Vol.2009-HPC-121 No.39
2009/8/6

アクセラレータによる四倍精度演算

中里直人[†1]　石川　正[†2]
牧野　淳一郎[†3]　湯浅　富久子[†2]

本論文では, GRAPE-DR や GPU などのメニーコアアクセラレータにおける, 四倍精度演算 (DD 演算) の性能評価について報告する. 四倍精度相当の演算は倍精度演算器を利用することでエミュレーションすることができる. しかしそのためには DD 演算あたり 20-23 回の倍精度演算が必要であり, DD 演算の演算性能は, 倍精度演算性能と比べると 20 分の 1 以下となる. また, 通常の CPU では, 利用できる論理レジスタが最大で 16 個であるため, DD 演算性能はさらに低下する. 一方で, メニーコアアクセラレータをベクトル型演算器と考えると, 実効的なレジスタ数が非常に多いため, ループアンローリングが効果的に働くことで, 演算レイテンシを完全に隠蔽できる. 我々は GRAPE-DR と GPU で DD 演算をおこなうための基本ライブラリとそれを利用するためのコンパイラを開発した. これにより, メニーコアアクセラレータでの DD 演算性能を調べた. 結果, 通常の CPU での DD 演算性能より 30 - 90 倍の高速化が可能であることがわかった.

Fast Quad-Precision Operations On Many-core Accelerators

Naohito Nakasato ,[†1] Tadashi Ishikawa ,[†2]
Jun Makino [†3] and Fukuko Yuasa [†2]

In this paper, we present a performance analysis of many-core accelerators like GPU and GRAPE-DR with a special attention to an emulation scheme of double-double (DD) operations. The emulation scheme requires 20 and 23 double operations for add and mul DD operations, respectively. Accordingly, a performance of DD operations on a general purpose CPU is at least 20 times slower than its performance of D operations. In addition, the CPU has only 16 logical registers so that an optimization technique called loop-unrolling to hide latency of D opearations is not effective. On the other hand, many-core accelerators have much many logical registers of more than 30,000. That is the loop-unrolling technique is highly effective on many-core accelerators to completely hide the operation latency. We have developed a basic library for RV770 GPU and GRAPE-DR and a compiler system for the many-core accelerators. The obtained performance of DD operations on RV770 GPU and GRAPE-DR is 30 - 90 times faster than the performance of the general purpose CPU.

1. はじめに

今日, 多くの数値計算, 数値シミュレーションの実装には, 倍精度浮動小数点演算 (IEEE 754 倍精度では仮数部 52 ビット, 指数部 11 ビット, 符号部 1 ビット) が利用されている. これは, 多くの問題では倍精度程度の演算精度があれば十分であり, また通常の CPU には倍精度演算器がハードウエアとして実装されているため, 倍精度演算が高速に実行できるためでもある.

一方で, 倍精度までの演算精度は必要ないような問題があり, その性質を利用することで, 効率のよい高性能計算を実現することができる. 一例として, 天文学における無衝突系の多体問題では, 位置の減算と加速度の積算を除いてはたかだか仮数部 5 ビットの演算精度があれば十分であることが示されている[7]. また, 倍精度演算では十分ではない問題も多く存在しており, そのような問題では, 四倍精度演算 (拡張の一例では仮数部 112 ビット) やさらに拡張した精度での演算が使用されている. このように, 必要十分となる演算精度は, 本来問題に依存しており, それは必ずしも自明ではない. 倍精度よりはるかに低い演算精度で十分な場合があり, 四倍精度でも十分ではない場合もある. もし, 問題の性質を調べることで必要十分な演算精度がわかっているならば, 必要となる演算精度に応じて演算を実装する手法を変えて, 効率的な高性能計算の実現が可能な場合がある. なぜなら, ここ数年大きく発展してきたマルチコアやメニーコアの計算機では, 演算精度により大きく演算性能が異なるためである.

現在, 広く利用されている x86 アーキテクチャCPU には, ストリーミング SIMD 拡張演算器 (SSE 演算器) が搭載されている. この演算器では, 同時に 4 個の単精度浮動小数点演算が可能であり, その演算性能 (Intel 社の Nehalem アーキテクチャでは 2 GHz 動作の場合, コアあたり 8 GFLOPS) は, 通常の倍精度演算器の計算性能 (同 2 GFLOPS) の 4 倍となる.

