

インテル® Advisor XE 2016 を 利用したコードのベクトル化

パフォーマンスを向上する際に一般的な問題を解決する

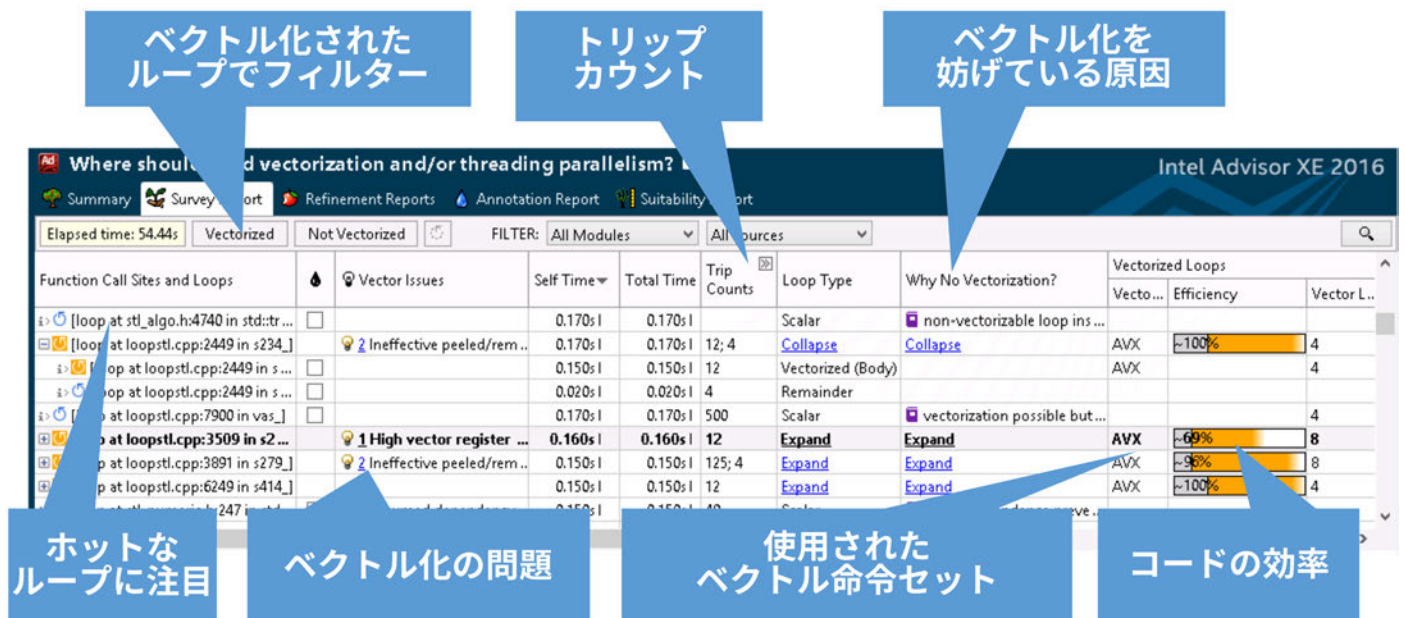
Kevin O'Leary インテル コーポレーション ソフトウェア・テクニカル・コンサルティング・エンジニア
Kirill Rogozhin インテル コーポレーション ソフトウェア開発マネージャー
Vadim Kartoshkin インテル コーポレーション テクニカルライター

多くの要因がプログラムの自動ベクトル化を妨げます。この記事では、コンパイラによるヒントがないとコードのベクトル化が困難である要因について検証します。ループのベクトル化は、アプリケーションのパフォーマンスを向上するために重要です。インテル® Advisor XE は、ベクトル化のプロセスをガイドします。

インテル® Advisor XE 2016 は、ベクトル化アドバイザーを含む動的な解析ツールです (図 1)。ベクトル化アドバイザーを利用することで、アプリケーションに含まれるすべてのループを調査し、次のことを確認できます。

- ベクトル化されたループとされなかったループ
- ループがベクトル化されなかった原因
- ベクトル化されたループのスピードアップとベクトル化の効率
- ベクトル化の効率を下げている要因
- メモリーレイアウトにより制約を受けているベクトル化されたループとされなかったループ

この記事では、ベクトル化アドバイザーの概要を提供し、次世代のインテル® Xeon Phi™ 製品 (開発コード名 Knights Landing) 上でのベクトル化を支援する新しい機能を紹介します。また、ベクトル化アドバイザーを利用して一般的な問題をベクトル化する方法を示します。

