オブジェクト指向並列化クラスライブラリの開発と性能評価

小野謙二[†,††] 玉木剛[†††] 野田茂穂[†]
岩田正子[†] 重谷隆之[†]

物理シミュレーションの領域分割型並列アプリケーション開発の効率化のため，オブジェクト指向技術によるフレームワークと並列化クラスライブラリを設計・開発した．開発した並列化データクラスは，配列に対する並列処理の機能をまとめたクラスライブラリであり，C/C++ と Fortran に対応している．フレームワークに組み込まれた並列化データクラスを用いて，楕円型偏微分方程式の反復解法をモデル化したベンチマークを f90 と C++ の混合コードで並列化した．このフレームワーク機能を利用したコードを Linux クラスタ上で並列実行した結果，非フレームワーク MPI 版と同程度の実行性能とスケーラビリティを確認した．

Development of Object-oriented Parallel Class Library and Performance Evaluation of Benchmark Code

Kenji Ono,[†,††] Tsuyoshi Tamaki,[†††] Shigeho Noda,[†]
Masako Iwata[†] and Takayuki Shigetani[†]

An Object-Oriented framework and class libraries with parallelism are designed and developed to enhance the construction of parallel physical simulators, which adopt domain decomposition method. Developed parallel data class is a class library that integrates a data structure and functions to manipulate the arrays for parallelisation. This data class can be used in both C/C++ and Fortran language. A benchmark code, which abstracts the part of a Jacobi iteration method for the elliptic partial differential equation, was ported using present framework and parallelized in C++/f90 mixed language. Measured timing results on Linux cluster showed us that the ported benchmark code using functions of the framework has almost the same performance and scalability of the original MPI benchmark code.

1. はじめに

物理現象の解析や工業製品の設計のため，シミュレーションは不可欠な技術となっている．特に，製品開発ではコストや性能の検討のため，高精度な計算結果を迅速に得たいというエンジニアの要求が高い．また，複雑な物理現象の解明のため，異なる物理現象の連成解析や非線形マルチスケール現象を扱うシミュレーション研究が進んでいる．これらの先端的な研究は，必然的に分散並列の大規模解析となる傾向にある．

従来から，流体解析や構造解析では領域分割型の並列計算法が研究され，並列化粒度の大きな効率の良い計算手法が提案されてきた．これらは MPI を用いた並列プログラムであるが，デバッグを含めたコード開発とメンテナンスが難しい点が問題であり，開発支援の仕組みが切望されている．この目的のために，オブジェクト指向プログラミング（Object-Orientd Programming，OOP）によるクラスライブラリ[1)~3)]，フレームワーク[4)~6)]，デバッガ[7)] などの並列プログラミング支援環境が研究されてきた．

複数の物理ソルバを連成したマルチフィジックス解析では，複数ソルバの実行制御・管理の問題が顕在化している．物理シュミレーションの数値解法は，各々の現象に対して効率の良い固有の解法として発展し，専門性の強い研究領域を形成している．1 人の研究者ですべての現象解析をカバーすることは難しいため，連成解析のシステム開発にあたり，研究者間のコラボレーションを促進する仕組みには大きな期待が寄せられている．具体的には，プログラム開発のガイドライン，あるいは共通機能を持つフレームワークを利用す

† 理化学研究所
RIKEN (The Institute of Physical and Chemical Research)
†† 北海道大学
Hokkaido University
††† 株式会社富士通長野システムエンジニアリング
Fujitsu Nagano Systems Engineering Co., Ltd.

ることにより，開発効率やプログラムの品質が向上すると考えられる．

プログラムの開発支援と実行管理の問題点に対して，著者らは物理シミュレーションのひな型の提供と複数のソルバコードを管理し，選択・連成実行可能なアプリケーションの機能を持つオブジェクト指向フレームワークを提案している[8]．このフレームワークは，オブジェクト指向技術の積極的な利用により，非定常/定常物理シミュレーションのソフトウェア構造の標準化・統一化を推進している．また，アプリケーションの開発効率化・高品質化・メンテナンス性の向上に貢献し，先端シミュレーション技術の迅速なパッケージングが期待できる．

本論文では，提案のフレームワークの概略と並列化データクラスの設計と実装について述べる．データクラスは並列化と連成問題を扱うためのものであるが，本論文では特に並列化に焦点をあてて詳しく説明する．さらに，このフレームワークを用いて楕円型方程式の反復解法プログラムの実行性能を測定し，並列性能について報告する．

以下，2 章では，フレームワークの概要と実装の関連部分について簡単に述べ，3 章では並列化データクラスの機能や実装，利用方法について詳しく述べる．4 章では具体的なベンチマークプログラムを用いてシステムの実行性能評価と考察を行う．

2. アプリケーションフレームワーク

著者らの提案するフレームワーク SPHERE[8]（Skeleton for PHysical and Engineering REsearch）は，時間発展型の物理シミュレーションへの適用を想定し設計している．登録した複数のプログラムに対して，起動時にキーワードを与えて選択実行が可能なアプリケーションとしても機能する．ユーザインタフェースに関しては，可読性・再利用性の高い XML により，パラメータ記述の統一を図った．このような機能の組合せにより，操作の一貫性を確保することを狙っている．

SPHERE は複数のソルバを登録できるプログラム構造となっている．この点を考慮すると，フレームワークにおける最大のクラス単位としては，ソルバとして機能するプログラム単位が適切であると考えられる．このクラスをソルバクラスと定義し，オブジェクト指向技術の流儀に従ったプログラミングを行う．通常のアプリケーションとしてのプログラム単位をソルバクラスとして扱い，複数のソルバクラスをフレームワークに登録する．

