新しい MPI-3 標準でパフォーマンスの課題に対処する

Mikhail Brinskiy インテル コーポレーション ソフトウェア開発エンジニア
Alexander Supalov インテル コーポレーション クラスター・ツール・アーキテクト
Michael Chuvelev インテル コーポレーション ソフトウェア・マネージャー
Evgeny Leksikov インテル コーポレーション

はじめに

インテル® MPI ライブラリー 5.0 およびインテル® MPI Benchmarks (IMB) 4.0 でサポートされる MPI-3 の非ブロッキング集合操作はパフォーマンスの利点をもたらします。ここでは、通信と計算のオーバーラップを測定する方法と、MPI アプリケーションで非ブロッキング集合操作を行う方法を紹介します。

メッセージ・パッシング・インターフェイス (MPI) 標準は、分散メモリーシステムでよく使用されるプログラミング・インターフェイスです。最新の MPI-3 標準には、非ブロッキング / 隣接集合操作、リモート・メモリー・アクセス (RMA) インターフェイスの拡張、ラージカウントのサポート、新しいツール・インターフェイスなどの主要な新機能があります。ラージカウントのサポートは大量のデータをシームレスに処理し、高速な一方向操作はリモート・メモリー・アクセス (RMA) ベースのアプリケーションを高速化し、非ブロッキング集合操作はアプリケーションの計算処理と通信処理をオーバーラップさせることでパフォーマンスを向上します。

ここでは、非ブロッキング集合操作 (NBC) と新しい RMA インターフェイスの 2 つの MPI-3 機能に注目します。NBC による通信と計算のオーバーラップの効果を検証し、新しい RMA 機能の利点を説明します。

パフォーマンスの測定には、MPI-3 標準をサポートするベータ版インテル® MPI ライブラリー 5.0[3] とベータ版インテル® MPI Benchmarks 4.0 (IMB) を使用しました。ベータ版 IMB 4.0 には非ブロッキング集合操作の効果を測定する **IMB-NBC** と、MPI-3 における RMA インターフェイスの重要なアップデートである **IMB-RMA** の 2 つの新しいバイナリーが追加されています。IMB と各ベンチマークの説明は、**http://software.intel.com/en-us/articles/intel-mpi-benchmarks (英語)** を参照してください。

非ブロッキング集合操作 (NBC)

非ブロッキングのポイントツーポイント操作のように、NBC セマンティクスにより通信と計算をオーバーラップさせることができます。また、NBC は、データ依存によって多くの集合アルゴリズムで生じる疑似同期の影響を緩和します。非ブロッキング集合操作の可能性については、**http://htor.inf.ethz.ch/publications/img/hoefler-nbc-standard.pdf (英語)** に詳しい分析があります。

一般に非ブロッキング操作 (集合およびポイントツーポイント) では、ユーザーの介入なしにバックグラウンドで処理が行われます。[1] しかし、MPI 標準では明示的に処理規則が定義されていないため、「高品質な」実装により非同期処理が行われます。そのため、以前の MPI ライブラリーの多くは非同期処理を提供せず、単に 1 つの MPI 呼び出しと別の MPI 呼び出しの間ですべての通信を実行していました。NBC の登場により、定期的に **MPI_Test()** ルーチンを呼び出して明示的にメッセージを処理するか、あるいは **MPI_Wait()** を 1 回呼び出して処理を完了することで、集合操作と通信をオーバーラップさせるという選択肢が増えました。

IMB は **MPI_Test()** または **MPI_Wait()** 呼び出しを通じて、よく使われる非ブロッキング通信、特に操作の開始と完了の間にアクティビティーが終了するケースにおいて、通信と集合操作のオーバーラップの効果を測定します。この手法では、ネットワークやオペレーティング・システム (OS) のノイズによって生じる不正確さを最小限に抑えられるため、通信時間と計算時間はほぼ同じであると仮定しています。IMB は、完全にベクトル化された 100x100 行列のベクトル積を計算することで、任意の時間 CPU をビジー状態にします。以下に、ベンチマークの処理の流れを示します。

