Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186-5140
Volume 14, Number 1, pages 68–73, January 2025

A Parallel Computation for McMurchie-Davidson Algorithm on the GPU

Haruto Fujii*, Yasuaki Ito*, Satoki Tsuji*†, Kanta Suzuki*, Nobuya Yokogawa*, Koji Nakano*, and Akihiko Kasagi†
*Graduate School of Advanced Science and Engineering, Hiroshima University
Kagamiyama 1-4-1, Higashi-Hiroshima, 739-8527, JAPAN
†Computing Laboratory, Fujitsu Limited
Kamikodanaka 4-1-1, Nakahara-ku, Kawasaki, 211-8588, JAPAN

***Abstract*—The two-electron repulsion integral (ERI) is a fundamental component in quantum chemistry, involving the calculation of integrals over four basis functions. Quantum chemistry computations require $O(M^4)$ 4 ERI calculations for M basis functions, making ERI evaluations a significant bottleneck in these computations. In this paper, we propose an efficient GPU-based parallelization of the McMurchie-Davidson algorithm, which recursively computes ERI. Our method focuses on optimizing the computations by leveraging the shell parameter, which characterizes the shape of the basis functions. The results demonstrate that the shell-based optimization achieves a performance improvement of up to 15% compared to the non-optimized approach.**

***Index Terms*—molecular integrals, two-electron repulsion integrals, quantum chemistry, GPU, CUDA**

I. はじめに

量子化学計算とは，分子の原子核座標や使用する基底関数を入力とし，Schrödinger 方程式を解くことで計算により分子の全エネルギーなどを求める手法である．計算時間の大半を占める代表的なものとして **2 電子反発積分 (ERI, two-electron repulsion integrals)** が存在している．ERI の具体的な計算量は，N 個の基底関数を用いて系を表現する場合，$O(N^4)$ となる．ERI の計算コストを削減できれば量子化学計算全体の時間を大きく削減できるため，様々な先行研究が行われている．また，ERI の計算手法に関して，McMurchie-Davidson 法 [1] や Obara-Saika 法 [2] など，異なる特性を持つ様々なアルゴリズムが提案されている．本研究では，McMurchie-Davidson 法を用いた計算に着目し，その高速化を目指す．

II. 2 電子反発積分 (ERI)

本項では，2 電子反発積分 (ERI) について説明する．ERI は，4 つの基底関数 $\chi_\mu, \chi_\nu, \chi_\lambda, \chi_\sigma$ に対して，

$$(\mu\nu|\lambda\sigma) = \iint \chi_\mu(r_1)\chi_\nu(r_1)\frac{1}{r_{12}}\chi_\lambda(r_2)\chi_\sigma(r_2)dr_1dr_2 \quad (1)$$

として定義される全空間での積分である．ここで，基底関数 χ に関するパラメータ r_1, r_2 は三次元座標 (x, y, z) として表され，r_{12} は r_1, r_2 間のユークリッド距離である．量子化学計算においては，基底関数の 4 つの組み合わせ毎に ERI を計算する必要がある．基底関数の組み合わせは N^4 通り存在するが，式 (1) の対称性より以下の等式

$$(\mu\nu|\lambda\sigma) = (\nu\mu|\lambda\sigma) = (\mu\nu|\sigma\lambda) = (\nu\mu|\sigma\lambda)$$
$$= (\lambda\sigma|\mu\nu) = (\lambda\sigma|\nu\mu) = (\sigma\lambda|\mu\nu) = (\sigma\lambda|\nu\mu)$$

が成立するため，実際に計算が必要な ERI の個数は $\frac{M(M+1)(M^2+M+2)}{8}$ となる．

さらに，量子化学計算において，基底関数 $\chi_\mu(r)$ は一般的にガウス型軌道 G_{μ_e} の線形和として式 (2) で定義される．

$$\chi_\mu(r) = \sum_{e=1}^{K_\mu} d_{\mu_e} G_{\mu_e}(r) \quad (2)$$

式 (2) 中における K_μ, d_{μ_e} は，それぞれ線形結合の項数と係数である．また，ガウス型軌道 G_{μ_e} は，

$$G_{\mu_e} = c_e\,(x - A_x)^{\mu_x}\,(y - A_y)^{\mu_y}\,(z - A_z)^{\mu_z}\exp\left(-\alpha_e\,|r - A|^2\right)$$

として定義される．ここで，$A = (A_x, A_y, A_z)$ は軌道中心，c_e は正規化係数，μ_x, μ_y, μ_z は方位量子数と呼ばれる軌道の形状を表す非負の整数であり，α_e は軌道の広がり方を表すパラメータである．即ち，基底関数とは，同一の中心座標 A 及び同一の方位量子数 (μ_x, μ_y, μ_z) を持ち，異なる係数 d_{μ_e} 及び α_e を持つガウス型軌道の集合である．方位量子数の和も非負の整数値となり，この値毎に軌道に名前がつけられることもある．具体的には，和が 0, 1, 2, 3 のものは s, p, d, f 軌道と呼ばれ，それ以上のものは f 以降のアルファベット順となっている．また，これらのパラメータは，基底関数や原子の種類によって一意に決定される．

