インテル® oneAPI スレッディング・ビルディング・ブロック

クロスアーキテクチャーのタスクベースのプログラミングの最適化

インテル コーポレーション
テクニカル・コンサルティング・エンジニア
James Tullos

intel®

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)

高速なアプリケーションのための高度なスケーリング

並列処理向けの柔軟な C++ ライブラリー

開発者に深いハードウェアの知識がなくてもアプリケーションで並列処理を表現できる簡単な方法

将来もアプリケーションのパフォーマンスをスケーリング

現在および将来のプラットフォームで計算負荷の高いワークロードのパフォーマンスを効率良く並列化およびスケーリング

ほかのスレッド化パッケージとの互換性

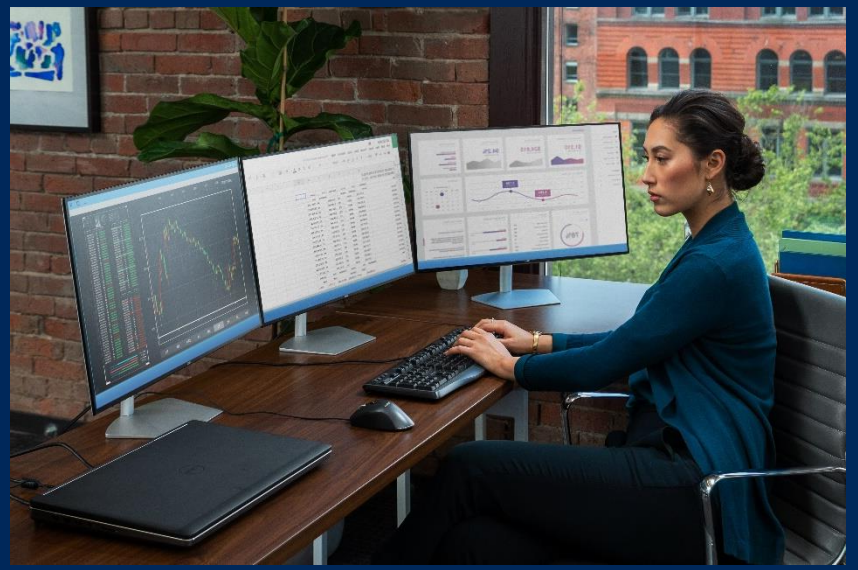
レガシーコードを保ちながら新しい実装にはインテル® oneTBB を使用、
ほかのスレッド化パッケージとシームレスに共存

アプリケーションの構成可能性を簡素化/拡張

CPU 上で構成可能でスケーラブルな並列処理を作成、強化されたアクセラレーター処理で拡張可能

intel oneTBB

並列プログラミングで最も広く利用されている C++ ライブラリーの 1 つ



xlsoft.com/jp/products/intel/perflib/tbb/

インテル® oneAPI ベース・ツールキットに含まれる

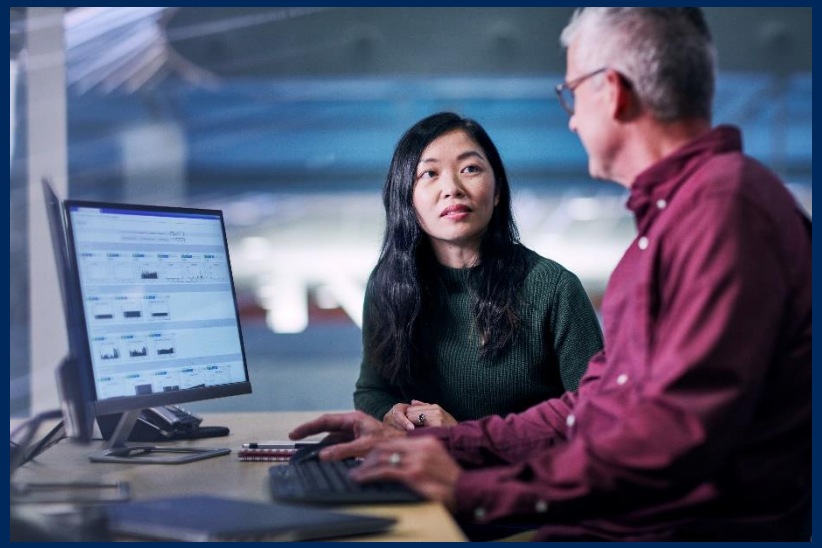
スケーラブルな並列プログラミングを容易に作成
インテル® oneTBB によるスレッド化

スレッド化を利用してマルチコア・パフォーマンスを活用

- CPU、GPU、FPGA、およびその他のアクセラレーターで計算負荷の高い作業を並列化
- 共有メモリー並列プログラミング向けの最新の C++ ライブラリーを使用した高レベルでシンプルなソリューションを提供
- 優れた移植性、構成の容易性、コスト、アプローチ、および将来にわたるスケーラビリティー

バージョン 2021 の新機能

- コードベースを刷新、使いやすさを向上、C++ 標準に準拠、ライブラリーを簡素化(このバージョンは、以前のバージョンと下位互換性がありません)[1]
- アクセラレーターへの対応を強化—再開可能なタスク、NUMA 対応のタスク領域など
- NUMA 対応アプリケーションの開発を簡素化して、構成可能でスケーラブルなパフォーマンスを実現するため task_arena インターフェイスを拡張
- 順序付きコンカレント・コンテナー、機能ノードと再開可能なタスクの相対的優先順位をサポートするフローグラフ API を完全にサポート
- インテル® oneTBB の Python* モジュールを Python2 から Python3 に更新
- フローグラフなどの並列アルゴリズムでインテル® Inspector をサポート



詳細
xlsoft.com/jp/products/intel/perflib/tbb/

インテル® oneAPI ベース・ツールキット
に含まれる

1.完全な機能リストおよび新しい機能へマップする手順は、
TBB の刷新: 背景、変更点および更新 (英語) を参照

TBB の歴史

- インテルで最初に成功したオープンソース・ソフトウェア・プロジェクト
- 当初 GPLv2 の下で提供された後、Apache* に変更
- OpenMP* などのほかのプログラミング・モデルから着想を得ている
- 現在はインテル® マス・カーネル・ライブラリーおよびインテル® データ・アナリティクス・アクセラレーション・ライブラリーを含む多くの製品で利用されている

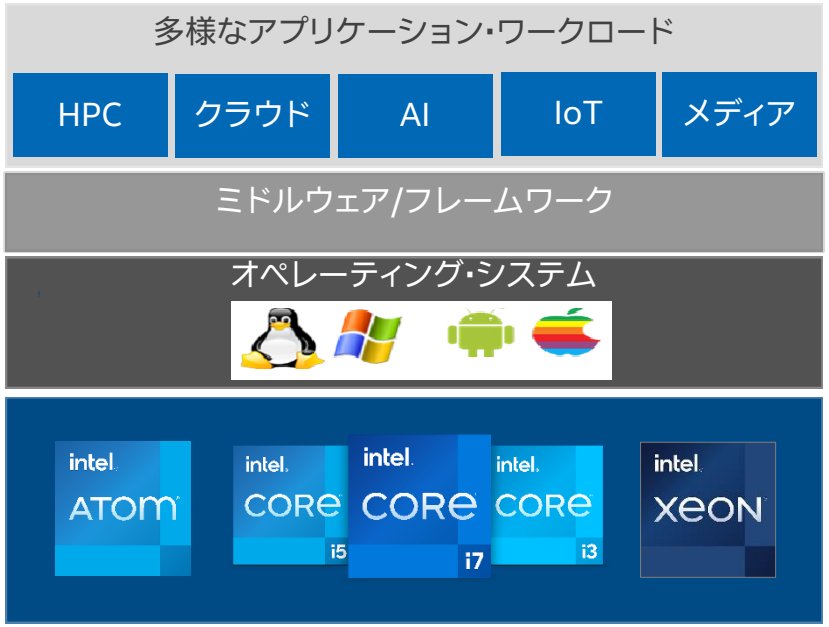
マルチスレッド化とヘテロジニアス・コンピューティングを容易にする
インテル® oneAPI スレッディング・ビルディング・ブロック (インテル® oneTBB)

インテル® oneTBB とは

1 つまたは複数のデバイス上のすべての CPU を利用してアプリケーションの並列化を容易にし、GPU、FPGA およびその他のアクセラレーターにオフロードするコードと相互運用可能な、高度にテンプレート化された C++ ライブラリー

インテル® oneTBB を使用する理由

- ハイパフォーマンス
- 簡単に使える API
- 開発期間を短縮
- プロダクション環境にも対応/スケーラブル



詳細: xlsoft.com/jp/products/intel/perflib/tbb/
フォーラム (英語):
http://software.intel.com/en-us/forums/intel-threading-building-blocks

インテル® oneTBB の入手

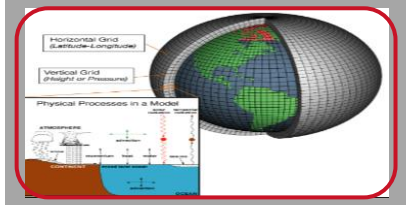
- インテル® oneAPI ベース・ツールキット (英語)
- スタンドアロン・コンポーネント (英語)
- オープンソース・バージョン (英語)

主なアプリケーション

- アニメーション・レンダリング
- 数値天気予報
- 海洋学と天体物理学
- AI/オートメーション
- 遺伝子工学
- 医療アプリケーション (画像処理、MRI 再構成)
- リモート・センシング・アプリケーション
- 社会経済学
- 金融部門 (デリバティブの価格付け、統計)
- 大量のデータファイルの更新
- ビッグデータの問題

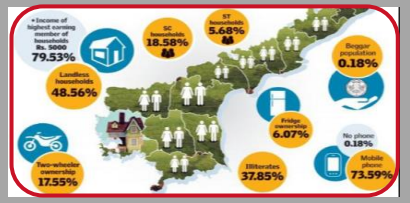
マルチスレッド化のメリットがあるアプリケーション

問題を並列に実行できるタスクに分割可能なアプリケーションや一部の数学/解析問題のように
問題そのものが超並列であるアプリケーション

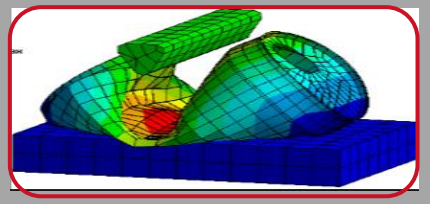


数値天気予報
- 数学的モデリング
- 今後の天気の予測
- データ同化

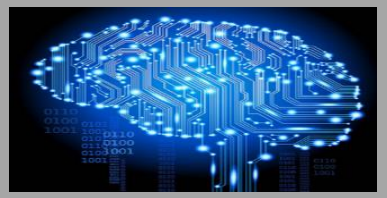
海洋学と天体物理学
- 海洋資源の研究
- PIC、PM、N 体シミュレーション
- 天体物理学の研究



