インテル® Xeon® スケーラブル・プロセッサーのインテル® AVX-512 のベクトル長の拡張機能

この記事は、インテル® デベロッパー・ゾーンに公開されている「The Intel® Advanced Vector Extensions 512 (Intel® AVX-512) Vector Length Extensions Feature on Intel® Xeon® Scalable Processors」の日本語参考訳です。

はじめに

インテル® Xeon® スケーラブル・プロセッサーは、増加するパフォーマンス要件に対応するため、要求が厳しい計算ワークロードのパフォーマンスを向上する新しい命令セット、インテル® アドバンスト・ベクトル・エクステンション 512 (インテル® AVX-512) をサポートしています。

インテル® AVX-512 命令セットの仕様は、いくつかのサブセットから構成されます。

- 基本命令
- 競合検出命令 (CDI)
- バイト (char または int8) およびワード (short または int16) 命令
- ダブルワード (int32 または int) およびクワッドワード (int64 または long) 命令
- ベクトル長の拡張 (VLE)

各サブセットの詳細は、「ベクトル化とインテル® Xeon® スケーラブル・プロセッサーを使用したパフォーマンスの向上」と「インテル® AVX-512 はインテル® Xeon® スケーラブル・プロセッサーの秘宝かもしれない」(英語) を参照してください。

インテル® AVX-512 は、最新のハードウェアおよび将来の世代の並列ハードウェアで、コードのベクトル化の効率を向上します。新しい命令は 512 ビット・レジスターを利用できるだけでなく、ベクトル化に利点をもたらす新しい機能を提供しており、また 32 個のベクトルレジスターを処理することができます。インテル® Xeon® スケーラブル・プロセッサーでは、インテル® AVX-512 のこれらの新機能を異なるサイズのレジスターでも利用することができるため、多くのアプリケーションでその利点を得ることができます。この新機能は、追加の直交機能であるベクトル長の拡張によって提供されます。

ベクトル長の拡張とは?

インテル® AVX-512 は、256 ビットをサポートするインテル® アドバンスト・ベクトル・エクステンション 2 (インテル® AVX2) 命令セットと比較して、2 倍のベクトルレジスターをサポートし、各ベクトルレジスターで 2 倍の単精度浮動小数点数または倍精度浮動小数点数をパックできます。レジスターが同時に処理するデータを多く保持でき、CPU サイクルごとに多くのワークを実行できます。

しかし、アプリケーションによっては (ベクトル化されたループが少ないアプリケーションやトリップカウントが小さいアプリケーションなど)、512 ビット・レジスターの利点が得られません。VLE 機能は、そのようなアプリケーションで、通常の 512 ビット (ZMM レジスター) に加えて、128 ビット (XMM レジスター) と 256 ビット (YMM レジスター) でほとんどのインテル® AVX-512 命令を利用できるようにします。これらのインテル®

AVX-512 命令は、異なるベクトル長で実行しても、より多くのレジスター (コアごとに 32 個) と opmask レジスター (8 個) を利用できます。

ほとんどのインテル® AVX-512 命令は、アルゴリズムやデータによって制限された異なるベクトル長で使用できるため、VLE の新機能を使用することで、アプリケーションの最適化の選択肢が広がります。

例

VLE 直交機能の例として、ここでは画像のヒストグラムを計算するサンプルコードを使用します。アルゴリズムとコードの説明は、「インテル® AVX-512 でベクトル化のパフォーマンスを向上する」を参照してください。サンプルコードもこの記事からダウンロードできます。これは、説明を目的としたサンプルコードです。

このサンプルコードには、ヒストグラムを計算する配列内の間接参照によるデータ依存関係が原因で、インテル® C++ コンパイラーによってインテル® AVX2 命令セットアーキテクチャー (ISA) でベクトル化されないループがあります。しかし、インテル® AVX-512 ISA を使用すると、コンパイラーは CDI サブセットの命令でこれらのループをベクトル化することができます。

CDI サブセットには、ベクトルレジスターのデータ競合を検出して、その情報をマスクベクトルに格納する機能があります。格納された情報は、ベクトル計算に使用され、前述の記事で説明されているように、配列の競合しない要素 (グレースケール値が同じもの) のみが同時に処理されます。