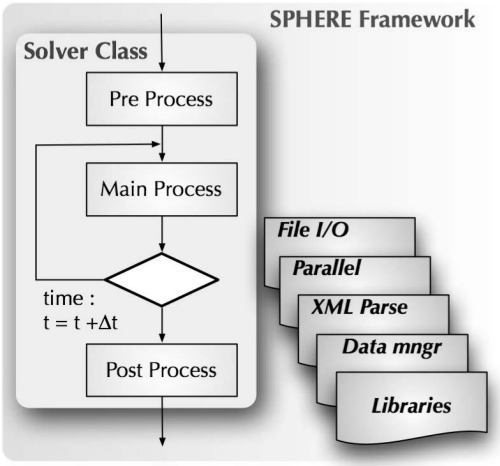

図 1 SPHERE フレームワークにおける抽象クラスの骨格とクラスライブラリ群
Fig. 1 Skeleton of abstracted class and class libraries in SPHERE framework.

SPHERE はオブジェクト指向言語である C++で記述されている．物理シミュレーションコードはある程度共通の構造でプログラムを記述できる点に着目して，プログラムを概念的にプリ・メイン・ポストの 3 つのプロセスに分割して考え，**図 1** のような制御構造を記述したプログラムのひな形（スケルトン）をソルバクラスの基底クラスとして作成する．フレームワークの利用者は，基底クラスを継承したソルバクラスを派生させ，派生したクラスに必要な処理をクラスメソッドとして実装し，プログラムを構築する．また，フレームワークは物理現象の解析プログラムを作成するうえで便利な関数群を提供する．この関数群は，データ管理，XML パラメータファイルのパース，ファイル入出力などの共通機能を持つ．

OOP ではプログラムの記述性を高めるために，クラスに対する演算が可能なオペレータオーバロードを用いた記述が用いられる．これはソースコードの可読性を高めるメリットがある一方で，テンポラリオブジェクトの生成をともなうため実行効率が上がらないという問題点がある[9]．一般に，科学技術アプリケーションは高い実行性能が求められるので，この点はデメリットとなる．これを軽減するため expression templates[10] と呼ばれる技法が提案され，C++ ライブラリである POOMA[2] で採用されている．POOMA はデータ並列言語で偏微分方程式の並列プログラムを記述し，C++ でも高い実行性能を確保している．提案のフレームワークでは，より簡単な実装により実行性能を確保している．具体的には，オペレータオー

バロードを抑制することと，配列変数をアドレス渡しの引数としてメソッドに渡すことにより，expression templates と等価な効果を得ている．

SPHERE はコンパイラがサポートするインタオペラビリティ，つまり混合言語プログラミングに対応している．フレームワークの main 関数は C++であるが，ここから呼ばれる関数として C，Fortran 言語による関数・サブルーチンの利用が可能である．これは，ソルバクラス内において Fortran サブルーチンをクラスメソッドとして実装するためである．システムコールなどの細かな記述は C/C++で，実行性能が重要視される処理は Fortran で記述することができ，言語の特性をうまく使い分けることができる．

以上に述べたように，手続き型の処理に従うクラス抽象化と混合言語のプログラミング環境の提供により，既存のソフトウェア資産である Fortran コードの移植性・親和性が高い点は，他の OOP 型フレームワークにはない本フレームワークの大きな特徴である．

3. 並列化データクラス

並列化処理と連成問題の取扱いは，フレームワークが担う重要な機能の 1 つである．分散並列計算の場合，プログラミングと境界条件の実装が複雑になる．SPHERE ではデータクラスを導入して，これらの低レベルの処理の記述を隠蔽し，概念的なレベルでのプログラミング環境を提供する．本論文では構造格子における領域分割型の並列処理を例として議論するが，本フレームワークでは扱えるデータ構造，計算手法，並列化手法について限定はない．

太田ら[11] は，CFD 解析における格子生成・計算・可視化などのプロセスを統合的に扱う目的で，OOP 技術を用いてプリ・ポスト・ソルバをモノリシックに構成した統合的な計算環境を構築し，データへのアクセス・並列管理を考慮したデータクラスの提案と評価を行った．このシステムでは，計算格子・変数・領域情報などの変数類をまとめたデータクラスに対して，流体計算プロセスの計算過程が独立に作用するクラス設計を行っている．さらに，領域分割型の並列化処理における境界条件処理と領域間の同期処理を共通インタフェースとして実装し，計算処理と並列処理の分離を試みている．これにより，逐次プログラムから並列プログラムへの移行は，簡単なライブラリコールの追加のみにより可能になる．また，ファイル入出力の記述性の向上にも寄与している．

一方，本フレームワークの目的は並列化・連成処理を含めた物理シミュレーションプログラムの開発支援と実行管理である．企業における実際の設計業務の現状を鑑み，プリポストは分離したシステム構成としている．設計で用いる実際の CFD システムは，不完全な形状データの修正や格子作成の試行錯誤のために専用ソフトの利用や商用アプリケーションを併用し，試行錯誤的な作業を行っているのが現状である．このため，格子生成プロセスと計算プロセスは分離されていることが多い．この点を考慮し，データクラスの抽象化を最もプリミティブな配列操作に限定している．これらの点で，フレームワークの利用目的と構成が明らかに異なる．

