インテル® コンパイラーを使用した OpenMP* による並列プログラミング

セッション 1: はじめに

IA Software User Society (iSUS)
編集長　すがわら　きよふみ

このセッションの目的

明示的な並列プログラミング手法として注目されてきた OpenMP* による並列プログラミングに加え、インテル® コンパイラーがサポートする OpenMP* 4.0 と 4.5 の機能を使用したベクトル・プログラミングとオフロード・プログラミングの概要をリフレッシュし、インテル® コンパイラー V19.1 でサポートされる OpenMP* 5.0 の機能と実装を紹介します。さらに新たなアクセラレーター・デバイスへのオフロードについて考えます

セッションの対象者

すでに OpenMP* でマルチスレッド・プログラミングを開発し、4.0 以降でサポートされる新たなベクトル化とオフロードを導入し、アプリケーションのパフォーマンス向上を計画する開発者

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

セッションリスト

セッション	説明
はじめに (13:35 – 14:20)	異なるバージョンのインテル® コンパイラーや異なるコンパイラー間で OpenMP* を使用する注意点や制限について説明します
OpenMP* のタスク機能 (14:30 – 15:30)	OpenMP* 3.1 で追加されたタスク機能が 4.0 から 4.5 でどのように進化したかを例を使用して説明し、最新の OpenMP* 5.0 で強化された新機能を紹介します
OpenMP* の SIMD 機能 (13:30 – 14:30)	OpenMP* のスレッド化機能を使用してプログラマーがマルチスレッドの動作をプログラミングしたように、OpenMP* 4.0 からは omp simd を使用してプログラマーが明示的にベクトル化もできるようになりました。OpenMP* simd に関連する機能を 4.0 から 5.0 までの進化を追って紹介します
OpenMP* のオフロード機能 (14:30 – 15:30)	OpenMP* 4.0 で追加されたオフロード機能を利用することで、これまで共有メモリー型並列処理に加え分散メモリー型の並列処理を表現できるようになりました。このセッションでは、注目されるヘテロジニアス・プログラミング環境での OpenMP* オフロード機能について説明します
OpenMP* 5.0 の注目する機能	セッション2、3、4でカバーされなかった OpenMP* 5.0 のそのほかの機能について紹介します
インテル® C++/Fortran コンパイラーのバージョン 19.1 を使用して GPU オフロードに備えましょう	oneAPI 向けのデータ並列 C++ (DPC++) へ移行する前に、現行のインテル® C++/Fortran コンパイラー V19.1 やインテル® oneAPI HPC ツールキットに含まれるベータ版インテル® C++/Fortran コンパイラー 2021 を使用して簡単にインテル® グラフィックスへのオフロードを行うソフトウェアを開発および検証方法を紹介します

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

内容

- はじめに (OpenMP* が必要とされる背景) と概要
  (OpenMP* とは、歴史、各バージョンの機能概要)
- OpenMP* の各バージョンの機能
  (4.0、4.5 および 5.0 の注目される新機能)
- 次世代インテル® コンパイラー (nextgen) の機能

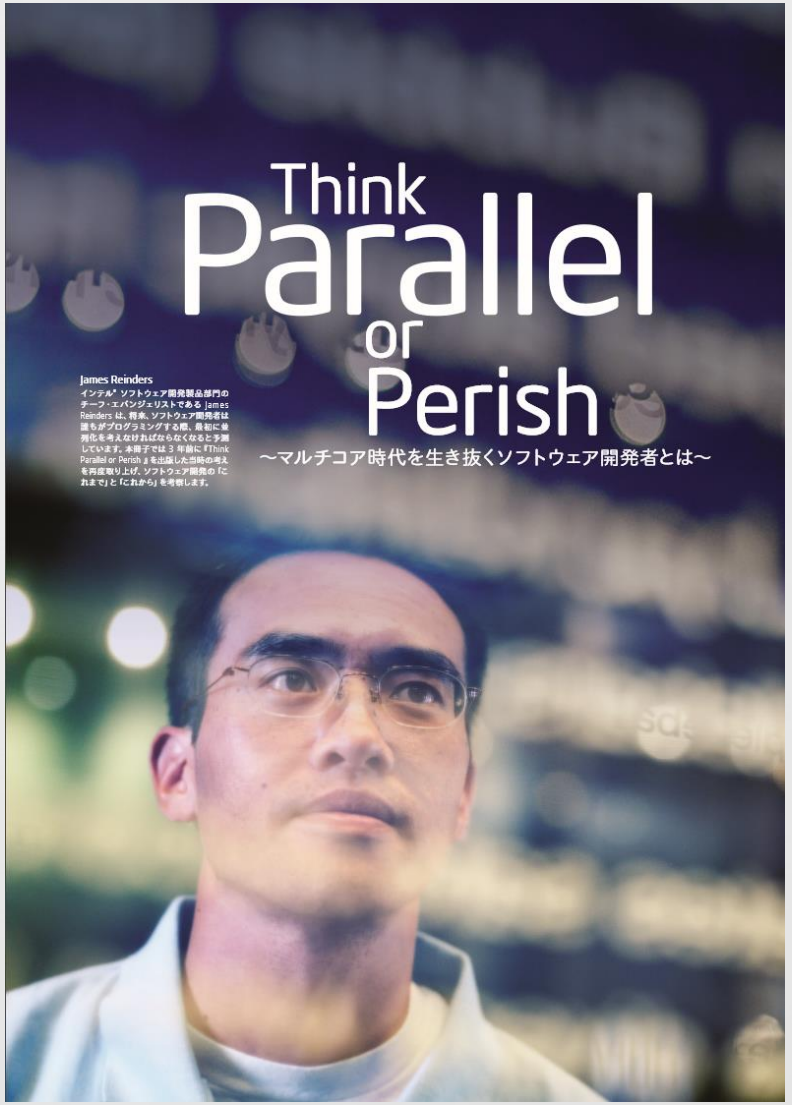
© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

はじめに

- コードの現代化 (モダニゼーション)
- OpenMP* が求められる背景
  - OpenMP* とは (歴史的背景)
  - OpenMP* を使用するには
  - インテル® コンパイラーにおける OpenMP* サポート