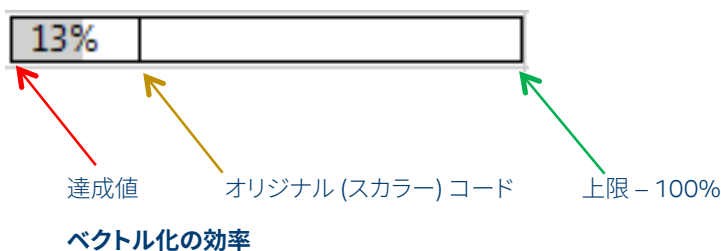


1 ベクトル化アドバイザー: 必要なすべてのデータに簡単にアクセス

ベクトル化の効率を向上するための 5 つのステップ

- 調査。**最初のステップでは、アプリケーションを調査します。このステップで、アプリケーションが時間を費やしているループが分かります。「ホットな」ループは、最適化による恩恵が最も得られる場所です。**図 2** は、アプリケーションの [Survey Report (調査レポート)] です。インテル® Advisor XE では、ループの種類 (ベクトル化されているかどうか) でフィルターすることができます。ベクトル化されなかったループには、ベクトル化を妨げている原因が表示されます。
- 推奨事項の確認。**ベクトル化の効率を向上する方法について具体的なアドバイスが得られます。また、ベクトル化を妨げている原因も表示されます。
- トリップカウント。**ループ反復のトリップカウントを個別の収集ステップとして収集します。ループがホットかどうかだけでなく、トリップカウントも把握することが重要です。トリップカウントが小さい場合、効率良くベクトル化するのに十分な反復がない可能性があります。トリップカウントがベクトル長の倍数かどうか確認することで、剰余ループが必要かどうか知ることができます。
- 依存性解析。**コードが正しい結果を生成するように、コンパイラーは、コンパイルしている言語のセマンティクスに対して保守的な見地に立たなければいけません。言語の規則に基づいて依存性が想定される場合、コンパイラーは依存性が存在すると仮定します。インテル® Advisor XE のような動的なツールを使用することにより、仮定した依存性が事実かどうか確認することができます。
- メモリー・アクセス・パターン (MAP) 解析。**データ構造がメモリー上にどのように配置され、ループでどのようにアクセスされるか知っていれば、アプリケーションのベクトル化の効率を大幅に引き上げることができます。メモリー参照がユニットストライド方式でアライメントされていることは非常に重要です。構造体配列 (AOS) から配列構造体 (SOA) へのデータ構造の変換のように、ベクトル化を支援するメモリーアクセス関連の手法があります。MAP 解析を使用すると、本質的にベクトル化が非効率なパターンを見つけ出すことができます。

Loops	Vecto...	Efficiency ▲	Estimated Gain	Vect...	Co.	Traits	Vector Widths	Self Time
④ [loop at lbpSUB.cpp:1280 in fPropagationS...	AVX	13%	0,53	4	0,53	Blends; Extracts; Inserts; Shuffles	128/256	2,312s
④ [loop at lbpGET.cpp:152 in fGetFracSite]	AVX	30%	2,38	8	2,34	Blends; Inserts; Masked Stores	128/256	0,030s
④ [loop at lbpGET.cpp:42 in fGetOneMassSite]	AVX	36%	2,86	8	2,79		256	0,100s
④ [loop at lbpGET.cpp:78 in fGetTotMassSite]	AVX	36%	2,86	8	2,79		256	0,010s
④ [loop at lbpGET.cpp:334 in fGetOneDirecSp ...]	AVX	38%	3,05	8	2,97	Type Conversions	128/256	0,011s
④ [loop at lbpBGK.cpp:840 in fCollisionBGK]	AVX	100%	2,05	2	2,05		128	0,080s



2 ベクトル化の効率: パフォーマンスのサーモメーター

ベクトル化アドバイザーは、ベクトル化によるコードの効率を予測できます。効率メトリックから、対応が必要な問題を含むループが分かります。ベクトル化されたループが効率的でない場合は、最初にベクトル化アドバイザーによってコードの効率を向上するための推奨事項が提供されていないか確認します。一般に、データ構造レイアウトはベクトル化の効率に大きく影響します。MAP 解析を実行することで、ベクトル化に適した方法でメモリーを参照しているか確認できます。

インテル® AVX-512 対応ハードウェアがない場合

次世代のインテル® Xeon Phi™ コプロセッサの実機がなくても、インテル® Advisor XE を利用して、コードを対応させることができます。インテル® コンパイラーの **-ax** オプションを使用して、複数のベクトル命令セット (インテル® AVX-512 を含む) 向けのコードを生成し、ベクトル化アドバイザーで解析します。

-ax オプションを指定してコードをコンパイルする

最初に、**-ax** オプションを指定して、コンパイラーに (デフォルトのコードパスに加え) 代替コードパスを含むバイナリーを生成するように指示します。例えば、次のコンパイラー・オプションは、インテル® SSE2 とインテル® AVX2 命令セット向けのコードを含むバイナリーを生成します。

```
-axCORE-AVX2
```

このオプションを指定すると、コンパイラーはインテル® SSE2 命令セット (**-x** または **-m** オプションで別の命令セットをデフォルトに指定しない限り、これがデフォルト) とインテル® AVX2 命令セット向けの代替コードパスを生成します。代替コードパスは、バイナリーを起動するシステムで対応する命令セットがサポートされている場合に実行されます。

ハイエンドのハードウェア (インテル® Xeon Phi™ 製品を搭載したマシンなど) 向けにコードを生成する場合は、最上位の命令セットを使用するコードの生成をコンパイラーに指示します。

