Vol.2012-ARC-202 No.11
Vol.2012-HPC-137 No.11
2012/12/13

地震動シミュレータGMSのOSCARコンパイラによる自動並列化

島岡 護[1,a)]　見神 広紀[1]　林 明宏[1]　和田 康孝[1,†1]　木村 啓二[1]　森田 秀和[2]　内山 邦男[2]
笠原 博徳[1]

概要：地震など自然災害から人命を救うために災害シミュレーションが注目を集めている．シミュレーションの高精度化とそれに伴うシミュレーション時間の増大を抑えるための処理の並列化が重要な課題となっている．本稿では防災科学技術研究所で開発された不連続格子を用いた差分法による地震動シミュレータ GMS(Ground Motion Simulator) の OSCAR 自動並列化コンパイラを用いた並列化手法について報告する．
POWER7 ベースの 128 コア SMP サーバ Hitachi SR16000 上での性能評価の結果、OSCAR 自動並列化により 1 コアでの実行と比べて 92 倍の速度向上をすることができた．同様に Xeon ベースの 64 コアブレード SMP サーバ Hitachi BS2000 上での性能評価の結果、OSCAR 自動並列化により 1 コアでの実行と比べて 43 倍の速度向上をすることができた．

Automatic Parallelization of Ground Motion Simulator

Abstract: To protect human's lives from disaster, disaster simulation attracts a lot of attensions in recent years. Parallelization of the simulation is an important issue to calculate more bigger and precise simulations. In this article, I report an automatic parallelization of GMS(Ground Motion Simulator) by OSCAR compiler. The performance evaluation shows that the automatic parallelization achives 92 times speedup on 128 cores Hitachi SR16000 with 16 POWER7 processors, and 43 times speedup on 64 cores Hitachi BS2000 with 8 Xeon processors, respectively.

1. はじめに

世界の地震の 2 割は日本で発生しており、地震、津波などの災害への対策は非常に重要である．災害に対して被害を最小限にするためには予兆、初動の早期発見、被害の事前予測とそれに基づいた対策が重要である．被害の正確な予測には災害のシミュレーションが必要であるが、シミュレーションの精度と計算時間の間は一般にトレードオフの関係となる．

精度の高いシミュレーションを高速に行うためにはシミュレーションのアルゴリズム自体の改良と高精度化に伴うシミュレーション時間の増大を抑えるプログラムの並列化による高速化が必要である．

akcelik らは有限要素法を用いた地震動シミュレータを MPI を用いて並列化し，ピッツバーグスーパーコンピュータセンターの hp AlphaServer SC を用いて 3000 コアで並列化効率 80%という高いスケーラビリティを示している [1]．

また furumura らは有限差分法を用いた地震動シミュレータを MPI，SMP を組み合わせて並列化し地球シミュレータを用いて性能評価を行い，本稿で性能評価を行う問題に近いサイズの問題に対して 128 コアで 8 コアから 6 倍近い速度向上を示している [2]．

本稿では SMP や cc-NUMA 等の主記憶共有型の並列計算器を対象として逐次プログラムとして記述された地震動シミュレータ (GMS: Ground Motion Simulator) の波動方程式計算部について当研究室で開発している自動並列化コ

[1] 早稲田大学
Waseda Univ, Shinjuku, Tokyo 169–8555, Japan
[2] 株式会社 日立製作所
Hitachi, Ltd
[†1] 現在，電気通信大学
Presently with The University of Electro-Communications
[a)] shimaoka"at"kasahara.cs.waseda.ac.jp

ンパイラ OSCAR により並列化を行った．逐次プログラムを自動並列化により高速化することにより，並列化に関する専門知識を必要とせずに並列計算の専門家の書いた並列化プログラムと同等のスケーラビリティを得ることを目標とする．

本稿が並列化の対象とする地震動シミュレータ (GMS: Ground Motion Simulator) とは防災科学技術研究所の青井，早川，藤原らにより提案された差分法を用いた波動伝播シミュレーションにより効率よく地震動の計算を行うシステムである [3,5,6]．GMS の特徴は媒質の違う領域ごとに不連続な格子を用いることにより効率的な差分計算を行うソルバーと，パラメータ設定や計算結果の評価を行うためのプログラム一式がシステム化されており，詳細なマニュアルと共に無償で提供されている点である [4]．また MPI で並列化した GMS は 2CPU からなるノードを 16 ノード，計 32 コアまでの CPU を持つ PC クラスタシステムによる性能評価では本稿の性能評価に用いた同じ問題について 1 コアに対して 16 コアで 14 倍，32 コアでは 17 倍の速度向上を得ている [3]．公開された GMS のシステム内には Fortran90 で記述されたソルバーのソースコードが含まれいる．本稿ではこの GMS ソルバーを自動並列化コンパイラ OSCAR により並列化し、Hitachi SR16000 および BS2000 上で性能評価を行う．自動並列化したプログラムを first touch によりメモリ配置を行う NUMA 型サーバにより性能を評価した結果，スレッドのバインドを適切に設定するとともに first touch を考慮して配列の初期化を並列化することで並列化によりスケーラブルな性能向上が得られたので，この結果について報告する．