1. 通信呼び出しにかかる時間 (**MPI_Ibcast()** から **MPI_Wait()** まで) を測定します。
2. 通信を開始します (**MPI_Ibcast()** を呼び出します)。
3. ステップ 1 で測定した時間分の計算を開始します。これにより、通信と計算にかかる時間がほぼ同じになるようにします。
4. 通信が完了するのを待ちます (**MPI_Wait()** 呼び出しを待機します)。

IMB-NBC ベンチマークは、上記の処理を実行して 4 つの測定時間を出力します。

> **time_pure** – CPU アクティビティーと同時に実行されなかった非ブロッキング操作の時間

> **time_CPU** – テスト CPU アクティビティーの時間 (time_pure に近い値になります)

> **time_ovrlp** – CPU アクティビティーと同時に実行された非ブロッキング操作の時間

> **overlap** – 次の式で予測されたオーバーラップの割合

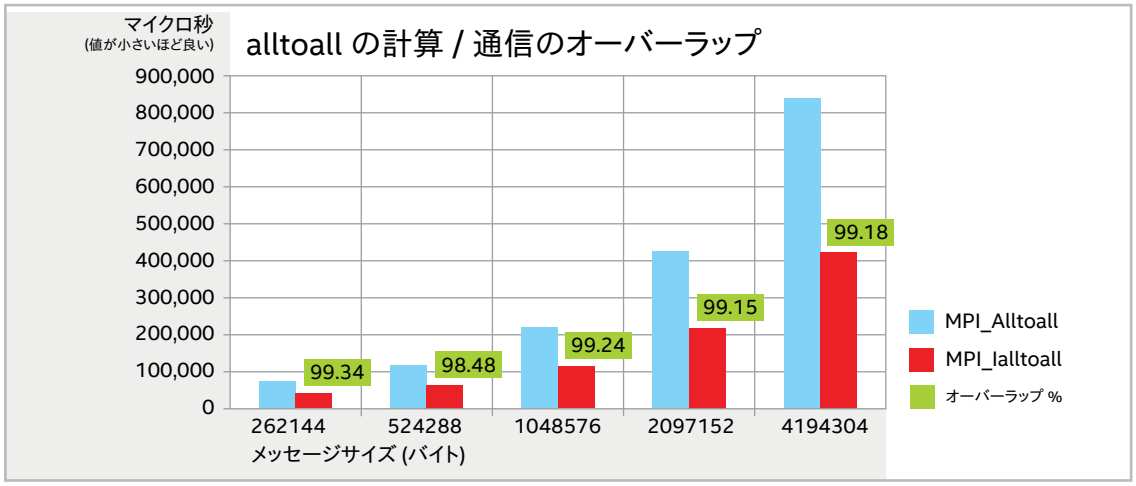
```
overlap = 100 * max(0, min(1, (time_pure + time_CPU - time_ovrlp) /
max(time_pure, time_CPU)))
```

インテル® MPI ライブラリーは、マルチスレッド・ライブラリーでのみ非同期メッセージ処理をサポートします。そのため、ここでは非ブロッキング集合操作のパフォーマンスを最大限に引き出すようにマルチスレッド・ライブラリーを使用しています。インテル® MPI ライブラリーで非同期処理を有効にするには、**MPICH_ASYNC_PROGRESS** 環境変数を 1 に設定してください。

