

—Original Paper—

A Performance Optimization of the Large-Scale Seismic Simulation on the Earth Simulator (ES2)

Yuichi Hirokawa^{1*}, Toshiyuki Masatsuki², Noriaki Nishikawa¹, Misako Iwasawa¹, and Toshiyuki Asano¹

The understanding and prediction of influences caused by seismic wave ranges from low to high frequency is important to mitigate the disaster of earthquakes by multilateral preparations. The structure of undergrounds causes the complex behavior of seismic wave propagation. The numerical simulation is an effective tool to predict the wave propagation. For metropolises on a large plain such as Kanto basin, however, the simulation requires enormous computing resources as well as techniques of high-performance computing. In this paper, we have developed the optimization method for the seismic simulation code on the Earth Simulator (ES2). We assume the finite difference method code “k-fdm3d” developed by KOZO KEKAKU ENGINEERING Inc.. Because the code adopt a stencil computation with the structured grid, we have optimized the memory-access and communication of the computation. For the reasons, the optimization method has applicability to simulation code on the other fields. The method enables a 570 million grids seismic simulation on ES2. The method also allows the simulation to achieve approximately 30% to the peak performance of ES2. For the results, we have established the optimization method to calculate the high-accuracy prediction of seismic wave propagation applicable for a large plain.

Keywords : Seismic wave, Simulation, The Earth Simulator, Parallelization, Optimization

Received 9 March 2016 ; Revised 28 April 2016 ; Accepted 9 May 2016

- 1 Center for Earth Information Science and Technology (CEIST), Japan Agency for Marine-Earth Science and Technology (JAMSTEC)
- 2 KOZO KEIKAKU ENGINEERING Inc.

*Corresponding author:

Yuichi Hirokawa

Center for Earth Information Science and Technology, Japan Agency for Marine-Earth Science and Technology
3173-25 Showa-machi, Kanazawa-ku, Yokohama, Kanagawa 236-0001, Japan
y-hiroka@jamstec.go.jp

Copyright by Japan Agency for Marine-Earth Science and Technology

— 原著論文 —

地球シミュレータ (ES2) における 大規模地震動シミュレーションの高速化

廣川 雄一^{1*}, 正月 俊行², 西川 憲明¹, 岩沢 美佐子¹, 浅野 俊幸¹

地震に対する防災・減災では、低～高周波数領域の地震動を事前に予測し、対策を立てておくことが重要である。地震動は地下構造の影響を受けながら複雑に伝播をしていく。この地震動の予測には数値シミュレーションが有効な手段であるが、都市が集中する大規模平野部などを対象とした解析では大規模並列計算が必要となる。そこで、本研究では地震動を解析する有限差分法コード（株式会社 構造計画研究所のk-fdm3d）の地球シミュレータ（ES2）向け高速化手法を開発した。本高速化手法は構造格子のステンシル計算を対象としたものであり、メモリアクセスと通信の効率化に着目し、高速化を行った。本高速化手法は地震動だけではなく、他の分野のコードにも応用することができる。ES2を用いた性能評価では570億点規模の大規模並列計算が実行可能であり、3割のピーク性能比を達成可能なことが確認できた。また、本研究の高速化手法により、大規模平野部などを対象とした低～高周波数領域の地震動を高速に計算する手法を確立することができた。

キーワード: 地震動, シミュレーション, 地球シミュレータ, 並列化, 高速化

2016年3月9日受領；2016年4月28日改訂稿受理；2016年5月9日受理

- 1 国立研究開発法人海洋研究開発機構 地球情報基盤センター
- 2 株式会社 構造計画研究所

*代表執筆者：

廣川 雄一

国立研究開発法人海洋研究開発機構 地球情報基盤センター

〒236-0001 神奈川県横浜市金沢区昭和町3173-25

y-hiroka@jamstec.go.jp

著作権：国立研究開発法人海洋研究開発機構

1. はじめに

地震に対する防災・減災では、想定地震での被害を予測し、事前に対策を立てておくことが重要である。地上付近にある構造物を評価する場合には、基礎的なデータとして、地震時に発生する波（以下、地震波とする）の影響を詳細に調べておく必要がある。地震波は地下構造などの影響を受け、反射や散乱をしながら複雑に伝播していく。この時間的・空間的に変動する地震波の把握には、非定常の数値シミュレーションが有効である。固有周期が2～20[s]の高層ビルなど長周期の構造物が対象であれば、大まかな目安として約0.5[Hz]までの周波数領域の地震動をシミュレーションする。一方、低～中層階ビルなど固有周期が2[s]以下の短周期の構造物では、0.5[Hz]以上の高い周波数領域までの地震動シミュレーションが必要となる。数値シミュレーションにおいて、高周波数領域の地震波を解析するためには、計算格子および時間分解能を増やす必要がある（田島ほか、2012）。このため、高周波数領域では、統計的グリーン関数法などの半経験的なモデルを併用して解析をするのが一般的である（司ほか、2009）。

近年、スーパーコンピュータを用いた地震動シミュレーションコード（Seism3D3）の高速化手法が検討されており、半経験的なモデルを用いずに高周波数領域まで解析ができていく（Furumura and Chen, 2005; Furumura and Saito, 2009）。また、高い計算性能が出ており、現実的な時間内で計算結果を得ることが可能である（古村, 2009a; 古村, 2010）。

本研究では都市が集中する大規模平野などを対象として、1～2[Hz]までの高周波数領域の地震動伝播を地球シミュレータ（以下、ES2とする）上で高速にシミュレーションする手法を開発した。シミュレーションコードとして株式会社構造計画研究所が保有する三次元波動伝播プログラム（k-fdm3d）を用い、ES2のアーキテクチャを考慮した高速化手法を開発し、性能評価を行った。また、本研究ではコードの保守性を高めるため、変更を最小限に留める高速化手法を開発した。本手法では3章で示すようにコンパイラ指示行の挿入および部分的なコードの書き換えのみで対応が可能である。