社会経済学
- 国家/世界の経済モデリング
- シナリオ計算/経済モデルの最適化

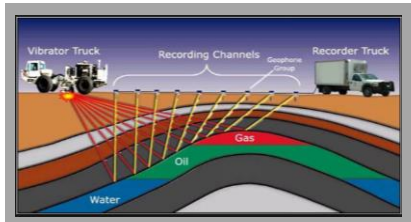


有限要素解析
- マルチフィジックス問題
- 船、ダム、超音速ジェット機などの巨大構造物の設計
- 偏微分方程式 (PDE) の計算

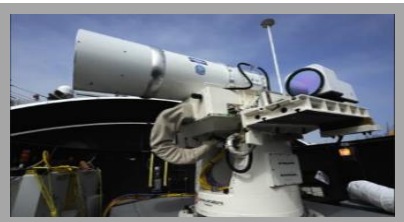


AI/オートメーション
- 画像処理
- エキスパート・システム
- 自然言語処理 (NLP)
- パターン認識

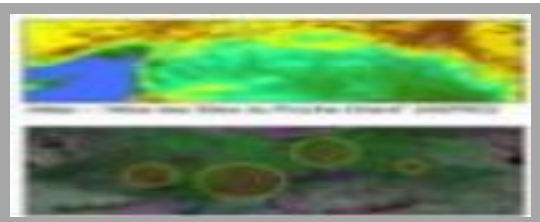
遺伝子工学
- DNA シーケンス解析



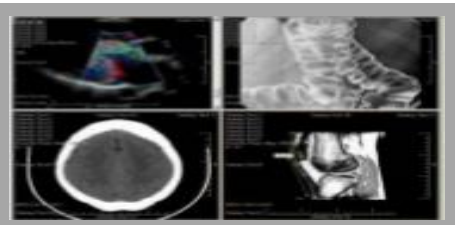
地震探査
- 地震波を使用した地球の特性の推測
- センサー解析



兵器研究/防衛
- 核兵器の性能証明
- プルトニウムの研究



リモート・センシング・アプリケーション
農業/林業で特別なフォーマットの読み取りに使用
- センサー画像データ
- ジオリファレンス情報
- センサーメタデータ



医療アプリケーション
- 医用画像処理
- 人体/脳の X 線検査
- MRI 再構成
- X 線画像の椎骨検出/分節
- 脳のファイバー・トラッキング

エネルギー資源探査
- 石油、天然ガスなどのエネルギー資源に関する情報の収集と管理
- 世界的なエネルギー危機、原子炉の安全性の記録

利点
インテル® oneTBB のスレッド化モデルを利用

- スレッドを操作する代わりにタスクを指定、論理タスクをスレッドにマップ (入れ子構造の並列処理を完全サポート)
- 実証済みの効率良い並列処理パターン
- ワークスチールを使用して実行時間が不明なタスクのロードバランスをサポート、低オーバーヘッドのポリモーフィズムを活用
- フローグラフ機能により依存性とデータフロー・グラフを簡単に表現
- 高レベルの並列アルゴリズム、コンカレント・コンテナー、低レベルのビルディング・ブロック (スケーラブル・メモリー・アロケーター、ロック、アトミック操作など) を提供

アプリケーション: 並列処理の 3 つの共通レイヤー

高水準のインテル® oneTBB 並列実行インターフェイスへのマップ方法

インテル® oneTBB は、これらの異なる並列処理レイヤーにそれぞれ高水準のインターフェースを提供して、
開発者が同じライブラリーを使用してすべてのレイヤーを活用できるようにする

メッセージ駆動レイヤー (インテル® oneTBB フローグラフ)

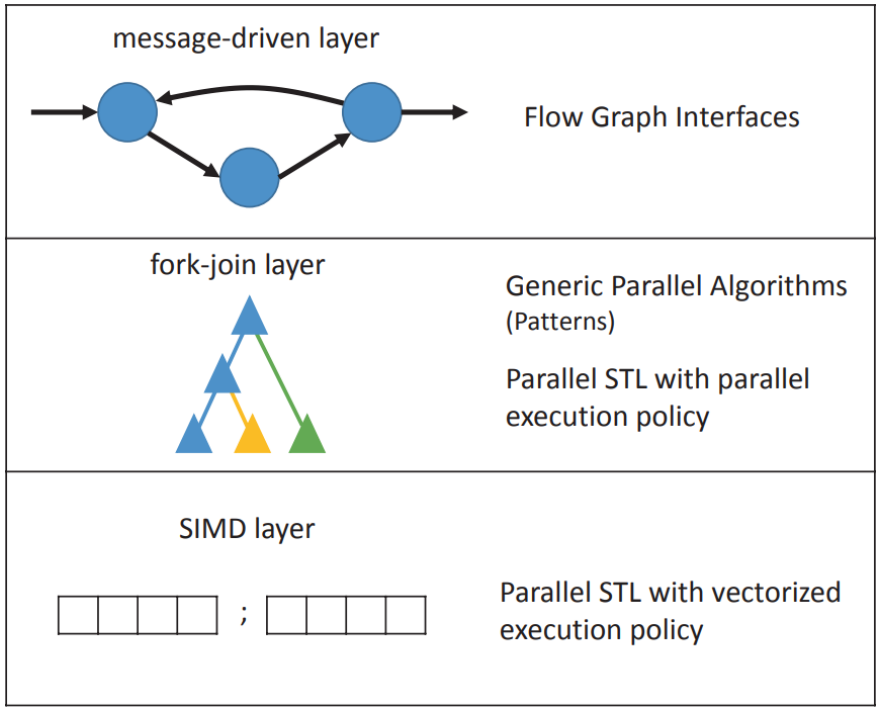
明示的なメッセージを利用して互いに通信する比較的大規模な計算として構造化された並列処理をキャプチャーします。このレイヤーの一般的なパターンには、ストリーミング・グラフ、データフロー・グラフ、依存関係グラフが含まれます。インテル® oneTBB では、これらのパターンはフローグラフ・インターフェイスによりサポートされます。

fork-join レイヤー - (インテル® oneTBB タスク)

シリアル計算が一連の並列タスクに分岐し、並列サブ計算が完了したときにのみ継続するパターンをサポートします。fork-join パターンの例には、機能並列処理 (タスク並列処理)、並列ループ、並列リダクション、パイプラインが含まれます。インテル® oneTBB は、汎用並列アルゴリズムでこれらのパターンをサポートします。

Single Instruction Multiple Data (SIMD) レイヤー

同じ操作を同時に複数のデータ要素に適用することによりデータ並列処理を活用するレイヤーです。この並列処理タイプは通常、各プロセッサー・コアで利用可能なベクトルユニットを使用するベクトル拡張命令 (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)、インテル® AVX2、インテル® AVX-512) を使用して実装されます。これらの拡張機能を活用するベクトル実装を提供するすべてのインテル® oneTBB ディストリビューションには Parallel STL 実装が含まれます。

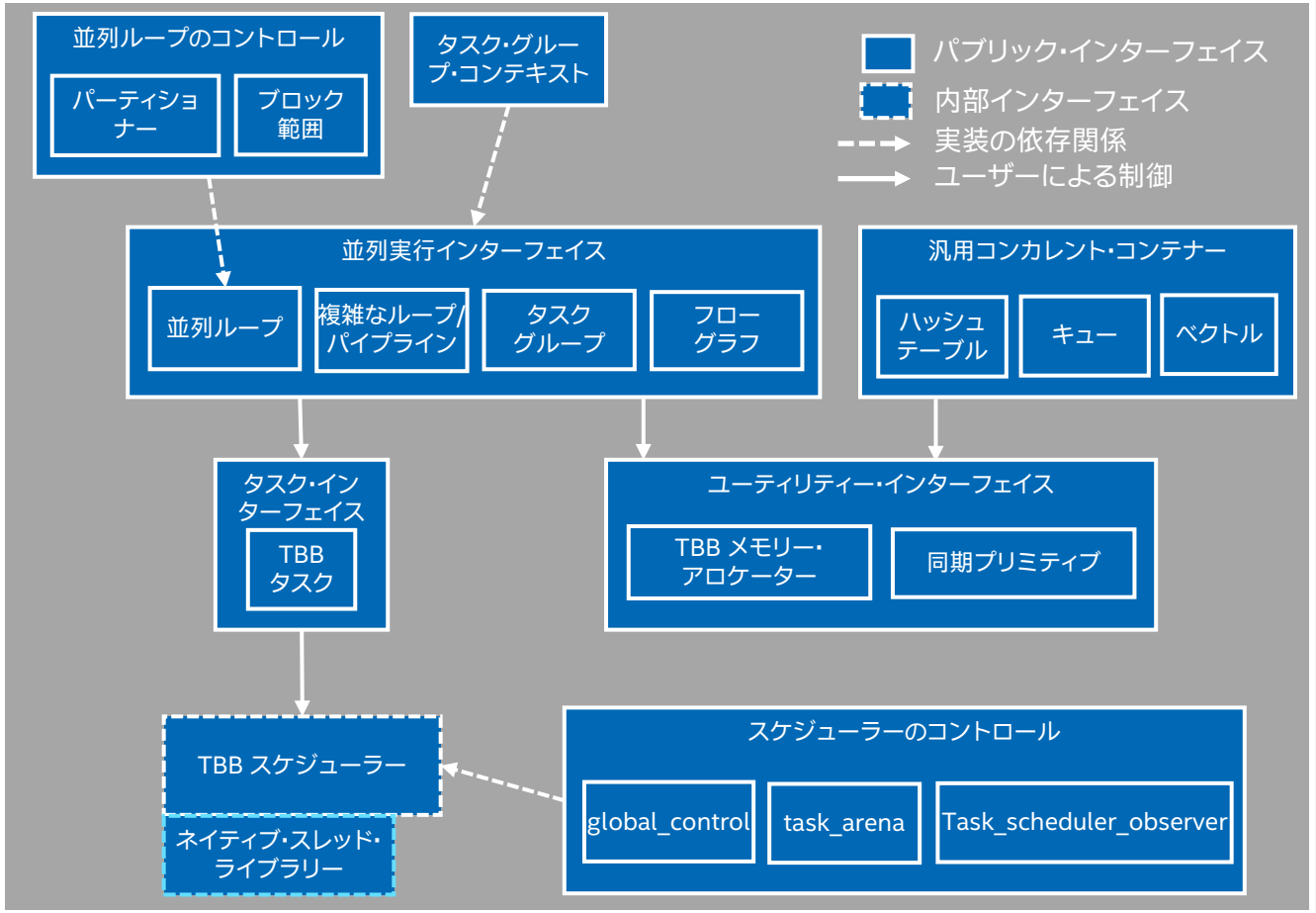


出典: Pro TBB (英語)