著者らの提案するデータクラスも基本的に文献 11) と同じ機能を持つが，実装は異なっている．本システムでは柔軟なプログラミング環境を提供するため，データクラスとそれらを管理するデータマネージャクラスの 2 層によりデータクラスを構成する．さらに，クラスの機能を多次元配列の管理と操作，および並列化に関する部分に限定した．これは，Fortran プログラムの移植性や混合言語でのアプリケーション開発，つまり，配列アドレスのポインタ渡しなど手続き型言語でプログラミングされたプログラムとの親和性，クラスメソッド経由のオーバヘッド抑制，プログラミングの柔軟性などを考慮しているためである．本フレームワークのクラス抽象化は，図 1 に示す最大単位であるソルバクラス，次いでプリ・メイン・ポストの各処理クラスであり，従来の手続き型のプログラムとの親和性が高い．これらの点が文献 11) との大きな違いとなっている．

以下に，各クラスの役割と機能について示す．

3.1 データクラスの役割と機能

データクラスは，一次元の配列データを多次元配列として扱うための機能を提供するクラス群である．データクラスは，内部で管理するデータ型をテンプレートとするテンプレートクラスとして実装されている．基底クラスは一次元の配列データを持つ．これから派生した n 次元のデータクラスは，n 次元のインデクスを用いて配列データにアクセスするインタフェースを提供する．

データクラスは，**図 2** に示す階層図のように，SklArayBase クラスを基底クラスとして，n 次元のインデックス情報のみを持つ SklArayidxN（N = 1, 2, 3, 4）クラス群，さらに SklArayidxN クラスにおいてスカラー/ベクトルを区別した SklScalar??, SklVector??クラスから構成される（?は任意の 1 文字）．SklVector?D と SklVector?DEx は，ベクトル要素の並びが異なるクラスで，SklScalar?D は逐次プログ

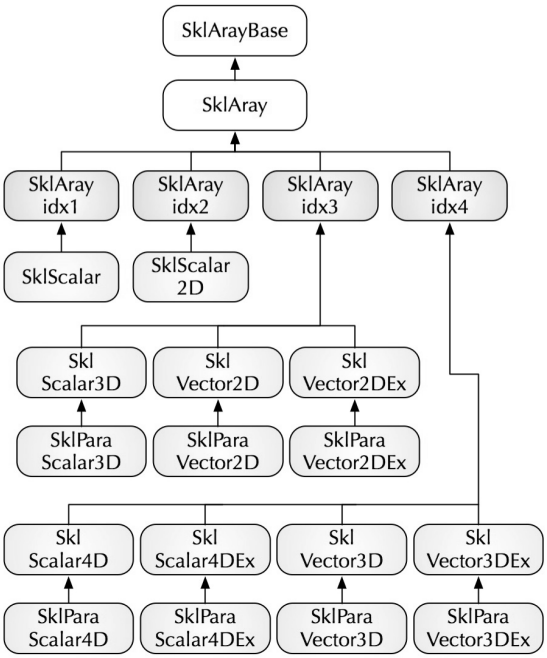

図 2 データクラスの階層図
Fig. 2 Hierarchy chart of data class.

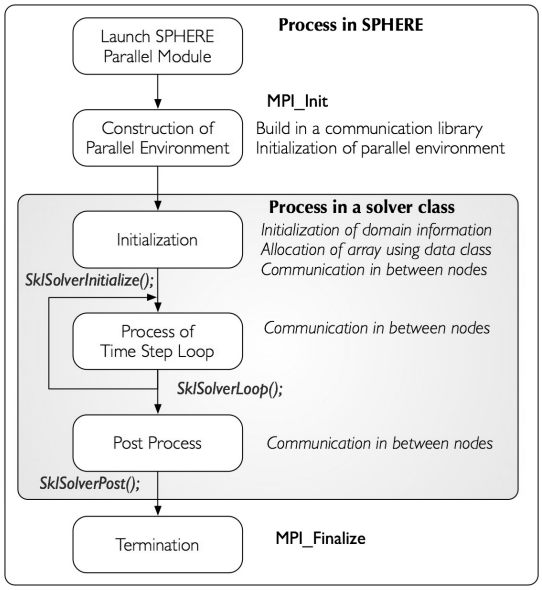

図 3 SPHERE の起動プロセス
Fig. 3 Launch process of SPHERE.

ラムで使用する標準データクラス，SklParaScalar?D は並列プログラムで利用する並列計算用のデータクラスを示す．データクラスは，配列データの実体に加えて，自身を他のデータクラスと区別するラベルやクラスタイプの識別子，データ型を識別する識別子を持つ．これらの情報はクラスメソッドを用いて取得することが可能である．また，外部境界条件設定や並列計算時の他領域とのデータ交換のためのバッファ領域となるガイドセルのサイズを，オブジェクト生成時に指定することができる．データクラスの共通機能としては，以下の機能がある．

1) ラベル名・クラスタイプ・データタイプの取得
2) 内部で保持している配列データ実体の先頭アドレスを返す実体データの取得
3) 配列サイズ・ガイドセルサイズの取得
4) 内部の実データにアクセスする n 次元のインデクスオペレータ
5) 並列実行時に仮想セルの情報を隣接するノード間で通信・更新するガイドセルの通信機能

3.2 マネージャクラスの役割と機能

マネージャクラスには，逐次処理のために利用されるデータマネージャクラスと並列処理用のパラレルマネージャクラスがある．データマネージャクラスは，データクラスの生成，登録，削除などの管理機能を持つ．データマネージャは，データクラスオブジェクトとオブジェクト生成時に指定されるラベルの組によって管理を行う．これにより，プログラムの必要な場所で，登録されているデータクラスオブジェクトを取得することができる．