本論文では、2章で大規模地震動シミュレーションの概要、3章で高速化手法、4章で並列性能について述べる。また、5章で考察、6章でまとめを示す。

2. 大規模地震動シミュレーション

本章では大規模地震動シミュレーションの概要を示す。2.1節ではシミュレーション手法の概要、2.2節ではES2の概要を示す。

2.1 三次元波動伝播プログラム（k-fdm3d）の概要

本節では三次元波動伝播プログラム（k-fdm3d）の計算手法を示す。支配方程式として、式（1）の等方な線形弾性体を仮定したNavierの式を用いる。

$$\rho \ddot{\mathbf{u}} = (\lambda + \mu) \nabla (\nabla \cdot \mathbf{u}) + \mu \Delta \mathbf{u} + \mathbf{F} \quad (1)$$

ここで、 ρ は密度、 $\mathbf{F}$ は体積力、 λ と μ は弾性などに関わるラメ定数、 $\ddot{\mathbf{u}}$ は粒子（局所領域）の加速度である。なお、 ∇ はナブラ演算子、 Δ はラプラス演算子、上付きのドット記号は時間方向の偏微分階数を表す。 $\ddot{\mathbf{u}}$ の時間発展を解くことにより、非定常な地震動伝播を解析することができる。数値シミュレーションでは空間4次精度の有限差分法を用い、陽的に $\ddot{\mathbf{u}}$ の時間発展を計算する。計算格子はデカルト座標系で、 $\ddot{\mathbf{u}}$ とその他の物理量をそれぞれ半格子点分ずらした位置で観測するStaggered格子を用いた。また、時間は2次精度で、半タイムステップずらした物理量で計算を行うVerlet Leap Frog法を適用した（古村, 2009b）。データ構造には高速に処理ができる構造格子を採用し、各物理量を倍精度または単精度の浮動小数点数型変数として扱った。プログラミング言語はFortran 90を用い、通信ライブラリにはMPIを用いた。

2.2 地球シミュレータ（ES2）の概要

本節ではシミュレーションを実行するES2の概要を示す。図1にES2の構成を示す。ES2はNEC SX9/E（以下、ノードとする）160台を128 GByte/sのネットワークで接続したベクトル型のスーパーコンピュータである（JAMSTEC, 2008）。

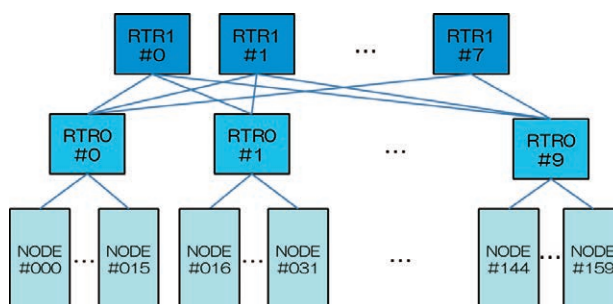


Fig. 1. The configuration of the Earth Simulator (ES2).

図1. 地球シミュレータ（ES2）の概要

ネットワークは2段のFat-Treeで、ノード間は1～3個のスイッチRTR0～1を経由して通信する。

図2に1ノードの構成を示す（板倉，2012）。1ノードには8個のCPUと128 GByteのメモリを搭載している。容量8 KByteのメモリバンク64個を1つのメインメモリユニット（MMU）にまとめ、16個のスイッチ（RTR）で256個のMMUを各CPUと接続している。1個のCPUには256回の倍精度浮動小数点数型演算を1度に実行できるベクトル処理部があり、ピーク性能は102.4 GFLOPSである。また、16 Byteを1ブロックとした16,384 wayのメモリアンタリーブにより、メモリを16,384個のバンクに分け、16,384バンクブロックを並列にLoad/Storeすることで2.5 Byte/FLOPを確保している。各CPUはメモリとの間にAssignable Data Buffer（以下、ADBとする）と呼ばれる容量256 KByte、4 Byte/FLOPの高速なキャッシュを備えている（Nakazato et al., 2008）。ADBはユーザーが格納もしくはバイパスする変数を指定可能なLeast Recently Used型のライトスルーキャッシュであり、2回以上参照される変数のLoadを高速化することができる（佐藤ほか，2009）。

3. ステンシル計算の高速化

本章では有限差分法などで現れる、計算時に隣接格子点を参照するステンシル計算のES2向け高速化手法を示す。k-fdm3dではステンシル計算の負荷は計算コード全体の約5割を占めており、実行速度に大きく影響する。残りの約5割は各物理量の時間発展計算や境界条件処理などである。計算コード全体での通信割合は条件によって変動するが、通常は1割程度である。本手法では3段階の手順を踏む。まず、3.1節で示すような配列寸法変更を行う。次に3.2～3.4節で示す高速化を行う。最後に3.5節で示す隣接間通信の高速化を行う。

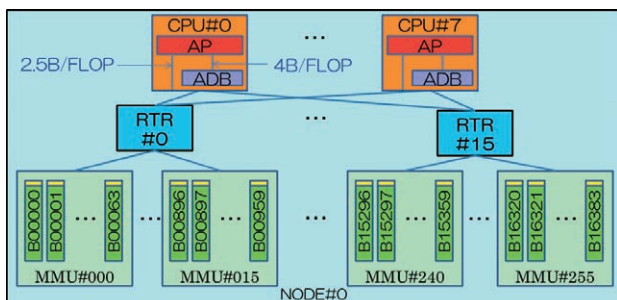


Fig. 2. The architecture of the NEC SX9/E.

