JAXA 内製 MPS 法プログラム P-Flow による大規模流体解析

宮 島 敬 明

理化学研究所　計算科学研究センター

１．はじめに

JAXA では、次期国産旅客機の開発に資する大規模流体解析プログラム P-Flow の研究開発を行っている。P-Flow は Moving Particle Semi-Implicit(MPS)法をベースに、水などの大変形を伴う非圧縮性流体を解析対象にしている。MPS 法は粒子系シミュレーションに分類され、計算対象を多数の仮想粒子として分割し、各粒子と近傍粒子との相互作用から物理量の計算を行う。P-Flow は大規模解析に対応すべく、近傍粒子の探索処理をスレッド並列化し、計算領域を複数プロセスに動的に分割して処理時間の短縮を図っている。本年度は、実際の航空機への応用を念頭に、多数ノードにおける P-Flow の適用可能性を検証した。

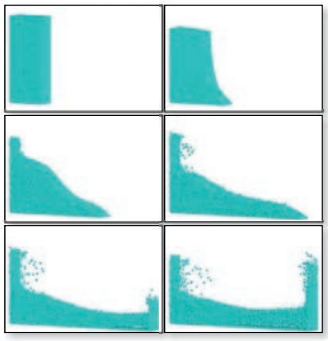


図 1:シミュレーション例

２．研究背景

航空機が着陸時に滑走路の終端を越えてしまうことをオーバーランと言う。その主要因の一つとして、滑走路にある水たまりが航空機の脚とタイヤに悪影響を及ぼすことが挙げられる。大変形を伴う水たまりと航空機の相互作用の数値解析は、計算領域を事前に格子で分割し、高精度に解析を行う有限要素法をベースとしたソルバーでは困難であったため、前例がない。そのため、実際の機体を用いた実験によってしか影響を知ることができず時間的・金銭的なコストが非常に大きかった。我々は、これら問題を解決すべく、MPS 法をベースとした大規模流体解析プログラム"P-Flow"の研究開発を行っている。

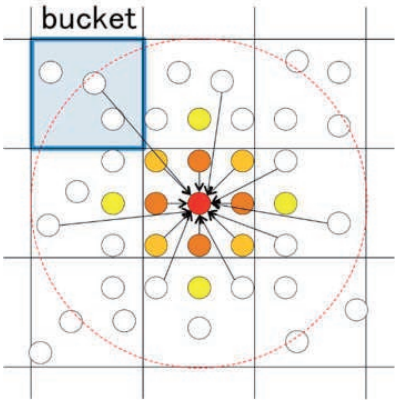


図 2:　バケット法を用いた近傍粒子探索

P-Flow のベースとなっている陽解法 MPS 法は、粒子系シミュレーションの 1 つであり、水などの非圧縮性流体を多数の仮想粒子に分割し、各粒子と近傍粒子との相互作用から物理量の計算を行う。大規模解析と計算精度向上のためには粒子の個数を増やす必要があるが、粒子数に比例して近傍粒子を探索する処理（近傍粒子探索）も所要時間が大幅に増加する。P-Flow は、図 2 に示すように近傍粒子探索にバケット法を採用した上でスレッド並列化し、計算領域を複数プロセスに動的に分割して処理時間の短縮を図っている。Reedbush-H を用いたこれまでの研究で、GPU 化と領域分割により近傍粒子探索は高速化されたが、動的領域分割とそれに伴う不規則な通信がボトルネックとなることがわかっている。

３．本年度の目標

本年度は以下の二つの問題の評価を行い、原因を探った。

（ア） 多数ノードにおける P-Flow の通信の挙動解析

P-Flow は ParMETIS を用いて、各プロセスの粒子が均等になるような負荷分散を行っている。右図は水中崩壊問題を 8MPI での負荷分散を示したもので、全粒子が縦方向に 8 つに区切られていることがわかる。通信が問題となる原因を特定する。

（イ） 大規模並列実行の実用化への課題の洗い出し

P-Flow は初期化の際に、シングルスレッドで粒子を生成している。このため、初期化に無視できない非常に長い時間がかかってしまうことがわかっている。これの原因と具体的なコードの場所を特定する。

