O-29

GPGPU による超粒子プラズマ・シミュレーション

Super-Particle in Cell Plasma Simulation by GPGPU Method

○内藤孝雄[1]，　相澤正満[2]，　長峰康雄[2]
Takao Naito[1], *Masamitsu Aizawa[2], Yasuo Nagamine[2]

We have studied the self-consistent super particle in cell (PIC) simulation. To reduce the execution time for these simulations, we have applied the so-called GPGPU method. We describe GPGPU method briefly, and report its effectiveness to the PIC simulation.

1.　目的

GPU(Graphics Processing Unit)は、画像処理を専門とする補助演算装置である。GPU による演算を物理数値計算シミュレーションに応用することが目的である。

2.　概要 [1]

近年、3DCG の発展とともに GPU の処理性能が劇的に向上している。

現在の GPU が 3D 描画をするためには、主に光源計算、陰影処理などのシェーダと呼ばれるソフトウェア命令を使用する。この処理は一度の命令で多数のデータを処理しなくてはならない。そのため、GPU は多数の計算コアを積んだ並列計算機となっている。

その並列性が注目され、GPU を汎用演算に使用する(GPGPU;General Purpose computing on GPU)という研究が 2004 年頃に始まったが、グラフィックスアプリケーション開発者の専門知識が必要であったため敷居が高かった。

そこで、GPU を製造している NVIDIA 社は、GPGPU を容易にするために CUDA(Compute Unified Device Architecture)というプログラムの開発環境を公開した。

CUDA の特徴は以下の通りである。

・C 言語を踏襲したプログラム言語
・クロスプラットフォーム
・3D グラフィックスの知識は不要
・無料で利用可能。

プロセッサがデータをやり取りするためには主記憶装置(メモリ)が必要である。CPU 用のメモリと GPU 用のメモリは独立していて、CPU と GPU がメモリを共有することが出来ない。

そのため、CUDA に限らず GPU の処理は、CPU 用のメモリから GPU 用のメモリにデータをコピーして 、CPU が GPU へ処理を指示し、処理が終わると CPU 用のメモリへデータを返すという流れである。(Fig.1)

CUDA は、並列処理を単一の関数で書く。各並列スレッドは、制御を決定するための ID を持つ。その ID によりアクセスするメモリなどを決定することで複数のデータの処理をしている。

また、コア数などの GPU の仕様を意識せずにスレッド数などを指定することも可能である。

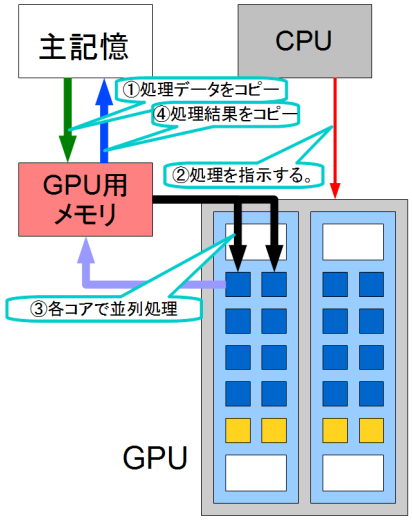

Fig.1 CUDA(GPU)の処理の流れ

数値計算で用いられる倍精度浮動小数点数の処理能力は、前世代のアーキテクチャでは理論上で単精度浮動小数点数の 1/8 の性能になっているが、最新の”Fermi”アーキテクチャで 1/2 の性能となっている。

3.　単純な並列化による GPGPU の性能確認

簡単に並列化できるプログラムの例として、各粒子が独立した単振動の運動方程式を解く。運動方程式は leapfrog 法により差分化する。

１：日大理工・院・量子　２：日大理工・教員・量子

実行環境は CPU は Intel Xeon Processor W5590(4core x2)、GPU は NVIDIA Tesla C1060 である。

CUDA に最適化する際には、各ステップの時間での結果を GPU 上のメモリに記憶し、計算終了時に GPU 上のメモリから CPU 側のメモリに転送する。

以下の表は、単精度、倍精度による、CPU による計算時間、GPGPU による計算時間(計算終了時の GPU から CPU への計算結果の転送を含めるか(A とする)、含めないか(B とする))をまとめたものである。但し、粒子数を N=131072(= 2^{17})、ステップ数を 1024 とする。

