

GPUを用いた分子動力学法と境界要素法の加速

○高橋徹、飯高敏晃、真瀬洋、戎崎俊一
(理研中央研)



GPGPUの勃興

GPU (Graphics Processor Unit)

利用目的：

- 本来、グラフィックスアクセラレータ
- 近年、多目的化

GPGPU (General Purpose computing with **GPU**)

→ 科学技術計算エンジンとしての活用

線形代数 (BLAS)、FFT、流体解析 (LB法)、有限要素法、etc の高速計算

最近のGPUの特徴

- 高い処理能力

演算器（グラフィックパイプライン）数の増加、動作周波数の向上。

- プログラマブル

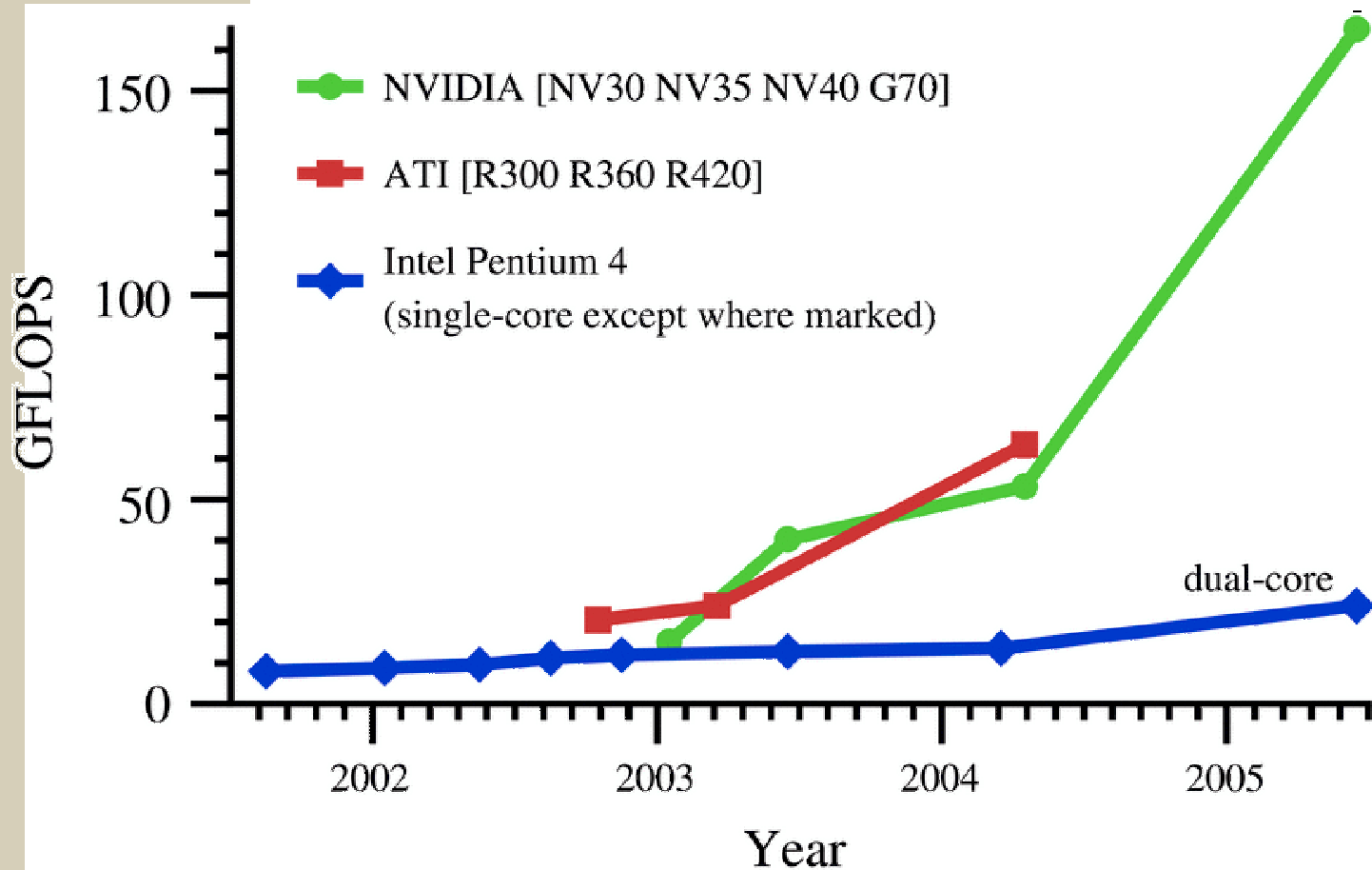
浮動小数点演算のサポート（float 型、ベクトル型）。

- 安価

年間数千億級のゲーム市場、量産効果、開発競争。

GPU ボードは2～7万円・PCとあわせて20万円程度。

GPUとCPUの性能比較



わたしたちの活動

- 間瀬&戎崎：熱輻射の計算
- 飯高：分子動力学法（MD）、量子計算
- 高橋：境界要素法（BEM）



GPGPU