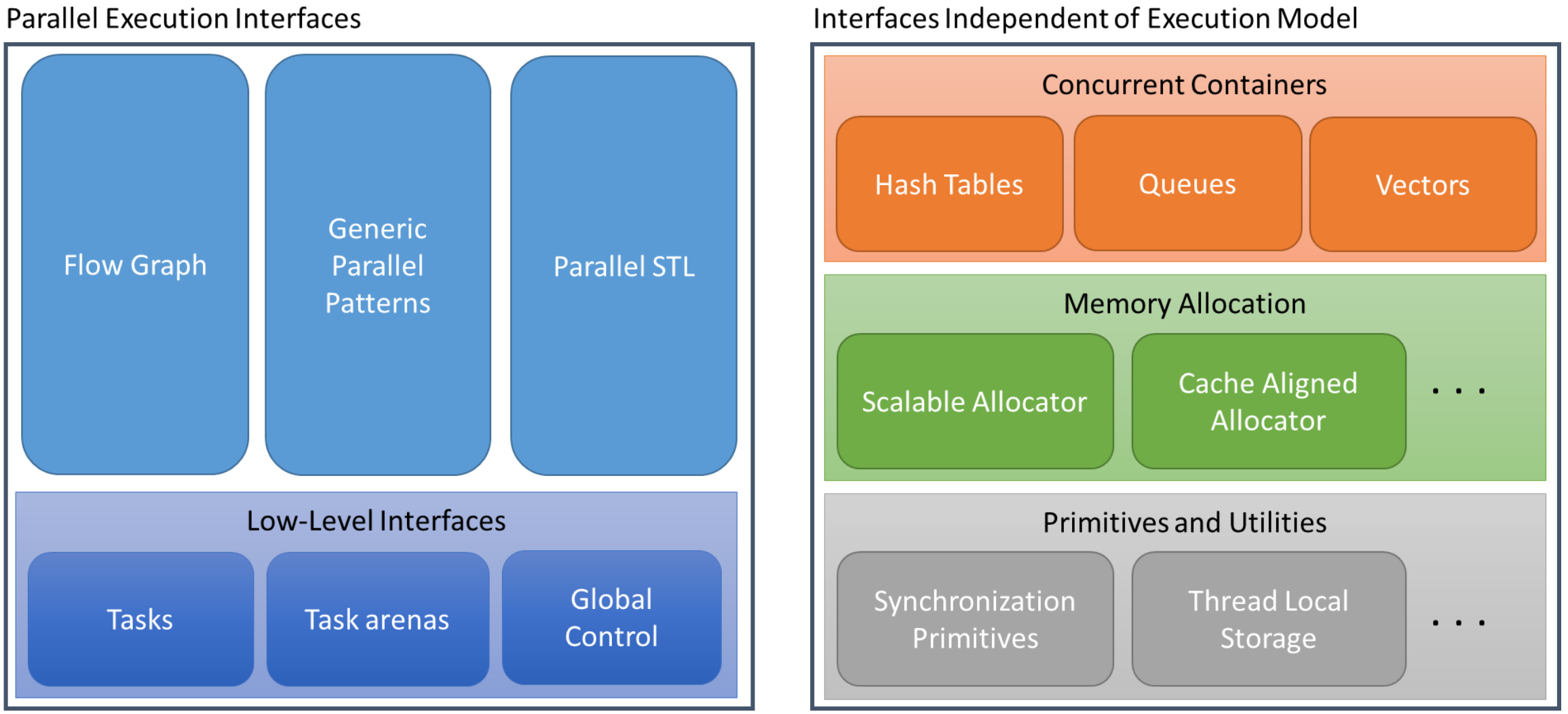
インテル® oneTBB アーキテクチャーの概要

高度にスケーラブルなスレッド・アプリケーションを開発するビルディング・ブロックのコレクション

- インテル® oneTBB は高水準の並列実行インターフェイスを含む
  - **並列ループ:** parallel_for、parallel_reduce、その他
  - **複雑なアルゴリズム:** パイプライン、タスクグループ
  - **フローグラフ:** データフローに依存しないグラフの表現
- すべて TBB タスク上に作成され、これらのタスクは TBB スケジューラー上で実行される
- スケジューラーのコントロールと並列ループのコントロールでパフォーマンスを厳密に制御
- コンカレント・コンテナー – キュー、ベクトル、その他はスレッドセーフでスレッド化しやすい
- スケーラブル・メモリー・アロケーター、同期プリミティブ



インテル® oneTBB の機能



インテル® oneTBB API の変更点

このリリースでインテル® oneTBB を刷新

- インテル® oneTBB 2021 では互換性を大幅に改善
- 最新の C++ (C++11) に対応
- 保守の負担を減らすため一部の冗長な機能を削除

変更点

- 最新の C++ に含まれる機能の削除
- PPL (Microsoft* 並列パターン・ライブラリー) との互換性の削除
- 冗長な機能と問題のある機能の削除

詳細は software.intel.com/articles/tbb-revamp (英語) を参照

インテル® oneTBB の変更点

最新の C++ (C++11 以降) の機能に含まれるインテル® oneTBB の機能

廃止/削除されたインテル® oneTBB の機能	代替オプション
tbb::atomic	std::atomic
tbb::flow::tuple (ヘルパークラスを含む)	std::tuple
tbb::mutex	std::mutex
tbb::recursive_mutex	std::recursive_mutex
tbb::critical_section (tbb::improper_lock を含む)	std::mutex
tbb::hash (tbb::hasher を含む)	std::hash
tbb::tbb_thread / std::thread / std::this_thread	std::thread (std::chrono に最小限の変更を追加)
std::lock_guard / std::unique_lock (ヘルパークラスを含む)	std::lock_guard / std::unique_lock
std::condition_variable (std::cv_status、std::timeout、std::no_timeout を含む)	std::condition_variable
tbb::aligned_space	std::aligned_storage
tbb::tbb_exception / tbb::captured_exception / tbb::movable_exception	インテル® oneTBB の正確な例外伝播により不要になりました

詳細は software.intel.com/articles/tbb-revamp (英語) を参照

インテル® oneTBB の変更点

ほかの既存の機能と重複する機能または実用性に乏しい機能

廃止/削除されたインテル® oneTBB の機能	代替オプション
タスク API (tbb::task、tbb::empty_task、tbb::task_list および関連する関数)	直接の代替オプションはありません。大部分のユースケースは tbb::task_group と tbb::flow_graph でカバーできます。タスクの優先順位は、フローグラフ・ノードの優先順位とスタティック領域レベルの優先順位でカバーできます。
tbb::task_scheduler_init	tbb::task_arena / tbb::global_control (ブロック終了プレビュー機能をサポートするように拡張されます)
tbb::pipeline (tbb::filter、tbb::thread_bound_filter を含む)	tbb::parallel_pipeline / tbb::flow::async_node、resumable tasks
tbb::flow::sender / tbb::flow:receiver / tbb::flow::continue_receiver	フローグラフ・クラスの未定義のベースタイプ
フローグラフ・ノードのアロケーター・テンプレート・パラメーター	代替オプションの予定なし
tbb::flow_async_msg、tbb::flow::streaming_node、tbb::flow::opencl_node	代替オプションの予定なし。非同期/ヘテロジニアス・アクティビティーには tbb::flow::async_node または再開可能なタスクを使用
(プレビュー) tbb::serial::parallel_for	task_arena または global_control でスレッド数を 1 に制限
(プレビュー) runtime_loader (tbbproxy ライブラリー)	代替オプションの予定なし
tbb::structured_taks_group (ヘルパークラスを含む)	tbb::task_group
tbb::parallel_do	tbb::parallel_for_each
tbb::flow::source_node	tbb::flow::input_node

詳細は software.intel.com/articles/tbb-revamp (英語) を参照

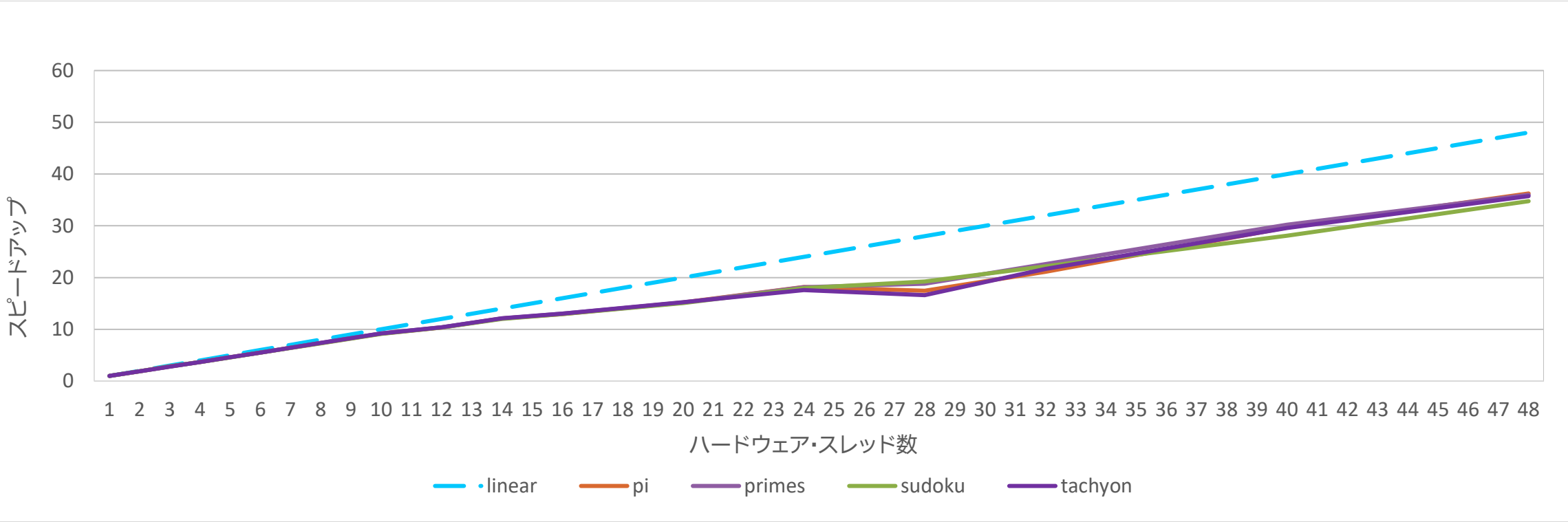
Microsoft* 並列パターン・ライブラリーとの互換性

Microsoft* 並列パターン・ライブラリー (PPL) との互換性のために提供されていたインテル® oneTBB の機能

廃止/削除されたインテル® oneTBB の機能	代替オプション
concurrency::critical_section	std::mutex
concurrency::reader_writer_lock (concurrency::improper_lock を含む)	std::shared_mutex (C++17 以前の環境で TBB により提供)
concurrency::parallel_invoke	tbb::parallel_invoke
concurrency::parallel_for (first, last, f)	tbb::parallel_for (first, last, f)
concurrency::parallel_for_each	tbb::parallel_for_each
concurrency::task_group (ヘルパークラスを含む)	tbb::task_group
concurrency::structured_task_group (ヘルパークラスを含む)	tbb::task_group

