Cell Broadband Engine に対する重力多体問題計算の最適化

総合研究大学院大学／国立天文台　富田 賢吾
大阪大学情報科学研究科　山崎 秀輔

SONY PLAYSTATION3 に搭載された Cell Broadband Engine に対し無衝突系重力多体問題計算プログラムを実装し、性能と精度の評価及び他アーキテクチャとの比較を行った。125Gflops という高いピーク性能を達成し、専用計算機 GRAPE-7 及び GPU を用いた CUNBODY-1 コードと比較して良好な価格対性能比及び消費電力対性能比を実現した。無衝突系重力多体問題には十分な精度を達成したが、SPE の単精度浮動小数点演算の丸め処理が切り捨てであることに起因して計算精度が低下していることを示した。このため Cell を用いて一般の科学計算を行うには注意を要し、適切な問題及び手法を選択する必要がある。

1　概要

重力多体問題とは次式に従って重力のみで相互作用する多粒子系の運動を計算する問題である。

$$\mathbf{a_i} = \sum_j -G\frac{m_j}{|r_i - r_j|^3}(\mathbf{r_i} - \mathbf{r_j}) \qquad (1)$$

（添え字 i は力を受ける粒子、j は力を及ぼす粒子を表し、以下それぞれ i 粒子・j 粒子と呼ぶ。$\mathbf{r}, \mathbf{a}$ はそれぞれ粒子の座標と加速度、G は万有引力定数。）このような問題は天文学にしばしば現れ、宇宙の大規模構造から銀河団や銀河、星団や惑星系に至るまで幅広い応用がある。

全粒子数 N に対し式 (1) は $O(N^2)$ の計算量を持つため、粒子数に伴い計算量は急激に増大する。これを取り扱うために東京大学（現在は国立天文台）のグループにより専用計算機 GRAPE[1] が開発されている。近年、汎用プロセッサの高速化に伴い、Intel や AMD の CPU に対して最適化を行った Phantom-GRAPE[2, 3] や、GPU を用いた CUNBODY-1[4, 5] 等の取り組みもなされている。今回我々は Cell Broadband Engine を搭載した SONY PLAYSTATION3 が安価で入手可能であることに着目し、重力多体問題計算プログラムを実装し性能評価を行った。

近年 Intel の Larrabee、AMD の Fusion や Torrenza、NVIDIA の CUDA など、汎用プロセッサと特定用途に特化した高速プロセッサを組み合わせて性能向上を図るという業界の動向があり、数値計算においても今後そのようなプロセッサが重要となる可能性が高い。Cell はヘテロジーニアスマルチコアプロセッサの先進的な実装の一つであり、今回の研究には数値計算におけるその有用性を検証する狙いがある。

2　衝突系と無衝突系 - 精度について

重力多体問題はその性質によって衝突系と無衝突系に分けられる。衝突系は考えている時間スケールで 2 体緩和が系の進化に重要な系で、球状星団や惑星系がその例である。このような問題では粒子間の相互作用を高精度に計算する必要がある。一方宇宙の大規模構造や銀河などの無衝突系では 2 体緩和は重要ではなく、計算精度は低くても良い。GRAPE の奇数シリーズはこのような問題を計算する専用計算機であり、現行の GRAPE-7[6] は 2 粒子間の相互作用で相対誤差 0.3%以下の計算精度がある。

Cell は高い単精度浮動小数点演算性能を持つが、倍精度は単精度の 1/14 と制限されている。また単精度浮動小数点の丸めが切り捨てという点で精度に不安がある。そこで今回は無衝突系の問題向けにプログラムを開発した。

3　プログラムの概要

我々は CellSDK2.1（libspe2.1）を用いてプログラムを開発した。コンパイラは PPE,SPE ともに SDK に含まれる IBM XLC(64bit) を用いた。プログラムは開発用計算機でクロスコンパイルし、PS3 に導入した FedoraCore6 上で動作する。SPE は 6 機使用しプログラムは全て同一である。各 SPE は独立に動作し、PPE と 1 対 1 の通信を行う。今回の計算は通信に対して計算の負荷が大きいためこのような実装でも十分通信の遅延を隠蔽できる。

