100 億超格子を用いた自動車の大規模流体解析への挑戦

小 野 謙 二

理化学研究所 計算科学研究機構

大 西 順 也

東京大学 生産技術研究所

１．はじめに

ものづくりの現場において，物理シミュレーションによる設計支援への期待は大きい。自動車設計では，衝突安全性の予測技術として構造解析が用いられるが，熱マネジメントや燃費，走行安定性などの空力性能設計には計算流体力学（Computational Fluid Dynamics, CFD）シミュレーションが用いられる。CFD は流動現象の非線形性をより精確に再現する場合，非常に多くの計算格子点を必要とするため，大規模な計算機資源を必要とするシミュレーションの一つである。

CFD は計算機の発展とともに，学術的・技術的にも大きく進展してきた。CFD のプロセスは，図１に示すように形状データや格子生成に関連する前処理，流体解析，可視化や統計処理などのポスト処理などのサブプロセスから構成される。現実の工業製品の設計に CFD を適用する場合には，現在でも多くの課題がある。まず，最初の課題として現象の数理的なモデリングが挙げられる。流動現象のマクロな運動は Navier-Stokes 方程式の型式で表現されるが，構成方程式は経験的に決められている。特に，混相流の場合には，構成方程式の形やパラメータなど未定なものも多い。比較的多くの研究がなされている乱流現象でさえ，種々のモデルや計算方法が提案されており，未だに決着はつかない。しかしながら，最近の巨大な計算機資源を用いて，できるだけ任意のパラメータを用いず，高い信頼性の結果を得ようとする試みがある。その手法の一つがラージエディシミュレーション（Large Eddy Simulation, LES）である[1]。乱流現象は大きな渦から小さい渦まで，さまざまなスケールの渦から構成されている。LES は，計算格子を細かくしていった極限で，格子で捉えられるスケールの渦は直接計算し，格子スケールより小さな渦は統計的に普遍性のあるモデルを利用する計算方法である。LES は経験的なパラメータが少なく，高い信頼性の結果が得られる計算方法として注目を集めている。

次の課題としては，計算対象の形状の複雑性がある。自動車は非常に多くの部品から構成され，形状も非常に複雑である。これらの形状データは CAD により作成されるが，流体計算を行う場合には，この CAD データから計算格子を作成する必要がある。この前処理プロセスにかかる時間と労力が大きな課題である。

３つめの課題は，シミュレーションにかかる時間である。シミュレーションは，流体計算時間だけでなく，前述の格子生成，および後述のポスト処理の時間も含む。CFD の全プロセスの時間を十分に短くすることが要求される。このため，各サブプロセスに必要な時間を短縮する並列処理のアプローチが必要となる。

大規模な計算機資源を活用したシミュレーションを実行するにあたり，CFD の全てのプロセスについて上記の課題がある。本研究では，各課題についてどのような方針で臨めば良いかを

明確にし，今後の大規模シミュレーションの普及促進を図る礎にすることを目的とする。

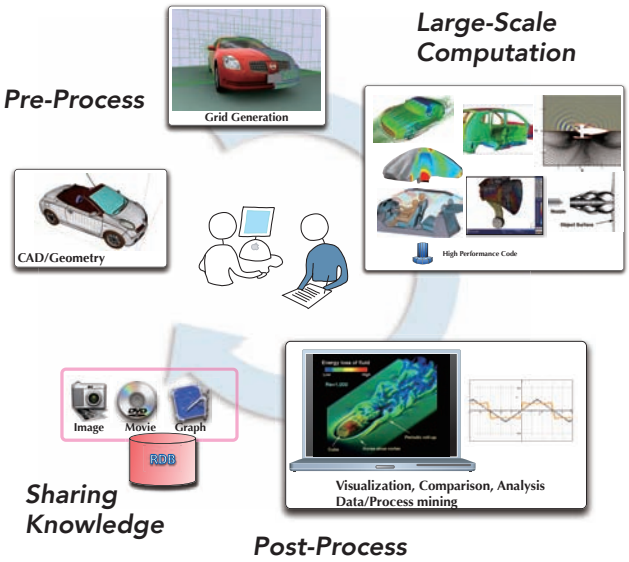


図 1　実務における CFD のプロセスの例

CAD データから計算格子作成，流体計算，可視化や統計処理，その後シミュレーション結果を活用して設計を行う。実務においては，これらの各プロセスをシームレスに実施し，短期間で設計ができるようにすることが要求される。