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

Think Parallel or Perish



あれから 11 年 …

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

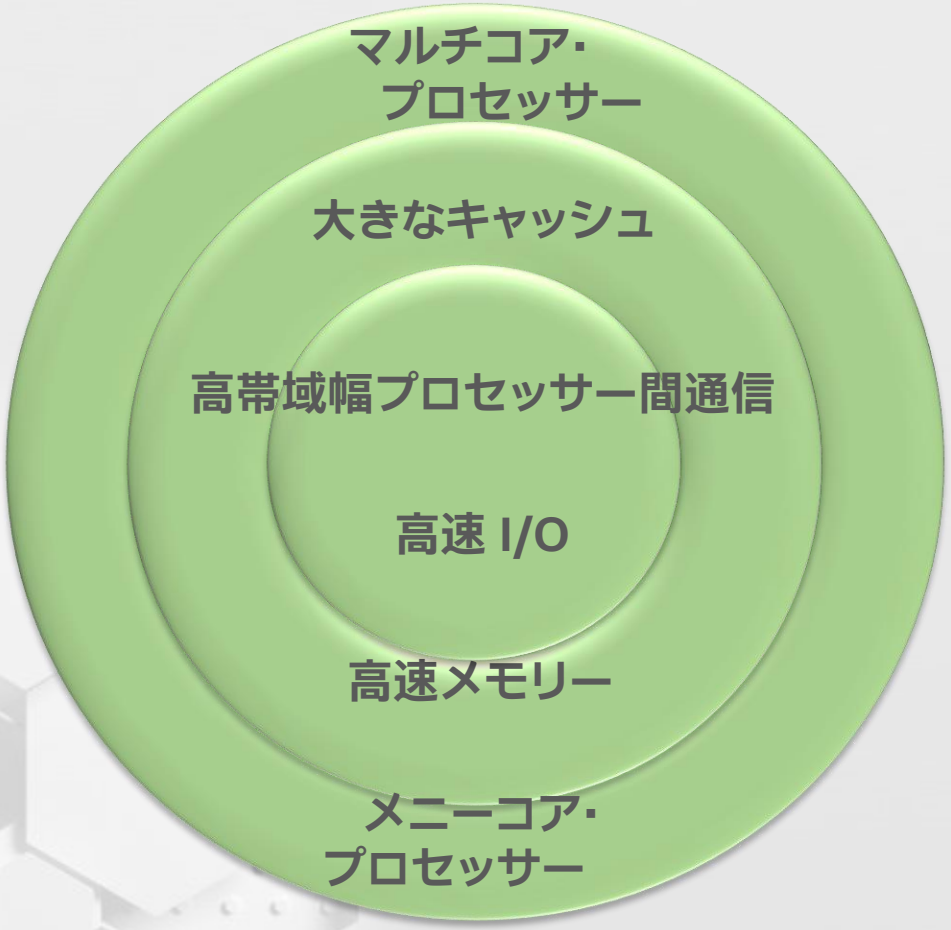
Vectorize or Die

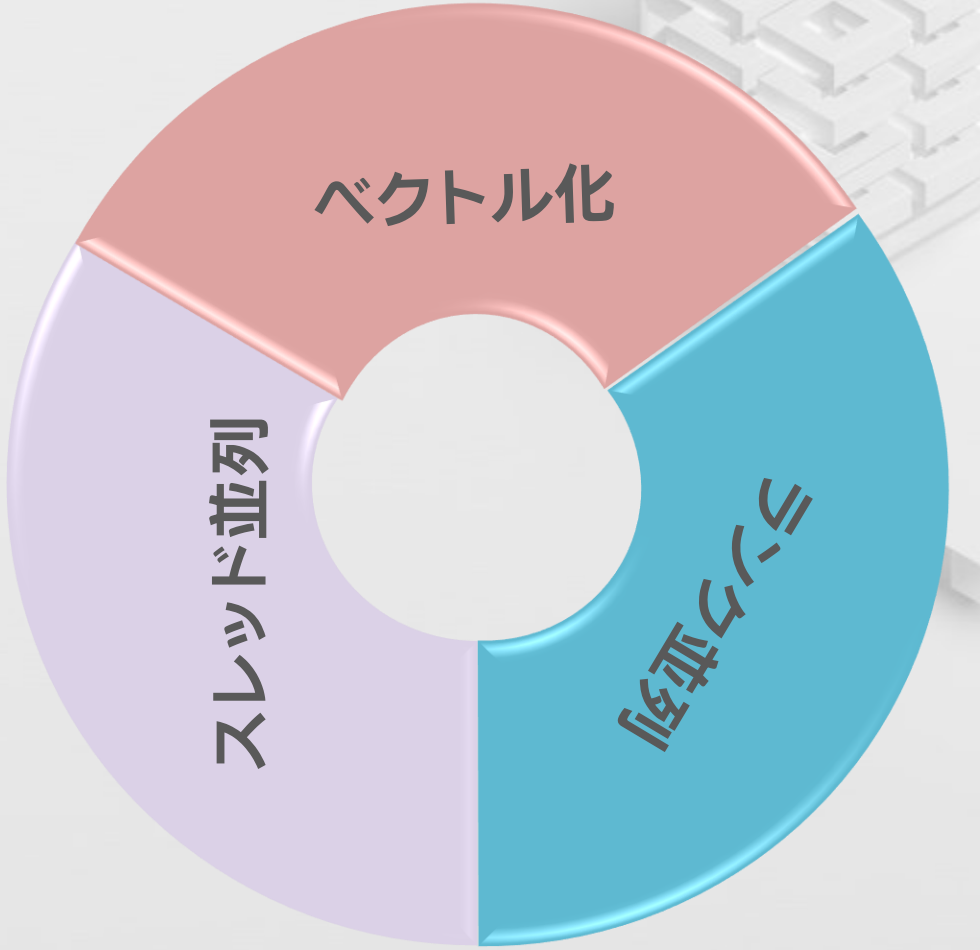
arallel or Perish

odernization

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

Code Modernization (コードのモダン化)





© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

マルチレベルの並列性を実装する 5 つのステップ

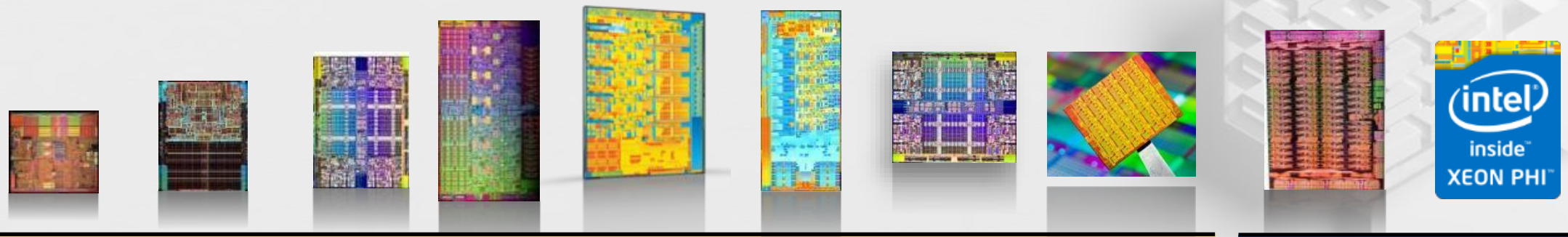
ステージ 1: 最適化ツールとライブラリーの活用

http://www.isus.jp/article/article-parallel/what-is-code-modernization/

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

Parallel + SIMD は前進への鍵 (少し前…)

インテル® Xeon® プロセッサーは、両者とも並列性を高めています



	インテル® Xeon® プロセッサー 64 ビット	インテル® Xeon® プロセッサー 開発コード名 Woodcrest EP	インテル® Xeon® プロセッサー 開発コード名 Nehalem EP	インテル® Xeon® プロセッサー 開発コード名 Westmere EP	インテル® Xeon® プロセッサー 開発コード名 Sandy Bridge EP	インテル® Xeon® プロセッサー 開発コード名 Ivy Bridge EP	インテル® Xeon® プロセッサー 開発コード名 Haswell EP	インテル® Xeon® プロセッサー 開発コード名 Skylake Server	インテル® Xeon Phi™ x100 コプロセッサー 開発コード名: Knights Corner	インテル® Xeon Phi™ x200 プロセッサー & コプロセッサー 開発コード名 Knights Landing
コア	1	2	4	6	8	12	18	28	61	72
スレッド	2	2	8	12	16	24	36	56	244	288
SIMD 幅	128	128	128	128	256	256	256	512	512	512

さらに多いコア　さらに多くのスレッド　より広いベクトル

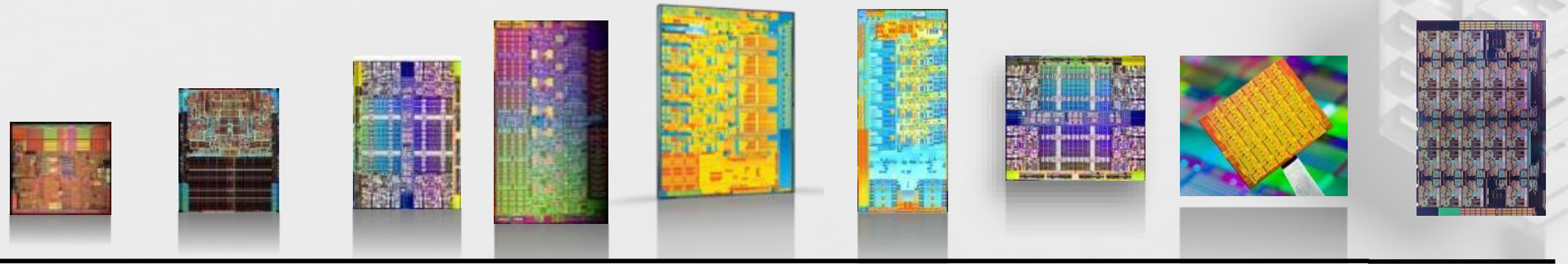
OpenMP* は parallel + SIMD を前進させる最も重要な機能の 1 つ

* ark.intel.com で公開されている出荷済の製品仕様

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

Parallel + SIMD は前進への鍵 (現在)

インテル® Xeon® プロセッサーは、両者とも並列性を高めています



	インテル® Xeon® プロセッサー 64 ビット	インテル® Xeon® プロセッサー 開発コード名 Woodcrest EP	インテル® Xeon® プロセッサー 開発コード名 Nehalem EP	インテル® Xeon® プロセッサー 開発コード名 Westmere EP	インテル® Xeon® プロセッサー 開発コード名 Sandy Bridge EP	インテル® Xeon® プロセッサー 開発コード名 Ivy Bridge EP	インテル® Xeon® プロセッサー 開発コード名 Haswell EP	インテル® Xeon® プロセッサー 開発コード名 Skylake Server	第 2 世代 インテル® Xeon スケーラブル・プロセッサー 開発コード名 Cascade Lake
コア	1	2	4	6	8	12	18	28	56
スレッド	2	2	8	12	16	24	36	56	112
SIMD 幅	128	128	128	128	256	256	256	512	512