詳細は software.intel.com/articles/tbb-revamp (英語) を参照

インテル® Xeon® プロセッサーにおけるインテル® oneTBB のパフォーマンス・スケーラビリティー



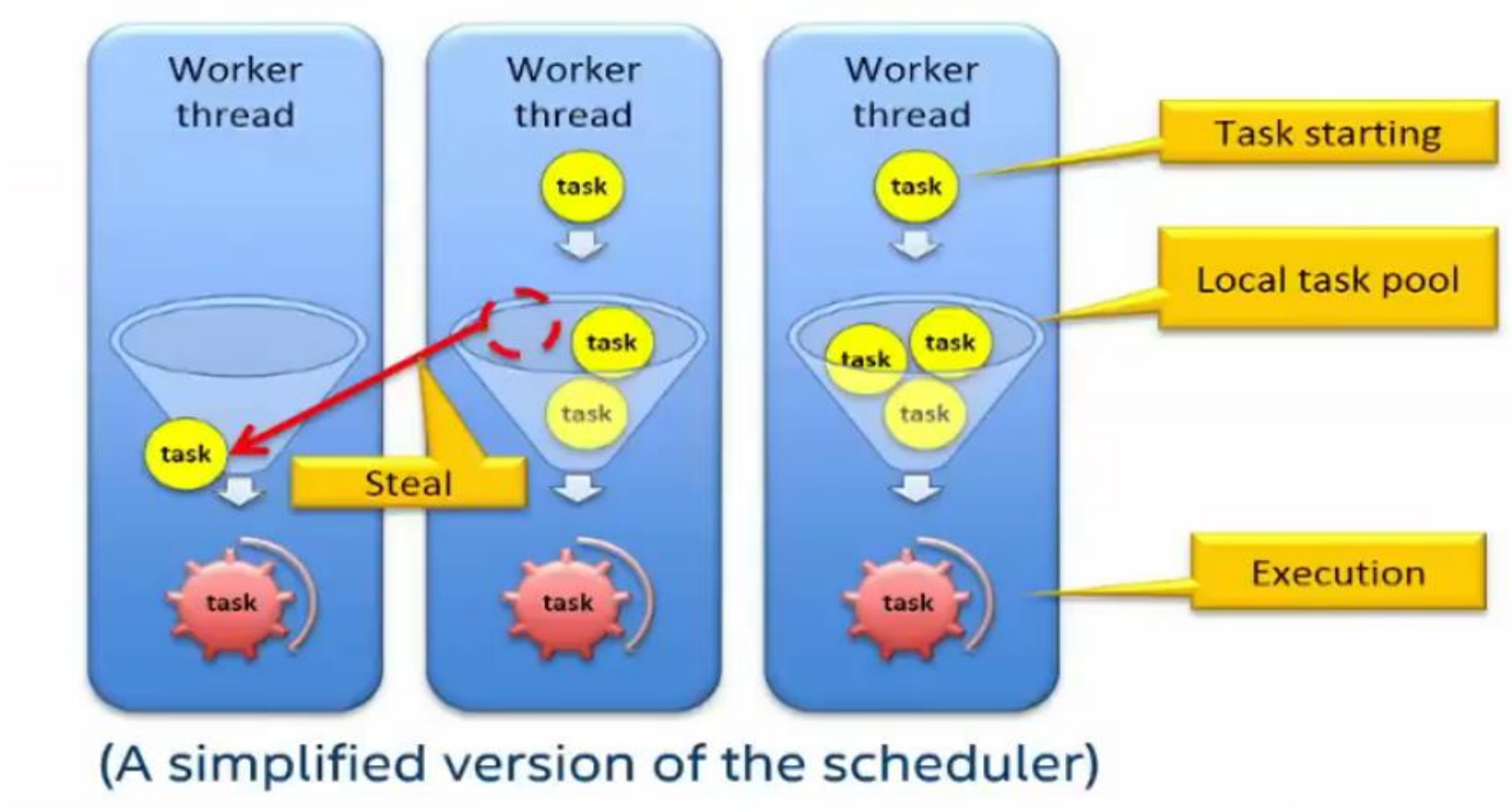
測定日: 性能の測定結果は **2020 年 11 月 13 日時点のテスト**に基づいています。また、現在公開中のすべてのセキュリティー・アップデートが適用されているとは限りません。

システム構成とワークロードのセットアップ: インテル® oneAPI DPC++/C++ コンパイラー、バージョン 2021.1。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)、バージョン 2021.1。ハードウェア: 2 x インテル® Xeon® Gold 6252 プロセッサー @ 2.10GHz (2 NUMA ノード)、192GB メインメモリー。オペレーティング・システム: Red Hat* Enterprise Linux* 8.0 (Ootpa)、カーネル 4.18.0-80.el8.x86_64。

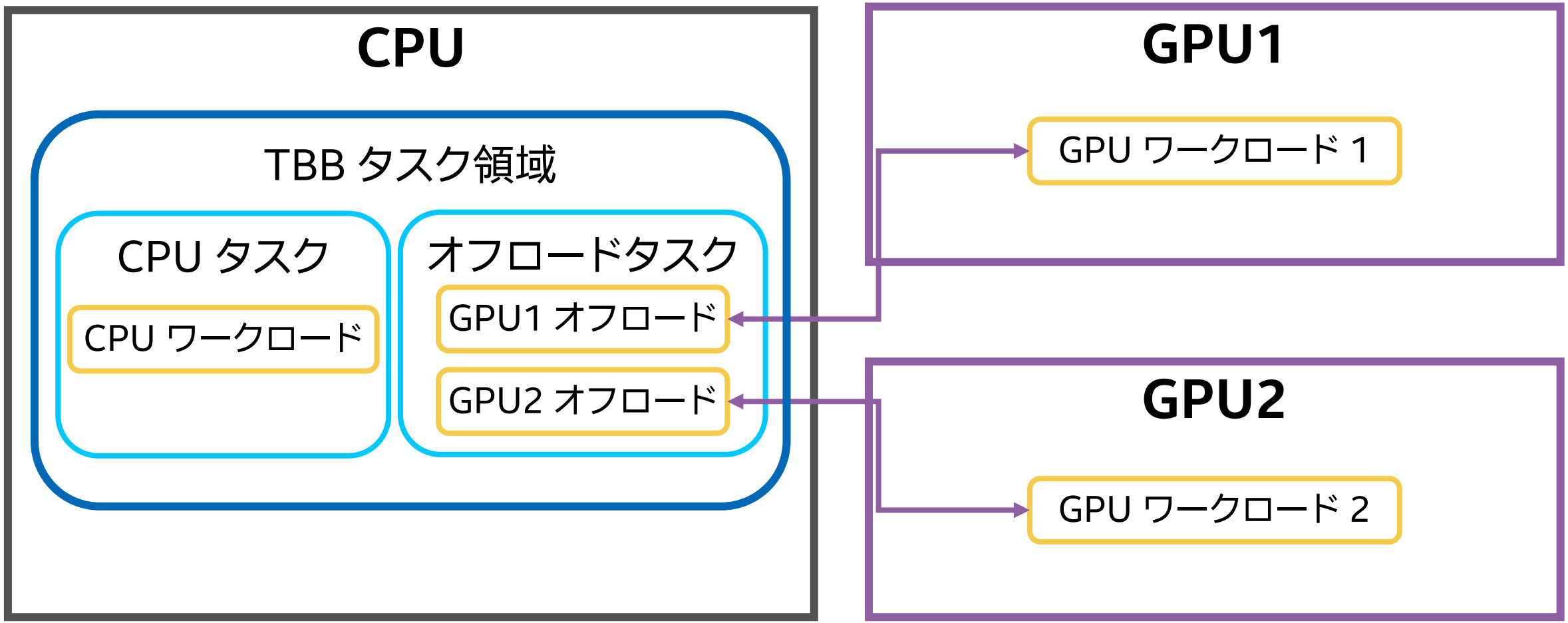
性能の測定結果はシステム構成の日付時点のテストに基づいています。また、現在公開中のすべてのセキュリティー・アップデートが適用されているとは限りません。詳細は、システム構成を参照してください。 絶対的なセキュリティーを提供できる製品またはコンポーネントはありません。

性能は、使用状況、構成、その他の要因によって異なります。詳細については、www.Intel.com/PerformanceIndex (英語) を参照してください。実際の費用と結果は異なる場合があります。

インテル® oneTBB のタスク実行



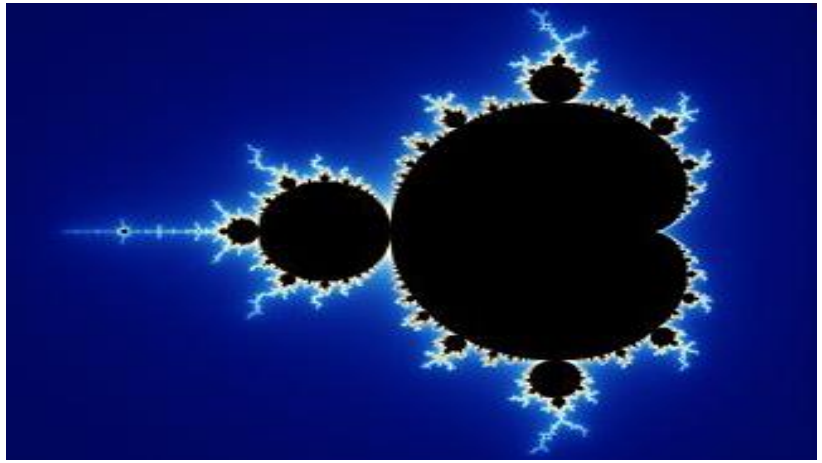
クロスアーキテクチャー環境での インテル® oneTBB の利用



汎用アルゴリズムで実証済みの並列パターンを再利用可能

```
int mandel(Complex c, int max_count) {

  int count = 0; Complex z = 0;

  for (int i = 0; i < max_count; i++) {

    if (abs(z) >= 2.0) break;

    z = z*z + c; count++;
for (int i = 0; i < max_row; i++) {

  for (int j = 0; j < max_col; j++ ) {

    p[i][j] = mandel( Complex(scale(i), scale(j)),
depth);

  }

}
```

シーケンシャル・バージョン

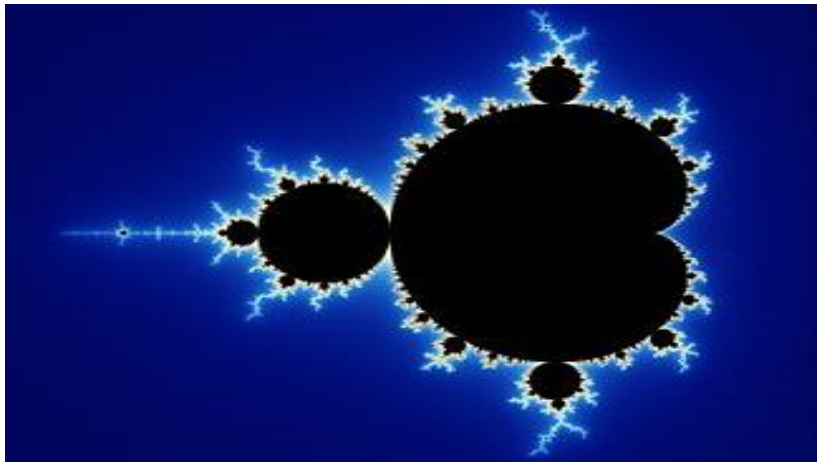


各ポイントで z = z*z + c は制限されているか?

