Vol.2018-HPC-165 No.36
2018/8/1

メニーコアプロセッサのための パラメータチューニング時間削減手法

岸谷 拓海[1,a)]　小松 一彦[1,b)]　撫佐 昭裕[1,c)]　佐藤 雅之[1,d)]　小林 広明[1,e)]

概要：高性能計算において性能を最大限に引き出すためには性能チューニングが必要である．しかしながら，コア数の増加や異種メモリの搭載などにより近年のプロセッサは複雑化が急速に進み，パラメータチューニングにかかる時間が増加している．本報告では，適切なパラメータを選択するパラメータチューニングのための時間を削減するための手法を提案する．津波シミュレーションを用いて評価を行い，提案手法によりパラメータチューニング時間を大幅に削減できることを明らかにした．

キーワード：自動チューニング，パラメータ探索空間削減，KNL，津波シミュレーション

Takumi Kishitani[1,a)]　Kazuhiko Komatsu[1,b)]　Akihiro Musa[1,c)]　Masayuki Sato[1,d)]
Hiroaki Kobayashi[1,e)]

1. はじめに

近年のプロセッサは，多数の演算コアを搭載することで，その演算性能を向上させてきた．また，メモリアクセス性能とメモリ容量への高い要求へ応えるべく，高いデータ転送能力を備えるメモリと，容量の大きなメモリの，異種のメモリを搭載するシステムが登場している [1], [2].

このように複雑化が進む計算システムにおいては，システムの性能を最大限に引き出すための複数のチューニングパラメータが用意されている．そのため，アプリケーションプログラム（以下，アプリケーション）を高い性能で実行するためには，計算システムとアプリケーションの両方の特徴を考慮したパラメータチューニングが必須となる．しかし，計算システムだけでなくアプリケーションも複雑化・巨大化が進み，大量のチューニングパラメータが存在する．その膨大なチューニングパラメータの組み合わせの中から適切なパラメータの組み合わせを探索するためには多くの時間が必要であり，パラメータ探索時間の削減が求められている．

[1] 東北大学
[a)] takumi.kishitani.p6@dc.tohoku.ac.jp
[b)] komatsu@tohoku.ac.jp
[c)] musa@tohoku.ac.jp
[d)] masa@tohoku.ac.jp
[e)] koba@tohoku.ac.jp

本報告では，まず，計算システムやアプリケーションの特徴を活用することで，パラメータ探索空間の削減を行う．そのために，アプリケーションを計算システムで実行する場合のボトルネックを予測し，そのボトルネックを解消するようにパラメータを選択することで，パラメータ探索空間を削減する．さらに，パラメータ探索時のアプリケーションの実行時間を可能な限り短くすることにより，パラメータチューニングにかかる時間の短縮を行う．

本報告で提案手法の評価に用いる津波シミュレーションアプリケーションは，リアルタイム津波浸水被害予報システムにおいて使用され，地震発生後いち早く被害を予測する重要な役割を担っている [3]．現在のリアルタイム津波浸水被害予測システムはスーパーコンピュータシステムに実装されているが，相補的に機能させるために，各地方自治体で津波シミュレーションを実行することが求められている．そのため，スーパーコンピュータシステムではなく，メニーコアプロセッサのような高性能プロセッサを搭載した計算システムを用いた高速な津波シミュレーションを実現することが必要であり，高速なシミュレーションを実現すらためのパラメータチューニングが重要である．

本報告の構成は以下通りである．1 節では本報告の背景について述べた．2 節ではメニーコアプロセッサにおけるシミュレーションの概要について述べる．3 節ではパラメータ探索空間の削減手法，およびパラメータチューニン

グにかかる時間を削減するための手法について提案する．4 節では提案手法によるパラメータチューニング時間の削減について津波シミュレーションを用いて評価を行う．5 節で関連研究について述べ，6 節で本報告をまとめる．

2. メニーコアプロセッサにおけるシミュレーション

本節では，まず，対象とするメニーコアプロセッサである Intel Xeon Phi Knights Landing（以下，KNL）を搭載した計算システムを取り上げ概要を述べる．

KNL は Intel Xeon Phi シリーズの第 2 世代のプロセッサであり，高い並列性，ベクトル演算，高いメモリバンド幅を提供するプロセッサである [1], [2].

KNL はタイルと呼ばれるユニットを複数接続し構成される．図 1 にタイルの概要を示す．各タイルには 2 つのコアが搭載され，2 つのコアは 1 MB の L2 キャッシュを共有している．またそれぞれのコアは 2 つのベクトル演算ユニット (VPU) を搭載しており，32 の単精度浮動小数点演算または 16 の倍精度浮動小数点演算を同時に実行することが可能である．さらに各コアはハイパースレッディング技術により最大で 4 スレッドを実行することが可能である．

図 2 に KNL のアーキテクチャの概要を示す．KNL は 38 のタイルを 2 次元メッシュ状に接続された構成である．38 のタイルのうち 2 つのタイルは製造の歩留まり改善のために用意されており，出荷時に無効化される．そのため KNL は最大で 36 タイル，72 コアを利用することができる．また，ハイパースレッディング技術により，最大で 288 スレッドを並列実行することが可能である．以上の計算資源を有効活用することにより，KNL は単精度で 6.9 Tflop/s，倍精度で 3.45 Tflop/s という高い性能を実現している．

