ヘテロジニアス分散システムにおける有限差分法

この記事は、インテル® デベロッパー・ゾーンに公開されている「Finite Differences on Heterogeneous Distributed Systems」の日本語参考訳です。

ソースコードのダウンロード(zip ファイル)

ここでは、有限差分法 (FD) の計算カーネルを分散システムで実行する方法を説明します。また、異なるノードや計算デバイスにより計算速度を大幅に向上できる可能性があるヘテロジニアス分散システムにおいて、ロード・インバランスに取り組むための手法を説明します。実装例としてサンプルコードも提供しています。

ここで使用するビルディング・ブロックは、一般に地震探査の RTM (Reverse Time Migration: リバース・タイム・マイグレーション) アルゴリズムで使用される FD 計算カーネルです。ISO-3DFD (等方性 3 次元有限差分) ステンシルにより実行される計算は、石油・天然ガスの探査において複雑な地下構造を正確に描画する上で重要な役割を果たします。ここでは、[1] と [2] で述べられている ISO-3DFD を利用し、インテル® Xeon® プロセッサー・ベースのホストとインテル® Xeon Phi™ コプロセッサーのハイブリッド・ハードウェア構成で分散 ISO-3DFD 計算カーネルを実行する簡単な MPI ベースの分散実装を導入しています。また、ロードバランスを解析し、パフォーマンスとスケーラビリティーを向上するのに役立つインテル® ソフトウェア・ツールについても取り上げます。

分散 ISO-3DFD

ここでは、ISO-3DFD 計算ステンシルの 1 次元分解を使用します。計算ドメインは MPI プロセス全体に分割されます。計算デバイス (インテル® Xeon® プロセッサーまたはインテル® Xeon Phi™ コプロセッサー) につき 1 つの MPI プロセスとします。この実装には、MPI プロセス間 (つまり、プロセッサーとコプロセッサー間) のハロ交換も含まれます。[1, 2] の説明のように FD ステンシルにドメイン分割が適用される場合、アルゴリズムの各タイムステップでサブドメイン間のハロ交換を実装する必要があります。これは、サブドメインの境界付近でドメインが値を更新する際に、隣接するサブドメインで値を計算する必要があるためです。

```
for(int k=1; k<=HL; k++)  // ステンシルの半分の長さ
    u_0 += W[k]*(
    U0(ix+k,iy  ,iz  ) + U0(ix-k,iy  ,iz  ) +
    U0(ix  ,iy+k,iz  ) + U0(ix  ,iy-k,iz  ) +
    U0(ix  ,iy  ,iz+k) + U0(ix  ,iy  ,iz-k));
```

3D ステンシルの次数は、その半分の長さ (*HL*) になります。例えば、8 番目のステンシルでは *HL=8/2=4* になります。隣接するサブドメイン間で交換されるハロの "幅" も *HL* と等しくなります。

このサンプルコードは対称モデルを使用します。つまり、ホスト・プロセッサーとコプロセッサーでコードが実行されます。これは、インテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサー上で実行する個別のプロセスを用いた完全に対称な MPI 実行により可能です。例えば、2 基のインテル® Xeon Phi™ コプロセッサー・カード (*hostname1-mic0*、*hostname1-mic1*) が x16 PCIe* スロットに装着された *hostname1* というホスト名の 2 ソケットのシステムと、2 つの実行バイナリー *rtm.cpu* (インテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX2) などのプロセッサー・アーキテクチャー向け) と *rtm.phi* (インテル® Xeon Phi™ コプロセッサー向け) がある場合、インテル® MPI ライブラリーを使用することで、MPI+OpenMP* 対称実行モードで両方の実行バイナリーを利用することができます。

```
mpirun \
-n 1 -host hostname1 -env I_MPI_PIN_DOMAIN=socket -env OMP_NUM_THREADS=14 ./rtm.cpu : \
-n 1 -host hostname1 -env I_MPI_PIN_DOMAIN=socket -env OMP_NUM_THREADS=14 ./rtm.cpu : \
-n 1 -host hostname1-mic0 –env OMP_NUM_THREADS=244 ./rtm.phi : \
-n 1 -host hostname1-mic1 –env OMP_NUM_THREADS=244 ./rtm.phi
```

