

# Hardware Acceleration for Boundary Element Methods

Toru Takahashi

Mech Sci Eng, Nagoya Univ / Comp Astro Phys, RIKEN



GPU Meeting (HaRiken08)

Harvard Univ, MA, USA  
Mar 10–11, 2008

# § 1. Introduction

## Points

---

- What is BEM? — It's a classical many-body problem.
- Any issue? — BEM requires **high computational cost**.
- How to overcome? — Two solutions.
  1. Apply (approximate) fast algorithm (FMM, B&H, PWTD,...)
  2. Vectorise and/or parallelise
    - ◇ General purpose computers (MPI, OpenMP, SSE,...)
    - ◇ Special purpose computers (**MDGRAPE**)
    - ◇ Quasi general purpose computers (**GPU**, PS3,...)

**How to implement?**

# Outline

---

§ 1. Introduction

§ 2. Boundary Element Method

§ 3. Hardware Accelerating Technique

§ 4. Implementation of BEM code

MDGRAPE-2, GeForce7800, GeForce8800

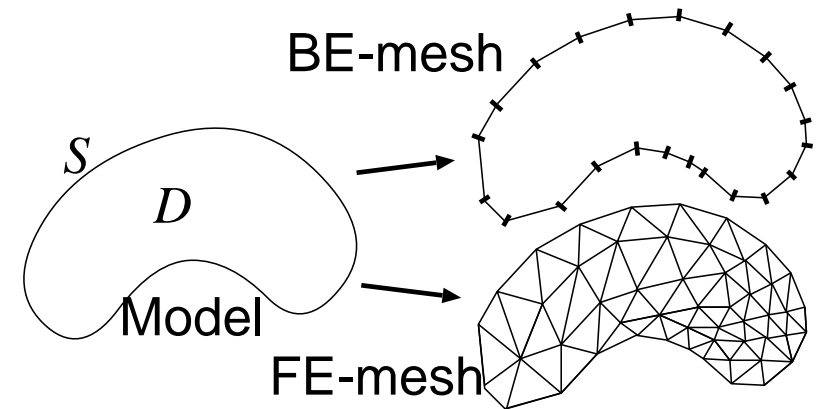
§ 5. Conclusion

# § 2. Boundary Element Method

## Features

---

- Numerical method to solve (initial-)boundary value problems
- Discretise only the boundary (surface) of the domain.
- Advantages
  - ◇ Easy to generate a mesh
  - ◇ Applicable to external problems
  - ◇ High accurate
- Disadvantages
  - ◇ Unsuitable for nonlinear/inhomogeneous problems
  - ◇ Requires **high computational cost**



# Formulation

- Focus on BEM for Helmholtz eq in 3D

- Statement of problem:

For a given wavenumber  $k$ , solve  $u$  from

$$\Delta u(\mathbf{x}) + k^2 u(\mathbf{x}) = 0 \quad \text{for } \mathbf{x} \in D \subset \mathbb{R}^3 \quad (1)$$

subject to boundary (and radiation) conditions.

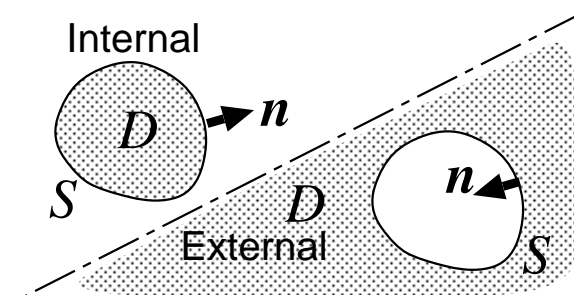
- Fundamental solution (Green's function in free space):

$$\Gamma_k(\mathbf{r}) := \frac{e^{ik|\mathbf{r}|}}{4\pi|\mathbf{r}|} \quad \text{for } \mathbf{r} \in \mathbb{R}^3 \quad (2)$$

- Boundary Integral Equation:

$$\frac{1}{2}u(\mathbf{x}) = \int_S \left( \underbrace{\Gamma_k(\mathbf{x} - \mathbf{y})q(\mathbf{y})}_{\text{single-layer}} - \underbrace{\frac{\partial \Gamma_k}{\partial n_y}(\mathbf{x}, \mathbf{y})u(\mathbf{y})}_{\text{double-layer}} \right) dS_y \quad \text{for } \mathbf{x} \in S, \quad (3)$$

where  $q := \partial u / \partial n$  (normal-flux).

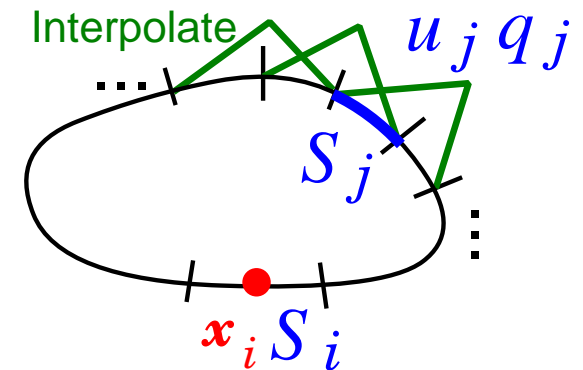


# Formulation (cont'd)

- Discretisation:

- ◇ Divide  $S$  into boundary elements  $S_j$
- ◇ Interpolate  $u$  and  $q$  with local basis functions  $\phi$ ; Introduce  $N$  samples of  $u_j$  and  $q_j$  on  $S$
- ◇ Let BIE be satisfied at  $N$  points  $\mathbf{x}_i$  on  $S$  (collocation points)

$$\begin{aligned} \sum_{j=1}^N \left( \int_{S_j} \frac{\partial \Gamma_k}{\partial n_y}(\mathbf{x}_i, \mathbf{y}) \phi(\mathbf{y}) dS_y \right) u_j \\ = \sum_{j=1}^N \left( \int_{S_j} \Gamma_k(\mathbf{x}_i - \mathbf{y}) \phi(\mathbf{y}) dS_y \right) q_j \quad (4) \end{aligned}$$



- Linear equations:

$$\sum_{j=1}^N A_{ij} z_j = b_i \quad \text{or} \quad \mathbf{A} \mathbf{z} = \mathbf{b} \quad \text{for } i = 1, \dots, N \quad (5)$$

where

- ◇  $A_{ij}$  stands for the influence to  $\mathbf{x}_i$  (receiver) from  $S_j$  (source).
- ◇  $\mathbf{A}$  is dense and asymmetric in general.

# Formulation (cont'd)

---

- Solution:
  - ◇ Direct approach:  $O(N^3)$   $\longrightarrow$  unrealistic.
  - ◇ Iterative approach:  $O(N^2 N_{\text{iter}})$ , where  $N_{\text{iter}}$  is number of iterations.
    - (Preconditioning is an open issue.)
    - Scheme
      1. Give  $z_0$  to  $z$
      2. If residual  $\|\mathbf{A}z - \mathbf{b}\| < \epsilon$ , then  $z$  is solution.
      3. Otherwise, update  $z$  and go to 2.
    - “matvec” is the most time-consuming part —  $O(N^2)$
    - How to accelerate matvec? (besides fast algorithm)

# § 3. Hardware Accelerating Technique

## Objective

---

Make MDGRAPE/GPU to calculate matvec or discretised layer-potential.

## Model

---

- Suppose a layer-potential assuming

$$\int_S K(\mathbf{x}, \mathbf{y}) \varphi(\mathbf{y}) dS_y, \quad (6)$$

where  $S$  is a boundary,  $K$  is a kernel function,  $\varphi$  is a density,  $\mathbf{x}$  is a collocation point on  $S$ .