図2. 1ノードの構成

3.1 配列寸法の変更

3.1.1 バンクコンフリクトの発生要因

Fortran 90で多次元配列を確保する際は、図3の様に左側の添字から順に一次元化を行い、メモリ上に配置する。

ES2では図4に示す様に、1行に16 Byte × 16,384の変数（8 Byteの倍精度は32,768個、4 Byteの単精度は65,536個）を左側から順に複数バンクに跨って並べていく。次の行以降も同様な方法で変数を配置していく。また、ES2では16 Byte（倍精度は2個、単精度は4個）を1ブロックとしてバンク単位でLoad/Storeを行う。実際の計算では1度で最大256回の演算を行うために、最大256個の変数をまとめてLoad/Storeする。図4における四角は倍精度の変数1個を表し、斜線はメモリからのLoad/Store、塗りつぶしは未アクセス領域、白抜きはパディング領域を表す。また、IDIMは多次元配列の1次元目寸法であり、 i, j, k は配列の添字を表す。

多次元配列の1次元目方向で倍精度の変数2個を2回Load/Storeする場合、図4（左上）のように複数バンクBank00000～00001にLoad/Storeが分散される。

2次元目方向で倍精度の変数2個を2回Load/Storeする場合は、図4（左中）のように特定のバンクBank00000にLoad/Storeが集中する“バンクコンフリクト”が発生する可能性がある。バンクコンフリクトが発生したバンクへのLoad/Storeは、先に実行中のLoad/Storeが完了するまで待ち状態となる。

3次元目方向で倍精度の変数2個を2回Load/Storeする場合にも図4（左下）のようなバンクコンフリクトが発生する可能性がある。

3.1.2 バンクコンフリクトの回避策

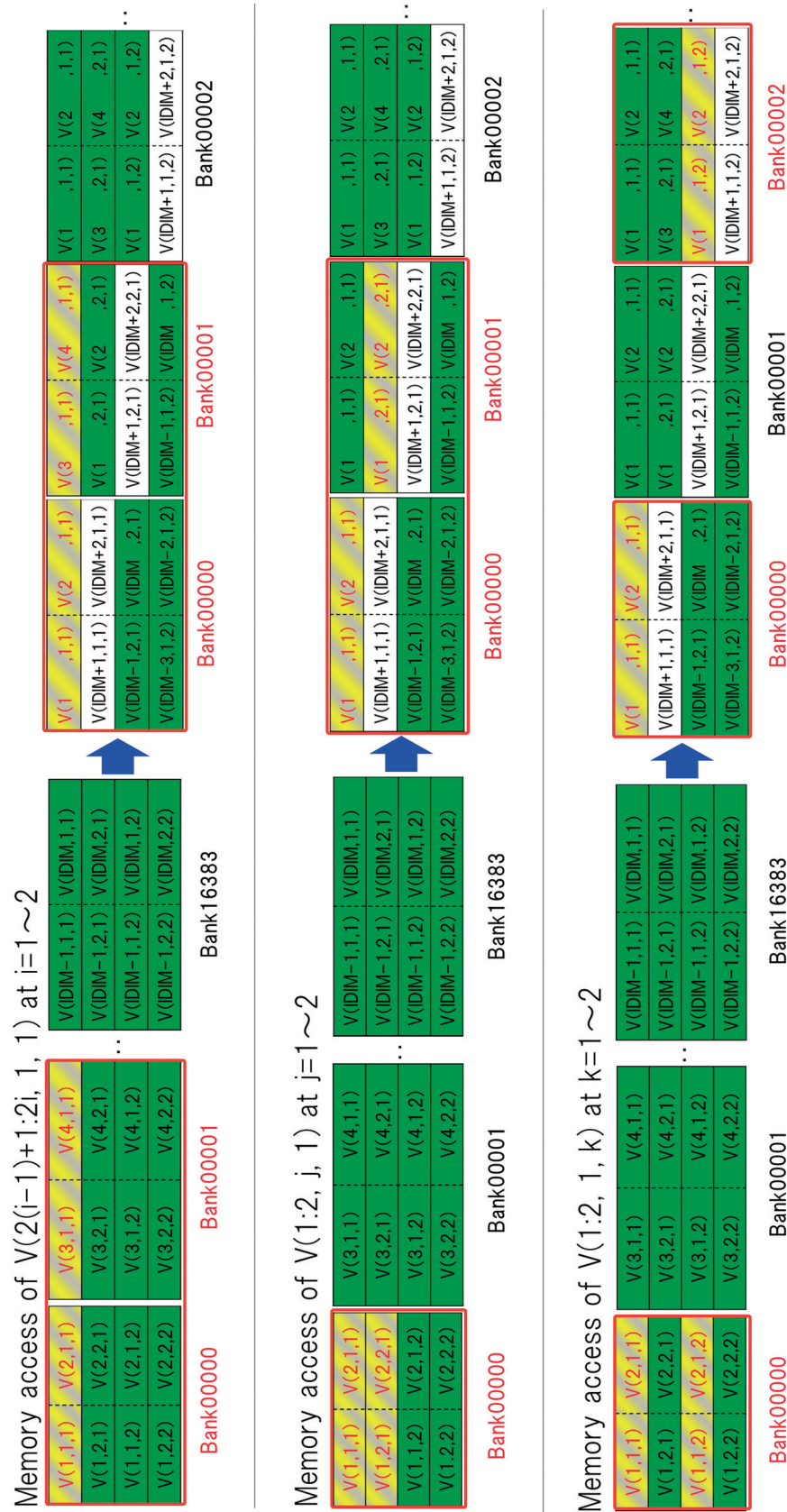
バンクコンフリクトの回避方法としては、配列寸法の奇数化が一般的である。本研究では更に性能を向上させるため、配列寸法の改良を行った。本高速化手法では多次元配列の1次元目寸法のみを変更する。また、配列寸法の変更はパディングでよく、拡張した部分は計算に使用しなくてよい。まず、ES2では16 Byteを1ブロックとしてバンク単位でLoad/Storeするため、式（2）に従って

Memory mapping of V(3,2)

V(1,1)	V(2,1)	V(3,1)	V(1,2)	V(2,2)	V(3,2)
--------	--------	--------	--------	--------	--------

Fig. 3. The memory mapping of an array in Fortran 90.