3.1　PPE プログラム

PPE プログラムは以下のように動作する。

1. SPE プログラムを起動する（initspe）
2. 粒子の初期値を読み込む（readnemo）
3. SPE に j 粒子を転送する (setjp)
4. i 粒子の加速度とポテンシャルエネルギーを求める (calcforce)
5. 粒子の軌道を積分し時間を進める
6. 必要な時間まで 3 に戻って繰り返す
7. 結果を出力する (writenemo)
8. SPE プログラムを停止する (finalize)

initspe ルーチンは 6 つの SPE を起動して全てが待機状態になるのを待つ。

readnemo ルーチンは粒子の質量と座標・速度の初期値を多体問題で標準的に使われる NEMO 形式のファイルから読み込む。データは倍精度で保持する。

setjp ルーチンはまず粒子のデータを単精度に変換し、アラインメントされた領域に連続的に配置する。その後 mailbox を用いて全 SPE に SETJP コマンドを発行し、各 SPE に取得すべき j 粒子の数とそのアドレスを通知する。PPE は SPE が転送終了を通知してくるまで待機する。一度に計算できる j 粒子の数が多い方が使い勝手が良いため、各 SPE で異なる j 粒子を保持する。またこの時粒子数は SPE 間で均等に、かつアラインメントを保つよう 32 個単位に分割する。今回は 1SPE あたり一度に最大 14400 粒子、全体で 86400 粒子を保持する。一度の計算で j 粒子が入りきらない場合には j 粒子を入れ替えて加速度を求め、その結果を合計すれば良い。

calcforce ルーチンは実際に加速度とポテンシャルエネルギーの計算を行う本プログラムの中心部である。このルーチンは setjp ルーチンと同様データを単精度に変換しメモリ上に配置する。次に全ての SPE に CALCFORCE コマンドを発行し、計算すべき i 粒子の数とアドレスを通知する。全ての SPE で i 粒子は共通である。SPE は i 粒子を ISET 個ずつ計算し、結果を随時転送する。PPE は SPE が計算し終わった部分の 6SPE 分の結果を合計していく。この間計算の終わった粒子について次の積分計算を進めることが可能だが、利便性と可読性を考慮してこのような実装とした。PPE は SPE と比べて低速であるが、PPE での積分は $O(N)$ に対し SPE の計算は $O(N^2)$ のため粒子数が多ければ性能への影響は小さい。

得られた加速度を用いて次のタイムステップの粒子の座標を計算する。軌道積分には 2 次精度リープフロッグ法を用いた（低精度な無衝突系の計算によく用いられる）。これを必要な時間まで繰り返し、結果の粒子のデータを writenemo ルーチンを用いて出力する。

finalize ルーチンは全 SPE に FINISH コマンドを発行し、全ての SPE の終了を待つ。

3.2 SPE プログラム

PPE によって起動されると SPE はまず待機状態になり、PPE からの mailbox を介したコマンドによって駆動される。コマンドは j 粒子の取得（SETJP）、加速度とポテンシャルエネルギーの計算（CALCFORCE）、終了（FINISH）の三つがある。

SETJP コマンドを受け取ると SPE はまず転送すべき j 粒子の個数と先頭アドレスを取得し、それに基づいて loadjp ルーチンを実行する。loadjp ルーチンは DMA を用いて j 粒子を取得する。この時粒子数が 4 の倍数でない場合は質量 0 の粒子を入れて 4 の倍数になるようにする。保持する j 粒子は SPE 毎に異なる。転送が終了すると mailbox で PPE に転送終了を通知し、SPE は待機状態に戻る。

CALCFORCE コマンドを受け取ると SPE はまず計算すべき i 粒子の個数と先頭アドレスを取得し、それに基づいて calcforce ルーチンを呼び出す。calcforce ルーチンは DMA を用いて i 粒子を取得し、calcforce_core ルーチンを呼び出して先に取得した j 粒子からの加速度を計算して結果を DMA で PPE に転送する一連の処理を行う。calcforce_core ルーチンの計算の内容と最適化については次節で詳述する。DMA 転送による遅延を隠蔽するためにダブルバッファリングを行っている。i 粒子は全ての SPE で共通である。できるだけ多くの j 粒子を SPE 上に保持できるように、一度に取得する i 粒子の数は転送の遅延を隠蔽できる範囲でできるだけ小さく取っている。