# How to formulate?

1. Discretise  $S$  into  $N$  piecewise-constant boundary elements,

$$\int_S K(\mathbf{x}_i, \mathbf{y}) \varphi(\mathbf{y}) dS_y \approx \sum_{j=1}^N \varphi_j \int_{S_j} K(\mathbf{x}_i, \mathbf{y}) dS_y \quad (7)$$

2. Apply  $G$ -points Gaussian quadrature formula

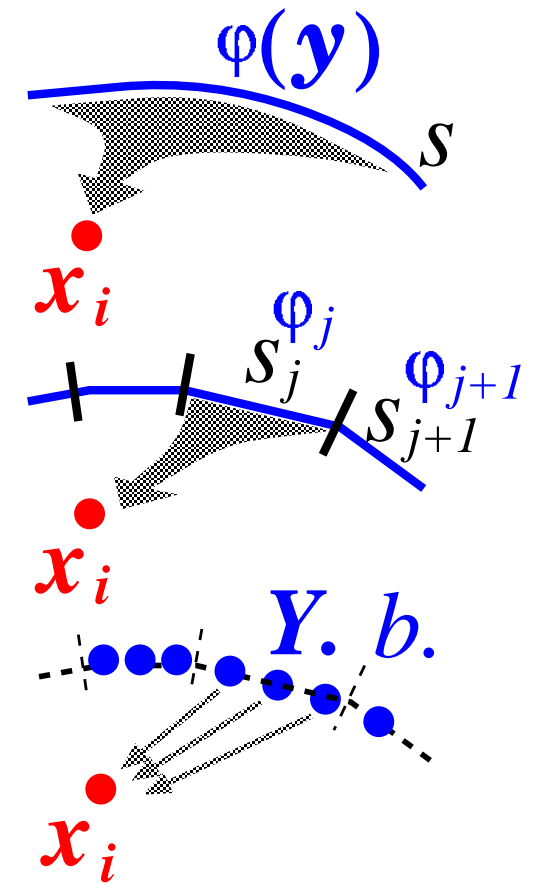
$$\approx \sum_{j=1}^N \varphi_j \sum_{k=1}^G w_k K(\mathbf{x}_i, \mathbf{y}_{j,k}) a_j \quad (8)$$

3. Combine two summations into one

$$= \sum_{j=1}^{NG} K(\mathbf{x}_i, \mathbf{Y}_j) b_j \quad \text{for } i = 1, \dots, N, \quad (9)$$

where

$$\begin{aligned} b_{G(j-1)+k} &:= \varphi_j w_k a_j \\ \mathbf{Y}_{G(j-1)+k} &:= \mathbf{y}_{j,k} \end{aligned} \quad (1 \leq j \leq N; 1 \leq k \leq G)$$



# Analogy to many-body problems

---

- Obviously,

$$\Phi(\mathbf{x}_i) = \sum_{j=1}^{NG} K(\mathbf{x}_i, \mathbf{Y}_j) b_j \quad \text{for } i = 1, \dots, N$$

forms a sort of many-body problems.

- ◇ Contains  $N$  receivers and  $NG$  sources.
  - ◇ Source strength is  $b_j$ .
  - ◇ Remark: Depending on problems, a quantity  $(\Phi, b_j, K)$  is a {real, complex} {scalar, vector}.
- Analogy allows to apply most of techniques in many-body to BEM.

*(The same occurs in vice versa; FMM was firstly invented for integral equations.)*

# Examples of “many-body expression”

- Helmholtz in 3D:

$$\int_S \Gamma_k(\mathbf{x}_i - \mathbf{y}) q(\mathbf{y}) dS_y \approx \sum_{j=1}^{NG} \frac{e^{ik|\mathbf{x}_i - \mathbf{Y}_j|}}{4\pi|\mathbf{x}_i - \mathbf{Y}_j|} Q_j$$

$$\int_S \frac{\partial \Gamma_k}{\partial n_y}(\mathbf{x}_i - \mathbf{y}) q(\mathbf{y}) dS_y \approx \sum_{j=1}^{NG} \frac{(1 - ik|\mathbf{x}_i - \mathbf{Y}_j|) e^{ik|\mathbf{x}_i - \mathbf{Y}_j|}}{4\pi|\mathbf{x}_i - \mathbf{Y}_j|} (\mathbf{x}_i - \mathbf{Y}_j) \cdot \mathbf{U}_j$$

- Maxwell in 3D:

$$\int_S \sqrt{\frac{\mu}{\epsilon}} \left[ -ik\mathbf{J}(\mathbf{y}) + \frac{1}{ik} \nabla(\nabla_{S_y} \cdot \mathbf{J}(\mathbf{y})) \right] \Gamma_k(\mathbf{x} - \mathbf{y}) dS_y$$

- Laplace in 3D:

$$\int_S \Gamma_0(\mathbf{x} - \mathbf{y}) q(\mathbf{y}) dS_y \quad : \text{Static limit } (k \rightarrow 0) \text{ of Helmholtz.}$$

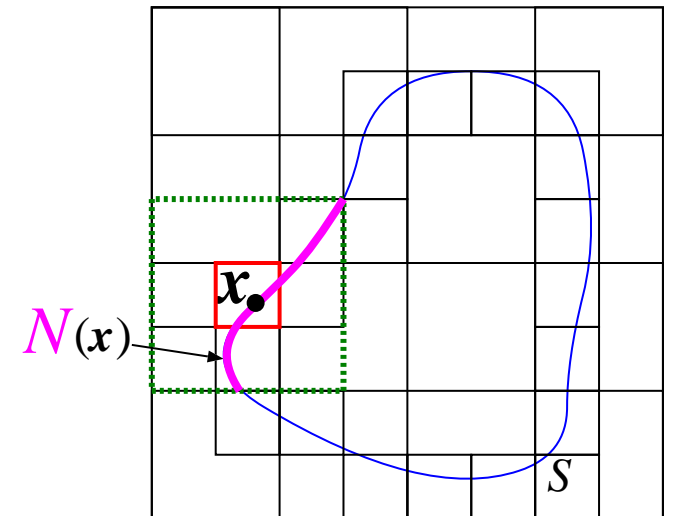
- Elastostatics in 3D:

$$\int_S G_{ij}(\mathbf{x} - \mathbf{y}) t_j(\mathbf{y}) dS_y = C_1 \int_S \Gamma_0(\mathbf{x} - \mathbf{y}) t_i dS_y + C_2 \int_S \frac{\partial \Gamma_0(\mathbf{x} - \mathbf{y})}{\partial n_y} \tau_j dS_y$$

# Correction

- $\Gamma_k$  and derivatives have a **singularity** of  $|\mathbf{x} - \mathbf{y}|^{-1, \dots}$  at  $\mathbf{x} = \mathbf{y}$ .
  - ◇ Cannot avoid by relaxation  $\epsilon^2$ ; Must evaluate properly.
  - ◇ Should care for **near-singularity** around  $\mathbf{x} = \mathbf{y}$ , too.
- How to correct  $\Phi$ ?

$$\begin{aligned} & \sum_{j=1}^{NG} K(\mathbf{x}_i, \mathbf{Y}_j) b_j \\ \Rightarrow & \sum_{\substack{j=1 \\ \mathbf{x}_i \neq \mathbf{Y}_j}}^{NG} K(\mathbf{x}_i, \mathbf{Y}_j) b_j \\ & - \sum_{\substack{\mathbf{Y}_j \in \mathcal{N}(\mathbf{x}_i) \\ \mathbf{x}_i \neq \mathbf{Y}_j}} K(\mathbf{x}_i, \mathbf{Y}_j) b_j + \int_{\mathcal{N}(\mathbf{x}_i)} K(\mathbf{x}_i, \mathbf{y}) \varphi(\mathbf{y}) dS_y. \end{aligned}$$



Define neighbouring boundary  $\mathcal{N}$  by oct-tree.