図3. メモリへの配置

Fig. 4. The memory access. left: original, right: optimized / upper: i -direction, middle: j -direction, lower: k -direction図4. メモリアクセス。左：オリジナル，右：最適化 / 上： i 方向，中： j 方向，下： k 方向

配列1次元目の寸法を16 Byte境界にアラインさせる。

$$IDIM \% (16 \text{ Byte} / N\text{Byte}) = 0 \quad (2)$$

ここで、 N は1個の変数が使用するByte数である。8 Byteの倍精度では $N = 8$ となる。

次に、1次元目の寸法 $IDIM$ が使用するバンク数を式(3)に従って奇数化する。式(3)の $\lceil \cdot \rceil$ は天井関数であり、+側の大きな値に向かって小数点以下を切上げる。

$$\lceil IDIM / (16 \text{ Byte} / N\text{Byte}) \rceil \% 2 = 1 \quad (3)$$

本手法では、式(2)と(3)を同時に満たすように $IDIM$ を決定する。

図4(右上)に本手法におけるメモリアクセスのイメージを示す。1次元目のアクセスについては変更前の図4(左上)と使用するバンク数は同じである。

図4(右中)に本手法における2次元目のメモリアクセスのイメージを示す。図4(左中)において左端のバンクBank00000に集中していたLoad/Storeが複数バンクBank00000~00001に分散される。

図4(右下)に本手法における3次元目のメモリアクセスのイメージを示す。3次元目についても図4(左下)において左端のバンクBank00000に集中していたLoad/Storeが複数バンクBank00000, 00002に分散される。

3.2 Z方向偏微分

Z方向の偏微分は図5に示すように変数 V （3次元配列）において K 方向（3次元目）の隣接アクセスを行う。図5の dzv はZ方向の偏微分であり、変数 V には粒子速度などの物理量が代入されている。また、“!CDIR”で始まる行はNEC SX専用のコンパイラ指示行である。NByte型変数 V の4個のLoadはメモリアクセス上では $IDIM * N\text{Byte}$ 飛びとなる。そこで、本研究では下記手法を用いた。

- ループを回す順序を $i \rightarrow k \rightarrow j$ に設定
(通常、 $i \rightarrow j \rightarrow k$ の順で回すループを変更)
- k を16段でアウターループアンローリング
(図5の“!CDIR OUTERUNROLL=16”に対応)

```

do j = 1, JDIM
!CDIR OUTERUNROLL=16
do k = 2, KDIM-2
!CDIR ON_ADB(V)
do i = 1, IDIM
dzv(i,j,k) = (V(i,j,k+1)-V(i,j,k))*R40 - (V(i,j,k+2)-V(i,j,k-1))*R41
enddo
enddo
enddo

```

Fig. 5. The optimized code in the direction Z.

図5. Z方向偏微分のFortran 90コード

- 2回以上Loadされる変数 V をADBにキャッシュ
(図5の“!CDIR ON_ADB(V)”に対応)

メモリアクセスの時間・空間局所性を高めるため、 j と k のループ交換、アウターループアンローリングとキャッシュ(ADB)のコンパイラ指示行を挿入した。また、ループアンロールの段数はトライ&エラーにより、ES2における最適値を求めた。

図6は図5のメモリアクセスのイメージである。図6の上段は初回のメモリアクセス、下段は初回以降のメモリアクセスである。横軸はメモリのバンク番号、縦軸はデータを表し、長方形1個は倍精度の変数256個を表す。また、ADBと記載された最上行はキャッシュを表し、ADBと記載されている部分はメモリではなくADBからLoadすることを表す。 $k \rightarrow k+1$ において変数 V の3個のLoadをADBから行うことが可能となり、Loadの効率化が図れる。なお、ライトスルー時のオーバーヘッドなどを回避するため、ADBにキャッシュする変数はRead Only Memoryとなる(Storeをしない)ものに限定する。

3.3 Y方向偏微分

Y方向の偏微分は図7に示すように変数 V において J 方向の隣接アクセスを行う。図7の dyv はY方向の偏微分であり、メモリアクセス上は $IDIM * N\text{Byte}$ 飛びとなる。そこで、本研究では下記手法を用いた。

- ループを $i \rightarrow j \rightarrow k$ の順に回す
- j を16段でアウターループアンローリング
(図7の“!CDIR OUTERUNROLL=16”に対応)
- 2回以上Loadされる変数 V をADBにキャッシュ
(図7の“!CDIR ON_ADB(V)”に対応)

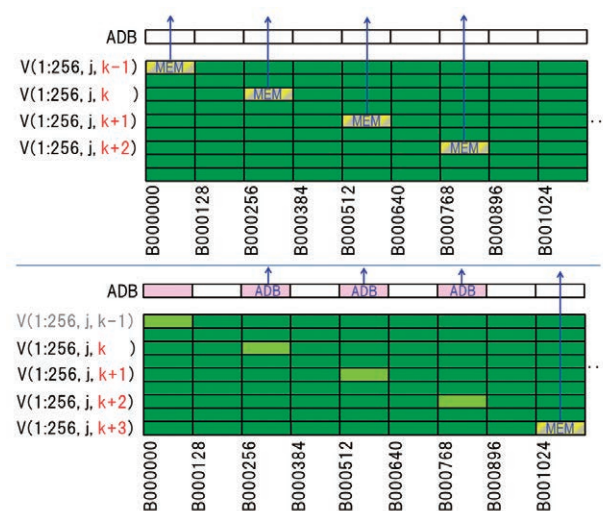


Fig. 6. The memory access in the direction Z.

