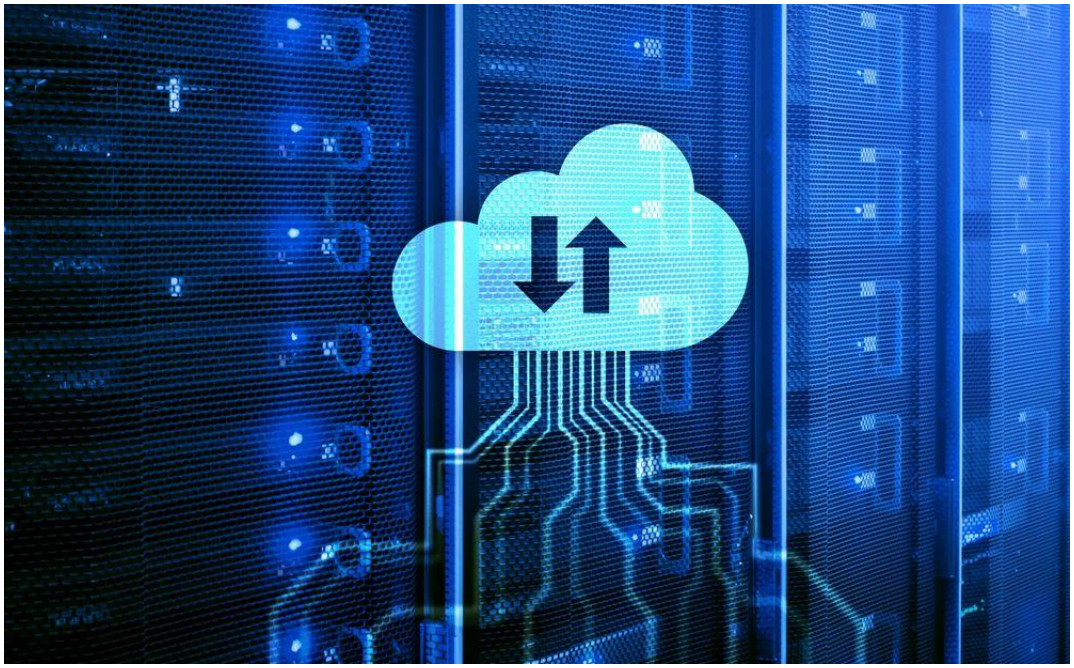
新しいインテル® SHMEM の紹介

この記事は、インテルのウェブサイトで公開されている「Introducing the New Intel® SHMEM」の日本語参考訳です。原文は更新される可能性があります。原文と翻訳文の内容が異なる場合は原文を優先してください。



はじめに

この記事では、OpenSHMEM プログラミング・モデルを拡張し、SYCL クロスプラットフォーム C++ プログラミング環境を使用してインテル® データセンター GPU をサポートするインテル® SHMEM を紹介します。

最初のオープンソース版であるインテル® SHMEM v1.0.0 リリース

グラフィックス・プロセシング・ユニット (GPU) は、ハイパフォーマンス・コンピューティング (HPC) や人工知能 (AI) アプリケーションで広く利用されています。インテル® データセンター GPU Max 1550 などの GPU は、数千の並列スレッドと膨大なメモリー帯域幅を提供します。これらは、CPU の負荷を軽減する絶好の可能性をもたらします。

複数の GPU が接続されたシステムでは、それらを独立したデバイスとして扱うか、インテル® MPI ライブラリー、インテル® oneAPI コレクティブ・コミュニケーション・ライブラリー (インテル® oneCCL)、インテル® SHMEM などの高性能通信ライブラリーを使用して、すべての GPU を使用するプログラムを作成できます。

インテル® MPI ライブラリーとインテル® oneCCL は、ホスト CPU コードから呼び出し可能なホスト中心のライブラリーですが、ホストとデバイスメモリーの両方のデータを処理できます。

インテル® SHMEM も同様の機能を備えていますが、GPU 上で動作する SYCL カーネルから通信関数を呼び出す機能が追加されています。これにより、計算と通信のオーバーラップを細かく制御し、多くの場合、複数のカーネル呼び出しによるオーバーヘッドを回避できます。

ほかのアクセラレーター・ベンダーも、それぞれのハードウェア・プラットフォーム向けに最適化された同様の機能を提供しています (例: NVIDIA の NVSHMEM、AMD の ROC_SHMEM)。