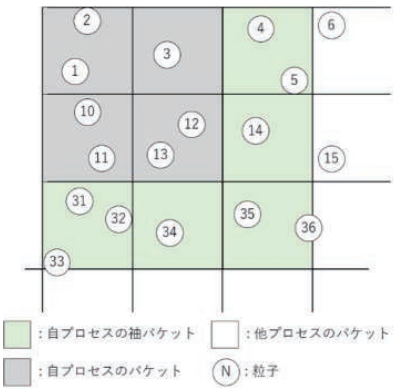


図 3: 通信パターン
粒子を持つ袖バケットがプロセス間通信を発生させる

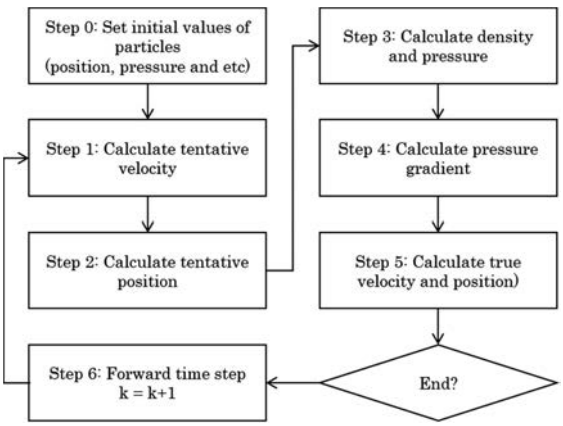
４．P-Flow

P-Flow は大規模シミュレーションを目的に研究開発されており、陽解法の MPS 法（Explicit MPS）を採用している。本節では、陽解法の MPS 法と複数ノードに並列化する際の通信手法について概略を述べる。

４．１ 支配方程式

MPS 法は以下の計算ステップを目的のシミュレーション時間に到達するまで繰り返す。なお、 $r_j - r_i$ は対象粒子 i と近傍粒子 j との距離計算を示す。

・Step 0）計算領域の初期値を設定
・Step 1）仮速度の計算
・Step 2）仮位置を計算
・Step 3）粒子数密度と圧力の計算
・Step 4）圧力勾配の計算
・Step 5）粒子の位置を計算
・Step 6）次の時間ステップへ（Step 1～6 を繰り返す）



バケット法を用いた近傍粒子探

以下に各計算ステップの概要を示す。

Step 0）シミュレーションの初期値を設定

MPS 法では初期値として、近傍粒子の距離の重み平均λ^0と初期の粒子数密度n^0 を求める。

$$\lambda^0 = \frac{\sum_{j \neq i'} (|r_j^0 - r_{i'}^0|)^2 \omega(|r_j^0 - r_{i'}^0|)}{\sum_{j \neq i'} \omega(|r_j^0 - r_{i'}^0|)}$$

$$n^0 = \sum_{j \neq i'} \omega(|r_j^0 - r_{i'}^0|)$$

Step 1）仮速度の計算

計算ループでは、まず下式を用いて各粒子の仮速度を求める。右辺 第 2 項と第 3 項はそれぞれ粘性と重力である。

$$u_i^* = u_i^k + \Delta t \left\{ \left(\nu \frac{2d}{\lambda^0 n^0} \sum_{j \neq i} (|u_j^k - u_{i'}^k|) \ \omega(|r_j - r_i|) \right) + g \right\}$$

ただし、 k はタイムステップ、 t は実時間、 i と j は粒子の番号、 ν は動粘性率、 d はシミュレーションの次元数、 g は重力加速度、 u_i^kは時刻 k での粒子 i の速度、 u_i^* は時刻 kでの粒子 i の仮速度である。

Step 2)仮位置を計算

続いて、 Step 1 で得られた仮速度を用いて各粒子の仮位置を求める。

$$r_i^* = r_i^k + \Delta t \, u_i^*$$

ただし、 r_i^k は時刻 k での粒子 i の位置である。

Step 3）粒子数密度と圧力の計算

その後、各粒子の次のタイムステップの圧力を求める。

$$n_i^* = \sum_{j \neq i} \omega(|r_j^* - r_{i'}^*|)$$

$$P_i^{k+1} = c^2 \frac{\rho^0}{n^0} (n_i^* - n^0)$$

ただし、 Pik+1 は時刻 k での粒子 i の圧力、 c は音速、 n_i^*は時刻 k での粒子 i の仮の粒子数密度である。

Step 4）圧力勾配の計算

そして、各粒子の圧力から圧力勾配を求める。

$$\langle \nabla P \rangle_i^{k+1} = \frac{d}{n^0} \sum_{j \neq i} \left(\frac{(P_j^{k+1} - P_i^{k+1})(r_j - r_{i'})}{|r_j - r_i|^2} \omega_{grad}(|r_j - r_i|) \right)$$

ただし、 ω_{grad}は後述する重み関数である。

Step 5）粒子の位置を計算

最後に、次のステップの最終的な速度と位置を求める。

$$u_i^{k+1} = u_i^* - \Delta t \left(\frac{1}{\rho} \nabla P \right)_i^{k+1}$$

$$r_i^{k+1} = r_i^* - \Delta t \left(\frac{1}{\rho} \nabla P \right)_i^{k+1}$$

ただし、 u_i^{k+1} と r_i^{k+1}は時刻 k + 1 での粒子 i の粒子の速度と位置である。

Step 6）次の時間ステップへ

次のタイムステップの計算を開始するために、粒子の最大速度uから実時間Δtを求める。

４．２ 複数ノード時の通信

前述の様に P-Flow では計算領域がバケットという座標で区切られており、P-Flow のノード並列化には粒子の位置と速度が変化した場合に物理量のやり取りが必要となる。各ステップで通信が必要となる物理量は以下の通りであり、7 回の通信が必要となる。