マンデルブロのスピードアップ

```
int mandel(Complex c, int max_count) {
  int count = 0; Complex z = 0;
  for (int i = 0; i < max_count; i++) {
    if (abs(z) >= 2.0) break;
    ++count;
  }
  return count;
}
```

並列アルゴリズム

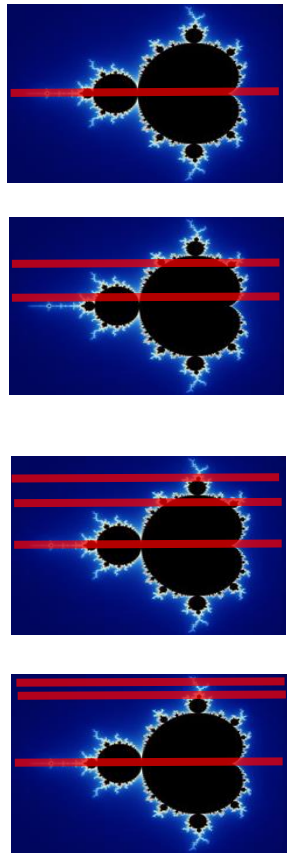


タスクは関数オブジェクト

C++ ラムダ関数を使用して関数オブジェクトをインラインで定義

```
  }
);
```

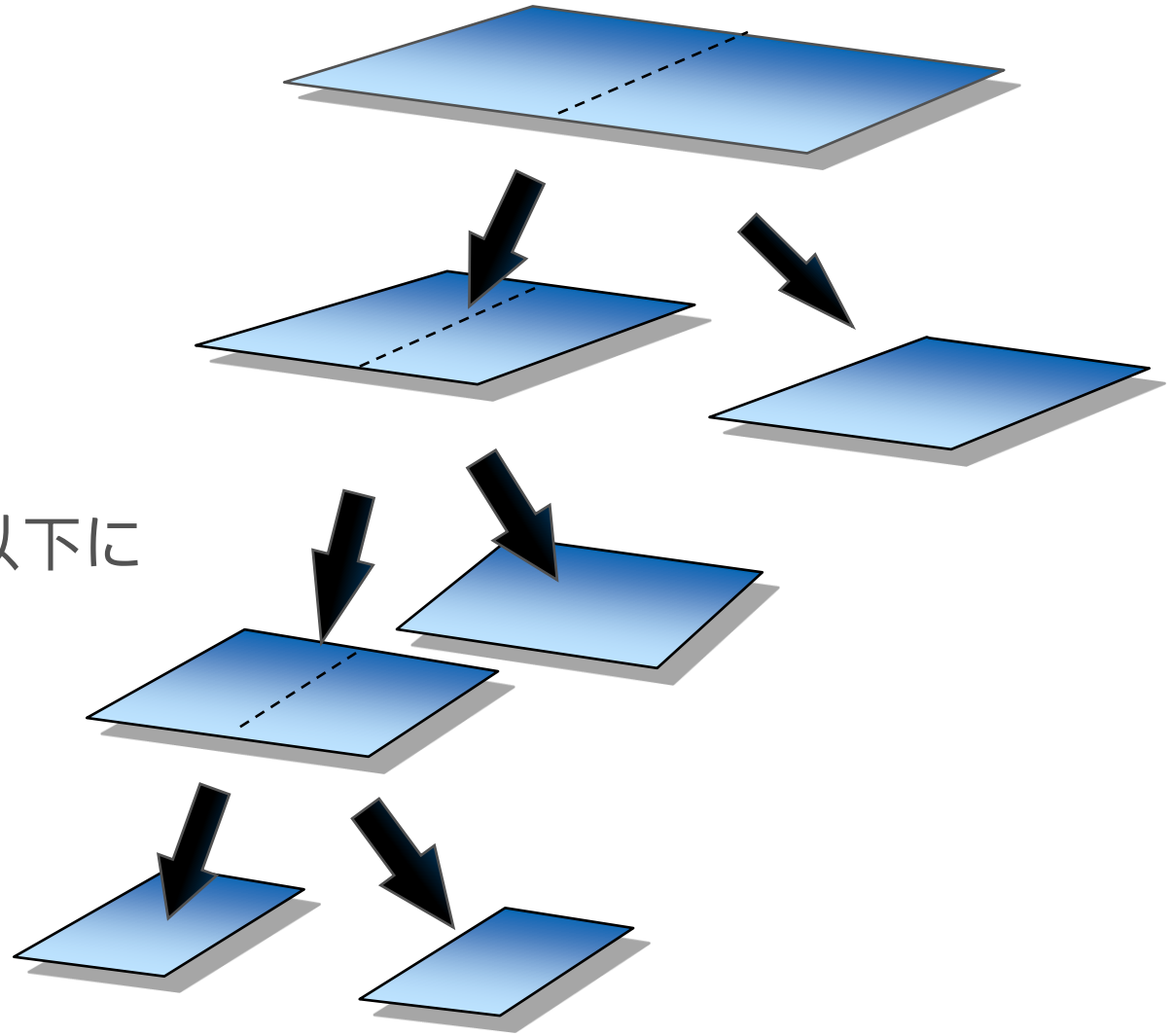
parallel_for 範囲をタスクを実行するサブ範囲に再帰的に分割



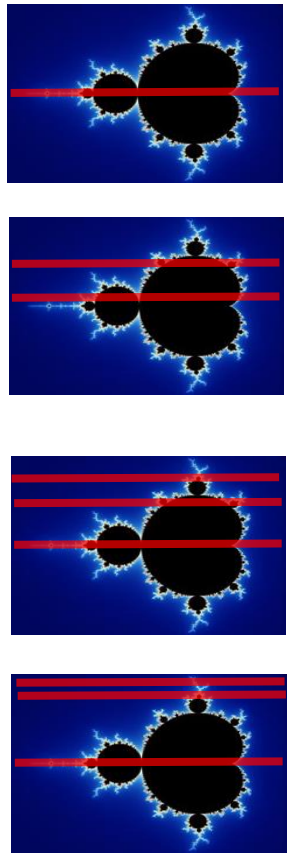
範囲を...

... grainsize 以下になるまで...