- **Host** computes correcting terms —  $O(N)$
- **Slave** computes the first term —  $O(N^2)$

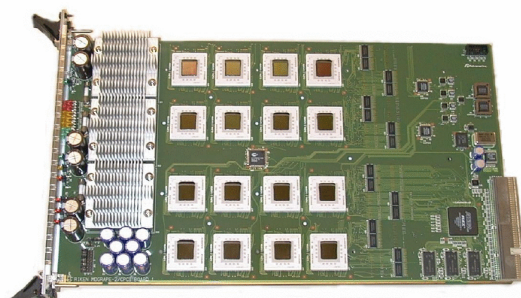
## § 4. Implementation of BEM Code

1. MDGRAPE-2
2. GeForce7800 (Cg)
3. GeForce8800 (CUDA)

# MDGRAPE-2 — Résumé

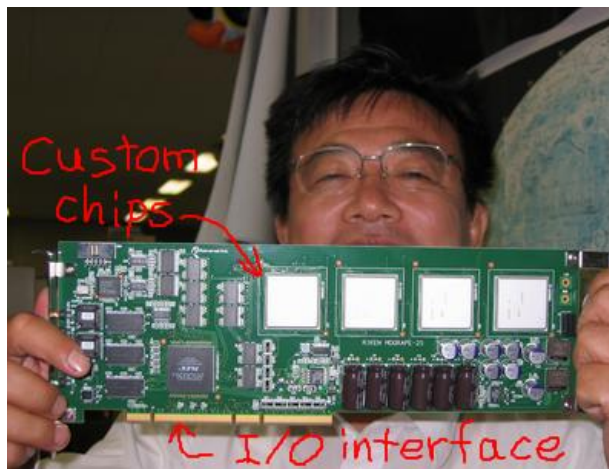
- Special-purpose computer for MD simulations (Narumi et al)
- Custom LSI to compute any type of two-body force.

MDGRAPE-2  
(Compact PCI)



128Gflops

MDGRAPE-2s  
(PCI)



64Gflops

MDGRAPE-3  
(PCI-X)



330Gflops

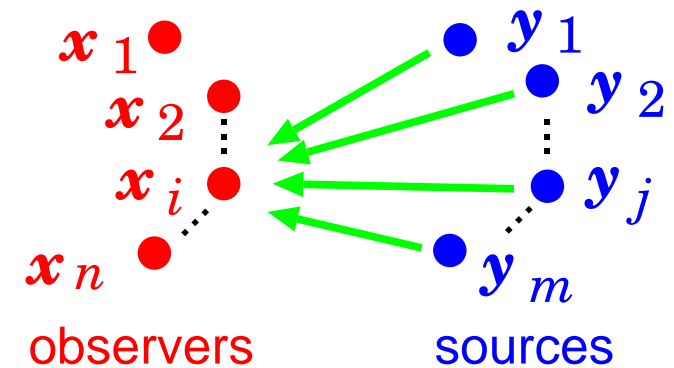
- A variant of GRAPE; A successor is MDGRAPE-3 (Taiji et al)

# Function

- $n$  receivers (i-particles): positions  $\mathbf{x}_I$  ( $I = 1, \dots, n$ )
- $m$  sources (j-particles): positions  $\mathbf{y}_J$  and charges  $b_J$  ( $J = 1, \dots, m$ )
- $\alpha$ : scaling parameter,  $g$ : a real-valued function of a real variable.
- All of them are given, MD2 computes

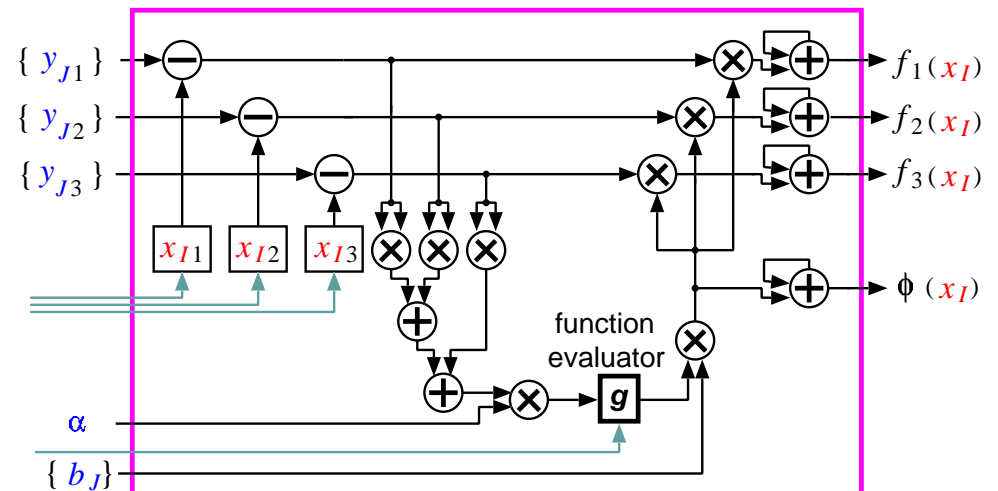
$$\phi(\mathbf{x}_i) := \sum_{j=1}^m b_j g(\alpha |\mathbf{x}_i - \mathbf{y}_j|^2)$$

$$\mathbf{f}(\mathbf{x}_i) := \sum_{j=1}^m b_j g(\alpha |\mathbf{x}_i - \mathbf{y}_j|^2) (\mathbf{x}_i - \mathbf{y}_j).$$



- A vector-parallel computer:

- ◇ vectorisation w.r.t.  $j$   
A hardwired pipeline
- ◇ parallelisation w.r.t.  $i$   
Multiple pipelines



MDGRAPE-2 pipeline

# How to set “molecules” in Helmholtz?

- Single-layer potential of 3D Helmholtz:
  - ◇ Separate any complex quantity into its real and imaginary parts

$$\begin{aligned}
 \int_S \Gamma_k(\mathbf{x}_i - \mathbf{y}) q(\mathbf{y}) dS_y &\simeq \sum_{j=1}^{NG} \frac{e^{ik|\mathbf{x}_i - \mathbf{Y}_j|}}{k|\mathbf{x}_i - \mathbf{Y}_j|} Q_j \\
 &= \sum_{j=1}^{NG} \frac{\cos(k|\mathbf{x}_i - \mathbf{Y}_j|)}{k|\mathbf{x}_i - \mathbf{Y}_j|} \Re[Q_j] - \sum_{j=1}^{NG} \frac{\sin(k|\mathbf{x}_i - \mathbf{Y}_j|)}{k|\mathbf{x}_i - \mathbf{Y}_j|} \Im[Q_j] \\
 &\quad + i \sum_{j=1}^{NG} \frac{\sin(k|\mathbf{x}_i - \mathbf{Y}_j|)}{k|\mathbf{x}_i - \mathbf{Y}_j|} \Re[Q_j] + i \sum_{j=1}^{NG} \frac{\cos(k|\mathbf{x}_i - \mathbf{Y}_j|)}{k|\mathbf{x}_i - \mathbf{Y}_j|} \Im[Q_j]
 \end{aligned}$$

- ◇ Each term corresponds to  $\sum_{j=1}^m b_j g(\alpha|\mathbf{x}_i - \mathbf{y}_j|^2)$  when  $b_j = \Re[Q_j]$  or  $\Im[Q_j]$ ,  $\alpha = k^2$ , and  $g(x) = \cos(x^{1/2})/x^{1/2}$  or  $\sin(x^{1/2})/x^{1/2}$ .