・Step 1)自プロセスのバケットから袖バケットへ移動した粒子の仮速度
・Step 2)自プロセスのバケットから袖バケットへ移動した粒子の仮位置，袖バケットに残った粒子の位置
・Step 3)袖バケットに残った粒子の圧力
・Step 5)自プロセスのバケットから袖バケットへ移動した粒子の位置と速度，袖バケットに残った粒子の位置
粒子系シミュレーションでは、各タイムステップで各バケット内部の粒子が移動するため、通信すべき粒子の数と通信先が常に異なる。P-Flow の現在の実装では、通信すべき粒子を順に通信バッファにコピーした後に、mpi_alltoall　で全プロセスから自プロセスへの通信量を求める。その後、各プロセスが全プロセスに対し mpi_{isend,irecv}で実際の通信を行う。このため、プロセス数 n の二乗で通信が発生してしまう。しかし、P-Flow は隣接するバケットにのみ粒子が移動するように設計されているため、隣接するプロセスへの通信のみを考慮すればよく、mpi_alltoall と全プロセスへの通信は不必要である。今回は、現在の実装について通信時間と通信パターンを理解することとした。

５．実験結果

ここからは、本年度得られた実験結果とその原因について述べる。

５．１　評価環境

P-Flow は以下の条件でコンパイルされ、Reedbush-H 上で測定を行った。対象問題は、水中崩壊問題である。なお、P-Flow は Fortran で記述され、OpenACC で GPU 化が行われている。

pgfortran 17.10-0 64-bit target on x86-64 Linux -tp haswell

mpif90 -tp haswell -Mpreprocess -Minfo=accel,mp -acc -O3 -ta=nvidia:cc60,fastmath,ptxinfo

５．２　評価結果

（ア） 多数ノードにおける P-Flow の通信の挙動解析

trans_x の通信パターンについて調べる。前述の通り、対象問題の実サイズは固定であるが、粒子の系が小さくなり、他の領域と接する面（＝袖バケット）に所属する粒子数が増加する。2MPI の場合、各タイムステップで各プロセスが通信する粒子の数は平均で 4 万個であった。しかし、これが 4MPI では 6 万個、8MPI では 11 万個と増加していた。GPU を使った今回の実装の様に、各プロセス担当するが CPU より多い実装の場合、負荷分散と分割の方法はよりシビアになると考えられる。これは、動的領域分割に利用している ParMETIS のプロセス番号の付け方によるものが大きいと考えられる。これは、分割前と分割後について領域が同一でも、プロセス番

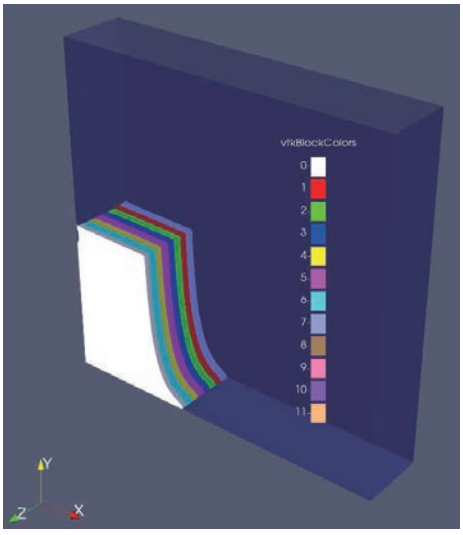


図 4: 8 プロセス時の負荷分散の様子

号が違えば粒子の移動が発生してしまうためである。

（イ） 大規模並列実行の実用化への課題の洗い出し

16GPU で 7,620,480 個の粒子の初期化（生成）に 1 時間以上かかってしまったために、スケーリングの測定が充分にできなかった。また、コードの見通しをよくするために、一時配列を随時 allocate していたが、これに予想以上に時間がかかっていたので、こちらの修正を行った。その他、OpenACC では変数への代入など簡単な逐次処理をすべて CPU で行わなければならないため、データのやり取りが発生してしまう点も問題であったので、最新のコンパイラのテストを行いたい。

６．まとめ

本年度は、通信の挙動解析と大規模化した際の課題の洗い出しを行った。それぞれの評価を行い、原因を特定した。前者の解決には領域分割手法の改良、後者の解決には動的配列の取得方法の変更などのテクニックを利用して改良することで対処できると考えられる。これらの解決と手法の提案は今後の課題とする。