図6. Z方向偏微分のメモリアクセス

メモリアクセスの時間・空間局所性を向上させるため、ループアンローリングおよびADB（キャッシュ）のコンパイラ指示行を挿入している。なお、ループアンロールの段数はES2における最適値をトライ&エラーで求めた。

図8は図7のメモリアクセスのイメージであり、長方形1個は倍精度の変数256個を表す。また、上段は初回メモリアクセス時、下段は初回以降のメモリアクセスである。本手法により、 $j \rightarrow j+1$ において変数 V の3個のLoadをADBから行うことが可能となる。

3.4 X方向偏微分

X方向の偏微分は図9に示すように変数 V において I 方向の隣接アクセスを行う。図9の dxv はX方向の偏微分である。メモリアクセス上は連続となるが、ES2（SX9）では機種固有の問題が発生する。

図10は図9のメモリアクセスのイメージである。1つの式で変数を4個Loadする場合、既にADBにキャッシュされているかどうかを判定せず、図10の様に特定のバンクに同時に4回Loadをする（撫佐, 2012）。この時、バンクコンフリクトが発生する。なお、NEC SX-7におい

```
do k = 1, KDIM
!CDIR OUTERUNROLL=16
do j = 2, JDIM-2
!CDIR ON_ADB(V)
do i = 1, IDIM
dxv(i,j,k) = (V(i,j+1,k)-V(i,j,k))*R40 - (V(i,j+2,k)-V(i,j-1,k))*R41
enddo
enddo
enddo
```

Fig. 7. The optimized code in the direction Y.

図7. Y方向偏微分のFortran 90コード

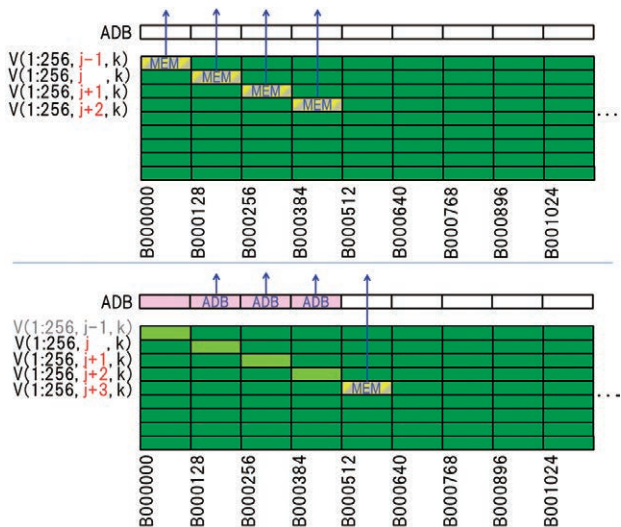


Fig. 8. The memory access in the direction Y.

図8. Y方向偏微分のメモリアクセス

てADBのキャッシュ状態を判定するMiss Status Handling Register（以下、MSHRとする）の実装により、2割の性能向上が報告されているが（佐藤ほか, 2008）、ES2では未実装である。そこで、本研究では下記手法を用いた（図11）。

- ループを回す順番を $j \rightarrow i \rightarrow k$ に強制的に変更（図11の行入替えと“!CDIR NOLOOPCHG”に対応）
- i を32段でアウターループアンローリング（図11の“!CDIR OUTERUNROLL=32”に対応）
- 2回以上Loadされる変数 V をADBにキャッシュ（図11の“!CDIR ON_ADB(V)”に対応）

ループアンロールの段数についてはES2で最適な値をトライ&エラーにより算出した。

図12は本手法適用時（図11）のメモリアクセスのイメージである。図中の縦長の長方形1個は倍精度の変数4個を表す。また、上段は初回メモリアクセス時、下段は初回以降のメモリアクセスである。最内側のループを j にすることにより、バンクコンフリクトを回避し、 $i \rightarrow i+1$

```
do k = 1, KDIM
do j = 1, JDIM
do i = 2, IDIM-2
dxv(i,j,k) = (V(i+1,j,k)-V(i,j,k))*R40 - (V(i+2,j,k)-V(i-1,j,k))*R41
enddo
enddo
enddo
```

Fig. 9. The memory access in the direction X.

図9. X方向偏微分のメモリアクセス

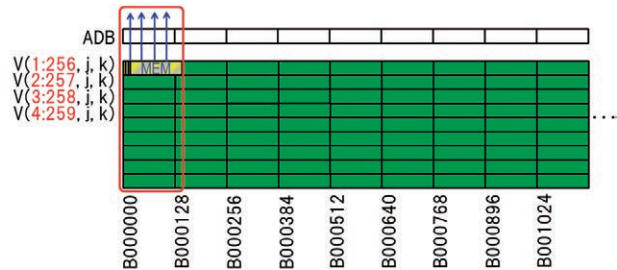


Fig. 10. The bank conflict in the direction X.