以下 2 章では GMS の概要，3 章では OSCAR 自動並列化コンパイラ，4 章では GMS に対して適用された並列化方法，5 章では性能評価についてそれぞれ述べる．

2. 地震動シミュレータ GMS

本章では地震動シミュレータ (GMS: Ground Motion Simulator) の概要について述べる．2.1 節では GMS 全体の概要とフローを示し，2.2 節では GMS ソルバーの概要について述べる．

2.1 GMS システム

本節では地震動シミュレータ (GMS: Ground Motion Simulator) システム全体の概要及び構成について述べる．

GMS とは青井, 早川, 藤原らにより提案された複数の点震源で近似された面的な広がりをもつ断層モデルと震源域から観測点を含む水平方向が数百 km，深さ方向が百 km 程度の盆地構造などを含む 3 次元的不均質地下構造モデルに対して差分法を用いた波動伝播シミュレーションにより効率よく地震動の計算を行うシステムである．GMS は差分計算を行うソルバーとともに，パラメータ設定や計算結果の評価を行うためのプログラム一式がシステム化されており, 詳細なマニュアルと共に無償で提供されている [3–6].

2.2 GMS ソルバー

本節では本稿で並列化の対象とする GMS ソルバーの概要を説明する．

GMS ソルバーは波動方程式を差分計算によって近似して計算する．GMS ソルバーの特徴は差分法により計算を行う格子点を媒質ごとに不連続な格子とすることにより計算量，必要なメモリを大幅に減らしている点である．精度の高いシミュレーションを行うために必要な格子点の間隔は地震波の伝播速度により変わる．地表面では媒質の変化から地震波の伝播が遅くなり，格子間隔を細かくとる必要がある．計算対象となる媒質全体を均質な間隔の格子に分割する場合，ごく表層の媒質のために全体を小さい間隔の格子に分割することになる．これは計算効率を落とすため，GMS ソルバーでは媒質ごとに不連続な格子とすることで地表面のみ細かい格子で分割し，それ以外の部分を粗い格子で分割することで効率的な計算が可能としている．

図 1 に GMS ソルバーのメインループ内の関数および関数間のデータ依存関係を示す．

GMS は地震波の伝播速度の異なる表層と下層にわけて計算を行う．図 1 では左側の関数が表層の計算となり右側の関数が下層の計算となる． 表層と下層の境界部での計算を除いて表層と下層は並列に計算が可能であるが，表層と下層の負荷はシミュレーションのパラメータ設定により異なり，また関数内に十分な並列性があるため，関数内の並列性を利用するほうが効率的な並列処理が可能である．また GMS では各格子点の速度を応力から計算し，応力を更新するというように速度と応力の計算を交互に繰り返している．図 1 では上部が各格子点の速度の計算，下部が各格子点の応力の計算に相当する．GMS では震源に対して振動を与える必要性があるが，震動を速度で与えるか応力で与えるかにより計算を行う位置が変わる．図 1 中の震源処理 A，震源処理 B がそれぞれ速度での震源の処理と応力での震源の処理の相当し，シミュレーションのパラメータ設定で与えられるフラグにより互いに排他的に実行される．5 章で性能評価に用いたシミュレーションパラメータ ex2 では震動は応力で与えられている．

3. 自動並列化コンパイラ OSCAR

本章では自動並列化コンパイラ OSCAR [8] の概要について述べる．

OSCAR 自動並列化コンパイラをでは従来のループ並列化に加え，粗粒度タスク並列処理，近細粒度並列処理を効果的に組み合わせたマルチグレイン並列処理を実現している．

粗粒度タスク並列化では，ソースプログラムを 基本ブ

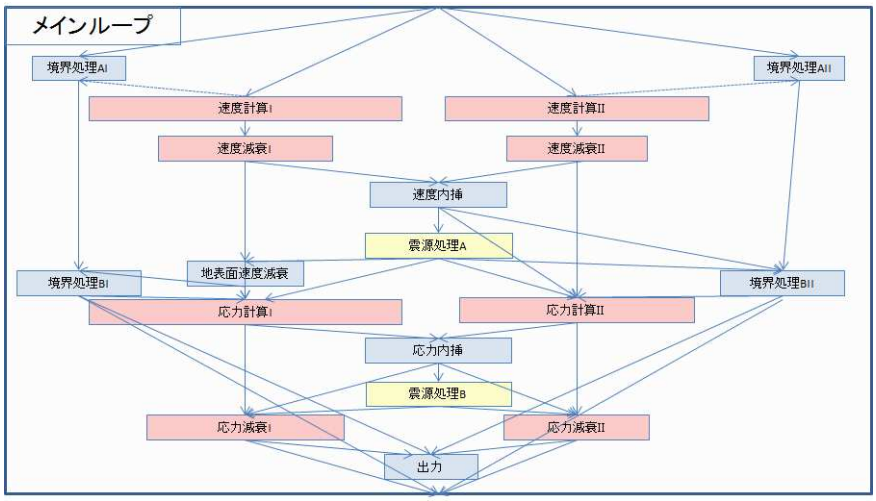

図 1　GMS ソルバーメインループ内の関数

ロックやループ，サブルーチン呼び出しの 3 種類の粗粒度タスクに分解後，最早実行可能条件解析によって，各粗粒度タスク間の並列性を抽出し，マクロタスクグラフ（MTG）を生成する [9]．粗粒度タスクがサブルーチンブロック，ループブロックである場合は，階層的にその内部を粗粒度タスクに分割して階層的 MTG を生成し，プログラム全域の階層的な並列性を抽出する．

