ISSN 2436-0066

TOHOKU
UNIVERSITY

東北大学
サイバーサイエンスセンター

大規模科学計算システム広報

SENAC

Vol.54 No.1 2021–1

Cyberscience
Center

Supercomputing System
Tohoku University
www.ss.cc.tohoku.ac.jp

大規模科学計算システム関連案内

＜大規模科学計算システム関連業務は、サイバーサイエンスセンター本館内の情報部情報基盤課が担当しています。＞

https://www.ss.cc.tohoku.ac.jp/

階	係・室名	電話番号(内線)* e-mail	主なサービス内容	サービス時間 平 日
一階	利用相談室	022-795-6153(6153) sodan@cc.tohoku.ac.jp 相談員不在時 022-795-3406(3406)	計算機利用全般に関する相談 大判プリンタ、利用者端末等の利用	8:30～17:15 8:30～21:00
	利用者談話室	(3444)	各センター広報の閲覧 自販機	8:30～21:00
	展 示 室*（分散コンピュータ博物館）*	*見学希望の方は共同利用支援係までご連絡ください。	歴代の大型計算機等の展示	9:00～17:00
三階	総務係	022-795-3407(3407) cc-som@grp.tohoku.ac.jp	総務に関すること	8:30～17:15
	会計係	022-795-3405(3405) cc-kaikei@grp.tohoku.ac.jp	会計に関すること、負担金の請求に関すること	8:30～17:15
	共同利用支援係 （受 付）	022-795-3406(3406) 022-795-6251(6251) uketuke@cc.tohoku.ac.jp	利用手続き、利用相談、講習会、ライブラリ、見学、アプリケーションに関すること	8:30～17:15
	共同研究支援係	022-795-6252(6252) rs-sec@cc.tohoku.ac.jp	共同研究、計算機システムに関すること	8:30～17:15
	ネットワーク係	022-795-6253(6253) i-network@grp.tohoku.ac.jp	ネットワークに関すること	8:30～17:15
	情報セキュリティ係	022-795-3410(3410) i-security@grp.tohoku.ac.jp	情報セキュリティに関すること	8:30～17:15
四階	研究開発部	022-795-6095(6095)		
五階	端末機室	(3445)	PC 端末機(X 端末)	8:30～17:15

＊（ ）内は東北大学内のみの内線電話番号です。青葉山・川内地区以外からは頭に 92 を加えます。

本誌の名前「SENAC」の由来

昭和 33 年に東北地区の最初の電子計算機として、東北大学電気通信研究所において完成されたパラメトロン式計算機の名前で SENAC-1(SENdai Automatic Computer-1)からとって命名された。

［巻頭言］

新スーパーコンピュータ AOBA 始動

東北大学サイバーサイエンスセンター
滝沢寛之

東北大学サイバーサイエンスセンターの新スーパーコンピュータ AOBA(あおば)が、2020 年 10 月 1 日より運用を開始しました。すでに本誌 SENAC でもいくつか紹介記事が掲載されていますが、巻頭言の場をお借りして、システム運用開始のご挨拶を一言申し上げます。

現在、スーパーコンピュータに親しみやすい愛称をつけることが一般的になっています。例えば、2020 年に稼働開始した理化学研究所計算科学研究センターの「富岳」などがその典型例として挙げられます。今回、東北大学サイバーサイエンスセンターでも、初めての試みとして新スーパーコンピュータの愛称を募集し、349 件ものご応募をいただきました。当初の想定を大きく上回る多数の応募をいただき、誠にありがとうございました。最終的に、愛称としてのふさわしさや親しみやすさを考慮して AOBA が選ばれました。AOBA は、応募総数の 1 割を超える 36 名の方々からご提案いただいた愛称です。その高い支持率を考えれば、多くの方にご賛同いただける親しみやすい愛称をつけることができたのではないかと思います。

言うまでもなく、AOBA という愛称は東北大学サイバーサイエンスセンターの位置する青葉山(あおばやま)キャンパスに因んだものです。仙台城址(本丸跡)周辺は青葉山と呼ばれており、その地域にある東北大学のキャンパスが青葉山キャンパスです。他にも、仙台駅周辺から山形県との県境まで広がる仙台市青葉区など、現在では「青葉」という地名や名称は仙台のいたるところで広く使われています。しかし、この地域がいつ頃から青葉山と呼ばれるようになったのか、結論から言うとあまり詳らかではないようです。仙台市図書館所蔵の「要説 宮城の郷土史」には、福島市の信夫山

にあった青葉山寂光寺というお寺が慶長 7 年(1602 年)に仙台城本丸近くに移され、それによってこの地に青葉山という地名が発生したという説が記述されています。このお寺の名前は青葉山(せいようざん)寂光寺と読むのだそうで、お寺自体はしばらくした後に別の場所(北山)に移されて明治には廃寺となっています。それにも関わらず、その山号に由来する地名だけは現在でも広く使われているということになり、長い歴史の中でいろいろな出来事や偶然が重なった結果として、この地域が青葉山(あおばやま)という地名になっていることが分かります。仙台には他にも多くの神社仏閣がありますが、その中で青葉山寂光寺の山号だけが独り歩きし、この地域全体を示すような地名にまで成長したとすれば興味深いですね。

2020 年 10 月、スーパーコンピュータ AOBA もこの青葉山の地に誕生しました。青葉山寂光寺の名称がこの地に定着して周りに大きな影響をもたらし続けているように、スーパーコンピュータ AOBA も青葉山の地から全世界に向けて成果を発信し続ける存在にしていきたいと思っています。そのために、東北大学サイバーサイエンスセンター スーパーコンピュータ運用関係者一同で一丸となって、利用者の方々のご期待に副えるように精進してまいります。

最後に、年明け早々暗い話は書きづらいので敢えて話題の中心にはしませんでしたが、2020 年は新型コロナウイルス感染対策に終始した年となりました。AOBA の構築にも多大な影響があったはずですが、多くの方々のご尽力のおかげで当初予定通りに運用を開始することができました。この場をお借りして御礼申し上げます。依然として新型コロナウイルスの猛威は留まることを知らず、先行きの見えない日々が続きますが、2021 年は少しでもよい年になることを祈念して、AOBA 運用開始記念のご挨拶とさせていただきます。なお、2020 年 10 月のスーパーコンピュータ AOBA の運用開始にあたって、多くの方々からお祝いの言葉をいただきました。コロナ禍の影響もあってシステム稼働式典を催すことはできませんでしたが、お寄せいただいたお言葉は AOBA の紹介動画とともに以下のアドレスでご覧いただけますので、ぜひご参照ください。

https://www.cc.tohoku.ac.jp/aobadebut.html

[大規模科学計算システム]

SX-Aurora TSUBASA でのプログラミング（並列化編）

— 共有並列化と分散並列化 —

林 康晴
日本電気株式会社

サイバーサイエンスセンターのスーパーコンピュータ AOBA に採用されている SX-Aurora TSUBASA は、Vector Engine(VE)カードを 8 枚搭載した Linux サーバー(Vector Host)が、InfiniBand の Fat-Tree ネットワークにより接続されたシステムです。各 VE カードは、8 個の CPU コアを搭載しています。SX-Aurora TSUBASA のハードウェア性能を十分に引き出すには、プログラムの並列化により、複数の CPU コアを有効に活用する必要があります。SX-Aurora TSUBASA は、並列処理環境としてプログラミング言語 Fortran、C、及び C++、並びに 通信ライブラリ MPI を用意しています。本稿は、第 1 章で並列処理の基本事項をご説明した後、第 2 章では NEC Fortran コンパイラ（以後、単にコンパイラと記します）による共有並列化、第 3 章では MPI ライブラリ NEC MPI による分散並列化を中心にご紹介します。

1. 並列処理

並列処理とは、一つの仕事を複数の小さな仕事に分割し、それらの仕事を複数同時に実行することです。例えば、2 重ループを四つの仕事に分割し、4 個の CPU コア上で並列実行すると、図 1.1 のようになります。この場合、外側の do j のループの 100 回の繰返しを 4 等分し、各 CPU コア上で並列に実行しています。

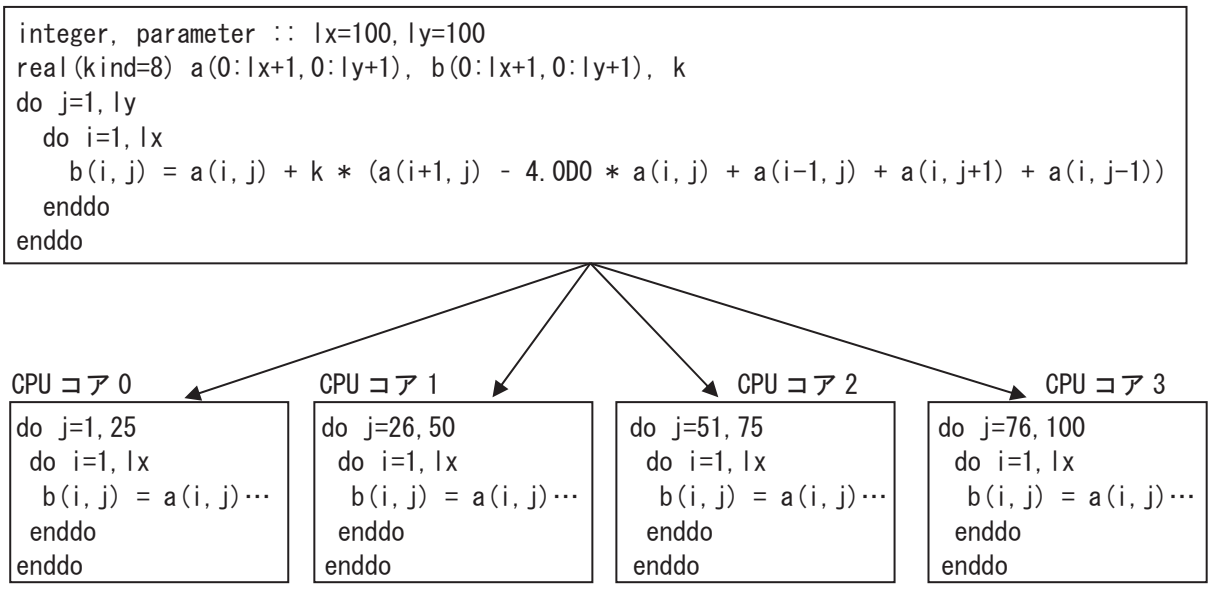


図 1.1　2 重ループの並列処理

1.1　並列化可能な条件

並列実行中、複数の仕事は、一般には実行タイミングの調整（同期）なしで、それぞれ独立に実行されます。そのため、異なる仕事内の計算の間には定まった実行順序はなく、一般には実行毎に異なる順序で実行されます。従って、ループが並列実行可能であるためには、ループの繰返しをどのような順序で実行しても実行結果が変わらないことが必要です。例えば、図 1.2 の二つのループ(a)、(b)は、いずれもループ中で確定される各配列要素 a(i) 及び a(i,1), (i=2,3,…,n) が、各ループ(a)、(b)の全ての繰返し中

でそれぞれ 1 度しか出現しません。そのため、ループの繰返しをどのような順序で実行しても結果は同じになるので、並列実行可能です。一方、図 1.3 の(a)、(b)の場合、いずれもループの k 回目の繰返しで値が確定される配列要素 a(k)が、(a)の場合 k+1 回目の繰返しで、(b)の場合 k-1 回目の繰返しでそれぞれ引用されます。そのため、ループの繰返しの実行順序に依存して、確定前の値を引用するか、確定後の値を引用するかが変わってしまい、並列化することはできません。図 1.3 の(c)の場合も、ループの実行終了時に変数 ifound に残る値が、ループの繰返しの実行順序に依存して変わってしまうので、並列化することはできません。このように、ループを並列化するためには、一つの繰返しで確定したデータ要素は、他の繰返しでは確定も引用もしないようにループを記述する必要があります。図 1.3 の(d)のように、ループ外への飛越しを含むループも、ループの繰返しの実行順序に依存して飛越しのタイミングが変わってしまうので、並列化することはできません。

図 1.4 の二つのループ(a)、(b)は、それぞれスカラ変数 t、s をループの全ての繰返しで確定するので、図 1.3 の条件に該当し、一見並列化できないように思えます。しかし(a)の場合、変数 t は、ループの繰返し毎に新たに確定した値を、その繰返しの中だけで引用しています。そのため、ループ実行後に変数 t の値を引用しないのであれば、仕事毎に作業変数を確保して、並列実行中は、変数 t の代わりにその作業変数を参照すれば並列化可能です。また(b)のように、同一の変数に同一の演算を繰返し適用するパターンは集計計算と呼ばれます。集計計算の場合も、やはり仕事毎に作業変数を確保して、並列実行中はその作業変数に各仕事の局所的な計算結果を格納し、並列実行終了時に全ての仕事の計算結果を集計することにより並列化できます。

多重ループの場合、並列化できるかどうかはループ毎に判断します。図 1.5 の内側の do i のループは並列化可能ですが、外側の do j のループは、左辺の a(i,j)と右辺の a(i,j-1)との間に図 1.3(a)と同様の依存関係があるので並列化できません。

```
integer, parameter :: n=100
real(kind=8), dimension(n) :: a, b
do i=2,n
  a(i) = b(i) + b(i-1)
enddo
```

(a)

```
integer, parameter :: n=100
real(kind=8), dimension(n,n) :: a
do i=2,n
   a(i,1) = a(i,2) + a(i-1,2)
enddo
```

(b)

図 1.2　並列化できるループ

```
integer, parameter :: n=100
real(kind=8), dimension(n) :: a
do i=2,n
  a(i) = a(i) + a(i-1)
enddo
```

(a) 依存

```
integer, parameter :: n=100
real(kind=8), dimension(n) :: a
do i=1,n-1
   a(i) = a(i) + a(i+1)
enddo
```

(b) 逆依存

```
integer, parameter :: n=100
real(kind=8), dimension(n) :: a
integer ifound
do i=1,n
  if(a(i).ge.0) ifound = i
enddo
```

(c) 出力依存

```
integer, parameter :: n=100
real(kind=8), dimension(n) :: a
do i=1,n
  if(a(i).ge.0) goto 100
enddo
100 continue
```

(d) 制御依存

図 1.3　並列化できないループ

```
integer, parameter :: n=100
real(kind=8) :: a(n), b(n), t
do i=1,n-1
  t = a(i) + a(i+1)
  b(i) = t
enddo
```

(a) 繰返し内作業変数

```
integer, parameter :: n=100
real(kind=8) :: a(n), s
do i=1,n
  s = s + a(i)
enddo
```

(b) 集計計算

図 1.4　作業変数の使用により並列化できるループ

```
integer, parameter :: n=100
real(kind=8), dimension(n,n) :: a
do j=2,n
  do i=1,n
    a(i,j) = a(i,j-1) + a(i,j)
  enddo
enddo
```

図 1.5　内側ループだけ並列化できる多重ループ

1.2 並列化の効果

一つのプログラムを N 個の CPU コア上で並列実行した場合、最大 N 倍のスピードアップ（実行時間の短縮）が期待できます。このスピードアップを最大にするには、まず、どれだけ多くの部分を並列化できたか（並列化率）が重要です。これは、図 1.6 のように逐次実行時間の 10 %分の仕事が並列化できない場合、残りの仕事をどれだけ多くの小さな仕事に分割して並列化しても、スピードアップは 10 倍未満になってしまうことからも分かります。

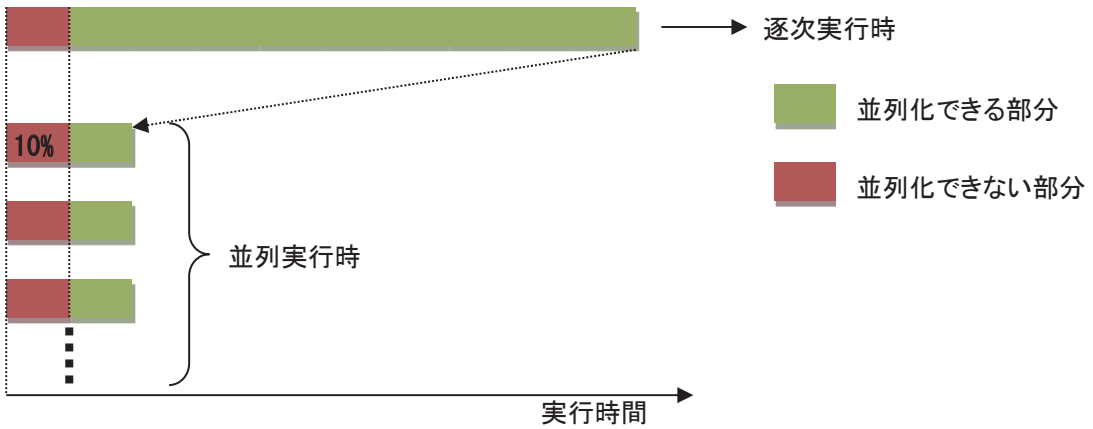


図 1.6　並列化率とスピードアップ（並列化率 90 %・スピードアップは 10 倍未満）

また、仕事を複数の小さな仕事に分割するための処理や、仕事間のデータのやり取り（通信）及び同期といった逐次実行時には必要なかった処理（オーバヘッド）をできるだけ小さくすることも重要です。これは、図 1.7 のように逐次実行時間の 10 %分のオーバヘッドが並列化により発生すると、スピードアップはやはり 10 倍未満になってしまうことからも分かります。特に、実行時間（粒度）の小さなループを並列化するような場合、オーバヘッドの方が大きすぎ、並列化によりかえって遅くなる場合もあります。オーバヘッド

を削減するには、できるだけ外側のループで並列化する、といった方法により、各仕事の粒度をできるだけ大きくすると効果的です。

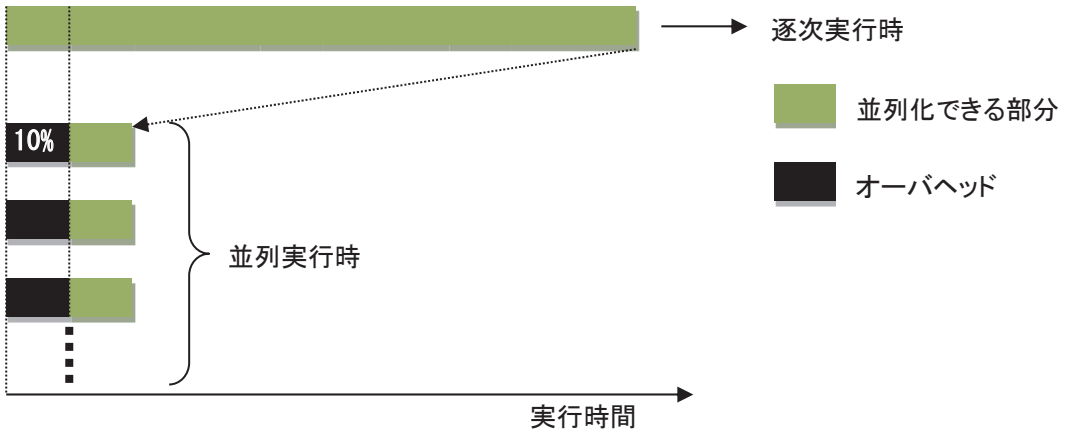


図 1.7　オーバヘッドとスピードアップ(オーバヘッド 10 %・スピードアップは 10 倍未満)

さらに、各仕事の実行時間のバランス(負荷バランス)をできるだけ均等化することも重要です。これは、図 1.8 のように複数の仕事のうち一つの仕事の実行に逐次実行時の 10 %分の時間がかかってしまったとすると、残りの仕事をどれだけ高速化しても、スピードアップは 10 倍以下になってしまうことからも分かります。

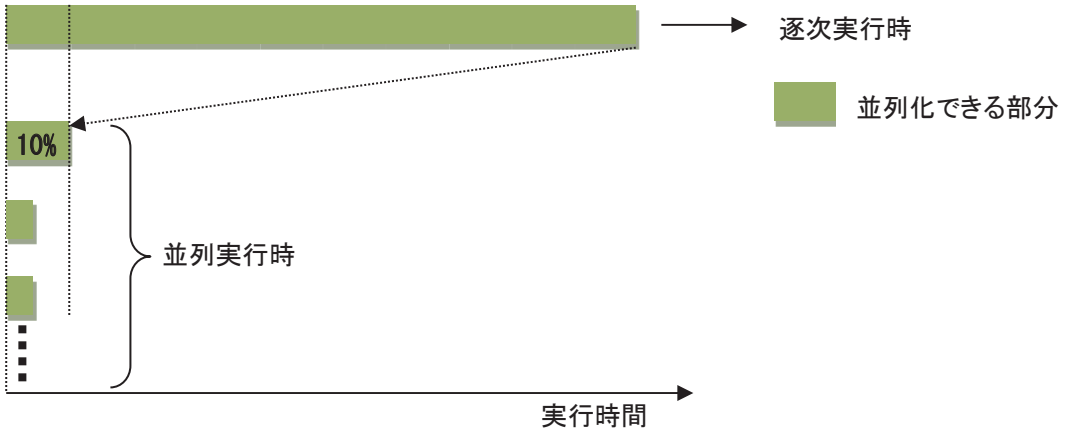


図 1.8　負荷バランスとスピードアップ(スピードアップは 10 倍以下)

このように、並列プログラムにおいて高いスピードアップを達成するには、高い並列化率・低いオーバヘッド・均等な負荷バランスの 3 点が重要です。

1.3　経過時間と CPU 時間

並列処理は、一つの仕事を複数の小さな仕事に分割して、複数の CPU コア上で同時に実行することにより、仕事の実行時間(経過時間)を減らします。一方、分割した全ての仕事を実行するための CPU コアの処理時間の総計(CPU 時間)を減らすことはできません。実際には、一つの仕事を複数の小さな仕事に分割するための処理や仕事間の同期などのオーバヘッドも発生するので、CPU 時間は並列化によりむしろ増加することに注意してください。

1.4 経過時間とチューニング

並列化の主たる目的であるプログラム実行の経過時間短縮のためには、アルゴリズムの工夫、高速なライブラリの使用、及びベクトル化の促進などにより、逐次プログラムとしての性能を並列化の前に十分チューニングしておくことが重要です。並列化によるスピードアップは、使用する CPU コアの個数が上限となりますが、逐次プログラムのチューニングによる高速化は、場合により数 10～数 100 倍に達することもあり、さらに CPU 時間も削減できるからです。

2. 共有並列化

一般に、コンピュータのプログラムは、プロセスにより実行されます。実行中のプロセスは、そのメモリ空間にプログラムのデータを格納しており、現在の実行状態をもっています。プロセスは、複数のスレッドから構成することもできます。一つのプロセス中の複数のスレッドは、それぞれ独自の実行状態をもち、別々の仕事を実行できますが、プロセスのメモリ空間は全てのスレッドが共有します。

共有並列化とは、一つのプロセス中の複数のスレッドによる並列処理のことです。SX-Aurora TSUBASA の各 VE カード内では、主記憶装置を共有する 8 個の CPU コア上で実行されるスレッドに仕事を割り当て、共有並列化を行うことができます。例として、図 1.1 の 2 重ループの共有並列化を考えます。第 1 章と同様に、外側の do j のループを四つの仕事に分割して別々のスレッドに割り当てます。この場合、データ領域に関しては、図 2.1 のように、2 次元配列 a、b を do j のループに対応する 2 次元目で分割し、分割された各部分領域の処理を各スレッドが担当することになります。ここで、例えば j=25 の繰返しを担当するスレッド 0 は、スレッド 1 の担当領域である配列要素 a(i,26), (i=1,2,…,lx) の値を引用する、といったように、配列の分割境界部分の処理では、自スレッドの担当領域外の配列要素を参照する必要があります。しかし、各スレッドは、メモリ上に配置された 2 次元配列 a、b の全ての領域を直接参照できるので、特に問題はありません。一方、DO 変数 i、j については注意が必要です。並列実行中、各スレッドは変数 i、j をそれぞれ独自のタイミングで参照します。そのため、配列 a、b と同様に、変数 i、j の領域を全てのスレッドが共有すると、あるスレッドが確定した変数 i、j の値を別のスレッドが引用してしまう可能性があり、実行が正しく行えません。従って、並列実行中に各スレッドが値を確定する変数は、図 2.2 のようにそれぞれのスレッドが固有の作業領域を割り付けて参照する必要があります。2 次元配列 a、b のような全てのスレッドがメモリ領域を共有するデータを SHARED データ、DO 変数 i、j のような各スレッドが専用のメモリ領域を割り付けるデータを PRIVATE データと呼びます。

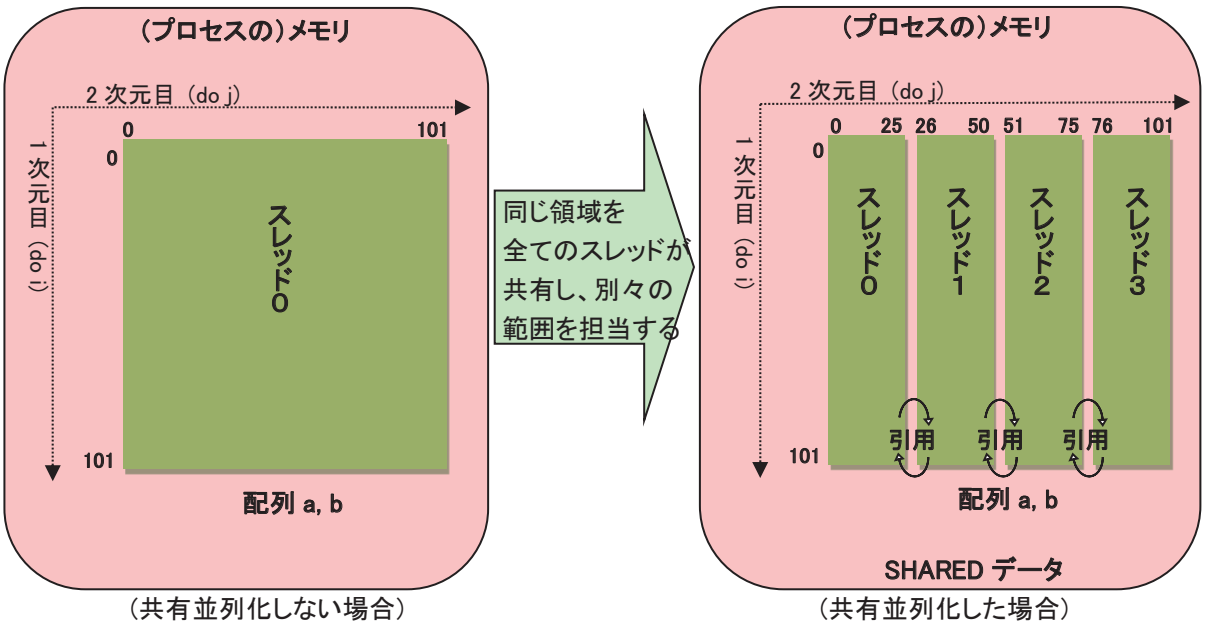


図 2.1　各スレッドが担当する配列 a、b の領域

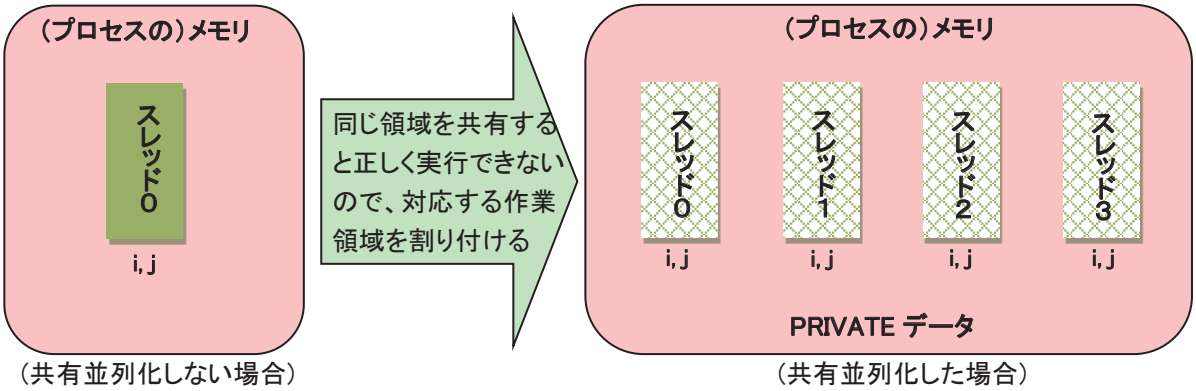


図 2.2　各スレッド専用の DO 変数 i、j 用の作業領域

2.1　自動並列化

並列化を行う場合、並列実行しても結果が不正にならないことを保証するために、通常はデータの依存関係の解析を行い、細心の注意を払ってプログラムの変形や指示文の挿入を行う必要があります。しかし、コンパイラの自動並列化機能を使用すると、コンパイラオプション-mparallel を翻訳時に指定するだけで、それらの作業をコンパイラが自動的に行います。

コンパイラは、まずプログラムを解析して並列実行可能なループや文の集まりを抽出し、次にそれらを複数の仕事に分割してスレッドに割り当て、経過時間を短縮します。また、SHARED データ・PRIVATE データの判定も自動的に行います。コンパイラの自動並列化対象は、図 2.3 のとおりです。

対象となる構文	DO ループ、配列式
対象ループ中に書ける文	代入文、IF 構文、GOTO 文、CONTINUE 文、CALL 文、CASE 構文
対象となる演算	四則演算、べき乗演算、論理演算、関係演算、型変換、組込み関数

図 2.3 コンパイラの自動並列化対象

2.2　コンパイラの最適化

ここでは、共有並列化の効果を高めるためにコンパイラが行う最適化をいくつかご紹介します。

2.2.1　ループ変形

ループ融合・ループ一重化などのループ変形による最適化が可能な場合、最適化後に共有並列化を行い、共有並列実行時の仕事の粒度を最大化します。図 2.4 にループ融合の例を示します。