図10. X方向偏微分のバンクコンフリクト

```
do k = 1, KDIM
!CDIR OUTERUNROLL=32
!CDIR NOLOOPCHG
do i = 2, IDIM-2
!CDIR ON_ADB(V)
do j = 1, JDIM
dxv(i,j,k) = (V(i+1,j,k)-V(i,j,k))*R40 - (V(i+2,j,k)-V(i-1,j,k))*R41
enddo
enddo
enddo
```

Fig. 11. The optimized code in the direction X.

図11. X方向偏微分のFortran 90コード

において4個のLoadの3個以上をADBから読出しすることが可能である。ES2では16ByteずつLoadしているため、倍精度では最大1個の変数を先読みし、ADBにキャッシュしている。図12下段の例では、変数 V の一次元目添字 $i = 2 \sim 5$ をLoadする際、実際には $i = 1 \sim 6$ をLoadしているため、 $i = 3 \sim 6$ の時にはADB上にキャッシュされたデータのみで計算が可能である。

3.5 通信隠蔽

有限差分法などで領域分割型の分散メモリ型並列を行う場合、分割領域の境界部分で隣接間通信（1対1通信）が必要となる。通常、隣接間通信をしている間は、計算は待ち状態となる。

そこで、通信と計算を同時に実行可能なNon-blocking通信を用いることにより、通信時間を計算時間の裏に隠蔽した。Non-blocking通信の実装には、通信データを保管しておく袖領域を設け、通信に依存しない領域を計算しながら同時に通信する重複領域法（黒川ほか, 2001）を用いた。

通信隠蔽は計算時間を通信時間よりも大きくしないと効果が期待できないため、可能な限り計算を通信の間に移動した。また、CPU間で複数の通信が同時に発生した場合、通信衝突が発生し、待ち時間が増大する可能性がある。このため、通信は出来るだけ1つにまとめて通信回数を削減し、通信タイミングを制御することで通信衝突の発生を軽減した。

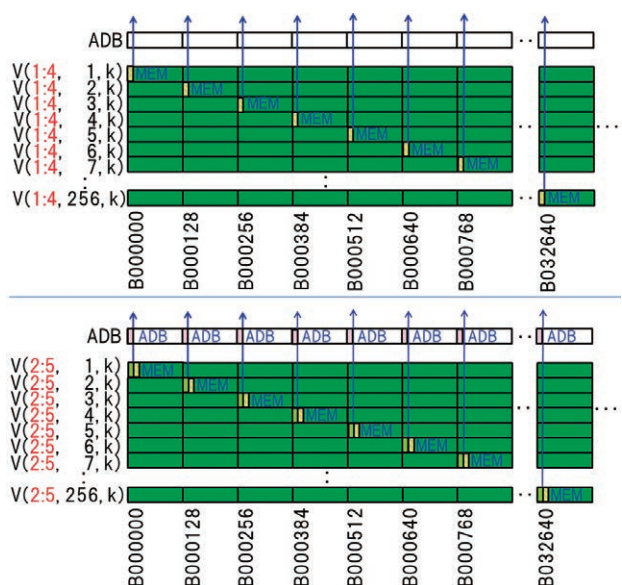


Fig. 12. The memory access in the direction X.

図12. X方向偏微分のメモリアクセス

4. 並列性能評価

本章ではk-fdm3dの並列計算性能を評価する。4.1節では高速化前後の性能を示す。4.2節では並列計算による高速化の評価を行う。4.3節では大規模問題への適正を評価する。また、4.4節では性能分析に係る補足データを示す。

4.1 高速化前後の実行時間

本節では高速化前後でのコード全体の性能比較を行う。表1に格子数 $IDIM = 1,024$, $JDIM = 1,280$, $KDIM = 340$, 500タイムステップ計算の1ノード、8CPU使用時における性能を示す。コード全体では倍精度で1.395倍、単精度で1.372倍高速化しており、約3割のピーク性能比が達成できていることが確認できる。

表2にこの時の性能概要を示す。3.1節のバンクコンフリクト回避策により、バンクコンフリクトの発生率が4割から3割に低減している。また、3.2～3.4節の最適化施策により、各方向の偏微分は1割以上高速化している。特に3.2節のZ方向偏微分では2.5倍以上高速化しており、ADBの有効性が確認できた。また、3.5節の通信隠蔽により、通信は約3割高速化した。

4.2 並列加速率

本節ではk-fdm3dの並列加速率（ストロング・スケーリング）を評価するため、問題サイズを $IDIM = 1,024$, $JDIM = 1,280$, $KDIM = 340$ に固定して、500タイムステップの計算時間を計測した。図13に並列加速率を示す。横軸はCPU数、縦軸は並列加速率を表す。なお、図13は10を底とする両対数グラフである。実線は理想的なスピードアップであり、プロットは実測値である。また、○は倍精度、×は単精度の結果を表す。領域分割法を用いる場合、CPU数が増えるにつれて並列粒度が小さくなるため、256回の演算を1度に行うベクトルプロセッサの性能を生かすことが難しくなる。特に、256CPU以上では並列化効率が5割未満となるため、この計算規模では256CPU未満で計算するのが適切と考えられる。

Table 1. The performance comparison of optimization.