また OSCAR コンパイラで並列化されたプログラムは最初に一度だけコア数分のスレッド生成を行い，それ以降はスレッド生成を行わないため，スレッド生成によるオーバーヘッドを低減できる．

OSCAR コンパイラは Fortran と C で記述されたソースコードを入力として，プログラムの解析を行い，並列化を行ったプログラムを OSCAR_API の指示文が挿入された元のソースコードと同じ C や Fortran として出力する [10]．OSCAR_API は OpenMP 互換であり SMP サーバ機では OpenMP コンパイラでコンパイルすることで並列化されたバイナリを得ることができる．

4. GMS ソルバーの並列化

本章では OSCAR コンパイラでの並列化の際に GMS ソルバーの逐次ソースコードに対して加えた変更点について述べる．

GMS ソルバーは Fortran90 で記述されている．OSCAR コンパイラは Fortran90 に未対応であるが，メインループの内部は出力関数を除いて Fortran77 相当で記述されているため OSCAR コンパイラで並列化可能であった．

また並列化に際しプログラムに以下の 3 点の変更を行い，並列性を向上させた．またその他に今回は問題の簡単化のために一時的に配列サイズ，震動を速度と応力のどちらかで与えるか決定するフラグを評価問題用に固定した他，より多くのタスク間の並列性を利用するために関数をインライン展開している．図 2 に OSCAR コンパイラによる GMS のマクロタスクグラフを示す．図 2 中のノード一つ一つがタスクであり，タスクからタスクへのエッジはデータ依存を表す．データ依存のエッジの先のタスクはデータ依存の元のタスクの実行が終了するまで計算を開始することができないことを表している．図の中央にある全てのタスクにデータ依存があるタスクが震源処理となる．図 2 中で横に並んだタスクは粗粒度並列処理が可能である．更に多くのタスクはイタレーションレベルの並列化が可能な doall ループである．

4.1 ループインターチェンジ

GMS ソルバーにおいて実行時間の多くを占めるタスクは，速度の計算部と応力の計算部である．それぞれのタスクは x,y,z 軸の 3 つに相当する 3 重ループを含むが，オリジナルのソースコードでは最も外側のループは z 軸となっている．表層の z 軸のループの回転数は今回性能評価に用いた ex2 や青木らの性能評価に用いられた大規模シミュレーション問題においても 30 前後であり，メニーコア上では z 軸のループのみでの並列化では並列性が使用プロセッサ数よりも下回ってしまう．そのため ex2 において 1020 回転である x 軸のループが最も外側になり，ついで ex2 において 720 回転の y 軸のループ，ex2 において 36 回転の z 軸のループとなるようにループインターチェンジを行った．これにより本稿で評価に使用した環境の最大のコア数は 128 のため最外側のループの並列化のみで十分な並列性を得ることが出来る．

4.2 作業配列へのコピー

本稿の評価で使用する Hitachi SR16000，BS2000 のような cc-NUMA 並列計算器では論理アドレスと物理アドレスとの割り当てが first touch policy で行われる．すなわち，