†1 会津大学
University of Aizu
†2 高エネルギー加速器研究機構
High Energy Accelerator Research Organization
†3 国立天文台
National Astronomical Observatory of Japan

	SP	DP	$N_{\rm unit}^{\rm SP}$	$N_{\rm unit}^{\rm DP}$	clock	note
CPU	51	26	16	8	3.2	Core i7 SSE2/3
Cell1	200	20	24	6	3.2	Cell Broadband Engine
Cell2	230	109	32	16	3.2	PowerXCell 8i
GPU1	933	78	240	28	1.5	GT200b
GPU2	1200	240	800	128	0.75	RV770 (DP add)
GRAPE-DR	390	195	512	256	0.38	SING

表 1 x86 CPU と様々なマルチコア・メニーコア計算機の演算性能: 演算性能の単位は GFLOPS であり, clock 周波数の単位は GHz である.

よって, 単精度で必要十分な問題であれば, SSE 演算器を利用することで最大 4 倍高速な計算が可能となる. 同様に, 最新の Graphic Processing Unit(GPU) チップには, 単精度浮動小数点演算器が非常に多数搭載されている. 例えば NVIDIA 社の GT200b GPU には, 合計 240 個単精度演算器が搭載されており, 1.5GHz 動作時の単精度積和演算での総演算性能は 933 GFLOPS である. また, AMD 社の RV770 GPU には, 合計 800 個の単精度演算器が搭載されており, 750MHz 動作時の単精度積和演算での総演算性能は 1200 GFLOPS である. 一方でこれらの GPU では, 倍精度演算器の数はそれほど潤沢ではなく, GT200b では 28 個であり, RV770 では 128 個相当であるため, 倍精度積和演算での演算性能はそれぞれ 78 と 240 GFLOPS である.

High Performance Computing(HPC) の分野では, GPU に加えて専用に開発されたマルチコア・メニーコア演算装置が利用されはじめており, その代表として PowerXCell 8i や GRAPE-DR がある. PowerXCell 8i は PLAYSTATION 3 に使用されている Cell Broadband Engine の倍精度演算性能を増強したものであり, 合計 16 個の倍精度演算器を搭載している. 3.2GHz 動作時の単精度・倍精度の演算性能はそれぞれ 230 と 109 GFLOPS である. GRAPE-DR[5),6)] は, 512 個の倍精度単精度共用の加算器と 512 個単精度相当乗算器を搭載している. 380MHz 動作時の単精度・倍精度の演算性能はそれぞれ 390 と 195 GFLOPS である.

表 1 に, 以上の様々なマルチコア・メニーコアアーキテクチャに搭載されている, 単精度・倍精度演算器の個数とその性能をまとめた. その設計方針や利用目的に応じて, これらのマルチコア・メニーコアアーキテクチャ計算機では, 単精度演算性能と倍精度演算性能の間に 2 倍から 10 倍の性能差がある. よって, 単精度で十分な問題においては, これらの計算機の利用が効果的な場合がある. ただし, PowerXCell 8i や GPU, GRAPE-DR とも, 自律して計算をすることができず, ホスト計算機に付随するアクセラレータとして利用される. そのため, マルチコア・メニーコアアクセラレータを有効に利用するためには, 大前提として並列計算が可能であることが必要であるだけでなく, 演算密度の高い問題でなければならない. 典型的な演算密度の高い問題として, 重力多体問題やブロック化された行列乗算がある. どちらも, アクセラレータ型演算装置により高性能な結果が得られている[3),8)].

本論文では, 別種の演算密度の高い計算として, 四倍精度演算のエミュレーションを考える. 以下, 2 章では四倍精度エミュレーション手法の概要を説明し, それがメニーコアアクセラレータに向いていることを説明する. 3, 4 章では, x86 アーキテクチャCPU とメニーコアアクセラレータにおける, 四倍精度エミュレーションの演算性能評価について述べる. 引き続き, 5 章にて, 実アプリケーションによる演算性能評価について紹介し, 最後に 6 章で, まとめと今後の展望について述べる.