4 つのガウス型軌道に関する ERI を以下の式 (3) のように定義した場合，

$$[\mu_e\nu_f|\lambda_g\sigma_h] = \iint G_{\mu_e}(r_1)G_{\nu_f}(r_1)\frac{1}{r_{12}}G_{\lambda_g}(r_2)G_{\sigma_h}(r_2) \quad (3)$$

式 (1) は，以下の式 (4) のように表すことができる．

$$(\mu\nu|\lambda\sigma) = \sum_{e=1}^{K_\mu}\sum_{f=1}^{K_\nu}\sum_{g=1}^{K_\lambda}\sum_{h=1}^{K_\sigma} d_{\mu_e}d_{\nu_f}d_{\lambda_g}d_{\sigma_h}\,[\mu_e\nu_f|\lambda_g\sigma_h] \quad (4)$$

また，先行研究として様々な種類の基底関数が提案されており，Basis Set Exchange [3]–[5] を利用することで Web 上のデータベースから様々な基底関数についての各種パラメータを取得することが可能である．

III. McMurchie-Davidson 法

本項では，式 (3) で表される ERI の計算手法の 1 つである McMurchie-Davidson 法 [1] について説明する．以下の 4 つのガウス型軌道からなる ERI($[ab|cd]$) について考える (正規化係数は省略).

$$G_a(r) = (x-A_x)^{a_x}(y-A_y)^{a_y}(z-A_z)^{a_z}\exp\left(-\alpha\left|r-A\right|^2\right)$$
$$G_b(r) = (x-B_x)^{b_x}(y-B_y)^{b_y}(z-B_z)^{b_z}\exp\left(-\beta\left|r-B\right|^2\right)$$
$$G_c(r) = (x-C_x)^{c_x}(y-C_y)^{c_y}(z-C_z)^{c_z}\exp\left(-\gamma\left|r-C\right|^2\right)$$
$$G_d(r) = (x-D_x)^{d_x}(y-D_y)^{d_y}(z-D_z)^{d_z}\exp\left(-\delta\left|r-D\right|^2\right)$$

MD アルゴリズムでは，以下の式 (5) を用いてガウス型軌道に関する ERI を計算する．

$$[ab|cd] = \frac{2\pi^{5/2}}{pp'\sqrt{p+p'}}\sum_{\substack{t=0\\u=0\\v=0}}^{\substack{a_x+b_x\\a_y+b_y\\a_z+b_z}}\sum_{\substack{t'=0\\u'=0\\v'=0}}^{\substack{c_x+d_x\\c_y+d_y\\c_z+d_z}} e^{t,u,v}_{t',u',v'}R^0_{t+t',u+u',v+v'} \quad (5)$$

式 (5) において，

$$e^{t,u,v}_{t',u',v'} = E^{a,b}_{t,u,v}E^{c,d}_{t',u',v'}(-1)^{t'+u'+v'}$$
$$E^{a,b}_{t,u,v} = E^{a_x,b_x}_t E^{a_y,b_y}_u E^{a_z,b_z}_v$$
$$E^{c,d}_{t',u',v'} = E^{c_x,d_x}_{t'} E^{c_y,d_y}_{u'} E^{c_z,d_z}_{v'}$$

であり，$p=\alpha+\beta, p'=\gamma+\delta$ である．

漸化式 $E^{a_x,b_x}_t, E^{a_y,b_y}_u, E^{a_z,b_z}_v$ は，ガウス型軌道 G_a, G_b に関する項である．例として，E^{a_x,b_x}_t は以下の漸化式 (6) によって定義される．

$$\begin{cases} E^{i,j}_t = \dfrac{1}{2p}E^{i-1,j}_{t-1} - (A_x-B_x)\dfrac{q}{\alpha}E^{i-1,j}_t + (t+1)E^{i-1,j}_{t+1} \\ E^{i,j}_t = \dfrac{1}{2p}E^{i,j-1}_{t-1} + (A_x-B_x)\dfrac{q}{\beta}E^{i,j-1}_t + (t+1)E^{i,j-1}_{t+1} \\ E^{0,0}_t = \exp\left(-q(A_x-B_x)^2\right) \\ E^{i,j}_t = 0 \quad \text{if } t<0 \text{ or } i+j<t \end{cases} \quad (6)$$

式 (6) において，$q=\frac{\alpha\beta}{\alpha+\beta}$ である．また，$E^{a_y,b_y}_u, E^{a_z,b_z}_v$ はそれぞれ y,z 座標の値を用いて計算可能であり，$E^{c_x,d_x}_{t'}E^{c_y,d_y}_{u'}E^{c_z,d_z}_{v'}$ は G_c, G_d について同様の式を用いて計算可能である．漸化式 E^{a_x,b_x}_t の評価は，後述する $R^0_{t,u,v}$ よりは計算コストが軽いものの，漸化式をそのまま再帰関数として実装すると計算時間の増加につながる．そこで，本研究では，この再帰式を展開し，(i,j,t) の組毎に最適化された専用の device 関数を作成することで計算コストの短縮を図った．