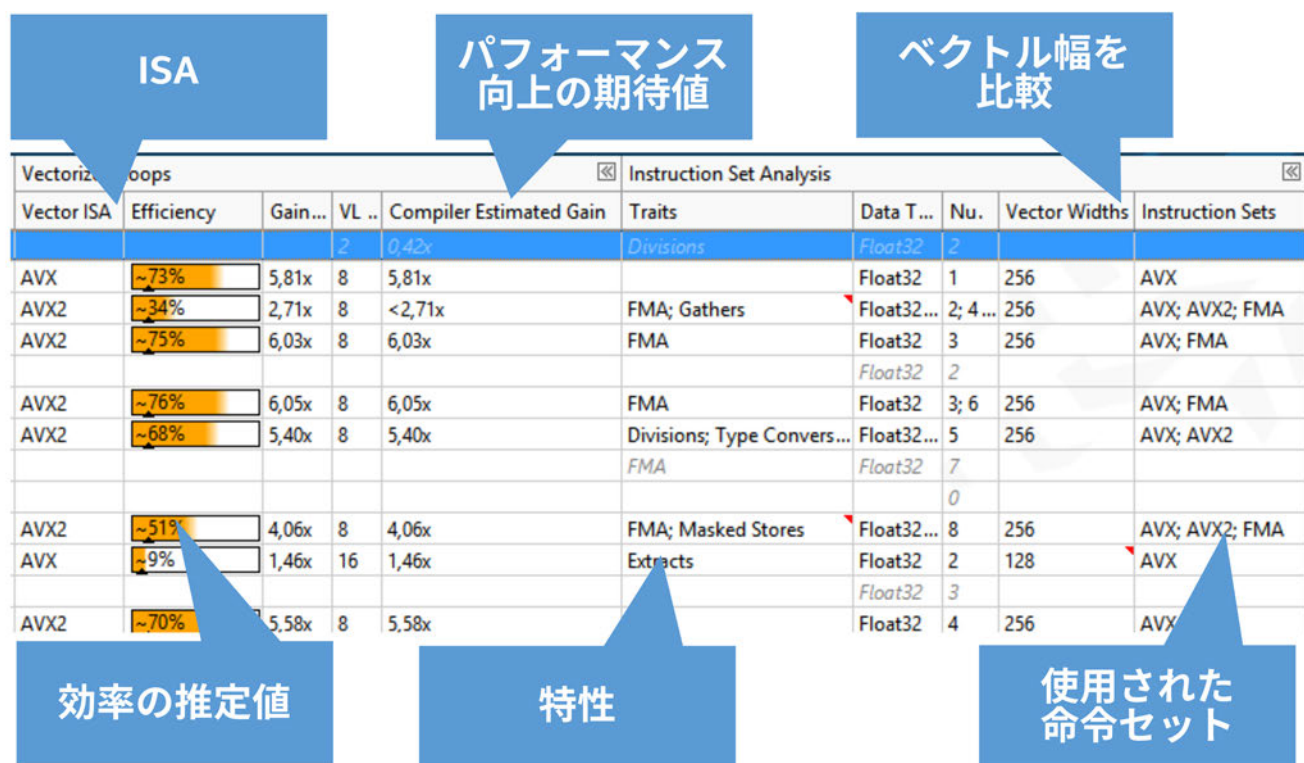
例えば、インテル® SSE4.1 以上をターゲットとし、インテル® AVX2 やインテル® AVX-512 命令セットをサポートするマシンではそれらの命令を使用するコードを生成する例について考えてみます。

この場合、コンパイラーは次のコードパスを含むコードを生成します。

```
-axCORE-AVX512, -axCORE-AVX2, -xsse4.1
```

- **-xsse4.1** は、ハードウェアが代替コードパスをサポートしていない場合、デフォルトのコードパスをインテル® SSE4.1 に変更します。
- **-axCORE-AVX2** は、インテル® AVX2 対応のハードウェア上で使用される 1 つ目の代替コードパスを設定します。
- **-axCORE-AVX512** は、インテル® AVX-512 対応のハードウェア上で使用される 2 つ目の代替コードパスを設定します。

各命令セット・アーキテクチャ (ISA) 向けに生成されたコードの違いは、インテル® Advisor XE の [Survey Report (調査レポート)] で、実行されなかったループを確認すると分かります。このレポートには、各 ISA で使用された特性に関する情報があり、パフォーマンスの期待値を比較することができます (図 3)。