2. 四倍精度エミュレーションの概要

四倍精度演算のエミュレーション (以下, DD 演算と呼ぶ) 手法は, Knuth[4)] と Dekker[1)] により, 加算の場合と乗算の場合がそれぞれ提案された. この手法は, 倍精度変数の組によって四倍精度 (相当) の変数を表現し, 倍精度演算の組み合わせで, 四倍精度加算と乗算を実行する. この加算と乗算には, それぞれ 20 演算,23 演算[⋆1]が必要である. DD 演算は, 4 語の倍精度変数を入力とし 2 語の倍精度変数を出力とするので, 1 語読み出しあたりの演算数は 4.6 - 5.0 となり演算密度が高い. 演算密度が高いこと自体は, 複数階層からなるキャッシュ機構をそなえた通常の CPU においても, 高性能演算を実行するために有効である. ただし, それらの CPU において倍精度演算のレイテンシは 3 から 4 であるため, ループアンローリングによるレイテンシの隠蔽が必須となる. 1 回の DD 演算に必要な一時変数の語数は加算では 9 であり, 乗算では 8 であるため, 利用可能なレジスタ数が多ければ多いほど, ループアンローリングが有効に働くことが考えられる[9)]. 一方で, x86 アーキテクチャCPU では, AMD64 拡張においても論理レジスタが 16 語[⋆2]であるためループアンローリングは有効ではなく, DD 演算の性能は演算のレイテンシに大きく影響を受ける. そのため, 以下に示すように x86 アーキテクチャCPU での DD 演算性能はそれほど高くない.

GPU や GRAPE-DR などのメニーコアアーキテクチャ計算機は, 一種のベクトル型演算器と考えることができる. ベクトル演算とは, ベクトルレジスタの大きさの段数による, ハー

⋆1 絶対値が 2^{996} を越える場合の処理は考えない
⋆2 SSE レジスタの片方のみを考慮

ドウエアループアンローリングと論理的に同等である．よって, メニーコアアーキテクチャで有効な演算処理とは, ループアンローリングが有効な演算処理ともいえる．このように考えると, DD 演算はメニーコアアーキテクチャに適していることがわかる．つまり, GPU(RV770 アーキテクチャの GPU) と GRAPE-DR は, いずれも演算ユニットあたりそれぞれ 256, 72 語のレジスタを持ち, さらにそれぞれ 240 個, 512 個の演算ユニットを搭載しているので, 実効的なレジスタ数は 3 万語を越える．これは, メニーコアアーキテクチャをハードウエアループアンローリング機構と考えたときに, アンローリング段数が通常の CPU とは比べものにならないくらい大きくできるということであり, 演算のレイテンシを実質上無視することができる．実際, 以下に示すように GPU と GRAPE-DR における DD 演算は高速である．

3. x86 アーキテクチャにおける DD 演算の性能

本章では, x86 アーキテクチャにおける DD 演算の演算性能評価について述べる．そのために, DD 演算による加算と乗算をおこなうコードを C 言語により作成し, 同一演算を繰り返し実行することで要した実行時間とクロック数を計測した．テストは, Intel Core2 6600(動作周波数 2.4 GHz), Intel Xeon E5410(2.33 GHz), Intel Core i7 920(2.67 GHz) および Dual-Core AMD Opteron 2214(2.2 GHz) でおこなった．いずれの場合も Linux kernel 2.6.18 上にて gcc version 4.1.2 を利用し, 最適化オプションは”-O2” とした．DD 加算と乗算を, それぞれ 256M 回繰り返した場合の結果を表 2 にしめす．レイテンシは, 256M 回にかかった総クロック数を 256M で割って得たクロック数である．レイテンシの結果は, マイクロアーキテクチャにより結果が異なり, Intel 社のプロセッサでは, Core MA から最新の Nehalem に移行することで改善はされているが, AMD K8 アーキテクチャには及ばないことがわかる．