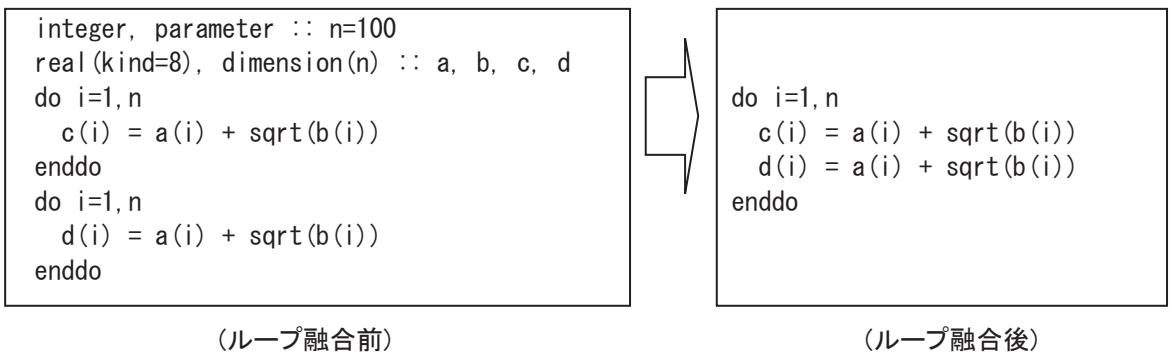


図 2.4　ループ融合の例

2.2.2 条件並列化

並列化できるかどうか又は並列化により高速化できるかどうかが翻訳時に判断できない場合、実行時に依存関係やループ長を調べて、共有並列化するかどうかを選択できるよう条件並列化を行います。図 2.5 では、条件並列化後の IF 構文の論理式 nx*ny > n によって、共有並列化により高速化するのに十分なループ長があるかどうかを判定しています。この際、ループ中の各演算の演算コストに基づいて、共有並列化の効果が期待できる値(n)を自動的に算出し、条件並列化を行います。また、論理式 id-ic==0 .or. abs(id-ic)>=nx によって、ループ中で確定される配列要素 y(ic+i)と y(id+i)の領域に重なりがなく、並列化可能であるかどうかを判定しています。

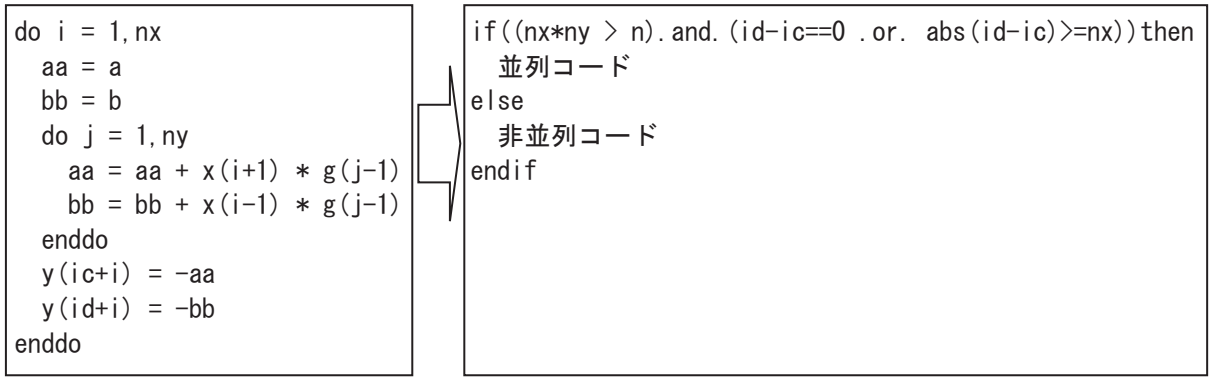


図 2.5　条件並列化の例

2.3 自動並列化促進のためのプログラミング

コンパイラオプション-mparallel の指定だけでは自動並列化できない場合でも、コンパイラ指示行の挿入やプログラムの修正により自動並列化できる場合もあります。ここでは、主な並列化用指示行とプログラムの修正例をご紹介します。

2.3.1 並列化指示行

コンパイラ指示行の書式は、図 2.6 のとおりです。

Fortran の場合：

```
!NEC$ コンパイラ指示オプション
```

C・C++の場合：

```
#pragma _NEC コンパイラ指示オプション
```

図 2.6　コンパイラ指示行の書式

自動並列化のために用意されている主なコンパイラ指示オプションは、次のとおりです。

- concurrent [schedule(種別[, 繰返し回数])] / noconcurrent
  直後のループの自動並列化を許可する / しないを指定します。コンパイラ指示オプション concurrent には、省略可能な schedule 指示句を指定して、ループの繰返しをどのようにスレッドに割り当てるかを指定できます。種別として指定できるのは、static、dynamic、又は runtime です。static

は、ループの繰返しを、スレッド番号の昇順にラウンドロビン方式で割り当てます。dynamic は、ループの繰返しを、その時点で仕事が割り当たっていないスレッドに割り当てます。runtime は、実行時に環境変数 OMP_SCHEDULE で指定された種別に従います。static 及び dynamic には、一度に割り当てるループの繰返し回数を指定できます。コンパイラ指示オプション noconcurrent は、粒度が小さく共有並列化するとかえって性能が低下するループに指定します。

- inner / noinner
  内側ループ又は一重ループを自動並列化の対象とする / しないを指定します。多重ループの内側ループは、既定値では自動並列化の対象にならないので、共有並列化したい場合、コンパイラ指示オプション inner を指定します。一重ループは、共有並列化の効果が翻訳時に不明の場合、自動並列化の対象になりませんが、コンパイラ指示オプション inner を指定することにより自動並列化の対象とすることができます。

- select_concurrent
  多重ループの内、直後のループを優先して自動並列化します。

- nosync
  ループ中で参照される配列に依存がないことを指定します。依存関係が翻訳時には分からないので自動並列化できないときでも、依存がないことを利用者が分かっている場合、コンパイラ指示オプション nosync によりそれをコンパイラに教えてやることによって、自動並列化できる場合があります。

- cncall
  Fortran の組込み関数以外の手続引用を含むループは、既定値では自動並列化の対象になりませんが、並列化しても問題ないことを利用者が分かっている場合、コンパイラ指示オプション cncall を指定すると自動並列化の対象とすることができます。自動並列化されたループ中で引用される手続の仮引数は、結合する実引数が SHARED データの場合、原則として SHARED データとなるので、そのような仮引数を手続中で確定した結果、図 1.3 のような依存関係が発生すると結果が不正となることに注意してください。

2.3.2 並列化のためのソース変形

図 2.7(a)の 2 重ループは、外側の do j のループには依存があるので並列化できませんが、コンパイラ指示オプション inner を内側の do i のループに指定すると、内側ループで自動並列化されます。ただし、内側ループを並列化すると、並列化のオーバヘッドが外側ループの繰返し回数分発生してしまいます。このような場合、図 2.7(b)のように内側ループと外側ループを利用者が交換すると、外側ループで自動並列化され並列化のオーバヘッドを削減できます。

```
integer, parameter :: n=10000,m=10000
real(kind=8), dimension(n,m) :: a
do j=1,m-1
   do i=1,n
      a(i,j) = a(i,j+1) + a(i,j)
   enddo
enddo
```

(a) ループ交換前

```
do i=1,n
 do j=1,m-1
   a(i,j) = a(i,j+1) + a(i,j)
 enddo
enddo
```

(b) ループ交換後

図 2.7　ループ交換の例

図 2.8(a)のループネストは、仮引数 w に関するデータ依存により、外側の do j のループを自動並列化できません。このような場合、図 2.8(b)のように仮配列引数 w を次元拡張し、do j のループの繰返し毎に異なる要素を参照するように修正すれば、自動並列化可能となります。この際、仮引数 w に対応する実引数にも同様の修正が必要となることに注意してください。また、手続 sub の引用元が、仮引数 w に対応する実引数を参照しておらず、配列 w が単なる作業変数として利用されている場合は、図 2.8(c)のように仮引数 w を手続 sub の局所変数に変更すると、ループネスト中ではコンパイラにより PRIVATE データとして割り付けられるので、自動並列化できます。

```
subroutine sub(a,b,w)
parameter(n=100,m=100)
real(kind=8) a(n,m),b(n,m),w(0:n)
do j=1,m
  do i=1,n
    w(i) = a(i,j) / dble(n)
  enddo
  do i=1,n
    b(i,j) = b(i,j) + (w(i-1)+w(i))/2.0
  enddo
enddo
```

(a) 仮引数 w に関する依存

```
subroutine sub(a,b,w)
parameter(n=100,m=100)
real(kind=8) a(n,m),b(n,m),w(0:n,m)
do j=1,m
 do i=1,n
   w(i,j) = a(i,j) / dble(n)
 enddo
 do i=1,n
   b(i,j) = b(i,j)+(w(i-1,j)+w(i,j))/2.0
 enddo
enddo
```

(b) 次元拡張後

```
subroutine sub(a,b)
parameter(n=100,m=100)
real(kind=8) a(n,m),b(n,m),w(0:n)
do j=1,m
  do i=1,n
    w(i) = a(i,j) / dble(n)
  enddo
  do i=1,n
    b(i,j) = b(i,j) + (w(i-1)+w(i))/2.0
  enddo
enddo
```

(c) 局所変数化後

図 2.8　仮引数の修正により自動並列化可能となる例

2.5 OpenMP による共有並列化

コンパイラは、OpenMP による手動並列化も用意しています。OpenMP は、the OpenMP Architecture Review Board (ARB)[1] によって策定された共有並列化のためのオープンな API であり、共有並列化のデファクト標準として広く使用されています。OpenMP では、主として、ソースプログラムに OpenMP 指示文を挿入することによって共有並列化を行います。利用者は、OpenMP 指示文に必要に応じて指示句を追加し、並列化方法を詳細に制御できます。

SX-Aurora TSUBASA で OpenMP を使用する場合、コンパイラオプション-fopenmp を翻訳時に指定してください。

2.5.1 OpenMP 使用時の注意点

自動並列化の場合、コンパイラがプログラムを解析し並列ループ間の不必要な同期を自動的に抑制します。一方 OpenMP による共有並列化時には、nowait 指示句を指定しない限り並列ループの直後に毎回同期が生成されます。同期が必要ない場合には、nowait 指示句を指定すると共有並列化のオーバヘッドを削減できます。

また自動並列化の場合、図 1.4 のような作業変数や集計変数は、コンパイラによって自動的に認識され適切に処理されます。一方 OpenMP による共有並列化時には、DO 変数以外の作業変数及び集計変数をそれぞれ private 指示句及び reduction 指示句中に指定する必要があります。

OpenMP の parallel do 指示文をループに指定して共有並列化した例を図 2.9 に示します。繰返し内作業変数 t を private 指示句中に、集計演算子+ 及び集計変数 s を reduction 指示句中に指定していることに注意してください。なお、DO 変数 i は、この場合既定値で PRIVATE データとなるので、private 指示句を指定する必要はありません。

```
integer, parameter :: n=100
real(kind=8) :: a(n), t, s
!$omp parallel do private(t) reduction(+: s)
do i=1,n-1
  t = a(i) + a(i+1)
  s = s + t
enddo
```

図 2.9　OpenMP による並列化例

2.6 性能解析

2.6.1 プログラム実行性能情報（PROGINF）

逐次(ベクトル)プログラムの場合と同様に、共有並列プログラムの場合も、プログラム実行性能情報(PROGINF)が使用できます。PROGINF は、環境変数 VE_PROGINF の値を YES 又は DETAIL に設定してプログラムを実行することによって採取できます。図 2.10 に、環境変数 VE_PROGINF の値を DETAIL に設定して共有並列実行した場合の出力例を示します。ここで図中の※は、共有並列実行時に固有の情報です。 **Max Active Threads の値により、同時に実行したスレッドの個数の最大値を確認できます。**

```
  Real Time (sec)                :          60.398424
  User Time (sec)                :         482.763682
  Vector Time (sec)              :         478.619978
  Inst. Count                    :      338419440810
  V. Inst. Count                 :      114026565255
  V. Element Count               :    29008594689434
  V. Load Element Count          :    16136470829032
  FLOP Count                     :    17271806827138
  MOPS                           :        71996.740007
  MOPS (Real)                    :       575841.184804
  MFLOPS                         :        35753.806267
  MFLOPS (Real)                  :       285964.533397
  A. V. Length                   :          254.402074
  V. Op. Ratio (%)               :           99.354820
  L1 Cache Miss (sec)            :            0.137676
  CPU Port Conf. (sec)           :            0.000000
  V. Arith. Exec. (sec)          :          179.948458
  V. Load Exec. (sec)            :          298.645140
  VLD LLC Hit Element Ratio (%)  :           44.909006
  FMA Element Count              :      5546911847400
  Power Throttling (sec)         :            0.000000
```

```
Thermal Throttling (sec)              :          0.000000
Max Active Threads                    :                 8  ※
Available CPU Cores                   :                 8  ※
Average CPU Cores Used                :          7.992985  ※
Memory Size Used (MB)                 :      16196.000000
Non Swappable Memory Size Used (MB)   :         98.000000
```

図 2.10　共有並列実行時の PROGINF

2.6.2 性能解析機能(FTRACE 機能)

逐次(ベクトル)プログラムの場合と同様に、共有並列プログラムの場合も、FTRACE 機能によって手続単位又は利用者が指定した区間単位の詳細な性能情報が取得できます。

基本的な使用方法は逐次プログラムの場合と同様ですが、コンパイラは、共有並列化する各ループネストを切り出して、それらを独立した手続に変形します。そのため共有並列化された各ループネストは、FTRACE の表示上は独立した一つの手続として表示されることに注意してください。切りだされた各ループネストは、ソースプログラム中の出現順に、元の手続名に加えて_$1、_$2、・・・とサフィックスが付いた名前の手続として表示されます。図 2.11 に、共有並列実行時の FTRACE の表示例を示します。_$1 のついた手続 jacobi_$1 は、手続 JACOBI の最初に出現する共有並列化されたループネストの性能情報です。-thread0、-thread1、・・・は、各スレッドの性能情報です。JACOBI のような$のついていない手続名は、手続 JACOBI 中の共有並列化されなかった部分の性能情報です。

```
FREQUENCY  EXCLUSIVE        AVER.TIME     MOPS    MFLOPS  V.OP  AVER.    VECTOR L1CACHE CPU PORT VLD LLC PROC.NAME
           TIME[sec](  % )    [msec]                       RATIO V.LEN     TIME    MISS     CONF HIT E.%

       8   239.724( 99.9) 29965.489  78821.8  39157.8 99.36 254.4  238.102  0.023    0.000   45.94 jacobi_$1
       2    59.932( 25.0) 29965.804  79126.6  39310.9 99.36 254.4   59.911  0.004    0.000   45.96  -thread0
       2    59.931( 25.0) 29965.335  79128.1  39311.5 99.36 254.4   59.876  0.004    0.000   45.95  -thread1
       2    59.931( 25.0) 29965.344  79130.3  39311.5 99.36 254.4   59.628  0.006    0.000   45.92  -thread2
       2    59.931( 25.0) 29965.471  77902.1  38697.1 99.35 254.4   58.687  0.009    0.000   45.91  -thread3
                :
       2     0.000(  0.0)     0.009    281.8      0.0  0.00   0.0    0.000  0.000    0.000    0.00 JACOBI
```

図 2.11　共有並列実行時の FTRACE

主に浮動小数点演算を実行する共有並列プログラムの性能を分析する場合、FTRACE 中の MFLOPS の値に着目してください。図 2.11 の場合、並列ループネスト jacobi_$1 は、4 個のスレッドそれぞれが約 39000MFLOPS の性能で実行したことが分かります。MFLOPS 値がスレッド間でほぼ均等となっているので、うまく共有並列化されていると判断できます。逆に図 2.12 のように、スレッド 0 及びスレッド 1 だけ MFLOPS 値が大きく、残りのスレッドが小さい場合は、負荷バランスが悪い、と判断できます。このようなプログラムは、コンパイラ指示オプション concurrent の schedule 指示句などを指定して負荷バランスを均等化すると、経過時間を短縮できる可能性があります。

FREQUENCY	EXCLUSIVE TIME[sec](%)	AVER. TIME [msec]	MOPS	MFLOPS	V. OP RATIO	AVER. V. LEN	VECTOR TIME	L1CACHE MISS	CPU PORT CONF	VLD LLC HIT E. %	PROC. NAME
8	240. 408(99. 9)	30050. 994	40443. 9	19958. 7	98. 70	254. 4	119. 740	0. 008	0. 000	46. 43	jacobi_$1
2	60. 103(25. 0)	30051. 341	80661. 4	40073. 5	99. 36	254. 4	60. 098	0. 002	0. 000	46. 43	-thread0
2	60. 102(25. 0)	30050. 955	80036. 4	39760. 9	99. 36	254. 4	59. 642	0. 003	0. 000	46. 43	-thread1
2	60. 102(25. 0)	30050. 833	538. 4	0. 0	0. 00	1. 0	0. 000	0. 002	0. 000	0. 00	-thread2
2	60. 102(25. 0)	30050. 848	538. 4	0. 0	0. 00	1. 0	0. 000	0. 002	0. 000	0. 00	-thread3

図 2.12　負荷バランスの悪い共有並列実行時の FTRACE

3. 分散並列化

分散並列化とは、それぞれ独自のメモリ空間をもつ複数のプロセスによる並列処理のことです。SX-Aurora TSUBASA では、一つ又は複数の VE カードの CPU コア上で実行されるプロセスに仕事を割り当て、分散並列化を行うことができます。

例として、図 1.1 の 2 重ループの分散並列化を考えます。第 1 章と同様に、外側の do j のループを四つの仕事に分割して別々のプロセスに割り当てます。この場合、データ領域に関しては、図 3.1 のように、2 次元配列 a、b を do j のループに対応する 2 次元目で分割し、分割された各部分領域の処理を各プロセスが担当することになります(分割配置・1 次元分割)。ここで、例えば j=25 の繰返しを担当するプロセス 0 は、プロセス 1 の担当領域である配列要素 a(i,26), (i=1,2,…,lx)の値を引用する、といったように、配列の分割境界部分の処理では、自プロセスの担当領域外の配列要素を参照する必要があります。ところが各プロセスは、他のプロセスに割り付けられた配列の領域を直接参照できないので、通信を行って必要な配列要素の値を取得しなければなりません。このように分散並列化の場合、共有並列化と異なり、他のプロセスの担当範囲のデータを参照するために、必要に応じて通信が必要です。一方 DO 変数 i、j については、図 3.2 のように各プロセスがそれぞれ専用の領域を割り付ける(重複配置)と、並列実行中、各プロセスは変数 i、j をそれぞれ独立に参照できるので通信の必要はありません。すなわち、分散並列実行時にどのような通信が必要となるかは、各データ要素をどのようにプロセスに割り付けるか(データマッピング)及び一つの仕事をどのように複数の小さな仕事へと分割してプロセスに割り当てるか(計算マッピング)に依存します。このように分散並列化においては、データマッピング、計算マッピング、及び通信の 3 点を全て考慮してプログラムする必要があります。

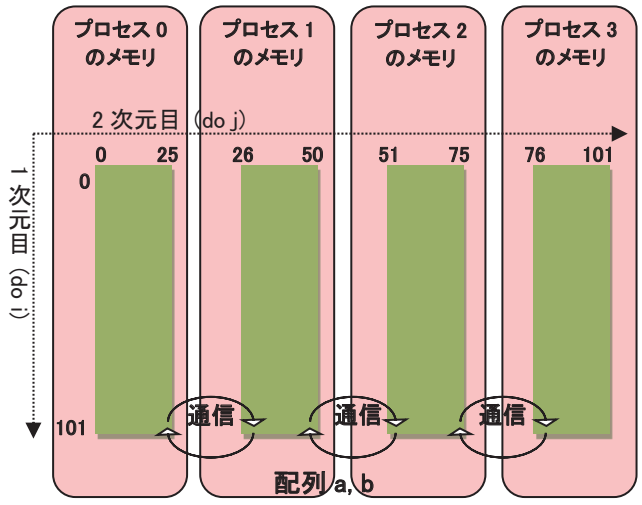


図 3.1　配列 a、b の分割配置(1 次元分割)

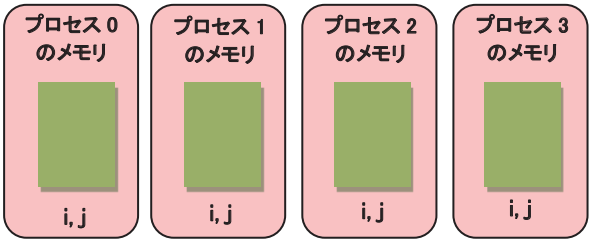


図 3.2　DO 変数 i、j の重複配置

3.1 データマッピングと性能

再び図 1.1 の 2 重ループの分散並列化を考えます。今度は図 3.3 のように、外側・内側のループをそれぞれ二つの仕事に分割してみます。この場合、データ領域に関しては、図 3.4 のように 2 次元配列 a、b を1次元目・2 次元目共に分割し、分割された各部分領域の処理を各プロセスが担当することになります（分割配置・2 次元分割）。ここでも、例えば j=50 の繰返しを担当するプロセス 0 は、プロセス 2 の担当領域である配列要素 a(i,51), (i=1,2,…,50)の値を引用する、といったように、配列の分割境界部分の処理では、自プロセスの担当領域外の配列要素を参照するために通信が必要です。

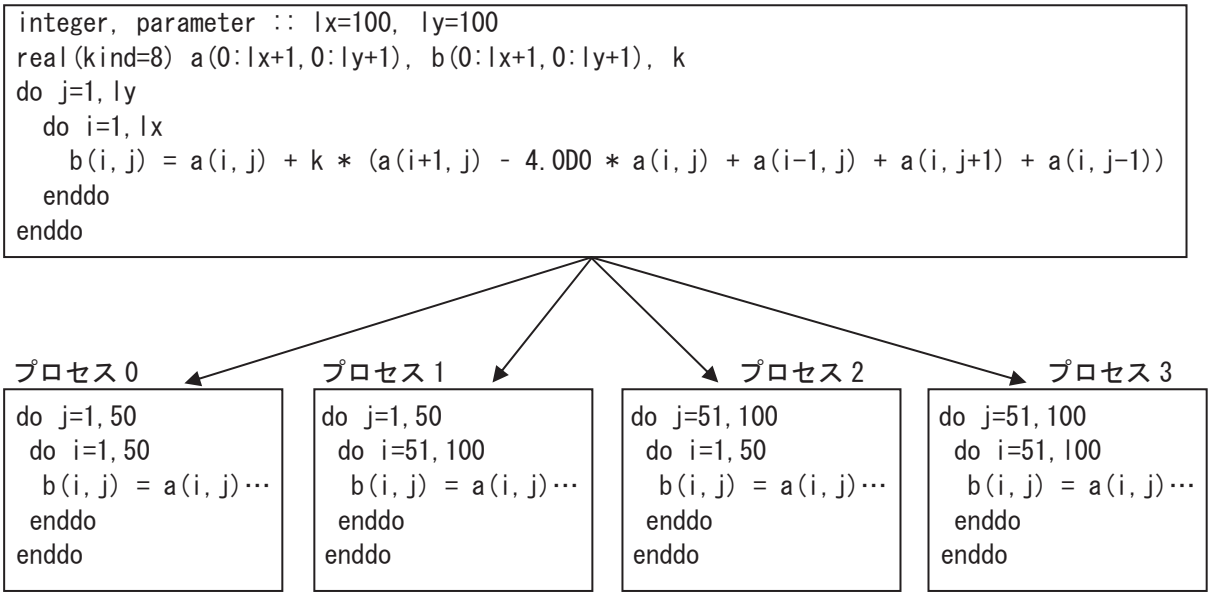


図 3.3　二つのループを共に並列化する例

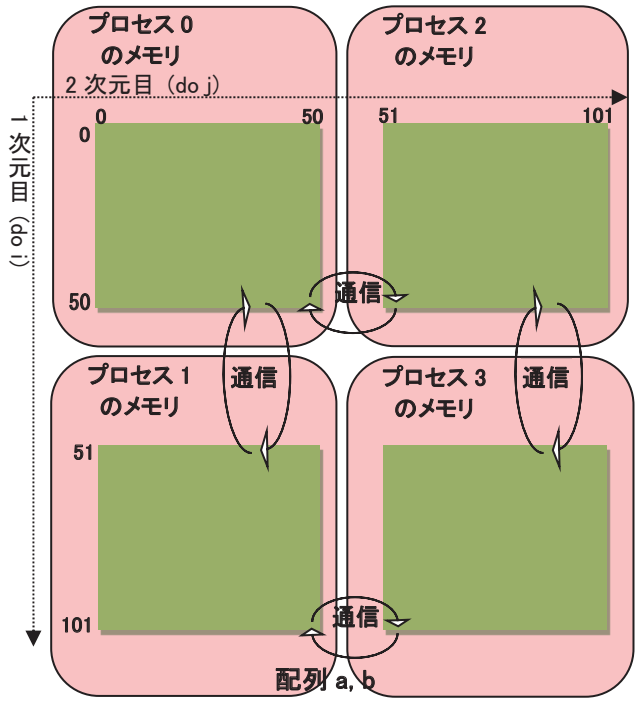


図 3.4　配列 a、b の分割配置(2 次元分割)

これを前節の外側ループだけによる分散並列化(1 次元分割)と比較してみます。各プロセスの通信に関しては、1 次元分割の場合、送受信回数はそれぞれ 2 回(2 次元目の上下方向)、送受信量はそれぞれ 1 次元目の寸法である lx*2 となります。一方 2 次元分割の場合、送受信回数はそれぞれ 4 回(1・2 次元目それぞれの上下方向)、送受信量はそれぞれ (lx/1 次元目分割数+ly/2 次元目分割数)*2 となります。従って、1 次元分割と比べると、2 次元分割の方が通信回数は多いですが、通信量はプロセスの個数が増えるほど少なくなります。1 回の通信時間は、典型的にはセットアップ時間+通信量/通信バンド幅と表せます。そのため、通信量が少なく(配列の寸法が小さく)セットアップ時間の影響が大きい場合には 1 次元分割の方が通信コストは少なく、逆に通信量が多く(配列の寸法が大きく)通信量の影響が大きい場合には 2 次元分割の方が通信コストは少なくなると考えられます。一方、各プロセスの演算性能を比較すると、ベクトル長は内側の do i のループに対応する配列の 1 次元目の寸法に依存します。そのため、1 次元目をプロセス間に分割することによってベクトル長が短くなりすぎると、ベクトル演算の効率が低下します。従って、配列の寸法がそれほど大きくない場合、2 次元目だけを分割したほうが演算時間は短い可能性があります。このように、これら二つの分散並列化方法のどちらが経過時間を短縮できるかは、プロセスの個数や配列の寸法に依存して変わります。分散並列化時には、通信コストや並列化後の演算性能を考慮して、データマッピングを決定することが重要です。

3.2 Message Passing Interface (MPI)

3.2.1 MPI 概要

MPI は、Message Passing Interface Forum[2] によって策定された通信ライブラリのオープンな規格であり、分散並列プログラミングのデファクト標準として広く使用されています。MPI を使用した分散並列化では、図 3.1 や図 3.4 中の通信部分を、MPI 手続の引用によってプログラムすることになります。

分散並列化時に高いスピードアップを達成するには、通信コストの削減が特に重要です。プロセス内のメモリアクセスコストと比較して、プロセス間の通信コストは圧倒的に大きいからです。

SX-Aurora TSUBASA は、高速な通信環境を提供するため、MPI ライブラリ NEC MPI を用意しています。NEC MPI は、プロセスの VE カードへの配置方法や通信サイズに応じて、VE カード上の主記憶装置

や InfiniBand のハードウェア機能を利用した最適な通信プロトコルを選択し、SX-Aurora TSUBASA の性能を最大限引き出しています。

3.2.2 MPI プログラミングの基本

MPI プログラミングでは、まず MPI 規格で規定されている各種定数などが定義されたヘッダファイル mpif.h（Fortran の場合）又は mpi.h（C・C++の場合）をインクルードします（Fortran の場合、ヘッダファイルをインクルードする代わりに、システムモジュール mpi 又は mpi_f08 を引用することもできます）。次に、手続 mpi_init 又は mpi_init_thread を引用して初期化を行い、手続 mpi_finalize を引用して MPI の使用を終了します。利用者は、この二つの引用の間で、1 対 1 通信、集団通信、又は片側通信により通信を行うことができます。

ほとんどの MPI 手続では、コミュニケータと呼ばれる引数で、通信を行うプロセスの集合を指定します。プログラム開始時の全てのプロセスに対応するコミュニケータは、MPI の定数 MPI_COMM_WORLD です。各プロセスは、ランクと呼ばれるコミュニケータ中で一意な整数値をもちます。ランクの値は、0 以上、プロセス数-1 以下です。手続 mpi_comm_size 及び mpi_comm_rank により、それぞれプロセスの個数及び自プロセスのランクを取得でき、これらを利用して各プロセスが行う処理をプログラムできます。

MPI プログラムの典型的な例として、図 1.1 の 2 重ループを外側ループで四つの仕事に分割し、分散並列化したプログラムを図 3.5 に示します。MPI プログラミングでは、プログラム全体のデータ・処理ではなく、各プロセス用に分割した後のデータ・処理をプログラムします。そのため、定数 ly の値を 25 に変更して（文番号 100）、配列の宣言（文番号 200）とループ長（文番号 300）を分割後の範囲に縮小しています。この際、通信時に受信したデータを配置する領域が必要なので、図 3.6 のように各プロセス上の配列 a の領域を大きめに割り付けていることに注意してください。このような隣接プロセスとの通信のための領域は袖領域と呼ばれます。並列ループ（文番号 300）実行前に、分割境界の処理に必要な配列 a の要素を、1 対 1 通信手続 mpi_sendrecv により隣接プロセス間で送受信（文番号 400）し、並列ループ実行中は、隣接プロセス上のデータの代わりに自プロセス上の袖領域を参照しています。両端のプロセスには片側の隣接プロセスが存在しないので、通信相手を設定する IF 構文（文番号 500）において、通信相手を示す変数 men、mep に、該当するプロセスが存在しないことを示す MPI の定数 MPI_PROC_NULL を設定しています。送受信先として定数 MPI_PROC_NULL を指定すると、その通信は実行されません。

```
      include "mpif.h"
100   integer, parameter :: lx=100, ly=25   ! ly は分割後の寸法
200   real(kind=8) a(0:lx+1,0:ly+1),b(0:lx+1,1:ly), k
   ! 初期化
      call mpi_init(ierr)
   ! プロセスの個数を変数 nproc に取得する
      call mpi_comm_size(MPI_COMM_WORLD,nproc,ierr)
   ! 自プロセスのランクを変数 me に取得する。
      call mpi_comm_rank(MPI_COMM_WORLD,me,ierr)
                :
   ! 通信相手の設定：両端のプロセスは特別処理を行う。
500   if(me .eq. 0)then
        men = MPI_PROC_NULL  ! 下端のプロセスの下隣は存在しない
      else
        men = me - 1  ! 下隣のプロセスのランク
      endif
      if(me .eq. nproc-1)then
        mep = MPI_PROC_NULL  ! 上端のプロセスの上隣は存在しない
      else
        mep = me + 1  ! 上隣のプロセスのランク
      endif
   ! 隣接プロセス間の通信：上方向 → 下方向
400   call mpi_sendrecv(a(1,ly),lx,MPI_DOUBLE_PRECISION,mep,0,a(1,0),       &  ! 上方向
     &     lx,MPI_DOUBLE_PRECISION,men,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE,ierr)
```

```
      call mpi_sendrecv(a(1,1),lx,MPI_DOUBLE_PRECISION,men,0,a(1,ly+1),      &  ! 下方向
     &     lx,MPI_DOUBLE_PRECISION,mep,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE,ierr)
  ! 並列化後のループ
300   do j=1,ly
        do i=1,lx
          b(i,j) = a(i,j) + k * (a(i+1,j)-4.0D0*a(i,j)+a(i-1,j)+a(i,j+1)+a(i,j-1))
        enddo
      enddo
          :
  ! 後始末処理
      call mpi_finalize(ierr)
      end
```

