「京」、HPCIにおける大規模並列化に向けたOpenFOAM高度化支援

OpenFOAM tuning towards massively parallelization at HPCI including the K computer

一般財団法人高度情報科学技術研究機構（RIST）
浅見　　暁、井上　義昭、青柳　哲雄

「京」を中核とするHPCI（High Performance Computing Infrastructure）の産業利用課題において、多く利用されているアプリケーションの一つが、流体シミュレーションコードOpenFOAMである。OpenFOAMは多種多様な流体解析のための標準ソルバーが実装されており、ライセンス費用が不要でカスタマイズ可能なオープンソースソフトウェアであることから、産業界では注目度が高く、多数の企業で利用されている。しかしながら、「京」における大規模な計算において、使用メモリ量、通信時間についての問題に直面し、OpenFOAM自体が、大規模並列での実行を想定していないことが判明した。例えば、当初1万プロセスでの実行がメモリ不足のため不可能であったが、改善により2万プロセス以上での実行を可能とした。また、約6500秒という極めて長い時間が初期処理に費やされていたが、通信処理の根本的な改善により、26秒まで短縮することができた。本稿では「京」の登録機関であるRISTが利用支援の一環として実施した移植支援、プリポスト支援、高速化支援の各内容について紹介する。また、支援で得た知見を利用者と共有するためのワークショップ開催や富士通株式会社コンパイラ開発部門への働きかけ、外部発表等についても紹介する。

1．はじめに［1］

表1は「京」やHPCIの産業利用課題で使用されているアプリケーションを課題数の順番で並べた一覧である。OpenFOAMは産業利用課題において2番目に多く利用されているアプリケーションであることが分かる。上位にランキングされるアプリケーションは、LAMMPSやOpenFOAMのようなオープンソースソフトウェア（OSS）や、FrontFlow/blueのような国家プロジェクトで開発されたアプリケーションである。いずれもライセンス費用が不要でカスタマイズが自由にできるところが企業にとっては魅力的であるが、オープンソースソフトウェアであるがゆえ、基本的に移植や利用に関する疑問、バグ対応、カスタマイズなどは、自力ですべて解決しなければならない。

OpenFOAMを利用する企業に対して有償でサポートを実施しているベンダーは数社あるが、「京」を利用する際の実践的なノウハウ、例えばビルドや効率的なステージングを行う実行シェルスクリプトの作成、高速化のためのノウハウを持っているベンダーはほとんどなく、利用者にとって「京」の利用は高いハードルとなっていた。従って、OpenFOAMを重点的な支援対象アプリケーションとした。

表１　「京」産業利用で利用課題数の多いアプリケーション[i]

利用プログラム	課題数	利用企業数	分野	種別
LAMMPS	17	10	計算化学（古典MD）	OSS
OpenFOAM	14	8	流体解析	OSS
FrontFlow/blue	10	8	流体解析	国家PJ
FrontFlow/red	9	6	流体解析	国家PJ
LS-DYNA	8	4	衝突解析	商用
FrontISTR	5	3	構造解析	国家PJ
GROMACS	5	3	計算化学（古典MD）	OSS
VSOP	5	3	計算化学（粗視化MD）	商用
MODYLAS	5	2	計算化学（古典MD）	国家PJ
ELSES	3	2	計算化学（電子構造計算）	国家PJ
OpenMX	2	2	計算化学（DFT）	OSS

２．OpenFOAMとは［２］［３］［４］

OpenFOAMとはOpen source Field Operation And Manipulationの略で、C++で記述された数値流体ソルバー群である。ライセンスはGPL（General Public License）に準じており、The OpenFOAM Foundation（http://openfoam.org/）から、オープンソースソフトウェアとして公開されている。特徴として、非圧縮流れ、圧縮流れ、多相流、燃焼、化学反応を伴う流れなど、多種多様な流体解析に対応するソルバー（interFoamなど標準ソルバーの数は200個余り[ii]）が実装されている。また利用者の目的に合致した標準ソルバーが無い場合には、C++でソルバーを自分で作りこむことも可能である。多種多様な流体解析や機能に対応しているため、ソースコードの規模は約240万行に上っている。GPLライセンスの場合、ソフトウェアを改変して提供する場合には、そのソフトウェアのソースコードも提供しなければならないという制約があるが、企業や個人の限られた範囲で使用する場合にはその制約はなく、自由に利用可能である。