Table.1 各粒子が独立した運動方程式の計算時間

	CPU[msec]	GPU[msec]	
		A	B
float	1357.41	332.73	1.48
double	1608.79	653.83	1.47

また、粒子数(N= 2^{x})を変更した時の倍精度計算時の CPU と GPGPU(転送時間含む)による計算時間の変化のグラフを以下に示す。

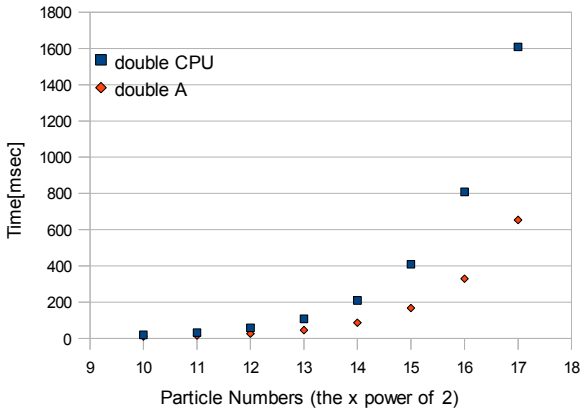

Fig.2 粒子数と計算時間の関係

Table.1 より、CPU と GPU よる計算時間はあまり差が無いように見えるが、GPU の A と B の計算時間の比較や Fig.2 による粒子数の増加と計算時間の関係より、GPU での計算時間の大半は、計算終了時の GPU から CPU への計算結果の転送が大半であることがわかる。このことから、GPU-CPU 間のメモリアクセスは最小限にとどめておいたほうがよいことがわかる。

そして、Tesla C1060 は理論上で単精度浮動小数点数の 1/8 の性能になっているが、Table.1 の GPU の B によると、float 精度も double 精度の場合も計算時間が変わらない。これは、計算時間を演算そのものよりも GPU 内部でのメモリアクセスの方が占有しているからと推測できる。

4.　超粒子プラズマ・シミュレーション

実用的な数値計算シミュレーションの例として、超粒子プラズマ・シミュレーションを行うにあたり、そのベースとして ES1 (Electro Static 1-dimensional program) を用いた。[2] [3] ES1 では、一次元の静電場における非線形な多粒子系のシミュレーションを行う。この際、磁場は考慮していない。また、セルフコンシステントであり、粒子が移動すると周囲の電場が変化するため、多くの計算が必要となる。そこで、並列化することにより計算時間の短縮をめざす。

シミュレーションの主な流れは、システム内に初速度を持った粒子を配置した後、fig.3 に示すようなサイクルに入る。

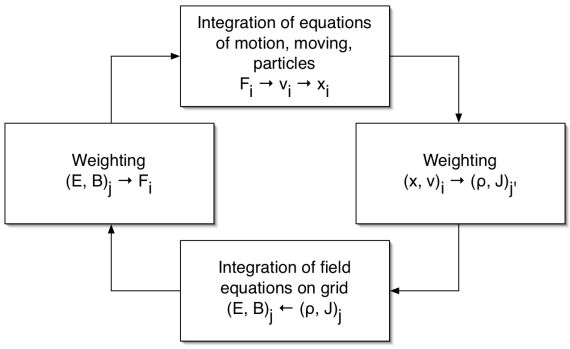

fig.3 ES1 での主要な計算サイクル

このサイクルは、荷電粒子の座標をもとに電荷密度を求めて電場を計算し、この新しい電場をもとに粒子を微小距離移動させて、新しい座標に更新するというものである。

今後、CPU による計算時間と、GPGPU による計算時間を比較して、セルフコンシステントなシステムでも GPGPU が有効かどうかを検証する。

5.　参考文献

[1] 青木尊之・額田彰：「はじめての CUDA プログラミング(工学社)」，2009 年

[2] 石黒静児：「講座 プラズマ計算機シミュレーション入門 II 3.静電粒子シミュレーション」，プラズマ・核融合学会誌，Vol.74，No.6，pp.591-597，1998 年

[3]島田要 2006 年度量子理工学専攻修士論文