漸化式 $R^0_{t+t',u+u',v+v'}$ は，以下の漸化式 (7) によって定義される，ERI 評価において最も計算コストの高い項である．

$$\begin{cases} R^n_{t,u,v} = (t-1)R^{n+1}_{t-2,u,v} + (P_x-P'_x)R^{n+1}_{t-1,u,v} \\ R^n_{t,u,v} = (u-1)R^{n+1}_{t,u-2,v} + (P_y-P'_y)R^{n+1}_{t,u-1,v} \\ R^n_{t,u,v} = (v-1)R^{n+1}_{t,u,v-2} + (P_z-P'_z)R^{n+1}_{t,u,v-1} \\ R^n_{0,0,0} = (-2\omega)^n F_n(\omega|P-P'|^2) \\ R^n_{t,u,v} = 0 \text{ if } t<0 \text{ or } u<0 \text{ or } v<0 \end{cases} \quad (7)$$

ここで，$P=(P_x,P_y,P_z)=\frac{\alpha A+\beta B}{\alpha+\beta}, P'=\left(P'_x,P'_y,P'_z\right)=\frac{\gamma C+\delta D}{\gamma+\delta}, \omega=\frac{pp'}{p+p'}, |P-P'|=(P_x-P'_x)^2+(P_y-P'_y)^2+(P_z-P'_z)^2$ である．式 (7) に含まれる $F_n(\omega|P-P'|^2)$ は Boys 関数 [6] と呼ばれる関数で，

$$F_j(T) = \int_0^1 u^{2j}\exp\left(-Tu^2\right)du \quad (8)$$

として計算可能な，解析的に解くことが不可能な関数である．本研究では，辻らの研究 [7] にて示された GPU 実装をそのまま利用した．そして，この関数の評価コストが非常に高いため，評価回数を減らすことで高速化を図ることが可能である．また，Boys 関数評価の特殊なケースとして，$T=0$ の場合は $F_j(T)=\int_0^1 u^{2j}du=\frac{1}{2j+1}$ となるため，この場合は極めて低い計算コストとなる．$T=0$ となるケースとしてもっとも多いのは，ERI に用いる 4 つのガウス関数の中心座標 (A,B,C,D) が全て同じ場合である．

本研究では，この漸化式 R に着目し高速化を行った．

IV. Shell ベースでの ERI

式 (8) における Boys 関数の定義から，Boys 関数の引数は n 及び $\omega|P-P'|^2$ によって決定されることが分かる．また，ω, P, P' の定義から，$\alpha,\beta,\gamma,\delta,A,B,C,D$ の値がそれぞれ等しいガウス型軌道に関する ERI では，使用する Boys 関数の値は同じであることが確認できる．しかし，基底関数に含まれるガウス型軌道は一般的に異なる α, A の値を持つため，基底関数の ERI に含まれるガウス型軌道の ERI を計算する場合，各計算毎に Boys 関数の値を評価する必要があり，無駄の多い実装となる．

この問題を解決するために，Ivan らの先行研究 [8] において，α, A の値が等しいガウス型軌道の集合である *shell* が提案された．shell の定義は以下の通りである．

$$S_\xi(k,r,A,\alpha) = \{d_\xi G_\xi(\{\xi_x,\xi_y,\xi_z\},r,A,\alpha) \mid \xi_x,\xi_y,\xi_z\geq 0, \xi_x+\xi_y+\xi_z=k\} \quad (9)$$

式中の d_ξ は，式 (2) のものと同様に係数パラメータである．$G_\xi(\{\xi_x,\xi_y,\xi_z\},r,A,\alpha)$ についても式 (2) と同様のガウス関数で，

$$G_\xi(\{\xi_x,\xi_y,\xi_z\},r,A,\alpha) = c(x-A_x)^{\xi_x}(y-A_y)^{\xi_y}(z-A_z)^{\xi_z}\exp\left(-\alpha|r-A|^2\right)$$

として定義される．これらの定義から，shell は α, A の値がそれぞれ等しく，方位量子数の和 $(\xi_x+\xi_y+\xi_z)$ も同じであるガウス型軌道の集合であるといえる．基底関数の場合と同様，この値毎に shell にも名前が付けられることがあり，順に s, p, d, f-shell となる．また，shell 内に含まれるガウス型軌道の個数は，式 (9) の定義から方位量子数の和によって決定されることが分かる．以下の表 I に，shell に含まれるガウス型軌道の個数を示す．これらの個数は，$t+u+v=k$ を満たす非負整数 (t,u,v) の組の個数は $\frac{(k+2)!}{k!2!}=\frac{(k+2)(k+1)}{2}$ として求められることから導出される．

4 つの shell(*ShellPair(SP)*) についての ERI は，以下の式 (10) のように定義される．ただし，座標 r 以外のパラメータを省略していることに注意されたい．

$$(\xi\eta|\theta\kappa) = \iint S_\xi(r_1)S_\eta(r_1)\frac{1}{r_{12}}S_\theta(r_2)S_\kappa(r_2)dr_1dr_2 \quad (10)$$

TABLE I
THE NUMBER OF GAUSSIAN-TYPE ORBITALS IN THE SHELL

shell	Gaussian-type orbitals
s	1
p	3
d	6
f	10
g	15
h	21
i	28
⋮	⋮

4 つの shell(*ShellPair(SP)*) に含まれるガウス型軌道から構成される ERI を計算する場合，一度計算した Boys 関数の値を使いまわすことが可能となる．この考えを元にしたものが shell ベースの ERI 計算である．