この記事では、OpenSHMEM、GPU、SYCL について簡単に説明してから、インテル® SHMEM とその用途について説明します。

OpenSHMEM

OpenSHMEM は、分散メモリーシステムにおける SPMD (Single Program Multiple Data) スタイルのプログラミングを可能にするライブラリー・インターフェイス標準です。SPMD を使用すると、スーパーコンピューターやコンピューター・クラスター全体で複数のコピーを実行する単一のプログラムを作成できます。コピーは、0 から N-1 までの一意のプロセシング要素 (PE) 識別子によって識別されます (N はコピーの総数)。OpenSHMEM は、効率的な分散プログラミングに役立ついくつかの操作を提供します。

- リモート・メモリー・アクセス (RMA)
- アトミックメモリー操作 (AMO)
- 同期
- 集合操作

OpenSHMEM プログラミングについては、分散コンピューティングに関する以前のブログシリーズの記事「OpenSHMEM によるメモリー空間の管理」(英語) で説明しました。

OpenSHMEM (英語) は、本稿執筆時点でバージョン 1.5 の包括的な仕様書 (英語) を提供しています。OpenSHMEM ライブラリーのいくつかの実装はオープンソースであり、無料でダウンロードできます。さまざまなトランスポート・メカニズムをサポートするオープン実装として、Sandia OpenSHMEM (英語) と、メッセージ・パッシング・インターフェイス (MPI) 上に OpenSHMEM を実装した OSHMPI (英語) を推奨します。

「分散通信入門: 分散コンピューティングと並列コンピューティングの歴史を振り返る」(英語) では、一般的な分散コンピューティングと並列コンピューティングの背景についてさらに詳しく説明しています。

要約すると、OpenSHMEM は、同じ実行可能プログラムのコピーが複数のノード上で起動され、各コピーが PE 番号を介してプログラム全体におけるワーカーとしての自身の ID を認識する計算モデルを提供します。では、PE はどのように相互にデータを転送するのでしょうか? ここで、各 PE 内のメモリー空間の特別な領域、つまり対称メモリーが重要な役割を果たします。OpenSHMEM の対称メモリーはリモートからアクセス可能であるため、他の PE はこのメモリーに対して読み取り、書き込み、その他の操作を実行できます。

OpenSHMEM には 2 つの形式の対称メモリーがあります。

- 静的変数とグローバル変数をサポートするデータセグメント
- `shmem_malloc()` などのルーチンを介して動的に割り当てられたオブジェクトをサポートする、対称ヒープと呼ばれる動的に割り当てられた領域

すべての PE は同じサイズの対称ヒープを持ちますが、ヒープは異なるアドレスから始まる可能性があります。オブジェクトは各ヒープにまとめて割り当てられるため、すべての PE でオブジェクト・セットの名前とサイズは同じになります。これにより、OpenSHMEM ではリモート・オブジェクトをローカルアドレスとリモート PE の ID の組み合わせで参照できます。そのため、共有メモリー・プログラミングと似た感覚で分散プログラミングを行うことが可能です。

OpenSHMEM は以下の操作を提供します。

- リモート・メモリー・アクセス: `put`、`get`、`put-with-signal`
- リモートアトミック操作: `store`、`fetch`、`swap`、`fetch-and-add`、およびビット単位のアトミック操作 (`and`、`or`、`xor`)
- 同期操作: `barrier`、`sync`、`wait` など
- 集合操作: `broadcast`、`alltoall`、`reduce`、`collect` (MPI の `AllGather` など)

SYCL による GPU プログラミング

SYCL は OpenCL の進化版です。GPU の並列コンピューティングの能力を活用したいアプリケーション向けに、クロスプラットフォームでオープンな C++ プログラミング環境を提供します。SYCL とその利点については、ブログ記事「SYCL で C++ ワークロードにマルチプラットフォーム並列処理を追加」(英語) とウェビナー (英語) で詳しく説明されています。

SYCL を使ったプログラミングを学習するリソースは数多くあり、その中には書籍も含まれています。

J. Reinders et. al. **Data Parallel C++: Programming Accelerated Systems using C++ and SYCL** (英語), 2nd Edition (2023), Apress, ISBN: 978-1-4842-9691-2

「データ並列 C++: SYCL* の oneAPI 実装」リソースページもあります。

基本的な考え方は、CPU 上で実行されるプログラムが GPU 上で並列「カーネル」を起動し、超並列計算を実行するというものです。SYCL には、ホストと GPU のメモリーシステム間でデータを手動または自動でコピーする機能も備わっています。これは「統合共有メモリー」と呼ばれます。