FINISH コマンドを受け取ると SPE は待機状態から脱け出しプログラムを終了する。

4 加速度の計算と最適化

calcforce_core ルーチンでの加速度とポテンシャルエネルギー（加速度の計算過程で得られる）の計算は次のように表現できる（eps はソフトニングパラメータという数値的な発散を抑えるための微小な定数）。

```
for(i=0;i<n;i++)
{
  for(j=0;j<n;j++)
  {
    dx=x[j]-x[i];
    dy=y[j]-y[i];
    dz=z[j]-z[i];
    r2=dx*dx+dy*dy+dz*dz+eps*eps;
    rinv=1.0/sqrt(r2);
    potij=m[j]*rinv;
    rinv2=rinv*rinv;
    atemp=potij*rinv2
    axij=atemp*dx;
    ayij=atemp*dy;
    azij=atemp*dz;
    pot[i]-=potij;
    ax[i]+=axij;
    ay[i]+=ayij;
    az[i]+=azij;
  }
}
```

これを SIMD 化とループアンロールで高速化する。今回は i 粒子を 8 個、j 粒子を 4 個同時に計算することでループ中のストールをほぼなくすことができた。

加速度を計算する際に $1.0/\sqrt{r^2}$ という計算が現れる。この時非常に有用なのは逆数平方根推定命令（spu_rsqrte）である。この命令により約 12bit 精度の逆数平方根の推定値を得ることができ、Newton-Raphson Iteration を一回行うことでほぼ単精度浮動小数点の精度（24bit）の値を得られる。Newton-Raphson Iteration は次の式で表される（x_0 が $1/\sqrt{a}$ の推定値）：

$$x_1 = -\frac{1}{2}x_0(ax_0^2 - 3) \tag{2}$$

5 精度と工夫

数値計算においては各種の数値誤差が問題となるが、今回は丸め誤差が問題となった。SPE の単精度浮動小数点

演算は丸め処理が四捨五入ではなく切り捨てである。そのため、加速度とポテンシャルを足し上げる際に、通常よりも大きな情報落ちが発生してしまう。これを補正するために加算を一度行う毎に仮数部の最下位ビットを強制的に 1 にする演算を入れ、切り捨てと逆のバイアスをかける。誤差の分散は悪化するが、平均の性質は改善する。

足し上げの誤差を補正しても、演算過程で発生する誤差は残る。これを直接的に補正することは困難であるが、この誤差は統計的には力の絶対値を系統的に変化させるように振舞うため、経験的に求めた補正係数をかけて対応している。ただしその値は系によって微妙に異なるため、完全にこの誤差を取り去ることはできなかった。

6 性能と精度の評価

以下に性能評価の結果を掲げる。一相互作用の演算数を 38 浮動小数点演算として性能を求める重力多体問題計算の慣例に倣い、粒子数を N、計算時間を T として $P = N * N * 38/T$(Gflops) で性能を評価した。計算時間 T は j 粒子を転送する setjp と加速度計算を行う calcforce に要した時間であり、PPE での積分の時間は含まない。

表 1: 性能測定結果

粒子数	性能 (Gflops)	粒子数	性能 (Gflops)
1024	84	16384	122
2048	107	32768	124
4096	112	65536	125
8192	120		

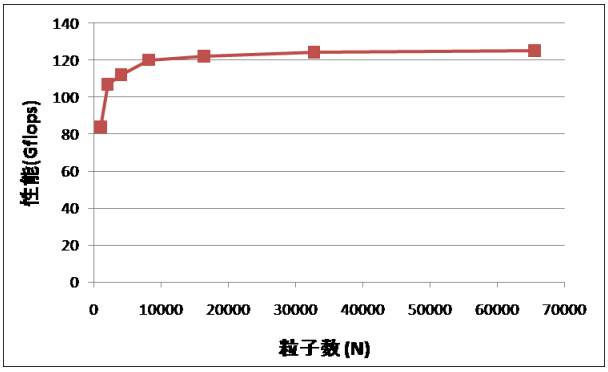

図 1: 性能測定結果

少ない粒子数からピークに近い性能が出ていることがわかる。実際の浮動小数点演算命令数（madd や nmsub を一命令と数える）は一相互作用あたり（誤差補正を含め）21 個であり、69Gflops に相当する。SPE6 機のピーク性能は madd や nmsub を一命令と数えると 76.8Gflops なので、これは SPE の性能のほぼ 90%に相当し十分性能を引き出すことができていると言える。