また，KNL には高バンド幅メモリの MCDRAM と，MCDRAM より低メモリバンド幅であるが大容量である DDR メモリの，2 種類の特徴の異なるメモリが搭載されている．高バンド幅メモリの MCDRAM の利用法として，プログラマが明示的に MCDRAM へのデータ配置を指定する方法と，ハードウェアにデータの管理を任せる方法がある．今日のアプリケーションはメモリバンド幅性能により実効性能が制限されていることが多く，メモリバンド幅性能の高い MCDRAM を有効に活用することは重要である．

多数のコアや複数種のメモリが搭載されている KNL の高い性能を最大限活用するためには，多数のコア及び VPU を有効利用するための最適化が重要であり，システムパラメータとしては，クラスターモード，メモリモードなど，そして利用法としてはスレッドアフィニティ，スレッド数など，各パラメータを状況に応じて適切な値に設定する必要がある．

クラスターモードは，all-to-all，Hemisphere，Quadrant，SNC(Sub Numa Cluster)-2，SNC-4 の 5 種類のモードがある．all-to-all，Hemisphere，Quadrant モードでは，メモリは UMA（Uniform Memory Access）として扱われる．これら 3 つのモードにおける違いは仮想的な領域に分割される領域数である．all-to-all モードの場合，仮想的な分割を行わず，チップ全体が 1 つの領域として扱われる．Hemisphere モードと Quadrant モードの場合は，メモリと L2 キャッシュのキャッシュタグを保存する L2 キャッシュタグディレクトリが仮想的に分割される．Hemisphere モードの場合はメモリと L2 キャッシュタグディレクトリを 2 つの領域に，Quadrant モードの場合は 4 つの領域に，それぞれ仮想的に分割される．この時，L2 キャッシュのキャッシュタグは，そのデータが配置されているメモリと同じ領域にある L2 キャッシュタグディレクトリに配置される．これにより，L2 キャッシュミス時のメモリアクセスを局所化することができ，メモリアクセスレイテンシを軽減することができる．

SNC モードの場合，メモリと L2 キャッシュタグディレクトリに加えて，コアも仮想的な領域に分割される．分割された領域はそれぞれ，NUMA（Non Uniform Memory Access）ノードとして使用される．NUMA ノードとして使用する場合，メモリとコアの距離が小さくなるため，UMA に比べてさらにメモリアクセスにかかるレイテンシを削減することが可能である．しかし，ほかの NUMA ノードに

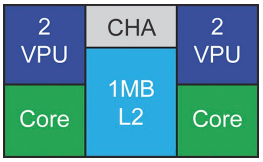

図 1　KNL のタイルの概要図

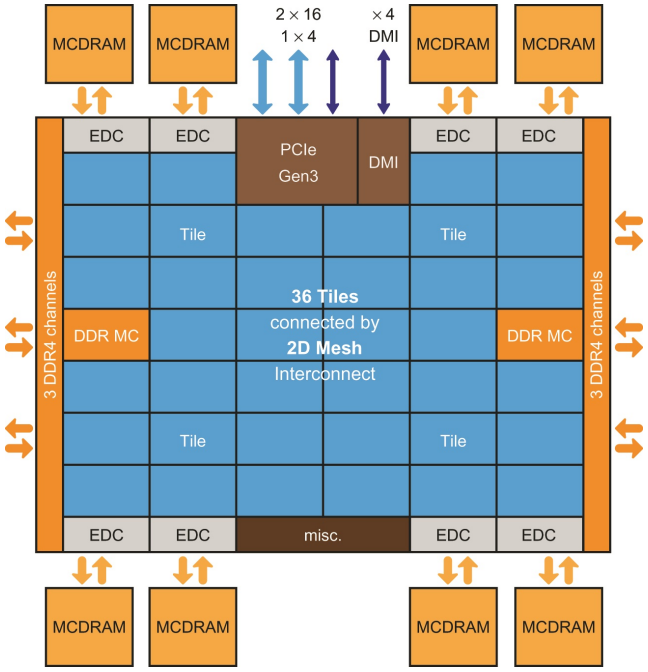

図 2　KNL のアーキテクチャの概要図

アクセスする際はアクセスレイテンシが大きくなるため，NUMA 向けの最適化を行うことが求められる．

メモリモードは，Flat モード，Cache モード，Hybrid モードの 3 種類がある．Flat モードは MCDRAM を使用する場合と使用しない場合の 2 通り，Hybrid モードは Cache 部分が 25%または 50%の 2 通りがあるため，合計で 5 通りのメモリモードがある．Flat モードでは MCDRAM と DDR メモリは同じメモリ空間空間を持つ．Flat モードではライブラリやユーティリティを用いて MCDRAM の利用を明示的に指示する必要がある．また，アプリケーションが MCDRAM の容量を超えるデータ容量を要求する場合は，再利用性の高いデータを明示的に MCDRAM に配置するように，アプリケーションコード内で指定する必要がある．Cache モードでは MCDRAM は DDR メモリのキャッシュメモリとして利用される．DDR メモリへのアクセスは MCDRAM キャッシュにおいてキャッシュミスが生じた場合のみ行われる．MCDRAM キャッシュはダイレクトマップで管理され，64 バイトのキャッシュラインを持つ．Cache モードにおける MCDRAM はハードウェアにより管理されるため，ソースコードの変更やツールなどを利用する必要が無く，アプリケーションを実行するだけでよい．Hybrid モードでは MCDRAM の容量の 25 %もしくは 50 %を Cache モード，残りの部分を Flat モードと分割して使用することが可能である．アプリケーションが Flat 部分の MCDRAM の容量を超えるデータ容量を要求する場合は，Flat モードの場合と同様に，再利用性の高いデータを MCDRAM に配置するように，アプリケーションコード内で明示的に指示する必要がある．