上記の簡単なシングルノードの例は、*rtm.cpu* と *rtm.phi* が [1, 2] の OpenMP* によるスレッド化によって並列化されていると想定しています。MPI は、ノード、プロセッサー、コプロセッサー間のデータ交換と同期に使用されます。OpenMP* は、MPI プロセスの計算処理を利用可能なプロセッサーのコアとコプロセッサーに分配します。上記の例は、複数のプロセッサーとコプロセッサーで構成される複数のノードで実行するように拡張することもできます。MPI 対称実行モードの詳細は [3] を参照してください。

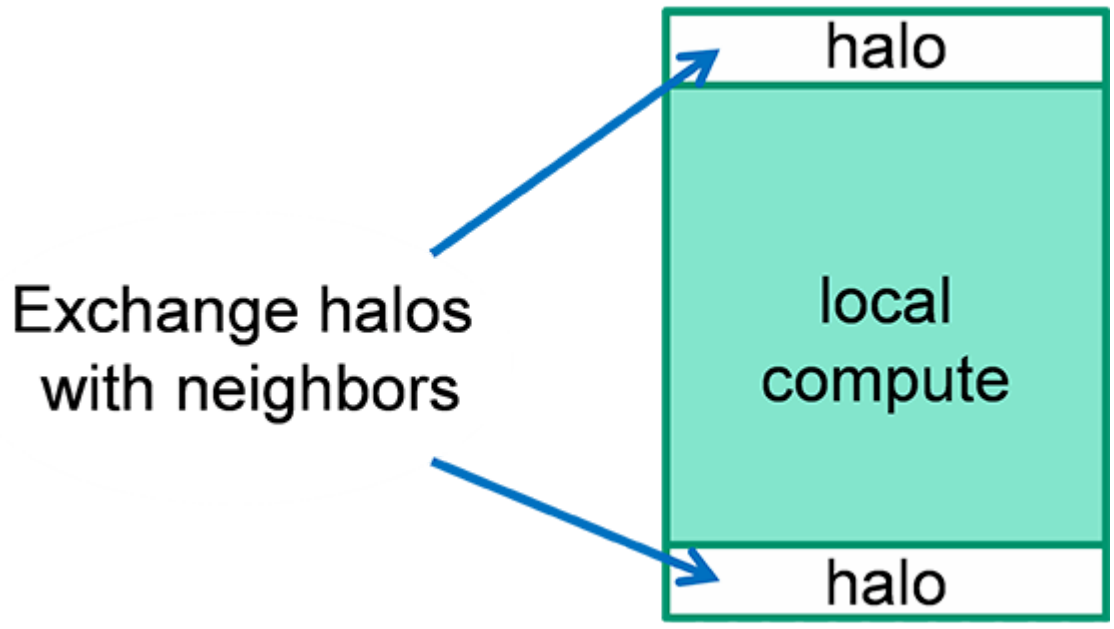
ここで示す簡単な MPI 実装は次の 2 点を想定します:

1) 各インテル® Xeon Phi™ コプロセッサーが独立した計算ノードとして動作することを想定しています。オフロード構文は使用していません。

2) 非ブロックの MPI 呼び出しにより非同期のハロ交換が可能です。各サブドメインの 2 種類の計算領域を考慮することで、計算と隣接するサブドメインとのハロ交換をオーバーラップさせることができます。