V. 提案アルゴリズム，GPU 実装

本章では，MD 法を用いた ERI 計算を更に効率化する為のアルゴリズム，及びその GPU 実装を提案する．前述した MD アルゴリズムには，Boys 関数の評価コストが高いことに加え，シグマ計算の過程で深い再帰を持つ関数 R が何度も呼ばれ，同じ値が何度も評価されるという冗長な計算が行われるという問題が存在する．以前に我々が提案した手法 [9] は，再帰関数 R を展開し，それらを相互依存性を持たずに計算可能な計算単位である batch に分割することで，冗長な計算を回避するものである．そして，batch 内の各要素を各一度のみ事前に評価することで，冗長な計算を回避した上で必要な $R^0_{t,u,v}$ の値を事前計算することが可能となる．

以下に，batch アルゴリズムの概要を示す．詳細は，以前の我々の論文を参照されたい．まず，$s_x = a_x + b_x + c_x + d_x, s_y = a_y + b_y + c_y + d_y, s_z = a_z + b_z + c_z + d_z, K = s_x + s_y + s_z$ を定義する．このとき，ERI の計算には，$0 \leq t \leq s_x, 0 \leq u \leq s_y, 0 \leq v \leq s_z$ を満たす $R^0_{t,u,v}$ を全て計算する必要がある．そして，必要な $R^0_{t,u,v}$ の漸化式を展開したもの $(R^n_{t,u,v})$ のうち，以下の条件を満たすものを batch $k(0 \leq k \leq K)$ に割り当てる．

$$k = t + u + v, 0 \leq n \leq K - k$$

ここで，batch k に含まれる $R^n_{t,u,v}$ の個数は，$t+u+v=k$ を満たす (t,u,v) の組は $\frac{(k+2)!}{k!2!} = \frac{(k+2)(k+1)}{2}$ であり，batch k には $0 \leq n \leq K-k$ を満たす $R^n_{t,u,v}$ が含まれることから $\frac{(k+2)(k+1)(K-k+1)}{2}$ である．また，$R^n_{t,u,v}$ の総数は，$\sum_{k=0}^{K} \frac{(k+2)(k+1)(K-k+1)}{2} = \frac{(K+1)(K+2)(K+3)(K+4)}{24}$ である．

そして，batch 0 から batch K までを順に計算することで，必要な $R^0_{t,u,v}$ の値を全て求めることが可能となる．これは，式 (3) 及び式 (7) の定義より明らかである．また，batch 内のそれぞれの $R^n_{t,u,v}$ の値は，式 (7) より相互依存性を持たないため，各 batch 内の値は全て並列に計算することが可能である．

次に，上述したアルゴリズムを GPU 上に実装する 2 通りの手法を説明する．

A. 基底関数ベース実装 *(1B1CI)*

本実装は，我々の以前の研究にて行ったものと同じものである．詳細は，先行研究についての論文を参照されたい．本実装は，II 章の内容を基にした実装であり，1 つの CUDA Block を 1 つの基底関数についての ERI に割り当てる．各 Block 内では，基底関数についての ERI に含まれるガウス型軌道についての ERI(式 (4) 参照) を順に計算する．即ち，本実装は二段階の並列化からなる．

一段階目の並列化として，基底関数についての ERI の計算をアルゴリズム 1 に示す．アルゴリズムの 1 行目に示す通り，このアルゴリズムでは基底関数についての ERI を並列に行う．このアルゴリズムの 2–5 行目では，式 (3) に示すガウス型軌道に関する ERI を順に評価することで，基底関数についての ERI の値を求めている．この時，ガウス型軌道に関する ERI の計算は，アルゴリズム 2 に示す手法で行われる．最終的に，求めた基底関数についての ERI の値はメモリ上の対応する位置に書き戻される．

二段階目の並列化として，ガウス型軌道についての ERI の計算をアルゴリズム 2 に示す．このアルゴリズムの 1–3 行目では，前述した batch アルゴリズムの各計算を行う，そのようにして得られた $R^0_{t,u,v}$ の値を用いて，5–8 行目では式 (3) についての計算を行う．ここでは，前述した batch アルゴリズムを Block 内の Thread を用いて並列に動作させる．また，batch アルゴリズムを GPU 上に実装する際，VI 章にて詳しく説明する Shared Memory のトリプルバッファリングを使用することにより，必要な Shared Memory の容量を削減している．

以後，本実装を 1B1CI(1Block-1ContractedIntegral) と呼称する．Contracted Integral とは，式 (2) に示す通り，基底関数がガウス型軌道の縮約形式で表されることから命名されたものである．

Algorithm 1 Parallel ERI algorithm for basis functions

Input: N basis functions
Output: ERIs for all combinations of four basis functions
1: **for all** $\mu, \nu, \lambda, \sigma$ **such that** $\mu \leq \nu$, $\lambda \leq \sigma$, **and** $(\mu, \nu) \leq (\lambda, \sigma)$ **do in parallel**
2: $t \leftarrow 0$ // Eq. (4)
3: **for all** e, f, g, h **do**
4: $t \leftarrow t + d_{\mu_e} d_{\nu_f} d_{\lambda_g} d_{\sigma_h} [\mu_e \nu_f | \lambda_g \sigma_h]$ // Algorithm 2
5: **end for**
6: $(\mu\nu|\lambda\sigma) \leftarrow t$
7: **end for**