２．計算格子生成

従来の実用的な計算手法の主流は，非構造格子や BFC 格子を用いた計算手法であり，計算格子の作成に大きな労力を必要とする。場合によっては，計算格子生成に全シミュレーションプロセスの 50%を超える時間を要する場合もある。

大規模な流体計算には，非常に多くの格子点をもつ計算格子が必要となる。例えば，10 億（1G）点の BFC 構造格子の生成を考えてみよう。3 次元の座標点データを単純に単精度で保持すると，x，y，z 各 4B で 1 点あたり 12B で，10 億点では 12GB となる。これに，計算に必要な識別子や付加的な情報もあるので，少なくとも 20GB 程度になる。非構造格子（四面体とする）の場合には，要素あたりの 4 頂点と接続情報，識別子などを含めて少なくとも 70B/要素は必要なので，10 億要素では 70GB のオーダーとなる。エンジニアが利用する PC での格子生成を考えると，現在でも 70GB のメモリ空間はかなり高額となるので現実的とは言えない。今後の計算規模は，100 億以上となる見込みであり，もはや PC での格子生成は難しい領域となる。したがって，前処理として計算格子生成はできなくなるため，計算時に格子生成を行う方法へ移行せざるを得ない。

シミュレーション利用の観点からも，計算時の格子生成には大きな利点がある。まず，大規模計算を実行する計算機は，殆どの場合，バッチ運用の計算機となる。キューの混み具合によって，利用するキュー（大抵，並列数と利用時間の上限があるため，混んでいるキューとそうでないキューなどがある）を選択することができる。状況に応じて，並列数を変更する場合，事前に計算格子を作成する方法では対応できない。また，最初に 512 プロセスで並列計算をして，リスタート時に 1,982 並列で計算をするという利用方法も考えられる。この様な柔軟な利

用を行うためにも計算開始時に格子を生成する技術は必須である。

計算精度の点からは，格子の直交性や解像度，形状の再現性などの課題がある。まず，格子の性質や後に述べる格子生成アルゴリズムと記憶容量の点から，直交格子を採用する。形状の再現性については，直交格子上で任意形状を反映する計算スキーム[2]の導入により対応する。

計算時の初期化段階で格子生成を実施するためには，格子生成アルゴリズムに幾つかの要件が必要になる。一つは，格子生成アルゴリズムのロバスト性である。入力となる形状データから格子を作成する場合にアルゴリズム的に破綻せず，かつ安定であることが必須である。この要請を考慮し，データ構造として単純な直交格子を利用する。直交格子を用いると記憶領域も小さくなり，ソルバーの要求メモリ量も低下するため，メリットは大きい。次の要求項目は，格子生成アルゴリズムの並列性である。分散並列処理で格子生成を行う必要があるためである。

上記の要件を満たす格子生成法として，直交格子データ構造を用いた交点計算に基づく格子生成法を開発した。その概要を図２に示す。計算領域と境界条件については，指定を支援するGUI アプリケーション FXgen を利用する。FXgen[1]は計算対象となる形状データを読み込み，計