パラレルマネージャクラスは，上記の機能に加え，並列実行用のデータクラス生成と並列実行環境の管理を行う．提供する機能として，以下のものがある．

1) 通信ライブラリの指定や並列実行可能性の判定
2) プロセスグループの作成を行う並列環境の初期化・終了メソッド
3) ノード ID，ノード数，隣接ノード ID などの並列実行環境の情報取得メソッド．ノード間通信は，C/C++ と Fortran に対応したメソッドを用意している．
4) ブロードキャスト，同期/非同期通信，ノード間演算処理，バリア，ガイドセルの更新などのノード間通信メソッド
5) 並列データクラスの実体化・登録メソッド

3.3 並列実行の処理の流れ

SPHERE では，マネージャクラスのオブジェクトをフレームワーク内に実体化し，これに対してソルバクラスがメソッドを用いてアクセスする実装としている．**図 3** に SPHERE 並列アプリケーションの起動時のプロセスを示す．アプリケーションが起動されると，まずノード間通信ライブラリの設定と並列実行環境の初期化を行う．その後，実行するソルバクラスのオブジェクトを生成し，個別の処理に入る．ソルバクラス

の内部は，プリ・メイン・ポストの 3 つの処理に分かれている．

領域の初期化では，計算空間の分割を行う．分割方法には，ユーザ指定とフレームワークによる自動分割がある．この自動分割機能は，領域分割型の並列処理に対応したもので MPI 並列を想定している．このとき，できるだけロードバランスが均等になり，かつ通信面の面積が最小になるように，実行プロセッサ数の情報をもとに三次元の各インデクス方向のループ分割数を決定する．並列化手法としては，MPI 並列のほかにもスレッド並列があり，両者を用いるハイブリッド並列もある．今後，マルチコアプロセッサを対象としたスレッド並列化支援の仕組みも取り入れる予定である．

並列プログラムの場合には，パラレルマネージャがデータクラスのオブジェクトの生成とデータマネージャへの登録を行う．これらのクラスの利用時には，並列実行の有無にかかわらずソルバを同一コードで記述できるように，実体化したオブジェクトを標準データクラスにキャストして使用する．

3.4 並列化関連クラス

図 4 に並列化関連のクラス階層を示す．SklBase クラスと SklParaIF クラスは抽象クラスである．SklParaIF クラスは，ノード間通信を行うクラスの基底クラスで，実際にはこのクラスから MPI 関数を用いて通信を行う SklParaMPI クラスを用いる．PVM や LAM など，他の通信ライブラリにも拡張できる．SklParaNodeInfo クラスと SklVoxelInfo クラスは，計算領域全体の格子数，ノード数，領域分割情報，自ノードのランクなどの並列処理情報を格納する．

SPHERE のソルバクラスは，内部に SklParaManager クラスのオブジェクトをメンバとして持ち，SklParaManager クラスに対して並列動作に関する要求を出す．**図 5** に並列化の機能を担当するクラスの構造を示す．SklParaManager クラスは，SklParaNodeInfo クラスと SklParaIF クラス，SklVoxelInfo クラスのオブジェクトをメンバとして持ち，(1) 外部（たとえば，ソルバクラス）からの要求に対して，(2, 3) 並列制御情報を返したり，(4–7) ノード間データ通信を行ったりする．並列環境に関する情報取得の要求は，SklParaManager のクラスメソッドのコールにより，(2, 3) クラスメンバである SklParaNodeInfo オブジェクトが保持する内容をソルバクラスへ返す．また，他ノードとの通信に関しては，(4, 5) SklParaMPI オブジェクトに対して通信要求を出す．このように，各ノードの通信は SklParaManager クラスがインタフェースとなっている．

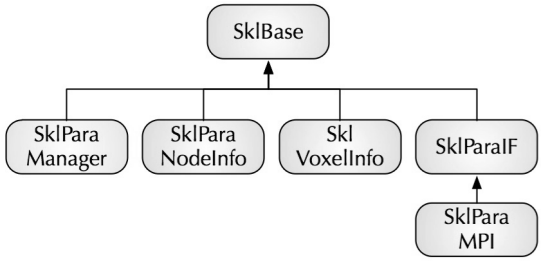

図 4 並列化に関連するクラスの階層
Fig. 4 Hierarchy chart for classes related to parallelisation.

3.5 ベンチマークコードの実装

次章で用いるベンチマークコードをフレームワークを用いて記述した疑似コードを示す．

プログラムの制御は図 1 のソルバクラス内の手続きに従い，SklSolverInitialize，SklSolverLoop，SklSolverPost がそれぞれプリ・メイン・ポスト処理に相当する．プリ処理では，計算領域の全要素数をパラメータ記述ファイルから読み込み，最適な分割数を選定する．続いて，並列データマネージャで配列領域の確保と初期化を行い，計算処理の初期化を行う．メイン処理では，ベンチマーク計算部分の hbmtf_jacobi を呼び出す．計算部分は f90 で書かれており，実際には Jacobi 反復法のアルゴリズムが書かれている．SklCommBndCell 関数の呼び出しにより配列 p の領域境界部分でガイドセルとして設けた 1 層分の通信を行っている．さらに，SklAllreduce 関数で gosa 変数の和をとっている．これらの領域間の同期処理と直後の境界条件処理は，文献 11) では共通インタフェース化している．本フレームワークでもその概念は意識しているが，実際は従来の手続き型と同様な実装をしている．これは，境界条件処理には様々なパターンがあり，必ずしも境界条件の設定と同期処理が同時に行われるとは限らないため，および多様な境界条件を追加・修正する場合のデータクラスの修正を避けるためである．

```
int SklSolverInitialize() {
 SklCfgGetVoxelSize(wi, wj, wk);
 GetCfgVoxelDivisionMethod(nx, ny, nz);
 ParaMngr.SklVoxelInit(wi,wj,wk,nx,ny,nz);
 VoxelInitilize();
 AllocateMatrix();
 InitilizeMatrix();
 return 1;
}
```