Algorithm 2 Parallel ERI algorithm for Gaussian-type orbitals

Input: Four Gaussian-type orbitals G_a, G_b, G_c, G_d
Output: ERI $[ab|cd]$
1: **for** $k = 0$ to K **do**
2: Compute R values in batch k **in parallel**
3: **end for**
4: $s \leftarrow 0$ // s is a shared variable, and parallel addition is performed on it
5: **for all** t, u, v, t', u', v' **do in parallel**
6: $s \leftarrow s + E^{a_x b_x}_t E^{a_y b_y}_u E^{a_z b_z}_v E^{c_x d_x}_{t'} E^{c_y d_y}_{u'} E^{c_z d_z}_{v'}$
7: $(-1)^{t'+u'+v'} R^0_{t+t', u+u', v+v'}$
8: **end for**
9: **return** $\frac{2\pi^{5/2}}{pp'\sqrt{p+p'}} \cdot s$

B. *Shell* ベース実装 *(1B1SP)*

本実装は，IV 章の内容をベースとした実装であり，1 つの CUDA Block を，1 つの ShellPair についての ERI に割り当てる．各 Block 内では，ShellPair についての ERI に含まれ

るガウス型軌道についての ERI を順に計算する．即ち，本実装も 1B1CI と同様に二段階の並列化からなる．

一段階目の並列化として，shell についての ERI の計算をアルゴリズム 3 に示す．アルゴリズムの 1 行目に示す通り，このアルゴリズムでは shell についての ERI を並列に行う．このアルゴリズムの 2 行目では，ShellPair 内のガウス型軌道についての ERI 計算にて用いる Boys 関数の値を全て評価する．3–5 行目では，式 (3) に示すガウス型軌道に関する ERI を順に評価する．各計算毎に，各ガウス型軌道が属する基底関数 $(\mu, \nu, \lambda, \sigma)$ に対応するメモリ位置に計算結果を書き戻す．そのため，本アルゴリズムを使用した場合についても，計算結果は基底関数についての ERI と同じものとなる．但し，各計算毎にメモリへの書き戻しが発生する為，メモリアクセスに伴うレイテンシが多数生じることで実行時間が増加する可能性もある．この時，ガウス型軌道についての ERI の計算には，アルゴリズム 2 に示す手法を用いて行われる．

二段階目の並列化であるガウス型軌道についての ERI の計算は，1B1CI と同様にアルゴリズム 2 を用いて行う．1B1CI との変更点として，1-3 行目に示す batch アルゴリズムについての計算は，アルゴリズム 3 の 2 行目で計算した Boys 関数の値を用いて行う．これを行うことにより，1B1CI の場合と比較し更に Boys 関数の評価回数を削減することが可能となる．また，本実装においても，1B1CI の場合と同様に Shared Memory のトリプルバッファリングを用いた実装を行っている．

1B1SP は，1B1CI と比較し，Boys 関数の計算回数が大幅に減少する代わりにメモリアクセス回数が増加するという特徴がある．以後，本実装を 1B1SP(1Block-1ShellPair) と呼称する．

Algorithm 3 Parallel ERI algorithm for shells

Input: M shells
Output: ERIs for all combinations of four basis functions
1: **for all** $\xi, \eta, \theta, \kappa$ **such that** $\xi \leq \eta$, $\theta \leq \kappa$, **and** $(\xi, \eta) \leq (\theta, \kappa)$ **do in parallel**
2: Compute all $F_n(\omega|P - P'|^2)$ values in $(\xi\eta|\theta\kappa)$ **in parallel**
3: **for all** Gaussian-type orbitals pair $(\xi_e, \eta_f, \theta_g, \kappa_h)$ in $\xi, \eta, \theta, \kappa$ **do**
4: $(\mu\nu|\lambda\sigma) \leftarrow d_\xi d_\eta d_\theta d_\kappa [\xi_e \eta_f | \theta_g \kappa_h]$ // Algorithm 2
5: **end for**
6: **end for**

VI. Shared Memory のトリプルバッファリング

本章では，batch アルゴリズムに必要な Shared Memory 容量を削減する為の手法である Shared Memory のトリプルバッファリングについて説明する．V 章に示す通り，batch アルゴリズムでは batch k に含まれる $R^n_{t,u,v}$ の値を batch 0 から batch K まで順番に計算する．また，式 (7) から，batch k に含まれる計算をする上では batch $k-1$ 及び batch $k-2$ の値があれば十分であるという事が分かる．この事から，batch アルゴリズムにて計算する $R^n_{t,u,v}$ の値を全て保持しておく必要はないといえる．この点に着目したのが本手法である Shared Memory のトリプルバッファリングである．

本手法では，Shared Memory 上に 3 つのバッファを用意し，それぞれのバッファには 1 つの batch に含まれる $R^n_{t,u,v}$ の値を格納するようにする．そして，図 1 に示すように使用するバッファを切り替えながら計算を進める．