- ◇ Drive MD2 in potential-mode term by term; 4 drives in total.
- Double-layer potential:
  - 12 drives in force-mode

# Benchmark

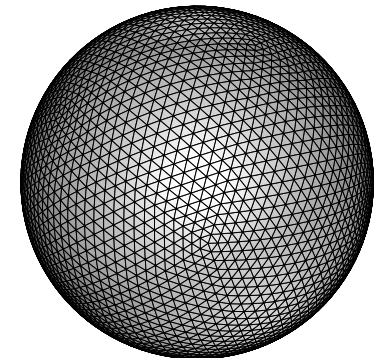
---

- Scattering problem: Solve total field  $u$  from

$$\frac{1}{2}u(\mathbf{x}) = u^\infty(\mathbf{x}) - \int_S \frac{\partial \Gamma}{\partial n_y}(\mathbf{x}, \mathbf{y})u(\mathbf{y})dS_y \quad \text{for } \mathbf{x} \in S$$

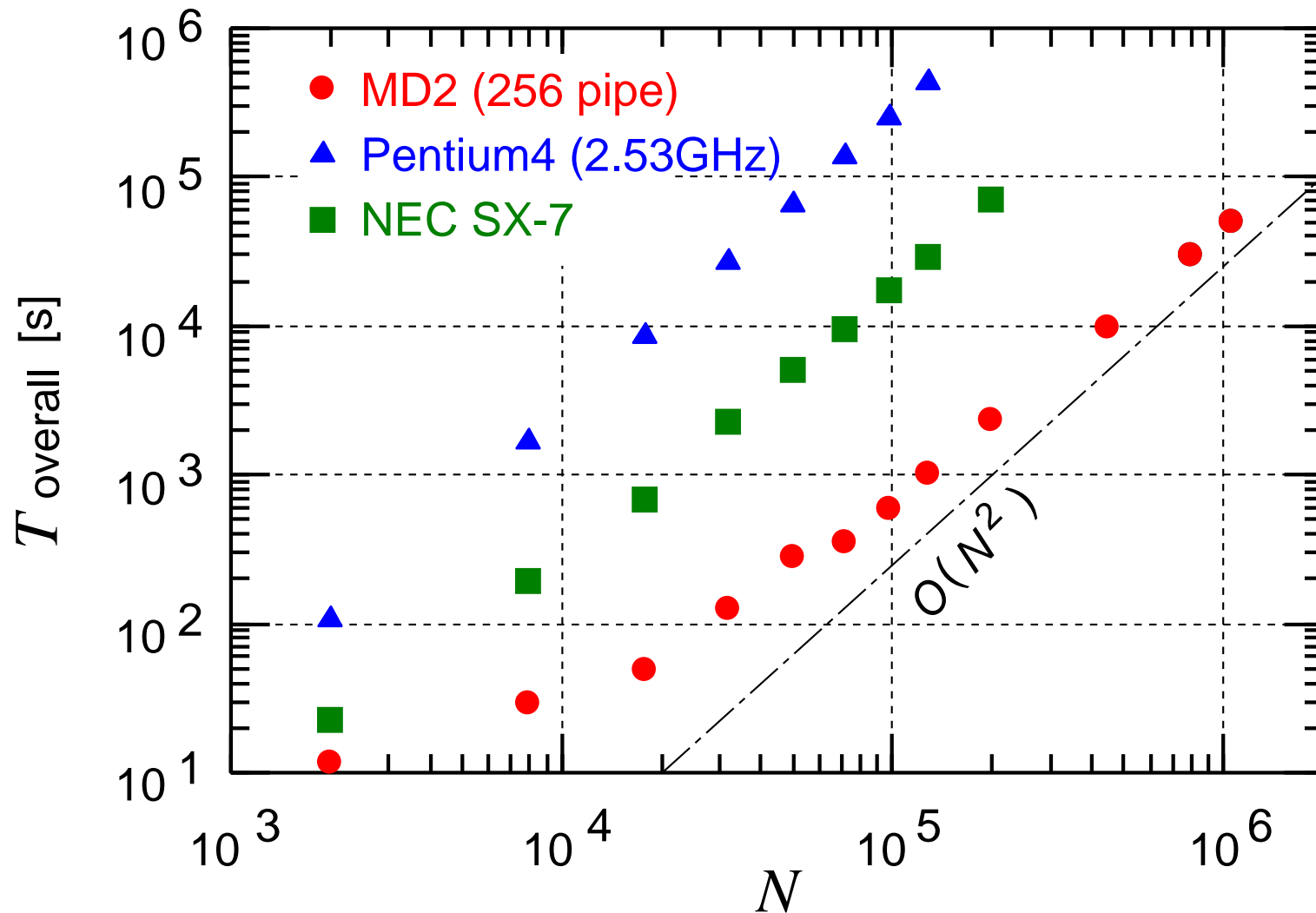
where incident field  $u^\infty = e^{ikx_3}$ .

- Exact solution is available.
- Numerical settings:
  - ◇ Wavenumber  $k = 5$
  - ◇ Elements  $N = 2,000 - 1,058,000$
  - ◇ Gaussian quadrature:  $G = 7$ 
    - 1M vs 7M particles at most
  - ◇ Preconditioned GMRES
  - ◇ (# of elements in  $\mathcal{N}$ )  $\lesssim 100$