CPU

計算

データ



GPU

計算
+
表示

GPU上の計算過程

SIMD (Single Instruction Multiple Data) 型の計算機

	入力	命令	出力
CG	画像データ (テクスチャ)	視角効果&ポリゴン への貼付	画面への描画
GPGPU	$A[i][j]$	$B[i][j]=\text{alpha}*A[i][j]$	$B[i][j]$

各画素配列成分に関して **パラレル処理** → 高速描画が可能
高速計算の可能性

分子動力学計算

✪ 粒子集団の運動方程式を数値的に解く

$$m_i \frac{d^2 r_i}{dt^2} = F_i$$

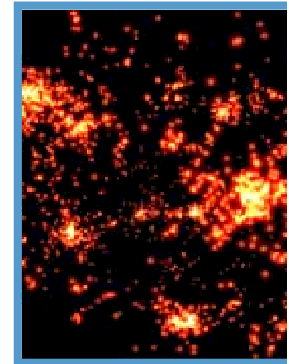
$$F_i = \sum_{j \neq i} f_{ij}$$

$O(N^2)$

$$f_{ij} = n_{ij} \frac{q_i q_j}{4\pi\epsilon_0 r^2}$$

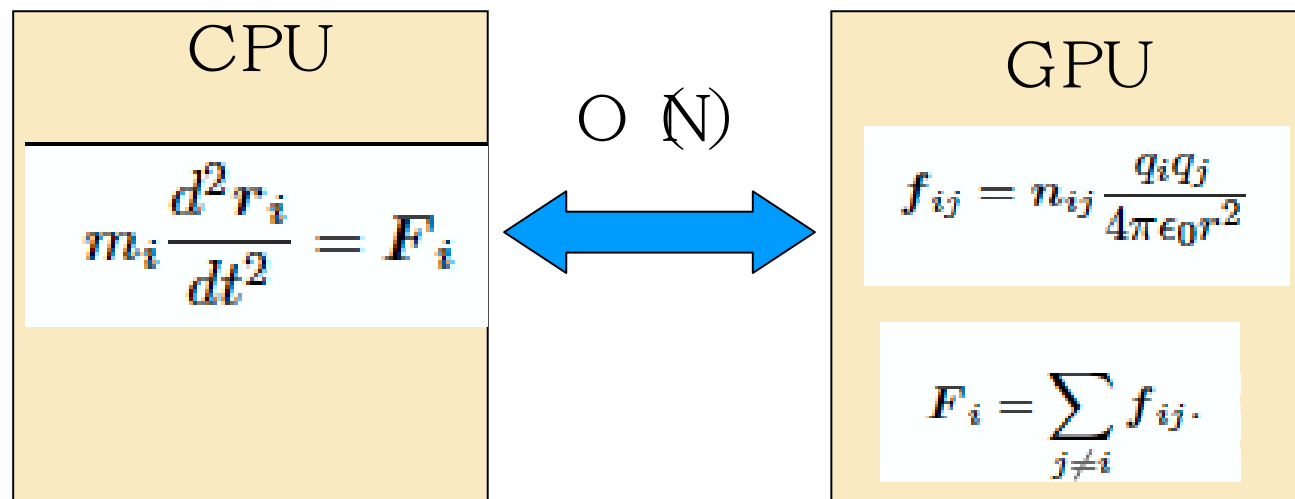
$$f_{ij} = n_{ij} \frac{G m_i m_j}{r^2}$$