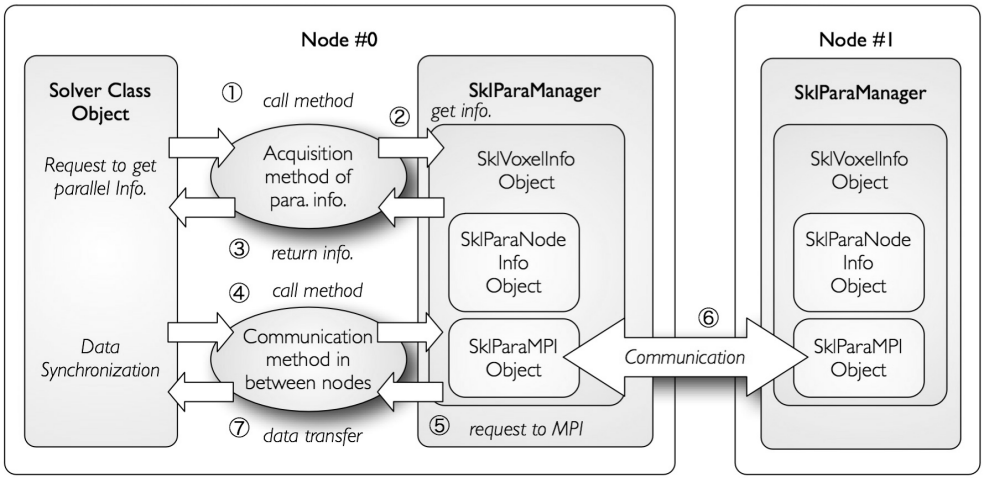

図 5 並列化に関連するクラスの構造およびソルバクラスとマネージャクラスの動作
Fig. 5 Class structure related to parallelisation and behavior between solver class and manager class.

```
int SklSolverLoop(const unsigned step) {
 hbmtf_jacobi_(nn, p, ...);
 return 1;
}

int SklSolverPost(void) {
 writeFile();
 return 1;
}

subroutine hbmtf_jacobi(nn, p, ...)
 integer, dimension(3) :: start, end
 real, dimension() :: p
 real(4) :: wgosa, gosa
 integer :: i,j,k,loop, nn, ierr
 do loop=1,nn
  gosa= 0.0
  do i,j,k=start(),end()
   p(i,j,k)=....
  enddo
  call SklCommBndCell(p, 1, ierr)
  call SklAllreduce(gosa, wgosa, ...)
  call BoundaryCondition()
 enddo
 return
end subroutine hbmtf_jacobi
```

表 1 評価用 PC の緒元と環境
Table 1 Specification and environment of evaluation PC.

CPU	Intel Pentium4 650（3.4 GHz, 2 MB cache）
Chipset	Intel i945G
Memory	2 GB（DDR2 533 MHz 1 GB × 2）
OS	FedraCore5 x86_64
Compiler	Intel Compiler 9.1（option = -O3）

4. 性能評価

開発したフレームワークを利用したアプリケーションの実行性能を確認するため，ベンチマークプログラムを用いて，逐次性能と並列性能を評価した．

4.1 評価システム

逐次性能評価の計算機は，**表 1** の諸元を持つ PC である．並列環境には，RIKEN Super Combined Cluster[12]（RSCC）を利用した．RSCC は**図 6** のように複合的な計算機群で構成されており，計算リソースの中心に合計 1024 ノード（2048CPU）の Linux クラスタシステム，単一プロセスで大量のメモリを要求するジョブに共有メモリベクトル計算機の SX-7 を有する．Linux クラスタはノード間を Infiniband（通信帯域：8 Gbps）で接続したクラスタ群であり，6 システムで構成されている．クラスタミドルウェアとしては，SCore を採用している．RSCC の中で今回の評価に利用したクラスタシステムとソフトウェアを**表 2** に示す．

