

実時間ゲリラ豪雨予測システムのための ファイルI/O調停ミドルウェア

齋藤 智之¹ 石川 裕² Gerofi Balazs² Guo-Yuan Lien² 三好 建正² 大塚 成徳² 富田 浩文²
西澤 誠也²

概要: 30 秒毎の最新気象観測データから 30 分後のゲリラ豪雨を予測するシステムを開発している。本システムでは、100 ケースの 30 秒アンサンブル気象シミュレーション結果と 30 秒毎の最新気象観測データを同化する。気象シミュレーションプログラムとデータ同化プログラムは別々に開発されており、データ交換はファイル渡しになっている。ファイル渡しによる非効率なデータ転送ではなく、気象系で使われている netCDF API を拡張し、2 つのジョブのプロセス間で効率の良いデータ転送を提供する File I/O Arbitrator (FARB) を提案する。MPI 通信ライブラリを通信層とした実装を行った。既存ファイル I/O を用いた場合に比べて、3 倍以上の性能向上、2.6 秒以上の I/O 処理時間削減を達成している。

File I/O Arbitrator Middleware for Real-Time Severe Weather Prediction System

TOMOYUKI SAITO¹ YUTAKA ISHIKAWA² GEROFI BALAZS² GUO-YUAN LIEN² TAKEMASA MIYOSHI²
SHIGENORI OTSUKA² HIROFUMI TOMITA² SEIYA NISHIZAWA²

Abstract: A high-resolution 30-minute numerical weather prediction system, using weather observable data that is available every 30 seconds, is being developed. In this system, 100-case ensemble simulations and data assimilation of results and observed data are performed every 30 seconds to predict 30 minute weather. Because the simulation and data assimilation programs are independently developed, data is exchanged via files. The middle-ware of a file I/O arbitrator, called FARB, is proposed to reduce the data exchange overhead between simulations and data assimilation. FARB extends the netCDF API, widely used as file I/O API in weather/climate applications, so that data is directory transferred between two jobs. FARB has been implemented using the MPI communication library. It performs more than three times faster than file I/O operations, and it reduces more than 2.6 second I/O processing time.

1. はじめに

我々は、次世代型フェーズドアレイ気象レーダーで取得可能な 30 秒毎の 3 次元観測データや次期気象衛星ひまわり 8 号・9 号で可能となる 2 分 30 秒毎の日本周辺雲画像データを用いて、30 秒毎に更新する 30 分天気予報を実現する実時間天気予測システムの研究開発を進めている [1]

*1. 本システムでは、30 秒サイクルで以下のジョブを実

行する：i) 天気予測シミュレーションを同時に 100 ケースシミュレーションするジョブ群実行、ii) これら結果と観測データを同化するジョブ実行、iii) 30 分あるいは 1 時間後の予測シミュレーションジョブ実行。現在、シミュレーションプログラムも同化プログラムも独立して開発されており、データはファイルから入出力している。計算ノードの性能向上に比べファイル I/O の性能向上は今後も大きな飛躍はない。実時間処理のためにはファイル I/O によるジョブ間のデータ交換でなく、プロセスのメモリ領域を直接 RDMA (Remote Direct Memory Access) 機能を用いて交換するミドルウェアを実現しデータ交換時間を短縮する

¹ 東京大学 情報理工学系研究科

² 理化学研究所 計算科学研究機構

*1 科学技術振興機構 CREST 「「ビッグデータ同化」の技術革新の創出によるゲリラ豪雨予測の実証」(研究代表者：三好建正)