GPU メモリー上に配列を作成して初期化するには、次のように記述します。

```
double *data = sycl::malloc_device<double>(10000, queue);
queue.parallel_for(sycl::range<1>(10000) [=](sycl::id<1> idx) {
    data[idx] = <something>;
}).wait();
```

このコード例は自動的に GPU コードにコンパイルされ、実行されると数千の GPU スレッド上で実行され、各スレッドが同時に単一の配列要素を初期化する可能性があります。もちろん、このような小規模の問題ではこのパワーはあまり意味がありませんが、大規模な問題では飛躍的な高速化につながる可能性があります。

GPU 上で SYCL カーネルを活用した実用的な例が、インテルのブログでいくつか紹介されています。

- モンテカルロ・シミュレーション・コード: マルチ GPU CUDA から C++ with SYCL への移行 (英語)
- AutoDock-GPU: SYCL を活用した科学および医学向け分子スクリーニング (英語)
- NAMD: SYCL を活用した高度に並列化されたスケーラブルなナノスケール・シミュレーション (英語)
- Ginkgo: SYCL を活用した線形代数 (英語)

SYCL を活用することで、インテル® oneAPI DPC++/C++ コンパイラー環境はインテル® データセンター GPU マックス・シリーズのダイレクト・プログラミングを可能にします。HPC およびデータセンター向けの新しい GPU 製品ラインナップの詳細については、以下の資料をご覧ください。

- インテル® データセンター GPU
- インテル® データセンター GPU マックス・シリーズ
- oneAPI GPU 最適化ガイド

インテル® SHMEM

インテル® SHMEM は、SYCL で実装されたデバイスカーネルでアプリケーションが OpenSHMEM 通信 API を使用できるようにする C++ ソフトウェア・ライブラリーです。インテル® データセンター GPU マックス・シリーズをはじめとする、インテルの HPC/AI 向け GPU をサポートしています。

最初のオープンソース・リリースであるインテル® SHMEM v1.0.0 が利用可能になりました!

インテル® SHMEM (英語) は、パーティション化されたグローバルアドレス空間 (PGAS) プログラミング・モデルを実装し、現在の OpenSHMEM 標準 (英語) のホスト開始操作のサブセットと、GPU カーネルから直接呼び出せる新しいデバイス開始操作が含まれています。以下は、インテル® SHMEM v1.0.0 で利用可能な機能の概要です。

- プログラミング・モデル、サポートされる API、サンプルプログラム、ビルドおよび実行手順などを詳細に説明した完全な仕様 (英語)。
- OpenSHMEM 1.5 準拠のポイントツーポイントのリモート・メモリー・アクセス (RMA)、アトミックメモリー操作、シグナリング、メモリーオーダー、同期操作をサポートするデバイスおよびホスト API。
- OpenSHMEM 集合操作に対するデバイスおよびホスト API のサポート。
- リモート・メモリー・アクセス、シグナリング、集合、メモリーオーダー、および同期操作の SYCL ワークグループおよびサブグループ・レベル拡張に対するデバイス API のサポート。
- OpenSHMEM 仕様の C11 汎用選択ルーチンを置き換える C++ テンプレート関数ルーチンのサポート。
- 高性能ネットワーク・サービス向けに適切な Libfabric (英語) プロバイダーを備えた Sandia OpenSHMEM (英語) で構成された GPU RDMA (リモート・ダイレクト・メモリー・アクセス) サポート。
- SHMEM 対称ヒープ用にデバイスメモリー (デフォルト) または統合共有メモリーを選択可能。

インテル® SHMEM では、OpenSHMEM API はホストコードと GPU コードの両方から呼び出すことができます。GPU スレッドから呼び出される OpenSHMEM 関数は、通常バージョンと「ワークグループ」拡張バージョンの両方で利用できます。`ishmem_put` などの通常関数は、利用可能なすべての GPU スレッドから独立して呼び出すことができます。一方、`ishmemx_put_work_group` などの「ワークグループ」拡張バージョンは、SYCL ワークグループ内のすべての GPU スレッドからまとめて呼び出され、連携して機能を実現します。