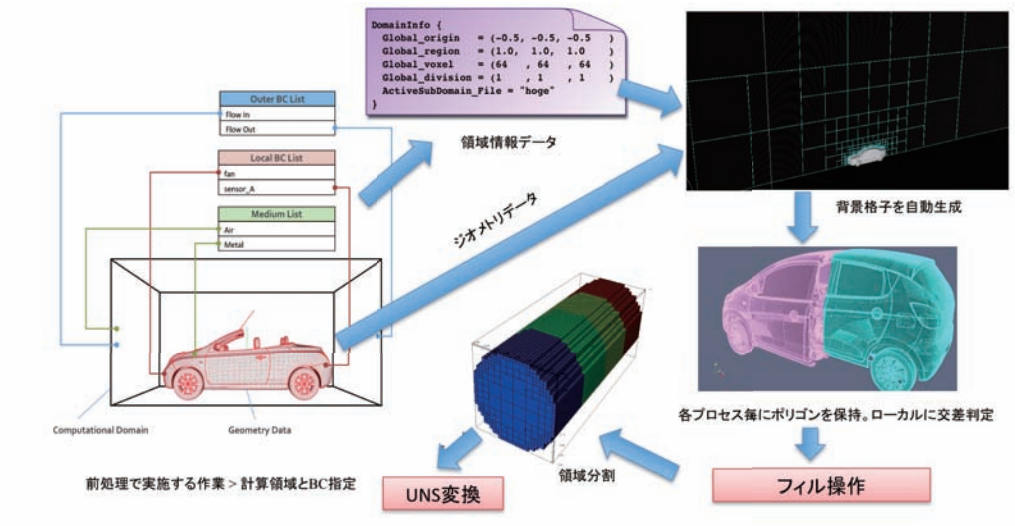


図２　計算時の初期化段階で格子を生成する方法

流体計算の初期化段階で，計算格子を自動生成する。入力データとして，形状を表す STL データと計算領域や境界条件を指定するメタファイルを用意する。メタファイルは，専用の GUI アプリケーションを利用して，事前にテキストデータとして作成する。自動生成された格子データは，オプションにより非構造格子データへも変換でき，非構造格子ソルバーの入力データとしても利用できる。

算領域を空間座標から適切に決めることができる。境界条件については，ロードしたポリゴンデータに識別子を付けて出力する。この識別子は，ソルバーでの境界条件の指定に利用する。流体解析ソルバーは，識別子を付与された形状データと計算領域情報などを記述したメタデータを読み込む。その後，計算領域情報から背景格子を自動生成し，物理量の定義点座標間を結ぶ線分と物体形状との交点計算を行う[2]。この交点計算はロバストであり，部分領域毎に独立に計算可能である。計算された定義点から物体表面までの距離を用いて，計算スキームを構成

[1] 東京大学生産技術研究所からオープンソースソフトウェアとしてリリース予定。

することにより，格子生成の自動化を実現した。

３．流体ソルバー

利用した流体解析ソルバーはFrontFlow/Violet Cartesian(FFV-C)である。このソルバーは，直交等間隔格子を用いた三次元非定常の熱流体解析プログラムである。理化学研究所VCADシステム研究プログラムで開発してきた熱流体解析システム V-Sphere::CBC システム[3]を参考にして，大規模並列計算用に再設計したものである。また，機能モジュール化を進め，下記に示すライブラリ群を利用したプログラミングを行っている。

3.1.　並列計算ミドルウェア CPMlib

FFV-CはCBCシステムと同様に，オブジェクト指向設計で開発している。V-Sphere::CBCの場合には,ミドルウェアとしてフレームワーク的な要素の強いミドルウェアV-Sphereを用いていたが，FFV-C ではより柔軟なプログラミングができるように設計したライブラリ CPMlib（Cartesian Partition Management Library）を開発して利用している。CPMlibは直交等間隔格子を用いた並列計算アプリケーションを記述するために必要な，領域分割処理，並列領域の情報管理，領域間通信の機能を提供する軽量のクラスライブラリである[2]。

3.2.　ポリゴン管理，交点計算ライブラリ Polylib，Cutlib

計算格子の自動生成のためには，ソルバーで形状データのロードと管理を行う必要がある。Polylib[3]は並列領域におけるポリゴンデータの管理機能を提供する。ポリゴンデータのロードと部分領域情報に基づくデータの分散保持，出力，移動，マイグレーションなどの機能をもつ。Cutlib[4]は，ポリゴンと任意線分との交点計算を行う機能を提供する。現在は，直交等間隔格子のみに対応しているが，不等間隔格子への拡張も予定している。

```
Steer {
   . . .
   TimeControl {
       AccelerationType  = "Time"                  //  加速時間の単位指定
       Acceleration       = 1.0
       DtType            = "CFLReferenceVelocity" //  参照速度基準のクーラン数
       DeltaT            = 0.2
       PeriodType        = "step"                 //  計算時間の単位指定
       CalculationPeriod = 1000
     }
   . . .
 } //Steer
```

図３　TextParserによるパラメータの記述

キーワードとキーワードに対する値（数値と文字列）を内部ツリーに保持し，保持したパラメータにアクセスするAPIを提供する。ブレースにより階層的なパラメータ記述ができる。また，自由形式の記述やコメントの挿入なども可能。

3.3. パラメータパーサーライブラリ TextParser

計算パラメータはテキストで行われることが多い。最も簡単な方法としては，平文でフラットなテキスト記述を行う方法があるが，パラメータの意味づけや記述性の点で柔軟ではない。