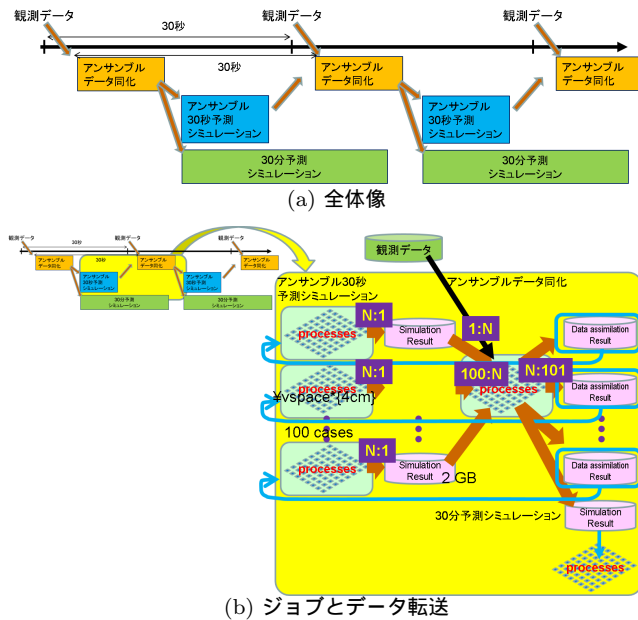


図 1 実時間ゲリラ豪雨予測システム

必要がある。

本稿では、実時間天気予測システムのために必要とされるジョブ間でのデータ転送を効率的に提供するカップリングミドルウェアである File I/O 調停ミドルウェア FARB を提案する。FARB は気象・気候シミュレーション系で使われている netCDF API[2] を拡張し、プロセスが保持するデータをそのデータを必要とする他のジョブのプロセスに直接転送する機構を提供する。次節においてゲリラ豪雨予測のための実時間天気予測システムを紹介した後、第 2 節において FARB を設計実装する。第 4 節において想定する問題サイズおよびデータ交換パターンにおける実行時間を FARB, MPI, netCDF, POSIX ファイル I/O で比較し、FARB の有効性を示す。

2. データ同化による実時間ゲリラ豪雨予測システム

2.1 概要

図 1(a) にシステムの全体像を示す。30 秒サイクルで、100 ケースアンサンブル 30 秒予測シミュレーション、100 ケース予測結果と観測データとの同化、データ同化された結果に基づく 30 分予測シミュレーションを繰り返す。このため、アンサンブルシミュレーションとデータ同化は 30 秒以内に終了しなければならない。データ同化結果は次のフェーズのアンサンブルシミュレーションの初期値として、また、30 分予測シミュレーションの初期値として使われる。

観測データとしては、次世代型フェーズドアレイ気象レーダー [3] や気象衛星ひまわりのデータがインターネット経由で受信されることを想定している。次世代型フェーズドアレイ気象レーダーからは、30 秒間隔で反射強度、ドッ

ラー速度、速度幅の 3 パラメータ (37.6MB/parameter) が取得できる。

数値天気予報モデルとしては、理研で開発されている LES 気象モデル SCALE[4]、気象庁で現業で運用されている NHM モデルも使う。データ同化には、局所アンサンブル変換カルマンフィルタ (LETKF) [5] を使用する。これらプログラムは MPI 並列化されており、いずれも独立して開発されており、入力データ、出力データはファイルに格納される。

本稿で想定する問題サイズは、水平方向 100 メートル解像度で 120 Km x 120 Km の領域、垂直方向 80 グリッドである。モデルが扱う変数は 16 であり、生成されるデータ量は 13.7GiB となる。

$$1200 \times 1200 \times 80 \times 16 \times 8\text{Byte} = 13.7\text{GiB}$$

100 ケースシミュレーションでは、約 1.3TiB のデータ量となる。

データ転送の観点で気象予測シミュレーションコードを概観する (図 1(b))。気象予測シミュレーションの並列化は領域分割され、各 MPI プロセスは担当領域を計算する。一つのファイルから各 MPI プロセスが担当する領域データが分配される。シミュレーション結果をファイルに格納するために、各 MPI プロセスが保持しているデータが集約される。ファイルアクセスパターンを入力ファイル数：プロセス数：出力ファイル数で表現すると、気象予測シミュレーションコードは、1:N:1 アクセスパターンといえる。なお、SCALE では並列ファイル I/O ライブラリ netCDF[2] を使ったファイルアクセスも可能である。

LETKF データ同化コードは、アンサンブルシミュレーション結果と観測データを同化する。観測データにはノイズが載っているため、品質保証のためにノイズ除去すると使える観測データ領域は動的に変わる。このため、シミュレーションと同じ領域分割による並列化では、モデルが扱っている分割領域周辺に観測データが存在しない領域を扱っているプロセスの計算負荷は少なくなり、負荷がアンバランスになる。第 2.2 節および第 2.3 節において、アンサンブルシミュレーションジョブとデータ同化ジョブの計算ノード割り当ておよびデータ交換パターンについて説明する。