2016年６月現在OpenFOAMは、The OpenFOAM Foundationから公開されているものと、ESIグループのOpenCFD社から公開されているものの２種類ある。いずれもWEBからダウンロード可能となっている。OpenCFD社（http://openfoam.com/）では、有償でプログラムのカスタマイズ、サポート、コンサルティングを行っており、The OpenFOAM Foundationからリリースされているを公開している。また、OpenFOAMは年に２、３回のバージョンアップがなされ、機能強化、改善が行われている。

３．支援内容

OpenFOAMは2013年から支援を開始し、

i　2015年8月1日現在。H24年度～H27年度までの「京」産業利用課題（トライアル・ユースと実証利用）の採択課題の申請書から、利用企業数の多い上位のソフトウェア（インハウスを除く）を集計。OpenFOAMの件数はOpenFOAM関連のソフト（Helyx-Core, iconCFD）を含めた数。

ii　http://cfd.direct/openfoam/user-guide/standard-solvers/

これまで14件の高度化支援を実施してきた。表 2 に「京」、HPCI課題における2015年度のOpenFOAMの利用状況と高度化支援内容の一部を示す。

表 2　OpenFOAM利用状況と高度化支援内容（2015年10月現在）

課題利用者	システム	使用OpenFOAM	ソルバー	データサイズ（セル数）	並列数	高度化支援内容
課題A	「京」	OpenFOAM 2.2.x	pisoFoam	128億～1000億	49152～98304	性能分析、冗長メモリ量の削減
課題B	「京」	OpenFOAM2.3.1 Helyx_core2.2.1	interDyMFoam waveDyMFoam	1億 500万	512 240	ビルド支援、ステージングの効率化、性能分析、性能改善案の検討
課題C	「京」 九州大学†FX10	OpenFOAM2.3.1	interDyMFoam	1.5億	384	ビルド支援、リスタートシェル、経過時間終了機能追加
課題D	「京」	OpenFOAM2.3.1	pimpleDyMFoam	5000万	1536～6144	ビルド支援、通信時間の改善
課題E	統計数理研究所‡UV2000	OpenFOAM2.2.1	pimpleConcFoam	1700万	600	性能分析、ボトルネック調査、性能改善案の検討

†：九州大学情報基盤研究開発センター
‡：統計数理研究所統計科学技術センター

上記に示す高度化支援内容の詳細は次の通りであり、以降で説明する。

(1) 利便性向上のための支援
　1）ビルド用パッチファイルの作成
　2）ステージングシェルスクリプトの作成
　3）経過時間指定による終了指定パッチファイルの作成
　4）リスタート計算用スクリプトの作成
(2) アプリケーションの大規模化、高速化の支援
　1）冗長メモリ量の削減
　2）通信時間の削減
　3）コンパイルオプションの最適化

高度化支援を通じて作成されたビルド用パッチファイルやシェルスクリプトは、すべて利用者へ提供可能である。

3. 1.　利便性向上

3. 1. 1.　ビルド用パッチファイルの作成

OpenFOAMをビルドする際にg++やインテルコンパイラなど、ビルド用Makefile内で数種類のコンパイラの選択が可能であるが、当然ながら「京」向けのコンパイラの選択肢はなく、そのままではビルド不可能である。また、OpenFOAMのバージョンや「京」の言語環境により、コンパイルエラーが多発するため、エラーの修正作業のみでもかなりの時間を要する。利用者の作業負荷軽減のため、「京」向けのコンパイラやオプションの設定、ソースコードの修正も含んだパッチファイル（OpenFOAMのバージョン、言語環境ごとに準備）、およびビルド手順書を作成した。ちなみに「京」向けのビルドは、コンパイルエラーを全て取り除いた状態でも、およそ 2 日間を要する。

3. 1. 2.　ステージングシェルスクリプトの作成［5］

「京」でプログラムを実行する場合、ログイ

ンノードから必要な入力データファイルを計算ノードへ送ったり、計算結果ファイルを計算ノードからログインノードへ戻したりする必要がある。この作業をステージング（それぞれステージイン、ステージアウト）と呼んでいる。

OpenFOAMの並列実行の場合、計算結果のファイル数が数千から数万ファイルと非常に多くなる。そのため、ステージアウト可能なファイル数の上限を超えてしまうと、ステージアウトに失敗する。これを避けるためには、ステージアウトの前にtarコマンドでファイルをまとめ、ファイル数を少なくすることが有効である。大規模計算の際にはファイル数だけでなく格納されるデータ量も多くなるため、ファイルをまとめる処理に非常に時間を要していたが、圧縮形式を見直し、ランクディレクトリを使用してtarコマンドを並列に実行することにより、tarコマンドの実行時間を約1/85（約7300秒を86秒）にすることができた（図1）。