+ アクセラレーター

さらに多いコア　さらに多くのスレッド　より広いベクトル

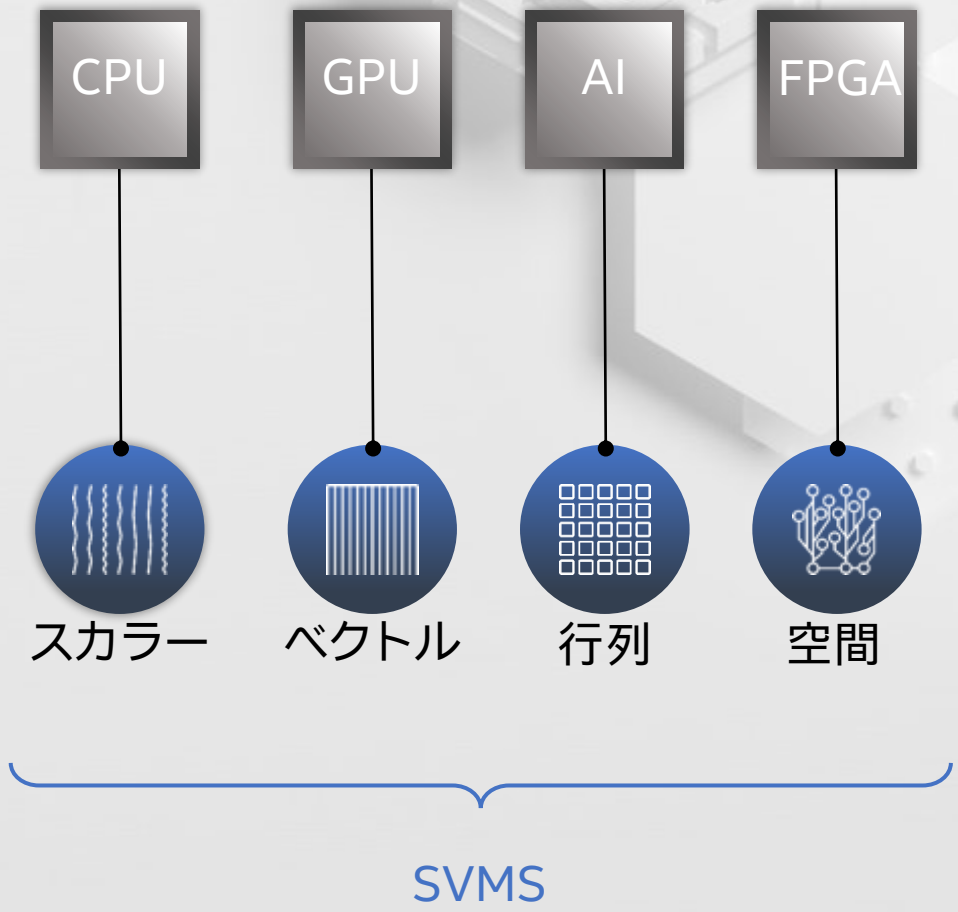
OpenMP* は parallel + SIMD を前進させる最も重要な機能の 1 つ

* ark.intel.com で公開されている出荷済の製品仕様

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

新たなプログラミングの課題

- さまざまなデータ構造を使用するハードウェアのセットが存在します
- 残念ながら、それらに対する共通のプログラミング言語や API はありません
- また、それぞれのプラットフォームにおいてツールのサポートに一貫性もありません
- プラットフォームごとに個別のソフトウェア投資/習得が必要です



© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® oneAPI

- **インテル® oneAPI** プロジェクトはさまざまなアーキテクチャーにわたる開発を容易にする統合プログラミング・モデルを提供
- データ並列 C++ 言語とインテル® oneAPI ライブラリー API により並列処理を表現
- 妥協のないパフォーマンス
- CPU、GPU、AI、および FPGA をサポート
- 業界標準およびオープン仕様ベース
- OpenMP*、Fortran、MPI などとの相互運用性

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

はじめに

- コードの現代化 (モダニゼーション)
- OpenMP* が求められる背景
  - OpenMP* とは (歴史的背景)
  - OpenMP* を使用するには
  - インテル® コンパイラーにおける OpenMP* サポート

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

OpenMP* API

ほぼ 23 年の技術計算/HPC における共有メモリー並列プログラミング (C/C++ と Fortran) 向けの業界標準:

- 最新の仕様: OpenMP* 5.0 (2018 年 11 月)
- ディレクティブ (C/C++ ではプラグマ) ベース
- ベンダーとプラットフォームにわたる移植性 - インテルは OpenMP* ARB (アーキテクチャー・レビュー委員会) のメンバーであり、OpenMP* のサポートを完全にコミット
- 各種並列性をサポート: スレッド、タスク、SIMD、オフロード
- インクリメンタルな並列処理をサポート

仕様ドキュメント、サンプル、および各種情報は www.openmp.org (英語) を参照してください

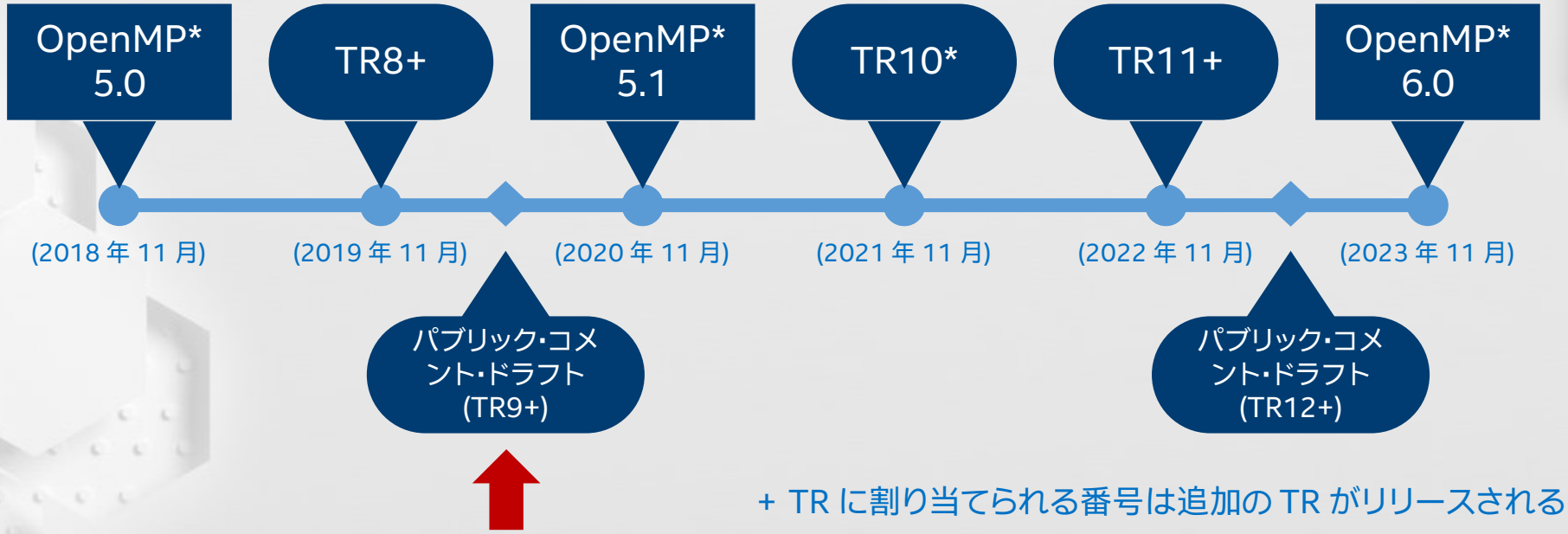
© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

OpenMP* の歴史

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

今後の OpenMP* ロードマップ

- 主要リリースに向けた今後の計画
- 主要リリースの間には、1 つのマイナーリリースが発表されます (例えば 5.1)
- (少なくとも) 1 つのテクニカルレポート (TR) と機能レビューが毎年行われます



+ TR に割り当てられる番号は追加の TR がリリースされると変更されます

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

コンパイラーが OpenMP* をサポートするか?

- OpenMP* は最も簡単なマルチスレッド・プログラミング・モデルであるが、コンパイラーが OpenMP* をサポートしている必要がある

- OpenMP* をサポートするコンパイラーは、_OPENMP マクロに仕様が公開された年月を数値で返す

仕様のバージョン	値	代表的なコンパイラー
OpenMP* 5.0 TR4	201611	インテル® コンパイラー V18.0、V19.0、V19.1
OpenMP* 4.5	201511	インテル® コンパイラー V17.0、gcc 6.1
OpenMP* 4.0	201307	インテル® コンパイラー V14.0、gcc 4.9 (offload 5.1)
OpenMP* 3.1	201107	インテル® コンパイラー V12.1、gcc 4.7
OpenMP* 3.0	200805	インテル® コンパイラー V11.0、gcc 4.4
OpenMP* 2.5	200505	インテル® コンパイラー V9.0、gcc 4.2

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

記述している構文が OpenMP* のどの仕様か?

OpenMP API 4.5 C/C++ Page 1

OpenMP

C/C++ 日本語翻訳版

OpenMP 4.5 API C/C++ シンタックス・リファレンス・カード

OpenMP アプリケーション・プログラム・インターフェイス (API) は、スケーラブルでポータブルな並列処理を実現します。ポータブルな並列アプリケーションを開発するためのシンプルで柔軟性のあるインターフェイスです。

OpenMP は、Unix プラットフォームと Windows プラットフォームを含む、すべてのアーキテクチャー上での C/C++ と Fortran による共有メモリー型並列プログラミングをサポートします。 仕様に関しては www.openmp.org をご覧ください。

このテキストの色 は、OpenMP 4.5 仕様の新機能もしくは変更点を指します。
[n.n.n] は、OpenMP API 4.5 仕様書のセクションを示します。
[n.n.n] は、OpenMP API 4.0 仕様書のセクションを示します。

C/C++ のディレクティブと構文

OpenMP の実行ディレクティブは、後続の構造化ブロックや OpenMP 構文に適用されます。各ディレクティブの部分は、C と C++ コンパイラーの標準規約に準拠します。構造化ブロックは、単一のステートメント

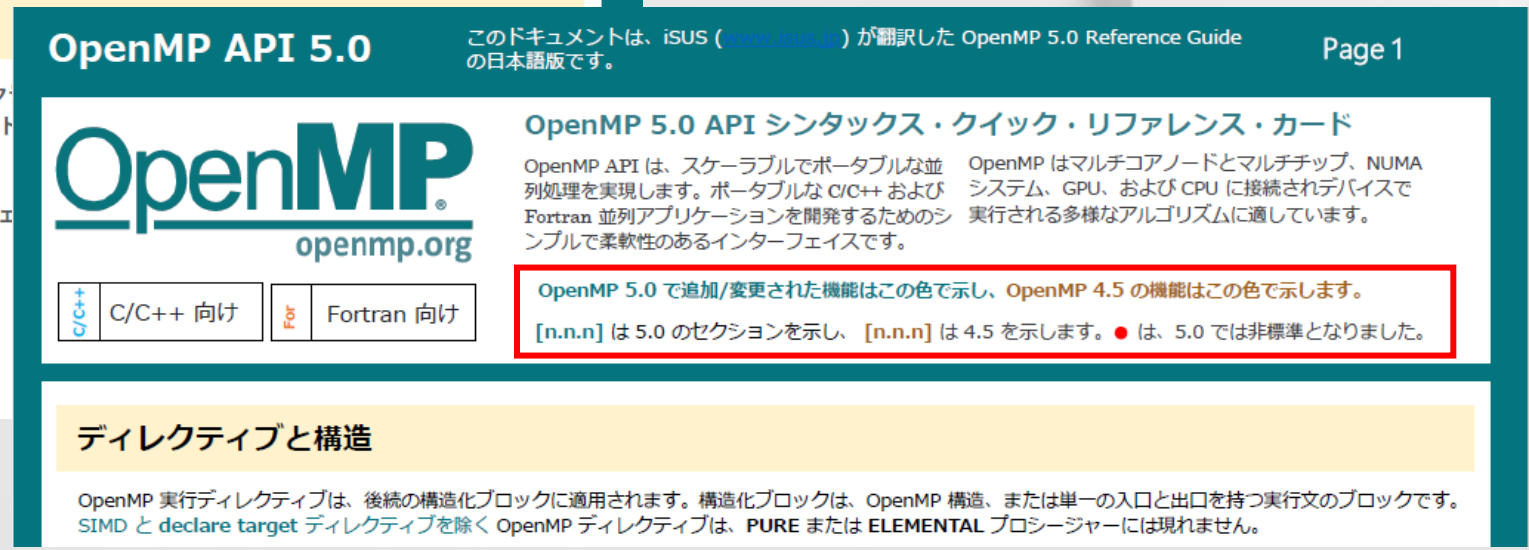
parallel [2.5] [2.5]
スレッドのチームを形成し、並列実行を開始します。

#pragma omp parallel [節 [,]節] ...]
構造化ブロック

節:
if([parallel:]スカラー式)
num_threads(整数式)

sections [2.7.2] [2.7.2]
一連の構造化ブロックを含む非反復型のワークシェアリング構文は、スレッドのチームに分散され実行されます。

#pragma omp sections [節[[,] 節] ...]
{
[#pragma omp section]
構造化ブロック



OpenMP API 5.0

このドキュメントは、iSUS ( ) が翻訳した OpenMP 5.0 Reference Guide の日本語版です。

Page 1

OpenMP
openmp.org

C/C++ C/C++ 向け For Fortran 向け

OpenMP 5.0 API シンタックス・クイック・リファレンス・カード

OpenMP API は、スケーラブルでポータブルな並列処理を実現します。ポータブルな C/C++ および Fortran 並列アプリケーションを開発するためのシンプルで柔軟性のあるインターフェイスです。

OpenMP はマルチコアノードとマルチチップ、NUMA システム、GPU、および CPU に接続されデバイスで実行される多様なアルゴリズムに適しています。

OpenMP 5.0 で追加/変更された機能はこの色で示し、OpenMP 4.5 の機能はこの色で示します。
[n.n.n] は 5.0 のセクションを示し、 [n.n.n] は 4.5 を示します。● は、5.0 では非標準となりました。

ディレクティブと構造

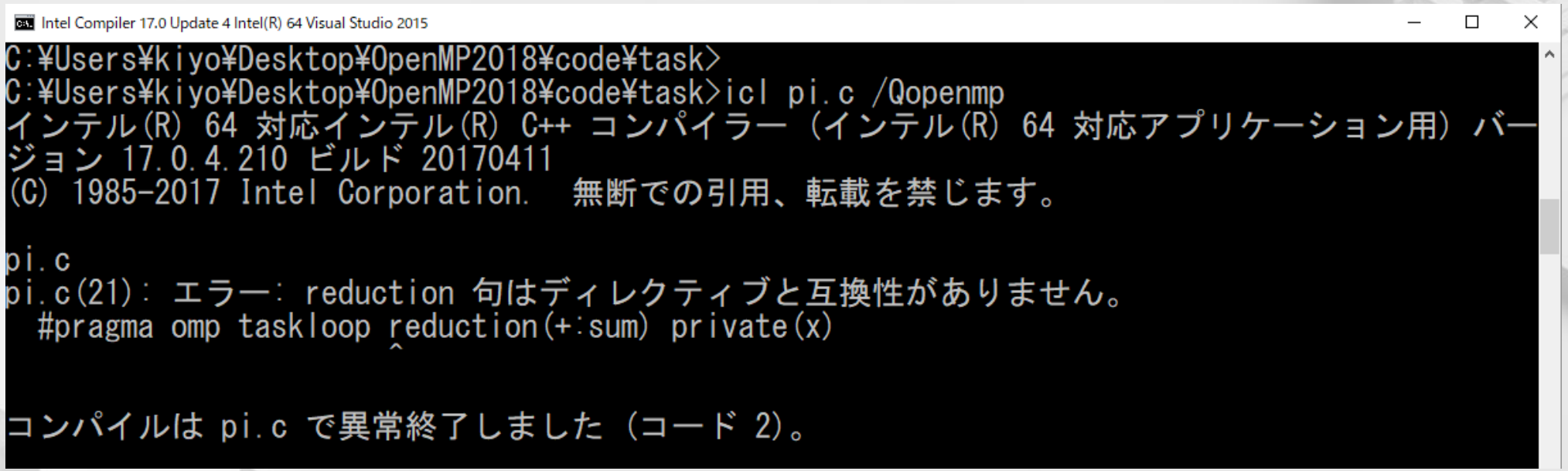
OpenMP 実行ディレクティブは、後続の構造化ブロックに適用されます。構造化ブロックは、OpenMP 構造、または単一の入口と出口を持つ実行文のブロックです。
SIMD と declare target ディレクティブを除く OpenMP ディレクティブは、PURE または ELEMENTAL プロシージャーには現れません。

http://openmp.org/wp/openmp-specifications/ (英語)

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ソースとコンパイラーのバージョンの不一致

ソースコードで使用する OpenMP* の仕様よりもコンパイラーがサポートする OpenMP* 仕様が古い場合、コンパイルエラーとなります



_OPENMP マクロでコンパイルを制御:

```
#if _OPENMP >= 201611
#pragma omp taskloop reduction(+:sum) private(x)
```


© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテルの OpenMP* ライブラリーのよくある質問

- ほかのコンパイラーとのソースとオブジェクトの互換性
- インテル® コンパイラーの異なるバージョンを使用するガイドライン
- インテル® コンパイラーとその他のコンパイラーを併用するガイドライン
- OpenMP* ライブラリーをその他のコンパイラーで使用する際の制限
- Visual Studio* でインテルの OpenMP* ライブラリーを使用する際の制限
- maxOS* で OpenMP* を使用する際の注意
- Fortran と C/C++ の混在
- インテルの OpenMP* のサポート・ライブラリー

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ほかのコンパイラーとのソースとオブジェクトの互換性

- インテル® C++ コンパイラーには、次のコンパイラーの OpenMP* サポートとソースおよびオブジェクト・レベルの互換性があります:
  - Microsoft* Visual C++* 2012 以降のコンパイラー (Windows*)
  - GCC 4.4.2 以降 (Linux*)
  - インテル® C++ コンパイラー 10.0 以降

コンパイラーが異なるとサポートされる OpenMP* 仕様も異なります。アプリケーションが使用する OpenMP* 機能に基づいて、どのバージョンの OpenMP* 仕様が必要かを判断してください

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

複数のソースコードの一部で OpenMP* を使用する

- もしくは異なるバージョンのコンパイラーでプロジェクトをビルドする場合

```
main() {
    printf("OpenMP version %d ¥n", _OPENMP);
    printf("OpenMP version %d ¥n", omp());
}
```

インテル® コンパイラー 17.0 でコンパイル

```
int omp(void){
return _OPENMP;
}
```

インテル® コンパイラー 16.0 でコンパイル

```
OpenMP version 201511
OpenMP version 201307
```

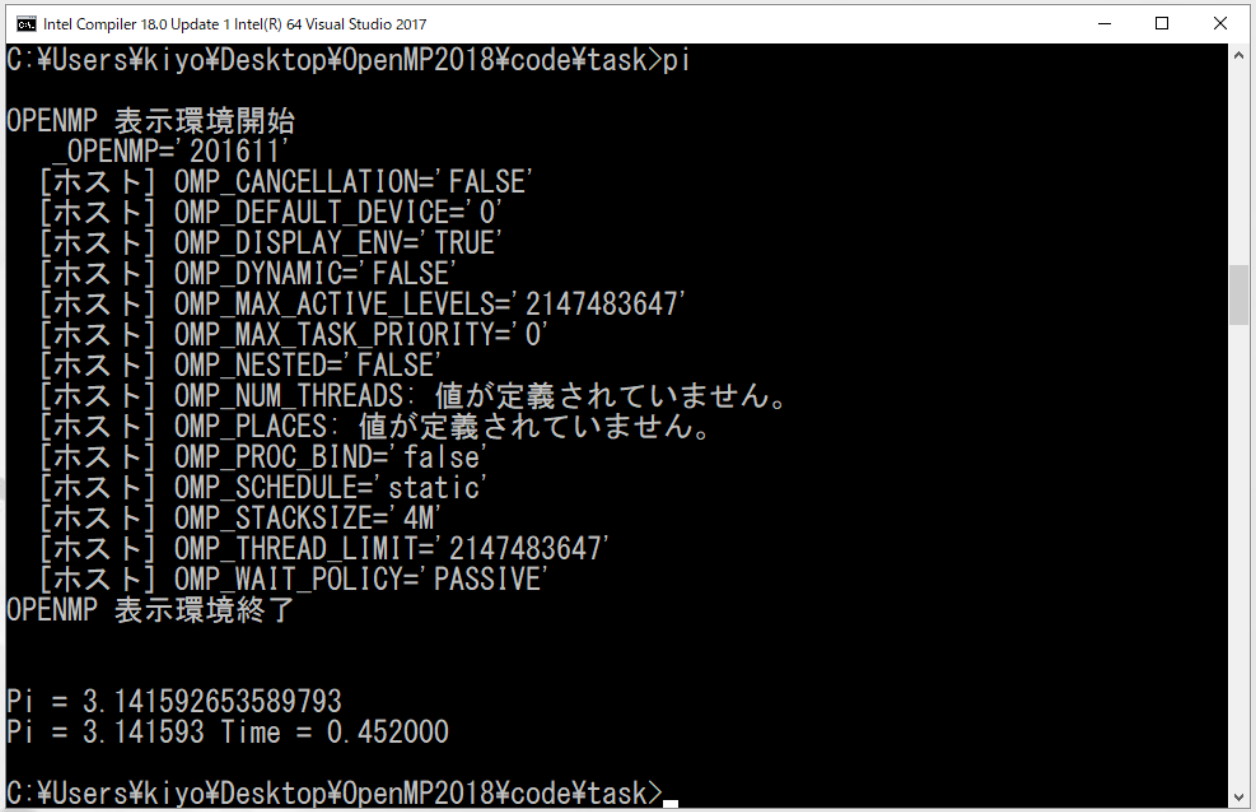
Visual C++* 2015 のコンパイラーでコンパイル

```
OpenMP version 201511
OpenMP version 200203
```


© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

実行環境を確認する便利な機能

OpenMP* 4.0 以降のライブラリーでは、環境変数 OMP_DISPLAY_ENV が利用できます。true に設定すると、実行時に情報を表示できます



true、false (デフォルト)、verbose が設定できます

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® コンパイラーの異なるバージョンを使用するガイドライン

リンク時または実行時の問題を回避するには、次の点に注意してください:

- 現行のインテル® C++ コンパイラーで生成されたオブジェクト・モジュールは、インテル® コンパイラー 10.0 より前のバージョンでコンパイルされたオブジェクト・モジュールとは互換性がありません
- OpenMP* スタティック・ライブラリーではなく、ダイナミック・ライブラリーを使用して、ライブラリーの複数のコピーが 1 つのプログラムにリンクされないようにします (Linux*、macOS*)

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® C++ コンパイラーとその他のコンパイラーを併用するガイドライン

リンク時または実行時の問題を回避するには、次の点に注意してください:

- 常にインテル® C++ コンパイラーの OpenMP* ライブラリーを使用してリンクします。これにより、異なるコンパイラーから OpenMP* ランタイム・ライブラリーの複数のコピーがリンクされるのを防ぎます
- 可能であれば、すべての OpenMP* ソースを同じコンパイラーでコンパイルします
- OpenMP* スタティック・ライブラリーではなく、ダイナミック・ライブラリーを使用して、ライブラリーの複数のコピーが 1 つのプログラムにリンクされないようにします

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

OpenMP* ライブラリーをその他のコンパイラーで使用する際の制限

オブジェクト・レベルの互換性には、threadprivate オブジェクトに次の制限があります:

- インテル® C++ コンパイラーは threadprivate データを参照する際、デフォルト (/Qopenmp-threadprivate:legacy) で GCC や Microsoft* Visual C++* コンパイラーとは異なる方法を用います
- コードで変数を threadprivate と宣言し、インテル® C++ コンパイラーと GCC/Visual C++* コンパイラーの両方でコンパイルする場合、インテル® C++ コンパイラーでコンパイルされたコードと GCC/Visual C++* コンパイラーでコンパイルされたコードでは、同じスレッドによって参照されていたとしても、異なる場所の変数が参照されます
- GCC/Visual C++* コンパイラーでコンパイルされたコードと同じ threadprivate 場所を参照させるには、インテル® C++ コンパイラーでのコンパイル時に、/Qopenmp-threadprivate:compat オプションを使用します

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

Visual Studio* でインテルの OpenMP* ライブラリーを使用する際の制限

Windows* システムで、Visual Studio* の Visual C++ 開発環境を一部変更すると、インテル® C++ コンパイラーと Visual C++* を使用して、インテルの OpenMP* ライブラリーを使用するアプリケーションを作成することができます

Microsoft Visual* C++ では、_OPENMP_NOFORCE_MANIFEST シンボルが定義されていなければなりません。定義されていない場合は、vcomp90 dll のマニフェストがインクルードされます。これにより、ビルドシステムでは問題ありませんが、この DLL がインストールされていない別のシステムにアプリケーションを移動すると問題が発生します

```
cl /MD /openmp /c f1.c f2.c
icl /MD /Qopenmp /c f3.c f4.c
link f1.obj f2.obj f3.obj f4.obj /out:app.exe /nodefaultlib:vcomp libiomp5md.lib
```


© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

macOS* で OpenMP* を使用する際の注意

- 古いバージョンの macOS* プラットフォームでは、インテル® C++ コンパイラーと GCC コンパイラーを併用してコンパイルできます
- macOS* 10.9 以降のプラットフォームには、GCC コンパイラーの代わりに Clang コンパイラーが含まれていますが、このコンパイラーは OpenMP* 実装をサポートしていません。Clang コンパイラーの将来のバージョンでは OpenMP* 実装がサポートされる可能性があります
- macOS* 10.9 以降 (Xcode* 5.x 以降) には GCC コンパイラーが含まれていませんが、インテル® C++ コンパイラーと一緒に GCC コンパイラーをインストールできます

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

Fortran と C/C++ の混在

- gcc コンパイラー、インテル® C++ コンパイラー、インテル® Fortran コンパイラーでコンパイルされた OpenMP* オブジェクト・ファイルを混在させることができます
- インテル® Fortran コンパイラーと gfortran コンパイラーでコンパイルされたオブジェクト・ファイルを混在させることはできません
- インテル® C++ コンパイラーを使用してアプリケーションをリンクするこができますが、-l オプションを使用して、リンク行で複数の gfortran ライブラリーを渡す必要があります

C ソースと Fortran ソースの混在

```
gfortran -fopenmp -c foo.f
icc -qopenmp -c ibar.c
icc -qopenmp foo.o bar.o -lgfortranbegin -lgfortran
```


© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

OpenMP* のサポートライブラリー

OS	ダイナミック・リンク	スタティック・リンク
Linux*	libiomp5.so	libiomp5.a
macOS*	libiomp5.dylib	libiomp5.a
Windows*	libiomp5md.lib libiomp5md.dll	なし

- OpenMP* を使用するオプションは、インテル製マイクロプロセッサーおよび互換マイクロプロセッサーの両方で利用可能ですが、両者では結果が異なります
- 両者の結果が異なる可能性のある OpenMP* 構造および機能は、ロック (内部的なものおよびユーザーが利用可能なもの)、SINGLE 構造、バリア (暗黙的および明示的)、並列ループ・スケジュール、リダクション、メモリーの割り当て、スレッド・アフィニティー、バインドです

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

インテル® コンパイラーの OpenMP* 関連のオプション

- /Qopenmp ; OpenMP* ディレクティブに基づいてコンパイラーがマルチスレッド・コードを生成するようにします。無効にするには /Qopenmp- を使用します
- /Qopenmp-stubs ; シーケンシャル・モードで OpenMP* プログラムをコンパイルします。OpenMP* ディレクティブは無視され、OpenMP* スタブ・ライブラリーがリンクされます (シーケンシャル)
- /Qopenmp-lib:<ver> ; リンクする OpenMP* ライブラリーのバージョンを選択します
  - compat – Microsoft* 互換の OpenMP* ランタイム・ライブラリーを使用します (デフォルト)
- ~~/Qopenmp-task:<arg> ; サポートする OpenMP* タスクモデルを選択します (廃止予定)~~
  - ~~omp – OpenMP* 3.0 タスクをサポートします (デフォルト)。intel – インテルのタスク・キューイングをサポートします~~
- /Qopenmp-threadprivate:<ver> ; 使用する threadprivate 実装を選択します
  - compat - GCC/Microsoft* 互換のスレッド・ローカル・ストレージを使用します。legacy - インテル互換の実装を使用します (デフォルト)
- /Qopenmp-simd ; OpenMP* SIMD コンパイルを有効にします。/Qopenmp を指定するとデフォルトで有効になります。無効にするには /Qopenmp-simd- を使用します
- /Qopenmp-offload[:<kind>] ; target プラグマの OpenMP* オフロードコンパイルを有効にします。このオプションは、インテル® MIC アーキテクチャーおよびインテル® グラフィックス・テクノロジーにのみ適用されます。/Qopenmp を指定するとデフォルトで有効になります。無効にするには /Qopenmp-offload- を使用します。target プラグマのデフォルトのデバイスを指定します
  - ~~host – オフロードの準備はしますが、ターゲットコードをホストシステムで実行します~~
  - mic - インテル® MIC アーキテクチャー、~~gfx – インテル® グラフィックス・テクノロジー~~

© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

まとめ

- 最初のセッションでは、OpenMP* が広く導入される背景と、これまでの歴史を振り返ってみました
- 異なるバージョンのインテル® コンパイラーや異なるコンパイラー間で OpenMP* を使用する注意点や制限についてもふれました
- OpenMP* は進化し続けています。今後も新たなハードウェアやその能力に合わせて多くの並列化向けの機能が追加されていきます

Intel、インテル、Intel ロゴ、Intel Inside、Intel Inside ロゴ、Xeon、Intel Xeon Phi は、アメリカ合衆国および / またはその他の国における Intel Corporation またはその子会社の商標です。
© 2020 iSUS *その他の社名、製品名などは、一般に各社の表示、商標または登録商標です。

ソフトウェア・セミナー