4.2 ベンチマークプログラム

性能評価に用いたベンチマークプログラムは，楕円型方程式を Jacobi 反復法で解くプログラムである．こ

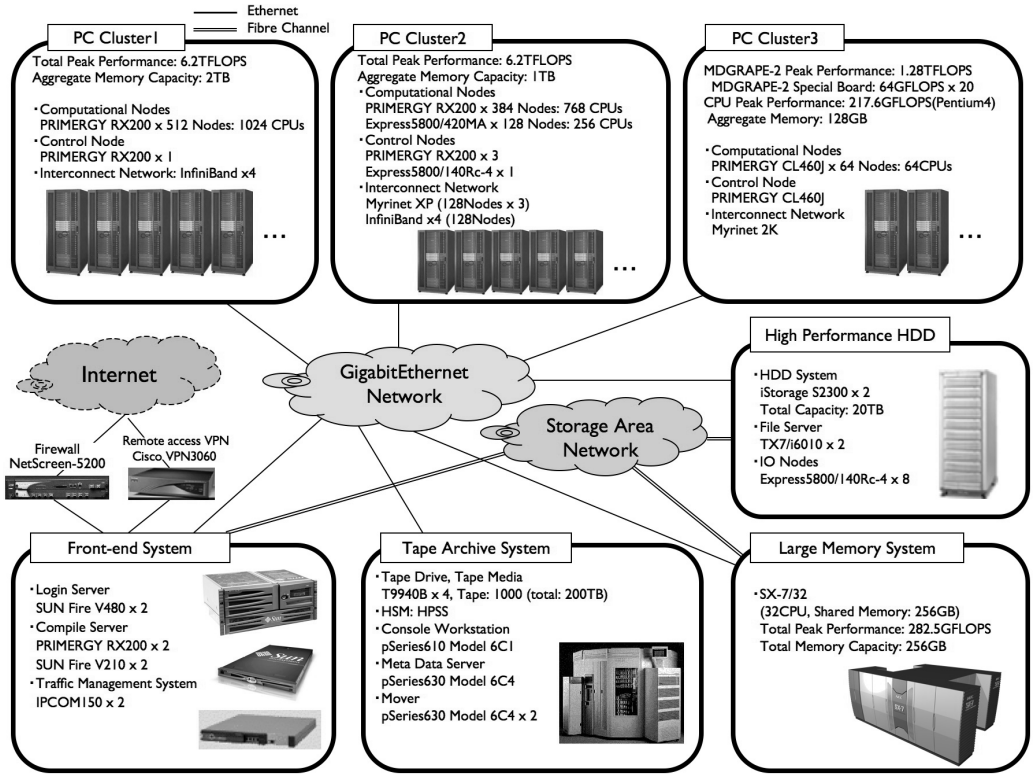

図 6 理研スーパコンバインドクラスタの構成
Fig. 6 Configuration of RIKEN super combined cluster.

表 2 RSCC の Linux クラスタの諸元と環境
Table 2 Specification and environment of Linux cluster in RSCC.

CPU	Intel Xeon（3.06 GHz，512 KB cache）2CPU/node
Memory	4 GB/node
OS	RedHat 8.0 kernel 2.4.12 + SCore∗
Middleware	SCore 5.6.1
Inter-connect	Infiniband（8 Gbps）
Compiler	Fujitsu Linux Parallel Language Package 1.0.0 (option = -Kfast，prefetch = 2，SSE2)

∗patch: Network Driver，PM/Myrinet，PM/InfiniBand

表 3 PC 上での逐次性能の測定結果
Table 3 Timing results for serial code on single CPU.

Code	Allocation	MFLOPS	Array size (i, j, k)
org_f77	static	1,021	256 × 128 × 128
org_f90	dynamic	907	256 × 128 × 128
org_C	static	1,144	128 × 128 × 256
org_C	dynamic	399	128 × 128 × 256
sph_f90	dynamic	921	256 × 128 × 128
sph_C++	dynamic	525	256 × 128 × 128
sph_C++	dynamic	537	128 × 128 × 256

れは，非圧縮性流体や電磁波のシミュレーションに現れる Poisson 方程式の求解プロセスとして用いられる．このベンチマークは，演算数とメモリのロードストアの比が約 1 対 1 であるため，演算速度に加えてメモリ帯域性能が大きく影響する特性がある．

オリジナルのプログラムは，C，Fortran77/90 などの言語，静的な配列と動的な配列への対応，逐次版と並列版などいくつかのバージョンが提供されている[13]．オリジナルコードの並列化は，領域分割による並列処理で MPI を用いて実装している．フレームワーク版のベンチマークプログラムは，オリジナルのプログラムからアルゴリズムを移植したものであり，並列化はオリジナルと同様に領域分割による並列処理である．

4.3 逐次性能

まず，PC 上で逐次性能の評価を行った．**表 3** にオリジナル版の Fortran と C のコード（org_∗），および SPHERE フレームワークを用いたコード（sph_∗）において，配列領域を動的および静的に確保した場合の実行性能の測定結果を示す．表中の MFLOPS 値は，ベンチマークコードで計算されるスコアである．

まず，オリジナルの C コードは動的なメモリ確保を行うと最適化が利きにくく，静的なメモリ確保を行っ

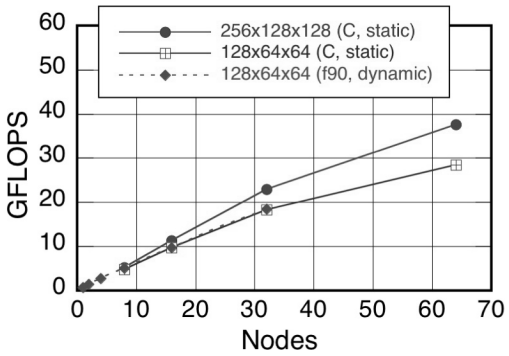

図 7 オリジナルコードの特性
Fig. 7 Characteristic of original codes on RSCC.

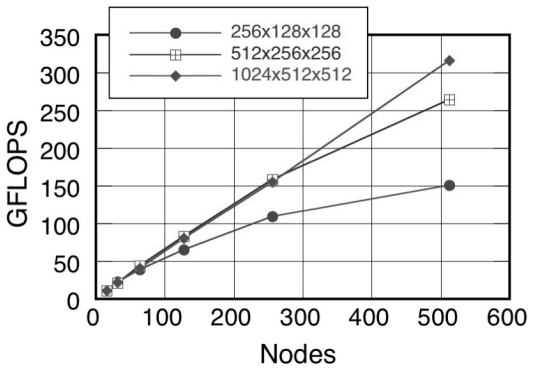

図 8 オリジナルコードの並列性能
Fig. 8 Parallel performance of original code on RSCC.