コンパイルされたアセンブラソース調べることで, いずれの場合にも倍精度演算には, SSE 命令である addsd/subsd/mulsd が利用されていることがわかった．x86 アーキテクチャは, 片方のソースレジスタが破壊される命令体系 (ISA) を採用しているため, 適宜 SSE レジスタ間コピー命令 (movsd) がコンパイラにより挿入されており, 演算に必要な総ステップ数は, DD 加算と乗算でそれぞれ 34, 37 ステップであった．この時に一時変数として利用されている SSE レジスタの数は, 加算の場合で 9, 乗算では 8 であった．よって, 論理レジスタを利用したループアンローリングはあまり効果的ではないと考えられる．一方, SSE 命令により同時に 2 個の倍精度演算を実行することにすれば, 実効演算性能は表 2 の倍程度にはなることが予想される．以上の結果から, 4 core 搭載の Nehalem アーキテクチャCPU で

	加算秒数	乗算秒数	加算 lat.	乗算 lat.	加算性能	乗算性能	clock	MA
Core2	4.545	5.829	40.6	52.2	59	46	2.4	Core MA
Xeon	4.502	6.002	39.0	52.2	60	45	2.33	Core MA
Core i7	3.617	4.621	36.0	46.0	73	58	2.67	Nehalem
Opteron	3.830	4.049	31.3	33.1	70	66	2.2	K8

表 2 DD 加算と乗算の各種 x86 アーキテクチャCPU における演算性能: 性能のコラムの単位は MFLOPS であり, clock 周波数の単位は GHz である．最後のコラムはマイクロアーキテクチャを示す．

あっても, 2.67 GHz 動作時の DD 演算の性能は加算・乗算それぞれの場合最大で 580, 460 MFLOPS 程度であると予測できる．

4. メニーコアアクセラレータにおける DD 演算の性能

我々は, GRAPE-DR と RV770 GPU で DD 演算による数値計算をおこなうための基盤ソフトウエアの研究をおこなってきた．具体的には GRAPE-DR と RV770 用の DD 演算ライブラリの開発と, それを有効利用するためのコンパイラの開発である．メニーコアアクセラレータにおいては, CPU と同等のベンチマークをおこなうことは困難であるため, 以下では, それぞれの場合について開発した演算ライブラリの詳細と, その性能評価について述べる．

4.1 GRAPE-DR の場合

GRAPE-DR の特徴をまとめると:

- 1 チップに 512 個の演算コアが搭載されている．
- それぞれのコアは同時に動作する浮動小数点加算器, 浮動小数点乗算器と整数演算器を内蔵する．
- 1 語は 36 ビットであり, 単精度変数は符号 1 ビット, 指数部 11 ビット, 仮数部 24 ビットである．倍精度変数 (仮数部が 60 ビット) の格納には 2 語必要である．
- それぞれのコアは 2 読み出し 1 書き込みの汎用レジスタを 32 語, 1 読み出し 1 書き込みの一時レジスタを 2 語, 1 読み出し 1 書き込みのメモリを 256 語内蔵する．
- 一時レジスタのみ連続する命令で利用可能
- 加えて, コア間で共有するメモリ領域を持つ．
- 命令は in-order で実行される．
- 外部メモリとの転送速度は $\sim 3 - 4$ GB s^{-1} である．

となる．

DD 演算を GRAPE-DR の演算命令で実装した結果, 加算は 21 ステップ, 乗算は 41 ステップ, 除算は 199 ステップとなった. GRAPE-DR では, 2 ソースレジスタ, 1 デスティネーションレジスタの ISA を採用しているため, 一部を除いてレジスタ間コピーは必要ない. 一方で GRAPE-DR の浮動小数点乗算器は単精度相当であるため, 乗算においては必要な演算数が増えている. 除算は倍精度での逆数計算と DD 加算と乗算の組み合わせで実装されている. 他に必要な演算も, 同様に実装することができる. 例えば逆数平方根の場合は, 倍精度での逆数平方根を求めるのに引き続いて DD 演算による Newton 法を適用をすることで計算できる.

GRAPE-DR では, 演算命令は in-order で実行されるため, 演算に必要なステップ数がわかれば演算性能を正確に予測することができる. ただし, 我々の実装では, 全ての命令は連続する 4 語のレジスタに対する SIMD 演算となっている. 以上をふまえると, DD 演算の GRAPE-DR での性能は, 動作周波数を 380 MHz とした時, 加算・乗算それぞれの場合 9.3 と 4.7 GFLOPS となる.

