Vol.2013-ARC-207 No.23
Vol.2013-HPC-142 No.23
2013/12/17

過渡現象数値計算のボトルネックとポスト処理連携

小林 泰三[1,a)]　森江 善之[1,b)]　高見 利也[1,c)]　青柳 睦[1,d)]

概要：エクサスケール・コンピューティングに向けての研究開発が進められている一方で，大学基盤センターでのスーパーコンピュータ利用の現場ではポスト処理があらたなボトルネックとして問題になってきている．特に過渡現象を対象にした数値シミュレーションでは，プレ・ポスト処理を含めた全計算時間のうちの９割をポスト処理が占めてしまう状況が出てきている．本稿では，ボトルネックの状況を議論し，その解決法としてのポスト処理連携を提案し，現在研究開発を進めている OpenFOAM でのポスト処理連携の進捗を報告する．

キーワード：連成計算，連携計算，過渡現象，数値シミュレーション，ポスト処理，OpenFOAM

A Bottleneck and Cooperation with the Post Processes in Numerical Calculation of Transient Phenomena

Taizo Kobayashi[1,a)]　Yoshiyuki Morie[1,b)]　Toshiya Takami[1,c)]　Mutsumi Aoyagi[1,d)]

Abstract: Research and development of exa-scale computing is advanced energetically. On the other hand, new bottlenecks appear in the post processing of large scale numerical simulations. A numerical simulation of transient phenomena takes more than 90% of time for the post processing. In this paper, the details of the bottlenecks are discussed. Then as a solution for the bottlenecks, cooperation between simulation and the post processing is proposed. An implementation of it for OpenFOAM is also reported.

Keywords: Coupled simulation, Cooperated calculation, Transient phenomena, Numrical simulation, Post processing, OpenFOAM

1. はじめに

「京」コンピュータの成功と，その後のエクサスケール・コンピューティングの研究開発により，膨大な CPU コアやアクセラレータを利用した超並列計算が実現されてきている [1]．この傾向は今後も暫く続くと予想され，ハードウェアではコア間やノード間通信の効率的な構造を探る研究がなされ，ソフトウェアではアプリケーションである数値計算の並列化の向上やノード間通信の効率化などの研究が盛んに行われている．また，大学基盤センターに導入されているスーパーコンピュータにもそれらの知見は還元されてきており，流体計算に限ってみても，FrontFlow[2], [3] や OpenFOAM[4] といった高性能でかつ使い易いソフトウェアが開発され広く利用されている．九州大学の情報基盤研究開発センターにてサービスを提供している OpenFOAM でも 1000 コア程度の並列計算は容易に実行可能である．このように，ハイパフォーマンスコンピューティングの最先端であるスーパーコンピュータとその周辺の研究開発の成果は基盤センターの利用者にも届いており，それほど計算機に詳しくない研究者でも基盤センターの大規模な並列計算機を利用して，これまで最も計算リソースが必要であった数値計算を効率よく実行可能な状況が整ってきている．

[1] 九州大学情報基盤研究開発センター
Research Institute for Information Technology, Kyushu University
[a)] tkoba@cc.kyushu-u.ac.jp
[b)] morie.yoshiyuki.404@m.kyushu-u.ac.jp
[c)] takami@cc.kyushu-u.ac.jp
[d)] aoyagi@cc.kyushu-u.ac.jp

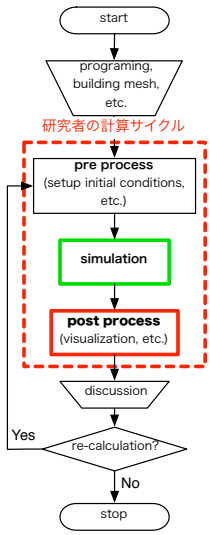

図 1 数値計算を用いた研究の流れ．赤破線で囲まれた部分が研究者が計算機上で行う作業であり，これまでは緑の数値シミュレーション部分が計算時間の大半を占めており，計算機科学分野での研究対象ももっぱらこの数値シミュレーションになっている．しかし，大規模計算が可能になった現在では，ポスト処理に多大な時間が掛かる様になり，新たなボトルネックになってきている．