表1. 8CPU使用時の性能比較

	経過時間[s]	GFLOPS (1CPU)
高速化前（倍精度）	219.622	22.416
高速化後（倍精度）	157.468	30.334
高速化前（単精度）	182.745	26.941
高速化後（単精度）	133.163	35.861

4.3 スケーラビリティ

本節ではk-fdm3dの大規模問題への適正（ウィーク・スケーリング）を評価するため、1 CPUあたりが計算する格子点数を $IDIM = 512$, $JDIM = 640$, $KDIM = 170$ に固定した。使用CPU数に比例して総格子点数を増加させ、500タイムステップの計算時間を計測した。8 CPUまでは1ノードを用い、16 CPU以降はノードあたり8 CPU全てを使用した。図14に測定結果を示す。横軸はCPU数、縦軸はスケーラビリティである。なお、図14は10を底とする片対数グラフである。実線は理想的なスケーラビリティ、プロットは実測値である。また、○は倍精度、×は単精度の結果である。ES2は通信に比べて計算が非常に速いため、通常のスカラ型計算機よりもスケーラビリティは低くなる傾向があるが、1,024 CPU使用時でも約8割のスケーラビリティを保っている。また、この時の総格子点数は約570億である。なお、8 CPU使用時に性能低下がみられたため、4.4節に8 CPU使用時の補足データを示す。

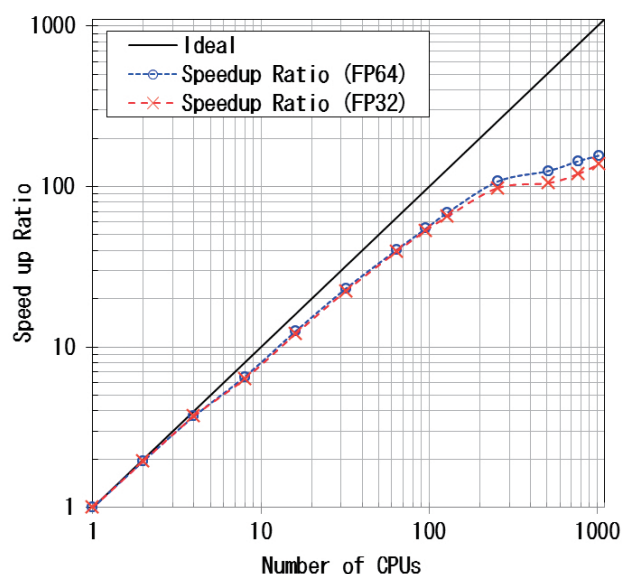


Fig. 13. The speed-up ratio (strong scaling)

図13. 並列加速率（ストロング・スケーリング）

Table 2. The summary of optimization technique.

表2. 8 CPU 使用時の性能概要

	バンクコンフリクト割合[%]	Z方向偏微分加速率	Y方向偏微分加速率	X方向偏微分加速率	通信加速率
高速化前(倍精度)	40.909				
高速化後(倍精度)	28.730	2.530	1.139	1.531	1.360
高速化前(単精度)	43.059				
高速化後(単精度)	30.947	2.671	1.150	1.167	1.304

4.4 複数CPU使用時のバンクコンフリクト

4.3節のスケーラビリティ（ウィーク・スケーリング）において、8 CPU使用時に性能が顕著に低下する要因としては、バンクコンフリクト発生割合の増加が挙げられる。

表3に8 CPU実行を1～8ノードを使用して計測した結果を示す。倍精度では1ノードあたり1 CPUを使用した場合、バンクコンフリクトの発生割合は18.270[%]である。一方、1ノードあたり8 CPUを使用した場合にはバンクコンフリクトの発生割合は28.721[%]まで上昇する。このバンクコンフリクトはMPI通信以外の部分で発生しており、経過時間は約12.855[%]増加する。また、単精度も同様の傾向を示している。

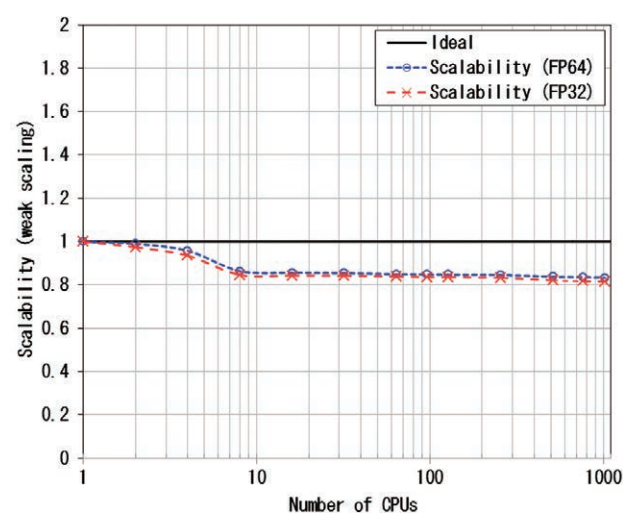


Fig. 14. The scalability (weak scaling)

図14. スケーラビリティ（ウィーク・スケーリング）

Table 3. The bank conflict caused by CPUs.

表3. 複数CPU使用時のバンクコンフリクト

	経過 時間[s]	バンクコン フリクト発生[s]	GFLOPS (1CPU)
8CPU × 1NODE (倍精度)	157.468	45.227	30.334
4CPU × 2NODE (倍精度)	143.903	30.950	33.235
2CPU × 4NODE (倍精度)	140.347	26.584	34.290
1CPU × 8NODE (倍精度)	139.531	25.492	34.548
8CPU × 1NODE (単精度)	133.163	41.196	35.861
4CPU × 2NODE (単精度)	121.649	29.648	39.289
2CPU × 4NODE (単精度)	119.211	26.216	40.159
1CPU × 8NODE (単精度)	117.291	23.324	41.173

5. 考察

本研究で提案したステンシル計算の高速化手法は、ES2において約3割のピーク性能比を達成した。また、最小限の変更により高速化が行えるため、ソースコードの派生バージョン増加やバグ発生を抑制することができる。本手法は、地震動シミュレーションだけではなく、構造格子を用いた有限差分法などでも有効であると考えられる。