インテル® SHMEM v1.0.0 の完全なソースコードは、GitHub リポジトリー (英語) からアクセスできます。

- インテル® SHMEM のビルド、インストール、および使用方法については、README.md (英語) に記載されています。
- インテル® SHMEM API を利用したプログラムの作成については、詳細なガイド (英語) を参照してください。
- さまざまな API の使用方法を示すサンプルコード (英語) も提供されています。

インテル® SHMEM の使用

分散 2D ステンシル演算を実行するプログラムを考えてみましょう。ステンシルでは、各反復処理において、行列内の各セルは、自身と周囲のセルに対する計算に置き換えられます (例えば、1970 年に John Conway が実装したライフゲームの実装がその一例です)。

このような計算を分散処理するには、各 PE がバンドやブロックなどの行列の領域を担当します。PE は、担当領域の端を処理する際に、隣接する PE の影響を受けるため、協調して処理する必要があります。

次のコードでは、大きな行列が行ごとに複数の PE に分散されています。各 PE は、行列の担当領域の更新値を計算し、次のタイムステップで使用するため、隣接する行列領域を担当する PE にデータを送信します。これは分散計算でよく見られるパターンです。複数の PE によって共有される領域は「ハロー」と呼ばれます。

GitHub (英語) 上のインテル® SHMEM ディストリビューションに同梱されている Jacobi 法 (英語) のサンプル・アプリケーション・ソースコードの一部を見てみましょう。

以下は、インテル® SHMEM を使用して SYCL でこれらの更新を実行する方法の 1 つです。

```
h.parallel_for(
  sycl::nd_range<2>{global_range, local_range}, sum_norm,
                    [=](sycl::nd_item<2> it, auto &sumr) {
    size_t iy = it.get_global_id(1) + iy_start;
    size_t ix = it.get_global_id(0) + 1;
    if (iy < iy_end && ix < (nx - 1)) {
      const real new_val = static_cast<real>(0.25) *
        (a[iy * nx + ix + 1] + a[iy * nx + ix - 1] +
         a[(iy + 1) * nx + ix] + a[(iy - 1) * nx + ix]);

      a_new[iy * nx + ix] = new_val;

      // apply boundary conditions
      if (iy_start == iy) {
        ishmem_float_p((real *) (a_new + top_iy * nx + ix), new_val,
                       top_pe);
      }
      if (iy_end - 1 == iy) {
        ishmem_float_p((real *) (a_new + bottom_iy * nx + ix),
                       new_val, bottom_pe);
      }
    }
  });
```

このコードでは、ある位置の更新値は隣接する 4 つの値の平均です。位置が領域の境界上にある場合、更新値は隣接する PE にも送信されます。

各タイムステップの後、すべての PE は `ishmem_barrier_all()` で同期し、ある PE が別の PE によって書き込まれた値を使用する前に、隣接する PE のすべての更新が完了するようにします。

すべての PE が同じ計算ノードに配置されている構成では、`ishmem_float_p` の呼び出しが Xe リンクを介して GPU 上の隣接 PE のメモリーにデータを書き込む単一のストア命令で解決されるため、このコードは非常に効率的です。

PE が配置されていない構成では、リモート更新は単一セルの計算に比べて時間がかかる可能性があるので、非ブロッキング `work_group` API を使用して、バックグラウンドでバッチ通信として実行するのが有効です。この場合、領域の端にある行の計算を実行する SYCL ワークグループは、インテル® SHMEM API の `ishmemx_put_work_group_nbi` を使用して、行全体の情報を隣接する PE に送信できます。

今すぐインテル® SHMEM を使い始めましょう

インテル® MPI ライブラリー、インテル® oneCCL (英語)、そしてインテル® SHMEM (英語) を使って、SYCL をサポートする幅広い GPU およびアクセラレーター・デバイス上で分散 HPC および AI アプリケーションを今すぐ実現しましょう。

オープンソースのインテル® SHMEM GitHub プロジェクト (英語) への皆様の貢献や質問をお待ちしています。プロジェクトへの貢献や問題の報告は、CONTRIBUTING.md (英語) に記載されているガイドラインに従ってください。

その他のご質問は、ishmem-discuss@intel.com までメールでお問い合わせください。

関連情報

- OpenSHMEM によるメモリー空間の管理 (英語)
- 分散通信入門 (英語)
- OpenSHMEM プロジェクト (英語)

インテル® SHMEM のリソース

- GitHub (英語)
- README.md (英語)
- インテル® SHMEM プログラムの書き方 (英語)
- インテル® SHMEM のサンプル (英語)