x_1-x_2}\int_0^{1-x_1-x_2-x_3-x_3}\int_0^{1-x_1-x_2-x_3-x_4}\frac{1}{D^2}dx_5dx_4dx_3dx_2dx_1$$

$$D=C(x_3M^2+x_6\lambda^2)$$
$$+(x_1+x_2+x_3)(x_4+x_5)^2+(x_1+x_2)^2(1-x_1-x_2)$$
$$+2x_3(x_1+x_2)(x_4+x_5)$$
$$x_6=1-x_1-x_2-x_3-x_4-x_5$$

を変数変換をして３次元インフラ問題に帰着して,演算量を大幅に削減したあと, $\lambda_n=10^{-n}(n=16,....,30)$ での積分値 I_n を計算して,1 次差分,2 次差分が 10 進 10 桁の範囲

で一定値になる事を検証する計算を行っています。帰着された３次元インフラ問題は以下の様になります。

$$I=\int_0^1\int_0^1\int_0^1\frac{J}{D^2}dxdydz$$
$$J=(1-x)^2yz$$
$$D=[(1-x+xy(1-y)][x(1-y)M^2+(1-x)(1-z)\lambda^2]$$
$$+[xy^2(1-xy)+(1-x)^2z^2+2x(1-x)y(1-y)z]$$

ここで M=0,M=91.19/0.1057 の場合を計算しています。
それぞれを case1,case2 としています。

変数変換区間[ε,1− ε]の ε は以下の事から定めています。
単調増加数列 I_n は収束するため,ある n_0 からは 2 乗収束するとして $(\frac{1}{2})^m=10^{-10}$ から I_{n_0+34} が極限値とします。
また n_0 は n=16,….,30 の計算を行うため,
$n_0=30\times2=60$ とし $n_0+34=94,\varepsilon=10^{-2\times94}$ から $\varepsilon=10^{-200}$ としています。刻み幅 h,分点数 N は

$$\int_0^1\frac{1}{xM^2+(1-x)\lambda^2}dx=\frac{1}{M^2-\lambda^2}\log(\frac{M^2}{\lambda^2})$$

$\lambda=10^{-30},M=100$ で相対誤差が 10^{-10} 以下となるのは $h=0.5^8$, 分点数 $N=2912$ と言う結果から,定めています。

ソースプログラムは分点の計算で 1 に近い数 $x=1-\varepsilon(\varepsilon\le2^{-65})$ は $x=\frac{e^y}{e^y+e^{-y}},\varepsilon=\frac{e^{-y}}{e^y+e^{-y}}$ より求める様に修正しています。

case1 の 2 次差分,case2 の 1 次差分は以下の様に 10 進 10 桁一致しています。

case1 2 次差分

n=16	5.301898110478400278
n=17	5.301898110478398252
n=18	5.301898110478398036
n=19	5.301898110478398013
n=20	5.301898110478398011
n=21	5.301898110478398010
n=22	5.301898110478398006
n=23	5.301898110478398055
n=24	5.301898110478398129
n=25	5.301898110478397037
n=26	5.301898110478403211
n=27	5.301898110478403097
n=28	5.301898110478319123

case2 1 次差分

n= 16	0.000038733248507
n= 17	0.000038733248507
n= 18	0.000038733248507
n= 19	0.000038733248507
n= 20	0.000038733248507
n= 21	0.000038733248507
n= 22	0.000038733248507
n= 23	0.000038733248507
n= 24	0.000038733248507
n= 25	0.000038733248507
n= 26	0.000038733248507
n= 27	0.000038733248507
n= 28	0.000038733248507
n= 29	0.000038733248507

またこの場合の実行時間は表 9 の様になっていてアクセラレータの効果がよくでています。

表9 5次元インフラ問題性能測定結果			
	実行時間(秒)一覧表		
ｹｰｽ	x5570	E5-2670	Phi5110P
case1	192628	7877	10769
case2	193515	7870	10787