2.2 ジョブ割り当て

本論文では、1 シミュレーションは 180 ノードで計算し、データ同化は 18,000 ノードで計算することを想定する。100 アンサンブルシミュレーションジョブ群とデータ同化ジョブは同時に実行しないため、18,000 計算ノードを交互に利用できる。図 2 に 100 アンサンブルシミュレーションジョブ群とデータ同化ジョブがシミュレーション領域をどのように分割して実行するかを示す。

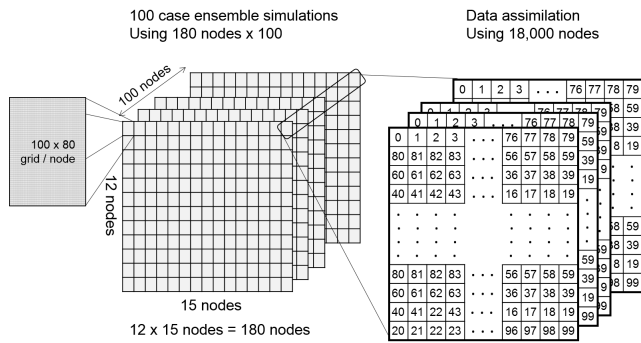
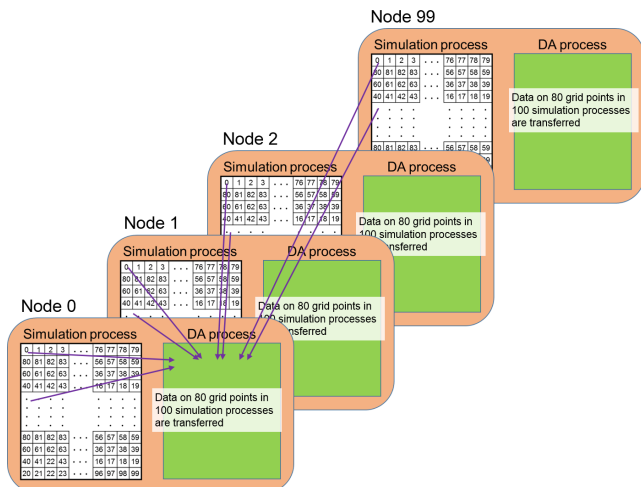


図 2 実行イメージ



計算ノード0のデータ同化プロセスに集められるデータを表現している。各シミュレーションプロセスから80要素のデータが転送される。

図 3 データ交換パターン

図2の左側は、アンサンブルシミュレーションジョブ群が実行している時のジョブ割り当てを示している。ここで、1つの面は1つのシミュレーションの領域を示している。1シミュレーションは180計算ノードで実行され、計算領域が分割されて処理される。1200 x 1200の格子に対して180計算ノードを割り当てる。すなわち、1計算ノードあたり100 x 80の格子を計算させる。

100アンサンブルシミュレーションにより、各100 x 80格子領域は100通りの結果をそれぞれのプロセスが保持している。100通りの100 x 80格子領域毎に100個の計算ノードがデータ同化する。図2の右側は、データ同化ジョブの計算ノードがアンサンブルシミュレーションプロセスが保持している各100 x 80格子領域のどの領域を計算するかを示している。本図においてシミュレーション領域の右上の100 x 80領域に対する100ケースのシミュレーション結果と観測データの同化は同じ100の計算ノードで計算される。それぞれの計算ノードを0から99の番号でラベル化したとすると、各格子点の数字はその格子点をデータ同化する計算ノード番号を示している。

表 1 FARB API

API 名	概要
nc_put_vara_init	書き込みパターンの初期化
nc_get_vara_init	読み込みパターンの初期化
nc_startall	データ交換の開始
nc_waitall	データ交換終了待ち