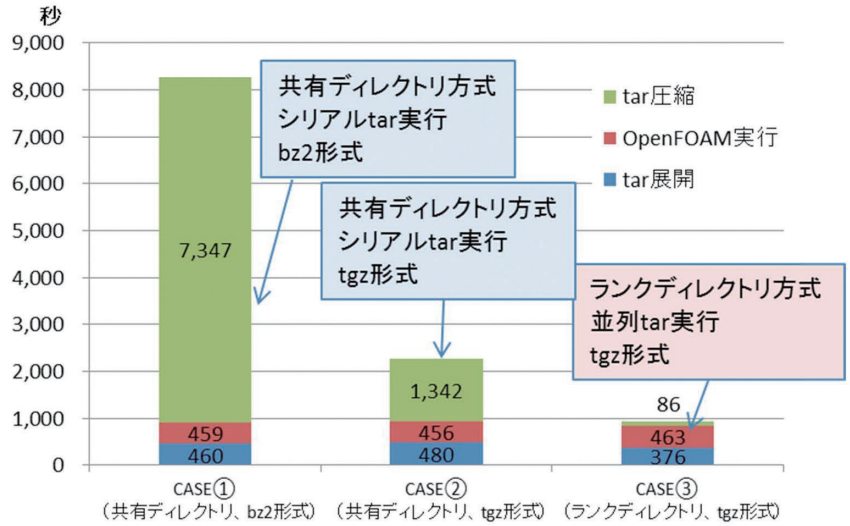

図1　ランクディレクトリと並列tarによる実行時間短縮（192プロセス）

3.1.3.　経過時間による終了指定パッチファイルの作成［6］

「京」では1ジョブの実行時間は最大で24時間という運用上の制限があるため、その中でジョブを終了させる必要がある。3.1.2のtarコマンド処理を含むプログラムの実行が24時間を超えてしまうジョブの場合、経過時間オーバーのエラーとなり、それまでの計算結果ファイルをステージアウトできない(計算結果が回収できない)恐れがある(図2の上部)。

OpenFOAMを24時間以内（あるいは指定時間内）に終了させたい場合には、計算ステップ数やシミュレーション時間で終了させる機能を用いて、小ステップ数での実行時間を参考に長時間ステップでの実行時間を見積もる必要があった。しかし、時間刻幅を自動調整する機能を使用した場合、見積もり通りとはならない場合もある。入力ファイル(controlDict)で指定した（希望終了）経過時間と実行開始からの経過時間をタイムステップ毎に比較し、指定した経過時間に到達していなければ次のタイムステップへ、指定した経過時間に到達すれば、次のタイムステップへは進まずに終了させる機能（図2の下部）を追加し、パッチファイルとして提供可能とした。このパッチファイルは「京」のようなファイルのステージングを必要とするシステムで有効である。