---

[2] CPMlibは，東京大学生産技術研究所からオープンソースソフトウェアとしてリリース予定。
[3] V-Polylibとして，http://vcad-hpsv.riken.jp/jp/からダウンロード可能。
[4] V-Cutlibとして，http://vcad-hpsv.riken.jp/jp/からダウンロード可能。

一方，汎用的な方法として，XML による記述がある。階層性や任意記述性が高い一方で，煩雑で見通しがわるい点もある。加えて，XML ライブラリはかなり大規模であり，スパコンへの対応などの点でインストールに手間取る面もある。これらの点を考慮し，シミュレーションのパラメータを見通しよく，簡便に記述するため，階層性と自由記述性を主とした軽量パーサーを開発[5]した。図３に TextParser によるパラメータ記述の一例を示す。ブレース”{}”により階層化を図り，パラメータのグルーピングが可能になる。また，C ライクなコメントもいれることもできる。

```
-----------------------------------------------------------------
Report of Timing Statistics PMlib version 1.2

Operator  : Kenji_Ono
Host name : Strontium.local
Date      : 2012/07/11 : 20:31:24

Parallel Mode                  :   OpenMP (24 threads)

Total execution time           = 5.960922e+00 [sec]
Total time of measured sections = 5.461881e+00 [sec]

Statistics per MPI process [Node Average]
Lavel       |call |            accumulated time            |       flop | messages[Bytes]
            |     |  avr[sec]  avr[%]   sdv[s] avr/call[s] |   avr         sdv         speed
------------------+------------+--------------------------+-------------------------------------
Search Vmax :  25   6.0695e-02   1.11  0.000e+00  2.4278e-03   4.325e+08  0.000e+00    6.64 Gflops
assign BC   :  75   2.1376e-03   0.04  0.000e+00  2.8502e-05   0.000e+00  0.000e+00    0.00 Mflops
Copy Array  :  50   3.1872e-01   5.84  0.000e+00  6.3744e-03   0.000e+00  0.000e+00    0.00 Mflops
Pseudo Vec  :  25   3.0864e+00  56.51  0.000e+00  1.2345e-01   2.180e+10  0.000e+00    6.58 Gflops
P Vec FluxBC:  25   1.3652e-02   0.25  0.000e+00  5.4609e-04   1.055e+07  0.000e+00  737.20 Mflops
Pvec. EE    :  25   1.3234e-01   2.42  0.000e+00  5.2938e-03   2.595e+08  0.000e+00    1.83 Gflops
Pvec. BC    :  25   2.2888e-03   0.04  0.000e+00  9.1552e-05   3.870e+04  0.000e+00   16.12 Mflops
PvecFace BC :  25   8.4519e-04   0.02  0.000e+00  3.3807e-05   1.325e+04  0.000e+00   14.95 Mflops
assign Array:  50   3.9125e-02   0.72  0.000e+00  7.8251e-04   0.000e+00  0.000e+00    0.00 Mflops
Div CC      :  25   2.7460e-01   5.03  0.000e+00  1.0984e-02   8.939e+08  0.000e+00    3.03 Gflops
Poi Src. BC :  25   9.2988e-03   0.17  0.000e+00  3.7195e-04   5.499e+06  0.000e+00  563.93 Mflops
Poi Setup   :  25   1.5258e-05   0.00  0.000e+00  6.1035e-07   0.000e+00  0.000e+00    0.00 Mflops
Poi SOR2SMA :  50   1.7074e-01   3.13  0.000e+00  3.4149e-03   7.370e+10  0.000e+00  401.97 Gflops
Poi BC      :  50   9.1314e-05   0.00  0.000e+00  1.8262e-06   0.000e+00  0.000e+00    0.00 Mflops
Prj Vec CF  :  25   5.7449e-01  10.52  0.000e+00  2.2979e-02   1.398e+09  0.000e+00    2.27 Gflops
Prj Vec CFBC:  25   1.0159e-02   0.19  0.000e+00  4.0636e-04   1.930e+04  0.000e+00    1.81 Mflops
Prj Vec CC  :  25   3.0391e-01   5.56  0.000e+00  1.2156e-02   1.038e+09  0.000e+00    3.18 Gflops
Vec. BC     :  25   4.4822e-05   0.00  0.000e+00  1.7929e-06   0.000e+00  0.000e+00    0.00 Mflops
Div CF      :  25   1.0764e-01   1.97  0.000e+00  4.3057e-03   2.019e+08  0.000e+00    1.75 Gflops
P Norm Dmax :  25   4.0273e-02   0.74  0.000e+00  1.6109e-03   8.651e+07  0.000e+00    2.00 Gflops
Updt Vec.   :  25   7.0383e-03   0.13  0.000e+00  2.8153e-04   0.000e+00  0.000e+00    0.00 Mflops
Oflow Vel   :  25   2.9467e-02   0.54  0.000e+00  1.1786e-03   1.455e+07  0.000e+00  471.01 Mflops
Avr Space   :  25   2.7546e-01   5.04  0.000e+00  1.1018e-02   6.632e+08  0.000e+00    2.24 Gflops
H Stdout    :  25   1.0623e-03   0.02  0.000e+00  4.2495e-05   0.000e+00  0.000e+00    0.00 Mflops
H Base      :  25   7.7629e-04   0.01  0.000e+00  3.1051e-05   0.000e+00  0.000e+00    0.00 Mflops
H DomainFlux:  25   3.5381e-04   0.01  0.000e+00  1.4152e-05   0.000e+00  0.000e+00    0.00 Mflops
Monitoring  :  25   1.3113e-05   0.00  0.000e+00  5.2452e-07   0.000e+00  0.000e+00    0.00 Mflops
H Component :  25   2.0313e-04   0.00  0.000e+00  8.1253e-06   0.000e+00  0.000e+00    0.00 Mflops
------------------+------------+--------------------------+-------------------------------------
Total       |              5.461881e+00                          1.005e+11             17.14 Gflops
```