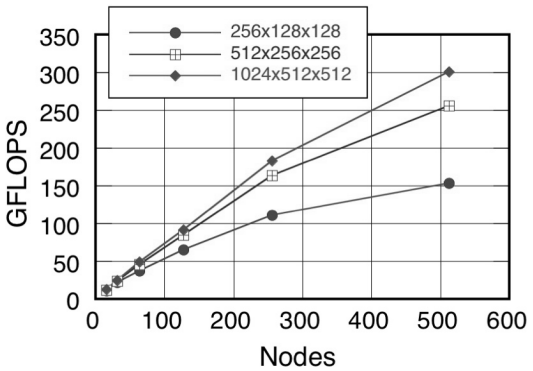

図 9 SPHERE コードの並列性能
Fig. 9 Parallel performance of SPHERE code on RSCC.

た場合の約 1/3 の実行性能となっている．一方，Fortran コードではメモリ確保の方式による顕著な実行性能の低下は見られない．これは RSCC におけるベンチマーク結果でも同様な傾向であった．

次に，SPHERE フレームワーク版の性能を見ると，f90 コードの結果はオリジナルとほぼ同じである．C++ コードの場合は配列サイズを 2 通り試しているが，オリジナルよりも有意に高い実行性能が得られている．これは，ループインデクスの計算でコンパイル時の最適化が利いているためと推察している．

したがって，単体性能としてはフレームワーク版はオリジナルと同程度以上の実行性能を持つと見ることができる．

4.4 RSCC における並列性能評価

まず，**図 7** に RSCC で測定したオリジナル版並列コードの性能測定結果を示す．ここではコンパイラとメモリ確保の方式に対する性能結果を比較する．横軸にはノード数，縦軸には実測性能を示す．配列サイズは，$256 \times 128 \times 128$（M），$128 \times 64 \times 64$（S）と表記する．配列サイズ S の場合，逐次性能測定の場合と同様に，f90 コードの動的メモリ確保と C コードの静的なメモリ確保のプログラムは，ほぼ同じ実行性能であることが確認された．また，配列サイズが大きくなるとスケーラビリティが向上する傾向が確認される．

次に，並列コードについてオリジナル版（f90）と SPHERE フレームワーク版（f90）の性能を RSCC の 16〜512 ノードを用いて測定した結果を**図 8** と**図 9** に示す．オリジナル版とフレームワーク版の双方とも配列サイズに対するスケーラビリティの傾向は同じである．つまり，配列サイズが小さい場合，ノード数が増加するに従い性能は飽和する．一方，配列サイズが大きくなると，スケーラビリティは向上する．全体の傾向を見ると，フレームワーク版の方が早く飽和している．これは，フレームワーク化のために若干のオーバヘッドが存在することを考えると当然の結果であろう．しかし，256 ノードまでの L，M，S の各配列サイズについて測定した実行性能は，むしろフレームワーク版の方が若干高い．これは，後述のメモリ配置の問題と考えられ，512 ノード程度の範囲内では，両者は同程度の性能と見るべきであろう．最終的に，SPHERE フレームワークを用いたコードは，RSCC の 512 ノード（1024CPUs）において 300 GFLOPS を超える性能を達成することが確認できた．**表 4** には，図 8 と図 9 の測定結果をオリジナル版とフレームワーク版の

表 4 RSCC におけるオリジナルと SPHERE コードの性能向上率（org_f90/sph_f90）
Table 4 Speed up ratio of original and SPHERE code on RSCC (org_f90/sph_f90).

Nodes	$256 \times 128 \times 128$	$512 \times 256 \times 256$	$1024 \times 512 \times 512$
16	1.00/1.00	1.00/1.00	1.00/1.00
32	2.01/2.06	1.87/1.99	1.92/1.91
64	3.46/3.52	3.92/3.88	3.62/3.90
128	5.87/6.13	7.33/7.25	7.15/7.25
256	9.79/10.50	13.95/13.94	13.83/14.42
512	13.53/14.43	23.37/21.84	28.19/23.74

性能向上率の点から示している．この表からも，解析規模が大きくなるとスケーラビリティが高くなる点が確認された．また，256 ノードまではフレームワーク版の方がスケーラビリティが高く，512 ノードのときにはオリジナル版の方が高くなっていることが分かる．

オリジナルのベンチマークコードと SPHERE フレームワーク利用のコードの実行性能を比較すると，両者はほぼ同じ性能を示すがノード数によって若干の違いが現れている．$1024 \times 512 \times 512$ の場合で，256 ノードのときには SPHERE コードの性能が高く，512 ノードのときにはオリジナルコードの方が性能が高い．メモリの配置により，このような問題が生じる場合がある．そこでフレームワークコードを利用して，配列サイズを固定したうえでガイドセルのサイズを 1〜10 まで変更し，PC の逐次ジョブとして性能を測定した．その結果，実行性能は 853〜1024 MFLOPS の間を示し，配列のメモリ上の配置に依存して性能差が出ることを確認した．並列実行時の性能差も，この点に起因していると考えられる．

5. ま と め

物理シミュレーションプログラムの開発支援のため，オブジェクト指向フレームワーク SPHERE を開発した．このフレームワークに，並列化データクラスを実装し，ベンチマークプログラムにより性能評価を実施した結果，次のことが明らかとなった．

1) 動的配列領域確保を行う C++ と f90 の混合言語コードの逐次性能は，静的なメモリ確保を行うプログラムの 9 割程度の実行速度である．
2) RSCC における並列性能は，SPHERE が提供する領域分割の仕組みがうまく機能し，スケーラブルな性能を達成でき，512 ノード，1024CPUs で 300 GFLOPS 超の性能を達成した．