その一方で，計算に必要な入力データを整備するプレ処理や，数値計算結果を議論するために必要な可視化やデータ解析等のポスト処理にかかる手間と時間が膨大になり，新たなボトルネックになってきている．

数値計算を行う研究者が計算機上で行う作業である計算全体を，図 1 に赤い破線で示した．緑の数値シミュレーションが最も計算リソースが必要であり，これまでは計算時間も計算全体の大半を占めており，計算機科学（HPC）でももっぱらこの数値シミュレーションを研究対象にしてきた．しかし，大規模計算が可能になった現在では，ポスト処理に多大な時間が掛かる様になり，新たなボトルネックになってきている．今後は図 1 に赤い破線で示したプレ・ポスト処理を含めた計算全体の状況も研究対象にする必要がある．

実際に，この計算全体としての環境を整える事が重要であるとの議論が「学際大規模情報基盤共同利用・共同研究拠点 (JHPCN)」[5] にて行われており，さまざまな研究開発が進められている．この JHPCN での 2012 年度採択課題の中で，プレ・ポスト処理を含む計算全体の環境を整える必要性を述べているものを表 1 に列挙する．

全 35 課題中 10 課題が計算全体の環境整備の必要性を明記している．加えて，「移動境界問題及び連成計算の大規模流体シミュレーションと動的負荷分散の評価（12-NA09）」のように，ポスト処理がネックになっていながらそれらが研究者の興味を惹かなかったり，計算科学での成果が上がったためにあえて報告に明記していない研究課題も存在

表 1 計算全体の環境整備に言及している JHPCN 採択課題

課題番号	課題名
12-DA03	生体分子の大規模分子動力学計算に対する時系列解析とその応用
12-MD02	次世代ジオスペースシミュレーション拠点の構築
12-MD03	高分子系粗視化シミュレーション基盤の計算機科学的高度化検討
12-MD04	壁乱流の大規模組織構造の解明がもたらすエネルギー高効率化への貢献
12-MD05	グリッドデータファームによる大規模分散ストレージの構築とサイエンスクラウド技術の研究
12-MD06	マルチパラメータサーベイ型シミュレーションを支えるシステム化技術に関する研究
12-NA05	分野横断型ハイパフォーマンス計算力学の新展開
12-NA10	超多自由度複雑流動現象解明のための計算科学
12-NA11	GPGPU による地震ハザード評価
12-NA16	災害影響評価のための大規模マルチフィジックス・シミュレータの高度・高性能化

するであろう．いずれにしても，課題番号 12-MD06 のように，マルチパラメータサーベイを行うためのプレ・ポスト処理を含めた環境の構築そのものを研究する課題が存在する．これは，実際の大規模計算環境を利用して研究を行っている現場の研究者には，主たる数値計算の高速化だけではなく，図 1 に赤い破線で示したプレ・ポスト処理を含む計算全体の高速化が重要であるとの認識が共有されてきていることを示している．

しかしながら，これらの知見を計算機科学の研究対象にするには若干の距離がある．例えば，表 1 に上げた課題には以下の特徴がある．

- 研究主体が HPC を利用する計算科学者である
- 論文投稿先が流体や土木・建築といった分野が多く，情報系が少ない
- フレームワーク化・一般利用化は研究段階であるものが多い

JHPCN の様な学際的な共同研究の場で明らかになってきたポスト処理を含む総合的な計算環境の問題は，この先のエクサスケール以降ますます重要になる．この認識を計算機科学の研究者間で共有するのが，本稿の大きな目的の一つである．

本稿の構成は，続く 2 節でポスト処理がボトルネックになっている数値計算の研究を事例として取り上げて，ボトルネックの詳細を議論する．つづく節にてボトルネックを解消するために進めている研究開発の進捗を報告する．