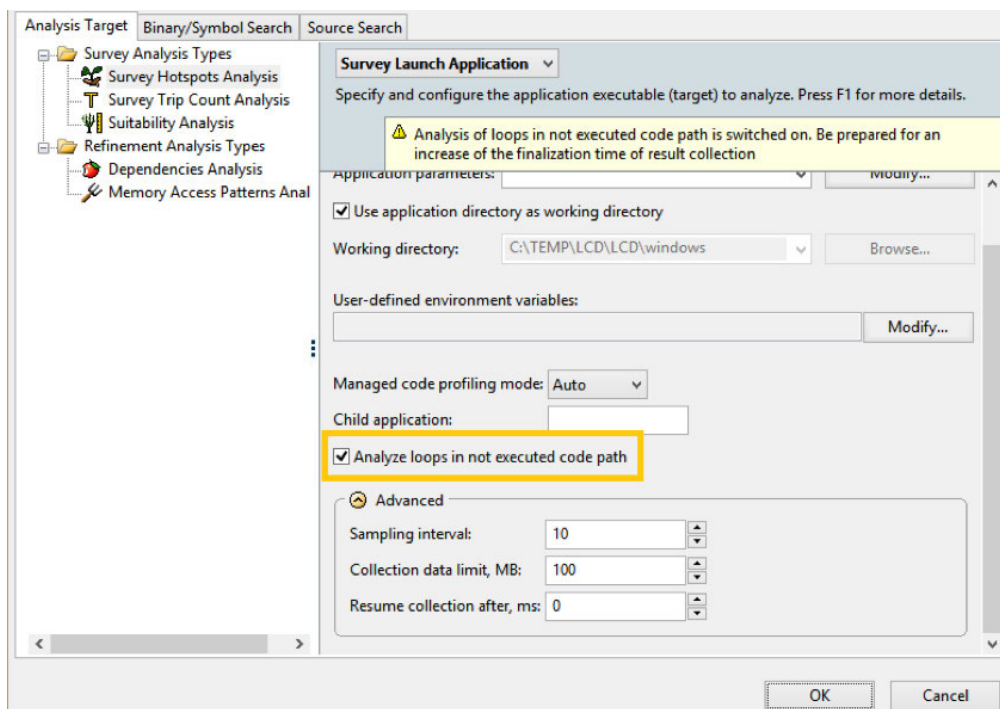


3 インテル® Advisor XE の [Survey Report (調査レポート)]

実行されなかったコードパスの解析を有効にする

ハードウェアの対応状況に応じて複数の命令セットから最適なものを使用するバイナリーをコンパイルしたら、インテル® Advisor XE でバイナリーに含まれるすべてのバージョンのベクトルループが解析されるように、次の操作を行います。

1. インテル® Advisor XE を実行します。
2. **プロジェクトのプロパティ (Ctrl+P)** で次の操作を行います。
 - a. バイナリーのパスを指定します。
 - b. [Analyze loops in not executed code path (実行されなかったコードパスのループを解析する)] チェックボックスをオンにします。



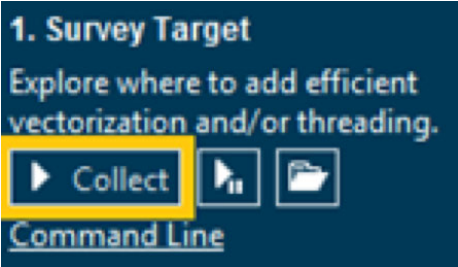
3. **[OK]** をクリックします。
- コマンドラインを使用する場合は (クラスターノード上の MPI アプリケーションなど)、次の構文を使用します。

```
mpirun -n 2 -gtool "advixe-cl -collect survey -support-multi-isa-binaries
-no-auto-finalize --project-dir=/tmp/my_proj" /tmp/bin/my_app
```

実行されなかったコードパスのループの確認

バイナリーを調査する

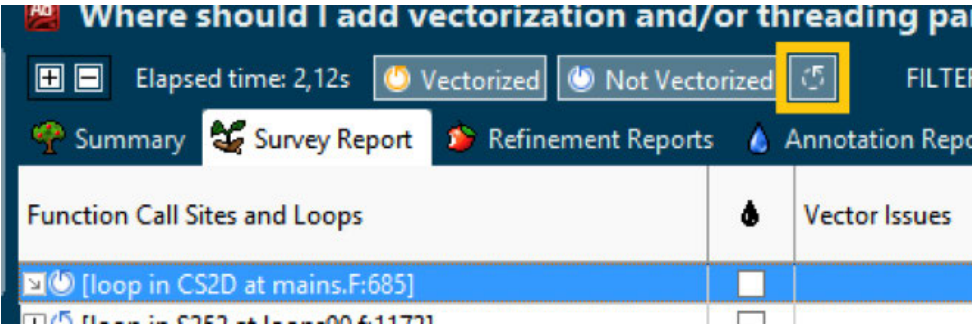
ワークフロー・タブで [Collect (収集)] ボタンをクリックします。












注: すべてのループ (現在のハードウェアで利用可能な命令セットと異なる命令セット向けの実行されなかったループを含む) の結果のファイナライズには、通常よりも長い時間がかかることがあります。

[Survey Report (調査レポート)] で実行されなかったループの表示を有効にする

調査解析結果の収集が完了したら、[Survey Report (調査レポート)] を確認します。対応するボタンをクリックして、調査グリッドで実行されなかったループの表示を有効にする必要があります。





ボタンをクリックすると、インテル® Advisor XE はグリッドをリフレッシュし、「親」ループの下に実行されなかったループを表示します。ベクトル化されたループ  を展開し、実行されたループとともに実行されなかったループを確認できます (図 4)。