スレッドアフィニティは，Compact，Scatter，Balanced の 3 種類がある．Compact に設定した場合，なるべく少ないコア数でアプリケーションを実行するようにスレッドが割り当てられる．そのため，1 つのコアに連続した 4 つのスレッドが割り当てられ，キャッシュヒット率が向上する可能性がある．これに加え，複数スレッドで VPU などの計算資源を共有できるため，計算資源を最大限に活用することができる．Scatter，Balanced に設定した場合，なるべく全てのコアを均等に使用するようにスレッドが割り当てられるため，少ないスレッド数で多くのコアを使用することが可能である．

これらのパラメータの中から適切なパラメータの組み合わせを見つけるためには，300 ものパラメータの組み合わせを探索する必要があり，全てのパラメータの組み合わせを探索するためには多くの時間が必要である．そのため，計算システムや使用するアプリケーションの特徴を考慮してチューニングパラメータの候補を絞ることで，パラメータチューニングにかかる時間を削減する必要がある．

3. パラメータチューニング時間削減手法

適切なパラメータの組み合わせを見つける最も確実な方法は，すべてのパラメータの組み合わせを探索することである．しかし，KNL には多くのパラメータがあり，それらの組み合わせは 300 通りもある．一般的に，シミュレーションの実行時間は長いため，全探索を行うことは難しい．そのため，パラメータの探索空間を削減し，且つ，1 パラメータあたりの実行時間を削減する必要がある．

本報告では，計算システム及びアプリケーションの特徴を利用して，パラメータ探索空間を絞り込むことで，適切なパラメータの組み合わせを探索する時間を削減する．まず，計算システムの演算性能とメモリ性能の比であるシステム Bytes/Flop と，アプリケーションコードにおける演算回数に対するメモリからのデータ転送量の比であるコード Bytes/Flop を比較することでボトルネックを予測する．そのボトルネックを解消するようなパラメータを選択候補とすることで，パラメータ探索空間を削減する．

次に，アプリケーションの特徴が変わらないような小さな入力データと，短縮したシミュレーション時間を用いて，1 つのパラメータの組み合わせあたりの実行時間を短縮する．アプリケーションの特徴が大きく変化しない程度の解像度の低い入力データを用いてパラメータの探索を行う．さらに，パラメータ探索時には，可能な限りシミュレーションの再現時間を短く設定し，適切なパラメータを探索する．これによって，チューニングパラメータの探索にかかる時間を削減する．

3.1 KNL のパラメータ選択方法

本節では，KNL におけるチューニングパラメータを絞り込む方法について，詳細を述べる．提案手法では，まず，コード Bytes/Flop とシステム Bytes/Flop を比較することによって，アプリケーションがメモリ律速か演算律速かを判断する．コード Bytes/Flop がシステム Bytes/Flop よりも大きい場合はメモリ律速であると判断し，メモリ性能を引き出すことができるようにパラメータの選択を行う．一方，コード Bytes/Flop がシステム Bytes/Flop よりも小さい場合は演算律速であると判断し，演算性能を引き出すことができるパラメータを選択する．

3.1.1 アプリケーションがメモリ律速の場合

アプリケーションがメモリ律速である場合は，メモリ性能が実行時間に大きな影響を及ぼすため，メモリ性能への影響が大きい順にパラメータの選択を行う．メモリ性能への影響は，メモリ性能を測定する Stream ベンチマークを用いて判断する．

表 1 に，クラスターモード，メモリモード，スレッドアフィニティ，スレッド数のそれぞれのパラメータを変化

表 1　各パラメータの最高メモリバンド幅及び最低メモリバンド幅 (GB/s)

パラメータ	最高メモリバンド幅	最低メモリバンド幅	最大メモリバンド幅差
クラスターモード	446.31	247.83	198.48
メモリモード	446.31	73.10	373.21
スレッドアフィニティ	446.57	164.71	281.86
スレッド数	446.31	348.48	97.83