次の 3 つの実験では、異なるオプションのセットを使用して、インテル® C++ コンパイラーでサンプルコードをコンパイルします。最後の実験では、コンパイラーに ZMM (512 ビット) レジスターの代わりに YMM (256 ビット) レジスターを使用するように指示しても、CDI 機能が使用されることが分かります。

実験 1: 最初に、インテル® AVX2 オプションでサンプルコードをコンパイルします。

```
icpc Histogram_Example.cpp -O3 -restrict -qopt-report -qopt-report-file=runAVX2.optrpt
-xCORE-AVX2 -lopencv_highgui -lopencv_core -lopencv_imgproc -o runAVX2
```

ここでは、最適化レポートを生成するため、インテル® C++ コンパイラーの `-qopt-report` オプションを追加しています。以下は、フィルターとヒストグラムを計算するループ (行 107 のループ) がベクトル化されなかったことを示す最適化レポートの部分です。

```
ループの開始 Histogram_Example.cpp(107,5)
    リマーク #15344: ループはベクトル化されませんでした: ベクトル依存関係がベクトル化を妨げています。最初の依存関係を以下に示します。詳細については、レベル 5 のレポートを使用してください。
ループの終了
```

実験 2: ZMM (512 ビット) レジスターを使用するようにインテル® C++ コンパイラーの `-qopt-zmm-usage=high` オプションを追加して、インテル® AVX-512 オプションでコンパイルすると、想定どおり、最適化レポートはループがベクトル化されたことを示します。

```
icpc Histogram_Example.cpp -O3 -restrict -qopt-report -qopt-report-file=runAVX512.optrpt
-xCORE-AVX512 -qopt-zmm-usage=high -lopencv_highgui -lopencv_core -lopencv_imgproc -o
runAVX512
```

```
ループの開始 Histogram_Example.cpp(107,5)
    リマーク #15300: ループがベクトル化されました。
ループの終了
```

-qopt-zmm-usage=high/low は、インテル® コンパイラー 18.0 で追加された新しいオプションで、インテル® Xeon® スケーラブル・プロセッサーにおいて SIMD ベクトル化をより柔軟に使用できるようにします。上記のように、この新しいオプションを -xCORE-AVX512 と一緒に使用します。また、コンパイラーに最適化レポートの生成を指示する -qopt-report も一緒に指定すると良いでしょう。最適化レポートは、開発者がコンパイラーによる最適化を理解し、さらなる最適化の可能性を探るのに役立ちます。インテル® コンパイラーの新機能の詳細は、「インテル® Xeon® スケーラブル・プロセッサー向け SIMD ベクトル化のチューニング」を参照してください。

-qopt-report を -qopt-report=5 に変更すると、コンパイラーはより詳細はベクトル化レポートを生成します。レポートから、ZMM レジスターに 16 個の float を格納するコードが生成されたことが分かります。

```
ループの開始 Histogram_Example.cpp(107,5)
    (…)
    リマーク #15416: ベクトル化のサポート: 不規則インデックス ストアが生成されました (変数
<hist2[*(image2+position*4)]>、マスク付き、インデックスの一部 メモリーからの読み取り
[ Histogram_Example.cpp(122,8) ])
    リマーク #15415: ベクトル化のサポート: 不規則インデックス ロードが生成されました (変数
<hist2[*(image2+position*4)]>、マスク付き、インデックスの一部 メモリーからの読み取り
[ Histogram_Example.cpp(122,8) ])
    リマーク #15305: ベクトル化のサポート: ベクトル長 16
    リマーク #15300: ループがベクトル化されました。
    (…)
ループの終了
```

また、アセンブリー・コード (-S オプションでインテル® C++ コンパイラーによって生成される) を確認すると、ループのヒストグラム計算 (ソースコードの行 122) がベクトル化されたことが分かります。

```
hist2[ int(image2[position]) ]++;
```

CDI サブセットの競合検出命令を ZMM レジスターで使用しています。

```
vpbroadcastmw2d %k2, %zmm6                                  #122.8
vpconflictd %zmm4, %zmm2{%k2}{z}                            #122.8
vpandd    %zmm6, %zmm2, %zmm5                               #122.8
```