Function Call Sites and Loops	Vectorized Loops				
	Vector ISA	Efficiency	Gain...	VL (Vector Length)	Compiler Estimated Gain
 [loop in S243 at loops90.f:1104]	AVX2	<div><div></div></div> ~76%	6,05x	8	6,05x
 [loop in S243 at loops90.f:1104]	AVX2			8	6,05x
 [loop in S243 at loops90.f:1104]					
 [loop in S243 at loops90.f:1104]	AVX512			16	3,10x
 [loop in S243 at loops90.f:1104]					
 [loop in S243 at loops90.f:1104]	AVX512			32	13,29x
 [loop in S243 at loops90.f:1104]	AVX512			16	5,49x
 [loop in S243 at loops90.f:1104]	AVX2			4	1,49x




4 実行されたループとされなかったループ

[Survey Report (調査レポート)] で確認すべき情報

ベクトル化されたループ (Vectorized Loops) 列を確認します ( ボタンで列幅を調整できます)。
インテル® Advisor XE は、実行されなかったループ  に関するコンパイラーによる診断情報も表示します。

- ベクトル ISA (Vector ISA) 列は、特定のコードパスの ISA を示します。
- ベクトル長 (VL (Vector Length)) 列は、ベクトルの長さを示します。この記事の執筆中に、インテル® Xeon Phi™ コプロセッサを搭載したマシンでサンプルコードを実行したところ、ベクトルループのほうがスカラーループよりも 16 から 32 命令 (ループに応じて) 少なく、優れたパフォーマンスが得られました。
- コンパイラーによって予測された期待値 (Compiler Estimated Gain) 列は、コンパイラーによって予測されたパフォーマンスの期待値を示します。期待値は、ターゲット ISA をサポートするハードウェアで同じループのスカラーバージョンを実行する場合との比較です。つまり、コンパイラーは、同じインテル® AVX2 対応マシンで実行した場合、ベクトル化されたループのほうがスカラーバージョンよりも 6.05 倍パフォーマンスが向上すると予測しています。また、インテル® AVX-512 対応マシンでは、13.29 倍のパフォーマンス向上を予測しています。

命令セット解析 (Instruction Set Analysis) 列で、インテル® AVX2 とインテル® AVX-512 のループを比較することも重要です。

Call Sites and Loops	Vectorized Loops 				Instruction Set Analysis 				
	Vector ISA	Eff...	Gain...	VL ..	Traits	Data T...	Nu.	Vector Widths	Instructio
p in S278 at loops90.f:1526]	AVX2		5,21x	8	FMA; Masked Stores	Float32...	16	256	AVX; AVX;
oop in S278 at loops90.f:1526]	AVX512			16	Unpacks; FMA; Mask Manip...	Float32...	18	512	AVX512F_
oop in S278 at loops90.f:1526]	AVX2			8	Masked Stores; FMA	Float32...	16	256	AVX; AVX;
oop in S278 at loops90.f:1526]	AVX512			16	FMA	Float32...	13	512	AVX512F_
loop in S278 at loops90.f:1526]					FMA	Float32...	3		

- 特性 (Traits) 列は、コードのパフォーマンスに (良くも、悪くも) 大きく影響する命令を示します。インテル® AVX-512 対応のループでは、インテル® Advisor XE はギャザー、圧縮、平方根の逆数、マスク付き操作などのインテル® AVX-512 でのみ有効な命令を特性 (Traits) として示します。
- ベクトル幅 (Vector Width) 列は、ベクトルレジスターの幅をビット単位で示します。この値は、ハードウェア固有です。
- 命令セット (Instruction Sets) 列は、個々の命令で使用された命令セットを示します。

アセンブリー表現を確認する

異なる ISA (インテル® AVX-512 など) 向けコードパスのアセンブリーを表示するには、調査グリッドでループを選択し、[Loop Assembly (ループ・アセンブリー)] タブをクリックします。

Function Call Sites and Loops		Vector Issues	Self Time▼	Total Time	Type	Vectorized Location Vector ISA
[loop in S491 at loops90.f:2911]	<input type="checkbox"/>		n/a	n/a	Scalar Versions	
[loop in S233 at loops90.f:983]	<input type="checkbox"/>		0,030s0	0,030s1	Vectorized (Body)	AVX
[loop in S161 at loops90.f:655]	<input type="checkbox"/>	1 Poss ...	0,030s0	0,030s1	Vectorized (Body)	AVX2
[loop in S341 at loops90.f:2254]	<input type="checkbox"/>	1 Assu ...	0,010s1	0,030s1	Vectorized Versions	AVX2
[loop in S414 at loops90.f:2484]	<input type="checkbox"/>		0,030s0	0,030s1	Vectorized (Body)	AVX2
[loop in S414 at loops90.f:2484]	<input type="checkbox"/>		0,030s0	0,030s1	Vectorized (Body)	AVX2
[loop in S414 at loops90.f:2484]	<input type="checkbox"/>		n/a	n/a	Vectorized (Peeled) [Not Executed]	AVX512
[loop in S414 at loops90.f:2484]	<input type="checkbox"/>		n/a	n/a	Peeled [Not Executed]	
[loop in S414 at loops90.f:2484]	<input type="checkbox"/>		n/a	n/a	Vectorized (Body) [Not Executed]	AVX512
[loop in S414 at loops90.f:2484]	<input type="checkbox"/>		n/a	n/a	Vectorized (Remainder) [Not Executed]	AVX512
[loop in S414 at loops90.f:2484]	<input type="checkbox"/>		n/a	n/a	Vectorized (Remainder) [Not Executed]	AVX2
[loop in S414 at loops90.f:2484]	<input type="checkbox"/>		n/a	n/a	Remainder [Not Executed]	
[loop in VBOR at loops90.f:3349]	<input type="checkbox"/>		0,020s0	0,020s1	Vectorized (Body)	AVX2