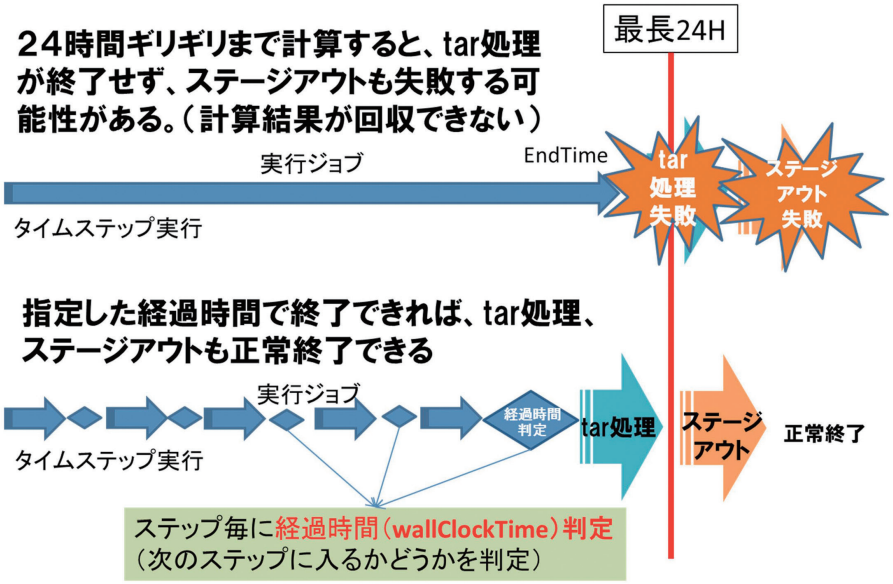

図 2　「京」での経過時間制限

3. 1. 4.　リスタート計算用スクリプトの作成［6］

OpenFOAMのリスタート計算は、コントロールファイルの他に、最終タイムスタンプディレクトリのデータのみ入力ファイルとしてステージインできれば、次の時刻からのリスタート計算が可能である。図 3 に示す全てのタイムスタンプディレクトリを含むres.tgzファイルをステージインした場合、リスタートに不要な古いタイムスタンプディレクトリを含むため、無駄なステージングの転送が発生してしまう。この無駄なステージングの転送時間削減のため、res.tgzファイルからリスタートに必要な最終タイムスタンプディレクトリのみを抽出し、リスタート計算時の入力データとして使えるようなシェルスクリプトを作成した。こちらも「京」のようなファイルのステージングを必要とするシステムで有効なシェルスクリプトとなっている。

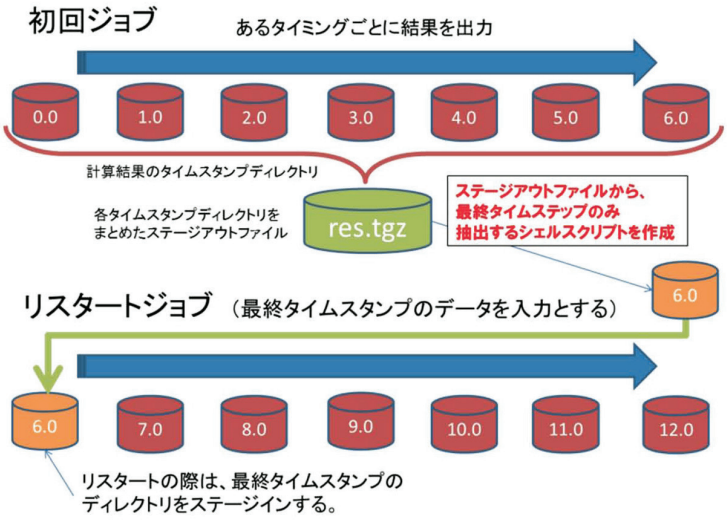

図 3　「京」での経過時間制限

3. 2.　アプリケーションの大規模化、高速化

3. 2. 1.　冗長メモリ量の削減［5］

「京」の1ノードあたりの搭載メモリ量は16Gバイトであり、そのうちプログラムで使用できるのはシステム領域を除いた約14Gバイトである。OpenFOAMの実行時[iii]の使用メモリ量を調査したところ、図4に示すように、ある程度の並列数まではMPIによる領域分割の効果で1ノード（8プロセス）あたりの使用メモリ量は減少していくが、192ノード（1536プロセス）を超えるあたりから増加に転じ、1536ノード（12288プロセス）を超えると、1ノードあたりのメモリ量を超え、異常終了することが判った。調査の結果、通信用の管理テーブル配列が冗長に確保されていたことが判り、必要な時に必要なだけ確保を行うよう改善をおこなった結果、2万プロセスを超える並列数での実行が可能となった。