2.3 データ転送パターン

100アンサンブルシミュレーションプロセス群(180プロセス x 100)が保持している格子領域をデータ同化プロセス18,000にどのように分配するかを説明するために、図2の右側で扱っている100 x 80格子領域の分配方法を図3に示す。図3は、各計算ノード上で実行しているシミュレーションプロセスとデータ同化プロセスを図示している。各計算ノードには、同一100 x 80格子領域を異なる初期値で計算しているシミュレーションプロセスとデータ同化するプロセスが同居している。

データ同化プロセス100個毎に100 x 80格子領域の100アンサンブル結果と観測データを同化させる。シミュレーションが保持している100 x 80格子領域を1次元のストライド長100のベクトルにして各データ同化プロセスに渡す。各アンサンブルシミュレーションプロセスがデータ同化プロセスに転送するデータサイズは

$$80 \text{ 格子点} \times 80 \text{ 垂直方向} \times 16 \text{ 変数} \times 8 \text{ Byte} = 819 \text{ KB}$$

となる。

本ジョブ割り当てでは、100アンサンブルシミュレーションプロセス群が保持している格子領域に対して100のデータ同化プロセスを割り当てており、100計算ノードに閉じていることになる。言い換えると図2の右側の各100 x 80格子領域毎に100個の計算ノードを使って100アンサンブルのシミュレーションプロセスと100データ同化プロセスがデータ交換するだけで、他の領域を処理しているアンサンブルプロセスやデータ同化プロセスとはデータ交換しない。すなわち、このような領域分割では、対象領域が大きくなっても計算ノード数を増やすことによってウィークスケールすることになる。なお、本論文では、これ以降、この100計算ノードで実行されているシミュレーションプロセス群とデータ同化プロセス群を「連成計算データ交換グループ」と呼ぶことにする。

3. 設計と実装

FARBは、netCDF APIを拡張している。新たに導入しているAPIを表1に示す。MPI通信ライブラリが提供する永続通信(Persistent Communication)と同様のインターフェイスをデータ書き出し、読み出しAPIに適用している。また、netCDFが提供しているファイル生成/オープンAPI nc_create, nc_openの引数の一つであるモードにNC_FARBという新たなモードを導入している。NC_FARBを

```
/* initialization part */
nc_create("data.nc", ncmode, &ncid);
nc_def_dim(ncid, "Y", dimlen[0], &dimids[0]);
nc_def_dim(ncid, "X", dimlen[1], &dimids[1]);
nc_def_var(ncid, "V0", NC_INT, 2, dimids, &vid[0]);
nc_def_var(ncid, "V1", NC_INT, 2, dimids, &vid[1]);
nc_enddef(ncid);

/* main computation */

/* data output */
for (i = 0; i < 2; i++)
    nc_put_vara(ncid, vid[i], startp, countp, buf+i*N);
```

図 4 netCDF による書き込みコード例

```
/* initialization part */
nc_open("data.nc", ncmode, &ncid);
nc_inq_var(ncid, "V0", &vid[0]);
nc_inq_var(ncid, "V1", &vid[1]);

/* data input */
for (i = 0; i < 2; i++)
    nc_get_vara(ncid, vid[i], startp, countp, buf+i*N);
```

図 5 netCDF による読み込みコード例

指定することにより, FARB API が利用できるようになる.

netCDF で記載されているプログラムをどのように修正して使うかを示すために, 図 4 および図 5 に netCDF のプログラム例を示す. 図 4 はファイルを作成しているプログラムで, ファイルフォーマットを定義した後, nc_put_vara 関数を使ってデータを書いている. 図 5 はファイルを読み込むプログラムで, nc_get_vara 関数を使ってデータを読んでいる.

図 6 は, 図 4 のプログラムを FARB で修正したプログラムである. 主計算およびデータ書き出しの前に, nc_put_vara_init API を呼び, データ書き出しパターンを登録する. 実際の書き込みは, nc_startall API を呼び出すことにより行われる. 本 API は非同期処理で, 実際にデータが書き出されるのを待つためには nc_waitall API を呼び出す.

図 7 は, 図 5 のプログラムを FARB で修正したプログラムである. 主計算およびデータ読み込みの前に, nc_get_vara_init API を呼び, データ読み込みパターンを登録する. 実際の書き込みは, nc_startall API を呼び出すことにより行われる. 本 API は非同期処理で, 実際にデータが読み込まれるのを待つためには nc_waitall API を呼び出す.