次に二体間の加速度の誤差と足し上げた加速度の誤差を Intel Pentium M プロセッサの単精度で計算した場合と比較した表を掲げる。ここでの誤差は Pentium で倍精度で計算した値に対する相対的な誤差である。正しい力に対し平行方向と垂直方向に分けてそれぞれ平均と分散を求めた。粒子分布として銀河等のモデルに使われる Plummer モデル（球形）と Exponential Disk モデル（円盤）を用いた。粒子数は二体間の加速度は $N = 256$、足し上げた加速度は $N = 4096$ で計算した。

表 2: 二体間相互作用の誤差

	Pentium 単精度		Cell	
モデル	Plummer	ExpDisk	Plummer	ExpDisk
平行方向平均	2.3e-10	-5.0e-10	8.0e-9	-9.0e-9
平行方向分散	1.0e-7	1.2e-7	2.0e-7	2.3e-7
垂直方向平均	∼0	∼0	∼0	∼0
垂直方向分散	7.6e-8	6.5e-8	1.1e-7	9.4e-8

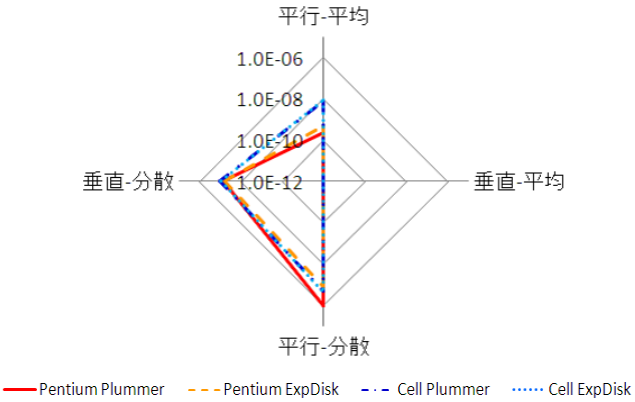

図 2: 二体間相互作用の誤差

表 3: 加速度の誤差

	Pentium 単精度		Cell	
モデル	Plummer	ExpDisk	Plummer	ExpDisk
平行方向平均	5.7e-12	2.6e-11	4.5e-8	-7.3e-9
平行方向分散	4.1e-9	7.1e-9	7.7e-7	1.9e-6
垂直方向平均	-1.1e-11	5.7e-11	-3.9e-9	1.6e-9
垂直方向分散	1.4e-8	1.5e-8	6.9e-7	1.7e-6

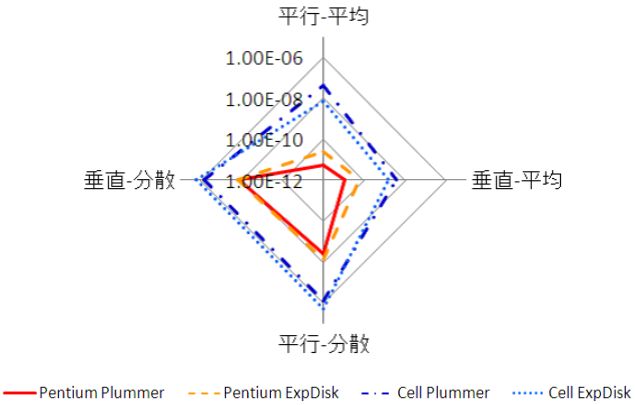

図 3: 加速度の誤差

二体間、足し上げた加速度の両方で、Pentium 単精度と比較して Cell では系統的に誤差が大きくなる。Cell ではバイアスを補正するために補正係数をかけているが、モデルによって誤差の平均値に補正しきれないバイアスが

残っている。このバイアスは十分小さいため今回の要求精度の範囲では大きな問題にはならないが、高精度の演算を行う場合には系統的なエラーとして問題になる。

Cell では系統的に誤差が大きくなるもののその誤差は二体間で最大でも 1.5e-6 であり、GRAPE-7 の相対誤差 0.3%と比べて十分良い。これは無衝突系の問題については十分精度の要件を満たしている。

7 専用計算機及び GPU との比較

以下に我々のコードと GRAPE-7 Model600 と GeForce 8800 シリーズで動作する CUNBODY-1 コードを比較した表を掲げる。ピーク性能は一相互作用あたり 38 浮動小数点演算として評価した性能である。ただし、Cell の消費電力については 90nm 版を 3.2GHz1.0V で駆動した場合の 6SPE の消費電力を示してある [7]。SPE 以外の消費電力は不明なので、実際にはこれよりかなり大きくなると推定される（おそらく数十 W 程度)。