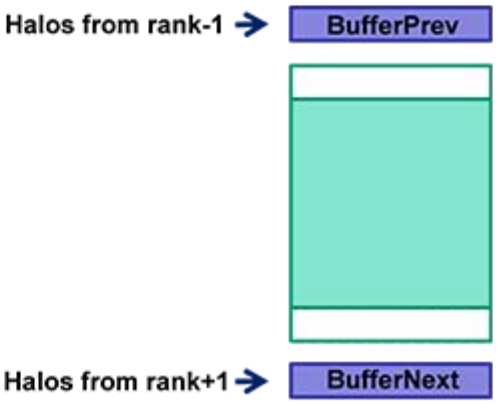
1. *ローカル計算*: 隣接する境界からの距離が *HL* よりも大きいポイント。つまり、同じサブドメイン内の以前に計算された値に基づいてステンシル計算が行われるポイント。
2. *ハロ計算*: 隣接する境界からの距離が *HL* 以下のポイント。つまり、隣接するサブドメインの値に基づいてステンシル計算が行われるポイント。

次のように図解することができます。



MPI 実装は、最も近くにあるハロを交換します。最初に、バッファー *BufferPrev* と *BufferNext* に、隣接するサブドメインから非同期に受け取ったハロを格納します。

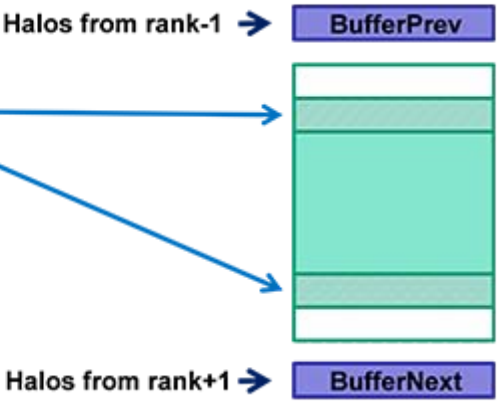
```
MPI_Irecv(BufferPrev, ..., rank-1,  ..,  &request[0]);
MPI_Irecv(BufferNext, ..., rank+1, ..., &request[1]);
```



次に、隣接するサブドメインで必要になる*ハロ計算*領域のポイントを更新し、

```
MPI_Irecv(BufferPrev, ..., rank-1,  ..,  &request[0]);
MPI_Irecv(BufferNext, ..., rank+1, ..., &request[1]);

{  compute halos to be sent to rank-1 and rank+1 }
```

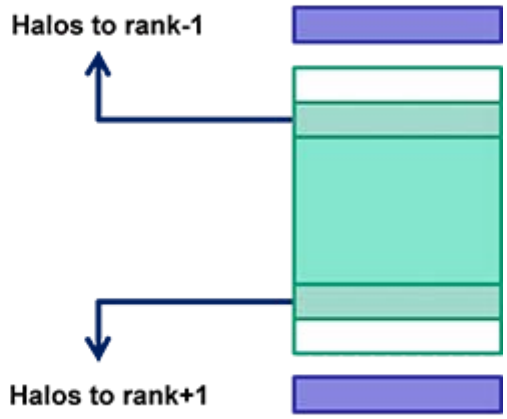


*ハロ計算*領域の更新した値を隣接するドメインへ非同期に送ります。

```
MPI_Irecv(BufferPrev, ..., rank-1,  ..,  &request[0]);
MPI_Irecv(BufferNext, ..., rank+1, ..., &request[1]);

{  compute halos to be sent to rank-1 and rank+1 }

MPI_Isend(DataForPrev, ...,  rank-1, ...,  &request[2]);
MPI_Isend(DataForNext, ...,  rank+1, ..., &request[3]);
```