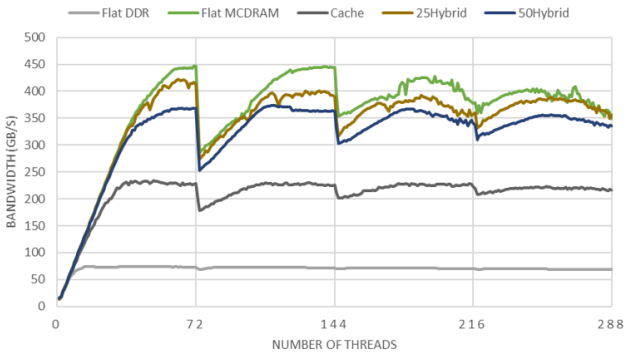

図 3　メモリモードの違いによるメモリ性能

させたときの最高メモリバンド幅，最低メモリバンド幅，またそれらの差を示す．表 1 を見ると，メモリモード，スレッドアフィニティ，クラスターモード，スレッド数の順にメモリバンド幅の差が大きいことが分かる．ただし，スレッドアフィニティはコアへのスレッドの割り当て方を設定するパラメータであり，スレッド数と密接な関係にあるため，スレッドアフィニティとスレッド数の選択は同時に行う．そのため，メモリモード，スレッドアフィニティとスレッド数，クラスターモードの順にパラメータの選択を行う．

Step1 メモリモードの選択

図 3 にクラスターモードを Quadrant，スレッドアフィニティを Balanced に固定し，それぞれのメモリモードに対しスレッド数を変化させたときのメモリバンド幅を示す．Stream ベンチマークにおける要求データ容量は約 13 GB とした．縦軸はメモリバンド幅，横軸はスレッド数を示す．図 3 を見ると，Flat モードにおいて MCDRAM を用いた場合にメモリバンド幅が最も高いことが分かる．また，メモリモードを Cache モードに設定した場合でも，MCDRAM を活用することができるが，キャッシュデータの管理によるオーバーヘッドが生じるため，Flat モードよりもメモリバンド幅が低下する．

メモリモード選択の指標となるのが，アプリケーションが要求するメモリ容量である．要求データ量が 16 GB よりも小さい場合，全てのデータを MCDRAM 内に配置することが可能であるため，Flat モードを選択する．したがって，要求データ量が 16 GB よりも小さい場合は，5 つの候補から 1 つに絞り込むことが可能である．

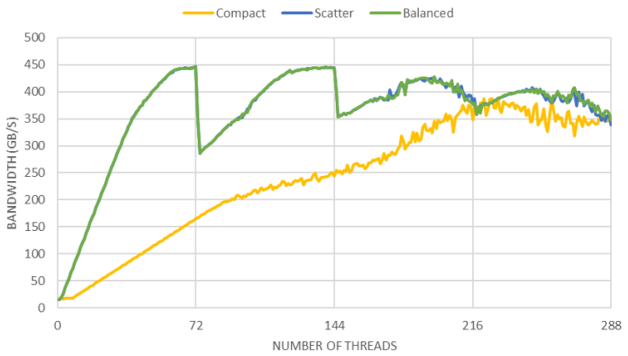

図 4　スレッドアフィニティの違いによるメモリ性能

一方，アプリケーションによる要求データ量が 16 GB よりも大きい場合は，全てのデータを MCDRAM に配置できないため，Cache モードを選択する．また，Flat モードや Hybrid モードは，再利用性の高いデータをコード中で明示的に指示する必要があるため，コードを変更せずにチューニングを行う場合は，候補から外れる．以上から，アプリケーションが要求するデータ量が 16 GB よりも大きい場合のメモリモードの候補は，Cache モードのみとなる．これにより，要求データ容量が 16 GB よりも大きい場合のメモリモードは Cache モードのみとなり，5 つの候補から 1 つに絞り込むことができる．

Step2 スレッドアフィニティとスレッド数の選択

図 4 にクラスターモードを Quadrant，メモリモードを Flat モードに固定し，それぞれのスレッドアフィニティに対してスレッド数を変化させたときのメモリバンド幅を示す．図 4 を見ると，スレッドアフィニティが Compact の場合とその他の場合を比較すると，Compact の場合のメモリバンド幅が低いことがわかる．これは，スレッド数が少ない時はすべてのコアを利用できておらず，メモリバンド幅を十分に引き出せていないためであり，また，スレッド数が多い時は，各コア内に複数スレッドが割り当てられることで，オーバーヘッドが大きくなるためである．そのため，アプリケーションがメモリ律速の場合は，Compact は候補から外れ，スレッドアフィニティは Scatter と Balanced が候補となる．これにより，3 つの候補から 2 つに絞り込むことができる．

また，KNL の性能を引き出すためには，すべてのコアを利用することが重要である．そのため，スレッドアフィニティが Scatter と Balanced の場合のスレッド数は，72，144，216，288 の 4 つが候補となる．図 4 を見ると，Scatter と Balanced のメモリバンド幅は，スレッド数が 72，または 144 の時に最も高いことがわかる．これは，スレッド数が少ない場合であっても，すべてのコアを利用でき，メモリバンド幅を十分に引き出せるためである．また，スレッド数が多い時はメモリバンド幅が低下していることわかる．これは，Compact の場合と同様に，各コア内に複数ス

