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 HPC チャレンジでは、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。

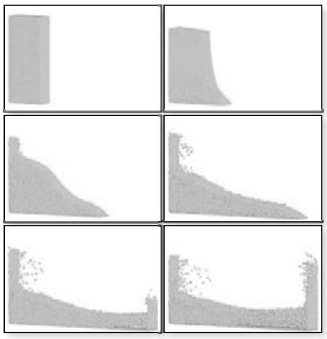


図 1:シミュレーション例

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。

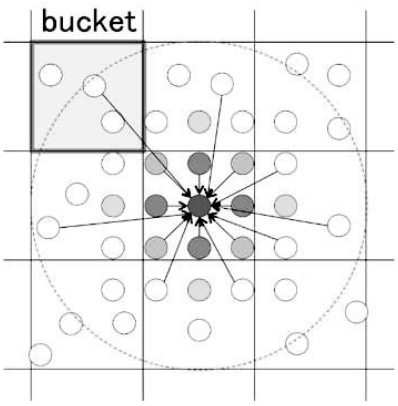


図 2: バケット法を用いた近傍粒子探索

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た。

３．本 HPC チャレンジでの目標

今回のチャレンジでは、多数ノードにおける P-Flow の適用可能性を検証すべく、以下の 3 つの課題について取り組んだ。

A) 弱スケーリングの測定

前述の水たまりと航空機の脚とタイヤにシミュレーションに求められる計算領域と精度を担保するには、数千万から 1 億粒子程度の大規模解析を行う必要がある。室谷ら[i]は津波の解析に MPS 法を用いており、理研の京コンピュータ 12,000 ノード(1.5P FLOPS)で 1.3 億粒子の解析を行っている。Reedbush-H の計算能力は、120 ノード使用時に 1.4P FLOPS であるため、同様の計算ができると考えた。今回は対象問題として、MPS 法において一般的なベンチマーク問題である水柱崩壊問題を対象に P-Flow の弱スケーリングの測定を行った。具体的には、総粒子数を 476,190（1GPU=1 プロセス）から 60,952,380（128GPU=128 プロセス）に変化させ、P-Flow の計算上のボトルネックの一つである、粒子密度の計算にかかる所要時間を計測した。

B) 多数ノードにおける P-Flow の通信の挙動解析

MPS 法は計算対象によって動的領域分割と通信パターンが大きく異なるため、実問題で大規模解析を行うことが今後の研究と実用化にあたり非常に重要である。P-Flow は計算領域をバケットという座標で区切り、物理的条件を用いて各タイムステップで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。(図 1) 今回は、課題 A において実際にどのような通信が発生するかの実測値を得ることを目標とした。P-Flow は 1 プロセスが 1GPU を占有する設計であるため、Reedbush-H を 120 ノード使用した場合には 240 プロセスでの動的領域分割とそれに伴う通信を行う。1 億粒子の解析の場合、1 プロセスが 40 万粒子程度（=1 億÷240）を担うことになり、ノード単体としての計算負荷とメモリ使用量は適切であると予想できるが、通信に関しては予想がしにくく、実データが必要である。

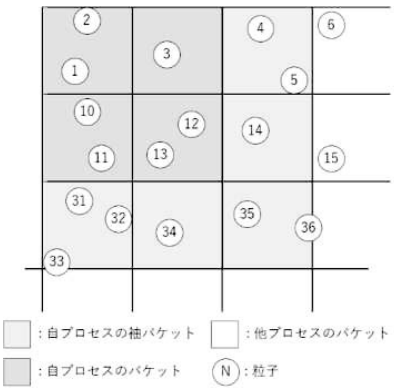


図 3: 通信パターン
粒子を持つ袖バケットがプロセス間通信を発生させる

C) 大規模並列実行の実用化への課題の洗い出し

P-Flow は、次期国産旅客機の開発に資することを目標としており、ユーザビリティの検討も行う必要がある。大規模並列実行時の実際のユースケースを模擬し、どの様な挙動（エラーや実行時間の内訳）を示すかを検証した。

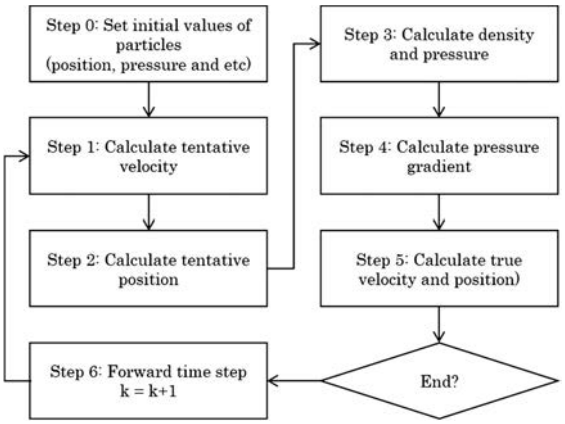
４．P-Flow

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。

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

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 HPC チャレンジで得られた実験結果について述べる。

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

A)　弱スケーリングの測定

1 プロセス（=1GPU）に 45 万程度の粒子を担当させ、2,4,8,16,32,64 プロセスでの弱スケーリングを計測した。対象問題の実サイズは固定であるため、粒子の系が小さくなり計算精度が上がることに相当する。プロセス番号 1 における処理時間の推移と左端に関数名を以下の表に示す。（関数についての詳細は省く。）全体の処理時間のである iteration は、2→64 プロセスで処理時間が 30.3 倍に増加してしまい、ほぼプロセス数に比例して処理時間が増加してしまった。