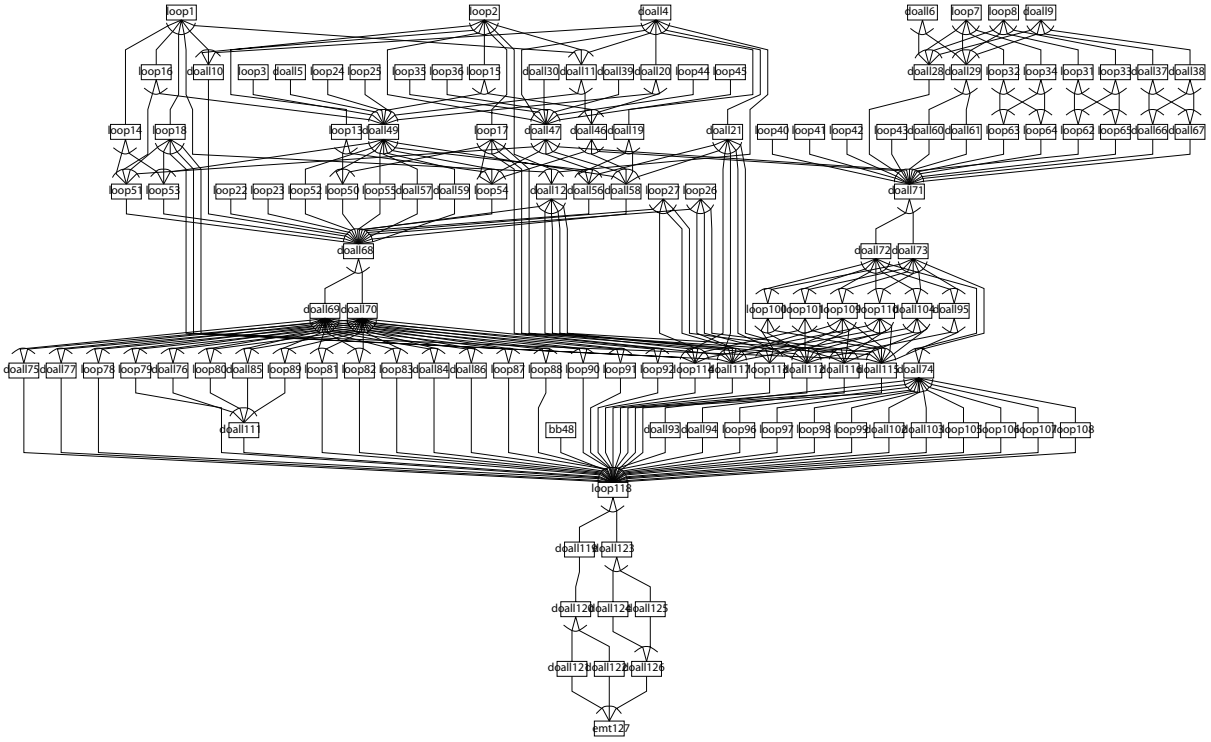

図 2　GMS MTG(ex2)

変数の配置場所は最初にその変数をアクセスしたノード近傍の物理メモリとなる．そのため変数の初期化部が並列化されない場合，全ての変数がメインスレッドを実行していたコアに近接するメモリに配置されることになり，他ノード上のコアの主記憶上に対するアクセスはリモートメモリアクセスとなり，並列処理時のメモリアクセスオーバーヘッドが大きくなる．変数の初期化は Fortran90 で記述された外部ツールによって行われており，OSCAR コンパイラは Fortran90 に未対応なため，今回は配列をコピーしその部分を並列化することで各プロセッサの近接のメモリに変数を分配した．また，ループインターチェンジにより配列のアクセスパターンが変化したため効率的な配列へのアクセスが可能となるように配列の次元の入れ替えを行った．

4.3　震源に対する処理部の分散化

GMS による地震動の計算では個々の問題ごとに別に設定された震源に対して震動を加える処理が必要となる．震源の位置は問題により異なりコンパイル時には未知であり，従来の GMS ソルバーの計算方法では震源に対する処理は全てのタスクに対してデータ依存があった．すなわちオリジナルのソース中で震源の計算以前に現れるタスクは OSCAR コンパイラによる粗粒度並列処理時においても全て震源の計算以前に終了する必要があり，オリジナルのソース中で震源の計算より後ろに現れるタスクは OSCAR コンパイラによる粗粒度並列処理時においても全て震源の計算終了後に計算を開始する必要性があった．これは不要な同期を必要とするため図 3 のように x 軸を下端から上端まで回転するループ中で震源の x 座標を確認し震源の x 座標とループ変数の値が一致した場合に震源への計算をする形に書き換えた．点震源が複数ある場合も一般的には断層としてある地点に集中すると考えられるため，この書き換えを行っても震源の計算は一部のコアに集中し，全コアで負荷分散されることにはならないと考えられるが，このループを並列化することにより震源の処理を全てのタスクが待機する必要はなくなった．

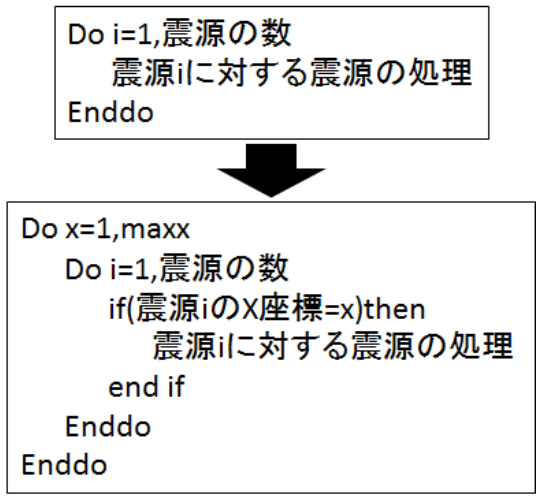

図 3　震源の処理の書き換え

5.　並列処理性能評価

本章では Hitachi SR16000 と BS2000 上での 4 章で述べ

た逐次ソースへの修正を行った GMS ソルバーの OSCAR コンパイラによる自動並列化の性能評価について述べる．

5.1 性能評価環境

本稿では Hitachi SR16000 および BS2000 を使い GMS ソルバーの実行時間を評価した．オリジナルの GMS を SR16000 では XL Fortran コンパイラ，BS2000 では Intel Fortran コンパイラで自動並列化したものに加え，4 章でより並列性の高い形の逐次プログラムに書き換えた GMS を OSCAR コンパイラで粗粒度タスク並列化したものについて性能を評価した．実行時間は GMS のメインループのうちファイルの入出力部を除いた部分に関して計測している．また，書き換えを行った GMS ソルバーのソースについて OSCAR コンパイラを用いずに XL Fortran コンパイラ，Intel Fortran コンパイラにより自動並列化を試みたが，インライン展開によりコードサイズが増大し各コンパイラの解析能力を超えたために並列化がされずコア数の増加による速度向上がなかったため本報告では XL Fortran 及び Intel Fortran についてはオリジナルの GMS の自動並列化について報告する．