... 再帰的に分割



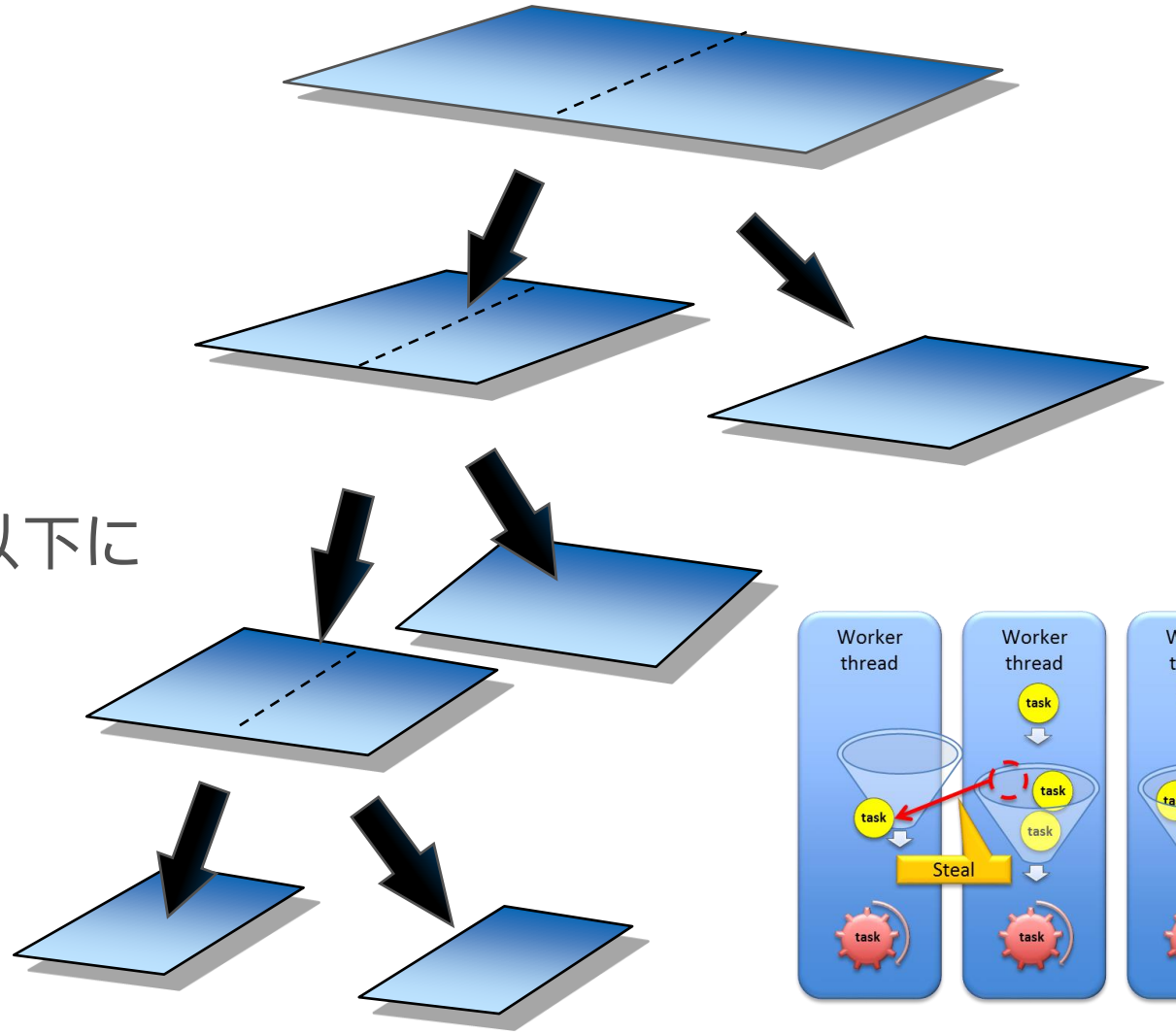
parallel_for 範囲をタスクを実行するサブ範囲に再帰的に分割



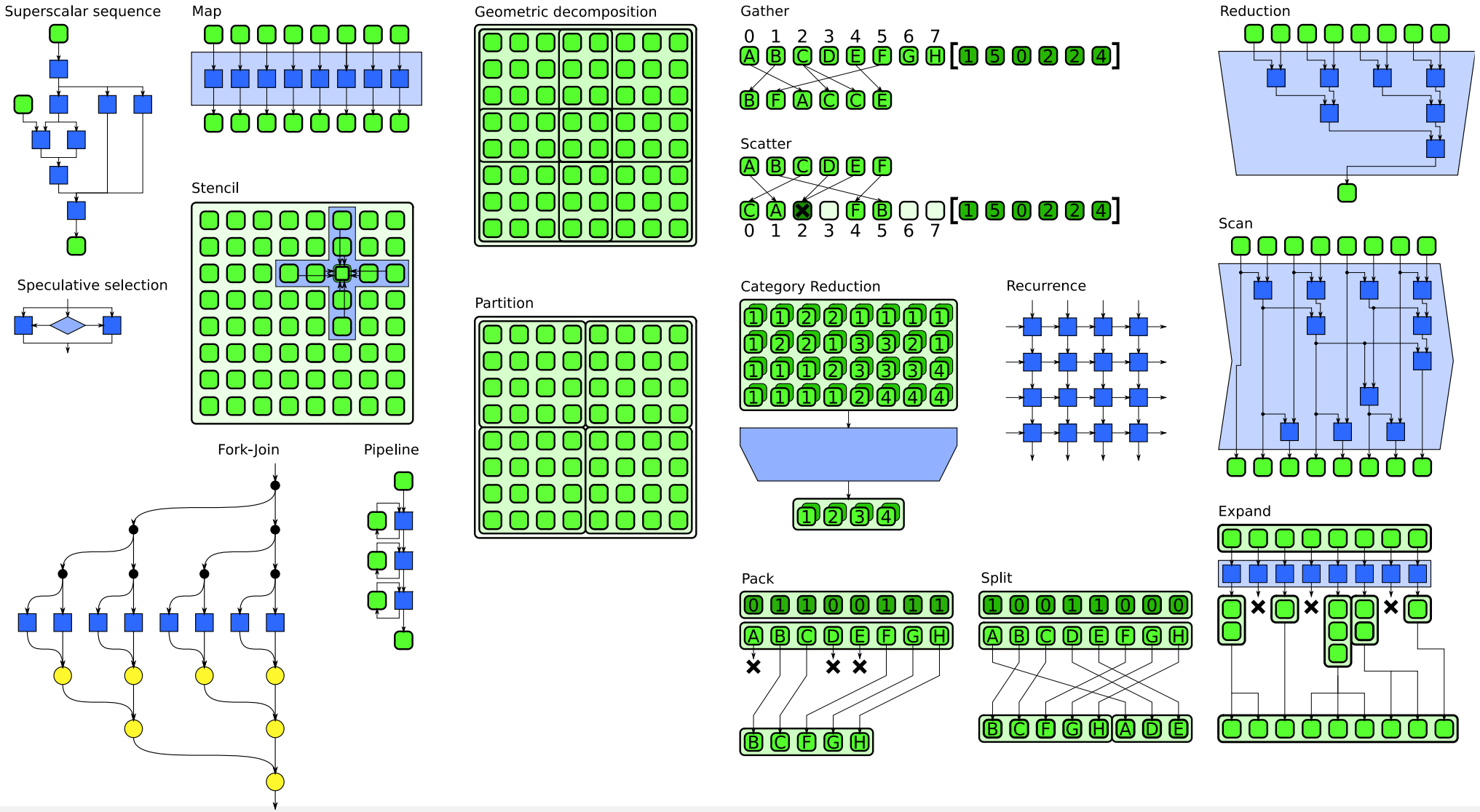
範囲を...

... grainsize 以下になるまで...

... 再帰的に分割



並列アルゴリズムの例


Superscalar sequence
Map
Geometric decomposition
Gather
Reduction
Stencil
Scatter
Speculative selection
Partition
Category Reduction
Recurrence
Scan
Fork-Join
Pipeline
Pack
Split
Expand

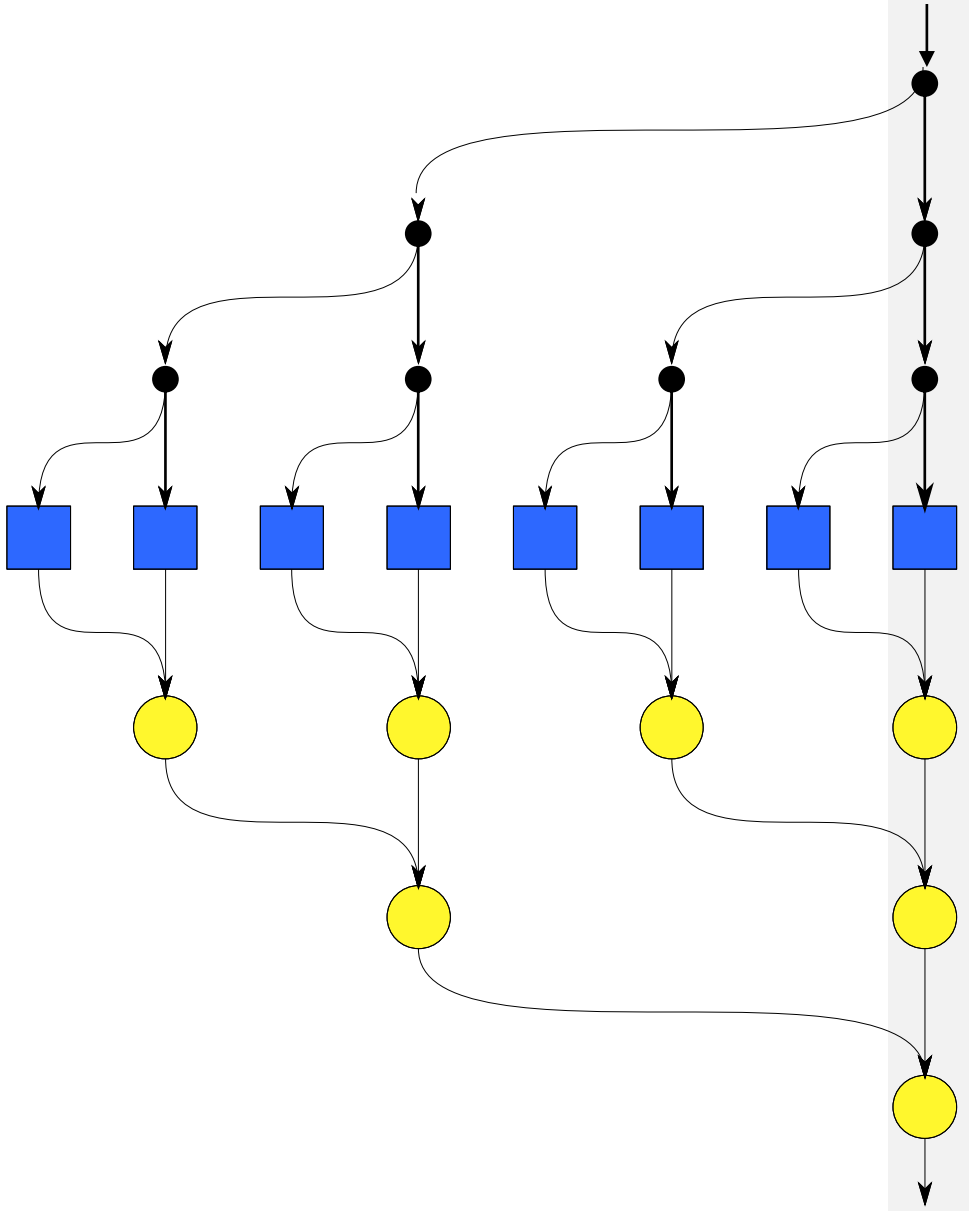
fork-join モデル

タスク数が少ない場合、または事前に分かっている場合

```
parallel_invoke( func1, func2, ...);
```

タスク数が多い場合、または事前に分かっていない場合

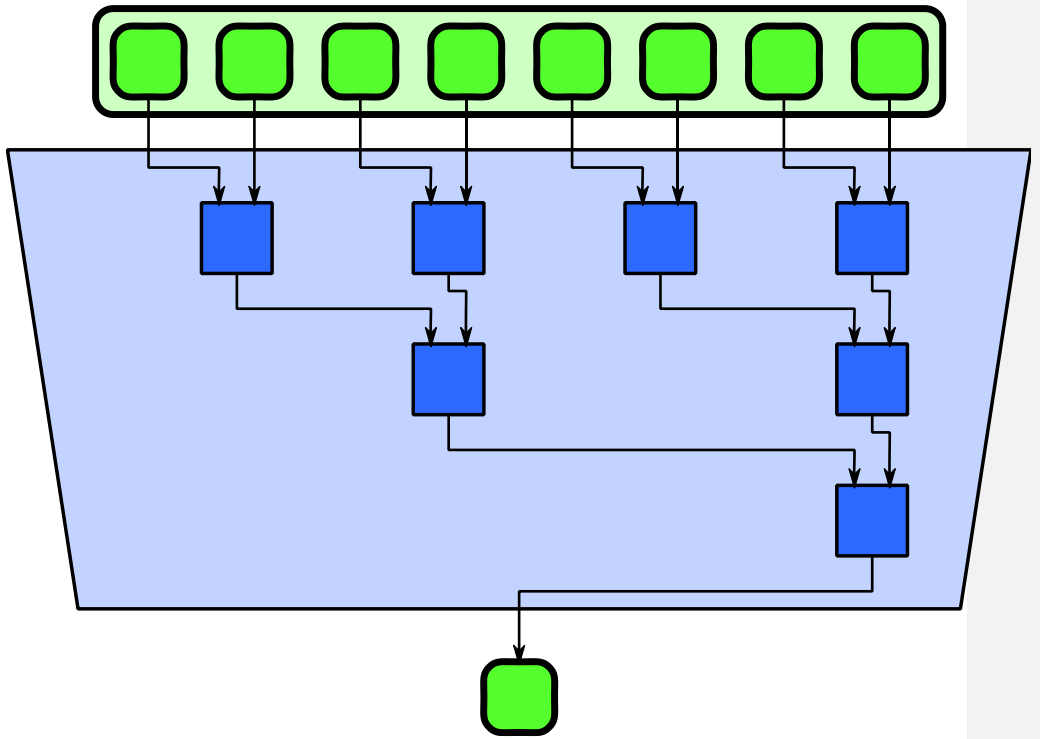
```
task_group g;
...
g.run( func1 );
...
g.run( func2 );
...
g.wait();
```



レデュースパターン

parallel_reduce 関数を使用

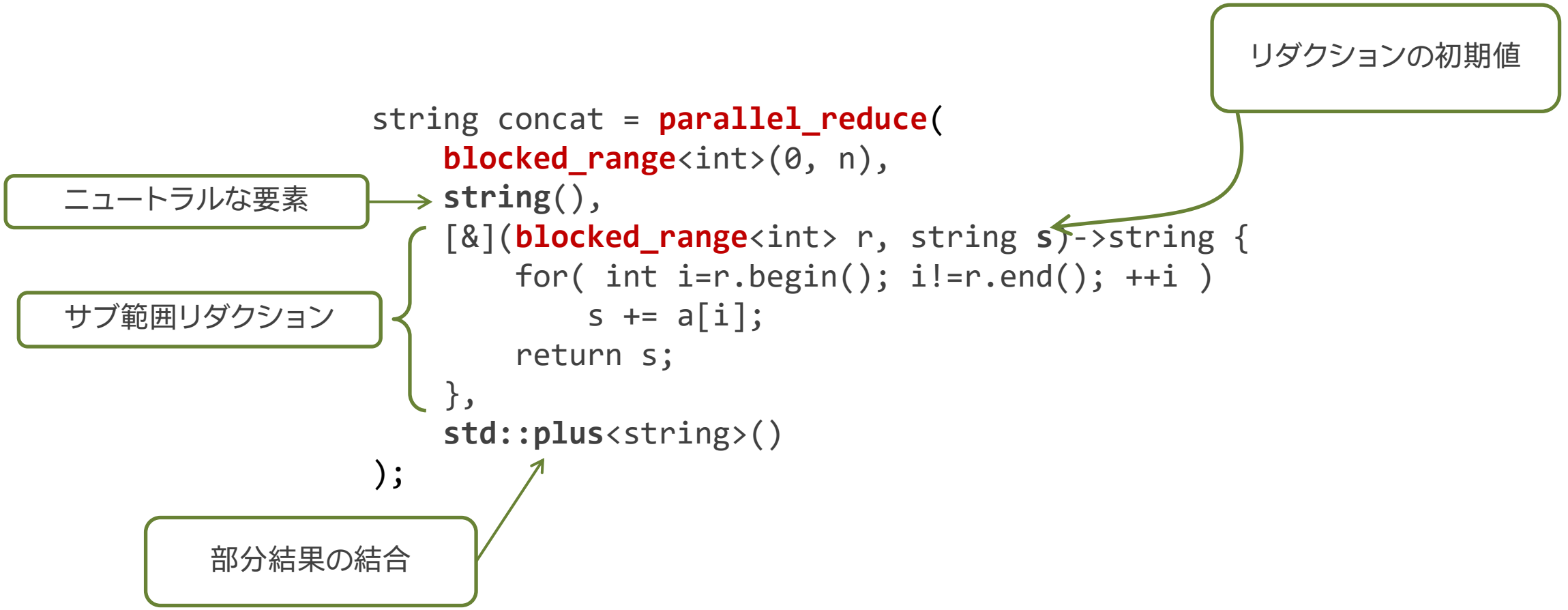
```
T sum = parallel_reduce(
    blocked_range<int>(0,n),
    0.f,
    [&](blocked_range<int> r, T s) -> T {
        for( int i=r.begin(); i!=r.end(); ++i )
           s += a[i];
        return s;
    },
    std::plus<T>()
);
```