4.2 RV770 の場合

AMD 社の GPU である RV770(Radeon 4850/4870/4890 等に搭載) は以下のような特徴を持っている:

- 1 チップに 160 個の演算コア (Thread Processor; TP) が搭載されている.
- TP は 5 個の単精度演算器を内蔵し, そのうちひとつは関数演算も可能.
- TP は 5 way の VLIW 演算器であり, 最大 5 個の単精度積和演算を実行する.
- 個々の TP において, 倍精度加算は 2 個の単精度演算器, 倍精度乗算は 4 個の単精度演算器により実行される.
- 16 個の TP が一組となり SIMD Engine と呼ばれるユニットを構成する.
- SIMD Engine 内の TP は全て同一の命令流を実行する.
- 1 チップには 10 組の SIMD Engine が搭載されている.
- TP には 256 語の倍精度変数レジスタを持つ[⋆1].
- SIMD Engine 内の TP が共有するメモリ領域をもつ
- SIMD Engine 内には複数階層のキャッシュ機構を内蔵する.
- 外部メモリとの転送速度は $\sim 63 - 120$ GB s^{-1} である.

GRAPE-DR と RV770 は, ほぼ同等の倍精度演算性能を持ち, 倍精度加算が倍精度乗算より 2 倍性能が高いという点も同様である. 一方で, RV770 の大きな利点は, 外部メモリとの高速な転送速度と, 演算コアがもつ豊富な汎用レジスタである. GRAPE-DR では, 全ての演算が連続する 4 語のレジスタに対する SIMD 演算となるため, 実効的なレジスタ数は $(32 + 256)/4 = 72$ 語にすぎない.

RV770 の内部動作については詳細が明らかではないため, 以下では GRAPE-DR と同様の仮定で性能を評価する. AMD 社の GPU でのプログラミングは, C 言語を拡張した環境である Brook +と仮想的なアセンブラである Intermidiate Luangage(IL) でおこなうことができる. 実際には Brook+のソースは IL に変換され, IL は GPU での実行時に, GPU のアーキテクチャに合わせた機械語に, デイバイスドライバによって変換される. 本研究では, 最大性能を引き出すため DD 演算を IL により実装した. これを RV770 用の機械語 (アセンブリソース) に変換した結果から, TP に供給される VLIW 命令の必要ステップ数を得た. 結果, 加算は 21 ステップ, 乗算は 23 ステップ, 除算は 53 ステップとなった. おおむね, CPU 用の実装と同等の演算数で実装することができた. 演算が in-order で実行されると仮定すると, 動作周波数を 750 MHz(Radeon HD4870 相当) とした時, RV770 は秒間 $160 \times 750 \times 10^6 = 1.2 \times 10^{11}$ 個の VLIW 命令を処理することができる. よって, DD 演算のでの性能は加算・乗算それぞれの場合 5.7 と 5.2 GFLOPS となる.

実際には, RV770 の TP は最大 5 命令が実行可能な VLIW 演算器である. 単純な単精度演算のみが実行可能な 4 個の演算器を x, y, z, w ユニットと呼び, 関数演算が可能な演算器は t ユニットと呼ばれている. このうち, 倍精度加算では xy または zw スロットが利用される. また, 倍精度乗算では $xyzw$ のスロットが同時に　利用される. そのため, この最大 5 演算器分の命令スロットがどれだけ埋まるかにより, 演算性能が上下する. 生成された RV770 のアセンブリの内訳を調べることで以下のことがわかった. DD 加算を 1 演算実装した場合, $xyzwt$ の全スロットが利用された命令は 2 ステップだけであり, 残りのうち 8 ステップで $xyzw$ のスロットが利用されており, さらに残りの 13 ステップでは xy または zw スロットのみが利用されていた. 乗算と除算についても同様に調べた結果を表 3 にしめす. 2 命令スロットしか利用していない場合の割合は, 命令に依存して 35% - 59%であるため, 複数の DD 演算が連続する場合には, 前後の VLIW 命令が融合されて性能が上昇することが予想される.