図 4 に SR16000 のアーキテクチャを示す．SR16000 は POWER7 プロセッサ [7] を 16 個搭載した 128 コア SMP サーバである．POWER7 は 1 プロセッサあたり 8 個のコアを持ち，8 コアで 32MB のオンチップ L3 キャッシュを共有している．各コアは 4GHz で動作し、256KB の L2 キャッシュを持つ．また，図 4 の通り SR16000 は各プロセッサごとに分散共有メモリを持つ NUMA アーキテクチャである．変数は分散共有メモリに first touch policy で配置される．SR16000 は 4 つのモジュールが搭載され 1 モジュールあたり 4 プロセッサが搭載され，モジュール内のプロセッサ間は完全結合網で接続されている．また 4 つのモジュールも完全結合網で接続される．SR16000 で使用した XL Fortran コンパイラのバージョンは 13.1 であり，コンパイルオプションは XL Fortran による自動並列化の際には-qsmp=auto, -O5, -q64，OSCAR コンパイラで並列化して出力した fortran ソースに対しては-qsmp=noauto, -O5, -q64 を使用した．また OSCAR コンパイラでは sched_setaffinity システムコールにより詳細なバインドの設定が可能であるが，XL Fortran の自動並列化の評価時には XLSMPOPTS 環境変数によりバインドを設定して評価を行った．

図 5 に BS2000 のアーキテクチャを示す．本稿で性能評価に用いた BS2000 は Xeon E7-8830 を 8 個搭載した 64 コア SMP サーバである．Xeon E7-8830 は 1 プロセッサあたり 8 個のコアを持ち，8 コアで 24MB の L3 キャッシュを共有している．各コアは 2.13GHz で動作し、256KB の L2 キャッシュを持つ．SR16000 同様分散共有メモリを持つ NUMA アーキテクチャであり，first touch policy となっている．BS2000 には 4 つのブレードが搭載され 1 ブレードあたり 2 プロセッサが搭載されている．それぞれのプロセッサは QPI により他の 3 つのプロセッサと接続されすべてのプロセッサ、分散メモリ間において 2hop 以内で通信が可能となっている．使用したコンパイラは 64bit 版 Intel Fortran コンパイラのバージョン 12.1.0 で，コンパイルオプションは-O2,-static,-xsse4.2,-mcmodel=large を共通として Intel Fortran コンパイラによる自動並列化の際には-parallel, OSCAR により自動並列化したソースに対しては-openmp を使用した．Intel Fortran の自動並列化の評価時には KMP_AFFINITY 環境変数によりバインドを設定して評価を行った．

性能評価に用いた問題は公開されている GMS パッケージに用意されていたシミュレーションパラメータ ex2 を利用した．表 2 に ex2 の格子点の数を示す．また ex2 は実時間 48 秒のシミュレーションを 6000 ステップに分割してシミュレーションを行うが，今回は評価時間の短縮のために先頭の 10 ステップのみの実行時間について評価した．先頭 10 ステップは 0.08 秒のシミュレーションに相当する．また GMS のメインループのうちファイルの入出力部を除いた部分に関して計測している．

表 2 ex2 格子点数

	x 軸	y 軸	z 軸
表層	1020	720	38
下層	340	240	105

5.2 評価結果

図 6 に SR16000 における XL Fortran 及び OSCAR により自動並列化した GMS の XL Fortran の逐次実行を基準とした速度向上率を示す．横軸に並列実行に使用した POWER7 コア数を取り，縦軸に GMS の XL Fortran の逐次実行を 1.0 とした速度向上率を示している．OSCAR の 1 コアでの速度向上率を見ると OSCAR が XL Fortran での逐次実行に対してい 1.49 倍速くなっていることがわかる．書き換えを行った GMS に対しては XL Fortran コンパイラによる並列化が出来なかったが書き換え自体は GMS ソルバーを高速化させていることがわかる．逐次実行の速度向上の要因の一つとしては配列の動的確保部を ex2 用に配列サイズを固定し静的に割り当てたことが考えられる．128 コア時の XL Fortran 逐次実行を基準とした速度向上率は XL Fotran コンパイラは 7.49 倍，OSCAR コンパイラは 137.28 倍であり，OSCAR コンパイラは同じ数のコアを利用して XL Fortran コンパイラの 18.33 倍速く計算が可能であった．逐次実行時より差が大きくなっていることから OSCAR のほうが XL Fortran コンパイラより並列化により速度向上が大きいことがわかる．