図 3.5　MPI プログラム例

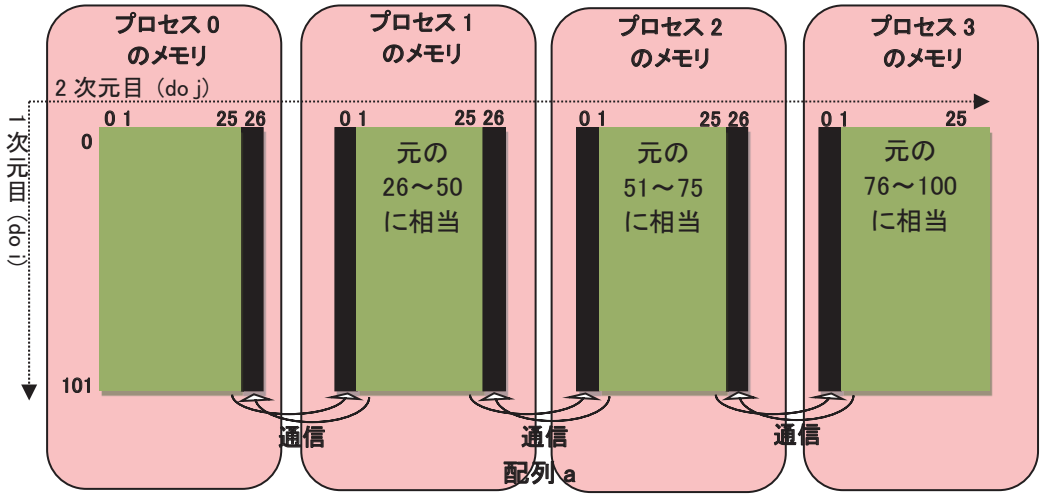


図 3.6　1 次元分割時の配列 a の割付け（黒色は袖領域）

3.3 NEC MPI の活用方法と注意点

3.3.1 MPI プログラムへの標準入力

MPI プログラムに標準入力から入力データを与える場合、図 3.7 のように、実行ファイルに対して直接入力データを与えると、正常に実行できない場合があります。

```
mpirun -np 4 /execdir/a.out < /datadir/inputfile
```

図 3.7　MPI プログラムへの不正な標準入力の例

このような場合、図 3.8 のように、コンパイラの環境変数 VE_FORT5 に入力ファイル名を設定して事前接続した上で、MPI プログラムを実行してください。

```
setenv VE_FORT5 /datadir/inputfile

mpirun -np 4 /execdir/a.out
```

図 3.8　MPI プログラムへの標準入力

3.3.2 プロセス毎に別々のファイルを対象にした入出力

環境変数 MPIRANK は、コミュニケータとして引数 MPI_COMM_WORLD を指定して、手続 mpi_comm_rank を呼び出すことにより得られるランクと同一の値をもっています。これを利用して、図 3.9 のようなシェルスクリプトを作成し（シェルスクリプトのファイル名を、例えば/execdir/run.sh とします）、図 3.10 のように、mpirun コマンドでそのシェルスクリプトを実行すると、各プロセスのランクと入出力ファイル名とを一対一に対応させ、プロセス毎に別々のファイルを対象にした入出力が可能です。

```
#!/bin/csh
setenv VE_FORT5 infile.$MPIRANK
setenv VE_FORT6 outfile.$MPIRANK

exec /execdir/a.out
```

図 3.9　環境変数 MPIRANK を利用した入出力のためのシェルスクリプト例

```
mpirun -np 4 /execdir/run.sh
```

図 3.10　シェルスクリプトを介した MPI プログラムの実行例

プログラム中で接続したファイルに対して入出力する場合は、手続 mpi_comm_rank により取得したランクを利用してファイル名を決めると、やはりプロセス毎に別々のファイルを対象にした入出力が可能です

3.3.3 プロセス毎に別々のファイルへの標準出力・標準エラー出力

複数の MPI プロセスが同時に標準出力又は標準エラー出力を行った場合、それらは入り交じって表示される場合があります。各プロセスの標準出力及び標準エラー出力を別々のファイルに容易に出力できるように、NEC MPI はシェルスクリプト/opt/nec/ve/bin/mpisep.sh を用意しています。このスクリプトを使用して、図 3.11 のように MPI プログラムを実行すると、環境変数 NMPI_SEPSELECT の実行時の値に応じて、次のように各プロセスの標準出力及び標準エラー出力が別々のファイルに出力されます。なお下記の u は通常 0 となり、r は環境変数 MPIRANK の値となります。

- 環境変数 NMPI_SEPSELECT の値が 1 の場合
  各プロセスの標準出力が、ファイル stdout.u:r に出力されます。

- 環境変数 NMPI_SEPSELECT の値が 2 の場合（既定値）
  各プロセスの標準エラー出力が、ファイル stderr.u:r に出力されます。

- 環境変数 NMPI_SEPSELECT の値が 3 の場合
  各プロセスの標準出力及び標準エラー出力が、それぞれファイル stdout.u:r 及び stderr.u:r に出力されます。

- 環境変数 NMPI_SEPSELECT の値が 4 の場合
  各プロセスの標準出力及び標準エラー出力が、共にファイル std.u:r に出力されます。

```
mpirun -np 4 /opt/nec/ve/bin/mpisep.sh  /execdir/a.out
```

図 3.11　シェルスクリプト /opt/nec/ve/bin/mpisep.sh を使用した実行例

3.3.5 MPI 集団手続デバッグ支援機能

NEC MPI は、実行時でなければ見つけにくい MPI プログラム特有のバグを検出する機能を用意しています。コンパイラオプション-mpiverify を指定して翻訳し、環境変数 NMPI_VERIFY の値を 3 又は 4 に設定して実行すると、集団手続に対する呼出し順序の誤りや、引数のプロセス間の不整合などの情報が出力されます。

3.3.6 Fortran からの MPI 使用時の注意点

Fortran プログラム中で引用する非ブロッキングな MPI 手続の実引数として、部分配列、配列式、配列ポインタ、又は形状引継ぎ配列を指定すると、引数がメモリ上連続であることを保証するために、コンパイラが一時配列を割り付けて実引数をコピーする場合があります。このとき、実際の引数としては、その一時配列が MPI 手続に渡されます。しかし、その一時配列は、非ブロッキングな MPI 手続の完了以前に解放されてしまうので、プログラムが正常に実行されない可能性があります。非ブロッキングな MPI 手続の実引数には、メモリ上連続な配列を指定してください。形状引継ぎ配列については、コンパイラオプション-fassume-contiguous を指定すると、コンパイラによる一時配列の生成を抑制できます。

3.4 ハイブリッド並列化

複数のプロセスだけによるプログラムの分散並列化をフラット並列化と呼びます。SX-Aurora TSUBASA では、分散並列化と第 2 章でご説明した共有並列化を併用することもできます。これをハイブリッド並列化と呼びます。ハイブリッド並列化は、MPI プログラムを自動並列化又は OpenMP により共有並列化するだけで使用可能です。

図 3.5 では、図 1.1 の 2 重ループを外側ループで四つの仕事に分割し、4 プロセスに割り当てフラット並列化しました。これを、それぞれ 2 スレッドからなる二つのプロセスに割り当てハイブリッド並列化するには、図 3.5 中の定数 ly の値を 50 に変更した上で、自動並列化のためのコンパイラオプション-mparallel を指定して翻訳し、共有並列化します。さらに、環境変数 OMP_NUM_THREADS 又は VE_OMP_NUM_THREADS の値を実行時に 2 に設定すると、各プロセスは 2 スレッドで実行されます。フラット並列化の場合、配列 a を各プロセス上に図 3.6 のように割り付けました。一方、ハイブリッド並列化の場合、配列 a を各プロセス上に図 3.12 のように割り付けることになります。一つのプロセス中の複数のスレッドは、そのプロセスの全てのデータ領域を直接参照できるので、スレッド間で通信を行う必要はなく、従ってスレッド間には袖領域も必要ないことに注意してください。

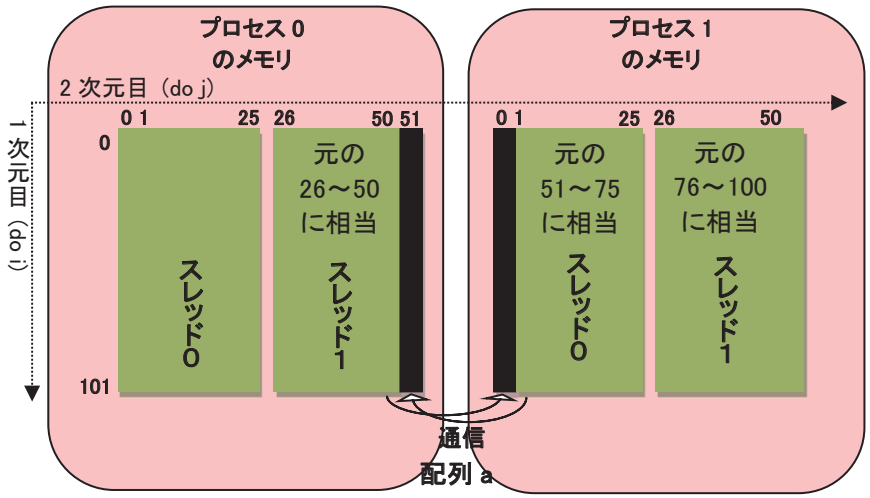


図 3.12　ハイブリッド並列化時の配列 a の割付け（黒色は袖領域）

一つの仕事を複数の小さな仕事に分割して、複数のプロセスに割り当てる場合（フラット並列化）と、各プロセス内の複数のスレッドに割り当てる場合（ハイブリッド並列化）とを比較すると、ハイブリッド並列化に

よりプロセスの総数が削減されます。そのため、プロセス数に依存してコストが増大する通信を含むようなプログラムは、ハイブリッド並列化により経過時間を短縮できる場合があります。その反面、分散並列化と共有並列化の 2 重のオーバヘッドが発生するので、プログラムによってはハイブリッド並列化により経過時間が増加する場合もあります。一方、コンパイラ・MPI の処理系内部のバッファや、全てのプロセスに重複配置するデータの個数は、プロセス数の減少とともに減少するので、ハイブリッド並列化の方がフラット並列化よりも使用メモリの総量を少なくできる傾向があります。

3.4.1 ハイブリッド並列化時の注意点

共有並列化の場合、既定値では各プロセスが VE カード内の全ての CPU コアを使用して共有並列実行を行います。そのため、ハイブリッド並列化時に一つの VE カード上で二つ以上のプロセスを実行すると、CPU コアの個数がスレッドの個数より少なくなり、大幅に性能が低下する可能性があります。ハイブリッド並列化時には、各 VE カード上のプロセスの個数を 1 にするか、共有並列化のスレッドの個数を指定する環境変数 OMP_NUM_THREADS 又は VE_OMP_NUM_THREADS を指定して、各 VE カード上のプロセスの個数とスレッドの個数の積が 8 以下となるようにしてください。

NEC MPI のスレッドサポートレベルは、MPI_THREAD_SERIALIZED です。従って、全てのスレッドが MPI 手続を呼び出せますが、OpenMP で共有並列化する場合、同一プロセス中の複数のスレッドが同時に MPI 手続を呼び出さないように利用者がプログラムする必要があります。

3.5 性能解析

3.5.1 MPI プログラム実行性能情報

プログラム実行時に環境変数 NMPI_PROGINF の値を、YES、ALL、DETAIL、又は ALL_DETAIL に設定することによって MPI プログラムの実行性能情報を採取できます。図 3.13 に、環境変数 NMPI_PROGINF の値を DETAIL に設定して実行した場合の出力例を示します。User Time が最小のプロセス(Min)、最大のプロセス(Max)、及び平均(Average)の情報が表示されており、プログラム全体の性能を簡便に把握できます。

```
MPI Program Information:
========================
Note: It is measured from MPI_Init till MPI_Finalize.
      [U,R] specifies the Universe and the Process Rank in the Universe.
      Times are given in seconds.


Global Data of 4 Vector processes   :          Min [U,R]          Max [U,R]      Average
=================================

Real Time (sec)                     :       60.203 [0,1]       60.203 [0,3]       60.203
User Time (sec)                     :       60.186 [0,0]       60.194 [0,1]       60.192
Vector Time (sec)                   :       60.060 [0,3]       60.154 [0,1]       60.123
Inst. Count                         :  45399907387 [0,0]  45758376971 [0,2]  45613254933
V. Inst. Count                      :  15364471112 [0,0]  15489373831 [0,2]  15427821557
V. Element Count                    : 3906811516058 [0,0] 3939689666616 [0,2] 3923254187337
V. Load Element Count               : 2172285365098 [0,0] 2190447895724 [0,2] 2181368777928
FLOP Count                          : 2323955699488 [0,3] 2342254563239 [0,1] 2333105131564
MOPS                                :    77806.308 [0,3]    78451.352 [0,2]    78130.852
MOPS (Real)                         :    77790.396 [0,0]    78437.995 [0,2]    78114.788
MFLOPS                              :    38608.708 [0,3]    38912.728 [0,2]    38762.091
MFLOPS (Real)                       :    38602.070 [0,3]    38906.154 [0,1]    38754.121
A. V. Length                        :      254.165 [0,3]      254.400 [0,1]      254.297
V. Op. Ratio (%)                    :       99.356 [0,3]       99.359 [0,1]       99.358
```

```
L1 Cache Miss (sec)                :         0.025 [0,1]         0.040 [0,3]         0.031
CPU Port Conf. (sec)               :         0.000 [0,0]         0.000 [0,0]         0.000
V. Arith. Exec. (sec)              :        23.589 [0,3]        23.748 [0,1]        23.665
V. Load Exec. (sec)                :        36.214 [0,3]        36.400 [0,1]        36.338
VLD LLC Hit Element Ratio (%)      :        45.808 [0,3]        45.869 [0,1]        45.832
FMA Element Count                  : 746347948500 [0,0] 752224704000 [0,1] 749286326250
Power Throttling (sec)             :         0.000 [0,0]         0.000 [0,0]         0.000
Thermal Throttling (sec)           :         0.000 [0,0]         0.000 [0,0]         0.000
Memory Size Used (MB)              :      4241.166 [0,0]      4242.500 [0,1]      4241.500


Overall Data of 4 Vector processes
==================================

Real Time (sec)                    :        60.203
User Time (sec)                    :       240.767
Vector Time (sec)                  :       240.494
GOPS                               :        78.131
GOPS (Real)                        :       312.459
GFLOPS                             :        38.762
GFLOPS (Real)                      :       155.016
Memory Size Used (GB)              :        16.568
```

図 3.13　MPI プログラムの実行時性能情報

3.5.2 性能解析機能(FTRACE 機能・NEC Ftrace Viewer)

コンパイラオプション-ftrace を指定して翻訳した MPI プログラムを実行すると、解析情報ファイル ftrace.out.u.r (u は通常 0、r は環境変数 MPIRANK の値)が各プロセスから出力されます。MPI プログラムの場合、コンパイラによる性能解析情報に加えて、通信時間などの情報が得られます。

SX-Aurora TSUBASA が用意する NEC Ftrace Viewer は、性能解析情報をグラフィカルに表示でき、並列プログラムの性能を分析する際にも有効です。例として、8 プロセスにより並列実行したある MPI プログラムの負荷バランスを解析してみます。まず、解析情報ファイルを読み込んだときに表示される Process Breakdown Chart を見ると、図 3.14 のように手続 ITERATE の実行時間(EXCLUSIVE TIME)が大きいことが分かります。手続 ITERATE の負荷バランスを把握するため、左上の「Chart」メニューから Function Statistics Chart を選択します。すると、図 3.15 中の折れ線グラフが示すように、手続 ITERATE の EXCLUSIVE TIME の標準偏差が大きく、負荷バランスが悪いことが分かります。状況をより詳細に調査するため、「Chart」メニューから Process Metrics Chart を選択します。次に、「Metrics Selection」メニューから、MPI COMM. TIME(MPI 通信時間)及び MPI IDLE TIME(MPI 待ち時間)を選択します。すると、図 3.16 のように、MPI 通信時間及び MPI 待ち時間を示す二つの折れ線グラフがほぼ重なって見えます。これは、MPI 通信時間のほとんどが MPI 待ち時間であることを意味しています。また、棒グラフで示される手続 ITERATE の EXCLUSIVE TIME と、折れ線グラフの MPI 待ち時間とを比較すると、EXCLUSIVE TIME に占める MPI 待ち時間が、両端のプロセスだけ短いことが分かります。従って、両端のプロセスの MPI 待ち時間以外の実行時間、つまり演算時間が他のプロセスよりも大きく、相対的に演算時間の小さな他のプロセスにおいて、MPI による通信時に待ちが発生していると推定できます。以上のことから、両端のプロセスの演算を他のプロセスに分配すれば、負荷バランスを改善できる可能性があることが分かります。このように、NEC Ftrace Viewer のグラフ機能を使用すると、負荷バランスなど複数の仕事間の関係を容易に把握できます。

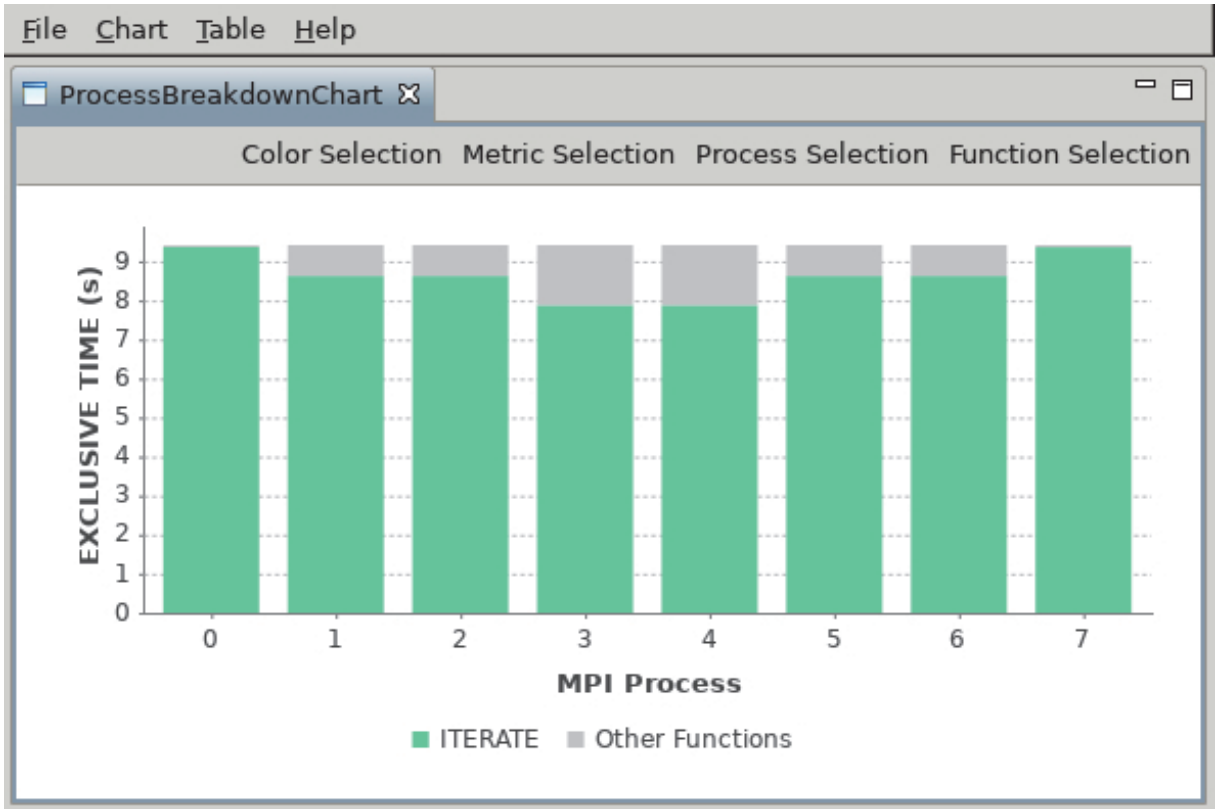


図 3.14　Process Breakdown Chart

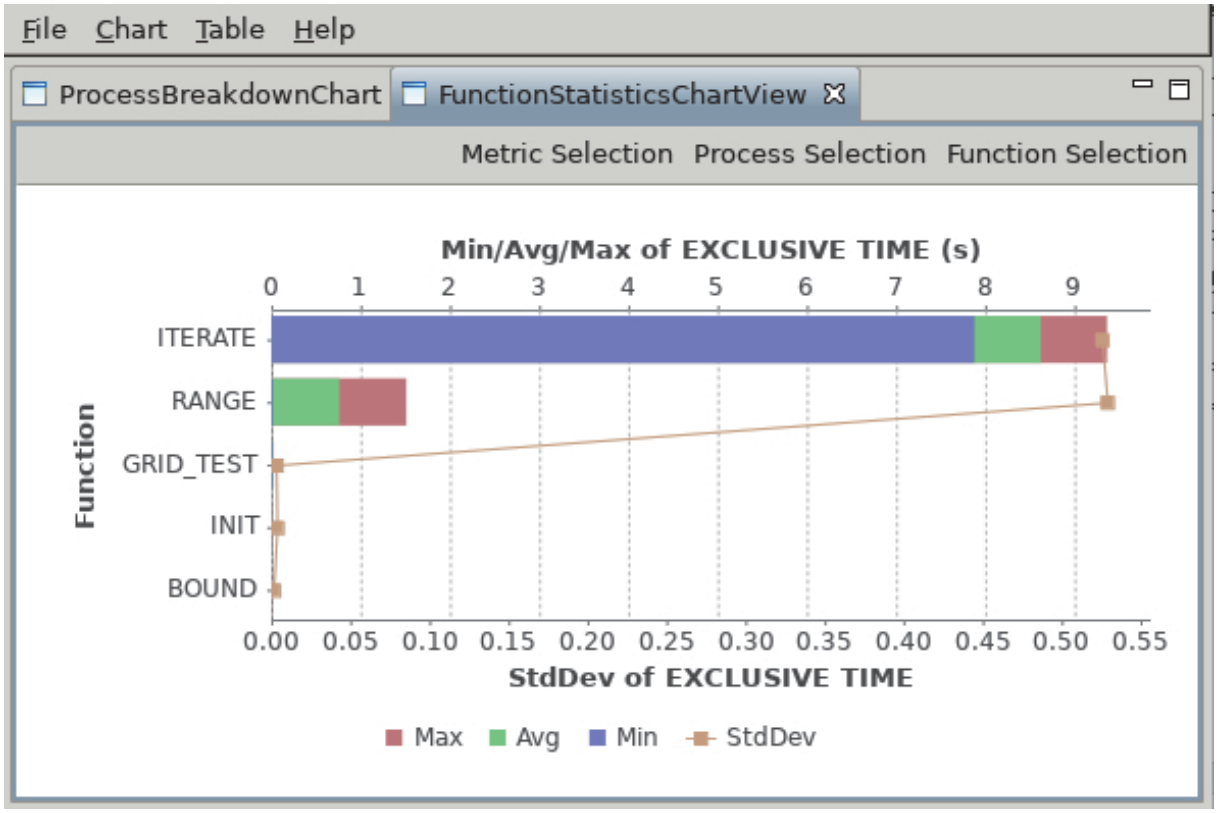


図 3.15　Function Statistics Chart

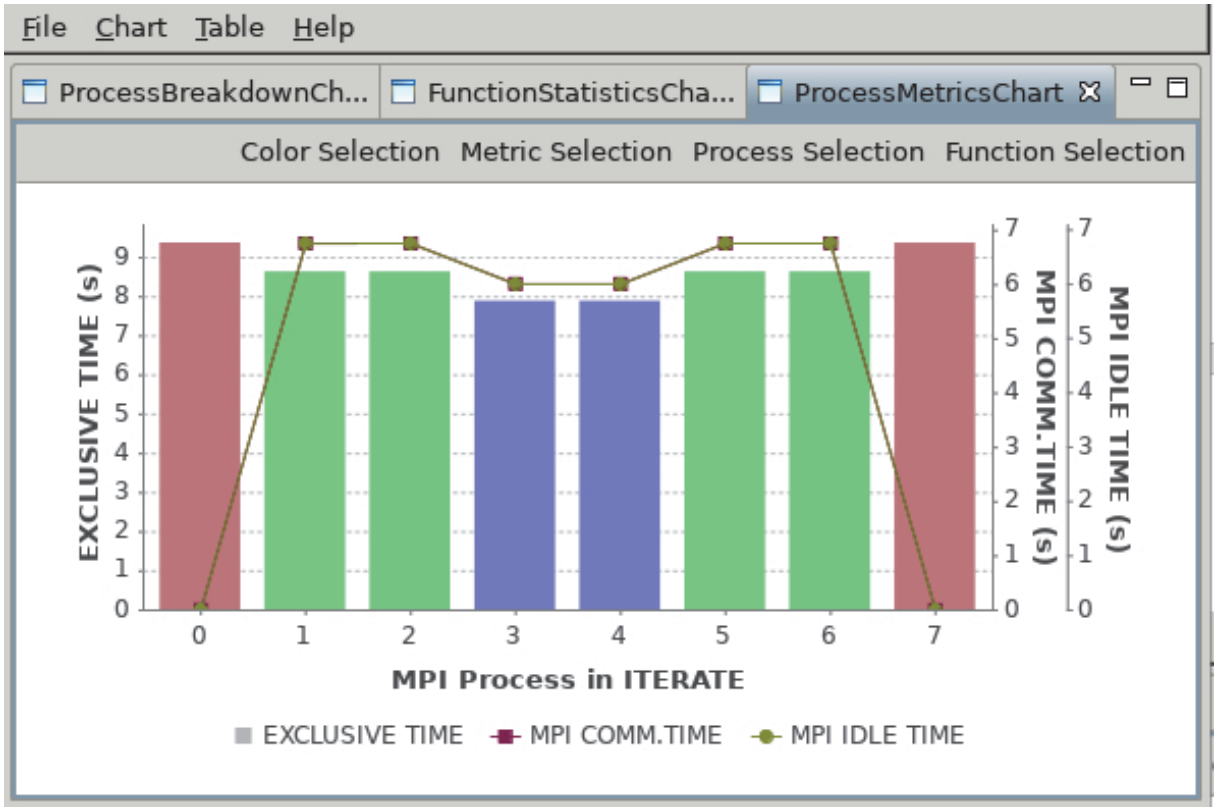


図 3.16　Process Metrics Chart

4. あとがき

本稿では、コンパイラの共有並列化機能 及び NEC MPI を使用した分散並列化の基本的事項を中心にご紹介しました。SX-Aurora TSUBASA が用意する並列処理環境の詳細な使用方法については、「Fortran コンパイラ ユーザーズガイド」[3]、「C/C++コンパイラ ユーザーズガイド」[4]、及び「NEC MPI ユーザーズガイド」[5] をご覧ください。また、プログラム実行性能情報(PROGINF)及び性能解析機能(FTRACE)の詳細な使用方法については、「PROGINF/FTRACE ユーザーズガイド」[6]及び「NEC Ftrace Viewer ユーザーズガイド」[7]をご覧ください。SX-Aurora TSUBASA をご使用になる上で、本稿がお役に立てば幸いです。

参考文献

[1] The OpenMP Architecture Review Board (http://openmp.org/)
[2] Message Passing Interface Forum (http://www.mpi-forum.org/)
[3] Fortran コンパイラ ユーザーズガイド、日本電気株式会社 (https://www.hpc.nec/documentation)
[4] C/C++コンパイラ ユーザーズガイド、日本電気株式会社 (https://www.hpc.nec/documentation)
[5] NEC MPI ユーザーズガイド、日本電気株式会社 (https://www.hpc.nec/documentation)
[6] PROGINF/FTRACE ユーザーズガイド、日本電気株式会社 (https://www.hpc.nec/documentation)
[7] NEC Ftrace Viewer ユーザーズガイド、日本電気株式会社 (https://www.hpc.nec/documentation)

［共同研究成果］

ナノアンテナの数値解析のための ブロックモーメント法の高速化

今野 佳祐 †, 陳 強 †
東北大学大学院工学研究科 通信工学専攻 †

1　まえがき

近年，ナノスケールの波長を有する帯域におけるアンテナ，すなわちナノアンテナの研究が盛んに行われている．ナノアンテナは，光センサの高感度化や光電変換の高効率化など，ナノデバイスの高機能化に役立つと期待されている．また，回折限界を超えたスポット径を実現できる局在型の表面プラズモンや，ナノアンテナ表面を伝播して放射にも寄与することが知られている表面プラズモンポラリトンなど，ナノアンテナ独特の現象も報告されており，基礎科学の側面でも注目されている．ナノアンテナは，主に金や銀，アルミニウム等の金属粒子からなるが，これらの粒子は特に可視光付近の帯域で誘電体として振舞うことが知られている．したがって，ナノアンテナの設計においては，一般的なマイクロ波帯のように金属は完全導体とみなすことができず，誘電体として扱う必要がある．

線状のナノアンテナにおいて，誘電体として振る舞う金属の比誘電率や波数が満たすべき分散関係式を導出し，表面プラズモンポラリトンモードの波数を求める研究がなされた [1]-[3]．このような表面プラズモンポラリトンモードは，真空中から光波を入射しても励振できないことが分かっており，誘電体を介して入射した光波によって励振される．Pocklington の積分方程式を用いた，線状のナノアンテナの数値解析法に関する研究が行われた [4]-[7]．これらの研究では，誘電体形状は線状とみなされ，その比誘電率や導電率は表面インピーダンスでモデリングされている．また，線状のナノアンテナの表面プラズモンポラリトンモードの波数を分散関係式から求め，Pocklington の積分方程式と組み合わせてナノアンテナを解析する手法も提案されている [8]，[9]．Pocklington の積分方程式を用いた線状ナノアンテナの数値解析法は，簡便で扱いやすいという利点はあるが，誘電体形状は線状とみなしているので，分極電流や分極電荷のモデリングに問題がある．

誘電体内部の分極電流を直接モデリングする数値解析法として，体積積分方程式を用いたモーメント法が挙げられる [10]-[12]．体積積分方程式を用いたモーメント法は，誘電体内部の分極電流を基底関数で展開し，その重みを数値的に求めることで誘電体内部の分極電流を得る手法である．Pocklington の積分方程式を用いたナノアンテナの数値解析法では，誘電体の分極電流が 1 次元の線電流で展開されるのに対し，体積積分方程式を用いたモーメント法では誘電体の分極電流が 3 次元の電流で展開されるので，不均質な誘電体や厚みを持った誘電体の数値解析に向いている．一方で，体積積分方程式を用いたモーメント法は，分極電流セグメント間の自己・相互インピーダンスの数値計算に非常に長い時間がかかるという問題点を持つことが知られている．

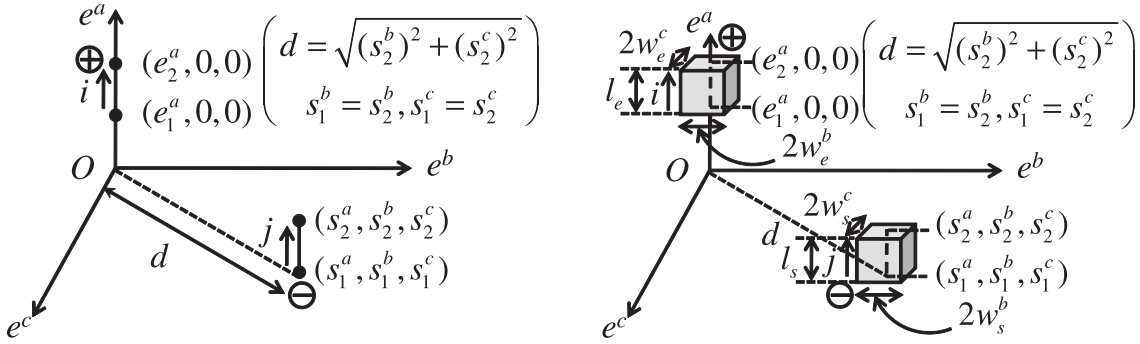
そこで本報告では，ナノアンテナの数値解析への応用を目指して，ベクトル型スーパーコンピュー

タによる体積積分方程式を用いたモーメント法の高速化を行う．誘電体内部の分極電流だけでなく，誘電体表面の分極電荷も考慮したブロックモーメント法のプログラムをベクトル化する [13], [14]．ベクトル化による高速化の有効性と，ベクトル化困難な部分を明らかにすることで，プログラムをチューニングする上での指針を示す．

2　体積積分方程式を用いたブロックモーメント法

本章では，端電荷を持つ線状モノポールセグメント間の自己・相互インピーダンス表示式を導出し，それを多重積分することでブロックモノポールセグメント間の自己・相互インピーダンス表示式を導出する [13]，[14]．ブロックモノポールセグメント間の自己・相互インピーダンスの表示式に現れる 5 重積分は座標変換を用いて 3 重積分に変換し，数値計算に要する時間を削減する [15]．

2.1　端電荷を持つモノポールセグメント間の自己・相互インピーダンス



(a) 端電荷を持つ一対の線状モノポールセグメントの一例と座標系.