enumerable_thread_specific クラスを使用

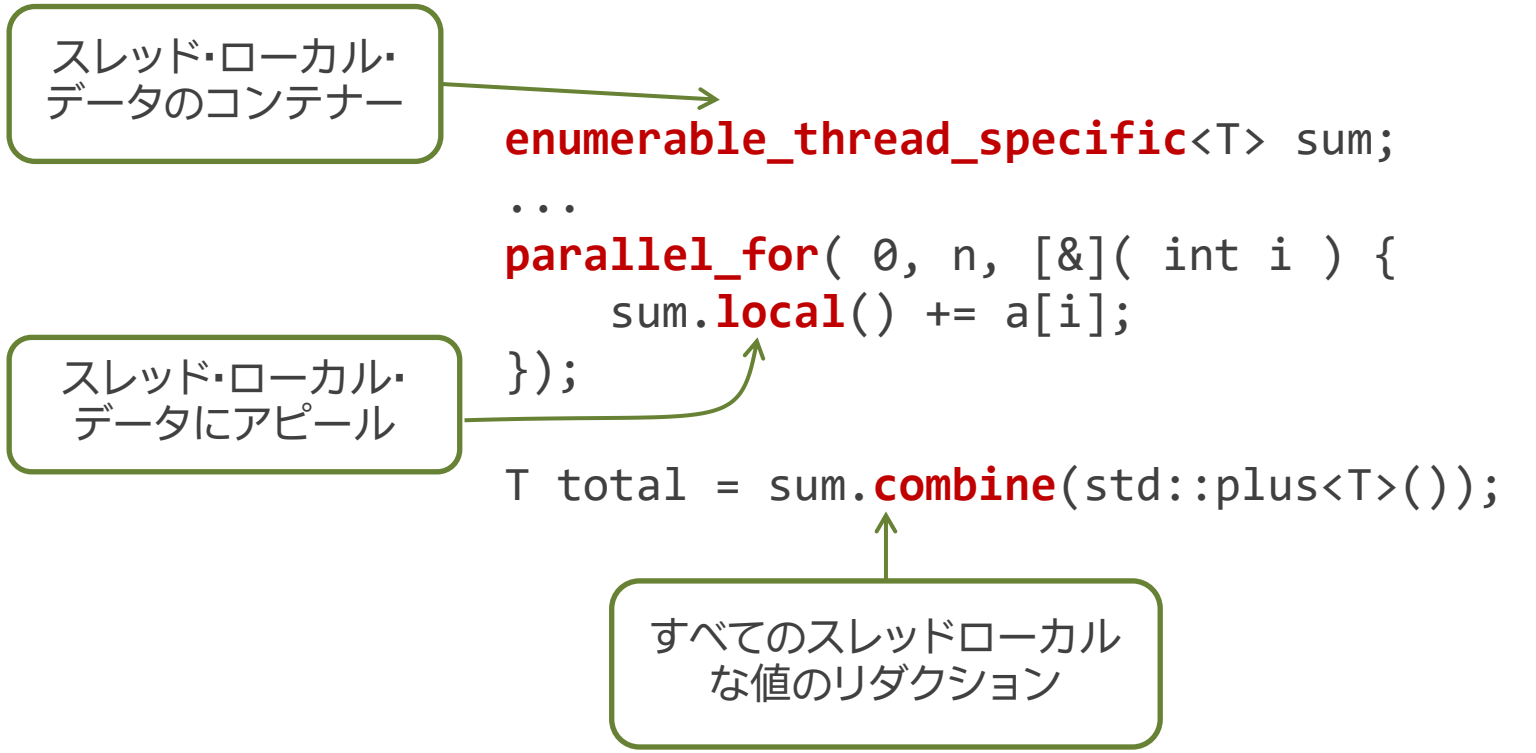
```
enumerable_thread_specific<T> sum;
parallel_for( 0, n, [&]( int i ) {
    sum.local() += a[i];
});
T total = sum.combine(std::plus<T>());
```

parallel_reduce 関数を使用したリダクション




```
string concat = parallel_reduce(
    blocked_range<int>(0, n),
    string(),
    [&](blocked_range<int> r, string s)->string {
        for( int i=r.begin(); i!=r.end(); ++i )
            s += a[i];
        return s;
    },
    std::plus<string>()
);
```

enumerable_thread_specific クラスを使用したリダクション




```
enumerable_thread_specific<T> sum;
...
parallel_for( 0, n, [&]( int i ) {
    sum.local() += a[i];
});

T total = sum.combine(std::plus<T>());
```

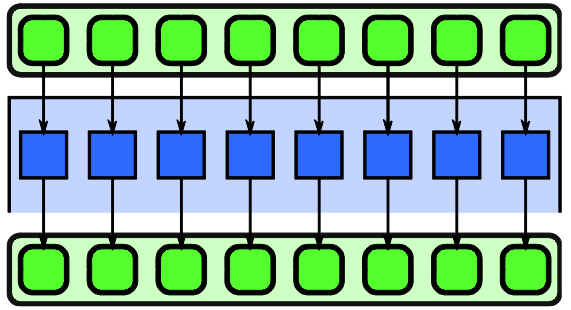
インテル® oneTBB のマップパターン

functor(i) をすべての i ∈ [lower, upper) に適用

```
parallel_for( lower, upper, func );
```

functor(i) を適用、指定されたステップで i を変更

```
parallel_for( lower, upper, stride, func );
```



functor(subrange) を range のすべての subrange に適用

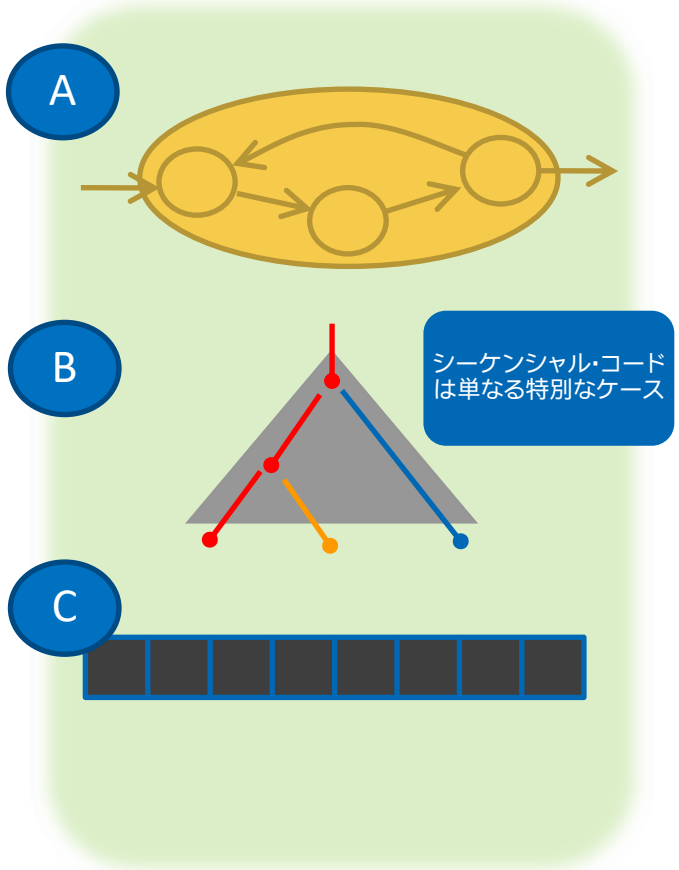
```
parallel_for( range, func );
```

parallel_for の例

```
void saxpy( float a, float x[], float (&y)[], size_t n ) {
    tbb::parallel_for( size_t(0), n, [&]( size_t i ) {
        y[i] += a * x[i];
    });
}
```

```
void saxpy( float a, float x[], float (&y)[], size_t n ) {
    size_t grain_size = 1000;
    tbb::parallel_for( tbb::blocked_range<size_t>(0, n, grain_size),
    [&]( tbb::blocked_range<size_t> r ) {
        for( size_t i = r.begin(); i != r.end(); ++i )
            y[i] += a * x[i];
    }
    );
}
```

CPU プログラミング・モデル階層



メッセージ駆動 (インテル® oneTBB フローグラフ)
(A) は単なる階層的なレイヤーであるため同じリソース/スケジューラーと (B) を使用

fork-join/タスク (インテル® oneTBB タスク)
予期しない CPU ロードに対する耐性および効率的な合成のサポート

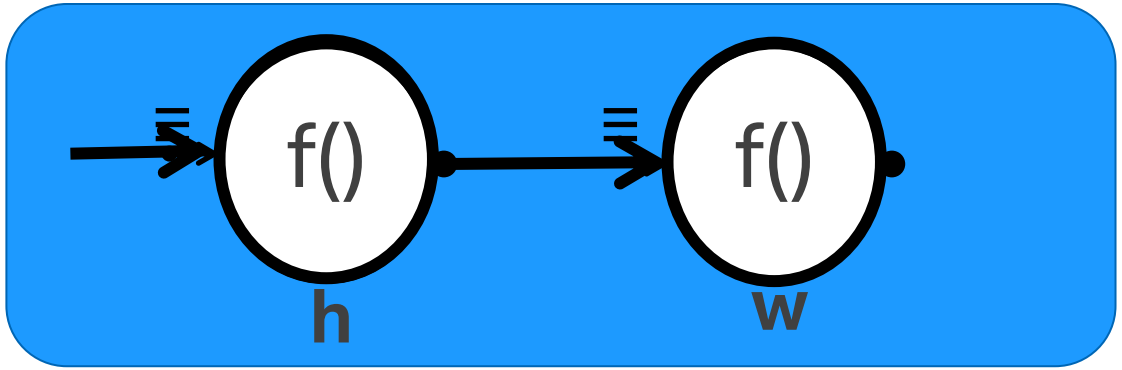
SIMD
コンパイラー・サポートが必要—C++ の Parallel STL に対する新しい標準化案ではこのレイヤーを同じソフトウェア・スタックに統合