$$f_{ij} = n_{ij} \epsilon_{ij} \left(-2 \left(\frac{C_{ij}}{r} \right)^{14} + \left(\frac{C_{ij}}{r} \right)^8 \right)$$



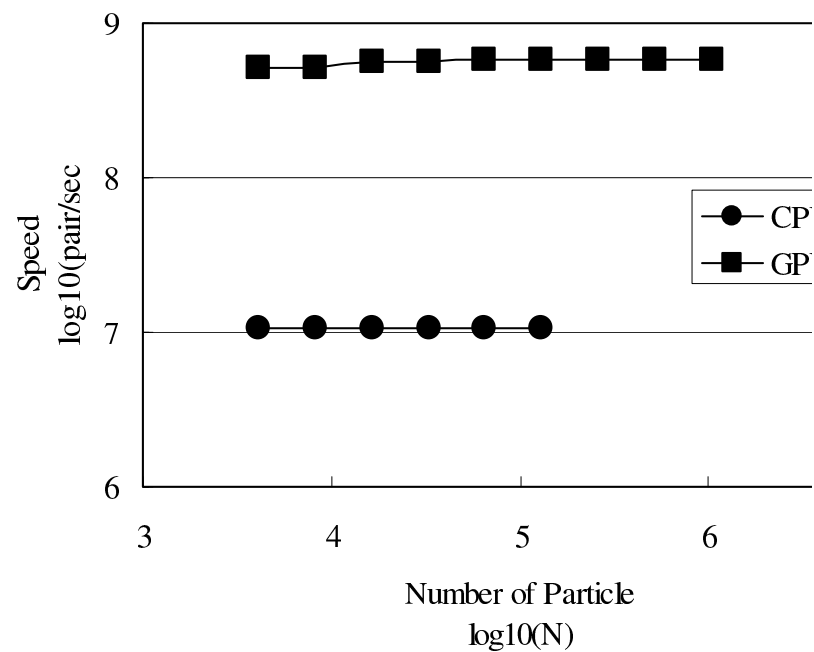
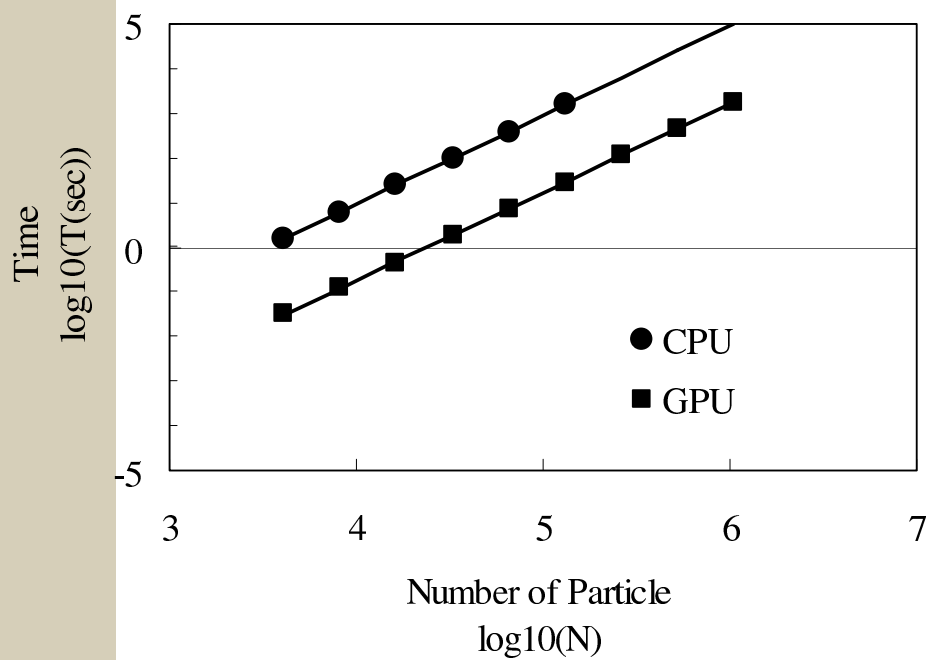
Co-processorとしてのGPU

