O-31

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

○向中野徹[1]，　内藤孝雄[1]，　長峰康雄[2]，　相澤正満[2]

*Toru Mukainakano[1], Takao Naito[1], Yasuo Nagamine[2] ,Masamitsu Aizawa[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。計算量が膨大になる上に、相互作用のある系をセルフコンシステントに扱わなくてはならない、プラズマの粒子シミュレーションを従来困難であった GPU により並列化をし、計算結果や計算時間を CPU と比較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、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) という研究が始まったが、グラフィックス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 は、C 言語を踏襲したプログラム言語であり、クロスプラットフォームに対応しており、3D グラフィックスの知識は不要で、無料で利用可能である、といったような特徴を持つ。

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。

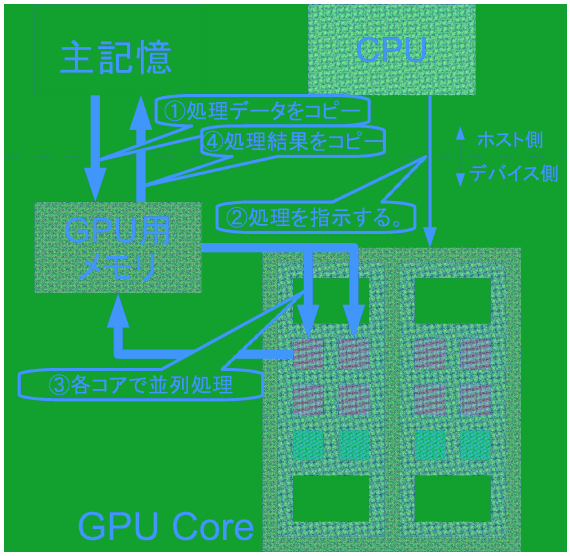

fig.1 CUDA(GPU)の処理の流れ

3.　超粒子プラズマ・シミュレーション

実用的な数値計算の例として、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) 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、毎時間ごとに多くの計算が必要となる。電場の計算と粒子の移動計算を、並列化することにより計算時間の短縮をめざす。

1：日大理工・院・量子　2：日大理工・教員・量子

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.2 に示すようなサイクルに入る。

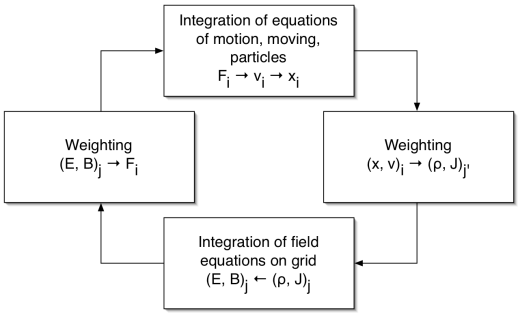

fig.2 ES1 での主要な計算サイクル

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。

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 GPGPU が有効かどうかを検証する。

4.　実行結果

以下の結果で共通している点は、格子点数は M=512、時間刻みは $\tilde{\Delta t}=0.03125$ ("~"は規格化により無次元化された量を表す)とする。初期条件は、粒子の座標は擬似乱数 Xorshift により均等にランダムに並べ、速度は $\tilde{v}=\pm 50$ のどちらかになるような所謂 2 ビーム不安定性と呼ばれるものをとる。また、GPGPU の計算では、CUDA のブロック数を 64 個、スレッド数を 128 個の、合計スレッド数 8192 個で計算した。

CPU と GPGPU の計算結果の比較を v-x 位相図の時間発展で視覚的に簡易的に比較した結果、時刻 $\tilde{t}=320\left(10240\text{steps}\right)$ で違いが見られ始めた(fig.3)。赤色のプロットは初期速度がプラスの粒子、緑色のプロットは初期速度がマイナスの粒子を表している。

次に、CPU と GPGPU の全体の計算結果のグラフを fig.4 に示す。

計算する時間を $\tilde{t}=128\left(4096\text{steps}\right)$ に固定し、粒子数を $N=2^{13}\sim 2^{21}$ で 2 のべき乗で変化させていく。CPU と GPGPU の両方で、N と計算時間が比例関係であり、増加量は CPU の方が大きい。

GPGPU のとき、粒子数が少ないときは比例関係になっていない。理由については以下のように説明できる。電荷密度の算出と粒子の移動の 2 つの工程は、計算時間が粒子数に依存している。結果的に全体の計算時間の大半が 2 つの工程で占めている。しかし、粒子数が少ないときにはそれ以外の工程のほうが全体の計算を占めているために、2 つの工程が全体の計算時間に影響が出なかったために、比例関係になっていなかったからである。

5.　結論・課題

従来困難であった相互作用のあるプラズマシミュレーションを、GPGPU により並列化した。CPU のシングルプロセスプログラムと比べて、かなりの高速化が実現できた。GPU の特殊な機構にシミュレーションモデルに最適化する事ができたことが理由に挙げられる。CPU と GPGPU での計算結果が異なる点関しては今後の課題とする。

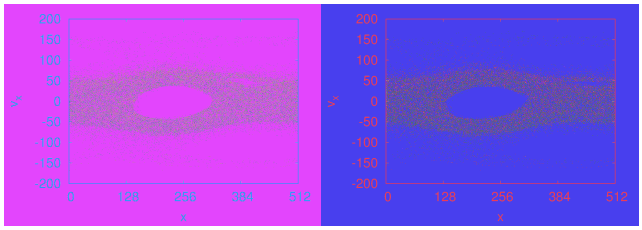

fig.3 CPU と GPGPU の v-x 位相図
($\tilde{t}=128\left(4096\text{steps}\right)$ 右が CPU、左が GPGPU)

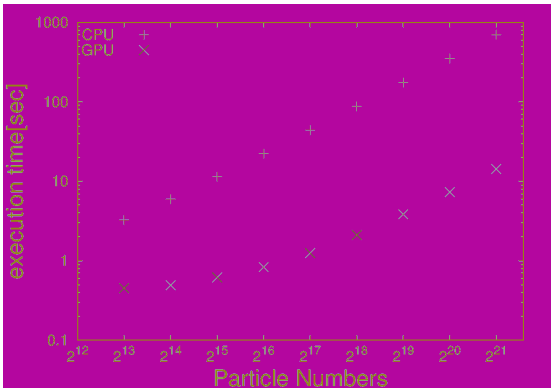

fig.4 CPU と GPGPU の全体の計算時間のグラフ

6.　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