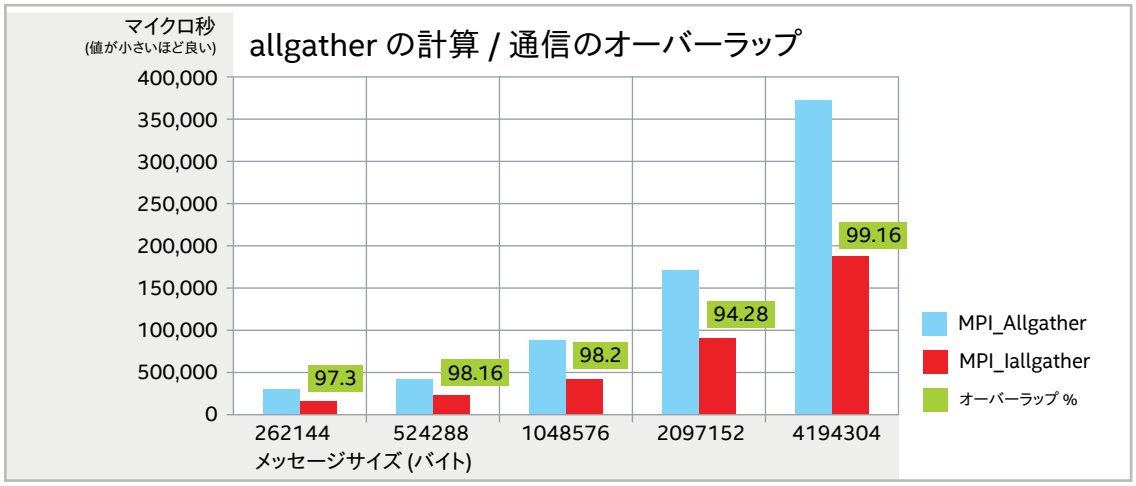
ここでは、非ブロッキング集合操作のオーバーラップを測定する **IMB-NBC** ベンチマークの結果と、**IMB-MPI1** で得られる通常の集合操作の測定結果を比較します。ブロッキング集合操作を実行してから計算を実行するのと、非ブロッキング集合操作により同量の計算とオーバーラップさせるのでは、どちらのほうが効率的かを検証します。グラフには非ブロッキング操作と CPU アクティビティーを同時に実行した場合 (IMB-NBC 出力の **time_ovrlp** 値) と、通信と計算を別々に行ってその時間を合計した場合の 2 つの測定時間を示します。2 つ目の値は、ブロッキング集合操作の時間 (**IMB-MPI1** ベンチマークで測定) とテスト CPU アクティビティーの時間 (**IMB-NBC** 出力の **time_CPU** 値) の合計です。

グラフに示す値は、4 つの IVT ノードで 48 プロセス (ノードあたり 12 プロセス) を使用した結果です。各ノードには、2 つのインテル® Xeon® プロセッサー E5-2697 と 1 つの Mellanox* Connectx*-3 InfiniBand* アダプターが搭載されていす。

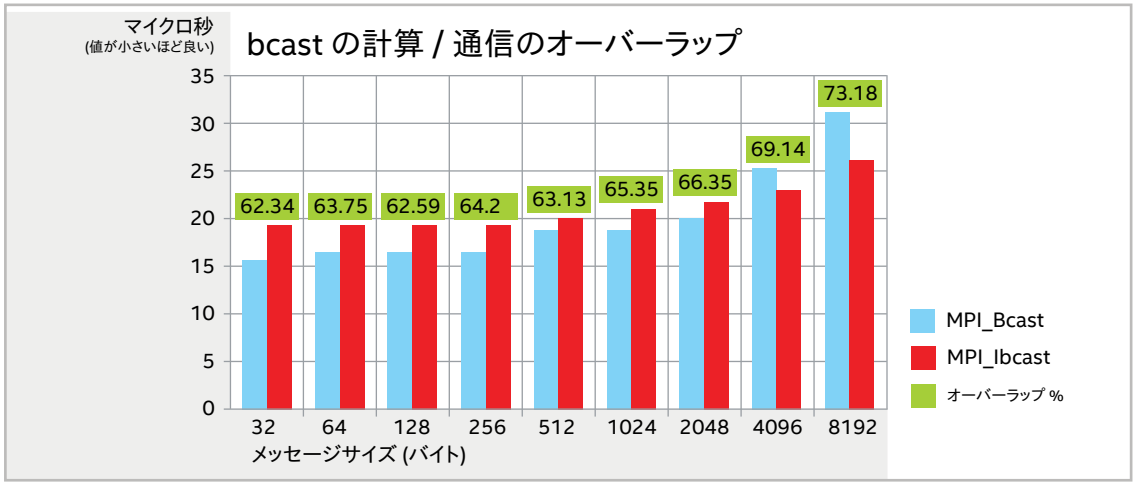
ialltoall および iallgather ベンチマークから得られるオーバーラップは理想に近いものです。図 1 と図 2 は、代表的な結果としてメッセージサイズの大きなもののみを示しています。非同期処理を行う非ブロッキング操作ではオーバーヘッドが大幅に増えるにもかかわらず (これは特に小さなメッセージサイズで顕著です)、通常のブロッキング集合操作と計算の組み合わせよりも効率的であることが分かります。しかし、常にそうなるとは限りません。図 3 は、小さなメッセージサイズのブロードキャスト・ベンチマークの結果です。オーバーラップの割合は比較的良いものの (約 60 ～ 70%)、通常のブロッキング・ブロードキャスト操作のほうが高速で、メッセージサイズが 2KB 以下の場合はパフォーマンス効率も良くなります。



