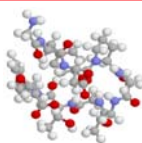


Background

1. Recently, the methods of computational chemistry become more applicable to macromolecules such as proteins with remarkable improvement of computational resources.
2. General Purpose GPU (GPGPU) is successfully applied for various problems in the field of high-performance computation using NVIDIA CUDA Environment.
3. GPGPU is already applied for some ab initio quantum chemistry calculations, but FMO method, which has high parallelization efficiency, remains untouched in GPGPU field.

Purpose of this Research

We developed FMO-MP2 algorithm suitable for GPU with CUDA environment. The performance is compared with GAMESS 2009 Jan. Chignolin, the smallest protein known today, is used as target compound.



Plan of Coding

1. Develop and optimize a reference serial program for a CPU.
2. Parallelize and optimize the time-consuming part of it for a GPU by using CUDA.
3. Coarse-grained parallel tasks in FMO-MP2 are mapped to a large-scale GPU cluster.

Algorithms

Features of CUDA GPGPU Development

1. Single thread multiple data (STMD) style.
2. Fast but small on-chip memory, large but slow external memory.
3. Double precision calculation is 10 times slower than single one.

Hartree-Fock :

Two-electron integrals were calculated with methods of McMurchie-Davidson and Rys quadrature, as suggested by Ufimtsev et. al. (2008) and Yasuda(2007).

RI-MP2 :

We use RI-MP2, which is the approximation of MP2. $O(N^4) \sim O(N^5)$ calculation which is dominant in total calculation is accelerated by CUBLAS matrix product library provided by NVIDIA Inc. Vogt et. al. (2008) succeeded in about 4 times acceleration of MP2 using CUDA. We have to pay attention that conventional MP2 in GAMESS is slower than RI-MP2.

Full-CD (Full – Cholesky Decomposition) :

RI needs different auxiliary basis for different basis sets and the approximation (HF/DFT/MP2). Therefore we implement Full-CD method which automatically generates a subset of product of a basis set as an auxiliary basis set for the targeting system.

We can adjust accuracy by changing for $\delta > \langle \mu\nu | \mu\nu \rangle - \sum_n L_{\mu\nu}^n L_{\mu\nu}^n$

Methods

- Method** FMO-RI-MP2/6-31G/SVP(CPU)
- Method** FMO-RI-MP2/6-31G/SVP(GPU)
- Method** FMO-FullCD($\approx 10^{-4}$)/6-31G(CPU)
- Method** FMO-FullCD ($\approx 10^{-4}$)/6-31G(GPU)

Comparison Program : GAMESS 2009 Jan.(CPU)

Results

Benchmark Environment

CPU : Intel Core 2 Quad Q9450 2.66GHz

GPU : NVIDIA GeForce GTX 295

OS : Fedora 8

Memory : 4GB

1 CPU core (1 process) and 1 GPU are used.

Each residue constitutes a fragment in FMO2.

Comparison of Speed

TABLE 1. Calculation Time for Chignolin and its Fragments

	(unit : sec)				
	GAMESS	RI(CPU)	RI(GPU)	CD(CPU)	CD(GPU)
Chignolin FMO-RHF	6834.00	-	-	-	-
Chignolin FMO-MP2	2565.20	204.29	27.39	405.66	44.45
TYR+TRP MP2	367.74	24.64	2.28	97.86	12.86
GLY MP2	0.88	0.56	0.17	0.29	0.05

Times for the generation of the auxiliary basis set on Full-CD are eliminated.

RI-MP2: 12 times faster than original GAMESS on the same CPU.

Full-CD, 6 times faster than original GAMESS on the same CPU.

Furthermore, 7-9 times acceleration is achieved on GPU.

Generally speaking, Full-CD which is more precise than RI needs larger auxiliary basis set and is expensive in calculation. However Full-CD does not need the two-electron-two-center integrals and the inversion of the large matrix.

Comparison of Accuracy

TABLE 2. MP2 Energy Calculation Result of Chignolin

	(unit : a.u.)				
	GAMESS	RI(CPU)	RI(GPU)	CD(CPU)	CD(GPU)
Chignolin FMO-MP2	-7.6566965	-7.6607912	-7.6608114	-7.6560510	-7.6556359

RI method has a error of 10^{-2} order and Full-CD($\approx 10^{-4}$) has a error of 10^{-3} order.

The error of float calculation on GPU is about 10^{-3} . The accuracy is similar to the error of Full-CD ($\approx 10^{-4}$).

Summary

1. FMO-MP2 was accelerated up to 100 times with RI-MP2 and full-CD executed on GPU.
2. Energy errors of the single-precision RI-MP2 or Full-CD were small enough for general usage.

Future Theme

1. Furthermore acceleration of RI-MP2
2. GPU implementation of RI-HF
3. Two-electron integral acceleration on GPU
4. Optimization of algorithm for automatic generation of auxiliary basis set
5. Multi GPU parallelization