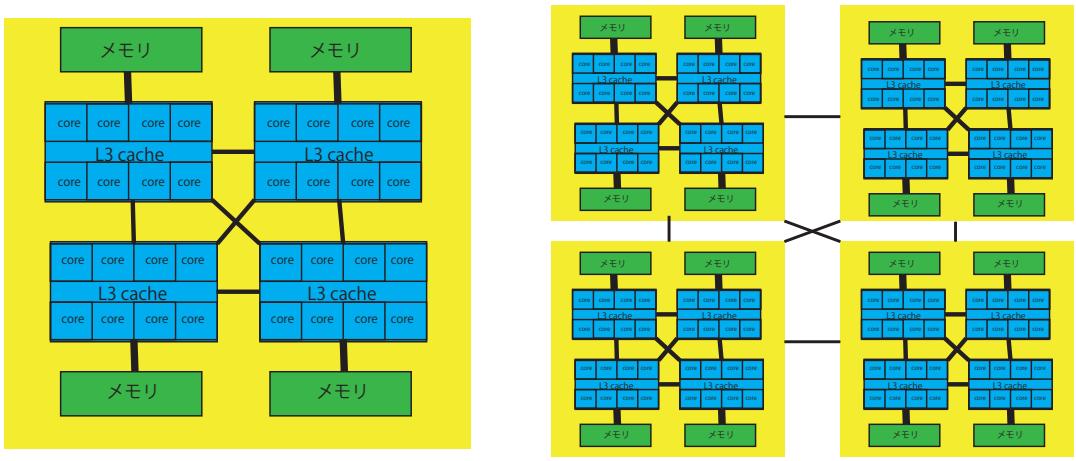

図 4　Hitachi SR16000 アーキテクチャ

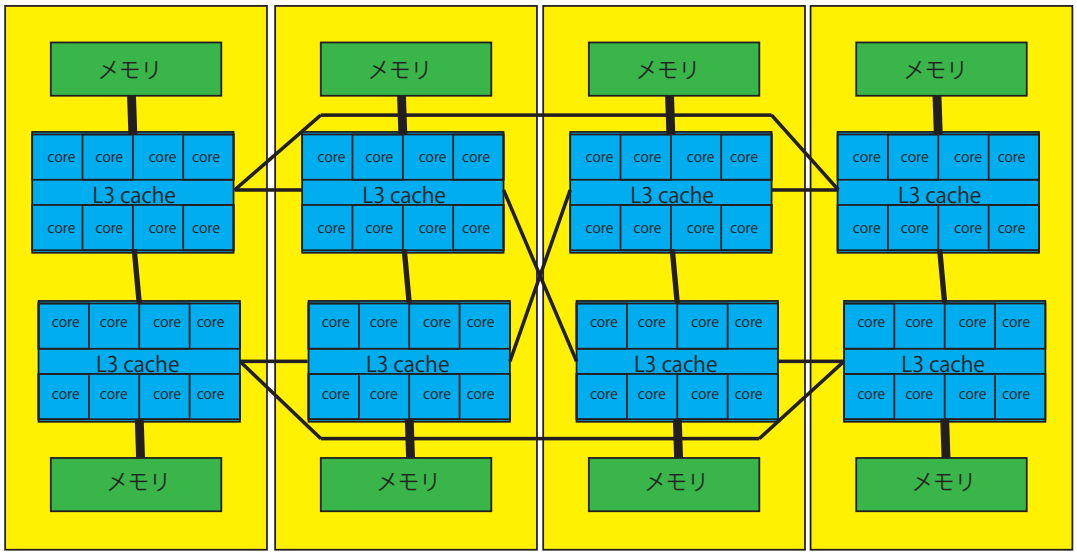

図 5　Hitachi BS2000 アーキテクチャ

図 6 の系列 SR16000 xlf を見ると SR16000 における XL Fortran 自動並列化では 32 コアで 11.4 倍の速度向上が最も大きく，64 コアでは 9.9 倍となり若干低下している．これはコア数の増加に従い他のコアの計算結果を参照する割合が大きくなる事に加え，first touch するコアと実際に計算に利用するコアの食い違いによる遠隔分散共有メモリのアクセス，スレッド生成オーバーヘッドなどが原因として考えられる．また XLSMPOPTS 環境変数によるバインドの設定ではどのスレッドがどのコアにバインドされるかは実行するたびに変わってしまうこともメモリアクセスの効率を悪化させていると思われる．

図に BS2000 における Intel Fortran 及び OSCAR により自動並列化した GMS の Intel Fortran の逐次実行を基準とした速度向上率を示す．横軸に並列実行に使用した Xeon コア数を取り，縦軸に GMS の Intel Fortran の逐次実行を 1.0 とした速度向上率を示している．OSCAR の 1 コアでの速度向上率を見ると OSCAR の逐次実行は Intel Fortran コンパイラの逐次実行より 1.09 倍速くなっていることがわかる．書き換えを行った GMS は Intel Fortran コンパイラでも並列化が出来なかったが BS2000 においても逐次実行の速度向上に寄与していることがわかる．64 コア時の Intel Fortran 逐次実行を基準とした速度向上率は Intel Fotran コンパイラは 23.67 倍，OSCAR コンパイラは 43.75 倍であり，OSCAR コンパイラは同じ数のコアを利用して Intel Fortran コンパイラの 2.08 倍速く計算が可能であった．