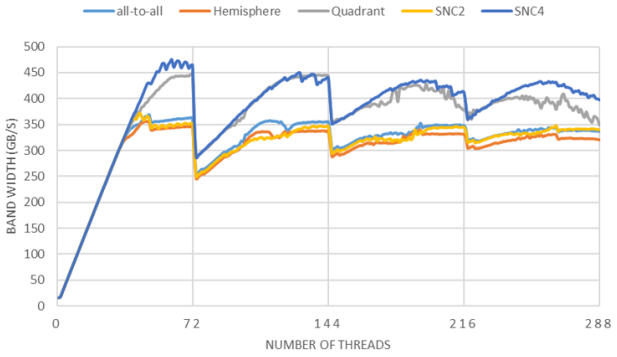

図 5　クラスターモードの違いによるメモリ性能

レッドが割り当てられることで，オーバーヘッドが大きくなるためである．そのため，アプリケーションがメモリ律速の場合のスレッド数の候補は，72 または，144 となる．これにより，4 つの候補から 2 つに絞り込むことができる．

Step3 クラスターモードの選択

図 5 にメモリモードを Flat モード，スレッドアフィニティを Balanced に固定し，それぞれのクラスターモードに対してスレッド数を変化させたときのメモリ性能を示す．図 5 より，SNC-4 に設定したときのメモリバンド幅が最も高いことが分かる．また，メモリが UMA として扱われるクラスターモードの中では，Quadrant が最もメモリバンド幅が高いことが分かる．これは，メモリアクセスレイテンシが他のクラスターモードに比べて小さいためである．

クラスターモードを選択する上で，まず NUMA 向けの最適化が行われているか否かが重要である．NUMA 向けの最適化が施されている場合は，最もメモリバンド幅が高い SNC-4 を選択する．NUMA 向けの最適化が行われていない場合のクラスターモードは，メモリを UMA として扱うクラスターモードの中で最もメモリバンド幅が高い Quadrant を選択する．NUMA 向け最適化が行われていない場合に SNC モードを選択すると，ほかの NUMA ノードへのメモリアクセスが頻繁に発生し，性能が低下してしまうためである．これにより，NUMA 向けの最適化が行われている場合と，行われていない場合のいずれにおいても，5 つの候補から 1 つに絞り込むことができる．

3.1.2　アプリケーションが演算律速である場合

アプリケーションが演算律速である場合，演算性能が実行時間に大きな影響を及ぼす．そのため，演算性能を測定できる演算律速なベンチマークの General Matrix Multiply（GEMM）ベンチマークや，High Performance Linpack ベンチマーク（HPL）の実行結果をパラメータ選択の判断基準として使用する．メモリ律速の場合と異なるのは，スレッドアフィニティの選択とスレッド数の選択である．アプリケーションが演算律速である場合，演算器をすべて使用したほうが実行時間が短くなる．そのため，すべての演算器を有効に利用できるスレッド数と，スレッドアフィニティを選択する必要がある．したがって，アプリケーションが演算律速である場合のスレッド数は 288 を選択する．これにより，すべてのスレッドを利用することが可能である．また，スレッドアフィニティは，Comapct を選択する．これは，スレッドアフィニティを Comapct に設定することで，各コアに割り当てられるスレッド番号が連続となり，キャッシュヒット率が向上するためである．以上により，アプリケーションが演算律速である場合は，スレッドアフィニティの候補は Comapct，スレッド数の候補は 288 となる．

3.2　シミュレーションのパラメータによるチューニング時間削減

本節では，アプリケーションの特徴を大きく変化させないように，実行時間の短くなるような入力データの利用，シミュレーション時間の調整を行うことにより，アプリケーションの実行時間を短縮し，チューニング時間の削減を試みる．実行時間が短くなるような解像度の低い入力データを用いて，アプリケーションの特性が大きく変化しない程度の短い再現時間でパラメータ探索行う．これによって，パラメータチューニングにかかる時間を削減することができる．

また，シミュレーションの再現時間を短くしてもコード実行中の特性が大きく変わらないことに着目する．アプリケーションの特性が変化しない程度まで再現時間を短く設定し，パラメータ探索を行う．これにより，短い再現時間でパラメータ探索ができるため，探索にかかる時間を削減することができる．実行時間の短くなるような解像度の低い入力データを用いる場合，本来の入力データを用いて実行した場合と特性が変化しない必要がある．そのため，提案手法においては，本来実行対象となっている入力データの縮小版のデータを採用する．

最小の再現時間の探索方法は，二分探索により行う．シミュレーションの再現時間とシミュレーションの実行時間はほぼ比例する傾向にある．そのため，再現時間とシミュレーションの実行時間が比例している間は，アプリケーション実行時の特徴がほとんど変化していないと考えることができる．したがって，アプリケーションの特徴が変化していないとみなせる間は再現時間を短くする．

探索に関する疑似コードを Algorithm1 に示す．まず，Step1 から Step4 で絞り込んだパラメータの中から，ある組み合わせ $PSET$ を 1 つ選択する．選択したパラメータの組み合わせに関して，解像度の低い入力データを用いて，最小の再現時間の探索を行う．ここでアルゴリズム中の df は再現時間とシミュレーションの実行時間が比例しているとみなせる許容値である．探索の終了条件は，再現時間とシミュレーション実行時間が比例しているとみなせなくなった場合，または，実行時間の変化が閾値 rf よりも小さ