1 MPI_Ialltoall と計算をオーバーラップさせた場合と MPI_Alltoall を実行してから計算を行った場合



2 MPI_Iallgather と計算をオーバーラップさせた場合と MPI_Allgather を実行してから計算を行った場合



3 MPI_Ibcast と計算をオーバーラップさせた場合と MPI_Bcast を実行してから計算を行った場合

これらの結果から次のことが分かります。

- 非ブロッキング集合操作は計算とオーバーラップすることで、特にメッセージサイズが中～大の場合、ブロッキング操作と比べて大幅なパフォーマンス向上をもたらします。
- 場合によっては、非ブロッキング操作と非同期メッセージ処理によって生じるオーバーヘッドがオーバーラップの利点を打ち消してしまうことがあります。これは、通常レイテンシーが重要になる小さなメッセージサイズで起こる傾向にあります。

新しい RMA インターフェイス

MPI-3 標準はさまざまなメモリーモデル、新しい通信と同期呼び出し、RMA ウィンドウの新しい作成方法を含むいくつかの主要機能の追加により、RMA インターフェイスを大幅に拡張します。異なるメモリーモデルのサポートは最も重要な新機能の 1 つで、キャッシュ・コヒーレンシーがサポートされる (ユニファイド・モデル) 場合とされない場合 (MPI-2 標準に特有の個別モデル) を区別します。この区別により、ハイパフォーマンスな MPI 実装が可能になるとともに、プログラミングが容易になります。同期セマンティクスの主な更新は、パッシブターゲット通信モードを現在の HPC のニーズに合わせるためのものです。MPI-3 標準で改善が試みられている以前の RMA インターフェイスの主な問題は、http://upc.lbl.gov/publications/bonachea-duell-mpi.pdf **(英語)** で説明されています。

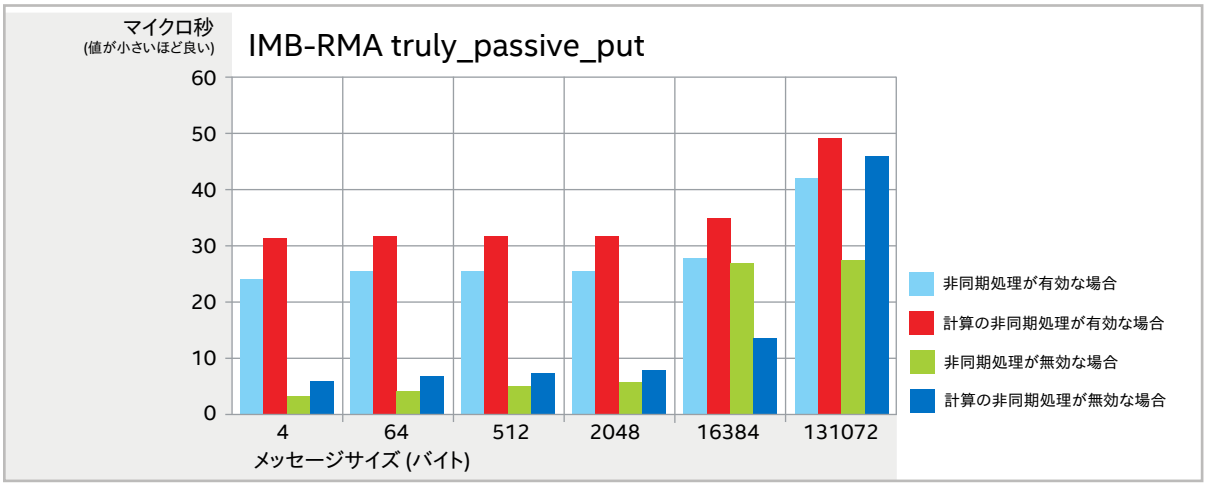
ベータ版 IMB 4.0 以前のバージョンでは、アクティブ通信モードのみを用いる RMA ベンチマークのセットが提供されていました。これらのベンチマークは、`IMB-EXT` モジュールにあります。ベータ版 IMB 4.0 は、MPI-3 の新しい RMA 機能とパッシブターゲット通信モードに注力しています。新しい IMB-RMA モジュールには 19 のベンチマークが含まれていおり、新しいアトミック関数、`MPI_Put()` および `MPI_Get()` 操作を使用するハロゲン交換、異なるターゲットへの同時 RMA 呼び出しを測定します。これらのベンチマークはすべて、パッシブターゲット通信モードを利用します。