図 8 に FARB アーキテクチャを図示する. 今後, 次世代システムソフトウェアスタックの一つとして設計している

```
NC_Request req[2]; NC_Status stat[2];
/* initialization part */
nc_create("farb:0", NC_FARB, &ncid);
nc_def_dim(ncid, "Y", dimlen[0], &dimids[0]);
nc_def_dim(ncid, "X", dimlen[1], &dimids[1]);
nc_def_var(ncid, "V0", NC_INT, 2, dimids, &vid[0]);
nc_def_var(ncid, "V1", NC_INT, 2, dimids, &vid[1]);
nc_enddef(ncid);

for (i = 0; i < 2; i++)
    nc_put_vara_init(ncid, vid[i], startp, countp, buf+i*N, req+i);

/* main computation */

/* data transfer */
nc_startall(2, req);
nc_waitall(2, req, stat);
```

図 6 FARB による書き込みコード例

```
NC_Request req[2]; NC_Status stat[2];
/* initialization part */
nc_open("farb:0", NC_FARB, &ncid);
nc_inq_var(ncid, "V0", &vid[0]);
nc_inq_var(ncid, "V1", &vid[1]);

for (i = 0; i < 2; i++)
    nc_get_vara_init(ncid, vid[i], startp, countp, buf+i*N, req+i);

/* data transfer */
nc_startall(2, req);
nc_waitall(2, req, stat);
```

図 7 FARB による読み込みコード例

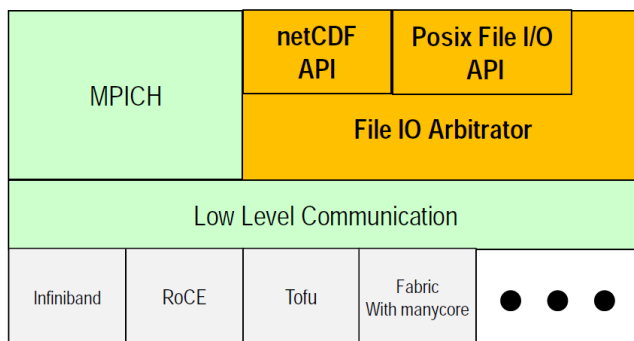


図 8 FARB アーキテクチャ

LLC(Low Level Communication Library)[6] 上に実装する予定でいる. LLC は RDMA 通信による one sided 通信を基本として, その上に two sided 通信機能等を構築している. LLC は, PC クラスタで主流の Infiniband ネットワーク, 京や FX10 の Tofu ネットワークでの実装を進めており, FARB は広い計算機プラットフォームで利用できるようになる. LLC は開発中で利用できないため, 本稿では, FARB の下位通信層として MPI 通信ライブラリを使用して実装した.

表 2 Oakleaf-FX の仕様

Specifications	
CPU (SPARC64™IXfx)	
Number of Cores	16
Clock	1.848 GHz
Memory	32 GB
Memory Bandwidth	85 GB/s
Interconnect (Tofu Interconnect)	
Bisection Bandwidth	5 GB/s
Topology	6-D Mesh/Torus

4. 評価

4.1 ベンチマークプログラム

本システムが目標とする 30 秒以内での 100 アンサンブルシミュレーションとデータ同化サイクルを実現するために必要とされる計算資源は、計算ノード性能およびインターコネクトネットワーク性能に依存する。第 2 節で示した通り、連成計算のためのデータ交換パターンは書き込みをするプロセス群と読み込むプロセス群から構成される。使用する計算ノード数によって連成計算のためのデータ転送容量も変わってくる。そこで、本システムのデータ交換パターンとデータ転送容量を $C(E, N, M, X)$ で表現する。 $C(E, N, M, X)$ のパラメータの意味は以下のとおりである。

- E : データ書き込みをするプロセスグループ数 (アンサンブル数)
- N : プロセスグループ内のプロセス数 (1 シミュレーションのプロセス数)
- M : データ読み込みをするプロセス数 (データ同化プロセス数)
- X : データサイズ