- ✦ 入出力データ $O(N)$
- ✦ 演算量 $O(N^2)$
- ✦ 先例：(MD) - GRAPE

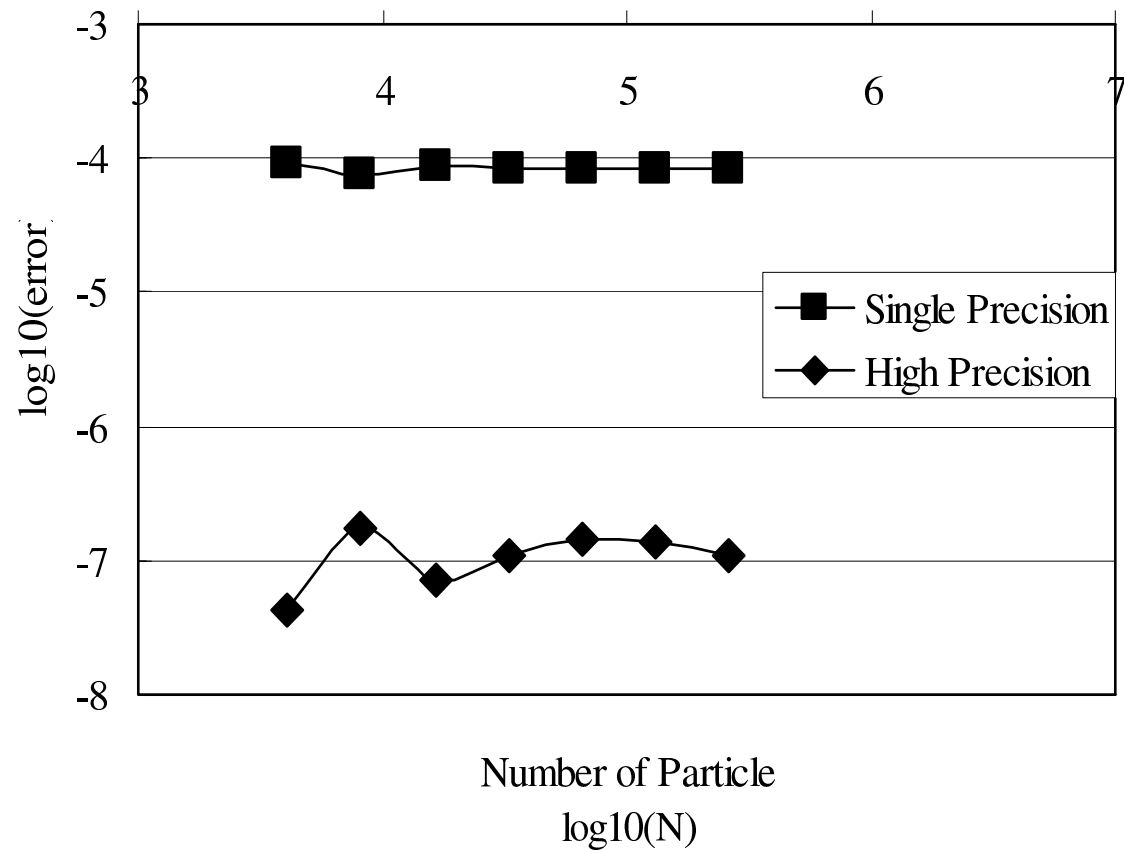


演
○

計算速度 (分子動力学)