新しい RMA ベンチマークの 1 つは、MPI 実装が通信の自然なパッシブモードをサポートするかどうかを示します。自然なパッシブモードでは、発信元のプロセスがアクセスエポックを完了できるように、ターゲットプロセスで MPI を呼び出す必要はありません。これは、前述の非同期メッセージ処理によって、あるいはハードウェアの通信オフロード機能によってサポートされる可能性があります。ベータ版インテル® MPI ライブラリー 5.0 の RMA 操作は、通常のポイントツーポイント・メッセージを用いて実装されているため、確実に完了するようにターゲットと発信元で処理される必要があります。前述したように、非同期メッセージ処理はインテル® MPI ライブラリーのマルチスレッド･バージョンでサポートされています。シングルスレッド･バージョンではパッシブ･ターゲット･モードはサポートされていません。そのため、ターゲットは MPI を呼び出して処理しなければなりません。ここでは、非同期処理を行った場合とシングルスレッド・バージョンを利用した場合の結果を比較し、インテル® MPI ライブラリーのパッシブ・モード・サポートの利点を評価します。

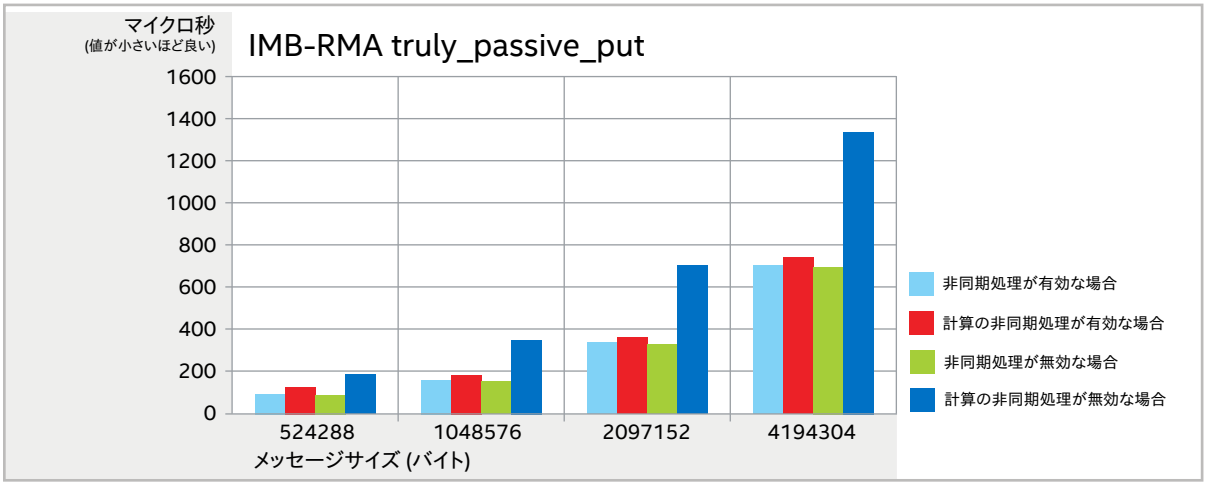
上記のベンチマークは次の 2 つの時間を測定します。

1. ターゲットプロセスが (確実に処理されるように) `MPI_Barrier()` 呼び出しを実行している間、発信元プロセスが `MPI_Put()` 操作の完了に費やした時間
2. ターゲットプロセスが MPI スタック外で計算を実行し `MPI_Barrier()` を呼び出す間、発信元プロセスが `MPI_Put()` 操作の完了に費やした時間。計算時間は、発信元プロセスがステップ 1 を完了するのに必要な時間とほぼ同じになるようにします。そのためには、`MPI_Put()` の完了に必要な時間の値を発信元プロセスからターゲットプロセスへ送ります。

ターゲットプロセスと発信元プロセスは、各 `MPI_Put()` 操作の前に同期されます。そのため、直接通信オフロード機能がない場合、あるいはメッセージ処理を実行するスレッドが異なる場合、ステップ 2 の時間はステップ 1 の時間よりも大きく (約 2 倍に) なるでしょう。ただし、MPI 実装が RMA でパッシブモードを提供している場合、これらの時間はほぼ同じになります。



4 非同期処理のパッシブ・ベンチマーク結果への影響 (小～中程度のメッセージサイズ)



5 非同期処理のパッシブ・ベンチマーク結果への影響 (大きなメッセージサイズ)

図 4 と**図 5** は、非同期メッセージ処理によりオーバーヘッドが著しく増加することを示しており、これはメッセージサイズが 128KB 以下の場合に顕著です。しかし、ラージ・メッセージサイズではオーバーヘッドの影響があまり見受けられず、ターゲットの計算パフォーマンスによって発信元の `MPI_Put()` 操作の完了が遅れることはありません。