2. 流体音研究における事例

ここでは，著者が研究に参加している流体音研究 [6] [7] を題材にして，ポスト処理がボトルネックになっている状況を議論する．

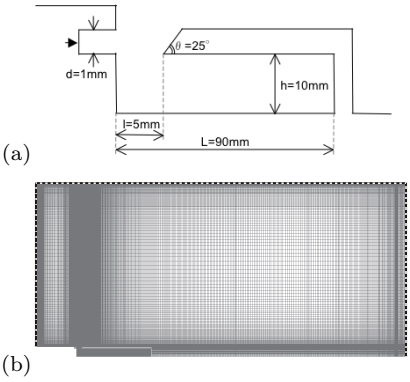

図 2 エアリード楽器のモデル寸法 (a) とメッシュ (b)

表 2 メッシュパラメータ

	Points	Cells	Faces
2D	158,762	78,492	314,856
3D	4,048,431	3,924,600	11,896,692

表 3 計算環境

	名称	Version
システム	Fujitsu PRIMERGY CX400	Xeon E5-2680 2.70GHz
ソフトウェア	OpenFOAM	2.2.2
コンパイラ	Intel Compiler XE, Intel 64	12.1.5
MPI	IMPI	4.0.3.008
ノード間通信	InfiniBand	

2.1 モデル

この研究は，エアリード楽器の発音機構を数値流体計算（CFD）を用いて解明するものであり，過渡現象数値計算の一つである．計算は，流体計算で広く使われているオープンソースの OpenFOAM に含まれている圧縮性 LES ソルバーの `rhoPisoFoam` を用いている．図 2 に示した様に，計算モデルはリコーダーの管端を閉じた形のものを用いており，メッシュは音波の伝搬領域を広めに確保して，上方と左右の境界は透過壁に設定してある．メッシュ数などのパラメータは表 2 に纏めた．計算環境は九州大学情報基盤センターでサービスしている高性能演算サーバであり，緒言を表 3 に纏めた．この環境で得られた計算結果を可視化したものが図 3 である．理論通りの周波数で楽音が発振している事が確認でき，Lighthill の音源や Howe のエネルギー推論などの解析が進んでいる．

2.2 処理の詳細

この流体音研究に必要な計算機上での作業を見てみよう．まず，ファイルハンドリングについて考察する．OpenFOAM のソルバーは，指定したステップ間隔毎に各物理量を独立したファイルとして保存する．また，並列計算する場合は，各 MPI プロセス毎に独立してファイルが生成される．従って小容量のファイルが多量に生成される事に

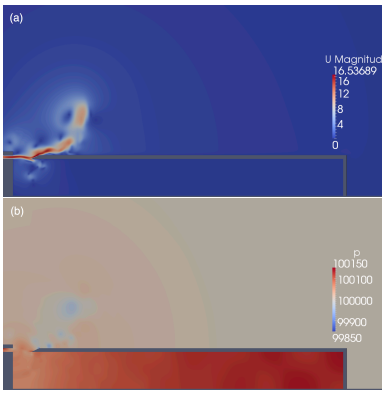

図 3 数値結果のスナップショット．速度分布 (a) と圧力分布 (b)

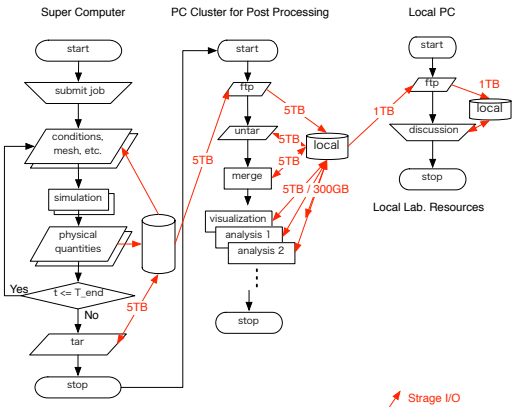

図 4 OpenFOAM を用いた典型的な流体計算の 1 サイクルをフローチャートで表したもの．スーパーコンピュータで数値シミュレーションを行い，ポスト処理は専用に用意された PC クラスター環境で行う．基本的に各処理は独立して I/O するのが判る．

なる．この仕様は，再計算や動的な境界条件の変更を可能にするなど，利用者の利便性に基づいた標準的なものである．生成される総ファイル数は式 (refeq:1) で見積もる事が出来る．

$$\text{総ファイル数} = \text{物理量} \times \text{並列数} \times \text{ステップ数/出力インターバル} \quad (1)$$

