

通信アルゴリズムの数式解説 (comm.cpp)

Open DEM Japan

2025年6月30日

MPI を用いて並列離散要素法 (DEM) を実行する際、本ファイルは**空間分割** + **通信バッファ管理** + **リング通信** の 3 機能を提供する。以下ではコードを抽象化し、アルゴリズムの背後にある数学的構造を順に導出する。

(1) プロセッサグリッドの決定——仮に全 MPI プロセス数を P 、その直交分割を (p_x, p_y, p_z) とすると

$$P = p_x p_y p_z, \quad (1)$$

が必要条件である。入力で特定軸の値が “*” であればコードは残りの値と P から自動決定する。NUMA / TWOLEVEL モードでは同一ノード内の**コアグリッド** (c_x, c_y, c_z) を追加で満たす

$$p_x = k_x c_x, \quad p_y = k_y c_y, \quad p_z = k_z c_z, \quad (2)$$

($k_\alpha \in \mathbb{Z}_{>0}$) も構築する。

(2) 一様分割における座標 → プロセス写像——直交箱の全長 $\mathbf{L} = (L_x, L_y, L_z)$ 、下端 \mathbf{x}_{lo} 、粒子座標 \mathbf{x} に対し

$$i_x = \lfloor p_x \frac{x - x_{lo,x}}{L_x} \rfloor, \quad i_y = \lfloor p_y \frac{y - x_{lo,y}}{L_y} \rfloor, \quad i_z = \lfloor p_z \frac{z - x_{lo,z}}{L_z} \rfloor, \quad (3)$$
$$\mathbf{i} = (i_x, i_y, i_z) \quad (0 \leq i_\alpha < p_\alpha)$$

を求め、三次元配列 $\text{grid2proc}[i_x][i_y][i_z] \in \{0, \dots, P-1\}$ で

$$\text{proc} = \text{grid2proc}(\mathbf{i}) \quad (4)$$

とする。三斜格子 (triclinic) では \mathbf{x} はラメダ座標 $\boldsymbol{\lambda}$ に変換済みであり、式 (3) はそのまま適用できる。

(3) 非一様分割と 1D 二分探索——軸 α の分割点 配列 $\{s_\alpha^j\}_{j=0}^{p_\alpha}$ によりプロセス境界を記述するとき、正規化座標 $\xi_\alpha = (x_\alpha - x_{lo,\alpha})/L_\alpha$ あるいは λ_α を入力として

$$i_\alpha = \max\{j \mid s_\alpha^j \leq \xi_\alpha\}, \quad (5)$$

を二分探索で得る (コードの `binary()`)。計算量は $O(\log p_\alpha)$ 。

(4) ゴースト幅の設定——相互作用半径の最大値を

$$r_{\max} = \begin{cases} \max_{i,j} r_{ij} & (\text{single モード}) \\ \max_{t \in \text{型}} r_t & (\text{multi モード}) \end{cases}$$

とし, ユーザ指定値 r_{user} が存在すれば

$$r_g = \max(r_{\text{max}}, r_{\text{user}}). \quad (6)$$

速度も同期する場合は余分に

$$d_{\text{vel}} = \text{size_velocity}$$

の通信成分が付加される.

(5) 通信バッファの動的伸長—送信バイト数 N が現在容量 M_{send} を超えるとき,

$$M_{\text{send}} \leftarrow \lceil \alpha N \rceil + N_{\text{extra}}, \quad \alpha = 1.5, N_{\text{extra}} = 1000, \quad (7)$$

で再確保を行う (`grow_send()`). 受信側も同様で $M_{\text{recv}} \leftarrow \lceil \alpha N \rceil$.

(6) リング通信アルゴリズム—全プロセスを $\{0, \dots, P-1\}$ とし, プロセス p が持つバッファ $b_p^{(0)}$ の総サイズ N を前節の規則で揃えてから

$$b_p^{(k+1)} = \begin{cases} b_{p-1}^{(k)} & \text{if } p \neq (p+1) \bmod P, \\ b_p^{(k)} & \text{otherwise,} \end{cases} \quad k = 0, 1, \dots, P-1, \quad (8)$$

を非ブロッキング受信 & 同期送信で回す. 各ステップでコールバック f を適用し, $k = P-1$ 到達後に必要なら最終更新を呼び出し側へ返す (`ring()`).

(7) プロセッサレイアウトの変更伝播—動的ロードバランスなどで

$$\text{layout} \in \{\text{UNIFORM}, \text{NONUNIFORM}, \text{TILED}\}$$

が変更されると, サブコミュニケータ $\{\text{Comm}_{\text{sub}}\}$ 全体に再帰的に新値を伝搬させる. 同様に分割境界 m_{split} , RCB (二分割再帰切断) パラメータ (`rcbnew`, `rcbcutfrac`, `rcbcutdim`) についても再帰伝搬が定義される.

(8) I/O 補助関数—プロセス 0 によるファイル読み込みは

$$\text{MPI_Bcast} : (m, l), \quad (9)$$

で他プロセスへ伝搬し, `fgets` が早期 EOF を返せば読み込みエラー (戻り値 1) とする.

以上により, `comm.cpp` は空間分解パラメータ (1)–(5), 通信領域幅 (6), バッファ管理則 (7), リング通信手順 (8), およびそれらの再設定伝搬を数学的に体系化した実装である. MPI の逐次一貫性と動的メモリ確保を保証しつつ, 粒子法に不可欠なゴーストセル同期を高効率に実現している.