5. 実アプリケーションでの演算性能

倍精度より高精度の演算が必要なアプリケーションで, メニーコアアクセラレータに適し

⋆1 実際には 128bit 幅の SIMD レジスタが 128 語

命令	5 slots	4 slots	3 slots	2 slots	1 slot	計
加算	2	8	0	13	0	22
乗算	5	9	1	8	0	23
除算	7	25	1	20	0	53

表 3 RV770 における DD 演算の VLIW 命令スロットの分布:

た計算として, ここではファインマンループ積分を考える[10]. その一例は, 以下のような多次元積分の評価に帰着される.

$$
\begin{aligned}
I &= \int_0^1 dx \int_0^{1-x} dy \int_0^{1-x-y} dz F(x,y,z), \\
F(x,y,z) &= D(x,y,z)^{-2} \\
D &= -xys - tz(1-x-y-z) + (x+y)\lambda^2 \\
&\quad +(1-x-y-z)(1-x-y)m_e^2 \\
&\quad +z(1-x-y)m_f^2. \qquad (1)
\end{aligned}
$$

ここで, s と t はパラメータであり, m_e と m_f は物理定数である. また, λ は仮想的な粒子の質量を表す. このような形の多次元積分を, s と t の異なる組み合わせについて計算する必要があり, その総数は 100 万組を超える.

二重指数型積分法を適用することで式 1 の積分は 3 重ループによる積算に帰着する. この時に, 最内ループでの演算は, 加算 10 演算, 乗算 15 演算そして除算 1 演算の合計 26 演算である. よって, 1 方向の領域分割数を N とした場合, この積分の評価には $26N^3$ 回の DD 演算が最低でも必要になる. 以下では, この最内ループ演算を GRAPE-DR と GPU で実行し得られた演算性能について報告する. この時, 外側のループの積算はホスト上で DD 演算にて実行した. ホスト上での DD 演算には qd ライブラリ[2] の version 2.3.7 を利用した.

5.1 本研究のコンパイラの概要

我々のコンパイラでは, 以下のような計算モデルを想定する:

$$S = \sum F(a,b,c,...). \qquad (2)$$

ここで F は, 任意の数の入力変数 $a, b, c, ...$ により計算される関数である. また, S は関数 F の累算によって計算される結果変数である. 式 1 の二重指数積分法による評価は, $a, b, c, ...$ を x, y, z の座標等と考えることで, 式 2 により計算することができる.

具体的に, 式 1 を 3 重ループで計算することを考えると, 最内ループの計算は x と y を固定して, z 方向にループで積算をすることになる. 異なる x と y に対して, z 方向のループ積算は式 2 により独立に計算することができる. 我々の計算モデルでは, 異なる x と y の組み合わせの計算を個別の演算ユニットに割り当てる. つまり, 元の 3 重ループにもどって考えると, 外側の x と y のループを適切にループアンローリングして, それぞれを別々の演算ユニット割り当てて計算することに相当する. このような計算は, 従来のベクトル演算処理に適した計算である. 従来のベクトル型演算器では, アンロールされた演算を, 内蔵している浮動小数点演算器でパイプライン処理していたが, マルチコアアクセラレータでは, 異なる変数の組は異なる浮動小数点演算ユニットで計算されることとなる.

z 方向のループにおいて, x と y は不変な変数であり, z はループの度に変更される変数である. また, 積算結果を保持する変数もループの度に変更される. その他の変数は, 全てのループにおいて不変であるため定数と考えられる. 我々のコンパイラでは, このような異なる変数の性質を明示的に指定することが特徴である. これにより, 最適化されたコードの自動生成が容易になる. 我々のコンパイラへの入力ソースコードの例を図 1 にしめす. ここで, `LMEM`, `BMEM`, `RMEM`, `CONST` ではじまる行で, 変数の種類を指定している. それぞれ, ローカルメモリ (Local MEMory), ブロードキャストメモリ (Broadcast MEMory), 結果用のローカルメモリ (Result MEMory) と, 定数 (CONSTanat) を定義している.