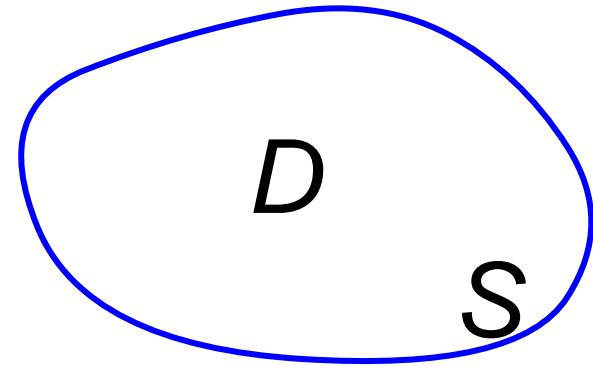
計算精度



境界要素法

境界積分方程式：

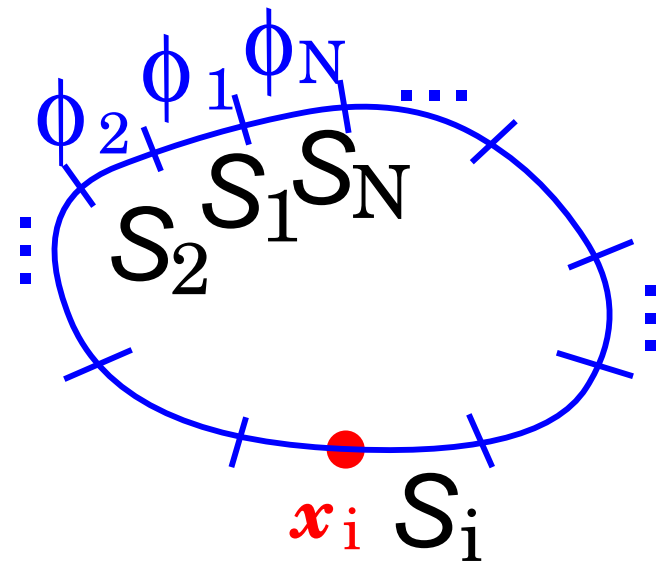
$$\int_S K(\mathbf{x}, \mathbf{y}) \varphi(\mathbf{y}) dS_y = f(\mathbf{x})$$



離散化：

$$\sum_{j=1}^N \left(\int_{S_j} K(\mathbf{x}_i, \mathbf{y}) dS_y \right) \varphi_j = f(\mathbf{x}_i)$$

for $i = 1, \dots, N$



反復解法を用いる場合，左辺の行列ベクトル積がネック — $O(N^2)$

数値計算（2D Laplace 方程式）

問題：一様流中に静止する円柱周りの速度ポテンシャル u を求めよ。

$$\frac{1}{2}u(\boldsymbol{x}) + \int_{\partial D} \frac{\partial \Phi(\boldsymbol{x} - \boldsymbol{y})}{\partial \nu(\boldsymbol{y})} u(\boldsymbol{y}) ds_{\boldsymbol{y}} = u_{\infty}(\boldsymbol{x}) \quad \boldsymbol{x} \in \partial D,$$