(b) 端電荷を持つ一対のブロックモノポールセグメントの一例と座標系.

図 1: 自己・相互インピーダンス表示式の座標系.

ここでは, 端電荷を持つ線状モノポールセグメント間の自己・相互インピーダンス表示式を導出する. 図 1(a) に, 端電荷を持つ一対の平行なモノポールセグメントを示す. e 系の座標が観測セグメント, s 系の座標が波源セグメントを意味し, 座標変換と平行移動をすれば, 互いに平行な位置関係にあるモノポールセグメントの座標は全てこの座標に変換できる. なお, 本論文におけるモノポールセグメントの配置は平行あるいは垂直に限定する. また, 図 1(a) に示したのはあくまで一例であり, 電流の向きと電荷の組み合わせに応じて, 平行配置の組み合わせは他に 7 種類, 垂直配置の組み合わせは 4 種類ある.

さて, 図 1(a) の観測セグメント上の電流分布を表す基底関数として, 以下の区分正弦関数を利用する.

$$\mathbf{f}_e(e^a)=\frac{\sinh jk_0(e^a-e_1^a)}{\sinh jk_0\,|e_2^a-e_1^a|}. \tag{1}$$

同様に, 波源セグメント上の電流分布を表す基底関数として, 以下の区分正弦関数を用いる.

$$\mathbf{f}_s(s^a) = \frac{\sinh jk_0(s_2^a - s^a)}{\sinh jk_0\,|s_2^a - s_1^a|}. \tag{2}$$

次に, (1) 式を連続の式に代入することによって, 観測セグメント上の線電荷密度を以下のように得る.

$$\begin{aligned}\nabla\cdot\mathbf{f}_e(e^a) + j\omega\rho_e(e^a) &= \delta(e^a - e_2^a)\\ \rho_e(e^a) &= \frac{1}{j\omega}\left\{-jk_0\frac{\sinh jk_0(e^a - e_1^a)}{\sinh jk_0\,|e_2^a - e_1^a|} + \delta(e^a - e_2^a)\right\}.\end{aligned} \tag{3}$$

同様に, (2) 式を連続の式に代入することにより, 波源セグメント上の線電荷密度を以下のように得る.

$$\begin{aligned}\nabla\cdot\mathbf{f}_s(s^a) + j\omega\rho_s(s^a) &= -\delta(s^a - s_1^a)\\ \rho_s(s^a) &= \frac{1}{j\omega}\left\{jk_0\frac{\sinh jk_0(s_2^a - s^a)}{\sinh jk_0\,|s_2^a - s_1^a|} - \delta(s^a - s_1^a)\right\}.\end{aligned} \tag{4}$$

ここで, 以下に示すセグメント間の自己・相互インピーダンスの一般的な表示式に (1)-(4) 式を代入して整理すると, 端電荷を持つ線状モノポールセグメント間の自己・相互インピーダンス表示式が, 単積分の形で (5) 式のように得られる [13].

$$\begin{aligned}Z_{ij} &= \frac{j\omega\mu_0}{4\pi}\int_{e_1^a}^{e_2^a}\int_{s_1^a}^{s_2^a}\mathbf{f}_e(e^a)\cdot\mathbf{f}_s(e^a)\frac{e^{-jk_0R_{ij}}}{R_{ij}}ds^ade^a + \frac{j\omega}{4\pi\varepsilon_0}\int_{e_1^a}^{e_2^a}\int_{s_1^a}^{s_2^a}\rho_e(e^a)\rho_s(s^a)\frac{e^{-jk_0R_{ij}}}{R_{ij}}ds^ade^a\\ &= C_AI_A + C_BI_B + C_CI_C - C_DI_D.\end{aligned} \tag{5}$$

ただし, (5) 式中の $C_A \sim C_D$, $I_A \sim I_D$, そして R_{ij} はそれぞれ, $R_{ij} = \sqrt{(e^a - s^a)^2 + (s_2^b) + (s_2^c)^2}$, $C_A = \frac{7.5}{\sin k_0\left|e_2^a - e_1^a\right|\sin k_0\left|s_2^a - s_1^a\right|}$, $I_A = \sum_{p=1}^{2}\sum_{q=1}^{2} e^{-jk_0(ne_1^a + ms_2^a)}mnI_{pq}$, $C_B = \frac{15}{j\sin k_0\left|s_2^a - s_1^a\right|}$, $I_B = \sum_{p=1}^{2} I_{Bp}$, $I_{Bp} = -me^{jk_0(me_2^a - ms_2^a)}\int_{v(s_1^a,e_2^a)}^{v(s_2^a,e_2^a)}\frac{e^{-z}}{z}dz$, $C_C = \frac{15}{j\sin k_0\left|e_2^a - e_1^a\right|}$, $I_C = \sum_{q=1}^{2} I_{Cq}$, $I_{Cq} = -ne^{jk_0(-ne_1^a + ns_1^a)}\int_{w(s_1^a,e_1^a)}^{w(s_1^a,e_2^a)}\frac{e^{-z}}{z}dz$, $C_D = \frac{30}{jk_0}$, $I_D = \frac{e^{-jk_0R_{21}}}{R_{21}}$ である. なお, I_A の表示式に含まれる I_{pq} は以下のようになる.

$$\begin{aligned}I_{pq} &= jk_0\frac{1+mn}{mn}\int_{e_1^a}^{e_2^a}\int_{s_1^a}^{s_2^a}\frac{e^{jk_0(ms^a + ne^a - R_{ij})}}{R_{ij}}ds^ade^a\\ &= \sum_{h=1}^{2}(-1)^h\{-e^{jk_0e_h^a(n+m)}\int_{v(s_1^a,e_h^a)}^{v(s_1^a,e_h^a)}\frac{e^{-z}}{z}dz - e^{jk_0s_h^a(m+n)}\int_{w(s_h^a,e_1^a)}^{w(s_h^a,e_2^a)}\frac{e^{-z}}{z}dz\\ &+ e^{jk_0ju_0}\int_{x(s_h^a,e_1^a)}^{x(s_h^a,e_2^a)}\frac{e^{-z}}{z}dz + e^{-jk_0ju_0}\int_{y(s_h^a,e_1^a)}^{y(s_h^a,e_2^a)}\frac{e^{-z}}{z}dz\}.\end{aligned} \tag{6}$$

ただし, $v(s_i^a, e_h^a) = jk_0(-ms_i^a + me_h^a + R_{hi})$, $w(s_h^a, e_i^a) = jk_0(-ne_i^a + ms_h^a + R_{ih})$, $x(s_h^a, e_i^a) = jk_0(-ms_h^a - ne_i^a + R_{ih} + ju_0)$, $y(s_h^a, e_i^a) = jk_0(-ms_h^a - ne_i^a + R_{ih} - ju_0)$, $u_0 = d\sqrt{\frac{1+mn}{1-mn}}$, $d = \sqrt{(s_2^b)^2 + (s_2^c)^2}$, $m = (-1)^{p-1}$, $n = (-1)^{q-1}$ であり, (6) 式において, $1+mn = 0$ なら $I_{pq} = 0$, $1-mn = 0$ なら最後の 2 項を無視する. I_{pq}, I_B, I_C の表示式において, z を含む全ての積分は 2 つの指数積分の差を取ることで数値的に積分できる. この指数積分の数値計算が最も時間のかかる部分である. なお, モノポールセグメントの配置が垂直の場合も同様の定式化が必要だが, そちらは省略する.

2.2　ブロックモノポールセグメント間の自己・相互インピーダンス

図 1(b) に, 端電荷を持つ一対の平行なブロックモノポールセグメントを示す. 観測・波源モノポールセグメントにおける電流の向き, 端電荷の符号と位置は前項の図 1(a) と同じである. 観測ブロックモノポールセグメントの電流分布を表す基底関数は, 以下のように定義される.

$$\mathbf{f}_e(e^a) = \frac{1}{4w_e^b w_e^c} \frac{\sinh jk_0(e^a - e_1^a)}{\sinh jk_0 \left|e_2^a - e_1^a\right|}. \tag{7}$$

同様に, 波源モノポール上の電流分布を表す基底関数は以下の式で定義される.

$$\mathbf{f}_s(s^a) = \frac{1}{4w_s^b w_s^c} \frac{\sinh jk_0(s_2^a - s^a)}{\sinh jk_0 \left|s_2^a - s_1^a\right|}. \tag{8}$$

これらの基底関数は, 先頭の係数部分を除けばそれぞれ (1) 式, (2) 式と一致する. 従って, ブロックセグメント中央を通る線状モノポールセグメント間の相互インピーダンス Z_{ij} を (5) 式によって求め, Z_{ij} を 4 重積分すると, ブロックモノポールセグメント間の自己・相互インピーダンス表示式が以下のように得られる.

$$Z_{ij}^{block} = \frac{1}{16w_e^b w_e^c w_s^b w_s^c} \int_{-w_e^b}^{w_e^b} \int_{-w_e^c}^{w_e^c} \int_{s_1^b - w_s^b}^{s_1^b + w_s^b} \int_{s_1^c - w_s^c}^{s_1^c + w_s^c} Z_{ij} ds^c ds^b de^c de^b. \tag{9}$$

Z_{ij} の計算には単積分が含まれるため, (9) 式は 5 重積分を含む表示式である. この 5 重積分は，座標変換を用いて 3 重積分の和に変換することができ，数値計算の収束性および計算時間が改善する [15]. 詳細は省略するが，本報告ではこの 3 重積分の表現を用いて数値計算を行う. ナノアンテナの数値解析では，これらの式を用いて誘電体ブロックセグメント間の自己・相互インピーダンス行列要素を計算し，得られた行列方程式を解くことで電流を求める.

3　数値計算結果

銀でできた幅 10 nm，長さ 100 nm のナノダイポールアンテナを垂直入射の TM 波で励振し，その BRCS(Bistatic Radar Cross Section) を求めた. 銀の複素比誘電率は Drude モデルによって与え，周波数は 500 THz とした [9]. ナノダイポールアンテナはブロックダイポールおよびブロックモノポールセグメントに分割した. アンテナ表面での分極電流の不連続性がナノアンテナの数値解析結果におよぼす影響を明らかにするため，アンテナ表面にブロックモノポールセグメントを配置したモデル (w/monopole) とそうでないモデル (w/o monopole) のそれぞれを数値解析した.

数値解析によって得られた BRCS は図 2 に示す. アンテナ表面における端電荷を考慮したモノポールセグメントの有無によって，BRCS が異なっていることが分かる. モノポールセグメントをアンテナ表面に配置しないと，誘電体の表面における分極電流の法線成分の不連続性が無視されてしまう. その結果，誘電体アンテナの共振周波数にずれが生じ，波長に対するアンテナ長が変わるため，BRCS が変化したと考えられる. このように，誘電体の表面における分極電流の法線成分の不連続性のモデリングはナノダイポールアンテナの特性に大きく影響することが分かる.

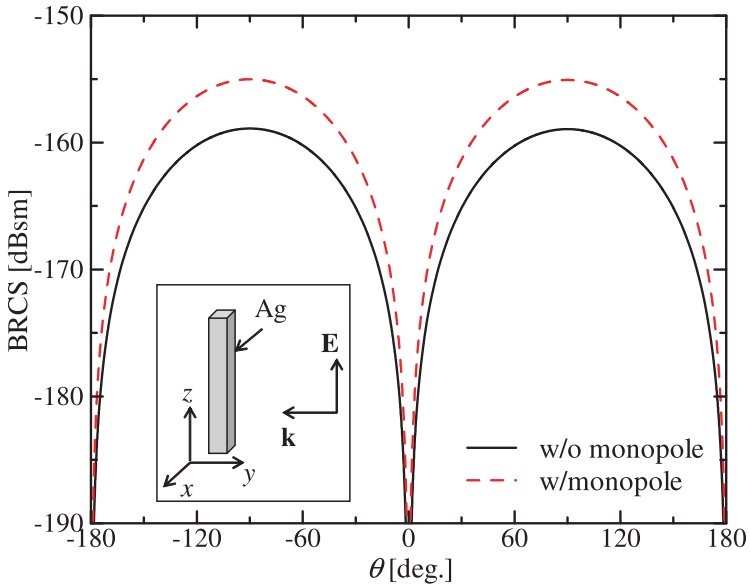


図 2: ナノダイポールアンテナの BRCS.

したがって，ナノアンテナの高精度な数値解析のためには，アンテナ表面へのブロックモノポールセグメントの配置が不可欠であると言える．

次に，ナノダイポールアンテナの数値解析に用いたプログラムをベクトル化チューニングした．サイバーサイエンスセンターのベクトル型スーパーコンピュータ SX-ACE を用いて，ブロックモノポールセグメント間の相互インピーダンスを数値計算したところ，ベクトル化チューニングを行わないときのベクトル演算率が約 0.03%，計算時間が 0.92 sec. であった．一方で，ベクトル化チューニングを行ったときは，ベクトル演算率が 2.7%，計算時間が 0.43 sec. であった．ベクトル演算率が向上したものの，最も計算時間を要する指数積分の箇所のベクトル化ができなかったため，計算時間は十分に低減できなかった．指数積分の箇所を除いた部分のベクトル演算率は 82.4%であり，計算時間は 0.02 sec. となるので， 指数積分の箇所のベクトル化が今後の課題である．

まえがきでも述べた通り，体積積分方程式を用いたモーメント法は，分極電流セグメント間の自己・相互インピーダンスの数値計算に非常に長い時間がかかることが知られている．分極電流セグメント間の自己・相互インピーダンスの数値計算は全ての分極電流セグメントの組み合わせに対して行うことになるので，その数値計算回数はセグメント数の 2 乗に比例する．また，誘電体に含まれるセグメント数はその体積に比例するので，数波長程度の誘電体の数値解析であっても，自己・相互インピーダンスの数値計算回数は 100 万回以上に達する．このような膨大な数値計算量を汎用のワークステーションで実施するのは極めて困難であるため，体積積分方程式を用いたモーメント法をベクトル型スーパーコンピュータ向けにチューニングすることは，ナノアンテナの数値解析の高速化において重要である．

4　まとめ

本報告では，ナノアンテナの数値解析のための体積積分方程式を用いたモーメント法の高速化を行った．プログラムにベクトル化チューニングを施した結果，指数積分の数値計算を除いた箇所のベクトル化に成功した．今後は，指数積分の箇所のベクトル化を行うなどして，SX-ACE の後継機である SX-Aurora におけるプログラムのベクトル化を行う．

参考文献

[1] Q. Cao and J. Jahns, “Azimuthally polarized surface plasmons as effective terahertz waveguides,” Opt. Exp., vol. 13, no. 2, pp. 511-518, 2005.

[2] L. Novotny, “Effective wavelength scaling for optical antennas,” Phys. Rev. Lett. 98(26), 266802 (2007).

[3] V. D. Miljković, T. Shegai, P. Johansson, and M. Käll, “Simulating light scattering from supported plasmonic nanowires,” Opt. Exp., vol. 20, no. 9, pp. 10816-10826, 2012.

[4] A. Locatelli, C. D. Angelis, D. Modotto, S. Boscolo, F. Sacchetto, M. Midrio, A.-D. Capobianco, F. M. Pigozzo, and C. G. Someda, “Modeling of enhanced field confinement and scattering by optical wire antennas,” Opt. Exp., vol. 17, no. 19, pp. 16792-1680, 2009.

[5] A. Locatelli, “Analysis of the optical properties of wire antennas with displaced terminals,” Opt. Exp., vol. 18, no. 9, pp. 9504-9510, 2010.

[6] K. Costa and V. Dmitriev, “Simple and efficient computational method to analyze cylindrical plasmonic nanoantennas,” Int. J. Antennas Propag., vol.2014, Article ID 675036, pp.1-8.

[7] J. L. de Souza and K. Q. da Costa, “Broadband wireless optical nano-link composed by dipole-loop nanoantennas,” IEEE Photonics Journal, vol. 10, no. 2, pp. 1-8, April 2018.

[8] J. Dorfmüller, R. Vogelgesang, W. Khunsin, C. Rockstuhl, C. Etrich, and K. Kern“Plasmonic nanowire antennas: experiment, Simulation, and Theory,” Nano Lett., vol. 10, no. 9, pp.3596-3603, 2010.

[9] M. Nafari, and J. M. Jornet, “Modeling and performance analysis of metallic plasmonic nano-antennas for wireless optical communication in nanonetworks,” IEEE Access, vol. 5, pp. 6389-6398, 2017.

[10] K. Tanaka and M. Tanaka, “Simulations of nanometric optical circuits based on surface plasmon polariton gap waveguide,” Appl. Phys. Lett. 82, 1158 (2003).

[11] K. Tanaka and M. Tanaka, “Simulation of an aperture in the thick metallic screen that gives high intensity and small spot size using surface plasmon polariton,” J. Microsc., Vol. 210, Pt 3 June 2003, pp. 294—300.

[12] K. Tanaka and M. Tanaka, “Simulation of practical nanometric optical circuits based on surface plasmon polariton gap waveguides,” Opt. Exp., vol. 13, no. 1, pp. 256-266, 2005.

[13] M.A. Tilston and K.G. Balmain, “On the suppression of asymmetric artifacts arising in an implementation of the thin-wire method of moments,” IEEE Trans. Antennas Propag., vol.38, no.2, pp.281-285, Feb. 1990.

[14] 陳　強，ザイ　フイチン，袁　巧微，澤谷　邦男，“誘電体に対するガラーキンモーメント法-端部電荷を考慮した直方体モノポール間の自己・相互インピーダンスの単積分化-，” 信学論(B)，Vol. J91-B，No. 9，pp. 926-939，2008 年 9 月．

[15] A. Köksal and J.F. Kauffman, “Mutual impedance of parallel and perpendicular coplanar surface monopoles,” IEEE Trans. Antennas Propag., vol.39, no.8, pp.1251-1256, Aug. 1991.

[16] J.H. Richmond and N.H. Geary, “Mutual impedance of nonplanar-skew sinusoidal dipoles,” IEEE Trans. Antennas Propag., vol.AP-23, no.3, pp.412-414, May 1975.

［共同研究成果］

金蒸気ー金ナノ粒子変換を伴う高エンタルピー水プラズマジェットの数値シミュレーション

茂田　正哉
大阪大学　接合科学研究所

金ナノ粒子の大量創製プロセスにおける熱・物質輸送過程の解明を目的とし，水を用いたプラズマジェットとその境界領域において蒸気分子からエアロゾル粒子へと集団的な変換過程にある金ナノ粒子群の対流・拡散輸送の数値シミュレーションを，東北大学サイバーサイエンスセンターのスーパーコンピュータ SX-ACE の 512 ノードを利用した並列計算によって実現した. その結果，水プラズマジェットへの周囲気体の巻込みとともに 2 層構造の渦輪列が生じ，金蒸気分子および金ナノ粒子はそれぞれ異なる対流スケールによって輸送されることが初めて明らかとなった.

1.　はじめに

直径がナノメートルスケールの超微粒子（ナノ粒子）はバルクの材料とは大きく異なる物質機能を示すことが知られており，特に金ナノ粒子は生体親和性が高く，光線力学的療法，光温熱治療，X 線 CT イメージング，ドラッグデリバリー，バイオセンサーへ応用が期待されている[1]. 近年, ナノ粒子の大量創製を可能にするツールとして, 高いエネルギー密度と化学的活性を有し，さらに外部電磁場によって制御可能[2, 3]な流体である高エンタルピープラズマに注目が集まっており，精力的に研究が進められている[4]. しかし高温の発光体である熱プラズマの流動場について実験によって得られる情報には限りがあるため，プロセスの効率的な制御のために必要な知見は今尚少ない. また流動場だけでなく，ナノ粒子の形成過程はマイクロ秒～ミリ秒の現象であるため，その集団的形成のメカニズムや輸送現象を直接計測することも困難である. そのため，高エンタルピープラズマによるナノ粒子創製プロセッシングは依然として現場の技術者や研究者の経験に依っているところが多く，また莫大な時間とコストを要しているのが現状である. そこで本研究では，金ナノ粒子の大量創製プロセスにおける熱・物質輸送現象の解明を目的として，身近な物質であり，かつ熱容量が大きい水を用いたプラズマジェットとその境界領域において蒸気分子からエアロゾル粒子へと集団的な変換過程にある金ナノ粒子群の対流・拡散輸送の数値シミュレーションを行なう.

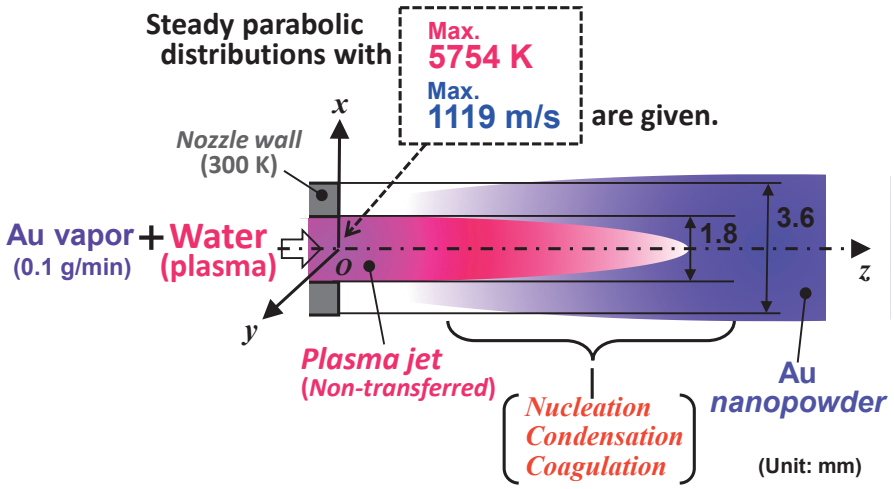


Fig. 1 Computational domain

2.　仮定および支配方程式

2.1　高エンタルピープラズマ－非電離気体共存系の熱流動場

高エンタルピープラズマ流の通常の生成条件では，圧力は大気圧と同程度で，プラズマおよび非電離気体を含む流体全域にわたって局所熱平衡が成り立ち，また光学的に薄いと仮定できる．このとき支配方程式は以下のような質量・運動量・エネルギーに関する保存式となる．

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \tag{1}$$

$$\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}) = -\nabla p + \nabla \cdot \left\{ \eta \left[(\nabla \boldsymbol{u}) + (\nabla \boldsymbol{u})^{tr} - \frac{2}{3}(\nabla \cdot \boldsymbol{u})\boldsymbol{I} \right] \right\} \tag{2}$$

$$\frac{\partial}{\partial t}(\rho h) + \nabla \cdot (\rho \boldsymbol{u} h) = \nabla \cdot \left(\frac{\lambda}{C_p} \nabla h \right) + \frac{\partial p}{\partial t} + \rho \boldsymbol{u} \cdot \nabla p - q_{rad} + q_{con} + \Phi \tag{3}$$

ここで，ρ は密度，$\boldsymbol{u}$ は速度ベクトル，t は時間，p は圧力，ηは粘性係数，$\boldsymbol{I}$ は単位行列，σは導電率，$\boldsymbol{E}$ は電場ベクトル，$\boldsymbol{B}$ は磁束密度ベクトル，h はエンタルピー，λ は熱伝導率，C_p は定圧比熱，q_{rad} は放射損失，q_{con} はナノ粒子の凝縮熱，Φは粘性散逸である．また tr は転置を意味する．

2.2　蒸気からナノ粒子への集団的変換および輸送

高エンタルピープラズマプロセスにおけるナノ粒子形成過程の概要は次の通りである．

① プラズマの高温場において原料が蒸発する
② その原料蒸気は温度低下に伴い過飽和状態となる
③ 多数の臨界核が生成する（均一核生成）
④ その臨界核に原料蒸気が凝縮することでナノ粒子が成長する（不均一凝縮）
⑤ 同時にナノ粒子同士も衝突・合体してより大きなナノ粒子となる（粒子間凝集）

ナノ粒子は，時間スケールの異なる均一核生成や不均一凝縮を経るのみならず，２～３桁に及ぶ直径差を持つ多数の粒子同士が衝突し合体しながら集団として成長していく．これまで分子動力学に基づいた数値計算も行われてはいるものの，現在のコンピュータ性能の限界から，数十個の核の生成過程を数ナノ秒間分ほど追跡することしかできていないため[5]，ナノ粒子群全体の成長を取り扱うことは実質的に不可能である．そこでエアロゾル学に基づく理論的および数値的なアプローチが有効とされている．

本稿では，簡潔なモデルによりナノ粒子の集団的形成過程を表現するために，ナノ粒子は局所的には同じ粒径を持つ球体であるとする．また帯電の効果は無視し，粒子温度は周囲の流体の温度と等しいとする．このとき支配方程式は以下のように記述できる[3, 6-8]．

$$\rho \frac{\partial}{\partial t}\left(\frac{n_p}{\rho}\right) + \rho \boldsymbol{u} \cdot \nabla\left(\frac{n_p}{\rho}\right) = \nabla \cdot \left[\rho D_p \nabla\left(\frac{n_p}{\rho}\right) \right] + J - 2\sqrt{2}\beta_0 n_p^{11/6} f^{1/6} + \nabla \cdot \left(K_{th} \eta \frac{n_p}{\rho} \nabla \ln T \right) \tag{4}$$

$$\rho \frac{\partial}{\partial t}\left(\frac{f}{\rho}\right) + \rho \boldsymbol{u} \cdot \nabla\left(\frac{f}{\rho}\right) = \nabla \cdot \left[\rho D_p \nabla\left(\frac{f}{\rho}\right) \right] + J g_c + \beta_0 (n_v - n_s) n_p^{1/3} f^{2/3} + \nabla \cdot \left(K_{th} \eta \frac{f}{\rho} \nabla \ln T \right) \tag{5}$$

$$\rho\frac{\partial}{\partial t}\left(\frac{n_v}{\rho}\right)+\rho\boldsymbol{u}\cdot\nabla\left(\frac{n_v}{\rho}\right)=\nabla\cdot\left[\rho D_v\nabla\left(\frac{n_v}{\rho}\right)\right]-Jg_c-\beta_0(n_v-n_s)n_p^{1/3}f^{2/3} \tag{6}$$

ここで，n は数密度，D は拡散係数，J は均一核生成率[9]，K_{th} は熱泳動係数[10]，T は温度，g は１つのナノ粒子に含まれるモノマー数の平均である．添え字 p, v, c および s はそれぞれ粒子，蒸気，臨界状態および飽和状態を表している．また f は次のように定義される変数である．

$$f=n_pg \tag{7}$$

また β_0 は衝突頻度に関するパラメーターであり，体積 v および質量 m を用いて以下のように表される．

$$\beta_0=\left(\frac{3v_v}{4\pi}\right)^{1/6}\sqrt{\frac{6k_BTv_v}{m_v}} \tag{8}$$

k_B はボルツマン定数である．式(4)および式(5)の右辺第 4 項は熱泳動を表している．式(4)～(6)の右辺に含まれる粒子成長に関わる項の詳しい導出については文献[7, 11]を参照されたい．