SourceTop Down Loop AnalyticsLoop Assembly Recommendations Compiler Diagnostic Details

Module: lcd_f90.exe!0x140055c67

	Address	Line	Assembly
remainder	0x140055c67		Block 1:
	0x140055c67 2484		add r8, 0x10
	0x140055c6b 2484		vpcmpd k1, k0, zmm1, zmm0, 0x2
	0x140055c72 2484		vpadd zmm1, k0, zmm1, zmm3
	0x140055c78 2485		vmovups zmm5, k1{z}, zmmword ptr [r15+rdx*1]
	0x140055c7f 2485		vmovups zmm6, k1{z}, zmmword ptr [r15+rbx*1]
	0x140055c86 2485		vmovups zmm4, k1{z}, zmmword ptr [r15+rbp*1]
	0x140055c8d 2485		vfmadd213ps zmm6, k0, zmm4, zmm5
	0x140055c93 2485		vmovups zmmword ptr [r15+rsi*1], k1, zmm6
	0x140055c9a 2484		add r15, 0x40
	0x140055c9e 2484		cmp r8, rax
	0x140055ca1 2484		jb 0x140055c67 <Block 1>

この機能を利用して、インテル® AVX2 とインテル® AVX-512 のコードパスの違いを確認できます。

ループのベクトル化に関する予備知識

一般的にベクトル化されたループは次の 3 つの部分で構成されます。

- **メインのベクトル本体:** 3 つの部分の中で最も高速です。
- **オプションのピール部分:** ループのアライメントされていないメモリー参照に使用されます。スカラーか、低速なベクトル命令を使用します。
- **剰余部分:** 反復回数 (トリップカウント) がベクトル長の倍数でない場合に生成されます。スカラーか、低速なベクトル命令を使用します。

ベクトルレジスター幅が長くなると、ループのピール/剰余部分の反復回数が多くなります。

- データはアライメントし、コンパイラーにアライメントされていることを伝えます。
- 反復回数は、ベクトル長の倍数になるようにします。

インテル® AVX-512 の診断例

RTM ステンシル・プロジェクト

ステンシル計算は、地震探査の RTM (Reverse Time Migration: リバース・タイム・マイグレーション) アルゴリズムの基本です。有限差分法により波動方程式を解きます。このサンプルは、3 次元の 25 ポイントステンシルです。**RTM ステンシル** サンプル用にインテル® AVX-512 コードを生成したところ、コンパイラーはインテル® AVX-512 では 25.28 倍、インテル® AVX2 では 9.59 倍のスピードアップを予測しました (図 5)。

ベクトル長が 2 倍のため、インテル® AVX-512 コードは 2 倍高速になると予測しましたが、コンパイラーによって示された期待値は 2.63 倍でした。

その理由は、インテル® Advisor XE により説明できます。インテル® AVX2 コードにはスカラーの剰余部分がありますが、この剰余部分がインテル® AVX-512 ではベクトル化されています。

Elapsed time: 7,58s Vectorized Not Vectorized FILTER: All Modules All Sources Loops All Threads									
Summary Survey Report Refinement Reports Annotation Report									
Function Call Sites and Loops	Vector Issues	Self Time	Total Time	Type	Why No V...	Vectorized Loops			
						Vector ISA	Efficiency	Gain...	VL (...)
[loop in co_basecase_simd at rtm_stencil.cpp:352]		22,496s	22,496s	Vectorized (Body)		AVX2	~100%	9,59x	8
[loop in co_basecase_simd at rtm_stencil.cpp:352]		22,496s	22,496s	Vectorized (Body)		AVX2		8	9,59x
[loop in co_basecase_simd at rtm_stencil.cpp:352]		n/a	n/a	Vectorized (Peeled) [Not Executed]		AVX512		16	9,20x
[loop in co_basecase_simd at rtm_stencil.cpp:352]		n/a	n/a	Peeled [Not Executed]					
[loop in co_basecase_simd at rtm_stencil.cpp:352]		n/a	n/a	Vectorized (Body) [Not Executed]		AVX512		16	25,28x
[loop in co_basecase_simd at rtm_stencil.cpp:352]		n/a	n/a	Vectorized (Remainder) [Not Executed]		AVX512		16	14,03x
[loop in co_basecase_simd at rtm_stencil.cpp:352]		n/a	n/a	Remainder [Not Executed]	uns.				