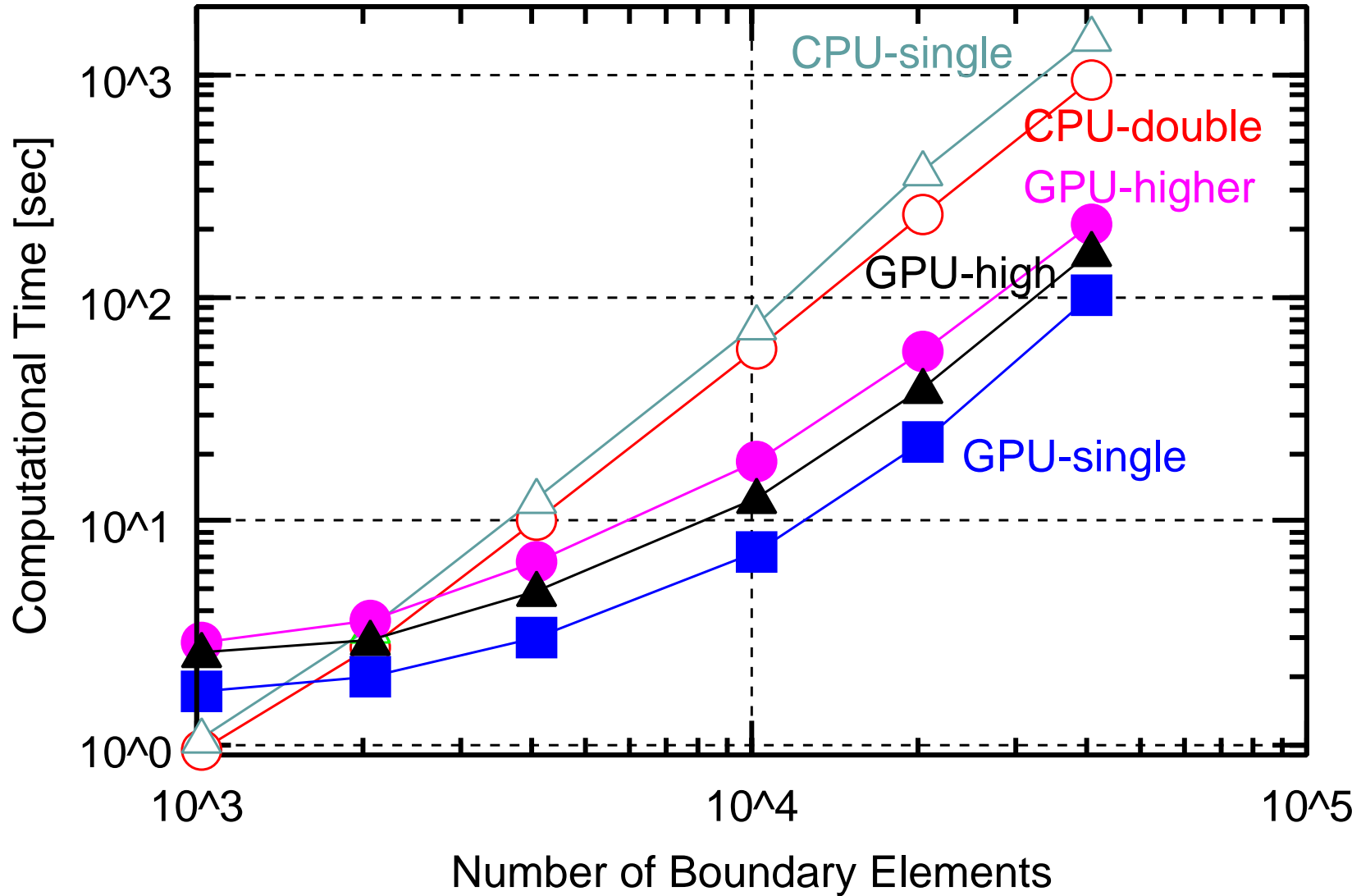
ここに、 $u_{\infty}(\boldsymbol{x}) := x_1$ 。