Algorithm 1 再現時間の探索アルゴリズム

1: $PSET \leftarrow$ パラメータの組み合わせ
2: $PSET$ に関する $KLSEC_{original}$ における実行時間を調査
3: $KLSEC_{opt} \leftarrow KLSEC_{original}$
4: $KLSEC_1 \leftarrow KLSEC_{orginal}$
5: $i \leftarrow 1$
6: **repeat**
7: $i \leftarrow i+1$
8: $KLSEC_i \leftarrow KLSEC_{i-1}/2$
9: $PSET$ に関する $KLSEC_i$ における実行時間を調査
10: **if** $\frac{ET_{PSET}(KLSEC_i)}{ET_{PSET}(KLSEC_{i-1})} \geq (0.5+df)$ **then**
11: **exit**
12: **end if**
13: $KLSEC_{opt} \leftarrow KLSEC_i$
14: **until** $|ET_{PSET}(KLSEC_i) - ET_{PSET}(KLSEC_{i-1})| < rf$

くなった場合である．探索が終了した時点で，$KLSEC_{opt}$ に格納されている値が，最小の再現時間である．

最後に探索によって得られた最小の再現時間と，解像度の低い入力データを用いて，3.1 節で絞り込まれたすべてのパラメータの組み合わせに対して探索を行う．これにより得られた最適なパラメータの組み合わせを，本来実行すべき入力データを用いた時のシミュレーションにおける適切なパラメータの組み合わせとして採用する．

4. 評価

本節では，提案するパラメータチューニング時間の削減手法の評価を行う．

4.1 評価環境

メニーコアプロセッサとして，表 2 に示す Intel Xeon Phi 7290B を使用する．Intel Xeon Phi 7290B の単精度理論演算性能は 6.912 TFlop/s であり，MCDRAM の Stream メモリバンド幅が 450 GB/s であることから，システム Bytes/Flop は 0.065 である．

評価アプリケーションとして津波浸水被害予測シミュレーション（以下，津波シミュレーション）を用いる．津波シミュレーションは，地震が発生した際の地震データから断層モデルを予測し，予測した断層モデルを使用して津波による浸水被害を予測する [3]．津波シミュレーションでは津波浸水被害予測として，津波が到達する時間，津波による最大浸水深，浸水開始時間，津波水位の変化を計算する．

津波をシミュレーションする手順として，津波の原因となる地盤変動量を計算し，その結果をもとに浸水計算が行われる．地盤変動量は Okada の方法 [4] を用いて地表の変位計算を行うことで求める．浸水計算は，Okada の方法によって求めた地盤変動量に基づいて，TUNAMI(Tohoku University's Numerical Analsis Model for Investigating Tsunami) モデル [5] を用いて行われる．

TUNAMI モデルでは，Staggerd Leap-frog 有限差分法を用いた数値計算法により非線形浅水方程式を解くことで，津波のシミュレーションを行う．この方法は，UNESCO（United Nations Educational, Scientific and Cultural Organization）に浸水予測の基礎方法として採用されている [6]．支配方程式は以下の式で与えられる．

$$\frac{\partial \eta}{\partial t} + \frac{\partial M}{\partial x} + \frac{\partial N}{\partial y} = 0, \quad (1)$$

$$\frac{\partial M}{\partial t} + \frac{\partial}{\partial x}\left(\frac{M^2}{D}\right) + \frac{\partial}{\partial y}\left(\frac{MN}{D}\right) + gD\frac{\partial \eta}{\partial x} + \frac{gn^2 M}{D^{7/3}}\sqrt{M^2+N^2} = 0, \quad (2)$$

$$\frac{\partial N}{\partial t} + \frac{\partial}{\partial x}\left(\frac{MN}{D}\right) + \frac{\partial}{\partial y}\left(\frac{N^2}{D}\right) + gD\frac{\partial \eta}{\partial y} + \frac{gn^2 N}{D^{7/3}}\sqrt{M^2+N^2} = 0, \quad (3)$$

$$g + \frac{1}{\rho}\frac{\partial P}{\partial z} = 0. \quad (4)$$

ここで，η は垂直方向の水面変位，g は重力加速度，D は全水深，n は Manning の粗度係数，P は静水圧，ρ は水密度，M 及び N は x 方向，y 方向のそれぞれにおける流量フラックスである．

津波シミュレーションにおいて実行時間に大きな影響を与えるパラメータとして，シミュレーションを行う地域を決める地形データと，津波を再現する時間が挙げられる．領域の構造が複雑であったり，グリッドサイズが小さく解像度の高い地形データを用いた場合，アプリケーションによる要求データ量は大きくなり，実行時間が長くなる．そのため，解像度の低い地形データを用いて実行した場合は実行時間が短く，解像度の高い地形データを用いて実行した場合は実行時間が長くなる傾向がある．また，津波を再現する時間を長くすることで演算量の増加にともない実行時間が長くなる．そのため，パラメータチューニング時間を削減するために，これらのパラメータを考慮することは重要である．

シミュレーションの実行には，高知県沿岸の地形データを用いる．計算領域は，810 m，270 m，90 m，30 m，10 m グリッドの 5 階層からなる．また，高知県沿岸の地形データを使用したときのアプリケーションによる要求データ量は約 15 GB である．軽量版の高知県沿岸の地形データの計算領域は，810m，270m，80m，30m グリッドの 4 階層からなる．軽量版の高知県沿岸の地形データを使用したときのアプリケーションによる要求データ量は約 2 GB である．また，津波シミュレーションのコード Bytes/Flop は約 1.85 である．津波シミュレーションを実行する際の津波再現時間は 1 時間（3600 秒）とする．