増加の主な原因は、赤字で書かれた通信を含む処理(xu_pre, trans_x, trans_p, xu_post_explicit, trans_xu)である。最下段の mpi_alltoall は全体の平均をとったもの記す。ほぼ通信時間で処理が占められている trans_x については、プロセス数の増加の 1/3 程度の比率で増加していた。しかし、xu_pre と xu_post_explicit については、2→64 プロセスで処理時間が 100 倍以上に増加してしまい、プロセス数の増加を大きく超えてしまった。処理時間の増加の原因の詳細については今後の課題とする。

# of MPI (=GPUs)	2	4	8	16	32	64
# of total particles	882,090	1,852,200	3,936,600	7,620,480	11,863,800	31,492,800
# of particles per process	441,045	463,050	492,075	476,280	370,744	492,075
# of buckets	110x110x22	140x140x28	180x180x36	225x225x45	260x260x52	360x360x72
# of iterations	4237	2840	1414	1402	913	257
timestep:	0.4	0.4	0.6	2.1	1.2	20.7
apportion_bkt:	0.0	0.0	0.0	0.0	0.1	0.1
Laplacian_u:	599.9	190.2	154.3	168.2	140.1	260.4
Surface_tension_f:	31.3	33.5	30.0	35.9	23.2	38.0
Isolated_particle:	1.7	1.8	1.9	1.8	1.5	1.9
xu_pre:	323.6	1312.0	3024.1	6350.7	10441.4	37280.3
index:	275.0	292.6	280.6	311.6	219.1	319.4
trans_x:	98.6	170.2	334.0	379.8	574.8	1124.2
cal_nden:	383.2	79.1	58.5	65.6	45.9	72.5
update_p_explicit:	33.3	35.0	32.0	37.4	24.0	38.3
trans_p:	191.5	778.2	1906.7	3849.1	6575.6	23317.4
grad_p_explicit:	635.0	161.3	137.2	139.7	111.3	190.2
xu_post_explicit:	288.0	1329.9	3084.1	6339.9	10794.9	37640.8
trans_xu:	177.9	263.5	485.7	551.8	812.5	1714.0
elastic_collision:	2.1	2.0	2.5	2.4	1.7	2.0
iteration:	3369.6	5002.4	9886.0	18603.4	30037.2	102408.0
mpi_alltoall*	248.8	1251.7	2960.9	6187.3	10428.7	36930.8

表 1: プロセス数と処理時間の推移

B)　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。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。

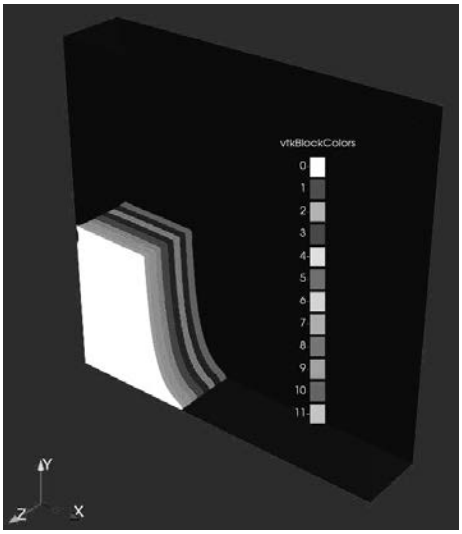


図 4: 8 プロセス時の負荷分散の様子

C)　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た。例えば、128GPU で 60,952,380 個の粒子の初期化（生成）に 2 時間半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。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。

６．まとめ

本 HPC チャレンジでは、弱スケーリングを中心に P-Flow の大規模並列化のフィージビリティ・スタディを行った。弱スケーリングでは、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るため、通信すべき粒子が増加することと実測値をとることができた。また、通信の方法に全プロセスを対象とするやりとり（mpi_alltoall+mpi_{isend,irecv}）を利用したことで、プロセスの増加を超えてて通信時間が増加することと実測値をとることができた。前者の解決には領域分割手法の改良、後者の解決には隣接プロセスのみを対象とした通信を行う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。

---

[i] K. Murotani, S. Koshizuka, M. Ogino and R. Shioya, ″Development of Explicit Moving Particle Simulation Framework and Zoom-up Tsunami Analysis System″, SC15, Regular Poster, post131s1, Austin, November 15-20, 2015