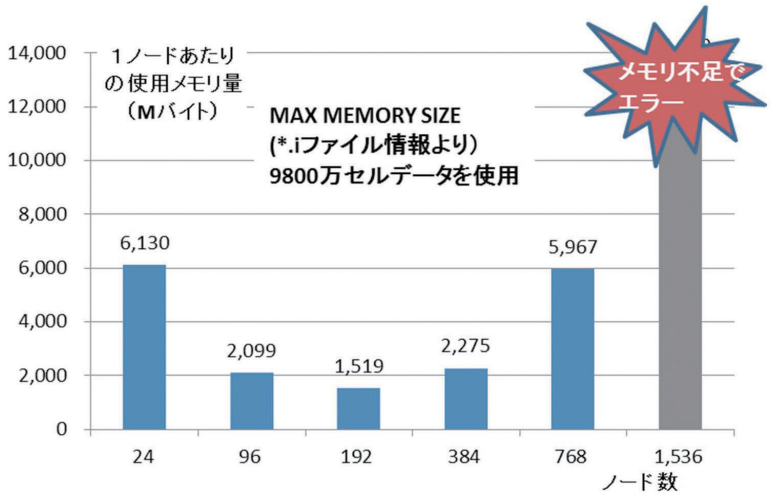

図4　1ノード（8プロセス）あたりの使用メモリ量

3. 2. 2.　通信時間の改善　［4］

OpenFOAMでは大規模並列実行時に、通信時間が性能のボトルネックとなった。特に集団通信については、MPIの集団通信（Alltoall）で実装できるところを、MPIの単一通信（Send/Recv）の繰り返しで実装されており、数千並列以上の大規模な並列数を想定したものにはなっていなかった。理由は定かではないが、開発時のMPIの集団通信の性能があまり良くなかった可能性がある。「京」のMPIの集団通信は最適化されているため、これらの関数を直接呼び出した方が高速な場合が多い。以下に事例を紹介する。

OpenFOAMに含まれるデータ交換を行うソースコードexchange.C（combineScatter/combineGather）では、全プロセスのテーブルの内容を集め、それを各プロセスへ転送するために、バイナリーツリー形式のMPI_Send/MPI_Recvを使用していた。図5に「京」の性能解析ツールfappの解析結果を示す。横軸はプロセスのランク番号、縦軸はMPI_probe通信の経過時間（通信の待ち時間）となっているが、ノコギリ状となっており、インバランスが大きいことが判った。分

iii　フラットMPIでの実行。1ノードに8プロセスを割り当て。

析の結果、図７の左図に示すように送信と受信のタイミングが適切でなく、通信完了までに長い時間を要していることが判った。（図５のMPI_probe時間の最大は約190秒となっている。）

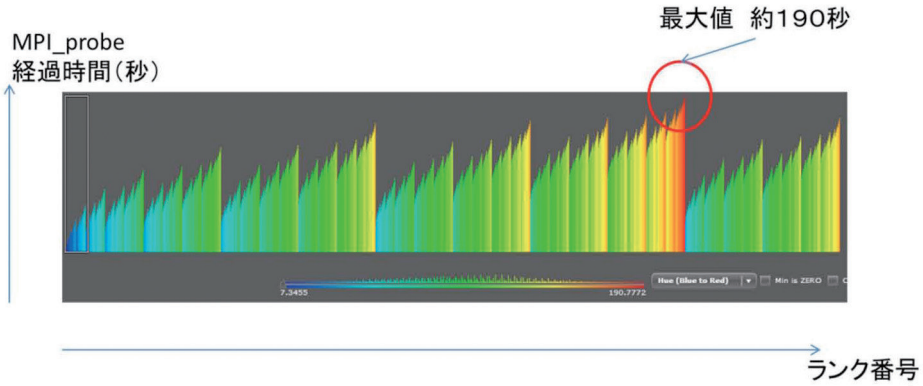

図５　MPI_probeのfapp MPI情報（1280プロセス）
（横軸はMPIランク番号、縦軸は時間（単位：秒）。カラーバー：赤色が最大値、青色が最小値を示す。）

通信を行うループの順番を逆順に変更（ループリバース）することにより、通信のタイミングが改善され、図６に示すようにMPI_probeの経過時間を約１/４（約190秒から約46秒へ）にすることができた。図７の右図に改善後のcombineScatterの通信イメージを示す。左図と比較して通信の順番が変わり、通信の効率が向上していることが判る。

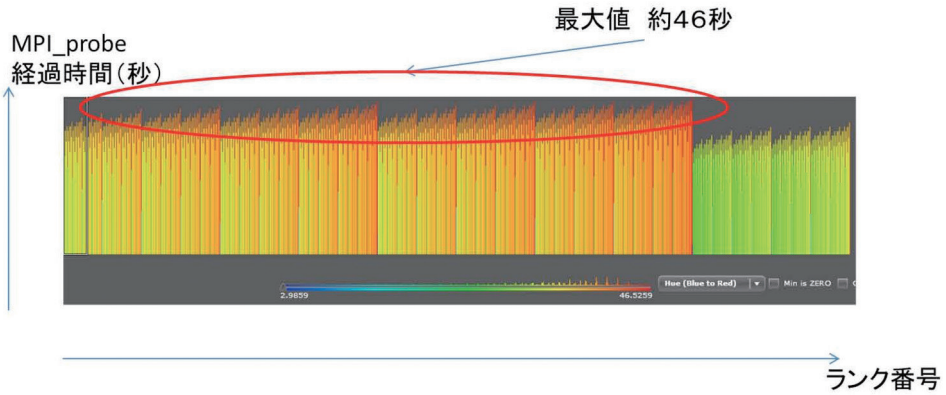

図６　改善後のMPI_probeのfapp MPI情報（1280プロセス）
（横軸はMPIランク番号、縦軸は時間（単位：秒）。カラーバー：赤色が最大値、青色が最小値を示す。）