この実験では、コンパイラーに ZMM レジスターの使用を指示するオプションを組み合わせて (-xCORE-AVX512 -qopt-zmm-usage=high) コードをコンパイルした場合に、想定どおりの結果が得られました。

実験 3: プログラムは ZMM レジスターの恩恵を受ける可能性が低く、より幅の狭いレジスターを使用する可能性が高いことをコンパイラーに伝える、 -xCORE-AVX512 -qopt-zmm-usage=low オプションを使用してコンパイルします (注: -qopt-zmm-usage=low は -xCORE-AVX512 のデフォルト)。

```
icpc Histogram_Example.cpp -O3 -restrict -qopt-report -qopt-report-file=runAVX512.optrpt -
xCORE-AVX512 -qopt-zmm-usage=low  -lopencv_highgui -lopencv_core -lopencv_imgproc -o
runAVX512
```

```
ループの開始 Histogram_Example.cpp(107,5)
    リマーク #15300: ループがベクトル化されました。
    リマーク #15321: コンパイラーは XMM/YMM ベクトルをターゲットとして選択しました。ZMM を使用するには、-
qopt-zmm-usage=high を指定してオーバーライドしてください。
ループの終了
```

-qopt-report を -qopt-report=5 に変更すると、コンパイラーはより詳細なレポートを出力します。

```
ループの開始 Histogram_Example.cpp(107,5)
    (…)
    リマーク #15416: ベクトル化のサポート: 不規則インデックス ストアが生成されました (変数
<hist2[*(image2+position*4)]>、マスク付き、インデックスの一部 メモリーからの読み取り
[ Histogram_Example.cpp(122,8) ])
    リマーク #15415: ベクトル化のサポート: 不規則インデックス ロードが生成されました (変数
<hist2[*(image2+position*4)]>、マスク付き、インデックスの一部 メモリーからの読み取り
[ Histogram_Example.cpp(122,8) ])
    リマーク #15305: ベクトル化のサポート: ベクトル長 8
    リマーク #15300: ループがベクトル化されました。
    (…)
ループの終了
```

上記のレポートから、今回は YMM (256 ビット) レジスターを使用するコードが生成されたことが分かります。アセンブリー・コードを確認すると、今回も競合検出命令を使用してループがベクトル化されていることが分かります。ただし、YMM レジスターが使用されています。

```
vpbroadcastmw2d %k2, %ymm5                                  #122.8
vpconflictd %ymm3, %ymm2{%k2}{z}                            #122.8
vpand     %ymm5, %ymm2, %ymm4                               #122.8
```

つまり、VLE 直交機能が使用されています。VLE 直交機能は、アプリケーションがより狭い幅のベクトル (このケースでは 256 ビット YMM レジスター) で CDI を利用できるようにします。これは、上記の実験 1 に示すとおり、インテル® AVX2 命令では不可能です。YMM レジスターを使用する場合であっても、アプリケーションはインテル® Xeon® スケーラブル・プロセッサーのより多くのレジスター (コアごとに 32 個) と opmask レジスター (8 個) を利用することができます。

まとめ

インテル® Xeon® スケーラブル・プロセッサーのインテル® AVX-512 命令セットは、コアごとのレジスター数の増加と 512 ビット・レジスターで実行可能なベクトル演算などの拡張されたベクトル処理機能により、いくつかのワークロードと用途でパフォーマンスを向上できます。

インテル® AVX-512 の新機能をより多くのアプリケーションで利用できるようにするため、新しい VLE 直交機能は、より多くのレジスターを利用しつつ、ほとんどのインテル® AVX-512 命令をより幅の狭いベクトルで実行できるようにします。この新機能は、128 ビットまたは 256 ビット・レジスターで SIMD 操作を実行するアプリケーションに利点をもたらします。

ここでは、インテル® AVX-512 命令により CDI を使用した自動ベクトル化の利点が得られるサンプルコードを用いて VLE 直交機能を説明しました。VLE は、より幅の狭いレジスター (XMM または YMM) での処理を可能にし、CDI によるベクトル化やより多くのレジスターを使用できるようにすることで、このようなアプリケーションの適用範囲を広げます。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。