表 4: 他アーキテクチャとの比較

	Cell B.E.	GRAPE-7	CUNBODY-1
最大粒子数	86400	24576	131072
ピーク性能 (Gflops)	125 （実測）	456 （理論値）	653 （実測）
チップ単価	2 万円 （推定）	36 万円 (Model600)	7 万 （GTX）
システム単価 （最小構成）	4 万円 （PS3）	45 万円〜 （ホスト込）	16 万円〜 （ホスト込）
消費電力	18W 本文参照	約 20W	約 200W?
価格対性能比 （円/Gflops）	320	987	245
電力対性能比 （W/Gflops）	0.14	0.04	0.31

表から Cell はピーク性能では劣るものの、最大粒子数とコストパフォーマンスおよび消費電力において GRAPE-7 と GPU の中間的性質を持つことがわかる。

GRAPE-7 や GPU の 2 つと比較したときの Cell の長所は、システム全体のコストパフォーマンスである。他の 2 つはシミュレーション実行時にホストとなる PC を必要とするが、PS3 は一般に PC とは独立に動作する。また近年は計算機の消費電力も問題であり、SPE の高い消費電力対性能比は注目に値する。最近発表された 45nm 版の Cell では消費電力は更に改善している。

8 まとめ

我々は PS3 に搭載された Cell Broadband Engine に対して重力多体問題計算プログラムを実装し、その性能評価を行った。結果無衝突系の問題には十分な精度と良好な価格対性能比及び消費電力対性能比が実現できることがわかった。しかし Cell には一般論として次のような実用上の問題点がある：

- 高精度の演算は低速
- Local Store 容量が小さい
- PPE が遅いため複雑な計算には不向き
- プログラミングが困難
- PS3 以外の実装は高価

これらを解決するために以下のような改善が望まれる：

- 浮動小数点演算の IEEE754 準拠
- 倍精度浮動小数点演算の高速化
- Local Store 容量の拡大
- 汎用コアの性能向上
- ミドルウェアやライブラリの改善

Cell と比べて GPU や汎用プロセッサの性能向上は著しく速い。Cell は現時点ではコストパフォーマンスで優位と言えるが、これらに対抗していくには今後動作周波数の向上や SPE 数の拡張などの性能向上が必要であろう。

今回の計算において特に切り捨て処理による誤差が計算の精度に大きく影響していることが問題となった。科学計算に従来型の Cell を用いる場合、要求精度に十分注意して適切な問題及び手法を選択しなければならない。

Cell はピーク性能は高いもののその性能を引き出すのは容易ではない。IBM による Cell と Opteron のハイブリッドな計算機”Roadrunner”[8] のような構成であれば、複雑な計算に対応しつつ必要な部分を Cell で高速化するという使い方が可能である。倍精度演算性能を強化した新型 Cell であれば、応用範囲が数値計算業界一般に広がる可能性は十分あると考えられる。

あらゆるプログラムを Cell 向けに最適化することは困難であるため、行列演算など一部の共通の計算を Cell で行うアクセラレータ的な使い方が現実的であると思われる。より幅広い用途に Cell を用いるためには使いやすいライブラリやミドルウェアが不可欠である。

参考文献

[1] the GRAPE project, http://grape.mtk.nao.ac.jp/grape/

[2] Phantom-GRAPE Home Page, http://grape.mtk.nao.ac.jp/ nitadori/phantom/

[3] Nitadori, K., Makino, J., & Hut, P., *New Astronomy* **12** (2006), p.169.

[4] CUNBODY-1 library Homepage, http://prograpе.jp/cs/

[5] Hamada, T., & Iitaka, T., ArXiv Astrophysics e-prints, astro-ph/0703100 (2007)

[6] K&F Computing Research, http://www.kfcr.jp/

[7] Takahashi, O., Cottier, S. Sang, H. D., Flachs, B., & Silberman, J., *IEEE Micro* **25** 5 (2005), 10–18

[8] Los Alamos Lab: High-Performance Computing: Roadrunner, http://www.lanl.gov/orgs/hpc/roadrunner/