今回の事例では，物理量は 17，並列数は 864，ステップ数が 5×10^5，インターバルは 200 であり，総ファイル数は 36,720,000 になる．ファイルの容量は，gzip 圧縮を掛けた状態で 1 ファイルが 200 kB のオーダーであるので，全体でおよそ 5 TB になる．このファイルをポスト処理に掛ける事になる．以下は，この全計算（研究における 1 計算サイクル）の詳細である．

(1) プレ処理: ～ 10 分
  (a) メッシュ生成: `blockMesh`
  (b) メッシュ分割: `decomposePar`
(2) シミュレーション: `rhoPisoFoam`, 20 時間
(3) ポスト処理

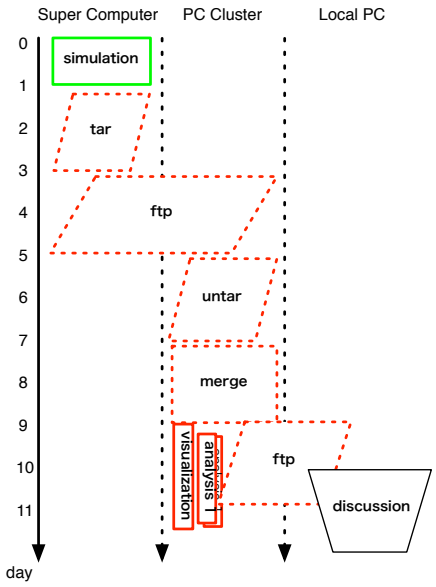

図 5 OpenFOAM を用いた典型的な流体計算の 1 サイクルを時系列で表したもの．スーパーコンピュータでの数値シミュレーション（緑）は 20 時間で終了する．しかし，ポスト処理（赤）を開始するまでに 3 千万ファイルを転送し，MPI プロセス事に分かれているファイルをマージする必要がある．これらは tar と ftp を利用した演算プロセスが殆どないストレージへのファイル I/O（赤破線）であるが，実測をするとそれぞれが 2 日を要する．

(a) tar でデータをファイルに纏める: 2 日
(b) ftp で纏めた tar ファイルをポスト処理環境に転送する: 2 日
(c) tar ファイルを展開する: 2 日
(d) 分散して出力されたファイルのマージ: `reconstructPar`, 2 日
(e) 可視化：`foamToVTK`, 2 日
(f) 各種解析：`vorticity`, `Lighthill`, `Howe`, `sample`, 2 日

これをフローチャートで表すと図 4 になる．この 1 計算サイクルで生成されるデータ量は，数値シミュレーションが 5 TB，可視化データが 300GB，各種解析データが 100MB である．図 5 に示す様に，処理時間は数値シミュレーションが 20 時間，ポスト処理の各段階で 2 ～ 3 日を要す．つまり，数値計算が上がってから可視化を経て研究の議論が出来るまでに 2 週間ほど掛かっており，総計算時間のうちでポスト処理に費やす割合が 9 割を超えることになる．実際の研究では，系の設計変更やアンサンブルやパラメータサーベイ等で，この計算サイクルを必要なだけ繰り返す事になる．そのオーダーは $10^{2\sim3}$ に達することも稀ではない．

以上の議論をもとにして，過渡現象数値シミュレーションの特徴を数値計算を行う研究者の立場とそれを受ける計算機科学の立場でそれぞれ纏めると以下になる．

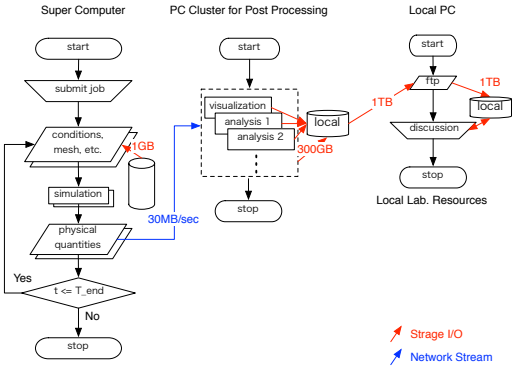

図 6 OpenFOAM を用いた典型的な流体計算の 1 サイクルをフローチャートで表したもの．スーパーコンピュータで数値シミュレーションを行い，ポスト処理は専用に用意された PC クラスター環境で行う．シミュレーション結果をネットワークを通して直接ポスト処理に掛ける事により，ストレージへの I/O を大幅に減らしている．