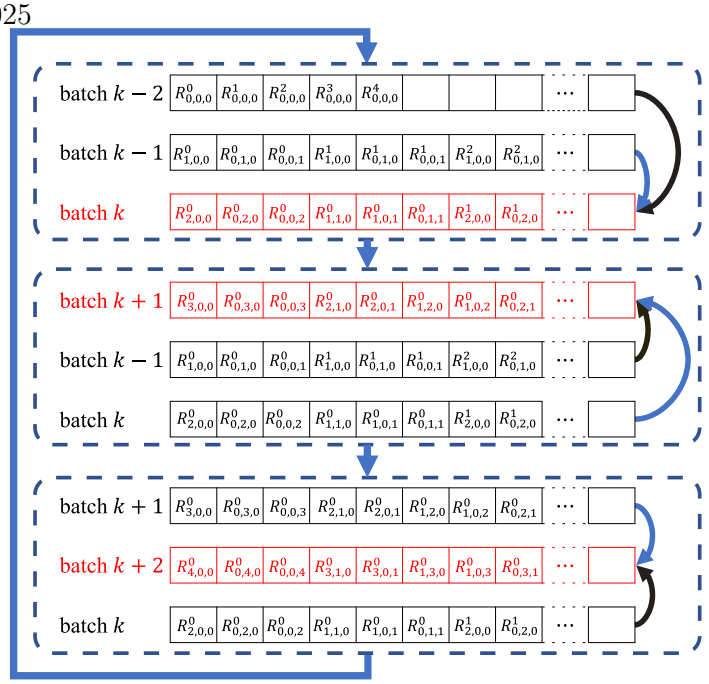

Fig. 1. Computation of R values for each batch using triple buffering of the shared memory

本実装は，全ての $R^n_{t,u,v}$ の値を保持しておく実装と比較し，必要な Shared Memory の容量が小さい．全ての $R^n_{t,u,v}$ の値を保持しておく実装の場合，必要な Shared Memory の要素数は

$$\sum_{k=0}^{K} \frac{(k+2)(k+1)(K-k+1)}{2} = \frac{(K+1)(K+2)(K+3)(K+4)}{24}$$

である．それに対し，本手法で用いるバッファのサイズは，

$$\max_{0 \leq k \leq K} \frac{(k+1)(k+2)(K-k+1)}{2}$$

である．これは，最も要素数の多い batch の値を保持できれば良い為である．バッファは 3 つ必要なので，実際はこの 3 倍の要素数が必要である．更に，アルゴリズム 2 の計算には $R^0_{t,u,v}$ の値を使用する為，この値も別途保持しておく必要がある．これには，

$$\sum_{k=0}^{K} \frac{(k+2)(k+1)}{2} = \frac{(K+1)(K+2)(K+3)}{6}$$

要素のメモリが必要である．

$0 \leq K \leq 24$ におけるトリプルバッファリングとナイーブな実装 (全ての $R^n_{t,u,v}$ の値を保持) について，それぞれのメモリ使用量を図 2 に，削減率を図 3 に示す．図 2 より，K が 7 以上の場合，トリプルバッファリングを用いた方が必要なメモリ量が少なくなるということが分かる．また，図 3 より，K の値が大きくなるほどメモリ量の削減率は大きくなり，最大で 0.344 倍，即ち約 65%の削減を達成した．これにより，Shared Memory 容量に制限のある GPU デバイス上であっても提案アルゴリズムを実装できる可能性がある．

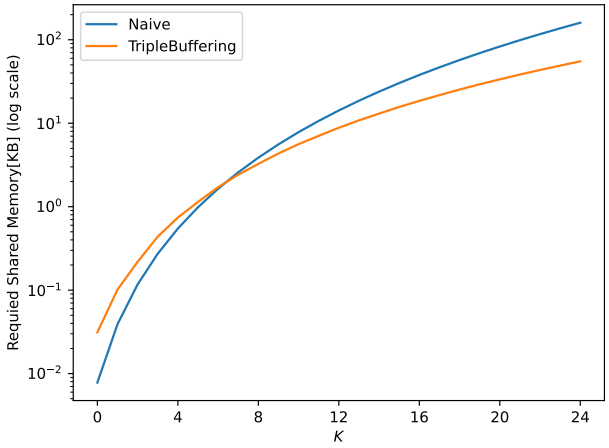

Fig. 2. Required shared memory

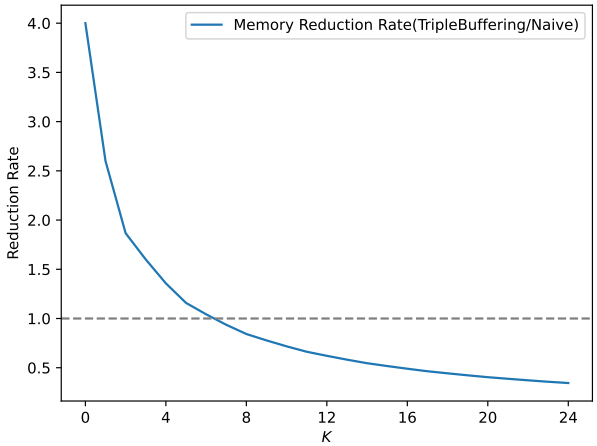

Fig. 3. Memory reduction using triple buffering over naive implementation

VII. 性能評価

本章では，提案アルゴリズムの性能評価を行う．本実験では，GPU として NVIDIA A100 Tensor Core GPU を，CPU として Intel Xeon Gold 6338 CPU をそれぞれ使用し，1B1CI と 1B1SP の ERI 計算時間をそれぞれ測定した．また，CUDA Block は 1B1CI では基底関数についての ERI の個数分，1B1SP では shell についての ERI の個数分起動し，各 Block 内の Thread 数は 256 とした．上記の条件で，次の 2 つの実験を行った．