図４　PMlib の出力結果

3.4. 性能情報ライブラリ PMlib

PMlib (Performance Monitor Library) は並列計算時の性能情報をサンプリングする機能を提供する。一般的には，gprof などのチューニングツールを利用することが多いが，チューニ

[5] TextParser は東京大学生産技術研究所からオープンソースソフトウェアとしてリリース予定。

ングツールの利用は，実行性能への影響が少なくないためデバッグ時に限られる。ハードウェアカウンタ情報を用いた機能を提供しているシステムもあるが，ハードウェアカウンタはシステム依存である。そこで，ユーザがある区間の浮動小数点演算数を申告し，その区間の時間を計測することで演算性能のログをとる機能を開発した。図４に示すようなレポートを実行終了時に表示する。また，詳細モードでは，プロセス毎の統計情報も表示できる。このライブラリは，ユーザがループ内の浮動小数点演算数をカウントして，区間ラベルを指定するという手間がかかるが，どの計算機でも利用可能であり，異なるプラットフォーム間の性能評価ができる。また，タイミング測定にかかるコストは無視できるほど小さく，実際の問題を計算している場合でも利用できる利点もある。

3.5. FFV-C ソルバー

FFV-C ソルバー[6]は非圧縮清流体を対象としているので，解析方法は Fractional Step 法に基づいている。有限体積法を用いて離散化しており，対流項スキームには形式的に MUSCL スキームを用いている。反復解法には，RedBalck-SOR，CG，CGS，BiCGStab などを持つ。プログラムはオブジェクト指向技術を利用しており，C++のメイン関数から前述のライブラリ群をインスタンスして利用，コードのカーネル部分は Fortran90 を用いた混合言語で記述している。

４．ポスト処理

ポスト処理については，今回は市販の可視化ソフトウェア FieldView を用いた。利用した可視化環境を図５に示す。今回は，計算結果を Server のローカルストレージ（SSD:900GB）にダウンロードし，Client と Server 間で通信しながら並列可視化を実施した。なお，可視化用のファイルフォーマットとして PLOT3D 形式を用いた。