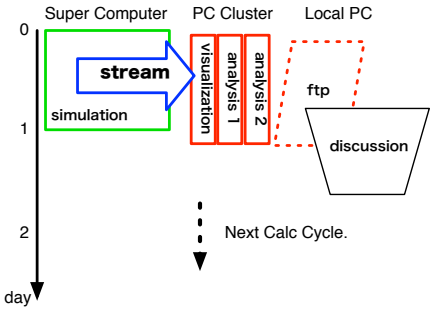

図 7 OpenFOAM を用いた典型的な流体計算の 1 サイクルを時系列で表したもの．スーパーコンピュータでの数値シミュレーションは 20 時間で終了する．そして，計算結果は直接ポスト処理を行うサーバにソケット通信で送られ，直ちにポスト処理が行われる．従って，利用者は数値シミュレーションの進行と殆ど同時にポスト処理結果に基づいた研究上の議論が開始できる．

- 現象が始まる前から終わるまで，或は安定するまで計算
  → 計算ステップ数が大きくなる：10^6 以上
- 一本の軌跡（計算）では統計的議論が困難
  → アンサンブルが必要：$\sim \times 10$
- パラメータ依存性の議論が必要
  → パラメータサーベイ：$\sim \times 10^{2\sim3}$
- 数値結果のデータ解析が複数段になることがある
  → 多段階プレ・ポスト処理，大小多寡さまざまなファイルハンドリング

そして，本節で詳細を見てきた様に，ボトルネックはプレ・ポスト処理のストレージに対するファイル I/O にある．

3. 連成・連携計算としてのポスト処理連携

前節でのポスト処理におけるボトルネックの詳細を受けて，ここではその解決法を議論する．

前節の議論でポスト処理のボトルネックがファイル I/O であることが明らかになったので，対処方法は，ファイル I/O の性能を改善するか，ファイル I/O を他の方法で置き換えるかになる．本稿では，連成・連携計算 [8][9] を応用して，数値シミュレーションからダイレクトにポスト処理にデータを流す方法を提案する．

図 4 に示した計算に，流体計算とポスト処理を連携すると図 6 になる．シミュレーション結果をネットワークを通して直接ポスト処理に掛ける事により，ストレージへの I/O を大幅に減らしている．これにより，tar や ftp やマージと云った処理が不要になっている．この効果を時系列で表すと図 7 になる．

3.1 OpenFOAM の特徴

実装の詳細に進む前に，OpenFOAM のソフトウェアとしての特徴を概観しておこう．

OpenFOAM は C++ を用いて徹底したオブジェクト指向で開発されており，以下の特徴を持っている．

(1) 研究者が望むソルバーを作成するのが極めて容易である
(2) 差分法や誤差の修正方法を，パラメータの指定と同様に設定できる
(3) 外部ライブラリの利用は最小限

（1）のソルバー作成は，解くべき偏微分方程式を解く順番に書き並べれば良く，殆ど疑似コードを書く程度の手間で完了する．例えば流体力学での質量保存則

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0 \qquad (2)$$

は，OpenFOAM では以下の記述で完了する．

```
solve
(
  fvm::ddt(rho) + fvm::div(rhoFlux, rho)
);
```

このように，ソルバーのコード自体はわずか数十行で済む場合が多い．従って，計算機やプログラミングの知識が少なくても計算対象の物理さえ理解していればソルバーを作成できる．

（2）の計算スキームは，ソルバー内にハードコードするのではなく，ユーザーが実行時に指定する方法で実現している．解法に関しては `fvSolution` 設定ファイルで指定し，スキームに関しては `fvSchemes` 設定ファイルで指定する．例えば上の質量保存即の計算では以下のように記述する．

“fvSolution”

```
solvers
{
    rho
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0.2;
    }
}
```

“fvSchemes”

```
ddtSchemes
{
    default         Euler;
}

divSchemes
{
    default         none;
    div(rhoFlux,rho) Gauss upwind;
}
```