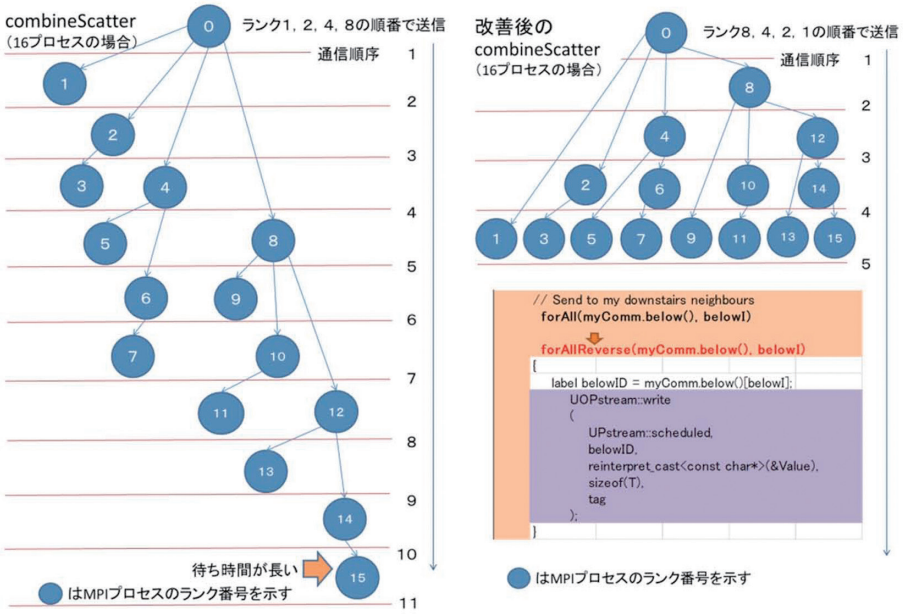

図 7　combineScatterの通信のイメージ

また、単一通信を用いた集団通信のパターンに着目し、MPI_Gather/MPI_Scatter関数やMPI_AlltoAll関数に置き換える修正を行った。この修正により、特に高並列実行時に爆発的に時間が増加してしまう初期処理時間（タイムステップループに入るまでの処理時間）を削減することができた。図 8 に初期処理（水色部分）、タイムステップループ部分（赤色部分）の測定結果を示す。オリジナルの状態では、約6500秒が初期処理に費やされていた。さらに並列数を増やして実行した場合、24時間でタイムステップループまで到達しなかったケースもあった。初期処理部分は 1 回のみの実行のため、多少時間がかかったとしても通常は無視してしまうことが多いが、 2 時間近く要するのでは、デバッグや実行結果の検証、計算機資源消費の観点から非常に効率が悪い。この初期処理部分を最終的には 1 /250（約6500秒から26秒）に短縮することができた。この通信部分の改善は、初期処理以外にタイムステップループの部分でも有効となっており、実行時間は図 8 （青色部分）より約 1 / 7 （447秒から65秒）となっている。実際の計算ではタイムステップループ部分が支配的になるため、約 7 倍程度の性能向上が得られると期待される。