5. まとめ

性能測定結果から以下の事が言えます。

(1) インフラ以外の問題では,GPGPU,Phi5110P の性能が良く出ています。特に演算量が多くなると,Phi5110P は演算器を効率的に使用できるので,より性能が良くなっています。

(2)インフラ問題では 4 倍精度で実行する場合,インフラ以外の問題と同じ傾向にあります。

(3)インフラ問題で 4 倍精度で精度の良い結果が得られない場合,合成拡張 4 倍精度が 6 倍精度の 2.5 倍の性能を示している様に,一般に合成数の演算量は合成数の数の 2 乗(加減算),3 乗(乗算)となりますので,合成数の数は 2 が適しています。

(4)4 倍精度超の演算が必要となる問題では,アクセラレータ Phi5110P の使用が有効となります。

以上から,Feynman 積分に於いては GPGPU,Phi5110P で ieee754-2008 形式の 4 倍精度演算器が追加されれば,精度的に問題なく性能が大幅(5 倍程度)に向上すると言えます。

6. おわりに

Feynman 積分に於いて,内部特異点を持たないケースでのアクセラレータの効果を検証しました。今回,スーパーコンピュータやサーバーでの自動並列化やインライン展開機能,mpi とスレッド並列のハイブリッド機能,内部特異点がある場合とそのノウハウの適用を充分に行っていないので,今後検討して行く予定です。尚,より詳細な結果や別分野の結果に関しては,高性能計算の扉[10]で 3 か月毎に更新しています。

謝辞 本研究は JSPS 科研費 23540328 の助成を受け、HPCI 戦略プログラム分野 5「物質と宇宙の起源と構造」の元で実施したものです。ループ積分の資料を提供いただいた高エネルギー加速器研究機構湯浅富久子准教授,GPGPU の実行環境を提供いただいた会津大学中里直人准教授に感謝いたします。

参考文献

1) 高エネルギー素粒子反応に対する高次補正を含む自動計算プログラム（その２）日本物理学会　第６６回年次大会
高エネルギー加速器研究機構:湯浅富久子,石川正,栗原良将,清水韶光,濱口信行　工学院大学院大学:加藤潔

2) 数値解析　森正武 2002 年　共立出版

3) Rediative Corrections to $e^{+}e^{-}$ Reactions in Electroweak Theory :
JUNPEI FUJIMOTO,Masatuka IGARASHI,NOBUYA NAKAZAWA,YOSHIMITSU SHIMIZU and KEIJIRO TOBIMATSU
Reprinted from Progress ofTheoretical Physics Supplement NO.100(1990)

4) High-precision calculation of multi-loop Feynman integrals by difference equation S.Laporta April,2000
arXiv:hep-12h/0102033 V1 3 Feb 2001

5) Algebraic-numerical evaluation of Feynman diagrams: two-loops self-energies
GiampieroPassriero,SandroUccirati
Nuclear Physhics B629(2002) 97-187

6) Progress on the Direct Computation Method
F.yuasa,T.ishikawa,N.Hamaguchi and Y.Shimizu(KEK)
E.de.Doncker/Westrn Michigan Univ.
K.Kato/kogakuuinUniv
ACAT2001 5-9 September 2001 at Brunel University

7) Quad-Double Arithmetic :
Algorithms,Implementation,and Application
Yozo Hida,XiaoyeS.Li,DavidH.Bailey October 30,2000

8) SR11000 での 4 倍精度、多倍長精度演算使用例
濱口信行（(株)日立製作所ソフトウエア事業部）
九州大学情報基盤研究開発センター　全国共同利用システム広報　Vol2 No.3 p.p.102-108,March 2009

9) 量子モンテカルロ法による物性スペクトル計算への多倍長演算の適用　濱口信行,石川正,岩野薫(高エネルギー加速器研究機構)　情報処理学会研究報告 IPSJ SIG Technical report 2013-HPC-141(12) 2013/10/3

10) 高性能計算の扉
http://www.jicfus.jp/field5/jp/promotion/hpcdoor