3.　計算手法

本研究では，水プラズマと非電離気体が相互作用しながら同時に存在する熱流動場を取り扱わなければならない．両者の間には大きな温度差に起因して粘性係数・熱伝導率・定圧比熱・導電率といった物性値だけでなく密度にも大きな差が生じている[12]．その一方で，流れ場におけるマッハ数は 10^{-3}～10^{-2} のオーダーにあり，工学的に有意な時間スケールでの流体運動を捉えるためには当該の系を「大きな密度変化を伴う非圧縮性流れ」として取り扱うべきである．すなわち，プラズマと周囲の低温気体との間に生じる速度，温度，密度，ナノ粒子濃度の急激な空間勾配を捉えながら，さらに時間ステップ幅を大きく取っても数値計算を安定的に進められるような計算手法が必要となる．そこで対流項をハイブリッド型 K-K スキーム[13]により，時間微分項を３次精度 Adams-Moulton-Bashforth 法により差分化し，改良型 PISO 法[14]と組み合わせることによって，上述の数値計算を実現することとする．高エンタルピープラズマ流動のシミュレーションに対するこれらの手法の有効性については文献[15]を参照されたい．なお，拡散項，圧力勾配項，熱泳動項には２次精度中心差分を用いる．

4.　計算条件

Fig. 1 に計算領域を示す．内径 1.8 mm，外径 3.6 mm の噴出孔から最高温度 5754 K，最高速度 1119 m/s の放物線型の分布をもつプラズマジェットが，400 K・大気圧の非電離水蒸気雰囲気の計算領域へ定常的に噴出する．ナノ粒子の原料である金は既にプラズマ生成部で蒸気になっているものとしてプラズマジェットと共に 0.1 g/min で供給される．

これらの条件の下，３次元の計算領域を設けた．座標系の原点をプラズマジェット噴出孔の中心に取り，各軸方向にそれぞれ 0.02 mm の幅を持つスタガード格子を用いて，時間ステップ幅を 0.02 ms として計算を進めた．

本計算は，東北大学サイバーサイエンスセンターの SX-ACE において MPI 並列によって 512 ノードを使用して実現した．

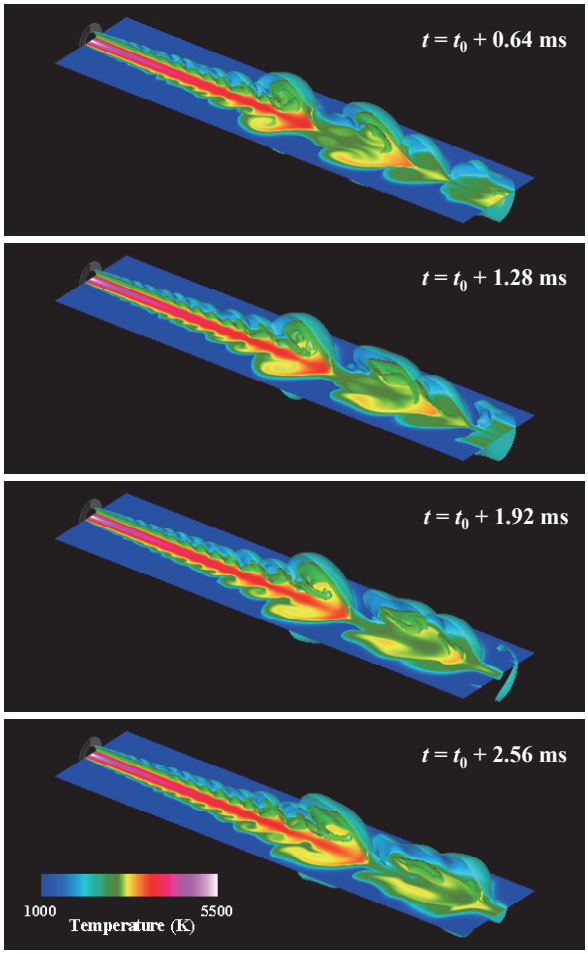


Fig. 2 Temperature distributions on $x = 0$ plane and isotherms of 2,000 K and 3,200 K in $y < 0$

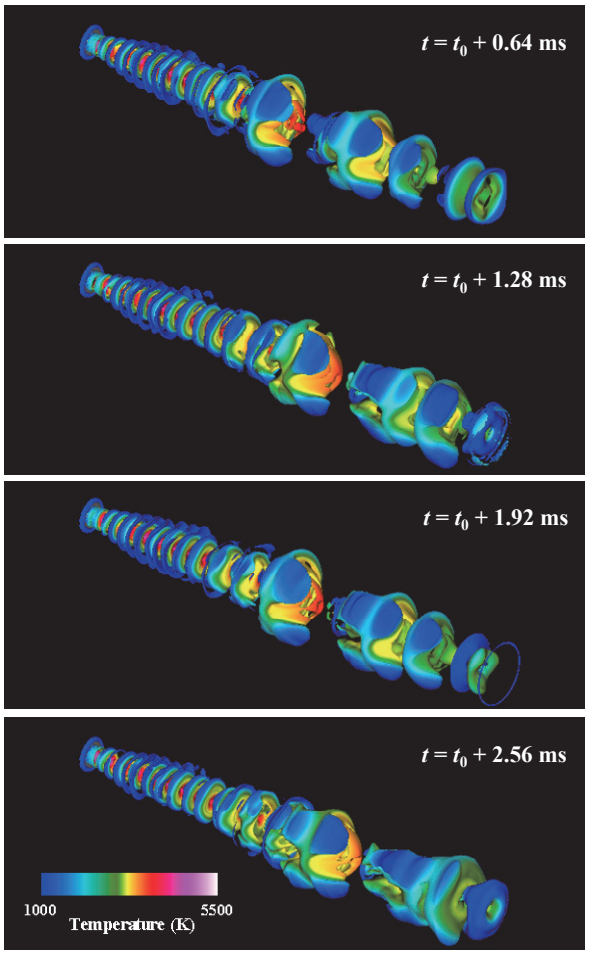


Fig. 3 Isosurfaces of the second invariant of velocity gradient tensor of 0.001 (normalized)

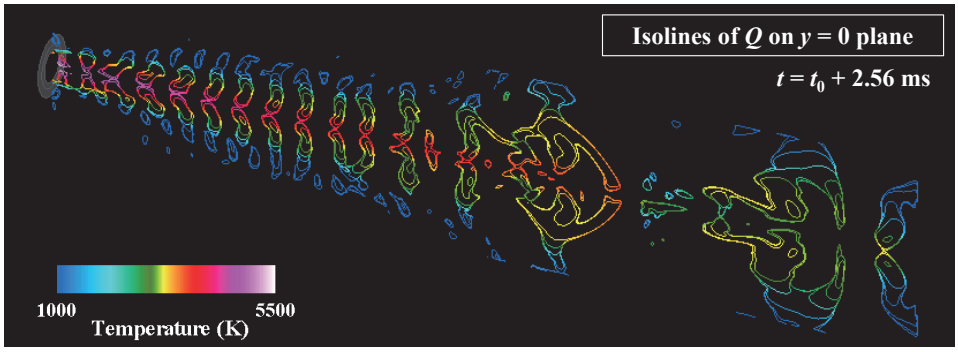


Fig. 4 Isolines of the second invariant of velocity gradient tensor on $y = 0$ plane

5.　計算結果

Fig. 2 に，ある時刻 t_0 から 0.64 ms，1.28 ms，1.92 ms，2.56 ms 経過した温度場を示す．Fig. 3 には同時刻における渦構造を示す．渦構造は速度勾配テンソルの第 2 不変量を噴出孔出口での平均流速と直径で無次元化した等値面 $Q = 0.001$ で表現されている．また Fig. 4 に時刻 t_0 から 2.56 ms 後の断面 $y = 0$ における渦構造を示す．水プラズマジェットと非電離気体の境界領域で Kelvin-Helmholtz 不安定性に起因する渦生成とそれによる巻込みが生じている．温度分布の模様を流脈として捉えると境界領域における渦列の層は 1 層であるように見えるが, Fig. 3 および Fig. 4

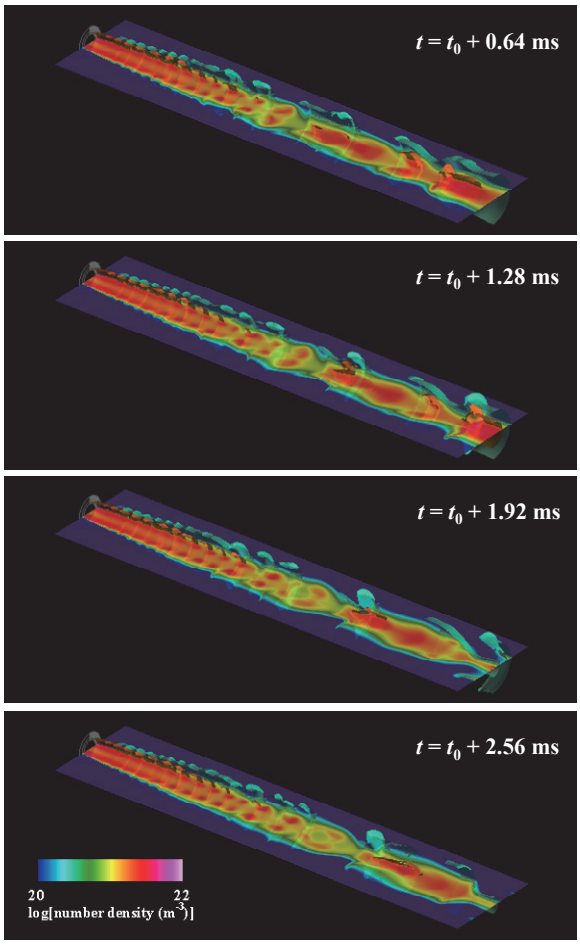


Fig. 5 Number density distributions of gold vapor molecules on $x = 0$ plane and isosurfaces of 3.2×10^{20} and 2.0×10^{21} in $y < 0$

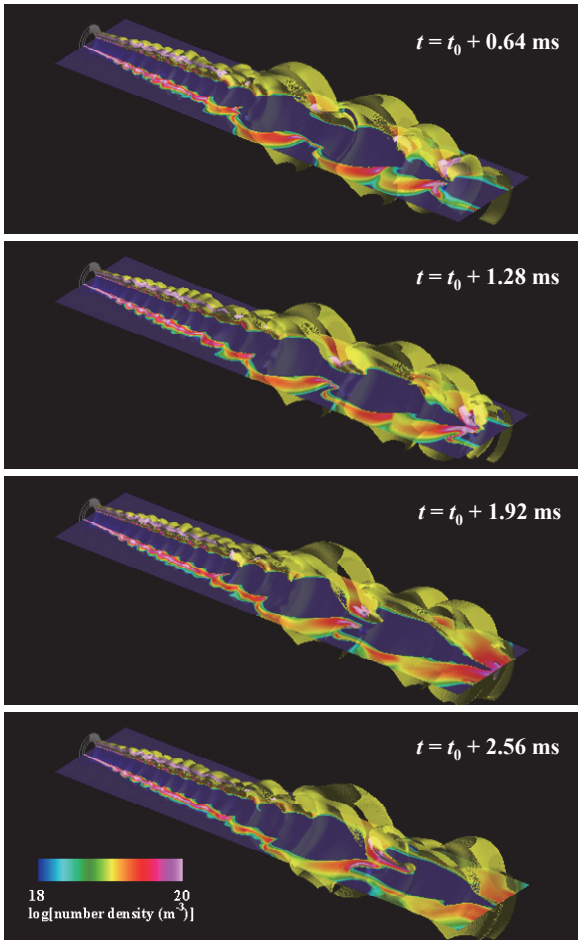


Fig. 6 Number density distributions of gold nanoparticles on $x = 0$ and isosurfaces of 1.0×10^{19} and 1.0×10^{20} in $y < 0$

から高温のプラズマジェットの内部から境界領域にかけて太い渦輪列が生じ，その周囲の低温領域には細い渦輪列が生じるという，2 層構造となっていることがわかる．

Fig. 5 および Fig. 6 に金蒸気分子および金ナノ粒子それぞれの数密度分布を示す．水プラズマジェット内部は高温であるため金ナノ粒子は存在せずに分子の状態をとっているが，プラズマ－非電離気体の境界領域でナノ粒子へと変換される．そのため金ナノ粒子群はプラズマ外に存在するが，上流域に多く存在するナノ粒子の数が下流域で減少している．これは拡散による希釈効果だけでなく，粒子間凝集（衝突・合体）による集団成長の表れでもある．

Fig. 5 において金蒸気の分子数密度が島状に高くなっている領域が見られるが，これは Fig. 2 における周囲の低温流体の巻込みによる温度低下によって分子群が濃化しているためである．Fig. 3 および Fig. 4 に示したプラズマ内外の渦輪列の 2 層構造から，金蒸気分子は大きな渦によって，金ナノ粒子は小さな渦によって，それぞれ異なる対流輸送スケールを持つことが明らかになったといえる．

6.　まとめ

金ナノ粒子の大量創製プロセスにおける熱・物質輸送現象の解明を目的として，水を用いたプラズマジェットとその境界領域において蒸気分子からエアロゾル粒子へと集団的な変換過程にある金ナノ粒子群の対流・拡散輸送の数値シミュレーションを行なった．東北大学サイバーサイエ

ンスセンターのスーパーコンピュータSX-ACEの512ノードを利用した並列計算によって数値シミュレーションが達成された．その結果，水プラズマジェットへの周囲気体の巻込みとともに2層構造の渦輪列が生じ，金蒸気分子および金ナノ粒子はそれぞれ異なる対流スケールによって輸送されることが初めて明らかとなった．

謝辞

本研究は，東北大学サイバーサイエンスセンターのスーパーコンピュータを利用することで実現することができました．計算コードの高速化およびデータ整理にあたっては同センター関係各位よりご協力をいただきました．シミュレーションに要する費用の一部は科学研究費補助金（基盤研究(B)：19H01887）によって賄われました．水プラズマの物性値モデルおよび条件設定にあたっては，渡辺隆行先生（九州大学 教授）より多大なる協力をいただきましたので，ここに感謝の意を表します．

参考文献

[1] Elahi, N., Kamali, M., Baghersad, M.H., “Recent biomedical applications of gold nanoparticles: A review,” Talanta, 184 (2018), pp. 537-556.

[2] Sato, T., Shigeta, M., Kato, D., Nishiyama, H., “Mixing and magnetic effects on a nonequilibrium argon plasma jet,” Int. J. Thermal Sci., 40 (2001), pp. 273-278.

[3] Shigeta, M., “Numerical Study of Axial Magnetic Effects on a Turbulent Thermal Plasma Jet for Nanopowder Production Using 3D Time-Dependent Simulation,” J. Flow Control, Measure. Visual., 6 (2018), pp. 107-123.

[4] Shigeta, M. and Murphy, A.B., “Thermal plasmas for nanofabrication,” J. Phys. D: Appl. Phys., 44 (2011), pp. 174025-(16 pages).

[5] Lümmen, N. and Kraska, T., “Homogeneous nucleation and growth in iron-platinum vapour investigated by molecular dynamics simulation,” Euro. Phys. J. D, 41 (2007), pp. 247-260.

[6] Shigeta, M., “Simple nonequilibrium model of collective growth and transport of metal nanomist in a thermal plasma process,” Theor. Appl. Mech. Jpn., 63 (2015), pp. 147-154.

[7] Shigeta, M., “Modeling and Simulation of a Turbulent-like Thermal Plasma Jet for Nanopowder Production,” IEEJ Trans. Electrical Electronic Eng., 14, (2019), pp. 16-28.

[8] Shigeta, M., “Simulating Turbulent Thermal Plasma Flows for Nanopowder Fabrication,” Plasma Chem. Plasma Process., 40 (2020), pp. 775-794.

[9] Girshick, S.L., Chiu, C.P. and McMurry, P.H., “Time-dependent aerosol models and homogeneous nucleation rates,” Aerosol Sci. Tech., 13 (1990), 465-477.

[10] Talbot, L., Cheng, R.K., Schefer, R.W. and Willis, D.R., “Thermophoresis of particles in a heated boundary layer,” J. Fluid Mech., 101 (1980), pp. 737-758.

[11] Nemchinsky, V.A. and Shigeta, M., “Simple equations to describe aerosol growth,” Modelling Simul. Mater. Sci. Eng., 20 (2012), pp. 045017-(11 pages).

[12] Aubreton, J., Elchinger, M. F., Vinson, J. M., “Transport Coefficients in Water Plasma: Part I: Equilibrium Plasma,” Plasma Chem. Plasma Process., 29 (2009), pp. 149-171.

[13]Komurasaki, S., “A Hydrothermal Convective Flow at Extremely High Temperature,” 7th Int. Conf. Comp. Fluid Dynamics, (2012), ICCFD7-3001.

[14]Oliveira, P.J. and Issa, R.I., “An improved PISO algorithm for the computation of buoyancy-driven flows,” Numer. Heat Transfer B 40 (2001), pp. 473-493.

[15] Shigeta, M, “Turbulence modelling of thermal plasmas flows,” J. Phys. D: Appl. Phys., 49 (2016), pp. 493001-(18 pages).

［共同研究成果］

格子ガス法流体解析モデルとニューラルネットワークの融合

松岡　浩
技術士事務所ＡＩコンピューティングラボ

筆者らは、東北大学サイバーサイエンスセンターの共同研究公募制度により、平成 30 年度から３年計画で「リカレントニューラルネットワークによる高解像度流体解析コードの開発」を行っている。本稿では、その２年目に当たる平成 31 年度(令和元年度)以降に行った研究のうち、“格子ガス法流体解析モデル”と“ニューラルネットワーク”の融合により実現を目指す３つの目標：

１．不完全な実世界情報でも利用可能なリアルタームデータ同化
２．どんなに激しい乱流でも安定的に計算可能な超並列整数計算
３．極めて単純な要素モデルからでも自己組織化可能な粘性制御

について、その趣旨とこれまで試みた基本的な方法を概説する。

なお、説明の都合上、１年前の SENAC Vol. 53 No.1(2020.1)に掲載した研究成果「リカレントニューラルネットワークによる実世界流れ場解析用時間発展計算モデルの探求」と重複する内容の記載がある。これは、説明の飛躍をなくして読みやすくするためであり、ご容赦願いたい。

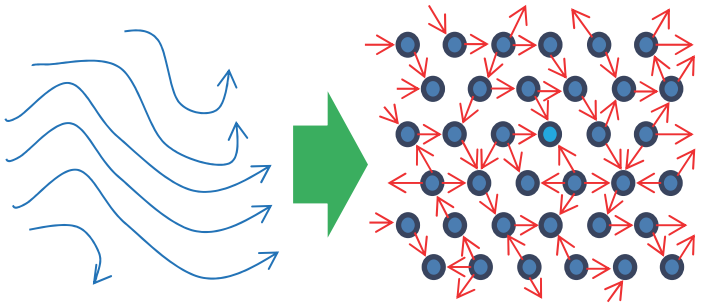
１．不完全な実世界情報でも利用可能なリアルタームデータ同化をめざして

（１）C. M. Teixeira が考案した高精度な格子ガスモデルから出発する。

本研究では、C. M. Teixeira が考案した４次元面心超立方体格子による「３速さ格子ガス法 54 速度モデル[1]」を出発点にして、ニューラルネットワークの計算原理を導入する。ニューラルネットワークは、入力信号の一部が欠落したり、間違っていたりしても、それなりに適切な答えを出力することができる。この特徴は、「計測システムを設置できない場所の物理場情報の欠落や、計測システムの故障等による間違った物理場情報の混在が一部にあったとしても、それなりに適切なシミュレーション結果を出力し続けることができる」という実世界応用への可能性を広げる。

また、Teixeira の 54 速度モデルに基づく格子流体解析の結果は、ナビエ・ストークス方程式を解く標準的な数値流体力学(非圧縮性流体)の結果と比較した場合、マッハ数に関する３次の精度まで一致することが Teixeira の論文[1]において証明されており（cf. 関連する重要論文[2]）、理論的にもそれなりの精度を期待できるものである。(cf. 関連する解説[3,4])

なお、格子ガス法流体解析の一般的な概要を**図１**に示す。



連続体流れ場
（連続体の流動）

仮想粒子の格子点挙動
（格子点上での衝突散乱
と格子点間の並進移動）

格子ガス法流体解析では、流体が存在する空間中に規則正しく格子点を配置し、特定の質量と運動量を担う多数の仮想粒子が、
①格子点上での「衝突散乱」、
②格子点間の「並進移動」
を繰り返しながら移動していく様子を疎視化(平均)して、マクロな流体物理量(密度、運動量、速度等)の挙動を得る。

図１．格子ガス法流体解析の一般的な概要

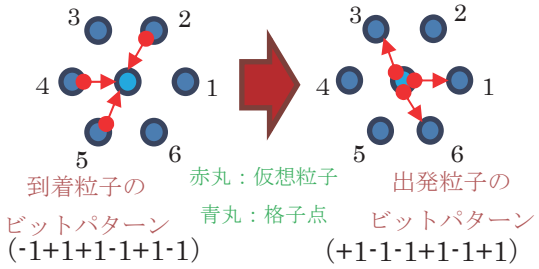
（２）格子ガス法の衝突散乱過程は多層パーセプトロンの計算に等しい。

格子ガス法流体解析では、流体が存在する空間(流体場)に一定間隔で多数の格子点を規則正しく配置する。次に、一定の質量を担う多数の“仮想粒子”が、これらの格子点上で互いに衝突散乱を繰り返しながら格子点間を次々と並進移動していく様子を計算で求める。そして、ある一定の時間が経過するごとに、近傍の格子点に存在する仮想粒子の質量や運動量を合計(疎視化)して、流体場各部におけるマクロな流体物理量(密度、流速等)の時間変化(スナップショット)を導く。

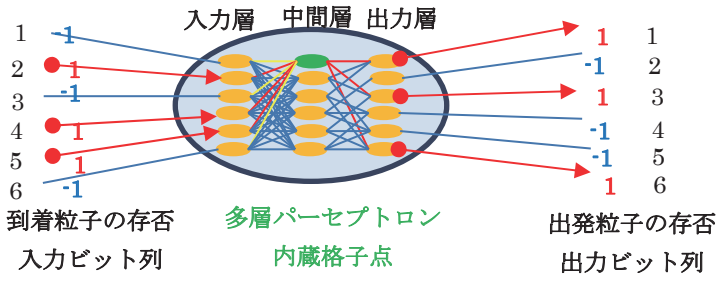
このとき、各格子点では、短い時間間隔⊿τが刻まれるたびに、近傍に存在するいくつかの格子点から仮想粒子が飛来する。これらの“到着粒子”は当該格子点上で他の到着粒子と衝突散乱を起こし、瞬時に飛行の向きを変えていろいろな近傍格子点に向かう“出発粒子”になる。このモデル上の仮定から、“到着粒子”も“出発粒子”も、近傍格子点間の移動をぴったり時間⊿τで行う必要があるため、仮想粒子がもちうる速度は、連続的ではなくいくつかの“離散ベクトル値”になる。Teixeira の 54 速度モデルでは、これらの離散ベクトル値を 54 個考え、ひとつの格子点には、それぞれが最大１個まで存在できるとする。従って、ひとつの格子点に存在できる仮想粒子の最大数は 54 個である。ただし、その内訳は、①６個の“静止粒子”、②速さが c で向きが異なる 24 個の“遅い運動粒子”、③速さが 2c で向きが異なる 24 個の“速い運動粒子”であり、静止粒子だけは例外で、同じ速度ゼロをもつ仮想粒子がひとつの格子点に複数存在してもよい。

ひとつの格子点における 54 種類の仮想粒子の存在状態は、54 ビットのビット列パターンで完全に表現できる。例えば、種類 i（i＝1, 2, ･･･, 54）の仮想粒子が存在する場合、i 番目のビットを「1」にし、存在しない場合は i 番目のビットを「0」にすればよい。この表現を用いれば、各格子点で生じる仮想粒子の衝突散乱は、「到着粒子の存否を表す 54 ビット列パターン」を、「出発粒子の存否を表す 54 ビット列パターン」へ変換する関数機能に他ならない。一般に、このような関数機能は、“多層パーセプトロン”で実現できる。従って、格子ガス法による時間発展計算を実行する立場からは、「流体場に存在する各格子点は、仮想粒子の衝突散乱過程を計算する“多層パーセプトロン”を内蔵している」と見なすことができる。

ここで、念のため、２次元格子ガス法 FHP モデル[5]における仮想粒子の衝突散乱過程を例にとり、その過程を計算できる具体的な多層パーセプトロンを構成してみよう。**図２**に衝突散乱過程の１例とその計算を実現できる多層パーセプトロンの構成例を示す。



２次元格子ガス法 FHP モデルでは、正三角形格子を用いる。各格子点の近傍格子点は６個あるが、左図では 2,4,5 から仮想粒子が３個飛来し、格子点上での衝突散乱の結果、1,3,6 の近傍格子点に向かって３個の仮想粒子が出発している。衝突散乱の前後で粒子数と運動量が保存されている。



左上図の衝突散乱過程の計算は、左図の３層パーセプトロンで実現できる。この例では、仮想粒子の存否を「1or0」ではなく「1or－1」に対応させているので注意。計算内容は、本文中で説明を参照されたい。

図２．２次元格子ガス法 FHP モデルにおける衝突散乱過程の１例とその計算を実現できる多層パーセプトロンの例

図２に示す多層パーセプトロンは、３層からなる。“入力層”は到着粒子の存否を表す入力値（±１）をそのまま中間層に伝える。中間層は入力ビット列のパターンを識別する。例えば、中間層の一番上に位置する緑色ニューロンにおいて、赤線入力に+1、黄線入力に−1 の重みを設定したとすれば、上から黄(−1)→赤(+1)→黄(−1) →赤(+1) →赤(+1) →黄(−1)の重みパターンになる。このとき、入力層への入力ビットパターンがこの重みパターンと一致するときにのみ積和が最大値(6)になる。この状態を、中間層ニューロンの活性化関数を“しきい値＝6 の階段関数”にすることによって識別する。そして、この中間層ニューロンの出力は、出力層のうち活性化すべきニューロン 1, 3, 6 にだけ接続しておく。出力層のニューロンは、接続されているシナプスの入力の重みをすべて+1 に設定し、活性化関数を“しきい値＝1 の階段関数”にする。このことで、いずれかひとつでも入力が+1 になれば活性化（出力＝+1）する。実際の適用時には、多数の衝突規則を実現する必要があり、それらが干渉しないように、このような階層を多段に重ねて適用する。

なお、衝突規則を確率的に適用する場合は、入力層のニューロンを増やし、そこに±１の入力を所定の確率でランダムに与えればよい。

（３）格子ガス法の時間発展過程はリカレント型多層パーセプトロンの時間展開計算に等しい。

格子ガス法流体解析における時間発展過程の全体もニューラルネットワークの計算で実現できることを示す。はじめに、仮想粒子の挙動をイメージしながら、

①“格子点”⇔“ニューロン”、

②“仮想粒子の存否”⇔“情報の内容（ここでは、±１とする。）”、

③“仮想粒子の格子点間移動経路”⇔“シナプス結合”

という対応を考える。これから、ただちに“相互結合型ニューラルネットワーク”の構造を連想できる。このとき、仮想粒子は、格子点間を相互に移動できるので、シナプス結合において、情報はニューロン間を双方向に流れうる。

次に、時間発展計算の手順を明示的に表現するため、上記の“相互結合型ニューラルネットワーク”を時間方向に展開する。このため、「時刻ステップｔにおけるニューロン」と「時刻ステップｔ＋1 におけるニューロン」を別のものとして扱い階層的に並べる。そして、「時刻ステップｔにおけるニューロン」の出力を「時刻ステップｔ＋1 におけるニューロン」の入力に伝えるシナプス結合を配置する。時間発展計算は、この繰り返しで実現できるため、「時刻ステップｔ＋1 におけるニューロン」の出力は、「時刻ステップｔにおけるニューロン」の入力にフィードバックされる。こうして、“リカレント型多層パーセプトロン”が得られる。このとき、各シナプス結合における情報の流れは、双方向ではなくひとつの向きになっている。イメージを**図３**に示す。

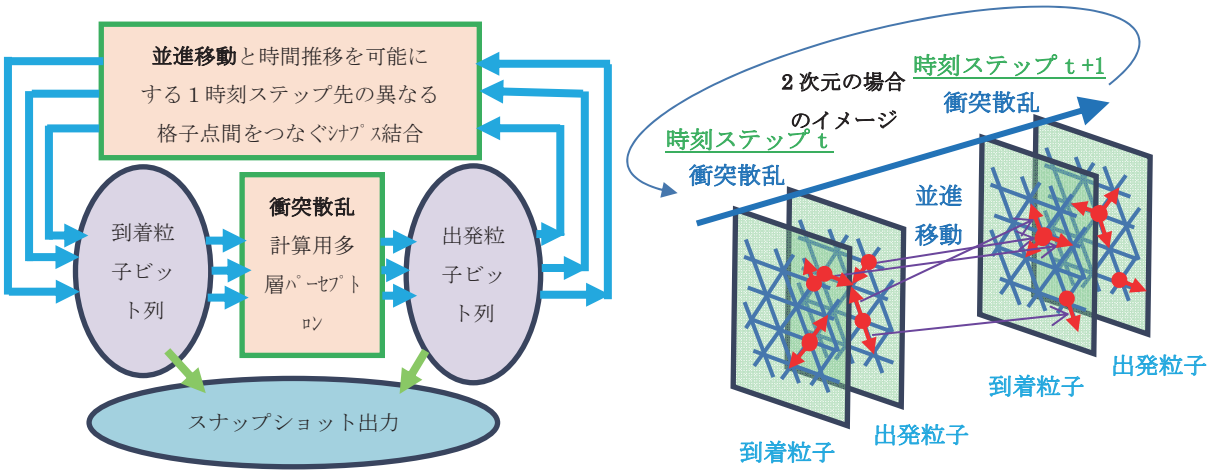


図３．格子ガス法の時間発展過程を計算するリカレント型多層パーセプトロン

（４）“誤差逆伝搬法”がリアルタイムデータ同化を実現するヒントになる。

“多層パーセプトロン”には、“誤差逆伝搬法”という強力な“教師あり学習法”が開発されている。実流体の過渡変化実測データまたは高精度なシミュレーションデータがあれば、例えば、ある場所のマクロな流体物理量の時系列データを教師データとして利用することができる。すなわち、各格子点に内蔵された多層パーセプトロンによる衝突散乱過程の繰り返し計算によって得られたマクロな流体物理量の時間変化が、実測データまたは高精度なシミュレーションデータに合うように、多層パーセプトロンの重みを学習させることができる。学習後の重みが、例えば、仮想粒子をある向きに加速するなど運動量保存則を一時的に破る操作に相当する場合もある。しかしながら、局所的な時空間平均では保存則を守るようにプログラムすることが可能である。そして、流体場のある場所の格子点に対するこのような操作の効果は、仮想粒子の衝突散乱過程を通じて短時間のうちに周辺の流体挙動に伝わり、物理的に矛盾のない新たな局所平衡状態をつくりだす。これは、「リアルタイムデータ同化」を実現する有力な手法開発のヒントになる。