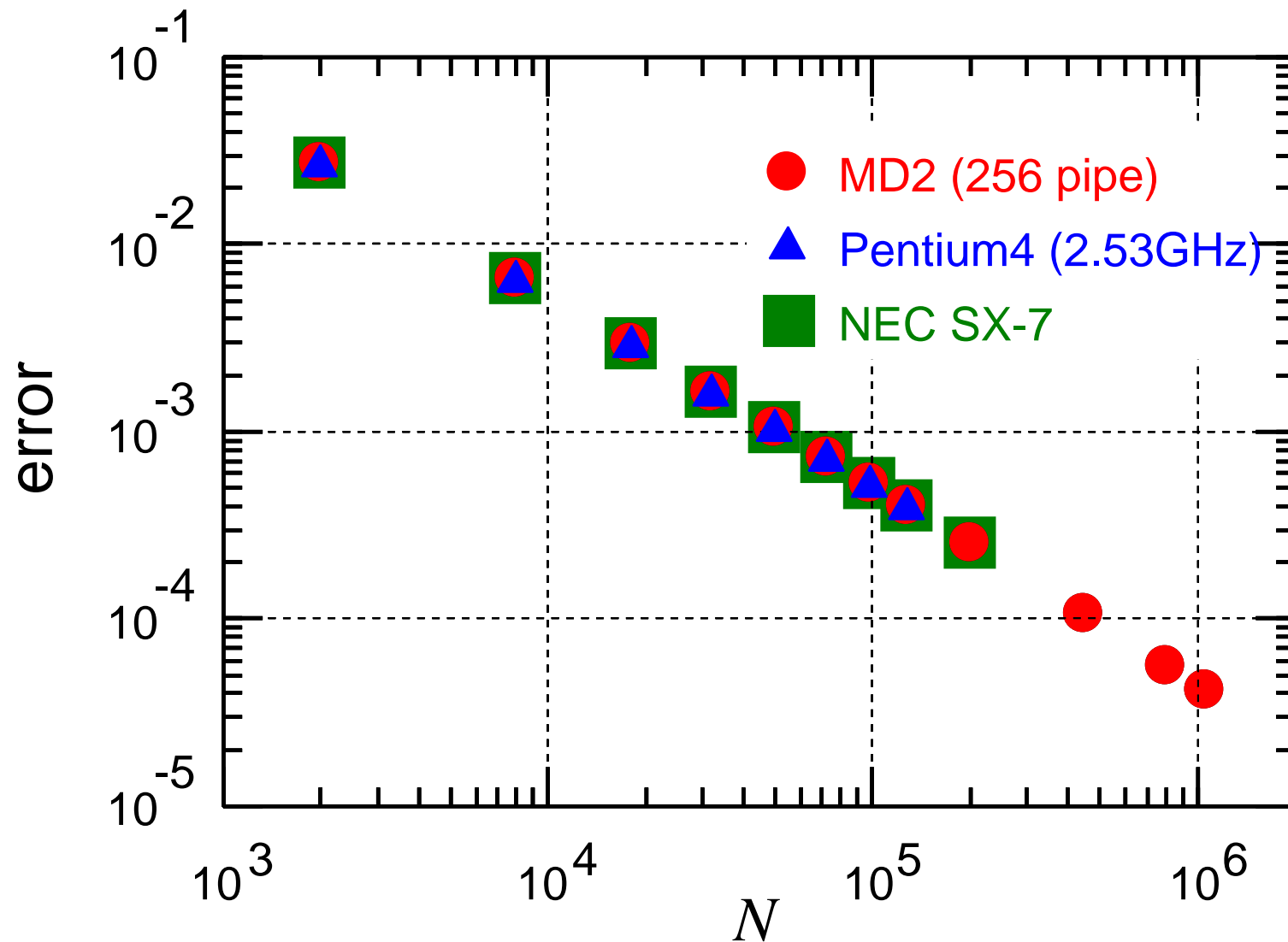
$N = 2000$

# Result: Timing



400x faster than Pen4-code and 30x faster than SX7-code

# Result: Accuracy

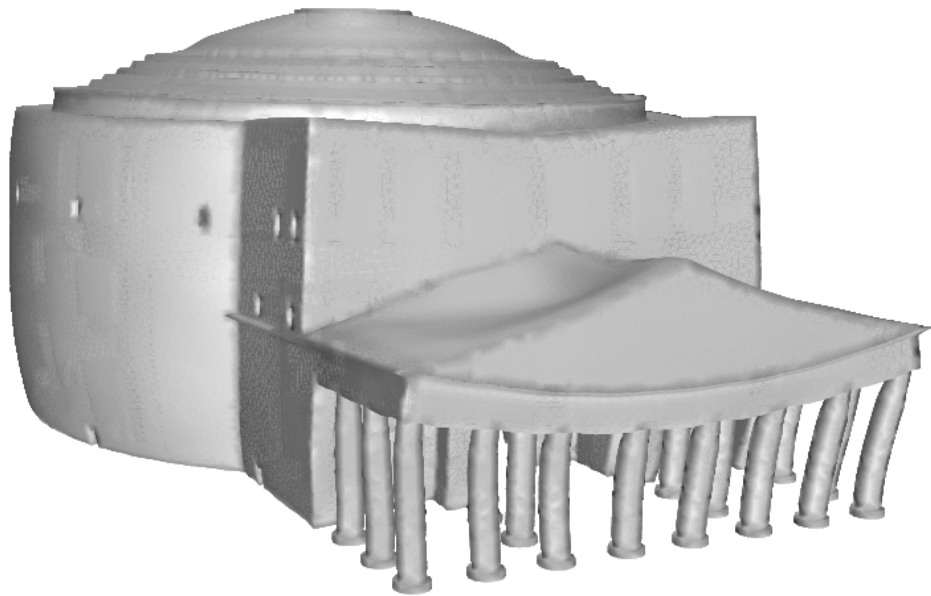


All of the results are in agreement with the exact solution

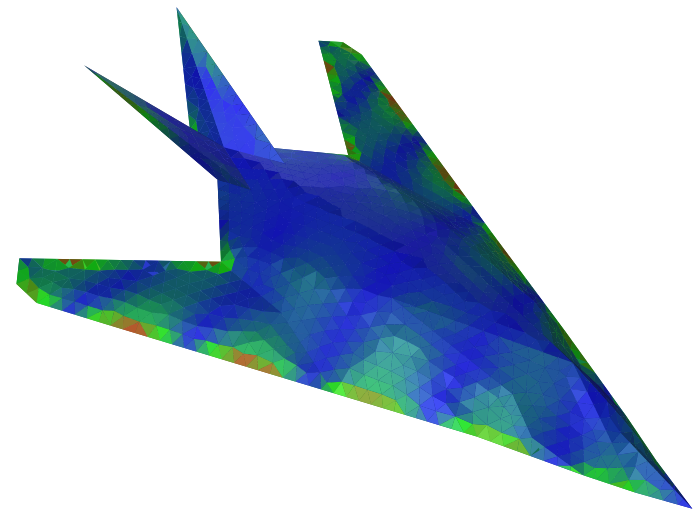
# Other applications

---

- Laplace and Helmholtz in 2 and 3D (2005)
- Elastostatics in 3D (2005)
- Electromagnetism in 3D (2006)
- Wave equation in 2D in time domain (2007)



Elastic deformation of “Pantheon”  
in Rome due to gravity



Surface current intensity of F-117  
subject to plane EM wave  
( $k = 20$ )

# Next step: MD2 + Fast algorithm?

---

- MD2+B&H's TREE algorithm?
  - ◇ In many-body,  $P^2M^2$  (Makino 1999, Kawai 2004) enables tree-code to run on GRAPE architecture.
    - (Unfortunately, my result was poor. Only 1.1x faster...)
  - ◇  $P^2M^2$  works for BEM for Laplace etc (Takahashi 2005)
  - ◇ However,  $P^2M^2$  could not be extended to BEMs in wave analyses (Helmholtz, Maxwell, etc).
    - Complexity remains  $O(N^2)$
    - Because operations are not “diagonal”.
- MD2+FMM?
  - ◇ I proposed a simple way, but it's not always faster than FMM.
    - Replacing M2L comp at the finest level with direct comp by MD2
- BEM needs a more flexible (programmable) engine like
  - ◇ GRAPE-DR, FPGA, and GPU