使用した計算機：

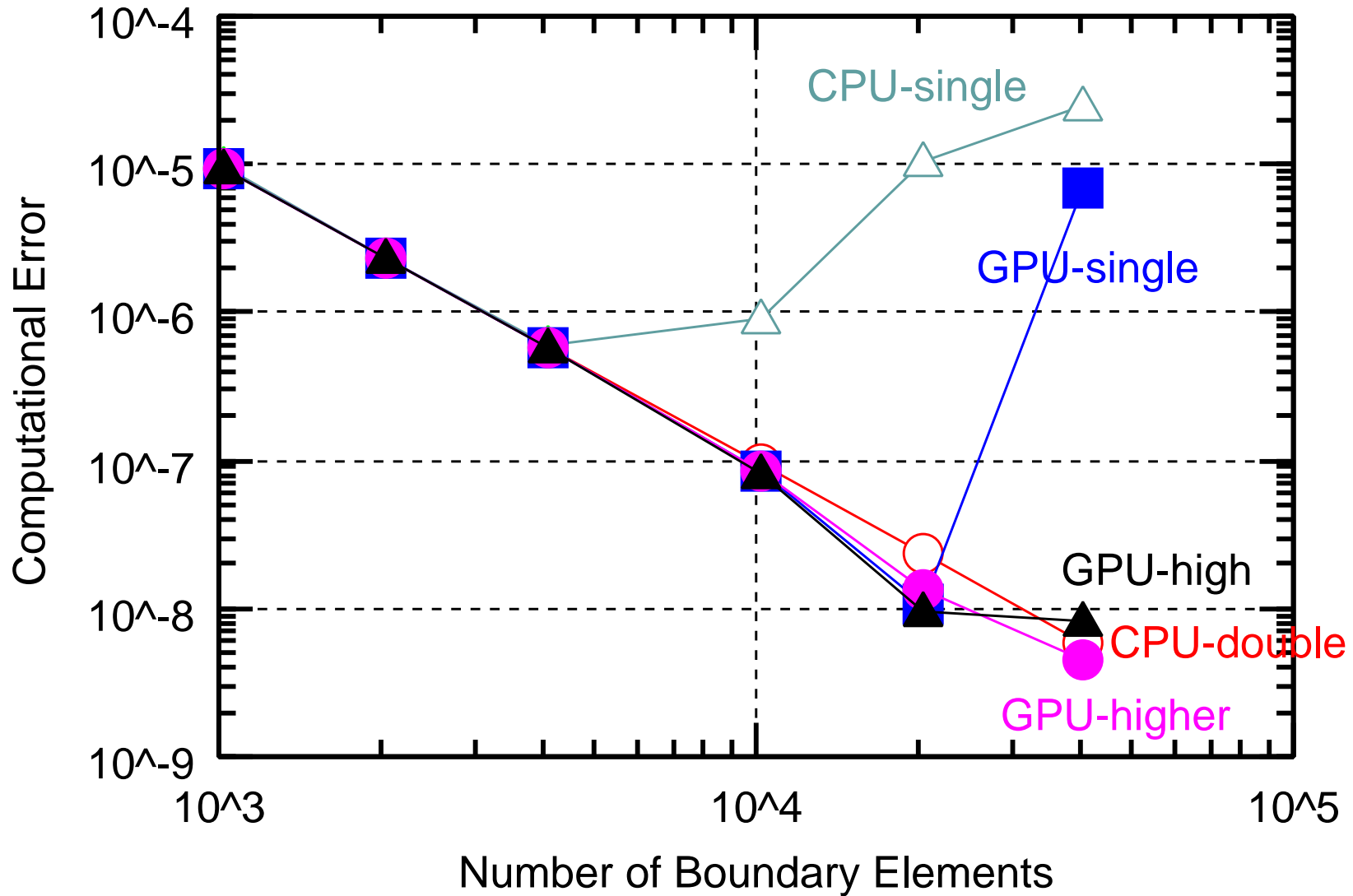
CPU は、Intel Xeon 3.20GHz。

GPU は、NVIDIA GeForce7800GT。

結果：計算時間



結果：計算精度



結論

GPU を利用して、MD と BEM の高速計算に成功した。

今後は、より良い実装法の開発や、高速アルゴリズムとの併用等が課題である。

謝辞

本研究の一部は、理研基礎特研制度と文科省科研費制度若手研究（B）の助成による。記して謝意を表す。

以上

付録

計算速度

Data	GF7800GTX	MDGRAPE-2[3]	MDGRAPE-3[4]
# of pipeline	32	4	20
clock (MHz)	400	100	460
peak (Gflops)	150	16.4	300
peak (Gpair/sec)	6	0.4	9
sustained (Gflops)	15	3.75	NA
sustained (Gpair/sec)	0.6	0.09	NA

- ✪ [3] R. Susukita et al., Phys. Commun. 155, 115 (2003).
- ✪ [4] M. Taiji et al., in Proceedings of SC'03, November 15-21, 2003, Phoenix Arizona, USA.

計算精度

Symbol	Data	GF7800GTX	MDGRAPE-2[3]	MDGRAPE-3[4]
r_i	position	32 bit float	40 bit fixed	40 bit fixed
f_{ij}	force	32 bit float	32 bit float	32 bit float
F_i	total force	32 bit float	64 bit float	80 bit fixed
	others	32 bit float	32 bit float	32 bit float

Table 4. comparison of data format

Cg言語 (GPU用C言語)

クーロン力の和の計算

$$f_{ij} = n_{ij} \frac{q_i q_j}{4\pi\epsilon_0 r^2}$$

```