（５）全ての機能を含めたリカレント型多層パーセプトロンを構成する。

格子ガス法流体解析における“衝突散乱”と“並進移動”の計算は、すべての場所で全く同じというわけではない。例えば、流体中に不浸透な静止した固体壁が存在する場合、通常、“粘着条件”としてバウンズバック条件が課せられる。この場合、到着粒子の衝突散乱計算によって得られた出発粒子(＝到着粒子に比べて分布の凸凹が緩やかになった“緩和粒子”)の存否計算の結果は捨てられ、到着粒子の速度を反転させた存否分布を出発粒子の分布とする（下図の緑線)。

さらに、実世界の流速計測情報をリアルタイムでフィードバックするような場合、外部環境条件に合うように個別粒子を巧みに加速するなど達人的な操作(“達人操作”)を行って、いわゆる“データ同化”(計測融合)を実現したい場合もある。また、後述する粘性制御のため、１時刻ステップ前の出発粒子の存否情報を用いる場合もある（下図の赤線)。

これらをまとめて**図４**を得る。

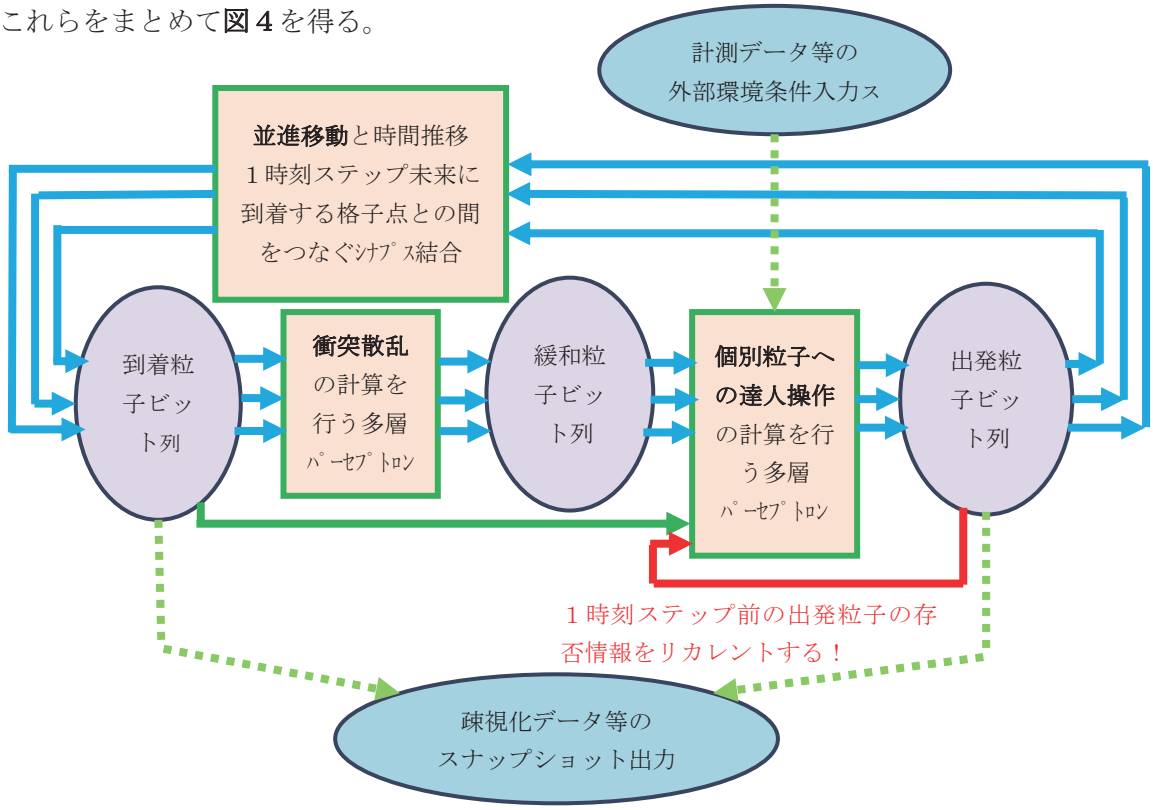


図４．全ての機能を含めたリカレント型多層パーセプトロン

２．どんなに激しい乱流でも安定的に計算可能な超並列整数計算を目指して

（１）“整数型多層パーセプトロン”による計算は乱流計算に貢献できる。

これまで考察してきたことを振り返ると、格子ガス法流体解析の時間発展計算過程は、“多層パーセプトロン”による計算の組合せで全てを実行できそうである。

特に、**図２**に示した３層パーセプトロンの例を考察すると、

①入力層への入力パターンが重みパターンと一致するときにのみ積和が最大値になるので、
　中間層ニューロンの活性化関数のしきい値を当該最大値に設定して入力パターンを選別する。

②活性化関数を“しきい値＝1 の階段関数”にすることで、いずれかひとつでも入力が+1 になればニューロンを活性化させて、出力＝+1 を実現する。

という２つの操作を組みあわせていることがわかる。これらの操作は他の“衝突散乱”の計算の場合にも広く適用できるものである。

また、“並進移動”については、シナプス結合のつなぎ方の問題であり、計算機上ではメモリ間の情報移動に過ぎないと考えても良いし、あるいは、格子点間にすべてのシナプス結合が存在していてどこの重みを“ゼロでなく+1”にするかの問題であると考えてもよい。

以上のことから、格子ガス法流体解析の時間発展計算の全過程は、各ニューロンの“入出力値”、“重み”、“階段型活性化関数のしきい値”をすべて微小な整数に設定した“整数型多層パーセプトロン”によって、微小整数の加減乗算のみで実行できることがわかる。

このことは、実数計算による時間発展計算を行う場合と比較して、計算の高速化と記憶容量の節約を可能にする。そして、流体計算の観点から最も重要なメリットは、実数計算の場合に生じる打切り誤差等の発生がなく、どんなに激しく変化する流れに対しても安定的に答えを出すことができる点にある。従って、乱流を解明する研究への応用が潜在的に期待される。

３．極めて単純な要素モデルからでも自己組織化可能な粘性制御をめざして

（１）格子ガスモデルのスケールで大胆な粘性発現機構を考えることも意義があるかも。

格子ガス法は、それが考案された初期、等方性やガリレイ不変性などの基本的要件を満たさないことが指摘されたが、いずれも改良された手法で克服されている。現在、格子ガス法が数々のメリットを持ちながらも、実用的な流体解析手法として一般的でない唯一の理由は、発現できる流体粘性を広い範囲で制御することが容易ではないからであろう。なお、格子ガス法の発展形態としては、実数計算を用いる“格子ボルツマン法”が成果を挙げている。しかし、筆者としては、実数計算が不要であるメリットを活かしたい。そこで、整数演算のメリットを残したままで、極めて単純な要素還元モデルから広範囲の流体粘性を自己組織化的に発現できる手法を探求した。

流体粘性は、結局のところ、ファンデルワールス力等の複雑な分子レベルの力学的相互作用から発現する。従って、流体粘性を高精度に扱うためには、分子レベルのスケールをシミュレーションする分子動力学を用いるのが自然なアプローチであろう。従って、分子レベルよりも桁違いに大きい格子ガスモデルのスケールで意図的な相互作用モデルを考案しても、そのモデルが発現する粘性効果は“大胆な近似”にしかならないとも思われる。

他方、ナビエ・ストークス方程式を解く王道的な“数値流体力学”は、マクロな連続体モデルであるにも拘わらず、粘性流体の挙動をかなり高精度にシミュレーションできる。また、格子ガスモデルと同様のスケールにおいて粒子間相互作用を設定する“粒子法”が開発されているが、いろいろな流体解析事例で成功を収めている。従って、格子ガスモデルのスケールで“大胆な近似”を導入しても、例えば統計平均的な効果により、マクロな疎視化スケールでは、現実の流体

挙動を高精度に模擬できる粘性発現機構になっている可能性があることも否定はできない。

（２）格子流体の粘性を自由に制御するには“等確率の常識”を超える必要がある。

「格子ガス法がマクロなスケールでどのような流体粘性を発現するか？」という問題は、より高いレイノルズ数領域の流体解析に格子ガス法を適用するため、「格子ガス法では、どこまで小さい値の流体粘性を発現できるか？」という点から重要である。

格子ガス法が提案された当初から、粘性に関する重要な論文がいくつも発表されている。例えば、Hénon は、「Viscosity of a Lattice Gas (1987)」[6]において格子ガス法によって発現される流体粘性の公式を導き、格子流体が発現する粘性は、格子点間隔を一定にした場合、衝突規則の詳細に依存して変化することを示した。また、(全ての衝突散乱が等確率で生じる場合) 粘性は必ず正の値になることを明らかにしている。

格子ガス法の粘性制御に関する論文例としては、Chen, Teixeria, Molvig は、「Digital physics approach to computational fluid dynamics: Some basic theoretical features (1997)」[7]において、通常の衝突散乱規則に“過緩和過程”を追加することで粘性をより小さくできることを示している。他方、Rothman は、「Negative-Viscosity Lattice Gases(1989)」[8]で負の粘性を発現させる方法を具体的に示すとともに、乱流解析への応用可能性を示唆している。いずれにしても、仮想粒子の衝突散乱過程を完全に等確率で行うのではなく、「到着粒子に特定の衝突規則だけを頻繁に適用したり、出発粒子を特定の向きだけに頻繁に放出させたり」というある種の“負のエントロピー発現操作”が提案内容である。すなわち、格子流体の粘性を自由に制御するには、“等確率の常識”を超える必要がある。

（３）幅広い流体粘性を発現させるため個別粒子への“達人操作”を導入してみる。

筆者らは、これまで“等確率の常識”を超えるいくつかの方法[9,10,11]を試してきた。例えば、１年前の SENAC Vol.53 No.1(2020.1)に掲載した研究成果[11]においては、「仮想粒子が並進する際の経路は直線ではなくジグザグに揺動する経路であると仮定し、同じ速度をもつ個別粒子がときどき接近しすぎてジグザグ経路を乗り換える際に、前回の接近の際に粒子が存在した側のジグザク経路を頻度多く選択する」というモデルを示した。このような“等確率の常識”を超える方法に共通する目標は、「望まれる粘性をぴたりと発現できるように、仮想粒子の衝突散乱過程において個別粒子の運動に影響を与える何等かの巧みな“達人操作”を見つける」という点にある。

筆者らの研究ではまだ本目標を達成していないが、本稿では、上述の１年前の試みとはまた別の試みを示す。ただし、「流体粘性は、流体速度の時間相関に関わるものである」という非平衡系統計力学の知識を参考にして、１時刻ステップ前の出発粒子の存否情報を“個別粒子に対して行う達人操作”を行うタイミングの判断基準のひとつにする点は１年前の試みから変えていない。

今回試みた具体的な**“個別粒子への達人操作”**は、“衝突散乱過程”のあと「ある向き(順向き)に出発するはずだった仮想粒子(緩和粒子)を、ある条件が整ったときにだけ確率的に逆向きに出発させる」というものである。このときの確率は、すべての向きに対して同じ値を適用する。

ただし、この操作は、あるひとつの粒子を意図的に逆向きに加速したことを意味する。従って、局所的な時空間平均で運動量保存則が破れないように、今回の研究では「順向きの加速と逆向きの加速は必ず交互に行う」ものとした。

（４）粘性制御のための達人操作も多層パーセプトロンの計算で実現できることを確認する。

今回試みた粘性制御のための“個別粒子への達人操作”も、多層パーセプトロンの計算で実現できることを確認しておく。その１例を**図５**に示す。

図５において、入力層は入力値±１をそのまま中間層に伝えるだけの役割を果たす。中間層は入力ビットパターンを識別する。赤線は+1、青線は−1 の重み設定を意味する。例えば、中間層の一番上に位置する黄色ニューロンにおいては、その７つの入力線は、上から赤(+1)→青(−1)→赤(+1)→赤(+1) →青(−1) →赤(+1) →青(−1)の重みパターンになっている。この中間層の重みパターンに一致するビットパターンが入力層にきた場合にのみ積和が最大値(７)になる。この状態を中間層ニューロンの活性化関数を“しきい値＝7 の階段関数”にすることによって識別する。このニューロンの出力は、出力層のうち活性化すべきニューロンである上のニューロンにだけ接続しておく。出力層のニューロンは、接続されているシナプスの入力重みをすべて+1 に設定し、活性化関数を“しきい値＝0 の階段関数”にする。このことで、いずれかひとつでも入力が+1 になれば活性化（出力＝+1）する。

なお、**図５**の例では、中間層のニューロンは、上から順に、後述する「順向き局所主流巻込操作」,「逆向き局所主流巻込操作」,「順向き近傍粒子引合操作」,「逆向き近傍粒子引合操作」を行うべき条件を識別するものである。

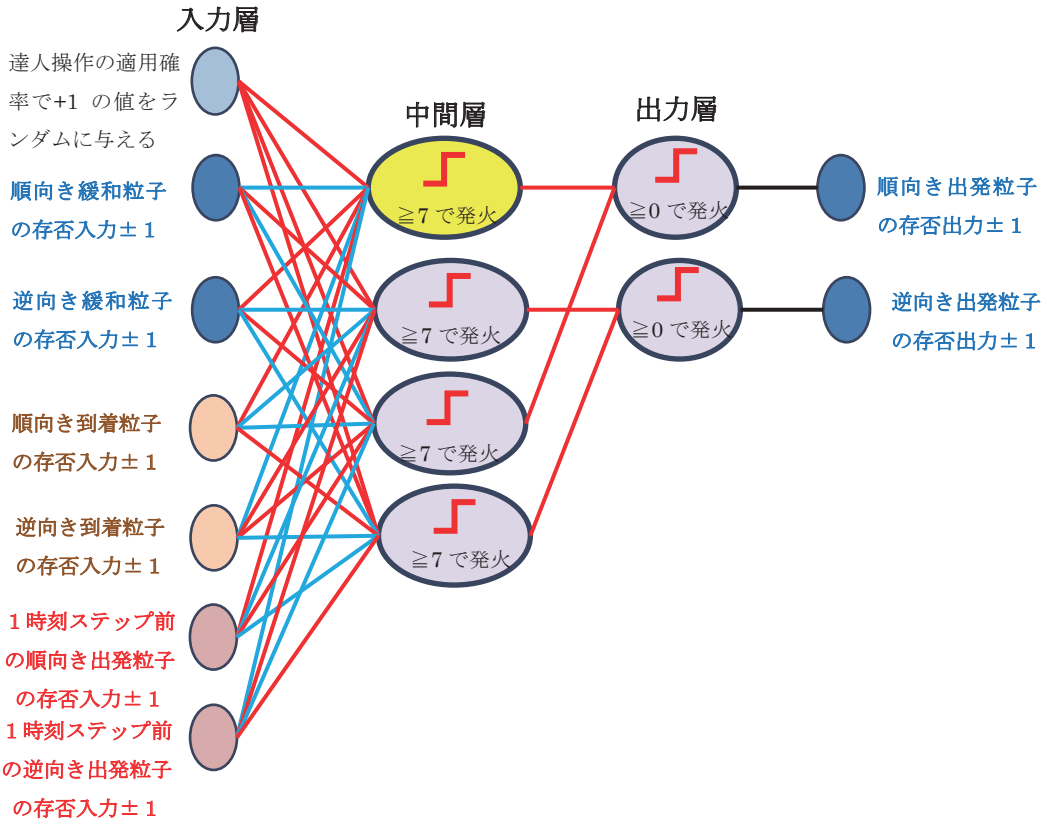


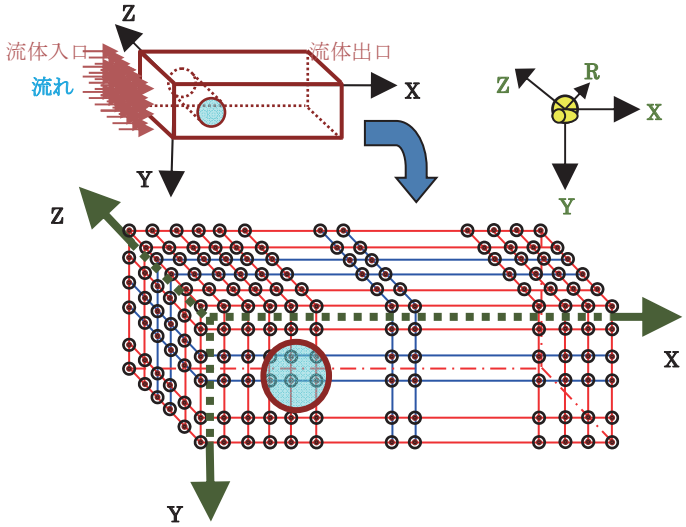
図５．粘性制御を行う多層パーセプトロン

（５）円柱後流の過渡変化計算により粘性変化の兆候を観察する。

上述した粘性制御の効果を確認するため、円柱後流のシミュレーション計算を行った。粘性に変化があれば、後流の様子が変化するはずである。

具体的には、東北大学サイバーサイエンスセンターのベクトル型スーパーコンピュータ SX-ACE を利用し、１CPU(４コア)による 40 分程度の小規模計算で求められる円柱後流の３次元過渡変化計算（約 530 万格子点で 12800 時刻ステップ、円柱軸に垂直なある平面上の流体挙動）を試みた。

計算条件等の詳細を**図６**に示す。



3次元空間中で多数の格子点をXYZの各方向に並べ、直方体形状の格子点配列を作る。各格子点は、その内部に4次元目の座標として R＝0,1,2,3 の位置を識別できる自由度をもつとする。これが左図であり、4次元面心超立方体格子を3次元空間へ投影した姿である。今回の数値シミュレーションでは、3次元縮退格子として、X方向に 864 個、Y方向に 384 個、Z方向に 16 個の格子点を配置した。

[過渡変化シミュレーションの条件]

シミュレーション計算を開始する時刻ステップ 0 の時点で、各格子点には、そこに存在できる仮想粒子の最大数の 20%の数の仮想粒子をランダムな向きの速度で配置する。この結果、疎視化して得られるマクロな流速はゼロであり、流体は、直方体形状の中で静止している。

次に、時刻ステップ 1 の時点から、＋X向きの速度をもつ仮想粒子をX＝0 の位置から注入していく。すると、時刻ステップが進むにつれて、流体全体が＋X向きのマクロな速度をもつようになる。このとき、＋X側の先にある直方体出口においては、出口直前に存在する格子点上の仮想粒子配置を、出口直後に存在する格子点の仮想粒子配置にコピーして、出口におけるマクロな流速の勾配がゼロになるという境界条件を近似的に実現した。また、±Y方向と±Z方向には、周期的境界条件を適用した。この流れの中の入り口に近い位置に、“Z方向の中心軸をもつ無限大の長さの円柱”を置き、その後流に生じる流体挙動を計算した。

図６．円柱後流の過渡変化シミュレーションを行う計算体系

シミュレーション結果は、あるZ断面上で疎視化された運動量分布を計算し、ParaView で過渡変化の動画とした。本稿では、この中からスナップショット画像を抜粋して表示する。

（６）流体粘性を変える“局所主流巻込操作(仮称)”の適用効果を観る。

“個別粒子への達人操作”として、「局所主流巻込操作（仮称)」なるものを考え、確率的に適用してみたので、以下、概要を示す。

マクロな速度がゼロで静止している流体が存在する場所にある格子点上では、移動することが可能なすべての向きから等方的に仮想粒子が到着し、衝突散乱後、すべての向きに等方的に仮想粒子が出発していると考えられる。もし、流体がある向き（順向き）にマクロな速度をもっていれば、順向き速度成分が正の速度をもった仮想粒子の方が、逆向き速度成分が正の速度をもった仮想粒子よりも、頻繁に到着し、かつ、頻繁に出発しているであろう。

次に、個々の格子点において、一直線上で相対する前後２つの向きへの仮想粒子の移動を考える。ある瞬間に順向きの到着粒子と出発粒子がともに存在し、逆向きの到着粒子と出発粒子がともに存在しない場合、局所的な流れの主流が瞬間的に順向きであると考える。このとき、格子点から逆向きに出発しようとした仮想粒子（緩和粒子）の一部について、主流に巻き込まれて速度を反転させ順向きの格子点に向かって並進していくように操作する。これを、ここでは、**“局所主流巻込操作（仮称)”**と呼ぶことにする。この場合のシミュレーション結果を**図７**に示す。

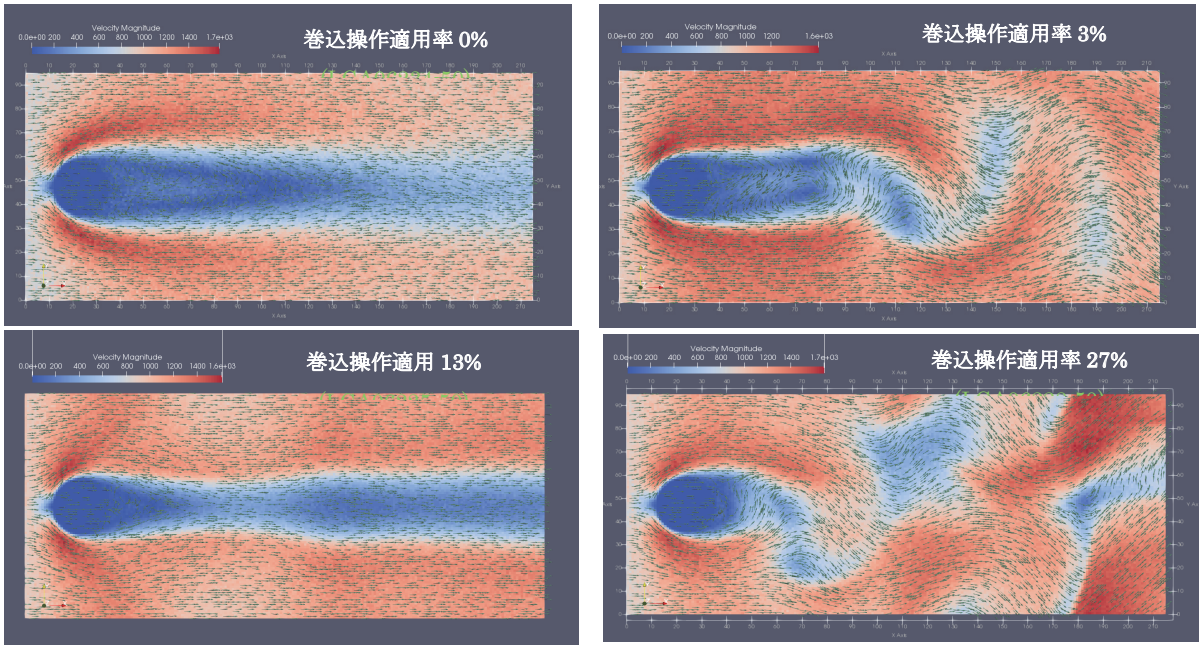


図７．“局所主流巻込操作(仮称)”の適用確率を変化させた場合の円柱後流の変化

(挙動が定常化した時点におけるある瞬間(12800 時刻ステップ目)の疎視化された運動量)

“局所主流巻込操作(仮称)”の適用確率を、0%､3%、13%、27%と変化させた場合、円柱後流の過渡変化挙動がかなり変化する。ここでは、挙動が定常化した時点におけるある瞬間（12800 時刻ステップ目）の疎視化された運動量を示す。同一格子点幅で同じ時刻ステップ経過後の状態であるが、シミュレーションを行った適用確率の範囲では、適用確率を上げるにしたがって生じる円柱後流の変化は、レイノルズ数が増加した場合に現れる変化に近く、定性的には、流体粘性が低下していることがわかる。

（７）流体粘性を変える“近傍粒子引合操作(仮称)”の適用効果を観る。

次に、“個別粒子への達人操作”として、「近傍粒子引合操作（仮称)」なるものを考え、確率的に適用してみた場合の結果を示す。

この場合も、まず、個々の格子点において、一直線上で相対する前後２つの向きへの仮想粒子の移動を考える。ある瞬間に順向きの出発粒子と逆向きの到着粒子がともに存在し、逆向きの出発粒子と順向きの到着粒子がともに存在しない場合、局所的には、その瞬間、順向き前方の方が後方よりも多くの仮想粒子が存在していたと考える。このとき、格子点から逆向きに出発しようとした仮想粒子（緩和粒子）の一部について、前方に多く存在する粒子からの粒子間引力が勝った結果、その速度を反転させ順向きの格子点に向かって並進していくように操作する。これを、ここでは、**“近傍粒子引合操作（仮称)”**と呼ぶことにする。この場合のシミュレーション結果を**図８**に示す。

“近傍粒子引合操作(仮称)”の適用確率を、0.1%､0.5%、2%、7%と変化させた場合、死水域の面積が変化する。また、このとき、円柱に衝突してバウンズバックする仮想粒子の X 成分運動量の変化から、円柱が流体から受けている抗力を求めた。この相対値は、それぞれ、5709, 4131, 3207, 2511 であり、死水域の減少とともに抗力が大きく減少した。円柱の配置体系、入口流入速度、格子点間隔等はすべて同じであるにも関わらず、抗力が大きく変化したことから、流体粘性が大きく変化したと考えられる。

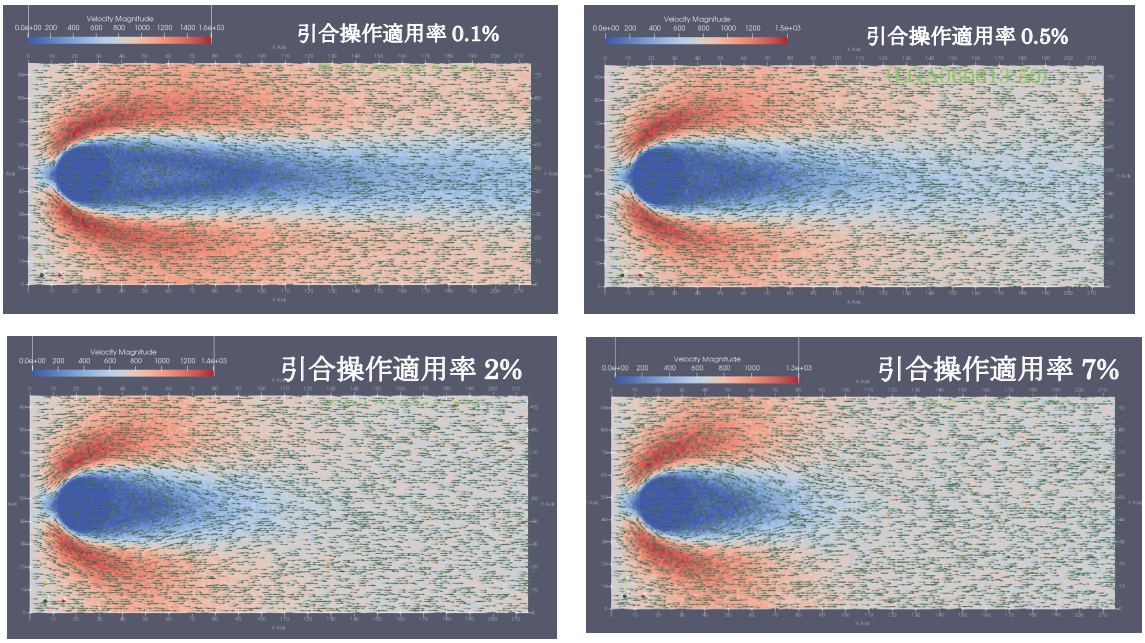


図８．“近傍粒子引合操作(仮称)”の適用確率を変化させた場合の円柱後流の変化

(挙動が定常化した時点におけるある瞬間(12800 時刻ステップ目)の疎視化された運動量)

４．まとめ

（１）今回試みた格子流体の粘性制御法に関する数値シミュレーションの結果について

今回試みた格子流体の粘性制御法は､『時刻ｔにおける衝突散乱の結果各方向に出発しようとする“緩和粒子”に対して、当該格子点に時刻ｔに到着した同方向の仮想粒子と時刻ｔ－１に出発した同方向の仮想粒子の有無に関する情報だけを観察し、ある条件が整った際にある確率で速度を反転させて出発させてやる』というものである。この方法の適用確率をいろいろ変えて円柱後流の挙動の違いを観察した結果、円柱の配置体系、入口流入速度、格子点間隔等はすべて同じであるにもかかわらず、円柱後流のうずの発生挙動や円柱に働く抗力が大きく変化することを確認した。現時点では定性的な評価にとどまるが、今回試みた方法は、流体粘性をかなり大きく変化させる効果がある。従って、格子ガス法において、高レイノルズ数領域の流体挙動を比較的少ない格子点数でも短時間に計算できる手法を開発する重要なヒントになるものと考えられる。

なお、流体粘性の制御については、各格子点における各方向の速度をもつ到着粒子と出発粒子の頻度をモニタリングすることでその時点の流体粘性を把握し、そのとき保持したい流体粘性の値に近づける操作をリアルタイムフィードバックで行うことを想定している。

（２）格子ガス法流体解析モデルとニューラルネットワークの融合について

本稿では、格子ガス法流体解析におけるすべての計算が、整数型多層パーセプトロンの組み合わせ計算として実行できる可能性を示した。“整数型”にこだわった理由は、実数計算に比べて計算速度の向上やメモリ節約の点で有利なこともあるが、格子ガス法がもつ最も重要なメリットである「誤差が発生しない安定した時間発展計算の実現」を大切にしたかったからである。これは、将来、乱流の解明に貢献できるかもしれないという期待感がある。

また、筆者らが行ってきたこれまでの研究では、ニューラルネットの学習原理を活用した「不完全な実世界情報でも利用可能なリアルタームデータ同化」という第一に掲げた目標を達成できていない。今後、この方向のシミュレーション事例を積み上げ、“格子ガス法流体解析モデル”と“ニューラルネットワーク”の融合が実用的な流体解析に貢献できることを示せたら幸いである。

謝辞

本稿で述べた研究課題では、“格子ガス法流体解析モデル”と“ニューラルネットワーク”の融合を目指して、いろいろなアイデアを多数試すことが非常に重要であった。この時期に、個々の計算モデルを試すたびに計算コードのチューニングに多くの時間をさくことは非効率的で悩ましい。この点、東北大学サイバーサイエンスセンターのベクトル型スーパーコンピュータ SX-ACE は、特別なチューニングをしないでも十分な計算速度を確保することができ、大変助かった。今後、さらに進化したベクトル機である SX-Aurora TSUBASA の利用が楽しみである。

また、利用にあたって同センター関係各位のご親切なご指導とご協力をいただき、心から感謝する次第である。今後とも同センターの有意義な活動を継続的に発展させられることを期待する。

参考文献

[1] Chistopher M. Teixeira, “Continuum Limit of Lattice Gas Fluid Dynamics”, MIT, 1993

[2] Uriel Frisch, Dominique d’Humières, Brosl Hasslacher, Pierre Lallemand, Yves Pomeau, Jean-Pierre Rivet, “Lattice Gas Hydrodynamics in Two and Three Dimensions”, Complex Systems, 1 (1987), pp. 649-707, 1987

[3] 松岡, 菊池,“多速さ格子ガス法実用化展開への手がかり”, SENAC Vol. 49 No. 4, pp. 1-15, 2016

[4] 松岡,“ビット演算による CFD と等価な高精度流体解析手法”, RIST News No. 64, pp. 17-28, 2018

[5] B. Haaslacher, U. Frisch, Y. Pomeau, “Lattice Gas Automata for the Navier-Stokes Equation”, Physical Review Letters Vol. 56, No. 14, pp. 1505-1508, 1986

[6] Michel Hénon, “Viscosity of a Lattice Gas”, Complex Systems, 1(1987), pp. 763-789, 1987

[7] Hudong Chen, Chris Teixeira, Kim Molvig, “Digital physics approach to computational fluid dynamics: Some basic theoretical features”, International Journal of Modern Physics C, Vol. 8, No. 4 (1997), pp. 675-684, 1997

[8] Daniel H. Rothman, “Negative-Viscosity Lattice Gases”, Journal of Statistical Physics, Vol. 56, Nos. 3/4, 1989

[9] 松岡, 菊池,“コンパクトな計算機によるリアルタイム流体解析の実現に向けて”, SENAC Vol. 51 No. 2, pp. 1-10, 2018

[10] 松岡, 菊池,“仮想粒子の並進移動過程に干渉効果を加味した流体解析の可能性”, SENAC Vol. 52 No. 2, pp. 18-27, 2019

[11] 松岡, 菊池,“リカレントニューラルネットワークによる実世界流れ場解析用時間発展計算モデルの探求”, SENAC Vol. 53 No. 1, pp. 25-33, 2020

［大学 ICT 推進協議会　2020 年度年次大会論文集より］

東北大学サイバーサイエンスセンター スーパーコンピュータ AOBA の紹介

山下 毅 [1)], 森谷 友映 [1)], 佐々木 大輔 [1)], 齋藤 敦子 [1)]
小野 敏 [1)], 大泉 健治 [1)], 滝沢 寛之 [2)]

1) 東北大学 情報部情報基盤課
2) 東北大学 サイバーサイエンスセンター

yamacta@tohoku.ac.jp

Introduction of Supercomputer “AOBA” at the Cyberscience Center of Tohoku University.