残り行における演算定義は, 通常のプログラムにおけるメイン関数に相当する. ここでは, ローカルメモリとブロードキャストメモリの変数を入力として, 適宜一時変数に結果を代入し, 最終的に結果用の変数に代入するような演算を定義する. 以下では, 図 1 により GRAPE-DR と RV770 用のコードを生成した.

5.2 GRAPE-DR の場合

図 1 のソースコードより, GRAPE-DR のアセンブリコードを生成すると, 最内ループ部分は 1079 ステップとなった. このうち 34 ステップ分の nop 命令を含む. これは, 主にレジスタ書き込みの競合を回避するために挿入されている.

この計算を GRAPE-DR チップが 1 個搭載された GRAPE-DR model 450 ボード上で実行した. N の大きさを, $256, 512, 1024, 2048$ と変化させると, それぞれの場合で $\sim 2.67, 3.85, 4.80, 5.46$ GFLOPS の DD 演算による演算性能を得た. ここで, 演算性能の計算には, 除算を 10 演算相当と換算し, $35N^3$ の DD 演算を実行したと仮定した. N に対する演算性能に変化が見られるのは, CPU とボード間とのデータ転送のオーバーヘッドのためである.

5.3 RV770 の場合

図 1 のソースコードより, RV770 のアセンブリコードを生成すると, 最内ループ部分は

```
LMEM xx, yy, cnt4;
BMEM x30_1, gw30;
RMEM res;
CONST tt, ramda, fme, fmf, s, one;

zz = x30_1*cnt4;
d = -xx*yy*s-tt*zz*(one-xx-yy-zz)+(xx+yy)*ramda**2 +
        (one-xx-yy-zz)*(one-xx-yy)*fme**2+zz*(one-xx-yy)*fmf**2;

res += gw30/d**2;
```

図 1 式 1 を二重指数型積分法により計算する場合の最内ループの積算例

VLIW 命令換算で 481 ステップとなった. RV770 は VLIW 演算器であるため, 式 1 のように DD 演算が連続する場合, VLIW の演算スロットが効率よく利用されることになる. そのため, 総ステップ数が GRAPE-DR と比べてかなり少なくなった. 実際, 得られたアセンブリを調べると, 多くのステップで 5,4 スロットが利用されており, 3,2,1 スロットのステップはそれぞれ 14,74,2 ステップであった. よって, 計算中おおよそ 81%の時間は, 最大効率で TP が動作することとなり, 4.2 で得た見積もりよりも高い演算性能が見込まれる. また, この時に実際に利用された物理的なレジスタ数は 39 であった. そのため, TP 単位でさらにループアンローリングを適用することで, 演算性能がより向上する見込みがある.

この計算を RV770 が 1 個搭載された Radeon HD4870 ボード (動作周波数は 750 MHz) 上で実行した. N の大きさを, $256, 512, 1024, 2048$ と変化させると, それぞれの場合で $\sim$ 6.43, 7.14, 7.46, 7.57 GFLOPS の DD 演算による演算性能を得た. ここで, 演算性能の計算では GRAPE-DR と同様の仮定をおいた. N に対する演算性能の依存性は, GRAPE-DR の場合と比べると弱い. これは, ボードとの実効データ転送速度が高速なためである. GRAPE-DR model 450 は PCI Express x4 で接続されているのに対して, HD4870 ボードは PCI Express x16 (gen2) で接続されている.

5.4 考　　察

表 4 に, GRAPE-DR と GPU, そして Inte Core i7 920(動作周波数は 2.67 GHz) で同様の計算をおこなった場合の経過時間をまとめてしめす. Core i7 では, N の大きさにかかわらず演算速度は 80 MFLOPS となった. これが 3 章で得られた予測性能よりも多少向上している理由は, 演算数を $35N^3$ で評価したためと考えられる. メニーコアアクセラレータを利用することで, 1 core を利用した CPU の演算性能と比較すると 33 - 94 倍の高速化が可能であることがわかった. 倍精度加算の理論性能で比較すると, GRAPE-DR(380 MHz) は Core i7(2670 MHz) の 37 倍高速であり, RV770 GPU(750 MHz) は 44 倍高速である. ボードとのデータ転送オーバーヘッドにもかかわらず, この比率を上回る演算性能向上率が得られたのは, DD 演算がメニーコアアクセラレータに適した演算であることをしめしている.