5 RTM ステンシルサンプルのスピードアップ

LCD ベクトル化ベンチマーク

このサンプルでは、コンパイラーは 2 つのバージョンの期待値を 12.20 倍 (インテル® AVX2) と 36.34 倍 (インテル® AVX-512) と予測しました。インテル® AVX-512 コードのほうが 2.97 倍高速ですが、ベクトル長は 2 倍です。RTM ステンシルサンプルでは、コンパイラーはインテル® AVX2 の剰余ループをベクトル化できませんでした。

このサンプルでは、インテル® AVX2 とインテル® AVX-512 の剰余部分はともにベクトル化されています。期待値が 2 倍を上回った理由は、インテル® AVX-512 のマスク付き操作によるものです。

インテル® AVX2:

Type	Why No Vector...	Vectorized Loops		Instruction Set Analysis					Advanced								
		Vector ISA	Efficien...	Gain...	VL (...)	Compiler Esti...	Traits	Data Types	Number of Ve...	Vector Widths	Instruction Sets	Transfor...	Unroll Fa...	Vectorization Details			
Vectorized (Body) [Not Executed]		AVX2			8	12,20x	FMA	Float32	4	256	AVX; FMA	Unrolled	2	Unaligned Access in Vector Loop			
Vectorized (Remainder) [Not Executed]		AVX512			16	16,88x	FMA	Float32; Int32...	6	512	AVX512F_512			Masked Loop Vectorization; Unaligned Access			
Vectorized (Remainder) [Not Executed]		AVX512			16	16,88x	FMA	Float32; Int32...	6	512	AVX512F_512			Masked Loop Vectorization; Unaligned Access			
Vectorized (Remainder) [Not Executed]		AVX2			4	2,68x	FMA	Float32	2	128	AVX; FMA			Unaligned Access in Vector Loop			
Vectorized (Remainder) [Not Executed]		AVX2			4	2,68x	FMA	Float32	2	128	AVX; FMA			Unaligned Access in Vector Loop			
Vectorized (Body) [Not Executed]		AVX512			32	36,34x	FMA	Float32	4	512	AVX512F_512	Unrolled	1	Unaligned Access in Vector Loop			
<																	
Source Top Down Loop Analytics Loop Assembly Recommendations Compiler Diagnostic Details																	
Module: lcd_exe.exe!0x14005c4b3																	
	Address	Line	Assembly										Total Time	%	Self Time	%	Traits
remainder	0x14005c4b3		Block 1:														
	0x14005c4b3	6252	vmovups xmm1, xmmword ptr [rcx+rsi*4]														FMA
	0x14005c4b8	6252	vmovups xmm0, xmmword ptr [rdi+rsi*4]														
	0x14005c4bd	6252	vfmadd213ps xmm1, xmm0, xmmword ptr [r8+rsi*4]														
	0x14005c4c3	6252	vmovups xmmword ptr [r8+rsi*4], xmm1														
	0x14005c4c9	6251	add rsi, 0x4														
	0x14005c4cd	6251	cmp rsi, r12														
	0x14005c4d0	6251	jb 0x14005c4b3 <Block 1>														

インテル® AVX-512:

Type	Why No Vector...	Vectorized Loops		Instruction Set Analysis					Advanced					
		Vector ISA	Efficiency	Gain...	VL (...)	Compiler Estimated ...	Traits	Data Types	Number of Ve...	Vector Widths	Instruction Sets	Transfor...	Unr...	Vectorization Details
Vectorized (Body) [Not Executed]		AVX2			8	12,20x	FMA	Float32	4	256	AVX; FMA	Unrolled	2	Unaligned Access in Vector Loop
Vectorized (Remainder) [Not Executed]		AVX512			16	16,88x	FMA	Float32; Int32...	6	512	AVX512F_512			Masked Loop Vectorization; Unaligned Access
Vectorized (Remainder) [Not Executed]		AVX512			16	16,88x	FMA	Float32; Int32...	6	512	AVX512F_512			Masked Loop Vectorization; Unaligned Access
Vectorized (Remainder) [Not Executed]		AVX2			4	2,68x	FMA	Float32	2	128	AVX; FMA			Unaligned Access in Vector Loop
Vectorized (Remainder) [Not Executed]		AVX2			4	2,68x	FMA	Float32	2	128	AVX; FMA			Unaligned Access in Vector Loop
Vectorized (Body) [Not Executed]		AVX512			32	36,34x	FMA	Float32	4	512	AVX512F_512	Unrolled	1	Unaligned Access in Vector Loop

Source

Top Down

Loop Analytics

Loop Assembly

Recommendations
























































Compiler Diagnostic Details

Module: lcd_exe.exe!0x14005bea8

Address	Line	Assembly	Total Time	%	Self Time	%	Traits
remainder	0x14005bea8	Block 1:					
	0x14005bea8 6251	add rsi, 0x10					
	0x14005beac 6251	vpcmpd k1, k0, xmm3, xmm2, 0x2					
	0x14005beb3 6251	vpaddq xmm3, k0, xmm3, xmm1					
	0x14005beb9 6252	vmovups xmm5, k1(z), xmmword ptr [r8+r15*1]					
	0x14005bec0 6252	vmovups xmm10, k1(z), xmmword ptr [r8+r16*1]					
	0x14005bec7 6252	vmovups xmm4, k1(z), xmmword ptr [r8+r17*1]					
	0x14005becd 6252	vfmadd213ps xmm10, k0, xmm4, xmm5					FMA
	0x14005bed4 6252	vmovups xmmword ptr [r8+rcx*1], k1, xmm10					
	0x14005bedb 6251	add r8, 0x40					
	0x14005bedf 6251	cmp rsi, rax					
	0x14005bee2 6251	jb 0x14005bea8 <Block 1>					