次に図 8 にそれぞれの 1 コアに対する速度国情率を示す．横軸に並列実行に使用したコア数を取り，縦軸に GMS のそれぞれ SR16000 XL Fortran，SR16000 OSCAR，BS2000 Intel Fortran，BS2000 OSCAR の逐次実行を 1.0 とした速度向上率を示している．

図 8 の系列 SR16000 oscar を見ると SR16000 では OSCAR 自動並列化コンパイラで 2 コアで 2.06 倍，4 コアで 3.63 倍，8 コアで 6.98 倍，16 コアで 14.29 倍，32 コアで 29.02 倍，64 コアで 55.20 倍，128 コアで 92.30 倍の速度向上が確認できる．同様に系列 BS2000 oscar を見ると，

表 1　評価環境

	SR16000	BS2000
CPU	POWER7	Xeon E7-8830
動作周波数	4GHz	2.13GHz
1 プロセッサあたりのコア数	8	8
L2 キャッシュ (1 コア)	256KB	256KB
L3 キャッシュ (1 プロセッサ)	32MB	24MB
搭載プロセッサ数	16	8
搭載 CPU コア数	128	64
搭載メモリ容量	1TB	256GB
コンパイラ	XL Fortran コンパイラ	Intel Fortran コンパイラ
バージョン	13.1	12.1.0

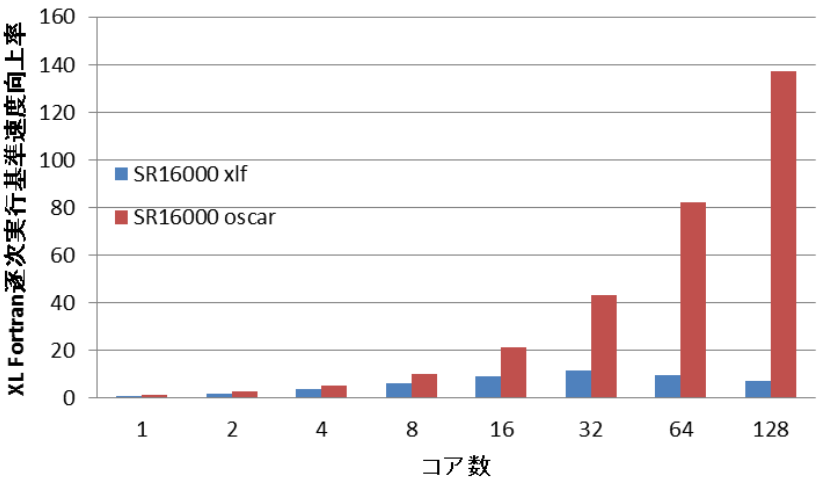

図 6　SR16000 速度向上率

BS2000 では OSCAR 自動並列化コンパイラで 2 コアで 1.86 倍，4 コアで 3.64 倍，8 コアで 7.77 倍，16 コアで 14.72 倍，32 コアで 27.73 倍，64 コアで 43.75 倍の速度向上を得た．また系列 BS2000 oscar の 8,16,32 コアの速度向上率は使用するプロセッサが分散するようにバインドした際の値となっている．なお逆に少数のプロセッサに集中してバインドすると 8 コアで 5.98 倍，16 コアで 11.72 倍，23.35 倍の速度向上留まった．これは L3 キャッシュやメモリの帯域を有効に利用できなかったためと考えられ，そのため GMS はメモリアクセスのコストが計算時間の多くを占めるものと推定される．

また，MPI で並列化した GMS [3] の同規模の問題による性能評価では 1 ノードつき 2CPU を持つノードを 16 ノード接続した PC クラスタシステムを用いて 1 コアに対して 16 コアでは 14 倍，32 コアでは 17 倍程度の速度向上率を示しているが，OSCAR コンパイラによる並列化により 16 コアにおいて SR16000 では 14.29 倍，BS2000 では 14.17 倍となりほぼ同等であり，32 コアにおいては SR16000 で 29.02 倍，BS2000 で 27.73 倍となり別の環境ではあるがより高い速度向上を自動並列化により得たことがわかる．

6. まとめ

本報告では地震動シミュレータ GMS の計算プログラム部の自動並列化コンパイラ OSCAR による高速化および Hitachi SR16000，BS2000 上での性能評価について報告した．GMS 逐次ソースコードを OSCAR コンパイラが解析・並列化しやすいように改変し、OSCAR コンパイラにより任意台数のプロセッサ用の並列プログラムを自動生成することにより，8 コアマルチコア Power7 ベース 128 コア SMP サーバ SR16000 上で 128 コアを用いて 1 コアに対して 92 倍，8 コアマルチコア Intel Xeon E7-8830 ベース外付けコヒーレントメカニズム 64 コア SMP ブレードサーバ BS2000 上で 64 コアを用いて 1 コアに対して 43 倍の速度向上をすることができた．今後は GMS の更なる性能向上とともに他の災害シミュレーションなども並列化を検討していく予定である．