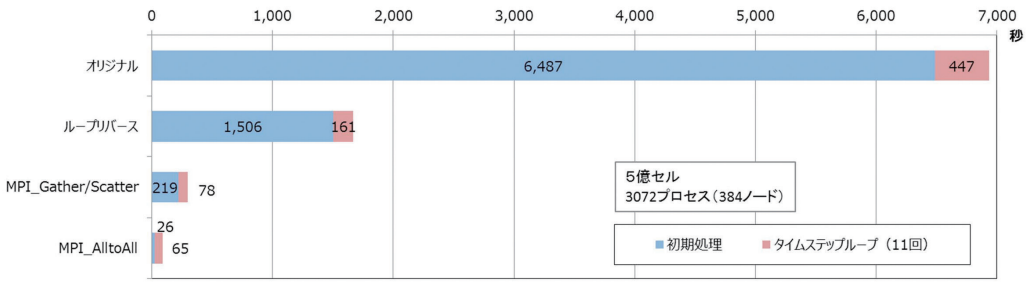

図 8　初期処理とタイムステップループの時間の推移

3. 2. 3.　コンパイルオプションの最適化［5］

「京」ではコンパイルオプションとして－Kfast[iv]の指定が推奨されているが、OpenFOAMの場合、このオプションを指定すると最適化の副作用のため、実行時エラーとなる。そこで、実行時間コストが大きく、実行時エラーが発生しない一部の演算関数に限定して－Kfastを指定して高速化を図り、またOpenFOAMのベクターマシン用のマクロを展開するオプションを指定するようにした。さらに富士通株式会社のコンパイラ開発部門から、－Kfastの最適化レベルに近く、副作用のない最適化オプションの情報を入手し、以降はこれらのオプションを各利用者へ提供している。

3. 2. 1冗長メモリ量の削減と、3. 2. 2通信時間の改善の修正（exchange.C）については、OpenCFD社へ情報提供を行い、同社で公開されているOpenFOAM+版へ反映されることになった。

4．その他の活動

4. 1.　OpenFOAMワークショップ［1］［5］［6］［7］［8］

OpenFOAMは「京」への移植、実行環境整備及び性能向上に多くの課題があり、「京」の産業利用に大きな障害となっていた。RISTでは、今まで述べてきたように1）利便性向上のための支援、2）アプリケーションの大規模化、高速化の支援を積極的に行ってきた。OpenFOAMの支援で得られた知見や利用ノウハウを利用者へ提供し、利用者間の情報交換の場として、OpenFOAMワークショップを開催している。OpenFOAMワークショップでは「京」やHPCIでの移植状況や高度化支援の紹介、産業利用課題参加企業の計算事例の紹介を中心に2013年10月に第1回目を開催、その後3年連続で毎年10月に開催しており、民間企業からの参加は、第1回43人 中38人（88％）、第 2 回65人 中53人（82％）、第3回84人中55人（65％）となっている。図9に2015年度に開催されたワークショップのアンケート結果の一部を示す。内容については、十分満足と満足で90％を超えていることが判った。民間企業からの参加が多数を占めている中で、「京」やHPCIでのOpenFOAMの大規模解析に向けた利用を促進し、受講者のスキルアップを通して産業界における計算科学研究の人材育成に貢献している。

iv　ターゲットマシン上で高速に実行するオブジェクトプログラムを作成するためのオプション。- O3とその他の最適化オプションを組み合わせたプリセットオプションとなる。

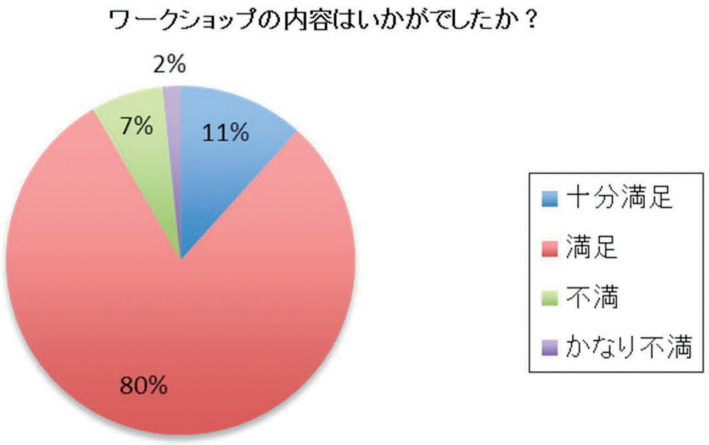

図 9　アンケート結果

4. 2.　富士通株式会社コンパイラ開発部門への働きかけ［9］

OpenFOAMの演算性能は、C++コンパイラの性能改善が寄与する部分が大きいため、富士通株式会社のコンパイラ開発部門とOpenFOAMの性能について情報交換や改善提案を行った。一例として、OpenFOAMの性能解析について挙げる。OpenFOAMをソース行番号に対応した性能解析を行うための－Nlineオプションでコンパイルを行うと、ライブラリのサイズが肥大化し、リンクや実行が不可能となった。OpenFOAM固有の問題であったが、これを富士通に伝えた結果、機能強化された評価用のコンパイラの提供を受けた。この評価用コンパイラの機能は、今後「京」の正式版コンパイラに反映される予定である。