**October, 2004**

# GPGPU for BEM on GT7800

---

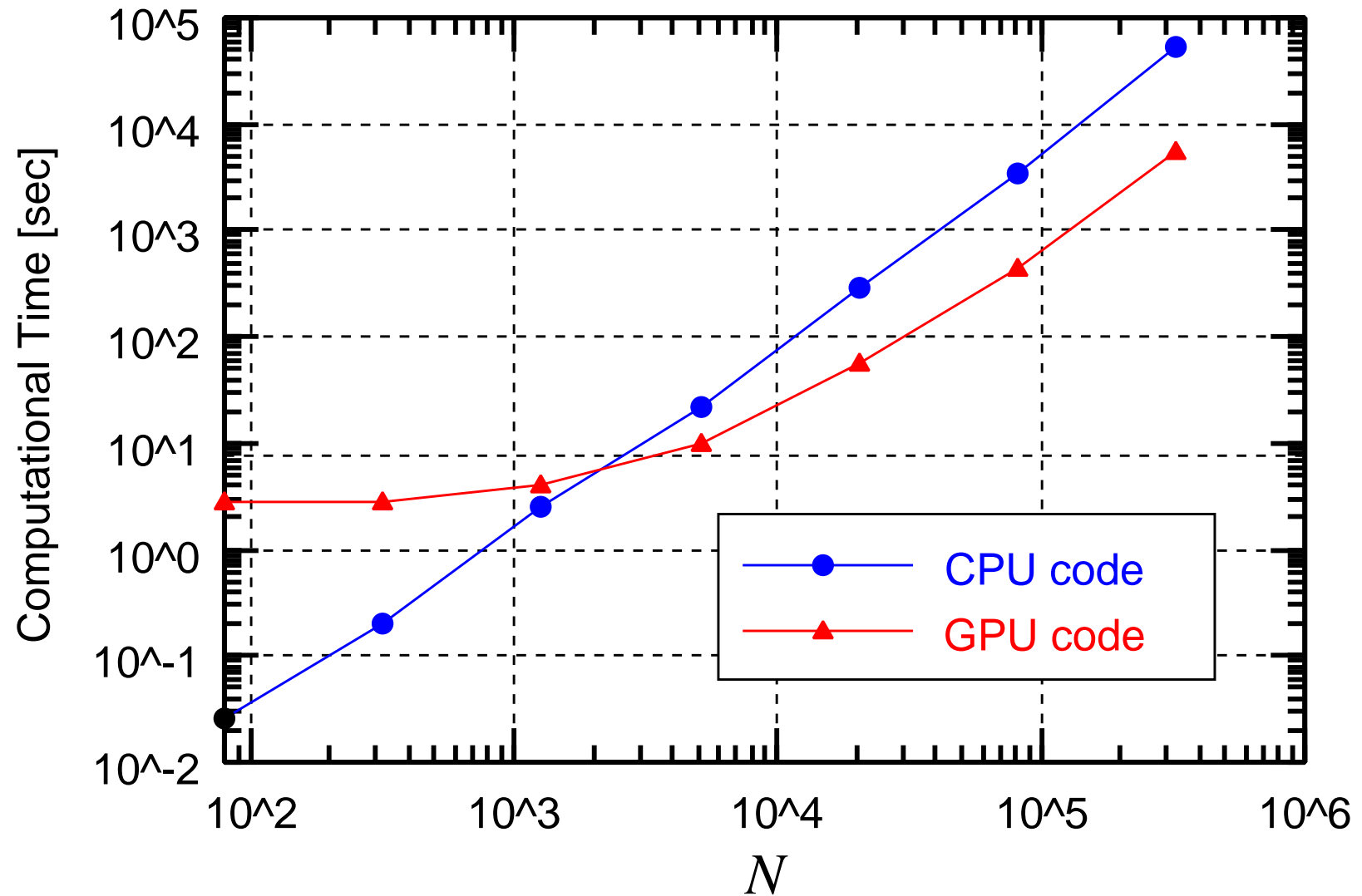
- Stimulated by
  - ◇ high-performance MD simulations on GPU by Iitaka
  - ◇ many other researchers in [www.gpgpu.org](http://www.gpgpu.org)
- Implemented an  $O(N^2)$  BEM code for Laplace eq
  - ◇ Followed the formulation of MDGRAPE-2
  - ◇ Programmed with Cg
  - ◇ Imported techniques in many-body (Mark Harris)  
(eg. “parallel-reduction” to accumulate j-particles)
- Corroborated with Iitaka, Mase, and Ebisuzaki (2005)
  - ◇ First attempt to run BEM on GPU.

# Benchmark

---

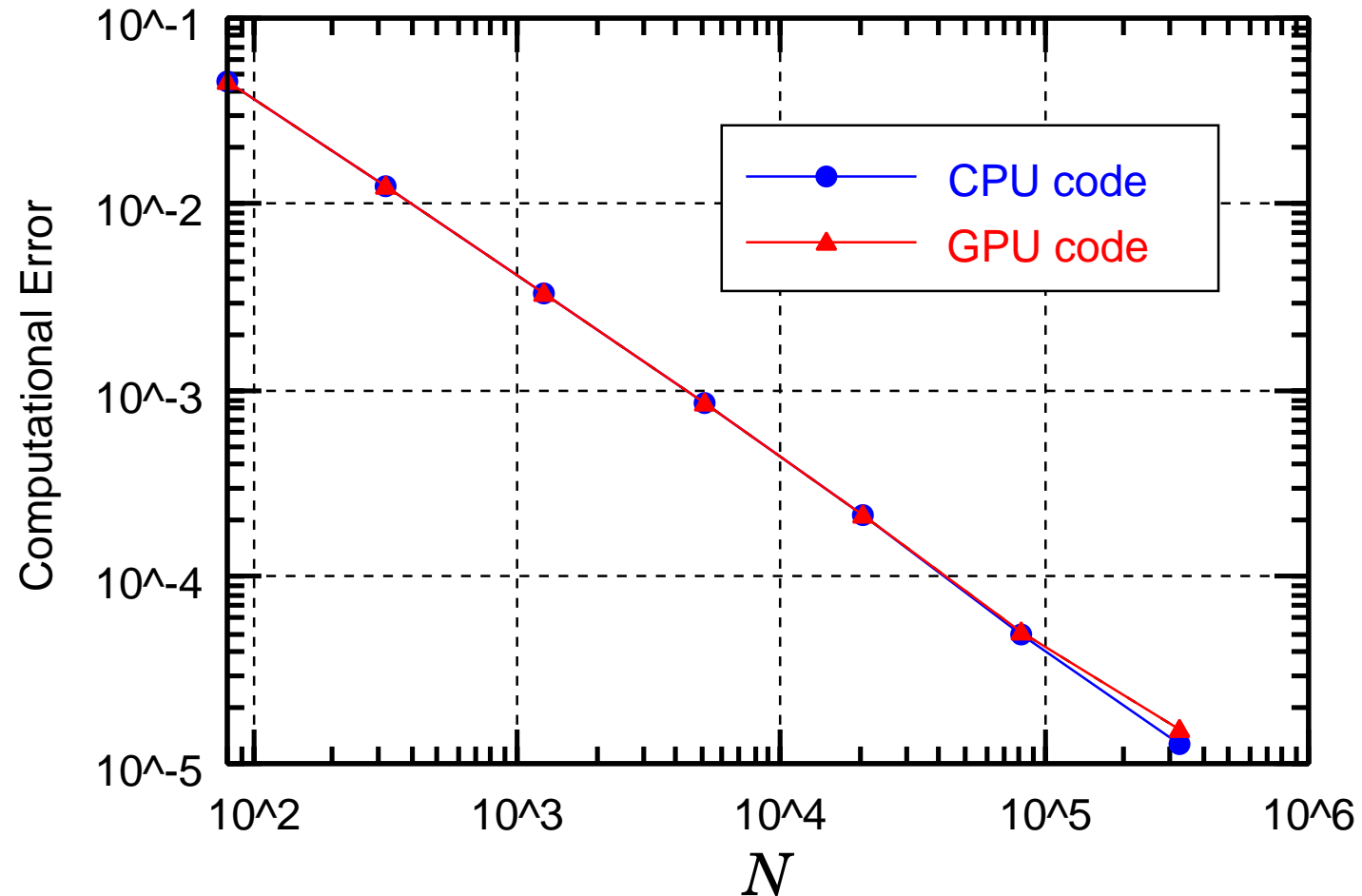
- Solve velocity potential around a sphere in uniform flow
- 3 points Gaussian formula, GMRES,...
- Computer:
  - ◇ **CPU**: Intel Xeon 3.20GHz
  - ◇ **GPU**: NVIDIA GeForce7800GT (256MB DDR3)

# Result: Timing



9.6 times faster for  $N = 327,680$ .