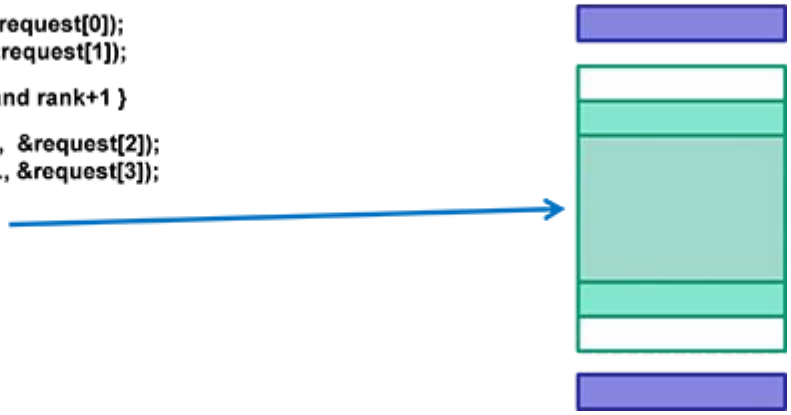
ローカル計算は隣接するサブドメインから提供されるの値に依存しないため、非同期のハロ交換と並行して、*ローカル計算* 領域を更新することができます。

```
MPI_Irecv(BufferPrev, …, rank-1,  ..,  &request[0]);
MPI_Irecv(BufferNext, …, rank+1, …, &request[1]);

{  compute halos to be sent to rank-1 and rank+1 }

MPI_Isend(DataForPrev, …,  rank-1, …,  &request[2]);
MPI_Isend(DataForNext, …,  rank+1, …, &request[3]);

{  compute remaining  inner points }
```



そして、*MPI_Waitall* 同期呼び出しにより、非同期ハロ交換をチェックします。最後に、転送バッファー *BufferPrev* と *BufferNext* にある受け取った値をサブドメインにコピーします。

```
MPI_Irecv(BufferPrev, …, rank-1,  ..,  &request[0]);
MPI_Irecv(BufferNext, …, rank+1, …, &request[1]);

{  compute halos to be sent to rank-1 and rank+1 }

MPI_Isend(DataForPrev, …,  rank-1, …,  &request[2]);
MPI_Isend(DataForNext, …,  rank+1, …, &request[3]);

{  compute remaining  inner points }

MPI_Waitall(4, request,  …);

Copy Halo-from-(rank-1) => Buffer-from-(rank-1)
Copy Halo-from-(rank+1)=> Buffer-from-(rank+1)
```

完全なコードは、この記事に添付されているサンプルコードを参照してください。[1, 2] の ISO-3DFD コードに MPI レイヤーを追加しています。各 MPI プロセスは、インテル® ターボ・モード、インテル® ハイパースレッディング・テクノロジー、ECC モード (インテル® Xeon Phi™ コプロセッサー) などのハードウェア設定やソフトウェアの最適化、そしてキャッシュ・ブロッキング、スレッド・アフィニティー、データ・プリフェッチの距離などを調整することで、パフォーマンスをチューニングできます。シングルプロセスの最適化とチューニングについては、元の記事をご覧ください。

ワークロード・バランス

ヘテロジニアス・システムにおいてワークロード・バランスは非常に重要です。この分散 ISO-3DFD には、プロセッサー・ソケットとインテル® Xeon Phi™ コプロセッサーに割り当てられる計算処理のスタティック・ロード・バランスを行うチューニング・パラメーターがあります。実行ファイルは、次の 2 つのコマンドライン・パラメーターを受け付けます。

factor_speed_phi: インテル® Xeon Phi™ コプロセッサー上での FD コードの実行速度を表す整数値
factor_speed_xeon: インテル® Xeon® プロセッサー上での FD コードの実行速度を表す整数値

ワークバランス係数は *factor_speed_phi / factor_speed_xeon* で、この値を利用してインテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサーへ割り当てる作業量を定義できます。インテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサー間のバランスは、MPI 起動時に定義できま