現在のところ，フレームワークで用意したデータクラスとしては構造格子のみであるが，今後，八分木構造や非構造格子などを順次追加の予定である．

なお，本フレームワークは，理化学研究所 VCAD システム研究プログラム[14] から配布を行っている．

謝辞 本研究の一部は，新エネルギー・産業技術総合開発機構（NEDO）の平成 16 年度産業技術研究助成事業の助成を受けている．また，理化学研究所の共同計算機システム RSCC を利用している．

参 考 文 献

1) Balay, S., Gropp, W., McInnes, L.C. and Smith, B.: PETSc 2.0 Users Manual, Rv. 2.0.16, Technical Report ANL-95/11 (1997).
2) http://www.nongnu.org/freepooma/ (1990).
3) Baden, S.B., Colella, P., Shalit, D. and Van Straalen, B.: Abstract KeLP, *10th SIAM Conference on Parallel Processing for Scientific Computing*, Portsmouth, Virginia (March 2001).
4) Wijesinghe, H.S., Hornung, R.D., Garcia, A.L. and Hadjiconstantinou, N.G.: Three-dimensional Hybrid Continuum-Atomistic Simulations for Multiscale Hydrodynamics, *Journal of Fluids Engineering*, Vol.126, pp.768–777 (2004).
5) Hornung, R.D. and Kohn, S.R.: Managing Application Complexity in the SAMRAI Object-Oriented Framework, *Concurrency and Computation: Practice and Experience* (*Special Issue*), Vol.14, pp.347–368 (2002).
6) Henshaw, W.D.: Overture: An Object-Oriented Framework for Overlapping Grid Applications, *AIAA conference on Applied Aerodynamics* (2002). also UCRL-JC-147889.
7) 松田勝之，武宮　博：データ可視化機能を持つ並列プログラムデバッグツール：vdebug，技術報告，JAERI–Data/Code 2000–014 (2000).
8) 小野謙二，玉木　剛：SPHERE—物理シミュレーションのフレームワークと実行環境の開発，日本計算工学会論文集，No.20060031 (2006).
9) Bulka, D. and Mayhew, D.: *Efficient C++ Performance Programming Techniques*, Addison-Wesley (1999).
10) Veldhuizen, T.: Expression Templates, C++ Report, Vol.7, No.5, pp.26–31 (1995).
11) 太田高志，白山　晋：オブジェクト指向フレームワークによる流体計算統合環境，日本計算工学会論文集，No.19990001 (1999).
12) http://accc.riken.jp/rscc/index.html
13) http://accc.riken.jp/HPC/HimenoBMT/index.html
14) http://vcad-hpsv.riken.jp

(平成 18 年 10 月 6 日受付)
(平成 19 年 1 月 18 日採録)

小野 謙二

昭和 41 年生．平成 2 年熊本大学大学院工学研究科修了．同年日産自動車（株）入社，中央研究所勤務．数値流体力学の車両開発への適用に従事．平成 12 年熊本大学大学院自然科学研究科生産システム科学専攻修了，博士（工学）．平成 13 年東京大学大学院工学系研究科助教授，Intelligent Modeling Laboratory 担当．平成 15 年ワシントン大学応用物理研究所客員研究員．平成 16 年独立行政法人理化学研究所ものつくり情報技術統合化研究プログラム製品機能シミュレーションチームチームリーダー．平成 18 年同所 VCAD システム研究プログラム機能情報シミュレーションチームチームリーダー．平成 17 年より北海道大学大学院工学研究科人間機械システムデザイン専攻客員助教授（併任）．数値流体力学，CFD システム開発，大規模データ可視化の研究に従事．IEEE，日本機械学会，日本流体力学会，可視化情報学会，自動車技術会各会員．

玉木 剛

昭和 45 年生．平成 7 年信州大学大学院工学系研究科情報工学専攻修士課程修了．同年（株）富士通長野システムエンジニアリング入社．並列環境での流体解析プログラムおよび可視化アプリケーションの開発に従事．

野田 茂穂

昭和 40 年生．平成 3 年信州大学工学部機械工学科学士卒業．同年富士通長野システムエンジニアリング（株）入社．数値流体解析の開発と適用のコンサルティングに従事．平成 16 年より信州大学大学院工学系研究科後期課程入学．平成 17 年より理化学研究所出向．主として血流のシミュレーションシステムの研究開発に従事．日本流体力学会会員．

岩田 正子

昭和 32 年生．昭和 56 年お茶の水女子大学理学部物理学科卒業．化学技術系コンサルタント会社にて高分子の用途開発に従事．平成 10 年お茶の水女子大学大学院修士課程理学研究科情報科学専攻修了．平成 13 年お茶の水女子大学大学院人間文化研究科博士後期課程複合領域科学専攻修了．平成 13 年（特）理化学研究所ものつくり情報技術統合化研究プログラム製品機能シミュレーションチーム研究員．平成 18 年（独）理化学研究所 VCAD システム研究プログラム機能情報シミュレーションチーム研究員．数値流体力学，高分子物理等をテーマとする．理学博士．日本物理学会，日本機械学会各会員．

重谷 隆之

昭和 41 年生．平成 8 年東京都立大学大学院博士課程修了．同年理化学研究所奨励研究員．平成 9 年理化学研究所・情報環境室（現情報基盤センター）技師．共同利用計算機の設計，運用，管理に従事．理学博士．