4.2 チューニングパラメータの選択

津波シミュレーションのコード Bytes/Flop が 1.85 であり，KNL のシステム Bytes/Flop の 0.065 よりも大きいことから，津波シミュレーションはメモリ律速なアプリケーションであると判断できる．そのため，メモリ律速なアプ

表 2　評価環境

Processor	Intel Xeon Phi 7290B
Peak Performance	6.912 TFlop/s (SP)
Memory system	MCDRAM and DDR4-2400
MCDRAM Capacity	16 GB
MCDRAM Bandwidth	450 GB/s
DDR Capacity	96 GB
DDR Bandwisth	115.2 GB/s
OS	CentOS 7.4.1708
Compiler	Intel Fortran Compiler 18.0.2

表 3　津波再現時間に関する調査結果

試行回数	津波再現時間	df_{PSET}	rf_{PSET}
1	1800	0.51	120.0
2	900	0.52	59.7
3	450	0.54	30.0
4	225	0.57	15.0

リケーションに対するチューニングパラメータの選択を行う．Step1 より，要求メモリ容量が 16 GB 以下であるため，メモリモードは Flat モードに絞り込むことができる．これにより，5 つの候補から 1 つに絞り込むことが可能である．次に，Step2 より，スレッドアフィニティは 3 つから Scatter，Balanced の 2 つに絞ることが可能である．Step3 より，クラスターモードは 5 つから，Quadrant に絞り込こめる．最後に Step4 より，スレッド数は 4 通りから 72, 144 スレッドの 2 つに絞り込める．以上より，提案手法により全パラメータの組み合わせ 300 通りから，4 通りに絞り込むことができる．

次に津波再現時間に関する探索を行う．解像度の低い地形データとして，高知県沿岸の軽量版の地形データを用いる．探索における閾値の設定は，アプリケーションの特徴変化を決める閾値 df は 0.05，終了条件の閾値 rf は 10 とした．表 3 に提案手法による，最小の津波再現時間の探索結果を示す．表中の df_{PSET} は，$KLSEC_i$ を用いた場合の $PSET$ の実行時間 $ET_{PSET}(KLSEC_i)$ と，$KLSEC_{i-1}$ を用いた場合の $PSET$ の実行時間 $ET_{PSET}(KLSEC_{i-1})$ の比を表す．また，rf_{PSET} は，$ET_{PSET}(KLSEC_i)$ と $ET_{i-1}(KLSEC_{i-1})$ の差を表す．試行回数 4 回目で $rate_{PSET}$ が閾値を超えているため，津波再現時間が 225 秒でアプリケーションの特徴が変化したと考えられる．そのため提案手法により，津波シミュレーションの特徴が変化しない最小の津波再現時間が 450 秒であることがわかった．

最後に，求めた最小の津波再現時間を用いて，適切なパラメータの組み合わせを探索する．Step1 から Step4 で絞り込みを行った 4 つパラメータの組み合わせに対して，津波再現時間を 450 秒に設定し，軽量版の高知県沿岸の地形データを用いて適切なパラメータの組み合わせを評価する．表 4 に 4 つのパラメータの組み合わせに関して，実行時間の短い順に並べたものを示す．表 4 を見ると，クラスター

表 4　提案手法により得られた最適なパラメータの組み合わせ

Rank	Cluster Mode	Memory Mode	Affinity	Threads
1	Quadrant	FlatMCDRAM	Balanced	144
2	Quadrant	FlatMCDRAM	Scatter	72
3	Quadrant	FlatMCDRAM	Scatter	144
4	Quadrant	FlatMCDRAM	Balanced	72

表 5　実際の実行時間の短い上位のパラメータの組み合わせ

Rank	Cluster Mode	Memory Mode	Affinity	Threads
1	Quadrant	FlatMCDRAM	Balanced	144
2	Quadrant	FlatMCDRAM	Scatter	144
3	Quadrant	Cache	Balanced	144
4	Quadrant	FlatMCDRAM	Balanced	72
5	all-to-all	FlatMCDRAM	Balanced	144

モードは Quadrant，メモリモードは Flat モード，スレッドアフィニティは Balanced，スレッド数は 144 の組み合わせが適切なパラメータの組み合わせであることがわかる．

選択されたパラメータの組み合わせが実際に適切なパラメータの組み合わせであるか評価を行う．表 5 に高知県沿岸の地形データを用いて，津波再現時間を 3600 秒に設定し，全パラメータの組み合わせにおいて調査を行ったときの，実行時間が短い上位 5 組の組み合わせを示す．表 5 を見ると，提案手法により選択したパラメータの組み合わせが，実際に最適なパラメータの組み合わせであることがわかる．