ベータ版インテル® MPI ライブラリー 5.0 でパッシブモードの通信モデルの動作を分析した結果、このライブラリーでは非同期処理でパッシブモードがサポートされているものの、改善の余地があることが分かりました。これには、RDMA 処理のネイティブサポートやさまざまなパフォーマンスの最適化 (例えば、オーバーヘッドが大きく、マルチスレッド・バージョンでのみ利用可能な非同期メッセージ処理の改善) などがあります。MPI-3 の新しい RMA インターフェイスは柔軟な一方向操作のインターフェイスを提供します。その利点については、**http://htor.inf.ethz.ch/publications/img/mpi_mpi_hybrid_programming.pdf (英語)** を参照してください。これらの利点により、MPI 実装のパフォーマンス効率が向上し、さまざまな PGAS 言語に対する競争力も上がるでしょう。

まとめ

MPI-3 標準は、MPI ユーザーがパフォーマンス効率の良いプログラムを記述できるように柔軟なインターフェイスを提供します。ここでは、ベータ版インテル® MPI ライブラリー 5.0 を例に 2 つの重要な機能、非ブロッキング集合操作と新しい RMA インターフェイスについて述べました。これらの機能によって MPI アプリケーションのパフォーマンスを向上できるでしょう。既存の MPI プログラムでは、新しい MPI-3 機能を利用するのにアプリケーションの変更が必要になるかもしれませんが、それだけの価値があることはここで示したように明らかです。

参考文献

1. Message Progression in Parallel Computing—To Thread or Not to Thread, Torsten Hoefler and Andrew Lumsdaine.
   http://htor.inf.ethz.ch/publications/index.php?pub=75 (英語)
2. A Message-Passing Interface Standard 3.0,
   http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf (英語)
3. ベータ版インテル® MPI ライブラリー 5.0 Readme:
   http://software.intel.com/en-us/articles/intel-mpi-library-50-beta-readme (英語)

インテル® MPI ライブラリーを試そう >

Parallel Universe 18 号の全文はこちらでご覧いただけます。