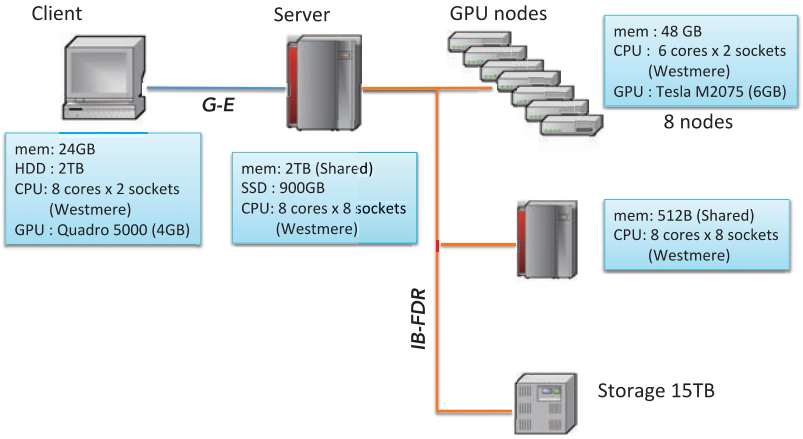


図５　可視化環境

５．計算結果

計算に利用したデータは，自動車会社から借用した実車の形状データである。STL 形式で 1,241 万ポリゴン，3.3GB のファイルサイズである。図６に自動格子生成アルゴリズムにより作成したエンジンルーム内断面の様子を示す。このケースは 4mm 幅の格子を用いたケースである。

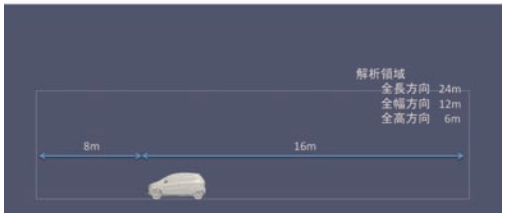
[6] FFV-C は東京大学生産技術研究所からオープンソースソフトウェアとしてリリース予定。

図 7 には車体形状と計算領域の大きさを示す。計算条件は，時速 100km/h 相当である。計算格子は 4mm の格子幅を用いた 290 億格子の計算を想定しているが，クーラン数 0.1 では，意味のある時間平均結果を得るまでに 290 時間を要する試算となる。この計算時間では，4800 ノードを 24 時間しても計算できない。そのため，異なる格子幅の３段階の計算格子を用いて計算する方法を採用した。これは粗い格子を用いて現象の時間を進め，準定常的になって順次細かい格子で計算する方法である。つまり，粗い格子を用いた計算結果からリスタートする場合に，細かい格子に補間する。これにより，計算初期の過渡的な現象を粗く近似し，意味のある時間領域で細かい計算を行い，精度の高い結果を短い計算時間で得ることができる。今回の一連の計算では，利用できる計算機資源の都合上，図 7(b)に示すジョブ１とジョブ２を FX10 Oakleaf 上で計算し，ジョブ３は京の上で計算した。

まず，ジョブ１とジョブ２の Poisson 方程式の反復履歴を図 8 に示す。補間リスタート時には，保存則を満たしていないことによる反復回数の増加が見られるが，時間とともに収束に向かうことが確認される。参考に，最終ステップのジョブ３（京で実行）の履歴を図 9 に示す。リスタート後，しばらく反復回数の多い時間があるが，これは格子が細かくなったことにより細かい渦が解像されるため，その圧力変動の影響と考えられる。この場合もある程度の時間の後，収束に向かう。

図 6　自動生成アルゴリズムにより作成した 4mm 幅格子の計算格子の断面図（車体近傍のみ）
ポリゴンと背景格子との交点計算の結果から固体と流体の識別を行い，そのごフラッドフィルアルゴリズムにより固体内部をフィルする。

	X	Y	Z	備考
解析領域サイズ	24.5 m	12.2 m	6.14 m	ブロッケージ比：約0.4%



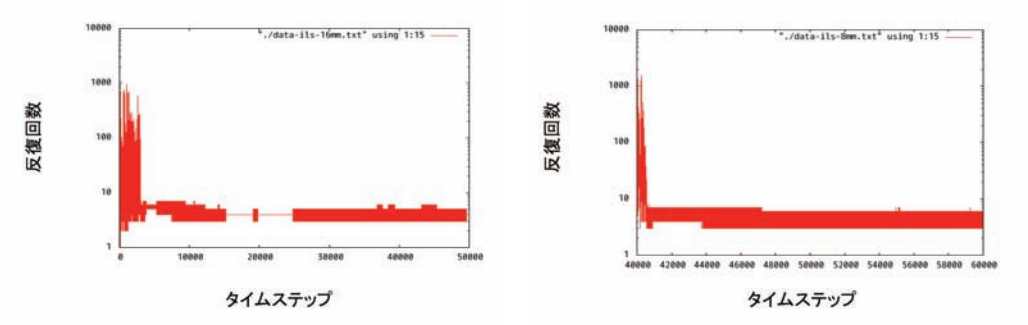
(a) 計算領域と対象

	X	Y	Z	備考
詳細モデル (F, ⊿x=4mm)	6144	3072	1536	約29G格子
中間モデル (C1, ⊿x=8mm)	3072	1536	768	約3.6G格子
疎視化モデル (C2, ⊿x=16mm)	1536	768	384	約0.45G格子