	$N = 256$	$N = 512$	$N = 1024$	$N = 2048$	clock
GRAPE-DR	0.21	1.21	7.83	55.1	380
RV770	0.09	0.66	5.03	39.7	750
Core i7	7.39	59.0	472		2670

表 4 式 1 を二重指数積分法によって DD 演算にて計算した場合の経過時間 (秒): clock 周波数の単位は MHz である. 比較のため Core i7 で同等の計算をおこなった場合の経過時間もしめす.

6. ま と め

本論文ではメニーコアアクセラレータにおける, Knuth と Dekker による四倍精度エミュレーションの演算性能評価をおこなった. RV770 GPU と GRAPE-DR はどちらも 200 個以上の倍精度演算器を搭載しているだけでなく, 個々の演算器が利用できる論理的なレジスタ数が通常の CPU よりも圧倒的に多いため, 依存性のある倍精度演算で問題となる演算レイテンシーを実質的に完全に隠蔽できる. そのため, DD 演算のように演算密度が高く依存性のある演算が連続するような計算には, メニーコアアクセラレータが効果的であることが予想される.

実際にファインマンループ積分の二重指数積分法による計算を, DD 演算により RV770 GPU と GRAPE-DR で実行するために, それぞれに対応した演算ライブラリとそれ有効利用するためのコンパイラを作成した. 得られた実演算性能は, Intel Core i7 の 1 core を利用した場合の 30-90 倍に達した. 倍精度演算の性能比を超えて高速化が可能であるということは, メニーコアアクセラレータによる DD 演算が様々な応用で有効に利用できることをしめす. 今後は, 他の DD 演算が必要な処理への適用を目指す. 特に, DD 演算による行列乗算の高速実行が可能となれば, 条件数の非常に大きい行列の処理に有効であることが予想される. また, 本研究の自然な拡張としては, より高精度の演算, 具体的には同様のエミュレーションによる八倍精度演算の実装が考えられる.

Vol.2009-HPC-121 No.39
2009/8/6

参 考 文 献

1) Dekker, T.: A Floating-Point Technique for Extending the Available Precision, *Numerische Mathematik*, Vol.18, pp.224–242 (1971).
2) Hida, Y., Li, X. and Bailey, D.: Algorithm for quad-double precision floating point arithmetic, *Proc. 15th IEEE Symposium on Computer Architecture*, pp. 287–302 (2001).
3) Kistler, M., Gunnels, J., Brokenshire, D. and Benton, B.: Programming the Linpack benchmark for the IBM PowerXCell 8i processor, *Scientific Programming*, Vol.17, pp.43–57 (2009).
4) Knuth, D.: *The Art of Computer Programming vol.2 Seminumerical Algorithms*, Addison Wesley, Reading, Massachusetts, first edition (1998).
5) Makino, J.: Specialized Hardware for Supercomputing, *SciDAC Review*, pp.54–65 (2009).
6) Makino, J., Hiraki, K. and Inaba, M.: GRAPE-DR: 2-Pflops massively-parallel computer with 512-core, 512-Gflops processor chips for scientific computing, *SC07* (2007).
7) Makino, J., Ito, T. and Ebisuzaki, T.: Error Analysis of the GRAPE-1 Special-Purpose N-Body Machine, *Publication of Astronomical Society of Japan*, Vol. 42, pp.717–736 (1990).
8) Sugimoto, D., Chikada, Y., Makino, J., Ito, T., Ebisuzaki, T. and Umemura, M.: A Special-Purpose Computer for Gravitational Many-Body Problems, *Nature*, Vol.345, pp.33–35 (1990).
9) 永井貴博， 吉田仁，黒田久泰，金田康正：SR11000 モデル J2 における 4 倍精度積和演算の高速化，情報処理学会誌:コンピューティングシステム， Vol.48, pp.214–222 (2007).
10) 湯浅富久子，濱口信行：二重指数関数型積分法の素粒子物理学への応用，情報処理学会研究報告 (2008-HPC-114)，pp.31–36 (2008).