実験 1：本実験では，単原子分子に関する ERI の実行時間を測定した．単原子分子としては水素原子を使用し，基底関数には Correlation-Consistent 基底関数セット [10] として定義される様々な基底関数のうち，cc-pVDZ から cc-pV6Z までの 5 種類を使用した．

実験 2：本実験では，多原子分子に関する実行時間の計測を行った．多原子分子としてベンゼン (C_6H_6)，ナフタレン ($C_{10}H_8$)，酸化銅 (CuO) を使用し，基底関数には STO-3G 基底関数セット [11] 及び 6-31G**基底関数セット [12] を使用した．本実験は，特に実際の分子系においてどれほどの高速化が可能であるかを検証することを主目的としている．

表 II に，実験 1 についての結果を示す．表 II 中の ERI の個数は，II 章で説明した通り，基底関数の個数を M とした場合は $\frac{M(M+1)(M^2+M+2)}{8}$ 個として算出している．結果より，単原子分子においては，1B1SP は 1B1CI よりも基本的に低速であるという結果が得られた．つまり，単原子分子で ERI 計算を行う場合，基本的には 1B1CI アルゴリズムを用いる方が高速であるという事になる．ここで，表 1 に示されている各 shell の個数に注目すると，各基底関数ごとに最も方位量子数の大きな軌道の個数が 1 となっている．例えば，cc-pVQZ では f-shell の個数が 1 つとなっている．この場合，最も計算コストの高い，4 つの f-shell からなる ERI は 1 つしか存在せず，1B1SP アルゴリズムの設計上 1Block で計算することになる．それに対し，1B1CI アルゴリズムの場合，最も方位量子数の大きな軌道の個数が 1 ではなく，例として cc-pVQZ では f 軌道の個数が 10 個である．そのため，4 つの f 軌道からなる ERI は $\frac{10(10+1)(10^2+10+2)}{8} = 1540$ 個存在している為，1B1CI アルゴリズムの設計上 1540 個の Block を割り当てて計算を行うことになる．その為，1B1SP は 1B1CI と比較し十分な並列度とならない計算部分が存在し，それが原因で 1B1CI よりも低速になっていると推察される．

また，表 II において，cc-pVQZ 基底関数セットを用いた場合の高速化率と比較し，cc-pV5Z, 6Z 基底関数セットを用いた場合の高速化率が高くなっている．これは，cc-pVQZ 基底関数セットの場合は shell に関する ERI の個数が少ないため GPU を用いた高速化効果があまり得られていないためであると推察される．

表 III に，実験 2 についての実験結果を示す．結果として，本論文での提案実装 (1B1SP) は，1B1CI と比較して最大 1.15 倍の高速化を達成した．また，実験結果から，すべてのケースにおいて 1.1 倍前後の高速化を達成していることが分かる．これは，提案実装 (1B1SP) は 1B1CI と比較し Boys 関数の評価回数を削減可能であるため，一定の効果を得られたという事を意味し，様々な分子系において効果を発揮する手法であることを示唆している．

実験 2 の結果である多原子分子における ERI 実行時間は，実験 1 の単原子分子におけるものと比較すると，全体的に高速化率が高くなっている．これには，大きく分けて 2 つの理由が考えられる．第一に，1B1SP の方が ERI の個数が多くなっているため，前述した 1B1SP アルゴリズムのデメリットである十分な並列度にならない場合がある点を緩和できている為高速になっていると考えられる．第二に，単原子分子と多原子分子では式 (8) で示した Boys 関数の評価において大きな違いがある．単原子分子の場合，全ての基底関数や shell が同一の軌道中心を持つ．その為，式 (7) 中の $|P - P'|$ が全て 0 となり，結果として Boys 関数の入力である T も全て 0 となる．その為，単原子分子の場合は ERI 評価が極めて低コストな計算となる為，shell についての ERI を行う最大の利点である Boys 関数の計算回数が抑えられるという点をほとんど活かせない．それに対し，多原子分子の場合，$|P - P'|$ のほとんどは 0 ではない値を取るため，Boys 関数の評価は高コストな計算となり，shell についての ERI を行うことによる Boys 関数評価回数が減る恩恵を大きく受けることが可能になる．