ベクトル化の詳細情報から、インテル® AVX-512 コードでは剰余ループでマスク付き操作が使用されており、16.88 倍のスピードアップにつながっています。一方、インテル® AVX2 の剰余ループは 2.60 倍のスピードアップにとどまっています。

マスク付き操作を利用して、インテル® AVX-512 バージョンでは「ピール」ループもベクトル化しています。次のループは、インテル® AVX-512 バージョンではピール、本体、剰余ループがベクトル化されます。

Function Call Sites and Loops	 Vect... Issues	 Self Tim...	Total Time	Type	Why No Vectori...	Vectorized Loops				Instruction Set Analysis					
						Vector ISA	Efficien...	Gain E...	VL (...)	Compiler Es...	Traits	Data Types	Number of Ve...	Vector Widths	Instruction Sets
  [loop in s234_at loopstl.cpp:2449]		0.030s	0.030s	Vectorized (Body)		AVX2	100%	11.52x	8	11.52x	FMA	Float32	4	256	AVX; FMA
  [loop in s234_at loopstl.cpp:2449]		0.030s	0.030s	Vectorized (Body)		AVX2			8	11.52x	FMA	Float32	4	256	AVX; FMA
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Peeled [Not Executed]							FMA	Float32	2		
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Peeled [Not Executed]							FMA	Float32	2		
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Scalar [Not Executed]	 non ...						FMA	Float32	4		
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Scalar [Not Executed]	 non ...						FMA	Float32	4		
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Remainder [Not Executed]							FMA	Float32	2		
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Remainder [Not Executed]							FMA	Float32	2		
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Vectorized (Peeled) [Not Executed]		AVX512			16	9.00x	FMA	Float32; Int32...	6	512	AVX512F_512
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Vectorized (Peeled) [Not Executed]		AVX512			16	9.00x	FMA	Float32; Int32...	6	512	AVX512F_512
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Vectorized (Remainder) [Not Executed]		AVX512			16	16.88x	FMA	Float32; Int32...	6	512	AVX512F_512
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Vectorized (Remainder) [Not Executed]		AVX512			16	16.88x	FMA	Float32; Int32...	6	512	AVX512F_512
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Vectorized (Body) [Not Executed]		AVX512			32	38.97x	FMA	Float32	4	512	AVX512F_512
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Vectorized (Body) [Not Executed]		AVX512			32	36.34x	FMA	Float32	4	512	AVX512F_512
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Vectorized (Remainder) [Not Executed]		AVX2			4	2.68x	FMA	Float32	2	128	AVX; FMA
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Vectorized (Remainder) [Not Executed]		AVX2			4	2.68x	FMA	Float32	2	128	AVX; FMA
  [loop in s234_at loopstl.cpp:2449]		n/a	n/a	Vectorized (Body) [Not Executed]		AVX2			8	12.20x	FMA	Float32	4	256	AVX; FMA

インテル® AVX-512 のベクトル化された剰余ループは、ベクトル長 16 をすべて使用しているのに対し、インテル® AVX2 バージョンはベクトル長 4 のみ—インテル® AVX2 の本体よりも少ない—を使用しています。インテル® AVX2 バージョンのピールループはスカラーのみであるのに対し、インテル® AVX-512 のピールループはベクトル化され、9.0 倍のスピードアップが期待できます。

要点のおさらい

インテル® Advisor XE では、次のことが可能です。

- 一度に、1 台のマシンで、複数の ISA 向けのコードを生成し、解析することができます。また、コンパイラー・レポートからパフォーマンスの期待値が得られます。
- 個々の命令で使用された ISA 固有の命令の「ファミリー」や、コードパス固有の特性を確認できます。

関連情報 (英語):

- インテル® C++ コンパイラーのコード生成オプション
- インテル® Fortran コンパイラーのコード生成オプション
- インテル® C++ コンパイラーの x、Qx オプション
- インテル® Fortran コンパイラーの x、Qx オプション
- インテル® C++ コンパイラーの ax、Qax オプション
- インテル® Fortran コンパイラーの ax、Qax オプション

まとめ

この記事では、インテル® Advisor XE 2016 の概要と、インテル® AVX-512 ISA 向けのコードをインテル® AVX2 対応プロセッサ・ベースのマシンで実行し、解析する機能を説明しました。また、サンプルを用いてベクトル化アドバイザーにより C++ STL コードをベクトル化する方法を示しました。

ハードウェアを最大限に活用するには、ベクトル化とスレッド化を導入してコードを現代化する必要があります。この記事で説明した系統的なアプローチを採用し、インテル® Parallel Studio XE の強力なツールを活用することで、簡単にこの作業を行うことができます。



インテル® Advisor XE を評価する
 インテル® Parallel Studio XE に含まれます >