4. 3.　外部発表（情報処理学会）［10］［11］

情報処理学会第151回ハイパフォーマンスコンピューティング研究会にて、清水建設株式会社、富士通株式会社と共著で「京」におけるOpenFOAMの性能向上について、下記の 2 件の発表を行った。

1）「京」コンピュータのTofu高機能バリア通信機能を活用して、データ型に合わせたテンプレートの追加や、OpenFOAM特有のPstreamBuffer全体データ交換形態を必要最小限の隣接データ交換形態に改良することにより、「京」において並列数の上限が6144であったところを、49152並列まで実行できるようにアプリケーション全体の実行効率を向上させた。

2）MPI_並列とThread並列を用いたHybrid並列の検討、および計算効率を向上させるための検討、及び試行を行った結果、MPI並列時と同等の性能を実現することができ、Hybrid並列化の有効性を示すことができた。

5．まとめ

高度化支援を通じてOpenFOAMの各種ソルバーの分析を行い、「京」やHPCIの計算機（FX10、UV2000）において、大規模実行をするための利便性の向上や、メモリ量の削減、通信方法の変更による性能向上を達成できた。OpenFOAMを含む高度化支援はこれまで108件実施しており、支援終了後のアンケートでは90％の利用者が満足と回答している。

今年度も高度化支援依頼を多数受けており、すでに新たな高速化への取り組みとし

て、対象物の変形に伴う移動メッシュや他の線形ソルバーについての分析、性能改善作業を開始している。今後ポスト「京」に向けてさらなる大規模での実行が可能となるよう、また利用者の期待に応えられるよう、引き続き支援を行っていきたいと考えている。

6．謝辞

高度化支援を行うにあたり、計算機リソースを提供していただいたHPCI課題参加企業各社に感謝する。また、「京」の運用を行っている理化学研究所計算科学研究機構を始め、九州大学情報基盤研究開発センター、統計数理研究所統計科学技術センターのスタッフの方々には、OpenFOAMを実行する際の計算機の利用方法などについて、多大なるご支援をいただいた。併せてここに深く感謝する。

7．参考文献

［1］ 新宮　哲（高度情報科学技術研究機構）：「京」を中核とするHPCI産業利用課題のご案内　第3回OpenFOAMワークショップ　2015年10月15日

［2］ http://www.geocities.co.jp/penguinitis2002/study/OpenFOAM/OpenFOAM-info.html

［3］ http://cfd.direct/openfoam/

［4］ 井上　義昭（高度情報科学技術研究機構）：「京」におけるOpenFOAMの性能評価　平成26年度「京」における高速化ワークショップ　2014年12月19日

［5］ 井上　義昭（高度情報科学技術研究機構）：「京」でのOpenFOAM支援内容と実績　第2回OpenFOAMワークショップ　2014年10月17日

［6］ 井上　義昭（高度情報科学技術研究機構）：「京」OpenFOAM移植＆支援状況　第3回OpenFOAMワークショップ　2015年10月15日

［7］ 山本　秀喜（高度情報科学技術研究機構）：OpenFOAM移植状況　第2回OpenFOAMワークショップ　2014年10月17日

［8］ 小林　寛（高度情報科学技術研究機構）：OpenFOAMワークショップ～性能評価手順と大規模実行時の問題点～　第3回OpenFOAMワークショップ　2015年10月15日

［9］ 千葉　修一（富士通）：「京」コンピュータにおける富士通の取り組みとOpenFOAMの評価　第3回OpenFOAMワークショップ　2015年10月15日

［10］ ファム　バン　フック（清水建設）、井上　義昭、浅見　暁（高度情報科学技術研究機構）、内山　学（清水建設）、千葉　修一（富士通）：「京」コンピュータでのC++型流体コードにおけるMPIの評価　第151回ハイパフォーマンスコンピューティング研究発表会　vol. 2015-HPC-151, no. 19, pp. 1-9, 2015-09-23

［11］ 内山　学、ファム　バン　フック（清水建設）、千葉　修一（富士通）、井上　義昭、浅見　暁（高度情報科学技術研究機構）：OpenFOAMによる流体コードのHybrid並列化の評価　第151回ハイパフォーマンスコンピューティング研究発表会　vol. 2015-HPC-151, no.20, pp.1-6, 2015-09-23