このように解法やスキームがソルバーコードから分離されている事で，実際の数値計算で現れる様々な不確定要素に対応している．例えば，同じ基礎方程式に従う系でも境界などの状況によって適切な解法や計算スキームが異なるが，そのような条件にも柔軟に対応できる．

（3）は移植性と計算結果の信頼性の面で有利である．OpenFOMA-2.2.2 現在では，依存しているライブラリはメッシュツールの scotch[10] と MPI のみであり，MPI の利用も send, Isend, receive, Ireceive, allreduce のみである．

これらの特徴により，OpneFOAM は，実際に研究や企業での開発で「使える」クラスライブラリとしての偏微分方程式ソルバー環境を実現している．実際，OpenCAE 学会 [11] では毎週の様に OpenFOAM 講習会を全国各地で開催している．これは，HPC においても C++ のオーバーヘッドがユーザーにとって大きな問題ではなくなっている事を示している．

3.2 OpenFOAM への実装

ここでは，図 6 を具体化する実装方法について議論する．具体的には，流体の圧縮性 LES ソルバーである `rhoPisoFoam` から MPI 並列のために分割された上方を纏める処理を担当する `reconstructPar` に対してソケット通信を経由してダイレクトに計算結果を送る事により実現できる．以下にその機能を実現するためのクラス設計について議論する．

前節で概観した様に，OpenFOAM は徹底したオブジェクト指向に基づいて C++ 言語で書かれている．従って，ポスト処理連携を実現するための通信機能を追加実装するには，OpenFOAM 自身のクラス設計を理解してその枠に沿う様に実装する必要がある．図 8 にファイル I/O に関す

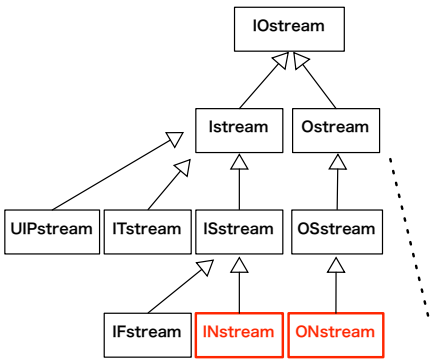

図 8 OpenFOAM のストリームクラスの継承関係図．赤枠の “INstream” と “ONstream” が新たに開発されているクラスである．この位置でクラスを実装すれば，現在ファイル I/O で行っている処理を最小限の変更でソケット等の通信に置き換える事が可能になる．

る stream クラス周辺の継承関係を示す．この図に示すように，フィアル I/O に関する事柄は IFstream, OFstream に纏められている．我々の最初の目的は，ファイル I/O をソケット通信等のネットワークストリームに置き換える事であるので，図 8 で赤く示した Nstream を実装すればよい．ただし，この新しく実装する Nstream は，IOstream だけではなくソケット通信などを扱うネットワークの抽象クラスも継承する必要がある．なぜならば，たとえ本稿で対象にしているポスト処理連携だけを実装するにしても，通信で発生するさまざまな不確定要素に対応して基盤センターでサービスとして提供できる程度の信頼性を確保するには，通信関連を司るクラス体系の存在は必須であるからである．さらに OpenFOAM 内でネットワークを利用する連成計算などを将来実現するためにはネットワーク越しの通信を管理する機構が必須であり，クラス設計の観点からはソケット通信を担当するクラスもネットワークのクラス体系に属していなければならない．現在はこのネットワークのクラス体系を設計している段階である．C++ を用いたネットワーククラス体系は，ACE[12] を参考にしている．

4. まとめと今後の展望

本稿では，近年の大規模計算，特に過渡現象の数値シミュレーションで大きなボトルネックになっているポスト処理について議論した．JHPCN での研究課題にみられるように，プレ・ポスト処理を含む計算サイクル全体を考慮した計算効率の向上が求められている．そしてこれらの問題は既に基盤センターに設置されているスーパーコンピュータの利用現場で発生していることから，エクサスケール時代にはより大きな問題になると予想される．