struct Output {  
    float4 color : COLOR;  
};
```

```
Output main(  
    float2 i : TEXCOORD0,  
    uniform int mx,  
    uniform int my,  
    uniform samplerRECT texture):COLOR
```

```
{  
    Output OUT;  
    float r,rr,k,l;  
    float2 j;  
    float3 ri,rj,rij,force;  
    const float eps2=1e-6;  
    force=0;  
    ri = texRECT(texture,i);  
    for(l=0.5; l <my; ++l){  
        for(k=0.5; k <mx; ++k){  
            j=float2(k,l);  
            rj = texRECT(texture,j);  
            rij= rj-ri;  
            rr = eps2+dot(rij,rij);  
            r = sqrt(rr);  
            force += rij/(r*rr) ;  
        }  
    }  
    OUT.color.xyz=force;  
    return OUT;  
}
```

多体問題専用計算機

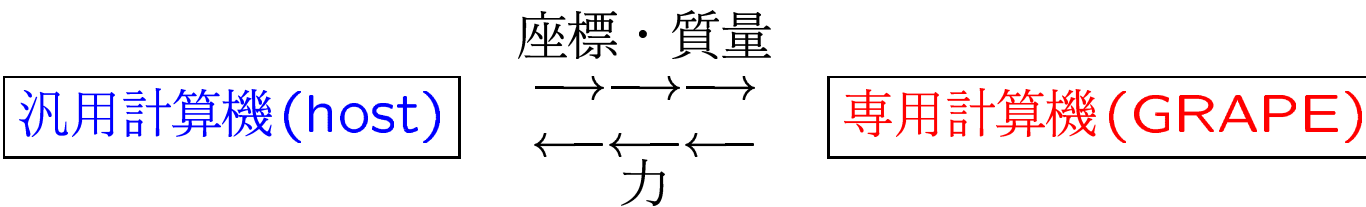
GRAPE ファミリー ('80年代~):

重力多体問題専用 GRAvityPipE (東大)、

分子動力学専用 MDGRAPE (東大 & 理研)、

後継機 MDGRAPE-3、GRAPE-DR、GRAPE-8、、、

$O(N^2)$ アルゴリズム用に開発: 粒子間力 $\mathbf{F}_i = \sum_{j=1}^N \mathbf{f}_{ij}$



高速アルゴリズムとの併用: 近傍計算はそのまま。遠方計算は?

J.Makino, Treecode with a special-purpose processor, 1991.