第 2 節で述べたとおり、本稿で想定するシステムでは、 $C(100, 180, 100, 819KB)$ となる。また、通信の観点では、 $C(100, 1, 100, 819KB)$ の性能は、一つの 100×80 格子領域における連成計算データ交換グループ内での通信性能となる。連成計算データ交換グループ内の通信はそのグループ内のプロセス間通信に閉じているため、ネットワークハードウェア的にも連成計算データ交換グループ内通信が閉じていれば $C(100, 1, 100, 819KB)$ の実行時間と $C(100, 180, 100, 819KB)$ の実行時間は同じになる。

4.2 FARB 性能

ベンチマークプログラムを東京大学情報基盤センターの Oakleaf-FX (富士通製 FX10) を用いて FARB のデータ交換能力を計測した。Oakleaf-FX の仕様を表 2 に示す。図 9 に $C(100, 1, 100, X)$ where $16kiB \leq X \leq 4MiB$ のデータ交換における通信にかかる時間を示す。計測区間を線形補間すると、転送サイズ $S(KB)$ に対する実行時間 $T(msec)$

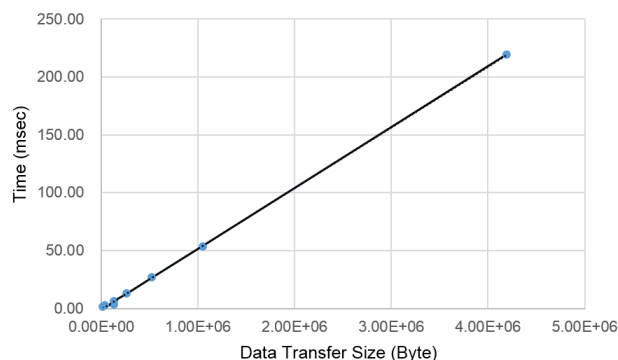


図 9 $C(100, 1, 100, X)$ における通信時間

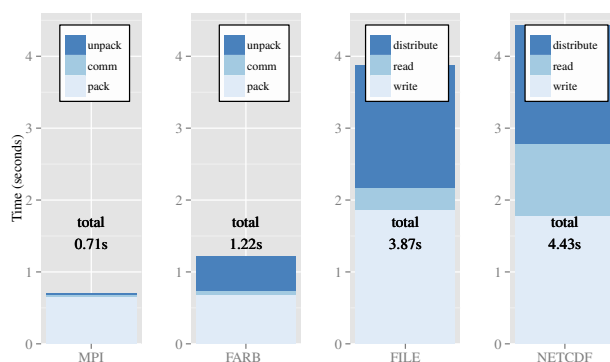


図 10 $C(100, 1, 100, 819KB)$ におけるデータ交換時間内訳

は以下の式となる。

$$T = S/20$$

想定データサイズ $S=819KB$ における実行時間はおよそ 41 msec となる。目標実行時間 30 秒に対してシミュレーションとデータ同化間の通信時間が 41 msec なので、通信コストは問題にならないことがわかる。

次に、 $C(100, 1, 100, 819KB)$ のデータ交換におけるデータのバック、アンパックも含めた時間について計測した。図 10 は、4 つのデータ交換手法の実行時間を比較している。MPI とは、通信ライブラリでデータ交換パターンを書き下した時の性能である。FARB では変数毎に書き出し/読み込み API が呼ばれているので、変数毎のバック・アンパック処理をしている。MPI 実装では変数領域がメモリ上で連続していることを利用して一括コピーしている点が違う。FILE とは、MPI 通信ライブラリでデータを集約して代表プロセスがファイルを生成、代表プロセスがファイルを読んでデータを分散した時の時間である。netCDF は、オリジナル netCDF の時間である。ファイル I/O や netCDF に比べて 3 倍以上の性能向上をしているが、MPI 通信ライブラリに比べて 1.7 倍遅くなっている。バックおよびアンパックの最適化が不十分のためであり、今後、バックおよびアンパックの最適化をしていく必要がある。実行時間の観点では 1.22 秒で終了しており、シミュレーションおよびデータ同化プロセス合わせて 28 秒以上の実行時間を使用