	モデル（格子点数）	ノード数【台】	ステップ数	解析時間【時間】	実時間【秒】	
ジョブ1	C2(0.45G)	4608	50000	2.0	2.87	初期値スタート
ジョブ2	C1(3.6G)	4608	20000	2.0	0.57	補間付きリスタート
ジョブ3	F(29G)	9216	10000	27.4	0.14	補間付きリスタート

(b) 計算格子

図 7　計算領域と計算格子



(a) Δx=16mm 格子（t=0〜2.3 sec.）　　(b) Δx=8mm 格子（t=2.3〜2.87 sec.）

図 8　計算格子幅の違いによる反復履歴

反復解法には，RB-SOR（反復 4 回固定）を前処理に用いた CG 法を用いた。（b）のリスタートでは，リスタート時に保存則を満たしていないので反復回数が多くなるが，時間とともに収束していく。

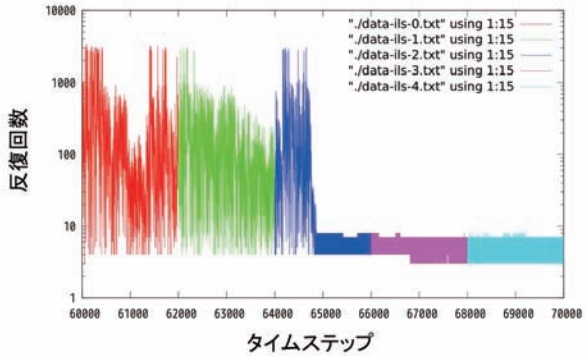


図 9　最も細かい格子での収束履歴（京での参考計算結果）

計算結果の可視化結果を図 10 に示す。解像度により，計算された流れ場の様子が異なることが分かる。両ケースとも，低解像度の計算結果である。最終的な 4mm での計算結果を図 11 に示す。非常に細かい渦が車体周りに生じていることが判る。

非表示

(a) Δx=16mm のケース

ルーフ後端で剥離が生じている。床下の流れの挙動は緩やか。

非表示

(b) Δx=8mm のケース

ルーフ後端での剥離がなくなり，床下にも渦が見える。車体後部で細かな渦が解像されている。

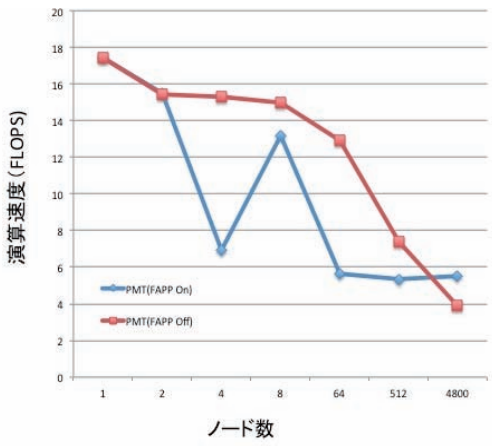
図 10　解像度による結果の違い

非表示

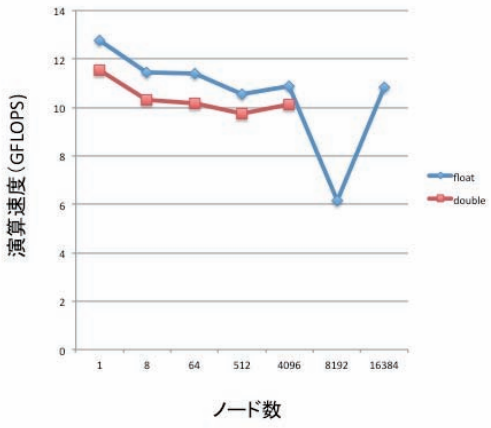
図 11　290 億格子を用いた車体周りの流れの様子

６．ベンチマーク

本節では,大規模計算のベンチマーク結果を述べる。ベンチマーク問題を FX10 と京で実施し，結果を比較する。ベンチマークは，FFV-C の組み込み例題として提供される三次元キャビティフロー問題を固定回数で反復する計算である。ノードあたりの要素数は，256x256x256(16M 点)とする Weak Scaling ベンチマークである。