YAMASHITA Takeshi[1)], MORIYA Tomoaki[1)], SASAKI Daisuke[1)], SAITO Atsuko[1)]
ONO Satoshi[1)], OIZUMI Kenji[1)], TAKIZAWA Hiroyuki[2)]

1) Information Infrastructure Division, Information Department, Tohoku Univ.
2) Cyberscience Center, Tohoku Univ.

概要

東北大学サイバーサイエンスセンターは，全国共同利用設備として大規模科学計算システムの整備と，HPCI の資源提供機関としての役割を担っている。本稿では 2020 年 10 月に運用を開始したスーパーコンピュータ AOBA と，ユーザの利用環境および本センターが実施する高速化支援活動について紹介する。

1　スーパーコンピュータ AOBA

東北大学サイバーサイエンスセンター (以下，本センター) では，2020 年 10 月からスーパーコンピュータ AOBA の運用を開始した。スーパーコンピュータ AOBA はサブシステム AOBA-A(SX-Aurora TSUBASA，日本電気株式会社製)，サブシステム AOBA-B(LX 406Rz-2，日本電気株式会社製) の 2 種類の計算機システムと，ストレージシステム (DDN SFA7990XE，DDN 社製)，大判プリンタ，講習会端末およびそれらを接続するネットワーク機器群で構成される。図 1 にシステム構成図を示す。

以下ではスーパーコンピュータ AOBA の 2 つの計算機システムとストレージシステムについて，ハードウェアおよびソフトウェアの特徴と，利用者環境について紹介する。

1.1　サブシステム AOBA-A(スーパーコンピュータ)

■ハードウェア　今回導入した SX-Aurora TSUBASA は, 前スーパーコンピュータシステムの SX-ACE と同じくベクトルアーキテクチャを継承している。アプリケーション演算処理を行うベクトルエンジン (以下, VE) 部と，主に OS 処理を行うベクトルホスト (以下，VH) 部により構成される。PCIe カードに搭載される VE 部はベクトルプロセッサおよび高速メモリから構成され，x86/Linux が動作する VH と PCIe 経由で接続される。

今回本センターが導入した VE(Type 20B) は，理論演算性能 2,456GFLOPS(倍精度) となるマルチコア (8 コア) ベクトルプロセッサを 1 基，主記憶は 48GB を搭載し, 1.53TB/s という高いメモリバンド幅でプロセッサと接続されることで，高い演算性能とメモリ性能の両立を実現している。本センターのサブシステム AOBA-A は，1VH と 8VE が構成単位となる B401-8 モデルを採用し，サブシステム全体では 72 個の VH と 576 個の VE で構成される。VE と VH を合わせたシステム全体の理論演算性能は，1.48PFLOPS(倍精度)，総主記憶容量は 45TB，総メモリバンド幅は 895.68TB/s となる。図 2 に AOBA-A を構成する B401-8 と，それに搭載される VE の外観を図 2 に示す。

■プログラミング言語　SX-ACE と同じく，アプリケーションの実効性能を向上させる高度な自動ベクトル化・自動並列化機能を備えた Fortran/C/C++ コンパイラが利用できる。自動並列化機能および

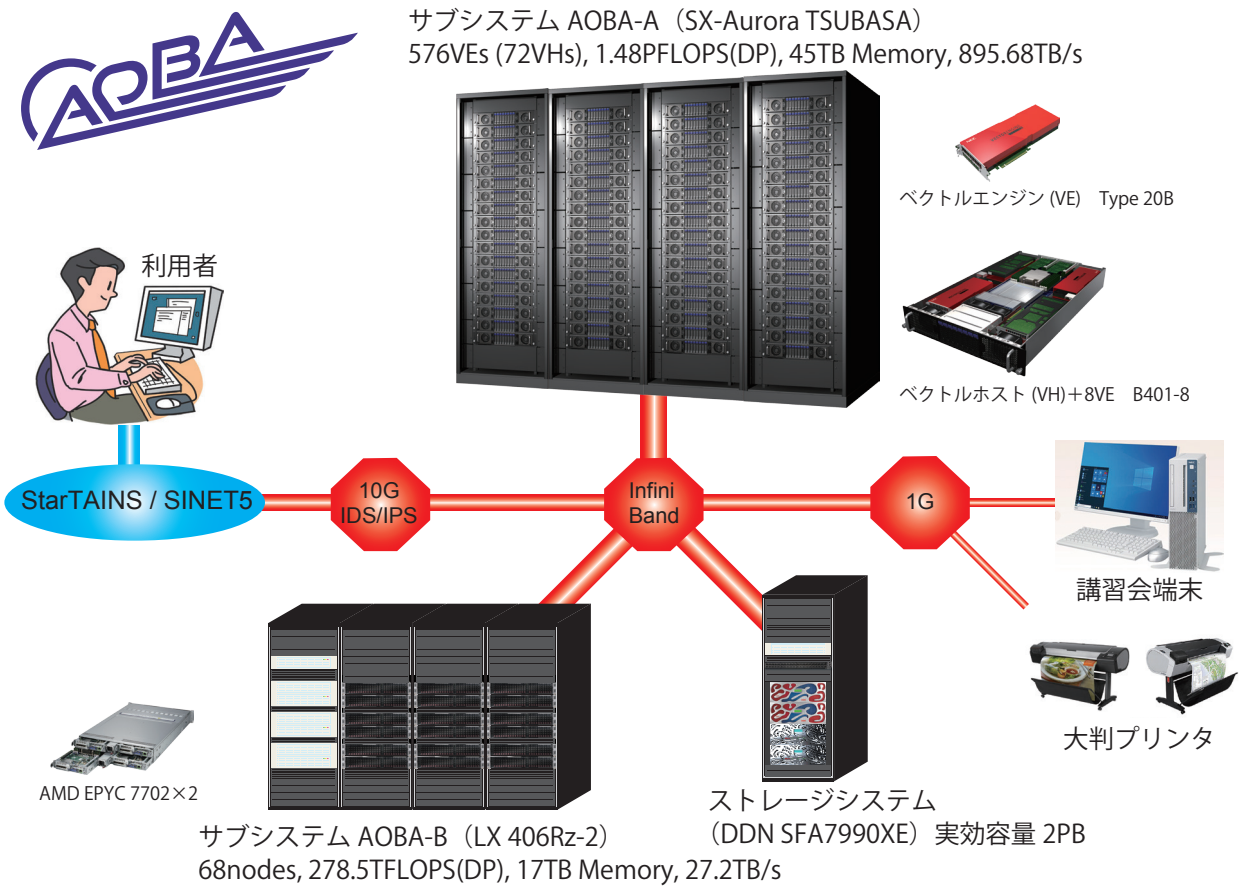


図 1　スーパーコンピュータ AOBA の構成

図 2　サブシステム AOBA-A(VE と B401-8)

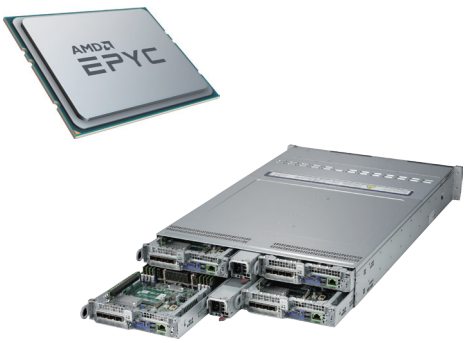


図 3　サブシステム AOBA-B(プロセッサと 4 ノードシャーシ)

OpenMP による共有メモリ並列実行と，システム構成に最適化された MPI ライブラリによる，分散メモリ並列実行が可能である。また科学技術計算ライブラリとして，VE に最適化された数学ライブラリのコレクション NEC Numeric Library Collection(NLC)が利用できる。

SX-ACE で動作していたプログラムをサブシステム AOBA-A に移植する場合には，そのプログラムを SX-Aurora TSUBASA 用のコンパイラでコンパイルし直す必要がある。なお，SX-Aurora TSUBASA 用コンパイラでは GNU コンパイラ互換性が強化され，指示行やコンパイルオプションが SX-ACE 用のものから変更されている。このため、移植時にはそれらの差異に注意が必要である。

■アプリケーション　サブシステム AOBA-A では，VE 向けに移植された商用アプリケーションの VASP や，オープンソースソフトウェア (OSS) の Quantum Espresso を利用できる。また，今後も VE 向けに移植されたアプリケーションを拡充する予定である。な

お，VASP の利用には利用者が契約したライセンスの提示が必要である。

1.2　サブシステム AOBA-B(並列コンピュータ)

■ハードウェア　今回導入した LX 406Rz-2 は，1 ノードに AMD EPYC プロセッサ 7702(64 コア) を 2 基と 256GB の主記憶装置を搭載し，合計 68 ノードで構成される。OpenMP，MPI を利用したノード内の並列処理は 128 並列まで可能で，ノードあたりの理論演算性能は 4.096TFLOPS(倍精度) である。サブシステム全体の理論演算性能は，278.5TFLOPS(倍精度)，総主記憶容量は 17TB，総メモリバンド幅は 27.2TB/s となる。サブシステム AOBA-B を構成する LX 406Rz-2 の 4 ノードシャーシと，それに搭載される AMD EYPC プロセッサの外観を図 3 に示す。

サブシステム AOBA-B は，ベクトル演算に不向きなプログラムや，商用アプリケーションや OSS の高速な実行を目的として導入された。

■プログラミング言語　Fortran/C/C++ コンパイラとして，AMD Optimizing C/C++ Compiler(AOCC)，GNU Compiler Collection(GCC) および，Intel Compiler(MKL，Intel MPI 含む) が利用できる。AOCC と GCC は OpenMPI ライブラリによる分散メモリ並列プログラムをコンパイル可能である。科学技術計算ライブラリとして，EPYC プロセッサに最適化された AMD Optimizing CPU Libraries (AOCL) が利用できる。Intel Compiler は旧システムからのソースコード移行用として，ライセンス数限定で利用できる。

■アプリケーション　商用アプリケーションとして Gaussian16 および VASP と，東北大学内利用者向けに MATLAB および Mathematica が利用できる。OSS として OpenFOAM および Quantum Espresso がインストールされている。なお，VASP の利用には利用者が契約したライセンスの提示が必要である。

1.3　ストレージシステム

ユーザのホーム領域として，高速アクセスかつ高密度ストレージである DDN SFA7990XE(DDN 社製) を導入した。図 4 にストレージシステムを示す。上図がストレージのコントローラー部で，下図がスピンドルの格納部である。SFA7990XE 上に ScaTeFS(日本電気株式会社製) の IO サーバを構築し，高速アクセス性能と IO サーバの耐障害性を確保した。ホーム領域は RAID6 で構成され，実効容量は 2PB である。

2　利用者環境

2.1　ログイン認証方式

図 5 に利用者向けサーバを示す。今回のシステムでは利用者の利便性とセキュリティの向上を考慮し，ログインサーバとフロントエンドサーバの 2 段構成としている。ログインサーバは外部ネットワークに公開され，緊急に対応が必要なセキュリティインシデントに迅速に対処可能としている。フロントエンドサーバはログインサーバからのみアクセス可能とし，利用者はフロントエンドサーバ上でソースコードのコンパイルやリクエストの投入を行う。

利用者の公開鍵は旧システムで利用していたものを引き続き利用できるので，ローカル PC に保存済みの秘密鍵とパスフレーズによるログインが可能である。

新規利用者は本センターウェブサイト上に提供される，公開暗号鍵ペア作成機能を用いてログインのための秘密鍵を作成する。

また，利用者のローカル PC とストレージシステム間で大規模なデータ転送を行うために，データ転送サーバも同じ鍵ペアを用いてログインと利用が可能である。

なお，GSI-SSH 認証によるログインはログインサーバを介さず，HPCI 利用者用のフロントエンドサーバから利用する。

2.2　プロジェクトコード

本センターではバッチリクエストの処理に NEC Network Queuing System V (以下，NQSV) を採用している。NQSV のジョブアカウント機能によって，ユーザが異なるプロジェクトで計算機資源を利用する際に，リクエスト単位の課金と予算管理を行うことが出来る。新システムでも引き続きこのプロジェクト

図 4　ストレージシステム

図 5　利用者向けサーバ

コードの機能を利用し，1 つの利用者番号で複数の請求先の使い分けを可能としている。プロジェクトコードと請求先の関係を図 6 に示す。

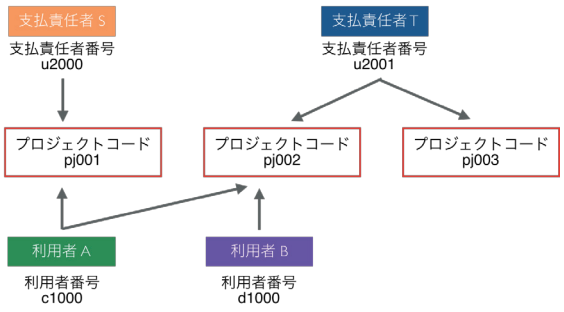


図 6　プロジェクトコード

■複数の請求先の利用　近年では研究費での利用に加え，課題採択形式で利用されるケースが増加している。プロジェクトコードを用いることにより，利用者 A は 1 つの利用者番号 (c1000) から請求先の異なる複数のプロジェクトコード (pj001, pj002) を使い分けることが可能である。バッチリクエスト投入の際に NQSV のジョブアカウント機能を用い，請求先としたいプロジェクトコードを指定することで利用者が複数の請求先を使い分けることが可能となる。また，支払責任者が複数のプロジェクトコード (pj002, pj003) を保有することも可能である。

■課題利用期間とプロジェクトコード　採択課題の利用期間が終了したものについては，該当するプロジェクトコードを無効にすることで利用者はリクエストを投入不可となる。また，利用可能な課題が追加された場合は，利用者番号に対してプロジェクトコードを追加設定することでリクエストの投入が可能となるので，それまで利用していた環境を引き続き利用することが可能である。

3　利用負担金と実行形態

3.1　利用負担金

大規模科学計算システムの利用負担金表を表 1 に示す。この表は大学・学術利用に適用され，民間企業利用は成果公開型の場合で本表記載の金額の 2 倍，成果非公開型の場合で本表記載の金額の 4 倍となる。

課金対象時間は各リクエストの利用 VE 数または利用ノード数と経過時間の積を秒単位で記録し，半年間の請求期毎に合算した後に時間単位に切り上げたものである。この課金対象時間に負担額を乗じた金額が請求金額となる。また，負担金を前払いすることで一定の課金対象時間まで利用することの出来る，定額制の導入も行った。定額制による利用は，年度途中に負担金を追加することによる利用継続も可能である。一定数の VE またはノードを研究グループで占有して利用する，占有利用も引き続き利用可能である。

3.2　サブシステム AOBA-A の実行形態

サブシステム AOBA-A で実行する場合の実行形態を表 2 に示す。今回導入したシステムでは，計算資源の効率的な利用と，リクエストの待ち時間短縮など利用者の利便性を考慮して，VH を共有する実行形態および VH を共有しない実行形態を利用者が選択できるようにした。それぞれの実行形態の例を図 7 に示す。どちらの実行形態も利用者の利便性を考慮し，最大経過時間を既定値 72 時間，最大値 720 時間として長時間のリクエスト実行を可能とした。

表 1　基本利用負担金【大学・学術利用】

区分	項目	利用形態	負担額及び課金対象時間
演算負担経費	スーパーコンピュータ	共有(無料)	利用 VE 数 1(実行数，経過時間の制限有) 無料
		共有(従量)	課金対象時間 ＝ (利用 VE 数÷ 8 を切り上げた数) ×経過時間 (秒) 課金対象時間 1 時間につき　125 円
		共有(定額)	負担額 10 万円につき課金対象時間 800 時間分使用可能
		占有	利用 VE 数 8　利用期間 3 ヶ月につき　270,000 円
	並列コンピュータ	共有(従量)	課金対象時間＝利用ノード数×経過時間 (秒) 課金対象時間 1 時間につき　22 円
		共有(定額)	負担額 10 万円につき課金対象時間 4,600 時間分使用可能
		占有	利用ノード数 1　利用期間 3 ヶ月につき　47,000 円
ファイル負担経費		共有	5TB まで無料，追加容量 1TB につき年額　3,000 円
		占有	10TB まで無料，追加容量 1TB につき年額　3,000 円
出力負担経費	大判プリンタによるカラープリント		フォト光沢用紙 1 枚につき　600 円 クロス紙 1 枚につき　1,200 円

備考

1. 負担額が無料となるのは専用のキューで実行されたものとし，制限時間を超えた場合は強制終了する。
2. 演算負担経費の課金対象時間については半期毎 (4 月から 9 月及び 10 月から 3 月) に合計し，1 時間未満を切上げて負担金を請求する。
3. 演算負担経費について定額制を選択した場合はスーパーコンピュータ及び並列コンピュータを課金対象時間の範囲内で共用できる。
4. 占有利用期間は年度を超えないものとし，期間中に障害，メンテナンス作業が発生した場合においても，原則利用期間の延長はしない。
5. ファイル負担経費については申請日から当該年度末までの料金とする。運用期間が 1 年に満たない場合は，月割りをもって計算した額とする。

表 2　サブシステム AOBA-A の実行形態

投入キュー名	利用可能 VE 数	最大メモリ	リクエストの実行形態	最大経過時間
sxf	1	48GB	無料の 1VE リクエスト (VH を共用する)	最大値 1 時間
sx	1	48GB	1VE リクエスト (VH を共用する)	既定値 72 時間 最大値 720 時間
sx	2〜256	12TB	8VE 単位で確保 (VH を共用しない)	
sxmix	2〜8	384GB	1VE 単位で確保 (VH を共用する)	
占有利用	契約 VE 数	48GB ×契約 VE 数	VE および VH を占有する	最大値 720 時間

表 3　サブシステム AOBA-B の実行形態

投入キュー名	利用可能ノード数	最大メモリ	リクエストの実行形態	実行時間制限
lx	1〜16	12TB	1 ノード単位で確保 (ノードを共用しない)	既定値 72 時間 最大値 720 時間
占有利用	契約ノード数	256GB ×契約ノード数	ノードを占有する	最大値 720 時間

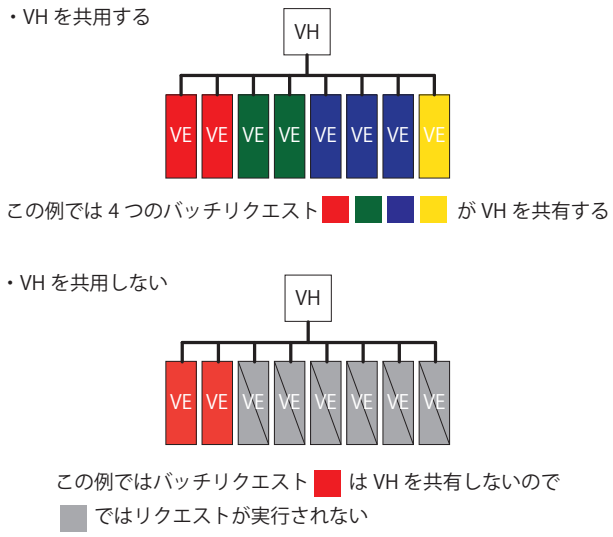


図 7　プロジェクトコード

■VH を共用する　投入したリクエストは他のリクエストと VH を共用して実行される。1VE を利用すると指定したリクエストは，VH を共有して実行される。また例として，2 個の VE を使うと指定したリクエストを sxmix キューに投入した場合，他 6 個の VE で別のリクエストが実行されることがある。利用する VE 数が 2〜8 個の場合，sxmix キューを選択すると実行に必要な VE 数が確保されやすく，リクエスト混雑時にも待ち時間を短縮することが出来る。

■VH を共用しない　投入したリクエストは他のリクエストと VH を共用しないで実行される。利用する VE 数を 2〜7 個と指定をしたリクエストを sx キューに投入した場合は，8 個の VE と 1 個の VH を確保する。他のリクエストと VH を共用しないため他リクエストのストレージへの I/O や VH 間通信の影響を受けにくく，演算時間のバラツキが少なくなる。

3.3　サブシステム AOBA-B の実行形態

サブシステム AOBA-B で実行する場合の実行形態を表 3 に示す。共有利用は lx キューのみであり，利用者は利用するノード数を 1 ノード単位で指定してリクエストを投入する。サブシステム AOBA-B でも利用者の利便性を考慮し，最大経過時間を既定値 72 時間，最大値 720 時間として長時間のリクエスト実行を可能とした。

4　高速化支援活動

本センターでは 1997 年より、ユーザアプリケーションの高精度化、大規模化の支援を目的とした高速化支援活動を，また 1999 年より共同研究制度を実施している。利用者、計算機科学を専門とするセンター教員、技術職員、およびベンダー技術者が連携してアプリケーションの高速化に取り組んでいる。

前スーパーコンピュータシステムの SX-ACE を運用した 5 年間においては，合計で 30 件の高速化支援を行った。単体性能では平均約 16.7 倍の性能向上を，並列性能では約 2.4 倍の性能向上を得ることが出来た。

図 8 に 1999 年から本センターで取り組んでいるセンター独自の共同研究、学際大規模情報基盤共同利用・共同研究拠点 (JHPCN) 課題および革新的ハイパフォーマンス・コンピューティング・インフラ (HPCI) 課題採択数の推移を示す。本センター独自の共同研究は恒常的に年 10 課題ほど実施されていることに加え、近年では JHPCN、HPCI を介した共同研究数が増加している。これは、センターの共同研究を通してユーザアプリケーションが高度化・大規模化し、JHPCN、HPCI 採択課題へとステップアップしており、我々の継続的な高速化支援活動が一定の成果を上げていると言える。

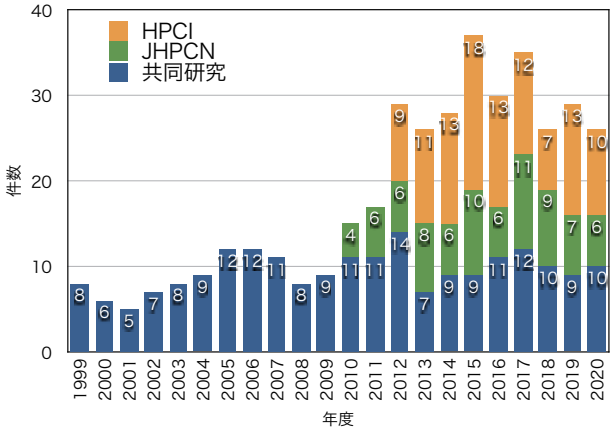


図 8　課題採択件数

5　おわりに

本稿では 2020 年 10 月に運用を開始した，サイバーサイエンスセンターのスーパーコンピュータ AOBA について紹介した。研究室のサーバでは実行できなかったプログラムやアイデアを実現する研究の強力なツールとして，最新鋭のスーパーコンピュータ AOBA をご活用いただければ幸いである。各システムの利用法の詳細，本センターからのお知らせ，問い合わせ，利用相談，高速化の依頼方法などついては本センターのウェブサイト[*1]を参照いただきたい。

[*1] https://www.ss.cc.tohoku.ac.jp/

［大学 ICT 推進協議会　2020 年度年次大会論文集より］

キャンパス無線 LAN と公衆無線 LAN の統合 – eduroam と Cityroam, OpenRoaming –

後藤英昭 [1)]，　原田寛之 [2)]，　漆谷重雄 [3)]

1) 東北大学 サイバーサイエンスセンター
2) 札幌学院大学 情報処理課
3) 国立情報学研究所

Integration of Campus Wireless LAN and Public Wireless LAN – eduroam, Cityroam, and OpenRoaming –

Hideaki Goto[1)], Hiroyuki Harada[2)], Shigeo Urushidani[3)]

1) Cyberscience Center, Tohoku University
2) Information Processing Division, Sapporo Gakuin University
3) National Institute of Informatics

概要

学生や生徒，教職員の学習・教育・研究環境を改善するために，社会全体の ICT 対応推進という側面も含めて，大学キャンパスのみならず市街地でも eduroam のサービスを提供したいという要求があり，各国が様々な枠組みを試しながらこれに対応してきた．しかしながら，現行の多くの公衆無線 LAN サービスにはローミングの仕組みがなく，eduroam の対応に多くの困難があった．一方，Wireless Broadband Alliance (WBA) によって提唱された次世代ホットスポット (NGH, Next Generation Hotspot) のコンセプトに沿う形で，公衆無線 LAN 向けのローミング基盤の構築が始まった．既報のとおり，著者らは eduroam を NGH 基盤に接続する実証実験システムを開発し，NGH の普及によって市街地の eduroam サービスの構築が容易になる可能性を示した．本報告では，WBA で開発が進められているローミング基盤である OpenRoaming を紹介し，eduroam の連携状況などの最新動向を示す．また，キャンパス内に公衆無線 LAN サービスを導入しようとする動きも出てきたことから，eduroam と公衆無線 LAN を統合する方向性を紹介，提案する．

1　はじめに

教育・研究機関向けの無線 LAN ローミング基盤である eduroam (エデュローム) は，執筆時点 (2020 年 9 月) で世界 107 か国 (地域)，国内 287 機関に導入されるに至っている [1]．初等・中等教育機関についても，海外では既にいくつかの国で導入が進み，社会全体の学習・教育環境の ICT 対応への貢献を目指して，日本国内でも募集が始まった [2]．学生や生徒，教職員の学習・教育・研究環境を改善するために，社会全体の ICT 対応推進という側面も含めて，大学キャンパスのみならず市街地でも eduroam のサービスを提供したいという要求がある．各国が様々な枠組みを試しながらこれに対応してきた．日本では世界的にも比較的早い時期の 2010 年に市街地サービスが始まったが，関東地域に限られるなど，十分とは言えなかった．欧州では，いくつかの都市において，大規模な展開が見られる．北欧諸国では，特に空港や鉄道駅などの公共施設での eduroam サービスが充実している．最近の特色ある例として，新型コロナウイルス禍における学生の学習環境確保のために，市街地に広く eduroam サービスを展開した，Research and Education Network for Uganda (RENU) の取り組みがある [3]．

現行の公衆無線 LAN サービスの多くにはローミングの仕組みがないことから，既に公衆無線 LAN の基地局がある場所でさえ，eduroam サービスの追加には技術面・経済面双方で多くの困難があった．一方，Wireless Broadband Alliance (WBA) によって提唱された次世代ホットスポット (NGH, Next Generation Hotspot) のコンセプトに沿う形で，公衆無線 LAN 向けのローミング基盤の構築が始まった．2019 年度年次大会 [2] で既報のとおり，著者らは eduroam

を NGH 基盤に接続する実証実験システムを開発し，NGH の普及によって市街地の eduroam サービスの構築が容易になる可能性を示した．WBA におけるローミング基盤は，2020 年に OpenRoaming として公表され，開発と展開が進められている [4]．本報告では，この OpenRoaming を紹介し，eduroam の連携状況などの最新動向を示す．また，キャンパス内に公衆無線 LAN サービスを導入しようとする動きも出てきたことから，eduroam と公衆無線 LAN を統合する方向性を紹介，提案する．

2　セキュア無線 LAN ローミング基盤 Cityroam

国内の公衆無線 LAN のセキュア化と NGH 導入を推進する目的で，第一著者が発起人となり，2017 年 1 月に「セキュア公衆無線 LAN ローミング研究会 (NGHSIG)」が発足した [5]．当研究会では，参加企業や開発者を増やしながら複数の通信事業者と協働で国内各地に Cityroam と呼ばれるセキュア公衆無線 LAN ローミング基盤の整備を進めている．この Cityroam と，ベースになった NGH テストベッド，及び，これらの開発背景・状況についての詳細は，文献 [6, 7] などで詳しく説明した．

現在主流のオープン Wi-Fi に基づくフリー Wi-Fi には，セキュリティ及び利便性の上で多くの問題があり，これらに対処するために次世代ホットスポット (NGH) が提唱されている．NGH は，無線 LAN ローミング環境において SSID (Service Set IDentifier) の自動選択と自動接続を実現する Passpoint 仕様 [8] に基づいており，Passpoint はまた，IEEE 802.1X [9] による安全な利用者認証に基づいている．Cityroam は，Passpoint/NGH をベースとして，複数の通信事業者に横断的な利用者認証の仕組みを提供する無線 LAN ローミング基盤で，安全で利便性の高い公衆無線 LAN の普及を目指している．ローミング基盤に参加する複数の通信事業者の団体は，ローミングフェデレーションと呼ばれることがあり，この観点で Cityroam はフェデレーションと呼べる．

図 1 に，Cityroam のウェブサイト [10] の様子を示す．Cityroam では，参加事業者が設置する無線 LAN 基地局で共通の SSID=“cityroam” が吹かれているのに加えて，Passpoint 用のビーコンも吹かれている．Cityroam の特徴の一つとして，eduroam との連携・統合が挙げられ，eduroam 利用者の便宜のために SSID=“eduroam” も併設されている．現在，主な参加

図 1　Cityroam ウェブサイト (2020 年 9 月).