できることになる。

先に述べたとおり、本稿で想定するシステムのデータ交換パターン $C(100, 180, 100, 819\text{KB})$ の実行時間は、FARB においては $C(100, 1, 100, 819\text{KB})$ の実行時間とほぼ変わらないだろう。一方でファイルシステムを使用する FILE および netCDF の性能に関しては、連成計算データ交換グループ毎にファイルシステムが存在するのならば、FARB と同様、想定システムのデータ交換時間は今回測定した結果と同じになるだろう。しかし、京コンピュータや FX10 においては、ローカルファイルシステムは計算ノード群で共有されているため、1 つの連成計算データ交換グループのデータ交換時間は増加すると考えられる。

5. 関連研究

5.1 カップリング

FARB は複数アプリケーションを連成実行するためのカップリングミドルウェアである。カップリングミドルウェアとしては、Model Coupling Toolkit (MCT)[7], [8], OASIS4 [9] や O-PALM [10] などがある。FARB が提供する機能と大気、海洋、大陸などのマルチフィジックスモデルを連成計算する地球システムモデルで使われているカップリング機能 [11] との関係性を以下にまとめる。

(1) アプリケーション間データ交換

MCT や OASIS4, O-PALM はデータ交換のために独自の API を定義している。一方、FARB は netCDF の API を踏襲し拡張することによってアプリケーション間で効率よくデータを交換できるようにしている。

(2) データ再構成 (regrid)

地球システムモデルの連成計算では、それぞれのモデルが異なる解像度によるシミュレーションを行う場合がある。このためカップリング機能として解像度に応じた再データ構成が必要となる。FARB は同じ解像度を持つ領域を共有することを想定しているため、このような機能を提供していない。汎用カップリングとして、データの再構成 (regrid) を可能とするためには、再構成に必要なアルゴリズムを定義するインターフェイスが必要となる。

(3) 時間発展管理

カップリングソフトウェアである OASIS4 [9] や O-PALM [10] は、アプリケーション実行タイミングを制御できる。例えば、OASIS4 では Driver と呼ばれる連成計算のための制御インターフェイスを提供している。Driver は、連成計算しているプログラム群の実行スケジュールとデータ交換方法を記述した設定ファイルに従って連成計算を制御する。FARB はこのような機能を提供していない。

5.2 POSIX IO API

FARB は netCDF ライブラリを拡張することにより、既存 netCDF ライブラリを使って記述しているアプリケーションプログラムに対して最小限の変更でアプリケーション間で高速にデータ交換が出来るようにしている。同様に、POSIX IO API[12] で記述されているアプリケーションプログラムに対して最小限の変更でアプリケーション間で高速にデータ交換が出来るようになれば、多くのアプリケーションの連携が可能となる。しかし、POSIX IO API は構造体のようなデータ構造を読み書きする API ではない。アプリケーションプログラムは POSIX IO API を用いて構造体データのメモリ領域を連続して書いているに過ぎない。ライブラリレベルでアプリケーションが読み書きしているデータ構造を知る方法がないために、FARB のような拡張をすることが出来ない。

POSIX IO API に対しては、ファイル単位で受け渡しを行う連成計算に対して直接データ交換を行う拡張が考えられる。

6. おわりに

ゲリラ豪雨予測のための実時間天気予測システムに必要とされるジョブ間データ転送ミドルウェア FARB を設計し MPI 通信ライブラリを用いた実装による予備実験の結果を示した。FARB は、アンサンブルシミュレーションジョブの各プロセスが保持するデータをそのデータを必要とするデータ同化ジョブのプロセスに直接転送する機構を提供する。これによりファイル I/O 処理ならびにデータ並び替えの処理を削減することが出来る。予備実験により、1 連成計算データ交換グループにおける既存ファイル I/O を用いた場合に比べて 3 倍以上の性能向上、実行時間としては 3.87 秒から 1.22 秒に短縮することがわかった。

想定システムでは 1 連成計算データ交換グループ数が 100 倍となる。FARB はグループ数が 100 倍になっても通信時間が増加する要因はない。FARB を使うことによりアンサンブルシミュレーションおよびデータ同化ジョブを 30 秒以内に終了させるために使用できる実行時間は、26.13 秒から 28.78 秒に増加することになる。一方で、既存ファイル I/O の場合にはファイルシステムと計算ノードの構成に依存するが、必ずしもグループ数 1 とグループ数 100 において同じ実行時間を保証するものではない。既存ファイルシステムにおけるスケーラビリティ問題の調査を今後進めていく予定である。また、本結果を踏まえて、今後、FARB の改良、実装、評価をしていく。

謝辞

本研究の一部は、科学技術振興機構 CREST「科学的発見・社会的課題解決に向けた各分野のビッグデータ利活用推進のための次世代アプリケーション技術の創出・高度化」

領域のなかの課題名「「ビッグデータ同化」の技術革新の創出によるゲリラ豪雨予測の実証」(研究代表者:三好建正)による.