TABLE II
COMPUTATION TIME OF ERIS FOR MONATOMIC MOLECULES

		monatonic Hydrogen (H)				
Basis function set		cc-pVDZ	cc-pVTZ	cc-pVQZ	cc-pV5Z	cc-pV6Z
Basis Functions	s-orbitals	2	3	4	5	6
	p-orbitals	3	6	9	12	15
	d-orbitals	0	6	12	18	24
	f-orbitals	0	0	10	20	30
	g-orbitals	0	0	0	15	30
	h-orbitals	0	0	0	0	21
	ERIs	120	1,035	198,765	3,088,855	32,012,001
Shells	s-shells	5	7	9	12	15
	p-shells	1	2	3	4	5
	d-shells	0	1	2	3	4
	f-shells	0	0	1	2	3
	g-shells	0	0	0	1	2
	h-shells	0	0	0	0	1
	ERIs	231	1,540	7,260	32,131	108,345
Computation time [s]	1B1CI	0.001	0.004	0.054	0.976	14.874
	1B1SP	0.001	0.020	0.294	3.742	38.545
	Speed-up	1.36	0.23	0.18	0.26	0.38

TABLE III
COMPUTATION TIME OF ERIS FOR POLYATOMIC MOLECULES

		Benzene (C_6H_6)		Naphthalene ($C_{10}H_8$)		Copper oxide (CuO)	
	Basis function set	STO-3G	6-31G**	STO-3G	6-31G**	STO-3G	6-31G**
Basis Functions	s-orbitals	18	30	28	46	6	8
	p-orbitals	18	54	30	84	12	18
	d-orbitals	0	36	0	60	6	18
	f-orbitals	0	0	0	0	0	10
	ERIs	222,111	26,357,430	1,464,616	164,629,585	45,150	1,103,355
Shells	s-shells	54	84	84	132	18	32
	p-shells	18	30	30	48	12	20
	d-shells	0	6	0	10	3	5
	f-shells	0	0	0	0	0	1
	ERIs	3,454,506	26,357,430	21,487,290	164,629,585	157,641	1,464,616
Computation time [s]	1B1CI	1.590	26.018	10.364	166.790	0.452	5.261
	1B1SP	1.427	22.451	9.444	145.868	0.395	4.732
	Speed-up	1.11	1.15	1.09	1.14	1.14	1.11

VIII. まとめ

本論文では，以前の我々の研究で提案した MD アルゴリズムを行うための効率の良い GPU 実装について，以前は基底関数についての ERI に 1 つの CUDA Block を割り当てていたところを，shell についての ERI に 1 つの CUDA Block を割り当てる形で拡張した．それぞれの ERI 計算の計算時間を計測したところ，多原子分子での実行において最大 1.15 倍の高速化を達成した．

REFERENCES

[1] L. E. McMurchie and E. R. Davidson, “One- and two-electron integrals over Cartesian Gaussian functions,” *Journal of Computational Physics*, vol. 26, no. 2, pp. 218–231, 1978.

[2] S. Obara and A. Saika, “Efficient recursive computation of molecular integrals over Cartesian Gaussian functions,” *The Journal of Chemical Physics*, vol. 84, no. 7, pp. 3963–3974, 04 1986.

[3] B. P. Pritchard, D. Altarawy, B. Didier, T. D. Gibsom, and T. L. Windus, “A new basis set exchange: An open, up-to-date resource for the molecular sciences community,” *J. Chem. Inf. Model.*, vol. 59, pp. 4814–4820, 2019.

[4] D. Feller, “The role of databases in support of computational chemistry calculations,” *J. Comput. Chem.*, vol. 17, pp. 1571–1586, 1996.

[5] K. L. Schuchardt, B. T. Didier, T. Elsethagen, L. Sun, V. Gurumoorthi, J. Chase, J. Li, and T. L. Windus, “Basis set exchange: A community database for computational sciences,” *J. Chem. Inf. Model.*, vol. 47, pp. 1045–1052, 2007.

[6] S. Boys and A. C. Egerton, “Electronic wave functions-I. A general method of calculation for the stationary states of any molecular system,” *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 200, no. 1063, pp. 542–554, 1950.

[7] S. Tsuji, Y. Ito, K. Nakano, and A. Kasagi, “Efficient GPU-accelerated bulk evaluation of the Boys function for quantum chemistry,” in *2023 Eleventh International Symposium on Computing and Networking (CANDAR)*, 2023, pp. 49–58.

[8] I. S. Ufimtsev and T. J. Martínez, “Quantum chemistry on graphical processing units. 1. Strategies for two-electron integral evaluation,” *Journal of Chemical Theory and Computation*, vol. 4, no. 2, pp. 222–231, 2008, pMID: 26620654.

[9] H. Fujii, Y. Ito, N. Yokogawa, K. Suzuki, S. Tsuji, K. Nakano, and A. Kasagi, “A GPU implementation of McMurchie-Davidson algorithm for two-electron repulsion integral computation,” in *in Proceedings of the 15th International Conference on Parallel Processing and Applied Mathematics (to appear)*.

[10] J. Dunning, Thom H., “Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen,” *The Journal of Chemical Physics*, vol. 90, no. 2, pp. 1007–1023, 01 1989.

[11] W. J. Hehre, R. F. Stewart, and J. A. Pople, “Self-consistent molecular-orbital methods. I. Use of Gaussian expansions of Slater – type atomic orbitals,” *The Journal of Chemical Physics*, vol. 51, no. 6, pp. 2657–2664, 09 1969.

[12] V. A. Rassolov, J. A. Pople, M. A. Ratner, and T. L. Windus, “6-31g* basis set for atoms K through Zn,” *J. Chem. Phys.*, vol. 109, pp. 1223–1229, 1998.