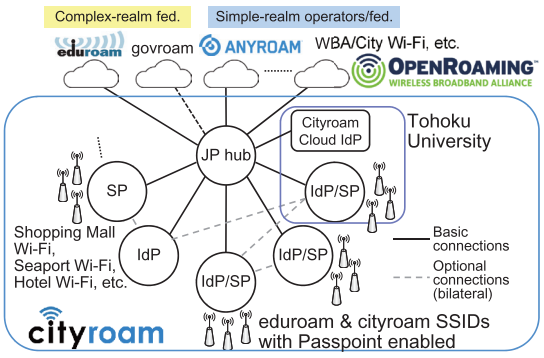


図 2　Cityroam の認証連携ネットワーク概略図.

事業者の拠点である京都市と長野市を中心として，国内各地の大規模ショッピングモールや飲食店，宿泊施設，公園，公共施設などに基地局が設置され，eduroam 及び Cityroam の両サービスが実現している．

図 2 に，Cityroam の認証連携ネットワークを示す．Cityroam に参加する無線 LAN サービスの事業者は，他の事業者の利用者を自社の基地局で受け入れ，ローミングサービスを展開できる．インターネットサービスプロバイダ (ISP) や各種ウェブサービスなど，無線 LAN サービスを自社では提供しないが，利用者アカウントを認証連携で提供できる事業者も，Cityroam に

参加することで，利用者に無線 LAN サービスを提供できるようになる．

本稿の執筆時点で，Cityroam には以下のようなアイデンティティプロバイダ (IdP) が接続されている [7]．

- eduroam
- ANYROAM
- Cityroam クラウド認証システム
- GlobalReach Odyssys Hotspot 2.0 Signup & Provisioning Service
  (一部の基地局のみで，デモ用に利用可能)
- 世界の携帯電話会社や ISP (実証実験中)
- WBA OpenRoaming (後述)

利用者は，いずれかのアカウントを持っていれば，国内各地の基地局で，Passpoint または 802.1X 認証により，安全かつ自動接続で無線 LAN が利用できる．

WBA では，NGH の技術開発と普及促進のために，2016 年から 3 年間，各年 1〜2 か月の期間で，City Wi-Fi Roaming と呼ばれるトライアルが開催された．このトライアルでは，世界各地の事業者や都市 Wi-Fi を結んで実証実験が行われた．著者らは，第二回となる 2017 年のトライアルに初参加，国際的な NGH 基盤との認証連携を実現した．翌 2018 年にもトライアルに参加し，認証連携システムの構築に技術的な支援を行い，共同実証実験を行った．トライアルの期間中は，世界の携帯電話会社や ISP とも認証連携を試すことができた．

3　WBA OpenRoaming

OpenRoaming [4] は，世界中の市民が利用できる，セキュアな公衆無線 LAN のためのローミング基盤である．まだ開発途中のため，流動的な仕様も残っている．

WBA では，NGH 実現のための仕様策定に留まらず，無線 LAN サービス事業者や携帯電話会社，ISP，都市 Wi-Fi，その他の様々なフリー Wi-Fi などを接続した，国際的なローミング基盤を実現しようとする構想があった．前述の City Wi-Fi Roaming トライアルは，準備的な位置付けにあったと考えられる．

公衆無線 LAN の国際的なローミングとしては，以前より，いくつかの事業者がブローカーとなって認証連携を実現する仕組みが存在している．いくつかの老舗はダイヤルアップ接続の時代からのローミング事業者である．国内の無線 LAN 事業者でも，国内他社や海外の事業者とローミングを実現している例がある．現在の方式では，訪問先ごとに異なる SSID を利用者が手作業で選択したり，あまり安全ではないウェブ認証の画面に ID・パスワードを入力する必要があるなど，利便性でもセキュリティの観点でも，問題が少なくない．NGH では，一つのアカウントを，場所や事業者に依らずにシームレスに利用できる環境の実現を目指している．

WBA などの広報によると，利用者は契約する事業者から発行された無線 LAN プロファイルを端末にインストールしておくだけで，安全かつ自動的に無線 LAN 接続が完了し，携帯電話並みの利便性が得られるとされる．また，携帯電話の SIM カードを用いた EAP-AKA (Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement, RFC 4187) やその改良版の EAP-AKA' (RFC 5448) にも対応しており，スマートフォンなどでは，プロファイルを別途導入することなく端末の初期状態のままで，もしくは，簡単な設定変更のみで，OpenRoaming 対応の公衆無線 LAN に接続できる．例えば，観光地がローミングに対応した公衆無線 LAN を提供していれば，そこを訪れた旅行者は，現地で情報をかき集めて無線 LAN 利用の登録をするといった煩わしさから開放され，その地に着いた時からすぐに無線 LAN を利用できるようになる．

4　Cityroam と eduroam の OpenRoaming 参加

2020 年春，WBA は通信事業者等に対して OpenRoaming への参加の呼びかけを行った．セキュア公衆無線 LAN ローミング研究会では，国内外の通信事業者とのローミングを実現しつつ，大規模ローミング基盤の実現・応用の研究のために，Cityroam を主体として OpenRoaming に参加することにした．また，eduroam の国際的な運用母体である GÉANT も OpenRoaming に参加した．2020 年 5 月 28 日に，WBA OpenRoaming のお披露目となるプレスリリースが出されたが，その中には初期サポートメンバーとして Cityroam 及び eduroam の名前も含まれている．

Cityroam は，Passpoint ベースのフリー Wi-Fi 向けとしては，おそらく世界初のローミングフェデレーションである．eduroam もフェデレーションであるが，市民一般向けの公衆無線 LAN の機能を大学などの参加機関でどのように提供するか，本稿の執筆時点でまだ検討中の段階である．OpenRoaming の基本機能は既に開発がある程度進んでいるが，大規模フェデ

レーションを相互接続するという観点での技術開発は継続中である．また，フリー Wi-Fi 向けの世界規模のローミングサービスを実現する上で，その効率的な運用方法や，運用規則の開発も必要である．研究会では，単に Cityroam を運用するだけではなく，運用で得られた知見を蓄積・提供し，技術開発を進めることで，世界規模の無線 LAN ローミング基盤の構築に貢献することを目指している．

eduroam の参加機関は，GÉANT または各国のシステムを通じて，OpenRoaming に参加できるようになる見込みである．また，国内の eduroam JP 参加機関は，Cityroam のシステムを介しても，OpenRoaming に参加可能である．以下に，実践的及び技術的な要点を解説する．

eduroam の参加機関が OpenRoaming に IdP として参加した場合，その構成員は自機関の eduroam アカウントを利用して，世界各地の OpenRoaming 対応基地局を利用できるようになる．ただし，OpenRoaming は eduroam のような一枚岩のシステムではなく，有償契約やフリー，特定用途向けなど，様々なカテゴリが存在するため，eduroam を受け入れているサイトのみでの利用に限定される．eduroam サービスを提供する方式には複数が考えられ，利便性の高い方から順に，以下のものが検討されている．

1. eduroam の SSID を併設する方式．一部に OpenRoaming の認証連携ネットワークが使われるが，従来の eduroam と同等のサービスが提供される．事業者は eduroam の SP として登録が必要．利用者側の設定変更は不要．(Cityroam はこの方式)
2. eduroam 用の RCOI (Roaming Consortium Organization Idenfitier) を基地局に追加する方法．事業者は eduroam の SP として登録が必要．利用者端末に Passpoint プロファイルの追加導入が必要 (eduroam Configuration Assistant Tool (CAT) などで対応可能)．
3. OpenRoaming 用の RCOI を利用する方法．事業者は eduroam の SP に準じたプライバシポリシーなどを守る必要があり，どのようなローミング契約とするか，現在検討中．利用者端末に Passpoint プロファイルの追加導入が必要．eduroam に非対応のサイトでは，端末が接続を試みた上で認証失敗となるので，利便性が低い問題がある．

OpenRoaming 上で eduroam アカウントを有効にしたい機関は，RadSec で利用される NAPTR (Naming Authority Pointer) レコードを自機関の DNS (Domain Name System) サーバに追加する必要がある．執筆時点で，Cityroam は OpenRoaming に接続済みであり，eduroam JP の参加機関もトライアルが可能である．

反対に，eduroam の参加機関が Cityroam または OpenRoaming を介して教育研究系以外の市民一般を受け入れ，公衆無線 LAN サービスを展開する場合は，Passpoint に対応したアクセスポイントを導入し，各ローミング基盤の仕様に従った設定と運用を行う必要がある．

5 eduroam と公衆無線 LAN の統合

5.1 公衆無線 LAN の学校・大学構内への導入

大学では，学会の大会や，セミナー・研修，研究打ち合わせなど，大小様々なイベントにおいて，eduroam のアカウントを所有していない訪問者に対して，キャンパス無線 LAN を利用するためのビジターアカウントを発行することが一般的に行われている．この場合，eduroam 用の一時利用アカウントを発行したり，自校独自の無線 LAN システムのアカウントを発行したり，その形態は様々である．eduroam 用のビジターアカウントを発行する手段としては，自校の eduroam IdP を利用する方法に加えて，国内では「eduroam JP 認証連携 ID サービス [11]」のビジター用アカウント発行機能を用いる方法，海外では eduroam Visitor Access (eVA) [12] を用いる方法などがある．訪問者があるたびに人手でアカウントを発行することは，手間が大きいことから，できるだけ行わないようにしたいという要求がある．米国の eduroam の運用機関である ANYROAM [13] では，携帯電話の Short Message Service (SMS) を利用してアカウントを発行するサービスがある．ANYROAM のアカウントは，すべての eduroam 参加機関で利用できるものではなく，ANYROAM を受け入れるように登録した機関のみで利用可能である．

もし，eduroam 参加機関からの訪問者に限らず，民間企業の研究者はもちろん，市民一般を受け入れられるような無線 LAN ローミングシステムがあれば，上述のような課題に対処できると考えられる．これまでの市街地における eduroam サービスとは反対に，学校等の構内に公衆無線 LAN を導入することを考える．利用者の故意あるいは無意識の不正利用に対して，機関が余計な責任を負うことがないようにするため，利用者認証があり，不正利用時の利用者追跡も可能な，

セキュアな公衆無線 LAN システムを導入することが望ましい．

2018 年度年次大会において，「キャンパス無線 LAN の市民への開放」という観点を紹介し，実例として札幌学院大学における Cityroam の運用について述べた [14]．大学を含め，学校には地域社会への貢献が常に求められており，無線 LAN サービスの提供は，市民の交流や学習支援，高大連携，産学連携などにおける ICT 活用環境の充実に寄与するものと考えられる．

総務省によれば，全国の約 9 割の学校が避難所に指定されており，被災時の無線 LAN サービス提供の重要性が指摘されている [15, 16]．被災時の通信手段の確保は重要であり，携帯電話網に加えて，大容量で無償の公衆無線 LAN サービスの重要性は高い．一方で，避難所における無線 LAN のセキュリティ問題・プライバシ問題も指摘されており，安全なシステムを導入する必要がある．大規模災害時は，大学キャンパスも避難所になることがあるが，その時その場でキャンパス無線 LAN の開放を決断，実施することは必ずしも容易ではない．東日本大震災では，「普段使いのシステムでなければ緊急時の活用が難しい」ことがよく指摘されていた．すなわち，日常的に利用可能なシステムであることが重要である．

以上のように，公衆無線 LAN システムのセキュア化とローミング対応が進むことにより，eduroam の市街地展開が容易になる一方で，学校には市民向けの安全な公衆無線 LAN システムが求められるようになってきている．これらを総合すると，「eduroam と公衆無線 LAN の統合」という方向性が考えられる．言い換えると，学校においても市街地においても，両対応の基地局システムを導入すればよいことになる．基地局システムが統一ないし類似したものとなることで，仕様や運用の簡素化が期待される．このような見方は，世界の eduroam コミュニティでも新しいものであるが，既に一部で議論が始まっている．eduroam は，現在は IdP と SP を含むローミングフェデレーションとして定義されているが，もし SP 側の統合が進めば，主に IdP を集めたフェデレーションとしての色が濃くなり，価値観が若干変わってくる可能性もある．

5.2　技術面・運用面の検討

既に eduroam を導入して訪問者を受け入れている機関であっても，非学術系の市民を受け入れることには，抵抗があるかもしれない．障壁として考えられるものの一つとして，機関のセキュリティポリシーやネットワーク利用ポリシーによる制約が考えられる．eduroam JP では，当初から，訪問者用のネットワークを学外ネットワーク扱いとして分離することを推奨してきた．もしネットワークがこのような構成になっていれば，若干のポリシー変更で市民の受け入れにも対応できるものと思われる．現在のポリシーが，ICT を活用した新しい教育・研究環境や BYOD (Bring Your Own Device) への対応，ICT 活用社会への貢献などの観点で，現代に相応しいものとなっているかどうかを確認し，必要に応じて修正することが望ましい．

もし，何らかの理由でポリシー変更による対応が不可能な場合は，キャンパス無線 LAN システムを自前で運用するのではなく，いわゆる「マネージド Wi-Fi」としてアウトソーシングすることで，機関のポリシーの適用対象から外すことも考えられる．

大学共同利用施設や，学会施設，会議施設など，構成員の常用が少なく，主に訪問者の一時的な利用のために設置する無線 LAN システムならば，さらに簡易な導入方法も考えられる．例えば，eduroam・公衆無線 LAN 共用の基地局を組み込んだ自動販売機があり，プレスリリース [17] にあるような公園に限らず，様々な場所に導入可能である．

基地局のネットワークを学外扱いにすることで，そのままでは構成員の学内サーバへのアクセスも学外扱いになる問題が生じるが，これは認証 VLAN を導入して，学内サーバへのバイパス経路を設けることによって解決できる．

訪問者を受け入れる場合，不正利用が懸念事項として挙げられるかもしれない．この点については，eduroam と同様であり，不正利用時の利用者追跡が可能な公衆無線 LAN システムを導入することによって，SP 側に無用な責任が生じることを回避する．Cityroam や OpenRoaming では，利用者追跡が可能なアカウントを発行する，信頼できる IdP のみを受け入れることができる．

キャンパス無線 LAN では，構成員の利用が最優先であり，もし訪問者の通信によってその利用に支障が出るようでは，本末転倒と言える．訪問者による帯域圧迫が懸念される場合は，訪問者の通信のみを帯域制限 (スロットリング) することも考えられる．

6　むすび

市民向けの国際無線 LAN ローミング基盤である OpenRoaming を紹介し，eduroam の連携状況などの最新動向を示した．国内のローミング基盤 Cityroam は，eduroam が統合されていることに特徴があり，国

内の様々な場所で eduroam/Cityroam 双方のサービスを展開している．eduroam と Cityroam は 2020 年 5 月に OpenRoaming に参加し，国内外の公衆無線 LAN において eduroam サービスを実現する基盤の構築が進められている．

一方，地域社会への貢献や災害対応などの観点で，学校への公衆無線 LAN の導入も望まれている．本稿では，eduroam と公衆無線 LAN を統合する方向性を紹介した．今後，国内外の通信事業者とのローミングを充実させ，利用できるアカウントとサービスエリアの拡充を図っていく予定である．

本研究の一部は，平成 31 年度，令和 2 年度国立情報学研究所公募型共同研究の助成を受けた．

参考文献

[1] eduroam JP: https://www.eduroam.jp/
(2020 年 9 月 7 日参照)

[2] 後藤英昭, 原田寛之, 中村素典, “キャンパス無線 eduroam と次世代ホットスポットの最新動向,” 大学 ICT 推進協議会 2019 年度年次大会 論文集 TH2-2, 2019.

[3] RENU PRESS RELEASE “eduroam now accessible off-campus!” https://renu.ac.ug/assets/docs/eduroam-press-release.pdf
(2020 年 9 月 7 日参照)

[4] WBA OpenRoaming: https://wballiance.com/openroaming/
(2020 年 9 月 7 日参照)

[5] セキュア公衆無線 LAN ローミング研究会 (NGH-SIG): https://nghsig.jp/
(2020 年 9 月 7 日参照)

[6] 後藤英昭, “次世代ホットスポット (NGH) の世界動向と NGH 対応 eduroam システムの開発,” 信学技報 IA2017–61/IN2017–60, pp.49–54, 2017.

[7] 後藤英昭, “安全で利便性の高い公衆無線 LAN を提供する次世代ホットスポット基盤 Cityroam” 東北大学サイバーサイエンスセンター 大規模科学計算システム広報 SENAC Vol.51, No.3, pp.16–19, 2018.

[8] Wi-Fi Alliance, “Passpoint – Wi-Fi ホットスポットネットワークへのシームレスでセキュアな接続を実現.” https://www.wi-fi.org/ja/discover-wi-fi/passpoint/
(2020 年 9 月 7 日参照)

[9] IEEE Std 802.1X-2010, “Port-Based Network Access Control.”

[10] Cityroam: https://cityroam.jp/
(2020 年 9 月 7 日参照)

[11] eduroam JP 認証連携 ID サービス: https://federated-id.eduroam.jp/
(2020 年 9 月 7 日参照)

[12] eduroam Visitor Access: https://www.surf.nl/en/eduroam-easy-access-to-wireless-networks/eduroam-visitor-access
(2020 年 9 月 7 日参照)

[13] ANYROAM: https://www.anyroam.net/
(2020 年 9 月 7 日参照)

[14] 原田寛之, 後藤英昭, “学術無線 LAN ローミング基盤 eduroam と次世代ホットスポット基盤 Cityroam のキャンパスへの展開,” 大学 ICT 推進協議会 2018 年度年次大会 論文集 MA1-5, 2018.

[15] 総務省, “2020 年に向け全国約 3 万箇所の Wi-Fi 整備を目指して (PDF).” https://www.soumu.go.jp/main_content/000556749.pdf
(2020 年 9 月 7 日参照)

[16] 総務省, “防災等に資する Wi-Fi 環境の整備計画 (PDF).” https://www.soumu.go.jp/main_content/000669467.pdf
(2020 年 9 月 7 日参照)

[17] コカ・コーラ ボトラーズジャパン株式会社, “国内初!! 国際的学術無線 LAN「eduroam」対応自動販売機を京都市の都市公園に設置.” https://www.ccbji.co.jp/news/detail.php?id=770
(2020 年 9 月 7 日参照)

［報　告］

「HPCwireJapan」にセンターの情報が掲載されました

高度情報科学技術研究機構(RIST)神戸センター・小柳義夫サイエンス・アドバイザーが我が国のコンピュータの開発史を発信している「HPCwireJapan」で本センターSENAC-1 等の情報が紹介されました。

詳細は、以下をご覧ください。

https://www.hpcwire.jp/archives/36466
https://www.hpcwire.jp/archives/36916

後藤准教授らの研究グループが国内初の OpenRoaming 基盤を開発し、サービスを開始

本センター・後藤英昭准教授らの研究グループおよび「セキュア公衆無線 LAN ローミング研究会」(同教員が開設、幹事)では、安全で利便性の高い次世代公衆無線 LAN の研究開発や啓発活動のために、認証連携基盤の開発を進め、通信事業者との協働によりセキュア無線 LAN ローミング基盤「Cityroam」を開設、運用しています。今回、この基盤を WirelessBroadband Alliance(WBA)による「OpenRoaming」に接続、国内初となる OpenRoaming 基盤を開発し、サービスを開始しました。国際的にも、OpenRoaming の広域サービス展開は先行事例の一つであり、WBA における技術・運用の開発に貢献しています。

詳細については、以下に掲載されています。

https://www.tohoku.ac.jp/japanese/2020/11/press20201125-03-openroaming.html

— SENAC 執筆要項 —

１．お寄せいただきたい投稿内容

サイバーサイエンスセンターでは、研究者・技術者・学生等の方々からの原稿を募集しております。以下の内容で募集しておりますので、皆さまのご投稿をお待ちしております。なお、一般投稿いただいた方には、謝礼として負担金の一部を免除いたします。

・一般利用者の方々が関心をもたれる事項に関する論説
・センターの計算機を利用して行った研究論文の概要
・プログラミングの実例と解説
・センターに対する意見、要望
・利用者相互の情報交換

２．執筆にあたってご注意いただく事項

(1)原稿は横書きです。
(2)術語以外は、「常用漢字」を用い、かなは「現代かなづかい」を用いるものとします。
(3)学術あるいは技術に関する原稿の場合、200 字～400 字程度のアブストラクトをつけてください。
(4)参考文献は通し番号を付し末尾に一括記載し、本文中の該当箇所に引用番号を記入ください。
　・雑誌：著者, タイトル, 雑誌名, 巻, 号, ページ, 発行年
　・書籍：著者, 書名, ページ, 発行所, 発行年

３．原稿の提出方法

原稿のファイル形式は Word を標準としますが、PDF での提出も可能です。サイズ*は以下を参照してください。ファイルは電子メールで提出してください。

－Word の場合－
　・用紙サイズ：A4
　・余白：上＝30㎜　下＝25㎜　左右＝25㎜　綴じ代＝0
　・標準の文字数（45 文字 47 行）

＜文字サイズ等の目安＞
　・表題＝ゴシック体 14pt 中央　・副題＝明朝体 12pt 中央
　・氏名＝明朝体 10.5pt 中央
　・所属＝明朝体 10.5pt 中央
　・本文＝明朝体 10.5pt
　・章・見出し番号＝ゴシック体 11pt～12pt
　　*余白サイズ、文字数、文字サイズは目安とお考えください。

４．その他

(1)執筆者には希望により本誌 PDF 版を進呈します。
(2)一般投稿を頂いた方には謝礼として、負担金の一部を免除いたします。免除額は概ね 1 ページ 1 万円を目安とします。詳細は共同利用支援係までお問い合わせください。
(3)投稿予定の原稿が 15 ページを超す場合は共同利用支援係まで前もってご連絡ください。
(4)初回の校正は、執筆者が行って、誤植の防止をはかるものとします。
(5)原稿の提出先は次のとおりです。

東北大学サイバーサイエンスセンター内　情報部情報基盤課共同利用支援係
e-mail　uketuke@cc.tohoku.ac.jp
TEL　　022-795-3406

スタッフ便り

2 年前にちょっとしたきっかけでトレッキングを始めました。山の中の緑、森林限界を超えた尾根での風、そして遠くまで見渡せる景色に魅了されました。昨年夏には、山小屋デビューと 2000m 級の山を体験するため鳥海山へ行く予定でした。しかし、新型コロナウイルスの第 2 波により山小屋は閉鎖、山行も中止することにしました。秋になり紅葉のシーズンには第 2 波も収束、仙台から日帰りできる栗駒山、熊野岳、月山を登り、初雪が降り始めてから面白山に登りました。面白山は宮城県と山形県の県境にある 1,264m の山です。仙山線面白山高原駅から紅葉川の渓谷にそって中面白山を超え、約 3 時間、面白山の頂上に立ちました。そこに待っていたのは、雲一つない青空と、鳥海山、月山、朝日連峰、飯豊連峰、蔵王連峰の雪をかぶった勇姿。しばらくその姿に見とれてしまいました。2021 年は新型コロナウイルスの感染も収まり、東北の山々を巡れるようになればと願っています。(A.M)

今更ですが、コロナ禍で、昨年すべての小中学校が 3 月初旬から 4 月上旬まで休校になり感じたことが学校給食のありがたみでした。私には子供が 3 人(中学生 2、小学生 1)いますが、ステイホーム中、妻曰く、よく食べるので、食費がかかって、かかってしょうがないということでした。特に、昼食は通常なら学校給食ですので、だいたい一人の給食費が 6 千円前後、3 人で 2 万円もかからないというのに、ステイホーム期間中は、お菓子だ、ジュースだ、アイスだと学校給食ではでるはずもないものがプラスされるので、昼食のエンゲル係数が半端ではなかったというのが現実でした。今般のコロナ禍を経験し、本当に学校給食の安さには、感謝しないといけないとつくづく感じたところです。ちなみに、仙台市立の小中学校では、給食 1 食の単価が小学校で約 290 円、中学校では約 345 円で、美味しさはさておき、子供の成長に必要な栄養が取れるように、学校栄養士さんがバランスのいい食事になるように毎日献立を考えてくれています。学校栄養士さんと学校給食法を立法してくださった国会議員の皆様方に、改めて感謝しなければならないと思う次第です。第 3 波が来ているという現時のコロナ禍で、家族及び職場以外の人との接触を避けつつ、日々、外出は、職場の往復と近所への買い物という静かな生活を営んでおります。ここサイバーサイエンスセンターには昨年 4 月に着任して以来、前任の理学部での送別会も「コロナが終息したらやりましょう。」といわれて送別されて、もう少しで、1 年が経ってしまうというこの状況。果たして、この状況がいつまで続くことやらと思いつつ、毎朝、体温を計測して日々の生活を送っています。(K.H)

令和 3 年 1 月発行
編集・発行　東北大学
サイバーサイエンスセンター
仙台市青葉区荒巻字青葉 6-3
郵便番号　980-8578
PDF 作成　株式会社　東誠社

スーパーコンピュータ AOBA システム一覧

計算機システム	機　種
サブシステム AOBA-A	SX-Aurora TSUBASA
サブシステム AOBA-B	LX 406Rz-2

サーバとホスト名

ログインサーバ	login.cc.tohoku.ac.jp
データ転送サーバ	file.cc.tohoku.ac.jp

サービス時間

利用システム名等	利用時間帯
サブシステム AOBA-A	連　続　運　転
サブシステム AOBA-B	連　続　運　転
サ ー バ	連　続　運　転
館内利用	平日 8:30～21:00

サブシステム AOBA-A の利用形態と制限値

<table>
<tr><th>利用形態</th><th>キュー名</th><th>VE 数※</th><th>実行形態</th><th>最大経過時間
既定値/最大値</th><th>メモリサイズ</th></tr>
<tr><td>無料</td><td>sxf</td><td>1</td><td>1VE</td><td>1 時間/1 時間</td><td rowspan="5">48GB×VE 数</td></tr>
<tr><td rowspan="3">共有</td><td rowspan="2">sx</td><td>1</td><td>1VE</td><td rowspan="3">72 時間/720 時間</td></tr>
<tr><td>2～256</td><td>8VE 単位で確保
（VH を共用しない）</td></tr>
<tr><td>sxmix</td><td>2～8</td><td>1VE 単位で確保
（VH を共用する）</td></tr>
<tr><td>占有</td><td colspan="4">個別設定</td></tr>
</table>

※2VE以上を利用した並列実行にはMPIの利用が必用

サブシステム AOBA-B の利用形態と制限値

<table>
<tr><th>利用形態</th><th>キュー名</th><th>ノード数※</th><th>最大経過時間
既定値/最大値</th><th>メモリサイズ</th></tr>
<tr><td>共有</td><td>lx</td><td>1～16</td><td>72 時間/720 時間</td><td rowspan="2">256GB×ノード数</td></tr>
<tr><td>占有</td><td colspan="3">個別設定</td></tr>
</table>

※2ノード以上を利用した並列実行にはMPIの利用が必用

目次

東北大学サイバーサイエンスセンター
大規模科学計算システム広報　Vol.54 No.1 2021–1

［巻頭言］
新スーパーコンピュータ AOBA 始動 ……滝沢　寛之　1

［大規模科学計算システム］
SX-Aurora TSUBASA でのプログラミング（並列化編）……林　康晴　3

［共同研究成果］
ナノアンテナの数値解析のためのブロックモーメント法の高速化
……今野　佳祐・陳　強　25

金蒸気－金ナノ粒子変換を伴う高エンタルピー水プラズマジェットの
数値シミュレーション ……茂田　正哉　32

格子ガス法流体解析モデルとニューラルネットワークの融合 ……松岡　浩　39

［大学 ICT 推進協議会　2020 年度年次大会論文集より］
東北大学サイバーサイエンスセンター
スーパーコンピュータ AOBA の紹介 ……山下　毅・森谷　友映・佐々木大輔　齋藤　敦子・小野　敏・大泉　健治　滝沢　寛之　50

キャンパス無線 LAN と公衆無線 LAN の統合
……後藤　英昭・原田　寛之・漆谷　重雄　56

［報　告］
「HPCwireJapan」にセンターの情報が掲載されました …… 62

後藤准教授らの研究グループが
国内初の OpenRoaming 基盤を開発し、サービスを開始…… 62

執筆要項 …… 63

スタッフ便り …… 64