(a) FX10　ノードあたり要素数は 256x256x256
倍精度，FAPP ON/OFF



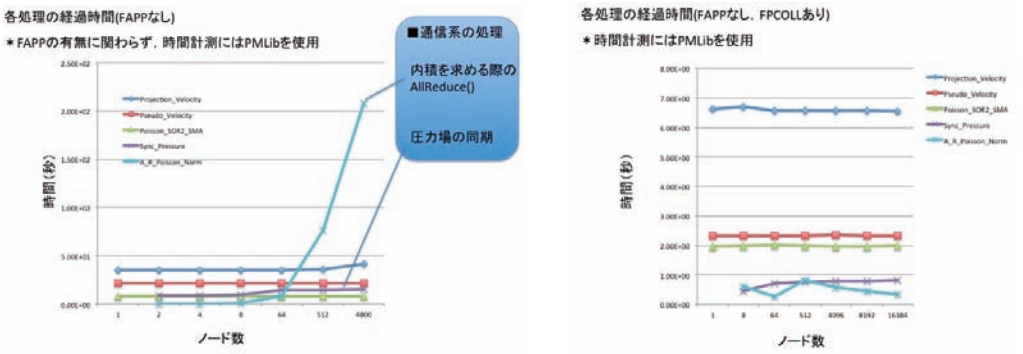
(b) 京　ノードあたり要素数は 128x128x128
FAPP OFF，単精度/倍精度

図 12　PMT ベンチマークのノードあたりの演算速度比較

まず，図 12 にノードあたりの演算性能を示す。問題規模は，FX10 では 16M 点，京では 2M 点とした。演算性能と通信性能のバランスの点では，FX10 の方が有利である。注目すべき点の一つは，FAPP を ON にしてハードウェアカウンタを用いた性能測定を行うと，測定性能が低下するケースがあった。FAPP については無視できないオーバーヘッドが生じていると思われる。そこで，FAPP を使わず PMlib での測定を実施した。FX10 と京を比較すると，コア数とメモリバンド幅の性能は FX10 の方が高いため，実行性能も高くなっていることが確認できた。京では，単精度と倍精度の比較も実施した。単精度の方がバンド幅を有効に利用できるため，若干性能が高くなっている。

次に，通信性能の比較結果を図 13 に示す。FX 10 ではノード数が増加すると，同期通信系の処理時間が増える。特に，集団通信の Allreduce 処理が指数的に増加している。一方，京では通信時間はノード数に依らずほぼ一定である。これはハードウェア機構が働いていると推察できる。

スケーラビリティの点では，FX10 が有利な問題設定になっているが，結果は逆に京の方が良いという結果であった。これは，テスト期間中の FX10 は，通信にコストがかかっているためと思われる。



(a) FX10　　(b) 京

図 13　通信性能の比較

7．まとめ

本研究では，大規模な実用計算を行う上で課題となる点として，格子生成，流体ソルバーの並列化，可視化処理の 3 点について検討した。格子生成については，計算時の初期化段階で自動格子生成を行う方法を提案し，290 億の自動格子生成を 9,216 プロセスで実行し，計算可能な格子を生成できることを示した。計算については，FFV-C ソルバーは十分なスケーラビリティを示すことが判った。また，可視化については，直交等間隔データの場合，290 億格子程度であれば，時間は多少かかるが，市販の可視化ソフトウェアでも可視化が可能であることが判った。

また，ベンチマークプログラムを FX10 と京で実施した。この結果，テスト期間中の FX10 は通信処理が何かの原因で遅くなっているという結果となった。これについては，再度評価したい。また，計時については，開発した PMlib はばらつきの少ないよい測定結果が得られたことを付け加えておく。

参　考　文　献

[1] 梶島岳夫，「乱流の数値シミュレーション」，養賢堂，1999.

[2] 赤坂 啓，小野 謙二，「非水密なポリゴン要素で構成された任意形状周りの直交格子を用いた非圧縮粘性流れ解析」，日本機械学会論文集 B 編，第 76 巻 764 号，pp. 536-545，2010.

[3] http://vcad-hpsv.riken.jp/jp/