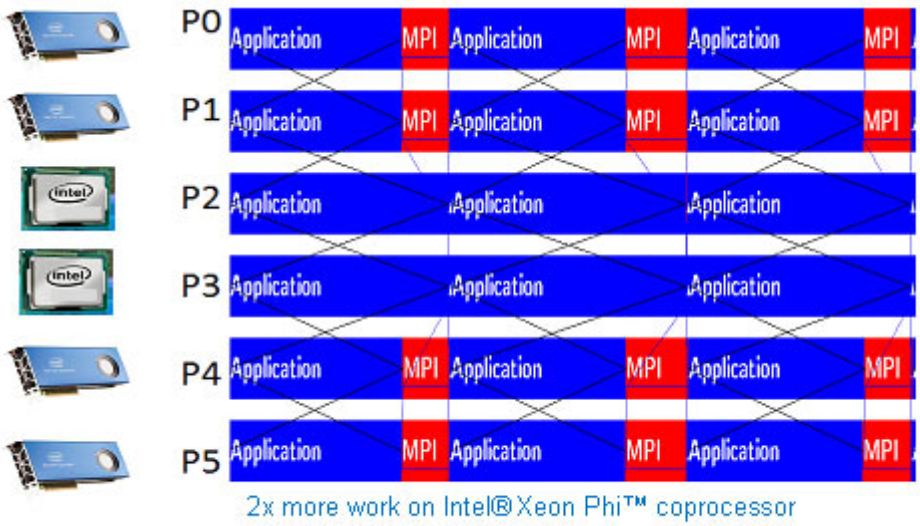
す。そのため、各種インテル® Xeon® プロセッサーおよびインテル® Xeon Phi™ コプロセッサーのモデルに柔軟に対応できます。

スタティック・ロード・バランスは、MPI メッセージ・プロファイリング・ツールを利用することで簡単に得られます。ここでは、インテル® Trace Analyzer & Collector (ITAC) [4] を使用して、MPI トレースを収集し、ロードバランスを解析します。

1. 最初に、ITAC のランタイム環境を設定します。
   ./opt/intel/itac_latest/bin/itacvars.sh
2. *mpirun* コマンドとともに "-t" オプションを指定して、MPI ジョブを実行します。これで、インテル® MPI ライブラリーのランタイムは、MPI メッセージトレースを収集し、結果を拡張子が **.stf* と **.prot* のファイルに保存します。
3. 結果を表示するには、*traceanalyzer* アプリケーションを起動して ITAC の GUI を利用します。*open* で **.stf* ファイルを選択し、*[Continue]* をクリックして *[Summary Page]* を表示します。
4. 次のウィンドウで、*[Charts] > [Event Timeline]* を選択し、マウスを使ってタイムラインを拡大して、プロセッサーとコプロセッサー間の通信パターンを確認します。

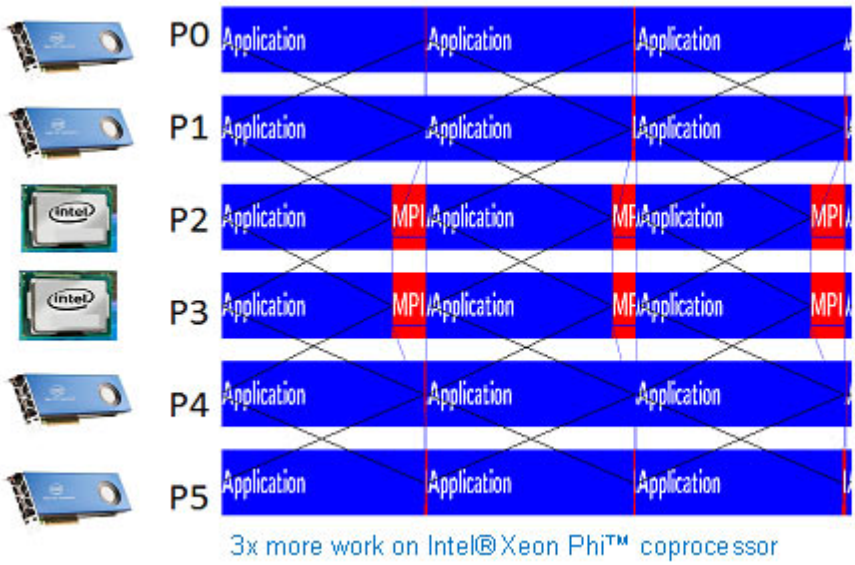
以下の図は、2 ソケット (MPI プロセス P2 と P3) と 4 基のインテル® Xeon Phi™ コプロセッサー (MPI プロセス P0、P1、P4、および P5) を搭載したシステムのロードバランス例を示します。タイムライン・グラフの水平バーは、各 MPI プロセスを表します。赤色の領域は通信/同期によりプロセスが一時的にブロックされたことを示し、青色の領域は通信を伴わない計算処理を表します。スタティック・ロード・バランスの目標は、赤色の領域を最小限に抑えるか、排除し、バランス良く、プロセッサー・ユニットの使用率を上げることです。