# Result: Accuracy



- Note: GPU-code was done in single-precision (`float`).
- Almost the same accuracy in this regime.

# Cg-based GPGPU

---

- It's OK, but Cg programming was quite tough!
- I gave up further development of BEM codes in 2006.

**2 years later...**

# GPGPU with CUDA

---

- Eyes was closing
  - ◇ CUDA released in 2006(?)
  - ◇ Hamada: CUNBODY (2007)
- I got up
  - ◇ Iitaka: “Introduction of Scientific Computing with GPU — even a *flog* can understand” (2007–8?)
  - ◇ Started, a month ago, to implement a BEM code on GT8800

# CUDA accelerating BEM for Helmholtz

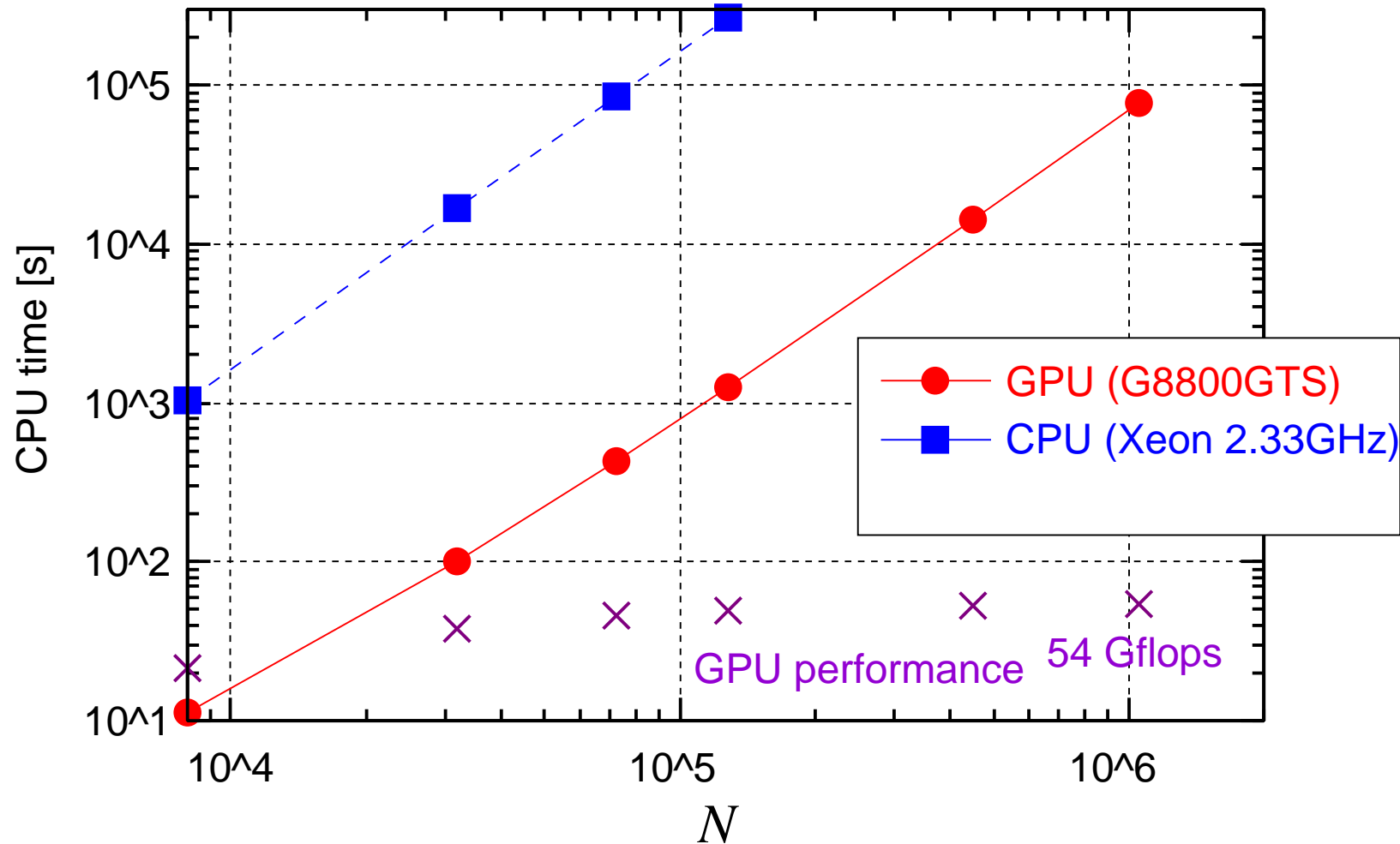
---

- Objective: Write a CUDA code to compute matvec or layer potential:

$$\int_S \left( \Gamma_k(\mathbf{x} - \mathbf{y})q(\mathbf{y}) - \frac{\partial \Gamma_k}{\partial n_y}(\mathbf{x}, \mathbf{y})u(\mathbf{y}) \right) dS_y \quad \text{where } \Gamma_k = \frac{e^{ik|\mathbf{x}-\mathbf{y}|}}{4\pi|\mathbf{x}-\mathbf{y}|}$$

