バイトニック・ソート

サンプル・ユーザーズ・ガイド

インテル® SDK for OpenCL* Applications - サンプル

資料番号: 325262-003JA

目次

著作権と商標について

本資料に掲載されている情報は、インテル製品の概要説明を目的としたものです。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を許諾するものではありません。製品に付属の売買契約書『Intel's Terms and Conditions of Sale』に規定されている場合を除き、インテルはいかなる責任を負うものではなく、またインテル製品の販売や使用に関する明示または黙示の保証 (特定目的への適合性、商品適格性、あらゆる特許権、著作権、その他知的財産権の非侵害性への保証を含む) に関してもいかなる責任も負いません。

「ミッション・クリティカルなアプリケーション」とは、インテル製品がその欠陥や故障によって、直接的または間接的に人身傷害や死亡事故が発生するようなアプリケーションを指します。そのようなミッション・クリティカルなアプリケーションのためにインテル製品を購入または使用する場合は、直接的か間接的かにかかわらず、あるいはインテル製品やそのいかなる部分の設計、製造、警告にインテルまたは委託業者の過失があったかどうかにかかわらず、製造物責任、人身傷害や死亡の請求を起因とするすべての賠償請求費用、損害、費用、合理的な弁護士費用をすべて補償し、インテルおよびその子会社、委託業者および関連会社、およびそれらの役員、経営幹部、従業員に何らの損害も与えないことに同意するものとします。

インテル製品は、予告なく仕様や説明が変更される場合があります。機能または命令の一覧で「留保」または「未定義」と記されているものがありますが、その「機能が存在しない」あるいは「性質が留保付である」という状態を設計の前提にしないでください。これらの項目は、インテルが将来のために留保しているものです。インテルが将来これらの項目を定義したことにより、衝突が生じたり互換性が失われたりしても、インテルは一切責任を負いません。この情報は予告なく変更されることがあります。この情報だけに基づいて設計を最終的なものとしないでください。

本資料で説明されている製品には、エラッタと呼ばれる設計上の不具合が含まれている可能性があり、公表されている仕様とは異なる動作をする場合があります。現在確認済みのエラッタについては、インテルまでお問い合わせください。

最新の仕様をご希望の場合や製品をご注文の場合は、お近くのインテルの営業所または販売代理店にお問い合わせください。

本資料で紹介されている資料番号付きのドキュメントや、インテルのその他の資料を入手するには、1-800-548-4725 (アメリカ合衆国) までご連絡いただくか、http://www.intel.com/design/literature.htm (英語) を参照してください。

インテル・プロセッサー・ナンバーはパフォーマンスの指標ではありません。プロセッサー・ナンバーは同一プロセッサー・ファミリー内の製品の機能を区別します。異なるプロセッサー・ファミリー間の機能の区別には用いません。詳細については、http://www.intel.co.jp/jp/products/processor_number/ 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。

Intel、インテル、Intel ロゴは、アメリカ合衆国およびその他の国における Intel Corporation の商標です。

* 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。

OpenCL および OpenCL ロゴは、Apple Inc. の商標であり、Khronos の使用許諾を受けて使用しています。Microsoft 製品のスクリーンショットは、Microsoft Corporation の許可を得て使用しています。

最適化に関する注意事項

インテル® コンパイラーは、互換マイクロプロセッサー向けには、インテル製マイクロプロセッサー向けと同等レベルの最適化が行われない可能性があります。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® SSE2)、インテル® ストリーミング SIMD 拡張命令 3 (インテル® SSE3)、ストリーミング SIMD 拡張命令 3 補足命令 (SSSE3) 命令セットに関連する最適化およびその他の最適化が含まれます。インテルでは、インテル製ではないマイクロプロセッサーに対して、最適化の提供、機能、効果を保証していません。本製品のマイクロプロセッサー固有の最適化は、インテル製マイクロプロセッサーでの使用を目的としています。インテル® マイクロアーキテクチャーに非固有の特定の最適化は、インテル製マイクロプロセッサー向けに予約されています。この注意事項の適用対象である特定の命令セットの詳細は、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。

改訂 #20110804

バイトニック・ソート・サンプルについて

バイトニック・ソート・サンプルは、OpenCL* C99 を使用して計算カーネルを実装し、いくつかのワークグループを並列に実行することでカーネルを並列化します。

このサンプルは、OpenCL* と SIMD (Single Instruction Multiple Data) バイトニック・ソート・ネットワークを使用して、任意の整数配列 (入力) をソートします。ここで紹介する実装は一般的なものなので、比較的簡単に*<key/value>* ソートを追加することができます。

目的

ソート・アルゴリズムは、最も広範に使用されているビルディング・ブロックです。このサンプルで実装するバイトニック・ソート・アルゴリズムは、バイトニック・シーケンスの特性とソート・ネットワークの原則に基づいています。OpenCL* ベクトルデータ型により効率良い SIMD 並列化を有効にします。