また，事例として取り上げた OpenFOAM による流体音研究に見られる様に，C++ のオーバーヘッドによる速度低下は実際の研究現場では問題ではなくなっている状況が起きつつある．数値シミュレーションの計算速度は今後も重要であるが，それと同等以上に必要であるのが，ソルバーや解法や計算スキームの表現のしやすさと，かつそれらの管理のしやすさであることを OpenFOAM の成功は示唆している．

HPC の立場としては，プレ・ポスト処理を含む計算全体の速度向上が新たな研究対象になる．プレ・ポスト処理を含む計算全体は，数値計算とデータマイニングや可視化の様に質的に全く異なる複数の処理を連携させなければならない．従って不確定要素へ対応するための管理機構と通信スキームの整備が必要になる．本稿では，その最初の試みとして，OpenFOAM にソケット通信の機能を実装し，Fstream を Nstream に置き換えることでソルバーとポスト処理アプリを連携させる設計の進捗を報告した．

今後は，OpenFOAM 内でのネットワーククラス体系の設計をすすめる．実装とテストは九州大学情報基盤研究開発センターの高性能演算サーバシステムと可視化ノードの環境で行い，ポスト処理におけるボトルネックの解消と，連成・連携計算の基礎環境の構築を目標にして研究開発を行う．

謝辞 新潟大学の大嶋拓也氏には OpenFOAM へのソケット実装実例の提供とご教示を戴いた．情報通信研究機構の渡邊英伸氏と九州大学の深沢圭一郎氏にはソケット通信ライブラリの提供とご教示を戴いた．OpenFOAM とソケット通信の実利用に関する知見を戴いた事に感謝致します．

参考文献

[1] 《特集》「スーパーコンピュータ「京」，情報処理学会誌：情報処理 Vol.53 No.8
[2] GUO, Yang, Chisachi KATO, and Yoshinobu YAMADE. “Basic Features of the Fluid Dynamics Simulation Software “FrontFlow/Blue”. ” 生産研究 58.1 (2006): 11-15.
[3] FrontFlow: マルチフィジックス流体シミュレーション・システム，http://www.ciss.iis.u-tokyo.ac.jp/rss21/theme/multi/fluid/fluid_softwareinfo.html
[4] OpenFOAM: Open Field Operation And Manipulation: The Open Source CFD Toolbox, 入手先 ⟨http://www.openfoam.com⟩
[5] JHPCN シンポジウム: 学際大規模情報基盤共同利用・共同研究拠点 第 5 回シンポジウム, http://jhpcn-kyoten.itc.u-tokyo.ac.jp/sympo/
[6] T. Iwasaki, T. Kobayashi, K. Takahashi, T. Takami, A. Nishida, M. Aoyagi: *“Numerical study on the function of tone holes of a recorder like instrument from the viewpoint of the aerodynamic sound theory”*, ICA2013/POMA, 19, 035024, 2013
[7] M. Miyamoto, Y. Ito, T. Iwasaki, T. Akamura, K. Takahashi, T. Takami, T. Kobayashi, A. Nishida, M. Aoyagi: *“Numerical study on acoustic oscillations of 2d and 3d flue organ pipe like instruments with compressible les”*, Acta Acustica united with Acustica, 99, 154171, 2013
[8] 押川雄大，小林泰三，森江善之，高見利也，青柳睦：ヘテロジニアスな並列計算環境を応用した連成・連係計算の提案,

情報処理学会研究報告. 2011-HPC-130(20),(SWoPP2011) pp.1-7, 2011-07-20
[9] 押川雄大，小林泰三，森江善之，高見利也，青柳睦：連成・連携計算によるデータ量削減の評価，情報処理学会研究報告. 2011-HPC-132(4), (Hokke19) pp.1-6, 2011-11-21
[10] Scoch: Software package and libraries for sequential mesh and hypergraph partitioning, 入手先 ⟨http://www.labri.fr/perso/pelegrin/scotch/⟩
[11] OpenCAE 学会: 一般社団法人オープン CAE 学会 http://www.opencae.jp
[12] ACE: The ADAPTIVE Communication Environment 入手先 ⟨http://www.dre.vanderbilt.edu/ schmidt/ACE.html⟩