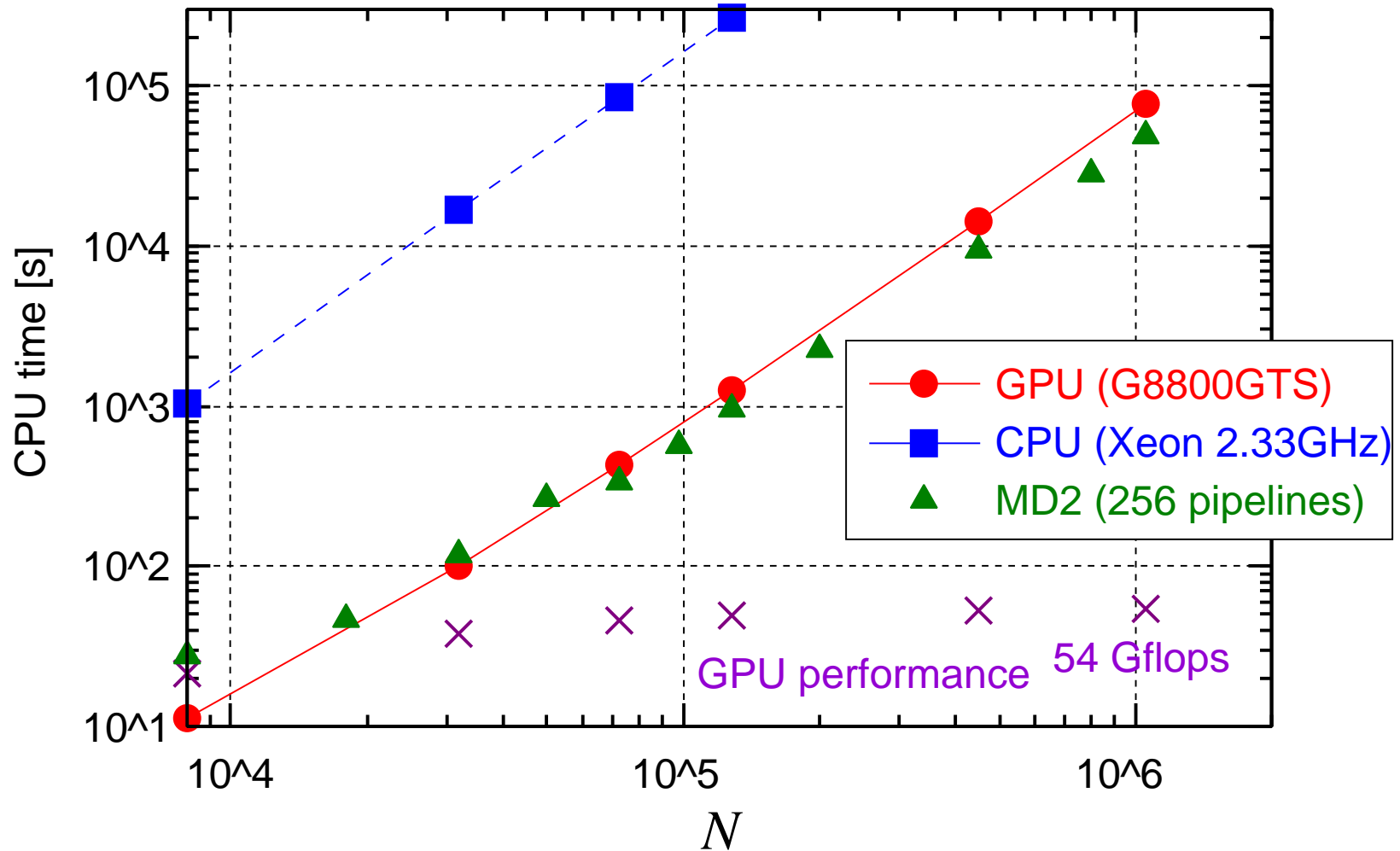
- Coding: many-body experts taught me all the techniques
  - ◇ Utilised **shared memory** to fetch j-particles' data (eg Hamada, GEM3)
  - ◇ **Accumulation** ( $\sum_j$ ) was done in pseudo double precision (Iitaka)  
Dekker's split, Knuth's trick, and dsfun90

# Benchmark: Timing & Performance



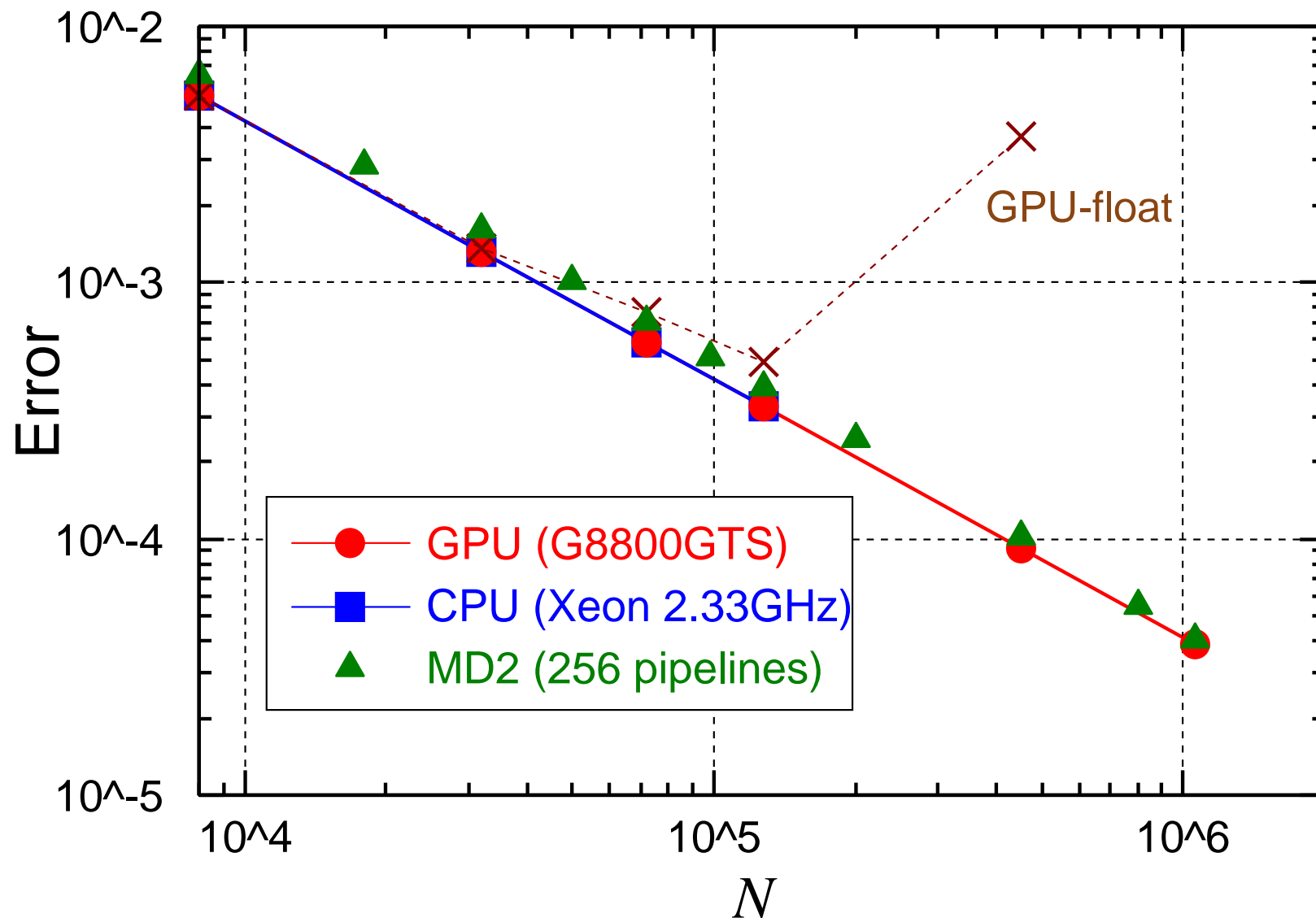
100 times faster & Sustained 54Gflops

# Benchmark: vs MD2



- MD2 is faster for  $N \gtrsim 5 \times 10^4$ . (Probably, MD3 is much more.)
- However, GPU is actually becoming a competitor to MDGRAPES.

# Benchmark: Accuracy



Implementation in fully single-precision should be avoided.  
(But, full double-precision may not be necessary.)

## § 5. Conclusion

- Wide variety of BEMs were successfully accelerated by MD-GRAPE and GPU.
- In the near future, implementation of “fast BEMs” on GPU should be challenged.
  - ◇ Gumerov and Duraiswami (Maryland): *Fast Multipole Methods on Graphics Processors* (2007)
  - ◇ Catch up and pass them soon!

# Acknowledgements

---

- MDGRAPE: Especially,
  - ◇ Ebisuzaki (RIKEN)
  - ◇ Kawai (K&F)
  - ◇ Narumi (Keio/RIKEN)
  - ◇ Ohno (RIKEN)
  - ◇ Susukita (KSIL)
  - ◇ Taiji (RIKEN)
- GPGPU:
  - ◇ Hamada (Nagasaki)
  - ◇ Iitaka (RIKEN)
  - ◇ Mase (RIKEN)

and...

Thank you for your attention!!