7. 謝辞

本研究は株式会社日立製作所との共同研究によって行われた．貴重な評価環境をご提供いただきましたことに感謝いたします．

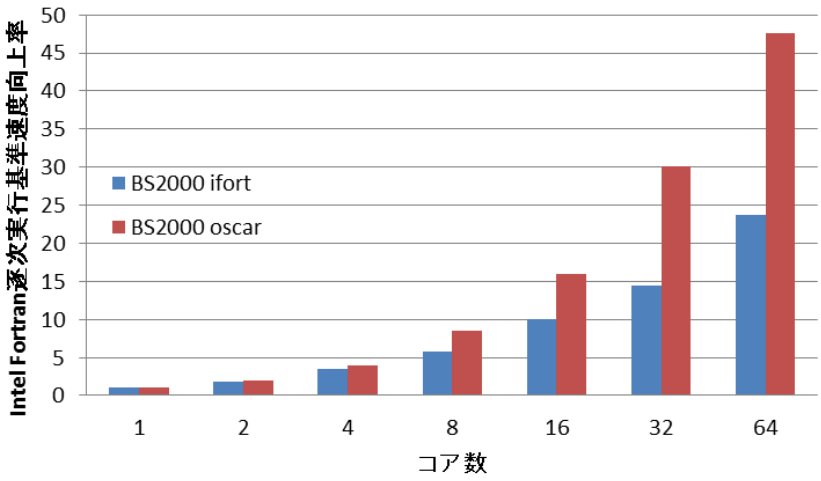

図 7　BS2000 速度向上率

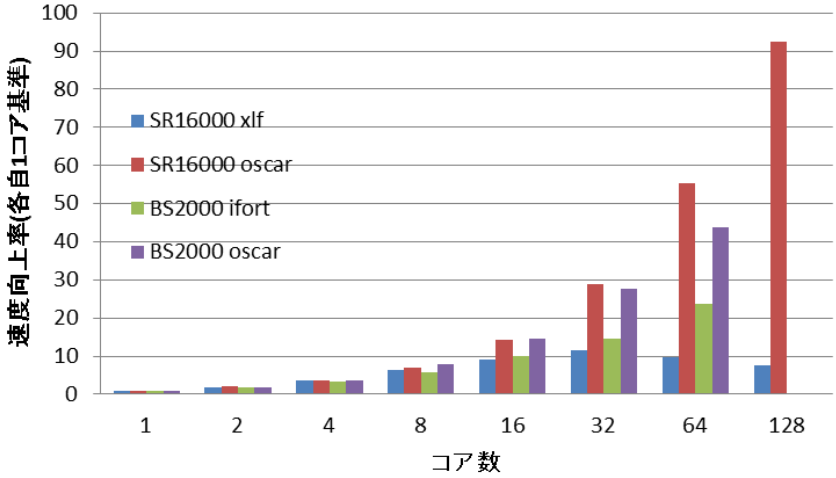

図 8　各自逐次実行を基準とした速度向上率

参考文献

[1] Akcelik, V.; Bielak, J.; Biros, G.; Epanomeritakis, I.; Fernandez, A.; Ghattas, O.; Kim, E. J.; Lopez, J.; O' Hallaron, D. R.; Tu, T.; Urbanic, J. (2003): "Highresolution forward and inverse earthquake modeling of terascale computers." In Proceedings of ACM/IEEESC2003, Phoenix, AZ.

[2] T. Furumura, L. Chen: "Parallel simulation of strong ground motions during recent and historical damaging earthquakes in Tokyo, Japan" Parallel Computing Volume 31, Issue 2, February 2005, Pages 149-65

[3] 青井 真，早川 俊彦，藤原 広行: "地震動シミュレータ:GMS", 物理探査 第 57 巻第 6 号 (2004) 651-666p

[4] NIED: "GMS ホームページ", http://www.gms.bosai.go.jp/GMS/

[5] 青井真，藤原広行: "不連続格子を用いた 4 次精度差分法による波形合成", 第 10 回日本地震工学シンポジウム論文集, Vol. 1, pp. 879-884(1998).

[6] Aoi, S. and H. Fujiwara: "3-D finite difference method using discontinuous grids", Bulletin of the Seismological Society of America, Vol. 89, pp. 918-930(1999).

[7] Dieter Wendel, Ronald Kalla, Robert Cargoni, Joachim Clables, Joshua Frierich, Roland Frech, James Kahle, Balaram Sinharoy, William Starke, Scott Taylor, Steve Weitzel, Sam G. Chu, Saiful Islam, Victor Zyuban: "The Implementation of POWER7: A Highly Parallel and Scarable Multi-Core High-end Server Processor", ISSCC 2010, pp.102-104(2010).

[8] 小幡 元樹, 白子 準, 神長 浩気, 石坂 一久, 笠原 博徳: "マルチグレイン並列処理のための階層的並列性制御手法", 情報処理学会論文誌, Vol. 44, No. 4, Apr., (2003).

[9] 本多 弘樹, 岩田 雅彦, 笠原 博徳: "Fortran プログラム粗粒度タスク間の並列性検出手法", 電子情報通信学会論文誌, Vol. J73-D-I, No. 12, pp. 951-960(1990)

[10] "OSCAR API 2.0", http://www.kasahara.elec.waseda.ac.jp/api2/regist.html