複数CPU使用時のバンクコンフリクト発生割合の上昇は、複数CPUがLoadをする際に、特定バンクにアクセスが集中し、衝突を起こす確率が上昇するためと考えられる。各CPUが使用するバンクを図15のように制限できればバンクコンフリクトは解消できると考えられる。図15の例では、各CPUが2,048バンク（32メインメモリユニット）ずつ排他的に使用することで、複数CPUによるバンクコンフリクトを回避する。このメモリアクセスの制御により、バンクコンフリクトが解消された場合、1,024 CPU使用時のスケーラビリティ（ウィーク・スケーリング）は約9割まで向上することが期待できる。

本高速化手法は他の計算機にも適用が可能である。ただし、3.4節のxとyのループ順序変更はNEC/SX-9以外では不要である。NEC/SX-ACEはMSHRを搭載しており、ループ順序変更による大きな性能向上は難しい。また、一般的なスカラーCPUではメモリアクセスが不連続となるため、却って性能が悪化する可能性が高い。

6. おわりに

本論文で開発したステンシル計算の高速化手法は、最小限の簡単な変更により、ES2において約3割のピーク性能比を達成することができる。また、570億格子点の大規模並列計算が可能である。以上により、大規模平野などにおける高周波数領域の地震動を高速に計算する手法を

確立することができた。

謝 辞

本研究は平成21～23年度「地球シミュレータ産業戦略利用プログラム」の一環として実施した。また、査読者である東京大学の三好 崇之博士と海洋機構 板倉 憲一博士より、的確で有益な助言を多数戴きました。記して感謝いたします。

参考文献

- Furumura, T. and L. Chen (2005), Parallel simulation of strong ground motions during recent and historical damaging earthquake in Tokyo, Japan, *Parallel Computing*, 31, 149–165.
- Furumura, T. and T. Saito (2009), Integrated Ground Motion and Tsunami Simulation for the 1944 Tonankai Earthquake Using High-Performance Supercomputers, *Journal of Disaster Research*, 4(2), 118–126.
- 古村孝志 (2009a), 3次元不均質場での波動伝播と強震動のシミュレーション, 平成21年度地球シミュレータ利用報告会, <<https://www.jamstec.go.jp/esc/projects/fy2009/13-furumura.pdf>>.
- 古村孝志 (2009b), 差分法による3次元不均質場での地震波伝播の大規模計算, 地震第2輯, 第61巻特集号, S83-S92.
- 古村孝志 (2010), 地震と津波発生伝播の大規模3次元シミュレーション, T2K (東大) 共同研究プロジェクト利用報告会2010.
- 板倉憲一 (2012), 地球シミュレータのアーキテクチャ・運用・成果, 京大セミナー, <<http://www.cs.kyoto-u.ac.jp/wp-content/uploads/2012/06/01itakura.pdf>>.
- JAMSTEC (2008), The outline of the Earth Simulator (ES2), <<http://www.jamstec.go.jp/esc/publication/leaflet/pdf/system1.pdf>>.
- 黒川原佳, 松澤照男, 姫野龍太郎, 重谷隆之 (2001), 並列CFD計算における非同期通信・計算重複法, 情報処理学会論文誌, ハイパフォーマンスコンピューティングシステム, 42, SIG 9 (HPS3), 54–63.
- 撫佐昭裕 (2012), 現場でのチューニング活動とリファクタリングカタログ, 第4回自動チューニング技術の現状と応用に関するシンポジウム.
- Nakazato S., S. Tagaya, N. Nakagomi, T. Watai, A. Sawamura (2008), Hardware Technology of the SX-9 (1)- Main System -, *NEC Technical Journal*, 3(4), 15–18.
- 佐藤義永, 撫佐昭裕, 江川隆輔, 滝沢寛之, 岡部公起, 小林広明 (2008), ベクトルプロセッサ用キャッシュメモリに

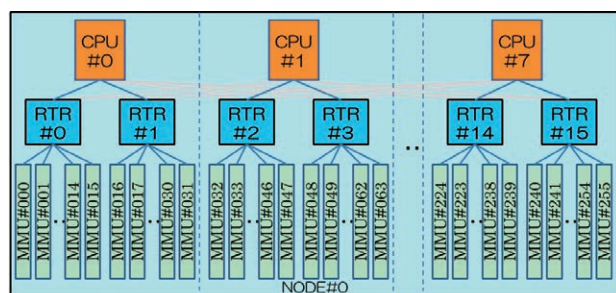


Fig. 15. The example of restricting bank allocation.

図15. バンク割当制限の例 (FlatMPI時)

におけるMSHRの性能評価, 次世代スーパーコンピューティング・シンポジウム2008資料集, 57–58.

佐藤義永, 永岡龍一, 撫佐昭裕, 江川隆輔, 滝沢寛之, 岡部公起, 小林広明 (2009), キャッシュメモリを有するベクトルプロセッサのためのプログラム最適化手法, 情報処理学会研究報告2009-ARC-184(6), 1–10.

田島礼子, 西條裕介, 正月俊行, 司宏俊, 廣川雄一 (2012), 3次元差分法による関東平野での広帯域地震動シミュレーションの検討, 2012年度日本建築学会大会学術講演梗概集, 構造II, 147–148.

司 宏俊, 西條裕介, 正月俊行, 内山不二男, 諸遊克己, 嶋村洋介, 戸井隆, 渡辺高志, 廣川雄一 (2009), 地震時の大規模平野の地盤挙動と斜面崩壊シミュレーション技術の開発, 平成21年度先端研究施設共用促進事業「地球シミュレータ産業戦略利用プログラム」利用成果報告書, 103–110.