この例では、2 基のインテル® Xeon® プロセッサー E5-2697 v2 製品ファミリー、64GB DDR3-1866MHz、4 基のインテル® Xeon Phi™ コプロセッサー 7120 PCIe x16 (それぞれ 61 コア、1.30GHz、16GB DDR5) で構成されたシステムを使用して、*factor_speed_phi* と *factor_speed_xeon* の 3 セットの値について考えてみます。1 つ目のケースは、*factor_speed_phi=10* と *factor_speed_xeon=5* (10/5= 2) で、インテル® Xeon Phi™ コプロセッサーのほうが 1 プロセッサー・ソケットよりも 2 倍高速に計算できると想定した場合です。MPI トレースは次のようになります。



上記のイベント・タイムラインから、プロセッサー (MPI プロセスランク P2 と P3) のほうが処理に時間がかかり、コプロセッサー (P0、P1、P4、および P5) はプロセッサーの処理が終わるのを待っている間、ブロック

されアイドル状態 (赤色の領域) になることが分かります。2 つ目のケースは、*factor_speed_phi=15* と *factor_speed_xeon=5* (15/5= 3) で、インテル® Xeon Phi™ コプロセッサーのほうが 1 プロセッサー・ソケットよりも 3 倍高速に計算できると想定した場合です。MPI トレースは次のようになります。



上記のイベント・タイムラインから、コプロセッサー (P0、P1、P4、および P5) のほうが処理に時間がかかり、プロセッサー (P2 と P3) はコプロセッサーの処理が終わるのを待っている間、ブロックされアイドル状態 (赤色の領域) になることが分かります。

最後に、3 つ目のケースは *factor_speed_phi=13* と *factor_speed_xeon=5* (13/5= 2.6) で、インテル® Xeon Phi™ コプロセッサーのほうが 1 プロセッサー・ソケットよりも 2.6 倍高速に計算できると想定した場合です。

このケースでは、ロード・インバランスが解消され、アイドル時間がなくなっています。つまり、最適なロードバランスを表しています。

上記のスタティック解析は、特定のハードウェア構成 (プロセッサー・モデル、インテル® Xeon Phi™ コプロセッサー・モデル、プロセッサー数、メモリー速度など) での結果です。ハードウェア構成やアルゴリズム実装に変更があった場合は、新たにスタティック解析を行う必要があります。

サンプルコード

前述の概念の実装例として、サンプルコードを添付しています。実際に、ビルド、実行、解析してみてください。MPI ジョブは、インテル® Xeon® プロセッサーとインテル® Xeon Phi™ コプロセッサーの両方、あるいはいずれか一方で、実行する MPI ランク数 (プロセス数) に設定することができます。

ISO-3DFD のシングル・プロセス・バージョンについては、すでに [1, 2] で紹介されているため、ここでは取り上げません。代わりに、ISO-3DFD を拡張し、プロセッサーとコプロセッサー間のハロ交換とロードバランスに対応した分散 ISO-3DFD を提供することで、ISO-3DFD の将来のバージョンでもサンプルコードを利用できるようにしています。

このサンプルコードのビルドに必要なものは、オリジナルの ISO-3DFD シングル・プロセス・バージョンの計算カーネル ([1,2] から入手可能) とインテル® ソフトウェア開発ツール (インテル® C/C++ コンパイラー、インテル® MPI ライブラリー、および ITAC) だけです。