インテル® oneTBB は**メッセージ駆動**レイヤーと **fork-join/タスク**レイヤーに必要なものを提供する C++ ライブラリー

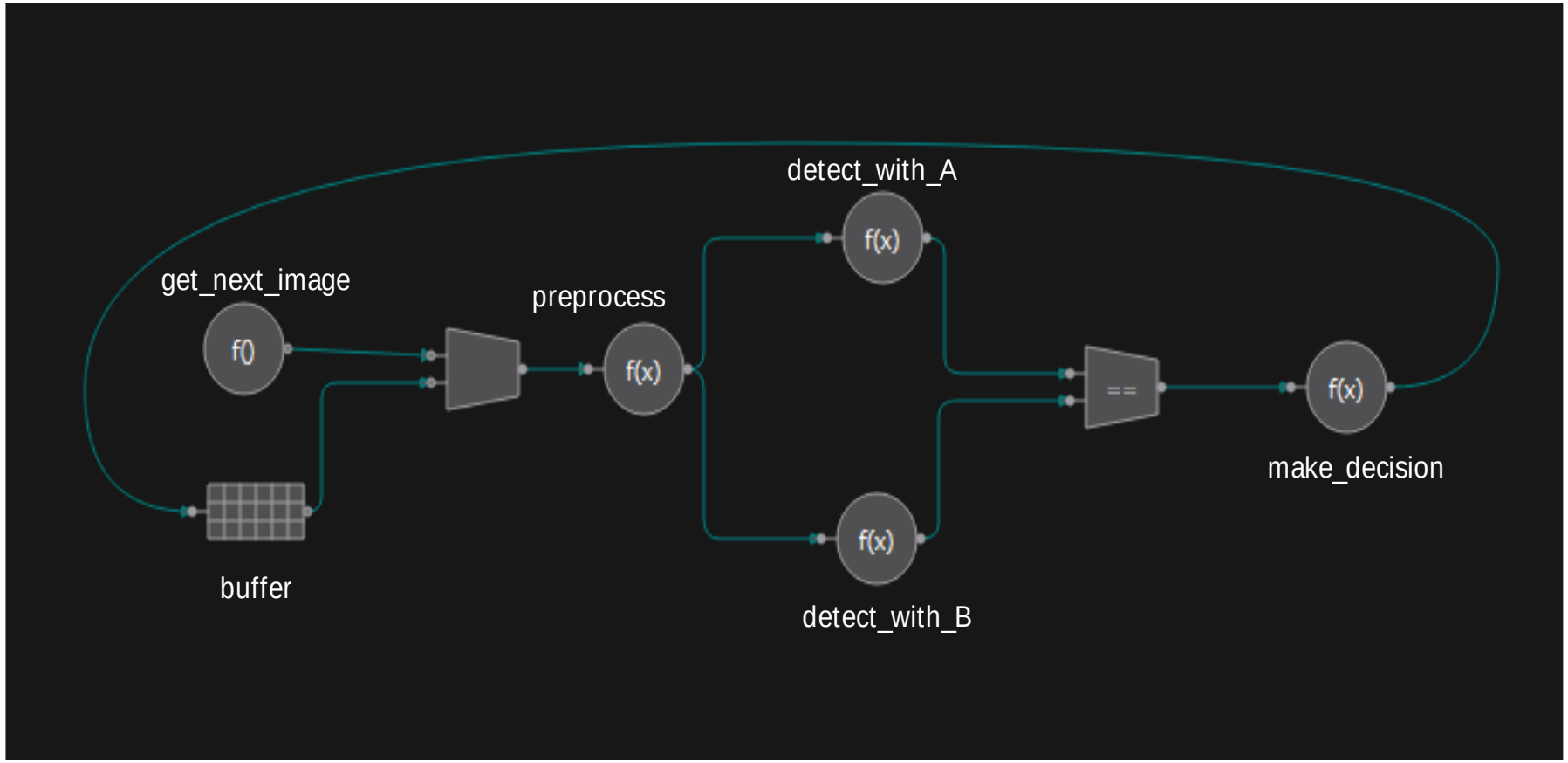
フローグラフの Hello World の例 (C++17 + プレビュー)

ノードとエッジを作成し、グラフを操作して、処理が完了するのを待機

```
tbb::flow::graph g;
tbb::flow::continue_node h( g,
    []( const continue_msg & ) { std::cout << “Hello “; } );
tbb::flow::continue_node w( tbb::flow::follows(h),
    []( const continue_msg & ) { std::cout << “World¥n“; } );

h.try_put(continue_msg());

g.wait_for_all();
```

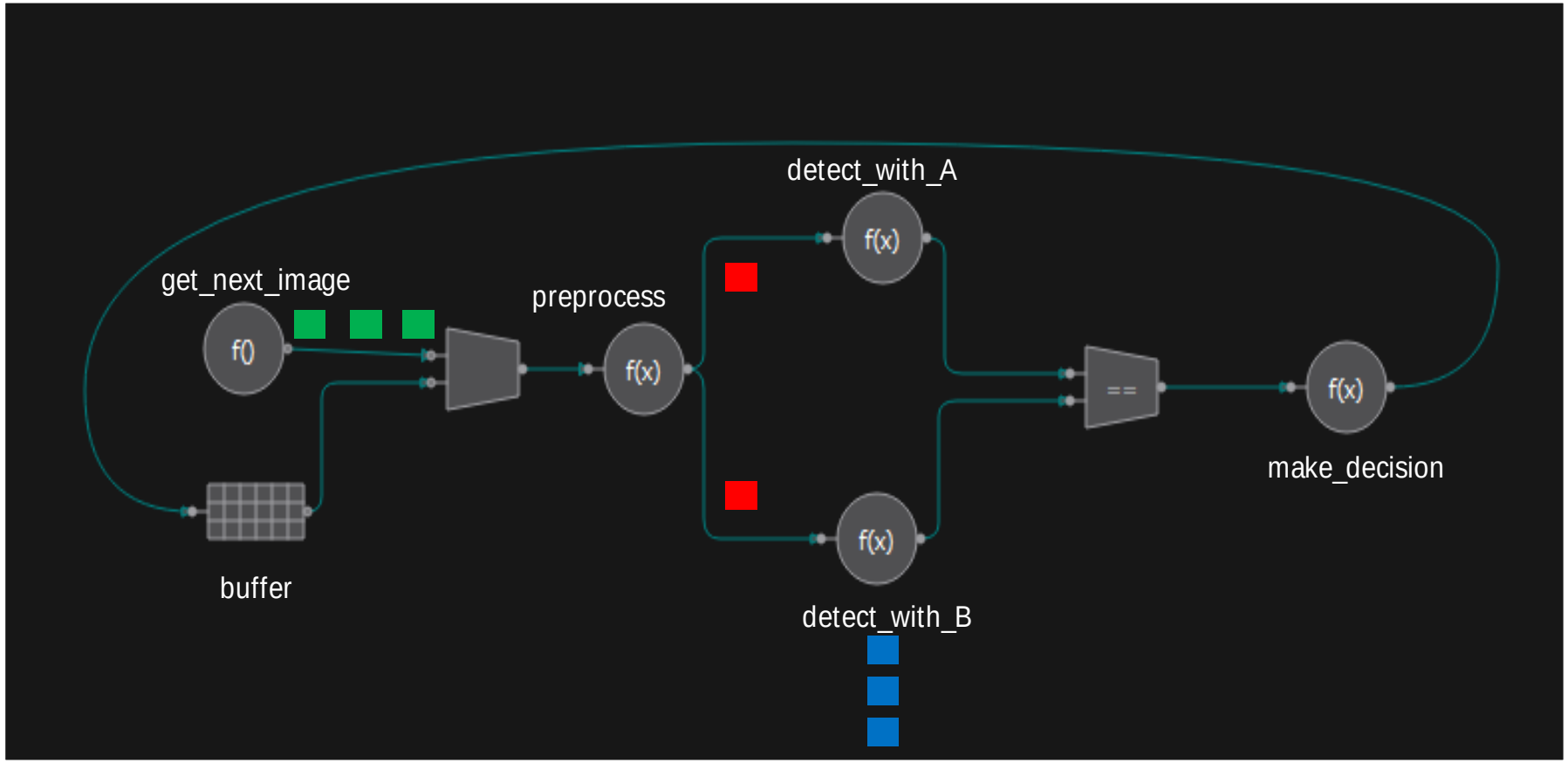


例: 特徴検出アルゴリズム



パイプライン、タスク並列処理、データ並列処理を表現可能

例: 特徴検出アルゴリズム



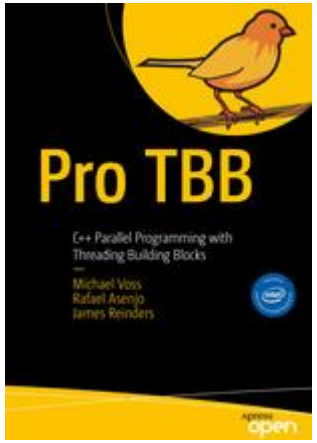
パイプライン、タスク並列処理、データ並列処理を表現可能

インテル® oneTBB、OpenMP*、インテル® oneMKL などによる入れ子の並列処理をサポート

関連情報

- インテル® oneAPI ベース・ツールキット
- スタンドアロン・コンポーネント (英語)
- オープンソース・バージョン (英語)

スレッディング・ビルディング・ブロックによる C++ 並列プログラミング (英語)



法務上の注意書きと最適化に関する注意事項

インテルのテクノロジーを使用するには、対応したハードウェア、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。詳細については、OEM または販売店にお問い合わせいただくか、http://www.intel.co.jp/ を参照してください。

実際の費用と結果は異なる場合があります。

インテルは、サードパーティーのデータについて管理や監査を行っていません。ほかの情報も参考にして、正確かどうかを評価してください。

最適化に関する注意事項: 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。
注意事項の改訂 #20110804https://software.intel.com/content/www/us/en/develop/articles/optimization-notice.html#opt-jp

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。構成の詳細は、補足資料を参照してください。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、https://www.intel.com/benchmarks (英語) を参照してください。

性能の測定結果はシステム構成の日付時点のテストに基づいています。また、現在公開中のすべてのセキュリティー・アップデートが適用されているとは限りません。詳細は、システム構成を参照してください。絶対的なセキュリティーを提供できる製品またはコンポーネントはありません。

本資料は、(明示されているか否かにかかわらず、また禁反言によるとよらずにかかわらず) いかなる知的財産権のライセンスも許諾するものではありません。

インテルは、明示されているか否かにかかわらず、いかなる保証もいたしません。ここにいう保証には、商品適格性、特定目的への適合性、および非侵害性の黙示の保証、ならびに履行の過程、取引の過程、または取引での使用から生じるあらゆる保証を含みますが、これらに限定されるわけではありません。


intel