4.3　パラメータチューニング時間削減に関する評価

本節では，パラメータチューニング時間をどれだけ削減できたか評価する．本来実行すべき地形データを用いて全パラメータの組み合わせに関して探索した場合の合計実行時間と，提案手法を用いて適切なパラメータの組み合わせに到達するまでに実行した合計時間を比較する．本来実行すべき地形データを用いて全パラメータの組み合わせを探索した場合の合計実行時間は 7277 分 35.52 秒であった．また，提案手法を用いて適切なパラメータの組み合わせに到達するまでにかかった総実行時間は 9 分 56.08 秒であった．したがって，全てのパラメータの組み合わせを探索するよりも約 120 時間のパラメータチューニング時間を削減することができることがわかる．

5.　関連研究

メニーコアプロセッサにおいて性能を引き出すために，数々の自動チューニングを用いた試みがなされている．片桐らは，KNL を用いて MPI と OpenMP のハイブリッド実行における自動チューニングの性能を評価している [7]．本提案手法では，単一プロセスでのプログラム実行を対象としていたが，片桐らの手法と組み合わせることで，ハイブリッド実行においても自動チューニングを行うことが可

能だと考えられる．

6. おわりに

メニーコアプロセッサや異種メモリを搭載したコンピュータの登場により，計算システムは複雑になっている．計算システムの複雑化に伴い，チューニングパラメータの数も膨大になり，適切なパラメータの組み合わせを探索するために多くの時間が必要である．さらに，科学技術計算アプリケーションの大規模化・高度化により，アプリケーションの実行時間の長時間が進んでいる．そのため，適切なパラメータの組み合わせを探索するためにかかる時間が膨大となっている．

本報告では，計算システムやアプリケーションの特徴を考慮したチューニング時間の削減を行った．これは，システム Bytes/Flop とアプリケーションのコード Bytes/Flop を比較することで，アプリケーションの特徴を予測し，計算システムにおけるパラメータ探索空間の削減を行うものである．さらに，アプリケーションの特性を変えない範囲で，入力データやシミュレーション時間を調整することで，1 パラメータあたりの実行時間を削減も見込める．KNL を搭載した計算システムと津波シミュレーションを用いて，提案するチューニングパラメータ削減手法の評価を行ったところ，アプリケーションのボトルネックを解消するようなパラメータを選択することにより，パラメータの組み合わせ数を 300 から 4 つに削減できることがわかった．また，本来実行すべきデータよりも解像度の低い地形データの活用と，津波再現時間を短くすることにより，1 パラメータの組み合わせを調べる実行時間を短くした．これによって，すべてのパラメータの組み合わせに関して調査するよりも，約 120 時間のパラメータチューニング時間を削減することが明らかとなった．

今後の課題として，以下のような点が挙げられる．本報告では，メモリ律速なアプリケーションである津波シミュレーションの評価のみを行った．そのため，本提案手法が演算律速なアプリケーションにおいて有効であることを今後評価する必要がある．また，提案手法のパラメータ探索は全範囲を対象としていたが，これを避ける他の既存手法と組み合わせた場合についても評価が必要である．

7. 謝辞

本研究の一部は大規模情報基盤共同利用・共同研究拠点の公募型共同研究（課題番号: jh180040），科研費基盤研究 S（課題番号: 17H06108）の支援を受けている．

参考文献

[1] Sodani, A., Gramunt, R., Corbal, J., Kim, H. S., Vinod, K., Chinthamani, S., Hutsell, S., Agarwal, R. and Liu, Y. C.: Knights Landing: Second-Generation Intel Xeon Phi Product, *IEEE Micro*, Vol. 36, No. 2, pp. 34–46 (online), DOI: 10.1109/MM.2016.25 (2016).

[2] Jeffers, J., Reinders, J. and Sodani, A.: *Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition*, Elsevier Science (2016).

[3] Musa, A., Watanabe, O., Matsuoka, H., Hokari, H., Inoue, T., Murashima, Y., Ohta, Y., Hino, R., Koshimura, S. and Kobayashi, H.: Real-time tsunami inundation forecast system for tsunami disaster prevention and mitigation, *The Journal of Supercomputing*, (online), DOI: 10.1007/s11227-018-2363-0 (2018).

[4] Okada, Y.: Surface deformation due to shear and tensile faults in a half-space, *Bulletin of the Seismological Society of America*, Vol. 75, No. 4, p. 1135 (online), available from ⟨http://dx.doi.org/⟩ (1985).

[5] Koshimura, S., Oie, T., Yanagisawa, H. and Imamura, F.: Developing Fragility Functions For Tsunami Damage Estimation Using Numerical Model and Post-Tsunami Data From Banda Aceh, Indonesia, *Coastal Engineering Journal*, Vol. 51, No. 03, pp. 243–273 (online), DOI: 10.1142/S0578563409002004 (2009).

[6] Goto, C., Ogawa, Y., Shuto, N. and Imamura, F.: Numerical Method of Tsunami Simulation with the Leap-frog Scheme, online, Intergorvernmental Oceanographic Commission of UNESCO, http://unesdoc.unesco.org/images/0012/001223/122367eb.pdf (1997).

[7] Katagiri, T., Ohshima, S. and Matsumoto, M.: Auto-Tuning on NUMA and Many-Core Environments with an FDM Code, *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 1399–1407 (online), DOI: 10.1109/IPDPSW.2017.27 (2017).