- [12] IEEE, T. and Group, T. O.: The Open Group Base Specifications Issue 7.

参考文献

- [1] 齋藤智之, 石川 裕, Balazs, G., 三好建正, 大塚成徳, 富田浩文, 西澤誠也: ジョブ間データ転送方式の検討, 情報処理学会研究報告, Vol. 2014-HPC-143(2), pp. 1–6 (2014).
- [2] Li, J., Liao, W.-k., Choudhary, A., Ross, R., Thakur, R., Gropp, W., Latham, R., Siegel, A., Gallagher, B. and Zingale, M.: Parallel netCDF: A High-performance Scientific I/O Interface, *Supercomputing, 2003 ACM/IEEE Conference*, IEEE, pp. 39–39 (online), available from <http://www.ece.northwestern.edu/~choudhar/Publications/LiLia03A.pdf> (2003).
- [3] Ushio, T., Wu, T. and Yoshida, S.: Review of Recent Progress in Lightning and Thunderstorm Detection Techniques in Asia, *Atmospheric Research*, Vol. 154, pp. 89–102 (2015).
- [4] Tomita, H.: SCALE-LES: Strategic development of large eddy simulation suitable to the future HPC, *Solution of Partial Differential Equations on the Sphere* (2012).
- [5] Takemasa, Miyoshi, Yamane, S. and Enomoto, T.: Localizing the Error Covariance by Physical Distances within a Local Ensemble Transform Kalman Filter (LETKF), *Scientific Online Letters on the Atmosphere (SOLA)*, Vol. 3 (2007).
- [6] 石川 裕, 堀 敦史, Balazs, G., 高木将通, 島田明男, 清水正明, 佐伯裕治, 白沢智輝, 中村 豪, 住元小田和 友仁: 次世代高性能並列計算機のためのシステムソフトウェアスタック, 情報処理学会研究報告, Vol. 2013-OS-125(3), pp. 1–7 (2013).
- [7] Larson, J., Jacob, R. and Ong, E.: The Model Coupling Toolkit: A New Fortran90 Toolkit for Building Multiphysics Parallel Coupled Models, *International Journal of High Performance Computing Applications*, Vol. 19, No. 3, pp. 277–292 (online), available from <http://hpc.sagepub.com/content/19/3/277.full.pdf> (2005).
- [8] Jacob, R., Larson, J. and Ong, E.: M×N Communication and Parallel Interpolation in Community Climate System Model Version 3 Using the Model Coupling Toolkit, *International Journal of High Performance Computing Applications*, Vol. 19, No. 3, pp. 293–307 (online), available from <http://hpc.sagepub.com/content/19/3/293.full.pdf> (2005).
- [9] Redler, R., Valcke, S. and Ritzdorf, H.: OASIS4—a coupling software for next generation earth system modelling, *Geoscientific Model Development*, Vol. 3, No. 1, pp. 87–104 (2010).
- [10] Piacentini, A., Morel, T., Thévenin, A. and Duchaine, F.: O-PALM: An Open Source Dynamic Parallel Coupler, *Proceedings of the IV International Conference on Computational Methods for Coupled Problems in Science and Engineering—Coupled Problems* (2011).
- [11] Valcke, S., Balaji, V., Craig, A., DeLuca, C., Dunlap, R., Ford, R., Jacob, R., Larson, J., RO'Kuinghttons, Riley, G. ほか: Coupling Technologies for Earth System Modelling, *Geoscientific Model Development*, (online), available from <http://www.mcs.anl.gov/uploads/cels/papers/P4036-0213.pdf> (2012).