README ファイルに、ビルドと実行手順があります。*Makefile* は、2 つの実行ファイルを生成します: プロセッサー上で実行する *dist-iso-3dfd* とインテル® Xeon Phi™ コプロセッサー上で実行する *dist-iso-3dfd.MIC*。変数 *VERSION_MIC* と *VERSION_CPU* で、使用する ISO-3DFD バージョンを選択できます。

スクリプト *run_example.sh* は、インテル® MPI ライブラリーを使用して 2 つの実行ファイルを起動します。詳細は、[3] を参照してください。このスクリプトは、2 基のプロセッサーと 2 基のインテル® Xeon Phi™ コプロセッサーで構成されたシステムでの実行用ですが、クラスターや異なるノード構成向けに簡単に拡張できます。スクリプト変数 *TRACE_OPTION=-t* を設定すると、MPI 通信のトレース情報が収集されるため、前述のスタティック・ロード・バランスを行うことができます。メイン・ソースコード *dist-iso-3dfd.cc* は、コマンドライン・オプションとしてパラメーター *factor_speed_phi* と *factor_speed_xeon* を受け付けます。「ワークバランス」セクションで述べたように、これらのパラメーターを使用して、スタティック・ワーク・バランスを行ってください。

スクリプトは、メイン実行ファイルで必要なすべてのコマンドライン・オプション (*nx ny nz #threads #iterations x-blocking y-blocking z-blocking factor_speed_phi factor_speed_xeon*) と追加の MPI および OpenMP* 設定もサポートしています。

ここで紹介したのは一例です。ISO-3DFD 計算カーネル向けの最適なコンパイラー・オプションとランタイム・パラメーターについては、[1,2] を参照してください。シングルノードと複数ノードのパフォーマンスについては、[5] を参照してください。

まとめ

ここでは、分散 FD (ISO-3DFD ステンシル計算カーネルの 1 次元分解) の実装例を説明しました。この実装は、デバイスごとに異なる計算スピードに対応するため、スタティック・ロード・バランスをサポートすることで、複数のプロセッサーとコプロセッサーで構成されるヘテロジニアス・システムをサポートします。

MPI メッセージ・プロファイリング・ツールを使用することで、次の場合にタイムステップごとの解析が可能です: 1) インテル® Xeon Phi™ コプロセッサーがプロセッサーの完了を待機している場合 2) インテル® Xeon® プロセッサーがインテル® Xeon Phi™ コプロセッサーの完了を待機している場合。

参考文献 (英語)

[1] "Optimizing and Running ISO 3DFD with Support for Intel® Xeon Phi™ Coprocessor." http://software.intel.com/en-us/articles/eight-optimizations-for-3-dimensional-finite-difference-3dfd-code-with-an-isotropic-iso

[2] "Characterization and Optimization Methodology Applied to Stencil Computations," in the book *High Performance Parallelism Pearls* (J. Reinders and J. Jeffers, editors).http://www.techenablement.com/characterization-optimization-methodology-applied-stencil-computations/

[3] "How to run Intel MPI on Xeon Phi" http://software.intel.com/en-us/articles/how-to-run-intel-mpi-on-xeon-phi

[4] Tutorial: Analyzing MPI Applications with Intel® Trace Analyzer and Intel® VTune™ Amplifier XE.http://software.intel.com/en-us/analyzing-mpi-apps-with-itac-and-vtune

[5] "Intel® Xeon Phi™ Coprocessor Energy Application Benchmarks." http://www.intel.com/content/www/xr/en/benchmarks/server/xeon-phi/xeon-phi-energy.html

本資料で紹介されている資料番号付きのドキュメントや、インテルのその他の資料を入手するには、1-800-548-4725 (アメリカ合衆国) までご連絡いただくか、http://www.intel.com/design/literature.htm (英語) を参照してください。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください