不均一メモリーアクセス (NUMA) とインテル® oneAPI スレッディング・ビルディング・ブロック (インテル® oneTBB)

インテル コーポレーション ソフトウェア・エンジニア Alex Katranov

インテル コーポレーション プリンシパル・エンジニア Michael Voss

概要

- インテル® oneAPI スレッディング・ビルディング・ブロック (インテル® oneTBB)
- インテル® oneTBB、不均一メモリーアクセス (NUMA) と構成の可能性
- インテル® oneTBB で NUMA に対応する既存の機能
- 将来への展望
- ケーススタディー: インテル® ディストリビューションの OpenVINO™ ツールキット
- まとめ

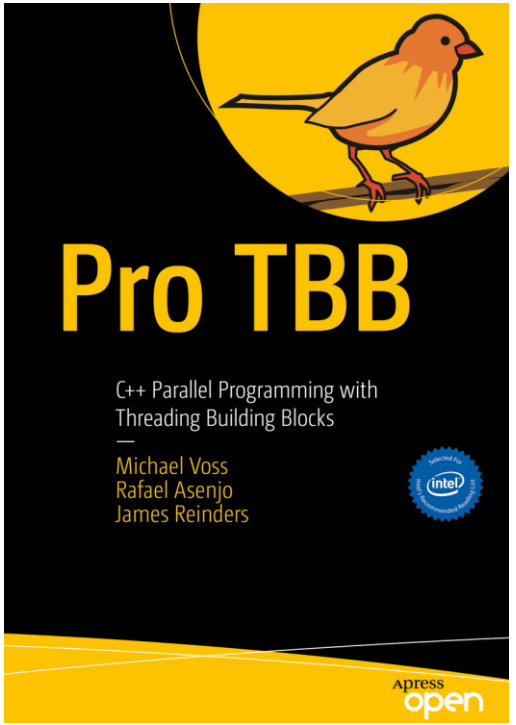
インテル® oneAPI スレッディング・ビルディング・ブロック (インテル® oneTBB)

インテル® oneTBB は並列プログラミング向けのポータブルな C++ ライブラリーです

GitHub* のオープンソース・プロジェクト、または Linux*、macOS*、Windows*、Android* などの商用サポート製品として入手できます

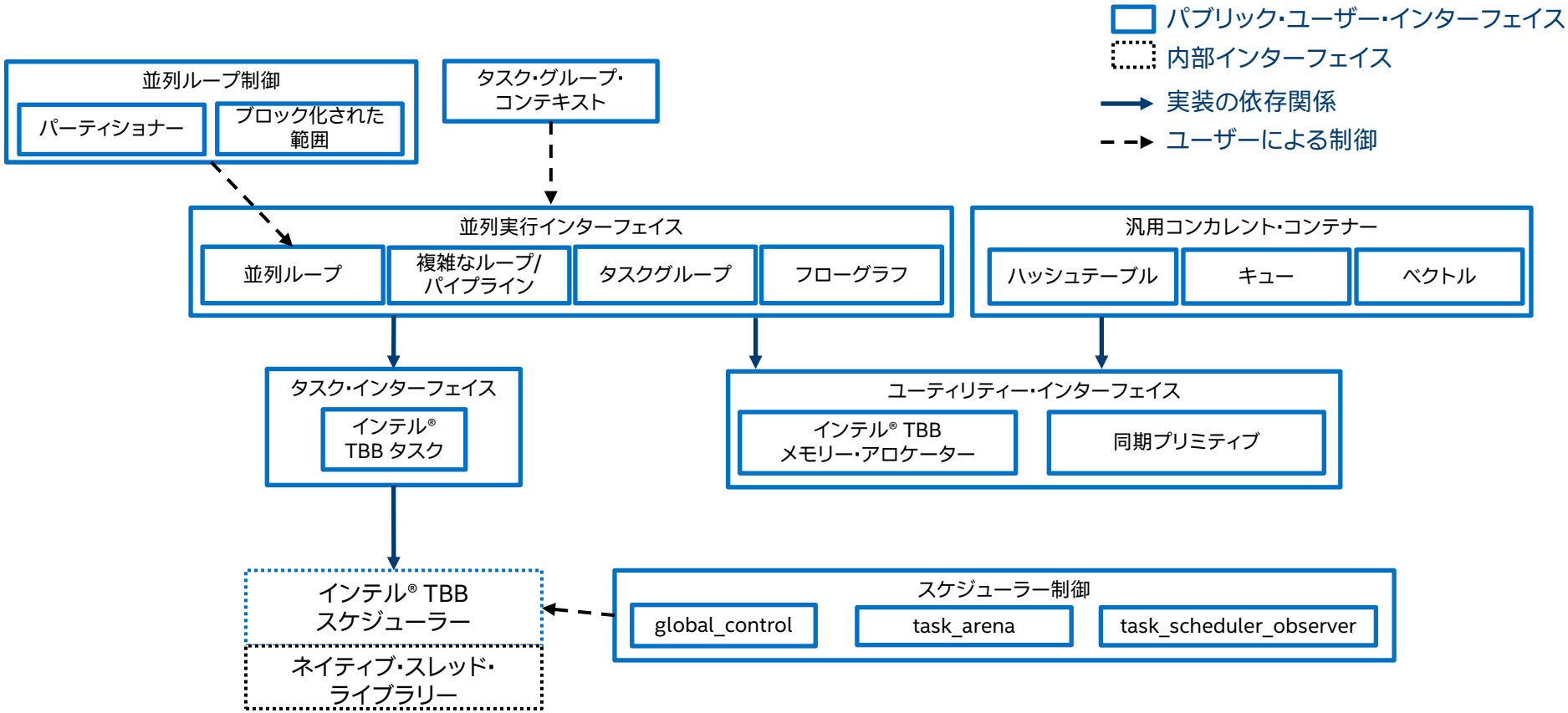
関連資料:

- https://software.intel.com/en-us/oneapi/onetbb (英語)
- threadingbuildingblocks.org (英語)
- 書籍『Pro TBB: スレッディング・ビルディング・ブロックを使用した C++ 並列プログラミング』



(注: インテル® TBB 2019 向けの例題)

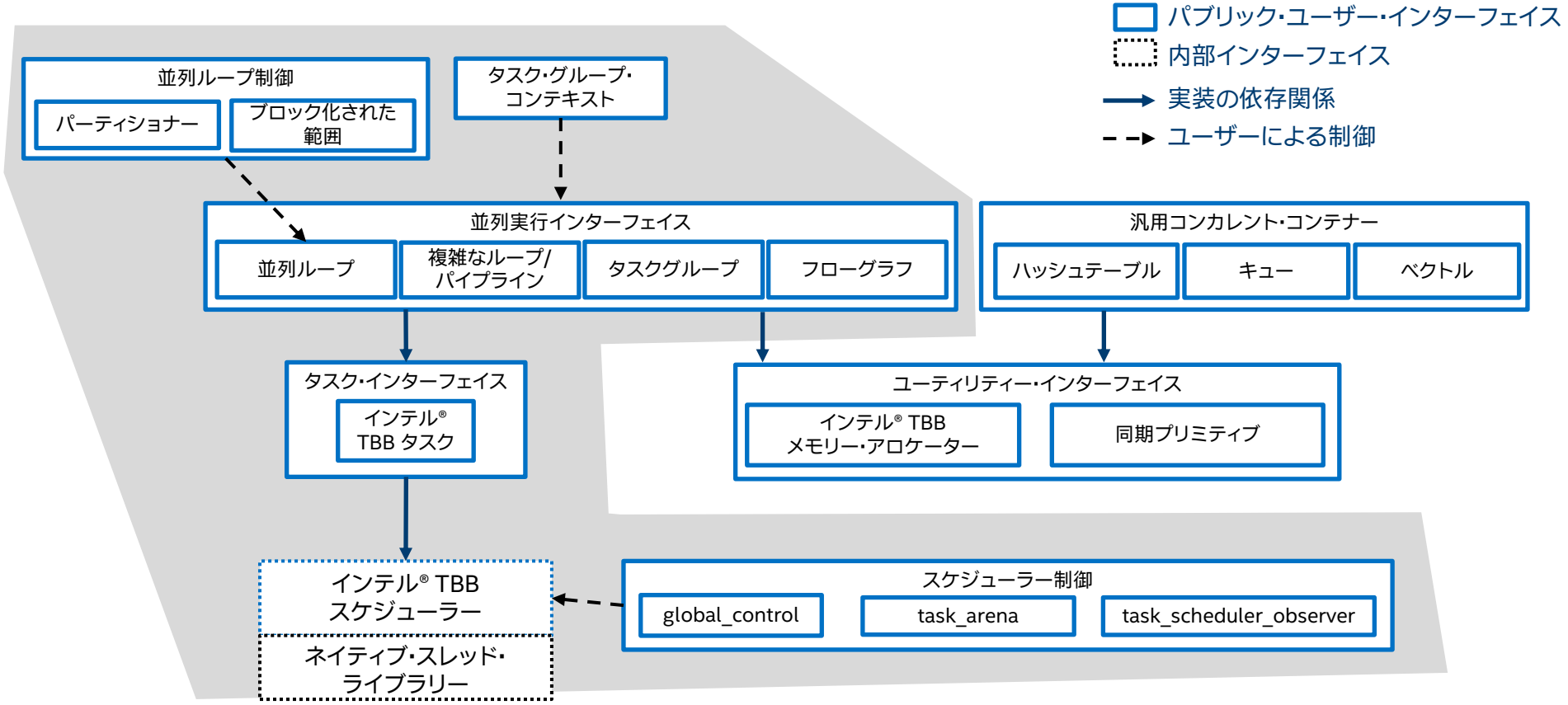
インテル® oneTBB アーキテクチャー:



スレッド化に OpenMP* API を使用する理由

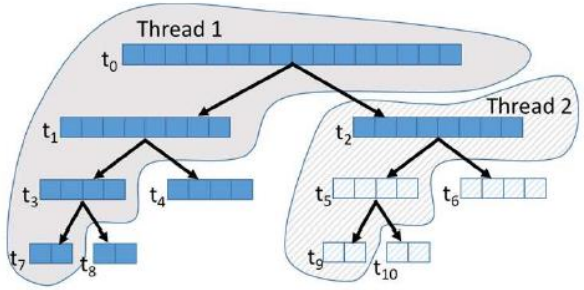
- OpenMP* API が適切に動作していればインテル® TBB に切り替える必要はありません
- しかし、インテル® TBB は HPC での利用において OpenMP* よりも優れている点があります:
  - 自然な C++ インターフェイス
  - 優れた構成の可能性により、入れ子になった並列処理 (ライブラリーでの並列処理を含む) のパフォーマンスが向上します
  - ワークロードのインバランスやマルチプログラム・システムのパフォーマンスを向上させるワークスチール・スケジューラー
- **例:** STAC-A2 とインテル® ディストリビューションの OpenVINO™ ツールキット。このプロジェクトは、OpenMP* を使用してスレッド化を実装しましたが、パフォーマンスの問題を解決するためインテル® TBB に切り替えました

インテル® oneTBB アーキテクチャー: 並列実行

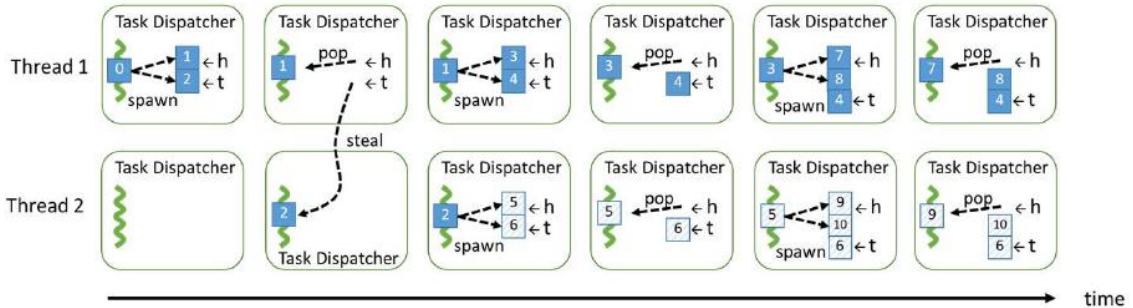


インテル® oneTBB、NUMA、および構成の可能性
インテル® TBB スケジューラーの手引き

```
tbb::parallel_for( 0, M, [=](int i) {
  for (int j = 0; j < M; ++j) {
    int c_index = i*M+j;
    for (int k = 0; k < M; ++k) {
      c[c_index] += a[i*M + k] * b[k*M+j];
    }
  }
});
```



(a) tasks as distributed by work-stealing across two threads



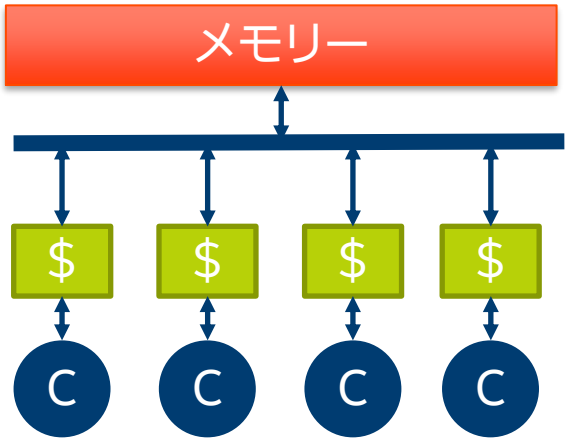
(b) the Task Dispatcher actions that acquire the tasks

構成の可能性
パフォーマンスの向上またはオーバーヘッドの増加?

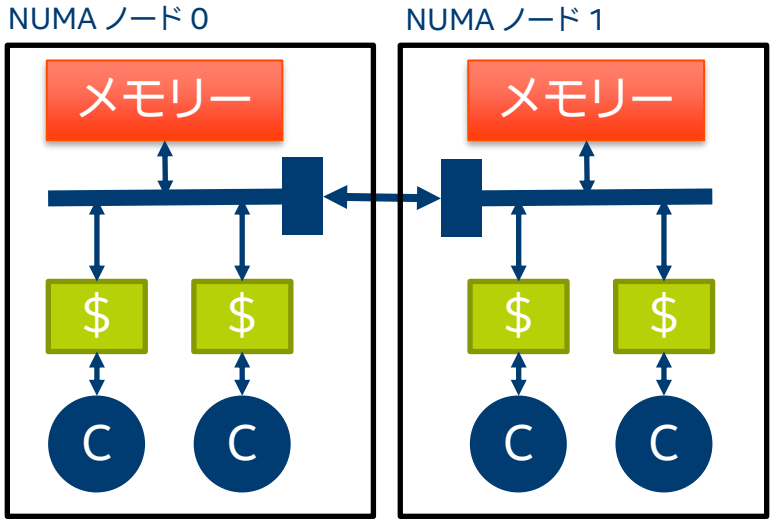
- 構成の可能性は、使用する並列コンポーネントがともに構成されている場合、並列モデルがどれだけうまく機能するかを示す尺度です
- インテル® TBB の動的な特性が優れた構成の可能性につながります
- 構成の可能性は、多くの実ワークロードでパフォーマンスを向上させます
- しかし、構成の可能性は無条件で得られるわけではありません。マイクロベンチマークやシンプルでフラットなバランスの取れたアプリケーションでは、オーバーヘッドか増加することがあります

インテル® oneTBB、NUMA、および構成の可能性
均一または不均一なメモリー・アクセス・アーキテクチャー

均一メモリー・アクセス・アーキテクチャー

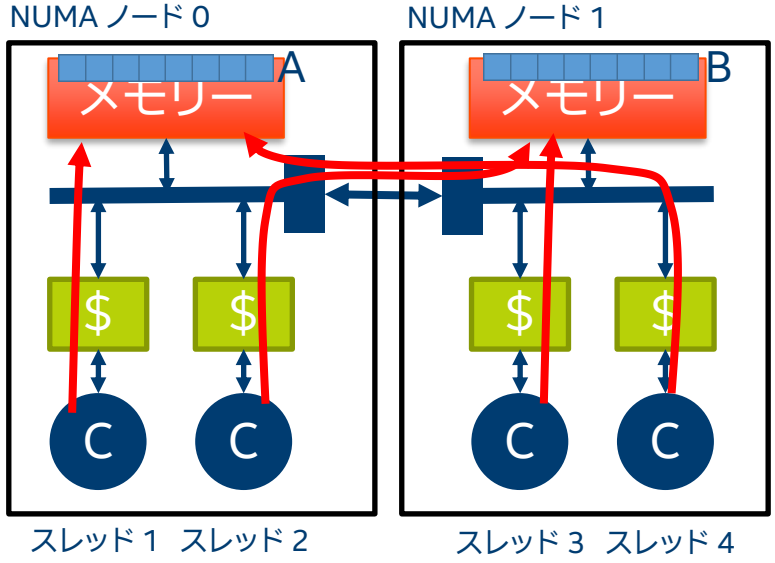


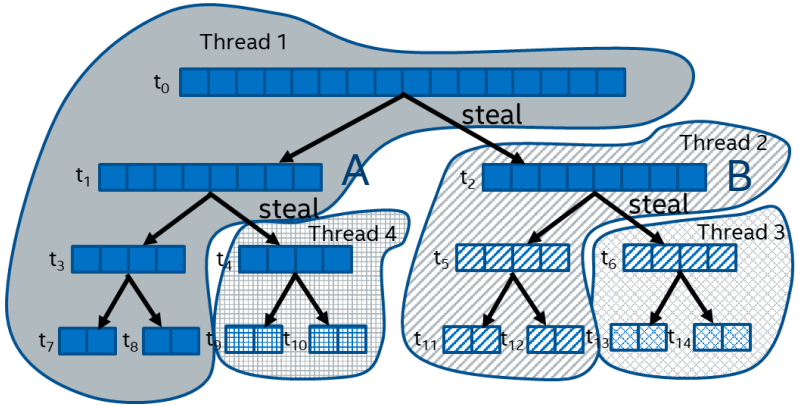
不均一メモリー・アクセス・アーキテクチャー



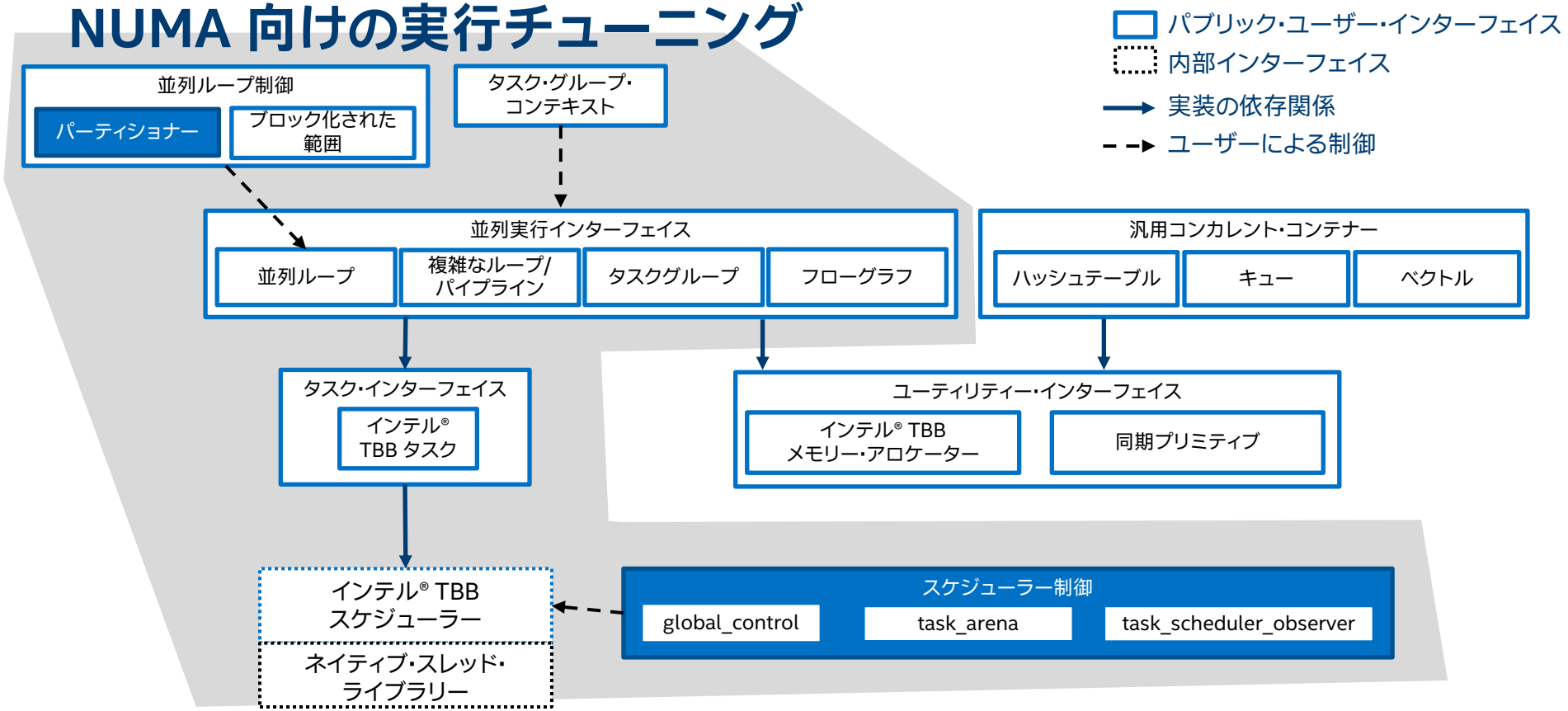
リモートメモリーへのアクセス
ローカルメモリーへのアクセスよりも時間がかかる

インテル® oneTBB、NUMA、および構成の可能性: 課題





インテル® oneTBB アーキテクチャー: NUMA 向けの実行チューニング



affinity_partitioner を使用して再スケジュール

- NUMA 向けの最適化は、ファーストタッチまたはネクストタッチのポリシーに依存します
- ページにタッチしたスレッドは、以降も同じページにアクセスし続けます
- affinity_partitioner によってこの機能を有効にできます

静的では構成の可能性が低下

動的では構成の可能性が向上

affinity_partitioner を使用して再スケジュール

- NUMA 向けの最適化は、ファーストタッチまたはネクストタッチのポリシーに依存します
- ページにタッチしたスレッドは、以降も同じページにアクセスし続けます
- affinity_partitioner によってこの機能を有効にできます

```
tbb::affinity_partitioner ap;
tbb::parallel_for( 0, N, [](int i) { /* ボディー */ }, ap);
// その他
tbb::parallel_for( 0, N, [](int i) { /* ボディー */ }, ap);
```

ワークスチールの動作を記録し、同じパーティションのオブジェクトを使用する後続の実行にスケジュールのヒントを提供します

静的では構成の可能性が低下

動的では構成の可能性が向上

affinity_partitioner を使用して再スケジュール

- NUMA 向けの最適化は、ファーストタッチまたはネクストタッチのポリシーに依存します
- ページにタッチしたスレッドは、以降も同じページにアクセスし続けます
- affinity_partitioner によってこの機能を有効にできます

```
tbb::affinity_partitioner ap;
tbb::parallel_for( 0, N, [](int i) { /* ボディー */ }, ap);
// その他
tbb::parallel_for( 0, N, [](int i) { /* ボディー */ }, ap);
```

ワークスチールの動作を記録し、同じパーティションのオブジェクトを使用する後続の実行にスケジュールのヒントを提供します

- スチールとチャンクサイズの自動選択を可能にし、構成の可能性を維持しますが、NUMA に最適なパフォーマンスは提供できません

静的では構成の可能性が低下

動的では構成の可能性が向上

静的なスケジュールを行うには static_partitioner を使用します

- static_partitioner を使ってスケジューラーをさらに制限できます
- static_partitioner は、参加するスレッドごとにほぼ同サイズのチャンクを作成します。チャンクはそれ以上分割されることはありません。そして、i 番目のチャンクはスケジューラーのヒントを使用して i 番目のスレッドに関連付けられます

静的では構成の可能性が低下

動的では構成の可能性が向上

静的なスケジュールを行うには static_partitioner を使用します

- static_partitioner を使ってスケジューラーをさらに制限できます
- static_partitioner は、参加するスレッドごとにほぼ同サイズのチャンクを作成します。チャンクはそれ以上分割されることはありません。そして、i 番目のチャンクはスケジューラーのヒントを使用して i 番目のスレッドに関連付けられます

```
tbb::parallel_for( 0, N, [](int i) { /* ボディー */ }, tbb::static_partitioner());
// その他
tbb::parallel_for( 0, N, [](int i) { /* ボディー */ }, tbb::static_partitioner());
```

静的では構成の可能性が低下

動的では構成の可能性が向上

静的なスケジュールを行うには static_partitioner を使用します

- static_partitioner を使ってスケジューラーをさらに制限できます
- static_partitioner は、参加するスレッドごとにほぼ同サイズのチャンクを作成します。チャンクはそれ以上分割されることはありません。そして、i 番目のチャンクはスケジューラーのヒントを使用して i 番目のスレッドに関連付けられます

```
tbb::parallel_for( 0, N, [](int i) { /* ボディー */ }, tbb::static_partitioner());
// その他
tbb::parallel_for( 0, N, [](int i) { /* ボディー */ }, tbb::static_partitioner());
```

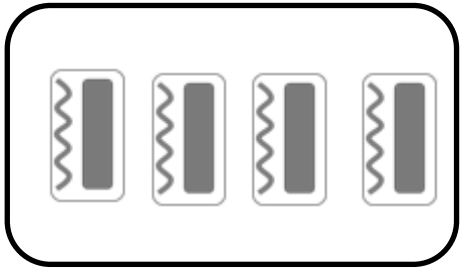
- ヒントを使用するとチャンクをスチールすることはできますが、それ以上は分割できません
- チャンクサイズを最大化するため、ロードバランスの可能性が制限されます

静的では構成の可能性が低下

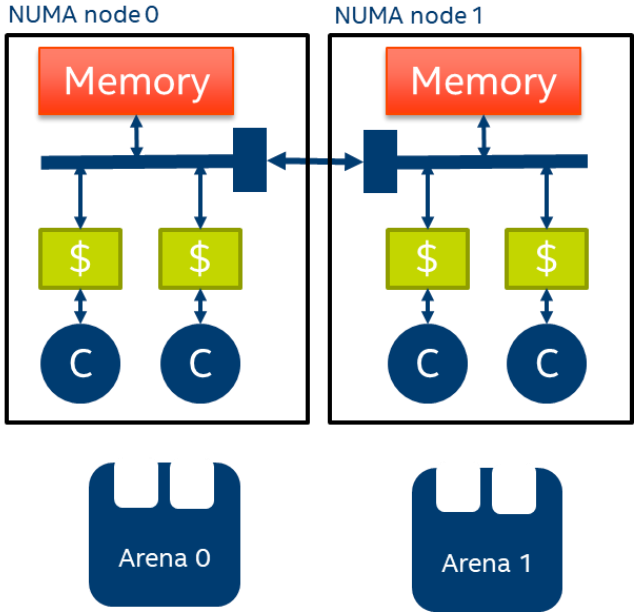
動的では構成の可能性が向上

tbb::task_arena を使用してワークを分離します

グローバル・スレッド・プール / マーケット



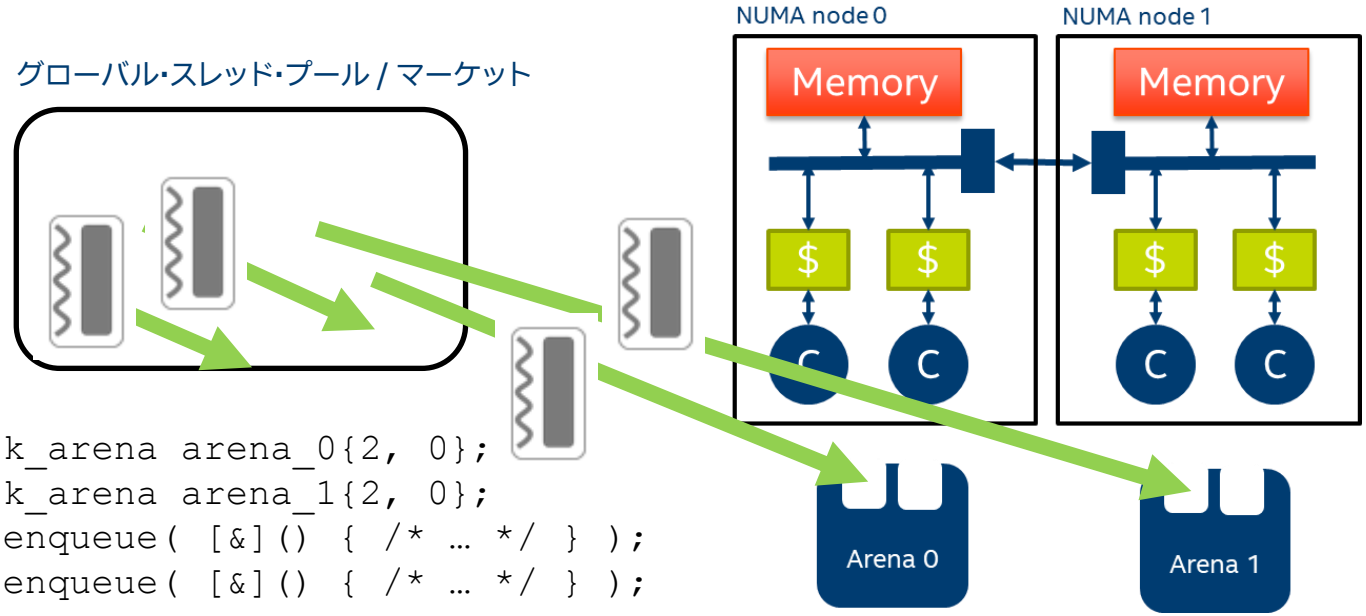
```
tbb::task_arena arena_0{2, 0};
tbb::task_arena arena_1{2, 0};
```



静的では構成の可能性が低下

動的では構成の可能性が向上

tbb::task_arena を使用してワークを分離します

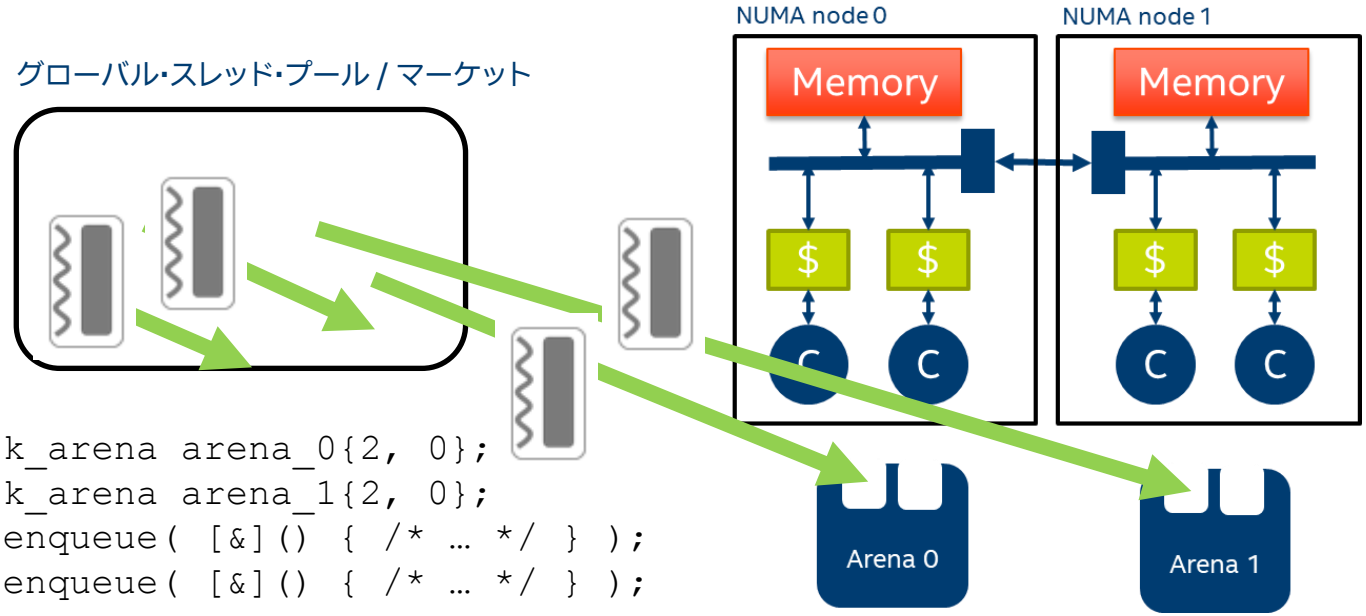



```
tbb::task_arena arena_0{2, 0};
tbb::task_arena arena_1{2, 0};
arena_0.enqueue( [&]() { /* … */ } );
arena_1.enqueue( [&]() { /* … */ } );
```

静的では構成の可能性が低下

動的では構成の可能性が向上

tbb::task_arena を使用してワークを分離します

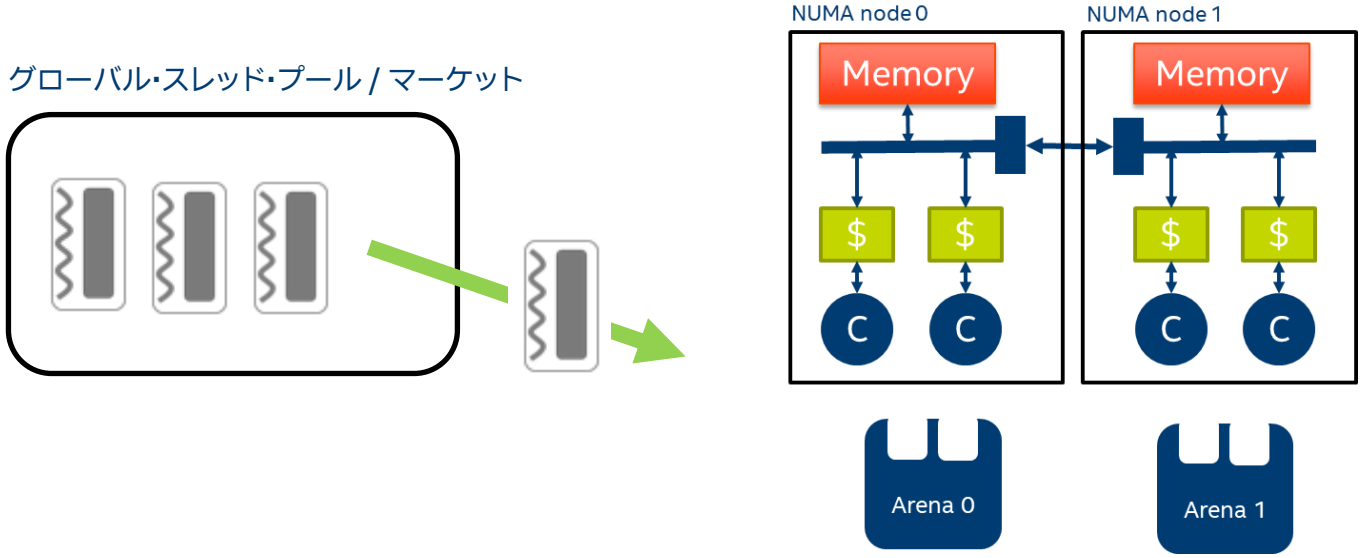



```
tbb::task_arena arena_0{2, 0};
tbb::task_arena arena_1{2, 0};
arena_0.enqueue( [&]() { /* … */ } );
arena_1.enqueue( [&]() { /* … */ } );
```

静的では構成の可能性が低下

動的では構成の可能性が向上

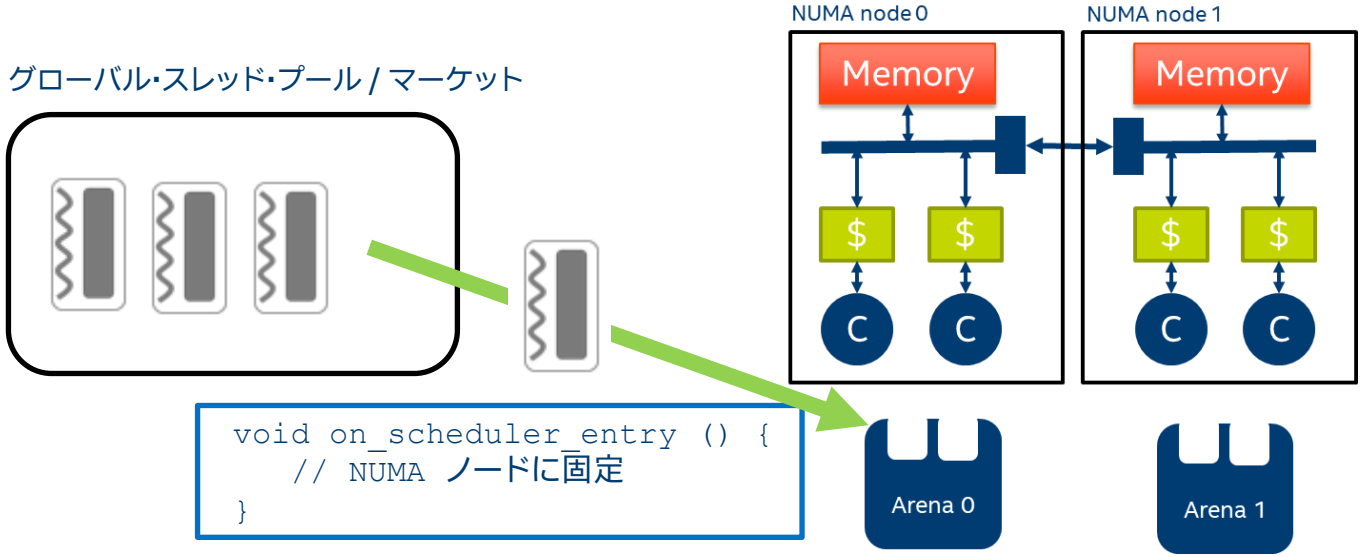
スレッドを task_arena に固定するには task_scheduler_observers を使用します



静的では構成の可能性が低下

動的では構成の可能性が向上

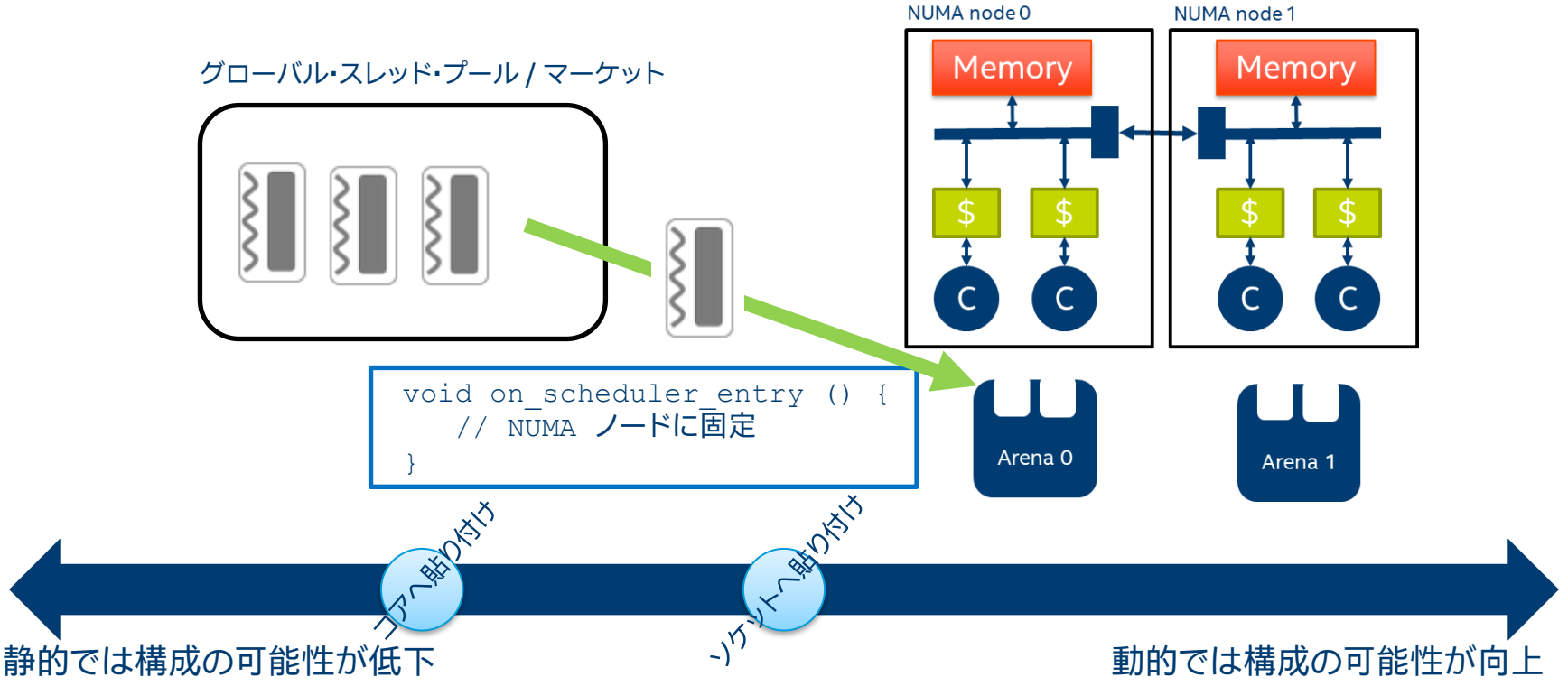
スレッドを task_arena に固定するには task_scheduler_observers を使用します



静的では構成の可能性が低下

動的では構成の可能性が向上

スレッドを task_arena に固定するには task_scheduler_observers を使用します



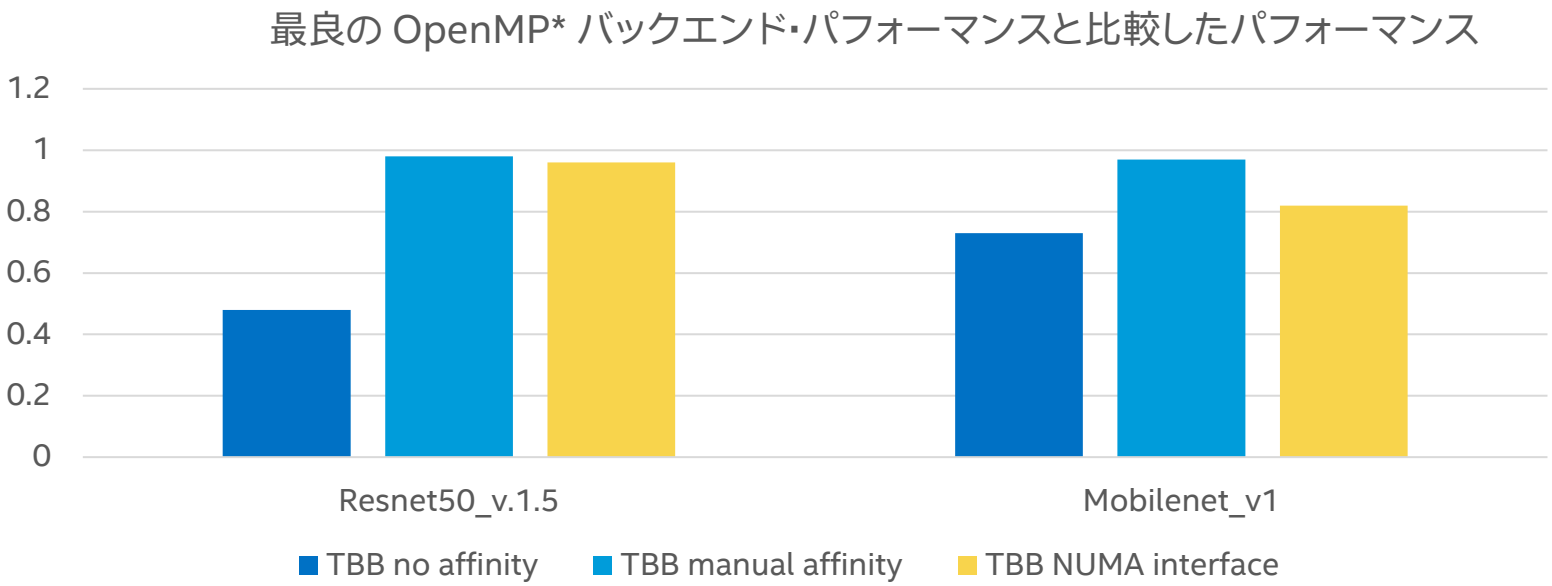
インテル® TBB 2020 の NUMA サポート: さらに簡単に

```
std::vector<int> numa_indexes = tbb::info::numa_nodes();

std::vector<tbb::task_arena> arenas(numa_indexes.size());
std::vector<tbb::task_group> task_groups(numa_indexes.size());

for(unsigned j = 0; j < numa_indexes.size(); j++) {
  arenas[j].initialize( tbb::task_arena::constraints(numa_indexes[j]) );
  arenas[j].execute( [&task_groups, &j](){
    task_groups[j].run([](){/*some parallel stuff*/});
  });
}

for(unsigned j = 0; j < numa_indexes.size(); j++) {
  arenas[j].execute([&task_groups, &j](){ task_groups[j].wait(); });
}
```

ケーススタディー: インテル® ディストリビューションの OpenVINO™ ツールキット

- OpenVINO™ ツールキットは、エッジでのディープラーニング推論を可能にします
- コンピューター・ビジョン・アクセラレーター (CPU、GPU、インテル® Movidius™ ニューラル・コンピュート・スティック、FPGA) 全体でヘテロジニアス実行をサポートします
- 以前のバージョンでは、CPU プラグインに OpenMP* API を使用していましたが、インテル® TBB オフロードでは構成の可能性が重要なパイプラインで最大 2 倍のパフォーマンスを達成できます

インテル® oneTBB のアフィニティーを使用して、アフィニティーが重要となる非パイプラインで OpenMP* と一致させます

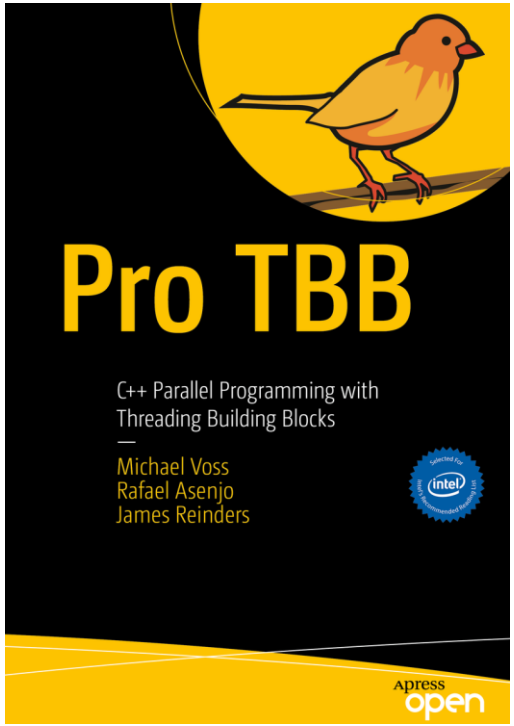


Example networks from the MLPerf Inference benchmark (ver0.5). Executed on system w/ two Intel® Xeon® Platinum 9242 Processors (4 NUMA nodes). Ubuntu* 18, running OpenVINO™ toolkit R3-based experimental private branch with early version of TBB 2020. Product interception: OpenVINO™ toolkit 2019 R4.

まとめ

- インテル® oneTBB には NUMA 向けのチューニングに利用できる機能が備わっています
- しかし、スケジューラーを制限すると構成の可能性を犠牲にすることがあります
- ここでは、affinity_partitioner、static_partitioner、task_arena および task_scheduler_observer について説明しました
- また、これを容易にする将来の機能について説明しました
- OpenVINO™ ツールキットにおける NUMA チューニングの影響を示しました

サンプルコードのソースは GitHub* で入手できます



書籍の無料のデジタルコピーは、
https://www.apress.com/gp/book/9781484243978
(英語) で入手できます。

パーティショナーのタイプ

“インテル® TBB アルゴリズムのチューニング: 粒度、局所性、並列性、および決定性”:
図 16-14
https://github.com/Apress/pro-TBB/blob/master/ch16/fig_16_14.cpp (英語)

分離のため task_arena を使用

“正当性とパフォーマンスのためのワーク分離” を参照:
図 12-10
https://github.com/Apress/pro-TBB/blob/master/ch12/fig_12_10.cpp (英語)

task_scheduler_observer でアフィニティーを制御

“スレッドとコアおよびタスクとスレッドのアフィニティーを制御”:
図 13-1
https://github.com/Apress/pro-TBB/blob/master/ch13/fig_13_01.cpp (英語)

NUMA システムのチューニング

“NUMA アーキテクチャーでのインテル® TBB”:
図 20-12
https://github.com/Apress/pro-TBB/blob/master/ch20/fig_20_10.cpp (英語)

法務上の注意書きと最適化に関する注意事項

インテル® テクノロジーの機能と利点はシステム構成によって異なり、対応するハードウェアやソフトウェア、またはサービスの有効化が必要となる場合があります。実際の性能はシステム構成によって異なります。

絶対的なセキュリティーを提供できるコンピューター・システムはありません。

テストでは、特定のシステムでの個々のテストにおけるコンポーネントの性能を文書化しています。ハードウェア、ソフトウェア、システム構成などの違いにより、実際の性能は掲載された性能テストや評価とは異なる場合があります。性能やベンチマーク結果について、さらに詳しい情報をお知りになりたい場合は、www.intel.com/benchmarks (英語) を参照してください。

性能に関するテストに使用されるソフトウェアとワークロードは、性能がインテル® マイクロプロセッサー用に最適化されていることがあります。SYSmark* や MobileMark* などの性能テストは、特定のコンピューター・システム、コンポーネント、ソフトウェア、操作、機能に基づいて行ったものです。結果はこれらの要因によって異なります。製品の購入を検討される場合は、他の製品と組み合わせた場合の本製品の性能など、ほかの情報や性能テストも参考にして、パフォーマンスを総合的に評価することをお勧めします。詳細については、 http://www.intel.com/performance (英語) を参照してください。

インテル® アドバンスト・ベクトル・エクステンション (インテル® AVX)* は、特定のプロセッサー演算で高いスループットを示します。

プロセッサーの電力特性の変動により、AVX 命令を利用すると、a) 一部の部品が定格周波数未満で動作する、b) インテル® ターボ・ブースト・テクノロジー 2.0 を使用する一部の部品が任意または最大のターボ周波数に達しない可能性があります。実際の性能はハードウェア、ソフトウェア、およびシステム構成によって異なります。詳細は、http://www.intel.com/go/turbo (英語) をご覧ください。

インテル® コンパイラーでは、インテル® マイクロプロセッサーに限定されない最適化に関して、他社製マイクロプロセッサー用に同等の最適化を行えないことがあります。これには、インテル® ストリーミング SIMD 拡張命令 2、インテル® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサーに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサー依存の最適化は、インテル® マイクロプロセッサーでの使用を前提としています。インテル® マイクロアーキテクチャーに限定されない最適化のなかにも、インテル® マイクロプロセッサー用のものがあります。この注意事項で言及した命令セットの詳細については、該当する製品のユーザー・リファレンス・ガイドを参照してください。

本資料に記載されている TCO などのコスト削減シナリオは、状況固有の多数の変動要因が加わることで、特定のインテル製品の購入が今後のコストとコスト削減にどのように影響するかについて理解を深めることができるようにするためのものです。状況が変わると、特定の製品の使用と導入に関連する説明されていないコストが発生する場合があります。本資料の内容は、一定レベルのコストを保証または確約するものではありません。

インテルは、本資料で参照しているサードパーティーのベンチマーク・データまたはウェブサイトについて管理や監査を行っていません。本資料で参照しているウェブサイトにアクセスし、本資料で参照しているデータが正確かどうかを確認してください。

質疑応答

intel
Software