アルゴリズム

長さ 2N*4 の配列で、このアルゴリズムは N ステージのソートを行います。最初のステージは 1 パスです。カーネルは、入力配列の各項目内で SIMD ソート・ネットワークを使用して、サイズ 4 のバイトニック・シーケンスを生成します。

後続のステージでは、パスは 1 ずつ増え、シーケンスのサイズは隣接する項目をマージするため倍になります。

バイトニック・ソート・ネットワークについては、「Bitonic Sort」(英語) (http://www.iti.fh- flensburg.de/lang/algorithmen/sortieren/bitonic/bitonicen.htm) を参照してください。SIMD データ型を利用するソート・ネットワークについては、「Efficient implementation of sorting on multi-core SIMD CPU architecture」(英語) (http://portal.acm.org/citation.cfm?id=1454159.1454171&coll=GUIDE&dl=GUIDE&CFID=105910684& CFTOKEN=82233064) を参照してください。

OpenCL* 実装

コードの概要

BitonicSort.cl ファイルのバイトニック・ソート OpenCL* カーネルは、各パスの指定されたステージを実行します。入力配列の各項目または項目のペア (パス番号による) は、一意のグローバル ID に対応しており、カーネルはこれによって各項目または項目のペアを識別します。ソートシーケンスは、BitonicSort.cpp ファイルの ExecuteSortKernel() 関数で繰り返しカーネル呼び出しを実行します。

制限事項

説明を単純にするため、サンプルコードの現在のバージョンでは、4*2^N 32 ビットの整数項目から成る入力配列のみ使用できます。N は正の整数です。

OpenCL* パフォーマンスの特性の理解

ベクトルデータ型を使用する利点

このサンプルは、ベクトルデータ型を使用してバイトニック・ソート・アルゴリズムを実装します。int4 や float4 のように、これらの型を明示的に使用することで、次の最適化が有効になります。

- 1 つの整数の代わりに、4 つの整数を処理できます。これにより、不要な分岐を排除し、メモリー帯域幅を抑え、CPU キャッシュ利用状況を最適化できます。
- 各ステージの最終パスでは、1 つのベクトル項目内でソート・ネットワークを利用できます。そのため、2 つの最終パスをマージして、各ステージの余分なカーネル呼び出しを排除し、実行オーバーヘッドを軽減できます。

SIMD レジスターの利用によって最大 4 倍のスピードアップが得られますが、これらの最適化は明示的なベクトルバージョンでさらに 25% のスピードアップをもたらします。そのため、全体では約 5 倍のスピードアップを達成できます。

ワークグループ・サイズについての考察

インテル・プラットフォームにおける有効なワークグループ・サイズは 1 ～ 8192 要素の範囲です。サンプルでは、実行時にデバイスに応じて最適なワークグループ・サイズを選択できるように、ローカル・ワークグループ・サイズに NULL 引数を使用しています。

リファレンス (ネイティブ) 実装

リファレンス実装は、BitonicSort.cpp ファイルの ExecuteSortReference() ルーチンで実行されます。このシングルスレッド・コードは、OpenCL* コードと同じバイトニック・ソート・シーケンスを実行しますが、入れ子構造の純粋なスカラー C ループを使用します。

サンプルの制御

サンプル実行ファイルは、コンソール・アプリケーションです。ソート方向と入力配列のサイズは、コマンドライン引数で設定します。コマンドラインが空の場合、サンプルはデフォルト値を使用します。

サンプルは、以下のコマンドライン引数をサポートします。

オプション	説明
`-h、--help`	このヘルプを表示し、終了します。
`-p、--platform number-or-string`	使用するデバイスのプラットフォームを選択します。
`-t、--type all | cpu | gpu | acc | default | <デバイスタイプの OpenCL* 定数>`	OpenCL* カーネルを実行するデバイスタイプを選択します。
`-d、--device number-or-string`	すべてを実行するデバイスを選択します。
`-r、--reverse-sort`	降順にソートします (デフォルトは昇順です)。
`-s、--size <integer>`	入力/出力配列のサイズを設定します。

参考文献

- H. W. Lang. Bitonic Sort at http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/bitonic/bitonicen.htm

- Jatin Chhugani, Anthony D. Nguyen, Victor W. Lee, William Macy, Mostafa Hagog, Yen-Kuang Chen, Akram Baransi, Sanjeev Kumar, Pradeep Dubey: Efficient implementation of sorting on multi-core SIMD CPU architecture. PVLDB 1(2): 1313-1324 (2008) at http://portal.acm.org/citation.cfm?id=1454159.1454171&coll=GUIDE&dl=GUIDE&CFID=105910684&CFTOKEN=82233064